MPC5553/MPC5554 Microcontroller Reference Manual

Devices Supported:

MPC5553
MPC5554

MPC5553/4RM
Rev. 3.1
10/2005

freescale™
semiconductor

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064, Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 26668334
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

MPC5553/4RM
Rev. 3.1
10/2005

Contents

Contents

Contents

Contents

Contents

Contents

Chapter 3
e200z6 Core Complex

Contents

Chapter 4
Reset

Contents

Contents

Contents

Contents

Chapter 9
Enhanced Direct Memory Access (eDMA)

Contents

Chapter 10
Interrupt Controller (INTC)

Contents

Contents

Chapter 12
External Bus Interface (EBI)

Contents

Contents

Contents

Chapter 13
Flash Memory

Contents

Chapter 14
Fast Ethernet Controller (FEC)

Contents

Chapter 15
Internal Static RAM (SRAM)

Contents

Contents

Chapter 18
Enhanced Time Processing Unit (eTPU)

Contents

Chapter 19
Enhanced Queued Analog-to-Digital Converter (eQADC)

Contents

Contents

Contents

Chapter 20
Deserial Serial Peripheral Interface (DSPI)

Contents

Contents

Chapter 21
Enhanced Serial Communication Interface (eSCI)

Contents

Contents

Chapter 22
FlexCAN2 Controller Area Network

Contents

Chapter 23
Voltage Regulator Controller (VRC) and POR Module

Contents

Chapter 24
IEEE 1149.1 Test Access Port Controller (JTAGC)

Contents

Chapter 25
Nexus Development Interface

Contents

Contents

Contents

Appendix A
MPC5553/MPC5554 Register Map

Contents

Contents

Paragraph Number	Title	Page Number

Contents

Paragraph Number	Title	Page Number

Chapter 1
Overview

1.1 Introduction

The MPC5553 and MPC5554 microcontrollers (MCU) are the first members of the MPC5500 family of next generation powertrain microcontrollers based on the PowerPC Book E architecture. The MPC5500 family contains a PowerPC™ processor core. This family of parts contains many new features coupled with high performance CMOS technology to provide significant performance improvement over the MPC565.

The e200z6 CPU of the MPC5500 family is based on the PowerPC Book E architecture. It is 100% user mode compatible (with floating point library) with the classic PowerPC instruction set. The Book E architecture has enhancements that improve the PowerPC architecture's fit in embedded applications. This core also has additional instructions, including digital signal processing (DSP) instructions, beyond the classic PowerPC instruction set.

The MPC5553 and MPC5554 of the MPC5500 family have two levels of memory hierarchy. The fastest accesses are to the unified cache (32-kilobytes in the MPC5554, 8-kilobytes in the MPC5553). The next level in the hierarchy contains the 96-kilobyte internal SRAM and internal Flash memory (2 Mbytes Flash in the MPC5554, 1.5 Mbytes in the MPC5553). Both the internal SRAM and the Flash memory can hold instructions and data. The external bus interface has been designed to support most of the standard memories used with the MPC5xx family.

The complex I/O timer functions of the MPC5500 family are performed by an enhanced time processor unit engines (eTPU) — two in the MPC5554, one in the MPC5553. Each eTPU engine controls 32 hardware channels. The eTPU has been enhanced over the TPU by providing 24-bit timers, double action hardware channels, variable number of parameters per channel, angle clock hardware, and additional control and arithmetic instructions. The eTPU can be programmed using a high-level programming language.

The less complex timer functions of the MPC5500 family are performed by the enhanced modular input/output system (eMIOS). The eMIOS' 24 hardware channels are capable of single action, double action, pulse width modulation (PWM) and modulus counter operation. Motor control capabilities include edge-aligned and center-aligned PWM.

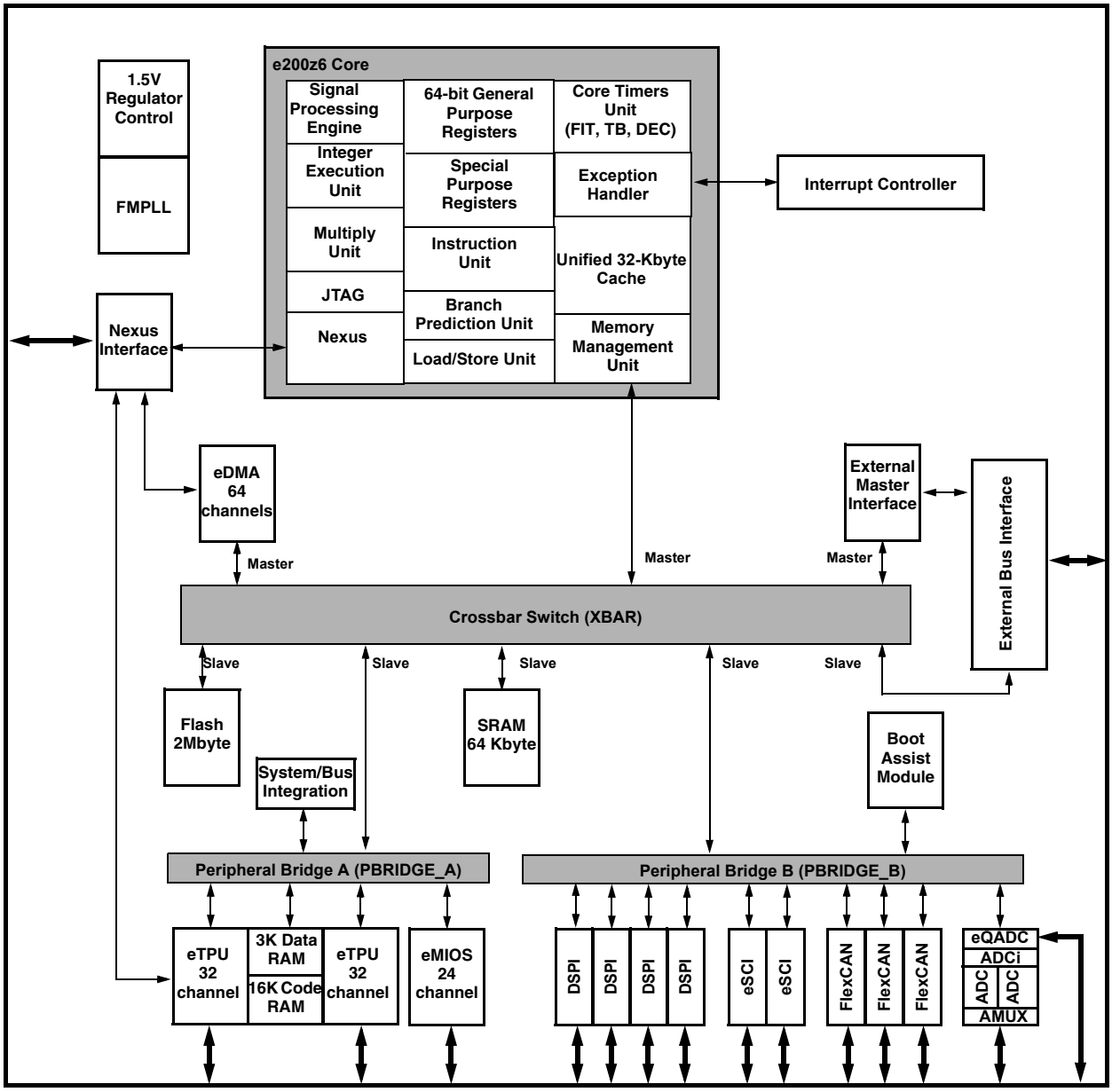
Off-chip communication is performed by a suite of serial protocols including controller area networks (FlexCANs) — three FlexCANs in the MPC5554 and two in the MPC5553, an enhanced deserial/serial peripheral interface (DSPI) — four in the MPC5554 and three in the MPC5553, and enhanced serial communications interfaces (eSCIs). The DSPIs support pin reduction through hardware serialization and deserialization of timer channels and general-purpose input/output (GPIO) signals.

The MCU of the MPC5553 and MPC5554 has an on-chip 40-channel enhanced queued dual analog to digital converter (eQADC).

The system integration unit (SIU) performs several chip-wide configuration functions. Pad configuration and general-purpose input and output (GPIO) are controlled from the SIU. External interrupts and reset control are also found in the SIU. The internal multiplexer submodule (SIU_DISR) provides multiplexing of eQADC trigger sources, daisy chaining the DSPIs, and external interrupt signal multiplexing.

The MPC5553 has a fast Ethernet controller (FEC) with a built-in FIFO and a DMA controller.

Figure 1-1 is a block diagram of the MPC5554 (MPC5500 family MCU), and Figure 1-2 is a block diagram of the MPC5553.



LEGEND

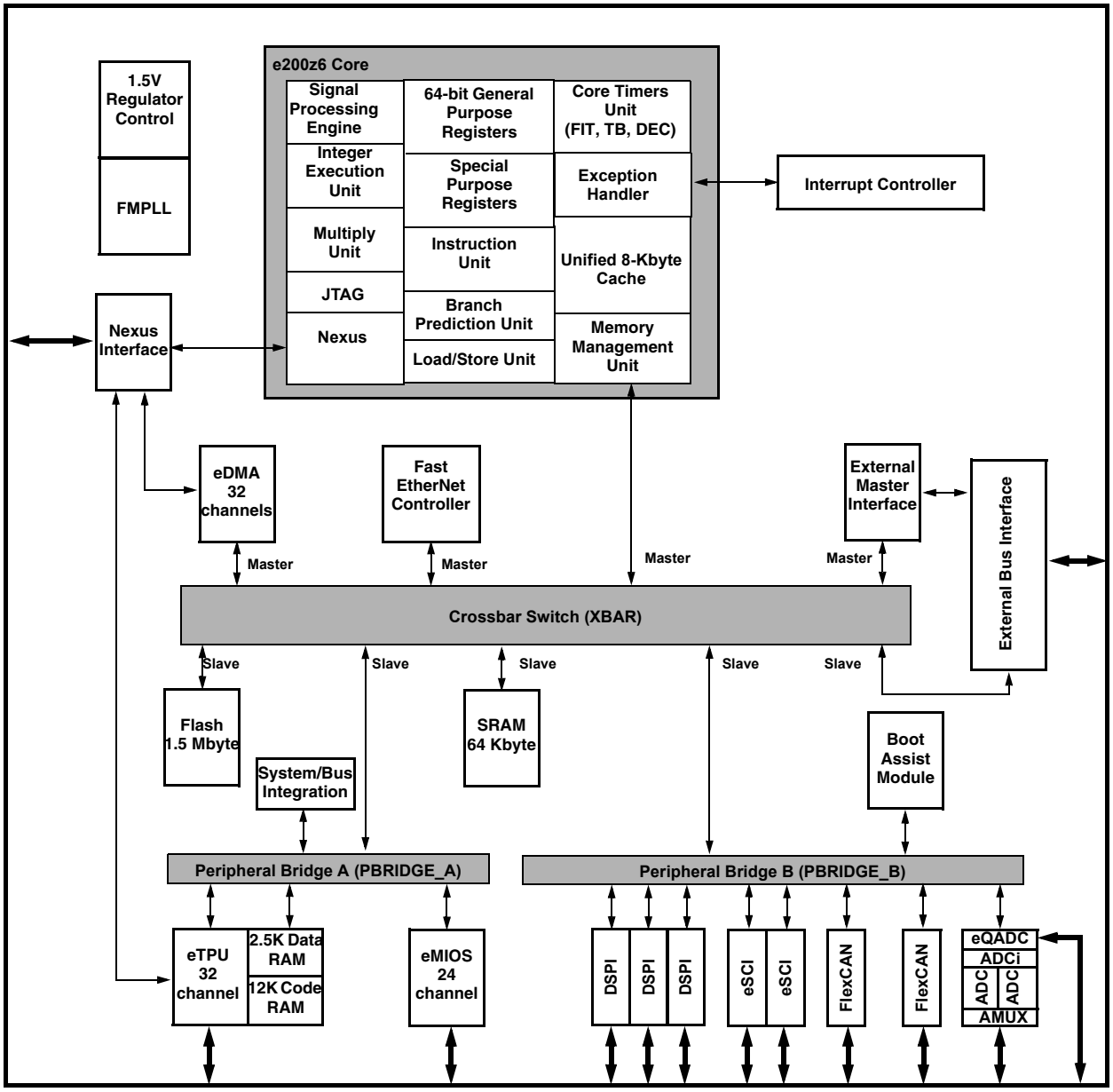
MPC5500 Device Module Acronyms

CAN – Controller area network (FlexCAN)
DSPI – Deserial/serial peripheral interface
DMA – Enhanced direct memory access
eMIOS – Enhanced modular I/O system
eQADC – Enhanced queued analog/digital converter
eSCI – Enhanced serial communications interface
eTPU – Enhanced time processing units
FMPLL – Frequency modulated phase-locked loop
SRAM – Static RAM

e200z6 Core Component Acronyms

DEC – Decrementer
FIT – Fixed interval timer
TB – Time base
WDT – Watchdog timer

Figure 1-1. MPC5554 Block Diagram



LEGEND

MPC5500 Device Module Acronyms

CAN – Controller area network (FlexCAN)
DSPI – Deserial/serial peripheral interface
DMA – Enhanced direct memory access
eMIOS – Enhanced modular I/O system
eQADC – Enhanced queued analog/digital converter
eSCI – Enhanced serial communications interface
eTPU – Enhanced time processing units
FMPLL – Frequency modulated phase-locked loop
SRAM – Static RAM

e200z6 Core Component Acronyms

DEC – Decrementer
FIT – Fixed interval timer
TB – Time base
WDT – Watchdog timer

Figure 1-2. MPC5553 Block Diagram

1.2 Features

This section provides a high-level description of the features found in the MPC5553 and MPC5554:

- Operating parameters
  - Fully static operation, 23.3–132 MHz
  - –40° to 150° C junction temperature
  - Low power design
    - Less than 1.2 Watts power dissipation
    - Designed for dynamic power management of core and peripherals
    - Software-controlled clock gating of peripherals
    - Separate power supply for stand-by operation for portion of internal SRAM
  - Fabricated in 0.13 μm process
  - 1.5V internal logic
  - Input and output pins with 3.0V–5.5V range
    - 35%/65% V_{DDE} CMOS switch levels (with hysteresis)
    - Selectable hysteresis
    - Selectable slew rate control
  - External bus and Nexus pins support 1.62V–3.6V operation
    - Selectable drive strength control
    - Unused pins configurable as GPIO
  - Designed with EMI reduction techniques
    - Frequency modulated phase-locked loop
    - On-chip bypass capacitance
    - Selectable slew rate and drive strength
- High performance e200z6 core processor
  - 32-bit PowerPC Book E compliant CPU
  - Thirty-two 64-bit general-purpose registers (GPRs)
  - Memory management unit (MMU) with 32-entry fully-associative translation look-aside buffer (TLB)
  - Branch processing unit
  - Fully pipelined load/store unit
  - 32 kilobyte unified cache (in the MPC5554), 8 kilobyte unified cache (in the MPC5553) with line locking
    - 8-way set associative in the MPC5554, 2-way set associative in the MPC5553
    - Two 32-bit fetches per clock
    - 8-entry store buffer
    - Way locking
    - Supports assigning cache as instruction or data only on a per way basis
    - Supports tag and data parity
  - Vectored interrupt support

  - Interrupt latency < 70 ns @132MHz (measured from interrupt request to execution of first instruction of interrupt exception handler)
  - Reservation instructions for implementing read-modify-write constructs (internal SRAM and Flash)
  - Signal processing engine (SPE) auxiliary processing unit (APU) operating on 64-bit GPRs
  - Floating point
    - IEEE® 754 compatible with software wrapper
    - Single precision in hardware, double precision with software library
    - Conversion instructions between single precision floating point and fixed point
  - Long cycle time instructions, except for guarded loads, do not increase interrupt latency in the MPC5554/MPC5553. To reduce latency in both the MPC5553 and the MPC5554, long cycle time instructions are aborted upon interrupt requests.
  - Extensive system development support through Nexus debug module
- System bus crossbar switch (XBAR)
  - 3 master ports in the MPC5554, 4 master ports in the MPC5553; 5 slave ports
  - 32-bit address bus, 64-bit data bus
  - Simultaneous accesses from different masters to different slaves (there is no clock penalty when a parked master accesses a slave)
- Enhanced direct memory access (eDMA) controller
  - 64 channels (MPC5554) or 32 channels (MPC5553) support independent 8-, 16-, 32-, or 64-bit single value or block transfers.
  - Supports variable sized queues and circular queues.
  - Source and destination address registers are independently configured to post-increment or remain constant.
  - Each transfer is initiated by a peripheral, CPU, or eDMA channel request.
  - Each eDMA channel can optionally send an interrupt request to the CPU on completion of a single value or block transfer.
- Interrupt controller (INTC)
  - 308 total interrupt vectors (MPC5554) or 212 total interrupt vectors (MPC5553)
    - 278 (MPC5554) or 191 (MPC5553) peripheral interrupt requests
    - plus 8 software settable sources
    - plus 22 reserved interrupts in the MPC5554, 13 reserved in the MPC5553
  - Unique 9-bit vector per interrupt source
  - 16 priority levels with fixed hardware arbitration within priority levels for each interrupt source
  - Priority elevation for shared resources
- Frequency modulated phase-locked loop (FMPLL)
  - Input clock frequency from 8 MHz to 20 MHz
  - Current controlled oscillator (ICO) range from 50 MHz to maximum device frequency
  - Reduced frequency divider (RFD) for reduced frequency operation without re-lock
  - Four selectable modes of operation
  - Programmable frequency modulation
  - Lock detect circuitry continuously monitors lock status
  - Loss of clock (LOC) detection for reference and feedback clocks

  - Self-clocked mode (SCM) operation
  - On-chip loop filter (reduces number of external components required)
  - Engineering clock output
- External bus interface (EBI)
  - 1.8V–3.3V I/O nominal I/O voltage
  - Memory controller with support for various memory types
  - MPC5554 specifications:
    - 32-bit data bus, 24-bit address bus with transfer size indication
  - MPC5553 specifications:
    - 416 BGA: 32-bit data bus, 24-bit address bus without transfer size indication
    - 324 BGA: 16-bit data bus, 20-bit address bus
    - 208 MAPBGA: no external bus
  - Selectable drive strength
  - Configurable bus speed modes
  - Support for external master accesses to internal addresses
  - Burst support
  - Bus monitor
  - Chip selects
    - In both the MPC5553 and MPC5554, four chip select ($\overline{\text{CS}}$[0:3]) signals; but the MPC5553 has no $\overline{\text{CS}}$ signals in the 208 MAPBGA package.
    - In the MPC5553 only, support for dynamic calibration with up to three calibration chip selects ($\overline{\text{CAL\_CS}}$[0] and $\overline{\text{CAL\_CS}}$[2:3])
  - Configurable wait states
- System integration unit (SIU)
  - Centralized GPIO control of 214 (MPC5554) or 198 (MPC5553) I/O and bus pins
  - Centralized pad control on a per-pin basis
  - System reset monitoring and generation
  - External interrupt inputs, filtering and control
  - Internal multiplexer submodule (SIU_DISR, SIU_ETISR, SIU_EIISR)
- Error correction status module (ECSM)
  - Configurable error-correcting codes (ECC) reporting for internal SRAM and Flash memories
- On-chip FLASH
  - 2 Mbytes (MPC5554) or 1.5 Mbytes (MPC5553) burst Flash memory
  - 256K × 64 bit configuration
  - Censorship protection scheme to prevent Flash content visibility
  - Hardware read-while-write feature that allows blocks to be erased/programmed while other blocks are being read (used for EEPROM emulation and data calibration)
  - 20 blocks (MPC5554) or 16 blocks (MPC5553) with sizes ranging from 16 Kbytes to 128 Kbytes to support features such as boot block, operating system block, and EEPROM emulation
  - Read while write with multiple partitions
  - Parallel programming mode to support rapid end of line programming

  - Hardware programming state machine
- Configurable cache memory, 32 kilobyte (MPC5554) / 0 – 8 kilobyte (MPC5553)
  - 8-way set-associative, unified (instruction and data) cache in the MPC5554
  - 2-way set-associative unified (instruction and data) cache in the MPC5553
- On-chip internal static RAM (SRAM)
  - 64 kilobyte general-purpose RAM of which 32 kilobytes are on standby power
  - ECC performs single bit correction, double bit error detection
- Boot assist module (BAM)
  - Enables and manages the transition of MCU from reset to user code execution in the following configurations:
    - User application can boot from internal or external Flash memory
    - Download and execution of code via FlexCAN or eSCI
- Enhanced modular I/O system (eMIOS)
  - 24 orthogonal channels with double action, PWM, and modulus counter functionality
  - Supports all DASM and PWM modes of MIOS14 (MPC5xx)
  - Four selectable time bases plus shared time or angle counter bus
  - DMA and interrupt request support
  - Motor control capability
- Enhanced time processor unit (eTPU)
  - MPC5554 has two eTPU engines, MPC5553 has one engine
  - Each eTPU engine is an event-triggered timer subsystem
  - High level assembler/compiler
  - 32 channels per engine
  - 24-bit timer resolution
  - 16 kilobyte shared code memory in the MPC5554, 12 kilobyte shared code memory in the MPC5553
  - 3 kilobyte (MPC5554) or 2.5 kilobyte (MPC5553)Shared data memory
  - Variable number of parameters allocatable per channel
  - Double match/capture channels
  - Angle clock hardware support
  - Shared time or angle counter bus for all eTPU and eMIOS modules
  - DMA and interrupt request support
  - Nexus class 3 debug support (with some class 4 support)
- Enhanced queued analog/digital converter (eQADC)
  - 2 independent ADCs with 12-bit A/D resolution
  - Common mode conversion range of 0–5V
  - 40 single-ended inputs channels, expandable to 65 channels with external multiplexers on 416 and 324 BGA packages
  - 34 single-ended inputs channels, expandable to 57 channels with external multiplexers on 208 BGA packages
  - 8 channels can be used as 4 pairs of differential analog input channels
  - 10-bit accuracy at 400 ksamples/s, 8-bit accuracy at 800 ksamples/s
  - Supports six FIFO queues with fixed priority.

  - Queue modes with priority-based preemption; initiated by software command, internal (eTPU and eMIOS), or external triggers
  - DMA and interrupt request support
  - Supports all functional modes from QADC (MPC5xx family)
- 4 (MPC5554) or 3 (MPC5553) deserial serial peripheral interface modules (DSPI)
  - SPI
    - Full duplex communication ports with interrupt and eDMA request support
    - Supports all functional modes from QSPI submodule of QSMCM (MPC5xx family)
    - Support for queues in RAM
    - 6 chip selects, expandable to 64 with external demultiplexers
    - Programmable frame size, baud rate, clock delay and clock phase on a per frame basis
    - Modified SPI mode for interfacing to peripherals with longer setup time requirements
  - Deserial serial interface (DSI)
    - Pin reduction by hardware serialization and deserialization of eTPU and eMIOS channels
    - Chaining of DSI submodules
    - Triggered transfer control and change in data transfer control (for reduced EMI)
- 2 enhanced serial communication interface (eSCI) modules
  - UART mode provides NRZ format and half or full duplex interface
  - eSCI bit rate up to 1 Mbps
  - Advanced error detection, and optional parity generation and detection
  - Word length programmable as 8 or 9 bits
  - Separately enabled transmitter and receiver
  - LIN Support
  - DMA support
  - Interrupt request support
- 3 (MPC5554) or 2 (MPC5553) FlexCANs
  - 64 message buffers each
  - Full implementation of the CAN protocol specification, Version 2.0B
  - Based on and including all existing features of the Freescale TouCAN module
  - Programmable acceptance filters
  - Short latency time for high priority transmit messages
  - Arbitration scheme according to message ID or message buffer number
  - Listen only mode capabilities
  - Programmable clock source: system clock or oscillator clock
- Nexus development interface (NDI)
  - Per IEEE®-ISTO 5001-2003
  - Real time development support for PowerPC core and eTPU engines through Nexus class 3 (some Class 4 support)
  - Data trace of eDMA accesses
  - Read and write access
  - Configured via the IEEE® 1149.1 (JTAG) port
  - High bandwidth mode for fast message transmission

  - Reduced bandwidth mode for reduced pin usage
- IEEE® 1149.1 JTAG controller (JTAGC)
  - IEEE® 1149.1-2001 test access port (TAP) interface
  - A JCOMP input that provides the ability to share the TAP. Selectable modes of operation include JTAGC/debug or normal system operation.
  - A 5-bit instruction register that supports IEEE® 1149.1-2001 defined instructions.
  - A 5-bit instruction register that supports additional public instructions.
  - Three test data registers: a bypass register, a boundary scan register, and a device identification register.
  - A TAP controller state machine that controls the operation of the data registers, instruction register and associated circuitry.
- Voltage regulator controller
  - Provides a low cost solution to power the core logic. It reduces the number of power supplies required from the customer power supply chip.
- POR block
  - Provides initial reset condition up to the voltage at which pins ($\overline{\text{RESET}}$) can be read safely. It does not guarantee the safe operation of the chip at specified minimum operating voltages.

1.3 MPC5553-Specific Modules

The MPC5553 has one module not found on the MPC5554, a Fast Ethernet Controller (FEC) module that supports the following features:

  - IEEE® 802.3 MAC (compliant with IEEE® 802.3 1998 edition)
  - Built-in FIFO and DMA controller
  - Support for different Ethernet physical interfaces:
    - 100Mbps IEEE® 802.3 MII
    - 10Mbps IEEE® 802.3 MII
    - 10Mbps 7-wire interface (industry standard)

1.4 MPC5500 Family Comparison

Table 1-1. MPC5500 Family Comparison

MPC5500 Device	MPC5554	MPC5553
Core	e200z6	e200z6
Cache	32k	8k
Memory Management Unit	32 entry	32 entry
Crossbar	3x5	4x5
Core Nexus	NZ6C3 (3+)	NZ6C3 (3+)
SRAM	64k	64k
Flash	2M	1.5M
External bus (EBI)	32 bit	32 bit
Direct Memory Access (eDMA)	64 channel	32 channel
DMA Nexus	Class 3	Class 3
Serial	2	2
eSCI_A	yes	yes
eSCI_B	yes	yes
Controller Area Network (CAN)	3	2
CAN_A	64 buf	64 buf
CAN_B	64 buf	no
CAN_C	64 buf	64 buf
DSPI	4	3
DSPI_A	yes	no
DSPI_B	yes	yes
DSPI_C	yes	yes
DSPI_D	yes	yes
eMIOS	24 channel	24 channel
eTPU	64 channel	32 channel
eTPU_A	yes	yes
eTPU_B	yes	no
Code memory	16k	12k
Parameter RAM	3k	2.5k
Interrupt controller	300 channel	200 channel
Analog to Digital Converter	40 channel	40 channel
ADC_A	yes	yes
ADC_B	yes	yes
Phase Lock Loop (PLL)	FM	FM
Voltage Regulator Controller (VRC)	yes	yes

1.5 Detailed Features

The following sections provided detailed information about each of the on-chip modules.

1.5.1 e200z6 Core Overview

The MPC5553 and MPC5554 use the e200z6 core explained in detail in the *e200z6 PowerPC™ Core Reference Manual*. The e200z6 CPU utilizes a seven stage pipeline for instruction execution. The instruction fetch 1, instruction fetch 2, instruction decode/register file read, execute1, execute2/memory access1, execute3/memory access2, and register writeback stages operate in an overlapped fashion, allowing single clock instruction execution for most instructions.

The integer execution unit consists of a 32-bit arithmetic unit (AU), a logic unit (LU), a 32-bit barrel shifter, a mask-insertion unit (MIU), a condition register manipulation unit (CRU), a count-leading-zeros unit (CLZ), a 32x32 hardware multiplier array, result feed-forward hardware, and support hardware for division.

Most arithmetic and logical operations are executed in a single cycle with the exception of multiply, which is implemented with a pipelined hardware array, and the divide instructions. The CLZ unit operates in a single clock cycle.

The instruction unit contains a program counter (PC) incrementer and a dedicated branch address adder to minimize delays during change of flow operations. Sequential prefetching is performed to ensure a supply of instructions into the execution pipeline. Branch target prefetching is performed to accelerate taken branches. Prefetched instructions are placed into an instruction buffer capable of holding 6 sequential instructions and 2 branch target instructions.

Branch target addresses are calculated in parallel with branch instruction decode, resulting in execution time of three clocks. Conditional branches which are not taken execute in a single clock. Branches with successful lookahead and target prefetching have an effective execution time of one clock.

Memory load and store operations are provided for byte, halfword, word (32-bit), and doubleword data with automatic zero or sign extension of byte and halfword load data. These instructions can be pipelined to allow effective single cycle throughput. Load and store multiple word instructions allow low overhead context save and restore operations. The load/store unit contains a dedicated effective address adder to allow effective address generation to be optimized.

The condition register unit supports the condition register (CR) and condition register operations defined by the PowerPC architecture. The condition register consists of eight 4-bit fields that reflect the results of certain operations, such as move, integer and floating-point compare, arithmetic, and logical instructions, and provide a mechanism for testing and branching.

Vectored and auto-vectored interrupts are supported by the CPU. Vectored interrupt support is provided to allow multiple interrupt sources to have unique interrupt handlers invoked with no software overhead.

The signal processing extension (SPE) APU supports vector instructions (SIMD) operating on 16- and 32-bit fixed-point data types, as well as 32-bit IEEE®-754 single-precision floating-point formats, and supports single-precision floating-point operations in a pipelined fashion. The 64-bit general-purpose register file is used for source and destination operands, and there is a unified storage model for single-precision floating-point data types of 32-bits and the normal integer type. Low latency fixed-point and floating-point add, subtract, multiply, divide, compare, and conversion operations are provided, and most operations can be pipelined.

1.5.2 System Bus Crossbar Switch

The system bus's XBAR multi-port crossbar switch supports simultaneous connections between three(MPC5554) or four (MPC5553) master ports and five slave ports. The crossbar supports a 32-bit address bus width and a 64-bit data bus width at all master and slave ports.

The crossbar allows for concurrent transactions to occur from any master port to any slave port. It is possible for all master ports and slave ports to be in use at the same time as a result of independent master requests. If a slave port is simultaneously requested by more than one master port, arbitration logic will select the higher priority master and grant it ownership of the slave port. All other masters requesting that slave port will be stalled until the higher priority master completes its transactions. By default, requesting masters will be treated with equal priority and will be granted access to a slave port in round-robin fashion, based upon the ID of the last master to be granted access.

1.5.3 eDMA

The enhanced direct memory access (eDMA) controller is a second-generation module capable of performing complex data movements via 64 (MPC5554) or 32 (MPC5553) programmable channels, with minimal intervention from the CPU. The hardware micro architecture includes a DMA engine which performs source and destination address calculations, and the actual data movement operations, along with an SRAM-based memory containing the transfer control descriptors (TCD) for the channels. This implementation is utilized to minimize the overall module size.

1.5.4 INTC

The interrupt controller (INTC) provides priority-based preemptive scheduling of interrupt requests, suitable for statically scheduled real-time systems. The INTC allows interrupt request servicing from 308 (MPC5554)/212(MPC5553) interrupt sources.

For high priority interrupt requests, the time from the assertion of the interrupt request from the peripheral to when the processor is executing the interrupt service routine (ISR) has been minimized. The INTC provides a unique vector for each interrupt request source for quick determination of which ISR needs to be executed. It also provides an ample number of priorities so that lower priority ISRs do not delay the execution of higher priority ISRs. To allow the appropriate priorities for each source of interrupt request, the priority of each interrupt request is software configurable.

When multiple tasks share a resource, coherent accesses to that resource must be supported. The INTC supports the priority ceiling protocol for coherent accesses. By providing a modifiable priority mask, the priority level can be raised temporarily so that no task can prempt another task that shares the same resource.

Multiple processors can assert interrupt requests to each other through software settable interrupt requests (by using application software to assert requests). These maskable interrupt requests can be used to split the software into a high priority portion and a low priority portion for servicing the interrupt requests. The high priority portion is initiated by a peripheral interrupt request, but then the ISR asserts a software settable interrupt request to finish the servicing in a lower priority ISR.

1.5.5 FMPLL

The frequency modulated PLL (FMPLL) allows the user to generate high speed system clocks from an 8MHz to 20MHz crystal oscillator or external clock generator. Further, the FMPLL supports programmable frequency modulation of the system clock. The PLL multiplication factor, output clock divider ratio, modulation depth, and modulation rate are all software configurable.

1.5.6 EBI

The external bus interface (EBI) controls data transfer across the crossbar switch to/from memories or peripherals in the external address space. The EBI also enables an external master to access internal address space. The EBI includes a memory controller that generates interface signals to support a variety of external memories. The EBI memory controller supports single data rate (SDR) burst mode Flash, external SRAM, and asynchronous memories. In addition, the EBI supports up to 4 regions (via chip selects), along with programmed region-specific attributes.

1.5.7 SIU

The MPC5553/MPC5554 system integration unit (SIU) controls MCU reset configuration, pad configuration, external interrupt, general-purpose I/O (GPIO), internal peripheral multiplexing, and the system reset operation. The reset configuration module contains the external pin boot configuration logic. The pad configuration module controls the static electrical characteristics of I/O pins. The GPIO module provides uniform and discrete input/output control of the I/O pins of the MCU. The reset controller performs reset monitoring of internal and external reset sources, and drives the RSTOUT pin. The SIU is accessed by the e200z6 core through the crossbar switch.

1.5.8 ECSM

The error correction status module (ECSM) provides status information regarding platform memory errors reported by error-correcting codes.

1.5.9 Flash

The MPC5554 provides 2 Mbytes of programmable, non-volatile, Flash memory storage. The MPC5553 provides 1.5 Mbytes of Flash memory. The non-volatile memory (NVM) can be used for instruction and/or data storage.

The MPC5553/MPC5554 Flash also contains a Flash bus interface unit (FBIU) that interfaces the system bus to a dedicated Flash memory array controller. The FBIU supports a 64-bit data bus width at the system bus port, and a 256-bit read data interface to Flash memory. The FBIU contains two 256-bit prefetch buffers, and a prefetch controller that prefetches sequential lines of data from the Flash array into the buffer. Prefetch buffer hits allow no-wait responses. Normal Flash array accesses are registered in the FBIU and are forwarded to the system bus on the following cycle, incurring three wait-states. Prefetch operations may be automatically controlled, and may be restricted to servicing a single bus master. Prefetches may also be restricted to being triggered for instruction or data accesses.

1.5.10 Cache

The e200z6 core supports a 32-Kbyte (MPC5554) / 8-Kbyte (MPC5553), 8-way (MPC5554) / 2-way (MPC5553) set-associative, unified (instruction and data) cache with a 32-byte line size. The cache improves system performance by providing low-latency data to the e200z6 instruction and data pipelines, which decouples processor performance from system memory performance. The cache is virtually indexed and physically tagged. The e200z6 does not provide hardware support for cache coherency in a multi-master environment. Software must be used to maintain cache coherency with other possible bus masters.

Both instruction and data accesses are performed using a single bus connected to the cache. Addresses from the processor to the cache are virtual addresses used to index the cache array. The memory management unit (MMU) provides the virtual to physical translation for use in performing the cache tag

compare. The MMU may also be configured so that virtual addresses are passed through to the cache as the physical address untranslated. If the physical address matches a valid cache tag entry, the access hits in the cache. For a read operation, the cache supplies the data to the processor, and for a write operation, the data from the processor updates the cache. If the access does not match a valid cache tag entry (misses in the cache) or a write access must be written through to memory, the cache performs a bus cycle on the system bus.

1.5.11 SRAM

The MPC5500 family's internal SRAM module provides a general-purpose 96-Kbyte memory block that supports mapped read/write accesses from any master. Included within the 96-Kbyte SRAM block is a 32-Kbyte block powered by a separate supply for standby operation, and ECC error correction and detection.

1.5.12 BAM

The boot assist module (BAM) is a block of read-only memory that is programmed once by Freescale and is identical for all MCUs with an e200z6 core. The BAM program is executed every time the MCU is powered-on or reset in normal mode. The BAM supports the following four modes of booting:

- Booting from internal Flash memory
- Single master booting from external memory
- Multi master booting from external memory with either no arbitration or external arbitration
- Serial boot loading (a program is downloaded into RAM via eSCI or the FlexCAN and then executed).

The BAM also reads the reset configuration half word (RCHW) from Flash memory (either internal or external) and configures the MPC5553 and MPC5554 hardware accordingly.

1.5.13 eMIOS

The enhanced modular I/O system (eMIOS) module provides the functionality to generate or measure time events. A unified channel (UC) module is employed that provides a superset of the functionality of all the MIOS channels, while providing a consistent user interface. This allows more flexibility as each unified channel can be programmed for different functions in different applications. In order to identify up to two timed events, each UC contains two comparators, a time base selector and registers. This structure is able to produce match events, which can be configured to measure or generate a waveform. Alternatively, input events can be used to capture the time base, allowing measurement of an input signal.

1.5.14 eTPU

The enhanced time processing unit (eTPU) is an enhanced co-processor designed for timing control. Operating in parallel with the CPU, the eTPU processes instructions and real-time input events, performs output waveform generation, and accesses shared data without host intervention. Consequently, for each timer event, the CPU setup and service times are minimized or eliminated. In the MPC5554 MCU, two eTPU engines are grouped together with shared instruction and data RAM to form a powerful time processing subsystem. The MPC5553 has one eTPU engine. High-level assembler/compiler and documentation allows customers to develop their own functions on the eTPU. The eTPU supports several features of older TPU versions, making it easy to port older applications.

1.5.15 eQADC

The enhanced queued analog to digital converter (eQADC) module provides accurate and fast conversions for a wide range of applications. The eQADC provides a parallel interface to two on-chip analog to digital converters (ADCs), and a single master-to-single slave serial interface to an off-chip external device. The two on-chip ADCs are architected to allow access to all the analog channels.

The eQADC transfers commands from multiple command FIFOs (CFIFOs) to the on-chip ADCs or to the external device. The module can also receive data from the on-chip ADCs or from an off-chip external device into multiple result FIFOs (RFIFOs) in parallel, independently of the CFIFOs. The eQADC supports software and external hardware triggers from other modules to initiate transfers of commands from the CFIFOs to the on-chip ADCs or to the external device. It also monitors the fullness of CFIFOs and RFIFOs, and accordingly generates eDMA or interrupt requests to control data movement between the FIFOs and the system memory, which is external to the eQADC.

1.5.16 DSPI

The deserial serial peripheral interface (DSPI) module provides a synchronous serial interface for communication between the MCU and external devices. The DSPI supports pin count reduction through serialization and deserialization of eTPU channels, eMIOS channels and memory-mapped registers. The channels and register content are transmitted using a SPI-like protocol. There are four identical DSPI modules (DSPI_A, DSPI_B, DSPI_C, and DSPI_D) on the MPC5554 MCU. The MPC5553 has three DSPI modules (DSPI_B, DSPI_C, and DSPI_D).

The DSPIs have three configurations:

- Serial peripheral interface (SPI) configuration where the DSPI operates as a SPI with support for queues
- Deserial serial interface (DSI) configuration where the DSPI serializes eTPU and eMIOS output channels and deserializes the received data by placing it on the eTPU and eMIOS input channels
- Combined serial interface (CSI) configuration where the DSPI operates in both SPI and DSI configurations interleaving DSI frames with SPI frames, giving priority to SPI frames

For queued operations, the SPI queues reside in system memory external to the DSPI. Data transfers between the memory and the DSPI FIFOs are accomplished through the use of the eDMA controller or through host software.

1.5.17 eSCI

The enhanced serial communications interface (eSCI) allows asynchronous serial communications with peripheral devices and other MCUs. It includes special support to interface to local interconnect network (LIN) slave devices.

1.5.18 FlexCAN

The MCU contains three (MPC5554) or two (MPC5553) controller area network (FlexCAN) modules. Each FlexCAN module is a communication controller implementing the CAN protocol according to CAN Specification version 2.0B. The CAN protocol was designed to be used primarily as a vehicle serial data bus, meeting the specific requirements of this field: real-time processing, reliable operation in the EMI environment of a vehicle, cost-effectiveness and required bandwidth. Each FlexCAN module contains 64 message buffers (MB).

1.5.19 NDI

The Nexus development interface (NDI) module provides real-time development support capabilities for the MPC5500 family's PowerPC-based MCU in compliance with the IEEE®-ISTO 5001-2003 standard. This development support is supplied for MCUs without requiring external address and data pins for internal visibility. The NDI module is an integration of several individual Nexus modules that are selected to provide the development support interface for the MPC5500 family. The NDI module interfaces to the host processor, to one or dual eTPU processors, and internal buses to provide development support as per the IEEE®-ISTO 5001-2003 standard. The development support provided includes program trace, data trace, watchpoint trace, ownership trace, run-time access to the MCU's internal memory map, and access to the PowerPC and eTPU internal registers during halt, via the auxiliary port. The Nexus interface also supports a JTAG only mode using only the JTAG pins.

1.5.20 JTAGC

The JTAG controller (JTAGC) module provides the means to test chip functionality and connectivity while remaining transparent to system logic when not in test mode. Testing is performed via a boundary scan technique, as defined in the IEEE® 1149.1-2001 standard. All data input to and output from the JTAGC module is communicated in serial format. The JTAGC module is compliant with the IEEE® 1149.1-2001 standard.

1.5.21 FEC (MPC5553 Only)

The fast Ethernet controller (FEC) of the MPC5553 supports several standard MAC-PHY interfaces to connect to an external Ethernet transceiver:

- 10/100 Mbps MII interface
- 10 Mbps 7-Wire interface that uses a subset of the MII pins

1.6 MPC5500 Family Memory Map

This section describes the MPC5500 family memory map. All addresses in the device, including those that are reserved, are identified in the tables. The addresses represent the physical addresses assigned to each module. Logical addresses are translated by the MMU into physical addresses.

Under software control of the MMU, the logical addresses allocated to modules may be changed on a minimum of a 4-Kbyte boundary. Peripheral modules may be redundantly mapped. The customer must use the MMU to prevent corruption.

Table 1-2 shows a detailed memory map.

Table 1-2. Detailed MPC5554/MPC5553 Family Memory Map

Address Range[1]	Allocated Size[1] (bytes)	Used Size (bytes)	Use
0x0000_0000–0x001F_FFFF (MPC5554) 0x0000_0000–0x0017_FFFF (MPC5553)	2 Mbytes	2 Mbytes (MPC5554) 1.5 Mbytes (MPC5553)	FLASH Memory Array
0x0020_0000–0x00FF_FBFF (MPC5554) 0x0018_0000–0x00FF_FBFF (MPC5553)	(14 Mbytes - 1 Kbyte) (MPC5554) (14.5 Mbytes - 1 Kbyte) (MPC5553)	N/A	Reserved
0x00FF_FC00–0x00FF_FFFF	1024	1024 bytes	FLASH Shadow Row
0x0100_0000–0x1FFF_FFFF	496 Mbytes	2 Mbytes	emulation mapping of FLASH Array
0x2000_0000–0x3FFF_FFFF	512 Mbytes	N/A	External Memory
0x4000_0000–0x4000_7FFF	32 Kbytes	32 Kbytes	Internal SRAM Array, Standby Powered
0x4000_8000–0x4000_FFFF	32 Kbytes	32 Kbytes	Internal SRAM Array
0x4001_0000–0xBFFF_FFFF	(2048 Mbytes–64 Kbytes)	N/A	Reserved
Bridge A Peripherals			
0xC000_0000–0xC3EF_FFFF	63 M	N/A	Reserved
0xC3F0_0000–0xC3F0_3FFF	16 K	—	Bridge A Registers
0xC3F0_4000–0xC3F7_FFFF	496K	N/A	Reserved
0xC3F8_0000–0xC3F8_3FFF	16 Kbytes	—	FMPLL Registers
0xC3F8_4000–0xC3F8_7FFF	16 Kbytes	48	External Bus Interface (EBI) Configuration Registers
0xC3F8_8000–0xC3F8_BFFF	16 Kbytes	28	Flash Configuration Registers
0xC3F8_C000–0xC3F8_FFFF	16 Kbytes	N/A	Reserved
0xC3F9_0000–0xC3F9_3FFF	16 Kbytes	2.5 Kbytes	System Integration Unit (SIU)
0xC3F9_4000–0xC3F9_FFFF	48 Kbytes	N/A	Reserved
0xC3FA_0000–0xC3FA_3FFF	16 Kbytes	1056	Modular Timer System (eMIOS/MTS)
0xC3FA_4000–0xC3FB_FFFF	112 Kbytes	N/A	Reserved
0xC3FC_0000–0xC3FC_3FFF	16 Kbytes	3 Kbytes	Enhanced Time Processing Unit (eTPU) Registers
0xC3FC_4000–0xC3FC_7FFF	16 Kbytes	N/A	Reserved
0xC3FC_8000–0xC3FC_BFFF	16 Kbytes	3 Kbytes (MPC5554) 2.5 Kbytes (MPC5553)	eTPU Shared Data Memory (Parameter RAM)

Table 1-2. Detailed MPC5554/MPC5553 Family Memory Map (continued)

Address Range[1]	Allocated Size[1] (bytes)	Used Size (bytes)	Use
0xC3FC_C000–0xC3FC_FFFF	16 Kbytes	3 Kbytes (MPC5554) 2.5 Kbytes (MPC5553)	eTPU Shared Data Memory (Parameter RAM) mirror
0xC3FD_0000–0xC3FD_3FFF	16 Kbytes	16 Kbytes (MPC5554) 12 Kbytes (MPC5553)	eTPU Shared Code RAM
0xC3FD_4000–0xC3FF_FFFF	176 Kbytes	N/A	Reserved
0xC400_0000–0xDFFF_FFFF	(512 Mbytes–64 Mbytes)	N/A	Reserved
Bridge B Peripherals			
0xE000_0000–0xFBFF_FFFF	(512 Mbytes–64 Mbytes)	N/A	Reserved
0xFC00_0000–0xFFEF_FFFF	63 Mbytes	N/A	Reserved
0xFFF0_0000–0xFFF0_3FFF	16 K	N/A	Bridge B Registers
0xFFF0_4000–0xFFF0_7FFF	16 K	N/A	System Bus Crossbar Switch (XBAR)
0xFFF0__8000–0xFFF0_FFFF	32 K	N/A	Reserved
0xFFF1_0000–0xFFF3_FFFF	192 K	N/A	Reserved
0xFFF4_0000–0xFFF4_3FFF	16 K	N/A	ECSM
0xFFF4_4000–0xFFF4_7FFF	16 K	N/A	DMA Controller 2 (eDMA)
0xFFF4_8000–0xFFF4_BFFF	16 K	N/A	Interrupt Controller (INTC)
0xFFF4_C000–0xFFF4_FFFF	16 K	N/A	Fast Ethernet Controller (FEC)[2] MPC5553 Only
0xFFFC_0000–0xFFF4_FFFF	15 K	N/A	Reserved -- MPC5554 Only
0xFFF5_0000–0xFFF7_FFFF	192 K	N/A	Reserved
0xFFF8_0000–0xFFF8_3FFF	16 Kbytes	164	Enhanced Queued Analog-to-Digital Converter (eQADC)
0xFFF8_4000–0xFFF8_FFFF	48 Kbytes	N/A	Reserved
0xFFF9_0000–0xFFF9_3FFF	16 Kbytes	200	Deserial Serial Peripheral Interface (DSPI_A)[3]
0xFFF9_4000–0xFFF9_7FFF	16 Kbytes	200	Deserial Serial Peripheral Interface (DSPI_B)
0xFFF9_8000–0xFFF9_BFFF	16 Kbytes	200	Deserial Serial Peripheral Interface (DSPI_C)
0xFFF9_C000–0xFFF9_FFFF	16 Kbytes	200	Deserial Serial Peripheral Interface (DSPI_D)
0xFFFA_0000–0xFFFA_FFFF	64 Kbytes	N/A	Reserved
0xFFFB_0000–0xFFFB_3FFF	16 Kbytes	44	Serial Communications Interface (SCI_A)
0xFFFB_4000–0xFFFB_7FFF	16 Kbytes	44	Serial Communications Interface (SCI_B)
0xFFFB_8000–0xFFFB_FFFF	32 Kbytes	N/A	Reserved
0xFFFC_0000–0xFFFC_3FFF	16 Kbytes	1152	Controller Area Network (FlexCAN_A)

Table 1-2. Detailed MPC5554/MPC5553 Family Memory Map (continued)

Address Range[1]	Allocated Size[1] (bytes)	Used Size (bytes)	Use
0xFFFC_4000–0xFFFC_7FFF	16 Kbytes	1152	Controller Area Network (FlexCAN_B)[3]
0xFFFC_8000–0xFFFC_BFFF	16 Kbytes	1152	Controller Area Network (FlexCAN_C)
0xFFFC_C000–0xFFFF_BFFF	192 Kbytes	N/A	Reserved
0xFFFF_C000–0xFFFF_FFFF[4]	16 Kbytes	16 Kbytes	Boot Assist Module (BAM)

1 If allocated size > used size, then the base address for the module is the lowest address of the listed address range, unless noted otherwise.

2 MPC5553 only, not in MPC5554

3 MPC5554 only, not in MPC5553

4 BAM address range is configured so that 4Kbyte BAM occupies 0xFFFF_F000–0xFFFF_FFFF

1.7 Multi-Master Operation Memory Map

When the MPC5553/MPC5554 MCU acts as a slave in a multi-master system, the external bus interface (EBI) translates the 24-bit external address to a 32-bit internal address. Table 1-3 lists the translation parameters.

Table 1-3. External to Internal Memory Map Translation Table for Slave Mode

Ext Addr[8:11][1]	Internal Addr[0:11]	Size (bytes)	Internal Slave	Internal Address Range
0b0xxx	N/A	8 Mbytes	N/A	N/A–Off-chip Flash access
0b10xx	0b0000_0000_00xx	4 Mbytes	Internal FLASH Array	0x0000_0000–0x003F_FFFF
0b1100	0b0100_0000_0000	1 Mbyte	Internal SRAM	0x4000_0000–0x400F_FFFF
0b1101	0b0110_0000_0000	1 Mbyte	Reserved[2]	0x6000_0000–0x600F_FFFF
0b1110	0b1100_0011_1111	1 Mbyte	Bridge A Peripherals	0xC3F0_0000–0xC3FF_FFFF
0b1111	0b1111_1111_1111	1 Mbyte	Bridge B Peripherals	0xFFF0_0000–0xFFFF_FFFF

1 Only the lower 24 address signals (addr[8:31]) are available off-chip.

2 Reserved for a future module that requires its own crossbar slave port.

Table 1-4 shows the memory map for the MPC5553/MPC5554 MCU acting as a slave in a multi-master system from the point of view of the external master.

Table 1-4. MPC5500 Family Slave Memory Map as Seen from an External Master

External Address Range[1]	Size (bytes)	Use
0x00_0000[2]–0x7F_FFFF	8 Mbytes	N/A–Used for off-chip memory accesses
0x80_0000–0x9F_FFFF(MPC5554) 0x80_0000–0x97_FFFF(MPC5553)	2 Mbytes(MPC5554) 1.5 Mbytes(MPC5553)	Slave FLASH[3]
0xA0_0000–0xBF_FFFF(MPC5554)0x98_0000 –0xBF_FFFF(MPC5553)	2 Mbytes(MPC5554) 2.5 Mbytes(MPC5553)	Reserved
0xC0_0000–0xC0_FFFF	64 Kbytes	Slave Internal SRAM
0xC1_0000–0xCF_FFFF	(1 Mbytes–64 Kbytes)	Reserved
0xD0_0000–0xDF_FFFF	1 Mbytes	Reserved
0xE0_0000–0xEF_FFFF	1 Mbytes	Slave Bridge A Peripherals
0xF0_0000–0xFF_FFFF	1 Mbytes	Slave Bridge B Peripherals

1 Only the lower 24 address signals (addr[8:31]) are available off-chip.

2 This address range is not part of the MPC5500 family slave memory map, rather it is shown to illustrate the addressing scheme for off-chip accesses in multi-master mode.

3 The shadow row of the slave FLASH is not accessible by an external master.

Table 1-5 shows the memory map for the MPC5553 and MPC5554family MCU configured as a master in multi-master system with another MPC5500 family MCU acting as the slave.

Table 1-5. MPC5500 Family Master Memory Map (Multi Master Mode)

Base Address	Size (bytes)	Use
On-Chip		
0x0000_0000	2 Mbytes(MPC55545) 1.5 Mbytes(MPC5553)	FLASH Array
0x0020_0000 0x0018_0000	(14 Mbytes–1024 bytes) (14.5 Mbytes – 1024 bytes)	Reserved
0x00FF_FC00	1024	FLASH Shadow Row
0x0100_0000	496 Mbytes	emulation mapping Flash
Off-Chip		
0x2000_0000	8 Mbytes[1]	External Memory
0x2080_0000	2 Mbytes	Slave FLASH
0x20A0_0000 0x2098_0000	2 Mbytes	Reserved
Not Addressable	1024	Slave FLASH Shadow Row
0x20C0_0000	64 Kbytes	Slave Internal SRAM
0x20C1_0000	(2 Mbytes–64 Kbytes)	Reserved
0x20E0_0000	1 Mbytes	Slave Bridge A Peripherals

Table 1-5. MPC5500 Family Master Memory Map (Multi Master Mode) (continued)

Base Address	Size (bytes)	Use
0x20F0_0000	1 Mbytes	Slave Bridge B Peripherals
On-Chip		
0x4000_0000	96 Kbytes	Internal SRAM
0x4001_8000	(2048 Mbytes–96 Kbytes)	Reserved
0xC000_0000	63 Mbytes	Reserved
0xC3F0_0000	1 Mbytes	Bridge A Peripherals
0xC400_0000	(1024 Mbytes–128 Mbytes)	Reserved
0xFC00_0000	63 Mbytes	Reserved
0xFFF0_0000	1 Mbyte	Bridge B Peripherals

[1] By using the 4 chip select signals, 32 Mbytes of external memory can be accessed by the master in a multi-master system.

1.8 Revision History

Substantive Changes since Rev 3.0
Features list - changed “40%/70% V_{DDE} CMOS switch levels (with hysteresis)” to be 35%/65%
Fixed reference to CAL_CS[0:2]. On MPC5553, CAL_CS[1] is not implemented.
Fixed a typo - “308212 interrupts” - split it to 308 for MPC5554 and 212 for MPC5553.
Updated both MPC5553 and MPC5554 block diagrams.
In features list, changed “96 kilobyte general-purpose RAM of which 32 kilobytes are on standby power” to be 64 kilobyte.

Chapter 2
Signal Description

This chapter describes the signals of the MPC5553 and the MPC5554 that connect off chip. It includes a table of signal properties, detailed descriptions of signals, and the I/O pin power/ground segmentation.

2.1 Block Diagram

Figure 2-1 shows the signals of the MPC5553, and Figure 2-2 shows the signals of the MPC5554.

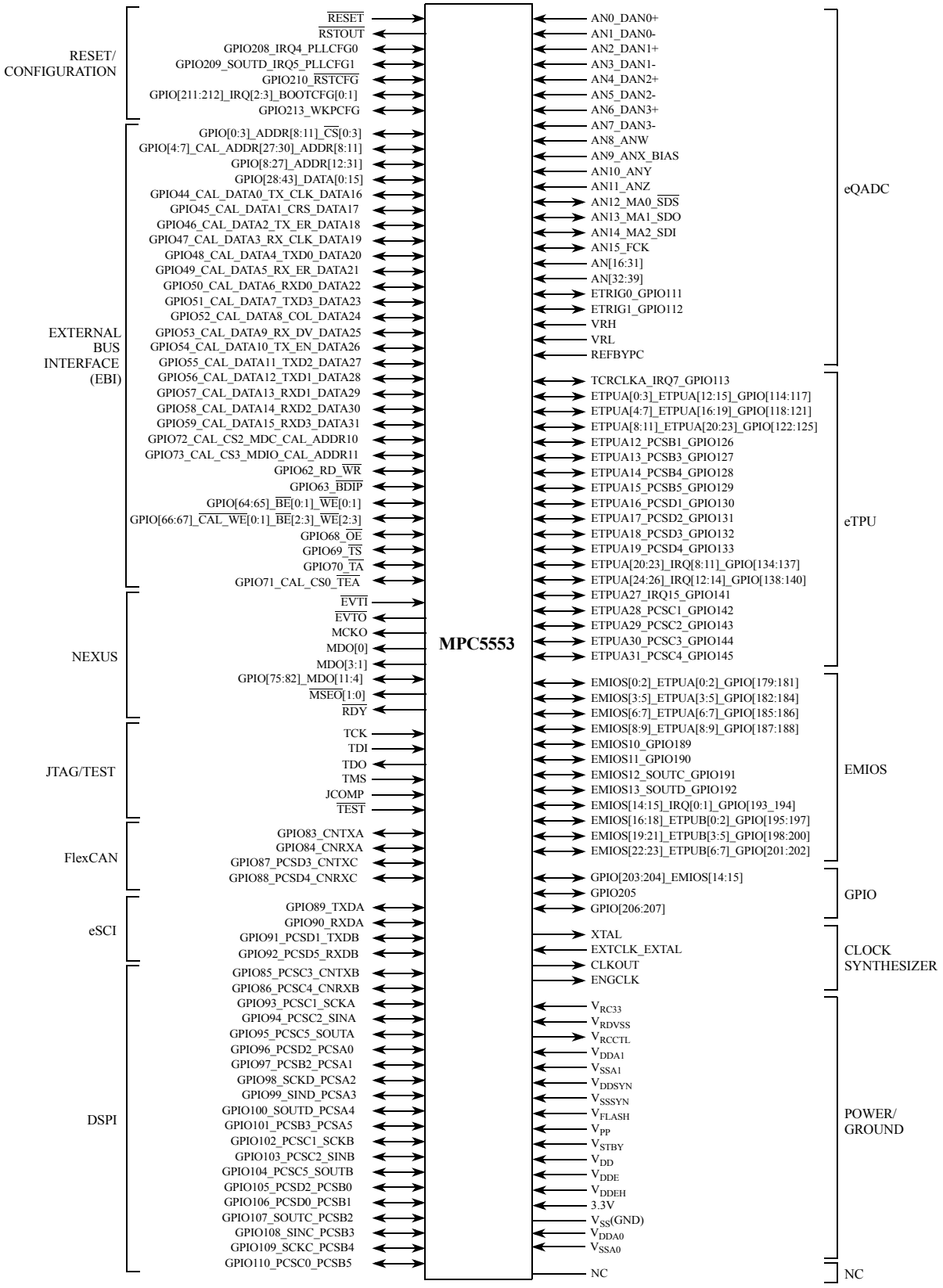


Figure 2-1. MPC5553 Signal Diagram

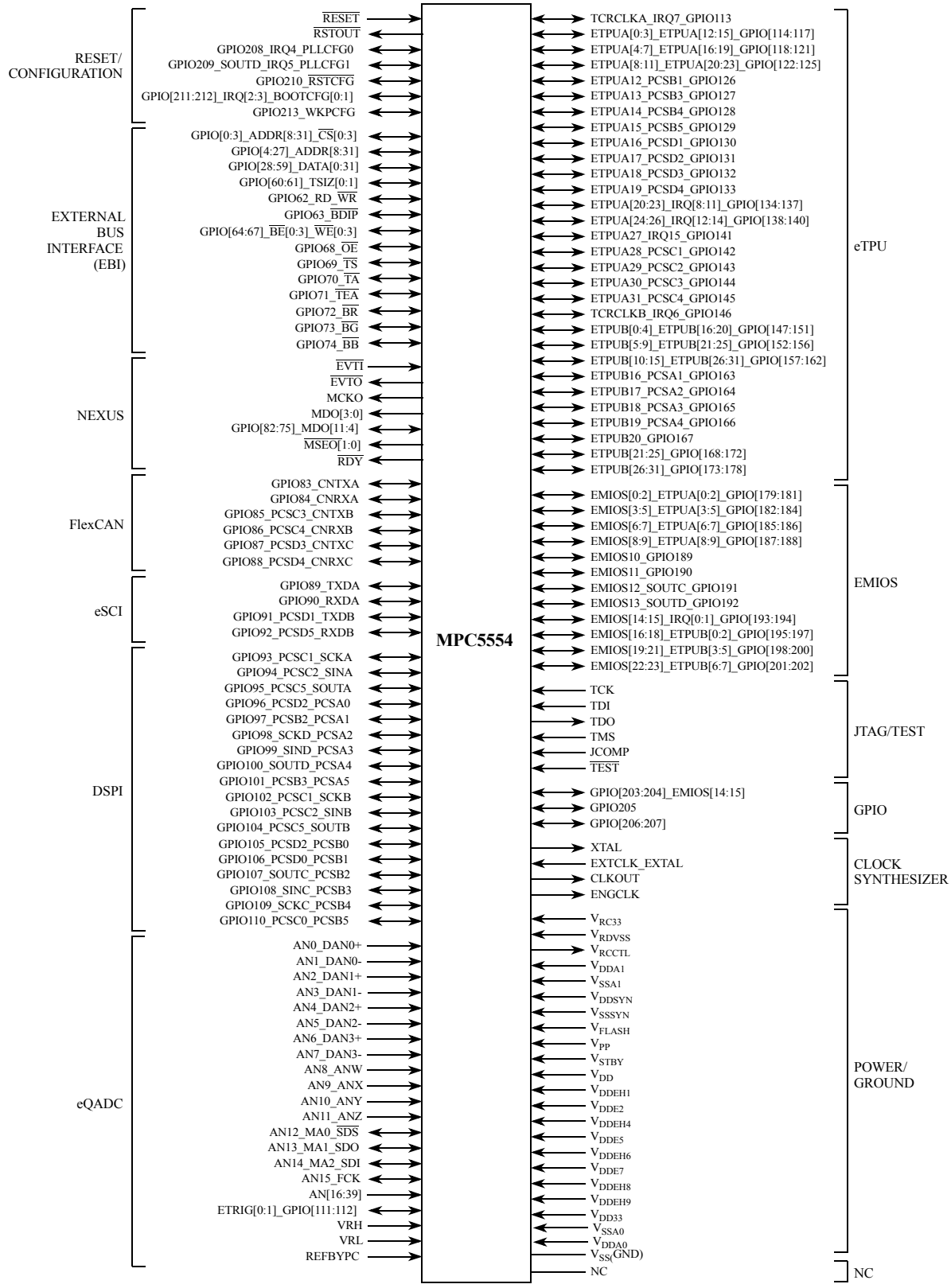


Figure 2-2. MPC5554 Signal Diagram

2.2 External Signal Description

Table 2-1 gives a summary of the MPC5553 external signals and properties, and Table 2-2 provides a summary of the MPC5554 external signals and properties. The Signal and Range column lists the signal name and range of each signal. The Function column lists all the functions multiplexed on a pin, beginning with the primary function. For example, for the pin CNTXB_PCSC3_GPIO85, CNTXB is the primary function, PCSC3 is the alternate function, and GPIO85 is the GPIO.

2.2.1 MPC5553 Signals Summary

Table 2-1 gives a summary of the MPC5553 external signals and properties.

Table 2-1. MPC5553 Signal Properties

Signal and Range	P/A/G	Function[1]	Description	I/O Type	Voltage[2]	Pad Type[3]	Status During Reset[4]	Status After Reset[5]	Package
Reset / Configuration (8)									
$\overline{\text{RESET}}$	P	RESET	External reset input	I	V_{DDEH6}	S	$\overline{\text{RESET}}$ / Up	$\overline{\text{RESET}}$ / Up	416 324 208
$\overline{\text{RSTOUT}}$	P	RSTOUT	External Reset Output	O	V_{DDEH6}	S	$\overline{\text{RSTOUT}}$ / Low	$\overline{\text{RSTOUT}}$ / High	416 324 208
PLLCFG0	P A G	PLLCFG0 IRQ4 GPIO208	FMPLL Mode Selection External Interrupt Request GPIO	I I I/O	V_{DDEH6}	M	PLLCFG / Up	— / Up	416 324 208
PLLCFG1	P A A2 G	PLLCFG1 IRQ5 SOUTD GPIO209	FMPLL mode selection External Interrupt Request DSPI D Data Output GPIO	I I O I/O	V_{DDEH6}	M	PLLCFG / Up	— / Up	416 324 208
$\overline{\text{RSTCFG}}$	P G	$\overline{\text{RSTCFG}}$ GPIO210	Reset configuration input GPIO	I I/O	V_{DDEH6}	S	$\overline{\text{RSTCFG}}$ / Up	— / Up	416 324
BOOTCFG0[6]	P A G	BOOTCFG0[6] IRQ2 GPIO211	Boot configuration input[6] External interrupt request GPIO	I I I/O	V_{DDEH6}	S	BOOTCFG / Down	— / Down	416 324
BOOTCFG1	P A G	BOOTCFG1 IRQ3 GPIO212	Boot configuration input External interrupt request GPIO	I I I/O	V_{DDEH6}	S	BOOTCFG / Down	— / Down	416 324 208
WKPCFG	P G	WKPCFG GPIO213	Weak pull configuration input GPIO	I I/O	V_{DDEH6}	S	WKPCFG / Up	— / Up	416 324 208
External Bus Interface (EBI)[7] (72)									
$\overline{\text{CS}}$[0]	P A G	$\overline{\text{CS}}$[0] ADDR[8][8] GPIO[0]	External chip selects External address bus[8] GPIO	O I/O I/O	V_{DDE2}	F	— / Up	— / Up[9]	416 324 208

Table 2-1. MPC5553 Signal Properties (continued)

Signal and Range	P/A/G	Function[1]	Description	I/O Type	Voltage[2]	Pad Type[3]	Status During Reset[4]	Status After Reset[5]	Package
$\overline{CS}$[1:3]	P A G	$\overline{CS}$[1:3] ADDR[9:11][8] GPIO[1:3]	External chip selects External address bus[8] GPIO	O I/O I/O	V_{DDE2}	F	— / Up	— / Up[9]	416 324
ADDR [8:11]	P A G	ADDR[8:11] [8] CAL_ADDR [27:30] GPIO[4:7]	External Address Bus[8, 10] Calibration Address Bus GPIO	I/O O I/O	V_{DDE2}	F	— / Up	— / Up[9]	416
ADDR [12:31]	P G	ADDR[12:31] GPIO[8:27]	External Address Bus[10] GPIO	I/O I/O	V_{DDE2}	F	— / Up	— / Up[9]	416 324
DATA [0:15]	P G	DATA[0:15] GPIO[28:43]	External Data Bus[10] GPIO	I/O I/O	V_{DDE3}	F	— / Up	— / Up[9]	416 324
DATA16	P A A2 G	DATA16 TX_CLK CAL_DATA0 GPIO44	External Data Bus[10] FEC Transmit Clock Calibration Data Bus GPIO	I/O I I/O I/O	V_{DDE3}	F	— / Up	— / Up[9]	416
DATA17	P A A2 G	DATA17 CRS CAL_DATA1 GPIO45	External Data Bus[10] FEC Carrier Sense Calibration Data Bus GPIO	I/O I I/O I/O	V_{DDE3}	F	— / Up	— / Up[9]	416
DATA18	P A A2 G	DATA18 TX_ER CAL_DATA2 GPIO46	External Data Bus[10] FEC Transmit Error Calibration Data Bus GPIO	I/O O I/O I/O	V_{DDE3}	F	— / Up	— / Up[9]	416
DATA19	P A A2 G	DATA19 RX_CLK CAL_DATA3 GPIO47	External Data Bus[10] FEC Receive Clock Calibration Data Bus GPIO	I/O I I/O I/O	V_{DDE3}	F	— / Up	— / Up[9]	416
DATA20	P A A2 G	DATA20 TXD0 CAL_DATA4 GPIO48	External Data Bus[10] FEC Transmit Data Calibration Data Bus GPIO	I/O O I/O I/O	V_{DDE3}	F	— / Up	— / Up[9]	416
DATA21	P A A2 G	DATA21 RX_ER CAL_DATA5 GPIO49	External Data Bus[10] FEC Receive Error Calibration Data Bus GPIO	I/O I I/O I/O	V_{DDE3}	F	— / Up	— / Up[9]	416
DATA22	P A A2 G	DATA22 RXD0 CAL_DATA6 GPIO50	External Data Bus[10] FEC Receive Data Calibration Data Bus GPIO	I/O I I/O I/O	V_{DDE3}	F	— / Up	— / Up[9]	416
DATA23	P A A2 G	DATA23 TXD3 CAL_DATA7 GPIO51	External Data Bus[10] FEC Transmit Data Calibration Data Bus GPIO	I/O O I/O I/O	V_{DDE3}	F	— / Up	— / Up[9]	416

Table 2-1. MPC5553 Signal Properties (continued)

Signal and Range	P/A/G	Function[1]	Description	I/O Type	Voltage[2]	Pad Type[3]	Status During Reset[4]	Status After Reset[5]	Package
DATA24	P A A2 G	DATA24 COL CAL_DATA8 GPIO52	External Data Bus[10] FEC Collision Detect Calibration Data Bus GPIO	I/O I I/O I/O	V_{DDE3}	F	— / Up	— / Up[9]	416
DATA25	P A A2 G	DATA25 RX_DV CAL_DATA9 GPIO53	External Data Bus[10] FEC Receive Data Valid Calibration Data Bus GPIO	I/O I I/O I/O	V_{DDE3}	F	— / Up	— / Up[9]	416
DATA26	P A A2 G	DATA26 TX_EN CAL_DATA10 GPIO54	External Data Bus[10] FEC Transmit Enable Calibration Data Bus GPIO	I/O O I/O I/O	V_{DDE3}	F	— / Up	— / Up[9]	416
DATA27	P A A2 G	DATA27 TXD2 CAL_DATA11 GPIO55	External Data Bus[10] FEC Transmit Data Calibration Data Bus GPIO	I/O O I/O I/O	V_{DDE3}	F	— / Up	— / Up[9]	416
DATA28	P A A2 G	DATA28 TXD1 CAL_DATA12 GPIO56	External Data Bus[10] FEC Transmit Data Calibration Data Bus GPIO	I/O O I/O I/O	V_{DDE3}	F	— / Up	— / Up[9]	416
DATA29	P A A2 G	DATA29 RXD1 CAL_DATA13 GPIO57	External Data Bus[10] FEC Receive Data Calibration Data Bus GPIO	I/O I I/O I/O	V_{DDE3}	F	— / Up	— / Up[9]	416
DATA30	P A A2 G	DATA30 RXD2 CAL_DATA14 GPIO58	External Data Bus[10] FEC Receive Data Calibration Data Bus GPIO	I/O I I/O I/O	V_{DDE3}	F	— / Up	— / Up[9]	416
DATA31	P A A2 G	DATA31 RXD3 CAL_DATA15 GPIO59	External Data Bus[10] FEC Receive Data Calibration Data Bus GPIO	I/O I I/O I/O	V_{DDE3}	F	— / Up	— / Up[9]	416
RD_$\overline{WR}$	P G	RD_$\overline{WR}$ GPIO62	External Read/Write GPIO	I/O I/O	V_{DDE2}	F	— / Up	— / Up[9]	416 324
$\overline{BDIP}$	P G	$\overline{BDIP}$ GPIO63	External Burst Data In Progress GPIO	O I/O	V_{DDE2}	F	— / Up	— / Up[9]	416 324
$\overline{WE}$[0:1]	P A G	$\overline{WE}$[0:1] $\overline{BE}$[0:1] GPIO[64:65]	External Write Enable External Byte Enable[11] GPIO	O O I/O	V_{DDE2}	F	— / Up	— / Up[9]	416 324

Table 2-1. MPC5553 Signal Properties (continued)

Signal and Range	P/A/G	Function[1]	Description	I/O Type	Voltage[2]	Pad Type[3]	Status During Reset[4]	Status After Reset[5]	Package
$\overline{WE}$[2:3]	P A A2 A3 G	$\overline{WE}$[2:3] $\overline{BE}$[2:3] $\overline{CAL\_WE}$[0:1] $\overline{CAL\_BE}$[0:1 GPIO[66:67]	External Write Enable External Byte Enable[11] Calibration Write Enable Calibration Byte Enable GPIO	O O O O I/O	V_{DDE2}	F	— / Up	— / Up[9]	416
$\overline{OE}$	P G	$\overline{OE}$ GPIO68	External Output Enable GPIO	O I/O	V_{DDE3}	F	— / Up	— / Up[9]	416 324 208
$\overline{TS}$	P G	$\overline{TS}$ GPIO69	External Transfer Start GPIO	I/O I/O	V_{DDE2}	F	— / Up	— / Up[9]	416 324
$\overline{TA}$	P G	$\overline{TA}$ GPIO70	External Transfer Acknowledge GPIO	I/O I/O	V_{DDE2}	F	— / Up	— / Up[9]	416 324
$\overline{TEA}$	P A G	$\overline{TEA}$ $\overline{CAL\_CS0}$ GPIO71	External Transfer Error Acknowledge Calibration Chip Select GPIO	I/O O I/O	V_{DDE2}	F	— / Up	— / Up[9]	416
$\overline{BR}$[12] (CAL_ADDR10)	— P A A2 G	CAL_ADDR10 MDC $\overline{CAL\_CS2}$ GPIO72	Calibration Address Bus FEC Management Clock Calibration Chip Select GPIO	O O O I/O	V_{DDE3}	F	— / Up	— / Up[9]	416
$\overline{BG}$[12] (CAL_ADDR11)	— P A A2 G	CAL_ADDR11 MDIO $\overline{CAL\_CS3}$ GPIO73	Calibration Address Bus FEC Management Data I/O Calibration Chip Select GPIO	O I/O I/O I/O	V_{DDE3}	F	— / Up	— / Up[9]	416
NEXUS (18)									
$\overline{EVTI}$	P	$\overline{EVTI}$	Nexus Event In	I	V_{DDE7}	F	I / Up	EVTI / Up	416 324 208
$\overline{EVTO}$	P	$\overline{EVTO}$	Nexus Event Out	O	V_{DDE7}	F	O / Low	EVTO / High	416 324 208
MCKO	P	MCKO	Nexus Message Clock Out	O	V_{DDE7}	F	O / Low	MCKO / Enabled[13]	416 324 208
MDO[0]	P	MDO[0][14]	Nexus Message Data Out	O	V_{DDE7}	F	O / High	MDO / Low	416 324 208
MDO[3:1]	P	MDO[3:1]	Nexus Message Data Out	O	V_{DDE7}	F	O / Low	MDO / Low	416 324 208

Table 2-1. MPC5553 Signal Properties (continued)

Signal and Range	P/A/G	Function[1]	Description	I/O Type	Voltage[2]	Pad Type[3]	Status During Reset[4]	Status After Reset[5]	Package
MDO[11:4]	P G	MDO[11:4] [15] GPIO[75:82]	Nexus Message Data Out GPIO	O I/O	V_{DDE7}	F	O / Low	— / Down	416 324
$\overline{\text{MSEO}}$[1:0]	P	$\overline{\text{MSEO}}$[1:0]	Nexus Message Start/End Out	O	V_{DDE7}	F	O / High	$\overline{\text{MSEO}}$ / High	416 324 208
$\overline{\text{RDY}}$	P	$\overline{\text{RDY}}$	Nexus Ready Output	O	V_{DDE7}	F	O / High	$\overline{\text{RDY}}$ / High	416 324
JTAG / TEST(6)									
TCK	P	TCK	JTAG Test Clock Input	I	V_{DDE7}	F	TCK / Down	TCK / Down	416 324 208
TDI	P	TDI	JTAG Test Data Input	I	V_{DDE7}	F	TDI / Up	TDI / Up	416 324 208
TDO	P	TDO	JTAG Test Data Output	O	V_{DDE7}	F	TDO / Up	TDO / Up	416 324 208
TMS	P	TMS	JTAG Test Mode Select Input	I	V_{DDE7}	F	TMS / Up	TMS / Up	416 324 208
JCOMP	P	JCOMP	JTAG TAP Controller Enable	I	V_{DDE7}	F	JCOMP / Down	JCOMP / Down	416 324 208
$\overline{\text{TEST}}$	P	$\overline{\text{TEST}}$	Test Mode Select	I	V_{DDE7}	F	$\overline{\text{TEST}}$ / Up	$\overline{\text{TEST}}$ / Up	416 324 208
FlexCAN (4)									
CNTXA	P G	CNTXA GPIO83	CAN_A Transmit GPIO	O I/O	V_{DDEH4}	S	— / Up	— / Up[16]	416 324 208
CNRXA	P G	CNRXA GPIO84	CAN_A Receive GPIO	I I/O	V_{DDEH4}	S	— / Up	— / Up	416 324 208
CNTXC	P A G	CNTXC PCSD3 GPIO87	CAN_C Transmit DSPI D Peripheral Chip Select 3 GPIO	O O I/O	V_{DDEH6}	M	— / Up	— / Up	416 324 208
CNRXC	P A G	CNRXC PCSD4 GPIO88	CAN_C Receive DSPI D Peripheral Chip Select 4 GPIO	I O I/O	V_{DDEH6}	M	— / Up	— / Up	416 324 208

Table 2-1. MPC5553 Signal Properties (continued)

Signal and Range	P/A/G	Function[1]	Description	I/O Type	Voltage[2]	Pad Type[3]	Status During Reset[4]	Status After Reset[5]	Package
SCI (4)									
TXDA	P G	TXDA GPIO89	SCI_A Transmit GPIO	O I/O	V_{DDEH6}	S	— / Up	— / Up	416 324 208
RXDA	P G	RXDA GPIO90	SCI_A Receive GPIO	I I/O	V_{DDEH6}	S	— / —	— / Up	416 324 208
TXDB	P A G	TXDB PCSD1 GPIO91	SCI_B Transmit DSPI D Peripheral Chip Select 1 GPIO	O O I/O	V_{DDEH6}	M	— / Up	— / Up	416 324 208
RXDB	P A G	RXDB PCSD5 GPIO92	SCI_B Receive DSPI D Peripheral Chip Select 5 GPIO	I O I/O	V_{DDEH6}	M	— / Up	— / Up	416 324 208
DSPI(20)									
CNTXB	P A G	CNTXB PCSC3 GPIO85	CAN_B Transmit (not functional) DSPI C Peripheral Chip Select 3 GPIO	O O I/O	V_{DDEH4}	M	— / Up	— / Up	416 324 208
CNRXB	P A G	CNRXB PCSC4 GPIO86	CAN_B Receive (not functional) DSPI C Peripheral Chip Select 4 GPIO	I O I/O	V_{DDEH4}	M	— / Up	— / Up	416 324 208
SCKA	P A G	SCKA PCSC1 GPIO93	— DSPI C Peripheral Chip Select 1 GPIO	— O I/O	V_{DDEH6}	M	— / Up	— / Up	416 324
SINA	P A G	SINA PCSC2 GPIO94	— DSPI C Peripheral Chip Select 2 GPIO	— O I/O	V_{DDEH6}	M	— / Up	— / Up	416 324
SOUTA	P A G	SOUTA PCSC5 GPIO95	— DSPI C Peripheral Chip Select 5 GPIO	— O I/O	V_{DDEH6}	M	— / Up	— / Up	416 324
PCSA0	P A G	PCSA0 PCSD2 GPIO96	— DSPI D Peripheral Chip Select 2 GPIO	— O I/O	V_{DDEH6}	M	— / Up	— / Up	416 324

Table 2-1. MPC5553 Signal Properties (continued)

Signal and Range	P/A/G	Function[1]	Description	I/O Type	Voltage[2]	Pad Type[3]	Status During Reset[4]	Status After Reset[5]	Package
PCSA1	P A G	PCSA1 PCSB2 GPIO97	— DSPI B Peripheral Chip Select 2 GPIO	— O I/O	V_{DDEH6}	M	— / Up	— / Up	416 324
PCSA2	P A G	PCSA2 SCKD GPIO98	— DSPI D Clock GPIO	— I/O I/O	V_{DDEH6}	M	— / Up	— / Up	416 324 208
PCSA3	P A G	PCSA3 SIND GPIO99	— DSPI D Data Input GPIO	— I I/O	V_{DDEH6}	M	— / Up	— / Up	416 324 208
PCSA4	P A G	PCSA4 SOUTD GPIO100	— DSPI D Data Output GPIO	— O I/O	V_{DDEH6}	M	— / Up	— / Up	416 324
PCSA5	P A G	PCSA5 PCSB3 GPIO101	— DSPI B Peripheral Chip Select 3 GPIO	— O I/O	V_{DDEH6}	M	— / Up	— / Up	416 324
SCKB	P A G	SCKB PCSC1 GPIO102	DSPI B Clock DSPI C Peripheral Chip Select 1 GPIO	I/O O I/O	V_{DDEH10}	M	— / Up	— / Up	416 324 208
SINB	P A G	SINB PCSC2 GPIO103	DSPI B Data Input DSPI C Peripheral Chip Select 2 GPIO	I O I/O	V_{DDEH10}	M	— / Up	— / Up	416 324 208
SOUTB	P A G	SOUTB PCSC5 GPIO104	DSPI B Data Output DSPI C Peripheral Chip Select 5 GPIO	O O I/O	V_{DDEH10}	M	— / Up	— / Up	416 324 208
PCSB0	P A G	PCSB0 PCSD2 GPIO105	DSPI B Peripheral Chip Select 0 DSPI D Peripheral Chip Select 2 GPIO	I/O O I/O	V_{DDEH10}	M	— / Up	— / Up	416 324 208
PCSB1	P A G	PCSB1 PCSD0 GPIO106	DSPI B Peripheral Chip Select 1 DSPI D Peripheral Chip Select 0 GPIO	O I/O I/O	V_{DDEH10}	M	— / Up	— / Up	416 324 208
PCSB2	P A G	PCSB2 SOUTC GPIO107	DSPI B Peripheral Chip Select 2 DSPI C Data Output GPIO	O O I/O	V_{DDEH10}	M	— / Up	— / Up	416 324 208

Table 2-1. MPC5553 Signal Properties (continued)

Signal and Range	P/A/G	Function[1]	Description	I/O Type	Voltage[2]	Pad Type[3]	Status During Reset[4]	Status After Reset[5]	Package
PCSB3	P A G	PCSB3 SINC GPIO108	DSPI B Peripheral Chip Select 3 DSPI C Data Input GPIO	O I I/O	V_{DDEH6}	M	— / Up	— / Up	416 324 208
PCSB4	P A G	PCSB4 SCKC GPIO109	DSPI B Peripheral Chip Select 4 DSPI C Clock GPIO	O I/O I/O	V_{DDEH6}	M	— / Up	— / Up	416 324 208
PCSB5	P A G	PCSB5 PCSC0 GPIO110	DSPI B Peripheral Chip Select 5 DSPI C Peripheral Chip Select 0 GPIO	O I/O I/O	V_{DDEH6}	M	— / Up	— / Up	416 324 208
eQADC(45)									
AN0	P A	AN0 DAN0+	Single Ended Analog Input 0 Positive Terminal Differential Input	I I	V_{DDA1}[17]	AE	I / —	AN0/ —	416 324 208
AN1	P A	AN1 DAN0-	Single Ended Analog Input 1 Negative Terminal Differential Input	I I	V_{DDA1}[17]	AE	I / —	AN1/ —	416 324 208
AN2	P A	AN2 DAN1+	Single Ended Analog Input 2 Positive Terminal Differential Input	I I	V_{DDA1}[17]	AE	I / —	AN2 / —	416 324 208
AN3	P A	AN3 DAN1-	Single Ended Analog Input 3 Negative Terminal Differential Input	I I	V_{DDA1}[17]	AE	I / —	AN3 / —	416 324 208
AN4	P A	AN4 DAN2+	Single Ended Analog Input 4 Positive Terminal Differential Input	I I	V_{DDA1}[17]	AE	I / —	AN4/ —	416 324 208
AN5	P A	AN5 DAN2-	Single Ended Analog Input 5 Negative Terminal Differential Input	I I	V_{DDA1}[17]	AE	I / —	AN5 / —	416 324 208
AN6	P A	AN6 DAN3+	Single Ended Analog Input 6 Positive Terminal Differential Input	I I	V_{DDA1}[17]	AE	I / —	AN6 / —	416 324 208

Table 2-1. MPC5553 Signal Properties (continued)

Signal and Range	P/A/G	Function[1]	Description	I/O Type	Voltage[2]	Pad Type[3]	Status During Reset[4]	Status After Reset[5]	Package
AN7	P A	AN7 DAN3-	Single Ended Analog Input 7 Negative Terminal Differential Input	I I	V_{DDA1}[17]	AE	I / —	AN7 / —	416 324 208
AN8	P A	AN8 ANW	Single Ended Analog Input 8 External Multiplexed Analog Input W	I I	V_{DDA1}[17]	AE	I / —	AN8/ —	416 324
AN9	P A A	AN9 ANX	Single Ended Analog Input 9 External Multiplexed Analog Input X	I I	V_{DDA1}[17]	AE	I / —	AN9 / —	416 324 208
AN10	P A	AN10 ANY	Single Ended Analog Input 10 External Multiplexed Analog Input Y	I I	V_{DDA1}[17]	AE	I / —	AN10/ —	416 324
AN11	P A	AN11 ANZ	Single Ended Analog Input 11 External Multiplexed Analog Input Z	I I	V_{DDA1}[17]	AE	I / —	AN11 / —	416 324 208
AN12	P A A	AN12 MA0 $\overline{SDS}$	Single Ended Analog Input 12 Mux Address 0 eQADC Serial Data Select	I O O	V_{DDEH9}	A, M	I / —	AN12/ —	416 324 208
AN13	P A A	AN13 MA1 SDO	Single Ended Analog Input 13 Mux Address 1 eQADC Serial Data Out	I O O	V_{DDEH9}	A, M	I / —	AN13/ —	416 324 208
AN14	P A A	AN14 MA2 SDI	Single Ended Analog Input 14 Mux Address 2 eQADC Serial Data In	I O I	V_{DDEH9}	A, M	I / —	AN14/ —	416 324 208
AN15	P A	AN15 FCK	Single Ended Analog Input 15 eQADC Free Running Clock	I O	V_{DDEH9}	A, M	I / —	AN15/ —	416 324 208
AN[16:18]	P	AN[16:18]	Single Ended Analog Input 16-18	I	V_{DDA1}[17]	AE	I / —	AN[16:18]/ —	416 324 208
AN[19:20]	P	AN[19:20]	Single Ended Analog Input 19-20	I	V_{DDA1}[17]	AE	I / —	AN[19:20]/ —	416 324
AN21	P	AN21	Single Ended Analog Input 21	I	V_{DDA1}[17]	AE	I / —	AN21/ —	416 324 208

Table 2-1. MPC5553 Signal Properties (continued)

Signal and Range	P/A/G	Function[1]	Description	I/O Type	Voltage[2]	Pad Type[3]	Status During Reset[4]	Status After Reset[5]	Package
AN[22:25]	P	AN[22:25]	Single Ended Analog Input 22-25	I	V_{DDA0}[17]	AE	I / —	AN[22:25]/ —	416 324 208
AN26	P	AN26	Single Ended Analog Input 26	I	V_{DDA0}[17]	AE	I / —	AN26/ —	416 324
AN[27:28]	P	AN[27:28]	Single Ended Analog Input 27-28	I	V_{DDA0}[17]	AE	I / —	AN[27:28]/ —	416 324 208
AN29	P	AN29	Single Ended Analog Input 29	I	V_{DDA0}[17]	AE	I / —	AN29/ —	416 324
AN[30:35]	P	AN[30:35]	Single Ended Analog Input 30-35	I	V_{DDA0}[17]	AE	I / —	AN[30:35]/ —	416 324 208
AN[36:39]	P	AN[36:39]	Single Ended Analog Input 36-39	I	V_{DDA1}[17]	AE	I / —	AN[36:39] / —	416 324 208
ETRIG[0:1]	P G	ETRIG[0:1] GPIO[111:112]	eQADC Trigger Input 0, 1 GPIO	I I/O	V_{DDEH8}	S	— / Up	— / Up	416
VRH	P	VRH	Voltage Reference High	I	V_{DDA0}[17]	VDDINT	— / —	VRH	416 324 208
VRL	P	VRL	Voltage Reference Low	I	V_{DDA0}[17]	VSSINT	— / —	VRL	416 324 208
REFBYPC	P	REFBYPC	Reference Bypass Capacitor Input	I	V_{DDA0}[17]	AE	— / —	REFBYPC	416 324 208
eTPU(33)									
TCRCLKA	P A G	TCRCLKA IRQ7 GPIO113	eTPU A TCR clock External interrupt request GPIO	I I I/O	V_{DDEH1}	S	— / Up	— / Up	416 324 208
ETPUA [0:3]	P A G	ETPUA[0:3] ETPUA[12:15] GPIO[114:117]	eTPU A channel eTPU A channel (output only) GPIO	I/O O I/O	V_{DDEH1}	S	— / WKPCFG	— / WKPCFG	416 324 208
ETPUA [4:7]	P A G	ETPUA[4:7] ETPUA[16:19] GPIO[118:121]	eTPU A channel eTPU A channel (output only) GPIO	I/O O I/O	V_{DDEH1}	S	— / WKPCFG	— / WKPCFG	416 324 208

Table 2-1. MPC5553 Signal Properties (continued)

Signal and Range	P/A/G	Function[1]	Description	I/O Type	Voltage[2]	Pad Type[3]	Status During Reset[4]	Status After Reset[5]	Package
ETPUA [8:11]	P A G	ETPUA[8:11] ETPUA[20:23] GPIO[122:125]	eTPU A channel eTPU A channel (output only) GPIO	I/O O I/O	V_{DDEH1}	S	— / WKPCFG	— / WKPCFG	416 324 208
ETPUA12	P A G	ETPUA12 PCSB1 GPIO126	eTPU A channel DSPI B peripheral chip select 1 GPIO	I/O O I/O	V_{DDEH1}	M	— / WKPCFG	— / WKPCFG	416 324 208
ETPUA13	P A G	ETPUA13 PCSB3 GPIO127	eTPU A channel DSPI B peripheral chip select 3 GPIO	I/O O I/O	V_{DDEH1}	M	— / WKPCFG	— / WKPCFG	416 324 208
ETPUA14	P A G	ETPUA14 PCSB4 GPIO128	eTPU A channel DSPI B peripheral chip select 4 GPIO	I/O O I/O	V_{DDEH1}	M	— / WKPCFG	— / WKPCFG	416 324 208
ETPUA15	P A G	ETPUA15 PCSB5 GPIO129	eTPU A channel DSPI B peripheral chip select 5 GPIO	I/O O I/O	V_{DDEH1}	M	— / WKPCFG	— / WKPCFG	416 324 208
ETPUA16	P A G	ETPUA16 PCSD1 GPIO130	eTPU A channel DSPI D peripheral chip select 1 GPIO	I/O O I/O	V_{DDEH1}	M	— / WKPCFG	— / WKPCFG	416 324 208
ETPUA17	P A G	ETPUA17 PCSD2 GPIO131	eTPU A channel DSPI D peripheral chip select 2 GPIO	I/O O I/O	V_{DDEH1}	M	— / WKPCFG	— / WKPCFG	416 324 208
ETPUA18	P A G	ETPUA18 PCSD3 GPIO132	eTPU A channel DSPI D peripheral chip select 3 GPIO	I/O O I/O	V_{DDEH1}	M	— / WKPCFG	— / WKPCFG	416 324 208
ETPUA19	P A G	ETPUA19 PCSD4 GPIO133	eTPU A channel DSPI D peripheral chip select 4 GPIO	I/O O I/O	V_{DDEH1}	M	— / WKPCFG	— / WKPCFG	416 324 208
ETPUA [20:23]	P A G	ETPUA[20:23] IRQ[8:11] GPIO[134:137]	eTPU A channel External interrupt request GPIO	I/O I I/O	V_{DDEH1}	M	— / WKPCFG	— / WKPCFG	416 324 208
ETPUA [24:26]	P A G	ETPUA[24:26] IRQ[12:14] GPIO[138:140]	eTPU A channel (output only) External interrupt request GPIO	O I I/O	V_{DDEH1}	S	— / WKPCFG	— / WKPCFG	416 324 208

Table 2-1. MPC5553 Signal Properties (continued)

Signal and Range	P/A/G	Function[1]	Description	I/O Type	Voltage[2]	Pad Type[3]	Status During Reset[4]	Status After Reset[5]	Package
ETPUA27	P A G	ETPUA27 IRQ15 GPIO141	eTPU A channel (output only) External interrupt request GPIO	O I I/O	V_{DDEH1}	S	— / WKPCFG	— / WKPCFG	416 324 208
ETPUA28	P A G	ETPUA28 PCSC1 GPIO142	eTPU A Channel (Output Only) DSPI C peripheral chip select 1 GPIO	O O I/O	V_{DDEH1}	M	— / WKPCFG	— / WKPCFG	416 324 208
ETPUA29	P A G	ETPUA29 PCSC2 GPIO143	eTPU A Channel (Output Only) DSPI C peripheral chip select 2 GPIO	O O I/O	V_{DDEH1}	M	— / WKPCFG	— / WKPCFG	416 324 208
ETPUA30	P A G	ETPUA30 PCSC3 GPIO144	eTPU A Channel DSPI C peripheral chip select 3 GPIO	I/O O I/O	V_{DDEH1}	M	— / WKPCFG	— / WKPCFG	416 324 208
ETPUA31	P A G	ETPUA31 PCSC4 GPIO145	eTPU A Channel DSPI C peripheral chip select 4 GPIO	I/O O I/O	V_{DDEH1}	M	— / WKPCFG	— / WKPCFG	416 324 208
EMIOS(24)									
EMIOS [0:9]	P A G	EMIOS[0:9] ETPUA[0:9] GPIO[179:188]	eMIOS channel eTPU A channel (output only) GPIO	I/O O I/O	V_{DDEH4}	S	— / WKPCFG	— / WKPCFG	416 324 208
EMIOS[10:11]	P G	EMIOS[10:11] GPIO[189:190]	eMIOS channel GPIO	I/O I/O	V_{DDEH4}	S	— / WKPCFG	— / WKPCFG	416 324 208
EMIOS12	P A G	EMIOS12 SOUTC GPIO191	EMIOS Channel (Output Only) DSPI C Data Output GPIO	O O I/O	V_{DDEH4}	M	— / WKPCFG	— / WKPCFG	416 324 208
EMIOS13	P A G	EMIOS13 SOUTD GPIO192	EMIOS Channel (Output Only) DSPI D Data Output GPIO	O O I/O	V_{DDEH4}	M	— / WKPCFG	— / WKPCFG	416 324 208
EMIOS [14:15]	P A G	EMIOS[14:15] IRQ[0:1] GPIO[193:194]	eMIOS channel (output only) External interrupt request GPIO	O I I/O	V_{DDEH4}	S	— / WKPCFG	— / WKPCFG	416 324 208

Table 2-1. MPC5553 Signal Properties (continued)

Signal and Range	P/A/G	Function[1]	Description	I/O Type	Voltage[2]	Pad Type[3]	Status During Reset[4]	Status After Reset[5]	Package
EMIOS [16:23]	P A G	EMIOS[16:23] ETPUB[0:7] GPIO[195:202]	eMIOS channel eTPU B channel (output only) GPIO	I/O O I/O	V_{DDEH4}	S	— / WKPCFG	— / WKPCFG	416 324 208
GPIO(5)									
GPIO[18] [203:204]	P A	EMIOS[14:15] GPIO[203:204] *Note: EMIOS is primary function*	EMIOS Channel (Output Only) GPIO	O I/O	V_{DDEH6}	S	— / Up	— / Up	416 324
GPIO205	P	GPIO205[19]	GPIO	I/O	V_{DDEH8}	M	— / Up	— / Up	416
GPIO [206:207]	P	GPIO[206:207][20] (can be selected as sources for the ADC trigger in the SIU_ETISR)	GPIO eQADC Trigger Input	I/O	V_{DDE3}	F	— / Up	— / Up	416 324 208
Clock Synthesizer (4)									
XTAL	P	XTAL	Crystal Oscillator Output	O	V_{DDSYN}	AE	O / —	XTAL[21] / —	416 324 208
EXTAL	P A	EXTAL [22] EXTCLK	Crystal Oscillator Input External Clock Input	I	V_{DDSYN}	AE	I / —	EXTAL[23] / —	416 324 208
CLKOUT	P	CLKOUT	System Clock Output	O	V_{DDE5}	F	CLKOUT / Enabled	CLKOUT / Enabled	416 324
ENGCLK	P	ENGCLK	Engineering Clock Output	O	V_{DDE5}	F	ENGCLK/ Enabled	ENGCLK / Enabled	416 324 208
Power / Ground (77)									
V_{RC33}	P	V_{RC33}[24]	Voltage Regulator Control Supply	I	3.3V	—	I / —	V_{RC33}	416 324 208
V_{RCVSS}	P	V_{RCVSS}	Voltage Regulator Control Ground	I	—	—	I / —	V_{RCVSS}	416 324 208
V_{RCCTL}	P	V_{RCCTL}	Voltage Regulator Control Output	O	3.3V	—	O / —	V_{RCCTL}	416 324 208
V_{DDA0}	P	V_{DDA0}[25]	Analog Power Input ADC0	I	5.0V	—	I / -	V_{DDA0}	416 324 208

Table 2-1. MPC5553 Signal Properties (continued)

Signal and Range	P/A/G	Function[1]	Description	I/O Type	Voltage[2]	Pad Type[3]	Status During Reset[4]	Status After Reset[5]	Package
V_{SSA0}	P	V_{SSA0}[25]	Analog Ground Input ADC0	I	—	—	I / -	V_{SSA0}	416 324 208
V_{DDA1}	P	V_{DDA1}[25]	Analog Power Input	I	5.0V	—	I / —	V_{DDA1}	416 324 208
V_{SSA1}	P	V_{SSA1}[25]	Analog Ground Input	I	—	—	I / —	V_{SSA1}	416 324 208
V_{DDSYN}	P	V_{DDSYN}	Clock Synthesizer Power Input	I	3.3V	—	I / —	V_{DDSYN}	416 324 208
V_{SSSYN}	P	V_{SSSYN}	Clock Synthesizer Ground Input	I	—	—	I / —	V_{SSSYN}	416 324 208
V_{FLASH}	P	V_{FLASH}	Flash Read Supply Input	I	3.3V	—	I / —	V_{FLASH}	416 324 208
V_{PP}	P	V_{PP}[26]	Flash Program/Erase Supply Input	I	5.0V	—	I / —	V_{PP}	416 324 208
V_{STBY}	P	V_{STBY}[27]	Internal SRAM Standby Power Input	I	TBD	—	I / —	V_{STBY}	416 324 208
V_{DD}	P	V_{DD}	Internal Logic Supply Input	I	1.5V	—	I / —	V_{DD}	416 324 208
V_{DDE}	P	V_{DDE}	External I/O Supply Input	I	1.8V - 3.3V	—	I / —	V_{DDE}	416 324 208
V_{DDEH}	P	V_{DDEH}[28]	External I/O Supply Input	I	3.3V - 5.0V	—	I / —	V_{DDEH}	416 324 208
V_{DD33}	P	V_{DD33}[29]	3.3V I/O Supply Input	I	3.3V	—	I / —	3.3V	416 324 208
V_{SS}	P	V_{SS}	Ground	—	—	—	I / —	V_{SS}	416 324 208

Table 2-1. MPC5553 Signal Properties (continued)

Signal and Range	P/A/G	Function[1]	Description	I/O Type	Voltage[2]	Pad Type[3]	Status During Reset[4]	Status After Reset[5]	Package
No Connect (2)									
NC[30]		NC	No Connect	—	—	—	—	—	416 324 208

1 For each pin in the table, each line in the Function column is a separate function of the pin. For all MPC5553 I/O pins the selection of primary pin function or secondary function or GPIO is done in the MPC5553 SIU except where explicitly noted.

2 The $V_{DD}E$ and VDDEH supply inputs are broken into segments. Each segment of slow I/O pins (VDDEH) may have a separate supply in the 3.3V to 5.0V range (+/- 5%). Each segment of fast I/O (VDDE) may have a separate supply in the 1.8V to 3.3V range (+/- 5%). Currently in the MPC5553 package, the V_{DDE2} and VDDE3 segments are shorted together into one segment. This segment is labelled V_{DDE2} in the ball map. See Table 2-4., "MPC5554 Power/Ground Segmentation" for a definition of the I/O pins that are powered by each segment.

3 The pad type is indicated by one of the abbreviations; F for fast, M for medium, S for slow, A for analog, AE for analog with ESD protection circuitry. Some pads may have two types, depending on which pad function is selected.

4 Terminology is O — output, I — input, up — weak pull up enabled, down — weak pull down enabled, low — output driven low, high — output driven high. A dash on the left side of the slash denotes that both the input and output buffers for the pin are off. A dash on the right side of the slash denotes that there is no weak pull up/down enabled on the pin. The signal name to the left or right of the slash indicates the pin is enabled.

5 Function after reset of GPI is general-purpose input. A dash on the left side of the slash denotes that both the input and output buffers for the pin are off. A dash on the right side of the slash denotes that there is no weak pull up/down enabled on the pin.

6 BOOTCFG0 does not function in the 208 package of the MPC5553.

7 The EBI is specified and tested at 1.8V and 3.3V.

8 ADDR[8:11] and $\overline{CS}$[0:3] pins must not be simultaneously configured to select ADDR[8:11]. Only one pin must be configured to provide the address input.

9 The function and state of these pins after execution of the BAM program is determined by the BOOTCFG[1:0] pins. See Table 16-6 for detail on the external bus interface (EBI) configuration after execution of the BAM program.

10 Although GPIO versus EBI function is specified in the SIU, when EBI function is chosen, the function must also be enabled in the EBI for these pins. The SIU and EBI configurations must match for proper operation.

11 GPIO versus EBI function for the $\overline{WE}$[0:3]_$\overline{BE}$[0:3]_GPIO[64:67] pins is specified in the SIU. When configured for EBI operation, the pin function of $\overline{WE}$[0:3] or $\overline{BE}$[0:3] is specified in the EBI_BR0 — EBI_BR3 registers for each chip select region.

12 The $\overline{BR}$ and $\overline{BG}$ functionality is not implemented on the MPC5553, it is replaced by calibration functionality. The pin name on the ball map, however, does remain $\overline{BR}$ and $\overline{BG}$. The primary functions are CAL_ADDR10 and CAL_ADDR11, respectively.

13 MCKO is only enabled if debug mode is enabled. Debug mode can be enabled before or after exiting System Reset (RSTOUT negated).

14 MDO[0] is driven high following a power-on reset until the system clock achieves lock, at which time it is then negated. There is an internal pull up on MDO[0].

15 The function of the MDO[11:4]_GPIO[75:82] pins is selected during a debug port reset by the $\overline{EVTI}$ pin. When functioning as MDO[11:4] the pad configuration specified by the SIU does not apply. See Section 2.3.3.5, "Nexus Message Data Out / GPIO (MDO[11:4]_GPIO[82:75])" for more detail on MDO[11:4] pin operation.

16 The function and state of the CAN_A and SCI_A pins after execution of the BAM program is determined by the BOOTCFG[0:1] pins. See Table 16-8 for detail on the CAN and SCI pin configuration after execution of the BAM program.

17 All analog input channels are connected to both ADC blocks. The supply designation for this pin(s) specifies only the ESD rail used.

18 Because other balls already are named EMIOS[14:15], the balls for these signals are named GPIO[203:204].

19 The GPIO[205] pin is a protect for pin for configuring an external boot for a double data rate memory.

20 The GPIO[206:207] pins are protect for pins for double data rate memory data strobes.

21 The function after reset of the XTAL pin is determined by the value of the signal on the PLLCFG[1] pin. When bypass mode is chosen XTAL has no function and should be grounded.

[22] When the FMPLL is configured for external reference mode, the V_{DDE5} supply affects the acceptable signal levels for the external reference. See Section 11.1.4.2, “External Reference Mode.”

[23] The function after reset of the EXTAL_EXTCLK pin is determined by the value of the signal on the PLLCFG[0:1] pins. If the EXTCLK function is chosen, the valid operating voltage for the pin is 1.6V to 3.6V. If the EXTAL function is chosen, the valid operating voltage is 3.3V. Refer to Table 11-1.

[24] V_{RC33} is the 3.3V input for the voltage regulator control.

[25] The $V_{DDA}n$ and $V_{SSA}n$ supply inputs are split into separate traces in the package substrate. Each trace is bonded to a separate pad location, which provides isolation between the analog and digital sections within each ADC. The digital power/ground use pad_vddint/pad_vssint pads respectively. The analog power/ground use spcr_filr_32_vdde/spcr_filr_32_vsse pads respectively.

[26] May be tied to 5.0V for both read operation and program/erase.

[27] The V_{STBY} pin should be tied to $V_{SSA}0$ if the battery backed internal SRAM is not used.

[28] The V_{DDEH9} segment may be powered from 3.0V to 5.0V for mux address or SSI functions, but must meet the $V_{DDA}1$ specifications of 4.5V to 5.25V for analog input function.

[29] All pins with pad type pad_fc will be driven to the high state if their VDDE segment is powered before V_{DD33}.

[30] The pins are reserved for the clock and inverted clock outputs for DDR memory interface. In the MPC5553/MPC5554 416-pin package, the two NC pins are isolated (not shorted together in the package substrate)

2.2.2 MPC5554 Signals Summary

Table 2-2 gives a summary of the MPC5554 external signals and properties.

Table 2-2. MPC5554 Signal Properties

Signal and Range[1]	Pin	P/A/G[2]	Function[3]	Description	I/O Type	Pad Type[4]	Status During Reset[5]	Status After Reset[6]
Reset / Configuration (8)								
$\overline{\text{RESET}}$	W26	P	$\overline{\text{RESET}}$	External reset input	I	S	$\overline{\text{RESET}}$ / Up	$\overline{\text{RESET}}$ / Up
$\overline{\text{RSTOUT}}$	V25	P	$\overline{\text{RSTOUT}}$	External reset output	O	S	$\overline{\text{RSTOUT}}$ / Low	$\overline{\text{RSTOUT}}$ / High
PLLCFG0	AB25	P A G	PLLCFG0 IRQ4 GPIO208	FMPLL mode selection External interrupt request GPIO	I I I/O	M	PLLCFG / Up	— / Up
PLLCFG1	AA24	P A A2 G	PLLCFG1 IRQ5 SOUTD GPIO209	FMPLL mode selection External interrupt request DSPI D data out GPIO	I I O I/O	M	PLLCFG / Up	— / Up
$\overline{\text{RSTCFG}}$	V26	P A	$\overline{\text{RSTCFG}}$ GPIO210	Reset configuration input GPIO	I I/O	S	$\overline{\text{RSTCFG}}$ / Up	— / Up
BOOTCFG[0:1]	AA25, Y24	P A G	BOOTCFG[0:1] IRQ[2:3] GPIO[211:212]	Boot configuration input External interrupt request GPIO	I I I/O	S	BOOTCFG / Down	— / Down
WKPCFG	Y23	P G	WKPCFG GPIO213	Weak pull configuration input GPIO	I I/O	S	WKPCFG / Up	— / Up
External Bus Interface (EBI)[7] (75)								
$\overline{\text{CS}}$[0:3]	P4, P3, P2, P1	P A G	$\overline{\text{CS}}$[0:3] ADDR[8:11] GPIO[0:3]	External chip selects External address bus GPIO	O I/O I/O	F	— / Up	— / Up[8]
ADDR[8:31]	V4, W3, W4, Y3, AA4, AA3, AB4, AB3, U1, V2, V1, W2, W1, Y2, Y1, AA2, AA1, AB2, AC1, AC2, AD1, AE1, AD2, AC3	P G	ADDR[8:31] GPIO[4:27]	External address bus[9] GPIO	I/O I/O	F	— / Up	— / Up[8]

Table 2-2. MPC5554 Signal Properties (continued)

Signal and Range[1]	Pin	P/A/G[2]	Function[3]	Description	I/O Type	Pad Type[4]	Status During Reset[5]	Status After Reset[6]
DATA[0:31]	AE8, AF9, AE9, AF10, AE10, AF12, AE11, AF13, AC11, AD11, AC12, AD12, AC14, AD13, AC15, AD14, AF3, AE4, AF4, AE5, AF6, AE6, AF7, AE7, AD5, AD6, AC6, AD7, AC7, AD8, AC9, AC10	P G	DATA[0:31] GPIO[28:59]	External data bus[9] GPIO	I/O I/O	F	— / Up	— / Up[8]
TSIZ[0:1]	T2, U2	P G	TSIZ[0:1] GPIO[60:61]	External transfer size[9] GPIO	I/O I/O	F	— / Up	— / Up[8]
RD_$\overline{WR}$	T3	P G	RD_$\overline{WR}$ GPIO62	External read/write GPIO	I/O I/O	F	— / Up	— / Up[8]
$\overline{BDIP}$	N1	P G	$\overline{BDIP}$ GPIO63	External burst data in progress GPIO	O I/O	F	— / Up	— / Up[8]
$\overline{WE}$[0:3]	R4, R3, R2, R1	P A G	$\overline{WE}$[0:3] $\overline{BE}$[0:3] GPIO[64:67]	External write enable External byte enable[10] GPIO	O O I/O	F	— / Up	— / Up[8]
$\overline{OE}$	AE12	P G	$\overline{OE}$ GPIO68	External output enable GPIO	O I/O	F	— / Up	— / Up[8]
$\overline{TS}$	V3	P G	$\overline{TS}$ GPIO69	External transfer start GPIO	I/O I/O	F	— / Up	— / Up[8]
$\overline{TA}$	U3	P G	$\overline{TA}$ GPIO70	External transfer acknowledge GPIO	I/O I/O	F	— / Up	— / Up[8]
$\overline{TEA}$	N2	P G	$\overline{TEA}$ GPIO71	External transfer error acknowledge GPIO	I/O I/O	F	— / Up	— / Up[8]
$\overline{BR}$	AE13	P G	$\overline{BR}$ GPIO72	External bus request[9] GPIO	I/O I/O	F	— / Up	— / Up[8]
$\overline{BG}$	AE14	P G	$\overline{BG}$ GPIO73	External bus grant[9] GPIO	I/O I/O	F	— / Up	— / Up[8]
$\overline{BB}$	AF14	P G	$\overline{BB}$ GPIO74	External bus busy[9] GPIO	I/O I/O	F	— / Up	— / Up[8]
NEXUS (18)								
$\overline{EVTI}$	F25	P	$\overline{EVTI}$	Nexus event in	I	F	I / Up	EVTI / Up

Table 2-2. MPC5554 Signal Properties (continued)

Signal and Range[1]	Pin	P/A/G[2]	Function[3]	Description	I/O Type	Pad Type[4]	Status During Reset[5]	Status After Reset[6]
$\overline{\text{EVTO}}$	F26	P	$\overline{\text{EVTO}}$	Nexus event out	O	F	O / Low	EVTO / High
MCKO	G24	P	MCKO	Nexus message clock out	O	F	O / Low	MCKO / Enabled[11]
MDO[3:0][12]	C22, D21, C23, B24	P	MDO[3:0]	Nexus message data out	O	F	O / Low	MDO / Low
MDO[11:4]	A22, B21, C20, A23, B22, C21, D20, B23	P G	MDO[11:4] GPIO[75:82][13]	Nexus message data out GPIO	O I/O	F	O / Low	— / Down
$\overline{\text{MSEO}}$[1:0]	G23, F23	P	$\overline{\text{MSEO}}$[1:0]	Nexus message start/end out	O	F	O / High	MSEO / High
$\overline{\text{RDY}}$	H23	P	$\overline{\text{RDY}}$	Nexus ready output	O	F	O / High	$\overline{\text{RDY}}$ / High
JTAG / TEST(6)								
TCK	D25	P	TCK	JTAG test clock input	I	F	TCK / Down	TCK / Down
TDI	D26	P	TDI	JTAG test data input	I	F	TDI / Up	TDI / Up
TDO	E25	P	TDO	JTAG test data output	O	F	TDO / Up[14]	TDO / Up
TMS	E24	P	TMS	JTAG test mode select input	I	F	TMS / Up	TMS / Up
JCOMP	F24	P	JCOMP	JTAG TAP controller enable	I	F	JCOMP / Down	JCOMP / Down
TEST	E26	P	TEST	Test mode select	I	F	$\overline{\text{TEST}}$ / Up	$\overline{\text{TEST}}$ / Up
FlexCAN (6)								
CNTXA	AD21	P G	CNTXA GPIO83	FlexCAN A transmit GPIO	O I/O	S	— / Up	— / Up[15]
CNRXA	AE22	P G	CNRXA GPIO84	FlexCAN A receive GPIO	I I/O	S	— / Up	— / Up[15]
CNTXB	AF22	P A G	CNTXB PCSC3 GPIO85	FlexCAN B transmit DSPI C peripheral chip select 3 GPIO	O O I/O	M	— / Up	— / Up
CNRXB	AF23	P A G	CNRXB PCSC4 GPIO86	FlexCAN B receive DSPI C peripheral chip select 4 GPIO	I O I/O	M	— / Up	— / Up
CNTXC	V23	P A G	CNTXC PCSD3 GPIO87	FlexCAN C transmit DSPI D peripheral chip select 3 GPIO	O O I/O	M	— / Up	— / Up

Table 2-2. MPC5554 Signal Properties (continued)

Signal and Range[1]	Pin	P/A/G[2]	Function[3]	Description	I/O Type	Pad Type[4]	Status During Reset[5]	Status After Reset[6]
CNRXC	W24	P A G	CNRXC PCSD4 GPIO88	FlexCAN C receive DSPI D peripheral chip select 4 GPIO	I O I/O	M	— / Up	— / Up
eSCI (4)								
TXDA	U24	P G	TXDA GPIO89	eSCI A transmit GPIO	O I/O	S	— / Up	— / Up[15]
RXDA	V24	P G	RXDA GPIO90	eSCI A receive GPIO	I I/O	S	— / —	— / —[15]
TXDB	W25	P A G	TXDB PCSD1 GPIO91	eSCI B transmit DSPI D peripheral chip select 1 GPIO	O O I/O	M	— / Up	— / Up
RXDB	W23	P A G	RXDB PCSD5 GPIO92	eSCI B receive DSPI D peripheral chip select 5 GPIO	I O I/O	M	— / Up	— / Up
DSPI(18)								
SCKA	R26	P A G	SCKA PCSC1 GPIO93	DSPI A clock DSPI C peripheral chip select 1 GPIO	I/O O I/O	M	— / Up	— / Up
SINA	R25	P A G	SINA PCSC2 GPIO94	DSPI A data input DSPI C peripheral chip select 2 GPIO	I O I/O	M	— / Up	— / Up
SOUTA	R24	P A G	SOUTA PCSC5 GPIO95	DSPI A data output DSPI C peripheral chip select 5 GPIO	O O I/O	M	— / Up	— / Up
PCSA0	T24	P A G	PCSA0 PCSD2 GPIO96	DSPI A peripheral chip select 0 DSPI D peripheral chip select 2 GPIO	I/O O I/O	M	— / Up	— / Up
PCSA1	T23	P A G	PCSA1 PCSB2 GPIO97	DSPI A peripheral chip select 1 DSPI B peripheral chip select 2 GPIO	O O I/O	M	— / Up	— / Up
PCSA2	T25	P A G	PCSA2 SCKD GPIO98	DSPI A peripheral chip select 2 DSPI D clock GPIO	O I/O I/O	M	— / Up	— / Up
PCSA3	P23	P A G	PCSA3 SIND GPIO99	DSPI A peripheral chip select DSPI D data input GPIO	O I I/O	M	— / Up	— / Up
PCSA4	U23	P A G	PCSA4 SOUTD GPIO100	DSPI A peripheral chip select DSPI D data output GPIO	O O I/O	M	— / Up	— / Up

Table 2-2. MPC5554 Signal Properties (continued)

Signal and Range[1]	Pin	P/A/G[2]	Function[3]	Description	I/O Type	Pad Type[4]	Status During Reset[5]	Status After Reset[6]
PCSA5	U25	P A G	PCSA5 PCSB3 GPIO101	DSPI A peripheral chip select 5 DSPI B peripheral chip select 3 GPIO	O O I/O	M	— / Up	— / Up
SCKB	P25	P A G	SCKB PCSC1 GPIO102	DSPI B clock DSPI C peripheral chip select GPIO	I/O O I/O	M	— / Up	— / Up
SINB	M26	P A G	SINB PCSC2 GPIO103	DSPI B data input DSPI C peripheral chip select GPIO	I O I/O	M	— / Up	— / Up
SOUTB	N23	P A G	SOUTB PCSC5 GPIO104	DSPI B data output DSPI C peripheral chip select GPIO	O O I/O	M	— / Up	— / Up
PCSB0	N25	P A G	PCSB0 PCSD2 GPIO105	DSPI B peripheral chip select DSPI D peripheral chip select GPIO	I/O O I/O	M	— / Up	— / Up
PCSB1	N26	P A G	PCSB1 PCSD0 GPIO106	DSPI B peripheral chip select DSPI D peripheral chip select GPIO	O I/O I/O	M	— / Up	— / Up
PCSB2	P26	P A G	PCSB2 SOUTC GPIO107	DSPI B peripheral chip select DSPI C data output GPIO	O O I/O	M	— / Up	— / Up
PCSB3	N24	P A G	PCSB3 SINC GPIO108	DSPI B peripheral chip select DSPI C data input GPIO	O I I/O	M	— / Up	— / Up
PCSB4	P24	P A G	PCSB4 SCKC GPIO109	DSPI B peripheral chip select DSPI C clock GPIO	O I/O I/O	M	— / Up	— / Up
PCSB5	R23	P A G	PCSB5 PCSC0 GPIO110	DSPI B peripheral chip select DSPI C peripheral chip select GPIO	O I/O I/O	M	— / Up	— / Up
eQADC(45)								
AN0	B7	P A	AN0 DAN0+	Analog input Differential analog input	I	AE	I / —	AN0 / —
AN1	A7	P A	AN1 DAN0–	Analog input Differential analog input	I	AE	I / —	AN1 / —
AN2	D9	P A	AN2 DAN1+	Analog input Differential analog input	I	AE	I / —	AN2 / —
AN3	C8	P A	AN3 DAN1–	Analog input Differential analog input	I	AE	I / —	AN3 / —
AN4	B8	P A	AN4 DAN2+	Analog input Differential analog input	I	AE	I / —	AN4 / —

Table 2-2. MPC5554 Signal Properties (continued)

Signal and Range[1]	Pin	P/A/G[2]	Function[3]	Description	I/O Type	Pad Type[4]	Status During Reset[5]	Status After Reset[6]
AN5	A8	P A	AN5 DAN2–	Analog input Differential analog input	I	AE	I / —	AN5 / —
AN6	D10	P A	AN6 DAN3+	Analog input Differential analog input	I	AE	I / —	AN6 / —
AN7	C9	P A	AN7 DAN3–	Analog input Differential analog input	I	AE	I / —	AN7 / —
AN8	C4	P A	AN8 ANW	Analog input Mux input	I I	AE	I / —	AN8 / —
AN9	D6	P A	AN9 ANX	Analog input Mux input	I I	AE	I / —	AN9 / —
AN10	D7	P A	AN10 ANY	Analog input Mux input	I I	AE	I / —	AN10 / —
AN11	A4	P A	AN11 ANZ	Analog input Mux input	I I	AE	I / —	AN11 / —
AN12	D15	P A A	AN12 MA0 $\overline{\text{SDS}}$	Analog input Mux address eQADC serial data select	I O O	A, M	I / —	AN12 / —
AN13	C15	P A A	AN13 MA1 SDO	Analog input Mux address eQADC serial data out	I O O	A, M	I / —	AN13 / —
AN14	B15	P A A	AN14 MA2 SDI	Analog input Mux address eQADC serial data in	I O I	A, M	I / —	AN14 / —
AN15	A15	P A	AN15 FCK	Analog input eQADC free running clock	I O	A, M	I / —	AN15 / —
AN[16:39]	A6, C5, D8, B5, B6, C7, B10, A10, D11, C11, B11, A11, A12, D12, C12, B12, B13, C13, D13, A13, B3, A3, D5, B4	P	AN[16:39]	Analog input	I	AE	I / —	AN[x] / —
ETRIG[0:1]	B16, A16	P G	ETRIG[0:1] GPIO[111:112]	eQADC trigger input GPIO	I I/O	S	— / Up	— / Up
VRH	A9	P	VRH	Voltage reference high	I	—	— / —	VRH
VRL	C10	P	VRL	Voltage reference low	I	—	— / —	VRL
REFBYPC	B9	P	REFBYPC	Reference Bypass Capacitor Input	I	AE	— / —	REFBYPC

Table 2-2. MPC5554 Signal Properties (continued)

Signal and Range[1]	Pin	P/A/G[2]	Function[3]	Description	I/O Type	Pad Type[4]	Status During Reset[5]	Status After Reset[6]
eTPU(66)								
TCRCLKA	N4	P A G	TCRCLKA IRQ7 GPIO113	eTPU A TCR clock External interrupt request GPIO	I I I/O	S	— / Up	— / Up
ETPUA[0:11]	N3, M4, M3, M2, M1, L4, L3, L2, L1, K4, K3, K2	P A G	ETPUA[0:11] ETPUA[12:23] GPIO[114:125]	eTPU A channel eTPU A channel (output only) GPIO	I/O O I/O	S	— / WKPCFG	— / WKPCFG
ETPUA12	K1	P A G	ETPUA12 PCSB1 GPIO126	eTPU A channel DSPI B peripheral chip select 1 GPIO	I/O O I/O	M	— / WKPCFG	— / WKPCFG
ETPUA13	J4	P A G	ETPUA13 PCSB3 GPIO127	eTPU A channel DSPI B peripheral chip select 3 GPIO	I/O O I/O	M	— / WKPCFG	— / WKPCFG
ETPUA14	J3	P A G	ETPUA14 PCSB4 GPIO128	eTPU A channel DSPI B peripheral chip select 4 GPIO	I/O O I/O	M	— / WKPCFG	— / WKPCFG
ETPUA15	J2	P A G	ETPUA15 PCSB5 GPIO129	eTPU A channel DSPI B peripheral chip select 5 GPIO	I/O O I/O	M	— / WKPCFG	— / WKPCFG
ETPUA16	J1	P A G	ETPUA16 PCSD1 GPIO130	eTPU A channel DSPI D peripheral chip select 1 GPIO	I/O O I/O	M	— / WKPCFG	— / WKPCFG
ETPUA17	H4	P A G	ETPUA17 PCSD2 GPIO131	eTPU A channel DSPI D peripheral chip select 2 GPIO	I/O O I/O	M	— / WKPCFG	— / WKPCFG
ETPUA18	H3	P A G	ETPUA18 PCSD3 GPIO132	eTPU A channel DSPI D peripheral chip select 3 GPIO	I/O O I/O	M	— / WKPCFG	— / WKPCFG
ETPUA19	H2	P A G	ETPUA19 PCSD4 GPIO133	eTPU A channel DSPI D peripheral chip select 4 GPIO	I/O O I/O	M	— / WKPCFG	— / WKPCFG
ETPUA[20:23]	H1, G4, G2, G1	P A G	ETPUA[20:23] IRQ[8:11] GPIO[134:137]	eTPU A channel External interrupt request GPIO	I/O I I/O	M	— / WKPCFG	— / WKPCFG
ETPUA[24:26]	F1, G3, F3	P A G	ETPUA[24:26] IRQ[12:14] GPIO[138:140]	eTPU A channel (output only) External interrupt request GPIO	O I I/O	S	— / WKPCFG	— / WKPCFG
ETPUA27	F2	P A G	ETPUA27 IRQ15 GPIO141	eTPU A channel (output only) External interrupt request GPIO	O I I/O	S	— / WKPCFG	— / WKPCFG

Table 2-2. MPC5554 Signal Properties (continued)

Signal and Range[1]	Pin	P/A/G[2]	Function[3]	Description	I/O Type	Pad Type[4]	Status During Reset[5]	Status After Reset[6]
ETPUA28	E1	P A G	ETPUA28 PCSC1 GPIO142	eTPU A Channel (Output Only) DSPI C peripheral chip select 1 GPIO	O O I/O	M	— / WKPCFG	— / WKPCFG
ETPUA29	E2	P A G	ETPUA29 PCSC2 GPIO143	eTPU A Channel (Output Only) DSPI C peripheral chip select 2 GPIO	O O I/O	M	— / WKPCFG	— / WKPCFG
ETPUA30	D1	P A G	ETPUA30 PCSC3 GPIO144	eTPU A Channel DSPI C peripheral chip select 3 GPIO	I/O O I/O	M	— / WKPCFG	— / WKPCFG
ETPUA31	D2	P A G	ETPUA31 PCSC4 GPIO145	eTPU A Channel DSPI C peripheral chip select 4 GPIO	I/O O I/O	M	— / WKPCFG	— / WKPCFG
TCRCLKB	M23	P A G	TCRCLKB IRQ6 GPIO146	eTPU B TCR clock External Interrupt Request GPIO	I I I/O	S	— / Up	— / Up
ETPUB[0:15]	M25, M24, L26, L25, L24, K26, L23, K25, K24, J26, K23, J25, J24, H26, H25, G26	P A G	ETPUB[0:15] ETPUB[16:31] GPIO[147:162]	eTPU B channel eTPU B channel (output only) GPIO	I/O O I/O	S	— / WKPCFG	— / WKPCFG
ETPUB16	D16	P A G	ETPUB16 PCSA1 GPIO163	eTPU B channel DSPI A peripheral chip select 1 GPIO	I/O O I/O	S	— / WKPCFG	— / WKPCFG
ETPUB17	D17	P A G	ETUB17 PCSA2 GPIO164	eTPU B channel DSPI A peripheral chip select 2 GPIO	I/O O I/O	S	— / WKPCFG	— / WKPCFG
ETPUB18	A17	P A G	ETUB18 PCSA3 GPIO165	eTPU B channel DSPI Aperipheral chip select 3 GPIO	I/O O I/O	S	— / WKPCFG	— / WKPCFG
ETPUB19	C16	P A G	ETUB19 PCSA4 GPIO166	eTPU B channel DSPI A peripheral chip select 4 GPIO	I/O O I/O	S	— / WKPCFG	— / WKPCFG
ETPUB[20:31]	A18, B17, C17, D18, A19, B18, C18, A20, B19, D19, C19, B20	P G	ETPUB[20:31] GPIO[167:178]	eTPU B channel GPIO	I/O I/O	S	— / WKPCFG	— / WKPCFG

Table 2-2. MPC5554 Signal Properties (continued)

Signal and Range[1]	Pin	P/A/G[2]	Function[3]	Description	I/O Type	Pad Type[4]	Status During Reset[5]	Status After Reset[6]
eMIOS(24)								
EMIOS[0:9]	AF15, AE15, AC16, AD15, AF16, AE16, AD16, AF17, AC17, AE17	P A G	EMIOS[0:9] ETPUA[0:9] GPIO[179:188]	eMIOS channel eTPU A channel (output only) GPIO	I/O O I/O	S	— / WKPCFG	— / WKPCFG
EMIOS10	AD17	P G	EMIOS10 GPIO189	eMIOS channel GPIO	I/O I/O	S	— / WKPCFG	— / WKPCFG
EMIOS11	AF18	P G	EMIOS11 GPIO190	eMIOS channel GPIO	I/O I/O	S	— / WKPCFG	— / WKPCFG
EMIOS12	AC18	P A G	EMIOS12 SOUTC GPIO191	eMIOS channel (output only) DSPI C data output GPIO	O O I/O	M	— / WKPCFG	— / WKPCFG
EMIOS13	AE18	P A G	EMIOS13 SOUTD GPIO192	eMIOS channel (output only) DSPI D data output GPIO	O O I/O	M	— / WKPCFG	— / WKPCFG
EMIOS[14:15]	AF19, AD18	P A G	EMIOS[14:15] IRQ[0:1] GPIO[193:194]	eMIOS channel (output only) External interrupt request GPIO	O I I/O	S	— / WKPCFG	— / WKPCFG
EMIOS[16:23]	AE19, AD19, AF20, AE20, AF21, AC19, AD20, AE21	P A G	EMIOS[16:23] ETPUB[0:7] GPIO[195:202]	eMIOS channel eTPU B channel (output only) GPIO	I/O O I/O	S	— / WKPCFG	— / WKPCFG
GPIO(5)								
GPIO[203:204][16]	H24, G25	P A	EMIOS[14:15] GPIO[203:204] *Note: EMIOS is primary function*	eMIOS channel (output only) GPIO	O I/O	S	— / Up	— / Up
GPIO205	A21	P	GPIO205[17]	GPIO	I/O	M	— / Up	— / Up
GPIO[206:207]	AF8, AD10	P	GPIO[206:207][18] (can be selected as sources for the ADC trigger in the SIU_ETISR)	GPIO	I/O	F	— / Up	— / Up
Clock Synthesizer (6)								
XTAL	AB26	P	XTAL	Crystal oscillator output	O	AE	O / —	XTAL[19] / —
EXTAL	AA26	P A	EXTAL EXTCLK[20]	Crystal oscillator input External clock input	I	AE	I / —	EXTAL[21] / —

Table 2-2. MPC5554 Signal Properties (continued)

Signal and Range[1]	Pin	P/A/G[2]	Function[3]	Description	I/O Type	Pad Type[4]	Status During Reset[5]	Status After Reset[6]
CLKOUT	AE24	P	CLKOUT	System clock output	O	F	CLKOUT / Enabled	CLKOUT / Enabled
ENGCLK	AF25	P	ENGCLK	Engineering clock output	O	F	O / Low	ENGCLK / Low
Power / Ground (76)								
V_{RC33}	AC25	P	V_{RC33}[22]	Voltage regulator control supply	I	—	I / —	V_{RC33}
V_{RCVSS}	Y25	P	V_{RCVSS}	Voltage regulator control ground	I	—	I / —	V_{RCVSS}
V_{RCCTL}	AB24	P	V_{RCCTL}	Voltage regulator control output	O	—	O / —	V_{RCCTL}
V_{DDA0}	C14	P	V_{DDA0}[23]	Analog power input	I	—	I / —	V_{DDA0}
V_{SSA0}	A14, B14	P	V_{SSA0}[23]	Analog ground input		—	I / —	V_{SSA0}
V_{DDA1}	A5	P	V_{DDA1}[23]	Analog power input	I	—	I / —	V_{DDA1}
V_{SSA1}	C6	P	V_{SSA1}[23]	Analog ground input	I	—	I / —	V_{SSA1}
V_{DDSYN}	AC26	P	V_{DDSYN}	Clock synthesizer power input	I	—	I / —	V_{DDSYN}
V_{SSSYN}	Y26	P	V_{SSSYN}	Clock synthesizer ground input	I	—	I / —	V_{SSSYN}
V_{FLASH}	U26	P	V_{FLASH}	Flash read supply input	I	—	I / —	V_{FLASH}
V_{PP}	T26	P	V_{PP}[24]	Flash program/erase supply input	I	—	I / —	V_{PP}
V_{STBY}	A2	P	V_{STBY}[25]	SRAM standby power input	I	—	I / —	V_{STBY}
V_{DD}	A24, B1, C2, C26, D3, E4, AB23, AC5, AC24, AD4, AD25, AE3, AE26, AF2	P	V_{DD}	Internal logic supply input	I	—	I / —	V_{DD}
V_{DDE2}	T1, T4, Y4, AB1, AF5, AC8, AF11, AC13, M10, N10, P10, R10, T10, M11, N11, P11, R11, U11, T12, U12, T13, U13, T14, U14, T15, U15	P	v_{DDE}	External I/O supply input	I	—	I / —	v_{DDE}
V_{DDE5}	AC21, AD22, AE23, AF24	P	v_{DDE}	External I/O supply input	I	—	I / —	v_{DDE}

Table 2-2. MPC5554 Signal Properties (continued)

Signal and Range[1]	Pin	P/A/G[2]	Function[3]	Description	I/O Type	Pad Type[4]	Status During Reset[5]	Status After Reset[6]
V_{DDE7}	B26, C25, D24, E23, K14, K15, K16, K17, L17, M17, N17	P	v_{DDE}	External I/O supply input	I	—	I / —	v_{DDE}
V_{DDEH1}	E3, F4	P	v_{DDEH}	External I/O high supply input	I	—	I / —	v_{DDEH}
V_{DDEH4}	AC20	P	v_{DDEH}	External I/O high supply input	I	—	I / —	v_{DDEH}
V_{DDEH6}	AA23, J23	P	v_{DDEH}	External I/O high supply input	I	—	I / —	v_{DDEH}
V_{DDEH8}	D22	P	v_{DDEH}	External I/O high supply input	I	—	I / —	v_{DDEH}
V_{DDEH9}	D14	P	V_{DDEH}[26]	External I/O high supply input	I	—	I / —	V_{DDEH}
V_{DD33}	C1, U4, AD9, A25, AD26	P	V_{DD33}[27]	3.3V I/O supply input	I	—	I / —	V_{DD33}
V_{SS}	A1, AF1, B2, AE2, C3, AD3, D4, AC4, D23, C24, B25, A26, AD24, AE 25, AF26, K10, K11, K12, K13, L10, L11, L12, L13, L14, L15, L16, M12, M13, M14, M15, M16, N12, N13, N14, N15, N16, P12, P13, P14, P15, P16, P17, R12, R13, R14, R15, R16, R17, T11, T16, T17, U10, U16, U17, AC23	P	V_{SS}	Ground		—	I / —	V_{SS}
No Connect (2)								
NC	AC22,AD23		NC[28]	No Connect	—	—	—	—

[1] This is the pin name that appears on the PBGA pinout.

[2] Primary, alternate, or GPIO function. Note that some pins may have a third function rather than, or in addtion to GPIO.

[3] For each pin in the table, each line in the function column is a separate function of the pin. For all MPC5554 I/O pins the selection of primary, secondary or tertiary function is done in the MPC5554 SIU except where explicitly noted.

[4] The pad type is indicated by one of the abbreviations; F for fast, M for medium, S for slow, A for analog, AE for analog with ESD protection circuitry. Some pads may have two types, depending on which pad function is selected.

[5] Terminology is O — output, I — input, up — weak pull up enabled, down — weak pull down enabled, low — output driven low, High — output driven high. A dash on the left side of the slash denotes that both the input and output buffers for the pin are off. A dash on the right side of the slash denotes that there is no weak pull up/down enabled on the pin. The signal name to the left or right of the slash indicates the pin is enabled.

[6] Function after reset of GPI is general-purpose input. A dash on the left side of the slash denotes that both the input and output buffers for the pin are off. A dash on the right side of the slash denotes that there is no weak pull up/down enabled on the pin.

[7] The EBI is specified and tested at 1.8V and 3.3V.

[8] The function and state of this pin after execution of the BAM program is determined by the BOOTCFG[0:1] pins. See Table 16-6 for detail on the external bus interface (EBI) configuration after execution of the BAM program.

[9] Although GPIO versus EBI function is specified in the SIU, when EBI function is chosen, the function must also be enabled in the EBI for these pins. The SIU and EBI configurations must match for proper operation.

[10] GPIO versus EBI function for the $\overline{\text{WE}}$[0:3]_$\overline{\text{BE}}$[0:3]_GPIO[64:67] pins is specified in the SIU. When configured for EBI operation, the pin function of $\overline{\text{WE}}$[0:3] or $\overline{\text{BE}}$[0:3] is specified in the EBI_BR0–EBI_BR3 registers for each chip select region.

[11] MCKO is only enabled if debug mode is enabled. Debug mode can be enabled before or after exiting System Reset (RSTOUT negated).

[12] MDO[0] is driven high following a power-on reset until the system clock achieves lock, at which time it is then negated. There is an internal pull up on MDO[0].

[13] The function of the MDO[11:4]_GPIO[75:82] pins is selected during a debug port reset by the $\overline{\text{EVTI}}$ pin. When functioning as MDO[11:4] the pad configuration specified by the SIU does not apply.

[14] The pull-up on TDO is only functional when not in JTAG mode, that is with JCOMP negated.

[15] The function and state of the FlexCAN_A and eSCI_A pins after execution of the BAM program is determined by the BOOTCFG[0:1] pins.

[16] Because other balls already are named EMIOS[14:15], the balls for these signals are named GPIO[203:204].

[17] The GPIO205 pin is a protect for pin for configuring an external boot for a double data rate memory.

[18] The GPIO[207:206] pins are protect for pins for double data rate memory data strobes.

[19] The function after reset of the XTAL pin is determined by the value of the signal on the PLLCFG[1] pin. When bypass mode is chosen XTAL has no function and should be grounded.

[20] When the FMPLL is configured for external reference mode, the V_{DDE5} supply affects the acceptable signal levels for the external reference. See Section 11.1.4.2, “External Reference Mode.”

[21] The function after reset of the EXTAL_EXTCLK pin is determined by the value of the signal on the PLLCFG[0:1] pins. If the EXTCLK function is chosen, the valid operating voltage for the pin is 1.6V to 3.6V. If the EXTAL function is chosen, the valid operating voltage is 3.3V. Refer to Table 11-1.

[22] V_{RC33} is the 3.3V input for the voltage regulator control.

[23] The $V_{DDA}n$ and $V_{SSA}n$ supply inputs are split into separate traces in the package substrate. Each trace is bonded to a separate pad location, which provides isolation between the analog and digital sections within each ADC. The digital power/ground use pad_vddint/pad_vssint pads respectively. The analog power/ground use spcr_filr_32_vdde/spcr_filr_32_vsse pads respectively.

[24] May be tied to 5.0V for both read operation and program/erase.

[25] The V_{STBY} pin should be tied to V_{SS} if the battery backed SRAM is not used.

[26] The V_{DDEH9} segment may be powered from 3.0V to 5.0V for mux address or SSI functions, but must meet the $V_{DDA}1$ specifications of 4.5V to 5.25V for analog input function.

[27] All pins will be driven to the high state if their V_{DDE} segment is powered before the V_{DD33} supply.

[28] The pins are reserved for the clock and inverted clock outputs for DDR memory interface. In the MPC5553/MPC5554 416-pin package, the two NC pins are isolated (not shorted together in the package substrate)

2.3 Detailed Signal Description

Below are detailed descriptions of the signals that occur on both the MPC5553 and the MPC5554. Some signals are implemented only on one device or the other; these signals are marked MPC5553 Only or MPC5554 Only. Signals not so marked function on both devices.

2.3.1 Reset / Configuration

2.3.1.1 External Reset Input (RESET)

The RESET input is asserted by an external device to reset the all modules of the MPC5553/MPC5554 MCU. The RESET pin should be asserted during a power-on reset. Refer to Section 4.2.1, "Reset Input (RESET)."

2.3.1.2 External Reset Output (RSTOUT)

The RSTOUT output is a push/pull output that is asserted during an internal MPC5553/MPC5554 reset. The pin may also be asserted by software without causing an internal reset of the MPC5553/MPC5554 MCU. Refer to Section 4.2.2, "Reset Output (RSTOUT)."

NOTE

During a power on reset, RSTOUT is tri-stated.

2.3.1.3 FMPLL Mode Selection / External Interrupt Request / GPIO (PLLCFG0_IRQ4_GPIO208)

PLLCFG0_IRQ4_GPIO208 are sampled on the negation of the RESET input pin, if the RSTCFG pin is asserted at that time. The values are used to configure the FMPLL mode of operation. The alternate function is external interrupt request input.

2.3.1.4 FMPLL Mode Selection / External Interrupt Request / DSPI / GPIO (PLLCFG1_IRQ5_SOUTD_GPIO209)

PLLCFG1_IRQ5_SOUTD_GPIO209 are sampled on the negation of the RESET input pin, if the RSTCFG pin is asserted at that time. The values are used to configure the FMPLL mode of operation. The alternate functions are external interrupt request input, and data output for the DSPI module D.

2.3.1.5 Reset Configuration Input / GPIO (RSTCFG_GPIO210)

The RSTCFG input is used to enable the BOOTCFG[0:1] and PLLCFG[0:1] pins during reset. If RSTCFG is negated during reset, the BOOTCFG and PLLCFG pins are not sampled at the negation of RSTOUT. In that case, the default values for BOOTCFG and PLLCFG are used. If RSTCFG is asserted during reset, the values on the BOOTCFG and PLLCFG pins are sampled and configure the boot and FMPLL modes.

2.3.1.6 Reset Configuration / External Interrupt Request / GPIO (BOOTCFG[0:1]_IRQ[2:3]_GPIO[211:212])

BOOTCFG[0:1]_IRQ[2:3]_GPIO[211:212] are sampled on the negation of the RSTOUT pin, if the RSTCFG pin is asserted at that time. The values are used by the BAM program to determine the boot configuration of the MPC5553/MPC5554. The alternate function is external interrupt request inputs. Note that in the 208 package of the MPC5553, BOOTCFG0 does not function.

2.3.1.7 Weak Pull Configuration / GPIO (WKPCFG_GPIO213)

WKPCFG_GPIO213 determines whether specified eTPU and EMIOS pins are connected to a weak pull up or weak pull down during and immediately after reset.

2.3.2 External Bus Interface (EBI)

2.3.2.1 External Chip Selects / External Address / GPIO (CS[0:3]_ADDR[8:11]_GPIO[0:3])

CS[0:3]_ADDR[8:11]_GPIO[0:3] are the external bus interface (EBI) chip select output signals. They can be individually configured as chip selects or GPIO. CS[1:3]_ADDR[9:11]_GPIO[1:3] are not pinned out in the 208 PBGA of the MPC5553.

2.3.2.2 External Address / Calibration Address / GPIO (ADDR[8:11]_CAL_ADDR[27:30]_GPIO[4:7]) — MPC5553 Only

ADDR[8:11]_ADDR[27:30]_GPIO[4:7] are the EBI address and calibration signals.

2.3.2.3 External Address / GPIO (ADDR[12:31]_GPIO[8:27])

ADDR[12:31]_GPIO[8:27] are the EBI address signals.

2.3.2.4 External Data — MPC5554 Only

Both the MPC5553 and the MPC5554 can be configured for 16-bit or 32-bit data bus operation.

2.3.2.4.1 External Data / GPIO (DATA[0:31]_GPIO[28:59]) — MPC5554 Only

DATA[0:31]_GPIO[28:59] are the EBI data signals. The multiplexing of DATA[0:31]_GPIO[28:59] occur in the MPC5554 only.

2.3.2.5 External Data — MPC5553 Only

2.3.2.5.1 External Data / GPIO (DATA[0:15]_GPIO[28:43]) — MPC5553 Only

2.3.2.5.2 DATA[0:15]_GPIO[28:43] are the EBI data signals. The data signals can be split as half data and half GPIO for 16-bit data bus operation.**External Data / FEC / Calibration Data / GPIO (DATA16_TX_CLK_CAL_DATA0_GPIO44) — MPC5553 Only**

DATA16_TX_CLK_CAL_DATA0_GPIO44 are the EBI, FEC transmit clock, and calibration data signals.

2.3.2.5.3 External Data / FEC / Calibration Data / GPIO (DATA17_CRS_CAL_DATA1_GPIO45) — MPC5553 Only

DATA17_CRS_CAL_DATA1_GPIO45 are the EBI, FEC carrier sense, and calibration signals.

2.3.2.5.4 External Data / FEC / Calibration Data / GPIO (DATA18_TX_ER_CAL_DATA2_GPIO46) — MPC5553 Only

DATA18_TX_ER_CAL_DATA2_GPIO46 are the EBI, FEC transmit error, and calibration signals.

2.3.2.5.5 External Data / FEC / Calibration Data / GPIO (DATA19_RX_CLK_CAL_DATA3_GPIO47) — MPC5553 Only

DATA19_RX_CLK_CAL_DATA3_GPIO47 are the EBI, FEC receive clock, and calibration signals.

2.3.2.5.6 External Data / FEC / Calibration Data / GPIO (DATA20_TXD0_CAL_DATA4_GPIO48) — MPC5553 Only

DATA20_TXD0_CAL_DATA4_GPIO48 are the EBI, FEC transmit data, and calibration signals.

2.3.2.5.7 External Data / FEC / Calibration Data / GPIO (DATA21_RX_ER_CAL_DATA5_GPIO49) — MPC5553 Only

DATA21_RX_ER_CAL_DATA5_GPIO49 are the EBI, FEC receive error, and calibration signals.

2.3.2.5.8 External Data / FEC / Calibration Data / GPIO (DATA22_RXD0_CAL_DATA6_GPIO50) — MPC5553 Only

DATA22_RXD0_CAL_DATA6_GPIO50 are the EBI, FEC receive data, and calibration signals.

2.3.2.5.9 External Data / FEC / Calibration Data / GPIO (DATA23_TXD3_CAL_DATA7_GPIO51) — MPC5553 Only

DATA23_TXD3_CAL_DATA7_GPIO51 are the EBI, FEC transmit data, and calibration signals.

2.3.2.5.10 External Data / FEC / Calibration Data / GPIO (DATA24_COL_CAL_DATA8_GPIO52) — MPC5553 Only

DATA24_COL_CAL_DATA8_GPIO52 are the EBI, FEC collision detect, and calibration signals.

2.3.2.5.11 External Data / FEC / Calibration Data / GPIO (DATA25_RX_DV_CAL_DATA9_GPIO53) — MPC5553 Only

DATA25_RX_DV_CAL_DATA9_GPIO53 are the EBI, FEC receive data valid, and calibration signals.

2.3.2.5.12 External Data / FEC / Calibration Data / GPIO (DATA26_TX_EN_CAL_DATA10_GPIO54) — MPC5553 Only

DATA26_TX_EN_CAL_DATA10_GPIO54 are the EBI, FEC transmit enable, and calibration signals.

2.3.2.5.13 External Data / FEC / Calibration Data / GPIO (DATA27_TXD2_CAL_DATA11_GPIO55) — MPC5553 Only

DATA27_TXD2_CAL_DATA11_GPIO55 are the EBI, FEC transmit data, and calibration signals.

2.3.2.5.14 External Data / FEC / Calibration Data / GPIO (DATA28_TXD1_CAL_DATA12_GPIO56) — MPC5553 Only

DATA28_TXD1_CAL_DATA12_GPIO56 are the EBI, FEC transmit data, and calibration signals.

2.3.2.5.15 External Data / FEC / Calibration Data / GPIO (DATA29_RXD1_CAL_DATA13_GPIO57) — MPC5553 Only

DATA29_RXD1_CAL_DATA13_GPIO57 are the EBI, FEC receive data, and calibration signals.

2.3.2.5.16 External Data / FEC / Calibration Data / GPIO (DATA30_RXD2_CAL_DATA14_GPIO58) — MPC5553 Only

DATA30_RXD2_CAL_DATA14_GPIO58 are the EBI, FEC receive data, and calibration signals.

2.3.2.5.17 External Data / FEC / Calibration Data / GPIO (DATA31_RXD3_CAL_DATA15_GPIO59) — MPC5553 Only

DATA31_RXD3_CAL_DATA15_GPIO59 are the EBI, FEC receive data, and calibration signals.

2.3.2.6 External Transfer Size / GPIO (TSIZ[0:1]_GPIO[60:61]) — MPC5554 Only

TSIZ[0:1]_GPIO[60:61] indicate the size of an external bus transfer when in external master operation or in slave mode. The TSIZ[0:1] signals are not driven by the EBI in single master operation. The MPC5553 has no TSIZ[0:1].

2.3.2.7 External Read/Write / GPIO (RD_$\overline{\text{WR}}$_GPIO62)

RD_$\overline{\text{WR}}$_GPIO62 indicates whether an external bus transfer is a read or write operation.

2.3.2.8 External Burst Data In Progress / GPIO ($\overline{\text{BDIP}}$_GPIO63)

$\overline{\text{BDIP}}$_GPIO63 indicates that an EBI burst transfer is in progress.

2.3.2.9 Write Enables

2.3.2.9.1 External Write/Byte Enable / GPIO (WE[0:3]_BE[64:67]_GPIO[64:67]) — MPC5554 Only

WE[0:3]_BE[0:3]_GPIO[64:67] specify which data pins contain valid data for an external bus transfer.

2.3.2.9.2 External Write/Byte Enable / GPIO (WE[0:1]_BE[0:1]_GPIO[64:65]) — MPC5553 Only

WE[0:1]_BE[0:1]_GPIO[64:65] specify which data pins contain valid data for an external bus transfer.

2.3.2.9.3 External Write/Byte Enable / Calibration Write Enable / GPIO (WE[2:3]_BE[2:3]_CAL_WE[0:1]_GPIO[66:67]) — MPC5553 Only

WE[2:3]_BE[2:3]_CAL_WE[0:1]_GPIO[66:67] specify which data pins contain valid data for an external bus transfer and provide write enables for calibration in the MPC5553.

2.3.2.10 External Output Enable / GPIO (OE_GPIO68)

OE_GPIO68 indicates that the EBI is ready to accept read data.

2.3.2.11 External Transfer Start / GPIO (TS_GPIO69)

TS_GPIO69 is asserted by the EBI owner to indicate the start of a transfer.

2.3.2.12 External Transfer Acknowledge (TA_GPIO70)

TA_GPIO70 is asserted by the EBI owner to acknowledge that the slave has completed the current transfer.

2.3.2.12.1 External Transfer Error Acknowledge / GPIO (TEA_GPIO71) — MPC5554 Only

TEA_GPIO71 indicates that an error occurred in the current external bus transfer.

2.3.2.12.2 External Transfer Error Acknowledge / Calibration Chip Select / GPIO (TEA_CAL_CS0_GPIO71) — MPC5553 Only

TEA_CA;_CS0_GPIO71 indicates that an error occurred in the current external bus transfer, and provides a calibration chip select function in the MPC5553.

2.3.2.13 Calibration Address[1] / FEC / Calibration Chip Select / GPIO (CAL_ADDR10_MDC_CAL_CS2_GPIO72) — MPC5553 Only

CAL_ADDR10_MDC_CAL_CS2_GPIO72 are the calibration and FEC management clock output signals. Note that BR is not implemented on the MPC5553.

2.3.2.14 External Bus Request / GPIO (BR_GPIO72) — MPC5554 Only

BR_GPIO72 is used by an external bus master to request ownership of the EBI from the arbiter. BR is only functional on the MPC5554.

2.3.2.15 Calibration Address[1] / FEC / Calibration Chip Select / GPIO (CAL_ADDR11_MDIO_CAL_CS3_GPIO73) — MPC5553 Only

CAL_ADDR11_MDIO_CAL_CS3_GPIO73 are the calibration and FEC management data in/out signals. Note that BG is not implemented on the MPC5553.

2.3.2.16 External Bus Grant / GPIO (BG_GPIO73) — MPC5554 Only

BG_GPIO73 is used by the external bus arbiter to give ownership of the EBI to the requesting master. BG is only functional on the MPC5554.

2.3.2.17 External Bus Busy / GPIO (BB_GPIO74) — MPC5554 Only

BB_GPIO74 is asserted by the current external bus master during a transfer to indicate that the EBI is busy. BB is only functional on the MPC5554.

2.3.3 Nexus

2.3.3.1 Nexus Event In (EVTI)

EVTI is an input that is read on the negation of TRST to enable or disable the Nexus debug port. After reset, the EVTI pin is used to initiate program and data trace synchronization messages or generate a breakpoint.

2.3.3.2 Nexus Event Out (EVTO)

EVTO is an output that provides timing to a development tool for a single watchpoint or breakpoint occurrence.

1. The BR and BG functionality is not implemented on the MPC5553, it is replaced by calibration functionality. The pin name on the ball map, however, does remain BR and BG.

2.3.3.3 Nexus Message Clock Out (MCKO)

MCKO is a free running clock output to the development tools which is used for timing of the MDO and MSEO signals.

2.3.3.4 Nexus Message Data Out (MDO[3:0])

MDO[3:0] are the trace message outputs to the development tools.

In addition to being a trace output, MDO[0] indicates the lock status of the system clock following a power-on reset. MDO[0] is driven high following a power-on reset until the system clock achieves lock, at which time it is then negated. There is an internal pull up on MDO[0].

2.3.3.5 Nexus Message Data Out / GPIO (MDO[11:4]_GPIO[82:75])

MDO[11:4]_GPIO[82:75] are the trace message outputs to the development tools for full port mode. These pins function as GPIO when the Nexus port controller (NPC) operates in reduced port mode.

2.3.3.6 Nexus Message Start/End Out ($\overline{\text{MSEO}}$[1:0])

$\overline{\text{MSEO}}$[1:0] are outputs that indicate when messages start and end on the MDO pins.

2.3.3.7 Nexus Ready Output ($\overline{\text{RDY}}$)

$\overline{\text{RDY}}$ is an output that indicates to the development tools the data is ready to be read from or written to the Nexus read/write access registers.

2.3.4 JTAG

2.3.4.1 JTAG Test Clock Input (TCK)

TCK provides the clock input for the on-chip test logic.

2.3.4.2 JTAG Test Data Input (TDI)

TDI provides the serial test instruction and data input for the on-chip test logic.

2.3.4.3 JTAG Test Data Output (TDO)

TDO provides the serial test data output for the on-chip test logic.

2.3.4.4 JTAG Test Mode Select Input (TMS)

TMS controls test mode operations for the on-chip test logic.

2.3.4.5 JTAG Compliance Input (JCOMP)

The JCOMP pin is used to enable the JTAG TAP controller.

2.3.4.6 Test Mode Enable Input ($\overline{\text{TEST}}$)

The $\overline{\text{TEST}}$ pin is used to place the chip in test mode. It should be negated for normal operation.

2.3.5 FlexCAN

2.3.5.1 FlexCAN A Transmit / GPIO (CNTXA_GPIO83)

CNTXA_GPIO83 is the transmit pin for the FlexCAN A module.

2.3.5.2 FlexCAN A Receive / GPIO (CNRXA_GPIO84)

CNRXA_GPIO84 is the receive pin for the FlexCAN A module.

2.3.5.3 FlexCAN B Transmit / DSPI C Chip Select / GPIO (CNTXB_PCSC3_GPIO85)

CNTXB_PCSC3_GPIO85 is the transmit pin for the FlexCAN B module. The alternate function is a peripheral chip select output for the DSPI C module.

2.3.5.4 FlexCAN B Receive / DSPI C Chip Select / GPIO (CNRXB_PCSC4_GPIO86)

CNRXB_PCSC4_GPIO86 is the receive pin for the FlexCAN B module. The alternate function is a peripheral chip select output for the DSPI C module.

2.3.5.5 FlexCAN C Transmit / DSPI D Chip Select / GPIO (CNTXC_PCSD3_GPIO87)

CNTXC_PCSD3_GPIO87 is the transmit pin for the FlexCAN C module. The alternate function is a peripheral chip select for the DSPI D module.

2.3.5.6 FlexCAN C Receive / DSPI D Chip Select / GPIO (CNRXC_PCSD4_GPIO88)

CNRXC_PCSD4_GPIO88 is the receive pin for the FlexCAN C module. The alternate function is a peripheral chip select for the DSPI D module.

2.3.6 eSCI

2.3.6.1 eSCI_A Transmit / GPIO (TXDA_GPIO89)

TXDA_GPIO89 is the transmit pin for the eSCI A module.

2.3.6.2 eSCI_A Receive / GPIO (RXDA_GPIO90)

RXDA_GPIO90 is the receive pin for the eSCI A module. The pin is an input only for the RXD function and does not have a weak pull device, but as GPIO the pin is input or output based on the SIU PCR configuration.

2.3.6.3 eSCI B Transmit / DSPI D Chip Select / GPIO (TXDB_PCSD1_GPIO91)

TXDB_PCSD1_GPIO91 is the transmit pin for the eSCI B module. The alternate function is a peripheral chip select output for the DSPI D module.

2.3.6.4 eSCI B Receive / DSPI D Chip Select / GPIO (RXDB_PCSD5_GPIO92)

RXDB_PCSD5_GPIO92 is the transmit pin for the eSCI B module. The secondary function is a peripheral chip select for the DSPI D module.

2.3.7 DSPI

2.3.7.1 DSPI A Clock[1] / PCSC1 / GPIO (SCKA_PCSC1_GPIO93)

SCKA_PCSC1_GPIO93 is the SPI clock pin for the DSPI A module; PCSC1 is a peripheral chip select output pin for the DSPI C module. SCKA is not implemented on the MPC5553.

2.3.7.2 DSPI A Data Input[1] / PCSC2 / GPIO (SINA_PCSC2_GPIO94)

SINA_PCSC2_GPIO94 is the data input pin for the DSPI A module; PCSC2 is a peripheral chip select output pin for the DSPI C module. SINA is not implemented on the MPC5553.

1. In the MPC5553, the DSPI A module is not implemented. The MPC5554 does implement the DSPI A module, and both the MPC5553 and MPC5554 implement the secondary and tertiary functions of the pin.

2.3.7.3 DSPI A Data Output[1] / PCSC5 / GPIO (SOUTA_PCSC5_GPIO95)

SOUTA_PCSC5_GPIO95 is the data output pin for the DSPI A module; PCSC5 is a peripheral chip select output pin for the DSPI C module. SOUTA is not implemented on the MPC5553.

2.3.7.4 DSPI A Chip Selec[1]t / PCSD2 / GPIO (PCSA0_PCSD2_GPIO96)

PCSA0_PCSD2_GPIO96 are peripheral chip select output pins for the DSPI A module. PCSA0 also serves as the slave select input ($\overline{SS}$) of the DSPI A module. PCSD2 is a peripheral chip select output pin for the DSPI D module. PCSA0 is not implemented on the MPC5553.

2.3.7.5 DSPI A Chip Select[1] / PCSB2 / GPIO (PCSA1_PCSB2_GPIO97)

PCSA1_PCSB2_GPIO97 are peripheral chip select output pins for the DSPI A module. PCSB2 is a peripheral chip select output pin for the DSPI B module. PCSA1 is not implemented on the MPC5553.

2.3.7.6 DSPI A Chip Select[1] / DSPI D Clock / GPIO (PCSA2_SCKD_GPIO98)

PCSA2_SCKD_GPIO98 is a peripheral chip select output pin for the DSPI A module. The alternate function is the $\overline{SPI}$ clock for the DSPI D module. PCSA2 is not implemented on the MPC5553.

2.3.7.7 DSPI A Chip Select[1] / DSPI D Data Input / GPIO (PCSA3_SIND_GPIO99)

PCSA3_SIND_GPIO99 is a peripheral chip select output pin for the DSPI A module. The alternate function is the data input for the DSPI D module. PCSA3 is not implemented on the MPC5553.

2.3.7.8 DSPI A Chip Select[1] / DSPI D Data Output / GPIO (PCSA4_SOUTD_GPIO100)

PCSA4_SOUTD_GPIO100 is a peripheral chip select output pin for the DSPI A module. The alternate function is the data output for the DSPI D module. PCSA4 is not implemented on the MPC5553.

2.3.7.9 DSPI A Chip Select[1] /PCSB3 / GPIO (PCSA5_PCSB3_GPIO101)

PCSA5_PCSB3_GPIO101 is a peripheral chip select output pin for the DSPI A module. PCSB3 is a peripheral chip select output pin for the DSPI B module. PCSA 5 is not implemented on the MPC5553.

1. In the MPC5553, the DSPI A module is not implemented. The MPC5554 does implement the DSPI A module, and both the MPC5553 and MPC5554 implement the secondary and tertiary functions of the pin.

2.3.7.10 DSPI B Clock / DSPI C Chip Select / GPIO (SCKB_PCSC1_GPIO102)

SCKB_PCSC1_GPIO102 is the SPI clock pin for the DSPI B module. The alternate function is a chip select output for the DSPI C module.

2.3.7.11 DSPI B Data Input / DSPI C Chip Select / GPIO (SINB_PCSC2_GPIO103)

SINB_PCSC2_GPIO103 is the data input pin for the DSPI B module. The alternate function is a chip select output for the DSPI C module.

2.3.7.12 DSPI B Data Output / DSPI C Chip Select / GPIO (SOUTB_PCSC5_GPIO104)

SOUTB_PCSC5_GPIO104 is the data output pin for the DSPI B module. The alternate function is a chip select output for the DSPI C module.

2.3.7.13 DSPI B Chip Select / DSPI D Chip Select / GPIO (PCSB0_PCSD2_GPIO105)

PCSB0_PCSD2_GPIO105 is a peripheral chip select output pin (slave select input pin for slave operation) for the DSPI B module. The alternate function is a chip select output for the DSPI D module.

2.3.7.14 DSPI B Chip Select / DSPI D Chip Select / GPIO (PCSB1_PCSD0_GPIO106)

PCSB1_PCSD0_GPIO106 is a peripheral chip select output pin for the DSPI B module. The alternate function is a chip select output (slave select input pin for slave operation) for the DSPI D module.

2.3.7.15 DSPI B Chip Select / DSPI C Data Output / GPIO (PCSB2_SOUTC_GPIO107)

PCSB2_SOUTC_GPIO107 is a peripheral chip select output pin for the DSPI B module. The alternate function is the data output for the DSPI C module.

2.3.7.16 DSPI B Chip Select / DSPI C Data Input / GPIO (PCSB3_SINC_GPIO108)

PCSB3_SINC_GPIO108 is a peripheral chip select output pin for the DSPI B module. The alternate function is the data input for the DSPI C module.

2.3.7.17 DSPI B Chip Select / DSPI C Clock / GPIO (PCSB4_SCKC_GPIO109)

PCSB4_SCKC_GPIO109 is a peripheral chip select output pin for the DSPI B module. The alternate function is the SPI clock for the DSPI C module.

2.3.7.18 DSPI B Chip Select / DSPI C Chip Select / GPIO (PCSB5_PCSC0_GPIO110)

PCSB5_PCSC0_GPIO110 is a peripheral chip select output pin for the DSPI B module. The alternate function is a chip select output (slave select input in slave mode) for the DSPI C module.

2.3.8 eQADC

2.3.8.1 Analog Input / Differential Analog Input (AN0_DAN0+)

AN0 is a single-ended analog input to the two on-chip ADCs. DAN0+ is the positive terminal of the differential analog input DAN0 (DAN0+ to DAN0–).

2.3.8.2 Analog Input / Differential Analog Input (AN1_DAN0–)

AN1 is a single-ended analog input to the two on-chip ADCs. DAN0– is the negative terminal of the differential analog input DAN0 (DAN0+ to DAN0–).

2.3.8.3 Analog Input / Differential Analog Input (AN2_DAN1+)

AN2 is a single-ended analog input to the two on-chip ADCs. DAN1+ is the positive terminal of the differential analog input DAN1 (DAN1+ to DAN1–).

2.3.8.4 Analog Input / Differential Analog Input (AN3_DAN1–)

AN3 is a single-ended analog input to the two on-chip ADCs. DAN1– is the negative terminal of the differential analog input DAN1 (DAN1+ to DAN1–).

2.3.8.5 Analog Input / Differential Analog Input (AN4_DAN2+)

AN4 is a single-ended analog input to the two on-chip ADCs. DAN2+ is the positive terminal of the differential analog input DAN2 (DAN2+ to DAN2–).

2.3.8.6 Analog Input / Differential Analog Input (AN5_DAN2–)

AN5 is a single-ended analog input to the two on-chip ADCs. DAN2– is the negative terminal of the differential analog input DAN2 (DAN2+ to DAN2–).

2.3.8.7 Analog Input / Differential Analog Input (AN6_DAN3+)

AN6 is a single-ended analog input to the two on-chip ADCs. DAN3+ is the positive terminal of the differential analog input DAN3 (DAN3+ to DAN3–).

2.3.8.8 Analog Input / Differential Analog Input (AN7_DAN3–)

AN7 is a single-ended analog input to the two on-chip ADCs. DAN3– is the negative terminal of the differential analog input DAN3 (DAN3+ to DAN3–).

2.3.8.9 Analog Input / Multiplexed Analog Input (AN8_ANW)

AN8 is an analog input pin. The alternate function, ANW, is an analog input in external multiplexed mode. This pin has reduced analog to digital conversion accuracy as compared to the AN[0:7] and AN[16:39] analog input pins.

2.3.8.10 Analog Input / Multiplexed Analog Input / Test BIAS (AN9_ANX_BIAS[1])

AN9 is an analog input pin. The alternate function, ANX, is an analog input in external multiplexed mode. This pin has reduced analog to digital conversion accuracy as compared to the AN[0:7] and AN[16:39] analog input pins. BIAS is only implemented on the MPC5553; AN9 and ANX are implemented on both the MPC5553 and the MPC5554.

2.3.8.11 Analog Input / Multiplexed Analog Input (AN10_ANY)

AN10 is an analog input pin. The alternate function, ANY, is an analog input in external multiplexed mode. This pin has reduced analog to digital conversion accuracy as compared to the AN[0:7] and AN[16:39] analog input pins.

2.3.8.12 Analog Input / Multiplexed Analog Input (AN11_ANZ)

AN11 is an analog input pin. The alternate function, ANZ, is an analog input in external multiplexed mode. This pin has reduced analog to digital conversion accuracy as compared to the AN[0:7] and AN[16:39] analog input pins.

2.3.8.13 Analog Input / Mux Address 0 / eQADC Serial Data Strobe (AN12_MA0_$\overline{\text{SDS}}$)

AN12_MA0_$\overline{\text{SDS}}$ is an analog input pin. The alternate function, MA0, is a MUX address pin. The second alternate function is the serial data strobe for the eQADC SSI. This pin has reduced analog to digital conversion accuracy as compared to the AN[0:7] and AN[16:39] analog input pins.

2.3.8.14 Analog Input / Mux Address 1 / eQADC Serial Data Out (AN13_MA1_SDO)

AN13_MA1_SDO is an analog input pin. The alternate function, MA1, is a MUX address pin. The second alternate function is the serial data output for the eQADC SSI. This pin has reduced analog to digital conversion accuracy as compared to the AN[0:7] and AN[16:39] analog input pins.

1. BIAS is only implemented on the MPC5553.

2.3.8.15 Analog Input / Mux Address 2 / eQADC Serial Data In (AN14_MA2_SDI)

AN14_MA2_SDI is an analog input pin. The alternate function, MA2, is a MUX address pin. The second alternate function is the serial data input for the eQADC SSI. This pin has reduced analog to digital conversion accuracy as compared to the AN[0:7] and AN[16:39] analog input pins.

2.3.8.16 Analog Input / eQADC Free Running Clock (AN15_FCK)

AN15_FCK is an analog input pin. The alternate function is the free running clock for the eQADC SSI. This pin has reduced analog to digital conversion accuracy as compared to the AN[0:7] and AN[16:39] analog input pins.

2.3.8.17 Analog Input (AN[16:39])

AN[16:39] are analog input pins.

2.3.8.18 External Trigger / GPIO (ETRIG[0:1]_GPIO[111:112])

ETRIG[0:1]_GPIO[111:112] are external trigger input pins for the eQADC.

2.3.8.19 Voltage Reference High (VRH)

VRH is the voltage reference high input pin for the eQADC.

2.3.8.20 Voltage Reference Low (VRL)

VRL is the voltage reference low input pin for the eQADC.

2.3.8.21 Reference Bypass Capacitor (REFBYPC)

REFBYPC is a bypass capacitor input for the eQADC. The REFBYPC pin is used to connect an external bias capacitor between the REFBYPC pin and VRL. The value of this capacitor should be 100nF.

2.3.9 eTPU

2.3.9.1 eTPU A TCR Clock / External Interrupt Request / GPIO (TCRCLKA_IRQ7_GPIO113)

TCRCLKA_IRQ7_GPIO113 is the TCR A clock input for the eTPU module. The alternate function is an external interrupt request input for the SIU module.

2.3.9.2 eTPU A Channel / eTPU A Channel (Output Only) / GPIO (ETPUA[0:11]_ETPUA[12:23]_GPIO[114:125])

ETPUA[0:11]_ETPUA[12:23]_GPIO[114:125] are input/output channel pins for the eTPU A module. The alternate function is for output channels of the eTPU A module; that is, when configured as ETPUA[12:23], the pins function as outputs only.

2.3.9.3 eTPU A Channel / DSPI / GPIO (ETPUA[12:19]_PCS*Xn*_GPIO[126:133])

ETPUA[12:19]_PCS*Xn*_GPIO[126:133] are input/output channel pins for the eTPU A module muxed with DSPI B and D pins.

2.3.9.4 eTPU A Channel / External Interrupt Request / GPIO (ETPUA[20:27]_IRQ[8:15]_GPIO[134:141])

ETPUA[20:27]_IRQ[8:15]_GPIO[134:141] are input/output channel pins for the eTPU A module muxed with interrupt request pins.

2.3.9.5 eTPU A Channel / DSPI / GPIO (ETPUA[28:31]_PCSC[1:4]_GPIO[142:145])

ETPUA[28:31]_PCSC[1:4]_GPIO[142:145] are input/output channel pins for the eTPU A module multiplexed with DSPI C pins.

2.3.9.6 eTPU B TCR Clock / External Interrupt Request / GPIO (TCRCLKB_IRQ6_GPIO146) — MPC5554 Only

TCRCLKB_IRQ6_GPIO146 is the TCR B clock input for the eTPU module. The alternate function is an external interrupt request input for the SIU module. This pin functions only on the MPC5554.

2.3.9.7 eTPU B Channel / eTPU B Channel (Output Only) / GPIO (ETPUB[0:15]_ETPUB[16:31]_GPIO[147:162]) — MPC5554 Only

ETPUB[0:15]_ETPUB[16:31]_GPIO[147:162] are 16 input/output channel pins for the eTPU B module. The alternate function is for output channels for the eTPU B module; that is, when configured as ETPUB[16:31], the pins function as outputs only. This pin functions only on the MPC5554.

2.3.9.8 eTPU B Channel / DSPI / GPIO (ETPUB[16:19]_PCSA[1:4]_GPIO[163:166]) — MPC5554 Only

ETPUB[16:19]_PCSA[1:4]_GPIO[163:166] are input/output channel pins for the eTPU B module and DSPI A functionality is the alternate. This pin functions only on the MPC5554.

2.3.9.9 eTPU B Channel / GPIO (ETPUB[20:31]_GPIO[167:178]) — MPC5554 Only

ETPUB[20:31]_GPIO[167:178] are input/output channel pins for the eTPU B module. This pin functions only on the MPC5554.

2.3.10 eMIOS

2.3.10.1 EMIOS Channel / eTPU A Channel (Output Only) / GPIO (EMIOS[0:9]_ETPUA[0:9]_GPIO[179:188])

EMIOS[0:9]_ETPUA[0:9]_GPIO[179:188] are input/output channel pins for the eMIOS module. The alternate function is output channels for the eTPU A module; that is, when configured as ETPUA[0:9], the pins function as outputs only.

2.3.10.2 EMIOS Channel / GPIO (EMIOS[10:11]_GPIO[189:190])

EMIOS[10:11]_GPIO[189:190] are input/output channel pins for the eMIOS module.

2.3.10.3 eMIOS Channel (Output Only) / DSPI C Data Output / GPIO (EMIOS12_SOUTC_GPIO191)

EMIOS12_SOUTC_GPIO191 is an output channel pin for the eMIOS module. The alternate function is the data output for the DSPI C module.

2.3.10.4 eMIOS Channel (Output Only) / DSPI D Data Output / GPIO (EMIOS13_SOUTD_GPIO192)

EMIOS[13]_SOUTD_GPIO[192] is an output channel pin for the eMIOS module. The alternate function is the data output for the DSPI D module.

2.3.10.5 eMIOS Channel (Output Only) / External Interrupt Request / GPIO (EMIOS[14:15]_IRQ[0:1]_GPIO[193:194])

EMIOS[14:15]_IRQ[0:1]_GPIO[193:194] are output channel pins for the eMIOS module. The alternate function is for external interrupt request inputs.

2.3.10.6 eMIOS Channel / eTPU Channel (Output Only) / GPIO (EMIOS[16:23]_ETPUB[0:7]_GPIO[195:202])

EMIOS[16:23]_ETPUB[0:7]_GPIO[195:202] are input/output channel pins for the eMIOS module. The alternate function is for output channels for the eTPU B module; that is, when configured as ETPUB[0:7], the pins function as outputs only.

2.3.11 GPIO

2.3.11.1 GPIO (GPIO[203:204]_EMIOS[14:15])

The GPIO[203:204]_EMIOS[14:15] pins' primary function is EMIOS[14:15]. When configured as EMIOS[14:15], the pins function as output channels for the eMIOS module. Because other balls already are named EMIOS[14:15], the balls for these signals are named GPIO[203:204]. The alternate function for these pins is GPIO.

2.3.11.2 GPIO (GPIO[205:207])

The GPIO[205:207] pins only have GPIO functionality. These pins are reserved for double data rate memory interface support. Note that the pad type for GPIO205 is medium driver and CMOS input buffer, 5V capability. The pad type for GPIO[206:207] is fast driver and CMOS input buffer (1.62V–1.98V). The GPIO[206:207] pins can be selected as sources for the ADC trigger in the SIU_ETISR. See Section 6.3.1.12.97, "Pad Configuration Registers 206 - 207 (SIU_PCR206 - SIU_PCR207)."

2.3.12 Clock Synthesizer

2.3.12.1 Crystal Oscillator Output (XTAL)

XTAL is the output pin for an external crystal oscillator.

2.3.12.2 Crystal Oscillator Input / External Clock Input (EXTAL_EXTCLK)

EXTAL is the input pin for an external crystal oscillator or an external clock source. The alternate function is the external clock input. The function of this pin is determined by the PLLCFG configuration pins.

2.3.12.3 System Clock Output (CLKOUT)

CLKOUT is an MPC5553/MPC5554 system clock output.

2.3.12.4 Engineering Clock Output (ENGCLK)

ENGCLK is a 50% duty cycle output clock with a maximum frequency of the MPC5553/MPC5554 system clock divided by two. ENGCLK is not synchronous to CLKOUT.

2.3.13 Power/Ground

2.3.13.1 Voltage Regulator Control Supply Input (V_{RC33})

V_{RC33} is the 3.3V supply input pin for the on-chip 1.5-V regulator control circuit.

2.3.13.2 Voltage Regulator Control Ground Input (V_{RCVSS})

V_{RCVSS} is the ground reference for the on-chip 1.5-V regulator control circuit.

2.3.13.3 Voltage Regulator Control Output (V_{RCCTL})

V_{RCCTL} is the output pin for the on-chip 1.5-V regulator control circuit.

2.3.13.4 eQADC Analog Supply (V_{DDA}*n*)

V_{DDA}*n* is the analog supply input pin for the eQADC.

2.3.13.5 eQADC Analog Ground Reference (V_{SSA}*n*)

V_{SSA}*n* is the analog ground reference input pin for the eQADC.

2.3.13.6 Clock Synthesizer Power Input (V_{DDSYN})

V_{DDSYN} is the power supply input for the FMPLL.

2.3.13.7 Clock Synthesizer Ground Input (V_{SSSYN})

V_{SSSYN} is the ground reference input for the FMPLL.

2.3.13.8 Flash Read Supply Input (V_{FLASH})

V_{FLASH} is the on-chip Flash read supply input.

2.3.13.9 Flash Program/Erase Supply Input (V_{PP})

V_{PP} is the on-chip Flash program/erase supply input.

2.3.13.10 SRAM Standby Power Input (V_{STBY})

V_{STBY} is the power supply input that is used to maintain a portion of the contents of internal SRAM during power down. If not used, V_{STBY} is tied toV_{SS}.

2.3.13.11 Internal Logic Supply Input (V_{DD})

V_{DD} is the 1.5V logic supply input.

2.3.13.12 External I/O Supply Input (V_{DDE})

V_{DDE} is the 1.8V to 3.3V +/– 10% external I/O supply input.

2.3.13.13 External I/O Supply Input ($V_{DDEH}n$)

$V_{DDEH}n$ is the 3.3V to 5.0V -10%/+5% external I/O supply input.

2.3.13.14 Fixed 3.3V Internal Supply Input (V_{DD33})

V_{DD33} is the 3.3V internal supply input.

2.3.13.15 Ground (V_{SS})

V_{SS} is the ground reference input.

2.3.14 I/O Power/Ground Segmentation

Table 2-3 gives the preliminary power/ground segmentation of the MPC5553 MCU and Table 2-4 gives the preliminary power/ground segmentation of the MPC5554 MCU. Each segment provides the power and ground for the given set of I/O pins. Each segment can be powered by any voltage within the allowed voltage range regardless of the power on the other segments. The power/ground segmentation applies regardless of whether a particular pin is configured for its primary function or GPIO.

Table 2-3. MPC5553 Power/Ground Segmentation[1]

Power Segment (V_{DDE})	V_{DDE} Package Ball Numbers	Voltage Range[2]	I/O Pins Powered by Segment
V_{DDEH1}	E3, F4	3.3V - 5.0V	ETPUA[0:31], TCRCLKA
V_{DDEH4}	AC20	3.3V - 5.0V	EMIOS[0:23], CNTXA, CNRXA, CNTXB, CNRXB
V_{DDEH6}	AA23	3.3V - 5.0V	$\overline{\text{RESET}}$, $\overline{\text{RSTOUT}}$, $\overline{\text{RSTCFG}}$, WKPCFG, BOOTCFG[0:1], PLLCFG[0:1], CNTXC, CNRXC, TXDA, RXDA, TXDB, RXDB, SCKA, SINA, SOUTA, PCSA[0:5], PCSB[3:5], GPIO[203:204]
V_{DDEH8}	D22	3.3V - 5.0V	ETRIG[0:1], GPIO[205]
V_{DDEH9}	D14	3.3V - 5.0V	AN12, AN13, AN14, AN15
V_{DDEH10}	J23	3.3V - 5.0V	SCKB, PCSB[0:2], SINB, SOUTB
V_{DDE2}[3]	T1, T4, Y4, AB1, AF5, AC8, AF11, AC13, M10, N10, P10, R10, T10, M11, N11, P11, R11, U11, T12, U12, T13, U13, T14, U14, T15, U15	1.8V - 3.3V	ADDR[8:31], $\overline{\text{WE}}$[0:3], $\overline{\text{CS}}$[0:3], $\overline{\text{BDIP}}$, RD_$\overline{\text{WR}}$, $\overline{\text{TS}}$, $\overline{\text{TA}}$, $\overline{\text{TEA}}$, DATA[0:31], $\overline{\text{OE}}$, $\overline{\text{BR}}$, $\overline{\text{BG}}$, GPIO[206:207] **Note:** V_{DDE2} and VDDE3 are separate segments in the MPC5553 pad ring. These segments are shorted together in the package substrate. The following pins are part of the VDDE3 segment: DATA[0:31], $\overline{\text{OE}}$, $\overline{\text{BR}}$, $\overline{\text{BG}}$.
V_{DDE5}	AC21, AD22, AE23, AF24	1.8V - 3.3V	CLKOUT, ENGCLK
V_{DDE7}	B26, C25, D24, E23, K14, K15, K16, K17, L17, M17, N17	1.8V–3.3V	MDO[11:0], $\overline{\text{EVTI}}$, $\overline{\text{EVTO}}$, MCKO, $\overline{\text{RDY}}$, $\overline{\text{MSEO}}$[1:0], TDO, TDI, TMS, TCK, JCOMP, $\overline{\text{TEST}}$
V_{DDSYN}	AC26	3.3V	XTAL, EXTAL
V_{RC33}	AC25	3.3V	V_{RCCTL}
V_{DDA0}	C14	5.0V	AN[22:35], VRH, VRL, REFBYPC
V_{DDA1}	A5	5.0V	AN[0:11, 16:21, 36:39]
V_{SSA0}	A14, B14	GND	—
V_{SSA1}	C6	GND	—
Other Power Segments			
V_{PP}	T26	4.5V–5.25V[4]	—
V_{FLASH}	U26	3.0V–3.6V	—
V_{DD33}	C1, U4, AD9, A25, AD26	3.0V–3.6V	—
V_{STBY}	A2	0.9V–1.1V	—

[1] This table applies only to the 416 package of the MPC5553.

[2] These are nominal voltages. All V_{DDE} and V_{DDEH} voltages are +/– 10% (V_{DDE} 1.62V to 3.6V, V_{DDEH} 3.0V to 5.5V). V_{RC33} is +/– 10%. V_{DDSYN} is +/– 10%. V_{DDA1} is + 5%, –10%.

[3] V_{DDE2} and VDDE3 are separate segments in the MPC5553 pad ring. These segments are shorted together in the package substrate. The following pins are part of the VDDE3 segment: DATA[0:31], $\overline{OE}$, $\overline{BR}$, $\overline{BG}$.

[4] During read operations, VPP can be as high as 5.3V and as low as 3.0V.

Table 2-4. MPC5554 Power/Ground Segmentation

Power Segment (V_{DDE})	V_{DDE} Package Ball Numbers	Voltage Range[1]	I/O Pins Powered by Segment
V_{DDEH1}	E3, F4	3.3V–5.0V	ETPUA[0:31], TCRCLKA
V_{DDEH4}	AC20	3.3V–5.0V	EMIOS[0:23], CNTXA, CNRXA, CNTXB, CNRXB
V_{DDEH6}[2]	AA23, J23	3.3V–5.0V	$\overline{RESET}$, $\overline{RSTOUT}$, $\overline{RSTCFG}$, WKPCFG, BOOTCFG[0:1], PLLCFG[0:1], CNTXC, CNRXC, TXDA, RXDA, TXDB, RXDB, SCKA, SINA, SOUTA, PCSA[0:5], SCKB, SINB, SOUTB, PCSB[0:5], GPIO[203:204], ETPUB[0:15], TCRCLKB
V_{DDEH8}	D22	3.3V–5.0V	ETPUB[16:31], ETRIG[0:1], GPIO205
V_{DDEH9}	D14	3.3V–5.0V	AN12, AN13, AN14, AN15
V_{DDE2}[3]	T1, T4, Y4, AB1, AF5, AC8, AF11, AC13, M10, N10, P10, R10, T10, M11, N11, P11, R11, U11, T12, U12, T13, U13, T14, U14, T15, U15	1.8V–3.3V	ADDR[8:31], $\overline{WE}$[0:3], $\overline{CS}$[0:3], $\overline{BDIP}$, RD_$\overline{WR}$, $\overline{TS}$, $\overline{TA}$, $\overline{TEA}$, TSIZ[0:1], **Note:** V_{DDE2} and VDDE3 are separate segments in the MPC5553 pad ring. These segments are shorted together in the package substrate. The following pins are part of the VDDE3 segment: DATA[0:31], GPIO[206:207], and $\overline{BR}$, $\overline{BB}$, $\overline{BG}$, $\overline{OE}$.
V_{DDE5}	AC21, AD22, AE23, AF24	1.8V–3.3V	CLKOUT, ENGCLK
V_{DDE7}	B26, C25, D24, E23, K14, K15, K16, K17, L17, M17, N17	1.8V–3.3V	MDO[11:0], $\overline{EVTI}$, $\overline{EVTO}$, MCKO, $\overline{RDY}$, $\overline{MSEO}$[1:0], TDO, TDI, TMS, TCK, JCOMP, $\overline{TEST}$
V_{DDA0}	C14	5.0V	AN[22:35], VRH, VRL, REFBYPC
V_{DDA1}	A5	5.0V	AN[0:11, 16:21, 36:39]
V_{SSA0}	A14, B14	GND	—
V_{SSA1}	C6	GND	—
V_{DDSYN}	AC26	3.3V	XTAL, EXTAL
V_{RC33}	AC25	3.3V	V_{RCCTL}
Other Power Segments			
V_{PP}	T26	4.5V–5.25V[4]	—
V_{FLASH}	U26	3.0V–3.6V	—
V_{DD33}	C1, U4, AD9, A25, AD26	3.0V–3.6V	—
V_{STBY}	A2	0.9V–1.1V	—

[1] These are nominal voltages. All V_{DDE} and V_{DDEH} voltages are +/– 10% (V_{DDE} 1.62V to 3.6V, V_{DDEH} 3.0V to 5.5V). V_{RC33} is +/– 10%. V_{DDSYN} is +/– 10%. V_{DDA1} is + 5%, –10%.

[2] When the FMPLL is configured for external reference mode, the V_{DDE5} supply affects the acceptable signal levels for the external reference. See Section 11.1.4.2, "External Reference Mode."

[3] V_{DDE2} and V_{DDE3} are separate segments in the MPC5554 pad ring. These segments are shorted together in the package substrate. The following pins are part of the V_{DDE3} segment: DATA[0:31], GPIO[206:207], $\overline{BR}$, $\overline{BB}$, $\overline{BG}$, and $\overline{OE}$.

[4] During read operations, V_{PP} can be as high as 5.3V and as low as 3.0V.

2.4 eTPU Pin Connections and Serialization

2.4.1 ETPUA[0:15]

The ETPUA[0:15] module channels connect to external pins or may be serialized out through the DSPIC module. A diagram for the ETPUA[0:15] / SOUTC connection is given in Figure 2-3. The full list of connections is given in Table 2-5. Although not shown in Figure 2-3, the output channels of ETPUA[12:15] are connected to the ETPUA[0:3]_ETPUA[12:15]_GPIO[114:117] pins.

The eTPU TCRA clock input is connected to an external pin only.

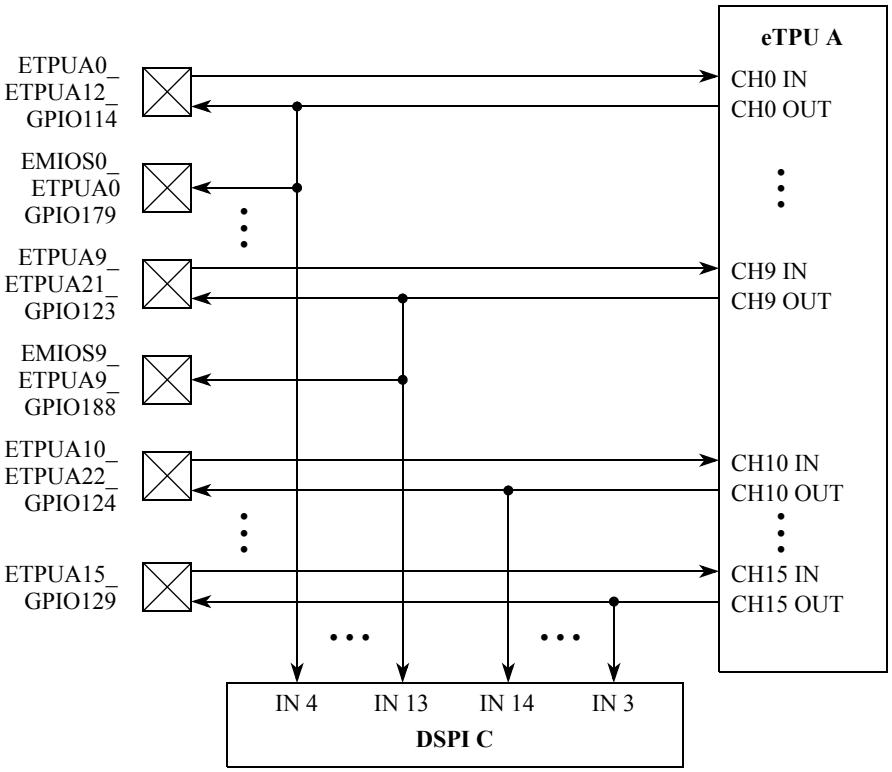


Figure 2-3. ETPUA[0:15]—DSPI C I/O Connections

Table 2-5. ETPUA[0:15]—DSPI C I/O Mapping

DSPI C Serialized Input	eTPU A Channel Output
15	11
14	10
13	9
12	8
11	7
10	6
9	5
8	4
7	3
6	2
5	1
4	0
3	15
2	14
1	13
0	12

2.4.2 ETPUA[16:31]

ETPUA[16:23,30:31] connect to external pins for both the input and output function. ETPUA[16:21,24:29] are serialized out on the DSPI B and DSPI D modules and ETPUA[22:23,30:31] are not serialized out. ETPUA[24:29] connect to external pins for only the output function. Figure 2-4 shows the connections for ETPUA16 and applies to ETPUA[16:21]. Figure 2-5 shows the connections for ETPUA24 and applies to TPUA[24:29]. The full ETPUA to DSPI B connections are given in Table 2-6, and ETPU A to DSPI D in Table 2-7. Although not shown in Figure 2-4, the output channels of ETPUA[16:23] are also connected to the ETPUA[4:11]_ETPUA[16:23]_GPIO[118:125] pins.

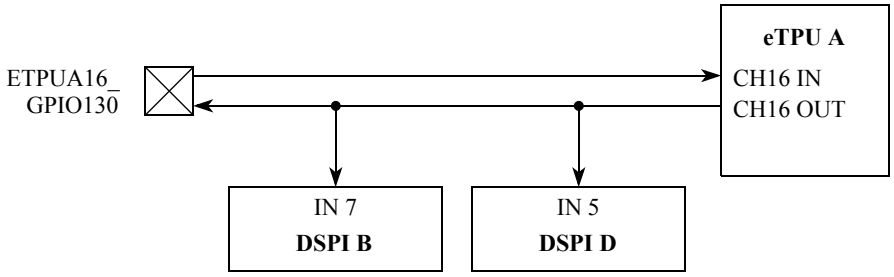


Figure 2-4. ETPUA[16:21]—DSPI B–DSPI D I/O Connections

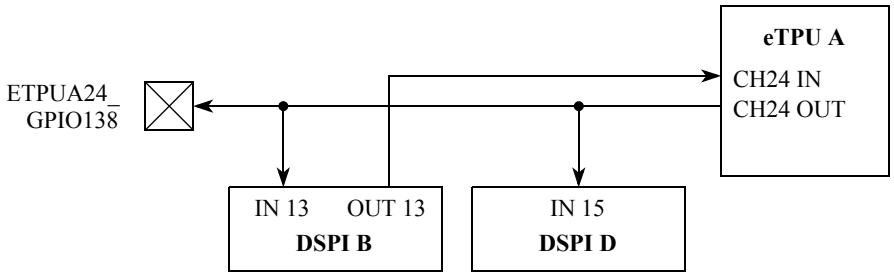


Figure 2-5. ETPUA[24:29]—DSPI B–DSPI D I/O Connections

Table 2-6. ETPUA[16:31]—DSPI B I/O Mapping

DSPI B Serialized Inputs / Outputs[1]	eTPU A Channel Output	eTPU A Channel Input
13	24	24
12	25	25
11	26	26
10	27	27
9	28	28
8	29	29
7	16	—
6	17	—
5	18	—
4	19	—
3	20	—
2	21	—

1 DSPI B serialized input channels 0, 1, 14, and 15 are connected to EMIOS channels. DSPI B serialized output channels 14, 15 are connected to EMIOS channels. DSPI B serialized output channels 0–7 are not connected.

Table 2-7. ETPUA[16:31]—DSPI D I/O Mapping

DSPI D Serialized Inputs[1]	eTPU A Channel Output
15	24
14	25
13	26
12	27
11	28
10	29
5	16

Table 2-7. ETPUA[16:31]—DSPI D I/O Mapping

DSPI D Serialized Inputs[1]	eTPU A Channel Output
4	17
3	18
2	19
1	20
0	21

[1] DSPI D serialized input channels 6–9 are connected to EMIOS channels.

2.4.3 ETPUB[0:31] — MPC5554 Only

The I/O connections for ETPUB[0:31] channels are given in Figure 2-6. The outputs of ETPUB[16:31] are connected to two pins. This allows the input and output of those channels to be connected to different pins. The outputs of ETPUB[16:31] are multiplexed on the ETPUB[0:15] pins. The outputs of ETPUB[0:7] are multiplexed on the EMIOS[16:23] pins so that the output channels of ETPUB[0:7] can be used when the normal pins for these channels are used by ETPUB[16:23] channels. The output channels of ETPUB[0:15] are serialized on DSPI A. The full ETPUB to DSPI A connections are given in Table 2-8.

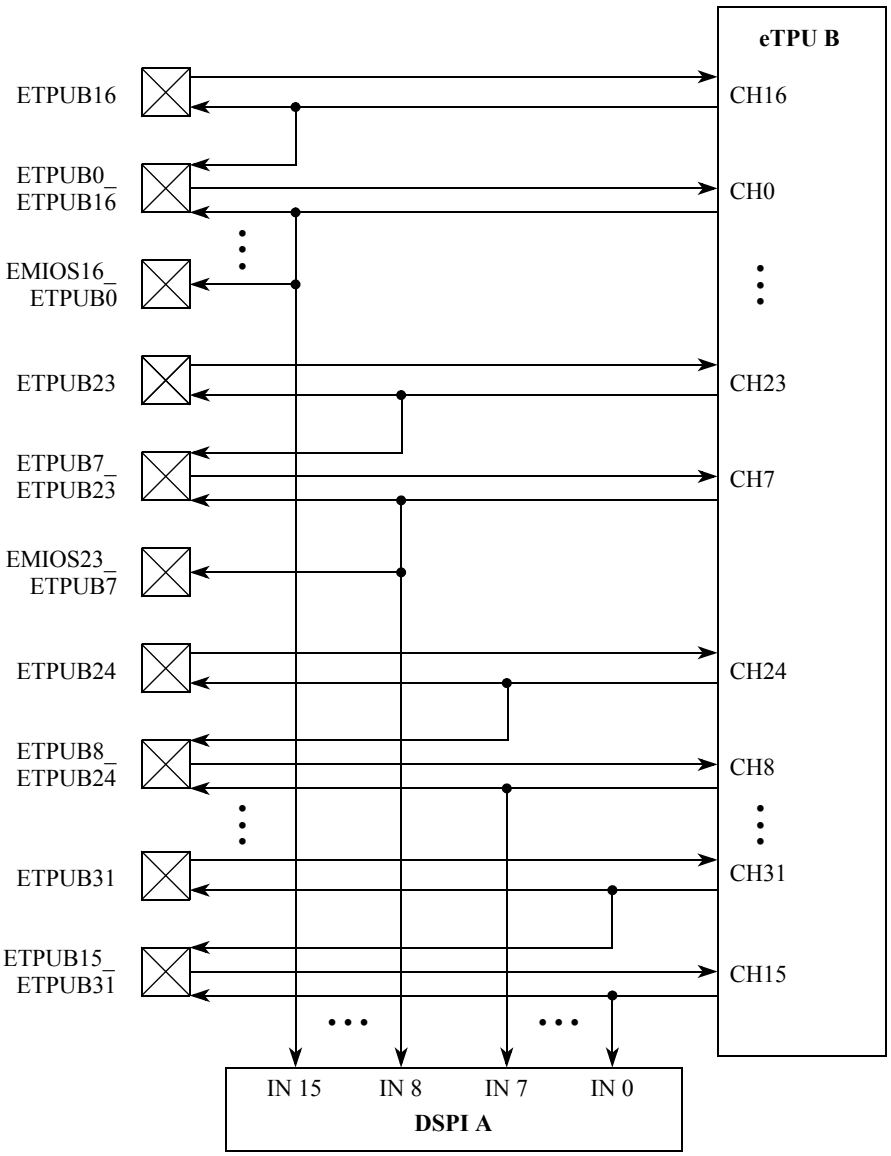


Figure 2-6. ETPUB[31:0]—DSPI A I/O Connections

Table 2-8. ETPUB[0:15]—DSPI A I/O Mapping

DSPI A Serialized Inputs	eTPU B Channel Output
15	0
14	1
13	2
12	3
11	4
10	5

Table 2-8. ETPUB[0:15]—DSPI A I/O Mapping

DSPI A Serialized Inputs	eTPU B Channel Output
9	6
8	7
7	8
6	9
5	10
4	11
3	12
2	13
1	14
0	15

2.5 eMIOS Pin Connections and Serialization

The eMIOS channels connect to external pins or may be serialized in and out of the MPC5553/MPC5554. The input and output channels of EMIOS[0:11, 16:23] connect to pins. Only the output channels of EMIOS[12:15] connect to pins. The output channels of EMIOS[10:13] may be serialized out, and the inputs of EMIOS[12:15] may be serialized in. The DSPI connections for EMIOS[10:11] are given in Figure 2-7, Figure 2-8 for EMIOS[12:13], and Figure 2-9 for EMIOS[14:15].

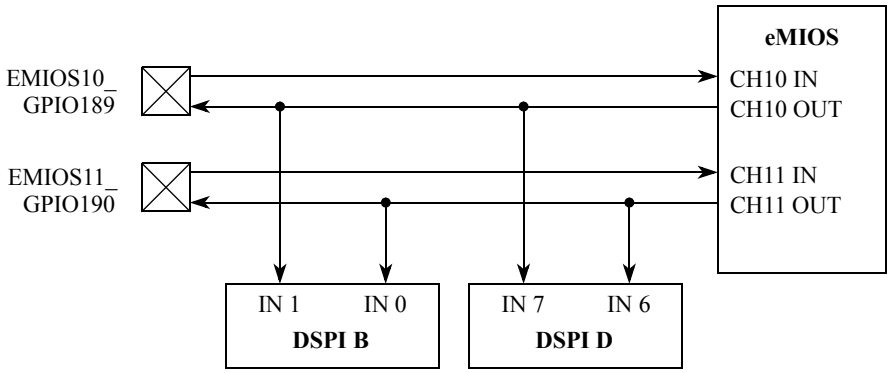


Figure 2-7. EMIOS[10:11]—DSPI B–DSPI D I/O Connections

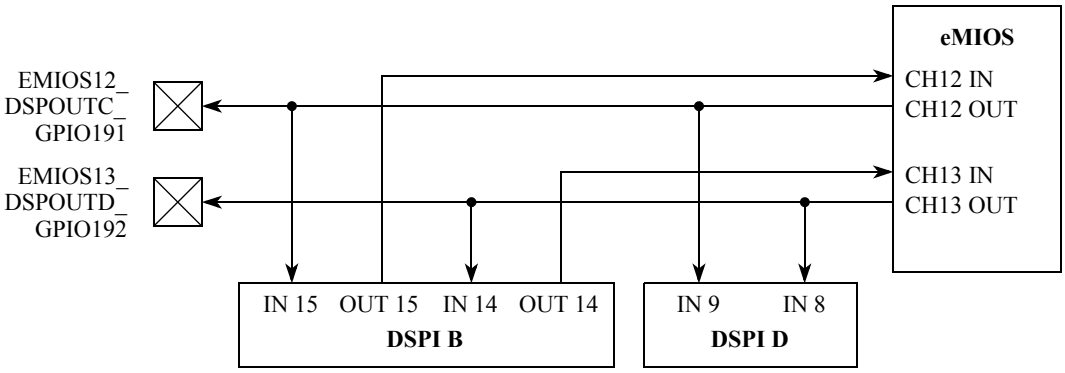


Figure 2-8. EMIOS[12:13]—DSPI B–DSPI D I/O Connections

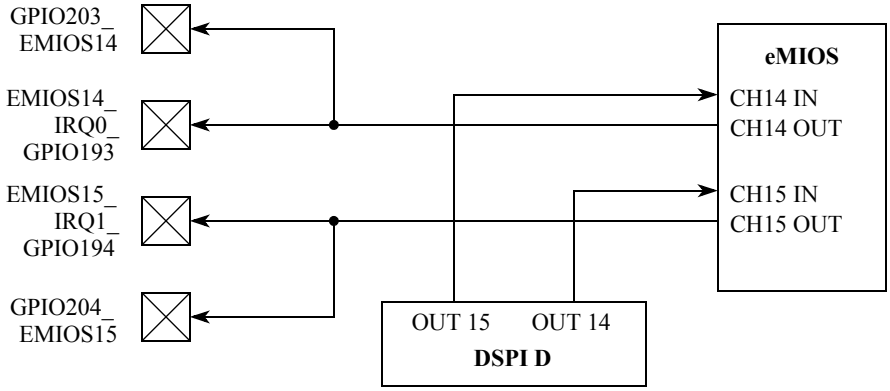


Figure 2-9. EMIOS[14:15]—DSPI D I/O Connections

2.6 Revision History

Substantive Changes since Rev 3.0
Table 2-2, Changed ADDR[12:31] to ADDR[8:31], changed GPIO[8:27] to GPIO[4:27]; Left pin listing as is - it correctly shows 24 pins.
Replaced existing Note on status after reset of MCKO in Table 2-1 and Table 2-2 with this note: "MCKO is only enabled if debug mode is enabled. Debug mode can be enabled before or after exiting System Reset (RSTOUT negated)."
Fixed typo (6 occurences of $V_{ddeh1}0$ changed to V_{ddeh10}.
Updated Section 2.3.13.12, "External I/O Supply Input (VDDE)" and Section 2.3.13.13, "External I/O Supply Input (VDDEHn) with: VDDE +/- 10% of 1.8 to 3.3 nominal and VDDEH -10%/+5% of the nominal 3.3 to 5.0 volts.
The ethernet signal TX_ER was changed from an input to an output.
Table 2-2 changed REFBYPC to say "Reference Bypass Capacitor Input" instead of "Reference Bypass Resistor Input"
Fixed voltage signal name subscripts. Removed V_{SUP} from MPC5553 signals diagram
Moved GPIO85_PCSC3_CNTXB and GPIO86_PCSC4_CNRXB from the FlexCAN group of signals to the DSPI group of signals in MPC5553 signals diagram (Figure 2-1) and in Table 2-1.
Updated GPIO[206:207] in table and in description to reflect a change in SIU PCR section (added Note to Section 6.3.1.12.97, "Pad Configuration Registers 206 - 207 (SIU_PCR206 - SIU_PCR207)" describing ETRIG functionality. NOTE: The GPIO[206:7] pins have the capability to trigger the ADCs. For the ETRIG functionality, these GPIO pins need to be set as GPIO and then select the GPIO ADC trigger in the eQADC Trigger Input Select Register (SIU_ETISR).")
In Section 2.3.6.2, "eSCI_A Receive / GPIO (RXDA_GPIO90)," changed an ambiguous input-only sentence to read "The pin is an input only for the RXD function and does not have a weak pull device, but as GPIO the pin is input or output based on the SIU PCR configuration."

Chapter 3
e200z6 Core Complex

3.1 Introduction

The core complex of the MPC5553/MPC5554 consists of the e200z6 core, a 32 Kbyte (MPC5554)/8 kilobyte (MPC5553) unified cache memory, a 32-entry memory management unit (MMU), a Nexus Class 3 block, and a bus interface unit (BIU). The e200z6 core is the central processing unit (CPU) in the MPC5553/MPC5554. The e200z6 core is part of a family of CPU cores that implement low-cost versions of the PowerPC Book E architecture. Refer to the *e200z6 PowerPC™ Core Reference Manual* for more information on the e200z6 core.

3.1.1 Block Diagram

Figure 3-1 shows a block diagram of the e200z6 core complex.

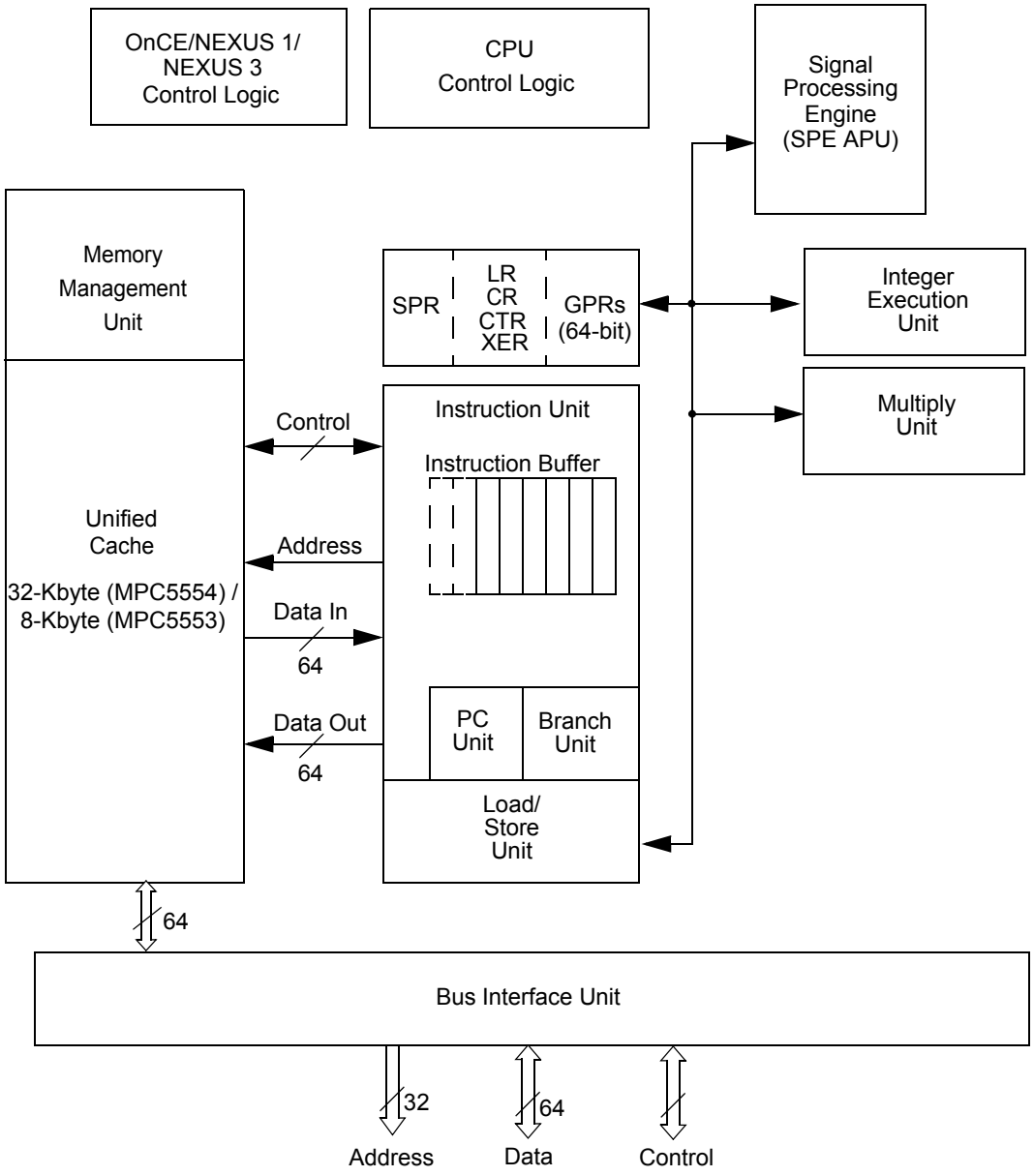


Figure 3-1. e200z6 Block Diagram

3.1.2 Overview

The e200z6 core integrates an integer execution unit, branch control unit, instruction fetch and load/store units, and a multi-ported register file capable of sustaining three read and two write operations per clock. Most integer instructions execute in a single clock cycle. Branch target prefetching is performed by the branch target address cache to allow single-cycle branches in many cases.

The e200z6 core complex is a single-issue, 32-bit PowerPC Book E compliant design with 64-bit general-purpose registers (GPRs). PowerPC Book E floating-point instructions are not supported in hardware, but are trapped and may be emulated by software. A signal processing extension (SPE) auxiliary processing unit (APU) is provided to support real-time fixed point and single-precision floating point operations using the general-purpose registers. All arithmetic instructions that execute in the core operate on data in the GPRs. The registers have been extended to 64-bits in order to support vector instructions defined by the SPE APU. These instructions operate on 16-bit or 32-bit data types, and produce vector or scalar results.

3.1.3 Features

The following is a list of some of the key features of the e200z6:

- Single issue, 32-bit PowerPC Book E compliant CPU
- In-order execution and retirement
- Precise exception handling
- Branch target address cache
  - — Dedicated branch address calculation adder
  - — Branch target prefetching
  - — Branch lookahead buffers of depth 2
- Load/store unit
  - — Pipelined operation supports throughput of one load or store operation per cycle
- 64-bit general-purpose register file
- Memory management unit (MMU) with 32-entry fully-associative TLB and multiple page size support
- 32 kilobyte, 8-way set associative unified cache in the MPC5554; 8 kilobyte, 2-way set-associative unified cache in the MPC5553
- Periodic timer and watchdog functions
- Signal processing extension APU supporting fixed-point and single-precision floating-point operations, using the 64-bit general-purpose register file
- Nexus class 3 real-time development unit
- Power management
  - — Low power design
  - — Dynamic power management of execution units, caches and MMUs

3.1.3.1 Instruction Unit Features

The features of the instruction unit are the following:

- 64-bit path to cache supports fetching of two 32-bit instructions per clock
- Instruction buffer holds up to 6 sequential instructions
- Dedicated PC incrementer supporting instruction prefetches
- Branch target address cache with dedicated branch address adder, and branch lookahead logic supporting single cycle execution of successful lookahead branches

3.1.3.2 Integer Unit Features

The integer unit supports single cycle execution of most integer instructions:

- 32-bit AU for arithmetic and comparison operations

- 32-bit LU for logical operations
- 32-bit priority encoder for count leading zeros function
- 32-bit single cycle barrel shifter for static shifts and rotates
- 32-bit mask unit for data masking and insertion
- Divider logic for signed and unsigned divides in 6-16 clocks with minimized execution timing
- Pipelined 32x32 hardware multiplier array supports 32x32->32 multiply with 3 clock latency, 1 clock throughput

3.1.3.3 Load/Store Unit Features

The load/store unit supports load, store, and the load multiple / store multiple instructions:

- 32-bit effective address adder for data memory address calculations
- Pipelined operation supports throughput of one load or store operation per cycle
- Dedicated 64-bit interface to memory supports saving and restoring of up to two registers per cycle for load multiple and store multiple word instructions

3.1.3.4 MMU Features

The features of the MMU are as follows:

- Virtual memory support
- 32-bit virtual and physical addresses
- 8-bit process identifier
- 32-entry fully associative TLB
- Support for nine page sizes (4, 16, 64, and 256 Kbytes, 1, 4, 16, 64, and 256 Mbytes)
- Entry flush protection

3.1.3.5 L1 Cache Features

The features of the cache are as follows:

- 32-kilobyte, 8-way set associative unified cache in the MPC5554; 8-kilobyte, 2-way set associative unified cache in the MPC5553.
- Copyback and writethrough support
- 8-entry store buffer
- Push buffer
- Linefill buffer
- 32-bit address bus plus attributes and control
- Separate unidirectional 64-bit read data bus and 64-bit write data bus
- Supports cache line locking
- Supports way allocation

3.1.3.6 BIU Features

The features of the e200z6 BIU are as follows:

- 32-bit address bus plus attributes and control
- Separate unidirectional 64-bit read data bus and 64-bit write data bus
- Overlapped, in-order accesses

3.1.4 Microarchitecture Summary

The e200z6 processor utilizes a seven stage pipeline for instruction execution. The instruction fetch 1, instruction fetch 2, instruction decode/register file read, execute1, execute2/memory access1, execute3/memory access2, and register writeback stages operate in an overlapped fashion, allowing single clock instruction execution for most instructions.

The integer execution unit consists of a 32-bit arithmetic unit (AU), a logic unit (LU), a 32-bit barrel shifter (shifter), a mask-insertion unit (MIU), a condition register manipulation unit (CRU), a count-leading-zeros unit (CLZ), a 32x32 hardware multiplier array, result feed-forward hardware, and support hardware for division.

Most arithmetic and logical operations are executed in a single cycle with the exception of multiply, which is implemented with a pipelined hardware array, and the divide instructions. A count-leading-zeros unit operates in a single clock cycle.

The instruction unit contains a PC incrementer and a dedicated branch address adder to minimize delays during change of flow operations. Sequential prefetching is performed to ensure a supply of instructions into the execution pipeline. Branch target prefetching is performed to accelerate taken branches. Prefetched instructions are placed into an instruction buffer capable of holding 6 sequential instructions.

Branch target addresses are calculated in parallel with branch instruction decode, resulting in execution time of three clocks. Conditional branches which are not taken execute in a single clock. Branches with successful lookahead and target prefetching have an effective execution time of one clock.

Memory load and store operations are provided for byte, halfword, word (32-bit), and double-word data with automatic zero or sign extension of byte and halfword load data as well as optional byte reversal of data. These instructions can be pipelined to allow effective single cycle throughput. Load and store multiple word instructions allow low overhead context save and restore operations. The load/store unit contains a dedicated effective address adder to allow effective address generation to be optimized.

The condition register unit supports the condition register (CR) and condition register operations defined by the PowerPC architecture. The condition register consists of eight 4-bit fields that reflect the results of certain operations, such as move, integer and floating-point compare, arithmetic, and logical instructions, and provide a mechanism for testing and branching.

Vectored and auto-vectored interrupts are supported by the CPU. Vectored interrupt support is provided to allow multiple interrupt sources to have unique interrupt handlers invoked with no software overhead.

The SPE APU supports vector instructions operating on 16- and 32-bit fixed-point data types, as well as 32-bit IEEE-754 single-precision floating-point formats, and supports single-precision floating-point operations in a pipelined fashion. The 64-bit general-purpose register file is used for source and destination operands, and there is a unified storage model for single-precision floating-point data types of 32-bits and the normal integer type. Low latency fixed-point and floating-point add, subtract, multiply, divide, compare, and conversion operations are provided, and most operations can be pipelined.

3.2 Core Registers and Programmer's Model

This section describes the registers implemented in the e200z6 core. It includes an overview of registers defined by the PowerPC Book E architecture, highlighting differences in how these registers are implemented in the e200z6 core, and provides a detailed description of core-specific registers. Full descriptions of the architecture-defined register set are provided in PowerPC Book E architecture.

The PowerPC Book E architecture defines register-to-register operations for all computational instructions. Source data for these instructions are accessed from the on-chip registers or are provided as immediate values embedded in the opcode. The three-register instruction format allows specification of a

target register distinct from the two source registers, thus preserving the original data for use by other instructions. Data is transferred between memory and registers with explicit load and store instructions only.

e200z6 extends the general-purpose registers to 64-bits for supporting SPE APU operations. PowerPC Book E instructions operate on the lower 32 bits of the GPRs only, and the upper 32 bits are unaffected by these instructions. SPE vector instructions operate on the entire 64-bit register. The SPE APU defines load and store instructions for transferring 64-bit values to/from memory.

Figure 3-2 and Figure 3-3 show the complete e200z6 register set. Figure 3-2 shows the registers that are accessible while in supervisor mode, and Figure 3-3 shows the set of registers that are accessible while in user mode. The number to the right of the special-purpose registers (SPRs) is the decimal number used in the instruction syntax to access the register (for example, the integer exception register (XER) is SPR 1).

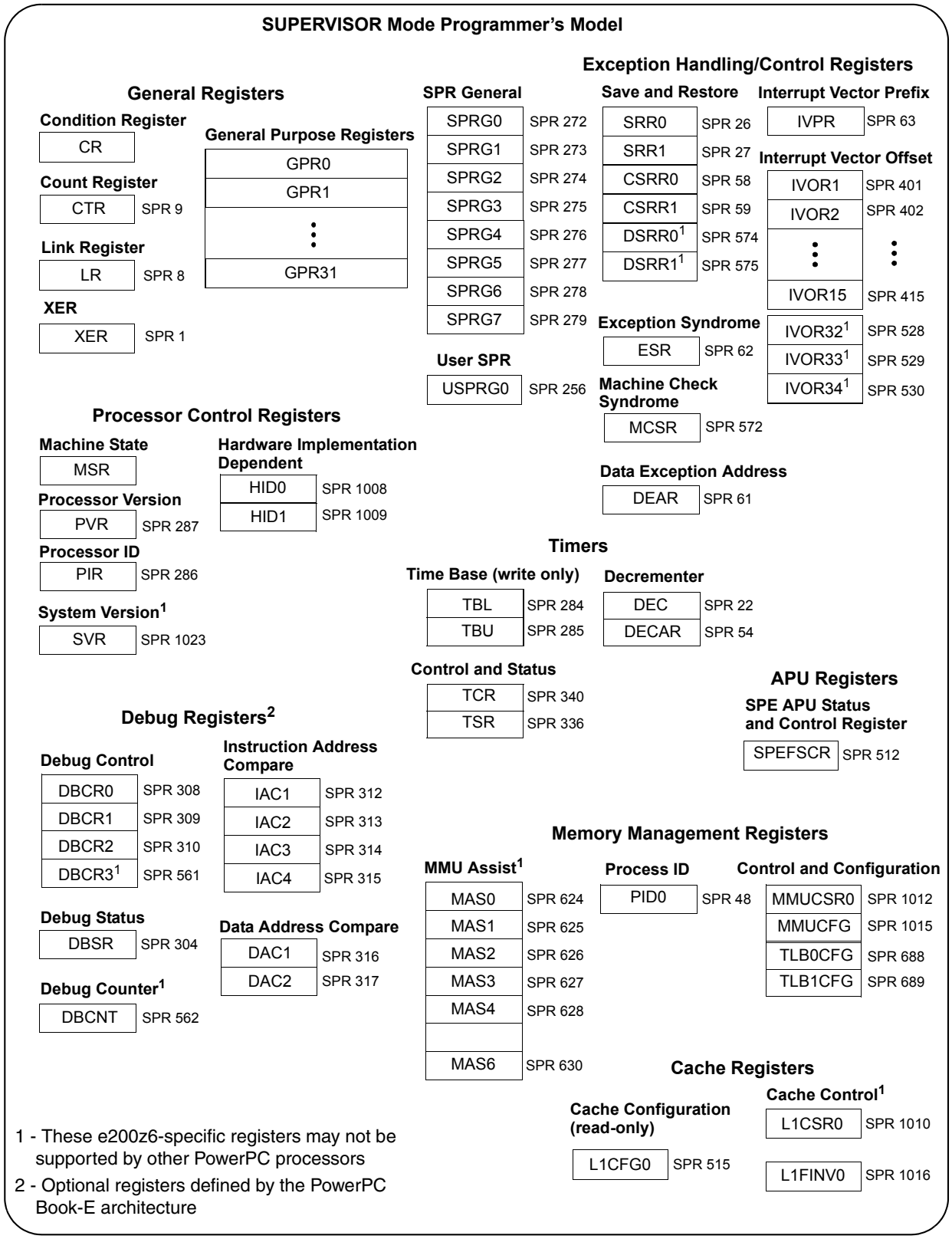


Figure 3-2. Supervisor Mode Programmer's Model

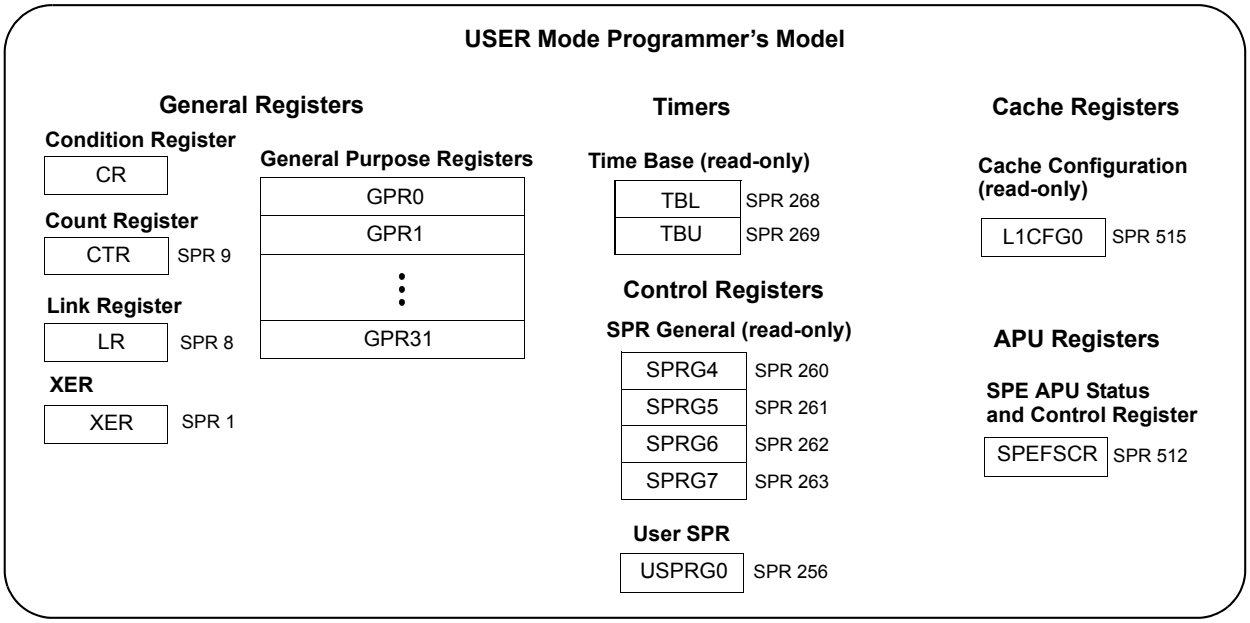


Figure 3-3. User Mode Programmer's Model

3.2.1 PowerPC Book E Registers

e200z6 supports most of the registers defined by the PowerPC Book E architecture. Notable exceptions are the floating point registers FPR0-FPR31 and FPSCR. The e200z6 does not support the Book E floating point architecture in hardware. The supported PowerPC Book E registers are described as follows:

3.2.1.1 User-Level Registers

The user-level registers can be accessed by all software with either user or supervisor privileges. They include the following:

- General-purpose registers (GPRs). The thirty-two 64-bit GPRs (GPR0–GPR31) serve as data source or destination registers for integer and SPE APU instructions and provide data for generating addresses. PowerPC Book E instructions affect only the lower 32 bits of the GPRs. SPE APU instructions are provided which operate on the entire 64-bit register.
- Condition register (CR). The 32-bit CR consists of eight 4-bit fields, CR0–CR7, that reflect results of certain arithmetic operations and provide a mechanism for testing and branching.
  The remaining user-level registers are SPRs. Note that the PowerPC architecture provides the **mtspr** and **mfspr** instructions for accessing SPRs.
- Integer exception register (XER). The XER indicates overflow and carries for integer operations.
- Link register (LR). The LR provides the branch target address for the branch conditional to link register (**bclr**, **bclrl**) instructions, and is used to hold the address of the instruction that follows a branch and link instruction, typically used for linking to subroutines.
- Count register (CTR). The CTR holds a loop count that can be decremented during execution of appropriately coded branch instructions. The CTR also provides the branch target address for the branch conditional to count register (**bcctr**, **bcctrl**) instructions.
- The time base facility (TB) consists of two 32-bit registers: time base upper (TBU) and time base lower (TBL). These two registers are accessible in a read-only fashion to user-level software.

- SPRG–-SPRG7. The PowerPC Book E architecture defines software-use special purpose registers (SPRGs). SPRG4–SPRG7 are accessible in a read-only fashion by user-level software. The e200z6 does not allow user mode access to the SPRG3 register (defined as implementation dependent by Book E).
- USPRG0. The PowerPC Book E architecture defines user software-use special purpose register USPRG0 which is accessible in a read-write fashion by user-level software.

3.2.1.2 Supervisor-Level Only Registers

In addition to the registers accessible in user mode, Supervisor-level software has access to additional control and status registers an operating system used for configuration, exception handling, and other operating system functions. The PowerPC Book E architecture defines the following supervisor-level registers:

- Processor control registers
  - Machine state register (MSR). The MSR defines the state of the processor. The MSR can be modified by the move to machine state register (**mtmsr**), system call (**sc)**, and return from exception (**rfi, rfci, rfdi)** instructions. It can be read by the move from machine state register (**mfmsr)** instruction. When an interrupt occurs, the contents of the MSR are saved to one of the machine state save/restore registers (SRR1, CSRR1, DSRR1).
  - Processor version register (PVR). This register is a read-only register that identifies the version (model) and revision level of the PowerPC processor.
  - Processor identification register (PIR). This read-only register is provided to distinguish the processor from other processors in the system.
- Storage control register
  - Process ID register (PID, also referred to as PID0). This register is provided to indicate the current process or task identifier. It is used by the MMU as an extension to the effective address, and by external Nexus 2/3/4 modules for ownership trace message generation. PowerPC Book E allows for multiple PIDs; e200z6 implements only one.
- Interrupt registers
  - Data exception address register (DEAR). After a data storage interrupt (DSI), alignment interrupt, or data TLB miss Interrupt, the DEAR is set to the effective address (EA) generated by the faulting instruction.
  - Software-use special purpose registers (SPRGs). The SPRG0–SPRG7 registers are provided for operating system use.
  - Exception syndrome register (ESR). The ESR register provides a syndrome to differentiate between the different kinds of exceptions which can generate the same interrupt.
  - Interrupt vector prefix register (IVPR) and the interrupt vector offset registers (IVOR1–IVOR15). These registers together provide the address of the interrupt handler for different classes of interrupts.
  - Save/restore registers (SRR0, SRR1). SRR0 holds the effective address for the instruction at which execution resumes when an **rfi** instruction is executed at the end of a non-critical class interrupt handler routine. SRR1 is used to save machine state on a non-critical interrupt, and stores the MSR register contents. The MSR value is restored when an **rfi** instruction is executed at the end of a non-critical class interrupt handler routine.
  - Critical save/restore registers (CSRR0, CSRR1). CSRR0 holds the effective address for the instruction at which execution resumes when an **rfci** instruction is executed at the end of a critical class interrupt handler routine. CSRR1 is used to save machine state on a critical interrupt, and stores the MSR register contents. The MSR value is restored when an **rfci** instruction is executed at the end of a critical class interrupt routine.

- Debug facility registers
  - Debug control registers (DBCR0–DBCR2). These registers provide control for enabling and configuring debug events.
  - Debug status register (DBSR). This register contains debug event status.
  - Instruction address compare registers (IAC1–IAC4). These registers contain addresses and/or masks which are used to specify instruction address compare debug events.
  - Data address compare registers (DAC1, DAC2). These registers contain addresses and/or masks which are used to specify data address compare debug events.
  - e200z6 does not implement the data value compare registers (DVC1, DVC2).
- Timer registers
  - The clock inputs for the timers are connected to the internal system clock.
  - Time base (TB). The TB is a 64-bit structure provided for maintaining the time of day and operating interval timers. The TB consists of two 32-bit registers, time base upper (TBU) and time base lower (TBL). The time base registers can be written to only by supervisor-level software, but can be read by both user and supervisor-level software.
  - Decrementer register (DEC). This register is a 32-bit decrementing counter that provides a mechanism for causing a decrementer exception after a programmable delay.
  - Decrementer auto-reload (DECAR). This register is provided to support the auto-reload feature of the Decrementer.
  - Timer control register (TCR). This register controls decrementer, fixed-interval timer, and watchdog timer options.
  - Timer status register (TSR). This register contains status on timer events and the most recent watchdog timer-initiated processor reset.

More details about these registers can be found in the PowerPC Book E architecture specifications.

3.2.2 Core-Specific Registers

The PowerPC Book E architecture allows implementation-specific registers. Those incorporated in the e200z6 core are as follows:

3.2.2.1 User-Level Registers

The user-level registers can be accessed by all software with either user or supervisor privileges. They include the following:

- Signal processing extension APU status and control register (SPEFSCR). The SPEFSCR contains all fixed-point and floating-point exception signal bits, exception summary bits, exception enable bits, and rounding control bits needed for compliance with the IEEE 754 standard.
- The L1 cache configuration register (L1CFG0). This read-only register allows software to query the configuration of the L1 Unified cache.

3.2.2.2 Supervisor-Level Registers

The following supervisor-level registers are defined in e200z6 core in addition to the PowerPC Book E registers described above:

- Configuration registers
  - Hardware implementation-dependent register 0 (HID0). This register controls various processor and system functions.

  - Hardware implementation-dependent register 1 (HID1). This register controls various processor and system functions.
- Exception handling and control registers
  - Debug save/restore registers (DSRR0, DSRR1). DSRR0 holds the effective address for the instruction at which execution resumes when an **rfdi** instruction is executed at the end of a debug interrupt handler routine. DSRR1 is used to save machine state on a debug interrupt, and stores the MSR register contents. The MSR value is restored when an **rfdi** instruction is executed at the end of a debug interrupt handler routine.
  - When enabled, the DSRR0 register is used to save the address of the instruction at which execution continues when **rfdi** executes at the end of a debug interrupt handler routine.
  - Interrupt vector offset registers (IVOR32-IVOR34). These registers provide the address of the interrupt handler for different classes of interrupts.
- Debug facility registers
  - Debug control register 3 (DBCR3)—This register provides control for debug functions not described in PowerPC Book E architecture.
  - Debug counter register (DBCNT)—This register provides counter capability for debug functions.
- Cache registers
  - L1 cache configuration register (L1CFG0) is a read-only register that allows software to query the configuration of the L1 Cache.
  - L1 cache control and status register (L1CSR0) control the operation of the L1 unified cache such as cache enabling, cache invalidation, cache locking, etc.
  - L1 cache flush and invalidate register (L1FINV0) controls software flushing and invalidation of the L1 unified cache.
- Memory management unit registers
  - MMU configuration register (MMUCFG) is a read-only register that allows software to query the configuration of the MMU.
  - MMU assist (MAS0-MAS4, MAS6) registers. These registers provide the interface to the core from the memory management unit.
  - MMU control and status register (MMUCSR0) controls invalidation of the MMU.
  - TLB configuration registers (TLBCFG0, TLBCFG1) are read-only registers that allow software to query the configuration of the TLBs.
- System version register (SVR). This register is a read-only register that identifies the version (model) and revision level of the system which includes an e200z6 PowerPC processor.

More details about these registers can be found in the e200z6 core reference.

3.2.3 e200Z6 Core Complex Features Not Supported in the MPC5553/MPC5554

The MPC5553/MPC5554 implements a subset of the e200z6 core complex features. The e200z6 core complex features that are *not* supported in the MPC5553/MPC5554 are described in Table 3-1.

Table 3-1. e200z6 Features Not Supported in the MPC5553/MPC5554 Core

Description	Function / Category
These events are disabled: External Debug Event (DEVT2) Unconditional Debug Event (UDE)	
The e200z6 Core Halted State and Stopped State are not supported.	Power Management
The following low power modes are not supported: Doze mode Nap mode Sleep mode Time base interrupt wake-up from low power mode is not supported.	Power Management
Core wake up is not supported. The MSR[WE] bit in the Machine State Register is not supported. The OCR[WKUP] bit in the Zen OnCE Control Register (OCR) has no effect.	Power Management
The machine check input pin is not supported. HID0 [EMCP] has no effect, and MCSR[MCP] always reads a negated value.	Machine Check
Least significant halfword of Processor Version Register (PVR) is 0x 0000, which contains these three bitfields: MBG Use = 0x00 MBG Rev = 0x0 MBG ID = 0x0 The PVR register has two bitfields in the MPC5553/MPC5554.	PVR Value
Reservation Management logic external to the e200z6 is not implemented.	Reservation Management
The System Version Register (SVR) of the e200z6 is 0x 0000_0000	Verification
The internal Time Base and Decrementer Counters are always enabled in the e200z6	Time Base
Timer External Clock is not supported.	Time Base
The CTXCR and ALTCXTCR registers are not supported.	Context Control

3.3 Functional Description

The following sections describe the function of the various blocks within the e200z6 core.

3.3.1 Memory Management Unit (MMU)

The memory management unit (MMU) is a 32-bit PowerPC Book E compliant implementation with a 32-entry fully associative translation lookaside buffer (TLB). The PowerPC Book E architecture divides the effective and real address space into pages. The page represents the granularity of effective address translation, permission control, and memory/cache attributes. The e200z6 MMU supports the following nine page sizes: 4K, 16K, 64K, 256K, 1M, 4M, 16M, 64M, and 256M.

3.3.1.1 Translation Lookaside Buffer (TLB)

The TLB consists of a 32-entry, fully associative content addressable memory (CAM) array. To perform a lookup, the CAM is searched in parallel for a matching TLB entry. The contents of this TLB entry are then concatenated with the page offset of the original effective address. The result constitutes the real (physical) address of the access. Table 3-2 shows the TLB entry bit definitions.

Table 3-2. TLB Entry Bit Definitions

Field	Comments
V	Valid bit for entry
TS	Translation address space (compared against AS bit)
TID[0:7]	Translation ID (compared against PID0 or ‘0’)
EPN[0:19]	Effective page number (compared against effective address)
RPN[0:19]	Real page number (translated address)
SIZE[0-3]	Page size (4K/16K/64K/256K/1M/4M/16M/64M/256Mbytes)
SX, SW, SR	Supervisor execute, write, and read permission bits
UX, UW, UR	User execute, write, and read permission bits
WIMGE	Translation attributes (Write-through required, cache-Inhibited, Memory coherence required, Guarded, Endian)
U0-U3	User bits -- used only by software
IPROT	Invalidation protect

The TLB is accessed indirectly through several MMU assist (MAS) registers; refer to Section 3.3.1.5, “MMU Assist Registers (MAS0–MAS4, MAS6) and the *e200z6 PowerPC*^{TM} *Core Reference Manual* for more details. Software can write and read the MMU Assist registers with **mtspr** (move to SPR) and **mfspr** (move from SPR) instructions. These registers contain information related to reading and writing a given entry within the TLB. Data is read from the TLB into the MAS registers with a **tlbre** (TLB read entry) instruction. Data is written to the TLB from the MAS registers with a **tlbwe** (TLB write entry) instruction.

3.3.1.2 Translation Flow

The effective address, concatenated with the address space value of the corresponding MSR bit (MSR[IS] or MSR[DS], is compared to the appropriate number of bits of the EPN field and the TS field of TLB entries. If the contents of the effective address plus the address space bit matches the EPN field and TS bit

of the TLB entry, that TLB entry is a candidate for a possible translation match. In addition to a match in the EPN field and TS, a matching TLB entry must match with the current process ID of the access (in PID0), or have a TID value of 0, indicating the entry is globally shared among all processes.

Figure 3-4 shows the translation match logic for the effective address plus its attributes, collectively called the virtual address, and how it is compared with the corresponding fields in the TLB entries.

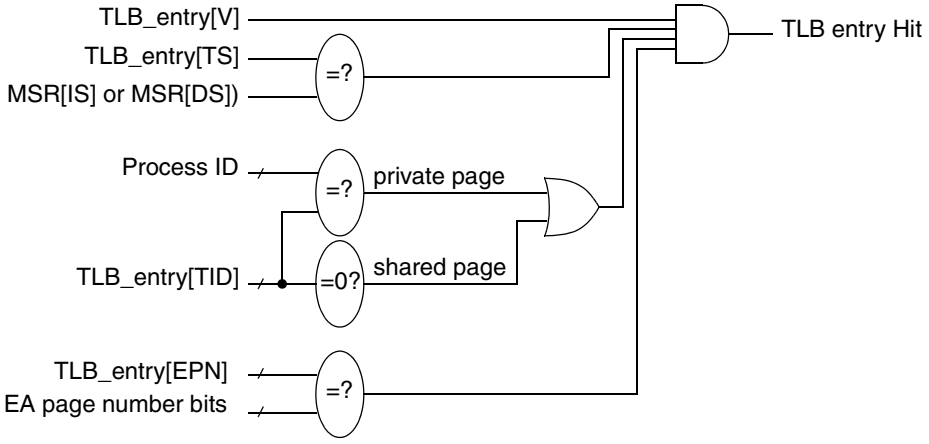


Figure 3-4. Virtual Address and TLB-Entry Compare Process

3.3.1.3 Effective to Real Address Translation

Instruction accesses are generated by sequential instruction fetches or due to a change in program flow (branches and interrupts). Data accesses are generated by load, store, and cache management instructions. The instruction fetch, branch, and load/store units generate 32-bit effective addresses. The MMU translates this effective address to a 32-bit real address which is then used for memory accesses. Figure 3-5 shows the effective to real address translation flow.

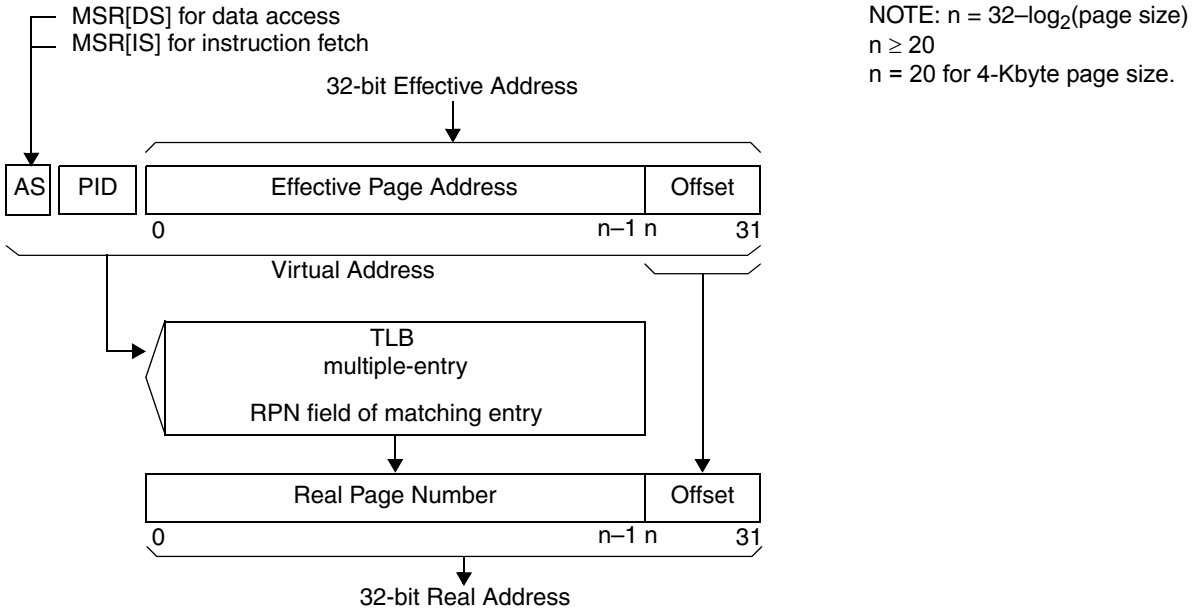


Figure 3-5. Effective to Real Address Translation Flow

3.3.1.4 Permissions

The application software may restrict access to virtual pages by selectively granting permissions for user mode read, write, and execute, and supervisor mode read, write, and execute on a per page basis. For example, program code might be execute-only and data structures may be mapped as read/write/no-execute.

The UX, SX, UW, SW, UR, and SR access control bits are provided to support selective permissions (access control):

- SR—Supervisor read permission. Allows loads and load-type cache management instructions to access the page while in supervisor mode.
- SW—Supervisor write permission. Allows stores and store-type cache management instructions to access the page while in supervisor mode.
- SX—Supervisor execute permission. Allows instruction fetches to access the page and instructions to be executed from the page while in supervisor mode.
- UR—User read permission. Allows loads and load-type cache management instructions to access the page while in user mode.
- UW—User write permission. Allows stores and store-type cache management instructions to access the page while in user mode.
- UX—User execute permission. Allows instruction fetches to access the page and instructions to be executed from the page while in user mode.

If the translation match was successful, the permission bits are checked as shown in Figure 3-6. If the access is not allowed by the access permission mechanism, the processor generates an instruction or data storage interrupt (ISI or DSI).

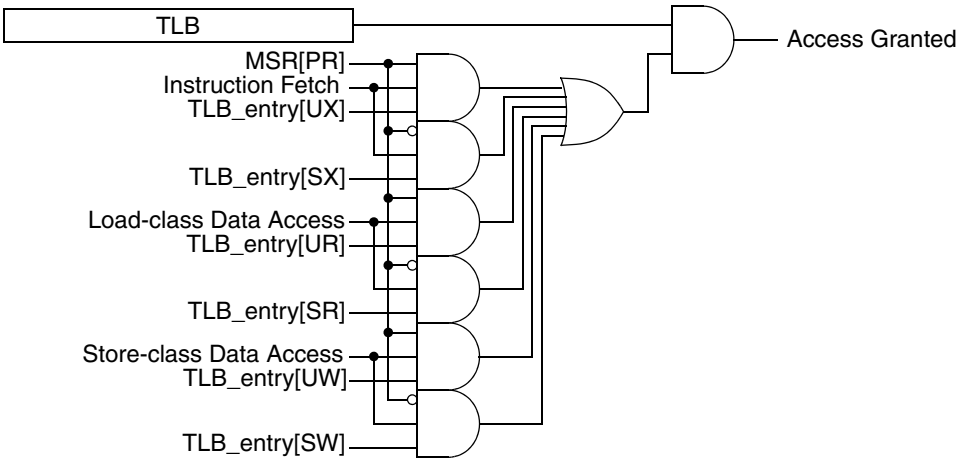


Figure 3-6. Granting of Access Permission

3.3.1.5 MMU Assist Registers (MAS0–MAS4, MAS6)

The e200z6 uses six special purpose registers (MAS0, MAS1, MAS2, MAS3, MAS4 and MAS6) to facilitate reading, writing, and searching the TLBs. The MAS registers can be read or written using the **mfspr** and **mtspr** instructions. The e200z6 does not implement the MAS5 register, present in other Freescale EIS designs, because the **tlbsx** instruction only searches based on a single SPID value.

Additional information on the MAS*n* registers is available in the *e200z6 PowerPC*^{TM} *Core Reference Manual*. The MAS0 register is shown in Figure 3-7.

	0 1	2 3	4 10	11 15	16 26	27 31
Field	—	TLBSEL	—	ESEL	—	NV
Reset	Undefined on *m_por* assertion, unchanged on *p_reset_b* assertion					
R/W	R/W					
SPR	SPR 624					

Figure 3-7. MAS Register 0 (MAS0) Format

MAS0 fields are defined in Table 3-3.

Table 3-3. MAS0—MMU Read/Write and Replacement Control

Bits	Name	Description
0–1	—	Reserved, should be cleared.
2–3	TLBSEL	Selects TLB for access 01 TLB1 (ignored by the e200z6, should be written to 01 for future compatibility)
4–10	—	Reserved, should be cleared.
11–15	ESEL	Entry select for TLB1
16–26	—	Reserved, should be cleared.
27–31	NV	Next replacement victim for TLB1 (software managed). Software updates this field; it is copied to the ESEL field on a TLB error.

The MAS1 register is shown in Figure 3-8.

	0	1	2 7	8 15	16 18	19	20 23	24 31
Field	VALID	IPROT	—	TID	—	TS	TSIZE	—
Reset	Undefined on *m_por* assertion, unchanged on *p_reset_b* assertion							
R/W	R/W							
SPR	SPR 625							

Figure 3-8. MMU Assist Register 1 (MAS1)

MAS1 fields are defined in Table 3-4.

Table 3-4. MAS1 —Descriptor Context and Configuration Control

Bits	Name	Description
0	VALID	TLB entry valid 0 This TLB entry is invalid. 1 This TLB entry is valid.
1	IPROT	Invalidation protect 0 Entry is not protected from invalidation. 1 Entry is protected from invalidation. Protects TLB entry from invalidation by **tlbivax** (TLB1 only), or flash invalidates through MMUCSR0[TLB1_FI].
2–7	—	Reserved, should be cleared.
8–15	TID	Translation ID bits This field is compared with the current process IDs of the effective address to be translated. A TID value of 0 defines an entry as global and matches with all process IDs.
16–18	—	Reserved, should be cleared.
19	TS	Translation address space This bit is compared with the IS or DS fields of the MSR (depending on the type of access) to determine if this TLB entry may be used for translation.
20–23	TSIZE	Entry page size Supported page sizes are: 0b00014 Kbytes 0b01104 Mbytes 0b001016 Kbytes 0b011116 Mbytes 0b001164 Kbytes 0b100064 Mbytes 0b0100256 Kbytes 0b1001256 Mbytes 0b01011 Mbyte All other values are undefined.
24–31	—	Reserved, should be cleared.

The MAS2 register is shown in Figure 3-9.

	0 — 19	20 — 26	27	28	29	30	31
Field	EPN	—	W	I	M	G	E
Reset	Undefined on *m_por* assertion, unchanged on *p_reset_b* assertion						
R/W	R/W						
SPR	SPR 626						

Figure 3-9. MMU Assist Register 2 (MAS2)

MAS2 fields are defined in Table 3-5.

Table 3-5. MAS2—EPN and Page Attributes

Bits	Name	Description
0–19	EPN	Effective page number
20–26	—	Reserved, should be cleared.

Table 3-5. MAS2—EPN and Page Attributes (continued)

Bits	Name	Description
27	W	Write-through required 0 This page is considered write-back with respect to the caches in the system. 1 All stores performed to this page are written through to main memory.
28	I	Cache inhibited 0 This page is considered cacheable. 1 This page is considered cache-inhibited.
29	M	Memory coherence required.The e200z6 does not support the memory coherence required attribute, and thus it is ignored. 0 Memory coherence is not required. 1 Memory coherence is required.
30	G	Guarded. The e200z6ignores the guarded attribute (other than for generation of the *p_hprot[4:2]* attributes on an external access), since no speculative or out-of-order processing is performed. 0 Access to this page are not guarded, and can be performed before it is known if they are required by the sequential execution model. 1 All loads and stores to this page are performed without speculation (that is, they are known to be required).
31	E	Endianness. Determines endianness for the corresponding page. 0 The page is accessed in big-endian byte order. 1 The page is accessed in true little-endian byte order.

The MAS3 register is shown in Figure 3-10.

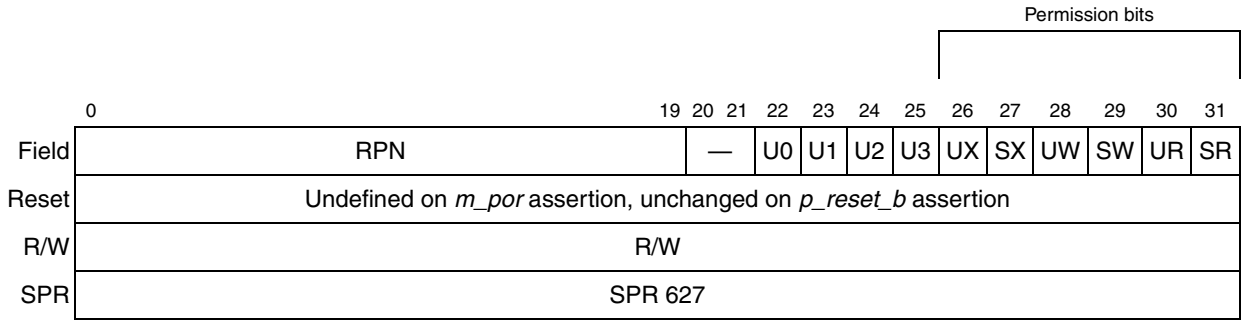


Figure 3-10. MMU Assist Register 3 (MAS3)

MAS3 fields are defined in Table 3-6

Table 3-6. MAS3—RPN and Access Control

Bits	Name	Description
0–19	RPN	Real page number Only bits that correspond to a page number are valid. Bits that represent offsets within a page are ignored and should be zero.
20–21	—	Reserved, should be cleared.
22–25	U0–U3	User bits
26–31	PERMIS	Permission bits (UX, SX, UW, SW, UR, SR)

The MAS4 register is shown in Figure 3-11.

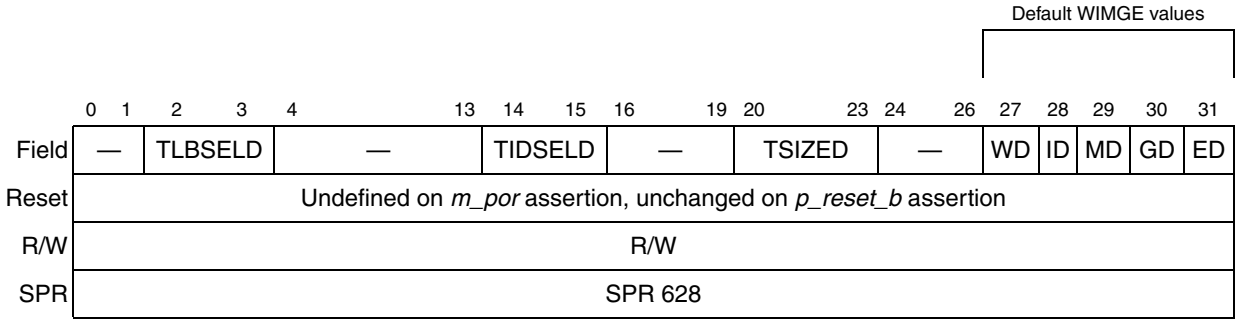


Figure 3-11. MMU Assist Register 4 (MAS4)

MAS4 fields are defined in Table 3-7.

Table 3-7. MAS4—Hardware Replacement Assist Configuration Register

Bits	Name	Description
0–1	—	Reserved, should be cleared.
2–3	TLBSELD	Default TLB selected 01 TLB1 (ignored by the e200z6, should be written to 01 for future compatibility)
4–13	—	Reserved, should be cleared.
14–15	TIDSELD	Default PID# to load TID from 00 PID0 01 Reserved, do not use 10 Reserved, do not use 11 TIDZ (0x00)) (Use all zeros, the globally shared value)
16–19	—	Reserved, should be cleared.
17–23	TSIZED	Default TSIZE value
24–26	—	Reserved, should be cleared.
27–31	DWIMGE	Default WIMGE values

The MAS6 register is shown in Figure 3-12.

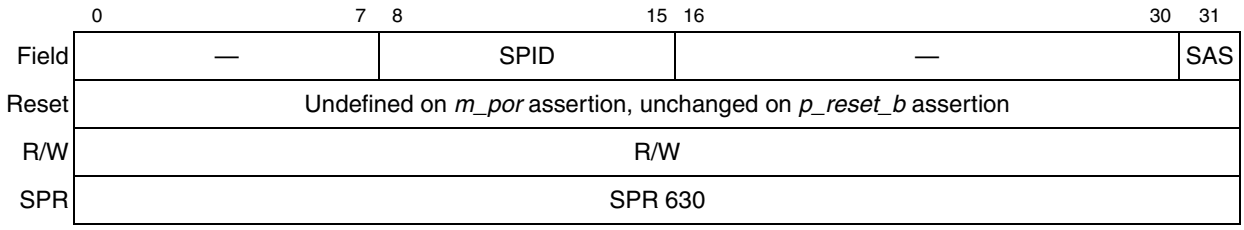


Figure 3-12. MMU Assist Register 6 (MAS6))

MAS6 fields are defined in Table 3-8.

Table 3-8. MAS6—TLB Search Context Register 0

Bits	Name	Description
0–7	—	Reserved, should be cleared.
8–15	SPID	PID value for searches
16–30	—	Reserved, should be cleared.
31	SAS	AS value for searches

3.3.2 L1 Cache

The e200z6 processor supports a 32-kilobyte, 8-way set-associative, unified (instruction and data) cache with a 32-byte line size in the MPC5554; the MPC5553 provides an 8-Kbyte, 2-way set associative unified cache with a 32-byte line size. The cache improves system performance by providing low-latency data to the e200z6 instruction and data pipelines, which decouples processor performance from system memory performance. The cache is virtually indexed and physically tagged. The e200z6 does not provide hardware support for cache coherency in a multi-master environment. Software must be used to maintain cache coherency with other possible bus masters.

Both instruction and data accesses are performed using a single bus connected to the cache. Addresses from the processor to the cache are virtual addresses used to index the cache array. The MMU provides the virtual to physical translation for use in performing the cache tag compare. If the physical address matches a valid cache tag entry, the access hits in the cache. For a read operation, the cache supplies the data to the processor, and for a write operation, the data from the processor updates the cache. If the access does not match a valid cache tag entry (misses in the cache) or a write access must be written through to memory, the cache performs a bus cycle on the system bus. Figure 3-13 shows a block diagram of the unified cache in the e200z6.

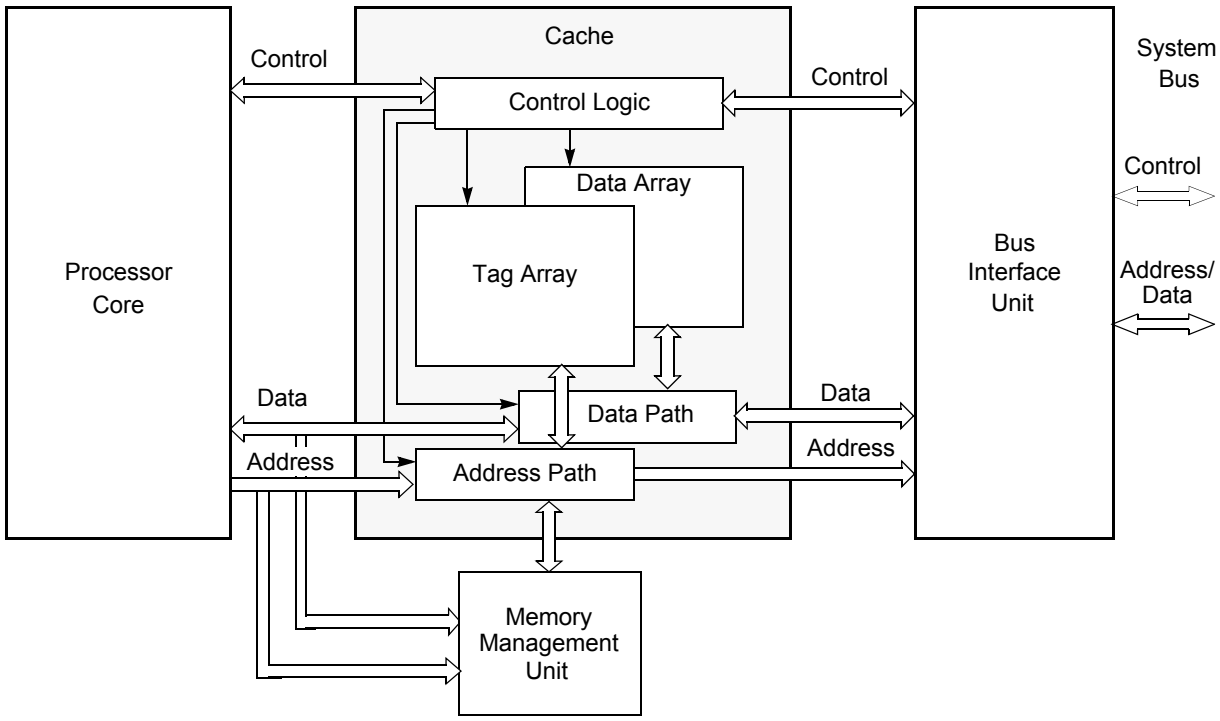


Figure 3-13. e200z6 Unified Cache Block Diagram

3.3.2.1 Cache Organization

The e200z6 cache is organized as eight (MPC5554)/two (MPC5553) ways of 128 sets with each line containing 32 bytes (four double-words) plus parity of storage. Figure 3-14 illustrates the cache organization, terminology used, the cache line format and cache tag formats.

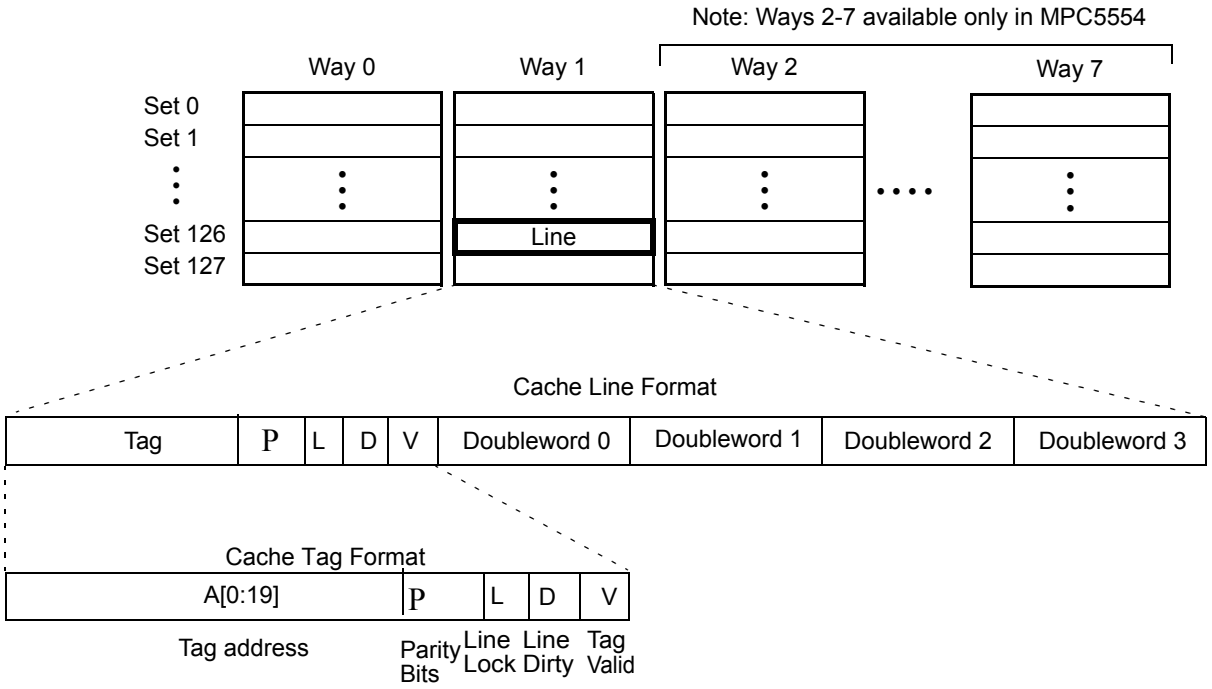


Figure 3-14. Cache Organization and Line Format

3.3.2.2 Cache Lookup

Once enabled, the unified cache will be searched for a tag match on all instruction fetches and data accesses from the CPU. If a match is found, the cached data is forwarded on a read access to the instruction fetch unit or the load/store unit (data access), or is updated on a write access, and may also be written-through to memory if required.

When a read miss occurs, if there is a TLB hit and the cache inhibit bit (WIMGE=0bx0xxx) of the hitting TLB entry is clear, the translated physical address is used to fetch a four double-word cache line beginning with the requested double-word (critical double-word first). The line is fetched and placed into the appropriate cache block and the critical double-word is forwarded to the CPU. Subsequent double-words may be streamed to the CPU if they have been requested and streaming is enabled via the DSTRM bit in the L1CSR0 register.

During a cache line fill, double-words received from the bus are placed into a cache linefill buffer, and may be forwarded (streamed) to the CPU if such a request is pending. Accesses from the CPU following delivery of the critical double-word may be satisfied from the cache (hit under fill, non-blocking) or from the linefill buffer if the requested information has been already received.

The cache always fills an entire line, thereby providing validity on a line-by-line basis. A cache line is always in one of the following states: invalid, valid, or dirty (and valid). For invalid lines, the V bit is clear, causing the cache line to be ignored during lookups. Valid lines have their V bit set and D bit cleared, indicating the line contains valid data consistent with memory. Dirty cache lines have the D and V bits set, indicating that the line has valid entries that have not been written to memory. In addition, a cache line may be locked (L bit set) indicating the line is not available for replacement.

The cache should be explicitly invalidated after a hardware reset; reset does not invalidate the cache lines. Following initial power-up, the cache contents will be undefined. The L, D and V bits may be set on some lines, necessitating the invalidation of the cache by software before being enabled.

Figure 3-15 illustrates the general flow of cache operation.

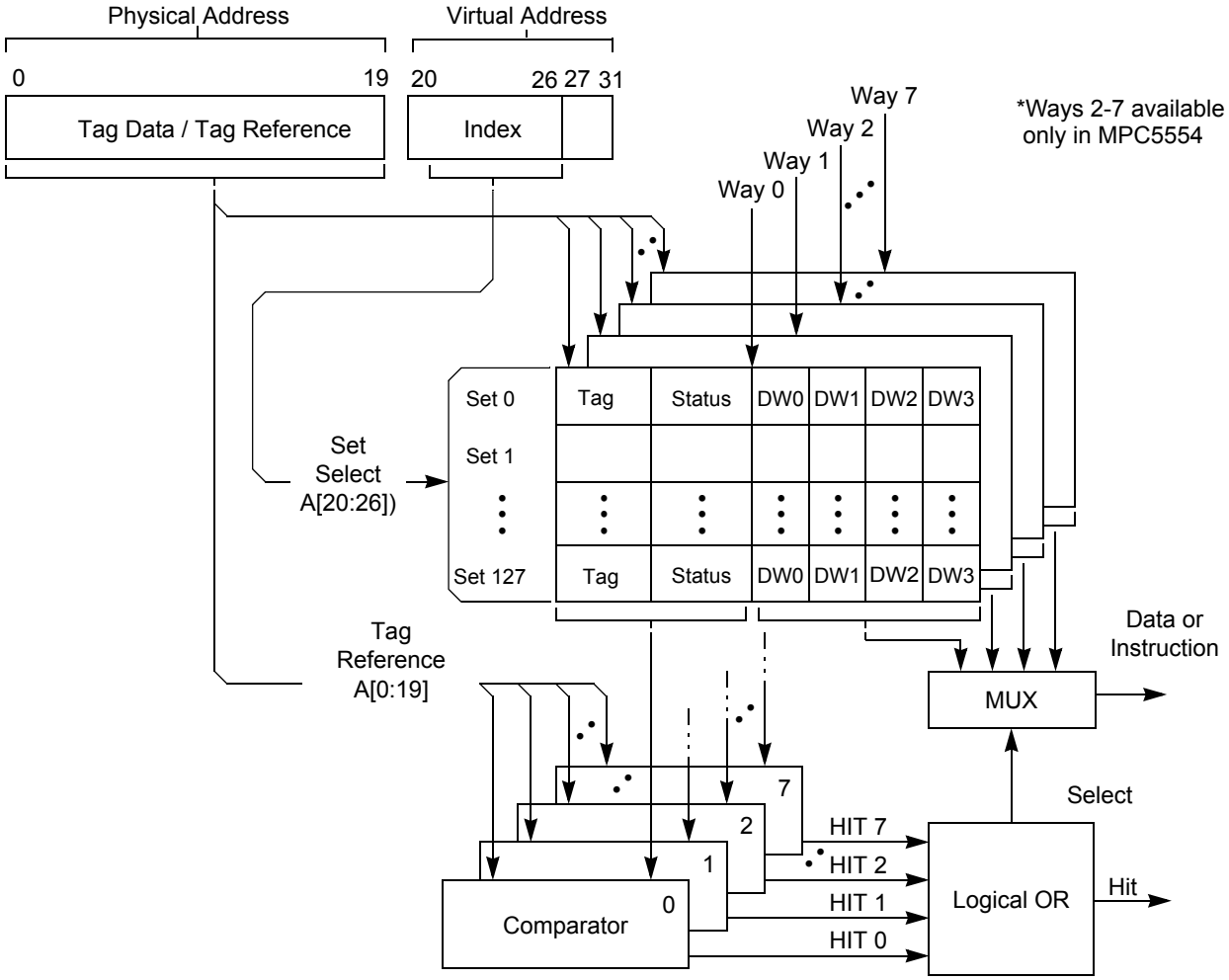


Figure 3-15. Cache Lookup Flow

To determine if the address is already allocated in the cache the following steps are taken:

1. The cache set index, virtual address bits A[20:26] are used to select one cache set. A set is defined as the grouping of eight lines (one from each way), corresponding to the same index into the cache array.
2. The higher order physical address bits A[0:19] are used as a tag reference or used to update the cache line tag field.
3. The eight tags from the selected cache set are compared with the tag reference. If any one of the tags matches the tag reference and the tag status is valid, a cache hit has occurred.
4. Virtual address bits A[27:28] are used to select one of the four double-words in each line. A cache hit indicates that the selected double-word in that cache line contain valid data (for a read access), or can be written with new data depending on the status of the W access control bit from the MMU (for a write access).

3.3.2.3 Cache Line Replacement Algorithm

On a cache read miss, the cache controller uses a pseudo-round-robin replacement algorithm to determine which cache line will be selected to be replaced. There is a single replacement counter for the entire cache. The replacement algorithm acts as follows: On a miss, if the replacement pointer is pointing to a way which is not enabled for replacement by the type of the miss access (the selected line or way is locked), it is incremented until an available way is selected (if any). After a cache line is successfully filled without error, the replacement pointer increments to point to the next cache way.

3.3.3 Interrupt Types

The interrupts implemented in the MPC5553/MPC5554 and the exception conditions that cause them are listed in Table 3-9.

Table 3-9. Interrupts and Conditions

Interrupt Type	Interrupt Vector Offset Register	Enables[1]	State Saved In	Causing Conditions
System reset	none, vector to 0xFFFF_FFFC			• Reset by assertion of $\overline{\text{RESET}}$ • Watchdog timer reset control. • Debug Reset Control.
Critical input	IVOR0	IVOR0 is not supported in the MPC5553/MPC5554		
Machine check	IVOR 1	ME	CSSR[0:1]	• Machine check exception and MSR[ME] =1. • ISI, ITLB Error on first instruction fetch for an exception handler • Parity error signaled on cache access • Write bus error on buffered store or cache line push
Data Storage	IVOR 2	—	SRR[0:1]	• Access control. • Byte ordering due to misaligned access across page boundary to pages with mismatched E bits. • Cache locking exception • Precise external termination error
Instruction Storage	IVOR 3	—	SRR[0:1]	• Access control. • Precise external termination error
External Input	IVOR 4[2]	EE, src	SRR[0:1]	External interrupt is asserted and MSR[EE]=1.
Alignment	IVOR 5	—	SRR[0:1]	• **lmw**, **stmw** not word aligned. • **lwarx** or **stwcx.** not word aligned. • **dcbz** with disabled cache or no cache present, or to W or I storage. • SPE ld and st instructions not properly aligned
Program	IVOR 6	—	SRR[0:1]	Illegal, Privileged, Trap, FP enabled, AP enabled, Unimplemented Operation.
Floating-point unavailable	IVOR 7	—	SRR[0:1]	MSR[FP]=0 and attempt to execute a Book E floating point operation.
System call	IVOR 8	—	SRR[0:1]	Execution of the System Call (**sc**) instruction
AP unavailable	IVOR 9	—	SRR[0:1]	Unused by e200z6

Table 3-9. Interrupts and Conditions (continued)

Interrupt Type	Interrupt Vector Offset Register	Enables[1]	State Saved In	Causing Conditions
Decrementer	IVOR 10	EE, DIE	SRR[0:1]	Decrementer timeout, and as specified in *Book E: Enhanced PowerPC™ Architecture, Rev 1.0*, Ch. 8, pg. 194-195 and in the *e200Z6 PowerPC™ Core Reference Manual,* Rev 0.
Fixed Interval Timer	IVOR 11	EE, FIE	SRR[0:1]	Fixed-interval timer timeout and as specified in *Book E: Enhanced PowerPC™ Architecture, Rev 1.0*, Ch. 8, pg. 195-196 and in the *e200Z6 PowerPC™ Core Reference Manual,* Rev 0.
Watchdog Timer	IVOR 12	CE, WIE	CSRR[0:1]	Watchdog timeout: as specified in *Book E: Enhanced PowerPC™ Architecture, Rev 1.0*, Ch. 8, pg. 196-197 and in the *e200Z6 PowerPC™ Core Reference Manual,* Rev 0.
Data TLB Error	IVOR 13	—	SRR[0:1]	Data translation lookup did not match a valid entry in the TLB
Instruction TLB Error	IVOR 14	—	SRR[0:1]	Instruction translation lookup did not match a valid entry in the TLB
Debug	IVOR 15	DE, IDM	CSSR[O:1]	Debugger when HIDO[DAPUEN] = 0. Caused by Trap, Instruction Address Compare, Data Address Compare, Instruction Complete, Branch Taken, Return from Interrupt, Interrupt Taken, Debug Counter, External Debug Event, Unconditional Debug Event
		DE, IDM	DSRR[0:1]	Debugger when HIDO[DAPUEN] = 1, and caused by same conditions as above.
Reserved	IVOR 16-31			
SPE Unavailable Exception	IVOR 32	—	SRR[0:1]	SPE APU instruction when MSR[SPE] = 0, and see Section 5.6.18 “SPE APU Unavailable Interrupt” in the *e200Z6 PowerPC™ Core Reference Manual,* Rev 0.
SPE Data Exception	IVOR 33	—	SRR[0:1]	SPE FP data exception and see Section 5.6.19 “SPE Floating-Point Data Interrupt” in the *e200Z6 PowerPC™ Core Reference Manual,* Rev 0.
SPE Round Exception	IVOR 34	—	SRR[0:1]	Inexact result from floating-point instruction. See Section 5.6.20 “SPE Floating-Point Round Interrupt” in the *e200Z6 PowerPC™ Core Reference Manual,* Rev 0.

[1] CE, ME, EE, DE are in the MSR. DIE, FIE, and WIE are in the TCR. “src” signifies the individual enable for each INTC source. The debug interrupt, IVOR 15, also requires EDM = 0 (EDM and IDM are in the DBCR0 register).

3.3.4 Bus Interface Unit (BIU)

The BIU encompasses control and data signals supporting instruction and data transfers. A data bus width of 64-bits is implemented. The memory interface supports read and write transfers of 8, 16, 24, 32, and 64 bits, supports burst transfers of four double-words, and operates in a pipelined fashion.

Single-beat transfers are supported for cache-inhibited read and write cycles, and write-buffer writes. Burst transfers of four double-words are supported for cache linefill and copyback operations.

3.3.5 Timer Facilities

The core provides a set of registers to provide fixed interval timing and watchdog functions for the system. All of these must be initialized during start-up. The registers associated with fixed interval timer and watchdog functions are the following:

- Timer control register (TCR) – provides control of the timer and watchdog facilities.
- Timer status register (TSR) – provides status of the timer facilities.
- Time base registers (TBU and TBL) – Two 32-bit registers (upper and lower) that are concatenated to provide a long-period, 64-bit counter.
- Decrementer register (DEC) – a decrementing counter that is updated at the same rate as the time base. The DEC provides a means of signaling an exception after a specified amount of time. The DEC is typically used as a general-purpose software timer. Note that the decrementer always runs when the system is clocked, and may be written to by software at any time.
- Decrementer auto reload register (DECAR) – provides a value that is automatically reloaded (if enabled) into the decrementer register when the decrementer reaches 0.

For more information on the fixed-interval timer, watchdog timer, and timer and counter registers, refer to the *e200z6 PowerPC*™ *Core Reference Manual* and *EREF: A Reference for Freescale Book E and the e500 core.*

3.3.6 Signal Processing Extension APU (SPE APU)

3.3.6.1 Overview

The PowerPC 32-bit Book E instructions operate on the lower (least significant) 32 bits of the 64-bit GPRs. New SPE instructions are defined that view the 64-bit register as being composed of a vector of two 32-bit elements, and some of the instructions also read or write 16-bit elements. These new instructions can also be used to perform scalar operations by ignoring the results of the upper 32-bit half of the register file. Some instructions are defined that produce a 64-bit scalar result. Vector fixed-point instructions operate on a vector of two 32-bit or four 16-bit fixed-point numbers resident in the 64-bit GPRs. Vector floating-point instructions operate on a vector of two 32-bit single-precision floating-point numbers resident in the 64-bit GPRs. Scalar floating-point instructions operate on the lower half of GPRs. These single-precision floating-point instructions do not have a separate register file; there is a single shared register file for all instructions. Figure 3-16 shows two different representations of the 64-bit GPRs. The shaded half is the only region operated on by the 32-bit PowerPC instructions.

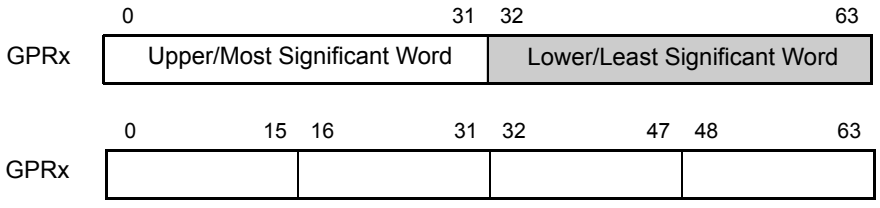


Figure 3-16. 64-bit General-Purpose Registers

3.3.7 SPE Programming Model

Not all SPE instructions record events such as overflow, saturation, and negative/positive result. See the description of the individual SPE instruction in the e200z6 core reference for information on which

conditions are recorded and where they are recorded. Most SPE instructions record conditions to the SPEFSCR. Vector compare instructions store the result of the comparison into the condition register (CR).

The e200z6 core has a 64-bit architectural accumulator register that holds the results of the SPE multiply accumulate (MAC) fixed-point instructions. The accumulator allows back-to-back execution of dependent fixed-point MAC instructions, something that is found in the inner loops of DSP code such as filters. The accumulator is partially visible to the programmer in that its results do not have to be explicitly read to use them. Instead, they are always copied into a 64-bit destination GPR specified as part of the instruction. The accumulator however, has to be explicitly cleared when starting a new MAC loop. Based upon the type of instruction, the accumulator can hold either a single 64-bit value or a vector of two 32-bit elements.

3.4 External References

In addition to the PowerPC Book E instructions, the MPC5554 supports e200z6 core specific instructions and SPE APU instructions. For further information see the following documents:

- *PowerPC Microprocessor Family: The Programming Environment for 32-bit Microprocessors*
- *Book E: Enhanced PowerPC™ Architecture*
- *EREF: A Reference for Freescale Book E and the e500 core*

3.5 Revision History

Substantive Changes since Rev 3.0
No changes.

Chapter 4
Reset

4.1 Introduction

The following reset sources are supported in the MPC5553/MPC5554 MCU:

- Power-on reset
- External reset
- Loss-of-lock reset
- Loss-of-clock reset
- Watchdog timer/debug reset
- JTAG reset
- Checkstop reset
- Software system reset
- Software external reset

All reset sources are processed by the reset controller, which is located in the SIU module. The reset controller monitors the reset input sources, and upon detection of a reset event, resets internal logic and controls the assertion of the RSTOUT pin. The RSTOUT signal may be automatically asserted by writing the SER bit in the SIU_SRCR to 1. The RSTOUT signal will stay asserted for a number of system clocks[1] determined by the configuration of the PLL (See Section 4.2.2, "Reset Output (RSTOUT)"). This does not reset the MPC5553/MPC5554 MCU. All other reset sources initiate an internal reset of the MCU.

For all reset sources, the BOOTCFG[0:1] and PLLCFG[0:1] signals can be used to determine the boot mode and the configuration of the FMPLL, respectively. If the RSTCFG pin is asserted during reset, the values on the BOOTCFG[0:1] pins are latched in the SIU_RSR 4 clock cycles prior to the negation of the RSTOUT pin, determining the boot mode. The values on the PLLCFG[0:1] pins are latched at the negation of the RSTOUT pin, determining the configuration of the FMPLL. If the RSTCFG pin is negated during reset, the FMPLL defaults to normal operation (PLL enabled) with a crystal reference and the boot mode (latched in the SIU_RSR) is defaulted to internal boot from Flash.

The reset status register (SIU_RSR) gives the source of the last reset and indicates whether a glitch has occurred on the RESET pin. The SIU_RSR is updated for all reset sources.

All reset sources initiate execution of the MPC5553/MPC5554 boot assist module (BAM) program with the exception of the software external reset.

The reset configuration half word (RCHW) provides several basic functions at reset. It provides a means to locate the boot code, determines if Flash memory is programmed or erased, enables or disables the watchdog timer, and if booting externally, sets the bus size. The location of the RCHW is specified by the state of the BOOTCFG[0:1] pins. These pins determine whether the RCHW is located in internal Flash, located in external memory, or whether a serial or CAN boot is configured. A complete description of the BOOTCFG[0:1] pins may be found in Chapter 2, "Signal Description." The BAM program reads the values of the BOOTCFG[0:1] pins from the BOOTCFG field of the SIU_RSR, then reads the RCHW from the specified location and uses the RCHW value to determine and execute the specified boot procedure. See Section 4.4.3, "Reset Configuration and Configuration Pins," for a complete description.

1. Unless noted otherwise, the use of 'clock' or 'clocks' in this section is a reference to the system clock.

4.2 External Signal Description

4.2.1 Reset Input (RESET)

The RESET pin is an active low input that is asserted by an external device during a power-on or external reset. The internal reset signal asserts only if the RESET pin is asserted for 10 clock cycles. Assertion of the RESET pin while the device is in reset causes the reset cycle to start over. The RESET pin also has an associated glitch detector which detects spikes greater than 2 clocks in duration that fall below the switch point of the input buffer logic.

4.2.2 Reset Output (RSTOUT)

The RSTOUT pin is an active low output that uses a push/pull configuration. The RSTOUT pin is driven to the low state by the MCU for all internal and external reset sources.

After the negation of the RESET input, if the PLL is configured for 1:1 (dual controller) mode or bypass mode, the RSTOUT signal is asserted for 16000 clocks, plus 4 clocks for sampling of the configuration pins. If the PLL is configured for any other operating mode, the RSTOUT signal is asserted for 2400 clocks, plus 4 clocks for sampling of the configuration pins. See Section 11.1.4, “FMPLL Modes of Operation” for details of PLL configuration.

The RSTOUT pin can also be asserted by a write to the SER bit of the system reset control register (SIU_SRCR).

NOTE

During a power on reset, RSTOUT is three-stated.

4.2.3 Reset Configuration (RSTCFG)

The RSTCFG input is used to enable the BOOTCFG[0:1] and PLLCFG[0:1] pins during reset. If RSTCFG is negated during reset, the BOOTCFG and PLLCFG pins are not sampled at the negation of RSTOUT. In that case, the default values for BOOTCFG and PLLCFG are used. If RSTCFG is asserted during reset, the values on the BOOTCFG and PLLCFG pins are sampled and configure the boot and FMPLL modes.

4.2.4 Weak Pull Configuration (WKPCFG)

WKPCFG determines whether specified eTPU and EMIOS pins are connected to a weak pull up or weak pull down during and immediately after reset.

4.2.5 Boot Configuration (BOOTCFG[0:1])

In the MPC5554, BOOTCFG determines the function and state of the following pins after execution of the BAM reset: CS[0:3], ADDR[12:31], DATA[0:31], TSIZ[0:1], RD_WR, BDIP, WE[0:3], OE, TS, TA, TEA, BR, BG, BB.

In the MPC5553, BOOTCFG determines the function and state of the following pins after a BAM reset: CS[0:3], ADDR[8:31], DATA[0:31], RD_WR, BDIP, WE[0:3], OE, TS, TA, TEA, BR, BG, TSIZ[0:1].

Note that BOOTCFG0 does not function in the 208 pin package of the MPC5553.

4.3 Memory Map/Register Definition

Table 4-1 summarizes the reset controller registers. The base address of the system integration unit is 0xC3F9_0000.

Table 4-1. Reset Controller Memory Map

Address	Register Name	Register Description	Size (bits)
Base (0xC3F9_000C) + 0xC	SIU_RSR	Reset status register	32
Base (0xC3F9_000C) + 0x10	SIU_SRCR	System reset control register	32

4.3.1 Register Descriptions

This section describes all the reset controller registers. It includes details about the fields in each register, the number of bits per field, the reset value of the register, and the function of the register.

4.3.1.1 Reset Status Register (SIU_RSR)

The reset status register (SIU_RSR) reflects the most recent source, or sources, of reset. This register contains one bit for each reset source. A bit set to logic 1 indicates the type of reset that occurred. Simultaneous reset requests cause more than one bit to be set at the same time. Once set, the reset source bits in the SIU_RSR remain set until another reset occurs. A software external reset causes the SERF bit to be set, but no previously set bits in the SIU_RSR will be cleared. Additional information about the SIU_RSR may be found in Section 6.3.1.2, "Reset Status Register (SIU_RSR)."

The SIU_RSR also contains the values latched at the last reset on the WKPCFG and BOOTCFG[0:1] pins and a RESET input pin glitch flag. The reset glitch flag bit (RGF) is cleared by writing a 1 to the bit. A write of 0 has no effect on the bit state. The SIU_RSR can be read at all times.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PORS	ERS	LLRS	LCRS	WDRS	CRS	0	0	0	0	0	0	0	0	SSRS	SERF
W																
Reset[1]	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0xC															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WKP CFG	0	0	0	0	0	0	0	0	0	0	0	0	BOOTCFG		RGF
W																
Reset	—[2]	0	0	0	0	0	0	0	0	0	0	0	0	—[3]		0
Reg Addr	Base + 0xC															

1 The RESET values for this register are defined for power-on reset only.

2 The RESET value of this bit or field is determined by the value latched on the associated pin or pins at the negation of the last reset.

3 The RESET value of this bit or field is determined by the value latched on the associated pin or pins at the negation of the last reset. BOOTCFG can also be loaded with a default instead of what is on the associated pin or pins.

Figure 4-1. Reset Status Register (SIU_RSR)

Table 4-2. SIU_RSR Field Descriptions

Bits	Name	Description
0	PORS	Power-on reset status 0 No power-on reset has occurred. 1 A power-on reset has occurred.
1	ERS	External reset status 0 No external reset has occurred. 1 An external reset has occurred. The ERS bit is also set during a POR event.
2	LLRS	Loss-of-lock reset status 0 No loss-of-lock reset has occurred. 1 A loss-of-lock reset has occurred.
3	LCRS	Loss-of-clock reset status 0 No loss-of-clock reset has occurred. 1 A loss-of-clock reset has occurred due to a loss of the reference or failure of the FMPLL.
4	WDRS	Watchdog timer/debug reset status 0 No watchdog timer or debug reset has occurred. 1 A watchdog timer or debug reset has occurred.
5	CRS	Checkstop reset status 0 No enabled checkstop reset has occurred. 1 An enabled checkstop reset has occurred.
6–13	—	Reserved.
14	SSRS	Software system reset status 0 No software system reset has occurred. 1 A software system reset has occurred.
15	SERF	Software external reset flag 0 No software external reset has occurred. 1 A software external reset has occurred.
16	WKPCFG	Weak pull configuration pin status 0 WKPCFG pin latched during the last reset was logic 0 and weak pull down is the default setting. 1 WKPCFG pin latched during the last reset was logic 1 and weak pull up is the default setting.
17–28	—	Reserved.
29–30	BOOTCFG	Reset configuration pin status. Holds the value of the BOOTCFG[0:1] pins that was latched 4 clocks before the last negation of the $\overline{\text{RSTOUT}}$ pin, if the $\overline{\text{RSTCFG}}$ pin was asserted. If the $\overline{\text{RSTCFG}}$ pin was negated at the last negation of $\overline{\text{RSTOUT}}$, the BOOTCFG field is set to the value 0b00. The BOOTCFG field is used by the BAM program to determine the location of the reset configuration half word. See Table 4-10 for a translation of the reset configuration half word location from the BOOTCFG field value.
31	RGF	$\overline{\text{RESET}}$ glitch flag. Set by the MCU when the $\overline{\text{RESET}}$ pin is asserted for more than 2 clocks clock cycles, but less than the minimum $\overline{\text{RESET}}$ assertion time of 10 consecutive clocks to cause a reset. This bit is cleared by the reset controller for a valid assertion of the $\overline{\text{RESET}}$ pin or a power-on reset or a write of 1 to the bit. 0 No glitch was detected on the $\overline{\text{RESET}}$ pin. 1 A glitch was detected on the $\overline{\text{RESET}}$ pin.

4.3.1.2 System Reset Control Register (SIU_SRCR)

The system reset control register (SIU_SRCR) allows software to generate either a software system reset or software external reset. The software system reset causes an internal reset sequence, while the software external reset only causes the external RSTOUT pin to be asserted. When written to 1, the SER bit automatically clears after a predetermined number of clock cycles (See Section 4.2.2, "Reset Output (RSTOUT)"). If the value of the SER bit is 1 and a 0 is written to the bit, the bit is cleared and the RSTOUT pin is negated regardless of whether the relevant number of clocks has expired.

The CRE bit in the SIU_SRCR allows software to enable a checkstop reset. If enabled, a checkstop reset will occur if the checkstop reset input to the reset controller is asserted. The checkstop reset is enabled by default.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SSR	SER	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset[1]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x10															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CRE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	1[1]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x10															

[1] The CRE bit is reset to 1 by POR. Other resets sources do not reset the bit value.

Figure 4-2. System Reset Control Register (SIU_SRCR)

Table 4-3. SIU_SRCR Field Descriptions

Bits	Name	Description
0	SSR	Software system reset. Writing a 1 to this bit causes an internal reset and assertion of the RSTOUT pin. The bit is automatically cleared by all reset sources except the software external reset. 0 Do not generate a software system reset. 1 Generate a software system reset.
1	SER	Software external reset. Writing a 1 to this bit causes an software external reset. The RSTOUT pin is asserted for a predetermined number of clock cycles (See Section 4.2.2, "Reset Output (RSTOUT)"), but the MCU is not reset. The bit is automatically cleared when the software external reset completes. 0 Do not generate an software external reset. 1 Generate an software external reset.
2–15	—	Reserved.

Table 4-3. SIU_SRCR Field Descriptions (continued)

Bits	Name	Description
16	CRE	Checkstop reset enable Writing a 1 to this bit enables a checkstop reset when the e200z6 core enters a checkstop state. The CRE bit defaults to checkstop reset enabled. This bit is reset at POR. 0 No reset occurs when the e200z6 core enters a checkstop state. 1 A reset occurs when the e200z6 core enters a checkstop state.
17–31	—	Reserved.

4.4 Functional Description

4.4.1 Reset Vector Locations

The reset vector contains a pointer to the instruction where code execution begins after BAM execution. The location of the reset vector is determined by boot mode, as illustrated in Table 4-4.

Table 4-4. Reset Vector Locations

Boot Mode	Reset Vector Location
External Boot	0x0000_0004 (assuming 0x0000_0000 has a valid RCHW)
Internal Boot	Next word address after the first valid RCHW found. The BAM searches the lowest address of each of the six low address space blocks in Flash memory for a valid RCHW. Hence, the possible reset vector locations are: 0x0000_0004 0x0000_4004 0x0001_0004 0x0001_C004 0x0002_0004 0x0003_0004
Serial Boot	Specified over serial download

4.4.2 Reset Sources

4.4.2.1 FMPLL Lock

A loss of lock of the FMPLL can cause a reset (provided the SIU is enabled by the FMPLL_SYNCR[LOLRE] bit). Furthermore, reset will remain asserted, regardless of the source of reset, until after the FMPLL has locked.

4.4.2.2 Flash High Voltage

There is no Flash access gating signal implemented in the MPC5553/MPC5554. However, the device is held in reset for a long enough period of time to guarantee that high voltage circuits are reset and stabilized and that Flash memory is accessible.

4.4.2.3 Reset Source Descriptions

For the following reset source descriptions refer to the reset flow diagrams in Figure 4-5 and Figure 4-6. Figure 4-5 shows the reset flow for assertion of the RESET pin. Figure 4-6 shows the internal processing of reset for all reset sources.

4.4.2.3.1 Power-on Reset

The power-on reset (POR) circuit is designed to detect a POR event and ensure that the RESET signal is correctly sensed. The POR is not intended to be used to detect falling power supply voltages. External supply monitoring should be provided. The output signals from the power-on reset circuits are active low signals. All power-on reset output signals are combined into one POR signal at the V_{DD} level and input to the reset controller. Although assertion of the power-on reset signal causes reset, the RESET pin must be asserted during a power-on reset to guarantee proper operation of the MCU.

The PLLCFG[0:1] and RSTCFG pins determine the configuration of the FMPLL. If the RSTCFG pin is asserted at the negation of RSTOUT, the PLLCFG[0:1] pins set the operating mode of the FMPLL. If RSTCFG is asserted anytime during the assertion of RSTOUT, the FMPLL will switch to the mode specified by the PLLCFG[0:1] pins. The values on the RSTCFG and the PLLCFG[0:1] pins must be kept constant once RSTCFG is asserted to avoid transient mode changes in the FMPLL. If RSTCFG is in the negated state at the negation of RSTOUT, the FMPLL defaults to enabled with a crystal reference. See Chapter 11, “Frequency Modulated Phase Locked Loop (FMPLL) and System Clocks,” for more details on the operation of the FMPLL and the PLLCFG[0:1] pins.

The signal on the WKPCFG pin determines whether weak pull up or pull down devices are enabled after reset on the eTPU and eMIOS pins. The WKPCFG pin is applied starting at the assertion of the internal reset signal, as indicated by the assertion of RSTOUT. Refer to Figure 4-4 and see Chapter 2, “Signal Description,” for information on WKPCFG and RSTOUT.

Once the RESET input pin is negated, the reset controller checks if the FMPLL is locked. The internal reset signal and RSTOUT are kept asserted until the FMPLL is locked. After the FMPLL is locked, the reset controller waits an additional predetermined number of clock cycles (See Section 4.2.2, “Reset Output (RSTOUT)”) before negating the RSTOUT pin. The WKPCFG and BOOTCFG[0:1] pins are sampled 4 clock cycles before the negation of RSTOUT, and the associated bits/fields are updated in the SIU_RSR (note that the BOOTCFG[0:1] pins are only sampled if RSTCFG is asserted). In addition, the PORS and ERS bits are set, and all other reset status bits are cleared in the reset status register.

4.4.2.3.2 External Reset

When the reset controller detects assertion of the RESET pin, the internal reset signal and RSTOUT are asserted. Starting at the assertion of the internal reset signal (as indicated by assertion of RSTOUT), the value on the WKPCFG pin is applied; at the same time the PLLCFG[0:1] values are applied if RSTCFG is asserted. Once the RESET pin is negated and the FMPLL loss of lock request signal is negated, the reset controller waits the predetermined number of clock cycles (see Section 4.2.2, “Reset Output (RSTOUT)”).

Once the clock count finishes, the WKPCFG and BOOTCFG[0:1] pins are sampled (note that the BOOTCFG[0:1] pins are only sampled if RSTCFG is asserted). The reset controller then waits 4 clock cycles before the negating RSTOUT, and the associated bits/fields are updated in the SIU_RSR. In addition, the ERS bit is set, and all other reset status bits in the SIU_RSR are cleared.

4.4.2.3.3 Loss-of-Lock Reset

A loss-of-lock reset occurs when the FMPLL loses lock and the loss-of-lock reset enable (LOLRE) bit in the FMPLL synthesizer control register (FMPLL_SYNCR) is set. The internal reset signal is asserted (as indicated by assertion of RSTOUT). Starting at the assertion of the internal reset signal (as indicated by

assertion of RSTOUT), the value on the WKPCFG pin is applied; at the same time the PLLCFG[0:1] values are applied if RSTCFG is asserted. Once the FMPLL locks, the reset controller waits until the predetermined clock count finishes (See Section 4.2.2, "Reset Output (RSTOUT)") and then the WKPCFG and BOOTCFG[0:1] pins are sampled (note that the BOOTCFG[0:1] pins are only sampled if RSTCFG is asserted). The reset controller then waits 4 clock cycles before negating RSTOUT, and the associated bits/fields are updated in the SIU_RSR. In addition, the LLRS bit is set, and all other reset status bits in the SIU_RSR are cleared. Refer to Chapter 11, "Frequency Modulated Phase Locked Loop (FMPLL) and System Clocks," for more information on loss-of-lock.

4.4.2.3.4 Loss-of-Clock Reset

A loss-of-clock reset occurs when the FMPLL detects a failure in either the reference signal or FMPLL output, and the loss-of-clock reset enable (LOCRE) bit in the FMPLL_SYNCR is set. The internal reset signal is asserted (as indicated by assertion of RSTOUT). Starting at the assertion of the internal reset signal (as indicated by assertion of RSTOUT), the value on the WKPCFG pin is applied; at the same time the PLLCFG[0:1] values are applied if RSTCFG is asserted. Once the FMPLL has a clock and is locked, the reset controller waits the the predetermined clock cycles (See Section 4.2.2, "Reset Output (RSTOUT)") before negating RSTOUT. When the clock count finishes the WKPCFG and BOOTCFG[0:1] pins are sampled (note that the BOOTCFG[0:1] pins are only sampled if RSTCFG is asserted). The reset controller then waits 4 clock cycles before the negating RSTOUT, and the associated bits/fields are updated in the SIU_RSR. In addition, the LCRS bit is set, and all other reset status bits in the SIU_RSR are cleared. Refer to Section 11.4.2.6, "Loss-of-Clock Detection," for more information on loss-of-clock.

4.4.2.3.5 Watchdog Timer/Debug Reset

A watchdog timer reset occurs when the e200z6 core watchdog timer is enabled, and a time-out occurs with the enable next watchdog timer (EWT) and watchdog timer interrupt status (WIS) bits set in the timer status register (TSR), and with the watchdog reset control (WRC) field in the timer control register (TCR) configured for a reset. The WDRS bit in the SIU_RSR is also set when a debug reset command is issued from a debug tool. To determine whether the WDRS bit was set due to a watchdog timer or debug reset, check the WRS field in the e200z6 core TSR. The effect of a watchdog timer or debug reset request is the same for the reset controller. Starting at the assertion of the internal reset signal (as indicated by assertion of RSTOUT), the value on the WKPCFG pin is applied; at the same time the PLLCFG[0:1] values are applied if RSTCFG is asserted. Once the FMPLL is locked, the reset controller waits the predetermined number of clock cycles (See Section 4.2.2, "Reset Output (RSTOUT)") before negating RSTOUT.. When the clock count finishes the WKPCFG and BOOTCFG[0:1] pins are sampled (note that the BOOTCFG[0:1] pins are only sampled if RSTCFG is asserted). The reset controller then waits 4 clock cycles before the negating RSTOUT, and the associated bits/fields are updated in the SIU_RSR. In addition, the WTRS bit is set, and all other reset status bits in the SIU_RSR are cleared. Refer to the e200z6 Core Guide for more information on the watchdog timer and debug operation.

4.4.2.3.6 Checkstop Reset

When the e200z6 core enters a checkstop state, and the checkstop reset is enabled (the CRE bit in the system reset control register (SIU_SRCR) is set), a checkstop reset occurs. Starting at the assertion of the internal reset signal (as indicated by assertion of RSTOUT), the value on the WKPCFG pin is applied; at the same time the PLLCFG[0:1] values are applied if RSTCFG is asserted. Once the FMPLL is locked, the reset controller waits a predetermined number of clock cycles (See Section 4.2.2, "Reset Output (RSTOUT)") before negating RSTOUT. When the clock count finishes the WKPCFG and BOOTCFG[0:1] pins are sampled (note that the BOOTCFG[0:1] pins are only sampled if RSTCFG is asserted). The reset controller then waits 4 clock cycles before the negating RSTOUT, and the associated

bits/fields are updated in the SIU_RSR. In addition, the CRS bit is set, and all other reset status bits in the SIU_RSR are cleared. Refer to e200z6 Core Guide for more information.

4.4.2.3.7 JTAG Reset

A system reset occurs when JTAG is enabled and either the EXTEST, CLAMP, or HIGHZ instructions are executed by the JTAG controller. Starting at the assertion of the internal reset signal (as indicated by assertion of RSTOUT), the value on the WKPCFG pin is applied; at the same time the PLLCFG[0:1] values are applied if RSTCFG is asserted.

Once the JTAG reset request has negated and the FMPLL is locked, the reset controller waits a predetermined number of clock cycles (See Section 4.2.2, "Reset Output (RSTOUT)") before negating RSTOUT.. When the clock count finishes the WKPCFG and BOOTCFG[0:1] pins are sampled (note that the BOOTCFG[0:1] pins are only sampled if RSTCFG is asserted), and their associated bits/fields are updated in the SIU_RSR. The reset source status bits in the SIU_RSR are unaffected. Refer to Chapter 24, "IEEE 1149.1 Test Access Port Controller (JTAGC)," for more information.

4.4.2.3.8 Software System Reset

A software system reset is caused by a write to the SSR bit in the system reset control register (SIU_SRCR). A write of 1 to the SSR bit causes an internal reset of the MCU. The internal reset signal is asserted (as indicated by assertion of RSTOUT). The value on the WKPCFG pin is applied starting at the assertion of the internal reset signal (as indicated by assertion of RSTOUT); at the same time the PLLCFG[0:1] values are applied if RSTCFG is asserted. Once the FMPLL locks, the reset controller waits a predetermined number of clock cycles (See Section 4.2.2, "Reset Output (RSTOUT)") before negating RSTOUT.. When the clock count finishes the WKPCFG and BOOTCFG[0:1] pins are sampled (note that the BOOTCFG[0:1] pins are only sampled if RSTCFG is asserted). The reset controller then waits 4 clock cycles before negating RSTOUT, and the associated bits/fields are updated in the SIU_RSR. In addition, the SSRS bit is set, and all other reset status bits in the SIU_RSR are cleared.

4.4.2.3.9 Software External Reset

A write of 1 to the SER bit in the SIU_SRCR causes the external RSTOUT pin to be asserted for a predetermined number of clocks (See Section 4.2.2, "Reset Output (RSTOUT)"). The SER bit automatically clears after the clock cycle expires. A software external reset does not cause a reset of the MCU, the BAM program is not executed, the PLLCFG[0:1], BOOTCFG[0:1], and WKPCFG pins are not sampled. The SERF bit in the SIU_RSR is set, but no other status bits are affected. The SERF bit in the SIU_RSR is not automatically cleared after the clock count expires, and remains set until cleared by software or another reset besides the software external reset occurs.

For a software external reset, the e200z6 core will continue to execute instructions, timers that are enabled will continue to operate, and interrupt requests will continue to be processed. It is the responsibility of the application to ensure devices connected to RSTOUT are not accessed during a software external reset, and to determine how to manage MCU resources.

4.4.3 Reset Configuration and Configuration Pins

The microcontroller and the BAM perform a reset configuration that allows certain functions of the MCU to be controlled and configured at reset. This reset configuration is defined by:

- Configuration pins
- A reset configuration half word (RCHW), if present
- Serial port, if a serial boot is used

The following sections describe these configuration pins and the RCHW.

4.4.3.1 RSTCFG Pin

Table 4-5 shows the RSTCFG pin settings for configuring the MCU to use a default or a custom configuration. Refer to Chapter 2, "Signal Description" for more information about the RSTCFG pin.

Table 4-5. RSTCFG Settings

RSTCFG	Description
1	Use default configuration of: – booting from internal flash – clock source is a crystal on FMPLL
0	Get configuration information from: – BOOTCFG[0:1] – PLLCFG[0:1]

4.4.3.2 WKPCFG Pin (Reset Weak Pull Up/Pull Down Configuration)

As shown in Table 4-6, the signal on the WKPCFG pin determines whether specific eTPU and eMIOS pins are connected to weak pull up or weak pull down devices during and after reset (see Chapter 2, "Signal Description," for the eTPU and eMIOS pins that are affected by WKPCFG). For all reset sources except the software external reset, the WKPCFG pin is applied starting at the assertion of the internal reset signal (as indicated by the assertion of RSTOUT). If the WKPCFG signal is logic high at this time, pull up devices will be enabled on the eTPU and eMIOS pins. If the WKPCFG signal is logic low at the assertion of the internal reset signal, pull down devices will be enabled on those pins. The value on WKPCFG must be held constant during reset to avoid oscillations on the eTPU and eMIOS pins caused by switching pull up/down states. The final value of WKPCFG is latched 4 clock cycles before the negation of RSTOUT. After reset, software may modify the weak pull up/down selection for all I/O pins through the PCRs in the SIU.

Table 4-6. WKPCFG Settings

WKPCFG	Description
0	Weak pull down applied to eTPU and eMIOS pins at reset
1	Weak pull up applied to eTPU and eMIOS pins at reset

Also refer to Chapter 2, "Signal Description" for information about the WKPCFG pin.

4.4.3.3 BOOTCFG[0:1] Pins (MCU Configuration)

In addition to specifying the RCHW location, the values latched on the BOOTCFG[0:1] pins at reset are used to initialize the internal Flash memory enabled/disabled state, and whether no arbitration or external arbitration of the external bus interface is selected. Additionally, the RCHW can determine either directly or indirectly how the MMU is configured, how the external bus is configured, CAN or eSCI module and pin configuration, Nexus enabling, and password selection.

Also refer to Chapter 2, "Signal Description" for information about the BOOTCFG pins.

4.4.3.4 PLLCFG[0:1] Pins

The role of PLLCFG pins in PLL configuration is explained in Section 11.1.4, "FMPLL Modes of Operation." Also refer to Chapter 2, "Signal Description" for information about the PLLCFG pins.

Table 4-7. PLLCFG[0:1] and $\overline{\text{RSTCFG}}$ in Configuration

$\overline{\text{RSTCFG}}$	PLLCFG0	PLLCFG1	Clock Mode	MODE	PLLSEL	PLLREF
1	PLLCFG pins ignored		Crystal reference (default)	1	1	1
0	0	0	Bypass Mode	0	0	0
0	0	1	External reference	1	1	0
0	1	0	Crystal reference	1	1	1
0	1	1	1:1 Mode	1	0	0

4.4.3.5 Reset Configuration Half Word

4.4.3.5.1 Reset Configuration Half Word Definition

The RCHW is read from either external memory or internal Flash memory. If a valid RCHW is not found, a CAN/SCI boot is initiated. The RCHW is a collection of control bits that specify a minimum MCU configuration after reset and define the desired mode of operation of the BAM program. At reset the RCHW provides a means to locate the boot code, determines if Flash memory is programmed or erased, enables or disables the watchdog timer, and if booting externally, sets the bus size. The user should refer to the appropriate register given by the RCHW bit descriptions for a detailed description of each control bit.

NOTE

Do not configure the RCHW to a 32-bit bus size for devices with only a 16-bit data bus.

If booting from internal Flash or external memory, the user must insure that the RCHW is the correct value for the desired configuration, and that it is located at the proper location in memory. The boot ID of the RCHW must be read as 0x5A. BOOT_BLOCK_ADDRESS is explained in Section 16.3.2.2.5, "Reset Configuration Half Word Read."

The fields of the RCHW are shown in Figure 4-3.

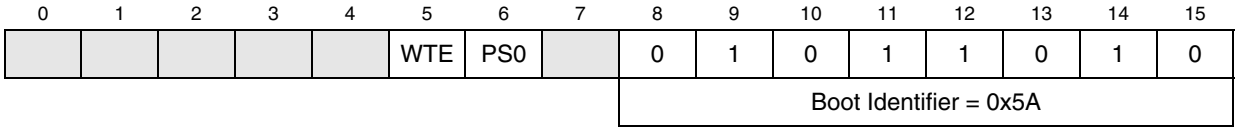


Figure 4-3. RCHW Fields

Table 4-8. Internal Boot RCHW Field Descriptions

Bits	Name	Description
0–4	—	Reserved: These bit values are ignored when the halfword is read. Write to 0 for future compatibility.
5	WTE	Watchdog timer enable. This is used to enable or disable the e200z6 watchdog timer through the BAM program. The configuration of the watchdog timer function is managed through the timer control register (TCR). 0 BAM does not write the e200z6 timebase registers (TBU and TBL) nor enable the e200z6 core watchdog timer. 1 BAM writes the e200z6 timebase registers (TBU and TBL) to 0x0000_0000_0000_0000 and enables the e200z6 core watchdog timer with a time-out period of 3 x 2^{17} system clock cycles. (Example: For 8 MHz crystal –> 12MHz system clock–> 32.7mS time-out. For 20 MHz crystal –> 30 MHz system clock –> 13.1mS time-out)
6	PS0	Port size. Defines the width of the data bus connected to the memory on $\overline{CS0}$. After system reset, CS0 is changed to a 16-bit port by the BAM which fetches the RCHW from either 16- or 32-bit external memories. Then the BAM reconfigures the EBI either as a 16-bit bus or a 32-bit bus, according to the settings of this bit. 0 32-bit CS0 port size 1 16-bit CS0 port size **Note:** Used only in external boot mode. Do not set the port to 32-bits if the device only has a 16-bit data bus.
7	—	Reserved: This bit value is ignored when the halfword is read. Write to 0 for future compatibility.
8–15	BOOTID [0:7]	Boot identifier. This field serves two functions. First, it is used to indicate which block in Flash memory contains the boot program. Second, it identifies whether the Flash memory is programmed or invalid. The value of a valid boot identifier is 0x5A (0b01011010). The BAM program checks the first half word of each Flash memory block starting at block 0 until a valid boot identifier is found. If all blocks in the low address space of the internal Flash are checked and no valid boot identifier is found, then the internal Flash is assumed to be invalid and a CAN/SCI boot is initiated. For an external boot, only block 0 is checked for a valid boot identifier, and if not found, a CAN/SCI boot is initiated.

4.4.3.5.2 Invalid RCHW

If the device is configured for a boot from internal Flash, a valid boot ID must be read at the lowest address of one of the six LAS blocks in internal Flash memory. If the device is configured for a boot from external memory, a valid boot ID must be read at 0x00_0000 of $\overline{CS0}$. Refer to Chapter 16, “Boot Assist Module (BAM)” for more information.

If a valid RCHW is not found, a serial boot is initiated. A serial boot does not use a RCHW. The watchdog timer is enabled. For serial boot entered from a failed external boot, the port size remains configured as 16 bits wide. For serial boot entered from a failed internal boot, the external bus is never configured and remains in the reset state of GPIO inputs.

4.4.3.5.3 Reset Configuration Half Word Source

The reset configuration half word (RCHW) specifies a minimal MCU configuration after reset. The RCHW also contains bits that control the BAM program flow. See Section 16.3.2.1.1, "Finding Reset Configuration Half Word" for information on the BAM using the RCHW. The RCHW is read and applied each time the BAM program executes, which is for every power-on, external, or internal reset event. The only exception to this is the software external reset. See Section 4.4.3.5, "Reset Configuration Half Word," for detailed descriptions of the bits in the RCHW. The RCHW is read from one of the following locations:

- The lowest address (0x00_0000) of an external memory device, enabled by chip select CS0 using either a 16- or 32-bit data bus
- The lowest address of one of the six low address space (LAS) blocks in the internal Flash memory. (2 x 16K; 2 x 48K; 2 x 64K)

At the negation of the $\overline{\text{RSTOUT}}$ pin, the BOOTCFG field in the RSR has been updated. If BOOTCFG0 is asserted, then the BAM program reads the RCHW from address 0x0000_0000 in the external memory connected to $\overline{\text{CS0}}$ (the BAM first configures the MMU and $\overline{\text{CS0}}$ such that address 0x0000_0000 is translated to 0x2000_0000 and then directed to $\overline{\text{CS0}}$). When BOOTCFG0 is asserted, BOOTCFG1 determines whether external arbitration must be enabled to fetch the RCHW.

If BOOTCFG0 and BOOTCFG1 are negated at the negation of the $\overline{\text{RSTOUT}}$ pin, then the BAM program attempts to read the RCHW from the first address of each of the 6 blocks in the low address space (LAS) of internal Flash. Table 4-9 shows the LAS addresses.

Table 4-9. LAS Block Memory Addresses

Block	Address
0	0x0000_0000
1	0x0000_4000
2	0x0001_0000
3	0x0001_C000
4	0x0002_0000
5	0x0003_0000

If the RCHW stored in either internal or external Flash is invalid (boot identifier field of RCHW is not 0x5A), or if BOOTCFG0 is negated and BOOTCFG1 is asserted at the negation of the $\overline{\text{RSTOUT}}$ pin, then RCHW is not applicable, and serial boot mode is performed. Table 4-10 summarizes the RCHW location options.

Note that the BOOTCFG[0:1] = 11 is a meaningless configuration for the MPC5553, because the arbitration pins and TSIZ have been removed.

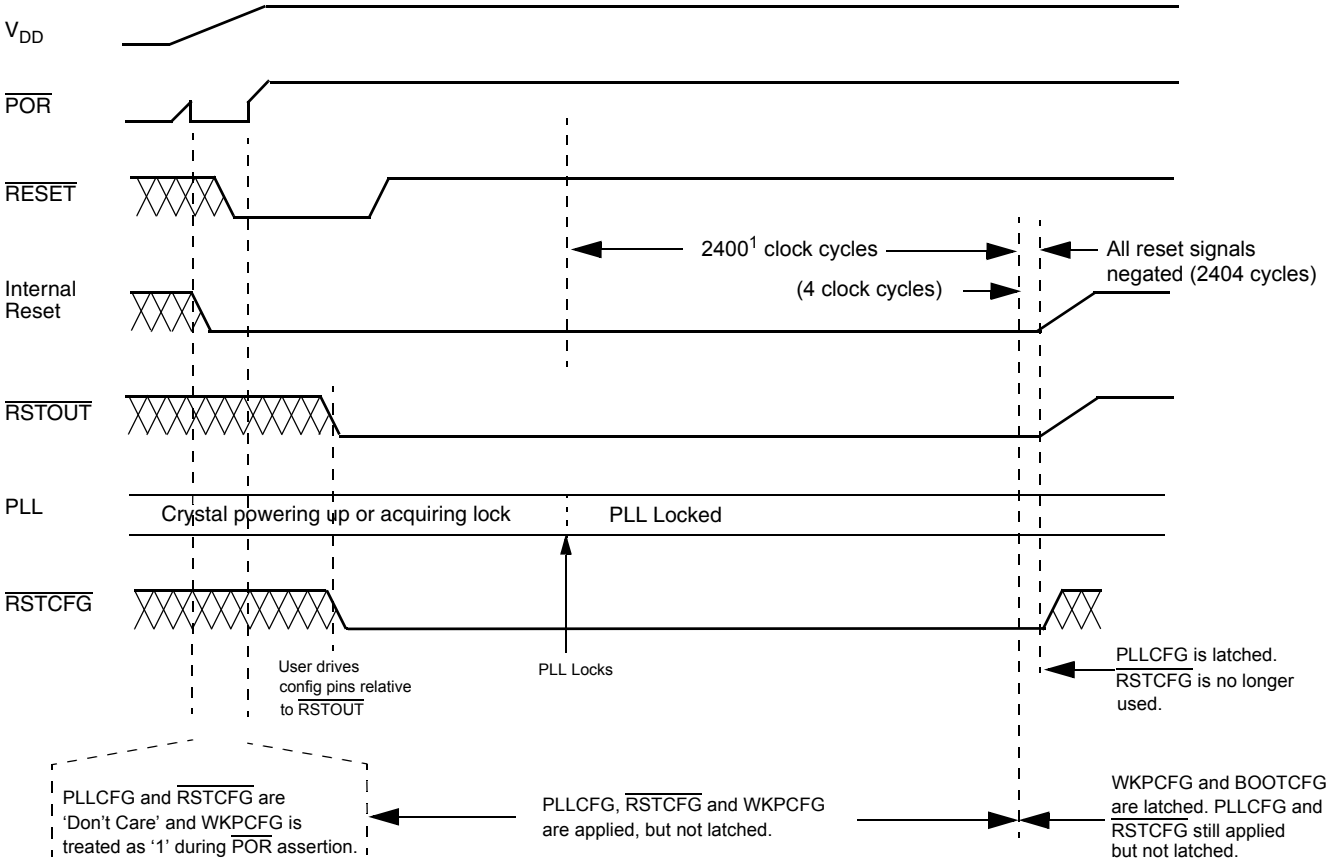
Table 4-10. MPC5553/MPC5554 Reset Configuration Half Word Sources

$\overline{\text{RSTCFG}}$	BOOTCFG0	BOOTCFG1	Boot Identifier Field (RCHW)	Boot Mode	Configuration Word Source
1	—	—	Valid	Internal	The lowest address of one of the six low address spaces (LAS) in internal Flash memory.
			Invalid	Serial	Not applicable
0	0	0	Valid	Internal	The lowest address of one of the six low address spaces (LAS) in internal Flash memory.
			Invalid	Serial	Not applicable
0	0	1	—	Serial	Not applicable
0	1	0	Valid	External Boot, No Arbitration	The lowest address (0x00_0000) of an external memory device, enabled by chip select CS0 using either 16- or 32-bit data bus
			Invalid	Serial	Not applicable
0	1	1	Valid [1]	External [1] Boot, External Arbitration	The lowest address (0x0000_0000) of an external memory device, enabled by chip select CS0 using either 16- or 32-bit data bus.
			Invalid	Serial	Not applicable

[1] External boot mode with external arbitration is not supported in the MPC5553.

4.4.4 Reset Configuration Timing

The timing diagram in Figure 4-4 shows the sampling of the BOOTCFG[0:1], WKPCFG, and PLLCFG[0:1] pins for a power-on reset. The timing diagram is also valid for internal/external resets assuming that V_{DD}, V_{DDSYN}, and V_{DDEH6} are within valid operating ranges. The values of the PLLCFG[0:1] pins are latched at the negation of the $\overline{\text{RSTOUT}}$ pin, if the $\overline{\text{RSTCFG}}$ pin is asserted at the negation of $\overline{\text{RSTOUT}}$. The value of the $\overline{\text{WKPCFG}}$ signal is applied at the assertion of the internal reset signal (as indicated by the assertion of $\overline{\text{RSTOUT}}$). The values of the WKPCFG and BOOTCFG[0:1] pins are latched 4 clock cycles before the negation of $\overline{\text{RSTOUT}}$ and stored in the reset status register (SIU_RSR). BOOTCFG[0:1] are latched only if $\overline{\text{RSTCFG}}$ is asserted. WKPCFG is not dependent on $\overline{\text{RSTCFG}}$.



[1] This clock count is dependent on the configuration of the FMPLL (See Section 4.2.2, "RSTOUT"). If the FMPLL is configured for 1:1 (dual controller) operation or for bypass mode, this clock count is 16000.

Figure 4-4. MPC5553/MPC5554 Reset Configuration Timing

4.4.5 Reset Flow

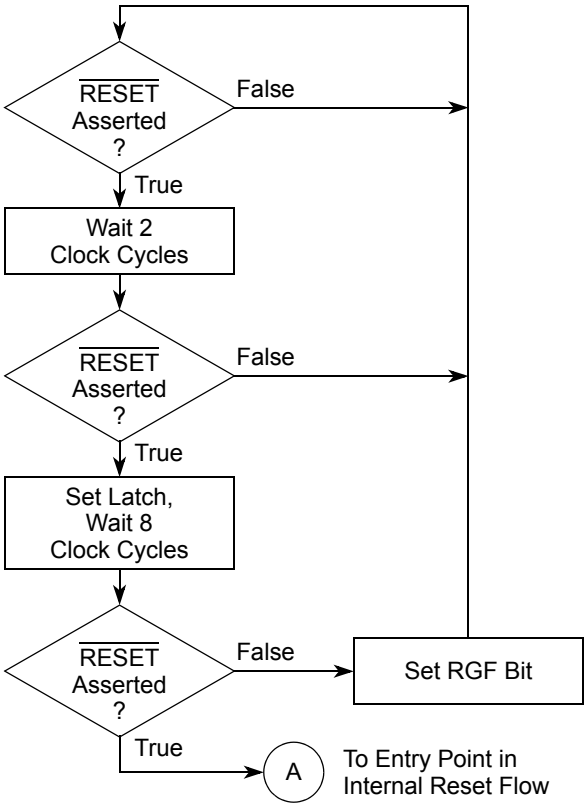
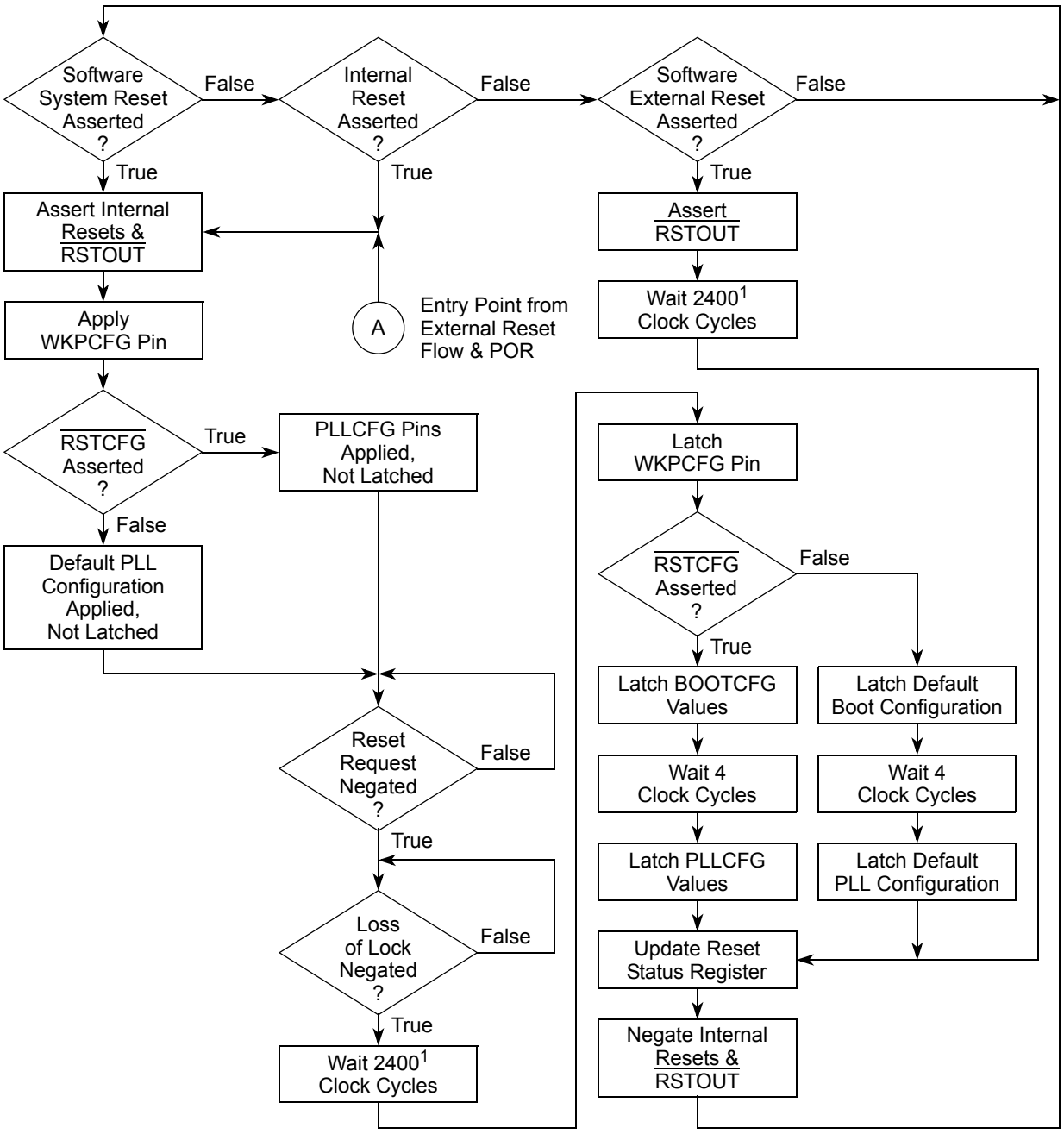


Figure 4-5. External Reset Flow Diagram



NOTES:
[1] The clock count is dependent on the configuration of the FMPLL (refer to Section 5.3.1.2, '$\overline{\text{RSTOUT}}$'). If the FMPLL is configured in 1:1 (dual controller) or bypass mode, this clock count is 16000.

Figure 4-6. Internal Reset Flow Diagram

4.5 Revision History

Substantive Changes since Rev 3.0
Updated Table 4-5 by swapping the 0 and 1.
Changed “Once the debug reset request has negated and the FMPLL is locked, the reset controller waits ...." to "Once the FMPLL is locked, the reset controller waits ...." in the Watchdog Timer/Debug Reset section.

Chapter 5
Peripheral Bridge (PBRIDGE_A, PBRIDGE_B)

5.1 Introduction

5.1.1 Block Diagram

The PBRIDGE is the interface between the system bus and on-chip peripherals as shown in Figure 5-1.

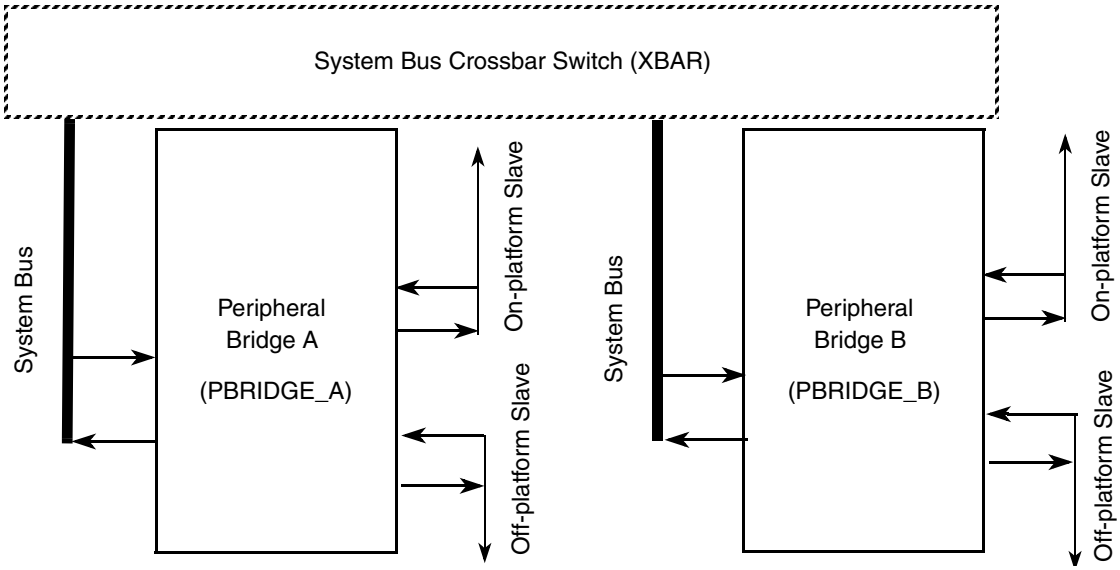


Figure 5-1. PBRIDGE Interface

5.1.2 Overview

There are two peripheral bridges, PBRIDGE_A and PBRIDGE_B, which act as interfaces between the system bus and lower bandwidth peripherals. In this manual, PBRIDGE refers to either of these bridges, as their functionality is identical. The only difference is the peripherals to which they connect. Accesses that fall within the address space of the PBRIDGE are decoded to provide individual module selects for peripheral devices on the slave bus interface.

5.1.2.1 Access Protections

The PBRIDGE provides programmable access protections for both masters and peripherals. It allows the privilege level of a master to be overridden, forcing it to user mode privilege, and allows masters to be designated as trusted or untrusted. Peripherals may require supervisor privilege level for access, may restrict access to a trusted master only, and may be write-protected. See Table 5-1 for a list of master/slave IDs and the peripherals associated with each master and slave. More information on access protection may be found in Section 13.3.2.9, “Flash Bus Interface Unit Access Protection Register (FLASH_BIUAPR).”

Table 5-1. Peripheral Bridge Master/Slave ID Table

XBAR Port	XBS port Module	Master ID	Peripheral
Master 0	e200z6 Core—CPU	0	
	e200z6—Nexus	1	
Master 1	eDMA	2	
Master 2	EBI	3	
Master 3 (MPC5553 only)	FEC (MPC5553 only)	4	
Slave 0	FLASH		
Slave 1	EBI		
Slave 3	L2SRAM		
Slave 6	PBRIDGE_A		PBRIDGE_A
			FMPLL
			EBI Control
			FLASH Control
			SIU
			eMIOS
			eTPU reg
			eTPU PRAM
			eTPU PRAM Mirror
			eTPU SCM

Table 5-1. Peripheral Bridge Master/Slave ID Table (continued)

XBAR Port	XBS port Module	Master ID	Peripheral
Slave 7	PBRIDGE_B		PBRIDGE_B
			XBAR
			ESCM
			eDMA Control
			INTC
			FEC (MPC5553 only)
			eQADC
			DSPIA (MPC5554 only)
			DSPIB
			DSPIC
			DSPID
			eSCIA
			eSCIB
			CANA
			CANB (MPC5554 only)
			CANC
			BAM

5.1.3 Features

The following list summarizes the key features of the PBRIDGE:

- Supports the slave interface signals. This interface is only meant for slave peripherals.
- Supports 32-bit slave peripherals. (Byte, halfword, and word reads and write are supported to each.)
- Supports a pair of slave accesses for 64-bit instruction fetches.
- Provides configurable per-module write buffering support.
- Provides configurable per-module and per-master access protections.

5.1.4 Modes of Operation

The PBRIDGE has only one operating mode.

5.2 External Signal Description

The PBRIDGE has no external signals.

5.3 Memory Map/Register Definition

The memory maps for the 32-bit PBRIDGE registers are shown in Table 5-2 and Table 5-3.

Table 5-2. PBRIDGE_A Memory Map

Address	Register Name	Register Description	Size (bits)
Base (0xC3F0_0000)	PBRIDGE_A_MPCR	Master privilege control register	32
Base + 0x004– Base + 0x01F	—	Reserved	—
Base + 0x020	PBRIDGE_A_PACR0	Peripheral access control register 0	32
Base + 0x024– Base + 0x03F	—	Reserved	—
Base + 0x040	PBRIDGE_A_OPACR0	Off-platform peripheral access control register 0	32
Base + 0x044	PBRIDGE_A_OPACR1	Off-platform peripheral access control register 1	32
Base + 0x048	PBRIDGE_A_OPACR2	Off-platform peripheral access control register 2	32
Base + 0x04C– Base + 0x053	—	Reserved	—

Table 5-3. PBRIDGE_B Memory Map

Address	Register Name	Register Description	Size (bits)
Base (0xFFF0_0000)	PBRIDGE_B_MPCR	Master privilege control register	32
Base + 0x004– Base + 0x01F	—	Reserved	—
Base + 0x020	PBRIDGE_B_PACR0	Peripheral access control register 0	32
Base + 0x024– Base + 0x027	—	Reserved	—
Base + 0x028	PBRIDGE_B_PACR2	Peripheral access control register 2	32
Base + 0x02C– Base + 0x03F	—	Reserved	—
Base + 0x040	PBRIDGE_B_OPACR0	Off-platform peripheral access control register 0	32
Base + 0x044	PBRIDGE_B_OPACR1	Off-platform peripheral access control register 1	32
Base + 0x048	PBRIDGE_B_OPACR2	Off-platform peripheral access control register 2	32
Base + 0x04C	PBRIDGE_B_OPACR3	Off-platform peripheral access control register 3	32
Base + 0x050– Base + 0x053	—	Reserved	—

5.3.1 Register Descriptions

There are three types of registers that control each PBRIDGE. All registers are 32-bit registers and can only be accessed in supervisor mode by trusted bus masters. Additionally, these registers must only be read from or written to by a 32-bit aligned access. PBRIDGE registers are mapped into the PBRIDGE_A and PBRIDGE_B address spaces. The protection and access fields of the MPR, PACR, and OPACR registers are 4 bits in width.

5.3.1.1 Master Privilege Control Register (PBRIDGE_*x*_MPCR)

Each master privilege control register (PBRIDGE_*x*_MPCR) specifies 4-bit access fields defining the access privilege level associated with a bus master in the platform, as well as specifying whether write accesses from this master are bufferable. The registers provide one field per bus master. Note that access field 4 is available only in the MPC5553.

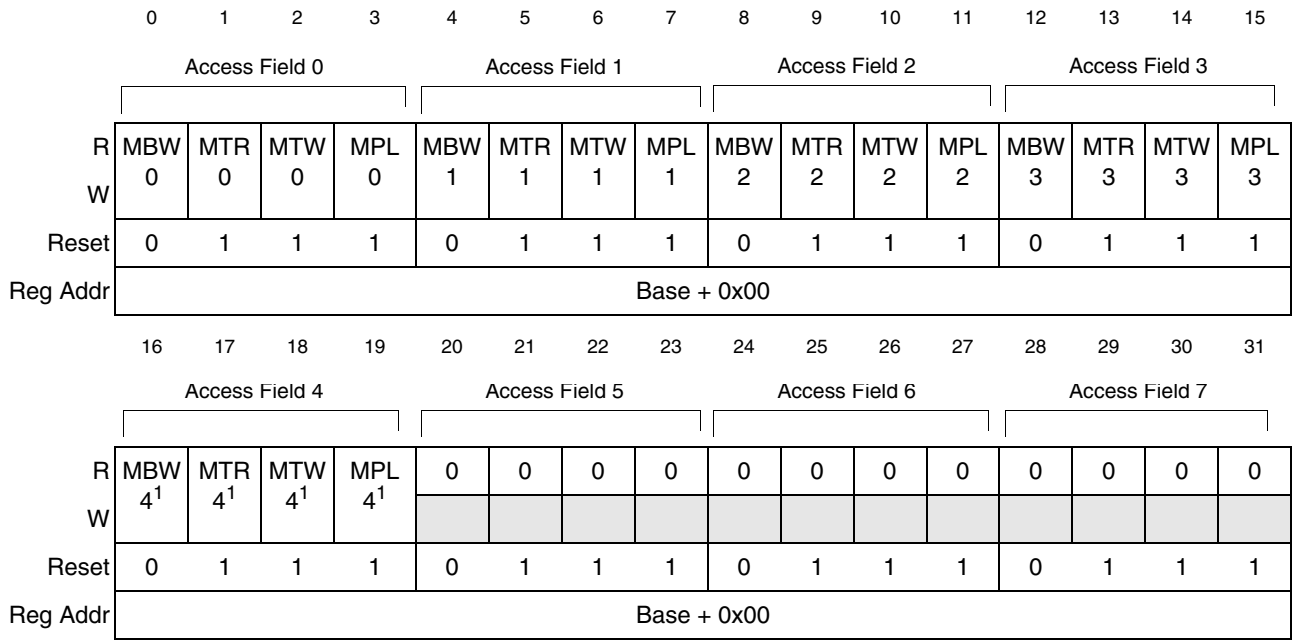


Figure 5-2. Master Privilege Control Registers (PBRIDGE_*x*_MPCR)

[1] Available only in the MPC5553

Table 5-4. PBRIDGE_*x*_MPCR Field Descriptions

Bits	Name	Description
0	MBW0	Master buffer writes. Determines whether the PBRIDGE is enabled to buffer writes from the CPU. Writes not able to be buffered by default. 0 Write accesses from the CPU are not bufferable 1 Write accesses from the CPU are allowed to be buffered
1	MTR0	Master trusted for reads. Determines whether the CPU is trusted for read accesses. Trusted by default. 0 The CPU is not trusted for read accesses. 1 The CPU is trusted for read accesses.

Table 5-4. PBRIDGE_*x*_MPCR Field Descriptions (continued)

Bits	Name	Description
2	MTW0	Master trusted for writes. Determines whether the master is trusted for write accesses. Trusted by default. 0 The CPU is not trusted for write accesses. 1 The CPU is trusted for write accesses.
3	MPL0	Master privilege level. Determines how the privilege level of the CPU is determined. Accesses not forced to user mode by default. 0 Accesses from the CPU are forced to user mode. 1 Accesses from the CPU are not forced to user mode.
4	MBW1	Master buffer writes. Determines whether the PBRIDGE is enabled to buffer writes from the Nexus. Writes not able to be buffered by default. 0 Write accesses from the Nexus are not bufferable 1 Write accesses from the Nexus are allowed to be buffered
5	MTR1	Master trusted for reads. Determines whether the Nexus is trusted for read accesses. Trusted by default. 0 The Nexus is not trusted for read accesses. 1 The Nexus is trusted for read accesses.
6	MTW1	Master trusted for writes. Determines whether the master is trusted for write accesses. Trusted by default. 0 The Nexus is not trusted for write accesses. 1 The Nexus is trusted for write accesses.
7	MPL1	Master privilege level. Determines how the privilege level of the Nexus is determined. Accesses not forced to user mode by default. 0 Accesses from the Nexus are forced to user mode. 1 Accesses from the Nexus are not forced to user mode.
8	MBW2	Master buffer writes. Determines whether the PBRIDGE is enabled to buffer writes from the eDMA. Writes not able to be buffered by default. 0 Write accesses from the eDMA are not bufferable 1 Write accesses from the eDMA are allowed to be buffered
9	MTR2	Master trusted for reads. Determines whether the eDMA is trusted for read accesses. Trusted by default. 0 The eDMA is not trusted for read accesses. 1 The eDMA is trusted for read accesses.
10	MTW2	Master trusted for writes. Determines whether the master is trusted for write accesses. Trusted by default. 0 The eDMA is not trusted for write accesses. 1 The eDMA is trusted for write accesses.
11	MPL2	Master privilege level. Determines how the privilege level of the eDMA is determined. Accesses not forced to user mode by default. 0 Accesses from the eDMA are forced to user mode. 1 Accesses from the eDMA are not forced to user mode.
12	MBW3	Master buffer writes. Determines whether the PBRIDGE is enabled to buffer writes from the EBI. Writes not able to be buffered by default. 0 Write accesses from the EBI are not bufferable 1 Write accesses from the EBI are allowed to be buffered

Table 5-4. PBRIDGE_*x*_MPCR Field Descriptions (continued)

Bits	Name	Description
13	MTR3	Master trusted for reads. Determines whether the EBI is trusted for read accesses. Trusted by default. 0 The EBI is not trusted for read accesses. 1 The EBI is trusted for read accesses.
14	MTW3	Master trusted for writes. Determines whether the master is trusted for write accesses. Trusted by default. 0 The EBI is not trusted for write accesses. 1 The EBI is trusted for write accesses.
15	MPL3	Master privilege level. Determines how the privilege level of the EBI is determined. Accesses not forced to user mode by default. 0 Accesses from the EBI are forced to user mode. 1 Accesses from the EBI are not forced to user mode.
16	MBW4[1]	Master buffer writes. Determines whether the PBRIDGE is enabled to buffer writes from the FEC. Writes not able to be buffered by default. 0 Write accesses from the FEC are not bufferable 1 Write accesses from the FEC are allowed to be buffered **Note:** Applies only to MPC5553. Reserved in MPC5554.
17	MTR4[1]	Master trusted for reads. Determines whether the FEC is trusted for read accesses. Trusted by default. 0 The FEC is not trusted for read accesses. 1 The FEC is trusted for read accesses. **Note:** Applies only to MPC5553. Reserved in MPC5554.
18	MTW4[1]	Master trusted for writes. Determines whether the master is trusted for write accesses. Trusted by default. 0 The FEC is not trusted for write accesses. 1 The FEC is trusted for write accesses. **Note:** Applies only to MPC5553. Reserved in MPC5554.
19	MPL4[1]	Master privilege level. Determines how the privilege level of the FEC is determined. Accesses not forced to user mode by default. 0 Accesses from the FEC are forced to user mode. 1 Accesses from the FEC are not forced to user mode. **Note:** Applies only to MPC5553. Reserved in MPC5554.
20–31	—	Reserved.

[1] Applies only to MPC5553. Reserved in MPC5554.

5.3.1.2 Peripheral Access Control Registers (PBRIDGE_*x*_PACR) and Off-Platform Peripheral Access Control Registers (PBRIDGE_*x*_OPACR)

Each of the PBRIDGE on-platform peripherals has a 4-bit access field in a peripheral access control register (PACR) that defines the access levels supported by the given module. A single PACR contains up to eight of these module-access fields, and the PACR register structure is shown in Table 5-2 and Table 5-3. The PACR registers with their access fields are shown in Figure 5-3. There are three PACR registers, one for bridge A and two for bridge B.

Also, each of the off-platform peripherals has a 4-bit access field in an off-platform peripheral access control register (PBRIDGE_*x*_OPACR) that defines the access levels supported by the given module. Each OPACR contains up to eight of these module-access fields, and the OPACR register structure is

shown in Table 5-2 and Table 5-3. The OPACR registers with their access fields are shown in Figure 5-4. Seven OPACR registers are used, three for bridge A, and four for bridge B.

NOTE

Not all members of the MPC5500 family have PBRIDGE_x_PACR and PBRIDGE_x_OPACR. On the parts that do not have them, writes to their addresses will receive a transfer error. If ensuring code compatibility across all family members is wanted, then writes to those addresses must be qualified with SIU_MIDR[PARTNUM].

The type of peripheral designated by each PACR and OPACR access field is shown in Table 5-6.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	Access Field 0				Access Field 1				Access Field 2				Access Field 3			
R W	BW0[1]	SP0	WP0	TP0	BW1	SP1	WP1	TP1	BW2	SP2	WP2	TP2	BW3	SP3	WP3	TP3
Reset A_PACR0	0	1[2]	0	1[2]	0	0	0	0	0	0	0	0	0	0	0	0
Reset B_PACR0	0	1[2]	0	1[2]	0	1[2]	0	0	0	0	0	0	0	0	0	0
Reset B_PACR2	0	1[2]	0	0	0	1[2]	0	0	0	1[2]	0	0	0	1[2]	0	0
Reg Addr	Base + 0x20 (PBRIDGE_A_PACR0 and PBRIDGE_B_PACR0); Base + 0x28 (PBRIDGE_B_PACR2)															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	Access Field 4				Access Field 5				Access Field 6				Access Field 7			
R W	BW4	SP4	WP4	TP4	BW5	SP5	WP5	TP5	BW6	SP6	WP6	TP6	BW7	SP7	WP7	TP7
Reset A_PACR0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset B_PACR0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset B_PACR2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x20 (PBRIDGE_A_PACR0 and PBRIDGE_B_PACR0); Base + 0x28 (PBRIDGE_B_PACR2)															

Figure 5-3. Peripheral Access Control Registers (PBRIDGE_*x*_PACR*n*)

1 In PBRIDGE_A_PACR0 and in PBRIDGE_B_PACR0, the BW0 bit is not writeable.

2 The default value is 0b0000 for PACR peripheral access fields that are unused or not connected.

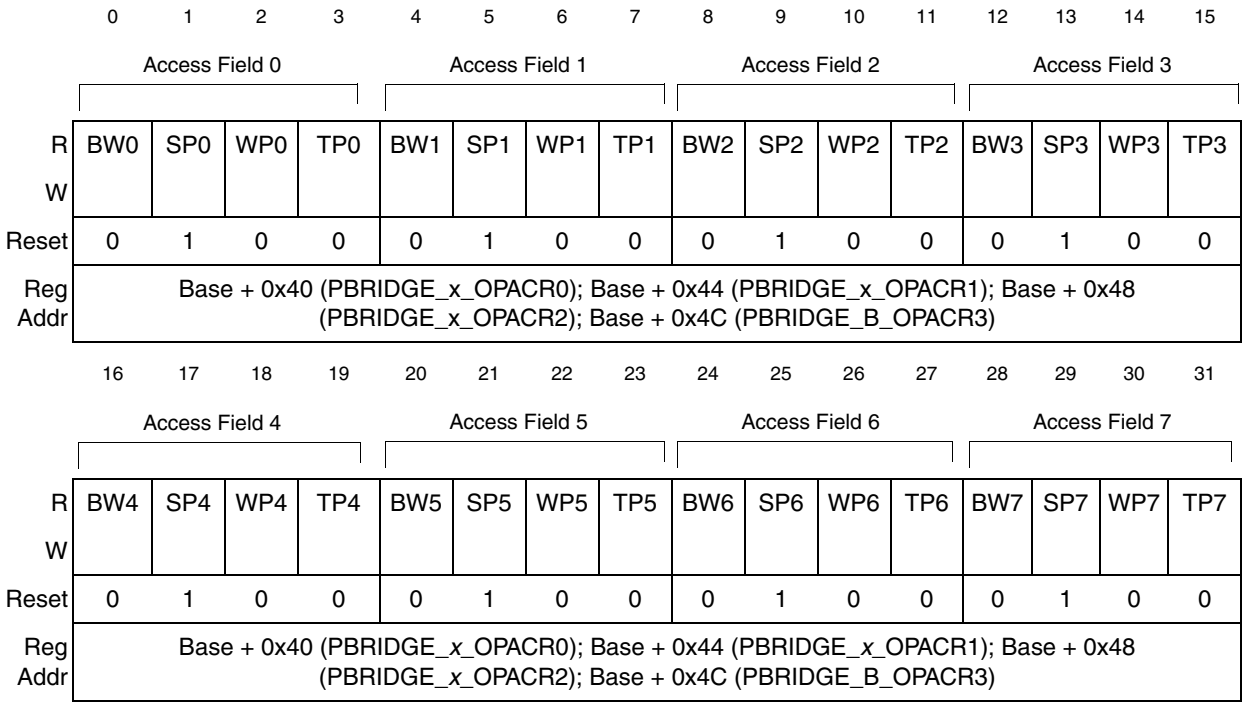


Figure 5-4. Off-platform Peripheral Access Control Registers (PBRIDGE_*x*_OPACR*n*)

Table 5-5. PBRIDGE_*x*_PACR*n* and PBRIDGE_*x*_OPACR*n* Field Descriptions

Bits	Name	Description
0, 4, 8, 12, 16, 20, 24, 28	BW*n*[1]	Buffer writes. Determines whether write accesses to this peripheral are allowed to be buffered. Write accesses not bufferable by default 0 Write accesses to this peripheral are not bufferable by the PBRIDGE. 1 Write accesses to this peripheral are allowed to be buffered by the PBRIDGE. **Note:** In PBRIDGE_A_PACR0 and PBRIDGE_B_PACR0, the BW0 bit is not writeable.
1, 5, 9, 13, 17, 21, 25, 29	SP*n*	Supervisor protect. Determines whether the peripheral requires supervisor privilege level for access. Supervisor privilege level required by default. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses. The PBRIDGE_x_MPCR[MPLy] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the slave bus.

Table 5-5. PBRIDGE_*x*_PACR*n* and PBRIDGE_*x*_OPACR*n* Field Descriptions (continued)

Bits	Name	Description
2, 6, 10, 14, 18, 22, 26, 30	WP*n*	Write protect. Determines whether the peripheral allows write accesses. Write accesses allowed by default. 0 This peripheral allows write accesses. 1 This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the slave bus.
3, 7, 11, 15, 19, 23, 27, 31	TP*n*	Trusted protect. Determines whether the peripheral allows accesses from an untrusted master. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the slave bus.

[1] In PBRIDGE_A_PACR0 and PBRIDGE_B_PACR0, the BW0 bit is not writeable.

Presence or absence of a module's 4-bit access field in either a PBRIDGE_x_PACR or PBRIDGE_x_OPACR is based on whether the associated peripheral is present on the device. When absent, the corresponding field is not implemented and will read as 0's. Writes will be ignored.

NOTE

Table 5-6 lists all of the access fields in the PACRs and OPACRs in both PBRIDGE_A and PBRIDGE_B, and each of the associated peripherals present on the MPC5553/MPC5554.

Table 5-6. PACR/OPACR Access Control Registers and Peripheral Mapping

Register	Register Address	Peripheral Access Field #	Peripheral Type	Access Field Default Value
PBRIDGE_A				
PBRIDGE_A_PACR0	PBRIDGE_A_Base + 0x020	0	PBRIDGE_A	0b0101
		1-7	—	0b0000
PBRIDGE_A_OPACR0	PBRIDGE_A_Base + 0x040	0	FMPLL	0b0100
		1	EBI Control	0b0100
		2	Flash Control	0b0100
		3	—	0b0100
		4	SIU	0b0100
		5-7	—	0b0100
PBRIDGE_A_OPACR1	PBRIDGE_A_Base + 0x044	0	eMIOS	0b0100
		1-7	—	0b0100

Table 5-6. PACR/OPACR Access Control Registers and Peripheral Mapping (continued)

Register	Register Address	Peripheral Access Field #	Peripheral Type	Access Field Default Value
PBRIDGE_A_OPACR2	PBRIDGE_A_Base + 0x048	0	eTPU	0b0100
		1	—	0b0100
		2	eTPU PRAM	0b0100
		3	eTPU PRAM Mirror	0b0100
		4	eTPU SCM	0b0100
		5-7	—	0b0100
PBRIDGE_B				
PBRIDGE_B_PACR0	PBRIDGE_B_Base + 0x020	0	PBRIDGE_B	0b0101
		1	XBAR	0b0100
		2-7	—	0b0000
PBRIDGE_B_PACR2	PBRIDGE_B_Base + 0x028	0	ESCM	0b0100
		1	eDMA	0b0100
		2	INTC	0b0100
		3[1]	FEC[1]	0b0100
		4-7	—	0b0000
PBRIDGE_B_OPACR0	PBRIDGE_B_Base + 0x040	0	eQADC	0b0100
		1-3	—	0b0100
		4[2]	DSPI_A[2]	0b0100
		5	DSPI_B	0b0100
		6	DSPI_C	0b0100
		7	DSPI_D	0b0100
PBRIDGE_B_OPACR1	PBRIDGE_B_Base + 0x044	0-3	—	0b0100
		4	eSCI_A	0b0100
		5	eSCI_B	0b0100
		6-7	—	0b0100
PBRIDGE_B_OPACR2	PBRIDGE_B_Base + 0x048	0	FlexCAN_A	0b0100
		1[2]	FlexCAN_B [2]	0b0100
		2	FlexCAN_C	0b0100
		3-7	—	0b0100
PBRIDGE_B_OPACR3	PBRIDGE_B_Base + 0x04C	0-6	—	0b0100
		7	BAM	0b0100

[1] In MPC5553 only, not present in MPC5554

[2] In MPC5554 only, not present in MPC5553

5.4 Functional Description

The PBRIDGE serves as an interface between a system bus and the peripheral (slave) bus. It functions as a protocol translator. Support is provided for generating a pair of 32-bit peripheral accesses when targeted by a 64-bit system bus access. No other bus-sizing access support is provided.

Accesses that fall within the address space of the PBRIDGE are decoded to provide individual module selects for peripheral devices on the slave bus interface.

5.4.1 Access Support

Aligned 64-bit accesses, aligned word and halfword accesses, as well as byte accesses are supported for 32-bit peripherals. Peripheral registers must not be misaligned, although no explicit checking is performed by the PBRIDGE.

NOTE

Data accesses that cross a 32-bit boundary are not supported.

5.4.2 Peripheral Write Buffering

The PBRIDGE provides programmable write buffering capability to allow certain write accesses to be buffered in the PBRIDGE for later completion, while terminating the system bus access early. This provides improved performance in systems where frequent writes to a slow peripheral are performed. Write buffering must only be enabled for masters and peripherals for which an error termination from the slave bus will either not occur, or is safe to ignore.

When write buffering is enabled, all accesses through the PBRIDGE will still occur in-order; no bypassing of buffered writes is supported.

Write buffering is controllable on a per-master and per-peripheral basis.

NOTE

Write buffering will cause the processor core to believe that the write has completed before it actually has completed in the peripheral. If write buffering is enabled for a peripheral, the actual write will take an additional two system clock cycles plus any additional system clock cycles that the register needs. Most registers in the MPC5500 family only will delay the write by two clock cycles, but some registers will take longer. This early termination, as seen by the processor core, can defeat the **mbar** or **msync** instruction between the write to clear a flag bit and the write to the INTC_EOIR. Refer to Section 10.4.3.1.2, “End-of-Interrupt Exception Handler.” Therefore, if write buffering is enabled for a peripheral that has a flag bit, insert instructions between the **mbar** or **msync** instruction and the write to the INTC_EOIR that will consume at least the number of system clock cycles that the actual write is delayed.

5.4.2.1 Read Cycles

Read accesses are possible with the PBRIDGE when the requested access size is 32-bits or smaller, and is not misaligned across a 32-bit boundary. 64-bit data reads (not instruction) are not supported.

5.4.2.2 Write Cycles

Write accesses are possible with the PBRIDGE when the requested access size is 32-bits or smaller. Misaligned writes that cross a 32-bit boundary are not supported. 64-bit data writes (not instruction) are not supported.

5.4.2.3 Buffered Write Cycles

Single clock write responses to the system bus are possible with the PBRIDGE when the requested write access is bufferable. If the requested access does not violate the permissions check, and if both master and peripheral are enabled for buffering writes, the PBRIDGE will internally buffer the write cycle. The write cycle is terminated early with zero system bus wait states. The access proceeds normally on the slave interface, but error responses are ignored.

All accesses are initiated and completed in order on the slave interface, regardless of buffering. If the buffer is full, a following write cycle will stall until it can either be buffered (if bufferable) or can be initiated. If the buffer has valid entries, a following read cycle will stall until the buffer is emptied and the read cycle can be completed.

5.4.3 General Operation

Slave peripherals are modules that contain readable/writable control and status registers. The system bus master reads and writes these registers through the PBRIDGE. The PBRIDGE generates module enables, the module address, transfer attributes, byte enables, and write data as inputs to the slave peripherals. The PBRIDGE captures read data from the slave interface and drives it on the system bus.

Separate interface ports are provided for on-platform and off-platform peripherals. The distinction between on-platform and off-platform is made to allow platform-based designs incorporating the PBRIDGE to separate the interface ports to allow for ease of timing closure. In addition, module selects and control register storage for on-platform peripherals are allocated at synthesis time, allowing only needed resources to be implemented. Off-platform module selects and control register storage do not have the same degree of configurability.

The modules that are on-platform and those that are off-platform are detailed in Table 5-7.

Table 5-7. On-Platform and Off-Platform Peripherals

On-Platform	Off-Platform
Enhanced Direct Memory Access (eDMA)	Deserial Serial Peripheral Interface (DSPI)
PBridge A and B	Enhanced Queued Analog-to-Digital Converter (eQADC)
Interrupt Controller (INTC)	Enhanced Serial Communication Interface (eSCI)
Error Correction Status Module (ECSM)	FlexCAN Controller Area Network
Fast Ethernet Controller (FEC)	Boot Assist Module (BAM)
System Bus Crossbar Switch (XBAR)	System Integration Unit (SIU)
	Enhanced Modular Input/Output Subsystem (eMIOS)
	Frequency Modulated Phase Locked Loop (FMPLL)
	Enhanced Time Processing Unit (eTPU)

Table 5-7. On-Platform and Off-Platform Peripherals (continued)

On-Platform	Off-Platform
	External Bus Interface (EBI)
	Flash Bus Interface Unit (FBIU)

The PBRIDGE occupies a 64 Mbyte portion of the address space. A 0.5 Mbyte portion of this space is allocated to on-platform peripherals. The remaining 63.5 Mbytes are available for off-platform devices. The register maps of the slave peripherals are located on 16-Kbyte boundaries. Each slave peripheral is allocated one 16-Kbyte block of the memory map, and is activated by one of the module enables from the PBRIDGE. Up to thirty-two 16-Kbyte external slave peripherals may be implemented, occupying contiguous blocks of 16 Kbytes. Two global external slave module enables are available for the remaining 63 Mbytes of address space to allow for customization and expansion of addressed peripheral devices. In addition, a single non-global module enable is also asserted whenever any of the 32 non-global module enables is asserted.

The PBRIDGE is responsible for indicating to slave peripherals if an access is in supervisor or user mode. The PBRIDGE may block user mode accesses to certain slave peripherals or it may allow the individual slave peripherals to determine if user mode accesses are allowed. In addition, peripherals may be designated as write-protected. The PBRIDGE supports the notion of trusted masters for security purposes. Masters may be individually designated as trusted for reads, trusted for writes, or trusted for both reads and writes, as well as being forced to look as though all accesses from a master are in user mode privilege level.

The PBRIDGE also supports buffered writes, allowing write accesses to be terminated on the system bus in a single clock cycle, and then subsequently performed on the slave interface. Write buffering is controllable on a per-peripheral basis. The PBRIDGE implements a two-entry 32-bit write buffer.

5.5 Revision History

Substantive Changes since Rev 3.0
Fixed Figure 5-2. There were 4 MBW4s listed, now they are MBW4, MTR4, MTW4 and MPL4.
Changed MPCR1 to MPCR in module memory map.
Added Note to Section 5.4.2, “Peripheral Write Buffering.”

Chapter 6
System Integration Unit (SIU)

6.1 Introduction

This chapter describes the MPC5553/MPC5554 system integration unit (SIU), which controls MCU reset configuration, pad configuration, external interrupt, general-purpose I/O (GPIO), internal peripheral multiplexing, and the system reset operation.

6.1.1 Block Diagram

Figure 6-1 is a block diagram of the SIU. The signals shown are external pins to the device. The SIU registers are accessed through the crossbar switch. Note that the power-on reset detection module, pad interface/pad ring module, and peripheral I/O channels are external to the SIU.

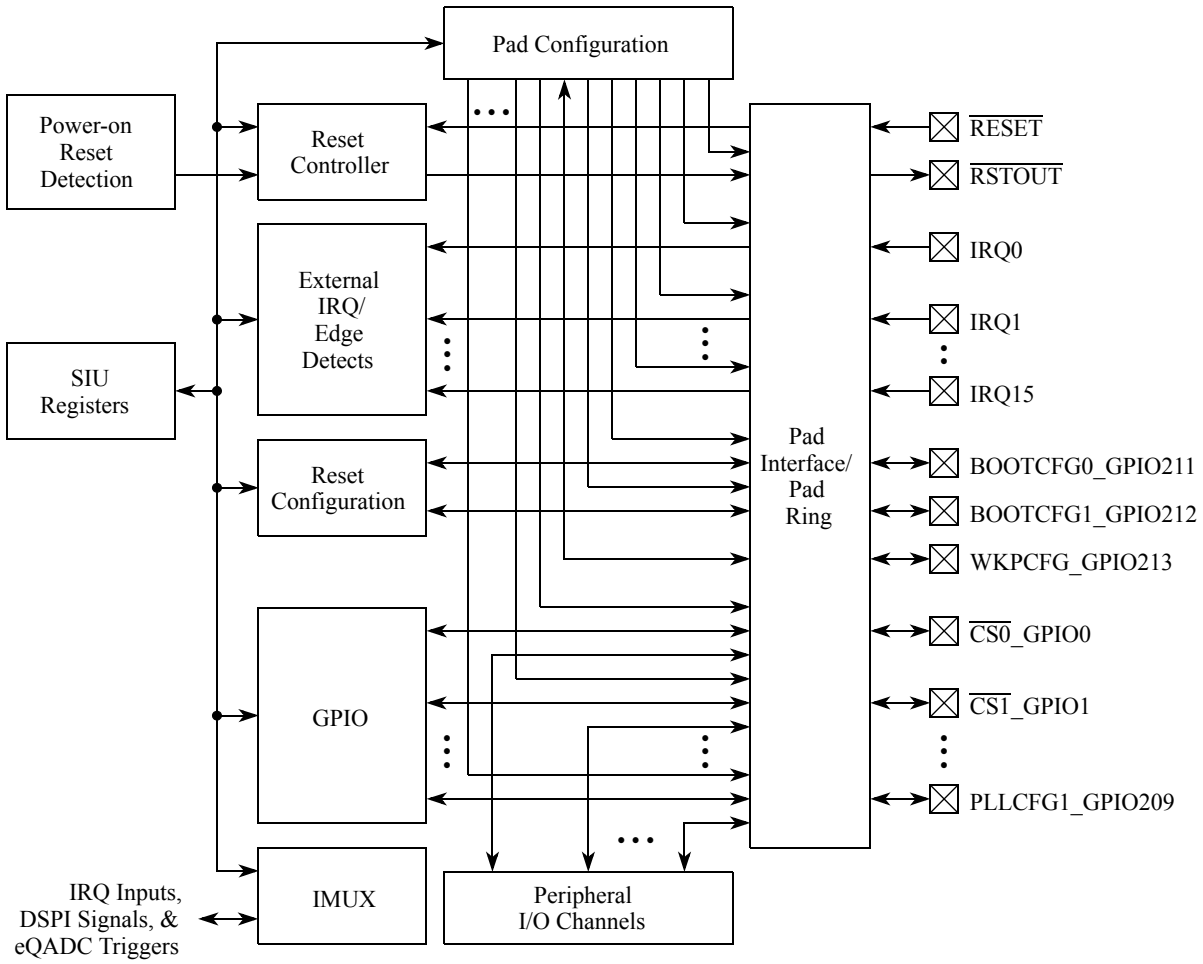


Figure 6-1. SIU Block Diagram

6.1.2 Overview

The MPC5553/MPC5554 system integration unit (SIU) controls MCU reset configuration, pad configuration, external interrupt, general-purpose I/O (GPIO), internal peripheral multiplexing, and the system reset operation. The reset configuration module contains the external pin boot configuration logic. The pad configuration module controls the static electrical characteristics of I/O pins. The GPIO module provides uniform and discrete input/output control of the I/O pins of the MCU. The reset controller performs reset monitoring of internal and external reset sources, and drives the RSTOUT pin. The SIU is accessed by the e200z6 core through the system bus crossbar switch (XBAR) and the peripheral bridge A (PBRIDGE_A).

6.1.3 Features

Features include the following:

- System configuration
  - MCU reset configuration via external pins
  - Pad configuration control
- System reset monitoring and generation
  - Power-on reset support
  - Reset status register providing last reset source to software
  - Glitch detection on reset input
  - Software controlled reset assertion
- External Interrupt
  - 16 (MPC5554) or 16 (MPC5553) interrupt requests
  - Rising or falling edge event detection
  - Programmable digital filter for glitch rejection
- GPIO
  - GPIO function on 214 I/O pins (MPC5554). There are 177 GPIO pins in the MPC5553.
  - Dedicated input and output registers for each GPIO pin.
- Internal Multiplexing
  - Allows serial and parallel chaining of DSPIs
  - Allows flexible selection of eQADC trigger inputs
  - Allows selection of interrupt requests between external pins and DSPI

6.1.4 Modes of Operation

6.1.5 Normal Mode

In normal mode, the SIU provides the register interface and logic that controls system configuration, the reset controller, and GPIO. The SIU continues operation with no changes in stop mode.

6.1.6 Debug Mode

SIU operation in debug mode is identical to operation in normal mode.

6.2 External Signal Description

Table 6-1 lists the external pins used by the SIU.

Table 6-1. SIU Signal Properties

Name	I/O Type	Pad Type	Function	Pull Up/Down[1]
Resets				
$\overline{\text{RESET}}$	Input	—	Reset Input	Up
$\overline{\text{RSTOUT}}$	Output	Slow	Reset Output	—

Table 6-1. SIU Signal Properties (continued)

Name	I/O Type	Pad Type	Function	Pull Up/Down[1]
System Configuration				
GPIO[0:210]	I/O	Slow	General-Purpose I/O	Up/Down
BOOTCFG0_ GPIO211	Input I/O	Slow	Boot Configuration Input / General-Purpose I/O	Down Up/Down
BOOTCFG1_ GPIO212	Input I/O	Slow	Boot Configuration Input / General-Purpose I/O	Down Up/Down
WKPCFG GPIO213	Input I/O	Slow	Weak Pull Configuration Pin / General-Purpose I/O	Up Up/Down
External Interrupt				
IRQ[0:15][2]	Input	Slow	External Interrupt Request Input	—[3]

1 Internal weak pull up/down. The reset weak pull up/down state is given by the pull up/down state for the primary pin function. For example, the reset weak pull up/down state of the BOOTCFG0_GPIO211 pin is weak pull up enabled.

2 The IRQ pins are multiplexed with other functions on the chip.

3 The weak pull up/down state at reset for the IRQ[0:15] depends on the pins that they are shared with. The weak pull up/down state for these pins is as follows: IRQ[0,1,4,5,6,7,12,13,14]: Up, IRQ[2,3,15]: Down, IRQ[8:11]: WKPCFG.

6.2.1 Detailed Signal Descriptions

6.2.1.1 Reset Input ($\overline{\text{RESET}}$)

The $\overline{\text{RESET}}$ pin is an active low input. The $\overline{\text{RESET}}$ pin is asserted by an external device during a power-on or external reset. The internal reset signal asserts only if the $\overline{\text{RESET}}$ pin asserts for 10 clock cycles. Assertion of the $\overline{\text{RESET}}$ pin while the device is in reset causes the reset cycle to start over. The $\overline{\text{RESET}}$ pin has a glitch detector which detects spikes greater than 2 clock cycles in duration that fall below the switch point of the input buffer logic of the VDDEH input pins. The switch point lies between the maximum VIL and minimum VIH specifications for the VDDEH input pins.

6.2.1.2 Reset Output ($\overline{\text{RSTOUT}}$)

The $\overline{\text{RSTOUT}}$ pin is an active low output that uses a push/pull configuration. The $\overline{\text{RSTOUT}}$ pin is driven to the low state by the MCU for all internal and external reset sources. After the negation of the $\overline{\text{RESET}}$ input, $\overline{\text{RSTOUT}}$ is asserted for 2404 clock cycles; except that if the PLL is configured for dual-controller mode, $\overline{\text{RSTOUT}}$ is asserted for 16004 clocks.

The $\overline{\text{RSTOUT}}$ pin can also be asserted for 2400 clock cycles by a write to the SER bit of the system reset control register (SIU_SRCR).

NOTE

During a power on reset, $\overline{\text{RSTOUT}}$ is tri-stated.

6.2.1.3 General-Purpose I/O Pins (GPIO[0:210])

The GPIO pins provide general-purpose input and output function. The GPIO pins are generally multiplexed with other I/O pin functions. Each GPIO input and output is separately controlled by an eight-bit input (SIU_GPDI) or output (SIU_GPDO) register. See Section 6.3.1.13, "GPIO Pin Data Output Registers 0–213 (SIU_GPDOn)" and Section 6.3.1.14, "GPIO Pin Data Input Registers 0–213 (SIU_GPDIn)".

6.2.1.4 Boot Configuration Pins (BOOTCFG[0:1])

The boot configuration pins specify the boot mode initiated by the boot assist module (BAM) program. BOOTCFG[0:1] are input pins that are sampled 4 clock cycles before the negation of the RSTOUT pin, and the values latched are stored in the reset status register (SIU_RSR). This occurs for all reset sources except a debug port reset and a software external reset. The BOOTCFG[0:1] pins are only sampled if the RSTCFG pin is asserted during reset. Otherwise, if the RSTCFG pin is negated during reset, the BOOTCFG[0:1] pins are not sampled, the BAM defaults to boot from internal Flash, and the BOOTCFG field in the SIU_RSR is set to the boot from internal Flash value (0b00). The latched BOOTCFG[0:1] values are also driven as output signals from the SIU.

The BOOTCFG pin values are used only if the RSTCFG pin is asserted during the assertion of RSTOUT. Otherwise, the default values for the BOOTCFG bits in the SIU_RSR are used.

6.2.1.5 I/O Pin Weak Pull Up Reset Configuration Pin (WKPCFG)

The WKPCFG pin is applied at the assertion of the internal reset signal (indicated by the assertion of RSTOUT), and is sampled 4 clock cycles before the negation of the RSTOUT pin. The value is used to configure whether the eTPU and eMIOS pins are connected to internal weak pull up or weak pull down devices after reset. The value latched on the WKPCFG pin at reset is stored in the reset status register (SIU_RSR), and is updated for all reset sources except the debug port reset and software external reset.

6.2.1.6 External Interrupt Request Input Pins (IRQ[0:15])

The IRQ[0:15] connect to the SIU IRQ inputs. SIU_ETISR select register 1 is used to select the IRQ[0:15] pins as inputs to the IRQs. The counter operates independently of IRQ or overrun flag bit clearing. Clearing an IRQ or overrun flag bit does not clear or reload the counter.

Rising or falling edge events are enabled by setting the corresponding bits in the SIU_IREER or the SIU_IFEER. If the same bit location is set in both registers, both rising and falling edge events will cause the corresponding IRQ Flag bit in Section 6.3.1.4, "External Interrupt Status Register (SIU_EISR)" to be set.

6.2.1.6.1 External Interrupts

The IRQ*n* pins map to 16 independent interrupt request outputs from the SIU. An interrupt request is asserted when the corresponding IRQ flag bit is set in Section 6.3.1.4, "External Interrupt Status Register (SIU_EISR)" with the corresponding dma/interrupt request enable bit set in Section 6.3.1.5, "DMA/Interrupt Request Enable Register (SIU_DIRER)," and the corresponding dma/interrupt select bit cleared in Section 6.3.1.6, "DMA/Interrupt Request Select Register (SIU_DIRSR)." The IRQ flag bit is set when an event as defined by the Section 6.3.1.9, "IRQ Rising-Edge Event Enable Register (SIU_IREER)," occurs on the corresponding IRQ*n* pin.

6.2.1.6.2 DMA Transfers

The IRQ*n* pins map to 16 independent DMA request outputs from the SIU. A DMA request is asserted when the corresponding IRQ flag bit is set in Section 6.3.1.4, "External Interrupt Status Register (SIU_EISR)," with the corresponding dma/interrupt request enable bit set in Section 6.3.1.5, "DMA/Interrupt Request Enable Register (SIU_DIRER)," and the corresponding dma/interrupt select bit set in Section 6.3.1.6, "DMA/Interrupt Request Select Register (SIU_DIRSR)." A DMA done signal is input to the SIU for each DMA request output. The assertion of a DMA done signal clears the corresponding IRQ Flag bit.

6.2.1.6.3 Overruns

An overrun interrupt request exists for each overrun flag in the SIU. In addition, there is one overrun interrupt request output from the SIU which is the logical OR of all of the overrun interrupt requests. An overrun interrupt request is asserted if any of the same bit locations are set in Section 6.3.1.7, "Overrun Status Register (SIU_OSR)," and Section 6.3.1.8, "Overrun Request Enable Register (SIU_ORER)." An overrun occurs if an edge triggered event occurs on an IRQ*n* pin while the corresponding IRQ flag bit is set in Section 6.3.1.4, "External Interrupt Status Register (SIU_EISR)."

6.2.1.6.4 Edge Detects

The IRQ*n* pins can be used as edge detect pins. Edge detect operation is enabled by selecting rising or falling edge events in Section 6.3.1.9, "IRQ Rising-Edge Event Enable Register (SIU_IREER)," with dma/interrupt requests disabled. The external IRQ status register reflects whether the desired edge has been captured on each pin.

6.3 Memory Map/Register Definition

Table 6-2 is the address map for the SIU registers. All register addresses are given as an offset of the SIU base address.

Table 6-2. SIU Address Map

Address	Register Name	Register Description	Size (bits)
Base (0xC3F9_0000)	—	Reserved	—
Base + 0x4	SIU_MIDR	MCU ID register	32
Base + 0x8	—	Reserved	—
Base + 0xC	SIU_RSR	Reset status register	32
Base + 0x10	SIU_SRCR	System reset control register	32
Base + 0x14	SIU_EISR	SIU external interrupt status register	32
Base + 0x18	SIU_DIRER	DMA/interrupt request enable register	32
Base + 0x1C	SIU_DIRSR	DMA/interrupt request select register	32
Base + 0x20	SIU_OSR	Overrun status register	32
Base + 0x24	SIU_ORER	Overrun request enable register	32
Base + 0x28	SIU_IREER	IRQ rising-edge event enable register	32
Base + 0x2C	SIU_IFEER	IRQ falling-edge event enable register	32

Table 6-2. SIU Address Map (continued)

Address	Register Name	Register Description	Size (bits)
Base + 0x30	SIU_IDFR	IRQ digital filter register	32
Base + 0x34– Base + 0x3F	—	Reserved	—
Base + 0x40– Base + 0x20C	SIU_PCR0– SIU_PCR230	Pad configuration registers 0–230	16
Base + 0x20E– Base + 0x5FF	—	Reserved	—
Base + 0x600– Base + 0x6D5	SIU_GPDO0– SIU_GPDO213	GPIO pin data output registers 0–213	8
Base + 0x6D6– Base + 0x7FF	—	Reserved	—
Base + 0x800– Base + 0x8D5	SIU_GPDI0– SIU_GPDI213	GPIO pin data input registers 0–213	8
Base + 0x8D6– Base + 0x8FF	—	Reserved	—
Base + 0x900– Base + 0x903	SIU_ETISR	eQADC trigger input select register	32
Base + 0x904– Base + 0x907	SIU_EIISR	External IRQ input select register	32
Base + 0x908– Base + 0x90B	SIU_DISR	DSPI input select register	32
Base + 0x90C– Base + 0x97F	—	Reserved	—
Base + 0x980	SIU_CCR	Chip configuration register	32
Base + 0x984	SIU_ECCR	External clock control register	32
Base + 0x988	SIU_CARH	Compare A high register	32
Base + 0x98C	SIU_CARL	Compare A low register	32
Base + 0x990	SIU_CBRH	Compare B high register	32
Base + 0x994	SIU_CBRL	Compare B low register	32
Base + 0x998– Base + 0x9FF	—	Reserved	—

6.3.1 Register Descriptions

6.3.1.1 MCU ID Register (SIU_MIDR)

The SIU_MIDR contains the part identification number and mask revision number specific to the device. The part number is a read-only field that is mask programmed with the part number of the device. The part number is changed if a new module is added to the device or a memory size is changed, for example. It is not changed for bug fixes or process changes. The mask number is a read-only field that is mask

programmed with the specific mask revision level of the device. The current value applies to revision 0 and will be updated for each mask revision.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PARTNUM															
W																
Reset MPC5553	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	1
Reset MPC5554	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x4															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	MASKNUM_MAJOR				MASKNUM_MINOR			
W																
Reset MPC5553	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset MPC5554	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x4															

Figure 6-2. MCU ID Register (SIU_MIDR)

Table 6-3. SIU_MIDR Field Descriptions

Bits	Name	Description
0–15	PARTNUM [0:15]	MCU part number. Read-only, mask programmed part identification number of the MCU. Reads 0x00 for the MPC5554. For the MPC5553 the value is 0x53.
16–23	—	Reserved
24–27	MASKNUM_MAJOR [0:3]	Major revision number of MCU mask. Read-only, mask programmed mask number of the MCU. Reads 0x0 for the initial mask set of the MPC5554 and the MPC5553, and will change sequentially for each mask set.
28–31	MASKNUM_MINOR [0:3]	Minor revision number of MCU mask. Read-only, mask programmed mask number of the MCU. Reads 0x0 for the initial mask set of the MPC5554 and the MPC5553, and will change sequentially for each mask set.

6.3.1.2 Reset Status Register (SIU_RSR)

The SIU_RSR reflects the most recent source, or sources of reset, and the state of configuration pins at reset. This register contains one bit for each reset source, indicating that the last reset was power-on reset (POR), external, software system, software external reset, watchdog, loss of PLL lock, loss of clock or checkstop reset. A reset status bit set to logic one indicates the type of reset that occurred. Once set, the

reset source status bits in the SIU_RSR remain set until another reset occurs. In the following cases more than one reset bit is set:

1. If a power-on reset request has negated and the MPC5553/MPC5554 is still in the resulting reset, and then an external reset is requested, both the power-on and external reset status bits will be set. In this case, the MPC5553/MPC5554 started the reset sequence due to a power-on reset request, but it ended the reset sequence after an external reset request.
2. If a software external reset is requested, the SERF flag bit is set, but no previously set bits in the SIU_RSR will be cleared. The SERF bit is cleared by writing a 1 to the bit location or when another reset source is asserted.
3. If any of the loss of clock, loss of lock, watchdog or checkstop reset requests occur on the same clock cycle, and no other higher priority reset source is requesting reset (See Table 6-4), the reset status bits for all of the requesting resets will be set.

Simultaneous reset requests are prioritized. When reset requests of different priorities occur on the same clock cycle, the lower priority reset request will be ignored. Only the highest priority reset request's status bit will be set. Except for a power-on reset request and condition 1 above, all reset requests of any priority are ignored until the MPC5553/MPC5554 exits reset.

Table 6-4. Reset Source Priorities

Reset Source	Priority
Power on reset (POR) and external reset (Group 0)	Highest
Software system reset (Group1)	
Loss of clock, loss of lock, watchdog, checkstop (Group2)	
Software external reset (Group 3)	Lowest

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PORS	ERS	LLRS	LCRS	WDRS	CRS	0	0	0	0	0	0	0	0	SSRS	SERF
W																
Reset[1]	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0xC															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WKP CFG[2]	0	0	0	0	0	0	0	0	0	0	0	0	BOOTCFG		RGF
W																
Reset[1]	U[2]													U[3]		0
Reg Addr	Base + 0xC															

1 The reset status register receives its reset values during power-on reset.

2 The reset value of the WKPCFG bit is determined by the value on the WKPCFG pin at reset.

3 The reset value of the BOOTCFG field is determined by the values on the BOOTCFG[0:1] pins at reset.

Figure 6-3. Reset Status Register (SIU_RSR)

Table 6-5. SIU_RSR Field Descriptions

Bits	Name	Description
0	PORS	Power-on reset status. 0 Another reset source has been acknowledged by the reset controller since the last assertion of the power-on reset input. 1 The power-on reset input to the reset controller has been asserted and no other reset source has been acknowledged since that assertion of the power-on reset input except an external reset.
1	ERS	External reset status. 0 The last reset source acknowledged by the reset controller was not a valid assertion of the $\overline{\text{RESET}}$ pin. 1 The last reset source acknowledged by the reset controller was a valid assertion of the $\overline{\text{RESET}}$ pin.
2	LLRS	Loss of lock reset status. 0 The last reset source acknowledged by the reset controller was not a loss of PLL lock reset. 1 The last reset source acknowledged by the reset controller was a loss of PLL lock reset.
3	LCRS	Loss of clock reset status. 0 The last reset source acknowledged by the reset controller was not a loss of clock reset. 1 The last reset source acknowledged by the reset controller was a loss of clock reset.
4	WDRS	Watchdog timer/debug reset status. 0 The last reset source acknowledged by the reset controller was not a watchdog timer or debug reset. 1 The last reset source acknowledged by the reset controller was a watchdog timer or debug reset.
5	CRS	Checkstop reset status. 0 The last reset source acknowledged by the reset controller was not an enabled checkstop reset. 1 The last reset source acknowledged by the reset controller was an enabled checkstop reset.
6–13	—	Reserved.
14	SSRS	Software system reset status. 0 The last reset source acknowledged by the reset controller was not a software system reset. 1 The last reset source acknowledged by the reset controller was a software system reset.
15	SERF	Software external reset flag. 0 This bit has been cleared from a 1 to a 0 by a write of 1 to it when it was a 1 or the software external reset input to the reset controller has not been asserted. 1 The software external reset input to the reset controller has been asserted while this bit was a 0.
16	WKPCFG	Weak pull configuration pin status 0 The WKPCFG pin latched during the last reset was a logical 0 and weak pull down is the default setting 1 The WKPCFG pin latched during the last reset was a logical 1 and weak pullup is the default setting
17–28	—	Reserved.

Table 6-5. SIU_RSR Field Descriptions (continued)

Bits	Name	Description
29–30	BOOTCFG	Reset configuration pin status. Holds the value of the BOOTCFG pins that were latched on the last negation of the RSTOUT pin, if the RSTCFG pin was asserted. If the RSTCFG pin was not asserted at the last negation of RSTOUT, and the lower half or least significant half word of the censorship control word equals 0xFFFF or 0x0000, the BOOTCFG field is set to the value 0b10. Otherwise, if the RSTCFG pin was negated at the last negation of RSTOUT and the lower half of the censorship control word does not equal 0xFFFF or 0x0000, then the BOOTCFG field is set to the value 0b00. The BOOTCFG field is used by the BAM program to determine the location of the reset configuration half word. See Table 4-10 for a translation of the reset configuration half word location from the BOOTCFG field value.
31	RGF	Reset glitch flag. Set by the reset controller when a glitch is detected on the RESET pin. This bit is cleared by the assertion of the power-on reset input to the reset controller, or a write of 1 to the RGF bit. See Section 6.4.2.1, "RESET Pin Glitch Detect," for more information on glitch detection. 0 No glitch has been detected on the RESET pin. 1 A glitch has been detected on the RESET pin.

6.3.1.3 System Reset Control Register (SIU_SRCR)

The system reset control register allows software to generate either a system or external reset. The software system reset causes an internal reset, while the software external reset only causes the external RSTOUT pin to be asserted. When written to 1, the SER bit automatically clears.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SSR[1]	SER	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x10															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CRE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	1[2]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x10															

[1] The SSR bit always reads as 0. A write of 0 to this bit has no effect.

[2] The CRE bit is set to 1 by POR. Other resets sources do not reset the bit value.

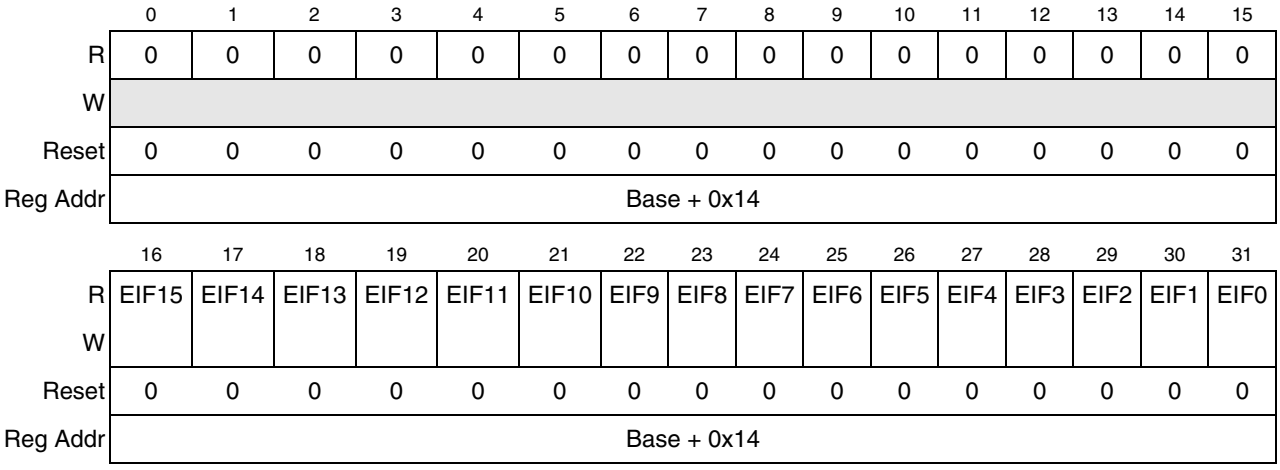
Figure 6-4. System Reset Control Register (SIU_SRCR)

Table 6-6. SIU_SRCR Field Descriptions

Bits	Name	Description
0	SSR	Software system reset. Used to generate a software system reset. Writing a 1 to this bit causes an internal reset. The software system reset is processed as a synchronous reset. The bit is automatically cleared on the assertion of any other reset source except a software external reset. 0 Do not generate a software system reset. 1 Generate a software system reset.
1	SER	Software external reset. Used to generate a software external reset. Writing a 1 to this bit causes the $\overline{\text{RSTOUT}}$ pin to be asserted for 2400 clocks, but the internal reset is not asserted. The bit is automatically cleared when the software external reset completes or any other reset source is asserted. Once a software external reset has been initiated, the $\overline{\text{RSTOUT}}$ pin is negated if this bit is cleared before the 2400 clock period expires. 0 Do not generate a software external reset. 1 Generate a software external reset. **Note:** If the PLL is configured for dual controller mode writing a 1 to SER causes the $\overline{\text{RSTOUT}}$ pin to be asserted for 16000 clocks. Refer to Section 4.2.2, "Reset Output (RSTOUT)."
2–15	—	Reserved.
16	CRE	Checkstop reset enable. Writing a 1 to this bit enables a reset when the checkstop reset request input is asserted. The checkstop reset request input is a synchronous internal reset source. The CRE bit defaults to checkstop reset enabled at POR. If this bit is cleared, it remains cleared until the next POR. 0 No reset occurs when the checkstop reset input to the reset controller is asserted. 1 A reset occurs when the checkstop reset input to the reset controller is asserted.
17–31	—	Reserved.

6.3.1.4 External Interrupt Status Register (SIU_EISR)

The external interrupt status register is used to record edge triggered events on the IRQ0 - IRQ15 inputs to the SIU. When an edge triggered event is enabled in the SIU_IREER or SIU_IFEER for an IRQ*n* input and then sensed, the corresponding SIU_EISR flag bit is set. The IRQ flag bit is set regardless of the state of the corresponding dma/interrupt request enable bit in SIU_DIRER. The IRQ flag bit remains set until cleared by software or through the servicing of a DMA request. The IRQ flag bits are cleared by writing a 1 to the bits. A write of 0 has no effect.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x14															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EIF15	EIF14	EIF13	EIF12	EIF11	EIF10	EIF9	EIF8	EIF7	EIF6	EIF5	EIF4	EIF3	EIF2	EIF1	EIF0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x14															

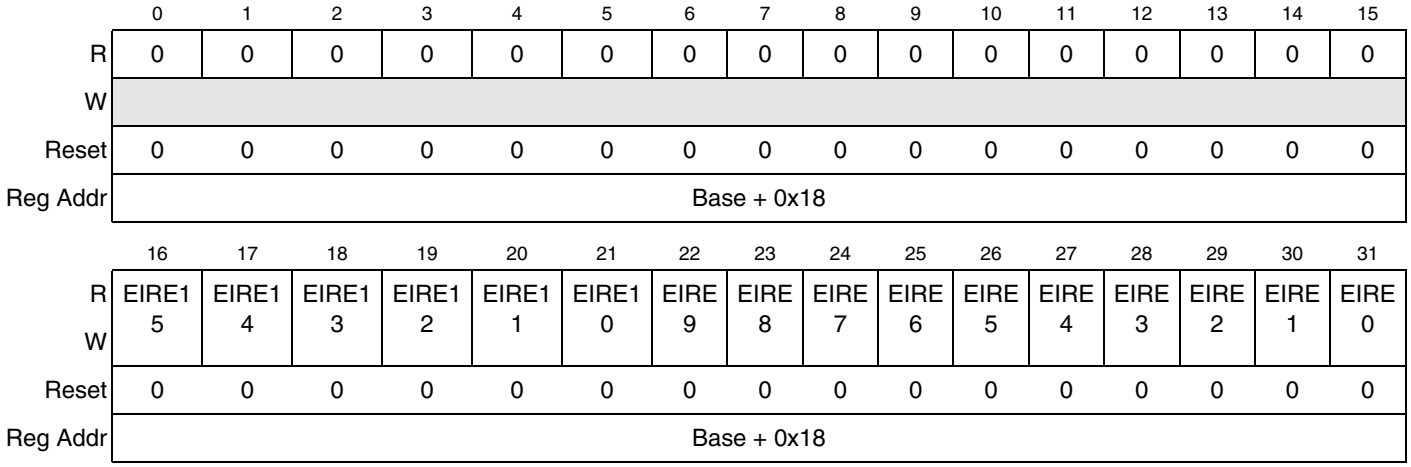
Figure 6-5. SIU External Interrupt Status Register (SIU_EISR)

Table 6-7. SIU_EISR Field Descriptions

Bits	Name	Function
0–15	—	Reserved
16–31	EIF*n*	External interrupt request flag *n*. This bit is set when an edge triggered event occurs on the corresponding IRQ*n* input. 0 No edge triggered event has occurred on the corresponding IRQ*n* input. 1 An edge triggered event has occurred on the corresponding IRQ*n* input.

6.3.1.5 DMA/Interrupt Request Enable Register (SIU_DIRER)

The SIU_DIRER allows the assertion of a DMA or interrupt request if the corresponding flag bit is set in the SIU_EISR. The external interrupt request enable bits enable the interrupt or DMA request. There is only one interrupt request from the SIU to the interrupt controller. The EIRE bits allow selection of which external interrupt request flag bits cause assertion of the one interrupt request signal.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x18															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	EIRE15	EIRE14	EIRE13	EIRE12	EIRE11	EIRE10	EIRE9	EIRE8	EIRE7	EIRE6	EIRE5	EIRE4	EIRE3	EIRE2	EIRE1	EIRE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x18															

Figure 6-6. SIU DMA/Interrupt Request Enable Register (SIU_DIRER)

Table 6-8. SIU_DIRER Field Descriptions

Bits	Name	Function
0–15	—	Reserved.
16–31	EIRE*n*	External interrupt request enable *n*. Enables the assertion of the interrupt request from the SIU to the interrupt controller when an edge triggered event occurs on the IRQ*n* pin. 0 External interrupt request is disabled. 1 External interrupt request is enabled.

6.3.1.6 DMA/Interrupt Request Select Register (SIU_DIRSR)

The SIU_DIRSR allows selection between a DMA or interrupt request for events on the IRQ0–IRQ3 inputs. The SIU_DIRSR selects between DMA and interrupt requests. If the corresponding bits are set in SIU_EISR and the SIU_DIRER, then the DMA/interrupt request select bit determines whether a DMA or interrupt request is asserted.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x1C															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	DIRS3	DIRS2	DIRS1	DIRS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x1C															

Figure 6-7. DMA/Interrupt Request Select Register (SIU_DIRSR)

Table 6-9. SIU_DIRER Field Descriptions

Bits	Name	Function
0–27	—	Reserved.
28–31	DIRS*n*	DMA/interrupt request select *n*. Selects between a DMA or interrupt request when an edge triggered event occurs on the corresponding IRQ*n* pin. 0 Interrupt request is selected. 1 DMA request is selected.

6.3.1.7 Overrun Status Register (SIU_OSR)

The SIU_OSR contains flag bits that record an overrun.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x20															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	OVF15	OVF14	OVF13	OVF12	OVF11	OVF10	OVF9	OVF8	OVF7	OVF6	OVF5	OVF4	OVF3	OVF2	OVF1	OVF0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x20															

Figure 6-8. Overrun Status Register (SIU_OSR)

Table 6-10. SIU_OSR Field Descriptions

Bits	Name	Function
0–15	—	Reserved.
16–31	OVF*n*	Overrun flag *n*. This bit is set when an overrun occurs on the corresponding IRQ*n* pin. 0 No overrun has occurred on the corresponding IRQ*n* pin. 1 An overrun has occurred on the corresponding IRQ*n* pin.

6.3.1.8 Overrun Request Enable Register (SIU_ORER)

The SIU_ORER contains bits to enable an overrun if the corresponding flag bit is set in the SIU_OSR. If any overrun request enable bit and the corresponding flag bit are set, the single combined overrun request from the SIU to the interrupt controller is asserted.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x24															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ORE15	ORE14	ORE13	ORE12	ORE11	ORE10	ORE9	ORE8	ORE7	ORE6	ORE5	ORE4	ORE3	ORE2	ORE1	ORE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x24															

Figure 6-9. Overrun Request Enable Register (SIU_ORER)

Table 6-11. SIU_ORER Field Descriptions

Bits	Name	Function
0–15	—	Reserved.
16–31	ORE*n*	Overrun request enable *n. E*nables the corresponding overrun request when an overrun occurs on the corresponding IRQ*n* pin. 0 Overrun request is disabled. 1 Overrun request is enabled.

6.3.1.9 IRQ Rising-Edge Event Enable Register (SIU_IREER)

The SIU_IREER allows rising edge triggered events to be enabled on the corresponding IRQ*n* pins. Rising and falling edge events can be enabled by setting the corresponding bits in both the SIU_IREER and SIU_IFEER.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x28															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IREE 15	IREE 14	IREE 13	IREE 12	IREE 11	IREE 10	IREE 9	IREE 8	IREE 7	IREE 6	IREE 5	IREE 4	IREE 3	IREE 2	IREE 1	IREE 0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x28															

Figure 6-10. IRQ Rising-Edge Event Enable Register (SIU_IREER)

Table 6-12. SIU_IREER Field Descriptions

Bits	Name	Function
0–15	—	Reserved.
16–31	IREE*n*	IRQ rising-edge event enable *n.* Enables rising-edge triggered events on the corresponding IRQ*n* pin. 0 Rising edge event is disabled. 1 Rising edge event is enabled.

6.3.1.10 IRQ Falling-Edge Event Enable Register (SIU_IFEER)

The SIU_IFEER allows falling edge triggered events to be enabled on the corresponding IRQ*n* pins. Rising and falling edge events can be enabled by setting the corresponding bits in both the SIU_IREER and SIU_IFEER.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x2C															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IFEE 15	IFEE 14	IREE 13	IFEE 12	IFEE 11	IFEE 10	IFEE 9	IFEE 8	IFEE 7	IFEE 6	IFEE 5	IFEE 4	IFEE 3	IFEE 2	IFEE 1	IFEE 0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x2C															

Figure 6-11. IRQ Falling-Edge Event Enable Register (SIU_IFEER)

Table 6-13. SIU_IFEER Field Descriptions

Bits	Name	Function
0–15	—	Reserved.
16–31	IFEE*n*	IRQ falling-edge event enable *n*. Enables falling-edge triggered events on the corresponding IRQ*n* pin. 0 Falling edge event is disabled. 1 Falling edge event is enabled.

6.3.1.11 IRQ Digital Filter Register (SIU_IDFR)

The SIU_IDFR specifies the amount of digital filtering on the IRQ0–IRQ15 pins. The digital filter length field specifies the number of system clocks that define the period of the digital filter and the minimum time a signal must be held in the active state on the IRQ pins to be recognized as an edge triggered event.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x30															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	DFL			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x30															

Figure 6-12. External IRQ Digital Filter Register (SIU_IDFR)

Table 6-14. SIU_IDFR Field Descriptions

Bits	Name	Function
0–27	—	Reserved.
28–31	DFL	Digital filter length. Defines the digital filter period on the IRQ*n* inputs according to the following equation: $$\text{Filter Period} = (\text{SystemClockPeriod} \times 2^{\text{DFL}}) + 1(\text{SystemClockPeriod})$$ For a 100-MHz system clock, this gives a range of 20ns to 328us. The minimum time of three clocks accounts for synchronization of the IRQ input pins with the system clock.

6.3.1.12 Pad Configuration Registers (SIU_PCR)

The following subsections define the SIU_PCRs for all device pins that allow configuration of the pin function, direction, and static electrical attributes. The information presented pertains to which bits and fields are active for a given pin or group of pins, and the reset state of the register. Note that the reset state of SIU_PCRs given in the following sections is that prior to execution of the BAM program. The BAM program may change certain SIU_PCRs based on the reset configuration. See the BAM section of the manual for more detail.

The SIU_PCRs are 16-bit registers that may be read or written as 16-bit values aligned on 16-bit boundaries, or as 32-bit values aligned on 32-bit address boundaries. Table 6-15 describes the SIU_PCR fields.

NOTE

Not all of the fields may be present in a given SIU_PCR, depending on the type of pad it controls. See the specific SIU_PCR definition.

All MPC5553/MPC5554 pin names begin with the primary function, followed by the alternate function, and then GPIO. In some cases the third function may not be GPIO. Those exceptions are noted in the documentation. For example, for SIU_PCR85 and the pin CNTXB_PCSC3_GPIO85, CNTXB is the primary function and PCSC3 is the alternate function. For identification of the source module for primary and alternate functions, and the description of these signals, see Chapter 2, “Signal Description” of this manual. Also see the chapter of the specific module that uses the signal for an additional signal description.

Table 6-15. SIU_PCR Field Descriptions

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>0–2</td><td>—</td><td>Reserved.</td></tr>
<tr><td>3–5</td><td>PA
[0:2]</td><td>Pin assignment. Selects the function of a multiplexed pad. A separate port enable output signal from the SIU is asserted for each value of this register. The size of the field can be from 1 to 3 bits, depending on the amount of multiplexing on the pad
<table>
<tr><th colspan="9">PA Bit Field</th><th rowspan="2">Pin Function[1]</th></tr>
<tr><th colspan="3">1-bit[2]
(2 Functions)</th><th colspan="3">2-bit[2]
(3 Functions)</th><th colspan="3">3-bit
(4 Functions)</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>Primary Function</td></tr>
<tr><td rowspan="7" colspan="3"></td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>Alternate Function 1</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>Primary Function</td></tr>
<tr><td rowspan="5" colspan="3"></td><td>1</td><td>0</td><td>0</td><td>Alternate Function 2</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Reserved</td></tr>
</table>
[1] For all SIU_PCRs that do not comply with these rules, the PA definition is given explicitly with the SIU_PCR definition.
[2] For future software compatibility, it is recommended that all PA fields be treated as 3-bit fields, with the unused bits written as 0.</td></tr>
<tr><td>6</td><td>OBE</td><td>Output buffer enable. Enables the pad as an output and drives the output buffer enable signal.
0 Output buffer for the pad is disabled.
1 Output buffer for the pad is enabled.</td></tr>
<tr><td>7</td><td>IBE</td><td>Input buffer enable. Enables the pad as an input and drives the input buffer enable signal.
0 Input buffer for the pad is disabled.
1 Input buffer for the pad is enabled.</td></tr>
<tr><td>8–9</td><td>DSC
[0:1]</td><td>Drive strength control. Controls the pad drive strength. Drive strength control pertains to pins with the fast I/O pad type.
00 10 pF Drive Strength
01 20 pF Drive Strength
10 30 pF Drive Strength
11 50 pF Drive Strength</td></tr>
<tr><td>10</td><td>ODE</td><td>Open drain output enable. Controls output driver configuration for the pads. Either open drain or push/pull driver configurations can be selected. This feature applies to output pins only.
0 Open drain is disabled for the pad (push/pull driver enabled).
1 Open drain is enabled for the pad.</td></tr>
<tr><td>11</td><td>HYS</td><td>Input hysteresis. Controls whether hysteresis is enabled for the pad.
0 Hysteresis is disabled for the pad.
1 Hysteresis is enabled for the pad.</td></tr>
</table>

Table 6-15. SIU_PCR Field Descriptions (continued)

Bits	Name	Description
12–13	SRC [0:1]	Slew rate control. Controls slew rate for the pad. Slew rate control pertains to pins with slow or medium I/O pad types, and the output signals are driven according to the value of this field. Actual slew rate is dependent on the pad type and load. See the electrical specification for this information 00 Minimum slew rate 01 Medium slew rate 10 Reserved 11 Maximum slew rate
14	WPE	Weak pull up/down enable. Controls whether the weak pull up/down devices are enabled/disabled for the pad. Pull up/down devices are enabled by default. 0 Weak pull device is disabled for the pad. 1 Weak pull device is enabled for the pad.
15	WPS	Weak pull up/down select. Controls whether weak pull up or weak pull down devices are used for the pad when weak pull up/down devices are enabled. The WKPCFG pin determines whether pull up or pull down devices are enabled at reset. The WPS bit determines whether weak pull up or pull down devices are used after reset, or for pads in which the WKPCFG pin does not determine the reset weak pull up/down state. 0 The pull down value is enabled for the pad. 1 The pull up value is enabled for the pad.

6.3.1.12.1 Pad Configuration Registers 0 - 3 (SIU_PCR0 - SIU_PCR3)

The SIU_PCR0 - SIU_PCR3 registers control the pin function, direction, and static electrical attributes of the $\overline{CS}$[0:3]_ADDR[8:11]_GPIO[0:3] pins.

SIU_BASE+0x40 - SIU_BASE+0x46 (4)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA[1]		OBE[2]	IBE[3]	DSC		ODE[4]	HYS	0	0	WPE[5]	WPS[5]
W																
RESET:	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1

1 The PA fields in PCR0 - 3 and PCR4 - 7 must not be configured simultaneously to select ADDR[8:11] as input. Only one pin is to be configured to provide the address input.

2 When configured as $\overline{CS}$[0:3] or ADDR[8:11], the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

3 When configured as $\overline{CS}$[0:3], ADDR[8:11] (only MPC5554), or GPI, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

4 When configured as $\overline{CS}$[0:3] or ADDR[8:11] (only MPC5554), the ODE bit should be set to zero.

5 See the EBI section for weak pull up settings when configured as $\overline{CS}$[0:3] or ADDR[8:11] (only MPC5554).

Figure 6-13. $\overline{CS}$[0:3]_ADDR[8:11]_GPIO[0:3] Pad Configuration Registers (SIU_PCR0 - SIU_PCR3)

See Table 6-15 for bit field definitions.

6.3.1.12.2 MPC5553: Pad Configuration Registers 4 - 7 (SIU_PCR4 - SIU_PCR7)

The SIU_PCR4 - SIU_PCR7 registers control the pin function, direction, and static electrical attributes of the ADDR[8:11]_CAL_ADDR[27:30]_GPIO[4:7] pins.

SIU_BASE+0x48 - SIU_BASE+0x4E (4)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	PA[1]			OBE[2]	IBE[3]	DSC		ODE[4]	HYS[5]	0	0	WPE[6]	WPS[6]
W																
RESET:	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1

1 The PA fields in PCR0 - 3 and PCR4 - 7 must not be configured simultaneously to select ADDR[8:11] as an input. Only one pin is to be configured to provide the address input.

2 When configured as ADDR[8:11] or CAL_ADDR[27:30], the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

3 When configured as ADDR[8:11], CAL_ADDR[27:30], or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

4 When configured as ADDR[8:11] or CAL_ADDR[27:30], the ODE bit should be set to zero.

5 If external master operation is enabled, the HYS bit should be set to zero.

6 See the EBI section for weak pull up settings when configured as ADDR[8:11] or CAL_ADDR[27:30].

Figure 6-14. MPC5553: ADDR[8:11]_CAL_ADDR[27:30]_GPIO[4:7] Pad Configuration Registers (SIU_PCR4 - SIU_PCR7)

See Table 6-15 for bit field definitions. The PA field for PCR4 - PCR7 is given in Table 6-16.

Table 6-16. PCR4 - PCR7 PA Field Definition

PA Field	Pin Function
0b000	GPIO[4:7]
0b001	ADDR[8:11]
0b010	Reserved
0b011	ADDR[8:11]
0b100	CAL_ADDR[27:30][1]

[1]For calibration only.

6.3.1.12.3 MPC5554: Pad Configuration Registers 4 - 27 (SIU_PCR4 - SIU_PCR27)

NOTE

The definition for PCR4–PCR7 in MPC5553 devices differs from the definition given in this section. For MPC5553 devices' PCR4–PCR7, see Section 6.3.1.12.2, "MPC5553: Pad Configuration Registers 4 - 7 (SIU_PCR4 - SIU_PCR7)."

The SIU_PCR4 - SIU_PCR27 registers control the pin function, direction, and static electrical attributes of the ADDR[8:31]_GPIO[4:27] pins (ADDR[12:31]_GPIO[8:27] for MPC5553).

SIU_BASE+0x48 - SIU_BASE+0x76 (24)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	PA[1]	OBE[2]	IBE[3]	DSC		ODE[4]	HYS[5]	0	0	WPE[6]	WPS[6]
W																
RESET:	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1

1 The PA fields in PCR0 - 3 and PCR4 - 7 must not be configured simultaneously to select ADDR[8:11]. Only one pin is configured to provide the address input.

2 When configured as ADDR[8:31] (ADDR[12;31] for MPC5553), the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

3 When configured as ADDR[8:31] (ADDR[12;31] for MPC5553) or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

4 When configured as ADDR[8:31] (ADDR[12;31] for MPC5553), the ODE bit should be set to zero.

5 If external master operation is enabled, the HYS bit should be set to zero.

6 See the EBI section for weak pull up settings when configured as ADDR[8:31].

Figure 6-15. MPC5554: ADDR[8:31]_GPIO[4:27] Pad Configuration Registers (SIU_PCR4 - SIU_PCR27)

See Table 6-15 for bit field definitions.

6.3.1.12.4 Pad Configuration Registers 28 - 59 (SIU_PCR28 - SIU_PCR59)

NOTE

The definitions for PCR44–PCR59 in MPC5553 devices differs from the definition given in this section. For MPC5553 devices' PCR44–PCR59, see Section 6.3.1.12.5, "MPC5553: Pad Configuration Register 44 (SIU_PCR44)" – Section 6.3.1.12.20, "MPC5553: Pad Configuration Register 59 (SIU_PCR59)

The SIU_PCR28 - SIU_PCR59 registers control the pin function, direction, and static electrical attributes of the DATA[0:31]_GPIO[28:59] pins.

SIU_BASE+0x78 - SIU_BASE+0xB6 (32)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	PA	OBE[1]	IBE[2]	DSC		ODE[3]	HYS[4]	0	0	WPE[5]	WPS[5]
W																
RESET:	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1

1 When configured as DATA[0:31], the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

2 When configured as DATA[0:31] or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

3 When configured as DATA[0:31], the ODE bit should be set to zero.

4 If external master operation is enabled, the HYS bit should be set to zero.

5 See the EBI section for weak pull up settings when configured as DATA[0:31].

Figure 6-16. DATA[0:31]_GPIO[28:59] Pad Configuration Registers (SIU_PCR28 - SIU_PCR59)

See Table 6-15 for bit field definitions.

6.3.1.12.5 MPC5553: Pad Configuration Register 44 (SIU_PCR44)

The SIU_PCR44 register controls the pin function, direction, and static electrical attributes of the DATA[16]_TX_CLK_CAL_DATA[0]_GPIO[44] pin.

SIU_BASE+0x98

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	PA[1]			OBE[2]	IBE[3]	DSC		ODE[4]	HYS[5]	0	0	WPE[6]	WPS[6]
W																
RESET:	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1

1 CAL_DATA[0] is for calibration only.

2 When configured as DATA[16], TX_CLK, or CAL_DATA[0], the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

3 When configured as DATA[16], TX_CLK, CAL_DATA[0], or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

4 When configured as DATA[16] or CAL_DATA[0], the ODE bit should be set to zero.

5 If external master operation is enabled, the HYS bit should be set to zero.

6 See the EBI section for weak pull up settings when configured as DATA[16] or CAL_DATA[0].

Figure 6-17. MPC5553: DATA[16]_TX_CLK_CAL_DATA[0]_GPIO[44] Pad Configuration Register (SIU_PCR44)

See Table 6-15 for bit field definitions.

6.3.1.12.6 MPC5553: Pad Configuration Register 45 (SIU_PCR45)

The SIU_PCR45 register controls the pin function, direction, and static electrical attributes of the DATA[17]_CRS_CAL_DATA[1]_GPIO[45] pin.

SIU_BASE+0x9A

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	PA[1]			OBE[2]	IBE[3]	DSC		ODE[4]	HYS[5]	0	0	WPE[6]	WPS[6]
W																
RESET:	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1

1 CAL_DATA[1] is for calibration only.

2 When configured as DATA[17], CRS, or CAL_DATA[1], the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

3 When configured as DATA[17], CRS, CAL_DATA[1], or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

4 When configured as DATA[17] or CAL_DATA[1], the ODE bit should be set to zero.

5 If external master operation is enabled, the HYS bit should be set to zero.

6 See the EBI section for weak pull up settings when configured as DATA[17] or CAL_DATA[1].

Figure 6-18. MPC5553: DATA[17]_CRS_CAL_DATA[1]_GPIO[45] Pad Configuration Register (SIU_PCR45)

See Table 6-15 for bit field definitions.

6.3.1.12.7 MPC5553: Pad Configuration Register 46 (SIU_PCR46)

The SIU_PCR46 register controls the pin function, direction, and static electrical attributes of the DATA[18]_TX_ERR_CAL_DATA[2]_GPIO[46] pin.

SIU_BASE+0x9C

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	PA[1]			OBE[2]	IBE[3]	DSC		ODE[4]	HYS[5]	0	0	WPE[6]	WPS[6]
W																
RESET:	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1

1 CAL_DATA[2] is for calibration only.

2 When configured as DATA[18], TX_ERR, or CAL_DATA[2], the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

3 When configured as DATA[18], TX_ERR, CAL_DATA[2], or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

4 When configured as DATA[18] or CAL_DATA[2], the ODE bit should be set to zero.

5 If external master operation is enabled, the HYS bit should be set to zero.

6 See the EBI section for weak pull up settings when configured as DATA[18] or CAL_DATA[2].

Figure 6-19. MPC5553: DATA[18]_TX_ERR_CAL_DATA[2]_GPIO[46] Pad Configuration Register (SIU_PCR46)

See Table 6-15 for bit field definitions.

6.3.1.12.8 MPC5553: Pad Configuration Register 47 (SIU_PCR47)

The SIU_PCR47 register controls the pin function, direction, and static electrical attributes of the DATA[19]_RX_CLK_CAL_DATA[3]_GPIO[47] pin.

SIU_BASE+0x9E

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	PA[1]			OBE[2]	IBE[3]	DSC		ODE[4]	HYS[5]	0	0	WPE[6]	WPS[6]
W																
RESET:	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1

1 CAL_DATA[3] is for calibration only.

2 When configured as DATA[19], RX_CLK, or CAL_DATA[3], the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

3 When configured as DATA[19], RX_CLK, CAL_DATA[3], or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

4 When configured as DATA[19] or CAL_DATA[3], the ODE bit should be set to zero.

5 If external master operation is enabled, the HYS bit should be set to zero.

6 See the EBI section for weak pull up settings when configured as DATA[19] or CAL_DATA[3].

Figure 6-20. MPC5553: DATA[19]_RX_CLK_CAL_DATA[3]_GPIO[47] Pad Configuration Register (SIU_PCR47)

See Table 6-15 for bit field definitions.

6.3.1.12.9 MPC5553: Pad Configuration Register 48 (SIU_PCR48)

The SIU_PCR48 register controls the pin function, direction, and static electrical attributes of the DATA[20]_TXD[0]_CAL_DATA[4]_GPIO[48] pin.

SIU_BASE+0xA0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	PA[1]			OBE[2]	IBE[3]	DSC		ODE[4]	HYS[5]	0	0	WPE[6]	WPS[6]
W																
RESET:	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1

1 CAL_DATA[4] is for calibration only.

2 When configured as DATA[20], TXD[0], or CAL_DATA[4], the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

3 When configured as DATA[20], TXD[0], CAL_DATA[4], or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

4 When configured as DATA[20] or CAL_DATA[4], the ODE bit should be set to zero.

5 If external master operation is enabled, the HYS bit should be set to zero.

6 See the EBI section for weak pull up settings when configured as DATA[20] or CAL_DATA[4].

Figure 6-21. MPC5553: DATA[20]_TXD[0]_CAL_DATA[4]_GPIO[48] Pad Configuration Register (SIU_PCR48)

See Table 6-15 for bit field definitions.

6.3.1.12.10 MPC5553: Pad Configuration Register 49 (SIU_PCR49)

The SIU_PCR49 register controls the pin function, direction, and static electrical attributes of the DATA[21]_RX_ERR_CAL_DATA[5]_GPIO[49] pin.

SIU_BASE+0xA2

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	PA[1]			OBE[2]	IBE[3]	DSC		ODE[4]	HYS[5]	0	0	WPE[6]	WPS[6]
W																
RESET:	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1

1 CAL_DATA[5] is for calibration only.

2 When configured as DATA[21], RX_ERR, or CAL_DATA[5], the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

3 When configured as DATA[21], RX_ERR, CAL_DATA[5], or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

4 When configured as DATA[21] or CAL_DATA[5], the ODE bit should be set to zero.

5 If external master operation is enabled, the HYS bit should be set to zero.

6 See the EBI section for weak pull up settings when configured as DATA[21] or CAL_DATA[5].

Figure 6-22. MPC5553: DATA[21]_RX_ERR_CAL_DATA[5]_GPIO[49] Pad Configuration Registers (SIU_PCR49)

See Table 6-15 for bit field definitions.

6.3.1.12.11 MPC5553: Pad Configuration Register 50 (SIU_PCR50)

The SIU_PCR50 register controls the pin function, direction, and static electrical attributes of the DATA[22]_RXD[0]_CAL_DATA[6]_GPIO[50] pin.

SIU_BASE+0xA4

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	PA[1]			OBE[2]	IBE[3]	DSC		ODE[4]	HYS[5]	0	0	WPE[6]	WPS[6]
W																
RESET:	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1

1 CAL_DATA[6] is for calibration only.

2 When configured as DATA[22], RXD[0], or CAL_DATA[6], the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

3 When configured as DATA[22], RXD[0], CAL_DATA[6], or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

4 When configured as DATA[22] or CAL_DATA[6], the ODE bit should be set to zero.

5 If external master operation is enabled, the HYS bit should be set to zero.

6 See the EBI section for weak pull up settings when configured as DATA[22] or CAL_DATA[6].

Figure 6-23. MPC5553: DATA[22]_RXD[0]_CAL_DATA[6]_GPIO[50] Pad Configuration Register (SIU_PCR50)

See Table 6-15 for bit field definitions.

6.3.1.12.12 MPC5553: Pad Configuration Register 51 (SIU_PCR51)

The SIU_PCR51 register controls the pin function, direction, and static electrical attributes of the DATA[23]_TXD[3]_CAL_DATA[7]_GPIO[51] pin.

SIU_BASE+0xA6

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	PA[1]			OBE[2]	IBE[3]	DSC		ODE[4]	HYS[5]	0	0	WPE[6]	WPS[6]
W																
RESET:	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1

1 CAL_DATA[7] is for calibration only.

2 When configured as DATA[23], TXD[3], or CAL_DATA[7], the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

3 When configured as DATA[23], TXD[3], CAL_DATA[7], or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

4 When configured as DATA[23] or CAL_DATA[7], the ODE bit should be set to zero.

5 If external master operation is enabled, the HYS bit should be set to zero.

6 See the EBI section for weak pull up settings when configured as DATA[23] or CAL_DATA[7].

Figure 6-24. MPC5553: DATA[23]_TXD[3]_CAL_DATA[7]_GPIO[51] Pad Configuration Register (SIU_PCR51)

See Table 6-15 for bit field definitions.

6.3.1.12.13 MPC5553: Pad Configuration Register 52 (SIU_PCR52)

The SIU_PCR52 register controls the pin function, direction, and static electrical attributes of the DATA[24]_COL_CAL_DATA[8]_GPIO[52] pin.

SIU_BASE+0xA8

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	PA[1]			OBE[2]	IBE[3]	DSC		ODE[4]	HYS[5]	0	0	WPE[6]	WPS[6]
W																
RESET:	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1

1 CAL_DATA[8] is for calibration only.

2 When configured as DATA[24], COL, or CAL_DATA[8], the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

3 When configured as DATA[24], COL, CAL_DATA[8], or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

4 When configured as DATA[24] or CAL_DATA[8], the ODE bit should be set to zero.

5 If external master operation is enabled, the HYS bit should be set to zero.

6 See the EBI section for weak pull up settings when configured as DATA[24] or CAL_DATA[8].

Figure 6-25. MPC5553: DATA[24]_COL_CAL_DATA[8]_GPIO[52] Pad Configuration Register (SIU_PCR52)

See Table 6-15 for bit field definitions.

6.3.1.12.14 MPC5553: Pad Configuration Register 53 (SIU_PCR53)

The SIU_PCR53 register controls the pin function, direction, and static electrical attributes of the DATA[25]_RX_DV_CAL_DATA[9]_GPIO[53] pin.

SIU_BASE+0xAA

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	PA[1]			OBE[2]	IBE[3]	DSC		ODE[4]	HYS[5]	0	0	WPE[6]	WPS[6]
W																
RESET:	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1

1 CAL_DATA[9] is for calibration only.

2 When configured as DATA[25], RX_DV, or CAL_DATA[9], the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

3 When configured as DATA[25], RX_DV, CAL_DATA[9], or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

4 When configured as DATA[25] or CAL_DATA[9], the ODE bit should be set to zero.

5 If external master operation is enabled, the HYS bit should be set to zero.

6 See the EBI section for weak pull up settings when configured as DATA[25] or CAL_DATA[9].

Figure 6-26. MPC5553: DATA[25]_RX_DV_CAL_DATA[9]_GPIO[53] Pad Configuration Register (SIU_PCR53)

See Table 6-15 for bit field definitions.

6.3.1.12.15 MPC5553: Pad Configuration Register 54 (SIU_PCR54)

The SIU_PCR54 register controls the pin function, direction, and static electrical attributes of the DATA[26]_TX_EN_CAL_DATA[10]_GPIO[54] pin.

SIU_BASE+0xAC

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	PA[1]			OBE[2]	IBE[3]	DSC		ODE[4]	HYS[5]	0	0	WPE[6]	WPS[6]
W																
RESET:	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1

1 CAL_DATA[10] is for calibration only.

2 When configured as DATA[26], TX_EN, or CAL_DATA[10], the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

3 When configured as DATA[26], TX_EN, CAL_DATA[10], or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

4 When configured as DATA[26] or CAL_DATA[10], the ODE bit should be set to zero.

5 If external master operation is enabled, the HYS bit should be set to zero.

6 See the EBI section for weak pull up settings when configured as DATA[26] or CAL_DATA[10].

Figure 6-27. MPC5553: DATA[26]_TX_EN_CAL_DATA[10]_GPIO[54] Pad Configuration Register (SIU_PCR54)

See Table 6-15 for bit field definitions.

6.3.1.12.16 MPC5553: Pad Configuration Register 55 (SIU_PCR55)

The SIU_PCR55 register controls the pin function, direction, and static electrical attributes of the DATA[27]_TXD[2]_CAL_DATA[11]_GPIO[55] pin.

SIU_BASE+0xAE

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	PA[1]			OBE[2]	IBE[3]	DSC		ODE[4]	HYS[5]	0	0	WPE[6]	WPS[6]
W																
RESET:	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1

1 CAL_DATA[11] is for calibration only.

2 When configured as DATA[27], TXD[2], or CAL_DATA[11], the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

3 When configured as DATA[27], TXD[2], CAL_DATA[11], or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

4 When configured as DATA[27] or CAL_DATA[11], the ODE bit should be set to zero.

5 If external master operation is enabled, the HYS bit should be set to zero.

6 See the EBI section for weak pull up settings when configured as DATA[27] or CAL_DATA[11].

Figure 6-28. MPC5553: DATA[27]_TXD[2]_CAL_DATA[11]_GPIO[55] Pad Configuration Register (SIU_PCR55)

See Table 6-15 for bit field definitions.

6.3.1.12.17 MPC5553: Pad Configuration Register 56 (SIU_PCR56)

The SIU_PCR56 register controls the pin function, direction, and static electrical attributes of the DATA[28]_TXD[1]_CAL_DATA[12]_GPIO[56] pin.

SIU_BASE+0xB0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	PA[1]			OBE[2]	IBE[3]	DSC		ODE[4]	HYS[5]	0	0	WPE[6]	WPS[6]
W																
RESET:	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1

1 CAL_DATA[12] is for calibration only.

2 When configured as DATA[28], TXD[1], or CAL_DATA[12], the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

3 When configured as DATA[28], TXD[1], CAL_DATA[12], or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

4 When configured as DATA[28] or CAL_DATA[12], the ODE bit should be set to zero.

5 If external master operation is enabled, the HYS bit should be set to zero.

6 See the EBI section for weak pull up settings when configured as DATA[28] or CAL_DATA[12].

Figure 6-29. MPC5553: DATA[28]_TXD[1]_CAL_DATA[12]_GPIO[56] Pad Configuration Register (SIU_PCR56)

See Table 6-15 for bit field definitions.

6.3.1.12.18 MPC5553: Pad Configuration Register 57 (SIU_PCR57)

The SIU_PCR57 register controls the pin function, direction, and static electrical attributes of the DATA[29]_RXD[1]_CAL_DATA[13]_GPIO[57] pin.

SIU_BASE+0xB2

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	PA[1]			OBE[2]	IBE[3]	DSC		ODE[4]	HYS[5]	0	0	WPE[6]	WPS[6]
W																
RESET:	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1

1 CAL_DATA[13] is for calibration only.

2 When configured as DATA[29], RXD[1], or CAL_DATA[13], the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

3 When configured as DATA[29], RXD[1], CAL_DATA[13], or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

4 When configured as DATA[29] or CAL_DATA[13], the ODE bit should be set to zero.

5 If external master operation is enabled, the HYS bit should be set to zero.

6 See the EBI section for weak pull up settings when configured as DATA[29] or CAL_DATA[13].

Figure 6-30. MPC5553: DATA[29]_RXD[1]_CAL_DATA[13]_GPIO[57] Pad Configuration Register (SIU_PCR57)

See Table 6-15 for bit field definitions.

6.3.1.12.19 MPC5553: Pad Configuration Register 58 (SIU_PCR58)

The SIU_PCR58 register controls the pin function, direction, and static electrical attributes of the DATA[30]_RXD[2]_CAL_DATA[14]_GPIO[58] pin.

SIU_BASE+0xB4

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	PA[1]			OBE[2]	IBE[3]	DSC		ODE[4]	HYS[5]	0	0	WPE[6]	WPS[6]
W																
RESET:	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1

1 CAL_DATA[14] is for calibration only.

2 When configured as DATA[30], RXD[2], or CAL_DATA[14], the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

3 When configured as DATA[30], RXD[2], CAL_DATA[14], or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

4 When configured as DATA[30] or CAL_DATA[14], the ODE bit should be set to zero.

5 If external master operation is enabled, the HYS bit should be set to zero.

6 See the EBI section for weak pull up settings when configured as DATA[30] or CAL_DATA[14].

Figure 6-31. MPC5553: DATA[30]_RXD[2]_CAL_DATA[14]_GPIO[58] Pad Configuration Register (SIU_PCR58)

See Table 6-15 for bit field definitions.

6.3.1.12.20 MPC5553: Pad Configuration Register 59 (SIU_PCR59)

The SIU_PCR59 register controls the pin function, direction, and static electrical attributes of the DATA[31]_RXD[3]_CAL_DATA[15]_GPIO[59] pin.

SIU_BASE+0xB6

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	PA[1]			OBE[2]	IBE[3]	DSC		ODE[4]	HYS[5]	0	0	WPE[6]	WPS[6]
W																
RESET:	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1

1 CAL_DATA[15] is for calibration only.

2 When configured as DATA[31], RXD[3], or CAL_DATA[15], the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

3 When configured as DATA[31], RXD[3], CAL_DATA[15], or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

4 When configured as DATA[31] or CAL_DATA[15], the ODE bit should be set to zero.

5 If external master operation is enabled, the HYS bit should be set to zero.

6 See the EBI section for weak pull up settings when configured as DATA[31] or CAL_DATA[15].

Figure 6-32. MPC5553: DATA[31]_RXD[3]_CAL_DATA[15]_GPIO[59] Pad Configuration Register (SIU_PCR59)

See Table 6-15 for bit field definitions.

6.3.1.12.21 MPC5554: Pad Configuration Registers 60 - 61 (SIU_PCR60 - SIU_PCR61)

NOTE

The MPC5553 does not implement PCRs 60–61. Treat these registers as reserved space.

The SIU_PCR60 - SIU_PCR61 registers control the pin function, direction, and static electrical attributes of the TSIZ[0:1]_GPIO[60:61] pins.

SIU_BASE+0xB8 - SIU_BASE+0xBA (2)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	PA	OBE[1]	IBE[2]	DSC		ODE[3]	HYS[4]	0	0	WPE[5]	WPS[5]
W																
RESET:	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1

1 When configured as TSIZ[0:1], the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

2 When configured as TSIZ[0:1] or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

3 When configured as TSIZ[0:1], the ODE bit should be set to zero.

4 If external master operation is enabled, the HYS bit should be set to zero.

5 See the EBI section for weak pull up settings when configured as TSIZ[0:1].

Figure 6-33. MPC5554: TSIZ[0:1]_GPIO[60:61] Pad Configuration Registers (SIU_PCR60 - SIU_PCR61)

See Table 6-15 for bit field definitions.

6.3.1.12.22 Pad Configuration Register 62 (SIU_PCR62)

The SIU_PCR62 register controls the pin function, direction, and static electrical attributes of the RD_$\overline{WR}$_GPIO[62] pin.

SIU_BASE+0xBC

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	PA	OBE[1]	IBE[2]	DSC		ODE[3]	HYS[4]	0	0	WPE[5]	WPS[5]
W																
RESET:	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1

1 When configured as RD_$\overline{WR}$, the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

2 When configured as RD_$\overline{WR}$ or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

3 When configured as RD_$\overline{WR}$, the ODE bit should be set to zero.

4 If external master operation is enabled, the HYS bit should be set to zero.

5 See the EBI section for weak pull up settings when configured as RD_$\overline{WR}$.

Figure 6-34. RD_$\overline{WR}$_GPIO[62] Pad Configuration Register (SIU_PCR62)

See Table 6-15 for bit field definitions.

6.3.1.12.23 Pad Configuration Register 63 (SIU_PCR63)

The SIU PCR63 register controls the pin function, direction, and static electrical attributes of the BDIP_GPIO[63] pin.

SIU_BASE+0xBE

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	PA	OBE[1]	IBE[2]	DSC		ODE[3]	HYS	0	0	WPE[4]	WPS[4]
W																
RESET:	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1

1 When configured as $\overline{\text{BDIP}}$, the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

2 When configured as $\overline{\text{BDIP}}$ or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

3 When configured as $\overline{\text{BDIP}}$, the ODE bit should be set to zero.

4 See the EBI section for weak pull up settings when configured as $\overline{\text{BDIP}}$.

Figure 6-35. $\overline{\text{BDIP}}$_GPIO[63] Pad Configuration Register (SIU_PCR63)

See Table 6-15 for bit field definitions.

6.3.1.12.24 MPC5553: Pad Configuration Registers 64 - 65 (SIU_PCR64 - SIU_PCR65)

The SIU_PCR64 - SIU_PCR65 registers control the pin function, direction, and static electrical attributes of the $\overline{\text{WE}}$[0:1]_$\overline{\text{BE}}$[0:1]_GPIO[64:65] pins. Note that the PA bit in the PCR64 - 65 registers selects between the write enable/byte enable and GPIO functions. The WEBS bit in the ebi base registers selects between the write enable and byte enable function.

SIU_BASE+0xC0 - SIU_BASE+0xC2 (2)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	PA	OBE[1]	IBE[2]	DSC		ODE[3]	HYS	0	0	WPE[4]	WPS[4]
W																
RESET:	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1

1 When configured as $\overline{\text{WE}}$[0:1] or $\overline{\text{BE}}$[0:1], the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

2 When configured as $\overline{\text{WE}}$[0:1] or $\overline{\text{BE}}$[0:1] or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

3 When configured as $\overline{\text{WE}}$[0:1] or $\overline{\text{BE}}$[0:1], the ODE bit should be set to zero.

4 See the EBI section for weak pull up settings when configured as $\overline{\text{WE}}$[0:1] or $\overline{\text{BE}}$[0:1].

Figure 6-36. MPC5553: $\overline{\text{WE}}$[0:1]_$\overline{\text{BE}}$[0:1]_GPIO[64:65] Pad Configuration Registers (SIU_PCR64 - SIU_PCR65)

See Table 6-15 for bit field definitions.

6.3.1.12.25 MPC5553: Pad Configuration Registers 66 - 67 (SIU_PCR66 - SIU_PCR67)

The SIU_PCR66 - SIU_PCR67 registers control the pin function, direction, and static electrical attributes of the $\overline{\text{WE}}$[2:3]_$\overline{\text{BE}}$[2:3]_CAL_$\overline{\text{WE}}$[0:1]_CAL_$\overline{\text{BE}}$[0:1]_GPIO[66:67] pins. Note that the PA bit in the

PCR66 - 67 registers selects between the write enable/byte enable and GPIO functions. The WEBS bit in the EBI base registers selects between the write enable and byte enable function.

SIU_BASE+0xC4 - SIU_BASE+0xC6 (2)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	PA			OBE[1]	IBE[2]	DSC		ODE[3]	HYS	0	0	WPE[4]	WPS[4]
W																
RESET:	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1

1 When configured as $\overline{WE}$[2:3], $\overline{BE}$[2:3], $\overline{CAL\_WE}$[0:1], or $\overline{CAL\_BE}$[0:1], the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

2 When configured as $\overline{WE}$[2:3], $\overline{BE}$[2:3], $\overline{CAL\_WE}$[0:1], $\overline{CAL\_BE}$[0:1], or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

3 When configured as $\overline{WE}$[2:3], $\overline{BE}$[2:3], $\overline{CAL\_WE}$[0:1], or $\overline{CAL\_BE}$[0:1], the ODE bit should be set to zero.

4 See the EBI section for weak pull up settings when configured as $\overline{WE}$[2:3], $\overline{BE}$[2:3], $\overline{CAL\_WE}$[0:1], or $\overline{CAL\_BE}$[0:1].

Figure 6-37. MPC5553: $\overline{WE}$[2:3]_$\overline{BE}$[2:3]_$\overline{CAL_WE}$[0:1]_$\overline{CAL_BE}$[0:1]_GPIO[66:67] Pad Configuration Registers (SIU_PCR66 - SIU_PCR67)

See Table 6-15 for bit field definitions. The PA field for the MPC5553's PCR66 - PCR67 is given in Table 6-17.

Table 6-17. MPC5553: PCR66 - PCR77 PA Field Definition

PA Field	Pin Function
0b000	GPIO[66:67]
0b001	$\overline{WE}$[2:3]_$\overline{BE}$[2:3]
0b010	Reserved
0b011	$\overline{WE}$[2:3]_$\overline{BE}$[2:3]
0b100	$\overline{CAL\_WE}$[0:1]_$\overline{CAL_BE}$[0:1][1]

1 For calibration only.

6.3.1.12.26 MPC5554: Pad Configuration Registers 64 - 67 (SIU_PCR64 - SIU_PCR67)

The SIU_PCR64 - SIU_PCR67 registers control the pin function, direction, and static electrical attributes of the $\overline{WE}$[0:3]_$\overline{BE}$[0:3]_GPIO[64:67] pins. Note that the PA bit in the PDMCR64 -67 registers selects between the write enable/byte enable and GPIO functions. The WEBS bit in the EBI base registers selects between the write enable and byte enable function.

SIU_BASE+0xC0 - SIU_BASE+0xC6 (4)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	PA	OBE[1]	IBE[2]	DSC		ODE[3]	HYS	0	0	WPE[4]	WPS[4]
W																
RESET:	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1

1 When configured as $\overline{WE}$[0:3] or $\overline{BE}$[0:3], the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

2 When configured as $\overline{WE}$[0:3] or $\overline{BE}$[0:3] or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

3 When configured as $\overline{WE}$[0:3] or $\overline{BE}$[0:3], the ODE bit should be set to zero.

4 See the EBI section for weak pull up settings when configured as $\overline{WE}$[0:3] or $\overline{BE}$[0:3].

Figure 6-38. MPC5554: $\overline{WE}$[0:3]_$\overline{BE}$[0:3]_GPIO[64:67] Pad Configuration Registers (SIU_PCR64 - SIU_PCR67)

See Table 6-15 for bit field definitions.

6.3.1.12.27 Pad Configuration Register 68 (SIU_PCR68)

The SIU_PCR68 register controls the pin function, direction, and static electrical attributes of the $\overline{OE}$_GPIO[68] pin. The $\overline{OE}$ function is not available in the 208 MAP BGA package. Only the GPIO function is available on this pin in the 208 package.

SIU_BASE+0xC8

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	PA	OBE[1]	IBE[2]	DSC		ODE[3]	HYS[4]	0	0	WPE[5]	WPS[5]
W																
RESET:	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1

1 When configured as $\overline{OE}$, the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

2 When configured as $\overline{OE}$ or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

3 When configured as $\overline{OE}$, the ODE bit should be set to zero.

4 If external master operation is enabled, the HYS bit should be set to one.

5 See the EBI section for weak pull up settings when configured as $\overline{OE}$.

Figure 6-39. $\overline{OE}$_GPIO[68] Pad Configuration Register (SIU_PCR68)

See Table 6-15 for bit field definitions.

6.3.1.12.28 Pad Configuration Register 69 (SIU_PCR69)

The SIU_PCR69 register controls the pin function, direction, and static electrical attributes of the TS_GPIO[69] pin.

SIU_BASE+0xCA

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	PA	OBE[1]	IBE[2]	DSC		ODE[3]	HYS[4]	0	0	WPE[5]	WPS[5]
W																
RESET:	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1

1 When configured as $\overline{TS}$, the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

2 When configured as $\overline{TS}$ or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

3 When configured as $\overline{TS}$, the ODE bit should be set to zero.

4 If external master operation is enabled, the HYS bit should be set to zero.

5 See the EBI section for weak pull up settings when configured as $\overline{TS}$.

Figure 6-40. $\overline{TS}$_GPIO[69] Pad Configuration Register (SIU_PCR69)

See Table 6-15 for bit field definitions.

6.3.1.12.29 Pad Configuration Register 70 (SIU_PCR70)

The SIU_PCR70 register controls the pin function, direction, and static electrical attributes of the $\overline{TA}$_GPIO[70] pin.

SIU_BASE+0xCC

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	PA	OBE[1]	IBE[2]	DSC		ODE[3]	HYS[4]	0	0	WPE[5]	WPS[5]
W																
RESET:	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1

1 When configured as $\overline{TA}$, the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

2 When configured as $\overline{TA}$, or GPIO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

3 When configured as $\overline{TA}$ and external master operation is enabled, the ODE bit should be set to zero.

4 If external master operation is enabled, the HYS bit should be set to zero.

5 See the EBI section for weak pull up settings when configured as $\overline{TA}$.

Figure 6-41. $\overline{TA}$_GPIO[70] Pad Configuration Register (SIU_PCR70)

See Table 6-15 for bit field definitions.

6.3.1.12.30 MPC5553: Pad Configuration Register 71 (SIU_PCR71)

The SIU_PCR71 register controls the pin function, direction, and static electrical attributes of the TEA_CAL_CS[0]_GPIO[71] pin.

SIU_BASE+0xCE

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	PA			OBE[1]	IBE[2]	DSC		ODE[3]	HYS[4]	0	0	WPE[5]	WPS[5]
W																
RESET:	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1

1 When configured as $\overline{\text{TEA}}$ or $\overline{\text{CAL\_CS}}$[0], the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

2 When configured as $\overline{\text{TEA}}$, $\overline{\text{CAL\_CS}}$[0], or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

3 When configured as $\overline{\text{TEA}}$ and external master operation is enabled, the ODE bit should be set to zero.

4 If external master operation is enabled, the HYS bit should be set to zero.

5 See the EBI section for weak pull up settings when configured as $\overline{\text{TEA}}$ or $\overline{\text{CAL\_CS}}$[0].

Figure 6-42. MPC5553: $\overline{\text{TEA}}$_$\overline{\text{CAL_CS}}$[0]_GPIO[71] Pad Configuration Register (SIU_PCR71)

See Table 6-15 for bit field definitions. The PA field for the MPC5553's PCR71 is given in Table 6-18.

Table 6-18. PCR71 PA Field Definition

PA Field	Pin Function
0b000	GPIO[71]
0b001	$\overline{\text{TEA}}$
0b010	Reserved
0b011	$\overline{\text{TEA}}$
0b100	$\overline{\text{CAL\_CS}}$[0][1]

1 For calibration only.

6.3.1.12.31 MPC5554: Pad Configuration Register 71 (SIU_PCR71)

The SIU_PCR71 register controls the pin function, direction, and static electrical attributes of the $\overline{\text{TEA}}$_GPIO[71] pin.

SIU_BASE+0xCE

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	PA	OBE[1]	IBE[2]	DSC		ODE[3]	HYS[4]	0	0	WPE[5]	WPS[5]
W																
RESET:	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1

1 When configured as $\overline{\text{TEA}}$, the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

2 When configured as $\overline{\text{TEA}}$ or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

3 When configured as $\overline{\text{TEA}}$ and external master operation is enabled, the ODE bit should be set to zero.

4 If external master operation is enabled, the HYS bit should be set to zero.

5 See the EBI section for weak pull up settings when configured as $\overline{\text{TEA}}$.

Figure 6-43. MPC5554: $\overline{\text{TEA}}$_GPIO[71] Pad Configuration Register (SIU_PCR71)

See Table 6-15 for bit field definitions.

6.3.1.12.32 MPC5553: Pad Configuration Register 72 (SIU_PCR72)

The SIU_PCR72 register controls the pin function, direction, and static electrical attributes of the BR(CAL_ADDR[10])_MDC_CAL_CS[2]_GPIO[72] pin. The BR function is not available on the MPC5554. Instead, its PA encoding is used for CAL_ADDR[10]. This register allows selection of the CAL_ADDR[10], MDC, CAL_CS[2], and GPIO functions.

SIU_BASE+0xD0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	PA[1]			OBE[2]	IBE[3]	DSC		ODE[4]	HYS[5]	0	0	WPE[6]	WPS[6]
W																
RESET:	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1

1 The BR function is not available on the MPC5554. Do not select 0b001 or 0b011 for the PA field except for CAL_ADDR[10].

2 When configured as CAL_ADDR[10], MDC, or CAL_CS[2], the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

3 When configured as CAL_ADDR[10], MDC, CAL_CS[2], or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

4 When configured as CAL_ADDR[10] or CAL_CS[2], the ODE bit should be set to zero.

5 If external master operation is enabled, the HYS bit should be set to zero.

6 See the EBI section for weak pull up settings when configured as CAL_ADDR[10] or CAL_CS[2].

Figure 6-44. MPC5553: BR(CAL_ADDR[10])_MDC_CAL_CS[2]_GPIO[72] Pad Configuration Register (SIU_PCR72)

See Table 6-15 for bit field definitions. The PA field for MPC5553's PCR72 is given in Table 6-19.

Table 6-19. PCR72 PA Field Definition

PA Field	Pin Function
0b000	GPIO[72]
0b001	CAL_ADDR[10][1]
0b010	MDC
0b011	CAL_ADDR[10][1]
0b100	CAL_CS[2][1]

1 For calibration only.

6.3.1.12.33 MPC5554: Pad Configuration Register 72 (SIU_PCR72)

The SIU_PCR72 register controls the pin function, direction, and static electrical attributes of the BR_GPIO[72] pin.

SIU_BASE+0xD0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	PA	OBE[1]	IBE[2]	DSC		ODE[3]	HYS[4]	0	0	WPE[5]	WPS[5]
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

1 When configured as $\overline{BR}$, the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

2 When configured as $\overline{BR}$ or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

3 When configured as $\overline{BR}$, and external master operation is enabled with external arbitration, the ODE bit should be set to zero.

4 If external master operation is enabled, the HYS bit should be set to zero.

5 See the EBI section for weak pull up settings when configured as $\overline{BR}$

Figure 6-45. MPC5554: $\overline{BR}$_GPIO[72] Pad Configuration Register (SIU_PCR72)

See Table 6-15 for bit field definitions.

6.3.1.12.34 MPC5553: Pad Configuration Register 73 (SIU_PCR73)

The SIU_PCR73 register controls the pin function, direction, and static electrical attributes of the $\overline{BG}$(CAL_ADDR[11])_MDIO_$\overline{CAL_CS}$[3]_GPIO[73] pin. The $\overline{BG}$ function is not available on the MPC5554. Instead, its PA encoding is used for CAL_ADDR[11]. This register allows selection of the CAL_ADDR[11], MDIO, $\overline{CAL\_CS}$[3], and GPIO functions.

SIU_BASE+0xD2

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	PA[1]			OBE[2]	IBE[3]	DSC		ODE[4]	HYS[5]	0	0	WPE[6]	WPS[6]
W																
RESET:	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1

1 The $\overline{BG}$ function is not available on the MPC5553. Do not select 0b001 or 0b011 for the PA field except for CAL_ADDR[11].

2 When configured as CAL_ADDR[11], MDIO, or $\overline{CAL\_CS}$[3], the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

3 When configured as CAL_ADDR[11], MDIO, $\overline{CAL\_CS}$[3], or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

4 When configured as CAL_ADDR[11] or $\overline{CAL\_CS}$[3], the ODE bit should be set to zero.

5 If external master operation is enabled, the HYS bit should be set to zero.

6 See the EBI section for weak pull up settings when configured as CAL_ADDR[11] or $\overline{CAL\_CS}$[3].

Figure 6-46. MPC5553: $\overline{BG}$(CAL_ADDR[11])_MDIO_$\overline{CAL_CS}$[3]_GPIO[73] Pad Configuration Register (SIU_PCR73)

See Table 6-15 for bit field definitions. The PA field for MPC5553's PCR73 is given in Table 6-20.

Table 6-20. PCR73 PA Field Definition

PA Field	Pin Function
0b000	GPIO[73]
0b001	CAL_ADDR[11][1]
0b010	MDIO
0b011	CAL_ADDR[11]<f-helvetica><st-superscript>
0b100	$\overline{\text{CAL\_CS}}$[3][1]

1 For calibration only.

6.3.1.12.35 MPC5554: Pad Configuration Register 73 (SIU_PCR73)

The SIU_PCR73 register controls the pin function, direction, and static electrical attributes of the $\overline{\text{BG}}$_GPIO[73] pin.

SIU_BASE+0xD2

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	PA	OBE[1]	IBE[2]	DSC		ODE[3]	HYS[4]	0	0	WPE[5]	WPS[5]
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

1 When configured as $\overline{\text{BG}}$, the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

2 When configured as $\overline{\text{BG}}$ or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

3 When configured as $\overline{\text{BG}}$, and external master operation is enabled with internal arbitration, the ODE bit should be set to zero.

4 If external master operation is enabled, the HYS bit should be set to zero.

5 See the EBI section for weak pull up settings when configured as $\overline{\text{BG}}$.

Figure 6-47. MPC5554: $\overline{\text{BG}}$_GPIO[73] Pad Configuration Register (SIU_PC73)

See Table 6-15 for bit field definitions.

6.3.1.12.36 MPC5554: Pad Configuration Register 74 (SIU_PCR74)

NOTE

The MPC5553 does not implement PCR74. Treat the register like reserved space.

The SIU_PCR74 register controls the pin function, direction, and static electrical attributes of the $\overline{\text{BB}}$_GPIO[74] pin.

SIU_BASE+0xD4

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	PA	OBE[1]	IBE[2]	DSC		ODE[3]	HYS[4]	0	0	WPE[5]	WPS[5]
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

1 When configured as $\overline{BB}$, the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

2 When configured as $\overline{BB}$ or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

3 When configured as $\overline{BB}$, and external master operation is enabled with internal arbitration, the ODE bit should be set to zero.

4 If external master operation is enabled, the HYS bit should be set to zero.

5 See the EBI section for weak pull up settings when configured as $\overline{BB}$.

Figure 6-48. MPC5554: $\overline{BB}$_GPIO[74] Pad Configuration Register (SIU_PCR74)

See Table 6-15 for bit field definitions.

6.3.1.12.37 Pad Configuration Register 75 - 82 (SIU_PCR75 - SIU_PCR82)

The SIU_PCR75 - SIU_PCR82 registers control the pin function, direction, and static electrical attributes of the MDO[4:11]_GPIO[75:82] pins. GPIO is the default function at reset for these pins. The full port mode (FPM) bit in the Nexus port controller (NPC) port configuration register controls whether the pins function as MDO[4:11] or GPIO[75:82]. The pad interface port enable for these pins is driven by the NPC block. When the FPM bit is set, the NPC enables the MDO port enable, and disables GPIO. When the FPM bit is cleared, the NPC disables the MDO port enable, and enables GPIO.

SIU_BASE+0xD6 - SIU_BASE+0xE4 (8)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	OBE[1]	IBE[1]	DSC		ODE[2]	HYS[3]	0	0	WPE[4]	WPS
W																
RESET:	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0

1 This bit applies only to GPIO operation. For GPO , the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting IBE to zero reduces power consumption.

2 The ODE bit should be set to zero for MDO operation.

3 The HYS bit has no affect on MDO operation.

4 The WPE bit should be set to zero for MDO operation.

Figure 6-49. MDO[4:11]_GPIO[75:82] Pad Configuration Register (SIU_PCR75 - SIU_PCR82)

See Table 6-15 for bit field definitions.

6.3.1.12.38 Pad Configuration Register 83 (SIU_PCR83)

The SIU_PCR83 register controls the pin function, direction, and static electrical attributes of the CNTXA_GPIO[83] pin.

SIU_BASE+0xE6

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	PA	OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

1 When configured as CNTX, the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

2 When configured as CNTX or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

Figure 6-50. CNTXA_GPIO[83] Pad Configuration Register (SIU_PCR83)

See Table 6-15 for bit field definitions.

6.3.1.12.39 Pad Configuration Register 84 (SIU_PCR84)

The SIU_PCR84 register controls the pin function, direction, and static electrical attributes of the CNRX_A_GPIO[84] pin.

SIU_BASE+0xE8

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	PA	OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

1 When configured as CNRX, the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

2 When configured as CNRX or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

Figure 6-51. CNRX_A_GPIO[84] Pad Configuration Register (SIU_PCR84)

See Table 6-15 for bit field definitions.

6.3.1.12.40 Pad Configuration Register 85 (SIU_PCR85)

The SIU_PCR85 register controls the pin function, direction, and static electrical attributes of the CNTXB_PCSC[3]_GPIO[85] pin. The CNTXB function is not available in the MPC5553 (this register allows selection of the PCSC[3] and GPIO functions).

SIU_BASE+0xEA

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA[1]		OBE[2]	IBE[3]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

1 The CNTXB function is not available on the MPC5553. Do not select 0b01 or 0b11 for the PA field.

2 When configured as CNTX (MPC5554 only) or PCS, the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

3 When configured as CNTX (MPC5554 only) or PCS or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

Figure 6-52. CNTXB_PCSC[3]_GPIO[85] Pad Configuration Register (SIU_PCR85)

See Table 6-15 for bit field definitions.

6.3.1.12.41 Pad Configuration Register 86 (SIU_PCR86)

The SIU_PCR86 register controls the pin function, direction, and static electrical attributes of the CNRX_B_PCSC[4]_GPIO[86] pin. The CNRX_B function is not available in MPC5553 (this register allows selection of the PCSC[4] and GPIO functions).

SIU_BASE+0xEC

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA[1]		OBE[2]	IBE[3]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

1 The CNRX_B function is not available on the MPC5553. Do not select 0b01 or 0b11 for the PA field.

2 When configured as CNRX or PCS, the OBE bit has no effect. When configured as GPO, the OBE bit should be set to one.

3 When configured as CNRX or PCS or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

Figure 6-53. CNRX_B_PCSC[4]_GPIO[86] Pad Configuration Register (SIU_PCR86)

See Table 6-15 for bit field definitions.

6.3.1.12.42 Pad Configuration Register 87 (SIU_PCR87)

The SIU_PCR87 register controls the pin function, direction, and static electrical attributes of the CNTXC_PCSD[3]_GPIO[87] pin.

SIU_BASE+0xEE

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA		OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

1 When configured as CNTX or PCS, the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

2 When configured as CNTX or PCS or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

Figure 6-54. CNTXC_PCSD[3]_GPIO[87] Pad Configuration Register (SIU_PCR87)

See Table 6-15 for bit field definitions.

6.3.1.12.43 Pad Configuration Register 88 (SIU_PCR88)

The SIU_PCR88 register controls the pin function, direction, and static electrical attributes of the CNRX_C_PCSD[4]_GPIO[88] pin.

SIU_BASE+0xF0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA		OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

1 When configured as CNRX or PCS, the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

2 When configured as CNRX or PCS or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

Figure 6-55. CNRX_C_PCSD[4]_GPIO[88] Pad Configuration Register (SIU_PCR88)

See Table 6-15 for bit field definitions.

6.3.1.12.44 Pad Configuration Register 89 (SIU_PCR89)

The SIU_PCR89 register controls the pin function, direction, and static electrical attributes of the TXD_A_GPIO[89] pin.

SIU_BASE+0xF2

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	PA	OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

1 When configured as TXD, the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

2 When configured as TXD or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. For SCI loop back operation the IBE bit must be set to one. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

Figure 6-56. TXD_A_GPIO[89] Pad Configuration Register (SIU_PCR89)

See Table 6-15 for bit field definitions.

6.3.1.12.45 Pad Configuration Register 90 (SIU_PCR90)

The SIU_PCR90 register controls the pin function, direction, and static electrical attributes of the RXD_A_GPIO[90] pin.

SIU_BASE+0xF4

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	PA	OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

1 When configured as RXD, the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

2 When configured as RXD or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

Figure 6-57. RXD_A_GPIO[90] Pad Configuration Register (SIU_PCR90)

See Table 6-15 for bit field definitions.

6.3.1.12.46 Pad Configuration Register 91 (SIU_PCR91)

The SIU_PCR91 register controls the pin function, direction, and static electrical attributes of the TXD_B_PCSD[1]_GPIO[91] pin.

SIU_BASE+0xF6

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA		OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

1 When configured as TXD or PCS, the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

2 When configured as TXD or PCS or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. For SCI loop back operation the IBE bit must be set to one. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

Figure 6-58. TXD_B_PCSD[1]_GPIO[91] Pad Configuration Register (SIU_PCR91)

See Table 6-15 for bit field definitions.

6.3.1.12.47 Pad Configuration Register 92 (SIU_PCR92)

The SIU_PCR92 register controls the pin function, direction, and static electrical attributes of the RXD_B_PCSD[5]_GPIO[92] pin.

SIU_BASE+0xF8

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA		OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

1 When configured as RXD or PCS, the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

2 When configured as RXD or PCS or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

Figure 6-59. RXD_B_PCSD[5]_GPIO[92] Pad Configuration Register (SIU_PCR92)

See Table 6-15 for bit field definitions.

6.3.1.12.48 Pad Configuration Register 93 (SIU_PCR93)

The SIU_PCR93 register controls the pin function, direction, and static electrical attributes of the SCKA_PCSC[1]_GPIO[93] pin. The SCKA function is not available in the MPC5553 (this register allows selection of the PCSC[1] and GPIO functions).

SIU_BASE+0xFA

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA[1]		OBE[2]	IBE[3]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

1 The SCKA function is not available on the MPC5553. Do not select 0b01 or 0b11 for the PA field.

2 When configured as SCK, the OBE bit should be set to one for master operation, and set to zero for slave operation. When configured as PCS, the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

3 When configured as SCK in slave operation, the IBE bit should be set to one. When configured as SCK in master operation, PCS, or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

Figure 6-60. SCKA_PCSC[1]_GPIO[93] Pad Configuration Register (SIU_PCR93)

See Table 6-15 for bit field definitions.

6.3.1.12.49 Pad Configuration Register 94 (SIU_PCR94)

The SIU_PCR94 register controls the pin function, direction, and static electrical attributes of the SIN_A_PCSC[2]_GPIO[94] pin. The SIN_A function is not available in the MPC5553 (this register allows selection of the PCSC[2] and GPIO functions).

SIU_BASE+0xFC

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA[1]		OBE[2]	IBE[3]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

1 The SIN_A function is not available on the MPC5553. Do not select 0b01 or 0b11 for the PA field.

2 When configured as SIN or PCS, the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

3 When configured as SIN, PCS, or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

Figure 6-61. SIN_A_PCSC[2]_GPIO[94] Pad Configuration Register (SIU_PCR94)

See Table 6-15 for bit field definitions.

6.3.1.12.50 Pad Configuration Register 95 (SIU_PCR95)

The SIU_PCR95 register controls the pin function, direction, and static electrical attributes of the SOUTA_PCSC[5]_GPIO[95] pin. The SOUTA function is not available in the MPC5553 (this register allows selection of the PCSC[5] and GPIO functions).

SIU_BASE+0xFE

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA[1]		OBE[2]	IBE[3]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

1 The SOUTA function is not available on the MPC5553. Do not select 0b01 or 0b11 for the PA field.

2 When configured as SOUT or PCS, the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

3 When configured as SOUT, PCS, or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

Figure 6-62. SOUTA_PCSC[5]_GPIO[95] Pad Configuration Register (SIU_PCR95)

See Table 6-15 for bit field definitions.

6.3.1.12.51 Pad Configuration Registers 96 (SIU_PCR96)

The SIU_PCR96 registers control the pin function, direction, and static electrical attributes of the PCSA[0]_PCSD[2]_GPIO[96] pin. The PCSA[0] function is not available in the MPC5553 (this register allows selection of the PCSD[2] and GPIO functions).

SIU_BASE+0x100

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA[1]		OBE[2]	IBE[3]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

1 The PCSA[0] function is not available on the MPC5553. Do not select 0b01 or 0b11 for the PA field.

2 When configured as PCSA[0], the OBE bit should be set to one for master operation, and set to zero for slave operation. When configured as PCSD[2], the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

3 When configured as PCSA[0] in slave operation, the IBE bit should be set to one. When configured as PCSA[0] in master operation, PCSD[2], or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

Figure 6-63. PCSA[0]_PCSD[2]_GPIO[96] Pad Configuration Register (SIU_PCR96)

See Table 6-15 for bit field definitions.

6.3.1.12.52 Pad Configuration Registers 97 (SIU_PCR97)

The SIU_PCR97 registers control the pin function, direction, and static electrical attributes of the PCSA[1]_PCSB[2]_GPIO[97] pin. The PCSA[1] function is not available in the MPC5553 (this register allows selection of the PCSB[2] and GPIO functions).

SIU_BASE+0x102

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA[1]		OBE[2]	IBE[3]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

1 The PCSA[1] function is not available on the MPC5553. Do not select 0b01 or 0b11 for the PA field.

2 When configured as PCS, the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

3 When configured as PCS or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

Figure 6-64. PCSA[1]_PCSB[2]_GPIO[97] Pad Configuration Register (SIU_PCR97)

See Table 6-15 for bit field definitions.

6.3.1.12.53 Pad Configuration Register 98 (SIU_PCR98)

The SIU_PCR98 register controls the pin function, direction, and static electrical attributes of the PCSA[2]_SCKD_GPIO[98] pin. The PCSA[2] function is not available in the MPC5553 (this register allows selection of the SCKD and GPIO functions).

SIU_BASE+0x104

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA[1]		OBE[2]	IBE[3]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

1 The PCSA[2] function is not available on the MPC5553. Do not select 0b01 or 0b11 for the PA field.

2 When configured as PCS, the OBE bit has no affect. When configured as SCK, the OBE bit should be set to one for master operation, and set to zero for slave operation. When configured as GPO, the OBE bit should be set to one.

3 When configured as SCK in slave operation, the IBE bit should be set to one. When configured as PCS or SCK in master operation or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

Figure 6-65. PCSA[2]_SCKD_GPIO[98] Pad Configuration Register (SIU_PCR98)

See Table 6-15 for bit field definitions.

6.3.1.12.54 Pad Configuration Register 99 (SIU_PCR99)

The SIU_PCR99 register controls the pin function, direction, and static electrical attributes of the PCSA[3]_SIN_D_GPIO[99] pin. The PCSA[3] function is not available in the MPC5553 (this register allows selection of the SIN_D and GPIO functions).

SIU_BASE+0x106

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA[1]		OBE[2]	IBE[3]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

[1]The PCSA[3] function is not available on the MPC5553. Do not select 0b01 or 0b11 for the PA field.
[2] When configured as PCS, the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.
[3] When configured as PCS or SIN or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

Figure 6-66. PCSA[3]_SIN_D_GPIO[99] Pad Configuration Register (SIU_PCR99)

See Table 6-15 for bit field definitions.

6.3.1.12.55 Pad Configuration Register 100 (SIU_PCR100)

The SIU_PCR100 register controls the pin function, direction, and static electrical attributes of the PCSA[4]_SOUTD_GPIO[100] pin. The PCSA[4] function is not available in the MPC5553 (this register allows selection of the SOUTD and GPIO functions).

SIU_BASE+0x108

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA[1]		OBE[2]	IBE[3]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

[1] The PCSA[4] function is not available on the MPC5553. Do not select 0b01 or 0b11 for the PA field.
[2] When configured as PCS or SOUT, the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.
[3] When configured as PCS or SOUT or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

Figure 6-67. PCSA[4]_SOUTD_GPIO[100] Pad Configuration Register (SIU_PCR100)

See Table 6-15 for bit field definitions.

6.3.1.12.56 Pad Configuration Registers 101 (SIU_PCR101)

The SIU_PCR101 register controls the pin function, direction, and static electrical attributes of the PCSA[5]_PCSB[3]_GPIO[101] pin. The PCSA[5] function is not available in the MPC5553 (this register allows selection of the PCSB[3] and GPIO functions).

SIU_BASE+0x10A

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA[1]		OBE[2]	IBE[3]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

1 The PCSA[5] function is not available on the MPC5553. Do not select 0b01 or 0b11 for the PA field.

2 When configured as PCS, the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

3 When configured as PCS or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

Figure 6-68. PCSA[5]_PCSB[3]_GPIO[101] Pad Configuration Register (SIU_PCR101)

See Table 6-15 for bit field definitions.

6.3.1.12.57 Pad Configuration Register 102 (SIU_PCR102)

The SIU_PCR102 register controls the pin function, direction, and static electrical attributes of the SCKB_PCSC[1]_GPIO[102] pin.

SIU_BASE+0x10C

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA		OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

1 When configured as SCK, the OBE bit should be set to one for master operation, and set to zero for slave operation. When configured as GPO, the OBE bit should be set to one.

2 When configured as SCK in slave operation the IBE bit should be set to one. When configured as SCK in master operation or PCS or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

Figure 6-69. SCKB_PCSC[1]_GPIO[102] Pad Configuration Register (SIU_PCR102)

See Table 6-15 for bit field definitions.

6.3.1.12.58 Pad Configuration Register 103 (SIU_PCR103)

The SIU_PCR103 register controls the pin function, direction, and static electrical attributes of the SIN_B_PCSC[2]_GPIO[103] pin.

SIU_BASE+0x10E

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA		OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

1 When configured as SIN, the OBE bit should be set to zero. When configured as PCS, the OBE bit should be set to one.

2 When configured as SIN or PCS, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

Figure 6-70. SIN_B_PCSC[2]_GPIO[103] Pad Configuration Register (SIU_PCR103)

See Table 6-15 for bit field definitions.

6.3.1.12.59 Pad Configuration Register 104 (SIU_PCR104)

The SIU_PCR104 register controls the pin function, direction, and static electrical attributes of the SOUTB_PCSC[5]_GPIO[104] pin.

SIU_BASE+0x110

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA		OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

1 When configured as SOUT or PCS, the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

2 When configured as SOUT or PCS or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

Figure 6-71. SOUTB_PCSC[5]_GPIO[104] Pad Configuration Register (SIU_PCR104)

See Table 6-15 for bit field definitions.

6.3.1.12.60 Pad Configuration Register 105 (SIU_PCR105)

The SIU_PCR105 register controls the pin function, direction, and static electrical attributes of the PCSB[0]_PCSD[2]_GPIO[105] pin.

SIU_BASE+0x112

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA		OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

1 When configured as PCS, the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

2 When configured as PCS or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

Figure 6-72. PCSB[0]_PCSD[2]_GPIO[105] Pad Configuration Register (SIU_PCR105)

See Table 6-15 for bit field definitions.

6.3.1.12.61 Pad Configuration Register 106 (SIU_PCR106)

The SIU_PCR106 register controls the pin function, direction, and static electrical attributes of the PCSB[1]_PCSD[0]_GPIO[106] pin.

SIU_BASE+0x114

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA		OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

1 When configured as PCSB[1], the OBE bit has no affect. When configured as PCSD[0], the OBE bit should be set to one for master operation, and set to zero for slave operation. When configured as GPO, the OBE bit should be set to one.

2 When configured as PCSD[0] in slave operation, the IBE bit should be set to one. When configured as PCS in master operation or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

Figure 6-73. PCSB[1]_PCSD[0]_GPIO[106] Pad Configuration Register (SIU_PCR106)

See Table 6-15 for bit field definitions.

6.3.1.12.62 Pad Configuration Register 107 (SIU_PCR107)

The SIU_PCR107 register controls the pin function, direction, and static electrical attributes of the PCSB[2]_SOUTC_GPIO[107] pin.

SIU_BASE+0x116

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA		OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

1 When configured as PCS or SOUT, the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

2 When configured as PCS or SOUT or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

Figure 6-74. PCSB[2]_SOUTC_GPIO[107] Pad Configuration Register (SIU_PCR107)

See Table 6-15 for bit field definitions.

6.3.1.12.63 Pad Configuration Register 108 (SIU_PCR108)

The SIU_PCR108 register controls the pin function, direction, and static electrical attributes of the PCSB[3]_SIN_C_GPIO[108] pin.

SIU_BASE+0x118

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA		OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

1 When configured as PCS or SIN, the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

2 When configured as PCS or SIN or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

Figure 6-75. PCSB[3]_SIN_C_GPIO[108] Pad Configuration Register (SIU_PCR108)

See Table 6-15 for bit field definitions.

6.3.1.12.64 Pad Configuration Register 109 (SIU_PCR109)

The SIU_PCR109 register controls the pin function, direction, and static electrical attributes of the PCSB[4]_SCKC_GPIO[109] pin.

SIU_BASE+0x11A

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA		OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

1 When configured as SCK, the OBE bit should be set to one for master operation, and set to zero for slave operation. When configured as PCS, the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

2 When configured as SCK in slave operation, the IBE bit should be set to one. When configured as PCS or SCK in master operation or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

Figure 6-76. PCSB[4]_SCKC_GPIO[109] Pad Configuration Register (SIU_PCR109)

See Table 6-15 for bit field definitions.

6.3.1.12.65 Pad Configuration Register 110 (SIU_PCR110)

The SIU_PCR110 register controls the pin function, direction, and static electrical attributes of the PCSB[5]_PCSC[0]_GPIO[110] pin.

SIU_BASE+0x11C

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA		OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

1 When configured as PCSB[5], the OBE bit has no affect. When configured as PCSC[0], the OBE bit should be set to one for master operation, and set to zero for slave operation. When configured as GPO, the OBE bit should be set to one.

2 When configured as PCSC[0] in slave operation, the IBE bit should be set to one. When configured as PCS in master operation or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

Figure 6-77. PCSB[5]_PCSC[0]_GPIO[110] Pad Configuration Register (SIU_PCR110)

See Table 6-15 for bit field definitions.

6.3.1.12.66 Pad Configuration Register 111 - 112 (SIU_PCR111 - SIU_PCR112)

The SIU_PCR111 - SIU_PCR112 registers control the pin function, direction, and static electrical attributes of the ETRIG[0:1]_GPIO[111:112] pins.

SIU_BASE+0x11E - SIU_BASE+0x120 (2)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	PA	OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

1 When configured as ETRIG, the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

2 When configured as ETRIG or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

Figure 6-78. ETRIG[0:1]_GPIO[111:112] Pad Configuration Register (SIU_PCR111 - SIU_PCR112)

See Table 6-15 for bit field definitions.

6.3.1.12.67 Pad Configuration Register 113 (SIU_PCR113)

The SIU_PCR113 register controls the pin function, direction, and static electrical attributes of the TCRCLKA_IRQ[7]_GPIO[113] pin.

SIU_BASE+0x122

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA		OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

1 When configured as TCRCLKA or IRQ, the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

2 When configured as TCRCLKA or IRQ or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption.When configured as GPI, the IBE bit should be set to one.

Figure 6-79. TCRCLKA_IRQ[7]_GPIO[113] Pad Configuration Register (SIU_PCR113)

See Table 6-15 for bit field definitions.

6.3.1.12.68 Pad Configuration Register 114 - 125 (SIU_PCR114 - SIU_PCR125)

The SIU_PCR114 - SIU_PCR125 registers control the pin function, direction, and static electrical attributes of the ETPUA[0:11]_ETPUA[12:23]_GPIO[114:125] pins. Only the output channels of ETPUA[12:23] are connected to pins. Both the input and output channels of ETPUA[0:11] are connected to pins.

SIU_BASE+0x124 - SIU_BASE+0x13A (12)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA		OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	U[3]

1 The OBE bit must be set to one for both ETPUA[0:11] and GPIO[114:125] when configured as outputs. When configured as ETPUA[12:23], the OBE bit has no affect.

2 The IBE bit must be set to one for both ETPUA[0:11] and GPIO[114:125] when configured as inputs. When configured as ETPUA[12:23] or when ETPUA[0:11] or GPIO[114:125] are configured as outputs, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register.

3 The weak pull up/down selection at reset for the ETPUA[0:11] pins is determined by the WKPCFG pin.

**Figure 6-80. ETPUA[0:11]_ETPUA[12:23]_GPIO[114:125]
Pad Configuration Register (SIU_PCR114 - SIU_PCR125)**

See Table 6-15 for bit field definitions.

6.3.1.12.69 Pad Configuration Register 126 (SIU_PCR126)

The SIU_PCR126 register controls the pin function, direction, and static electrical attributes of the ETPUA[12]_PCSB[1]_GPIO[126] pin.

SIU_BASE+0x13C

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA		OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	U[3]

1 When configured as PCS, the OBE bit has no affect. The OBE bit must be set to one for both ETPUA and GPIO when configured as outputs.

2 The IBE bit must be set to one for both ETPUA and GPIO when configured as inputs. When configured as PCS, or ETPUA or GPO outputs, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register.

3 The weak pull up/down selection at reset for the ETPUA[12] pin is determined by the WKPCFG pin.

Figure 6-81. ETPUA[12]_PCSB[1]_GPIO[126] Pad Configuration Register (SIU_PCR126)

See Table 6-15 for bit field definitions.

6.3.1.12.70 Pad Configuration Register 127 (SIU_PCR127)

The SIU_PCR127 register controls the pin function, direction, and static electrical attributes of the ETPUA[13]_PCSB[3]_GPIO[127] pin.

SIU_BASE+0x13E

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA		OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	U[3]

1 When configured as PCS, the OBE bit has no affect. The OBE bit must be set to one for both ETPUA and GPIO when configured as outputs.

2 The IBE bit must be set to one for both ETPUA and GPIO when configured as inputs. When configured as PCS, or ETPUA or GPO outputs, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register.

3 The weak pull up/down selection at reset for the ETPUA[13] pin is determined by the WKPCFG pin.

Figure 6-82. ETPUA[13]_PCSB[3]_GPIO[127] Pad Configuration Register (SIU_PCR127)

See Table 6-15 for bit field definitions.

6.3.1.12.71 Pad Configuration Register 128 (SIU_PCR128)

The SIU_PCR128 register controls the pin function, direction, and static electrical attributes of the ETPUA[14]_PCSB[4]_GPIO[128] pin.

SIU_BASE+0x140

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA		OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	U[3]

1 When configured as PCS, the OBE bit has no affect. The OBE bit must be set to one for both ETPUA and GPIO when configured as outputs.

2 The IBE bit must be set to one for both ETPUA and GPIO when configured as inputs. When configured as PCS, or ETPUA or GPO outputs, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register.

3 The weak pull up/down selection at reset for the ETPUA[14] pin is determined by the WKPCFG pin.

Figure 6-83. ETPUA[14]_PCSB[4]_GPIO[128] Pad Configuration Register (SIU_PCR128)

See Table 6-15 for bit field definitions.

6.3.1.12.72 Pad Configuration Register 129 (SIU_PCR129)

The SIU_PCR129 register controls the pin function, direction, and static electrical attributes of the ETPUA[15]_PCSB[5]_GPIO[129] pin.

SIU_BASE+0x142

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA		OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	U[3]

1 When configured as PCS, the OBE bit has no affect. The OBE bit must be set to one for both ETPUA and GPIO when configured as outputs.

2 The IBE bit must be set to one for both ETPUA and GPIO when configured as inputs. When configured as PCS, or ETPUA or GPO outputs, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register.

3 The weak pull up/down selection at reset for the ETPUA[15] pin is determined by the WKPCFG pin.

Figure 6-84. ETPUA[15]_PCSB[5]_GPIO[129] Pad Configuration Register (SIU_PCR129)

See Table 6-15 for bit field definitions.

6.3.1.12.73 Pad Configuration Register 130 (SIU_PCR130)

The SIU_PCR130 register controls the pin function, direction, and static electrical attributes of the ETPUA[16]_PCSD[1]_GPIO[130] pin.

SIU_BASE+0x144

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA		OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	U[3]

1 When configured as PCS, the OBE bit has no affect. The OBE bit must be set to one for both ETPUA and GPIO when configured as outputs.

2 The IBE bit must be set to one for both ETPUA and GPIO when configured as inputs. When configured as PCS, or ETPUA or GPO outputs, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register.

3 The weak pull up/down selection at reset for the ETPUA[16] pin is determined by the WKPCFG pin.

Figure 6-85. ETPUA[16]_PCSD[1]_GPIO[130] Pad Configuration Register (SIU_PCR130)

See Table 6-15 for bit field definitions.

6.3.1.12.74 Pad Configuration Register 131 (SIU_PCR131)

The SIU_PCR131 register controls the pin function, direction, and static electrical attributes of the ETPUA[17]_PCSD[2]_GPIO[131] pin.

SIU_BASE+0x146

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA		OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	U[3]

1 When configured as PCS, the OBE bit has no affect. The OBE bit must be set to one for both ETPUA and GPIO when configured as outputs.

2 The IBE bit must be set to one for both ETPUA and GPIO when configured as inputs. When configured as PCS, or ETPUA or GPO outputs, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register.

3 The weak pull up/down selection at reset for the ETPUA[17] pin is determined by the WKPCFG pin.

Figure 6-86. ETPUA[17]_PCSD[2]_GPIO[131] Pad Configuration Register (SIU_PCR131)

See Table 6-15 for bit field definitions.

6.3.1.12.75 Pad Configuration Register 132 (SIU_PCR132)

The SIU_PCR132 register controls the pin function, direction, and static electrical attributes of the ETPUA[18]_PCSD[3]_GPIO[132] pin.

SIU_BASE+0x148

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA		OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	U[3]

1 When configured as PCS, the OBE bit has no affect. The OBE bit must be set to one for both ETPUA and GPIO when configured as outputs.

2 The IBE bit must be set to one for both ETPUA and GPIO when configured as inputs. When configured as PCS, or ETPUA or GPO outputs, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register.

3 The weak pull up/down selection at reset for the ETPUA[18] pin is determined by the WKPCFG pin.

Figure 6-87. ETPUA[18]_PCSD[3]_GPIO[132] Pad Configuration Register (SIU_PCR132)

See Table 6-15 for bit field definitions.

6.3.1.12.76 Pad Configuration Register 133 (SIU_PCR133)

The SIU_PCR133 register controls the pin function, direction, and static electrical attributes of the ETPUA[19]_PCSD[4]_GPIO[133] pin.

SIU_BASE+0x14A

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA		OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	U[3]

1 When configured as PCS, the OBE bit has no affect. The OBE bit must be set to one for both ETPUA and GPIO when configured as outputs.

2 The IBE bit must be set to one for both ETPUA and GPIO when configured as inputs. When configured as PCS, or ETPUA or GPO outputs, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register.

3 The weak pull up/down selection at reset for the ETPUA[19] pin is determined by the WKPCFG pin.

Figure 6-88. ETPUA[19]_PCSD[4]_GPIO[133] Pad Configuration Register (SIU_PCR133)

See Table 6-15 for bit field definitions.

6.3.1.12.77 Pad Configuration Register 134 - 141 (SIU_PCR134 - SIU_PCR141)

The SIU_PCR134 - SIU_PCR141 registers control the pin function, direction, and static electrical attributes of the ETPUA[20:27]_IRQ[8:15]_GPIO[134:141] pins. Only the output channels of ETPUA[24:27] are connected to pins. Both the input and output channels of ETPUA[20:23] are connected to pins.

SIU_BASE+0x14C - SIU_BASE+0x15A (8)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA		OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	U[3]

1 When configured as ETPUA[24:27] or IRQ, the OBE bit has no affect. The OBE bit must be set to one for both ETPUA[20:23] and GPIO[134:141] when configured as outputs.

2 When configured as ETPUA[24:27] or IRQ or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. The IBE bit must be set to one for both ETPUA[20:23] and GPIO[134:141] when configured as inputs.

3 The weak pull up/down selection at reset for the ETPUA[20:27] pins is determined by the WKPCFG pin.

Figure 6-89. ETPUA[20:27]_IRQ[8:15]_GPIO[134:141] Pad Configuration Register (SIU_PCR134 - SIU_PCR141)

See Table 6-15 for bit field definitions.

6.3.1.12.78 Pad Configuration Register 142 (SIU_PCR142)

The SIU_PCR142 register controls the pin function, direction, and static electrical attributes of the ETPUA[28]_PCSC[1]_GPIO[142] pin. Only the output channel of ETPUA[28] is connected to the pin.

SIU_BASE+0x15C

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA		OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	U[3]

1 When configured as ETPUA or PCS, the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

2 When configured as ETPUA, PCS, or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. The IBE bit must be set to one for GPIO when configured as input.

3 The weak pull up/down selection at reset for the ETPUA[28] pin is determined by the WKPCFG pin

Figure 6-90. ETPUA[28]_PCSC[1]_GPIO[142] Pad Configuration Register (SIU_PCR142)

See Table 6-15 for bit field definitions.

6.3.1.12.79 Pad Configuration Register 143 (SIU_PCR143)

The SIU_PCR143 register controls the pin function, direction, and static electrical attributes of the ETPUA[29]_PCSC[2]_GPIO[143] pin. Only the output channel of ETPUA[29] is connected to the pin.

SIU_BASE+0x15E

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA		OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	U[3]

1 When configured as ETPUA or PCS, the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

2 When configured as ETPUA, PCS, or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. The IBE bit must be set to one for GPIO when configured as input.

3 The weak pull up/down selection at reset for the ETPUA[29] pin is determined by the WKPCFG pin

Figure 6-91. ETPUA[29]_PCSC[2]_GPIO[143] Pad Configuration Register (SIU_PCR143)

See Table 6-15 for bit field definitions.

6.3.1.12.80 Pad Configuration Register 144 (SIU_PCR144)

The SIU_PCR144 register controls the pin function, direction, and static electrical attributes of the ETPUA[30]_PCSC[3]_GPIO[144] pin.

SIU_BASE+0x160

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA		OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	U[3]

1 When configured as PCS, the OBE bit has no affect. When configured as ETPUA output or GPO, the OBE bit should be set to one.

2 When configured as ETPUA output, PCS, or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. The IBE bit must be set to one for ETPUA or GPIO when configured as input.

3 The weak pull up/down selection at reset for the ETPUA[30] pin is determined by the WKPCFG pin

Figure 6-92. ETPUA[30]_PCSC[3]_GPIO[144] Pad Configuration Register (SIU_PCR144)

See Table 6-15 for bit field definitions.

6.3.1.12.81 Pad Configuration Register 145 (SIU_PCR145)

The SIU_PCR145 register controls the pin function, direction, and static electrical attributes of the ETPUA[31]_PCSC[4]_GPIO[145] pin.

SIU_BASE+0x162

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA		OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	U[3]

1 When configured as PCS, the OBE bit has no affect. When configured as ETPUA output or GPO, the OBE bit should be set to one.

2 When configured as ETPUA output, PCS, or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. The IBE bit must be set to one for ETPUA or GPIO when configured as input.

3 The weak pull up/down selection at reset for the ETPUA[31] pin is determined by the WKPCFG pin

Figure 6-93. ETPUA[31]_PCSC[4]_GPIO[145] Pad Configuration Register (SIU_PCR145)

See Table 6-15 for bit field definitions.

6.3.1.12.82 MPC5554: Pad Configuration Register 146 (SIU_PCR146)

NOTE

The MPC5553 does not implement PCR146–178. Treat those registers as reserved space.

The SIU_PCR146 register controls the pin function, direction, and static electrical attributes of the TCRCLKB_IRQ[6]_GPIO[146] pin.

SIU_BASE+0x164

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA		OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

1 When configured as TCRCLKB or IRQ, the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

2 When configured as TCRCLKB or IRQ or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

Figure 6-94. MPC5554: TCRCLKB_IRQ[6]_GPIO[146] Pad Configuration Register (SIU_PCR146)

See Table 6-15 for bit field definitions.

6.3.1.12.83 MPC5554: Pad Configuration Register 147 - 162 (SIU_PCR147 - SIU_PCR162)

NOTE

The MPC5553 does not implement PCR146–178. Treat those registers as reserved space.

The SIU_PCR147 - SIU_PCR162 registers control the pin function, direction, and static electrical attributes of the ETPUB[0:15]_ETPUB[16:31]_GPIO[147:162] pins. Both the input and output channels

of ETPUB[0:15] are connected to these pins and only the output channels of ETPUB[16:31] are connected to these pins.

SIU_BASE+0x166 - SIU_BASE+0x184 (16)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA		OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	U[3]

1 The OBE bit must be set to one for both ETPUB[0:15] and GPIO[147:162] when configured as outputs. When configured as ETPUB[16:31], the OBE bit has no affect.

2 When configured as ETPUB or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. The IBE bit must be set to one for both ETPUB[0:15] and GPIO[147:162] when configured as inputs.

3 The weak pull up/down selection at reset for the ETPUB[0:15] pins is determined by the WKPCFG pin.

Figure 6-95. MPC5554: ETPUB[0:15]_ETPUB[16:31]_GPIO[147:162] Pad Configuration Register (SIU_PCR147 - SIU_PCR162)

See Table 6-15 for bit field definitions.

6.3.1.12.84 MPC5554: Pad Configuration Register 163 (SIU_PCR163)

NOTE

The MPC5553 does not implement PCR146–178. Treat those registers as reserved space.

The SIU_PCR163 register controls the pin function, direction, and static electrical attributes of the ETPUB[16]_PCSA[1]_GPIO[163] pin. Both the input and output channel of ETPUB[16] are connected to the pin.

SIU_BASE+0x186

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA		OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	U[3]

1 The OBE bit must be set to one for both ETPUB and GPIO when configured as outputs. When configured as PCS, the OBE bit has no affect.

2 When configured as ETPUB or GPIO outputs, or configured as PCS, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. The IBE bit must be set to one for both ETPUB and GPIO when configured as inputs.

3 The weak pull up/down selection at reset for the ETPUB[16] pin is determined by the WKPCFG pin.

Figure 6-96. MPC5554: ETPUB[16]_PCSA[1]_GPIO[163] Pad Configuration Register (SIU_PCR163)

See Table 6-15 for bit field definitions.

6.3.1.12.85 MPC5554: Pad Configuration Register 164 (SIU_PCR164)

NOTE

The MPC5553 does not implement PCR146–178. Treat those registers as reserved space.

The SIU_PCR164 register controls the pin function, direction, and static electrical attributes of the ETPUB[17]_PCSA[2]_GPIO[164] pin. Both the input and output channel of ETPUB[17] are connected to the pin.

SIU_BASE+0x188

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA		OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	U[3]

1 The OBE bit must be set to one for both ETPUB[17] and GPIO[164] when configured as outputs. When configured as PCS, the OBE bit has no affect.

2 When configured as ETPUB or GPIO outputs, or configured as PCS, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. The IBE bit must be set to one for both ETPUB[17] and GPIO[164] when configured as inputs.

3 The weak pull up/down selection at reset for the ETPUB[17] pin is determined by the WKPCFG pin.

Figure 6-97. MPC5554: ETPUB[17]_PCSA[2]_GPIO[164] Pad Configuration Register (SIU_PCR164)

See Table 6-15 for bit field definitions.

6.3.1.12.86 MPC5554: Pad Configuration Register 165 (SIU_PCR165)

NOTE

The MPC5553 does not implement PCR146–178. Treat those registers as reserved space.

The SIU_PCR165 register controls the pin function, direction, and static electrical attributes of the ETPUB[18]_PCSA[3]_GPIO[165] pin. Both the input and output channel of ETPUB[18] are connected to the pin.

SIU_BASE+0x18A

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA		OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	U[3]

1 The OBE bit must be set to one for both ETPUB[18] and GPIO[165] when configured as outputs. When configured as PCS, the OBE bit has no affect.

2 When configured as ETPUB or GPIO outputs, or configured as PCS, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. The IBE bit must be set to one for both ETPUB[18] and GPIO[165] when configured as inputs.

3 The weak pull up/down selection at reset for the ETPUB[18] pin is determined by the WKPCFG pin.

Figure 6-98. MPC5554: ETPUB[18]_PCSA[3]_GPIO[165] Pad Configuration Register (SIU_PCR165)

See Table 6-15 for bit field definitions.

6.3.1.12.87 Pad Configuration Register 166 (SIU_PCR166)

NOTE

The MPC5553 does not implement PCR146–178. Treat those registers as reserved space.

The SIU_PCR166 register controls the pin function, direction, and static electrical attributes of the ETPUB[19]_PCSA[4]_GPIO[166] pin. Both the input and output channel of ETPUB[19] are connected to the pin.

SIU_BASE+0x18C

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA		OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	U[3]

1 The OBE bit must be set to one for both ETPUB[19] and GPIO[166] when configured as outputs. When configured as PCS, the OBE bit has no affect.

2 When configured as ETPUB or GPIO outputs, or configured as PCS, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. The IBE bit must be set to one for both ETPUB[19] and GPIO[166] when configured as inputs.

3 The weak pull up/down selection at reset for the ETPUB[19] pin is determined by the WKPCFG pin.

Figure 6-99. ETPUB[19]_PCSA[4]_GPIO[166] Pad Configuration Register (SIU_PCR166)

See Table 6-15 for bit field definitions.

6.3.1.12.88 MPC5554: Pad Configuration Register 167 - 178 (SIU_PCR167 - SIU_PCR178)

NOTE

The MPC5553 does not implement PCR146–178. Treat those registers as reserved space.

The SIU_PCR167 - SIU_PCR178 registers control the pin function, direction, and static electrical attributes of the ETPUB[20:31]_GPIO[167:178] pins. Both the inputs and outputs of ETPUB[20:31] are connected to these pins.

SIU_BASE+0x18E - SIU_BASE+0x1A4 (12)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	PA	OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	U[3]

1 The OBE bit must be set to one for both ETPUB[20:31] and GPIO[167:178] when configured as outputs.

2 When configured as ETPUB or GPIO outputs, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. The IBE bit must be set to one for both ETPUB[20:31] and GPIO[167:178] when configured as inputs.

3 The weak pull up/down selection at reset for the ETPUB[20:31] pins is determined by the WKPCFG pin.

Figure 6-100. MPC5554: ETPUB[20:31]_GPIO[167:178]
Pad Configuration Register (SIU_PCR167 - SIU_PCR178)

See Table 6-15 for bit field definitions.

6.3.1.12.89 Pad Configuration Register 179 - 188 (SIU_PCR179 - SIU_PCR188)

The SIU_PCR179 - SIU_PCR188 registers control the pin function, direction, and static electrical attributes of the EMIOS[0:9]_ETPUA[0:9]_GPIO[179:188] pins. Both the input and output functions of EMIOS[0:9], and only the output channels of ETPUA[0:9] are connected to pins.

SIU_BASE+0x1A6 - SIU_BASE+0x1B8 (10)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA		OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	U[3]

1 The OBE bit must be set to one for both EMIOS[0:9] and GPIO[179:188] when configured as outputs.

2 When configured as EMIOS, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption.The IBE bit must be set to one for both EMIOS[0:9] and GPIO[179:188] when configured as inputs.

3 The weak pull up/down selection at reset for the EMIOS[0:9] pins is determined by the WKPCFG pin.

Figure 6-101. EMIOS[0:9]_ETPUA[0:9]_GPIO[179:188]
Pad Configuration Register (SIU_PCR179 - SIU_PCR188)

See Table 6-15 for bit field definitions.

6.3.1.12.90 Pad Configuration Register 189 - 190 (SIU_PCR189 - SIU_PCR190)

The SIU_PCR189 - SIU_PCR190 registers control the pin function, direction, and static electrical attributes of the EMIOS[10:11]_GPIO[189:190] pins. Both the input and output functions of EMIOS[10:11] are connected to pins.

SIU_BASE+0x1BA - SIU_BASE+0x1BC (2)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	PA	OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	U[3]

1 The OBE bit must be set to one for both EMIOS[10:11] and GPIO[189:190] when configured as outputs.

2 When configured as EMIOS or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. The IBE bit must be set to one for both EMIOS[10:11] and GPIO[189:190] when configured as inputs.

3 The weak pull up/down selection at reset for the EMIOS[10:11] pins is determined by the WKPCFG pin.

Figure 6-102. EMIOS[10:11]_GPIO[189:190] Pad Configuration Register (SIU_PCR189 - SIU_PCR190)

See Table 6-15 for bit field definitions.

6.3.1.12.91 Pad Configuration Register 191 (SIU_PCR191)

The SIU_PCR191 register controls the pin function, direction, and static electrical attributes of the EMIOS[12]_SOUTC_GPIO[191] pin. Only the output of EMIOS[12] is connected to the pin.

SIU_BASE+0x1BE

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA		OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	U[3]

1 The OBE bit must be set to one for GPIO[191] when configured as an output.

2 When configured as EMIOS or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. The IBE bit must be set to one for GPIO[191] when configured as an input.

3 The weak pull up/down selection at reset for the EMIOS[12] pin is determined by the WKPCFG pin.

Figure 6-103. EMIOS[12]_SOUTC_GPIO[191] Pad Configuration Register (SIU_PCR191)

See Table 6-15 for bit field definitions.

6.3.1.12.92 Pad Configuration Register 192 (SIU_PCR192)

The SIU_PCR191 register controls the pin function, direction, and static electrical attributes of the EMIOS[13]_SOUTD_GPIO[192] pin. Only the output of EMIOS[13] is connected to the pin.

SIU_BASE+0x1C0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA		OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	U[3]

1 The OBE bit must be set to one for GPIO[192] when configured as an output.

2 When configured as EMIOS or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. The IBE bit must be set to one for GPIO[192] when configured as an input.

3 The weak pull up/down selection at reset for the EMIOS[13] pin is determined by the WKPCFG pin.

Figure 6-104. EMIOS[13]_SOUTD_GPIO[192] Pad Configuration Register (SIU_PCR192)

See Table 6-15 for bit field definitions.

6.3.1.12.93 Pad Configuration Register 193 - 194 (SIU_PCR193 - SIU_PCR194)

The SIU_PCR193 - SIU_PCR194 registers control the pin function, direction, and static electrical attributes of the EMIOS[14:15]_IRQ[0:1]_GPIO[193:194] pins. Only the output functions of EMIOS[14:15] are connected to pins.

SIU_BASE+0x1C2 - SIU_BASE+0x1C4 (2)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA		OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	U[3]

1 The OBE bit must be set to one for GPIO[193:194] when configured as outputs.

2 When configured as EMIOS or IRQ or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. The IBE bit must be set to one for GPIO[193:194] when configured as inputs.

3 The weak pull up/down selection at reset for the EMIOS[14:15] pins is determined by the WKPCFG pin.

**Figure 6-105. EMIOS[14:15]_IRQ[0:1]_GPIO[193:194]
Pad Configuration Register (SIU_PCR193 - SIU_PCR194)**

See Table 6-15 for bit field definitions.

6.3.1.12.94 Pad Configuration Register 195 - 202 (SIU_PCR195 - SIU_PCR202)

MPC5553: The SIU_PCR195 - SIU_PCR202 registers control the pin function, direction, and static electrical attributes of the EMIOS[16:23]_ETPUB[0:7]_GPIO[195:202] pins. Both the input and output functions of EMIOS[16:23] are connected to pins. The secondary function, ETPUB[0:7] is unavailable.

MPC5554: The SIU_PCR195 - SIU_PCR202 registers control the pin function, direction, and static electrical attributes of the EMIOS[16:23]_ETPUB[0:7]_GPIO[195:202] pins. Both the input and output functions of EMIOS[16:23], and only the output channels of ETPUB[0:7] are connected to pins.

SIU_BASE+0x1C6 - SIU_BASE+0x1D4 (8)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA		OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	U[3]

1 The OBE bit must be set to one for both EMIOS[16:23] and GPIO[195:202] when configured as outputs.

2 When configured as EMIOS or eTPU, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption.The IBE bit must be set to one for both EMIOS[16:23] and GPIO[195:202] when configured as inputs.

3 The weak pull up/down selection at reset for the EMIOS[0:9] pins is determined by the WKPCFG pin.

Figure 6-106. EMIOS[16:23]_ETPUB[0:7]_GPIO[195:202] Pad Configuration Register (SIU_PCR195 - SIU_PCR202)

See Table 6-15 for bit field definitions.

6.3.1.12.95 Pad Configuration Register 203 - 204 (SIU_PCR203 - SIU_PCR204)

The SIU_PCR203 - SIU_PCR204 registers control the pin function, direction, and static electrical attributes of the GPIO[203:204]_EMIOS[14:15] pins. Only the output functions of EMIOS[14:15] are connected to these pins. These pins are named GPIO[203:204] because other balls are already named EMIOS[14:15]. The primary function of these pins is EMIOS, however, out of reset, they are configured for GPIO use. These pins are not affected by WKPCFG (see Section 2.3.1.7, “Weak Pull Configuration / GPIO (WKPCFG_GPIO213)).

SIU_BASE+0x1D6 - SIU_BASE+0x1D8 (2)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	PA[1]	OBE[2]	IBE[3]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

1 MPC5554 only: The PA bit should be set to one for EMIOS and cleared to zero when used as GPIO.

2 When configured as EMIOS the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

3 When configured as EMIOS or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

Figure 6-107. GPIO[203:204]_EMIOS[14:15] Pad Configuration Register (SIU_PCR203 - SIU_PCR204)

See Table 6-15 for bit field definitions.

6.3.1.12.96 Pad Configuration Registers 205 (SIU_PCR205)

The SIU_PCR205 register controls the direction and static electrical attributes of the GPIO[205] pin. This register is separate from the PCRs for GPIO[206:207] since GPIO[205] is a medium pad type with slew rate control and GPIO[206:207] are fast pad types with drive strength control. The PA bit is not implemented for this PCR since GPIO is the only pin function.

SIU_BASE+0x1DA

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	OBE[1]	IBE[2]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

1 When configured as GPO, the OBE bit should be set to one.

2 When configured as GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. When configured as GPI, the IBE bit should be set to one. Setting the IBE bit to zero reduces power consumption.

Figure 6-108. GPIO[205] Pad Configuration Registers (SIU_PCR205)

See Table 6-15 for bit field definitions.

6.3.1.12.97 Pad Configuration Registers 206 - 207 (SIU_PCR206 - SIU_PCR207)

The SIU_PCR206 - SIU_PCR207 registers control the pin function, direction, and static electrical attributes of the GPIO[206:207] pins. These registers are separate from the PCR for GPIO[205] since GPIO[206:207] are fast pad types with drive strength control and GPIO[205] is a medium pad type with slew rate control. The PA bit is not implemented for these PCRs since GPIO is the only pin function.

NOTE

The GPIO[206:7] pins have the capability to trigger the ADCs. For the ETRIG functionality, these GPIO pins need to be set as GPIO and then select the GPIO ADC trigger in the (SIU_ETISR) - see Section 6.3.1.15, "eQADC Trigger Input Select Register (SIU_ETISR)."

SIU_BASE+0x1DC - SIU_BASE+0x1DE (2)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	OBE[1]	IBE[2]	DSC		ODE	HYS	0	0	WPE	WPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

1 When configured as GPO, the OBE bit should be set to one.

2 When configured as GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. When configured as GPI, the IBE bit should be set to one. Setting the IBE bit to zero reduces power consumption.

Figure 6-109. GPIO[206:207] Pad Configuration Registers (SIU_PCR206 - SIU_PCR207)

See Table 6-15 for bit field definitions.

6.3.1.12.98 Pad Configuration Register 208 (SIU_PCR208)

The SIU_PCR208 register controls the pin function, direction, and static electrical attributes of the PLLCFG[0]_IRQ[4]_GPIO[208] pin.

SIU_BASE+0x1E0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA[1]		OBE[2]	IBE[3]	0	0	ODE	HYS[4]	SRC		WPE	WPS
W																
RESET:	0	0	0	0	1	1	0	0	0	0	0	1	0	0	1	1

1 The PLLCFG function applies only during reset when the $\overline{\text{RSTCFG}}$ pin is asserted during reset. The PA field should be set to 0b10 for IRQ[4] and set to 0b00 for GPIO[208].

2 When configured as IRQ, the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

3 When configured as IRQ or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

4 When configured as IRQ, the HYS bit should be set to one.

Figure 6-110. PLLCFG[0]_IRQ[4]_GPIO[208] Pad Configuration Register (SIU_PCR208)

See Table 6-15 for bit field definitions.

6.3.1.12.99 Pad Configuration Register 209 (SIU_PCR209)

The SIU_PCR209 register controls the pin function, direction, and static electrical attributes of the PLLCFG[1]_IRQ[5]_SOUTD_GPIO[209] pins.

SIU_BASE+0x1E2

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	PA[1]			OBE[2]	IBE[3]	0	0	ODE	HYS[4]	SRC		WPE	WPS
W																
RESET:	0	0	0	0	1	1	0	0	0	0	0	1	0	0	1	1

1 The PLLCFG function applies only during reset when the $\overline{\text{RSTCFG}}$ pin is asserted during reset. The PA field should be set to 0b010 for IRQ[5], 0b100 for SOUTD, and 0b000 for GPIO[209].

2 When configured as IRQ, the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

3 When configured as IRQ or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

4 When configured as IRQ, the HYS bit should be set to one.

Figure 6-111. PLLCFG[1]_IRQ[5]_SOUTD_GPIO[209] Pad Configuration Register (SIU_PCR209)

See Table 6-15 for bit field definitions.

6.3.1.12.100Pad Configuration Register 210 (SIU_PCR210)

The SIU_PCR210 register controls the pin function, direction, and static electrical attributes of the $\overline{\text{RSTCFG}}$_GPIO[210] pin.

SIU_BASE+0x1E4

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	PA[1]	OBE[2]	IBE[3]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1	1

1 $\overline{\text{RSTCFG}}$ function is only applicable during reset. The PA bit must be set to zero for GPIO operation

2 When configured as GPO, the OBE bit should be set to one.

3 When configured as GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. When configured as GPI, the IBE bit should be set to one.

Figure 6-112. $\overline{\text{RSTCFG}}$_GPIO[210] Pad Configuration Register (SIU_PCR210)

See Table 6-15 for bit field definitions.

6.3.1.12.101 Pad Configuration Register 211 - 212 (SIU_PCR211 - SIU_PCR212)

The SIU_PCR211 - SIU_PCR212 registers control the pin function, direction, and static electrical attributes of the BOOTCFG[0:1]_IRQ[2:3]_GPIO[211:212] pins.

SIU_BASE+0x1E6 - SIU_BASE+0x1E8 (2)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA[1]		OBE[2]	IBE[3]	0	0	ODE	HYS[4]	SRC		WPE	WPS
W																
RESET:	0	0	0	0	1	1	0	0	0	0	0	1	0	0	1	0

1 The BOOTCFG function applies only during reset when the $\overline{\text{RSTCFG}}$ pin is asserted during reset. The PA field should be set to 0b10 for IRQ[2:3] and set to 0b00 for GPIO[211:212].

2 When configured as IRQ, the OBE bit has no affect. When configured as GPO, the OBE bit should be set to one.

3 When configured as IRQ or GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. Setting the IBE bit to zero reduces power consumption. When configured as GPI, the IBE bit should be set to one.

4 When configured as IRQ, the HYS bit should be set to one.

Figure 6-113. BOOTCFG[0:1]_IRQ[2:3]_GPIO[211:212] Pad Configuration Register (SIU_PCR211 - SIU_PCR212)

See Table 6-15 for bit field definitions.

6.3.1.12.102 Pad Configuration Register 213 (SIU_PCR213)

The SIU_PCR213 register controls the pin function, direction, and static electrical attributes of the WKPCFG_GPIO[213] pin.

SIU_BASE+0x1EA

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	PA[1]	OBE[2]	IBE[3]	0	0	ODE	HYS	SRC		WPE	WPS
W																
RESET:	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1	1

1 WKPCFG function is only applicable during reset. The PA bit must be set to zero for GPIO operation

2 When configured as GPO, the OBE bit should be set to one.

3 When configured as GPO, the IBE bit may be set to one to reflect the pin state in the corresponding GPDI register. When configured as GPI, the IBE bit should be set to one.

Figure 6-114. WKPCFG_GPIO[213] Pad Configuration Register (SIU_PCR213)

See Table 6-15 for bit field definitions.

6.3.1.12.103Pad Configuration Register 214 (SIU_PCR214)

The SIU_PCR214 register controls the enabling/disabling and drive strength of the ENGCLK pin. The ENGCLK pin is enabled and disabled by setting and clearing the OBE bit. The ENGCLK pin is enabled during reset.

SIU_BASE+0x1EC

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	OBE	0	DSC		0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0

Figure 6-115. ENGLCK Pad Configuration Register (SIU_PCR214)

See Table 6-15 for bit field definitions.

6.3.1.12.104Pad Configuration Register 215 (SIU_PCR215)

The SIU_PCR215 register controls the pin function, direction, and static electrical attributes of the AN[12]_MA[0]_SDS pin.

SIU_BASE+0x1EE

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA[1]		0	0	0	0	ODE	0	SRC		0	0
W																
RESET:	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0

1 Input and output buffers are enabled/disabled based on PA selection. Both input and output buffer disabled for AN[12] function. Output buffer only enabled for MA[0] and $\overline{\text{SDS}}$ functions.

Figure 6-116. AN[12]_MA[0]_$\overline{\text{SDS}}$ Pad Configuration Register (SIU_PCR215)

See Table 6-15 for bit field definitions. The PA field for PCR215 is given in Table 6-21.

Table 6-21. PCR215 PA Field Definition

PA Field	Pin Function
0b00	SDS
0b01	Reserved
0b10	MA[0]
0b11	AN[12]

6.3.1.12.105Pad Configuration Register 216 (SIU_PCR216)

The SIU_PCR216 register controls the pin function, direction, and static electrical attributes of the AN[13]_MA[1]_SDO pin.

SIU_BASE+0x1F0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA[1]		0	0	0	0	ODE	0	SRC		0	0
W																
RESET:	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0

[1] Input and output buffers are enabled/disabled based on PA selection. Both input and output buffer disabled for AN[13] function. Output buffer only enabled for MA[1] and SDO functions.

Figure 6-117. AN[13]_MA[1]_SDO Pad Configuration Register (SIU_PCR216)

See Table 6-15 for bit field definitions. The PA field for PCR216 is given in Table 6-22.

Table 6-22. PCR216 PA Field Definition

PA Field	Pin Function
0b00	SDO
0b01	Reserved
0b10	MA[1]
0b11	AN[13]

6.3.1.12.106Pad Configuration Register 217 (SIU_PCR217)

The SIU_PCR217 register controls the pin function, direction, and static electrical attributes of the AN[14]_MA[2]_SDI pin.

SIU_BASE+0x1F2

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PA[1]		0	0	0	0	ODE	HYS	SRC		WPE[2]	WPS[3]
W																
RESET:	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0

1 Input and output buffers are enabled/disabled based on PA selection. Both input and output buffer disabled for AN[14] function. Output buffer only enabled for MA[2] function and input buffer only enabled for SDI functions.

2 The WPE bit should be set to zero when configured as an analog input or MA[2], and set to one when configured as SDI.

3 The WPS bit should be set to one when configured as SDI.

Figure 6-118. AN[14]_MA[2]_SDI Pad Configuration Register (SIU_PCR217)

See Table 6-15 for bit field definitions. The PA field for PCR217 is given in Table 6-23.

Table 6-23. PCR217 PA Field Definition

PA Field	Pin Function
0b00	SDI
0b01	Reserved
0b10	MA[2]
0b11	AN[14]

6.3.1.12.107Pad Configuration Register 218 (SIU_PCR218)

The SIU_PCR218 register controls the pin function, direction, and static electrical attributes of the AN[15]_FCK pin.

SIU_BASE+0x1F4

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	PA[1]	0	0	0	0	ODE	0	SRC		0	0
W																
RESET:	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

1 Input and output buffers are enabled/disabled based on PA selection. Both input and output buffer disabled for AN[15] function. Output buffer only enabled for FCK function.

Figure 6-119. AN[15]_FCK Pad Configuration Register (SIU_PCR218)

See Table 6-15 for bit field definitions. The PA field for PCR218 is given in Table 6-24.

Table 6-24. PCR218 PA Field Definition

PA Field	Pin Function
0b0	FCK
0b1	AN[15]

6.3.1.12.108Pad Configuration Register 219 (SIU_PCR219)

The SIU_PCR219 register controls the drive strength of the MCKO pin.

SIU_BASE+0x1F6

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	DSC		0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0

Figure 6-120. MCKO Pad Configuration Register (SIU_PCR219)

See Table 6-15 for bit field definitions.

6.3.1.12.109Pad Configuration Register 220 - 223 (SIU_PCR220 - SIU_PCR223)

The SIU_PCR220 - SIU_PCR223 registers control the drive strength of the MDO[0:3] pins.

SIU_BASE+0x1F8 -SIU_BASE+0x1FE (4)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	DSC		0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0

Figure 6-121. MDO[0:3] Pad Configuration Register (SIU_PCR220 - SIU_PCR223)

See Table 6-15 for bit field definitions.

6.3.1.12.110Pad Configuration Register 224 - 225 (SIU_PCR224 - SIU_PCR225)

The SIU_PCR224 - SIU_PCR225 registers control the drive strength of the $\overline{\text{MSEO}}$[0:1] pins.

SIU_BASE+0x200 -SIU_BASE+0x202 (2)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	DSC		0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0

Figure 6-122. $\overline{\text{MSEO}}$[0:1] Pad Configuration Register (SIU_PCR224 - SIU_PCR225)

See Table 6-15 for bit field definitions.

6.3.1.12.111Pad Configuration Register 226 (SIU_PCR226)

The SIU_PCR226 register controls the drive strength of the $\overline{\text{RDY}}$ pin.

SIU_BASE+0x204

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	DSC		0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0

Figure 6-123. $\overline{\text{RDY}}$ Pad Configuration Register (SIU_PCR226)

See Table 6-15 for bit field definitions.

6.3.1.12.112Pad Configuration Register 227 (SIU_PCR227)

The SIU_PCR227 register controls the drive strength of the $\overline{\text{EVTO}}$ pin.

SIU_BASE+0x206

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	DSC		0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0

Figure 6-124. $\overline{\text{EVTO}}$ Pad Configuration Register (SIU_PCR227)

See Table 6-15 for bit field definitions.

6.3.1.12.113Pad Configuration Register 228 (SIU_PCR228)

The SIU_PCR228 register controls the drive strength of the TDO pin.

SIU_BASE+0x208

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	DSC		0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0

Figure 6-125. TDO Pad Configuration Register (SIU_PCR228)

See Table 6-15 for bit field definitions.

6.3.1.12.114Pad Configuration Register 229 (SIU_PCR229)

The SIU_PCR229 register controls the enabling/disabling and drive strength of the CLKOUT pin. The CLKOUT pin is enabled and disabled by setting and clearing the OBE bit. The CLKOUT pin is enabled during reset.

SIU_BASE+0x20A

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	OBE	0	DSC		0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0

Figure 6-126. CLKOUT Pad Configuration Register (SIU_PCR229)

See Table 6-15 for bit field definitions.

6.3.1.12.115Pad Configuration Register 230 (SIU_PCR230)

The SIU_PCR230 register controls the slew rate of the $\overline{\text{RSTOUT}}$ pin.

SIU_BASE+0x20C

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	SRC		0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0

Figure 6-127. $\overline{\text{RSTOUT}}$ Pad Configuration Register (SIU_PCR230)

See Table 6-15 for bit field definitions.

6.3.1.13 GPIO Pin Data Output Registers 0–213 (SIU_GPDO*n*)

The definition of the 8-bit SIU_GPDO*n* registers, with each register specifying the drive data for a single GPIO pin, is given in Figure 6-128. The *n* notation in the name of the 214 SIU_GPDO*n* registers corresponds to the pins with the same GPIO pin numbers. For example, PDO0 is the pin data output bit for the $\overline{\text{CS0}}$_GPIO0 pin and is found in SIU_GPDO0, and PDO213 is the pin data output bit for the WKPCFG_GPIO213 pin and is found in SIU_GPDO213. The GPDO address for a particular pin is equal to the GPIO pin number with an offset of SIU_BASE + 0x600.

The SIU_GPDO*n* registers are written to by software to drive data out on the external GPIO pin. Each register drives a single external GPIO pin, which allows the state of the pin to be controlled independently from other GPIO pins. Writes to the SIU_GPDO*n* registers have no effect on pin states if the pins are configured as inputs by the associated Pad Configuration Registers. The SIU_GPDO*n* register values are automatically driven to the GPIO pins without software update if the direction of the GPIO pins is changed from input to output.

Writes to the SIU_GPDO*n* registers have no effect on the state of the corresponding pins when the pins are configured for their primary function by the corresponding PCR.

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	0
W								PDO*n*
Reset	0	0	0	0	0	0	0	0
Reg Addr	SIU_BASE + 0x600 + *n*							

Figure 6-128. GPIO Pin Data Output Register 0–213 (SIU_GPDO*n*)

Table 6-25. SIU_GPDO*n* Field Descriptions

Name	Description
PDO*n*	Pin data out. Stores the data to be driven out on the external GPIO pin associated with the register. If the register is read, it will return the value written. 0 V_{OL} is driven on the external GPIO pin when the pin is configured as an output. 1 V_{OH} is driven on the external GPIO pin when the pin is configured as an output.

6.3.1.14 GPIO Pin Data Input Registers 0–213 (SIU_GPDI*n*)

The definition of the 8-bit SIU_GPDI*n* registers, with each register specifying the drive data for a single GPIO pin, is given in Figure 6-129. The *n* notation in the name of the 178 (MPC5553) or 214 (MPC5554) SIU_GPDI*n* registers corresponds to the pins with the same GPIO pin numbers. For example, PDI0 is the pin data input bit for the CS0_GPIO0 pin and is found in SIU_GPDI0, and PDI213 is the pin data input bit for the WKPCFG_GPIO213 pin and is found in SIU_GPDI213. The GPDI address for a particular pin is equal to the GPIO pin number with an offset of SIU_BASE + 0x800. Gaps exist in this memory space where the pin is not available in the package.

The SIU_GPDI*n* registers are read-only registers that allow software to read the input state of an external GPIO pin. Each register represents the input state of a single external GPIO pin. If the GPIO pin is configured as an output, and the input buffer enable (IBE) bit is set in the associated pad configuration register, the SIU_GPDI*n* register reflects the actual state of the output pin.

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	PDI*n*
W								
Reset	0	0	0	0	0	0	0	0
Reg Addr	SIU_BASE + 0x800 + *n*							

Figure 6-129. GPIO Pin Data Input Register 0–213 (SIU_GPDI*n*)

Table 6-26. SIU_GPDI*n* Field Description

Name	Description
PDI*n*	Pin data in. This bit reflects the input state on the external GPIO pin associated with the register. If PCR*n*[IBE] = 1, then: 0 Signal on pin is less than or equal to V_{IL}. 1 Signal on pin is greater than or equal to V_{IH}.

6.3.1.15 eQADC Trigger Input Select Register (SIU_ETISR)

The SIU_ETISR selects the source for the eQADC trigger inputs. The eQADC trigger numbers 0-5 specified by TSEL(0-5) correspond to CFIFO numbers 0-5. To calculate the CFIFO number that each trigger is connected to, divide the DMA channel number by 2. So, for example, eQADC CFIFO 1 (connected to DMA channel 2) can be triggered by eTPUA[31], eMIOS[11] or ETRIG[1]. To select a trigger, the corresponding TSEL must be initialized.

When an eQADC trigger is connected, the timer output is connected to the eQADC CFIFO trigger input. To trigger the eQADC, the eTPU output must change to the state that the eQADC recognizes as a trigger. Bear in mind there are rising or falling edges, and low or high gated trigger types, so it might be possible to have the eQADC trigger immediately if desired.

Table 6-27. Trigger Interconnections

TSEL Field (Trigger Number)	eQADC CFIFO	EQADC DMA Channel	eTPUA Channel	eMIOS Channel	ETRIG Input
0	0	0	eTPUA30	eMIOS10	ETRIG0
1	1	2	eTPUA31	eMIOS11	ETRIG1
2	2	4	eTPUA29	eMIOS15	ETRIG0
3	3	6	eTPUA28	eMIOS14	ETRIG1
4	4	8	eTPUA27	eMIOS13	ETRIG0
5	5	10	eTPUA26	eMIOS12	ETRIG1

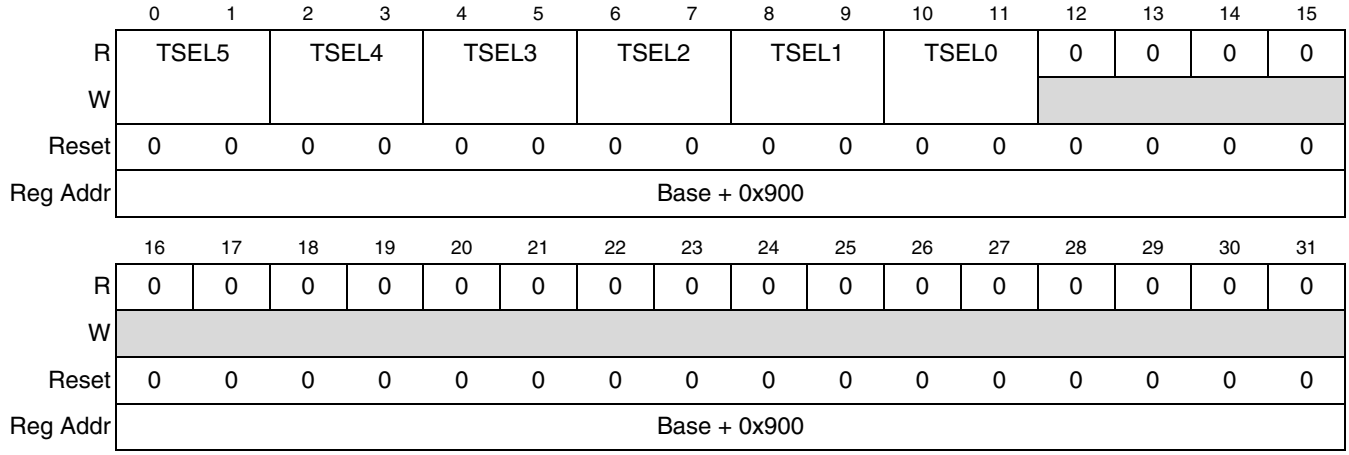


Figure 6-130. eQADC Trigger Input Select Register (SIU_ETISR)

Table 6-28. SIU_ETISR Field Descriptions

Bits	Name	Description
0–1	TSEL5 [0:1]	eQADC trigger input select 5. Specifies the input for eQADC trigger 5. 00 GPIO207 01 ETPUA26 channel 10 EMIOS12 channel 11 ETRIG1 pin
2–3	TSEL4 [0:1]	eQADC trigger input select 4. Specifies the input for eQADC trigger 4. 00 GPIO206 01 ETPUA27 channel 10 EMIOS13 channel 11 ETRIG0 pin
4–5	TSEL3 [0:1]	eQADC trigger input select 3. Specifies the input for eQADC trigger 3. 00 GPIO207 01 ETPUA28 channel 10 EMIOS14 channel 11 ETRIG1 pin
6–7	TSEL2 [0:1]	eQADC trigger input select 2. Specifies the input for eQADC trigger 2 00 GPIO206 01 ETPUA29 channel 10 EMIOS15 channel 11 ETRIG0 pin
8–9	TSEL1 [0:1]	eQADC trigger input select 1. Specifies the input for eQADC trigger 1 00 GPIO207 01 ETPUA31 channel 10 EMIOS11 channel 11 ETRIG1 pin
10–11	TSEL0 [0:1]	eQADC trigger input select 0. Specifies the input for eQADC trigger 0 00 GPIO206 01 ETPUA30 channel 10 EMIOS10 channel 11 ETRIG0 pin
12–31	—	Reserved.

6.3.1.16 External IRQ Input Select Register (SIU_EIISR)

The SIU_EIISR selects the source for the external interrupt/DMA inputs.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ESEL15		ESEL14		ESEL13		ESEL12		ESEL11		ESEL10		ESEL9		ESEL8	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x904															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ESEL7		ESEL6		ESEL5		ESEL4		ESEL3		ESEL2		ESEL1		ESEL0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x904															

Figure 6-131. External IRQ Input Select Register 1 (SIU_EIISR)

Table 6-29. SIU_EIISR Field Descriptions

Bits	Name	Description
0–1	ESEL15 [0:1]	External IRQ input select 15. Specifies the input for IRQ 15. 00 IRQ15 pin 01 DSPI_B15 serialized input (EMIOS12 pin) 10 DSPI_C0 serialized input (ETPUA12 pin) 11 DSPI_D1 serialized input (ETPUA20 pin)
2–3	ESEL14 [0:1]	External IRQ input select 14. Specifies the input for IRQ14. 00 IRQ14 pin 01 DSPI_B14 serialized input (EMIOS13 pin) 10 DSPI_C15 serialized input (ETPUA11 pin) 11 DSPI_D0 serialized input (ETPUA21 pin)
4–5	ESEL13 [0:1]	External IRQ input select 13. Specifies the input for IRQ13. 00 IRQ13 pin 01 DSPI_B13 serialized input (ETPUA24 pin) 10 DSPI_C14 serialized input (ETPUA10 pin) 11 DSPI_D15 serialized input (ETPUA24 pin)
6–7	ESEL12 [0:1]	External IRQ input select 12. Specifies the input for IRQ12. 00 IRQ12 pin 01 DSPI_B12 serialized input (ETPUA25 pin) 10 DSPI_C13 serialized input (ETPUA9 pin) 11 DSPI_D14 serialized input (ETPUA25 pin)
8–9	ESEL11 [0:1]	External IRQ input select 11. Specifies the input for IRQ11. 00 IRQ11 pin 01 DSPI_B11 serialized input (ETPUA26 pin) 10 DSPI_C12 serialized input (ETPUA8 pin) 11 DSPI_D13 serialized input (ETPUA26 pin)
10–11	ESEL10 [0:1]	External IRQ input select 10. Specifies the input for IRQ10. 00 IRQ10 pin 01 DSPI_B10 serialized input (ETPUA27 pin) 10 DSPI_C11 serialized input (ETPUA7 pin) 11 DSPI_D12 serialized input (ETPUA27 pin)

Table 6-29. SIU_EIISR Field Descriptions (continued)

Bits	Name	Description
12–13	ESEL9 [0:1]	External IRQ input select 9. Specifies the input for IRQ9. 00 IRQ9 pin 01 DSPI_B9 serialized input (ETPUA28 pin) 10 DSPI_C10 serialized input (ETPUA6 pin) 11 DSPI_D11 serialized input (ETPUA28 pin)
14–15	ESEL8 [0:1]	External IRQ input select 8. Specifies the input for IRQ8. 00 IRQ8 pin 01 DSPI_B8 serialized input (ETPUA29 pin) 10 DSPI_C9 serialized input (ETPUA5 pin) 11 DSPI_D10 serialized input (ETPUA29 pin)
16–17	ESEL7 [0:1]	External IRQ input select 7. Specifies the input for IRQ7. 00 IRQ7 pin 01 DSPI_B7 serialized input (ETPUA16 pin) 10 DSPI_C8 serialized input (ETPUA4 pin) 11 DSPI_D9 serialized input (EMIOS12 pin)
18–19	ESEL6 [0:1]	External IRQ input select 6. Specifies the input for IRQ6. 00 IRQ6 pin (for MPC5553, 0b00 is Reserved) 01 DSPI_B6 serialized input (ETPUA17 pin) 10 DSPI_C7 serialized input (ETPUA3 pin) 11 DSPI_D8 serialized input (EMIOS13 pin) **Note:** IRQ6 functions only on the MPC5554. It is not functional on the MPC5553.
20–21	ESEL5 [0:1]	External IRQ input select 5. Specifies the input for IRQ5. 00 IRQ5 pin 01 DSPI_B5 serialized input (ETPUA18 pin) 10 DSPI_C6 serialized input (ETPUA2 pin) 11 DSPI_D7 serialized input (EMIOS10 pin)
22–23	ESEL4 [0:1]	External IRQ input select 4. Specifies the input for IRQ4. 00 IRQ4 pin 01 DSPI_B4 serialized input (ETPUA19 pin) 10 DSPI_C5 serialized input (ETPUA1 pin) 11 DSPI_D6 serialized input (EMIOS11 pin)
24–25	ESEL3 [0:1]	External IRQ input select 3. Specifies the input for IRQ3. 00 IRQ3 pin 01 DSPI_B3 serialized input (ETPUA20 pin) 10 DSPI_C4 serialized input (ETPUA0 pin) 11 DSPI_D5 serialized input (ETPUA16 pin)
26–27	ESEL2 [0:1]	External IRQ input select 2. Specifies the input for IRQ2. 00 IRQ2 pin 01 DSPI_B2 serialized input (ETPUA21 pin) 10 DSPI_C3 serialized input (ETPUA15 pin) 11 DSPI_D4 serialized input (ETPUA17 pin)

Table 6-29. SIU_EIISR Field Descriptions (continued)

Bits	Name	Description
28–29	ESEL1 [0:1]	External IRQ input select 1. Specifies the input for IRQ1. 00 IRQ1 pin 01 DSPI_B1 serialized input (EMIOS10 pin) 10 DSPI_C2 serialized input (ETPUA14 pin) 11 EMIOS15 pin
30–31	ESEL0 [0:1]	External IRQ input select 0. Specifies the input for IRQ0. 00 IRQ0 pin 01 DSPI_B0 serialized input (EMIOS11 pin) 10 DSPI_C1 serialized input (ETPUA5 pin) 11 EMIOS14 pin

6.3.1.17 DSPI Input Select Register (SIU_DISR)

The SIU_DISR specifies the source of each DSPI data input, slave select, clock input, and trigger input to allow serial and parallel chaining of the DSPI modules. For MPC5553, see Figure 6-132. For MPC5554 see Figure 6-133.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	SINSELB		SSSELB		SCKSELB		TRIGSELB	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x908															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SINSELC		SSSELC		SCKSELC		TRIGSELC		SINSELD		SSSELD		SCKSELD		TRIGSELD	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x908															

Figure 6-132. MPC5553 DSPI Input Select Register (SIU_DISR)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SINSELA		SSSELA		SCKSELA		TRIGSELA		SINSELB		SSSELB		SCKSELB		TRIGSELB	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x908															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SINSELC		SSSELC		SCKSELC		TRIGSELC		SINSELD		SSSELD		SCKSELD		TRIGSELD	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x908															

Figure 6-133. MPC5554 DSPI Input Select Register (SIU_DISR)

Table 6-30. SIU_DISR Field Descriptions

Bits	Name	Description
0–1	SINSELA [0:1]	DSPI_A data input select. Specifies the source of the DSPI_A data input. 00 SINA_GPIO94 pin 01 SOUTB 10 SOUTC 11 SOUTD **Note:** MPC5553: Bits 0–1 are reserved
2–3	SSSELA [0:1]	DSPI_A slave select input select. Specifies the source of the DSPI_A slave select input. 00 PCSA0_GPIO96 pin 01 PCSB0 (Master) 10 PCSC0 (Master) 11 PCSD0 (Master) MPC5553: Bits 2–3 are reserved
4–5	SCKSELA [0:1]	DSPI_A clock input select. Specifies the source of the DSPI_A clock input. 00 SCKA_GPIO93 pin 01 SCKB (Master) 10 SCKC (Master) 11 SCKD (Master) MPC5553: Bits 4–5 are reserved
6–7	TRIGSELA [0:1]	DSPI_A trigger input select. Specifies the source of the DSPI_A trigger input. 00 No Trigger 01 PCSB4 10 PCSC4 11 PCSD4 MPC5553: Bits 6–7 are reserved
8–9	SINSELB [0:1]	DSPI_B data input select. Specifies the source of DSPI_B data input. 00 SINB_PCSC2_GPIO103 pin 01 SOUTA (not available for MPC5553) 10 SOUTC 11 SOUTD

Table 6-30. SIU_DISR Field Descriptions (continued)

Bits	Name	Description
10–11	SSSELB [0:1]	DSPI_B slave select input select. Specifies the source of the DSPI_B slave select input. 00 PCSB0_PCSD2_GPIO105 pin 01 PCSA0 (Master) (not available for MPC5553) 10 PCSC0 (Master) 11 PCSD0 (Master)
12–13	SCKSELB [0:1]	DSPI_B clock input select. Specifies the source of the DSPI_B clock input. 00 SCKB_PCSC1_GPIO102 pin 01 SCKA (Master) (not available for MPC5553) 10 SCKC (Master) 11 SCKD (Master)
14–15	TRIGSELB [0:1]	DSPI_B trigger input select. Specifies the source of the DSPI_B trigger input for master or slave mode. 00 Reserved 01 PCSA4 (not available for MPC5553) 10 PCSC4 11 PCSD4
16–17	SINSELC [0:1]	DSPI_C data input select. Specifies the source of the DSPI_C data input. 00 PCSA2_SINC_GPIO108 pin 01 SOUTA (not available for MPC5553) 10 SOUTB 11 SOUTD
18–19	SSSELC [0:1]	DSPI_C slave select input select. Specifies the source of the DSPI_C slave select input. 00 PCSB5_PCSC0_GPIO110 pin 01 PCSA0 (Master) (not available for MPC5553) 10 PCSB0 (Master) 11 PCSD0 (Master)
20–21	SCKSELC [0:1]	DSPI_C clock input select. Specifies the source of the DSPI_C clock input when in slave mode. 00 PCSB4_SCKC_GPIO109 pin 01 SCKA (Master) (not available for MPC5553) 10 SCKB (Master) 11 SCKD (Master)
22–23	TRIGSELC [0:1]	DSPI_C trigger input select. Specifies the source of the DSPI_C trigger input for master or slave mode. 00 Reserved 01 PCSA4 (not available for MPC5553) 10 PCSB4 11 PCSD4
24–25	SINSELD [0:1]	DSPI_D data input select. Specifies the source of the DSPI_D data input. 00 PCSA3_SIND_GPIO99 pin 01 SOUTA (not available for MPC5553) 10 SOUTB 11 SOUTC

Table 6-30. SIU_DISR Field Descriptions (continued)

Bits	Name	Description
26–27	SSSELD [0:1]	DSPI_D slave select input select. Specifies the source of the DSPI_D slave select input. 00 PCSB1_PCSD0_GPIO106 pin 01 PCSA0 (Master) (not available for MPC5553) 10 PCSB0 (Master) 11 PCSC0 (Master)
28–29	SCKSELD [0:1]	DSPI_D clock input select. Specifies the source of the DSPI_D clock input in slave mode. 00 PCSA2_SCKD_GPIO98 pin 01 SCKA (Master) (not available for MPC5553) 10 SCKB (Master) 11 SCKC (Master)
30–31	TRIGSELD [0:1]	DSPI_D trigger input select. Specifies the source of the DSPI_D trigger input for master or slave mode. 00 Reserved 01 PCSA4 (not available for MPC5553) 10 PCSB4 11 PCSC4

6.3.1.18 Chip Configuration Register (SIU_CCR)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	MATCH	DISNEX
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X[1]
Reg Addr	Base + 0x980															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TEST[2]
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x980															

1 When system reset negates, the value in this bit depends on the censorship control word and the boot configuration bits.

2 This bit is reset with a power on reset.

Figure 6-134. Chip Configuration Register (SIU_CCR)

Table 6-31. SIU_CCR Field Descriptions

Bits	Name	Description
0–13	—	Reserved.
14	MATCH	Compare register match. Holds the value of the match input signal to the SIU. The match input is asserted if the values in the SIU_CARH/SIU_CARL and SIU_CBRH/SIU_CBRL are equal. The MATCH bit is reset by the synchronous reset signal. 0 The content of SIU_CARH/SIU_CARL does not match the content of SIU_CBRH/SIU_CBRL 1 The content of SIU_CARH/SIU_CARL matches the content of SIU_CBRH/SIU_CBRL
15	DISNEX	Disable Nexus. Holds the value of the Nexus disable input signal to the SIU. When system reset negates, the value in this bit depends on the censorship control word and the boot configuration bits. 0 Nexus disable input signal is negated. 1 Nexus disable input signal is asserted.
16–30	—	Reserved.
31	TEST	Test mode enable. Allows reads or writes to undocumented registers used only for production tests. Since these production test registers are undocumented, estimating the impact of errant accesses to them is impossible. The application should not change this bit from its negated state at reset. 0 Undocumented production test registers can not be read or written. 1 Undocumented production test registers can be read or written.

6.3.1.19 External Clock Control Register (SIU_ECCR)

The SIU_ECCR controls the timing relationship between the system clock and the external clocks ENGCLK and CLKOUT. All bits and fields in the SIU_ECCR are read/write and are reset by the synchronous reset signal.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x984															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	ENGDIV						0	0	0	0	EBTS	0	EBDF	
W																
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
Reg Addr	Base + 0x984															

Figure 6-135. External Clock Control Register (SIU_ECCR)

Table 6-32. SIU_ECCR Field Descriptions

Bits	Name	Description
0–17	—	Reserved.
18–23	ENGDIV [0:5]	Engineering clock division factor. Specifies the frequency ratio between the system clock and ENGCLK. The ENGCLK frequency is divided from the system clock frequency according to the following equation: $\text{Engineering clock frequency} = \frac{\text{System clock frequency}}{\text{ENGDIV} \times 2}$ **Note:** Clearing ENGDIV to 0 is reserved. Synchronization between ENGCLK and CLKOUT cannot be guaranteed.
24–27	—	Reserved.
28	EBTS	External bus tap select. Changes the phase relationship between the system clock and CLKOUT. Changing the phase relationship so that CLKOUT is advanced in relation to the system clock increases the output hold time of the external bus signals to a non-zero value. It also increases the output delay times, increases the input hold times to non-zero values, and decreases the input setup times. Refer to the Electrical Specifications for how the EBTS bit affects the external bus timing. 0 External bus signals have zero output hold times. 1 External bus signals have non-zero output hold times. **Note:** The EBTS bit must not be modified while an external bus transaction is in progress.
29	—	Reserved.
30–31	EBDF [0:1]	External bus division factor. Specifies the frequency ratio between the system clock and the external clock, CLKOUT. The EBDF field must not be changed during an external bus access or while an access is pending. The CLKOUT frequency is divided from the system clock frequency according to the descriptions below. This divider must be kept as divide-by-2 when operating in dual controller mode. 00 Reserved 01 Divide by 2 10 Reserved 11 Divide by 4

6.3.1.20 Compare A High Register (SIU_CARH)

The compare registers are not intended for general application use, but are used temporarily by the BAM during boot and intended optionally for communication with calibration tools. After reset, calibration tools can immediately write a non-zero value to these registers. The application code, using the registers then as read only, can read them to determine if a calibration tool is attached and operate appropriately.

The compare registers can be used just like 128 bits of memory mapped RAM that is always zero out of reset, or they can perform a 64 bit to 64 bit compare. The compare function is continuous (combinational logic - not requiring a start or stop). The compare result appears in the MATCH bit in the SIU_CCR register.

The SIU_CARH holds the 32-bit value that is compared against the value in the SIU_CBRH register. The CMPAH field is read/write and is reset by the synchronous reset signal.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CMPAH															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x988															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CMPAH															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x988															

Figure 6-136. Compare A High Register (SIU_CARH)

6.3.1.21 Compare A Low Register (SIU_CARL)

The SIU_CARL register holds the 32-bit value that is compared against the value in the SIU_CBRL register. The CMPAL field is read/write and is reset by the synchronous reset signal.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CMPAL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x98C															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CMPAL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x98C															

Figure 6-137. Compare A Low Register (SIU_CARL)

6.3.1.22 Compare B High Register (SIU_CBRH)

The SIU_CBRH holds the 32-bit value that is compared against the value in the SIU_CARH. The CMPBH field is read/write and is reset by the synchronous reset signal.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CMPBH															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x990															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CMPBH															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x990															

Figure 6-138. Compare B High Register (SIU_CBRH)

6.3.1.23 Compare B Low Register (SIU_CBRL)

The SIU_CBRL holds the 32-bit value that is compared against the value in the SIU_CARL. The CMPBL field is read/write and is reset by the synchronous reset signal.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CMPBL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x994															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CMPBL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x994															

Figure 6-139. Compare B Low Register (SIU_CBRL)

6.4 Functional Description

The following sections provide an overview of the SIU operation.

6.4.1 System Configuration

6.4.1.1 Boot Configuration

The BOOTCFG[0:1] pins are used to determine the boot mode initiated by the BAM program, and whether external arbitration is selected for external booting. The BAM program uses the BOOTCFG field to determine where to read the reset configuration word, and whether to initiate a CAN or SCI boot. See Section 16.3.2.2.5, "Reset Configuration Half Word Read" of the BAM chapter for detail on the RCHW. Table 6-33 defines the boot modes specified by the BOOTCFG[0:1] pins. If the RSTCFG pin is asserted during the assertion of RSTOUT, except in the case of a software external reset, the BOOTCFG pins are latched 4 clock cycles prior to the negation of the RSTOUT pin and are used to update the SIU_RSR and the BAM boot mode. Otherwise, if RSTCFG is negated during the assertion of RSTOUT, the BOOTCFG pins are ignored and the boot mode defaults to 'Boot from Internal Flash Memory.'

Table 6-33. BOOTCFG[0:1] Configuration

Value	Meaning
0b00	Boot from Internal Flash Memory
0b01	CAN/SCI Boot
0b10	Boot from External Memory (No Arbitration)
0b11	Boot from External Memory (External Arbitration)

6.4.1.2 Pad Configuration

The pad configuration registers (SIU_PCR) in the SIU allow software control of the static electrical characteristics of external pins. The pad configuration registers allow control over the following external pin characteristics:

- Weak pull up/down enable/disable
- Weak pull up/down selection
- Slew-rate selection for outputs
- Drive strength selection for outputs
- Input buffer enable (when direction is configured for output)
- Input hysteresis enable/disable
- Open drain/push-pull output selection
- Multiplexed function selection
- Data direction selection

The pad configuration registers are provided to allow centralized control over external pins that are shared by more than one module. Each pad configuration register controls a single pin.

6.4.2 Reset Control

The reset controller logic is located in the SIU. See the Reset section of this manual for detail on reset operation.

6.4.2.1 $\overline{\text{RESET}}$ Pin Glitch Detect

The reset controller provides a glitch detect feature on the $\overline{\text{RESET}}$ pin. If the reset controller detects that the $\overline{\text{RESET}}$ pin is asserted for more than 2 clock cycles, the clock cycles the reset controller sets the RGF bit without affecting any of the other bits in the reset status register. The RGF bit remains set until cleared by software or the $\overline{\text{RESET}}$ pin is asserted for 10 clock cycles. The reset controller does not respond to assertions of the $\overline{\text{RESET}}$ pin if a reset cycle is already being processed.

6.4.3 External Interrupt

There are sixteen external interrupt inputs IRQ0–IRQ15 to the SIU. The IRQ*n* inputs can be configured for rising or falling edge events or both. Each IRQ*n* input has a corresponding flag bit in the external interrupt status register (SIU_EISR). The flag bits for the IRQ[4:15] inputs are OR'ed together to form one interrupt request to the interrupt controller (OR function performed in the integration glue logic). The flag bits for the IRQ[0:3] inputs can generate either an interrupt request to the interrupt controller or a DMA transfer request to the DMA controller. Table 6-140 shows the DMA and interrupt request connections to the interrupt and DMA controllers.

The SIU contains an overrun request for each IRQ and one combined overrun request which is the logical OR of the individual overrun requests. Only the combined overrun request is used in the MPC5553/MPC5554, and the individual overrun requests are not connected.

Each IRQ pin has a programmable filter for rejecting glitches on the IRQ signals. The filter length for the IRQ pins is specified in the external IRQ digital filter register (SIU_IDFR).

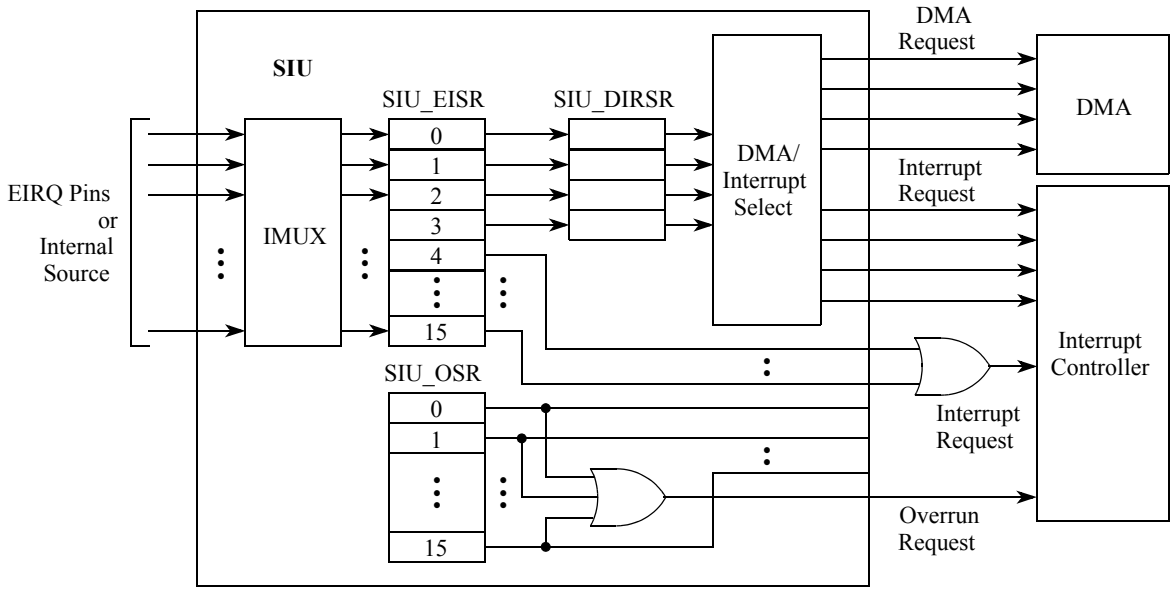


Figure 6-140. SIU DMA/Interrupt Request Diagram

6.4.4 GPIO Operation

All GPIO functionality is provided by the SIU for the MPC5553/MPC5554. Each MPC5553/MPC5554 pin that has GPIO functionality has an associated pin configuration register in the SIU where the GPIO function is selected for the pin. In addition, each MPC5553/MPC5554 pin with GPIO functionality has an input data register (SIU_GPDI*n*_*n*) and an output data register (SIU_GPDO*n*_*n*).

6.4.5 Internal Multiplexing

The internal multiplexing select registers SIU_ETISR, SIU_EIISR, and SIU_DISR provide selection of the source of the input for the eQADC external trigger inputs, the SIU external interrupts, and the DSPI signals that are used in serial and parallel chaining of the DSPI modules.

A block diagram of the internal multiplexing feature is given in Figure 6-141. The figure shows the multiplexing of four external signals to an output from the SIU. A two bit SEL field from an SIU select register is used to select the input of the multiplexor.

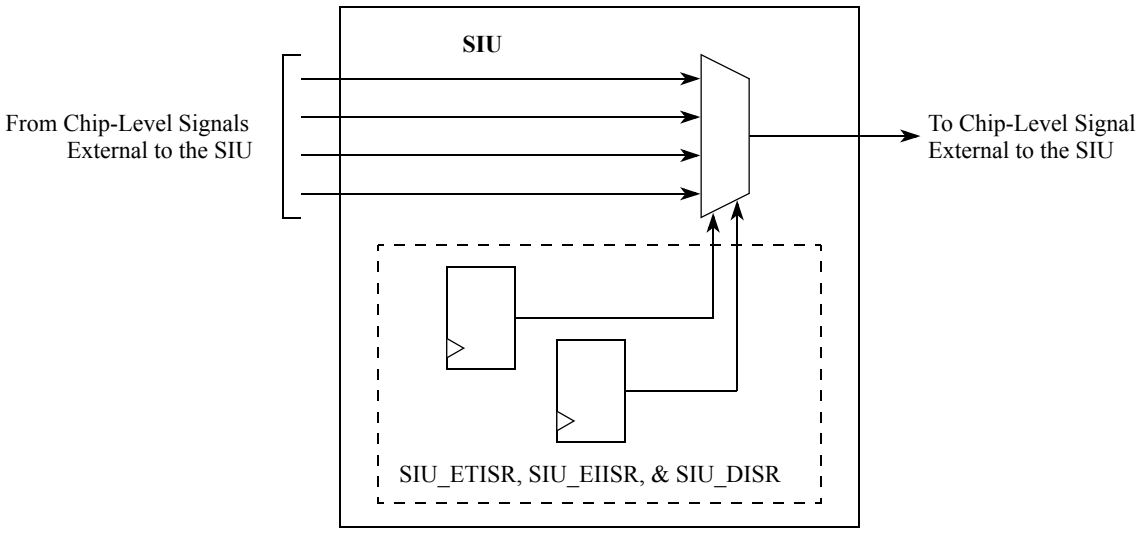


Figure 6-141. Four-to-One Internal Multiplexing Block Diagram

6.4.5.1 eQADC External Trigger Input Multiplexing

The six eQADC external trigger inputs can be connected to either an external pin, an eTPU channel, or an eMIOS channel. The input source for each eQADC external trigger is individually specified in the eQADC trigger input select register (SIU_ETISR). An example of the multiplexing of an eQADC external trigger input is given in Figure 6-142. As shown in the figure, the ETRIG0 input of the eQADC can be connected to either the ETRIG0_GPIO111 pin, the ETPUA30 channel, the EMIOS10 channel, or the GPIO206 pin. The remaining ETRIG inputs are multiplexed in the same manner (see Section 6.3.1.15, “eQADC Trigger Input Select Register (SIU_ETISR)” for the SIU_ETISR[TSEL0]–SIU_ETISR[TSEL5] bit definitions). Note that if an ETRIG input is connected to an eTPU or eMIOS channel, the external pin used by that channel can be used by the alternate function on that pin.

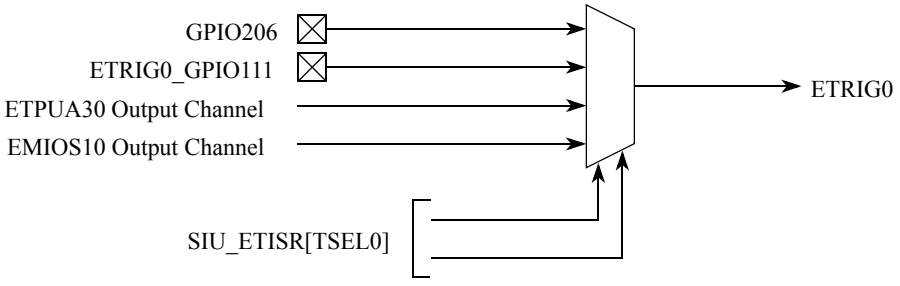


Figure 6-142. eQADC External Trigger Input Multiplexing

6.4.5.2 SIU External Interrupt Input Multiplexing

The sixteen SIU external interrupt inputs can be connected to either an external pin or to serialized output signals from a DSPI module. The input source for each SIU external interrupt is individually specified in the external IRQ input select register (SIU_EIISR). An example of the multiplexing of an SIU external interrupt input is given in Figure 6-143. As shown in the figure, the IRQ0 input of the SIU can be connected to either the IRQ0_GPIO203 pin, the DSPI_B0 serial input signal, the DSPI_C1 deserialized output signal, or the DSPI_D2 deserialized output signal. The remaining IRQ inputs are multiplexed in the same manner. The inputs to the IRQ from each DSPI module are offset by one so that if more than one DSPI module is connected to the same external device type, a separate interrupt can be generated for each device. This also applies to DSPI modules connected to external devices of different type that have status bits in the same bit location of the deserialized information.

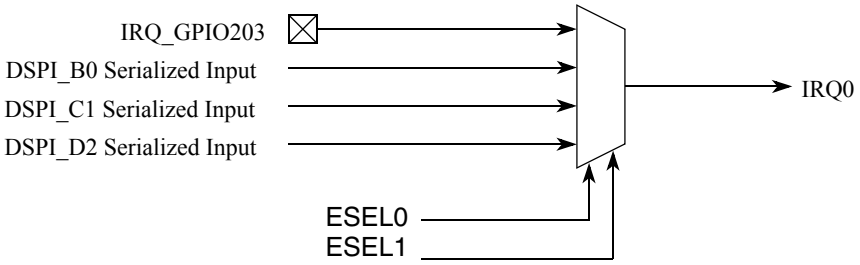


Figure 6-143. DSPI Serialized Input Multiplexing

6.4.5.3 Multiplexed Inputs for DSPI Multiple Transfer Operation

Each DSPI module can be combined in a serial or parallel chain (multiple transfer operation). Serial chaining allows SPI operation with an external device that has more bits than one DSPI module. An example of a serial chain is shown in Figure 6-144. In a serial chain, one DSPI module operates as a master, the second, third, or fourth DSPI modules operate as slaves. The data output (SOUT) of the master is connected to the data input (SIN) of the slave. The SOUT of a slave is connected to the SIN of subsequent slaves until the last module in the chain, where the SOUT is connected to an external pin, which connects to the input of an external SPI device. The slave DSPI and external SPI device use the master peripheral chip select (PCS) and clock (SCK). The trigger input of the master allows a slave DSPI to trigger a transfer when a data change occurs in the slave DSPI and the slave DSPI is operating in change in data mode. The trigger input of the master is connected to MTRIG output of the slave. If more than two DSPIs are chained in change in data mode, a chain must be connected of MTRIG outputs to trigger inputs through the slaves with the last slave MTRIG output connected to the master trigger input.

Parallel chaining allows the PCS and SCK from one DSPI to be used by more than one external SPI device, thus reducing pin utilization of the MPC5553/MPC5554 MCU. An example of a parallel chain is shown in Figure 6-145. In this example, the SOUT and SIN of the two DSPIs connect to separate external SPI devices, which share a common PCS and SCK.

To support multiple transfer operation of the DSPIs, an input multiplexor is required for the SIN, $\overline{SS}$, SCK IN, and trigger signals of each DSPI. The input source for the SIN input of a DSPI can be a pin or the SOUT of any of the other three DSPIs. The input source for the $\overline{SS}$ input of a DSPI can be a pin or the PCS0 of any of the other three DSPIs. The input source for the SCK input of a DSPI can be a pin or the SCK output of any of the other three DSPIs. The input source for the trigger input can be the PCSS output of any of the other three DSPIs. The input source for each DSPI SIN, $\overline{SS}$, SCK, and trigger signal is individually specified in the DSPI input select register (SIU_DISR).

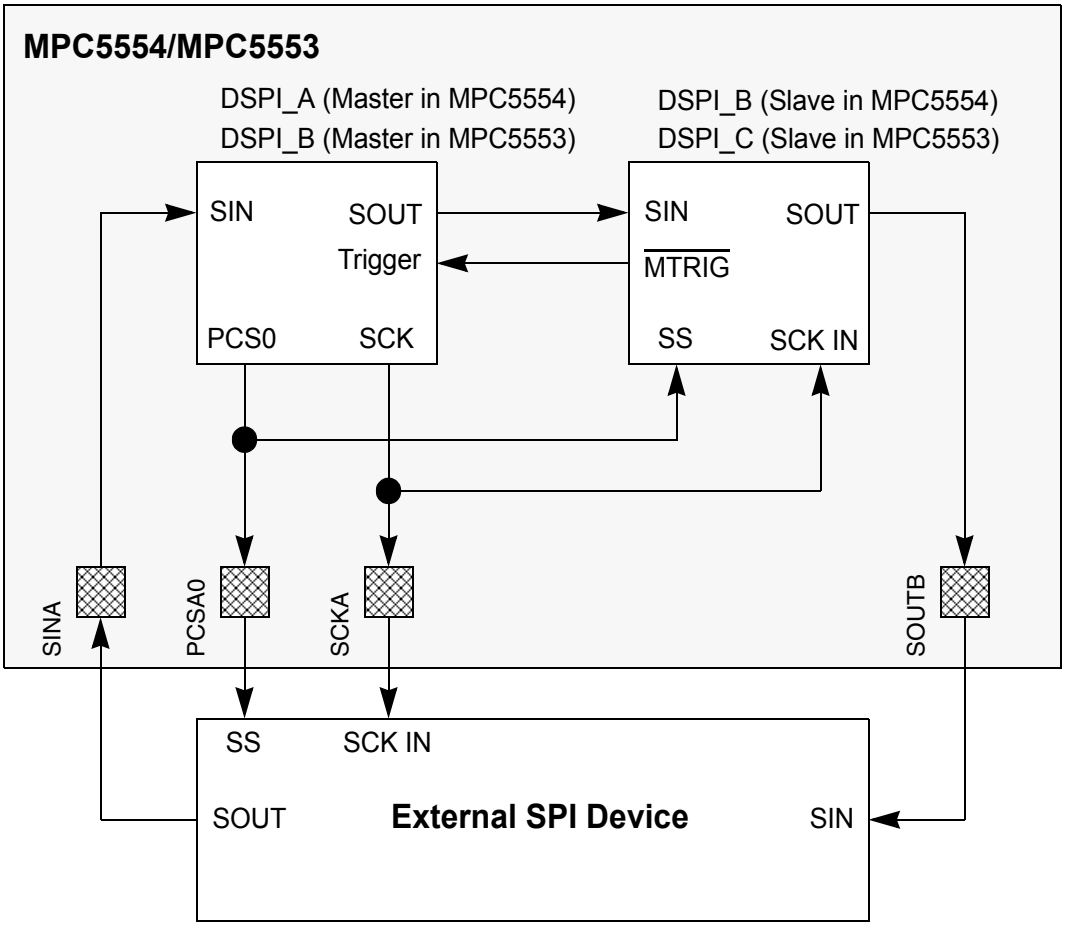


Figure 6-144. DSPI Serial Chaining

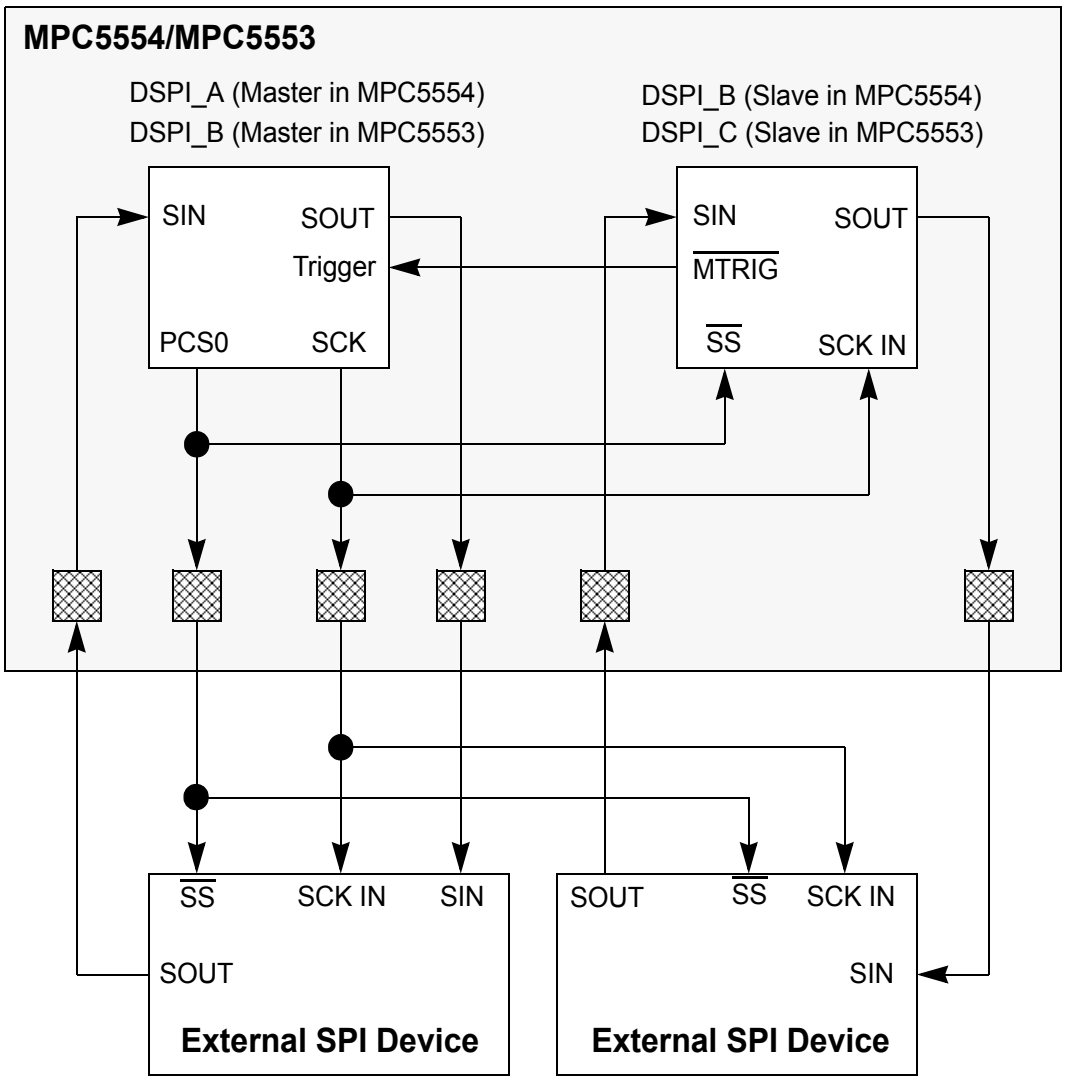


Figure 6-145. DSPI Parallel Chaining

6.5 Revision History

Substantive Changes since Rev 3.0
Changed HYS bit footnotes to say “set to zero” instead of “set to one” for PCRs 4-62 and 69-74.
Removed ns values from SRC bit field in Table 6-15.
Put back in the SIU_GPDO section description (Section 6.3.1.13, “GPIO Pin Data Output Registers 0–213 (SIU_GPDOn)”). It is the same as Rev2.2 of the RM now.
Bits 0-1 for SINSELA. Setting of 11 was "SOUT", changed to "SOUTD" in Table 6-30.
“Reserved” changed to “No Trigger” in Table 6-30 for TRIGSELA bit field.
Changed ODE wording in Table 6-15 to say “Open drain is disabled for the pad (push/pull driver enabled).”
Added Note to Section 6.3.1.12.97, “Pad Configuration Registers 206 - 207 (SIU_PCR206 - SIU_PCR207)” describing ETRIG functionality. NOTE: The GPIO[206:7] pins have the capability to trigger the ADCs. For the ETRIG functionality, these GPIO pins need to be set as GPIO and then select the GPIO ADC trigger in the eQADC Trigger Input Select Register (SIU_ETISR).”
Changed name of MPC5553 signal to be GPIO[203:204]_EMIOS[14:15] (same as MPC5554 now) instead of the other way around and updated the description in Section 6.3.1.12.95, “Pad Configuration Register 203 - 204 (SIU_PCR203 - SIU_PCR204).

Chapter 7
Crossbar Switch (XBAR)

7.1 Introduction

This chapter describes the multi-port crossbar switch (XBAR), which supports simultaneous connections between four (three for MPC5554) master ports and five slave ports. XBAR supports a 32-bit address bus width and a 64-bit data bus width at all master and slave ports.

7.1.1 Block Diagram

Figure 7-1 shows a block diagram of the crossbar switch. Table 7-1 gives the crossbar switch port for each master and slave, and the assigned and fixed ID number for each master.

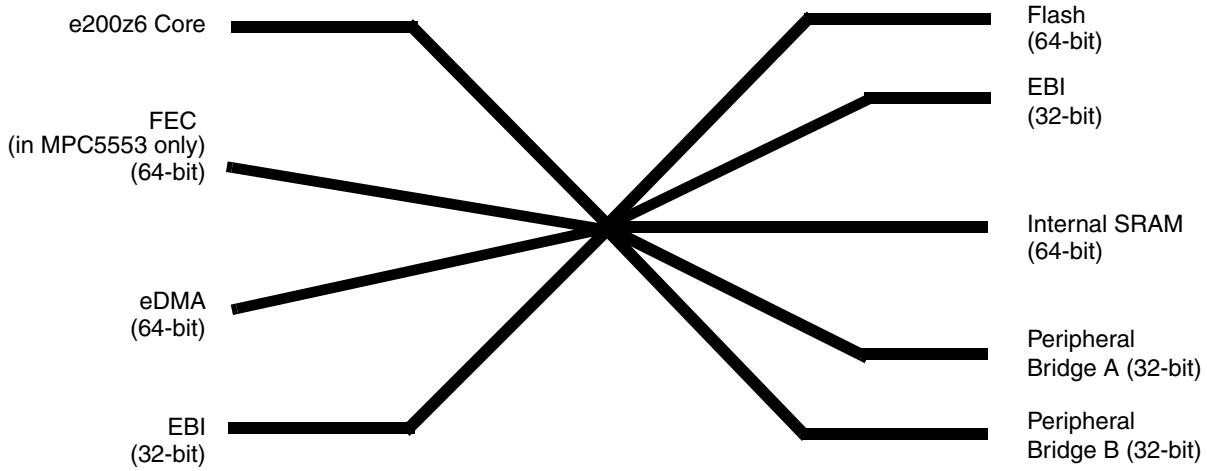


Figure 7-1. XBAR Block Diagram

Table 7-1. XBAR Switch Ports

Module	XBAR Port	Master ID
e200z6 core - CPU instruction/data	Master 0	0
e200z6 - Nexus		1
eDMA	Master 1	2
External Bus Interface	Master 2	3

Table 7-1. XBAR Switch Ports

Module	XBAR Port	Master ID
FEC (MPC5553 only)	Master 3	4
Flash	Slave 0	
External Bus Interface	Slave 1	
Internal SRAM	Slave 3	
Peripheral Bridge A (PBRIDGE_A)	Slave 6	
Peripheral Bridge B (PBRIDGE_B)	Slave 7	

7.1.2 Overview

The XBAR allows for concurrent transactions to occur from any master port to any slave port. It is possible for all master ports and slave ports to be in use at the same time as a result of independent master requests. If a slave port is simultaneously requested by more than one master port, arbitration logic will select the higher priority master and grant it ownership of the slave port. All other masters requesting that slave port will be stalled until the higher priority master completes its transactions.

By default, requesting masters will be granted access based on a fixed priority. A round-robin priority mode also is available. In this mode, requesting masters will be treated with equal priority and will be granted access to a slave port in round-robin fashion, based upon the ID of the last master to be granted access. A block diagram of the XBAR is shown in Figure 7-1.

The XBAR can place each slave port in a low-power park mode so that particular slave port will not dissipate any power transitioning address, control or data signals when not being actively accessed by a master port. There is a one-cycle arbitration overhead for exiting low power park mode.

Each slave port can independently support multiple master priority schemes. Each slave port has a hardware input that selects the master priority scheme so the user can dynamically change master priority levels on a slave-port-by-slave-port basis.

7.1.3 Features

- Four (Three for MPC5554) master ports:
  - e200z6 core
  - eDMA
  - EBI
  - FEC (MPC5553 only)
- Five slave ports
  - Flash (refer to Section Chapter 13, “Flash Memory” for information on accessing Flash)
  - EBI
  - Internal SRAM
  - Peripheral bridge A

— Peripheral bridge B
- 32-bit address, 64-bit data paths
- Fully concurrent transfers between independent master and slave ports

7.1.4 Modes of Operation

7.1.4.1 Normal Mode

In normal mode, the XBAR provides the register interface and logic that controls crossbar switch configuration.

7.1.4.2 Debug Mode

The XBAR operation in debug mode is identical to operation in normal mode.

7.2 Memory Map/Register Definition

The memory map for the XBAR program-visible registers is shown in Table 7-2.

Table 7-2. XBAR Register Memory Map

Address	Register Name	Register Description	Size (bits)
Base (0xFFF0_4000)	XBAR_MPR0	Master priority register for slave port 0	32
Base + 0x004–Base + 0x00F	—	Reserved	—
Base + 0x010	XBAR_SGPCR0	General-purpose control register for slave port 0	32
Base + 0x014–Base + 0x0FF	—	Reserved	—
Base + 0x100	XBAR_MPR1	Master priority register for slave port 1	32
Base + 0x104–Base + 0x10F	—	Reserved	—
Base + 0x110	XBAR_SGPCR1	General-purpose control register for slave port 1	32
Base + 0x114–Base + 0x02FF	—	Reserved	—
Base + 0x300	XBAR_MPR3	Master priority register for slave port 3	32
Base + 0x304–Base + 0x30F	—	Reserved	—
Base + 0x310	XBAR_SGPCR3	General-purpose control register for slave port 3	32
Base + 0x314–Base + 0x05FF	—	Reserved	—
Base + 0x600	XBAR_MPR6	Master priority register for slave port 6	32

Table 7-2. XBAR Register Memory Map (continued)

Address	Register Name	Register Description	Size (bits)
Base + 0x604– Base + 0x60F	—	Reserved	—
Base + 0x610	XBAR_SGPCR6	General-purpose control register for slave port 6	32
Base + 0x614– Base + 0x06FF	—	Reserved	—
Base + 0x700	XBAR_MPR7	Master priority register for slave port 7	32
Base + 0x704– Base + 0x70F	—	Reserved	—
Base + 0x710	XBAR_SGPCR7	General-purpose control register for slave port 7	32
Base + 0x714– Base + 0x3_FFFF	—	Reserved	—

7.2.1 Register Descriptions

There are two registers for each slave port of the XBAR. The registers can only be accessed in supervisor mode using 32-bit accesses.

The slave SGPCR also features a bit (RO), which when written with a 1, will prevent all slave registers for that port from being written to again until a reset occurs. The registers will still be readable, but future write attempts will have no effect on the registers and will be terminated with an error response.

7.2.1.1 Master Priority Registers (XBAR_MPR*n*)

The XBAR_MPR for a slave port sets the priority of each master port when operating in fixed priority mode. They are ignored in round-robin priority mode unless more than one master has been assigned high priority by a slave.

NOTE

Masters must be assigned unique priority levels.

The master priority register can only be accessed in supervisor mode with 32-bit accesses. Once the RO (read only) bit has been set in the slave general-purpose control register, the master priority register can only be read. Attempts to write to it will have no effect on the MPR and will result in an error response.

NOTE

XBAR_MPR should be written with a read/modify/write for code compatibility.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset for MPC5554	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset for MPC5553	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x000 (XBAR_MPR0); 0x100 (XBAR_MPR1); 0x300 (XBAR_MPR3); 0x600 (XBAR_MPR6); 0x700 (XBAR_MPR7)															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	MSTR3[1]		0	0	MSTR2		0	0	MSTR1		0	0	MSTR0	
W																
Reset for MPC5554	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0
Reset for MPC5553	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0
Reg Addr	Base + 0x000 (XBAR_MPR0); 0x100 (XBAR_MPR1); 0x300 (XBAR_MPR3); 0x600 (XBAR_MPR6); 0x700 (XBAR_MPR7)															

Figure 7-2. Master Priority Registers (XBAR_MPR*n*)

[1] MSTR3 is supported only in the MPC5553.

Table 7-3. XBAR_MPR*n* Descriptions

Bits	Name	Description
0–17	—	Reserved.
18-19	MSTR3	Master 3 priority. Set the arbitration priority for master port 3 on the associated slave port. 00 This master has the highest priority when accessing the slave port. 01 This master has the 2nd highest priority when accessing the slave port 10 This master has the 3rd highest priority when accessing the slave port.. 11 This master has the lowest priority when accessing the slave port. **Note:** This field is supported only in the MPC5553.
20-21	—	Reserved.
22–23	MSTR2	Master 2 priority. Set the arbitration priority for master port 2 on the associated slave port. 00 This master has the highest priority when accessing the slave port. 01 This master has the 2nd highest priority when accessing the slave port 10 This master has the 3rd highest priority when accessing the slave port.. 11 This master has the lowest priority when accessing the slave port.
24–25	—	Reserved.
26–27	MSTR1	Master 1 priority. Set the arbitration priority for master port 1 on the associated slave port. 00 This master has the highest priority when accessing the slave port. 01 This master has the 2nd highest priority when accessing the slave port 10 This master has the 3rd highest priority when accessing the slave port.. 11 This master has the lowest priority when accessing the slave port.

Table 7-3. XBAR_MPR*n* Descriptions (continued)

Bits	Name	Description
28–29	—	Reserved.
30–31	MSTR0	00 This master has the highest priority when accessing the slave port. 01 This master has the 2nd highest priority when accessing the slave port 10 This master has the 3rd highest priority when accessing the slave port.. 11 This master has the lowest priority when accessing the slave port.

7.2.1.2 Slave General-Purpose Control Registers (XBAR_SGPCR*n*)

The XBAR_SGPCR*n* of a slave port controls several features of the slave port, including the following:

- Round-robin or fixed arbitration policy for a particular slave port
- Write protection of any slave port registers
- Parking algorithm used for a slave port

The PARK field indicates which master port this slave port will park on when no active access attempts are being made to the slave and the parking control field is set to park on a specific master. XBAR_SGPCR*n*[PARK] should only be programmed to select master ports that are actually available on the device, otherwise undefined behavior will result. The low-power park feature can result in an overall power savings if the slave port is not saturated; however, an extra clock of latency will result whenever any master tries to access a slave (not being accessed by another master) because it will not be parked on any master.

The XBAR_SGPCR can only be accessed in supervisor mode with 32-bit accesses. Once the RO (read only) bit has been set in the XBAR_SGPCR, the XBAR_SGPCR and the SBAR_MPR can only be read. Attempts to write to them will have no effect and will result in an error response.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RO[1]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W		—[2]														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x010 (SGPCR0); 0x110 (SGPCR1); 0x310 (SGPCR3); 0x610 (SGPCR6); 0x710 (SGPCR7)															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	ARB		0	0	PCTL		0	PARK		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x010 (SGPCR0); 0x110 (SGPCR1); 0x310 (SGPCR3); 0x610 (SGPCR6); 0x710 (SGPCR7)															

1 Once this bit is set, only hardware reset will clear it

2 Some of these unused bits are writeable and readable, but the bits serve no function. Setting any of these bits has no effect on the operation of this module.

Figure 7-3. Slave General-Purpose Control Registers (XBAR_SGPCR*n*)

Table 7-4. XBAR_SGPCR*n* Field Descriptions

Bits	Name	Description
0	RO	Read only. Used to force all of a slave port's registers to be read only. Once written to 1 it can only be cleared by hardware reset. This bit is cleared by hardware reset. 0 All this slave port's registers can be written. 1 All this slave port's registers are read only and cannot be written (attempted writes have no effect and result in an error response).
1–21	—	Reserved.
22–23	ARB	Arbitration mode. Used to select the arbitration policy for the slave port. This field is initialized by hardware reset. 00 Fixed priority using MPR 01 Round-robin priority 1x Reserved
24–25	—	Reserved.
26–27	PCTL	Parking control. Used to select the parking algorithm used by the slave port. This field is initialized by hardware reset. 00 When no master is making a request, the arbiter will park the slave port on the master port defined by the PARK control field. 01 POL – Park on last. When no master is making a request, the arbiter will park the slave port on the last master to own the slave port. 10 LPP – Low power park. When no master is making a request, the arbiter will park the slave port on no master and will drive all slave port outputs to a safe state. 11 Reserved
28	—	Reserved.
29–31	PARK	Park. Used to determine which master port this slave port parks on when no masters are actively making requests . PCTL must be set to 00. 000 Park on master port 0 001 Park on master port 1 010 Park on master port 2 011 Park on master port 3 (Applies to MPC5553 only) 100 Illegal master port 101 Illegal master port 110 Illegal master port 111 Illegal master port

7.3 Functional Description

This section describes the functionality of the XBAR in more detail.

7.3.1 Overview

The main goal of the XBAR is to increase overall system performance by allowing multiple masters to communicate concurrently with multiple slaves. In order to maximize data throughput it is essential to keep arbitration delays to a minimum.

This section examines data throughput from the point of view of masters and slaves, detailing when the XBAR will stall masters, or insert bubbles on the slave side.

7.3.2 General Operation

When a master makes an access to the XBAR from an idle master state, the access will be taken immediately by the XBAR. If the targeted slave port of the access is available (that is, the requesting master is currently granted ownership of the slave port), the access will be immediately presented on the slave port. It is possible to make single clock (zero wait state) accesses through the XBAR by a granted master. If the targeted slave port of the access is busy or parked on a different master port, the requesting master will simply see wait states inserted until the targeted slave port can service the master's request. The latency in servicing the request will depend on each master's priority level and the responding slave's access time.

Because the XBAR appears to be just another slave to the master device, the master device will have no knowledge of whether or not it actually owns the slave port it is targeting. While the master does not have control of the slave port it is targeting it will simply be wait-stated.

A master will be given control of a targeted slave port only after a previous access to a different slave port has completed, regardless of its priority on the newly targeted slave port. This prevents deadlock from occurring when a master has an outstanding request to slave port A that has a long response time, has a pending access to a different slave port B, and a lower priority master also makes a request to the different slave port B. In this case, the lower priority master will be granted bus ownership of slave port B after a cycle of arbitration, assuming the higher priority master's slave port A access is not terminated.

Once a master has control of the slave port it is targeting, the master will remain in control of that slave port until it either gives up the slave port by running an IDLE cycle, leaves that slave port for its next access, or loses control of the slave port to a higher priority master with a request to the same slave port. However, since all masters run a fixed-length burst transfer to a slave port, it will retain control of the slave port until that transfer sequence is completed. In round-robin arbitration mode, the current master will be forced to hand off bus ownership to an alternately requesting master at the end of its current transfer sequence.

When a slave bus is being idled by the XBAR, it can be parked on the master port indicated by the PARK bits in the XBAR_SGPCR (slave general-purpose control register), or on the last master to have control of the slave port. This can be done in an attempt to save the initial clock of arbitration delay that would otherwise be seen if the master had to arbitrate to gain control of the slave port. The slave port can also be put into low power park mode in attempt to save power.

7.3.3 Master Ports

The XBAR will terminate an access and it will not be allowed to pass through the XBAR unless the master currently is granted access to the slave port to which the access is targeted. A master access will be taken if the slave port to which the access decodes is either currently servicing the master or is parked on the master. In this case the XBAR will be completely transparent and the master's access will be immediately seen on the slave bus and no arbitration delays will be incurred. A master access will be stalled if the access decodes to a slave port that is busy serving another master, parked on another master or is in low-power park mode.

If the slave port is currently parked on another master or is in low power park mode, and no other master is requesting access to the slave port, then only one clock of arbitration will be incurred. If the slave port

is currently serving another master of a lower priority and the master has a higher priority than all other requesting masters, then the master will gain control over the slave port as soon as the data phase of the current access is completed. If the slave port is currently servicing another master of a higher priority, then the master will gain control of the slave port once the other master releases control of the slave port if no other higher priority master is also waiting for the slave port.

A master access will be responded to with an error if the access decodes to a location not occupied by a slave port. This is the only time the XBAR will directly respond with an error response. All other error responses received by the master are the result of error responses on the slave ports being passed through the XBAR.

7.3.4 Slave Ports

The goal of the XBAR with respect to the slave ports is to keep them 100% saturated when masters are actively making requests. In order to do this the XBAR must not insert any bubbles onto the slave bus unless absolutely necessary.

There is only one instance when the XBAR will force a bubble onto the slave bus when a master is actively making a request. This occurs when a handoff of bus ownership occurs and there are no wait states from the slave port. A requesting master which does not own the slave port will be granted access after a one clock delay.

The only other time the XBAR will have control of the slave port is when no masters are making access requests to the slave port and the XBAR is forced to either park the slave port on a specific master, or place the slave port into low power park mode. In these cases, the XBAR will force IDLE for the transfer type.

7.3.5 Priority Assignment

Each master port must be assigned a unique 3 bit priority level in fixed priority mode. If multiple master ports are assigned the same priority level within a register (XBAR_MPR) undefined behavior will result.

7.3.6 Arbitration

XBAR supports two arbitration schemes; a simple fixed-priority comparison algorithm, and a round-robin fairness algorithm. The arbitration scheme is independently programmable for each slave port.

7.3.6.1 Fixed Priority Operation

When operating in fixed-priority arbitration mode, each master is assigned a unique priority level in the XBAR_MPR. If two masters both request access to a slave port, the master with the highest priority in the selected priority register will gain control over the slave port.

Any time a master makes a request to a slave port, the slave port checks to see if the new requesting master's priority level is higher than that of the master that currently has control over the slave port (if any). The slave port does an arbitration check at every clock edge to ensure that the proper master (if any) has control of the slave port.

If the new requesting master's priority level is higher than that of the master that currently has control of the slave port, the higher priority master will be granted control at the termination of any currently pending access, assuming the pending transfer is not part of a burst transfer.

A new requesting master must wait until the end of the fixed-length burst transfer, before it will be granted control of the slave port. But if the new requesting master's priority level is lower than that of the master that currently has control of the slave port, the new requesting master will be forced to wait until the master that currently has control of the slave port is finished accessing the current slave port.

7.3.6.2 Round-Robin Priority Operation

When operating in round-robin mode, each master is assigned a relative priority based on the master number. This relative priority is compared to the ID of the last master to perform a transfer on the slave bus. The highest priority requesting master will become owner of the slave bus at the next transfer boundary (accounting for fixed-length burst transfers). Priority is based on how far ahead the ID of the requesting master is to the ID of the last master. ID is defined by the master port number.

Once granted access to a slave port, a master may perform as many transfers as desired to that port until another master makes a request to the same slave port. The next master in line will be granted access to the slave port when the current transfer is completed, or possibly on the next clock cycle if the current master has no pending access request.

As an example of arbitration in round-robin mode, assume the three masters have ID's 0, 1, and 2. If the last master of the slave port was master 1, and masters 0 and 2 make simultaneous requests, they will be serviced in the order 2 and then 0 assuming no further requests are made.

As another example, if master 1 is waiting on a response from a slow slave and has no further pending access to that slave, no other masters are requesting, and master 0 then makes a request, master 0's request will be granted on the next clock (assuming that master 1's transfer is not a burst transfer), and the request information for master 0 will be driven to the slave as a pending access. If master 2 were to make a request after master 0 has been granted access, but prior to master 0's access being accepted by the slave, master 0 will continue to be granted the slave port, and master 2 will be delayed until the next arbitration boundary, which occurs once the transfer is complete. The round-robin pointer will have been reset to 0, so master 1 could actually be granted the bus next if it has another request which occurs prior to the completion of master 0's transfer. This implies a worst case latency of N transfers for a system with N masters.

Parking may still be used in round-robin mode, but will not affect the round-robin pointer unless the parked master actually performs a transfer. Handoff will occur to the next master in line after one cycle of arbitration.

The slave port does an arbitration check at every clock edge to ensure that the proper master (if any) has control of the slave port.

A new requesting master must wait until the end of the fixed-length burst transfer, before it will be granted control of the slave port. But if the new requesting master's priority level is lower than that of the master that currently has control of the slave port, the new requesting master will be forced to wait until the master that currently has control of the slave port is finished accessing the current slave port.

7.3.6.2.1 Parking

If no master is currently making a request to the slave port then the slave port will be parked. It will park in one of three places, dictated by the PCTL field in the XBAR_SGPCR.

- If the park on specific master mode is selected, then the slave port will park on the master designated by the PARK field. The behavior here is the same as for the POL mode with the exception that a specific master will be parked on instead of the last master to access the slave port. If the master designated by the PARK field tries to access the slave port it will not pay an arbitration penalty, while any other master will pay a one clock penalty.
- If the park on last (POL) mode is selected, then the slave port will park on the last master to access it, passing that master's signals through to the slave bus. When that master accesses the slave port again it will not pay any arbitration penalty; however, if any other master wishes to access the slave port a one clock arbitration penalty will be imposed.
- If the low power park (LPP) mode is selected, then the slave port will enter low power park mode. It will not recognize any master as being in control of it and it will not select any master's signals to pass through to the slave bus. In this case all slave bus activity will effectively halt because all slave bus signals will not be toggling. This can save power if the slave port will not be in use for some time. However, when a master does make a request to the slave port it will be delayed by one clock since it will have to arbitrate to acquire ownership of the slave port.

7.4 Revision History

Substantive Changes since Rev 3.0
Changed MPR's MSTR bit descriptions to include definition for "11" instead of being Reserved.

Chapter 8
Error Correction Status Module (ECSM)

8.1 Introduction

The MPC5553/MPC5554 includes error-correcting code (ECC) implementations to improve the quality and reliability of internal SRAM and internal flash memories. The error correction status module (ECSM), provides a means for the application to collect information on memory errors reported by ECC and/or generic access error information.

8.1.1 Overview

The ECSM provides a set of registers that configure and report ECC errors for the MPC5553/MPC5554 device including accesses to SRAM and Flash memory. The application may configure the device for the types of memory errors to be reported, and then query a set of read-only status and information registers to identify any errors that have been signalled.

There are two types of ECC errors: correctable and non-correctable. A correctable ECC error is generated when only one bit is wrong in a 64-bit double word. In this case it is corrected automatically by hardware, and no flags or other indication is set that the error occurred. A non-correctable ECC error is generated when 2 bits in a 64-bit double word are incorrect. Non-correctable ECC errors cause an interrupt, and if enabled, additional error details are available in the ECSM.

Error correction is implemented on 64 bits of data at a time, using 8 bits for ECC for every 64-bit double word. ECC is checked on reads, and calculated on writes per the following:

1. 64 bits containing the desired byte / half word / word or double word in memory is read, and ECC checked.
2. If the access is a write, then
   - — The new byte / half word / word / double word is merged into the 64 bits.
   - — New ECC bits are calculated.
   - — The 64 bits and the new ECC bits are written back.

In order to use ECC with SRAM, the SRAM memory must be written to before ECC is enabled. See Section 15.5, “Initialization/Application Information.”

8.1.2 Features

The ECSM includes these features:

- Configurable for reporting non-correctable errors
- Registers for capturing ECC information for RAM access errors
- Registers for capturing ECC information for Flash access errors

8.2 Memory Map/Register Definition

This section details the programming model for the ECSM. Table 8-1 is the memory map for the ECSM registers.

Table 8-2. ECSM Memory Map

Address	Register Name	Register Description	Size (bits)
Base (0xFFF4_0000) + 0x016	ECSM_SWTCR	Software watchdog timer control register[1]	16
Base + 0x018– Base + 0x01A	—	Reserved	—
Base + 0x01B	ECSM_SWTSR	Software watchdog timer service register[1]	8
Base + 0x01C– Base + 0x01E	—	Reserved	—
Base + 0x01F	ECSM_SWTIR	Software watchdog timer interrupt register[1]	8
Base + 0x020– Base + 0x023	—	Reserved	—
Base (0xFFF4_0000) + 0x024– Base + 0x027	FBOMCR	FEC Burst Optimization Master Control Register (MPC5553 Only)	32
Base + 0x028– Base + 0x042	—	Reserved	—
Base (0xFFF4_0000) + 0x043	ECSM_ECR	ECC configuration register	8
Base + 0x044– Base + 0x046	—	Reserved	—
Base + 0x047	ECSM_ESR	ECC status register	8
Base + 0x048– Base + 0x049	—	Reserved	—
Base + 0x04A	ECSM_EEGR	ECC error generation register	16
Base + 0x04B– Base + 0x04F	—	Reserved	—
Base + 0x050	ECSM_FEAR	Flash ECC address register	32
Base + 0x054– Base + 0x055	—	Reserved	—
Base + 0x056	ECSM_FEMR	Flash ECC master register	8
Base + 0x057	ECSM_FEAT	Flash ECC attribute register	8
Base + 0x058	ECSM_FEDRH	Flash ECC data high register	32
Base + 0x05C	ECSM_FEDRL	Flash ECC data low register	32
Base + 0x060	ECSM_REAR	RAM ECC address register	32
Base + 0x064– Base + 0x065	—	Reserved	—
Base + 0x066	ECSM_REMR	RAM ECC master register	8
Base + 0x067	ECSM_REAT	RAM ECC attributes register	8
Base + 0x068	ECSM_REDRH	RAM ECC data high register	32
Base + 0x06C	ECSM_REDRL	RAM ECC data low register	32
Base + 0x070– Base + 0x07F	—	Reserved	—

[1] These registers provide control and configuration for a software watchdog timer, and are included as part of a standard Freescale ECSM module incorporated in the MPC5553/MPC5554. The e200z6 core also provides this functionality and is the preferred method for watchdog implementation. See Section 8.2.1.1.

8.2.1 Register Descriptions

Attempted accesses to reserved addresses result in an error termination, while attempted writes to read-only registers are ignored and do not terminate with an error. Unless noted otherwise, writes to the programming model must match the size of the register; for example, an *n*-bit register only supports *n*-bit writes, etc. Attempted writes of a different size than the register width produce an error termination of the bus cycle and no change to the targeted register.

8.2.1.1 Software Watchdog Timer Control, Service, and Interrupt Registers (ECSM_SWTCR, ECSM_SWTSR, and ECSM_SWTIR)

These registers provide control and configuration for a software watchdog timer, and are included as part of a standard Freescale ECSM module incorporated in the MPC5553/MPC5554. The e200z6 core also provides this functionality and is the preferred method for watchdog implementation. In order to optimize code portability to other members of this eSys-based MPU family, use of the watchdog registers in the ECSM is not recommended.

The values in these registers should be left in their reset state. Any change from reset values may cause an unintentional ECSM_SWTIR_SWTIC interrupt.

8.2.1.2 ECC Registers

There are a number of program-visible registers for the sole purpose of reporting and logging of memory failures. These registers include the following:

- ECC configuration register (ECSM_ECR)
- ECC status register (ECSM_ESR)
- Flash ECC address register (ECSM_FEAR)
- Flash ECC master number register (ECSM_FEMR)
- Flash ECC attributes register (ECSM_FEAT)
- Flash ECC data register (ECSM_FEDR)
- RAM ECC address register (ECSM_REAR)
- RAM ECC master number register (ECSM_REMR)
- RAM ECC attributes register (ECSM_REAT)
- RAM ECC data register (ECSM_REDR)

The details on the ECC registers are provided in the subsequent sections.

8.2.1.3 ECC Configuration Register (ECSM_ECR)

The ECSM_ECR is an 8-bit control register for specifying whether memory errors are reported during RAM or Flash accesses. The occurrence of a non-correctable error causes the current access to be terminated with an error condition. In many cases, this error termination is reported directly by the initiating bus master. The ECC reporting logic in the ECSM provides an optional error interrupt mechanism to signal non-correctable memory errors. In addition to the interrupt generation, the ECSM captures specific information (memory address, attributes and data, bus master number, etc.) which may be useful for subsequent failure analysis.

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	ERNCR	EFNCR
W								
Reset	0	0	0	0	0	0	0	0
Reg Addr	Base (0xFFF4_0000) + 0x43							

Figure 8-1. ECC Configuration Register (ECSM_ECR)

Table 8-3. ECSM_ECR Field Definitions

Bits	Name	Description
0–5	—	Reserved.
6	ERNCR	Enable RAM non-correctable reporting. The occurrence of a non-correctable multi-bit RAM error generates a ECSM ECC interrupt request as signalled by the assertion of ECSM_ESR[RNCE]. The faulting address, attributes and data are also captured in the REAR, REMR, REAT and REDR registers. 0 Reporting of non-correctable RAM errors is disabled. 1 Reporting of non-correctable RAM errors is enabled.
7	EFNCR	Enable Flash non-correctable reporting. The occurrence of a non-correctable multi-bit Flash error generates a ECSM ECC interrupt request as signalled by the assertion of ECSM_ESR[FNCE]. The faulting address, attributes and data are also captured in the FEAR, FEMR, FEAT and FEDR registers. 0 Reporting of non-correctable Flash errors is disabled. 1 Reporting of non-correctable Flash errors is enabled.

8.2.1.4 ECC Status Register (ECSM_ESR)

The ECSM_ESR is an 8-bit control register for signaling which types of properly-enabled ECC events have been detected. The ESR signals the last, properly-enabled memory event to be detected. The generation of the ECSM ECC interrupt request is defined by the boolean equation:

```
ECSM_ECC_IRQ

    =  ECSM_ECR[ERNCR] &  ECSM_ESR[RNCE]     // ram,   noncorrectable error

    |  ECSM_ECR[EFNCR] &  ECSM_ESR[FNCE]     // Flash, noncorrectable error
```

where the combination of a properly-enabled category in the ECSM_ECR and the detection of the corresponding condition in the ECSM_ESR produces the interrupt request.

The ECSM allows a maximum of one bit of the ECSM_ESR to be asserted at any given time. This preserves the association between the ECSM_ESR and the corresponding address and attribute registers, which are loaded on each occurrence of an properly-enabled ECC event. If there is a pending ECC interrupt and another properly-enabled ECC event occurs, the ECSM hardware automatically handles the ECSM_ESR reporting, clearing the previous data and loading the new state and thus guaranteeing that only a single flag is asserted.

To maintain the coherent software view of the reported event, the following sequence in the ECSM error interrupt service routine is suggested:

1. Read the ECSM_ESR and save it.
2. Read and save all the address and attribute reporting registers.

3. Re-read the ECSM_ESR and verify the current contents matches the original contents. If the two values are different, go back to step 1 and repeat.
4. When the values are identical, write a 1 to the asserted ECSM_ESR flag to negate the interrupt request.

In the event that multiple status flags are signaled simultaneously, ECSM records the event with the RNCE as highest priority, and then FNCE.

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	RNCE	FNCE
W							w1c	w1c
Reset	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x47							

Figure 8-2. ECC Status Register (ECSM_ESR)

Table 8-4. ECSM_ESR Field Definitions

Bits	Name	Description
0–5	—	Reserved.
6	RNCE	RAM non-correctable error. The occurrence of a properly-enabled non-correctable RAM error generates an ECSM ECC interrupt request. The faulting address, attributes and data are also captured in the REAR, REMR, REAT and REDR registers. To clear this interrupt flag, write a 1 to this bit. Writing a 0 has no effect. 0 No reportable non-correctable RAM error has been detected. 1 A reportable non-correctable RAM error has been detected.
7	FNCE	Flash non-correctable error. The occurrence of a properly-enabled non-correctable Flash error generates an ECSM ECC interrupt request. The faulting address, attributes and data are also captured in the FEAR, FEMR, FEAT and FEDR registers. To clear this interrupt flag, write a 1 to this bit. Writing a 0 has no effect. 0 No reportable non-correctable Flash error has been detected. 1 A reportable non-correctable Flash error has been detected.

8.2.1.5 ECC Error Generation Register (ECSM_EEGR)

The ECSM_EEGR is a 16-bit control register used to force the generation of double-bit data errors in the internal SRAM. This capability provides a mechanism to allow testing of the software service routines associated with memory error logging.The intent is to generate errors during data write cycles, such that subsequent reads of the corrupted address locations generate ECC events, double-bit noncorrectable errors that are terminated with an error response.

If an attempt to force a non-correctable error (by asserting ECSM_EEGR[FRCNCI] or ECSM_EEGR[FRC1NCI]) and ECSM_EEGR[ERRBIT] equals 64, then no data error will be generated.

NOTE

The only allowable values for the 2 control bit enables {FRCNCI, FR1NCI} are {0,0}, {1,0} and {0,1}. The value {1,1} results in undefined behavior.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	FRC NCI	FR1 NCI	0	ERRBIT[0:6]						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x4A															

Figure 8-3. ECC Error Generation (ECSM_EEGR) Register

Table 8-5. ECSM_EEGR Field Definitions

Bits	Name	Description
0–5	—	Reserved.
6	FRCNCI	Force internal SRAM continuous noncorrectable data errors. 0 No internal SRAM continuous 2-bit data errors are generated. 1 2-bit data errors in the internal SRAM are continuously generated. The assertion of this bit forces the internal SRAM controller to create 2-bit data errors, as defined by the bit position specified in ERRBIT[0:6] and the overall odd parity bit, continuously on every write operation. The normal ECC generation takes place in the RAM controller, but then the polarity of the bit position defined by ERRBIT and the overall odd parity bit are inverted to introduce a 2-bit ECC error in the RAM.
7	FR1NCI	Force internal SRAM one noncorrectable data errors. 0 No internal SRAM single 2-bit data errors are generated. 1 One 2-bit data error in the internal SRAM is generated. The assertion of this bit forces the internal SRAM controller to create one 2-bit data error, as defined by the bit position specified in ERRBIT[0:6] and the overall odd parity bit, on the first write operation after this bit is set. The normal ECC generation takes place in the internal SRAM controller, but then the polarity of the bit position defined by ERRBIT and the overall odd parity bit are inverted to introduce a 2-bit ECC error in the RAM. After this bit has been enabled to generate a single 2-bit error, it must be cleared before being set again to properly re-enable the error generation logic.
8	—	Reserved

9–15	ERRBIT	Error bit position. Defines the bit position which is complemented to create the data error on the write operation. The bit specified by this field plus the odd parity bit of the ECC code are inverted.
		The internal SRAM controller follows a vector bit ordering scheme where LSB=0. Errors in the ECC syndrome bits can be generated by setting this field to a value greater than the internal SRAM width. The following association between the ERRBIT field and the corrupted memory bit is defined:
		`if ERRBIT = 0, then RAM[0] is inverted`
		`if ERRBIT = 1, then RAM[1] is inverted`
		`...`
		`if ERRBIT = 63, then RAM[63] is inverted`
		`if ERRBIT = 64, then ECC Parity[0] is inverted`
		`if ERRBIT = 65, then ECC Parity[1] is inverted`
		`...`
		`if ERRBIT = 71, then ECC Parity[7] is inverted`
		For ERRBIT values greater than 71, no bit position is inverted.

8.2.1.6 Flash ECC Address Register (ECSM_FEAR)

The ECSM_FEAR is a 32-bit register for capturing the address of the last, properly-enabled ECC event in the Flash memory. Depending on the state of the ECSM_ECR, an ECC event in the Flash causes the address, attributes and data associated with the access to be loaded into the ECSM_FEAR, ECSM_FEMR, ECSM_FEAT, and ECSM_FEDR registers, and the appropriate flag (F1BC or FNCE) in the ECSM_ESR to be asserted.

The address that is captured in ECSM_FEAR is the Flash page address as seen on the system bus. Refer to Section 13.3.2.7, "Address Register (FLASH_AR)" to retrieve the double word address.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FEAR															
W																
Reset	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U
Reg Addr	Base + 0x50															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FEAR															
W																
Reset	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U
Reg Addr	Base + 0x50															

1 "U" signifies a bit that is uninitialized. Refer to the Preface of the book.

Figure 8-4. Flash ECC Address Register (ECSM_FEAR)

Table 8-6. ECSM_FEAR Field Descriptions

Bits	Name	Description
0–31	FEAR [0:31]	Flash ECC address. Contains the faulting access address of the last, properly-enabled Flash ECC event.

8.2.1.7 Flash ECC Master Number Register (ECSM_FEMR)

The FEMR is an 8-bit register for capturing the XBAR bus master number of the last, properly-enabled ECC event in the Flash memory. Depending on the state of the ECSM_ECR, an ECC event in the Flash causes the address, attributes and data associated with the access to be loaded into the ECSM_FEAR, ECSM_FEMR, ECSM_FEAT and ECSM_FEDR registers, and the appropriate flag (FNCE) in the ECSM_ESR to be asserted.

	0	1	2	3	4	5	6	7
R	0	0	0	0	FEMR			
W								
Reset	0	0	0	0	U	U	U	U
Reg Addr	ECSM Base + 0x56							

1 "U" signifies a bit that is uninitialized. Refer to the Preface of the book.

Figure 8-5. Flash ECC Master Number Register (ECSM_FEMR)

Table 8-7. ECSM_FEMR Field Descriptions

Name	Description	Value
0–3	—	Reserved.
4–7	FEMR [0:3]	Flash ECC master number. Contains the XBAR bus master number of the faulting access of the last, properly-enabled Flash ECC event. The reset value of this field is undefined.

8.2.1.8 Flash ECC Attributes Register (ECSM_FEAT)

The ECSM_FEAT is an 8-bit register for capturing the XBAR bus master attributes of the last, properly-enabled ECC event in the Flash memory. Depending on the state of the ECSM_ECR register, an ECC event in the Flash causes the address, attributes, and data associated with the access to be loaded into the ECSM_FEAR, ECSM_FEMR, ECSM_FEAT, and ECSM_FEDRs, and the appropriate flag (FNCE) in the ECSM_ESR to be asserted.

	0	1	2	3	4	5	6	7
R	WRITE	SIZE			PROT0	PROT1	PROT2	PROT3
W								
Reset	U	U	U	U	U	U	U	U
Reg Addr	Base + 0x57							

1 "U" signifies a bit that is uninitialized. Refer to the Preface of the book.

Figure 8-6. Flash ECC Attributes Register (ECSM_FEAT)

Table 8-8. ECSM_FEAT Field Descriptions

Bits	Name	Description
0	WRITE	Write. The reset value of this field is undefined. 0 System bus read access 1 System bus write access
1–3	SIZE [0:2]	Size. The reset value of this field is undefined. 000 8-bit System bus access 001 16-bit System bus access 010 32-bit System bus access 011 64-bit System bus access 1xx Reserved

Table 8-8. ECSM_FEAT Field Descriptions (continued)

Bits	Name	Description
4	PROT0	Protection: cache. The reset value of this field is undefined. 0 Non-cacheable 1 Cacheable
5	PROT1	Protection: buffer. The reset value of this field is undefined. 0 Non-bufferable 1 Bufferable
6	PROT2	Protection: mode. The reset value of this field is undefined. 0 User mode 1 Supervisor mode
7	PROT3	Protection: type. The reset value of this field is undefined. 0 I-Fetch 1 Data

8.2.1.9 Flash ECC Data High Register (ECSM_FEDRH)

The ECSM_FEDRH and ECSM_FEDRL are 32-bit registers for capturing the data associated with the last, properly-enabled ECC event in the Flash memory. Depending on the state of the ECSM_ECR, an ECC event in the Flash causes the address, attributes and data associated with the access to be loaded into the ECSM_FEAR, ECSM_FEMR, ECSM_FEAT and ECSM_FEDRs, and the appropriate flag (FNCE) in the ECSM_ESR to be asserted.

The data captured on a multi-bit non-correctable ECC error is undefined.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FEDH															
W																
Reset	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U
Reg Addr	Base + 0x58															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FEDH															
W																
Reset	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U
Reg Addr	Base + 0x58															

[1] "U" signifies a bit that is uninitialized. Refer to the Preface of the book.

Figure 8-7. Flash ECC Data High Register (ECSM_FEDRH)

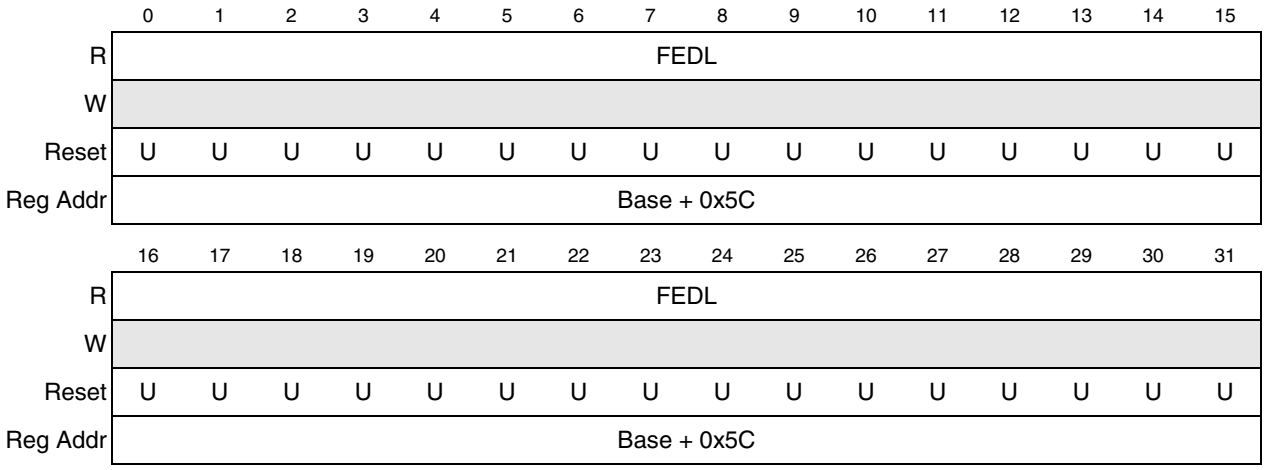
Table 8-9. ECSM_FEDRH Field Descriptions

Bits	Name	Description
0–31	FEDH [0:31]	Flash ECC data. Contains the data associated with the faulting access of the last, properly-enabled Flash ECC event. The register contains the data value taken directly from the data bus. The reset value of this field is undefined.

8.2.1.10 Flash ECC Data Low Registers (ECSM_FEDRL)

The ECSM_FEDRH and ECSM_FEDRL are 32-bit registers for capturing the data associated with the last, properly-enabled ECC event in the Flash memory. Depending on the state of the ECSM_ECR, an ECC event in the Flash causes the address, attributes and data associated with the access to be loaded into the ECSM_FEAR, ECSM_FEMR, ECSM_FEAT and ECSM_FEDRs, and the appropriate flag (FNCE) in the ECSM_ESR to be asserted.

The data captured on a multi-bit non-correctable ECC error is undefined.



1 "U" signifies a bit that is uninitialized. Refer to the Preface of the book.

Figure 8-8. Flash ECC Data Low Register (ECSM_FEDRL)

Table 8-10. ECSM_FEDRL Field Descriptions

Bits	Name	Description
0–31	FEDL [0:31]	Flash ECC data. Contains the data associated with the faulting access of the last, properly-enabled Flash ECC event. The register contains the data value taken directly from the data bus. The reset value of this field is undefined.

8.2.1.11 RAM ECC Address Register (ECSM_REAR)

The ECSM_REAR is a 32-bit register for capturing the address of the last, properly-enabled ECC event in the RAM memory. Depending on the state of the ECSM_ECR, an ECC event in the RAM causes the address, attributes and data associated with the access to be loaded into the ECSM_REAR, ECSM_REMR, ECSM_REAT and ECSM_REDRs, and the appropriate flag (RNCE) in the ECSM_ESR to be asserted.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	REAR															
W																
Reset	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U
Reg Addr	Base + 0x60															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	REAR															
W																
Reset	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U
Reg Addr	Base + 0x60															

1 "U" signifies a bit that is uninitialized. Refer to the Preface of the book.

Figure 8-9. RAM ECC Address Register (ECSM_REAR)

Table 8-11. ECSM_REAR Field Descriptions

Bits	Name	Description
0–31	REAR [0:31]	RAM ECC address. Contains the faulting access address of the last, properly-enabled RAM ECC event. The reset value of this field is undefined.

8.2.1.12 RAM ECC Master Number Register (ECSM_REMR)

The REMR is an 8-bit register for capturing the XBAR bus master number of the last, properly-enabled ECC event in the RAM memory. Depending on the state of the ECSM_ECR, an ECC event in the RAM causes the address, attributes and data associated with the access to be loaded into the ECSM_REAR, ECSM_REMR, ECSM_REAT and ECSM_REDRs, and the appropriate flag (RNCE) in the ECSM_ESR to be asserted.

	0	1	2	3	4	5	6	7
R	0	0	0	0	REMR			
W								
Reset	0	0	0	0	U	U	U	U
Reg Addr	Base + 0x66							

1 "U" signifies a bit that is uninitialized. Refer to the Preface of the book.

Figure 8-10. RAM ECC Master Number Register (ECSM_REMR)

Table 8-12. ECSM_REMR Field Descriptions

Bits	Name	Description
0–3	—	Reserved.
4-7	REMR [0:3]	RAM ECC master number. Contains the XBAR bus master number of the faulting access of the last, properly-enabled RAM ECC event. The reset value of this field is undefined.

8.2.1.13 RAM ECC Attributes Register (ECSM_REAT)

The ECSM_REAT is an 8-bit register for capturing the XBAR bus master attributes of the last, properly-enabled ECC event in the RAM memory. Depending on the state of the ECSM_ECR, an ECC event in the RAM causes the address, attributes and data associated with the access to be loaded into the ECSM_REAR, ECSM_REMR, ECSM_REAT and ECSM_REDRs, and the appropriate flag (RNCE) in the ECSM_ESR to be asserted.

	0	1	2	3	4	5	6	7
R	WRITE	SIZE			PROT0	PROT1	PROT2	PROT3
W								
Reset	U	U	U	U	U	U	U	U
Reg Addr	Base + 0x67							

[1] "U" signifies a bit that is uninitialized. Refer to the Preface of the book.

Figure 8-11. RAM ECC Attributes Register (ECSM_REAT)

Table 8-13. ECSM_REAT Field Descriptions

Bits	Name	Description
0	WRITE	Write. The reset value of this field is undefined. 0 System bus read access 1 System bus write access
1–3	SIZE [0:2]	Size. The reset value of this field is undefined. 000 8-bit system bus access 001 16-bit system bus access 010 32-bit system bus access 011 64-bit system bus access 1xx Reserved
4	PROT0	Protection: cache. The reset value of this field is undefined. 0 Non-cacheable 1 Cacheable
5	PROT1	Protection: buffer. The reset value of this field is undefined. 0 Non-bufferable 1 Bufferable
6	PROT2	Protection: mode. The reset value of this field is undefined. 0 User mode 1 Supervisor mode
7	PROT3	Protection: type. The reset value of this field is undefined. 0 I-Fetch 1 Data

8.2.1.14 RAM ECC Data High Register (ECSM_REDRH)

The ECSM_REDRH and ECSM_REDRL are 32-bit registers for capturing the data associated with the last, properly-enabled ECC event in the RAM memory. Depending on the state of the ECSM_ECR, an ECC event in the RAM causes the address, attributes and data associated with the access to be loaded into

the ECSM_REAR, ECSM_REMR, ECSM_REAT, and ECSM_REDRH and ECSM_REDRL, and the appropriate flag (RFNCE) in the ECSM_ESR to be asserted.

The data captured on a multi-bit non-correctable ECC error is undefined.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	REDH															
W																
Reset	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U
Reg Addr	Base + 0x68															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	REDH															
W																
Reset	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U
Reg Addr	Base + 0x68															

1 “U” signifies a bit that is uninitialized. Refer to the Preface of the book.

Figure 8-12. RAM ECC Data High Register (ECSM_REDRH)

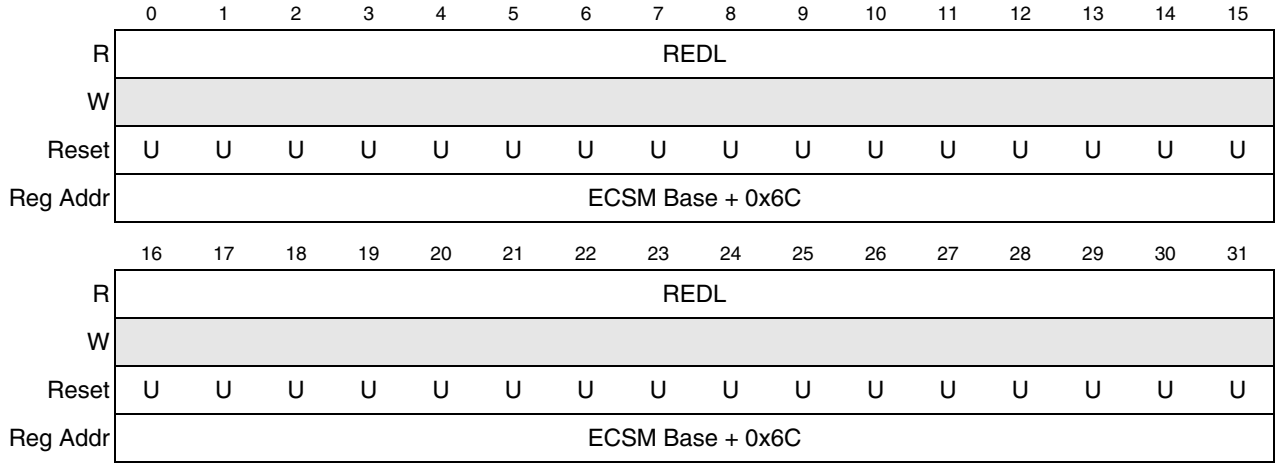
Table 8-14. ECSM_REDRH Field Descriptions

Bits	Name	Description
0–31	REDH [0:31]	RAM ECC data. Contains the data associated with the faulting access of the last, properly-enabled RAM ECC event. The register contains the data value taken directly from the data bus. The reset value of this field is undefined.

8.2.1.15 RAM ECC Data Low Registers (ECSM_REDRL)

The ECSM_REDRH and ECSM_REDRL are 32-bit registers for capturing the data associated with the last, properly-enabled ECC event in the RAM memory. Depending on the state of the ECSM_ECR, an ECC event in the RAM causes the address, attributes and data associated with the access to be loaded into the ECSM_REAR, ECSM_REMR, ECSM_REAT, ECSM_REDRH, and ECSM_REDRL, and the appropriate flag (RFNCE) in the ECSM_ESR to be asserted.

The data captured on a multi-bit non-correctable ECC error is undefined.



1 "U" signifies a bit that is uninitialized. Refer to the Preface of the book.

Figure 8-13. RAM ECC Data Low Register (ECSM_REDRL)

Table 8-15. ECSM_REDRL Field Descriptions

Bits	Name	Description
0–31	REDL [0:31]	RAM ECC data. Contains the data associated with the faulting access of the last, properly-enabled RAM ECC event. The register contains the data value taken directly from the data bus. The reset value of this field is undefined.

8.3 Initialization/Application Information

In order to use the ECC mechanism for internal SRAM accesses, it is essential for the ECC check bits to be initialized after power on. See Section 15.5, "Initialization/Application Information."

All non-correctable ECC errors cause a data storage interrupt (IVOR2) regardless of whether non-correctable reporting is enabled. A data storage interrupt handler can determine:

— The destination asserted an error, the ESR[XTE] bit will be set.

— The address where the error occurred, using the data exception address register (DEAR).

However, details of the ECC error are not reported unless non-correctable reporting is enabled by setting bits ERNCR and EFNCR in the ECSM_ECR. When these bits are set and a non-correctable ECC error occurs, error information is recorded in other ECSM registers and an interrupt request is generated on vector 9 of the INTC. If properly enabled, this INTC vector 9 can cause an external interrupt (IVOR4) along with the data storage interrupt (IVOR2).

To avoid the external interrupt (IVOR4) being generated, the application enables non-correctable reporting in the ECSM, but does not enable that its interrupt be recognized. The INTC_PSR[PRI] value for the ECC error interrupt request is left at its reset value of 0. The 0 priority level is the lowest priority and is never recognized, resulting in only the data storage interrupt (IVOR2) being taken.

8.4 Revision History

Substantive Changes since Rev 3.0
Added Section 8.2.1.5, “ECC Error Generation Register (ECSM_EEGR) as well as updated memory map to show EEGR.

Chapter 9
Enhanced Direct Memory Access (eDMA)

9.1 Introduction

This chapter describes the MPC5553/MPC5554's enhanced direct memory access (eDMA) controller, a second-generation module capable of performing complex data transfers with minimal intervention from a host processor.

9.1.1 Block Diagram

Figure 9-1 is a block diagram of the eDMA module.

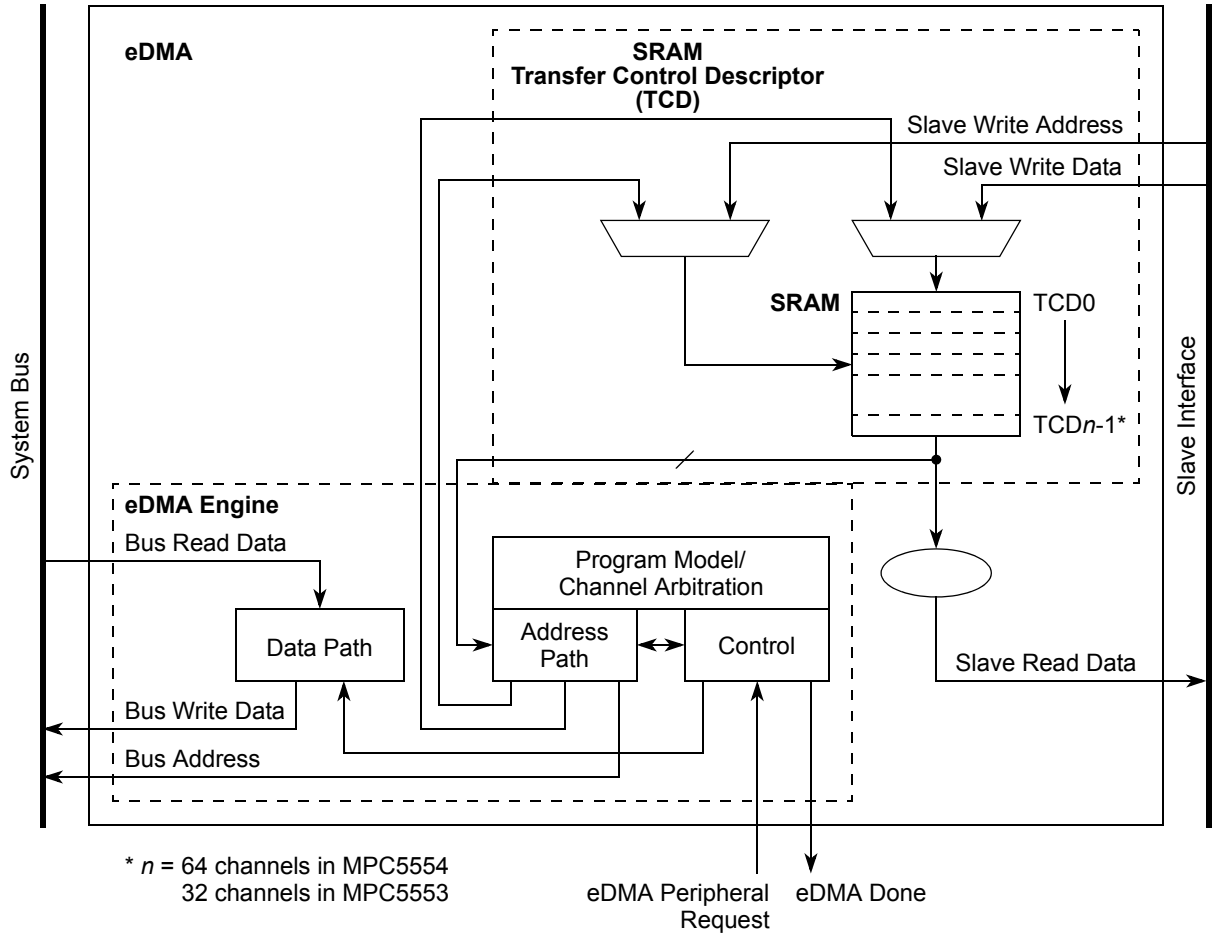


Figure 9-1. eDMA Block Diagram

9.1.2 Overview

The enhanced direct memory access (eDMA) controller hardware microarchitecture includes a DMA engine which performs source and destination address calculations, and the actual data movement operations, along with SRAM-based local memory containing the transfer control descriptors (TCD) for the channels.

9.1.3 Features

The eDMA is a highly-programmable data transfer engine, which has been optimized to minimize the required intervention from the host processor. It is intended for use in applications where the data size to be transferred is statically known, and is *not* defined within the data packet itself. The eDMA module features:

- All data movement via dual-address transfers: read from source, write to destination
  - Programmable source, destination addresses, transfer size, plus support for enhanced addressing modes

- 64-channel (MPC5554) or 32-channel (MPC5553) implementation performs complex data transfers with minimal intervention from a host processor
  - 32 bytes of data registers, used as temporary storage to support burst transfers (refer to SSIZE bit)
  - Connections to the crossbar switch for bus mastering the data movement
- Transfer control descriptor (TCD) organized to support two-deep, nested transfer operations
  - 32-byte TCD per channel stored in local memory
  - An inner data transfer loop defined by a minor byte transfer count
  - An outer data transfer loop defined by a major iteration count
- Channel activation via one of three methods:
  - Explicit software initiation
  - Initiation via a channel-to-channel linking mechanism for continual transfers
  - Peripheral-paced hardware requests (one per channel)

NOTE

For all three methods, one activation per execution of the minor loop is required

- Support for fixed-priority and round-robin channel arbitration
- Channel completion reported via optional interrupt requests
  - One interrupt per channel, optionally asserted at completion of major iteration count
  - Error terminations are enabled per channel, and logically summed together to form two optional error interrupts (MPC5554) or a single error interrupt (MPC5553).
- Support for scatter/gather DMA processing
- Any channel can be programmed so that it can be suspended by a higher priority channel's activation, before completion of a minor loop.

Throughout this chapter, *n* is used to reference the channel number. Additionally, data sizes are defined as byte (8-bit), half-word (16-bit), word (32-bit) and double-word (64-bit).

9.1.4 Modes of Operation

9.1.4.1 Normal Mode

In normal mode, the eDMA is used to transfer data between a source and a destination. The source and destination can be a memory block or an I/O block capable of operation with the eDMA.

9.1.4.2 Debug Mode

If enabled by EDMA_CR[EDBG] and the CPU enters debug mode, the eDMA will not honor any service requests when the debug input signal is asserted. If the signal is asserted during transfer of a block of data described by a minor loop in the current active channel's TCD, the eDMA will continue operation until completion of the minor loop.

9.2 External Signal Description

The eDMA has no external signals.

9.3 Memory Map/Register Definition

The eDMA's programming model is partitioned into two regions: the first region defines a number of registers providing control functions, while the second region corresponds to the local transfer control descriptor memory.

Some registers are implemented as two 32-bit registers, and include an "H" and "L" suffix, signaling the "high" and "low" portions of the control function. Table 9-1 is a 32-bit view of the eDMA's memory map.

Table 9-1. eDMA 32-bit Memory Map

Address	Register Name	Register Description	Size (bits)
Base (0xFFF4_4000)	EDMA_CR	eDMA control register	32
Base + 0x0004	EDMA_ESR	eDMA error status register	32
Base + 0x0008	EDMA_ERQRH	eDMA enable request high register (MPC5554 only)	32
Base + 0x000C	EDMA_ERQRL	eDMA enable request low register	32
Base + 0x0010	EDMA_EEIRH	eDMA enable error interrupt high register (MPC5554 only)	32
Base + 0x0014	EDMA_EEIRL	eDMA enable error interrupt low register	32
Base + 0x0018	EDMA_SERQR	eDMA set enable request register	8
Base + 0x0019	EDMA_CERQR	eDMA clear enable request register	8
Base + 0x001A	EDMA_SEEIR	eDMA set enable error interrupt register	8
Base + 0x001B	EDMA_CEEIR	eDMA clear enable error interrupt register	8
Base + 0x001C	EDMA_CIRQR	eDMA clear interrupt request register	8
Base + 0x001D	EDMA_CER	eDMA clear error register	8
Base + 0x001E	EDMA_SSBR	eDMA set start bit register	8
Base + 0x001F	EDMA_CDSBR	eDMA clear done status bit register	8
Base + 0x0020	EDMA_IRQRH	eDMA interrupt request high register (MPC5554 only)	32
Base + 0x0024	EDMA_IRQRL	eDMA interrupt request low register	32
Base + 0x0028	EDMA_ERH	eDMA error high register (MPC5554 only)	32
Base + 0x002C	EDMA_ERL	eDMA error low register	32
Base + 0x0030– Base + 0x00FF	—	Reserved	—
Base + 0x0100	EDMA_CPR0	eDMA channel 0 priority register	8
Base + 0x0101	EDMA_CPR1	eDMA channel 1 priority register	8
Base + 0x0102	EDMA_CPR2	eDMA channel 2 priority register	8
Base + 0x0103	EDMA_CPR3	eDMA channel 3 priority register	8
Base + 0x0104	EDMA_CPR4	eDMA channel 4 priority register	8
Base + 0x0105	EDMA_CPR5	eDMA channel 5 priority register	8
Base + 0x0106	EDMA_CPR6	eDMA channel 6 priority register	8
Base + 0x0107	EDMA_CPR7	eDMA channel 7 priority register	8

Table 9-1. eDMA 32-bit Memory Map (continued)

Address	Register Name	Register Description	Size (bits)
Base + 0x0108	EDMA_CPR8	eDMA channel 8 priority register	8
Base + 0x0109	EDMA_CPR9	eDMA channel 9 priority register	8
Base + 0x010A	EDMA_CPR10	eDMA channel 10 priority register	8
Base + 0x010B	EDMA_CPR11	eDMA channel 11 priority register	8
Base + 0x010C	EDMA_CPR12	eDMA channel 12 priority register	8
Base + 0x010D	EDMA_CPR13	eDMA channel 13 priority register	8
Base + 0x010E	EDMA_CPR14	eDMA channel 14 priority register	8
Base + 0x010F	EDMA_CPR15	eDMA channel 15 priority register	8
Base + 0x0110	EDMA_CPR16	eDMA channel 16 priority register	8
Base + 0x0111	EDMA_CPR17	eDMA channel 17 priority register	8
Base + 0x0112	EDMA_CPR18	eDMA channel 18 priority register	8
Base + 0x0113	EDMA_CPR19	eDMA channel 19 priority register	8
Base + 0x0114	EDMA_CPR20	eDMA channel 20 priority register	8
Base + 0x0115	EDMA_CPR21	eDMA channel 21 priority register	8
Base + 0x0116	EDMA_CPR22	eDMA channel 22 priority register	8
Base + 0x0117	EDMA_CPR23	eDMA channel 23 priority register	8
Base + 0x0118	EDMA_CPR24	eDMA channel 24 priority register	8
Base + 0x0119	EDMA_CPR25	eDMA channel 25 priority register	8
Base + 0x011A	EDMA_CPR26	eDMA channel 26 priority register	8
Base + 0x011B	EDMA_CPR27	eDMA channel 27 priority register	8
Base + 0x011C	EDMA_CPR28	eDMA channel 28 priority register	8
Base + 0x011D	EDMA_CPR29	eDMA channel 29 priority register	8
Base + 0x011E	EDMA_CPR30	eDMA channel 30 priority register	8
Base + 0x011F	EDMA_CPR31	eDMA channel 31 priority register	8
NOTE: Channels 32-63 Are Available only in the MPC5554			
Base + 0x0120	EDMA_CPR32	eDMA channel 32 priority register	8
Base + 0x0121	EDMA_CPR33	eDMA channel 33 priority register	8
Base + 0x0122	EDMA_CPR34	eDMA channel 34 priority register	8
Base + 0x0123	EDMA_CPR35	eDMA channel 35 priority register	8
Base + 0x0124	EDMA_CPR36	eDMA channel 36 priority register	8
Base + 0x0125	EDMA_CPR37	eDMA channel 37 priority register	8
Base + 0x0126	EDMA_CPR38	eDMA channel 38 priority register	8
Base + 0x0127	EDMA_CPR39	eDMA channel 39 priority register	8

Table 9-1. eDMA 32-bit Memory Map (continued)

Address	Register Name	Register Description	Size (bits)
Base + 0x0128	EDMA_CPR40	eDMA channel 40 priority register	8
Base + 0x0129	EDMA_CPR41	eDMA channel 41 priority register	8
Base + 0x012A	EDMA_CPR42	eDMA channel 42 priority register	8
Base + 0x012B	EDMA_CPR43	eDMA channel 43 priority register	8
Base + 0x012C	EDMA_CPR44	eDMA channel 44 priority register	8
Base + 0x012D	EDMA_CPR45	eDMA channel 45 priority register	8
Base + 0x012E	EDMA_CPR46	eDMA channel 46 priority register	8
Base + 0x012F	EDMA_CPR47	eDMA channel 47 priority register	8
Base + 0x0130	EDMA_CPR48	eDMA channel 48 priority register	8
Base + 0x0131	EDMA_CPR49	eDMA channel 49 priority register	8
Base + 0x0132	EDMA_CPR50	eDMA channel 50 priority register	8
Base + 0x0133	EDMA_CPR51	eDMA channel 51 priority register	8
Base + 0x0134	EDMA_CPR52	eDMA channel 52 priority register	8
Base + 0x0135	EDMA_CPR53	eDMA channel 53 priority register	8
Base + 0x0136	EDMA_CPR54	eDMA channel 54 priority register	8
Base + 0x0137	EDMA_CPR55	eDMA channel 55 priority register	8
Base + 0x0138	EDMA_CPR56	eDMA channel 56 priority register	8
Base + 0x0139	EDMA_CPR57	eDMA channel 57 priority register	8
Base + 0x013A	EDMA_CPR58	eDMA channel 58 priority register	8
Base + 0x013B	EDMA_CPR59	eDMA channel 59 priority register	8
Base + 0x013C	EDMA_CPR60	eDMA channel 60 priority register	8
Base + 0x013D	EDMA_CPR61	eDMA channel 61 priority register	8
Base + 0x013E	EDMA_CPR62	eDMA channel 62 priority register	8
Base + 0x013F	EDMA_CPR63	eDMA channel 63 priority register	8
Base + 0x0140– Base + 0x0FFF	—	Reserved	—
Base + 0x1000	TCD00	eDMA transfer control descriptor 00	256
Base + 0x1020	TCD01	eDMA transfer control descriptor 01	256
Base + 0x1040	TCD02	eDMA transfer control descriptor 02	256
Base + 0x1060	TCD03	eDMA transfer control descriptor 03	256
Base + 0x1080	TCD04	eDMA transfer control descriptor 04	256
Base + 0x10A0	TCD05	eDMA transfer control descriptor 05	256
Base + 0x10C0	TCD06	eDMA transfer control descriptor 06	256

Table 9-1. eDMA 32-bit Memory Map (continued)

Address	Register Name	Register Description	Size (bits)
Base + 0x10E0	TCD07	eDMA transfer control descriptor 07	256
Base + 0x1100	TCD08	eDMA transfer control descriptor 08	256
Base + 0x1120	TCD09	eDMA transfer control descriptor 09	256
Base + 0x1140	TCD10	eDMA transfer control descriptor 10	256
Base + 0x1160	TCD11	eDMA transfer control descriptor 11	256
Base + 0x1180	TCD12	eDMA transfer control descriptor 12	256
Base + 0x11A0	TCD13	eDMA transfer control descriptor 13	256
Base + 0x11C0	TCD14	eDMA transfer control descriptor 14	256
Base + 0x11E0	TCD15	eDMA transfer control descriptor 15	256
Base + 0x1200	TCD16	eDMA transfer control descriptor 16	256
Base + 0x1220	TCD17	eDMA transfer control descriptor 17	256
Base + 0x1240	TCD18	eDMA transfer control descriptor 18	256
Base + 0x1260	TCD19	eDMA transfer control descriptor 19	256
Base + 0x1280	TCD20	eDMA transfer control descriptor 20	256
Base + 0x12A0	TCD21	eDMA transfer control descriptor 21	256
Base + 0x12C0	TCD22	eDMA transfer control descriptor 22	256
Base + 0x12E0	TCD23	eDMA transfer control descriptor 23	256
Base + 0x1300	TCD24	eDMA transfer control descriptor 24	256
Base + 0x1320	TCD25	eDMA transfer control descriptor 25	256
Base + 0x1340	TCD26	eDMA transfer control descriptor 26	256
Base + 0x1360	TCD27	eDMA transfer control descriptor 27	256
Base + 0x1380	TCD28	eDMA transfer control descriptor 28	256
Base + 0x13A0	TCD29	eDMA transfer control descriptor 29	256
Base + 0x13C0	TCD30	eDMA transfer control descriptor 30	256
Base + 0x13E0	TCD31	eDMA transfer control descriptor 31	256
NOTE: Transfer Control Descriptors 32-63 Are Available only in the MPC5554			
Base + 0x1400	TCD32	eDMA transfer control descriptor 32	256
Base + 0x1420	TCD33	eDMA transfer control descriptor 33	256
Base + 0x1440	TCD34	eDMA transfer control descriptor 34	256
Base + 0x1460	TCD35	eDMA transfer control descriptor 35	256
Base + 0x1480	TCD36	eDMA transfer control descriptor 36	256
Base + 0x14A0	TCD37	eDMA transfer control descriptor 37	256
Base + 0x14C0	TCD38	eDMA transfer control descriptor 38	256

Table 9-1. eDMA 32-bit Memory Map (continued)

Address	Register Name	Register Description	Size (bits)
Base + 0x14E0	TCD39	eDMA transfer control descriptor 39	256
Base + 0x1500	TCD43	eDMA transfer control descriptor 40	256
Base + 0x1520	TCD41	eDMA transfer control descriptor 41	256
Base + 0x1540	TCD42	eDMA transfer control descriptor 42	256
Base + 0x1560	TCD43	eDMA transfer control descriptor 43	256
Base + 0x1580	TCD44	eDMA transfer control descriptor 44	256
Base + 0x15A0	TCD45	eDMA transfer control descriptor 45	256
Base + 0x15C0	TCD46	eDMA transfer control descriptor 46	256
Base + 0x15E0	TCD47	eDMA transfer control descriptor 47	256
Base + 0x1600	TCD48	eDMA transfer control descriptor 48	256
Base + 0x1620	TCD49	eDMA transfer control descriptor 49	256
Base + 0x1640	TCD50	eDMA transfer control descriptor 50	256
Base + 0x1660	TCD51	eDMA transfer control descriptor 51	256
Base + 0x1680	TCD52	eDMA transfer control descriptor 52	256
Base + 0x16A0	TCD53	eDMA transfer control descriptor 53	256
Base + 0x16C0	TCD54	eDMA transfer control descriptor 54	256
Base + 0x16E0	TCD55	eDMA transfer control descriptor 55	256
Base + 0x1700	TCD56	eDMA transfer control descriptor 56	256
Base + 0x1720	TCD57	eDMA transfer control descriptor 57	256
Base + 0x1740	TCD58	eDMA transfer control descriptor 58	256
Base + 0x1760	TCD59	eDMA transfer control descriptor 59	256
Base + 0x1780	TCD60	eDMA transfer control descriptor 60	256
Base + 0x17A0	TCD61	eDMA transfer control descriptor 61	256
Base + 0x17C0	TCD62	eDMA transfer control descriptor 62	256
Base + 0x17E0	TCD63	eDMA transfer control descriptor 63	256

9.3.1 Register Descriptions

Reading reserved bits in a register will return the value of zero. Writes to reserved bits in a register will be ignored. Reading or writing to a reserved memory location will generate a bus error.

Many of the control registers have a bit width that matches the number of channels implemented in the module, or 64-bits in size. These registers are implemented as two 32-bit registers, and include an “H” and “L” suffixes, signaling the “high” and “low” portions of the control function. Note that for the MPC5553, only the Low register is implemented for its 32 channels. High (H) registers are reserved on the MPC5553 and accessing them will generate a bus error.

9.3.1.1 eDMA Control Register (EDMA_CR)

The 32-bit EDMA_CR defines the basic operating configuration of the eDMA.

For the MPC5554 the eDMA arbitrates channel service requests in four groups (0, 1, 2, 3) of 16 channels each; the MPC5553 arbitrates channel service requests in two groups (0, 1). For the MPC5553/MPC5554, group 0 contains channels 0-15 and group 1 contains channels 16-31; but for the MPC5554 only, group 2 contains channels 32-47, and group 3 contains channels 48-63.

Arbitration within a group can be configured to use either a fixed priority or a round robin. In fixed priority arbitration, the highest priority channel requesting service is selected to execute. The priorities are assigned by the channel priority registers (see Section 9.3.1.15). In round robin arbitration mode, the channel priorities are ignored and the channels within each group are cycled through, from channel 15 down to channel 0, without regard to priority.

The group priorities operate in a similar fashion. In group fixed priority arbitration mode, channel service requests in the highest priority group are executed first where priority level 3 (in the MPC5554; priority level 1 for the MPC5553) is the highest and priority level 0 is the lowest. The group priorities are assigned in the GRP*n*PRI fields of the eDMA control register (EDMA_CR). All group priorities must have unique values prior to any channel service requests occur, otherwise a configuration error will be reported. In group round robin mode, the group priorities are ignored and the groups are cycled through, from group 3 down to group 0, without regard to priority.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0000															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	GRP3PRI[1]		GRP2PRI[1]		GRP1PRI[2]		GRP0PRI[3]		0	0	0	0	ERGA	ERCA	EDBG	0
W																
Reset	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0000															

Figure 9-2. eDMA Control Register (EDMA_CR)

1 Available only in the MPC5554.

2 In the MPC5553, only bit 21 is used

3 In the MPC5553, only bit 23 is used

Table 9-2. EDMA_CR Field Descriptions

Bits	Name	Description
0–15	—	Reserved.
16–17	GRP3PRI	Channel group 3 priority. Group 3 priority level when fixed priority group arbitration is enabled. **Note:** Available only in the MPC5554

Table 9-2. EDMA_CR Field Descriptions (continued)

Bits	Name	Description
18–19	GRP2PRI	Channel group 2 priority. Group 2 priority level when fixed priority group arbitration is enabled. **Note:** Available only in the MPC5554
20–21	GRP1PRI	Channel group 1 priority. Group 1 priority level when fixed priority group arbitration is enabled. **Note:** In the MPC5553, only bit 21 is used
22–23	GRP0PRI	Channel group 0 priority. Group 0 priority level when fixed priority group arbitration is enabled. **Note:** In the MPC5553, only bit 23 is used
24–27	—	Reserved.
28	ERGA	Enable round robin group arbitration. 0 Fixed priority arbitration is used for selection among the groups. 1 Round robin arbitration is used for selection among the groups.
29	ERCA	Enable round robin channel arbitration. 0 Fixed priority arbitration is used for channel selection within each group. 1 Round robin arbitration is used for channel selection within each group.
30	EDBG	Enable debug. 0 The assertion of the system debug control input is ignored. 1 The assertion of the system debug control input causes the eDMA to stall the start of a new channel. Executing channels are allowed to complete. Channel execution will resume when either the system debug control input is negated or the EDBG bit is cleared.
31	—	Reserved.

9.3.1.2 eDMA Error Status Register (EDMA_ESR)

The EDMA_ESR provides information concerning the last recorded channel error. Channel errors can be caused by a configuration error (an illegal setting in the transfer control descriptor or an illegal priority register setting in fixed arbitration mode) or an error termination to a bus master read or write cycle.

A configuration error is caused when the starting source or destination address, source or destination offsets, minor loop byte count, and the transfer size represent an inconsistent state. The addresses and offsets must be aligned on 0-modulo-transfer_size boundaries, and the minor loop byte count must be a multiple of the source and destination transfer sizes. All source reads and destination writes must be configured to the natural boundary of the programmed transfer size respectively.

In fixed arbitration mode, a configuration error is caused by any two channel priorities being equal within a group, or any group priority levels being equal among the groups. For either type of priority configuration error, the ERRCHN field is undefined. All channel priority levels within a group must be unique and all group priority levels among the groups must be unique when fixed arbitration mode is enabled.

If a scatter/gather operation is enabled upon channel completion, a configuration error is reported if the scatter/gather address (DLAST_SGA) is not aligned on a 32-byte boundary. If minor loop channel linking is enabled upon channel completion, a configuration error is reported when the link is attempted if the TCD.CITER.E_LINK bit does not equal the TCD.BITER.E_LINK bit. All configuration error conditions except scatter/gather and minor loop link error are reported as the channel is activated and assert an error interrupt request if enabled. When properly enabled, a scatter/gather configuration error is reported when

the scatter/gather operation begins at major loop completion. A minor loop channel link configuration error is reported when the link operation is serviced at minor loop completion.

If a system bus read or write is terminated with an error, the data transfer is immediately stopped and the appropriate bus error flag is set. In this case, the state of the channel's transfer control descriptor is updated by the eDMA engine with the current source address, destination address, and minor loop byte count at the point of the fault. If a bus error occurs on the last read prior to beginning the write sequence, the write will execute using the data captured during the bus error. If a bus error occurs on the last write prior to switching to the next read sequence, the read sequence will execute before the channel is terminated due to the destination bus error.

The occurrence of any type of error causes the eDMA engine to stop the active channel, and the appropriate channel bit in the eDMA error register to be asserted. At the same time, the details of the error condition are loaded into the EDMA_ESR. The major loop complete indicators, setting the transfer control descriptor DONE flag and the possible assertion of an interrupt request, are *not* affected when an error is detected. Once the error status has been updated, the eDMA engine continues to operate by servicing the next appropriate channel. A channel that experiences an error condition is not automatically disabled. If a channel is terminated by an error and then issues another service request before the error is fixed, that channel will execute and terminate with the same error condition.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0004															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	GPE	CPE	ERRCHN						SAE	SOE	DAE	DOE	NCE	SGE	SBE	DBE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0004															

Figure 9-3. eDMA Error Status Register (EDMA_ESR)

Table 9-3. EDMA_ESR Field Descriptions

Bits	Name	Description
0	VLD	Logical OR of all EDMA_ERH and EDMA_ERL status bits. 0 No EDMA_ER bits are set. 1 At least one EDMA_ER bit is set indicating a valid error exists that has not been cleared.
1–15	—	Reserved.
16	GPE	Group priority error. 0 No group priority error. 1 The last recorded error was a configuration error among the group priorities indicating not all group priorities are unique.
17	CPE	Channel priority error. 0 No channel priority error. 1 The last recorded error was a configuration error in the channel priorities within a group, indicating not all channel priorities within a group are unique.

Table 9-3. EDMA_ESR Field Descriptions (continued)

Bits	Name	Description
18–23	ERRCHN [0:5]	Error channel number — this is the channel number of the last recorded error (excluding GPE and CPE errors) **Note:** Do not rely on the number in the ERRCHN field for group and channel priority errors. Group and channel priority errors need to be resolved by inspection. The application code must interrogate the priority registers to find groups or channels with duplicate priority level.
24	SAE	Source address error. 0 No source address configuration error. 1 The last recorded error was a configuration error detected in the TCD.SADDR field, indicating TCD.SADDR is inconsistent with TCD.SSIZE.
25	SOE	Source offset error. 0 No source offset configuration error. 1 The last recorded error was a configuration error detected in the TCD.SOFF field, indicating TCD.SOFF is inconsistent with TCD.SSIZE.
26	DAE	Destination address error. 0 No destination address configuration error. 1 The last recorded error was a configuration error detected in the TCD.DADDR field, indicating TCD.DADDR is inconsistent with TCD.DSIZE.
27	DOE	Destination offset error. 0 No destination offset configuration error. 1 The last recorded error was a configuration error detected in the TCD.DOFF field, indicating TCD.DOFF is inconsistent with TCD.DSIZE.
28	NCE	NBYTES/CITER configuration error. 0 No NBYTES/CITER configuration error. 1 The last recorded error was a configuration error detected in the TCD.NBYTES or TCD.CITER fields, indicating the following conditions exist: • TCD.NBYTES is not a multiple of TCD.SSIZE and TCD.DSIZE, or • TCD.CITER is equal to zero, or • TCD.CITER.E_LINK is not equal to TCD.BITER.E_LINK.
29	SGE	Scatter/gather configuration error. 0 No scatter/gather configuration error. 1 The last recorded error was a configuration error detected in the TCD.DLAST_SGA field, indicating TCD.DLAST_SGA is not on a 32-byte boundary. This field is checked at the beginning of a scatter/gather operation after major loop completion if TCD.E_SG is enabled.
30	SBE	Source bus error. 0 No source bus error. 1 The last recorded error was a bus error on a source read.
31	DBE	Destination bus error. 0 No destination bus error. 1 The last recorded error was a bus error on a destination write.

9.3.1.3 eDMA Enable Request Registers (EDMA_ERQRH, EDMA_ERQRL)

The EDMA_ERQRH and EDMA_ERQRL provide a bit map for the 64 (MPC5554) or 32 (MPC5553) implemented channels to enable the request signal for each channel. For the MPC5554, EDMA_ERQRH supports channels 63–32, while EDMA_ERQRL covers channels 31–00. For the MPC5553,

EDMA_ERQRL maps to channels 31-0. EDMA_ERQRH is reserved on the MPC5553 and accessing it will result in a bus error.

The state of any given channel enable is directly affected by writes to these registers; the state is also affected by writes to the EDMA_SERQR and EDMA_CERQR. The EDMA_CERQR and EDMA_SERQR are provided so that the request enable for a *single* channel can easily be modified without the need to perform a read-modify-write sequence to the EDMA_ERQRH and EDMA_ERQRL.

Both the DMA request input signal and this enable request flag must be asserted before a channel's hardware service request is accepted. The state of the eDMA enable request flag does *not* affect a channel service request made explicitly through software or a linked channel request.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ERQ 63	ERQ 62	ERQ 61	ERQ 60	ERQ 59	ERQ 58	ERQ 57	ERQ 56	ERQ 55	ERQ 54	ERQ 53	ERQ 52	ERQ 51	ERQ 50	ERQ 49	ERQ 48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0008															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERQ 47	ERQ 46	ERQ 45	ERQ 44	ERQ 43	ERQ 42	ERQ 41	ERQ 40	ERQ 39	ERQ 38	ERQ 37	ERQ 36	ERQ 35	ERQ 34	ERQ 33	ERQ 32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0008															

Figure 9-4. eDMA Enable Request High Register (EDMA_ERQRH)—MPC5554 Only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ERQ 31	ERQ 30	ERQ 29	ERQ 28	ERQ 27	ERQ 26	ERQ 25	ERQ 24	ERQ 23	ERQ 22	ERQ 21	ERQ 20	ERQ 19	ERQ 18	ERQ 17	ERQ 16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x000C															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERQ 15	ERQ 14	ERQ 13	ERQ 12	ERQ 11	ERQ 10	ERQ 09	ERQ 08	ERQ 07	ERQ 06	ERQ 05	ERQ 04	ERQ 03	ERQ 02	ERQ 01	ERQ 00
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x000C															

Figure 9-5. eDMA Enable Request Low Register (EDMA_ERQRL)

Table 9-4. EDMA_ERQRH, EDMA_ERQRL Field Descriptions

Bits	Name	Description
0–31	ERQ*n*	Enable DMA hardware service request *n.* 0 The DMA request signal for channel n is disabled. 1 The DMA request signal for channel n is enabled.

As a given channel completes the processing of its major iteration count, there is a flag in the transfer control descriptor that may affect the ending state of the EDMA_ERQR bit for that channel. If the TCD.D_REQ bit is set, then the corresponding EDMA_ERQR bit is cleared after the major loop is complete, disabling the DMA hardware request. Otherwise if the D_REQ bit is cleared, the state of the EDMA_ERQR bit is unaffected.

9.3.1.4 eDMA Enable Error Interrupt Registers (EDMA_EEIRH, EDMA_EEIRL)

The EDMA_EEIRH and EDMA_EEIRL provide a bit map for the 64 channels (32 in the MPC5553) to enable the error interrupt signal for each channel. For the MPC5554, EDMA_EEIRH supports channels 63–32, while EDMA_EEIRL covers channels 31–00. For the MPC5553, EDMA_EEIRL maps to channels 31-0. EDMA_EEIRH is reserved on the MPC5553 and accessing it will result in a bus error.

The state of any given channel's error interrupt enable is directly affected by writes to these registers; it is also affected by writes to the EDMA_SEEIR and EDMA_CEEIR. The EDMA_SEEIR and EDMA_CEEIR are provided so that the error interrupt enable for a *single* channel can easily be modified without the need to perform a read-modify-write sequence to the EDMA_EEIRH and EDMA_EEIRL.

Both the DMA error indicator and this error interrupt enable flag must be asserted before an error interrupt request for a given channel is asserted.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EEI63	EEI62	EEI61	EEI60	EEI59	EEI58	EEI57	EEI56	EEI55	EEI54	EEI53	EEI52	EEI51	EEI50	EEI49	EEI48
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0010															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EEI47	EEI46	EEI45	EEI44	EEI43	EEI42	EEI41	EEI40	EEI39	EEI38	EEI37	EEI36	EEI35	EEI34	EEI33	EEI32
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0010															

Figure 9-6. eDMA Enable Error Interrupt High Register (EDMA_EEIRH)—MPC5554 Only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EEI31	EEI30	EEI29	EEI28	EEI27	EEI26	EEI25	EEI24	EEI23	EEI22	EEI21	EEI20	EEI19	EEI18	EEI17	EEI16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0014															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EEI15	EEI14	EEI13	EEI12	EEI11	EEI10	EEI09	EEI08	EEI07	EEI06	EEI05	EEI04	EEI03	EEI02	EEI01	EEI00
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0014															

Figure 9-7. eDMA Enable Error Interrupt Low Register (EDMA_EEIRL)

Table 9-5. EDMA_EEIRH, EDMA_EEIRL Field Descriptions

Bits	Name	Description
0–31	EEI*n*	Enable error interrupt *n.* 0 The error signal for channel *n* does not generate an error interrupt. 1 The assertion of the error signal for channel *n* generate an error interrupt request.

9.3.1.5 eDMA Set Enable Request Register (EDMA_SERQR)

The EDMA_SERQR provides a simple memory-mapped mechanism to set a given bit in the EDMA_ERQRH or EDMA_ERQRL to enable the DMA request for a given channel. The data value on a register write causes the corresponding bit in the EDMA_ERQRH or EDMA_ERQRL to be set. Setting bit 1 (SERQ*n*) provides a global set function, forcing the entire contents of EDMA_ERQRH and EDMA_ERQRL to be asserted. Reads of this register return all zeroes. For the MPC5553, bit 2 (SERQ1) is not used.

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	0
W		SERQ[0:6]						
Reset	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0018							

Figure 9-8. eDMA Set Enable Request Register (EDMA_SERQR)

Table 9-6. EDMA_SERQR Field Descriptions

Bits	Name	Description
0	—	Reserved.
1–7	SERQ [0:6]	Set enable request. 0–63 Set the corresponding bit in EDMA_ERQRH or EDMA_ERQRL 64–127 Set all bits in EDMA_ERQRH and EDMA_ERQRL

9.3.1.6 eDMA Clear Enable Request Register (EDMA_CERQR)

The EDMA_CERQR provides a simple memory-mapped mechanism to clear a given bit in the EDMA_ERQRH or EDMA_ERQRL to disable the DMA request for a given channel. The data value on a register write causes the corresponding bit in the EDMA_ERQRH or EDMA_ERQRL to be cleared. Setting bit 1 (CERQ*n*) provides a global clear function, forcing the entire contents of the EDMA_ERQRH and EDMA_ERQRL to be zeroed, disabling all DMA request inputs. Reads of this register return all zeroes. For the MPC5553, bit 2 (CERQ1) is not used.

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	0
W		CERQ[0:6]						
Reset	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0019							

Figure 9-9. eDMA Clear Enable Request Register (EDMA_CERQR)

Table 9-7. EDMA_CERQR Field Descriptions

Bits	Name	Description
0	—	Reserved.
1–7	CERQ [0:6]	Clear enable request. 0–63 Clear corresponding bit in EDMA_ERQRH or EDMA_ERQRL 64–127 Clear all bits in EDMA_ERQRH and EDMA_ERQRL

9.3.1.7 eDMA Set Enable Error Interrupt Register (EDMA_SEEIR)

The EDMA_SEEIR provides a simple memory-mapped mechanism to set a given bit in the EDMA_EEIRH or EDMA_EEIRL to enable the error interrupt for a given channel. The data value on a register write causes the corresponding bit in the EDMA_EEIRH or EDMA_EEIRL to be set. Setting bit 1 (SEEI*n*) provides a global set function, forcing the entire contents of EDMA_EEIRH or EDMA_EEIRL to be asserted. Reads of this register return all zeroes. For the MPC5553, bit 2 (SEEI1) is not used.

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	0
W		SEEI[0:6]						
Reset	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x001A							

Figure 9-10. eDMA Set Enable Error Interrupt Register (EDMA_SEEIR)

Table 9-8. EDMA_SEEIR Field Descriptions

Bits	Name	Description
0	—	Reserved.
1–7	SEEI [0:6]	Set enable error interrupt. 0–63 Set the corresponding bit in EDMA_EEIRH or EDMA_EEIRL 64–127 Set all bits in EDMA_EEIRH and EDMA_EEIRL

9.3.1.8 eDMA Clear Enable Error Interrupt Register (EDMA_CEEIR)

The EDMA_CEEIR provides a simple memory-mapped mechanism to clear a given bit in the EDMA_EEIRH or EDMA_EEIRL to disable the error interrupt for a given channel. The data value on a register write causes the corresponding bit in the EDMA_EEIRH or EDMA_EEIRL to be cleared. Setting bit 1 (CEEI*n*) provides a global clear function, forcing the entire contents of the EDMA_EEIRH or EDMA_EEIRL to be zeroed, disabling error interrupts for all channels. Reads of this register return all zeroes. For the MPC5553, bit 2 (CEEI1) is not used.

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	0
W		CEEI[0:6]						
Reset	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x001B							

Figure 9-11. eDMA Clear Enable Error Interrupt Register (EDMA_CEEIR)

Table 9-9. EDMA_CEEIR Field Descriptions

Bits	Name	Description
0	—	Reserved.
1–7	CEEI [0:6]	Clear enable error interrupt 0–63 Clear corresponding bit in EDMA_EEIRH or EDMA_EEIRL 64–127 Clear all bits in EDMA_EEIRH and EDMA_EEIRL

9.3.1.9 eDMA Clear Interrupt Request Register (EDMA_CIRQR)

The EDMA_CIRQR provides a simple memory-mapped mechanism to clear a given bit in the EDMA_IRQRH or EDMA_IRQRL to disable the interrupt request for a given channel. The given value on a register write causes the corresponding bit in the EDMA_IRQRH or EDMA_IRQRL to be cleared. Setting bit 1 (CINT*n*) provides a global clear function, forcing the entire contents of the EDMA_IRQRH or EDMA_IRQRL to be zeroed, disabling all DMA interrupt requests. Reads of this register return all zeroes. For the MPC5553, bit 2 (CINT1) is not used.

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	0
W		CINT[0:6]						
Reset	0	0	0	0	0	0	0	0
Reg Addr	Base + 0X001C							

Figure 9-12. eDMA Clear Interrupt Request (EDMA_CIRQR) Fields

Table 9-10. EDMA_CIRQR Field Descriptions

Bits	Name	Description
0	—	Reserved.
1–7	CINT [0:6]	Clear interrupt request. 0–63 Clear the corresponding bit in EDMA_IRQRH or EDMA_IRQRL 64–127 Clear all bits in EDMA_IRQRH or EDMA_IRQRL

9.3.1.10 eDMA Clear Error Register (EDMA_CER)

The EDMA_CER provides a simple memory-mapped mechanism to clear a given bit in the EDMA_ERH or EDMA_ERL to disable the error condition flag for a given channel. The given value on a register write causes the corresponding bit in the EDMA_ERH or EDMA_ERL to be cleared. Setting bit 1 (CERR*n*) provides a global clear function, forcing the entire contents of the EDMA_ERH and EDMA_ERL to be zeroed, clearing all channel error indicators. Reads of this register return all zeroes. For the MPC5553, bit 2 (CERR1) is not used.

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	0
W		CERR[0:6]						
Reset	0	0	0	0	0	0	0	0
Reg Addr	Base + 0X001D							

Figure 9-13. eDMA Clear Error Register (EDMA_CER)

Table 9-11. EDMA_CER Field Descriptions

Bits	Name	Description
0	—	Reserved.
1–7	CERR [0:6]	Clear error indicator. 0–63 Clear corresponding bit in EDMA_ERH or EDMA_ERL 64–127 Clear all bits in EDMA_ERH and EDMA_ERL

9.3.1.11 eDMA Set START Bit Register (EDMA_SSBR)

The EDMA_SSBR provides a simple memory-mapped mechanism to set the START bit in the TCD of the given channel. The data value on a register write causes the START bit in the corresponding transfer control descriptor to be set. Setting bit 1 (SSB*n*) provides a global set function, forcing all START bits to be set. Reads of this register return all zeroes. For the MPC5553, bit 2 (SSB1) is not used.

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	0
W		SSB[0:6]						
Reset	0	0	0	0	0	0	0	0
Reg Addr	Base + 0X001E							

Figure 9-14. eDMA Set START Bit Register (EDMA_SSBR)

Table 9-12. EDMA_SSBR Field Descriptions

Bits	Name	Description
0	—	Reserved.
1–7	SSB [0:6]	Set START bit (channel service request). 0–63 Set the corresponding channel's TCD.START 64–127 Set all TCD.START bits

9.3.1.12 eDMA Clear DONE Status Bit Register (EDMA_CDSBR)

The EDMA_CDSBR provides a simple memory-mapped mechanism to clear the DONE bit in the TCD of the given channel. The data value on a register write causes the DONE bit in the corresponding transfer control descriptor to be cleared. Setting bit 1 (CDSB*n*) provides a global clear function, forcing all DONE bits to be cleared. Reads of this register return all zeroes. For the MPC5553, bit 2 (CDSB1) is not used.

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	0
W		CDSB[0:6]						
Reset	0	0	0	0	0	0	0	0
Reg Addr	Base + 0X001F							

Figure 9-15. eDMA Clear DONE Status Bit Register (EDMA_CDSBR)

Table 9-13. EDMA_CDSBR Field Descriptions

Bits	Name	Description
0	—	Reserved.
1–7	CDSB [0:6]	Clear DONE status bit. 0–63 Clear the corresponding channel's DONE bit 64–127 Clear all TCD DONE bits

9.3.1.13 eDMA Interrupt Request Registers (EDMA_IRQRH, EDMA_IRQRL)

The EDMA_IRQRH and EDMA_IRQRL provide a bit map for the 64 channels signaling the presence of an interrupt request for each channel. For the MPC5554, EDMA_IRQRH supports channels 63–32, while EDMA_IRQRL covers channels 31–00. For the MPC5553, EDMA_IRQRL maps to channels 31-0. EDMA_IRQRH is reserved on the MPC5553 and accessing it will result in a bus error.

The eDMA engine signals the occurrence of a programmed interrupt upon the completion of a data transfer as defined in the transfer control descriptor by setting the appropriate bit in this register. The outputs of this register are directly routed to the interrupt controller (INTC). During the execution of the interrupt service routine associated with any given channel, it is software's responsibility to clear the appropriate bit, negating the interrupt request. Typically, a write to the EDMA_CIRQR in the interrupt service routine is used for this purpose.

The state of any given channel's interrupt request is directly affected by writes to this register; it is also affected by writes to the EDMA_CIRQR. On writes to the EDMA_IRQRH or EDMA_IRQRL, a 1 in any bit position clears the corresponding channel's interrupt request. A 0 in any bit position has no affect on the corresponding channel's current interrupt status. The EDMA_CIRQR is provided so the interrupt

request for a *single* channel can easily be cleared without the need to perform a read-modify-write sequence to the EDMA_IRQRH and EDMA_IRQRL.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	INT 63	INT 62	INT 61	INT 60	INT 59	INT 58	INT 57	INT 56	INT 55	INT 54	INT 53	INT 52	INT 51	INT 50	INT 49	INT 48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0020															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	INT 47	INT 46	INT 45	INT 44	INT 43	INT 42	INT 41	INT 40	INT 39	INT 38	INT 37	INT 36	INT 35	INT 34	INT 33	INT 32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0020															

Figure 9-16. eDMA Interrupt Request High Register (EDMA_IRQRH)—MPC5554 Only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	INT 31	INT 30	INT 29	INT 28	INT 27	INT 26	INT 25	INT 24	INT 23	INT 22	INT 21	INT 20	INT 19	INT 18	INT 17	INT 16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0024															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	INT 15	INT 14	INT 13	INT 12	INT 11	INT 10	INT 09	INT 08	INT 07	INT 06	INT 05	INT 04	INT 03	INT 02	INT 01	INT 00
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0024															

Figure 9-17. eDMA Interrupt Request Low Register (EDMA_IRQRL)

Table 9-14. EDMA_IRQRH, EDMA_IRQRL Field Descriptions

Bits	Name	Description
0–31	INT*n*	eDMA interrupt request *n.* 0 The interrupt request for channel n is cleared. 1 The interrupt request for channel n is active.

9.3.1.14 eDMA Error Registers (EDMA_ERH, EDMA_ERL)

The EDMA_ERH and EDMA_ERL provide a bit map for the 64 channels signaling the presence of an error for each channel. For the MPC5554, EDMA_ERH supports channels 63–32, while EDMA_ERL covers channels 31–00. For the MPC5553, EDMA_ERL maps to channels 31-0. EDMA_ERH is reserved on the MPC5553 and accessing it will result in a bus error.

The eDMA engine signals the occurrence of a error condition by setting the appropriate bit in this register. The outputs of this register are enabled by the contents of the EDMA_EEIR, then logically summed across groups of 16, 32, and 64 channels (MPC5554) or 16 and 32 channels (MPC5553) to form several group

error interrupt requests which is then routed to the interrupt controller. During the execution of the interrupt service routine associated with any DMA errors, it is software's responsibility to clear the appropriate bit, negating the error interrupt request. Typically, a write to the EDMA_CER in the interrupt service routine is used for this purpose. Recall the normal DMA channel completion indicators, setting the transfer control descriptor DONE flag and the possible assertion of an interrupt request, are *not* affected when an error is detected.

The contents of this register can also be polled and a non-zero value indicates the presence of a channel error, regardless of the state of the EDMA_EEIR. The EDMA_ESR[VLD] bit is a logical OR of all bits in this register and it provides a single bit indication of any errors. The state of any given channel's error indicators is affected by writes to this register; it is also affected by writes to the EDMA_CER. On writes to EDMA_ERH or EDMA_ERL, a 1 in any bit position clears the corresponding channel's error status. A 0 in any bit position has no affect on the corresponding channel's current error status. The EDMA_CER is provided so the error indicator for a *single* channel can easily be cleared.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ERR 63	ERR 62	ERR 61	ERR 60	ERR 59	ERR 58	ERR 57	ERR 56	ERR 55	ERR 54	ERR 53	ERR 52	ERR 51	ERR 50	ERR 49	ERR 48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0028															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERR 47	ERR 46	ERR 45	ERR 44	ERR 43	ERR 42	ERR 41	ERR 40	ERR 39	ERR 38	ERR 37	ERR 36	ERR 35	ERR 34	ERR 33	ERR 32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0028															

Figure 9-18. eDMA Error High Register (EDMA_ERH)—MPC5554 Only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ERR 31	ERR 30	ERR 29	ERR 28	ERR 27	ERR 26	ERR 25	ERR 24	ERR 23	ERR 22	ERR 21	ERR 20	ERR 19	ERR 18	ERR 17	ERR 16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Base + 0X002C															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERR 15	ERR 14	ERR 13	ERR 12	ERR 11	ERR 10	ERR 09	ERR 08	ERR 07	ERR 06	ERR 05	ERR 04	ERR 03	ERR 02	ERR 01	ERR 00
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Base + 0X002C															

Figure 9-19. eDMA Error Low Register (EDMA_ERL)

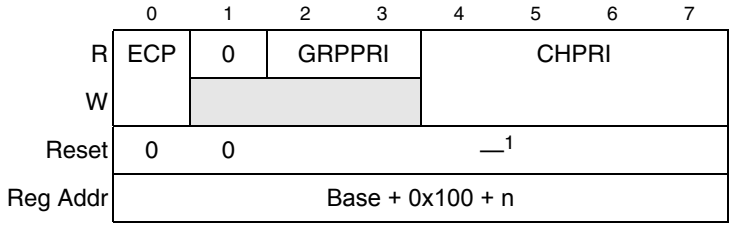
Table 9-15. EDMA_ERH, EDMA_ERL Field Descriptions

Bits	Name	Description
0–31	ERR*n*	eDMA Error *n.* 0 An error in channel *n* has not occurred. 1 An error in channel *n* has occurred.

9.3.1.15 eDMA Channel *n* Priority Registers (EDMA_CPR*n*)

When the fixed-priority channel arbitration mode is enabled (EDMA_CR[ERCA] = 0), the contents of these registers define the unique priorities associated with each channel within a group. The channel priorities are evaluated by numeric value; that is, 0 is the lowest priority, 1 is the next higher priority, then 2, 3, etc. Software must program the channel priorities with unique values, otherwise a configuration error will be reported. The range of the priority value is limited to the values of 0 through 15. When read, the GRPPRI bits of the EDMA_CPR*n* register reflect the current priority level of the group of channels in which the corresponding channel resides. GRPPRI bits are not affected by writes to the EDMA_CPR*n* registers. The group priority is assigned in the EDMA_CR. See Figure 9-2 and Table 9-2 for the EDMA_CR definition.

Channel preemption is enabled on a per-channel basis by setting the ECP bit in the EDMA_CPR*n* register. Channel preemption allows the executing channel's data transfers to be temporarily suspended in favor of starting a higher priority channel. Once the preempting channel has completed all of its minor loop data transfers, the preempted channel is restored and resumes execution. After the restored channel completes one read/write sequence, it is again eligible for preemption. If any higher priority channel is requesting service, the restored channel will be suspended and the higher priority channel will be serviced. Nested preemption (attempting to preempt a preempting channel) is not supported. Once a preempting channel begins execution, it cannot be preempted. Preemption is only available when fixed arbitration is selected for both group and channel arbitration modes.



[1]The reset value for the group and channel priority fields, GRPPRI[0–1] and CHPRI[0–3], is equal to the corresponding channel number for each priority register; that is, EDMA_CPR31[GRPPRI] = 0b01 and EDMA_CPR31[CHPRI] = 0b1111.

Figure 9-20. eDMA Channel *n* Priority Register (EDMA_CPR*n*)

Table 9-16. EDMA_CPR*n* Field Descriptions

Bits	Name	Description
0	ECP	Enable channel preemption. 0 Channel *n* cannot be suspended by a higher priority channel's service request. 1 Channel *n* can be temporarily suspended by the service request of a higher priority channel.
1	—	Reserved.

Table 9-16. EDMA_CPR*n* Field Descriptions (continued)

Bits	Name	Description
2–3	GRPPRI [0:1]	Channel *n* current group priority. Group priority assigned to this channel group when fixed-priority arbitration is enabled. These two bits are read only; writes are ignored.
4–7	CHPRI [0:3]	Channel *n* arbitration priority. Channel priority when fixed-priority arbitration is enabled.

9.3.1.16 Transfer Control Descriptor (TCD)

Each channel requires a 256-bit transfer control descriptor for defining the desired data movement operation. The channel descriptors are stored in the local memory in sequential order: channel 0, channel 1,... channel 63 (MPC5554) or channel 0, channel 1,... channel 31 (MPC5553). The definitions of the TCD are presented as twenty-three variable-length fields. Table 9-17 is field list of the basic TCD structure.

Table 9-17. TCD*n* 32-bit Memory Structure

eDMA Bit Offset	Length	TCD*n* Field Name	TCD*n* Abbreviation
0x1000 + (32 x n) + 0	32	Source Address	SADDR
0x1000 + (32 x n) + 32	5	Source address modulo	SMOD
0x1000 + (32 x n) + 37	3	Source data transfer size	SSIZE
0x1000 + (32 x n) + 40	5	Destination address modulo	DMOD
0x1000 + (32 x n) + 45	3	Destination data transfer size	DSIZE
0x1000 + (32 x n) + 48	16	Signed Source Address Offset	SOFF
0x1000 + (32 x n) + 64	32	Inner Minor Byte Count	NBYTES
0x1000 + (32 x n) + 96	32	Last Source Address Adjustment	SLAST
0x1000 + (32 x n) + 128	32	Destination Address	DADDR
0x1000 + (32 x n) + 160	1	Channel-to-channel Linking on Minor Loop Complete	CITER.E_LINK
0x1000 + (32 x n) + 161	6	Current “Major” Iteration Count or Link Channel Number	CITER or CITER.LINKCH
0x1000 + (32 x n) + 167	9	Current Major Iteration Count	CITER
0x1000 + (32 x n) + 176	16	Destination Address Offset (Signed)	DOFF
0x1000 + (32 x n) + 192	32	Last Destination Address Adjustment / Scatter Gather Address	DLAST_SGA
0x1000 + (32 x n) + 224	1	Channel-to-channel Linking on Minor Loop Complete	BITER.E_LINK
0x1000 + (32 x n) + 225	6	Starting Major Iteration Count or Link Channel Number	BITER or BITER.LINKCH
0x1000 + (32 x n) + 231	9	Starting Major Iteration Count	BITER
0x1000 + (32 x n) +240	2	Bandwidth Control	BWC
0x1000 + (32 x n) + 242	6	Link Channel Number	MAJOR.LINKCH
0x1000 + (32 x n) + 248	1	Channel Done	DONE

Table 9-17. TCD*n* 32-bit Memory Structure (continued)

0x1000 + (32 x n) + 249	1	Channel Active	ACTIVE
0x1000 + (32 x n) + 250	1	Channel-to-channel Linking on Major Loop Complete	MAJOR.E_LINK
0x1000 + (32 x n) + 251	1	Enable Scatter/Gather Processing	E_SG
0x1000 + (32 x n) + 252	1	Disable Request	D_REQ
0x1000 + (32 x n) + 253	1	Channel Interrupt Enable When Current Major Iteration Count is Half Complete	INT_HALF
0x1000 + (32 x n) + 254	1	Channel Interrupt Enable When Current Major Iteration Count Complete	INT_MAJ
0x1000 + (32 x n) + 255	1	Channel Start	START

Figure 9-21 and Table 9-18 define the fields of the TCDn structure.

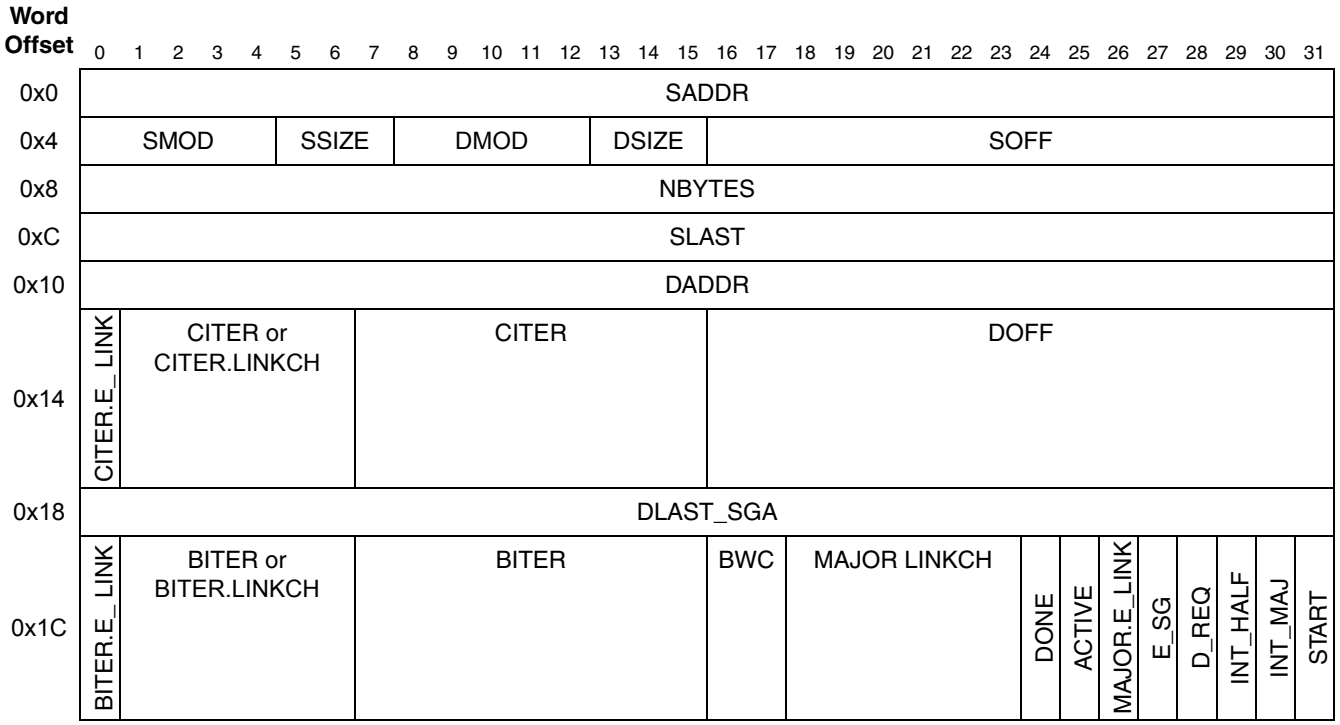


Figure 9-21. TCD Structure

NOTE

The TCD structures for the eDMA channels shown in Figure 9-21 are implemented in internal SRAM. These structures are not initialized at reset. Therefore, all channel TCD parameters must be initialized by the application code before activating that channel.

Table 9-18. TCD*n* Field Descriptions

Bits / Word Offset [n:n]	Name	Description
0–31 / 0x0 [0:31]	SADDR [0:31]	Source address. Memory address pointing to the source data. Word 0x0, bits 0–31.
32–36 / 0x4 [0:4]	SMOD [0:4]	Source address modulo. 0 Source address modulo feature is disabled. non-0 This value defines a specific address range which is specified to be either the value after SADDR + SOFF calculation is performed or the original register value. The setting of this field provides the ability to easily implement a circular data queue. For data queues requiring power-of-2 “size” bytes, the queue should start at a 0-modulo-size address and the SMOD field should be set to the appropriate value for the queue, freezing the desired number of upper address bits. The value programmed into this field specifies the number of lower address bits that are allowed to change. For this circular queue application, the SOFF is typically set to the transfer size to implement post-increment addressing with the SMOD function constraining the addresses to a 0-modulo-size range.
37–39 / 0x4 [5:7]	SSIZE [0:2]	Source data transfer size. 000 8-bit 001 16-bit 010 32-bit 011 64-bit 100 Reserved 101 32-byte burst (64-bit x 4) 110 Reserved 111 Reserved The attempted specification of a ‘reserved’ encoding will cause a configuration error.
40–44 / 0x4 [8:12]	DMOD [0:4]	Destination address modulo. See the SMOD[0:5] definition.
45–47 / 0x4 [13:15]	DSIZE [0:2]	Destination data transfer size. See the SSIZE[0:2] definition.
48–63 / 0x4 [16:31]	SOFF [0:15]	Source address signed offset. Sign-extended offset applied to the current source address to form the next-state value as each source read is completed.
64–95 / 0x8 [0:31]	NBYTES [0:31]	Inner “minor” byte transfer count. Number of bytes to be transferred in each service request of the channel. As a channel is activated, the contents of the appropriate TCD is loaded into the eDMA engine, and the appropriate reads and writes performed until the complete byte transfer count has been transferred. This is an indivisible operation and cannot be stalled or halted. Once the minor count is exhausted, the current values of the SADDR and DADDR are written back into the local memory, the major iteration count is decremented and restored to the local memory. If the major iteration count is completed, additional processing is performed. **Note:** The NBYTES value of 0x0000_0000 is interpreted as 0x1_0000_0000, thus specifying a 4 GByte transfer.
96–127 / 0xC [0:31]	SLAST [0:31]	Last source address adjustment. Adjustment value added to the source address at the completion of the outer major iteration count. This value can be applied to “restore” the source address to the initial value, or adjust the address to reference the next data structure.

Table 9-18. TCD*n* Field Descriptions (continued)

Bits / Word Offset [n:n]	Name	Description
128–159 / 0x10 [0:31]	DADDR [0:31]	Destination address. Memory address pointing to the destination data.
160 / 0x14 [0]	CITER.E_LINK	Enable channel-to-channel linking on minor loop completion. As the channel completes the inner minor loop, this flag enables the linking to another channel, defined by CITER.LINKCH[0:5]. The link target channel initiates a channel service request via an internal mechanism that sets the TCD.START bit of the specified channel. If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJOR.E_LINK channel linking. 0 The channel-to-channel linking is disabled. 1 The channel-to-channel linking is enabled. **Note:** This bit must be equal to the BITER.E_LINK bit otherwise a configuration error will be reported.
161–166 / 0x14 [1:6]	CITER [0:5] or CITER.LINKCH [0:5]	Current “major” iteration count or link channel number. If channel-to-channel linking is disabled (TCD.CITER.E_LINK = 0), then • No channel-to-channel linking (or chaining) is performed after the inner minor loop is exhausted. TCD bits [161:175] are used to form a 15-bit CITER field. otherwise • After the minor loop is exhausted, the eDMA engine initiates a channel service request at the channel defined by CITER.LINKCH[0:5] by setting that channel’s TCD.START bit.
167–175 / 0x14 [7:15]	CITER [6:14]	Current “major” iteration count. This 9 or 15-bit count represents the current major loop count for the channel. It is decremented each time the minor loop is completed and updated in the transfer control descriptor memory. Once the major iteration count is exhausted, the channel performs a number of operations (for example, final source and destination address calculations), optionally generating an interrupt to signal channel completion before reloading the CITER field from the beginning iteration count (BITER) field. **Note:** When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field. **Note:** If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.
176–191 / 0x14 [16:31]	DOFF [0:15]	Destination address signed offset. Sign-extended offset applied to the current destination address to form the next-state value as each destination write is completed.
192–223 / 0x18 [0:31]	DLAST_SGA [0:31]	Last destination address adjustment or the memory address for the next transfer control descriptor to be loaded into this channel (scatter/gather). If scatter/gather processing for the channel is disabled (TCD.E_SG = 0) then • Adjustment value added to the destination address at the completion of the outer major iteration count. This value can be applied to “restore” the destination address to the initial value, or adjust the address to reference the next data structure. Otherwise • This address points to the beginning of a 0-modulo-32 byte region containing the next transfer control descriptor to be loaded into this channel. This channel reload is performed as the major iteration count completes. The scatter/gather address must be 0-modulo-32 byte, otherwise a configuration error is reported.

Table 9-18. TCD*n* Field Descriptions (continued)

Bits / Word Offset [n:n]	Name	Description
224 / 0x1C [0]	BITER.E_LINK	Enables channel-to-channel linking on minor loop complete. As the channel completes the inner minor loop, this flag enables the linking to another channel, defined by BITER.LINKCH[0:5]. The link target channel initiates a channel service request via an internal mechanism that sets the TCD.START bit of the specified channel. If channel linking is disabled, the BITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJOR.E_LINK channel linking. 0 The channel-to-channel linking is disabled. 1 The channel-to-channel linking is enabled. **Note:** When the TCD is first loaded by software, this field must be set equal to the corresponding CITER field, otherwise a configuration error will be reported. As the major iteration count is exhausted, the contents of this field is reloaded into the CITER field.
225–230 / 0x1C [1:6]	BITER [0:5] or BITER.LINKCH[0:5]	Starting “major” iteration count or link channel number. If channel-to-channel linking is disabled (TCD.BITER.E_LINK = 0), then • No channel-to-channel linking (or chaining) is performed after the inner minor loop is exhausted. TCD bits [225:239] are used to form a 15-bit BITER field. Otherwise • After the minor loop is exhausted, the eDMA engine initiates a channel service request at the channel, defined by BITER.LINKCH[0:5], by setting that channel’s TCD.START bit. **Note:** When the TCD is first loaded by software, this field must be set equal to the corresponding CITER field, otherwise a configuration error will be reported. As the major iteration count is exhausted, the contents of this field is reloaded into the CITER field.
231–239 / 0x1C [7:15]	BITER [6:14]	Starting major iteration count. As the transfer control descriptor is first loaded by software, this field must be equal to the value in the CITER field. As the major iteration count is exhausted, the contents of this field is reloaded into the CITER field. **Note:** If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.
240–241 / 0x1C [16:17]	BWC [0:1]	Bandwidth control. This two-bit field provides a mechanism to effectively throttle the amount of bus bandwidth consumed by the eDMA. In general, as the eDMA processes the inner minor loop, it continuously generates read/write sequences until the minor count is exhausted. This field forces the eDMA to stall after the completion of each read/write access to control the bus request bandwidth seen by the system bus crossbar switch (XBAR). 00 No eDMA engine stalls 01 Reserved 10 eDMA engine stalls for 4 cycles after each r/w 11 eDMA engine stalls for 8 cycles after each r/w
242–247 / 0x1C [18:23]	MAJOR.LINKC H [0:5]	Link channel number. If channel-to-channel linking on major loop complete is disabled (TCD.MAJOR.E_LINK = 0) then • No channel-to-channel linking (or chaining) is performed after the outer major loop counter is exhausted. Otherwise • After the major loop counter is exhausted, the eDMA engine initiates a channel service request at the channel defined by MAJOR.LINKCH[0:5] by setting that channel’s TCD.START bit.

Table 9-18. TCD*n* Field Descriptions (continued)

Bits / Word Offset [n:n]	Name	Description
248 / 0x1C [24]	DONE	Channel done. This flag indicates the eDMA has completed the outer major loop. It is set by the eDMA engine as the CITER count reaches zero; it is cleared by software or hardware when the channel is activated (when the channel has begun to be processed by the eDMA engine, not when the first data tranfer occurs). **Note:** This bit must be cleared in order to write the MAJOR.E_LINK or E_SG bits.
249 / 0x1C [25]	ACTIVE	Channel active. This flag signals the channel is currently in execution. It is set when channel service begins, and is cleared by the eDMA engine as the inner minor loop completes or if any error condition is detected.
250 / 0x1C [26]	MAJOR.E_LINK	Enable channel-to-channel linking on major loop completion. As the channel completes the outer major loop, this flag enables the linking to another channel, defined by MAJOR.LINKCH[0:5]. The link target channel initiates a channel service request via an internal mechanism that sets the TCD.START bit of the specified channel. NOTE: To support the dynamic linking coherency model, this field is forced to zero when written to while the TCD.DONE bit is set. 0 The channel-to-channel linking is disabled. 1 The channel-to-channel linking is enabled.
251 / 0x1C [27]	E_SG	Enable scatter/gather processing. As the channel completes the outer major loop, this flag enables scatter/gather processing in the current channel. If enabled, the eDMA engine uses DLAST_SGA as a memory pointer to a 0-modulo-32 address containing a 32-byte data structure which is loaded as the transfer control descriptor into the local memory. NOTE: To support the dynamic scatter/gather coherency model, this field is forced to zero when written to while the TCD.DONE bit is set. 0 The current channel's TCD is "normal" format. 1 The current channel's TCD specifies a scatter gather format. The DLAST_SGA field provides a memory pointer to the next TCD to be loaded into this channel after the outer major loop completes its execution.
252 / 0x1C [28]	D_REQ	Disable hardware request. If this flag is set, the eDMA hardware automatically clears the corresponding EDMA_ERQH or EDMA_ERQL bit when the current major iteration count reaches zero. 0 The channel's EDMA_ERQH or EDMA_ERQL bit is not affected. 1 The channel's EDMA_ERQH or EDMA_ERQL bit is cleared when the outer major loop is complete.
253 / 0x1C [29]	INT_HALF	Enable an interrupt when major counter is half complete. If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the EDMA_ERQH or EDMA_ERQL when the current major iteration count reaches the halfway point. Specifically, the comparison performed by the eDMA engine is (CITER == (BITER >> 1)). This halfway point interrupt request is provided to support double-buffered (aka ping-pong) schemes, or other types of data movement where the processor needs an early indication of the transfer's progress. CITER = BITER = 1 with INT_HALF enabled will generate an interrupt as it satisfies the equation (CITER == (BITER >> 1)) after a single activation. 0 The half-point interrupt is disabled. 1 The half-point interrupt is enabled.

Table 9-18. TCD*n* Field Descriptions (continued)

Bits / Word Offset [n:n]	Name	Description
254 / 0x1C [30]	INT_MAJ	Enable an interrupt when major iteration count completes. If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the EDMA_ERQH or EDMA_ERQL when the current major iteration count reaches zero. 0 The end-of-major loop interrupt is disabled. 1 The end-of-major loop interrupt is enabled.
255 / 0x1C [31]	START	Channel start. If this flag is set, the channel is requesting service. The eDMA hardware automatically clears this flag after the channel begins execution. 0 The channel is not explicitly started. 1 The channel is explicitly started via a software initiated service request.

9.4 Functional Description

This section provides an overview of the microarchitecture and functional operation of the eDMA module.

9.4.1 eDMA Microarchitecture

The eDMA module is partitioned into two major modules: the eDMA engine and the transfer control descriptor local memory. Additionally, the eDMA engine is further partitioned into four submodules, which are detailed below.

- eDMA engine
  - Address path: This module implements registered versions of two channel transfer control descriptors: channel ‘x’ and channel ‘y,’ and is responsible for all the master bus address calculations. All the implemented channels provide the exact same functionality. This hardware structure allows the data transfers associated with one channel to be preempted after the completion of a read/write sequence if a higher priority channel service request is asserted while the first channel is active. Once a channel is activated, it runs until the minor loop is completed unless preempted by a higher priority channel. This capability provides a mechanism (optionally enabled by EDMA_CPR*n*[ECP]) where a large data move operation can be preempted to minimize the time another channel is blocked from execution.

    When any other channel is activated, the contents of its transfer control descriptor is read from the local memory and loaded into the registers of the other address path channel{x,y}. Once the inner minor loop completes execution, the address path hardware writes the new values for the TCD*n*.{SADDR, DADDR, CITER} back into the local memory. If the major iteration count is exhausted, additional processing is performed, including the final address pointer updates, reloading the TCDn.CITER field, and a possible fetch of the next TCDn from memory as part of a scatter/gather operation.
  - Data path: This module implements the actual bus master read/write datapath. It includes 32 bytes of register storage (matching the maximum transfer size) and the necessary mux logic to support any required data alignment. The system read data bus is the primary input, and the system write data bus is the primary output.

    The address and data path modules directly support the 2-stage pipelined system bus. The address path module represents the 1st stage of the bus pipeline (the address phase), while the data path module implements the 2nd stage of the pipeline (the data phase).

— Program model/channel arbitration: This module implements the first section of eDMA's programming model as well as the channel arbitration logic. The programming model registers are connected to the slave bus (not shown). The eDMA peripheral request inputs and eDMA interrupt request outputs are also connected to this module (via the Control logic).

— Control: This module provides all the control functions for the eDMA engine. For data transfers where the source and destination sizes are equal, the eDMA engine performs a series of source read, destination write operations until the number of bytes specified in the inner 'minor loop' byte count has been moved.

A minor loop interation is defined as the number of bytes to transfer (*n*bytes) divided by the transfer size. Transfer size is defined as the following"

if (ssize < dsize)

transfer size = destination transfer size (# of bytes)

else

transfer size = source transfer size (# of bytes)

Minor loop TCD variables are soff, smod, doff, dmod, nbytes, saddr, daddr, bwc, active, and start . Major loop TCD variables are dlast, slast, citer, biter, done, d_req, int_maj, major_lnkch, and int_half.

For descriptors where the sizes are not equal, multiple access of the smaller size data are required for each reference of the larger size. As an example, if the source size references 16-bit data and the destination is 32-bit data, two reads are performed, then one 32-bit write.

- TCD local memory
  — Memory controller: This logic implements the required dual-ported controller, handling accesses from both the eDMA engine as well as references from the slave bus. As noted earlier, in the event of simultaneous accesses, the eDMA engine is given priority and the slave transaction is stalled. The hooks to a BIST controller for the local TCD memory are included in this module.
  — Memory array: The TCD is implemented using a single-ported, synchronous compiled RAM memory array.

9.4.2 eDMA Basic Data Flow

The basic flow of a data transfer can be partitioned into three segments. As shown in Figure 9-22, the first segment involves the channel service request. In the diagram, this example uses the assertion of the eDMA peripheral request signal to request service for channel *n*. Channel service request via software and the TCDn.START bit follows the same basic flow as an eDMA peripheral request. The eDMA peripheral request input signal is registered internally and then routed to through the eDMA engine, first through the control module, then into the program model/channel arbitration module. In the next cycle, the channel arbitration is performed, either using the fixed-priority or round-robin algorithm. After the arbitration is complete, the activated channel number is sent through the address path and converted into the required address to access the TCD local memory. Next, the TCD memory is accessed and the required descriptor read from the local memory and loaded into the eDMA engine address path channel{x,y} registers. The TCD memory is organized 64-bits in width to minimize the time needed to fetch the activated channel's descriptor and load it into the eDMA engine address path channel{x,y} registers.

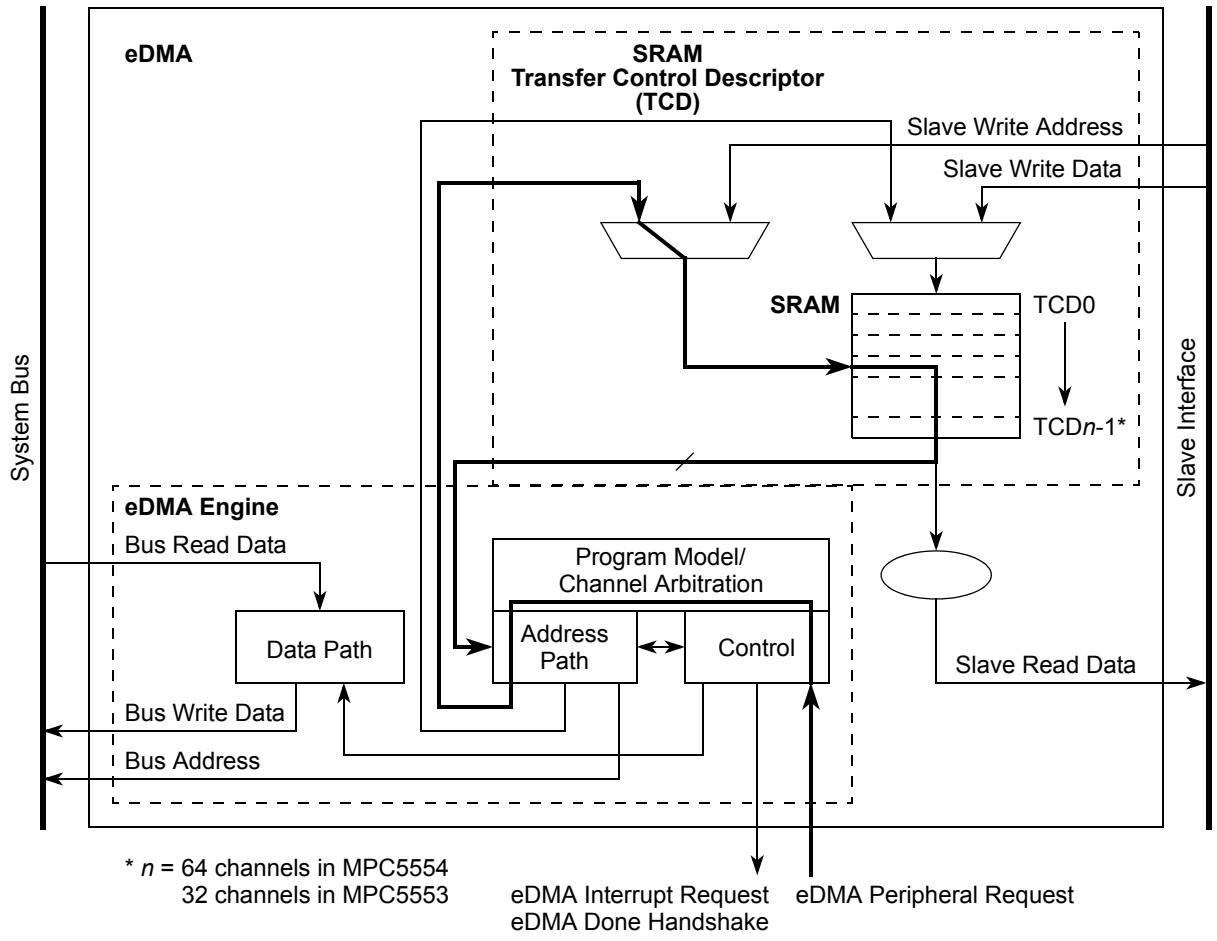


Figure 9-22. eDMA Operation, Part 1

In the second part of the basic data flow as shown in Figure 9-23, the modules associated with the data transfer (address path, data path and control) sequence through the required source reads and destination writes to perform the actual data movement. The source reads are initiated and the fetched data is temporarily stored in the data path module until it is gated onto the system bus during the destination write.

This source read/destination write processing continues until the inner minor byte count has been transferred. The eDMA Done Handshake signal is asserted at the end of the minor byte count transfer.

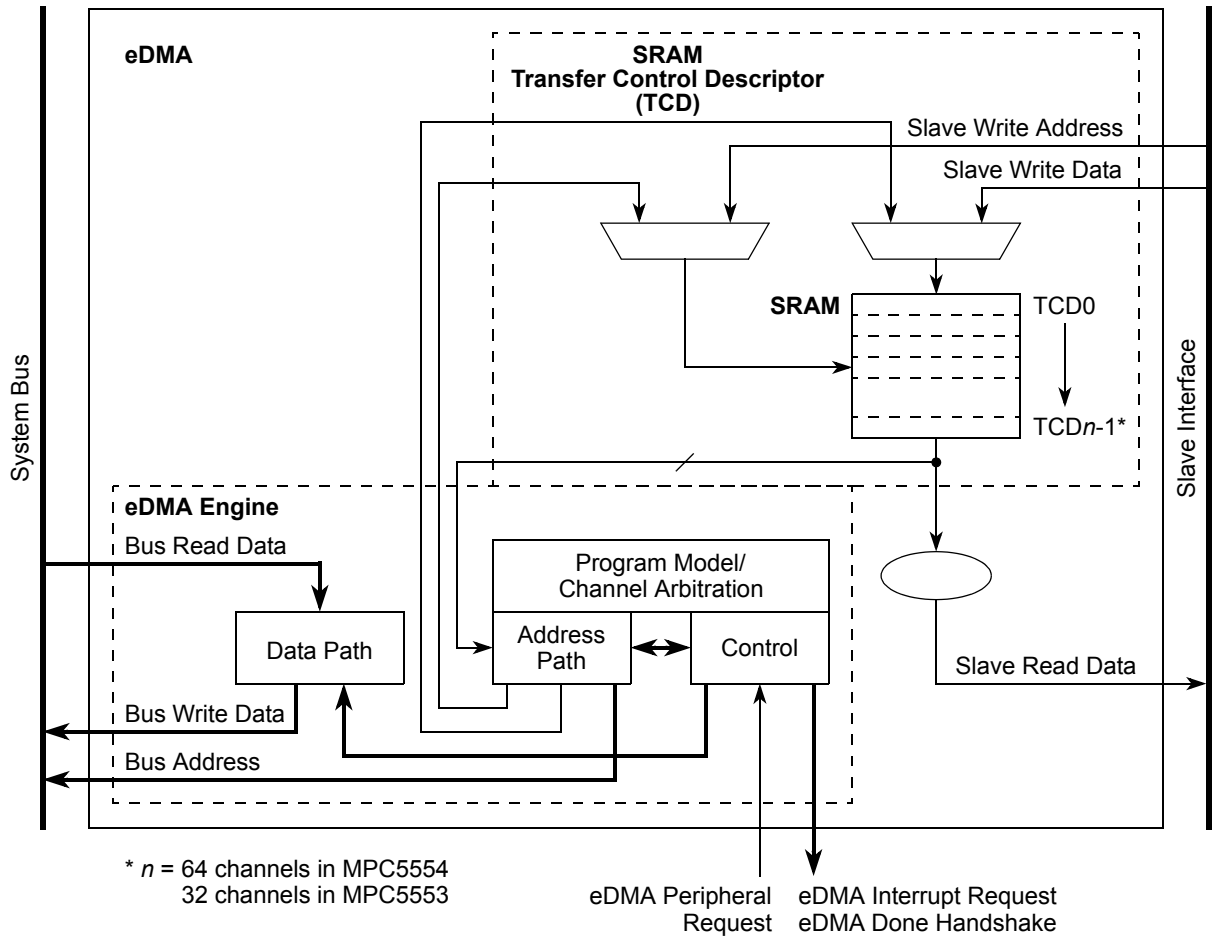


Figure 9-23. eDMA Operation, Part 2

Once the inner minor byte count has been moved, the final phase of the basic data flow is performed. In this segment, the address path logic performs the required updates to certain fields in the channel's TCD: for example., SADDR, DADDR, CITER. If the outer major iteration count is exhausted, then there are additional operations which are performed. These include the final address adjustments and reloading of the BITER field into the CITER. Additionally, assertion of an optional interrupt request occurs at this time, as does a possible fetch of a new TCD from memory using the scatter/gather address pointer included in

the descriptor. The updates to the TCD memory and the assertion of an interrupt request are shown in Figure 9-24.

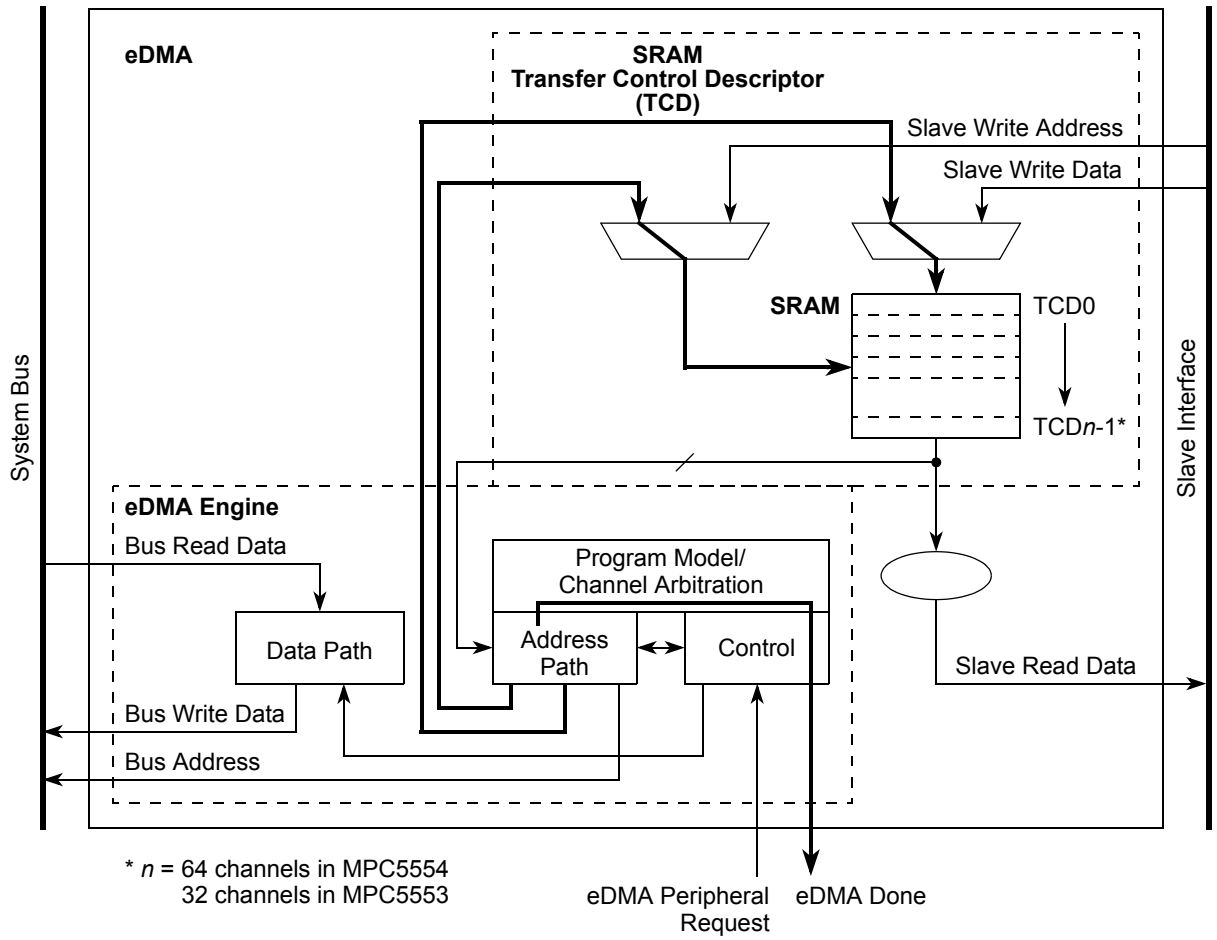


Figure 9-24. eDMA Operation, Part 3

9.4.3 eDMA Performance

This section addresses the performance of the eDMA module, focusing on two separate metrics. In the traditional data movement context, performance is best expressed as the peak data transfer rates achieved using the eDMA. In most implementations, this transfer rate is limited by the speed of the source and destination address spaces. In a second context where device-paced movement of single data values to/from peripherals is dominant, a measure of the requests that can be serviced in a fixed time is a more useful metric. In this environment, the speed of the source and destination address spaces remains important, but the microarchitecture of the eDMA also factors significantly into the resulting metric.

The peak transfer rates for several different source and destination transfers are shown in Table 9-19. The following assumptions apply to Table 9-19 and Table 9-20:

- Internal SRAM can be accessed with zero wait-states when viewed from the system bus data phase.
- All slave reads require two wait-states, and slave writes three wait-states, again viewed from the system bus data phase.
- All slave accesses are 32-bits in size.

Table 9-19 presents a peak transfer rate comparison, measured in Mbytes per second.

Table 9-19. eDMA Peak Transfer Rates (Mbytes/Sec)

System Speed, Width	Internal SRAM-to-Internal SRAM	32-Bit Slave-to-Internal SRAM	Internal SRAM-to-32-Bit Slave
66.7 MHz, 32 bit	133.3	66.7	53.3
66.7 MHz, 64 bit	266.7	66.6	53.3
83.3 MHz, 32 bit	166.7	83.3	66.7
83.3 MHz, 64 bit	333.3	83.3	66.7
100.0 MHz, 32 bit	200.0	100.0	80.0
100.0 MHz, 64 bit	400.0	100.0	80.0
133.3 MHz, 32 bit	266.7	133.3	106.7
133.3 MHz, 64 bit	533.3	133.3	106.7
150.0 MHz, 32 bit	300.0	150.0	120.0
150.0 MHz, 64 bit	600.0	150.0	120.0

Where the internal-SRAM-to-internal-SRAM transfers occur at the core's datapath width; that is, either 32- or 64-bits per access. For all transfers involving the slave bus, 32-bit transfer sizes are used. In all cases, the transfer rate includes the time to read the source plus the time to write the destination.

The second performance metric is a measure of the number of DMA requests that can be serviced in a given amount of time. For this metric, it is assumed the peripheral request causes the channel to move a single slave-mapped operand to/from internal SRAM. The same timing assumptions used in the previous example apply to this calculation. In particular, this metric also reflects the time required to activate the channel. The eDMA design supports the following hardware service request sequence:

- Cycle 1: eDMA peripheral request is asserted.
- Cycle 2: The eDMA peripheral request is registered locally in the eDMA module and qualified. (TCD.START bit initiated requests start at this point with the registering of the slave write to TCD bit 255).
- Cycle 3: Channel arbitration begins.
- Cycle 4: Channel arbitration completes. The transfer control descriptor local memory read is initiated.
- Cycle 5 - 6: The first two parts of the activated channel's TCD is read from the local memory. The memory width to the eDMA engine is 64 bits, so the entire descriptor can be accessed in four cycles.
- Cycle 7: The first system bus read cycle is initiated, as the third part of the channel's TCD is read from the local memory. Depending on the state of the crossbar switch, arbitration at the system bus may insert an additional cycle of delay here.
- Cycle 8 – *n*: The last part of the TCD is read in. This cycle represents the 1st data phase for the read, and the address phase for the destination write.

  The exact timing from this point is a function of the response times for the channel's read and write accesses. In this case of an slave read and internal SRAM write, the combined data phase time is 4 cycles. For an SRAM read and slave write, it is 5 cycles.
- Cycle *n* + 1: This cycle represents the data phase of the last destination write.
- Cycle *n* + 2: The eDMA engine completes the execution of the inner minor loop and prepares to write back the required TCDn fields into the local memory. The control/status fields at word offset

0x1C in TCDn are read. If the major loop is complete, the MAJOR.E_LINK and E_SG bits are checked and processed if enabled.

- Cycle *n* + 3: The appropriate fields in the first part of the TCDn are written back into the local memory.
- Cycle *n* + 4: The fields in the second part of the TCDn are written back into the local memory. This cycle coincides with the next channel arbitration cycle start.
- Cycle *n* + 5: The next channel to be activated performs the read of the first part of its TCD from the local memory. This is equivalent to Cycle 4 for the first channel's service request.

Assuming zero wait states on the system bus, DMA requests can be processed every 9 cycles. Assuming an average of the access times associated with slave-to-SRAM (4 cycles) and SRAM-to-slave (5 cycles), DMA requests can be processed every 11.5 cycles (4 + (4+5)/2 + 3). This is the time from Cycle 4 to Cycle "*n* + 5." The resulting peak request rate, as a function of the system frequency, is shown in Table 9-20. This metric represents millions of requests per second.

Table 9-20. eDMA Peak Request Rate (MReq/Sec)

System Frequency (MHz)	Request Rate (Zero Wait States)	Request Rate (with Wait States)
66.6	7.4	5.8
83.3	9.2	7.2
100.0	11.1	8.7
133.3	14.8	11.6
150.0	16.6	13.0

A general formula to compute the peak request rate (with overlapping requests) is:

PEAKreq = freq / [entry + (1 + read_ws) + (1 + write_ws) + exit]

where:

PEAKreq - peak request rate

freq - system frequency

entry - channel startup (4 cycles)

read_ws - wait states seen during the system bus read data phase

write_ws - wait states seen during the system bus write data phase

exit - channel shutdown (3 cycles)

For example: consider a system with the following characteristics:

- Internal SRAM can be accessed with one wait-state when viewed from the system bus data phase.
- All slave reads require two wait-states, and slave writes three wait-states, again viewed from the system bus data phase.
- System operates at 150 MHz.

For an SRAM to slave transfer,

PEAKreq = 150 MHz / [4 + (1 + 1) + (1 + 3) + 3] cycles = 11.5 Mreq/sec

For an slave to SRAM transfer,

PEAKreq = 150 MHz / [4 + (1 + 2) + (1 + 1) + 3] cycles = 12.5 Mreq/sec

Assuming an even distribution of the two transfer types, the average peak request rate would be:

PEAKreq = (11.5 Mreq/sec + 12.5 Mreq/sec) / 2 = 12.0 Mreq/sec

The minimum number of cycles to perform a single read/write, zero wait states on the system bus, from a cold start (where no channel is executing, eDMA is idle) are the following:

- 11 cycles for a software (TCD.START bit) request
- 12 cycles for a hardware (eDMA peripheral request signal) request

Two cycles account for the arbitration pipeline and one extra cycle on the hardware request resulting from the internal registering of the eDMA peripheral request signals. For the peak request rate calculations above, the arbitration and request registering is absorbed in or overlap the previous executing channel.

NOTE

When channel linking or scatter/gather is enabled, a two-cycle delay is imposed on the next channel selection and startup. This allows the link channel or the scatter/gather channel to be eligible and considered in the arbitration pool for next channel selection.

9.5 Initialization / Application Information

9.5.1 eDMA Initialization

A typical initialization of the eDMA would have the following sequence:

1. Write the EDMA_CR if a configuration other than the default is desired.
2. Write the channel priority levels into the EDMA_CPR*n* registers if a configuration other than the default is desired.
3. Enable error interrupts in the EDMA_EEIRL and/or EDMA_EEIRH registers if so desired.
4. Write the 32-byte TCD for each channel that may request service.
5. Enable any hardware service requests via the EDMA_ERQRH and/or EDMA_ERQRL registers.
6. Request channel service by either software (setting the TCD.START bit) or by hardware (slave device asserting its eDMA peripheral request signal).

Once any channel requests service, a channel is selected for execution based on the arbitration and priority levels written into the programmer's model. The eDMA engine will read the entire TCD, including the primary transfer control parameter shown in Table 9-21, for the selected channel into its internal address path module. As the TCD is being read, the first transfer is initiated on the system bus unless a configuration error is detected. Transfers from the source (as defined by the source address, TCD.SADDR) to the destination (as defined by the destination address, TCD.DADDR) continue until the specified number of bytes (TCD.NBYTES) have been transferred. When the transfer is complete, the eDMA engine's local TCD.SADDR, TCD.DADDR, and TCD.CITER are written back to the main TCD memory and any minor loop channel linking is performed, if enabled. If the major loop is exhausted, further post processing is executed: for example, interrupts, major loop channel linking, and scatter/gather operations, if enabled.

Table 9-21. TCD Primary Control and Status Fields

TCD Field Name	Description
START	Control bit to explicitly start channel when using a software initiated DMA service (Automatically cleared by hardware)
ACTIVE	Status bit indicating the channel is currently in execution
DONE	Status bit indicating major loop completion (Cleared by software when using a software initiated DMA service)
D_REQ	Control bit to disable DMA request at end of major loop completion when using a hardware-initiated DMA service
BWC	Control bits for “throttling” bandwidth control of a channel
E_SG	Control bit to enable scatter-gather feature
INT_HALF	Control bit to enable interrupt when major loop is half complete
INT_MAJ	Control bit to enable interrupt when major loop completes

Figure 9-25 shows how each DMA request initiates one minor loop transfer (iteration) without CPU intervention. DMA arbitration can occur after each minor loop, and one level of minor loop DMA preemption is allowed. The number of minor loops in a major loop is specified by the beginning iteration count (biter).

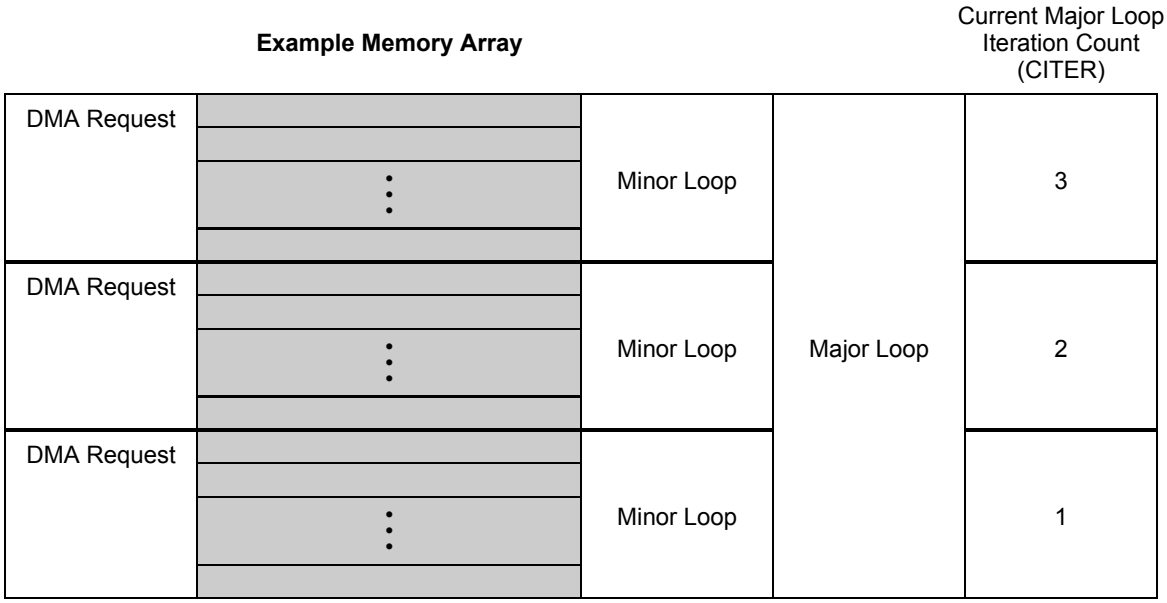


Figure 9-25. Example of Multiple Loop Iterations

Figure 9-26 lists the memory array terms and how the TCD settings interrelate.

<table>
<tr><td>xADDR:
(Starting Address)</td><td>xSIZE:
(Size of one data transfer)
•
•
•</td><td>Minor Loop (NBYTES in Minor Loop, often the same value as xSIZE)</td><td rowspan="3">Offset (xOFF): Number of bytes added to current address after each transfer (Often the same value as xSIZE)

Each DMA Source (S) and Destination (D) has its own:
• Address (xADDR)
• Size (xSIZE)
• Offset (xOFF)
• Modulo (xMOD)
• Last Address Adjustment (xLAST) where x = S or D

Peripheral queues typically have size and offset equal to NBYTES</td></tr>
<tr><td>•
•
•</td><td>•
•
•</td><td>Minor Loop</td></tr>
<tr><td>xLAST: Number of bytes added to current address after Major Loop (Typically used to loop back)</td><td>•
•
•</td><td>Last Minor Loop</td></tr>
</table>

Figure 9-26. Memory Array Terms

9.5.2 DMA Programming Errors

The eDMA performs various tests on the transfer control descriptor to verify consistency in the descriptor data. Most programming errors are reported on a per channel basis with the exception of two errors: group priority error and channel priority error, or EDMA_ESR[GPE] and EDMA_ESR[CPE], respectively.

For all error types other than group or channel priority errors, the channel number causing the error is recorded in the EDMA_ESR. If the error source is not removed before the next activation of the problem channel, the error will be detected and recorded again.

Channel priority errors are identified within a group once that group has been selected as the active group. For the example below, all of the channel priorities in group 1 are unique, but some of the channel priorities in group 0 are the same:

1. The eDMA is configured for fixed group and fixed channel arbitration modes.
2. Group 1 is the highest priority and all channels are unique in that group.
3. Group 0 is the next highest priority and has two channels with the same priority level.
4. If group 1 has any service requests, those requests will be executed.
5. Once all of group 1 requests have completed, group 0 will be the next active group.
6. If Group 0 has a service request, then an undefined channel in group 0 will be selected and a channel priority error will occur.
7. This will repeat until the all of group 0 requests have been removed or a higher priority group 1 request comes in.

In this sequence, for item 2, the eDMA acknowledge lines will assert only if the selected channel is requesting service via the eDMA peripheral request signal. If interrupts are enabled for all channels, the user will receive an error interrupt, but the channel number for the EDMA_ER and the error interrupt request line are undetermined because they reflect the ‘undefined’ channel. A group priority error is global and any request in any group will cause a group priority error.

If priority levels are not unique, the highest (channel/group) priority that has an active request will be selected, but the lowest numbered (channel/group) with that priority will be selected by arbitration and executed by the eDMA engine. The hardware service request handshake signals, error interrupts and error reporting will be associated with the selected channel.

9.5.3 DMA Request Assignments

The assignments between the DMA requests from the modules to the channels of the eDMA are shown in Table 9-22. The source column is written in C language syntax. The syntax is module_instance.register[bit].

Note that the MPC5554 has 64 channels but the MPC5553 has 32 channels, and in Table 9-22 channels 0-31 function for both the MPC5553/MPC5554, but only channels 32-63 function for the MPC5554.

Table 9-22. DMA Request Summary for eDMA

DMA Request	Channel	Source	Description
eQADC_FISR0_CFFF0	0	EQADC.FISR0[CFFF0]	eQADC Command FIFO 0 Fill Flag
eQADC_FISR0_RFDF0	1	EQADC.FISR0[RFDF0]	eQADC Receive FIFO 0 Drain Flag
eQADC_FISR1_CFFF1	2	EQADC.FISR1[CFFF1]	eQADC Command FIFO 1 Fill Flag
eQADC_FISR1_RFDF1	3	EQADC.FISR1[RFDF1]	eQADC Receive FIFO 1 Drain Flag
eQADC_FISR2_CFFF2	4	EQADC.FISR2[CFFF2]	eQADC Command FIFO 2 Fill Flag
eQADC_FISR2_RFDF2	5	EQADC.FISR2[RFDF2]	eQADC Receive FIFO 2 Drain Flag
eQADC_FISR3_CFFF3	6	EQADC.FISR3[CFFF3]	eQADC Command FIFO 3 Fill Flag
eQADC_FISR3_RFDF3	7	EQADC.FISR3[RFDF3]	eQADC Receive FIFO 3 Drain Flag
eQADC_FISR4_CFFF4	8	EQADC.FISR4[CFFF4]	eQADC Command FIFO 4 Fill Flag
eQADC_FISR4_RFDF4	9	EQADC.FISR4[RFDF4]	eQADC Receive FIFO 4 Drain Flag
eQADC_FISR5_CFFF5	10	EQADC.FISR5[CFFF5]	eQADC Command FIFO 5 Fill Flag
eQADC_FISR5_RFDF5	11	EQADC.FISR5[RFDF5]	eQADC Receive FIFO 5 Drain Flag
DSPIB_SR_TFFF	12	DSPIB.SR[TFFF]	DSPIB Transmit FIFO Fill Flag
DSPIB_SR_RFDF	13	DSPIB.SR[RFDF]	DSPIB Receive FIFO Drain Flag
DSPIC_SR_TFFF	14	DSPIC.SR[TFFF]	DSPIC Transmit FIFO Fill Flag
DSPIC_SR_RFDF	15	DSPIC.SR[RFDF]	DSPIC Receive FIFO Drain Flag
DSPID_SR_TFFF	16	DSPID.SR[TFFF]	DSPID Transmit FIFO Fill Flag
DSPID_SR_RFDF	17	DSPID.SR[RFDF]	DSPID Receive FIFO Drain Flag
eSCIA_COMBTX	18	ESCIA.SR[TDRE] || ESCIA.SR[TC] || ESCIA.SR[TXRDY]	eSCIA combined DMA request of the Transmit Data Register Empty, Transmit Complete, and LIN Transmit Data Ready DMA requests
eSCIA_COMBRX	19	ESCIA.SR[RDRF] || ESCIA.SR[RXRDY]	eSCIA combined DMA request of the Receive Data Register Full and LIN Receive Data Ready DMA requests
eMIOS_GFR_F0	20	EMIOS.GFR[F0]	eMIOS channel 0 Flag

Table 9-22. DMA Request Summary for eDMA (continued)

DMA Request	Channel	Source	Description
eMIOS_GFR_F1	21	EMIOS.GFR[F1]	eMIOS channel 1 Flag
eMIOS_GFR_F2	22	EMIOS.GFR[F2]	eMIOS channel 2 Flag
eMIOS_GFR_F3	23	EMIOS.GFR[F3]	eMIOS channel 3 Flag
eMIOS_GFR_F4	24	EMIOS.GFR[F4]	eMIOS channel 4 Flag
eMIOS_GFR_F8	25	EMIOS.GFR[F8]	eMIOS channel 8 Flag
eMIOS_GFR_F9	26	EMIOS.GFR[F9]	eMIOS channel 9 Flag
eTPU_CDTRSR_A_DTRS0	27	ETPU.CDTRSR_A[DTRS0]	eTPUA Channel 0 Data Transfer Request Status
eTPU_CDTRSR_A_DTRS1	28	ETPU.CDTRSR_A[DTRS1]	eTPUA Channel 1 Data Transfer Request Status
eTPU_CDTRSR_A_DTRS2	29	ETPU.CDTRSR_A[DTRS2]	eTPUA Channel 2 Data Transfer Request Status
eTPU_CDTRSR_A_DTRS14	30	ETPU.CDTRSR_A[DTRS14]	eTPUA Channel 14 Data Transfer Request Status
eTPU_CDTRSR_A_DTRS15	31	ETPU.CDTRSR_A[DTRS15]	eTPUA Channel 15 Data Transfer Request Status
The Below Requests Are Only Available in the MPC5554			
DSPIA_SR_TFFF	32	DSPIAISR[TFFF]	DSPIA Transmit FIFO Fill Flag
DSPIA_SR_RFDF	33	DSPIA.SR[RFDF]	DSPIA Receive FIFO Drain Flag
eSCIB_COMBTX	34	ESCIB.SR[TDRE] || ESCIB.SR[TC] || ESCIB.SR[TXRDY]	eSCIB combined DMA request of the Transmit Data Register Empty, Transmit Complete, and LIN Transmit Data Ready DMA requests
eSCIB_COMBRX	35	ESCIB.SR[RDRF] || ESCIB.SR[RXRDY]	eSCIB combined DMA request of the Receive Data Register Full and LIN Receive Data Ready DMA requests
eMIOS_GFR_F6	36	EMIOS.GFR[F6]	eMIOS channel 6 Flag
eMIOS_GFR_F7	37	EMIOS.GFR[F7]	eMIOS channel 7 Flag
eMIOS_GFR_F10	38	EMIOS.GFR[F10]	eMIOS channel 10 Flag
eMIOS_GFR_F11	39	EMIOS.GFR[F11]	eMIOS channel 11 Flag
eMIOS_GFR_F16	40	EMIOS.GFR[F16]	eMIOS channel 16 Flag
eMIOS_GFR_F17	41	EMIOS.GFR[F17]	eMIOS channel 17 Flag
eMIOS_GFR_F18	42	EMIOS.GFR[F18]	eMIOS channel 18 Flag
eMIOS_GFR_F19	43	EMIOS.GFR[F19]	eMIOS channel 19 Flag
eTPU_CDTRSR_A_DTRS12	44	ETPU.CDTRSR_A[DTRS12]	eTPUA Channel 12 Data Transfer Request Status

Table 9-22. DMA Request Summary for eDMA (continued)

DMA Request	Channel	Source	Description
eTPU_CDTRSR_A_DTRS13	45	ETPU.CDTRSR_A[DTRS13]	eTPUA Channel 13 Data Transfer Request Status
eTPU_CDTRSR_A_DTRS28	46	ETPU.CDTRSR_A[DTRS28]	eTPUA Channel 28 Data Transfer Request Status
eTPU_CDTRSR_A_DTRS29	47	ETPU.CDTRSR_A[DTRS29]	eTPUA Channel 29 Data Transfer Request Status
SIU_EISR_EIF0	48	SIU.SIU_EISR[EIF0]	SIU External Interrupt Flag 0
SIU_EISR_EIF1	49	SIU.SIU_EISR[EIF1]	SIU External Interrupt Flag 1
SIU_EISR_EIF2	50	SIU.SIU_EISR[EIF2]	SIU External Interrupt Flag 2
SIU_EISR_EIF3	51	SIU.SIU_EISR[EIF3]	SIU External Interrupt Flag 3
eTPU_CDTRSR_B_DTRS0	52	ETPU.CDTRSR_B[DTRS0]	eTPUB Channel 0 Data Transfer Request Status
eTPU_CDTRSR_B_DTRS1	53	ETPU.CDTRSR_B[DTRS1]	eTPUB Channel 1 Data Transfer Request Status
eTPU_CDTRSR_B_DTRS2	54	ETPU.CDTRSR_B[DTRS2]	eTPUB Channel 2 Data Transfer Request Status
eTPU_CDTRSR_B_DTRS3	55	ETPU.CDTRSR_B[DTRS3]	eTPUB Channel 3 Data Transfer Request Status
eTPU_CDTRSR_B_DTRS12	56	ETPU.CDTRSR_B[DTRS12]	eTPUB Channel 12 Data Transfer Request Status
eTPU_CDTRSR_B_DTRS13	57	ETPU.CDTRSR_B[DTRS13]	eTPUB Channel 13 Data Transfer Request Status
eTPU_CDTRSR_B_DTRS14	58	ETPU.CDTRSR_B[DTRS14]	eTPUB Channel 14 Data Transfer Request Status
eTPU_CDTRSR_B_DTRS15	59	ETPU.CDTRSR_B[DTRS15]	eTPUB Channel 15 Data Transfer Request Status
eTPU_CDTRSR_B_DTRS28	60	ETPU.CDTRSR_B[DTRS28]	eTPUB Channel 28 Data Transfer Request Status
eTPU_CDTRSR_B_DTRS29	61	ETPU.CDTRSR_B[DTRS29]	eTPUB Channel 29 Data Transfer Request Status
eTPU_CDTRSR_B_DTRS30	62	ETPU.CDTRSR_B[DTRS30]	eTPUB Channel 30 Data Transfer Request Status
eTPU_CDTRSR_B_DTRS31	63	ETPU.CDTRSR_B[DTRS31]	eTPUB Channel 31 Data Transfer Request Status

9.5.4 DMA Arbitration Mode Considerations

9.5.4.1 Fixed Group Arbitration, Fixed Channel Arbitration

In this mode, the channel service request from the highest priority channel in the highest priority group will be selected to execute. If the eDMA is programmed so the channels within one group use ‘fixed’ priorities, and that group is assigned the highest ‘fixed’ priority of all groups, it is possible for that group to take all the bandwidth of the eDMA controller; that is, no other groups will be serviced if there is always at least one DMA request pending on a channel in the highest priority group when the controller arbitrates the next DMA request. The advantage of this scenario is that latency can be small for channels that need to be serviced quickly. Preemption is available in this scenario only.

9.5.4.2 Round Robin Group Arbitration, Fixed Channel Arbitration

The occurrence of one or more DMA requests from one or more groups, the channel with the highest priority from a specific group will be serviced first. Groups are serviced starting with the highest group number with a service request and rotating through to the lowest group number containing a service request.

Once the channel request is serviced, the group round robin algorithm will select the highest pending request from the next group in the round robin sequence. Servicing continues round robin, always servicing the highest priority channel in the next group in the sequence, or just skipping a group if it has no pending requests.

If a channel requests service at a rate that equals or exceeds the round robin service rate, then that channel will always be serviced before lower priority channels in the same group, and thus the lower priority channels will never be serviced. The advantage of this scenario is that no one group will consume all the eDMA bandwidth. The highest priority channel selection latency is potentially greater than fixed/fixed arbitration. Excessive request rates on high priority channels could prevent the servicing of lower priority channels in the same group.

9.5.4.3 Round Robin Group Arbitration, Round Robin Channel Arbitration

Groups will be serviced as described in Section 9.5.4.2, but this time channels will be serviced in channel number order. Only one channel is serviced from each requesting group for each round robin pass through the groups.

Within each group, channels are serviced starting with the highest channel number and rotating through to the lowest channel number without regard to channel priority levels.

Because channels are serviced in round robin manner, any channel that generates DMA requests faster than a combination of the group round robin service rate and the channel service rate for its group will not prevent the servicing of other channels in its group. Any DMA requests that are not serviced are simply lost, but at least one channel will be serviced.

This scenario ensures that all channels will be guaranteed service at some point, regardless of the request rates. However, the potential latency could be quite high. All channels are treated equally. Priority levels are not used in round robin/round robin mode.

9.5.4.4 Fixed Group Arbitration, Round Robin Channel Arbitration

The highest priority group with a request will be serviced. Lower priority groups will be serviced if no pending requests exist in the higher priority groups.

Within each group, channels are serviced starting with the highest channel number and rotating through to the lowest channel number without regard to the channel priority levels assigned within the group.

This scenario could cause the same bandwidth consumption problem as indicated in Section 9.5.4.1, but all the channels in the highest priority group will get serviced. Service latency will be short on the highest priority group, but could potentially get very much longer and longer as the group priority decreases.

9.5.5 DMA Transfer

9.5.5.1 Single Request

To perform a simple transfer of '*n*' bytes of data with one activation, set the major loop to 1 (TCD.CITER = TCD.BITER = 1). The data transfer will begin after the channel service request is acknowledged and the channel is selected to execute. Once the transfer is complete, the TCD.DONE bit will be set and an interrupt will be generated if properly enabled.

For example, the following TCD entry is configured to transfer 16 bytes of data. The eDMA is programmed for one iteration of the major loop transferring 16 bytes per iteration. The source memory has a byte wide memory port located at 0x1000. The destination memory has a word wide port located at 0x2000. The address offsets are programmed in increments to match the size of the transfer; one byte for the source and four bytes for the destination. The final source and destination addresses are adjusted to return to their beginning values.

TCD.CITER = TCD.BITER = 1
TCD.NBYTES = 16
TCD.SADDR = 0x1000
TCD.SOFF = 1
TCD.SSIZE = 0
TCD.SLAST = –16
TCD.DADDR = 0x2000
TCD.DOFF = 4
TCD.DSIZE = 2
TCD.DLAST_SGA= –16
TCD.INT_MAJ = 1
TCD.START = 1 (Should be written last after all other fields have been initialized)
All other TCD fields = 0

This would generate the following sequence of events:

1. Slave write to the TCD.START bit requests channel service.
2. The channel is selected by arbitration for servicing.
3. eDMA engine writes: TCD.DONE = 0, TCD.START = 0, TCD.ACTIVE = 1.
4. eDMA engine reads: channel TCD data from local memory to internal register file.
5. The source to destination transfers are executed as follows:
   a) read_byte(0x1000), read_byte(0x1001), read_byte(0x1002), read_byte(0x1003)
   b) write_word(0x2000) -> first iteration of the minor loop
   c) read_byte(0x1004), read_byte(0x1005), read_byte(0x1006), read_byte(0x1007)

d) write_word(0x2004) -> second iteration of the minor loop
e) read_byte(0x1008), read_byte(0x1009), read_byte(0x100a), read_byte(0x100b)
f) write_word(0x2008) -> third iteration of the minor loop
g) read_byte(0x100c), read_byte(0x100d), read_byte(0x100e), read_byte(0x100f)
h) write_word(0x200c) -> last iteration of the minor loop -> major loop complete

6. eDMA engine writes: TCD.SADDR = 0x1000, TCD.DADDR = 0x2000, TCD.CITER = 1 (TCD.BITER).
7. eDMA engine writes: TCD.ACTIVE = 0, TCD.DONE = 1, EDMA_IRQR*n* = 1.
8. The channel retires.

The eDMA goes idle or services the next channel.

9.5.5.2 Multiple Requests

The next example is the same as previous with the exception of transferring 32 bytes via two hardware requests. The only fields that change are the major loop iteration count and the final address offsets. The eDMA is programmed for two iterations of the major loop transferring 16 bytes per iteration. After the channel's hardware requests are enabled in the EDMA_ERQR, channel service requests are initiated by the slave device (ERQR should be set after TCD). Note that TCD.START = 0.

TCD.CITER = TCD.BITER = 2
TCD.NBYTES = 16
TCD.SADDR = 0x1000
TCD.SOFF = 1
TCD.SSIZE = 0
TCD.SLAST = –32
TCD.DADDR = 0x2000
TCD.DOFF = 4
TCD.DSIZE = 2
TCD.DLAST_SGA= –32
TCD.INT_MAJ = 1
TCD.START = 0 (Should be written last after all other fields have been initialized)
All other TCD fields = 0

This would generate the following sequence of events:

1. First hardware (eDMA peripheral request) request for channel service.
2. The channel is selected by arbitration for servicing.
3. eDMA engine writes: TCD.DONE = 0, TCD.START = 0, TCD.ACTIVE = 1.
4. eDMA engine reads: channel TCD data from local memory to internal register file.
5. The source to destination transfers are executed as follows:
   a) read_byte(0x1000), read_byte(0x1001), read_byte(0x1002), read_byte(0x1003)
   b) write_word(0x2000) -> first iteration of the minor loop
   c) read_byte(0x1004), read_byte(0x1005), read_byte(0x1006), read_byte(0x1007)

d) write_word(0x2004) -> second iteration of the minor loop
e) read_byte(0x1008), read_byte(0x1009), read_byte(0x100a), read_byte(0x100b)
f) write_word(0x2008) -> third iteration of the minor loop
g) read_byte(0x100c), read_byte(0x100d), read_byte(0x100e), read_byte(0x100f)
h) write_word(0x200c) -> last iteration of the minor loop

6. eDMA engine writes: TCD.SADDR = 0x1010, TCD.DADDR = 0x2010, TCD.CITER = 1.
7. eDMA engine writes: TCD.ACTIVE = 0.
8. The channel retires -> one iteration of the major loop.

The eDMA goes idle or services the next channel.

9. Second hardware (eDMA peripheral request) requests channel service.
10. The channel is selected by arbitration for servicing.
11. eDMA engine writes: TCD.DONE = 0, TCD.START = 0, TCD.ACTIVE = 1.
12. eDMA engine reads: channel TCD data from local memory to internal register file.
13. The source to destination transfers are executed as follows:
    a) read_byte(0x1010), read_byte(0x1011), read_byte(0x1012), read_byte(0x1013)
    b) write_word(0x2010) -> first iteration of the minor loop
    c) read_byte(0x1014), read_byte(0x1015), read_byte(0x1016), read_byte(0x1017)
    d) write_word(0x2014) -> second iteration of the minor loop
    e) read_byte(0x1018), read_byte(0x1019), read_byte(0x101a), read_byte(0x101b)
    f) write_word(0x2018) -> third iteration of the minor loop
    g) read_byte(0x101c), read_byte(0x101d), read_byte(0x101e), read_byte(0x101f)
    h) write_word(0x201c) -> last iteration of the minor loop -> major loop complete
14. eDMA engine writes: TCD.SADDR = 0x1000, TCD.DADDR = 0x2000, TCD.CITER = 2 (TCD.BITER).
15. eDMA engine writes: TCD.ACTIVE = 0, TCD.DONE = 1, EDMA_IRQR*n* = 1.
16. The channel retires -> major loop complete.

The eDMA goes idle or services the next channel.

9.5.5.3 Modulo Feature

The modulo feature of the eDMA provides the ability to easily implement a circular data queue in which the size of the queue is a power of 2. MOD is a 5-bit bitfield for both the source and destination in the TCD, and it specifies which lower address bits are allowed to increment from their original value after the address + offset calculation. All upper address bits remain the same as in the original value. A setting of 0 for this field disables the modulo feature.

Table 9-23 shows how the transfer addresses are specified based on the setting of the MOD field. Here a circular buffer is created where the address wraps to the original value while the 28 upper address bits (0x1234567x) retain their original value. In this example the source address is set to 0x12345670, the offset is set to 4 bytes and the mod field is set to 4, allowing for a 2^4 byte (16-byte) size queue.

Table 9-23. Modulo Feature Example

Transfer Number	Address
1	0x12345670
2	0x12345674
3	0x12345678
4	0x1234567C
5	0x12345670
6	0x12345674

9.5.6 TCD Status

9.5.6.1 Minor Loop Complete

There are two methods to test for minor loop completion when using software initiated service requests. The first method is to read the TCD.CITER field and test for a change. Another method may be extracted from the sequence shown below. The second method is to test the TCD.START bit AND the TCD.ACTIVE bit. The minor loop complete condition is indicated by both bits reading zero after the TCD.START was written to a one. Polling the TCD.ACTIVE bit may be inconclusive because the active status may be missed if the channel execution is short in duration.

The TCD status bits execute the following sequence for a software activated channel:

1. TCD.START = 1, TCD.ACTIVE = 0, TCD.DONE = 0 (channel service request via software)
2. TCD.START = 0, TCD.ACTIVE = 1, TCD.DONE = 0 (channel is executing)
3. TCD.START = 0, TCD.ACTIVE = 0, TCD.DONE = 0 (channel has completed the minor loop and is idle) or
4. TCD.START = 0, TCD.ACTIVE = 0, TCD.DONE = 1 (channel has completed the major loop and is idle)

The best method to test for minor loop completion when using hardware initiated service requests is to read the TCD.CITER field and test for a change. The hardware request and acknowledge handshakes signals are not visible in the programmer's model.

The TCD status bits execute the following sequence for a hardware activated channel:

1. eDMA peripheral request asserts (channel service request via hardware)
2. TCD.START = 0, TCD.ACTIVE = 1, TCD.DONE = 0 (channel is executing)
3. TCD.START = 0, TCD.ACTIVE = 0, TCD.DONE = 0 (channel has completed the minor loop and is idle) or
4. TCD.START = 0, TCD.ACTIVE = 0, TCD.DONE = 1 (channel has completed the major loop and is idle)

For both activation types, the major loop complete status is explicitly indicated via the TCD.DONE bit.

The TCD.START bit is cleared automatically when the channel begins execution regardless of how the channel was activated.

9.5.6.2 Active Channel TCD Reads

The eDMA will read back the true TCD.SADDR, TCD.DADDR, and TCD.NBYTES values if read while a channel is executing. The true values of the SADDR, DADDR, and NBYTES are the values the eDMA engine is currently using in its internal register file and not the values in the TCD local memory for that channel. The addresses (SADDR and DADDR) and NBYTES (decrements to zero as the transfer progresses) can give an indication of the progress of the transfer. All other values are read back from the TCD local memory.

9.5.6.3 Preemption Status

Preemption is only available when fixed arbitration is selected for both group and channel arbitration modes. A preempt-able situation is one in which a preempt-enabled channel is running and a higher priority request becomes active. When the eDMA engine is not operating in fixed group, fixed channel arbitration mode, the determination of the relative priority of the actively running and the outstanding requests become undefined. Channel and/or group priorities are treated as equal (or more exactly, constantly rotating) when round-robin arbitration mode is selected.

The TCD.ACTIVE bit for the preempted channel remains asserted throughout the preemption. The preempted channel is temporarily suspended while the preempting channel executes one iteration of the major loop. Two TCD.ACTIVE bits set at the same time in the overall TCD map indicates a higher priority channel is actively preempting a lower priority channel.

9.5.7 Channel Linking

Channel linking (or chaining) is a mechanism where one channel sets the TCD.START bit of another channel (or itself) thus initiating a service request for that channel. This operation is automatically performed by the eDMA engine at the conclusion of the major or minor loop when properly enabled.

The minor loop channel linking occurs at the completion of the minor loop (or one iteration of the major loop). The TCD.CITER.E_LINK field are used to determine whether a minor loop link is requested. When enabled, the channel link is made after each iteration of the minor loop except for the last. When the major loop is exhausted, only the major loop channel link fields are used to determine if a channel link should be made. For example, with the initial fields of:

TCD.CITER.E_LINK = 1
TCD.CITER.LINKCH = 0xC
TCD.CITER value = 0x4
TCD.MAJOR.E_LINK = 1
TCD.MAJOR.LINKCH = 0x7

will execute as:

1. Minor loop done -> set channel 12 TCD.START bit
2. Minor loop done -> set channel 12 TCD.START bit
3. Minor loop done -> set channel 12 TCD.START bit
4. Minor loop done, major loop done -> set channel 7 TCD.START bit

When minor loop linking is enabled (TCD.CITER.E_LINK = 1), the TCD.CITER field uses a nine bit vector to form the current iteration count.

When minor loop linking is disabled (TCD.CITER.E_LINK = 0), the TCD.CITER field uses a 15-bit vector to form the current iteration count. The bits associated with the TCD.CITER.LINKCH field are concatenated onto the CITER value to increase the range of the CITER.

NOTE

After configuration, the TCD.CITER.E_LINK bit and the TCD.BITER.E_LINK bit must be equal or a configuration error will be reported. The CITER and BITER vector widths must be equal in order to calculate the major loop, half-way done interrupt point.

Table 9-24 summarizes how a DMA channel can "link" to another DMA channel, i.e, use another channel's TCD, at the end of a loop.

Table 9-24. Channel Linking Parameters

Desired Link Behavior	TCD Control Field Name	Description
Link at end of Minor Loop	citer.e_link	Enable channel-to-channel linking on minor loop completion (current iteration)
	citer.linkch	Link channel number when linking at end of minor loop (current iteration)
Link at end of Major Loop	major.e_link	Enable channel-to-channel linking on major loop completion
	major.linkch	Link channel number when linking at end of major loop

9.5.8 Dynamic Programming

This section provides recommended methods to change the programming model during channel execution.

9.5.8.1 Dynamic Channel Linking and Dynamic Scatter/Gather

Dynamic channel linking and dynamic scatter/gather is the process of changing the TCD.MAJOR.E_LINK or TCD.E_SG bits during channel execution. These bits are read from the TCD local memory at the *end* of channel execution thus allowing the user to enable either feature during channel execution.

Because the user is allowed to change the configuration during execution, a coherency model is needed. Consider the scenario where the user attempts to execute a dynamic channel link by enabling the TCD.MAJOR.E_LINK bit at the same time the eDMA engine is retiring the channel. The TCD.MAJOR.E_LINK would be set in the programmer's model, but it would be unclear whether the actual link was made before the channel retired.

The following coherency model is recommended when executing a dynamic channel link or dynamic scatter/gather request:

1. Set the TCD.MAJOR.E_LINK bit
2. Read back the TCD.MAJOR.E_LINK bit
3. Test the TCD.MAJOR.E_LINK request status:
   a) If the bit is set, the dynamic link attempt was successful.

b) If the bit is cleared, the attempted dynamic link did not succeed, the channel was already retiring.

This same coherency model is true for dynamic scatter/gather operations. For both dynamic requests, the TCD local memory controller forces the TCD.MAJOR.E_LINK and TCD.E_SG bits to zero on any writes to a channel's TCD once that channel's TCD.DONE bit is set indicating the major loop is complete.

NOTE

The user must clear the TCD.DONE bit before writing the TCD.MAJOR.E_LINK or TCD.E_SG bits. The TCD.DONE bit is cleared automatically by the eDMA engine once a channel begins execution.

9.6 Revision History

Substantive Changes since Rev 3.0
Changed 2 instances of EDMA_INTR to be "EDMA_ERQH or EDMA_ERQL".

Chapter 10
Interrupt Controller (INTC)

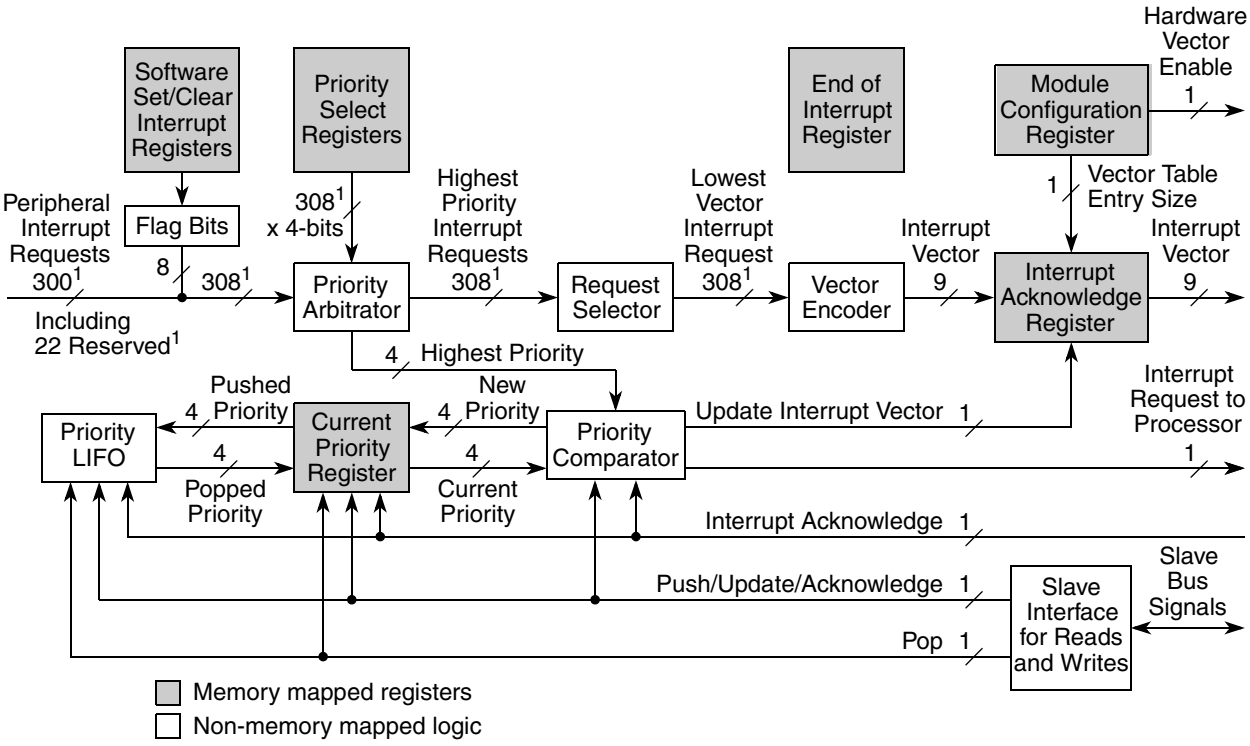
10.1 Introduction

This chapter describes the interrupt controller (INTC), which schedules interrupt requests (IRQs) from software and internal peripherals to the e200z6 core. The INTC provides interrupt prioritization and preemption, interrupt masking, interrupt priority elevation, and protocol support.

Interrupts implemented by the MPC5553 and the MPC5554 are defined in the *e200z6 PowerPC™ Core Reference Manual*, Rev 0.

10.1.1 Block Diagram

Figure 4-1 shows details of the interrupt controller.



[1] The total number of interrupt sources in the MPC5553 is 212, which includes 191 peripheral, 13 reserved sources, and 8 software sources.

Figure 10-1. INTC Block Diagram

10.1.2 Overview

Interrupt functionality for the MPC5553/MPC5554 is handled between the e200z6 core and the interrupt controller. The CPU core has 19 exception sources, each of which can interrupt the core. One exception source is from the interrupt controller (INTC). The INTC provides priority-based preemptive scheduling of interrupt requests. This scheduling scheme is suitable for statically scheduled hard real-time systems. The INTC is optimized for a large number of interrupt requests. It is targeted to work with a PowerPC book E processor and automotive powertrain applications where the ISRs nest to multiple levels.

Figure 10-2 displays the interrupt sources for the MPC5553. Figure 10-3 displays the interrupt sources for the MPC5554. Refer to Table 10-9 for interrupt source vector details.

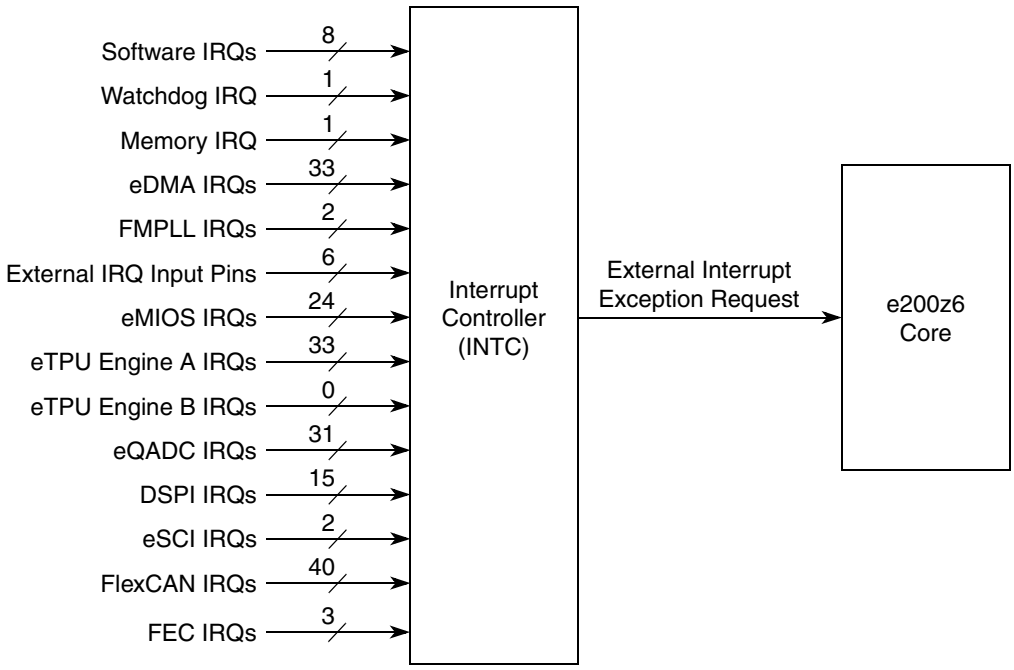


Figure 10-2. MPC5553 INTC Software Vector Mode

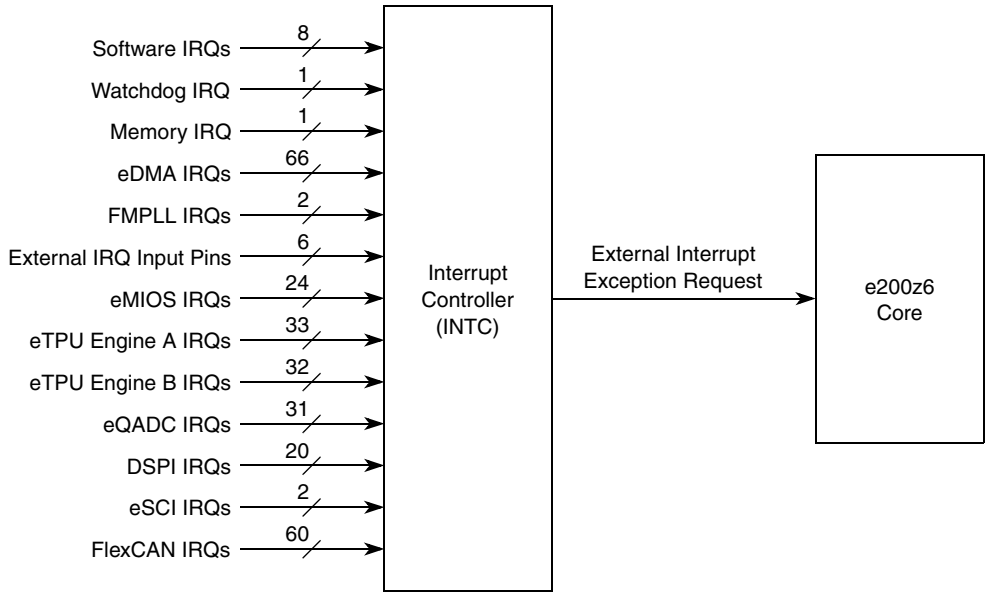


Figure 10-3. MPC5554 INTC Software Vector Mode

Two modes are available to determine the vector for the interrupt request source: software vector mode and hardware vector mode. In software vector mode, as shown in Figure 10-2, the e200z6 branches to a common interrupt exception handler whose location is determined by an address derived from special purpose registers IVPR and IVOR4. The interrupt exception handler reads the INTC_IACKR to determine

the vector of the interrupt request source. Typical program flow for software vector mode is shown in Figure 10-4.

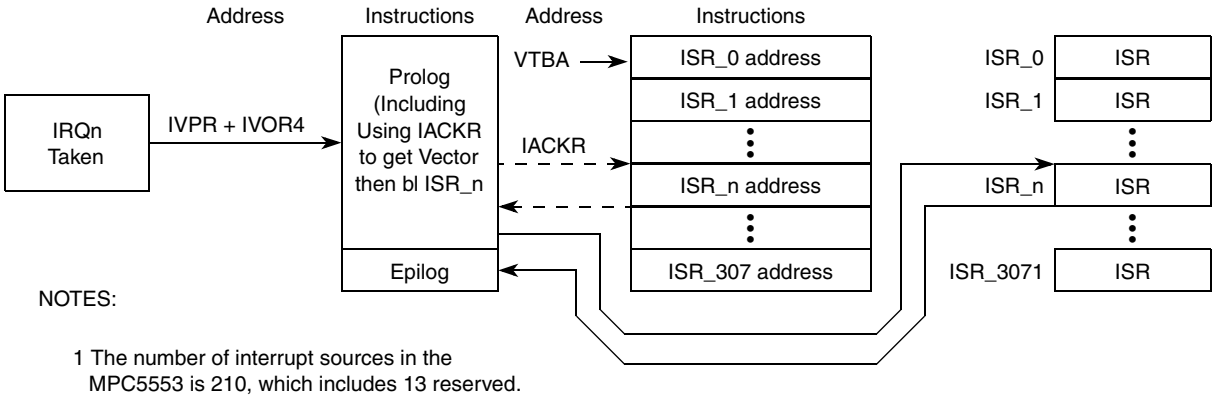


Figure 10-4. Program Flow - Software Vector Mode

In hardware vector mode, the e200z6 branches to a unique interrupt exception handler whose location is unique for each interrupt request source. Typical program flow for hardware vector mode is shown in Figure 10-5.

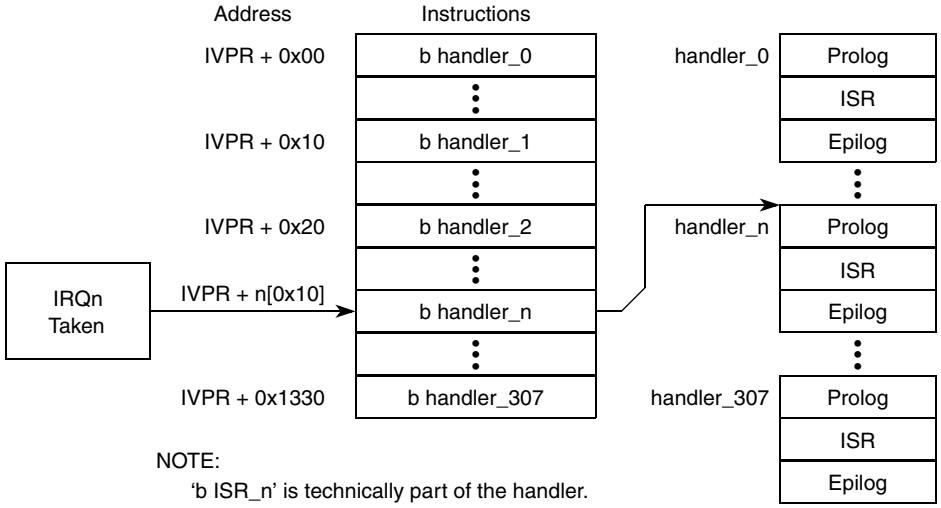


Figure 10-5. Program Flow - Hardware Vector Mode

For high priority interrupt requests in these target applications, the time from the assertion of the interrupt request from the peripheral to when the processor is performing useful work to service the interrupt request needs to be minimized. The INTC may be optimized to support this goal through the hardware vector mode, where a unique vector is provided for each interrupt request source. It also provides 16 priorities so that lower priority ISRs do not delay the execution of higher priority ISRs. Since each individual application will have different priorities for each source of interrupt request, the priority of each interrupt request is configurable.

When multiple tasks share a resource, coherent accesses to that resource need to be supported. The INTC supports the priority ceiling protocol for coherent accesses. By providing a modifiable priority mask, the priority level can be raised temporarily so that no task can preempt another task that shares the same resource.

Multiple processors can assert interrupt requests to each other through software settable interrupt requests, i.e., by using application software to assert an interrupt request. These same software settable interrupt requests also can be used to break the work involved in servicing an interrupt request into a high priority portion and a low priority portion. The high priority portion is initiated by a peripheral interrupt request, but then the ISR can assert a software settable interrupt request to finish the servicing in a lower priority ISR.

10.1.3 Features

Features include the following:

- Total number of interrupt vectors is 308 (MPC5554) or 212 (MPC5553) of which
  - 278 (MPC5554) or 191 (MPC5553) are peripheral interrupt request sources,
  - 8 are software settable sources, and
  - 22 (MPC5554) or 13 (MPC5553) are reserved sources.
- 9-bit unique vector for each interrupt request source in hardware vector mode.
- Each interrupt source can be programmed to one of 16 priorities.
- Preemption.
  - Preemptive prioritized interrupt requests to processor.
  - ISR at a higher priority preempts ISRs or tasks at lower priorities.
  - Automatic pushing or popping of preempted priority to or from a LIFO.
  - Ability to modify the ISR or task priority. Modifying the priority can be used to implement the priority ceiling protocol for accessing shared resources.
- Low latency - three clocks from receipt of interrupt request from peripheral to interrupt request to processor.

10.1.4 Modes of Operation

The interrupt controller has two handshaking modes with the processor: software vector mode and hardware vector mode. The state of the hardware vector enable bit, INTC_MCR[HVEN], determines which mode is used.

In debug mode the interrupt controller operation is identical to its normal operation of software vector mode or hardware vector mode.

10.1.4.1 Software Vector Mode

In software vector mode, there is a common interrupt exception handler address which is calculated by hardware as shown in Figure 10-6. The upper half of the interrupt vector prefix register (IVPR) is added to the offset contained in the external input interrupt vector offset register (IVOR4). Note that since bits IVOR4[28:31] are not part of the offset value, the vector offset must be located on a quad-word (16-byte) aligned location in memory.

In software vector mode, the interrupt exception handler software must read the INTC interrupt acknowledge register (INTC_IACKR) to obtain the vector associated with the corresponding peripheral or software interrupt request. The INTC_ACKR contains a 32-bit address composed of a vector table base address (VTBA) plus an offset which is the interrupt vector (INTVEC). The address is then used to branch to the corresponding routine for that peripheral or software interrupt source.

Figure 10-6. Software Vector Mode: Interrupt Exception Handler Address Calculation

Reading the INTC_IACKR acknowledges the INTC's interrupt request and negates the interrupt request to the processor. The interrupt request to the processor will not clear if a higher priority interrupt request arrives. Even in this case, INTVEC will not update to the higher priority request until the lower priority interrupt request is acknowledged by reading the INTC_IACKR. The reading also pushes the PRI value in the INTC current priority register (INTC_CPR) onto the LIFO and updates PRI in the INTC_CPR with the priority of the interrupt request. The INTC_CPR masks any peripheral or software settable interrupt request at the same or lower priority of the current value of the PRI field in INTC_CPR from generating an interrupt request to the processor.

The last actions of the interrupt exception handler must be the write to the end-of-interrupt register (INTC_EOIR). Writing to the INTC_EOIR signals the end of the servicing of the interrupt request. The INTC's LIFO is popped into the INTC_CPR's PRI field by writing to the INTC_EOIR, and the size of a write does not affect the operation of the write. Those values and sizes written to this register neither update the INTC_EOIR contents nor affect whether the LIFO pops. For possible future compatibility, write four bytes of all 0s to the INTC_EOIR. The timing relationship between popping the LIFO and disabling recognition of external input has no restriction. The writes can happen in either order.

However, disabling recognition of the external input before popping the LIFO eases the calculation of the maximum pipe depth at the cost of postponing the servicing of the next interrupt request.

10.1.4.2 Hardware Vector Mode

In hardware vector mode, the interrupt exception handler address is specific to the peripheral or software settable interrupt source rather than being common to all of them. No IVOR is used. The interrupt exception handler address is calculated by hardware as shown in Figure 10-7. The upper half of the interrupt vector prefix register (IVPR) is added to an offset which corresponds to the peripheral or software interrupt source which caused the interrupt request. The offset matches the value in the Interrupt Vector field, INTC_IACKR[INTVEC]. Each interrupt exception handler address is aligned on a quad word (16-byte) boundary. IVOR4 is unused in this mode, and software does not need to read INTC_IACKR to get the interrupt vector number.

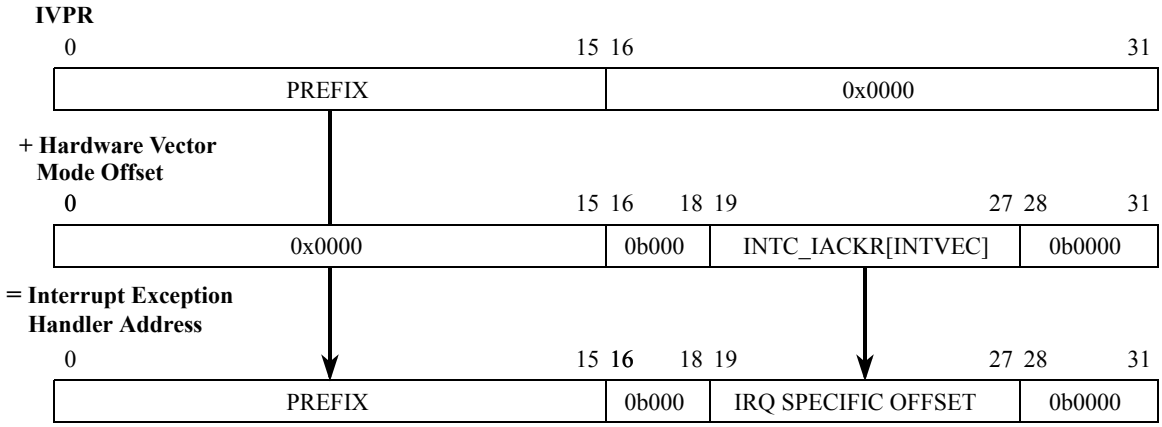


Figure 10-7. Hardware Vector Mode: Interrupt Exception Handler Address Calculation

The processor negates INTC's interrupt request when automatically acknowledging the interrupt request. However, the interrupt request to the processor will not negate if a higher priority interrupt request arrives. Even in this case, the interrupt vector number will not update to the higher priority request until the lower priority request is acknowledged by the processor.

The assertion of the interrupt acknowledge signal pushes the PRI value in the INTC_CPR onto the LIFO and updates PRI in the INTC_CPR with the new priority.

10.2 External Signal Description

The INTC does not have any direct external MCU signals. However, there are sixteen external pins which can be configured in the SIU as external interrupt request input pins. When configured in this function, an interrupt on the pin sets a corresponding SIU external interrupt flag. These flags can cause one of five peripheral interrupt requests to the interrupt controller. See Table 10-1 for a list of the external interrupt pins. See the SIU chapter for more information on these pins.

Table 10-1. External Interrupt Signals

Signal/Range Abbreviation[1]	Pin	P/A/G[2]	Function[3]	Description	I/O Type	Reset Function/ State[4]	Post Reset Function/ State[5]
EMIOS[14:15]	AF19: AD18	P A G	EMIOS[14:15] IRQ[0:1] GPIO[193:194]	eMIOS channel (output only) External interrupt request GPIO	O I I/O	— / WKPCFG	— / WKPCFG
BOOTCFG[0:1]	AA25: Y24	P A G	BOOTCFG[0:1] IRQ[2:3] GPIO[211:212]	Boot configuration input External interrupt request GPIO	I I I/O	BOOTCFG / Down	— / Down
PLLCFG0	AB25:	P A G	PLLCFG0 IRQ4 GPIO208	FMPLL mode selection External Interrupt Request GPIO	I I I/O	PLLCFG / Up	— / Up

Table 10-1. External Interrupt Signals (continued)

Signal/Range Abbreviation[1]	Pin	P/A/G[2]	Function[3]	Description	I/O Type	Reset Function/ State[4]	Post Reset Function/ State[5]
PLLCFG1	AA24	P	PLLCFG1	FMPLL mode selection	I	PLLCFG / Up	— / Up
		A	IRQ5	External Interrupt Request	I		
		A2	SOUTD	DSPI D Data Output	O		
		G	GPIO209	GPIO	I/O		
TCRCLKB[6]	M23	P	TCRCLKB	eTPU B TCR clock	I	— / Up	— / Up
		A	IRQ6	External Interrupt Request	I		
		G	GPIO146	GPIO	I/O		
TCRCLKA	N4	P	TCRCLKA	eTPU A TCR clock	I	— / Up	— / Up
		A	IRQ7	External interrupt request	I		
		G	GPIO113	GPIO	I/O		
ETPUA[20:23]	H1:G4 G2:G1	P	ETPUA[20:23]	eTPU A channel	I/O	— / WKPCFG	— / WKPCFG
		A	IRQ[8:11]	External interrupt request	I		
		G	GPIO[134:137]	GPIO	I/O		
ETPUA[24:26]	F1:G3: F3	P	ETPUA[24:26]	eTPU A channel (output only)	O	— /WKPCFG	— /WKPCFG
		A	IRQ[12:14]	External interrupt request	I		
		G	GPIO[138:140]	GPIO	I/O		
ETPUA27	F2	P	ETPUA27	eTPU A channel (output only)	O	— /WKPCFG	— /WKPCFG
		A	IRQ15	External interrupt request	I		
		G	GPIO141	GPIO	I/O		

1 This is the name that appears on the PBGA pinout.

2 Primary, alternate, or GPIO function.

3 For each pin in the table, each line in the function column is a separate function of the pin. For all MPC5554/MPC5553 I/O pins the selection of primary, secondary or tertiary function is done in the MPC5554/MPC5553 SIU except where explicitly noted.

4 Terminology is O - output, I - input, Up - weak pull up enabled, Down - weak pull down enabled, Low - output driven low, High - output driven high.

5 Function after reset of GPI is general-purpose input.

6 This signal appears only in the MPC5554, it is not implemented in the MPC5553.

10.3 Memory Map/Register Definition

Table 10-2 is the INTC memory map. INTC_BASE for the MPC5553/MPC5554 is located at 0xFFF4_8000.

Table 10-2. INTC Memory Map

Address	Register Name	Register Description	Size (bits)
Base (0xFFF4_8000)	INTC_MCR	INTC module configuration register	32
Base + 0x4	—	Reserved	—

Table 10-2. INTC Memory Map (continued)

Address	Register Name	Register Description	Size (bits)
Base + 0x8	INTC_CPR	INTC current priority register	32
Base + 0xC	—	Reserved	—
Base + 0x10	INTC_IACKR	INTC interrupt acknowledge register [1]	32
Base + 0x14	—	Reserved	—
Base + 0x18	INTC_EOIR	INTC end-of-interrupt register	32
Base + 0x1C	—	Reserved	—
Base + 0x20	INTC_SSCIR0	INTC software set/clear interrupt register 0	8
Base + 0x21	INTC_SSCIR1	INTC software set/clear interrupt register 1	8
Base + 0x22	INTC_SSCIR2	INTC software set/clear interrupt register 2	8
Base + 0x23	INTC_SSCIR3	INTC software set/clear interrupt register 3	8
Base + 0x24	INTC_SSCIR4	INTC software set/clear interrupt register 4	8
Base + 0x25	INTC_SSCIR5	INTC software set/clear interrupt register 5	8
Base + 0x26	INTC_SSCIR6	INTC software set/clear interrupt register 6	8
Base + 0x27	INTC_SSCIR7	INTC software set/clear interrupt register 7	8
Base + 0x28– Base + 0x3C	—	Reserved	—
Base + 0x40– Base + 0x173 (MPC5554) or Base + 0x110 (MPC5553)	INTC_PSR*n*	INTC priority select register 0–307 (MPC5554)[2] INTC priority select register 0–211 (MPC5553) [3]	8

1 When the HVEN bit in the INTC_MCR is asserted, a read of the INTC_IACKR has no side effects.

2 In the MPC5554, the PRI fields are reserved for peripheral interrupt requests whose vectors are 147, 148, 150, 151, 154, 175, 194–201, 282 and 301–307.

3 In the MPC5553, the PRI fields are reserved for peripheral interrupt requests whose vectors are 147, 148, 150, 151, 154, 175, 197-201, 210, 211.

10.3.1 Register Descriptions

With the exception of the INTC_SSCI*n* and INTC_PSR*n* registers, all of the registers are 32 bits in width. Any combination of accessing the 4 bytes of a register with a single access is supported, provided that the access does not cross a register boundary. These supported accesses include types and sizes of 8 bits, aligned 16 bits, and aligned 32 bits.

Although INTC_SSCI*n* and INTC_PSR*n* and 8 bits wide, they can be accessed with a single 16-bit or 32-bit access, provided that the access does not cross a 32-bit boundary.

In software vector mode, the side effects of a read of the INTC interrupt acknowledge register (INTC_IACKR) are the same regardless of the size of the read. In either software or hardware vector mode, the size of a write to the INTC end-of-interrupt register (INTC_EOIR) does not affect the operation of the write.

10.3.1.1 INTC Module Configuration Register (INTC_MCR)

The INTC_MCR is used to configure options of the INTC.

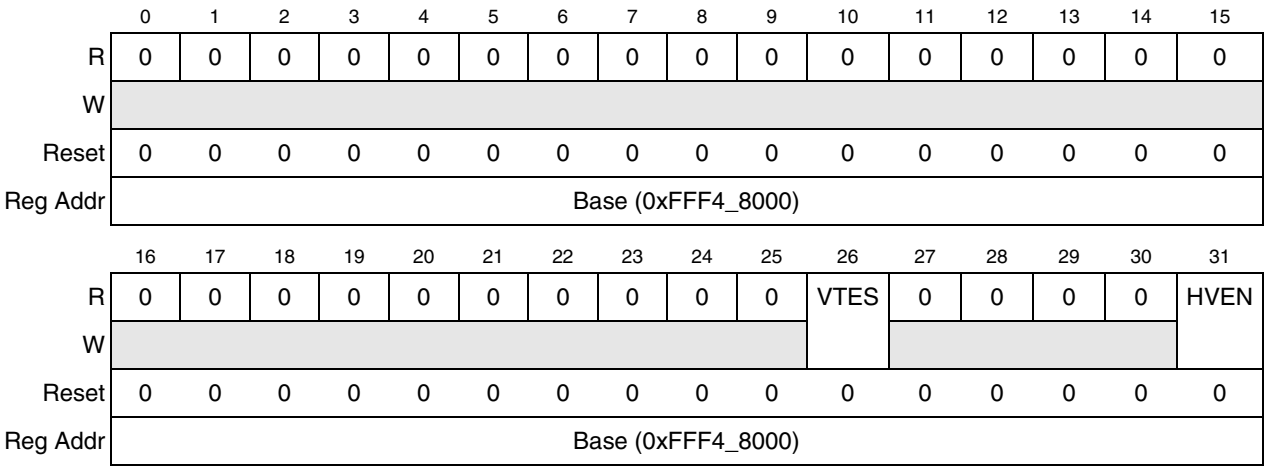


Figure 10-8. INTC Module Configuration Register (INTC_MCR)

Table 10-3. INTC_MCR Field Descriptions

Bits	Name	Description
0–25	—	Reserved.
26	VTES	Vector table entry size. Controls the number of ‘0’s to the right of INTVEC in Section 10.3.1.3, “INTC Interrupt Acknowledge Register (INTC_IACKR). If the contents of INTC_IACKR are used as an address of an entry in a vector table as in software vector mode, then the number of rightmost ‘0’s will determine the size of each vector table entry. VTES impacts software vector mode operation but also affects INTC_IACKR[INTVEC] position in both hardware vector mode and software vector mode. 0 4 bytes (Normal expected use) 1 8 bytes
27–30	—	Reserved.
31	HVEN	Hardware vector enable. Controls whether the INTC is in hardware vector mode or software vector mode. Refer to Section 10.1.4, “Modes of Operation”, for the details of the handshaking with the processor in each mode. 0 Software vector mode 1 Hardware vector mode

10.3.1.2 INTC Current Priority Register (INTC_CPR)

The INTC_CPR masks any peripheral or software settable interrupt request set at the same or lower priority as the current value of the INTC_CPR[PRI] field from generating an interrupt request to the processor. When the INTC interrupt acknowledge register (INTC_IACKR) is read in software vector mode or the interrupt acknowledge signal from the processor is asserted in hardware vector mode, the value of PRI is pushed onto the LIFO, and PRI is updated with the priority of the preempting interrupt request. When the INTC end-of-interrupt register (INTC_EOIR) is written, the LIFO is popped into the INTC_CPR’s PRI field.

The masking priority can be raised or lowered by writing to the PRI field, supporting the PCP. Refer to Section 10.5.5, "Priority Ceiling Protocol."

NOTE

On some eSys MCUs, a store to raise the PRI field which closely precedes an access to a shared resource can result in a non-coherent access to that resource unless an **mbar** or **msync** followed by an **isync** sequence of instructions is executed between the accesses. An **mbar** or **msync** instruction is also necessary after accessing the resource but before lowering the PRI field. Refer to Section 10.5.5.2, "Ensuring Coherency."

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x8															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	PRI			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
Reg Addr	Base + 0x8															

Figure 10-9. INTC Current Priority Register (INTC_CPR)

Table 10-4. INTC_CPR Field Descriptions

Bits	Name	Description
0–27	—	Reserved.
28–31	PRI [0:3]	Priority. PRI is the priority of the currently executing ISR according to the field values defined in Table 10-5.

Table 10-5. PRI Values

PRI	Meaning
1111	Priority 15 (highest priority)
1110	Priority 14
1101	Priority 13
1100	Priority 12
1011	Priority 11
1010	Priority 10
1001	Priority 9
1000	Priority 8

Table 10-5. PRI Values (continued)

PRI	Meaning
0111	Priority 7
0110	Priority 6
0101	Priority 5
0100	Priority 4
0011	Priority 3
0010	Priority 2
0001	Priority 1
0000	Priority 0 (lowest priority)

10.3.1.3 INTC Interrupt Acknowledge Register (INTC_IACKR)

The INTC_IACKR provides a value that can be used to load the address of an ISR from a vector table. The vector table can be composed of addresses of the ISRs specific to their respective interrupt vectors.

Also, in software vector mode, the INTC_IACKR has side effects from reads. The side effects are the same regardless of the size of the read. Reading the INTC_IACKR does not have side effects in hardware vector mode.

NOTE

The INTC_IACKR must not be read speculatively while in software vector mode. Therefore, for future compatibility, the TLB entry covering the INTC_IACKR must be configured to be guarded.

In software vector mode, the INTC_IACKR must be read before setting MSR[EE]. No synchronization instruction is needed after reading the INTC_IACKR and before setting MSR[EE].

However, the time for the processor to recognize the assertion or negation of the external input to it is not defined by the book E architecture and can be greater than 0. Therefore, insert instructions between the reading of the INTC_IACKR and the setting of MSR[EE] that will consume at least two processor clock cycles. This length of time will allow the negation of the interrupt request to propagate through the processor before MSR[EE] is set.

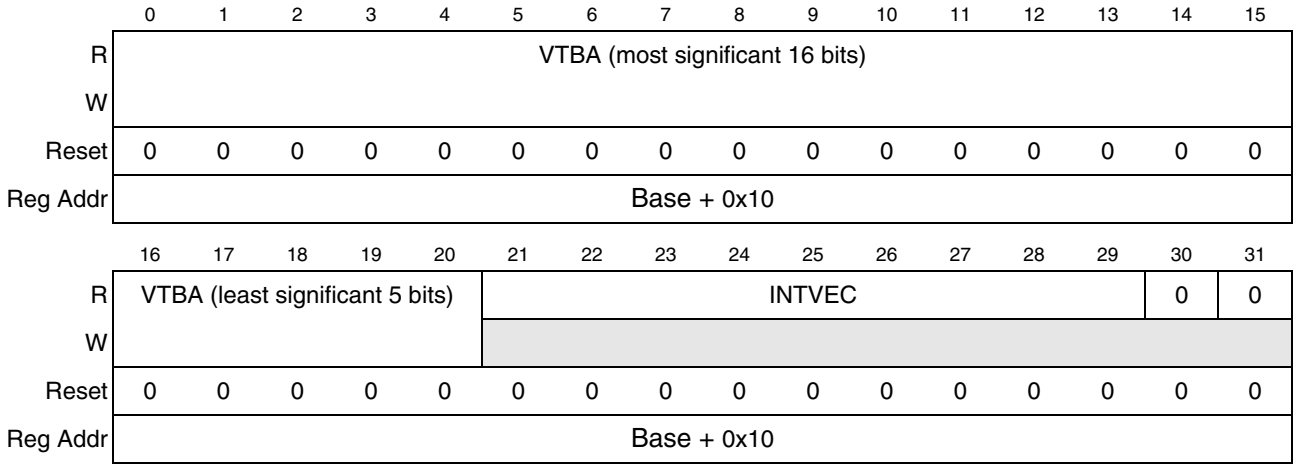


Figure 10-10. INTC Interrupt Acknowledge Register (INTC_IACKR)

Table 10-6. INTC_IACKR Field Descriptions

Bits	Name	Description
0–20	VTBA	Vector table base address. Can be the base address of a vector table of addresses of ISRs. The VTBA only uses the leftmost 20 bits when the VTES bit in INTC_MCR is asserted.
21–29	INTVEC	Interrupt vector. Vector of the peripheral or software settable interrupt request that caused the interrupt request to the processor. When the interrupt request to the processor asserts, the INTVEC is updated, whether the INTC is in software or hardware vector mode. **Note:** If INTC_MCR[VTES] = 1, then INTVEC field is shifted left one position to bits 20–28. VTBA is then shortened by one bit to bits 0–19.
30–31	—	Reserved.

10.3.1.4 INTC End-of-Interrupt Register (INTC_EOIR)

Writing to the INTC_EOIR signals the end of the servicing of the interrupt request. When the INTC_EOIR is written, the priority last pushed on the LIFO is popped into INTC_CPR. The values and size of data written to the INTC_EOIR are ignored. Those values and sizes written to this register neither update the INTC_EOIR contents or affect whether the LIFO pops. For possible future compatibility, write four bytes of all 0's to the INTC_EOIR.

Reading the INTC_EOIR has no effect on the LIFO.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x18															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x18															

Figure 10-11. INTC End-of-Interrupt Register (INTC_EOIR)

10.3.1.5 INTC Software Set/Clear Interrupt Registers (INTC_SSCIR0 – INTC_SSCIR7)

The INTC_SSCIR*n* support the setting or clearing of software settable interrupt requests. These registers contain eight independent sets of bits to set and clear a corresponding flag bit by software. With the exception of being set by software, this flag bit behaves the same as a flag bit set within a peripheral. This flag bit generates an interrupt request within the INTC just like a peripheral interrupt request. Writing a 1 to SET*n* will leave SET*n* unchanged at 0 but will set CLR*n*. Writing a 0 to SET*n* will have no effect. CLR*n* is the flag bit. Writing a 1 to CLR*n* will clear it. Writing a 0 to CLR*n* will have no effect. If a 1 is written to a pair SET*n* and CLR*n* bits at the same time, CLR*n* will be asserted, regardless of whether CLR*n* was asserted before the write.

Although INTC_SSCI*n* is 8 bits wide, it can be accessed with a single 16-bit or 32-bit access, provided that the access does not cross a 32-bit boundary.

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	CLR*n*
W							SET*n*	
Reset	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x20 + *n*							

Figure 10-12. INTC Software Set/Clear Interrupt Register 0–7 (INTC_SSCIR0–INTC_SSCIR7)

Table 10-7. INTC_SSCIR0–INTC_SSCIR7 Field Descriptions

Bits	Name	Description
0–5	—	Reserved.

Table 10-7. INTC_SSCIR0–INTC_SSCIR7 Field Descriptions (continued)

Bits	Name	Description
6	SET*n*	Set flag bits. Writing a 1 will set the corresponding CLR*n* bit. Writing a 0 will have no effect. Each SET*n* always will be read as a 0.
7	CLR*n*	Clear flag bits. CLR*n* is the flag bit. Writing a 1 to CLR*n* will clear it provided that a 1 is not written simultaneously to its corresponding SET*n* bit. Writing a 0 to CLR*n* will have no effect. 0 Interrupt request not pending within INTC. 1 Interrupt request pending within INTC.

10.3.1.6 INTC Priority Select Registers (INTC_PSR0 – INTC_PSR307)

The INTC_PSR*n* support the selection of an individual priority for each source of interrupt request. The unique vector of each peripheral or software settable interrupt request determines which INTC_PSR*n* is assigned to that interrupt request. The software settable interrupt requests 0–7 are assigned vectors 0–7, and their priorities are configured in INTC_PSR0–INTC_PSR7, respectively. The peripheral interrupt requests are assigned vectors 8–307 (MPC5554)/8–211 (MPC5553) and their priorities are configured in INTC_PSR8 through INTC_PSR307 (MPC5554) / INTC_PSR8 through INTC_PSR211 (MPC5553), respectively.

Although INTC_PSR*n* is 8 bits wide, it can be accessed with a single 16-bit or 32-bit access, provided that the access does not cross a 32-bit boundary.

NOTE

The PRI*n* field of an INTC_PSR*n* must not be modified while its corresponding peripheral or software settable interrupt request is asserted.

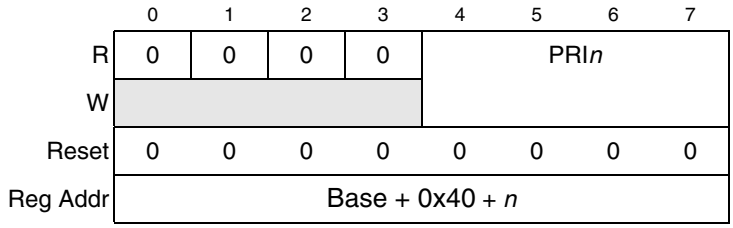


Figure 10-13. INTC Priority Select Register (INTC_PSR0–INTC_PSR307)

Table 10-8. INTC_PSR0–INTC_PSR307 Field Descriptions

Bits	Name	Description
0–3	—	Reserved.
4–7	PRI*n*	Priority select. Selects the priority for the interrupt requests. Refer to the field values in Table 10-5.

10.4 Functional Description

10.4.1 Interrupt Request Sources

The INTC has two types of interrupt requests, peripheral and software settable. The assignments between the interrupt requests from the modules to the vectors for input to the e200z6 are shown in Table 10-9. The Offset column lists the IRQ specific offsets when using hardware vector mode. The Source column is written in C language syntax. The syntax is 'module_register[bit].' Interrupt requests from the same module location or ORed together. The individual interrupt priorities are selected in INTC_PSR*n*, where the specific select register is assigned according to the vector.

Table 10-9. INTC: Interrupt Request Sources

Offset	Vector	Source[1] MPC5553	Source[1] MPC5554	Description
Software				
0x0000	0	INTC_SSCIR0[CLR0]	INTC_SSCIR0[CLR0]	INTC software settable Clear flag 0
0x0010	1	INTC_SSCIR1[CLR1]	INTC_SSCIR1[CLR1]	INTC software settable Clear flag 1
0x0020	2	INTC_SSCIR2[CLR2]	INTC_SSCIR2[CLR2]	INTC software settable Clear flag 2
0x0030	3	INTC_SSCIR3[CLR3]	INTC_SSCIR3[CLR3]	INTC software settable Clear flag 3
0x0040	4	INTC_SSCIR4[CLR4]	INTC_SSCIR4[CLR4]	INTC software settable Clear flag 4
0x0050	5	INTC_SSCIR5[CLR5]	INTC_SSCIR5[CLR5]	INTC software settable Clear flag 5
0x0060	6	INTC_SSCIR6[CLR6]	INTC_SSCIR6[CLR6]	INTC software settable Clear flag 6
0x0070	7	INTC_SSCIR7[CLR7]	INTC_SSCIR7[CLR7]	INTC software settable Clear flag 7
Watchdog / ECC				
0x0080	8	ECSM_SWTIR[SWTIC]	ECSM_SWTIR[SWTIC]	ECSM Software Watchdog Interrupt flag
0x0090	9	ECSM_ESR[RNCE] ECSM_ESR[FNCE]	ECSM_ESR[RNCE] ECSM_ESR[FNCE]	ECSM combined interrupt requests: Internal SRAM Non-Correctable Error and Flash Non-Correctable Error
eDMAC				
0x00A0	10	EDMA_ERL[ERR31:ERR0]	EDMA_ERL[ERR31:ERR0]	eDMA channel Error flags 31–0
0x00B0	11	EDMA_IRQRL[INT00]	EDMA_IRQRL[INT00]	eDMA channel Interrupt 0
0x00C0	12	EDMA_IRQRL[INT01]	EDMA_IRQRL[INT01]	eDMA channel Interrupt 1
0x00D0	13	EDMA_IRQRL[INT02]	EDMA_IRQRL[INT02]	eDMA channel Interrupt 2
0x00E0	14	EDMA_IRQRL[INT03]	EDMA_IRQRL[INT03]	eDMA channel Interrupt 3
0x00F0	15	EDMA_IRQRL[INT04]	EDMA_IRQRL[INT04]	eDMA channel Interrupt 4
0x0100	16	EDMA_IRQRL[INT05]	EDMA_IRQRL[INT05]	eDMA channel Interrupt 5
0x0110	17	EDMA_IRQRL[INT06]	EDMA_IRQRL[INT06]	eDMA channel Interrupt 6
0x0120	18	EDMA_IRQRL[INT07]	EDMA_IRQRL[INT07]	eDMA channel Interrupt 7
0x0130	19	EDMA_IRQRL[INT08]	EDMA_IRQRL[INT08]	eDMA channel Interrupt 8

Table 10-9. INTC: Interrupt Request Sources (continued)

Offset	Vector	Source[1] MPC5553	Source[1] MPC5554	Description
0x0140	20	EDMA_IRQRL[INT09]	EDMA_IRQRL[INT09]	eDMA channel Interrupt 9
0x0150	21	EDMA_IRQRL[INT10]	EDMA_IRQRL[INT10]	eDMA channel Interrupt 10
0x0160	22	EDMA_IRQRL[INT11]	EDMA_IRQRL[INT11]	eDMA channel Interrupt 11
0x0170	23	EDMA_IRQRL[INT12]	EDMA_IRQRL[INT12]	eDMA channel Interrupt 12
0x0180	24	EDMA_IRQRL[INT13]	EDMA_IRQRL[INT13]	eDMA channel Interrupt 13
0x0190	25	EDMA_IRQRL[INT14]	EDMA_IRQRL[INT14]	eDMA channel Interrupt 14
0x01A0	26	EDMA_IRQRL[INT15]	EDMA_IRQRL[INT15]	eDMA channel Interrupt 15
0x01B0	27	EDMA_IRQRL[INT16]	EDMA_IRQRL[INT16]	eDMA channel Interrupt 16
0x01C0	28	EDMA_IRQRL[INT17]	EDMA_IRQRL[INT17]	eDMA channel Interrupt 17
0x01D0	29	EDMA_IRQRL[INT18]	EDMA_IRQRL[INT18]	eDMA channel Interrupt 18
0x01E0	30	EDMA_IRQRL[INT19]	EDMA_IRQRL[INT19]	eDMA channel Interrupt 19
0x01F0	31	EDMA_IRQRL[INT20]	EDMA_IRQRL[INT20]	eDMA channel Interrupt 20
0x0200	32	EDMA_IRQRL[INT21]	EDMA_IRQRL[INT21]	eDMA channel Interrupt 21
0x0210	33	EDMA_IRQRL[INT22]	EDMA_IRQRL[INT22]	eDMA channel Interrupt 22
0x0220	34	EDMA_IRQRL[INT23]	EDMA_IRQRL[INT23]	eDMA channel Interrupt 23
0x0230	35	EDMA_IRQRL[INT24]	EDMA_IRQRL[INT24]	eDMA channel Interrupt 24
0x0240	36	EDMA_IRQRL[INT25]	EDMA_IRQRL[INT25]	eDMA channel Interrupt 25
0x0250	37	EDMA_IRQRL[INT26]	EDMA_IRQRL[INT26]	eDMA channel Interrupt 26
0x0260	38	EDMA_IRQRL[INT27]	EDMA_IRQRL[INT27]	eDMA channel Interrupt 27
0x0270	39	EDMA_IRQRL[INT28]	EDMA_IRQRL[INT28]	eDMA channel Interrupt 28
0x0280	40	EDMA_IRQRL[INT29]	EDMA_IRQRL[INT29]	eDMA channel Interrupt 29
0x0290	41	EDMA_IRQRL[INT30]	EDMA_IRQRL[INT30]	eDMA channel Interrupt 30
0x02A0	42	EDMA_IRQRL[INT31]	EDMA_IRQRL[INT31]	eDMA channel Interrupt 31
PLL				
0x02B0	43	FMPLL_SYNSR[LOCF]	FMPLL_SYNSR[LOCF]	FMPLL Loss of Clock Flag
0x02C0	44	FMPLL_SYNSR[LOLF]	FMPLL_SYNSR[LOLF]	FMPLL Loss of Lock Flag
SIU				
0x02D0	45	SIU_OSR[OVF15:OVF0]	SIU_OSR[OVF15:OVF0]	SIU combined overrun interrupt requests of the external interrupt Overrun Flags
0x02E0	46	SIU_EIISR[EIF0]	SIU_EIISR[EIF0]	SIU External Interrupt Flag 0
0x02F0	47	SIU_EIISR[EIF1]	SIU_EIISR[EIF1]	SIU External Interrupt Flag 1
0x0300	48	SIU_EIISR[EIF2]	SIU_EIISR[EIF2]	SIU External Interrupt Flag 2

Table 10-9. INTC: Interrupt Request Sources (continued)

Offset	Vector	Source[1] MPC5553	Source[1] MPC5554	Description
0x0310	49	SIU_EIISR[EIF3]	SIU_EIISR[EIF3]	SIU External Interrupt Flag 3
0x0320	50	SIU_EIISR[EIF15:EIF4]	SIU_EIISR[EIF15:EIF4]	SIU External Interrupt Flags 15–4
eMIOS				
0x0330	51	EMIOS_GFR[F0]	EMIOS_GFR[F0]	eMIOS channel 0 Flag
0x0340	52	EMIOS_GFR[F1]	EMIOS_GFR[F1]	eMIOS channel 1 Flag
0x0350	53	EMIOS_GFR[F2]	EMIOS_GFR[F2]	eMIOS channel 2 Flag
0x0360	54	EMIOS_GFR[F3]	EMIOS_GFR[F3]	eMIOS channel 3 Flag
0x0370	55	EMIOS_GFR[F4]	EMIOS_GFR[F4]	eMIOS channel 4 Flag
0x0380	56	EMIOS_GFR[F5]	EMIOS_GFR[F5]	eMIOS channel 5 Flag
0x0390	57	EMIOS_GFR[F6]	EMIOS_GFR[F6]	eMIOS channel 6 Flag
0x03A0	58	EMIOS_GFR[F7]	EMIOS_GFR[F7]	eMIOS channel 7 Flag
0x03B0	59	EMIOS_GFR[F8]	EMIOS_GFR[F8]	eMIOS channel 8 Flag
0x03C0	60	EMIOS_GFR[F9]	EMIOS_GFR[F9]	eMIOS channel 9 Flag
0x03D0	61	EMIOS_GFR[F10]	EMIOS_GFR[F10]	eMIOS channel 10 Flag
0x03E0	62	EMIOS_GFR[F11]	EMIOS_GFR[F11]	eMIOS channel 11 Flag
0x03F0	63	EMIOS_GFR[F12]	EMIOS_GFR[F12]	eMIOS channel 12 Flag
0x0400	64	EMIOS_GFR[F13]	EMIOS_GFR[F13]	eMIOS channel 13 Flag
0x0410	65	EMIOS_GFR[F14]	EMIOS_GFR[F14]	eMIOS channel 14 Flag
0x0420	66	EMIOS_GFR[F15]	EMIOS_GFR[F15]	eMIOS channel 15 Flag
eTPU_A				
0x0430	67	ETPU_MCR[MGEA] ETPU_MCR[MGEB] ETPU_MCR[ILFA] ETPU_MCR[ILFB] ETPU_MCR[SCMMISF]	ETPU_MCR[MGEA] ETPU_MCR[MGEB] ETPU_MCR[ILFA] ETPU_MCR[ILFB] ETPU_MCR[SCMMISF]	eTPU Global Exception
0x0440	68	ETPU_CISR_A[CIS0]	ETPU_CISR_A[CIS0]	eTPU Engine A Channel 0 Interrupt Status
0x0450	69	ETPU_CISR_A[CIS1]	ETPU_CISR_A[CIS1]	eTPU Engine A Channel 1 Interrupt Status
0x0460	70	ETPU_CISR_A[CIS2]	ETPU_CISR_A[CIS2]	eTPU Engine A Channel 2 Interrupt Status
0x0470	71	ETPU_CISR_A[CIS3]	ETPU_CISR_A[CIS3]	eTPU Engine A Channel 3 Interrupt Status
0x0480	72	ETPU_CISR_A[CIS4]	ETPU_CISR_A[CIS4]	eTPU Engine A Channel 4 Interrupt Status
0x0490	73	ETPU_CISR_A[CIS5]	ETPU_CISR_A[CIS5]	eTPU Engine A Channel 5 Interrupt Status
0x04A0	74	ETPU_CISR_A[CIS6]	ETPU_CISR_A[CIS6]	eTPU Engine A Channel 6 Interrupt Status
0x04B0	75	ETPU_CISR_A[CIS7]	ETPU_CISR_A[CIS7]	eTPU Engine A Channel 7 Interrupt Status
0x04C0	76	ETPU_CISR_A[CIS8]	ETPU_CISR_A[CIS8]	eTPU Engine A Channel 8 Interrupt Status

Table 10-9. INTC: Interrupt Request Sources (continued)

Offset	Vector	Source[1] MPC5553	Source[1] MPC5554	Description
0x04D0	77	ETPU_CISR_A[CIS9]	ETPU_CISR_A[CIS9]	eTPU Engine A Channel 9 Interrupt Status
0x04E0	78	ETPU_CISR_A[CIS10]	ETPU_CISR_A[CIS10]	eTPU Engine A Channel 10 Interrupt Status
0x04F0	79	ETPU_CISR_A[CIS11]	ETPU_CISR_A[CIS11]	eTPU Engine A Channel 11 Interrupt Status
0x0500	80	ETPU_CISR_A[CIS12]	ETPU_CISR_A[CIS12]	eTPU Engine A Channel 12 Interrupt Status
0x0510	81	ETPU_CISR_A[CIS13]	ETPU_CISR_A[CIS13]	eTPU Engine A Channel 13 Interrupt Status
0x0520	82	ETPU_CISR_A[CIS14]	ETPU_CISR_A[CIS14]	eTPU Engine A Channel 14 Interrupt Status
0x0530	83	ETPU_CISR_A[CIS15]	ETPU_CISR_A[CIS15]	eTPU Engine A Channel 15 Interrupt Status
0x0540	84	ETPU_CISR_A[CIS16]	ETPU_CISR_A[CIS16]	eTPU Engine A Channel 16 Interrupt Status
0x0550	85	ETPU_CISR_A[CIS17]	ETPU_CISR_A[CIS17]	eTPU Engine A Channel 17 Interrupt Status
0x0560	86	ETPU_CISR_A[CIS18]	ETPU_CISR_A[CIS18]	eTPU Engine A Channel 18 Interrupt Status
0x0570	87	ETPU_CISR_A[CIS19]	ETPU_CISR_A[CIS19]	eTPU Engine A Channel 19 Interrupt Status
0x0580	88	ETPU_CISR_A[CIS20]	ETPU_CISR_A[CIS20]	eTPU Engine A Channel 20 Interrupt Status
0x0590	89	ETPU_CISR_A[CIS21]	ETPU_CISR_A[CIS21]	eTPU Engine A Channel 21 Interrupt Status
0x05A0	90	ETPU_CISR_A[CIS22]	ETPU_CISR_A[CIS22]	eTPU Engine A Channel 22 Interrupt Status
0x05B0	91	ETPU_CISR_A[CIS23]	ETPU_CISR_A[CIS23]	eTPU Engine A Channel 23 Interrupt Status
0x05C0	92	ETPU_CISR_A[CIS24]	ETPU_CISR_A[CIS24]	eTPU Engine A Channel 24 Interrupt Status
0x05D0	93	ETPU_CISR_A[CIS25]	ETPU_CISR_A[CIS25]	eTPU Engine A Channel 25 Interrupt Status
0x05E0	94	ETPU_CISR_A[CIS26]	ETPU_CISR_A[CIS26]	eTPU Engine A Channel 26 Interrupt Status
0x05F0	95	ETPU_CISR_A[CIS27]	ETPU_CISR_A[CIS27]	eTPU Engine A Channel 27 Interrupt Status
0x0600	96	ETPU_CISR_A[CIS28]	ETPU_CISR_A[CIS28]	eTPU Engine A Channel 28 Interrupt Status
0x0610	97	ETPU_CISR_A[CIS29]	ETPU_CISR_A[CIS29]	eTPU Engine A Channel 29 Interrupt Status
0x0620	98	ETPU_CISR_A[CIS30]	ETPU_CISR_A[CIS30]	eTPU Engine A Channel 30 Interrupt Status
0x0630	99	ETPU_CISR_A[CIS31]	ETPU_CISR_A[CIS31]	eTPU Engine A Channel 31 Interrupt Status
eQADC				
0x0640	100	EQADC_FISR*x*[TORF] EQADC_FISR*x*[RFOF] EQADC_FISR*x*[CFUF]	EQADC_FISR*x*[TORF] EQADC_FISR*x*[RFOF] EQADC_FISR*x*[CFUF]	eQADC combined overrun interrupt request s from all of the FIFOs: Trigger Overrun, Receive FIFO Overflow, and command FIFO Underflow
0x0650	101	EQADC_FISR0[NCF]	EQADC_FISR0[NCF]	eQADC command FIFO 0 Non-Coherency Flag
0x0660	102	EQADC_FISR0[PF]	EQADC_FISR0[PF]	eQADC command FIFO 0 Pause Flag
0x0670	103	EQADC_FISR0[EOQF]	EQADC_FISR0[EOQF]	eQADC command FIFO 0 command queue End of Queue Flag
0x0680	104	EQADC_FISR0[CFFF]	EQADC_FISR0[CFFF]	eQADC Command FIFO 0 Fill Flag
0x0690	105	EQADC_FISR0[RFDF]	EQADC_FISR0[RFDF]	eQADC Receive FIFO 0 Drain Flag

Table 10-9. INTC: Interrupt Request Sources (continued)

Offset	Vector	Source[1] MPC5553	Source[1] MPC5554	Description
0x06A0	106	EQADC_FISR1[NCF]	EQADC_FISR1[NCF]	eQADC command FIFO 1 Non-Coherency Flag
0x06B0	107	EQADC_FISR1[PF]	EQADC_FISR1[PF]	eQADC command FIFO 1 Pause Flag
0x06C0	108	EQADC_FISR1[EOQF]	EQADC_FISR1[EOQF]	eQADC command FIFO 1 command queue End of Queue Flag
0x06D0	109	EQADC_FISR1[CFFF]	EQADC_FISR1[CFFF]	eQADC Command FIFO 1 Fill Flag
0x06E0	110	EQADC_FISR1[RFDF]	EQADC_FISR1[RFDF]	eQADC Receive FIFO 1 Drain Flag
0x06F0	111	EQADC_FISR2[NCF]	EQADC_FISR2[NCF]	eQADC command FIFO 2 Non-Coherency Flag
0x0700	112	EQADC_FISR2[PF]	EQADC_FISR2[PF]	eQADC command FIFO 2 Pause Flag
0x0710	113	EQADC_FISR2[EOQF]	EQADC_FISR2[EOQF]	eQADC command FIFO 2 command queue End of Queue Flag
0x0720	114	EQADC_FISR2[CFFF]	EQADC_FISR2[CFFF]	eQADC Command FIFO 2 Fill Flag
0x0730	115	EQADC_FISR2[RFDF]	EQADC_FISR2[RFDF]	eQADC Receive FIFO 2 Drain Flag
0x0740	116	EQADC_FISR3[NCF]	EQADC_FISR3[NCF]	eQADC command FIFO 3 Non-Coherency Flag
0x0750	117	EQADC_FISR3[PF]	EQADC_FISR3[PF]	eQADC command FIFO 3 Pause Flag
0x0760	118	EQADC_FISR3[EOQF]	EQADC_FISR3[EOQF]	eQADC command FIFO 3 command queue End of Queue Flag
0x0770	119	EQADC_FISR3[CFFF]	EQADC_FISR3[CFFF]	eQADC Command FIFO 3 Fill Flag
0x0780	120	EQADC_FISR3[RFDF]	EQADC_FISR3[RFDF]	eQADC Receive FIFO 3 Drain Flag
0x0790	121	EQADC_FISR4[NCF]	EQADC_FISR4[NCF]	eQADC command FIFO 4 Non-Coherency Flag
0x07A0	122	EQADC_FISR4[PF]	EQADC_FISR4[PF]	eQADC command FIFO 4 Pause Flag
0x07B0	123	EQADC_FISR4[EOQF]	EQADC_FISR4[EOQF]	eQADC command FIFO 4 command queue End of Queue Flag
0x07C0	124	EQADC_FISR4[CFFF]	EQADC_FISR4[CFFF]	eQADC Command FIFO 4 Fill Flag
0x07D0	125	EQADC_FISR4[RFDF]	EQADC_FISR4[RFDF]	eQADC Receive FIFO 4 Drain Flag
0x07E0	126	EQADC_FISR5[NCF]	EQADC_FISR5[NCF]	eQADC command FIFO 5 Non-Coherency Flag
0x07F0	127	EQADC_FISR5[PF]	EQADC_FISR5[PF]	eQADC command FIFO 5 Pause Flag
0x0800	128	EQADC_FISR5[EOQF]	EQADC_FISR5[EOQF]	eQADC command FIFO 5 command queue End of Queue Flag
0x0810	129	EQADC_FISR5[CFFF]	EQADC_FISR5[CFFF]	eQADC Command FIFO 5 Fill Flag
0x0820	130	EQADC_FISR5[RFDF]	EQADC_FISR5[RFDF]	eQADC Receive FIFO 5 Drain Flag
DSPI				
0x0830	131	DSPI_BSR[TFUF] DSPI_BSR[RFOF]	DSPI_BSR[TFUF] DSPI_BSR[RFOF]	DSPI_B combined overrun interrupt requests: Transmit FIFO Underflow and Receive FIFO Overflow
0x0840	132	DSPI_BSR[EOQF]	DSPI_BSR[EOQF]	DSPI_B transmit FIFO End of Queue Flag

Table 10-9. INTC: Interrupt Request Sources (continued)

Offset	Vector	Source[1] MPC5553	Source[1] MPC5554	Description
0x0850	133	DSPI_BSR[TFFF]	DSPI_BSR[TFFF]	DSPI_B Transmit FIFO Fill Flag
0x0860	134	DSPI_BSR[TCF]	DSPI_BSR[TCF]	DSPI_B Transfer Complete Flag
0x0870	135	DSPI_BSR[RFDF]	DSPI_BSR[RFDF]	DSPI_B Receive FIFO Drain Flag
0x0880	136	DSPI_CSR[TFUF] DSPI_CSR[RFOF]	DSPI_CSR[TFUF] DSPI_CSR[RFOF]	DSPI_C combined overrun interrupt requests: Transmit FIFO Underflow and Receive FIFO Overflow
0x0890	137	DSPI_CSR[EOQF]	DSPI_CSR[EOQF]	DSPI_C transmit FIFO End of Queue Flag
0x08A0	138	DSPI_CSR[TFFF]	DSPI_CSR[TFFF]	DSPI_C Transmit FIFO Fill Flag
0x08B0	139	DSPI_CSR[TCF]	DSPI_CSR[TCF]	DSPI_C Transfer Complete Flag
0x08C0	140	DSPI_CSR[RFDF]	DSPI_CSR[RFDF]	DSPI_C Receive FIFO Drain Flag
0x08D0	141	DSPI_DSR[TFUF] DSPI_DSR[RFOF]	DSPI_DSR[TFUF] DSPI_DSR[RFOF]	DSPI_D combined overrun interrupt requests: Transmit FIFO Underflow and Receive FIFO Overflow
0x08E0	142	DSPI_DSR[EOQF]	DSPI_DSR[EOQF]	DSPI_D transmit FIFO End of Queue Flag
0x08F0	143	DSPI_DSR[TFFF]	DSPI_DSR[TFFF]	DSPI_D Transmit FIFO Fill Flag
0x0900	144	DSPI_DSR[TCF]	DSPI_DSR[TCF]	DSPI_D Transfer Complete Flag
0x0910	145	DSPI_DSR[RFDF]	DSPI_DSR[RFDF]	DSPI_D Receive FIFO Drain Flag
eSCI				
0x0920	146	ESCIA_SR[TDRE] ESCIA_SR[TC] ESCIA_SR[RDRF] ESCIA_SR[IDLE] ESCIA_SR[OR] ESCIA_SR[NF] ESCIA_SR[FE] ESCIA_SR[PF] ESCIA_SR[BERR] ESCIA_SR[RXRDY] ESCIA_SR[TXRDY] ESCIA_SR[LWAKE] ESCIA_SR[STO] ESCIA_SR[PBERR] ESCIA_SR[CERR] ESCIA_SR[CKERR] ESCIA_SR[FRC] ESCIA_SR[OVFL]	ESCIA_SR[TDRE] ESCIA_SR[TC] ESCIA_SR[RDRF] ESCIA_SR[IDLE] ESCIA_SR[OR] ESCIA_SR[NF] ESCIA_SR[FE] ESCIA_SR[PF] ESCIA_SR[BERR] ESCIA_SR[RXRDY] ESCIA_SR[TXRDY] ESCIA_SR[LWAKE] ESCIA_SR[STO] ESCIA_SR[PBERR] ESCIA_SR[CERR] ESCIA_SR[CKERR] ESCIA_SR[FRC] ESCIA_SR[OVFL]	Combined Interrupt Requests of ESCI Module A: Transmit Data Register Empty, Transmit Complete, Receive Data Register Full, Idle line, Overrun, Noise Flag, Framing Error Flag, and Parity Error Flag interrupt requests, SCI Status Register 2 Bit Error interrupt request, LIN Status Register 1 Receive Data Ready, Transmit Data Ready, Received LIN Wakeup Signal, Slave TimeOut, Physical Bus Error, CRC Error, Checksum Error, Frame Complete interrupts requests, and LIN Status Register 2 Receive Register Overflow
0x0930	147	Reserved	Reserved	Reserved
0x0940	148	Reserved	Reserved	Reserved

Table 10-9. INTC: Interrupt Request Sources (continued)

Offset	Vector	Source[1] MPC5553	Source[1] MPC5554	Description
0x0950	149	ESCIB_SR[TDRE] ESCIB_SR[TC] ESCIB_SR[RDRF] ESCIB_SR[IDLE] ESCIB_SR[OR] ESCIB_SR[NF] ESCIB_SR[FE] ESCIB_SR[PF] ESCIB_SR[BERR] ESCIB_SR[RXRDY] ESCIB_SR[TXRDY] ESCIB_SR[LWAKE] ESCIB_SR[STO] ESCIB_SR[PBERR] ESCIB_SR[CERR] ESCIB_SR[CKERR] ESCIB_SR[FRC] ESCIB_SR[OVFL]	ESCIB_SR[TDRE] ESCIB_SR[TC] ESCIB_SR[RDRF] ESCIB_SR[IDLE] ESCIB_SR[OR] ESCIB_SR[NF] ESCIB_SR[FE] ESCIB_SR[PF] ESCIB_SR[BERR] ESCIB_SR[RXRDY] ESCIB_SR[TXRDY] ESCIB_SR[LWAKE] ESCIB_SR[STO] ESCIB_SR[PBERR] ESCIB_SR[CERR] ESCIB_SR[CKERR] ESCIB_SR[FRC] ESCIB_SR[OVFL]	Combined Interrupt Requests of ESCI Module B: Transmit Data Register Empty, Transmit Complete, Receive Data Register Full, Idle line, Overrun, Noise Flag, Framing Error Flag, and Parity Error Flag interrupt requests, SCI Status Register 2 Bit Error interrupt request, LIN Status Register 1 Receive Data Ready, Transmit Data Ready, Received LIN Wakeup Signal, Slave TimeOut, Physical Bus Error, CRC Error, Checksum Error, Frame Complete interrupts requests, and LIN Status Register 2 Receive Register Overflow
0x0960	150	Reserved	Reserved	Reserved
0x0970	151	Reserved	Reserved	Reserved
FlexCAN_A and FlexCAN_B				
0x0980	152	CANA_ESR[BOFF_INT]	CANA_ESR[BOFF_INT]	FLEXCAN_A Bus off Interrupt
0x0990	153	CANA_ESR[ERR_INT]	CANA_ESR[ERR_INT]	FLEXCAN_A Error Interrupt
0x09A0	154	Reserved	Reserved	Reserved
0x09B0	155	CANA_IFRL[BUF0]	CANA_IFRL[BUF0]	FLEXCAN_A Buffer 0 Interrupt
0x09C0	156	CANA_IFRL[BUF1]	CANA_IFRL[BUF1]	FLEXCAN_A Buffer 1 Interrupt
0x09D0	157	CANA_IFRL[BUF2]	CANA_IFRL[BUF2]	FLEXCAN_A Buffer 2 Interrupt
0x09E0	158	CANA_IFRL[BUF3]	CANA_IFRL[BUF3]	FLEXCAN_A Buffer 3 Interrupt
0x09F0	159	CANA_IFRL[BUF4]	CANA_IFRL[BUF4]	FLEXCAN_A Buffer 4 Interrupt
0x0A00	160	CANA_IFRL[BUF5]	CANA_IFRL[BUF5]	FLEXCAN_A Buffer 5 Interrupt
0x0A10	161	CANA_IFRL[BUF6]	CANA_IFRL[BUF6]	FLEXCAN_A Buffer 6 Interrupt
0x0A20	162	CANA_IFRL[BUF7]	CANA_IFRL[BUF7]	FLEXCAN_A Buffer 7 Interrupt
0x0A30	163	CANA_IFRL[BUF8]	CANA_IFRL[BUF8]	FLEXCAN_A Buffer 8 Interrupt
0x0A40	164	CANA_IFRL[BUF9]	CANA_IFRL[BUF9]	FLEXCAN_A Buffer 9 Interrupt
0x0A50	165	CANA_IFRL[BUF10]	CANA_IFRL[BUF10]	FLEXCAN_A Buffer 10 Interrupt
0x0A60	166	CANA_IFRL[BUF11]	CANA_IFRL[BUF11]	FLEXCAN_A Buffer 11 Interrupt
0x0A70	167	CANA_IFRL[BUF12]	CANA_IFRL[BUF12]	FLEXCAN_A Buffer 12 Interrupt
0x0A80	168	CANA_IFRL[BUF13]	CANA_IFRL[BUF13]	FLEXCAN_A Buffer 13 Interrupt
0x0A90	169	CANA_IFRL[BUF14]	CANA_IFRL[BUF14]	FLEXCAN_A Buffer 14 Interrupt

Table 10-9. INTC: Interrupt Request Sources (continued)

Offset	Vector	Source[1] MPC5553	Source[1] MPC5554	Description
0x0AA0	170	CANA_IFRL[BUF15]	CANA_IFRL[BUF15]	FLEXCAN_A Buffer 15 Interrupt
0x0AB0	171	CANA_IFRL[BUF31I:BUF16]	CANA_IFRL[BUF31I:BUF16]	FLEXCAN_A Buffers 31 - 16 Interrupts
0x0AC0	172	CANA_IFRH[BUF63I:BUF32]	CANA_IFRH[BUF63I:BUF32]	FLEXCAN_A Buffers 63 - 32 Interrupts
0x0AD0	173	CANC_ESR[BOFF_INT]	CANC_ESR[BOFF_INT]	FLEXCAN_C Bus off Interrupt
0x0AE0	174	CANC_ESR[ERR_INT]	CANC_ESR[ERR_INT]	FLEXCAN_C Error Interrupt
0x0AF0	175	Reserved	Reserved	Reserved
0x0B00	176	CANC_IFRL[BUF0]	CANC_IFRL[BUF0]	FLEXCAN_C Buffer 0 Interrupt
0x0B10	177	CANC_IFRL[BUF1]	CANC_IFRL[BUF1]	FLEXCAN_C Buffer 1 Interrupt
0x0B20	178	CANC_IFRL[BUF2]	CANC_IFRL[BUF2]	FLEXCAN_C Buffer 2 Interrupt
0x0B30	179	CANC_IFRL[BUF3]	CANC_IFRL[BUF3]	FLEXCAN_C Buffer 3 Interrupt
0x0B40	180	CANC_IFRL[BUF4]	CANC_IFRL[BUF4]	FLEXCAN_C Buffer 4 Interrupt
0x0B50	181	CANC_IFRL[BUF5]	CANC_IFRL[BUF5]	FLEXCAN_C Buffer 5 Interrupt
0x0B60	182	CANC_IFRL[BUF6]	CANC_IFRL[BUF6]	FLEXCAN_C Buffer 6 Interrupt
0x0B70	183	CANC_IFRL[BUF7]	CANC_IFRL[BUF7]	FLEXCAN_C Buffer 7 Interrupt
0x0B80	184	CANC_IFRL[BUF8]	CANC_IFRL[BUF8]	FLEXCAN_C Buffer 8 Interrupt
0x0B90	185	CANC_IFRL[BUF9]	CANC_IFRL[BUF9]	FLEXCAN_C Buffer 9 Interrupt
0x0BA0	186	CANC_IFRL[BUF10]	CANC_IFRL[BUF10]	FLEXCAN_C Buffer 10 Interrupt
0x0BB0	187	CANC_IFRL[BUF11]	CANC_IFRL[BUF11]	FLEXCAN_C Buffer 11 Interrupt
0x0BC0	188	CANC_IFRL[BUF12]	CANC_IFRL[BUF12]	FLEXCAN_C Buffer 12 Interrupt
0x0BD0	189	CANC_IFRL[BUF13]	CANC_IFRL[BUF13]	FLEXCAN_C Buffer 13 Interrupt
0x0BE0	190	CANC_IFRL[BUF14]	CANC_IFRL[BUF14]	FLEXCAN_C Buffer 14 Interrupt
0x0BF0	191	CANC_IFRL[BUF15]	CANC_IFRL[BUF15]	FLEXCAN_C Buffer 15 Interrupt
0x0C00	192	CANC_IFRL[BUF31:BUF16]	CANC_IFRL[BUF31:BUF16]	FLEXCAN_C Buffers 31 - 16 Interrupts
0x0C10	193	CANC_IFRH[BUF63:BUF32]	CANC_IFRH[BUF63:BUF32]	FLEXCAN_C Buffers 63 - 32 Interrupts
FEC				
0x0C20	194	EIR[TXF]	Reserved	FEC Transmit Frame flag
0x0C30	195	EIR[RXF]	Reserved	FEC Receive Frame flag

Table 10-9. INTC: Interrupt Request Sources (continued)

Offset	Vector	Source[1] MPC5553	Source[1] MPC5554	Description
0x0C40	196	EIR[HBERR] EIR[BABR] EIR[BABT] EIR[GRA] EIR[TXB] EIR[RXB] EIR[MII] EIR[EBERR] EIR[LC] EIR[RL] EIR[UN]	Reserved	Combined Interrupt Requests of the FEC Ethernet Interrupt Event Register: Heartbeat Error, Babbling Receive Error, Babbling Transmit Error, Graceful Stop Complete, Transmit Buffer, Receive Buffer, Media Independent Interface, Ethernet Bus Error, Late Collision, Collision Retry Limit, and Transmit FIFO Underrun
0x0C50	197	Reserved	Reserved	Reserved
0x0C60	198	Reserved	Reserved	Reserved
0x0C70	199	Reserved	Reserved	Reserved
0x0C80	200	Reserved	Reserved	Reserved
0x0C90	201	Reserved	Reserved	Reserved
eMIOS				
0x0CA0	202	EMIOS_GFR[F16]	EMIOS_GFR[F16]	eMIOS channel 16 Flag
0x0CB0	203	EMIOS_GFR[F17]	EMIOS_GFR[F17]	eMIOS channel 17 Flag
0x0CC0	204	EMIOS_GFR[F18]	EMIOS_GFR[F18]	eMIOS channel 18 Flag
0x0CD0	205	EMIOS_GFR[F19]	EMIOS_GFR[F19]	eMIOS channel 19 Flag
0x0CE0	206	EMIOS_GFR[F20]	EMIOS_GFR[F20]	eMIOS channel 20 Flag
0x0CF0	207	EMIOS_GFR[F21]	EMIOS_GFR[F21]	eMIOS channel 21 Flag
0x0D00	208	EMIOS_GFR[F22]	EMIOS_GFR[F22]	eMIOS channel 22 Flag
0x0D10	209	EMIOS_GFR[F23]	EMIOS_GFR[F23]	eMIOS channel 23 Flag
eDMA				
0x0D20	210	Reserved	EDMA_ERRH[ERR63:ERR32]	eDMA channel Error flags 63 - 32
0x0D30	211	Reserved	EDMA_IRQRH[INT32]	eDMA channel Interrupt 32
0x0D40	212	—	EDMA_IRQRH[INT33]	eDMA channel Interrupt 33
0x0D50	213	—	EDMA_IRQRH[INT34]	eDMA channel Interrupt 34
0x0D60	214	—	EDMA_IRQRH[INT35]	eDMA channel Interrupt 35
0x0D70	215	—	EDMA_IRQRH[INT36]	eDMA channel Interrupt 36
0x0D80	216	—	EDMA_IRQRH[INT37]	eDMA channel Interrupt 37
0x0D90	217	—	EDMA_IRQRH[INT38]	eDMA channel Interrupt 38
0x0DA0	218	—	EDMA_IRQRH[INT39]	eDMA channel Interrupt 39
0x0DB0	219	—	EDMA_IRQRH[INT40]	eDMA channel Interrupt 40

Table 10-9. INTC: Interrupt Request Sources (continued)

Offset	Vector	Source[1] MPC5553	Source[1] MPC5554	Description
0x0DC0	220	—	EDMA_IRQRH[INT41]	eDMA channel Interrupt 41
0x0DD0	221	—	EDMA_IRQRH[INT42]	eDMA channel Interrupt 42
0x0DE0	222	—	EDMA_IRQRH[INT43]	eDMA channel Interrupt 43
0x0DF0	223	—	EDMA_IRQRH[INT44]	eDMA channel Interrupt 44
0x0E00	224	—	EDMA_IRQRH[INT45]	eDMA channel Interrupt 45
0x0E10	225	—	EDMA_IRQRH[INT46]	eDMA channel Interrupt 46
0x0E20	226	—	EDMA_IRQRH[INT47]	eDMA channel Interrupt 47
0x0E30	227	—	EDMA_IRQRH[INT48]	eDMA channel Interrupt 48
0x0E40	228	—	EDMA_IRQRH[INT49]	eDMA channel Interrupt 49
0x0E50	229	—	EDMA_IRQRH[INT50]	eDMA channel Interrupt 50
0x0E60	230	—	EDMA_IRQRH[INT51]	eDMA channel Interrupt 51
0x0E70	231	—	EDMA_IRQRH[INT52]	eDMA channel Interrupt 52
0x0E80	232	—	EDMA_IRQRH[INT53]	eDMA channel Interrupt 53
0x0E90	233	—	EDMA_IRQRH[INT54]	eDMA channel Interrupt 54
0x0EA0	234	—	EDMA_IRQRH[INT55]	eDMA channel Interrupt 55
0x0EB0	235	—	EDMA_IRQRH[INT56]	eDMA channel Interrupt 56
0x0EC0	236	—	EDMA_IRQRH[INT57]	eDMA channel Interrupt 57
0x0ED0	237	—	EDMA_IRQRH[INT58]	eDMA channel Interrupt 58
0x0EE0	238	—	EDMA_IRQRH[INT59]	eDMA channel Interrupt 59
0x0EF0	239	—	EDMA_IRQRH[INT60]	eDMA channel Interrupt 60
0x0F00	240	—	EDMA_IRQRH[INT61]	eDMA channel Interrupt 61
0x0F10	241	—	EDMA_IRQRH[INT62]	eDMA channel Interrupt 62
0x0F20	242	—	EDMA_IRQRH[INT63]	eDMA channel Interrupt 63
eTPU_B				
0x0F30	243	—	ETPU_CISR_B[CIS0]	eTPU Engine B Channel 0 Interrupt Status
0x0F40	244	—	ETPU_CISR_B[CIS1]	eTPU Engine B Channel 1 Interrupt Status
0x0F50	245	—	ETPU_CISR_B[CIS2]	eTPU Engine B Channel 2 Interrupt Status
0x0F60	246	—	ETPU_CISR_B[CIS3]	eTPU Engine B Channel 3 Interrupt Status
0x0F70	247	—	ETPU_CISR_B[CIS4]	eTPU Engine B Channel 4 Interrupt Status
0x0F80	248	—	ETPU_CISR_B[CIS5]	eTPU Engine B Channel 5 Interrupt Status
0x0F90	249	—	ETPU_CISR_B[CIS6]	eTPU Engine B Channel 6 Interrupt Status
0x0FA0	250	—	ETPU_CISR_B[CIS7]	eTPU Engine B Channel 7 Interrupt Status

Table 10-9. INTC: Interrupt Request Sources (continued)

Offset	Vector	Source[1] MPC5553	Source[1] MPC5554	Description
0x0FB0	251	—	ETPU_CISR_B[CIS8]	eTPU Engine B Channel 8 Interrupt Status
0x0FC0	252	—	ETPU_CISR_B[CIS9]	eTPU Engine B Channel 9 Interrupt Status
0x0fd0	253	—	ETPU_CISR_B[CIS10]	eTPU Engine B Channel 10 Interrupt Status
0x0fe0	254	—	ETPU_CISR_B[CIS11]	eTPU Engine B Channel 11 Interrupt Status
0x0ff0	255	—	ETPU_CISR_B[CIS12]	eTPU Engine B Channel 12 Interrupt Status
0x1000	256	—	ETPU_CISR_B[CIS13]	eTPU Engine B Channel 13 Interrupt Status
0x1010	257	—	ETPU_CISR_B[CIS14]	eTPU Engine B Channel 14 Interrupt Status
0x1020	258	—	ETPU_CISR_B[CIS15]	eTPU Engine B Channel 15 Interrupt Status
0x1030	259	—	ETPU_CISR_B[CIS16]	eTPU Engine B Channel 16 Interrupt Status
0x1040	260	—	ETPU_CISR_B[CIS17]	eTPU Engine B Channel 17 Interrupt Status
0x1050	261	—	ETPU_CISR_B[CIS18]	eTPU Engine B Channel 18 Interrupt Status
0x1060	262	—	ETPU_CISR_B[CIS19]	eTPU Engine B Channel 19 Interrupt Status
0x1070	263	—	ETPU_CISR_B[CIS20]	eTPU Engine B Channel 20 Interrupt Status
0x1080	264	—	ETPU_CISR_B[CIS21]	eTPU Engine B Channel 21 Interrupt Status
0x1090	265	—	ETPU_CISR_B[CIS22]	eTPU Engine B Channel 22 Interrupt Status
0x10A0	266	—	ETPU_CISR_B[CIS23]	eTPU Engine B Channel 23 Interrupt Status
0x10B0	267	—	ETPU_CISR_B[CIS24]	eTPU Engine B Channel 24 Interrupt Status
0x10C0	268	—	ETPU_CISR_B[CIS25]	eTPU Engine B Channel 25 Interrupt Status
0x10D0	269	—	ETPU_CISR_B[CIS26]	eTPU Engine B Channel 26 Interrupt Status
0x10E0	270	—	ETPU_CISR_B[CIS27]	eTPU Engine B Channel 27 Interrupt Status
0x10F0	271	—	ETPU_CISR_B[CIS28]	eTPU Engine B Channel 28 Interrupt Status
0x1100	272	—	ETPU_CISR_B[CIS29]	eTPU Engine B Channel 29 Interrupt Status
0x1110	273	—	ETPU_CISR_B[CIS30]	eTPU Engine B Channel 30 Interrupt Status
0x1120	274	—	ETPU_CISR_B[CIS31]	eTPU Engine B Channel 31 Interrupt Status
DSPI_A				
0x1130	275	—	DSPIA_ISR[TFUF] DSPIA_ISR[RFOF]	DSPI_A combined overrun interrupt requests: Transmit FIFO Underflow and Receive FIFO Overflow
0x1140	276	—	DSPIA_ISR[EOQF]	DSPI_A transmit FIFO End of Queue Flag
0x1150	277	—	DSPIA_ISR[TFFF]	DSPI_A Transmit FIFO Fill Flag
0x1160	278	—	DSPIA_ISR[TCF]	DSPI_A Transfer Complete Flag
0x1170	279	—	DSPIA_ISR[RFDF]	DSPI_A Receive FIFO Drain Flag
FlexCAN_B				

Table 10-9. INTC: Interrupt Request Sources (continued)

Offset	Vector	Source[1] MPC5553	Source[1] MPC5554	Description
0x1180	280	—	CANB_ESR[BOFF_INT]	FLEXCAN_B Bus off Interrupt
0x1190	281	—	CANB_ESR[ERR_INT]	FLEXCAN_B Error Interrupt
0x11A0	282	—	Reserved	Reserved
0x11B0	283	—	CANB_IFRL[BUF0]	FLEXCAN_B Buffer 0 Interrupt
0x11C0	284	—	CANB_IFRL[BUF1]	FLEXCAN_B Buffer 1 Interrupt
0x11D0	285	—	CANB_IFRL[BUF2]	FLEXCAN_B Buffer 2 Interrupt
0x11E0	286	—	CANB_IFRL[BUF3]	FLEXCAN_B Buffer 3 Interrupt
0x11F0	287	—	CANB_IFRL[BUF4]	FLEXCAN_B Buffer 4 Interrupt
0x1200	288	—	CANB_IFRL[BUF5]	FLEXCAN_B Buffer 5 Interrupt
0x1210	289	—	CANB_IFRL[BUF6]	FLEXCAN_B Buffer 6 Interrupt
0x1220	290	—	CANB_IFRL[BUF7]	FLEXCAN_B Buffer 7 Interrupt
0x1230	291	—	CANB_IFRL[BUF8]	FLEXCAN_B Buffer 8 Interrupt
0x1240	292	—	CANB_IFRL[BUF9]	FLEXCAN_B Buffer 9 Interrupt
0x1250	293	—	CANB_IFRL[BUF10]	FLEXCAN_B Buffer 10 Interrupt
0x1260	294	—	CANB_IFRL[BUF11]	FLEXCAN_B Buffer 11 Interrupt
0x1270	295	—	CANB_IFRL[BUF12]	FLEXCAN_B Buffer 12 Interrupt
0x1280	296	—	CANB_IFRL[BUF13]	FLEXCAN_B Buffer 13 Interrupt
0x1290	297	—	CANB_IFRL[BUF14]	FLEXCAN_B Buffer 14 Interrupt
0x12A0	298	—	CANB_IFRL[BUF15]	FLEXCAN_B Buffer 15 Interrupt
0x12B0	299	—	CANB_IFRL[BUF31:BUF16]	FLEXCAN_B Buffers 31 - 16 Interrupts
0x12C0	300	—	CANB_IFRH[BUF63:BUF32]	FLEXCAN_B Buffers 63 - 32 Interrupts
0x12D0	301	—	Reserved	Reserved
0x12E0	302	—	Reserved	Reserved
0x12F0	303	—	Reserved	Reserved
0x1300	304	—	Reserved	Reserved
0x1310	305	—	Reserved	Reserved
0x1320	306	—	Reserved	Reserved
0x1330	307	—	Reserved	Reserved

[1] Interrupt requests from the same module location are ORed together.

NOTE

The INTC has no spurious vector support. Therefore, if an asserted peripheral or software settable interrupt request, whose PRI*n* value in INTC_PSR0–INTC_PSR307 is higher than the PRI value in INTC_CPR, negates before the interrupt request to the processor for that peripheral or software settable interrupt request is acknowledged, the interrupt request to the processor still can assert or will remain asserted for that peripheral or software settable interrupt request. In this case, the interrupt vector will correspond to that peripheral or software settable interrupt request. Also, the PRI value in the INTC_CPR will be updated with the corresponding PRI*n* value in INTC_PSR*n*.

Furthermore, clearing the peripheral interrupt request's enable bit in the peripheral or, alternatively, setting its mask bit has the same consequences as clearing its flag bit. Setting its enable bit or clearing its mask bit while its flag bit is asserted has the same effect on the INTC as an interrupt event setting the flag bit.

10.4.1.1 Peripheral Interrupt Requests

An interrupt event in a peripheral's hardware sets a flag bit which resides in that peripheral. The interrupt request from the peripheral is driven by that flag bit.

The time from when the peripheral starts to drive its peripheral interrupt request to the INTC to the time that the INTC starts to drive the interrupt request to the processor is three clocks.

10.4.1.2 Software Settable Interrupt Requests

The software set/clear interrupt registers (INTC_SSCIR*x*_*x*) support the setting or clearing of software-settable interrupt requests. These registers contain eight independent sets of bits to set and clear a corresponding flag bit by software. With the exception of being set by software, this flag bit behaves the same as a flag bit set within a peripheral. This flag bit generates an interrupt request within the INTC just like a peripheral interrupt request.

An interrupt request is triggered by software writing a 1 to the SET*n* bit in INTC software set/clear interrupt registers (INTC_SSCIR0–INTC_SSCIR7). This write sets the corresponding CLR*n* bit, which is a flag bit, resulting in the interrupt request. The interrupt request is cleared by writing a 1 to the CLR*n* bit. Specific behavior includes the following:

- Writing a 1 to SET*n* leaves SET*n* unchanged at '0' but sets the flag bit (which is the CLR*n* bit).
- Writing a 0 to SET*n* has no effect.
- Writing a 1 to CLR*n* clears the flag (CLRx) bit.
- Writing a 0 to CLR*n* has no effect.
- If a 1 is written to a pair of SET*n* and CLR*n* bits at the same time, the flag (CLRx) is set, regardless of whether CLR*n* was asserted before the write.

The time from the write to the SET*n* bit to the time that the INTC starts to drive the interrupt request to the processor is four clocks.

10.4.1.3 Unique Vector for Each Interrupt Request Source

Each peripheral and software settable interrupt request is assigned a hardwired unique 9-bit vector. Software settable interrupts 0–7 are assigned vectors 0–7, respectively. The peripheral interrupt requests are assigned vectors 8 to as high as needed to cover all of the peripheral interrupt requests.

10.4.2 Priority Management

The asserted interrupt requests are compared to each other based on their PRI*n* values in INTC priority select registers (INTC_PSR0–INTC_PSR307). The result of that comparison also is compared to PRI in INTC current priority register (INTC_CPR). The results of those comparisons are used to manage the priority of the ISR being executed by the processor. The LIFO also assists in managing that priority.

10.4.2.1 Current Priority and Preemption

The priority arbitrator, selector, encoder, and comparator submodules shown in Figure 10-1 are used to compare the priority of the asserted interrupt requests to the current priority. If the priority of any asserted peripheral or software settable interrupt request is higher than the current priority, then the interrupt request to the processor is asserted. Also, a unique vector for the preempting peripheral or software settable interrupt request is generated for INTC interrupt acknowledge register (INTC_IACKR), and if in hardware vector mode, for the interrupt vector provided to the processor.

10.4.2.1.1 Priority Arbitrator Submodule

The priority arbitrator submodule compares all the priorities of all of the asserted interrupt requests, both peripheral and software settable. The output of the priority arbitrator submodule is the highest of those priorities. Also, any interrupt requests which have this highest priority are output as asserted interrupt requests to the request selector submodule.

10.4.2.1.2 Request Selector Submodule

If only one interrupt request from the priority arbitrator submodule is asserted, then it is passed as asserted to the vector encoder submodule. If multiple interrupt requests from the priority arbitrator submodule are asserted, then only the one with the lowest vector is passed as asserted to the vector encoder submodule. The lower vector is chosen regardless of the time order of the assertions of the peripheral or software settable interrupt requests.

10.4.2.1.3 Vector Encoder Submodule

The vector encoder submodule generates the unique 9-bit vector for the asserted interrupt request from the request selector submodule.

10.4.2.1.4 Priority Comparator Submodule

The priority comparator submodule compares the highest priority output from the priority arbitrator submodule with PRI in INTC_CPR. If the priority comparator submodule detects that this highest priority is higher than the current priority, then it asserts the interrupt request to the processor. This interrupt request to the processor asserts whether this highest priority is raised above the value of PRI in INTC_CPR or the PRI value in INTC_CPR is lowered below this highest priority. This highest priority then becomes the new priority which will be written to PRI in INTC_CPR when the interrupt request to the processor is acknowledged. Interrupt requests whose PRI*n* in INTC_PSR*n* are zero will not cause a preemption because their PRI*n* will not be higher than PRI in INTC_CPR.

10.4.2.2 LIFO

The LIFO stores the preempted PRI values from the INTC_CPR. Therefore, because these priorities are stacked within the INTC, if interrupts need to be enabled during the ISR, at the beginning of the interrupt exception handler the PRI value in the INTC_CPR does not need to be loaded from the INTC_CPR and stored onto the context stack. Likewise at the end of the interrupt exception handler, the priority does not need to be loaded from the context stack and stored into the INTC_CPR.

The PRI value in the INTC_CPR is pushed onto the LIFO when the INTC_IACKR is read in software vector mode or the interrupt acknowledge signal from the processor is asserted in hardware vector mode. The priority is popped into PRI in the INTC_CPR whenever the INTC_EOIR is written.

Although the INTC supports 16 priorities, an ISR executing with PRI in the INTC_CPR equal to 15 will not be preempted. Therefore, the LIFO supports the stacking of 15 priorities. However, the LIFO is only 14 entries deep. An entry for a priority of 0 is not needed because of how pushing onto a full LIFO and popping an empty LIFO are treated. If the LIFO is pushed 15 or more times than it is popped, the priorities first pushed are overwritten. A priority of 0 would be an overwritten priority. However, the LIFO will pop ‘0’s if it is popped more times than it is pushed. Therefore, although a priority of 0 was overwritten, it is regenerated with the popping of an empty LIFO.

The LIFO is not memory mapped.

10.4.3 Details on Handshaking with Processor

10.4.3.1 Software Vector Mode Handshaking

10.4.3.1.1 Acknowledging Interrupt Request to Processor

A timing diagram of the interrupt request and acknowledge handshaking in software vector mode, along with the handshaking near the end of the interrupt exception handler, is shown in Figure 10-14. The INTC examines the peripheral and software settable interrupt requests. When it finds an asserted peripheral or software settable interrupt request with a higher priority than PRI in INTC current priority register (INTC_CPR), it asserts the interrupt request to the processor. The INTVEC field in INTC interrupt acknowledge register (INTC_IACKR) is updated with the preempting interrupt request’s vector when the interrupt request to the processor is asserted. The INTVEC field retains that value until the next time the interrupt request to the processor is asserted. The rest of the handshaking is described in Section 10.1.4.1, “Software Vector Mode.”

10.4.3.1.2 End-of-Interrupt Exception Handler

Before the interrupt exception handling completes, INTC end-of-interrupt register (INTC_EOIR) must be written. When it is written, the LIFO is popped so that the preempted priority is restored into PRI of the INTC_CPR. Before it is written, the peripheral or software settable flag bit must be cleared so that the peripheral or software settable interrupt request is negated.

NOTE

To ensure proper operation across all eSys MCUs, execute an **mbar** or **msync** instruction between the access to clear the flag bit and the write to the INTC_EOIR.

When returning from the preemption, the INTC does not search for the peripheral or software settable interrupt request whose ISR was preempted. Depending on how much the ISR progressed, that interrupt request may no longer even be asserted. When PRI in INTC_CPR is lowered to the priority of the

preempted ISR, the interrupt request for the preempted ISR or any other asserted peripheral or software settable interrupt request at or below that priority will not cause a preemption. Instead, after the restoration of the preempted context, the processor will return to the instruction address that it was to next execute before it was preempted. This next instruction is part of the preempted ISR or the interrupt exception handler's prolog or epilog.

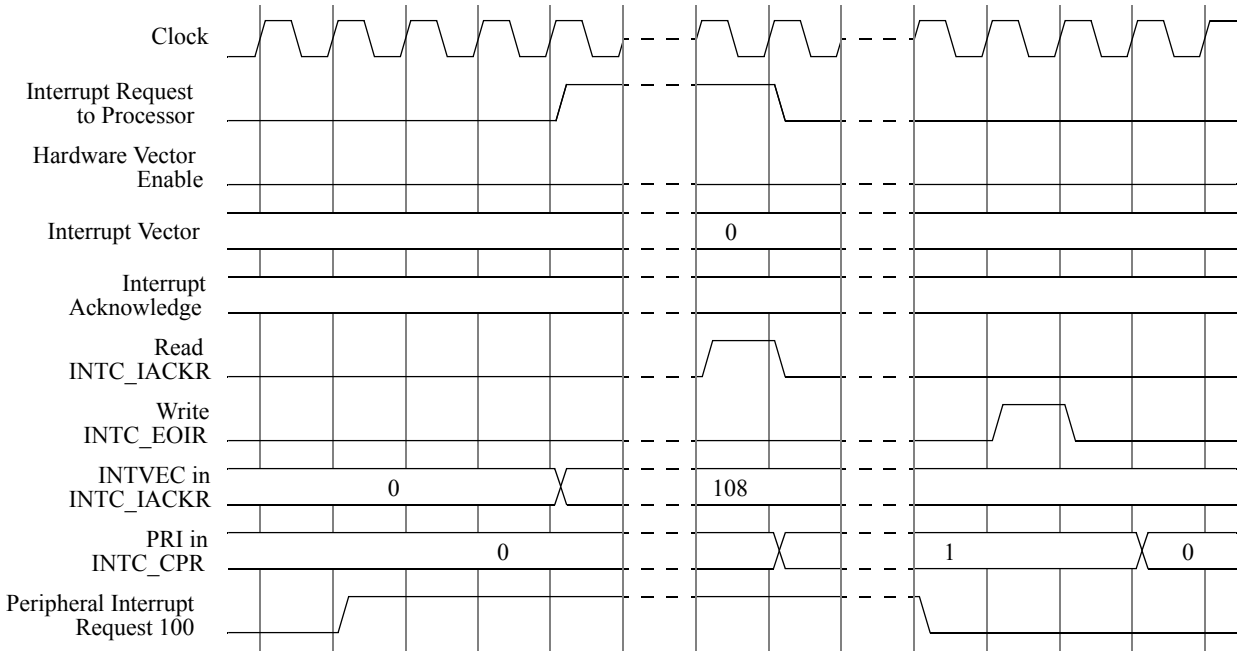


Figure 10-14. Software Vector Mode Handshaking Timing Diagram

10.4.3.2 Hardware Vector Mode Handshaking

A timing diagram of the interrupt request and acknowledge handshaking in hardware vector mode, along with the handshaking near the end of the interrupt exception handler, is shown in Figure 10-15. As in software vector mode, the INTC examines the peripheral and software settable interrupt requests, and when it finds an asserted one with a higher priority than PRI in INTC_CPR, it asserts the interrupt request to the processor. The INTVEC field in the INTC_IACKR is updated with the preempting peripheral or software settable interrupt request's vector when the interrupt request to the processor is asserted. The INTVEC field retains that value until the next time the interrupt request to the processor is asserted. In addition, the value of the interrupt vector to the processor matches the value of the INTVEC field in the INTC_IACKR. The rest of the handshaking is described in Section 10.1.4.2, "Hardware Vector Mode."

The handshaking near the end of the interrupt exception handler, that is the writing to the INTC_EOIR, is the same as in software vector mode. Refer to Section 10.4.3.1.2, "End-of-Interrupt Exception Handler."

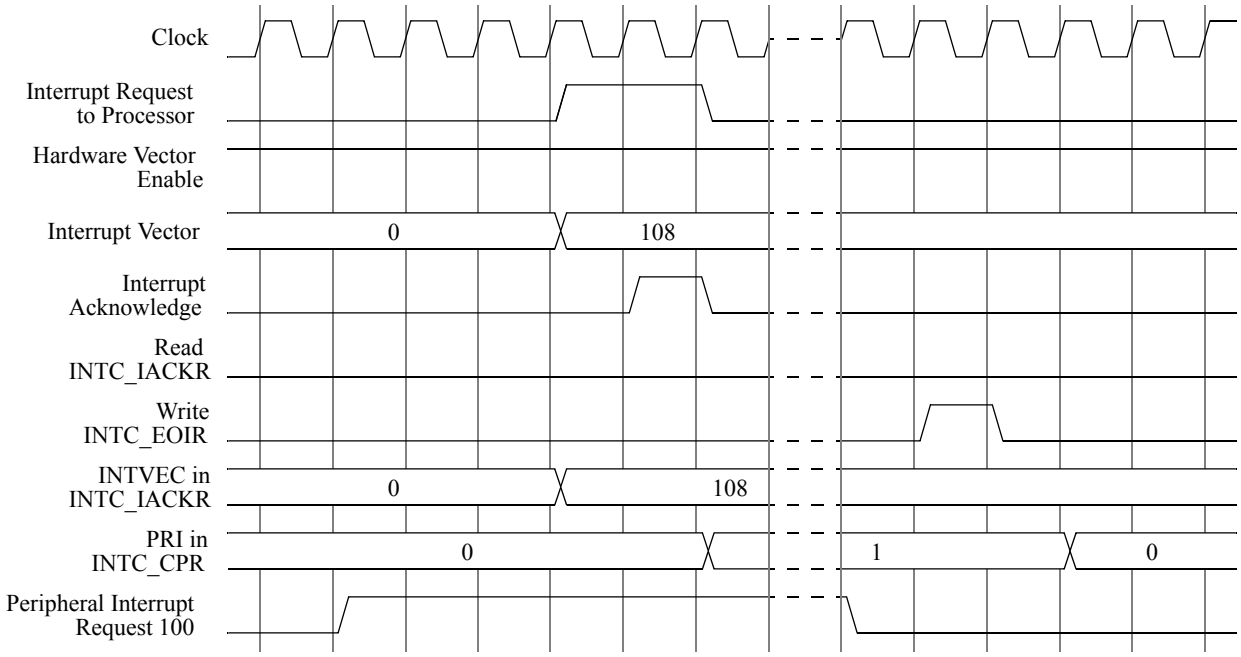


Figure 10-15. Hardware Vector Mode Handshaking Timing Diagram

10.5 Initialization/Application Information

10.5.1 Initialization Flow

After exiting reset, all of the PRI*n* fields in INTC priority select registers (INTC_PSR0–INTC_PSR307) will be zero, and PRI in INTC current priority register (INTC_CPR) will be 15. These reset values will prevent the INTC from asserting the interrupt request to the processor. The enable or mask bits in the peripherals are reset such that the peripheral interrupt requests are negated. An initialization sequence for allowing the peripheral and software settable interrupt requests to cause an interrupt request to the processor is:

```
interrupt_request_initialization:
configure VTES and HVEN in INTC_MCR
configure VTBA in INTC_IACKR
raise the PRIn fields in INTC_PSRn
set the enable bits or clear the mask bits for the peripheral interrupt requests
lower PRI in INTC_CPR to zero
enable processor recognition of interrupts
```

10.5.2 Interrupt Exception Handler

These example interrupt exception handlers use PowerPC Book E assembly code.

10.5.2.1 Software Vector Mode

```
interrupt_exception_handler:
code to create stack frame, save working register, and save SRR0 and SRR1
```

```
lis       r3,INTC_IACKR@ha              # form adjusted upper half of INTC_IACKR address
lwz       r3,INTC_IACKR@l(r3)           # load INTC_IACKR, which clears request to processor
lwz       r3,0x0(r3)                    # load address of ISR from vector table
wrteei    1                             # enable processor recognition of interrupts

code to save rest of context required by e500 EABI

mtlr      r3                            # move INTC_IACKR contents into link register
blrl                                    # branch to ISR; link register updated with epilog
                                        # address

epilog:
code to restore most of context required by e500 EABI

# Popping the LIFO after the restoration of most of the context and the disabling of processor
# recognition of interrupts eases the calculation of the maximum stack depth at the cost of
# postponing the servicing of the next interrupt request.
mbar                                    # ensure store to clear flag bit has completed
lis       r3,INTC_EOIR@ha               # form adjusted upper half of INTC_EOIR address
li        r4,0x0                        # form 0 to write to INTC_EOIR
wrteei    0                             # disable processor recognition of interrupts
stw       r4,INTC_EOIR@l(r3)            # store to INTC_EOIR, informing INTC to lower priority

code to restore SRR0 and SRR1, restore working registers, and delete stack frame

rfi


vector_table_base_address:
address of ISR for interrupt with vector 0
address of ISR for interrupt with vector 1
          .
          .
          .
address of ISR for interrupt with vector 510
address of ISR for interrupt with vector 511


ISRx:
code to service the interrupt event
code to clear flag bit which drives interrupt request to INTC

blr                                     # return to epilog
```

10.5.2.2 Hardware Vector Mode

This interrupt exception handler is useful with processor and system bus implementations that support a hardware vector. This example assumes that each `interrupt_exception_handlerx` only has space for four instructions, and therefore a branch to `interrupt_ exception_handler_continuedx` is needed.

```
interrupt_exception_handlerx:
b         interrupt_exception_handler_continuedx# 4 instructions available, branch to continue


interrupt_exception_handler_continuedx:
code to create stack frame, save working register, and save SRR0 and SRR1
```

```
wrteei    1                             # enable processor recognition of interrupts

code to save rest of context required by e500 EABI

bl        ISRx                          # branch to ISR for interrupt with vector x


epilog:
code to restore most of context required by e500 EABI

# Popping the LIFO after the restoration of most of the context and the disabling of processor
# recognition of interrupts eases the calculation of the maximum stack depth at the cost of
# postponing the servicing of the next interrupt request.
mbar                                    # ensure store to clear flag bit has completed
lis       r3,INTC_EOIR@ha               # form adjusted upper half of INTC_EOIR address
li        r4,0x0                        # form 0 to write to INTC_EOIR
wrteei    0                             # disable processor recognition of interrupts
stw       r4,INTC_EOIR@l(r3)            # store to INTC_EOIR, informing INTC to lower priority

code to restore SRR0 and SRR1, restore working registers, and delete stack frame

rfi


ISRx:
code to service the interrupt event
code to clear flag bit which drives interrupt request to INTC

blr                                     # branch to epilog
```

10.5.3 ISR, RTOS, and Task Hierarchy

The RTOS and all of the tasks under its control typically execute with PRI in INTC current priority register (INTC_CPR) having a value of 0. The RTOS will execute the tasks according to whatever priority scheme that it may have, but that priority scheme is independent and has a lower priority of execution than the priority scheme of the INTC. In other words, the ISRs execute above INTC_CPR priority 0 and outside the control of the RTOS, the RTOS executes at INTC_CPR priority 0, and while the tasks execute at different priorities under the control of the RTOS, they also execute at INTC_CPR priority 0.

If a task shares a resource with an ISR and the PCP is being used to manage that shared resource, then the task's priority can be elevated in the INTC_CPR while the shared resource is being accessed.

An ISR whose PRI*n* in INTC priority select registers (INTC_PSR0–INTC_PSR307) has a value of 0 will not cause an interrupt request to the processor, even if its peripheral or software settable interrupt request is asserted. For a peripheral interrupt request, not setting its enable bit or disabling the mask bit will cause it to remain negated, which consequently also will not cause an interrupt request to the processor. Since the ISRs are outside the control of the RTOS, this ISR will not run unless called by another ISR or the interrupt exception handler, perhaps after executing another ISR.

10.5.4 Order of Execution

An ISR with a higher priority can preempt an ISR with a lower priority, regardless of the unique vectors associated with each of their peripheral or software settable interrupt requests. However, if multiple peripheral or software settable interrupt requests are asserted, more than one has the highest priority, and that priority is high enough to cause preemption, the INTC selects the one with the lowest unique vector

regardless of the order in time that they asserted. However, the ability to meet deadlines with this scheduling scheme is no less than if the ISRs execute in the time order that their peripheral or software settable interrupt requests asserted.

The example in Table 10-10 shows the order of execution of both ISRs with different priorities and the same priority.

Table 10-10. Order of ISR Execution Example

Step	Step Description	Code Executing At End of Step						PRI in INTC_CPR at End of Step
		RTOS	ISR108[1]	ISR208	ISR308	ISR408	Interrupt Exception Handler	
1	RTOS at priority 0 is executing.	X						0
2	Peripheral interrupt request 100 at priority 1 asserts. Interrupt taken.		X					1
3	Peripheral interrupt request 400 at priority 4 is asserts. Interrupt taken.					X		4
4	Peripheral interrupt request 300 at priority 3 is asserts.					X		4
5	Peripheral interrupt request 200 at priority 3 is asserts.					X		4
6	ISR408 completes. Interrupt exception handler writes to INTC_EOIR.						X	1
7	Interrupt taken. ISR208 starts to execute, even though peripheral interrupt request 300 asserted first.			X				3
8	ISR208 completes. Interrupt exception handler writes to INTC_EOIR.						X	1
9	Interrupt taken. ISR308 starts to execute.				X			3
10	ISR308 completes. Interrupt exception handler writes to INTC_EOIR.						X	1
11	ISR108 completes. Interrupt exception handler writes to INTC_EOIR.						X	0
12	RTOS continues execution.	X						0

[1] ISR108 executes for peripheral interrupt request 100 because the first eight ISRs are for software settable interrupt requests.

10.5.5 Priority Ceiling Protocol

10.5.5.1 Elevating Priority

The PRI field in INTC current priority register (INTC_CPR) is elevated in the OSEK PCP to the ceiling of all of the priorities of the ISRs that share a resource. This protocol therefore allows coherent accesses of the ISRs to that shared resource.

For example, ISR1 has a priority of 1, ISR2 has a priority of 2, and ISR3 has a priority of 3. They all share the same resource. Before ISR1 or ISR2 can access that resource, they must raise the PRI value in INTC_CPR to 3, the ceiling of all of the ISR priorities. After they release the resource, they must lower the PRI value in INTC_CPR to prevent further priority inversion. If they do not raise their priority, then ISR2 can preempt ISR1, and ISR3 can preempt ISR1 or ISR2, possibly corrupting the shared resource. Another possible failure mechanism is deadlock if the higher priority ISR needs the lower priority ISR to release the resource before it can continue, but the lower priority ISR can not release the resource until the higher priority ISR completes and execution returns to the lower priority ISR.

Using the PCP instead of disabling processor recognition of all interrupts reduces the priority inversion time when accessing a shared resource. For example, while ISR3 can not preempt ISR1 while it is accessing the shared resource, all of the ISRs with a priority higher than 3 can preempt ISR1.

10.5.5.2 Ensuring Coherency

A scenario can exist that can cause non-coherent accesses to the shared resource. As an example, ISR1 and ISR2 both share a resource. ISR1 has a lower priority than ISR2. ISR1 is executing, and it writes to the INTC_CPR. The instruction following this store is a store to a value in a shared coherent data block. Either just before or at the same time as the first store, the INTC asserts the interrupt request to the processor because the peripheral interrupt request for ISR2 has asserted. As the processor is responding to the interrupt request from the INTC, and as it is aborting transactions and flushing its pipeline, it is possible that both of these stores will be executed. ISR2 thereby thinks that it can access the data block coherently, but the data block has been corrupted.

OSEK uses the GetResource and ReleaseResource system services to manage access to a shared resource. To prevent this corruption of a coherent data block, modifications to PRI in INTC_CPR can be made by those system services with the following code sequences.

```
GetResource:
raise PRI

mbar
isync

ReleaseResource:
mbar
lower PRI
```

10.5.6 Selecting Priorities According to Request Rates and Deadlines

The selection of the priorities for the ISRs can be made using rate monotonic scheduling or a superset of it, deadline monotonic scheduling. In RMS, the ISRs which have higher request rates have higher priorities. In DMS, if the deadline is before the next time the ISR is requested, then the ISR is assigned a priority according to the time from the request for the ISR to the deadline, not from the time of the request for the ISR to the next request for it.

For example, ISR1 executes every 100 µs, ISR2 executes every 200 µs, and ISR3 executes every 300 µs. ISR1 has a higher priority than ISR2 which has a higher priority than ISR3. However, if ISR3 has a deadline of 150 µs, then it has a higher priority than ISR2.

The INTC has 16 priorities, which could be much less than the number of ISRs. In this case, the ISRs should be grouped with other ISRs that have similar deadlines. For example, a priority could be allocated for every time the request rate doubles. ISRs with request rates around 1 ms would share a priority, ISRs with request rates around 500 µs would share a priority, ISRs with request rates around 250 µs would share a priority, etc. With this approach, a range of ISR request rates of 2^{16} could be covered, regardless of the number of ISRs.

Reducing the number of priorities does cause some priority inversion which reduces the processor's ability to meet its deadlines. It also allows easier management of ISRs with similar deadlines that share a resource. They can be placed at the same priority without any further priority inversion, and they do not need to use the PCP to access the shared resource.

10.5.7 Software Settable Interrupt Requests

The software settable interrupt requests can be used in two ways. They can be used to schedule a lower priority portion of an ISR and for processors to interrupt other processors in a multiple processor system.

10.5.7.1 Scheduling a Lower Priority Portion of an ISR

A portion of an ISR needs to be executed at the PRI*n* value in INTC priority select registers (INTC_PSR0–INTC_PSR307), which becomes the PRI value in INTC current priority register (INTC_CPR) with the interrupt acknowledgement. The ISR, however, can have a portion of it which does not need to be executed at this higher priority. Therefore, executing this later portion which does not need to be executed at this higher priority can block the execution of ISRs which do not have a higher priority than the earlier portion of the ISR but do have a higher priority than what the later portion of the ISR needs. This priority inversion reduces the processor's ability to meet its deadlines.

One option is for the ISR to complete the earlier higher priority portion, but then schedule through the RTOS a task to execute the later lower priority portion. However, some RTOSs can require a large amount of time for an ISR to schedule a task. Therefore, a second option is for the ISR, after completing the higher priority portion, to set a SET*n* bit in INTC software set/clear interrupt registers (INTC_SSCIR0–INTC_SSCIR7). Writing a 1 to SET*n* causes a software settable interrupt request. This software settable interrupt request, which usually will have a lower PRI*n* value in the INTC_PSR*n*, therefore will not cause priority inversion.

10.5.7.2 Scheduling an ISR on Another Processor

Since the SET*n* bits in the INTC_SSCIR*n* are memory mapped, processors in multiple processor systems can schedule ISRs on the other processors. One application is that one processor simply wants to command another processor to perform a piece of work, and the initiating processor does not need to use the results of that work. If the initiating processor is concerned that processor executing the software settable ISR has not completed the work before asking it to again execute that ISR, it can check if the corresponding CLR*n* bit in INTC_SSCIR*n* is asserted before again writing a 1 to the SET*n* bit.

Another application is the sharing of a block of data. For example, a first processor has completed accessing a block of data and wants a second processor to then access it. Furthermore, after the second processor has completed accessing the block of data, the first processor again wants to access it. The accesses to the block of data must be done coherently. The procedure is that the first processor writes a 1 to a SET*n* bit on the second processor. The second processor, after accessing the block of data, clears the

corresponding CLR*n* bit and then writes 1 to a SET*n* bit on the first processor, informing it that it now can access the block of data.

10.5.8 Lowering Priority Within an ISR

In implementations without the software-settable interrupt requests in the INTC software set/clear interrupt registers (INTC_SSCIR0–INTC_SSCIR7), the only way—besides scheduling a task through an RTOS—to prevent priority inversion with an ISR whose work spans multiple priorities (as described in Section 10.5.7.1, "Scheduling a Lower Priority Portion of an ISR,") is to lower the current priority. However, the INTC has a LIFO whose depth is determined by the number of priorities.

NOTE

Lowering the PRI value in INTC current priority register (INTC_CPR) within an ISR to below the ISR's corresponding PRI value in INTC priority select registers (INTC_PSR0–INTC_PSR307) allows more preemptions than the depth of the LIFO can support.

Therefore, the INTC does not support lowering the current priority within an ISR as a way to avoid priority inversion.

10.5.9 Negating an Interrupt Request Outside of its ISR

10.5.9.1 Negating an Interrupt Request as a Side Effect of an ISR

Some peripherals have flag bits which can be cleared as a side effect of servicing a peripheral interrupt request. For example, reading a specific register can clear the flag bits, and consequently their corresponding interrupt requests too. This clearing as a side effect of servicing a peripheral interrupt request can cause the negation of other peripheral interrupt requests besides the peripheral interrupt request whose ISR presently is executing. This negating of a peripheral interrupt request outside of its ISR can be a desired effect.

10.5.9.2 Negating Multiple Interrupt Requests in One ISR

An ISR can clear other flag bits besides its own flag bit. One reason that an ISR clears multiple flag bits is because it serviced those other flag bits, and therefore the ISRs for these other flag bits do not need to be executed.

10.5.9.3 Proper Setting of Interrupt Request Priority

Whether an interrupt request negates outside of its own ISR due to the side effect of an ISR execution or the intentional clearing a flag bit, the priorities of the peripheral or software settable interrupt requests for these other flag bits must be selected properly. Their PRI*n* values in INTC priority select registers (INTC_PSR0–INTC_PSR307) must be selected to be at or lower than the priority of the ISR that cleared their flag bits. Otherwise, those flag bits still can cause the interrupt request to the processor to assert. Furthermore, the clearing of these other flag bits also has the same timing relationship to the writing to INTC end-of-interrupt register (INTC_EOIR) as the clearing of the flag bit that caused the present ISR to be executed. Refer to Section 10.4.3.1.2, "End-of-Interrupt Exception Handler," for more information.

A flag bit whose enable bit or mask bit is negating its peripheral interrupt request can be cleared at any time, regardless of the peripheral interrupt request's PRI*n* value in INTC_PSR*n*.

10.5.10 Examining LIFO contents

Normally the user does not need to know the contents of the LIFO. One may not even know how deeply the LIFO is nested. However, if one should want to read the contents, they are not memory mapped. The contents still can be read by popping the LIFO and reading the PRI field in the INTC current priority register (INTC_CPR). The code sequence is:

```
pop_lifo:
store to INTC_EOIR
load INTC_CPR, examine PRI, and store onto stack
if PRI is not zero or value when interrupts were enabled, branch to pop_lifo
```

When the examination is complete, the LIFO can be restored using this code sequence:

```
push_lifo:
load stacked PRI value and store to INTC_CPR
load INTC_IACKR
if stacked PRI values are not depleted, branch to push_lifo
```

10.6 Revision History

Substantive Changes since Rev 3.0
In section Section 10.1.4.1, “Software Vector Mode,” third paragraph, after the sentence, "The interrupt request to the processor will not clear if a higher priority interrupt request arrives," added the sentence "Even in this case, INTVEC will not update to the higher priority request until the lower priority interrupt request is acknowledged by reading the INTC_IACKR."
In section Section 10.1.4.2, “Hardware Vector Mode,” second paragraph, changed the sentence, "Even if a higher priority interrupt request arrives while waiting for this interrupt acknowledge, the interrupt request to the processor will negate for at least one clock," to "However, the interrupt request to the processor will not negate if a higher priority interrupt request arrives. Even in this case, the interrupt vector number will not update to the higher priority request until the lower priority request is acknowledged by the processor."
Updated Table 10-9 to reflect MPC5553 interrupt sources.

Chapter 11
Frequency Modulated Phase Locked Loop (FMPLL) and System Clocks

11.1 Introduction

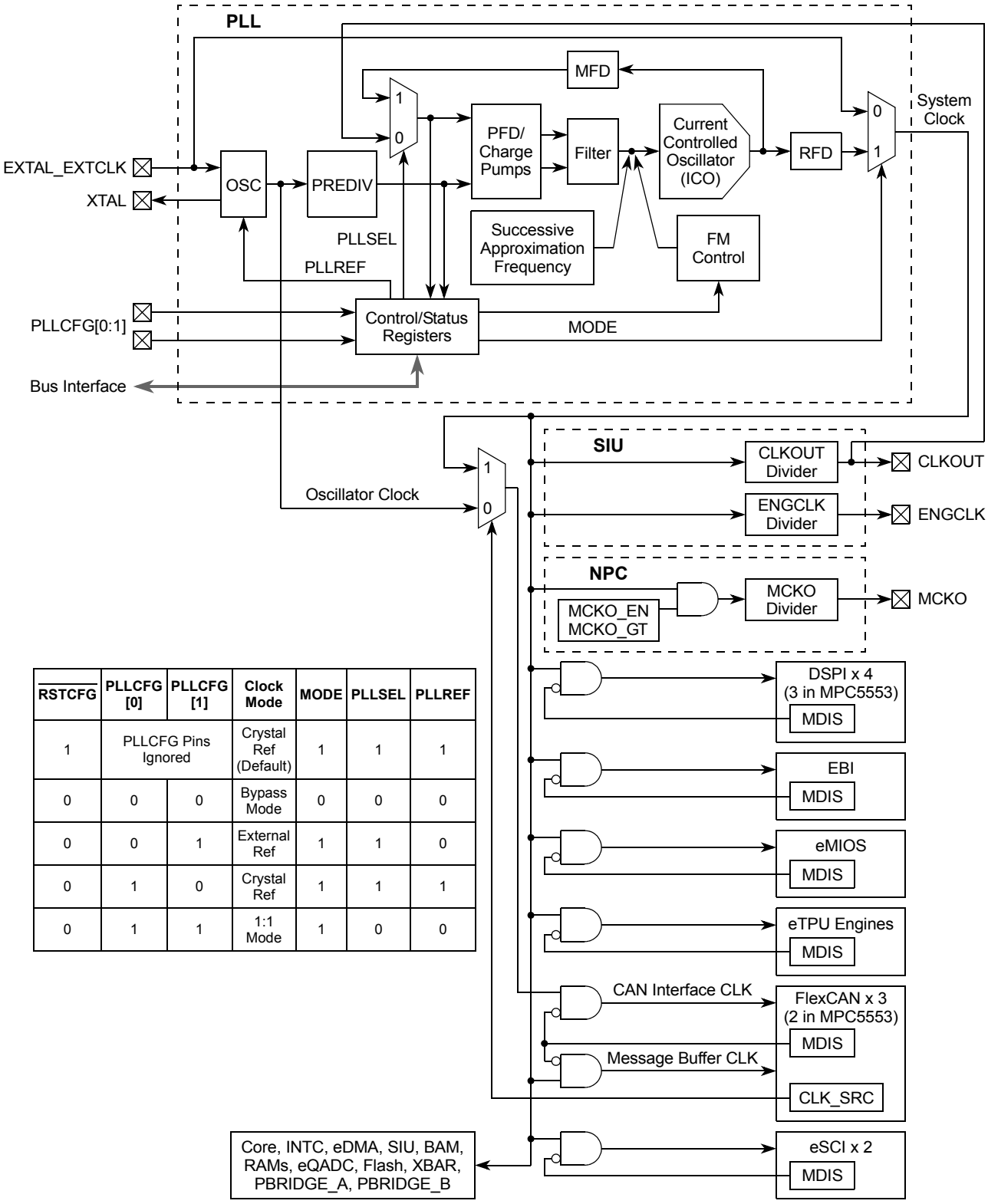
This section describes the features and function of the FMPLL module.

11.1.1 Block Diagrams

This section contains block diagrams that illustrate the FMPLL, the clock architecture, and the various FMPLL and clock configurations that are available on the MPC5553/MPC5554. The following diagrams are provided:

- Figure 11-1, “FMPLL and Clock Architecture”
- Figure 11-2, “FMPLL Bypass Mode”
- Figure 11-3, “FMPLL External Reference Mode”
- Figure 11-4, “FMPLL Crystal Reference Mode Without FM”
- Figure 11-5, “FMPLL Crystal Reference Mode With FM”
- Figure 11-6, “FMPLL Dual-Controller (1:1) Mode”

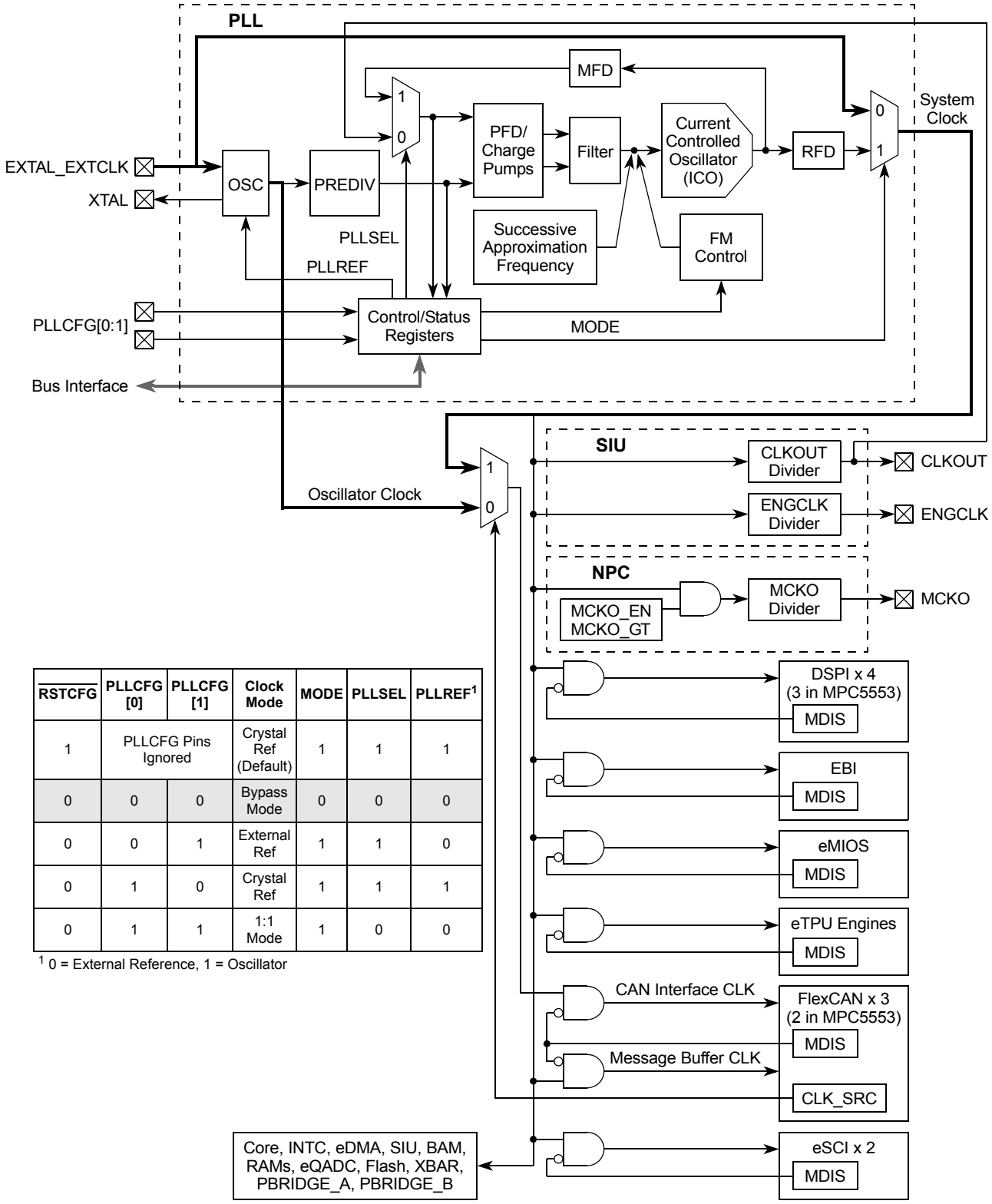
11.1.1.1 FMPLL and Clock Architecture



RSTCFG	PLLCFG [0]	PLLCFG [1]	Clock Mode	MODE	PLLSEL	PLLREF
1	PLLCFG Pins Ignored		Crystal Ref (Default)	1	1	1
0	0	0	Bypass Mode	0	0	0
0	0	1	External Ref	1	1	0
0	1	0	Crystal Ref	1	1	1
0	1	1	1:1 Mode	1	0	0

Figure 11-1. FMPLL Block and Clock Architecture

11.1.1.2 FMPLL Bypass Mode

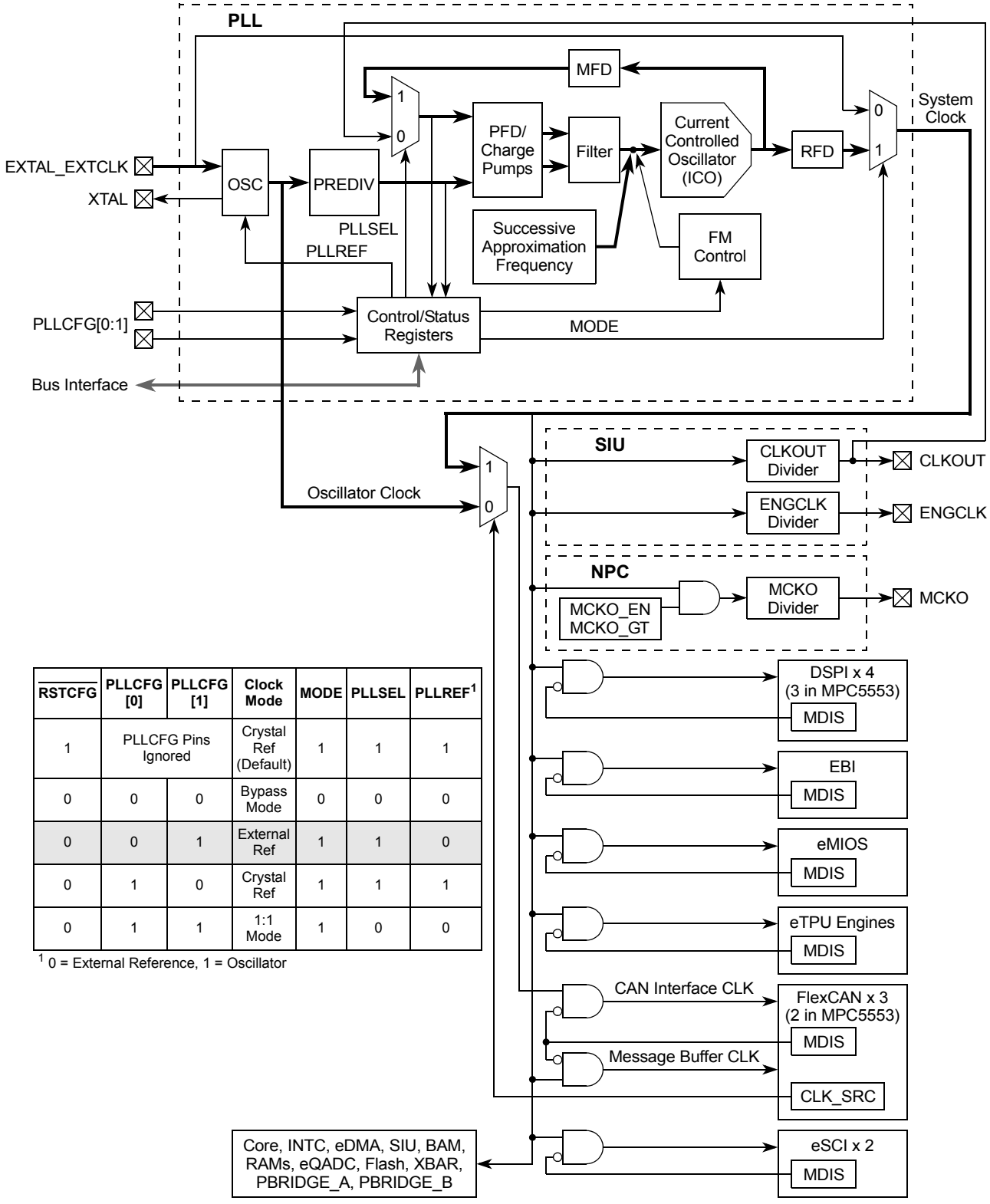


$\overline{\text{RSTCFG}}$	PLLCFG [0]	PLLCFG [1]	Clock Mode	MODE	PLLSEL	PLLREF[1]
1	PLLCFG Pins Ignored		Crystal Ref (Default)	1	1	1
0	0	0	Bypass Mode	0	0	0
0	0	1	External Ref	1	1	0
0	1	0	Crystal Ref	1	1	1
0	1	1	1:1 Mode	1	0	0

[1] 0 = External Reference, 1 = Oscillator

Figure 11-2. FMPLL Bypass Mode

11.1.1.3 FMPLL External Reference Mode

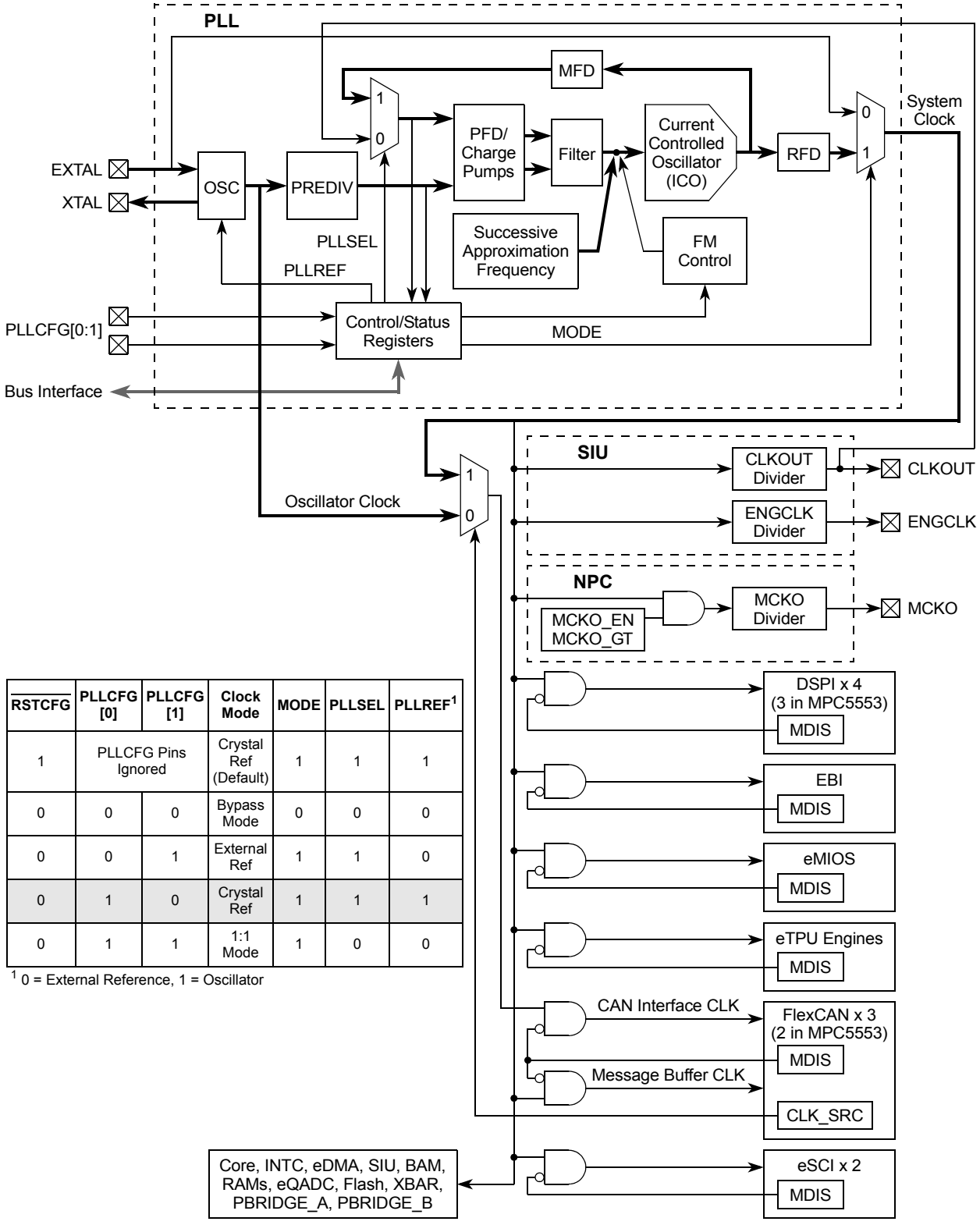


RSTCFG	PLLCFG [0]	PLLCFG [1]	Clock Mode	MODE	PLLSEL	PLLREF[1]
1	PLLCFG Pins Ignored		Crystal Ref (Default)	1	1	1
0	0	0	Bypass Mode	0	0	0
0	0	1	External Ref	1	1	0
0	1	0	Crystal Ref	1	1	1
0	1	1	1:1 Mode	1	0	0

[1] 0 = External Reference, 1 = Oscillator

Figure 11-3. FMPLL External Reference Mode

11.1.1.4 FMPLL Crystal Reference Mode Without FM

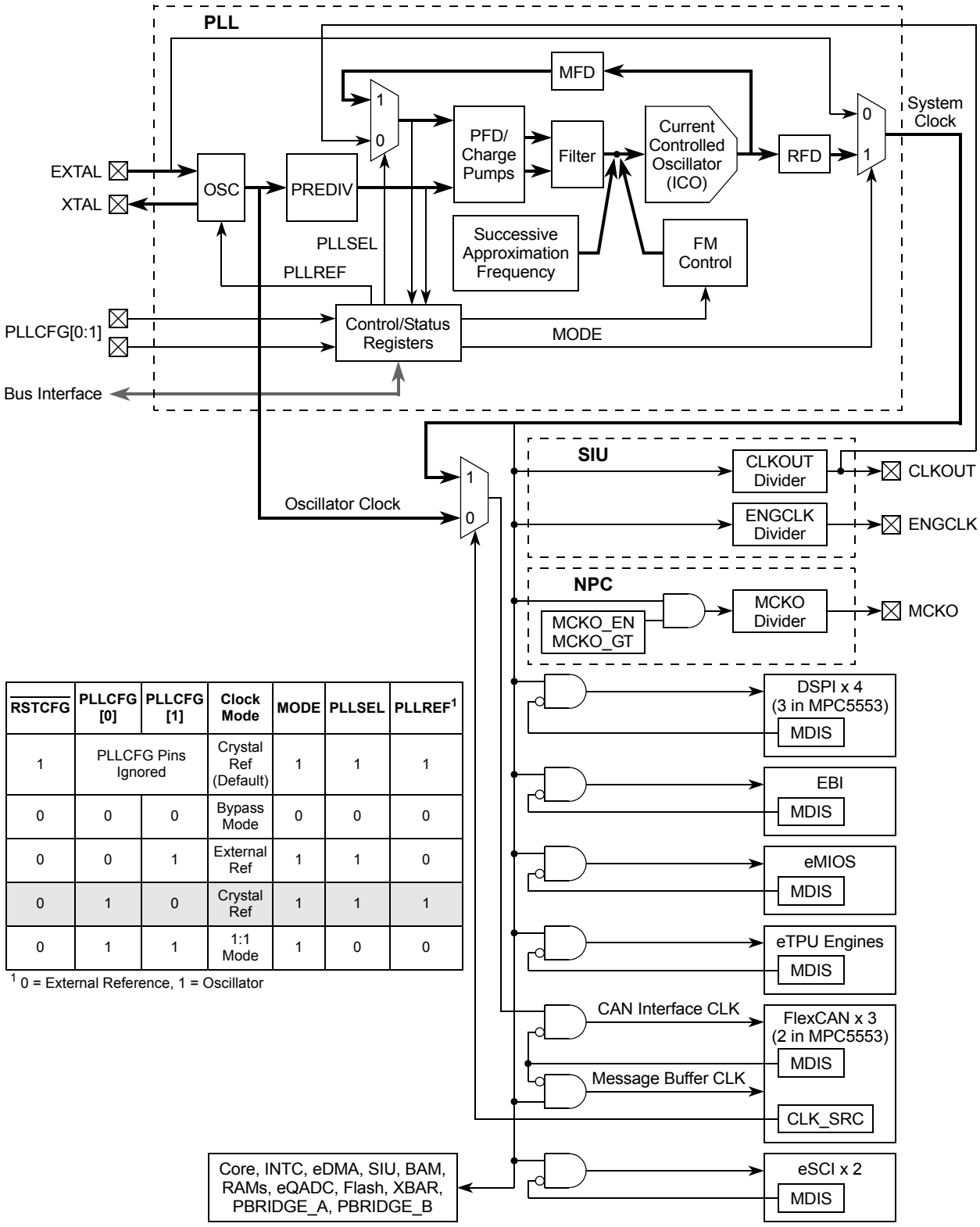


$\overline{\text{RSTCFG}}$	PLLCFG [0]	PLLCFG [1]	Clock Mode	MODE	PLLSEL	PLLREF[1]
1	PLLCFG Pins Ignored		Crystal Ref (Default)	1	1	1
0	0	0	Bypass Mode	0	0	0
0	0	1	External Ref	1	1	0
0	1	0	Crystal Ref	1	1	1
0	1	1	1:1 Mode	1	0	0

[1] 0 = External Reference, 1 = Oscillator

Figure 11-4. FMPLL Crystal Reference Mode without FM

11.1.1.5 FMPLL Crystal Reference Mode With FM

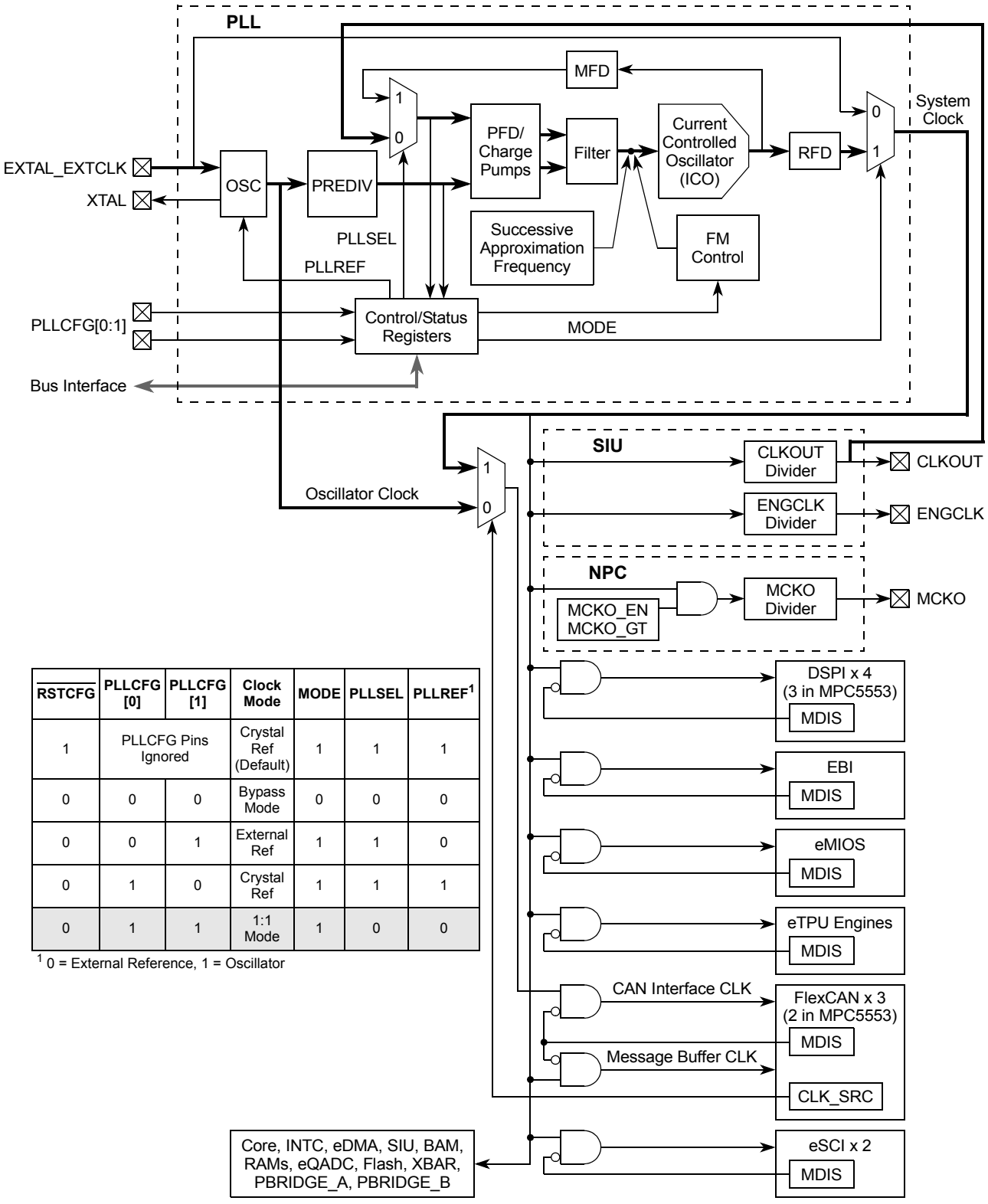


$\overline{\text{RSTCFG}}$	PLLCFG [0]	PLLCFG [1]	Clock Mode	MODE	PLLSEL	PLLREF[1]
1	PLLCFG Pins Ignored		Crystal Ref (Default)	1	1	1
0	0	0	Bypass Mode	0	0	0
0	0	1	External Ref	1	1	0
0	1	0	Crystal Ref	1	1	1
0	1	1	1:1 Mode	1	0	0

[1] 0 = External Reference, 1 = Oscillator

Figure 11-5. FMPLL Crystal Reference Mode with FM

11.1.1.6 FMPLL Dual-Controller Mode (1:1)



RSTCFG	PLLCFG [0]	PLLCFG [1]	Clock Mode	MODE	PLLSEL	PLLREF[1]
1	PLLCFG Pins Ignored		Crystal Ref (Default)	1	1	1
0	0	0	Bypass Mode	0	0	0
0	0	1	External Ref	1	1	0
0	1	0	Crystal Ref	1	1	1
0	1	1	1:1 Mode	1	0	0

[1] 0 = External Reference, 1 = Oscillator

Figure 11-6. FMPLL Dual Controller (1:1) Mode

11.1.2 Overview

The frequency modulated phase locked loop (FMPLL) allows the user to generate high speed system clocks from an 8 MHz to 20 MHz crystal oscillator or external clock generator. Further, the FMPLL supports programmable frequency modulation of the system clock. The FMPLL multiplication factor, output clock divider ratio, modulation depth, and modulation rate are all controllable through a bus interface.

11.1.3 Features

The FMPLL has the following major features:

- Input clock frequency from 8 MHz to 20 MHz
- Current controlled oscillator (ICO) range from 48 MHz to 132 MHz
- Reference frequency pre-divider (PREDIV) for finer frequency synthesis resolution
- Reduced frequency divider (RFD) for reduced frequency operation without forcing the FMPLL to re-lock
- Four modes of operation:
  - Bypass mode.
  - Crystal reference mode (default mode for MPC5554 and 324 and 416 packages of the MPC5553). Refer to Section 11.1.4.1, “Crystal Reference (Default Mode).”
  - External reference mode. Refer to Section 11.1.4.2, “External Reference Mode.”
  - PLL dual-controller (1:1) mode for EXTAL_EXTCLK to CLKOUT skew minimization.
- Programmable frequency modulation
  - Modulation enabled/disabled via bus interface
  - Triangle wave modulation
  - Register programmable modulation depth (+/-1%-2% deviation from center frequency)
  - Register programmable modulation frequency dependent on reference frequency
- Lock detect circuitry reports when the FMPLL has achieved frequency lock and continuously monitors lock status to report loss of lock conditions
  - User-selectable ability to generate an interrupt request upon loss of lock. (See Chapter 10, “Interrupt Controller (INTC),” for details.)
  - User-selectable ability to generate a system reset upon loss of lock. (See Chapter 4, “Reset,” for details.)
- Loss of clock (LOC) detection for reference and feedback clocks
  - User-selectable ability to generate an interrupt request upon loss of clock. (See Chapter 10, “Interrupt Controller (INTC),” for details.)
  - User-selectable ability to generate a system reset upon loss of clock (See Chapter 4, “Reset,” for details.)
- Self-clocked mode (SCM) operation in event of input clock failure

11.1.4 FMPLL Modes of Operation

The FMPLL operational mode is configured during reset. For the MPC5554, the FMPLL mode defaults to crystal reference mode. The 324 and 416 package sizes of the MPC5553 also default to crystal reference mode. For the MPC5554 and the 324 and 416 package sizes of the MPC5553, if the user should desire to change from this mode, the RSTCFG and PLLCFG[0:1] package pins must be driven to the appropriate state for the desired mode from the time RSTOUT asserts until it negates. As shown in Table 11-1, if RSTCFG is not asserted during reset, the state of the PLLCFG package pins is ignored, and the FMPLL will operate in the default crystal reference mode. The table also shows that to enter any other mode RSTCFG must be asserted during reset.

Note that because the 208 package size of the MPC5553 has no RSTCFG pin, after reset the 208 resets to the values of PLLCFG before reset. The device does not reset to the crystal reference mode as do the other MPC5553/MPC5554 packages.

Table 11-1 shows clock mode selection during reset configuration for the MPC5554 and for the 416 and 324 pin packages of the MPC5553. Additional information on reset configuration options for the FMPLL can be found in Chapter 4, "Reset."

Table 11-1. Clock Mode Selection in 416 Pin and 324 Pin Packages

Package Pins			Clock Mode	Synthesizer Status Register (FMPLL_SYNSR)[1] Bits		
RSTCFG	PLLCFG[0]	PLLCFG[1]		MODE	PLLSEL	PLLREF
1	PLLCFG pins ignored.		Crystal reference (default mode)	1	1	1
0	1	0				
0	0	1	External reference	1	1	0
0	0	0	Bypass Mode	0	0	0
0	1	1	Dual-Controller Mode	1	0	0

[1] See Section 11.3.1.2, "Synthesizer Status Register (FMPLL_SYNSR)" for more information on these bits.

Table 11-2 shows clock mode selection for the MPC5553 208 pin package.

Table 11-2. Clock Mode Selection in 208 Pin Package

Package Pins		Clock Mode	Synthesizer Status Register (FMPLL_SYNSR)[1] Bits		
PLLCFG[0]	PLLCFG[1]		MODE	PLLSEL	PLLREF
1	0	Crystal Reference	1	1	1
0	1	External Reference	1	1	0
0	0	Bypass	0	0	0
1	1	Dual-Controller	1	0	0

1 See Section 11.3.1.2, "Synthesizer Status Register (FMPLL_SYNSR)" for more information on these bits.

11.1.4.1 Crystal Reference (Default Mode)

In crystal reference mode, the FMPLL receives an input clock frequency (EXTAL) from the crystal oscillator circuit (and, in the MPC5553, the pre-divider) and multiplies the frequency to create the FMPLL output clock. The user must supply a crystal oscillator that is within the appropriate input frequency range, the crystal manufacture's recommended external support circuitry, and short signal route from the MCU to the crystal.

The external support circuitry for the crystal oscillator is shown in Figure 11-7. Example component values are shown as well. Note that the actual circuit should be reviewed with the crystal manufacturer. A block diagram illustrating crystal reference mode is shown in Figure 11-4.

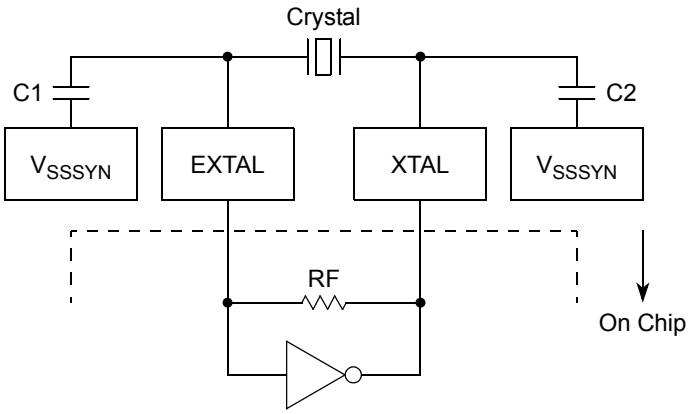


Figure 11-7. Crystal Oscillator Network

In crystal reference mode, the FMPLL can generate a frequency modulated clock or a non-modulated clock (locked on a single frequency). The modulation rate, modulation depth, output clock divide ratio (RFD), and whether the FMPLL is modulating or not can be programmed by writing to the FMPLL registers. Crystal reference is the default clock mode for the MPC5554 and the 324 and 416 packages of the MPC5553. It is not necessary to force PLLCFG[0:1] to enter this mode. In the 208 package size, because it has no RSTCFG pin, the crystal reference mode can only be selected through the PLLCFG pins.

11.1.4.2 External Reference Mode

This external reference mode functions the same as crystal reference mode except that EXTAL_EXTCLK is driven by an external clock generator rather than a crystal oscillator. Also, the input frequency range in external reference mode is the same as in the crystal reference mode. To enter external clock mode, the default FMPLL configuration must be overridden by following the procedure outlined in Section 11.1.4, "FMPLL Modes of Operation." A block diagram illustrating external reference mode is shown in Figure 11-3.

NOTE

In addition to supplying power for the CLKOUT signal, when the FMPLL is configured for external clock mode of operation, the V_{DDE5} supply voltage also controls the voltage level at which the signal presented to the EXTAL_EXTCLK pin causes a switch in the clock logic levels. The EXTAL_EXTCLK will accept a clock source with a voltage range of 1.6V to 3.6V, however the transition voltage is determined by V_{DDE5} supply voltage divided by 2. As an example, if V_{DDE5} is 3.3V, then the clock will transition at approximately 1.6V. The V_{DDE5} supply voltage and the voltage level of the external clock reference must be compatible, or the device will not clock properly.

11.1.4.3 Bypass Mode

In FMPLL bypass mode, the FMPLL is completely bypassed and the user must supply an external clock on the EXTAL_EXTCLK pin. The external clock is used directly to produce the internal system clocks. In bypass mode, the analog portion of the FMPLL is disabled and no clocks are generated at the FMPLL output. Consequently, frequency modulation is not available. In bypass mode the pre-divider is bypassed and has no effect on the system clock.

To enter bypass mode, the default FMPLL configuration must be overridden by following the procedure outlined in Section 11.1.4, "FMPLL Modes of Operation." A block diagram illustrating bypass mode is shown in Figure 11-2.

11.1.4.4 Dual-Controller Mode (1:1)

FMPLL dual-controller mode is used by the slave MCU device of a dual-controller system. The slave FMPLL will facilitate skew reduction between the input and output clock signals. To enter dual-controller mode, the default FMPLL configuration must be overridden by the procedure outlined in Section 11.1.4, "FMPLL Modes of Operation."

In this mode, the system clock runs at twice the frequency of the EXTAL_EXTCLK input pin and is phase aligned. Note that crystal operation is not supported in dual-controller mode and an external clock must be provided. In this mode, the frequency and phase of the signal at the EXTAL_EXTCLK pin and the CLKOUT pin of the slave MCU are matched. A block diagram illustrating dual-controller mode (1:1) is shown in Figure 11-6.

Frequency modulation is not available when configured for dual-controller mode for both the master and slave devices. Enabling frequency modulation on the device supplying the reference clock to the slave in dual-controller mode will produce unreliable clocks on the slave.

NOTE

When configured for dual-controller mode, the CLKOUT clock divider on the slave device must not be changed from its reset state of divide-by-2. Increasing or decreasing this divide ratio will produce unpredictable results from the FMPLL.

11.2 External Signal Description

Table 11-3 lists external signals used by the FMPLL during normal operation.

Table 11-3. PLL External Pin Interface

Name	I/O Type	Function	Pull
PLLCFG0_GPIO208	I/O	Configures the mode during reset. GPIO used otherwise.	Up
PLLCFG1_GPIO209	I/O	Configures the mode during reset. GPIO used otherwise.	Up
XTAL	Output	Output drive for external crystal	—
EXTAL_EXTCLK	Input	Crystal/external clock input	—
V_{DDSYN}	Power	Analog power supply (3.3V +/–10%)	—
V_{SSSYN}	Ground	Analog ground	—

11.3 Memory Map/Register Definition

Table 11-4 shows the FMPLL memory map locations.

Table 11-4. FMPLL Module Memory Map

Address	Register Name	Register Description	Size (bits)
Base (0xC3F8_0000)	FMPLL_SYNCR	Synthesizer control register	32
Base + 0x04	FMPLL_SYNSR	Synthesizer status register	32
Base + 0x08	—	Reserved	—
Base + 0x0C	—	Reserved	—
Base + 0x10	—	Reserved	—
Base + 0x14	—	Reserved	—
Base + 0x18	—	Reserved	—
Base + 0x1C	—	Reserved	—

11.3.1 Register Descriptions

The clock operation is controlled by the synthesizer control register (FMPLL_SYNCR) and status is reported in the synthesizer status register (FMPLL_SYNSR). The following sections describe these registers in detail.

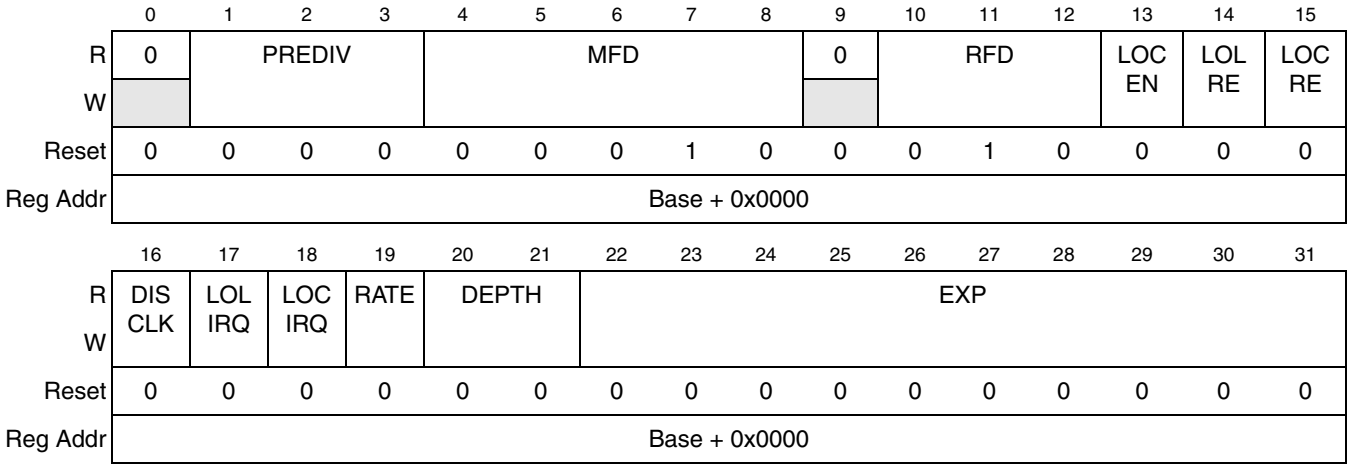
11.3.1.1 Synthesizer Control Register (FMPLL_SYNCR)

The synthesizer control register (FMPLL_SYNCR) contains bits for defining the clock operation for the system.

NOTE

To ensure proper operation across all MPC5500 MCUs, execute an mbar or msync instruction between the write to change the FMPLL_SYNCR[MFD] and the read to check the lock status shown by FMPLL_SYNSR[LOCK].

Furthermore, buffering writes to the FMPLL, as controlled by PBRIDGE_A_OPACR[BW0], must be disabled.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PREDIV			MFD					0	RFD			LOC EN	LOL RE	LOC RE
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0
Reg Addr	Base + 0x0000															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DIS CLK	LOL IRQ	LOC IRQ	RATE	DEPTH		EXP									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0000															

Figure 11-8. Synthesizer Control Register (FMPLL_SYNCR)

Table 11-5. FMPLL_SYNCR Field Descriptions

Bits	Name	Description
0	—	Reserved.
1–3	PREDIV [0:2]	The PREDIV bits control the value of the divider on the input clock. The output of the pre-divider circuit generates the reference clock to the FMPLL analog loop. When the PREDIV bits are changed, the FMPLL will immediately lose lock. To prevent an immediate reset, the LOLRE bit must be cleared before writing the PREDIV bits. In 1:1 (dual-controller) mode, the PREDIV bits are ignored and the input clock is fed directly to the analog loop. 000 Divide by 1 001 Divide by 2 010 Divide by 3 011 Divide by 4 100 Divide by 5 101–111Reserved **Note:** Programming a PREDIV value such that the ICO operates outside its specified range will cause unpredictable results and the FMPLL will not lock. Refer to the *MPC5553/MPC5554 Data Sheet* for details on the ICO range. **Note:** To avoid unintentional interrupt requests, disable LOLIRQ before changing PREDIV and then reenable it after acquiring lock.

Table 11-5. FMPLL_SYNCR Field Descriptions (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>4–8</td><td>MFD
[0:4]</td><td>Multiplication factor divider. The MFD bits control the value of the divider in the FMPLL feedback loop. The value specified by the MFD bits establish the multiplication factor applied to the reference frequency. The decimal equivalent of the MFD binary number is substituted into the equation from Table 11-10 for F_{sys} to determine the equivalent multiplication factor.
When the MFD bits are changed, the FMPLL loses lock. At this point, if modulation is enabled, the calibration sequence is reinitialized. To prevent an immediate reset, the LOLRE bit must be cleared before writing the MFD bits. In dual-controller mode, the MFD bits are ignored and the multiplication factor is equivalent to 2X. In bypass mode the MFD bits have no effect.
Note: Programming an MFD value such that the ICO operates outside its specified range will cause unpredictable results and the FMPLL will not lock. Refer to the MPC5553/MPC5554 Data Sheet for details on the ICO range.
Note: To avoid unintentional interrupt requests, disable LOLIRQ before changing MFD and then reenable it after acquiring lock.</td></tr>
<tr><td>9</td><td>—</td><td>Reserved.</td></tr>
<tr><td>10–12</td><td>RFD
[0:2]</td><td>Reduced frequency divider. The RFD bits control a divider at the output of the FMPLL. The value specified by the RFD bits establish the divisor applied to the FMPLL frequency.

<table><tr><th>RFD[0:2]</th><th>Output Clock Divide Ratio</th></tr><tr><td>000</td><td>Divide by 1</td></tr><tr><td>001</td><td>Divide by 2</td></tr><tr><td>010</td><td>Divide by 4</td></tr><tr><td>011</td><td>Divide by 8</td></tr><tr><td>100</td><td>Divide by 16</td></tr><tr><td>101</td><td>Divide by 32</td></tr><tr><td>110</td><td>Divide by 64</td></tr><tr><td>111</td><td>Divide by 128</td></tr></table>
Changing the RFD bits does not affect the FMPLL; hence, no re-lock delay is incurred. Resulting changes in clock frequency are synchronized to the next falling edge of the current system clock. However these bits must only be written when the lock bit (LOCK) is set, to avoid exceeding the allowable system operating frequency. In bypass mode, the RFD bits have no effect.</td></tr>
<tr><td>13</td><td>LOCEN</td><td>Loss-of-clock enable. The LOCEN bit determines whether the loss of clock function is operational. See Section 11.4.2.6, “Loss-of-Clock Detection” and Section 11.4.2.6.1, “Alternate/Backup Clock Selection” for more information.
In bypass mode, this bit has no effect.
LOCEN does not affect the loss of lock circuitry.
0 Loss of clock disabled.
1 Loss of clock enabled.</td></tr>
</table>

Table 11-5. FMPLL_SYNCR Field Descriptions (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>14</td><td>LOLRE</td><td>Loss-of-lock reset enable. The LOLRE bit determines how the integration module (the SIU) handles a loss of lock indication. When operating in crystal reference, external reference, or dual-controller mode, the FMPLL must be locked before setting the LOLRE bit. Otherwise reset is immediately asserted. The LOLRE bit has no effect in bypass mode.
0 Ignore loss of lock - reset not asserted.
1 Assert reset on loss of lock. Reset will remain asserted, regardless of the source of reset, until after the FMPLL has locked.</td></tr>
<tr><td>15</td><td>LOCRE</td><td>Loss-of-clock reset enable. The LOCRE bit determines how the integration module (the SIU) handles a loss of clock condition when LOCEN=1. LOCRE has no effect when LOCEN=0. If the LOCF bit in the SYNSR indicates a loss of clock condition, setting the LOCRE bit causes an immediate reset. In bypass mode LOCRE has no effect.
0 Ignore loss of clock - reset not asserted.
1 Assert reset on loss of clock.</td></tr>
<tr><td>16</td><td>DISCLK</td><td>Disable CLKOUT. The DISCLK bit determines whether CLKOUT is active. When CLKOUT is disabled it is driven low.
0 CLKOUT driven normally
1 CLKOUT driven low</td></tr>
<tr><td>17</td><td>LOLIRQ</td><td>Loss-of-lock interrupt request. The LOLIRQ bit enables an interrupt request for LOLF when it (LOLIRQ) is asserted and when LOLF is asserted. If either LOLF or LOLIRQ is negated, the interrupt request is negated. When operating in crystal reference, external reference, or dual-controller mode, the FMPLL must be locked before setting the LOLIRQ bit. Otherwise an interrupt is immediately requested. The LOLIRQ bit has no effect in bypass mode

0 Ignore loss of lock - interrupt not requested
1 Request interrupt</td></tr>
<tr><td>18</td><td>LOCIRQ</td><td>Loss-of-clock interrupt request. The LOCIRQ bit determines how the integration module (the SIU) handles a loss of clock condition when LOCEN=1. LOCIRQ has no effect when LOCEN=0. If the LOCF bit in the SYNSR indicates a loss of clock condition, setting (or having previously set) the LOCIRQ bit causes an interrupt request. In bypass mode LOCIRQ has no effect.
0 Ignore loss of clock - interrupt not requested
1 Request interrupt on loss of clock.</td></tr>
<tr><td>19</td><td>RATE</td><td>Modulation rate. Controls the rate of frequency modulation applied to the system frequency. The allowable modulation rates are shown below. Changing the rate by writing to the RATE bit will initiate the FM calibration sequence.

<table>
<tr><th>RATE</th><th>Modulation Rate (Hz)</th></tr>
<tr><td>0</td><td>FM = F_{ref} / 80</td></tr>
<tr><td>1</td><td>FM = F_{ref} / 40</td></tr>
</table>
Note: To avoid unintentional interrupt requests, clear LOLIRQ before changing RATE.</td></tr>
</table>

Table 11-5. FMPLL_SYNCR Field Descriptions (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>20–21</td><td>DEPTH
[0:1]</td><td>Controls the frequency modulation depth and enables the frequency modulation. When programmed to a value other than 0x0, the frequency modulation is automatically enabled. The programmable frequency deviations from the system frequency are shown below. Upon a change in the depth value to other than 0x0, the calibration sequence will be reinitialized.
<table>
<tr><th>DEPTH[1]</th><th>DEPTH[0]</th><th>Modulation Depth (% of Fsys)</th></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1.0 ± 0.2</td></tr>
<tr><td>1</td><td>0</td><td>2.0 ± 0.2</td></tr>
<tr><td>1</td><td>1</td><td>Reserved</td></tr>
</table>
Note: To avoid unintentional interrupt requests, clear LOLIRQ before changing DEPTH.</td></tr>
<tr><td>22–31</td><td>EXP
[0:9]</td><td>Expected difference value. Holds the expected value of the difference of the reference and the feedback counters. See Section 11.4.3.3, “FM Calibration Routine” to determine the value of these bits. This field is written by the application before entering calibration mode.</td></tr>
</table>

11.3.1.2 Synthesizer Status Register (FMPLL_SYNSR)

The synthesizer status register (FMPLL_SYNSR) is a 32-bit register. Only the LOLF and LOCF flag bits are writable in this register. Writes to bits other than the LOLF and LOCF have no effect.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0004															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	LOLF	LOC	MODE	PLL SEL	PLL REF	LOCKS	LOCK	LOCF	CALD ONE	CAL PASS
W							w1c							w1c		
Reset	0	0	0	0	0	0	0	0	—[1]	—[1]	—[1]	—[1]	—[2]	0	0	0
Reg Addr	Base + 0x0004															

1 Reset state determined during reset configuration. (See Section 11.1.4, “FMPLL Modes of Operation,” for more information.)

2 Reset state determined during reset.

3 “w1c” signifies that this bit is cleared by writing a 1 to it.

Figure 11-9. Synthesizer Status Register (FMPLL_SYNSR)

Table 11-6. FMPLL_SYNSR Field Descriptions

Bits	Name	Description
0–21	—	Reserved.
22	LOLF	Loss-of-lock flag. Provides the interrupt request flag. This is a write 1 to clear (w1c) bit; to clear the flag, the user must write a 1 to the bit. Writing 0 has no effect. This flag will not be set, and an interrupt will not be requested, if the loss of lock condition was caused by a system reset, a write to the FMPLL_SYNCR which modifies the MFD bits, or enabling frequency modulation. If the flag is set due to a system failure, writing the MFD bits or enabling FM will not clear the flag. Asserting reset will clear the flag. This flag bit is sticky in the sense that if lock is reacquired, the bit will remain set until either a write 1 or reset is asserted. 0 Interrupt service not requested 1 Interrupt service requested **Note:** Upon a loss of lock that is not generated by a system reset, a write to the FMPLL_SYNCR that modifies the MFD or PREDIV bits, or an enabling of frequency modulation, the LOLF will be set *if* LOLIRQ is set. If the FMPLL reacquires lock and at that point either of the three above steps are executed, the LOLF will again be set. To avoid an unintentional interrupt from being generated, LOLIRQ must be cleared prior to changing MFD or PREDIV, or prior to enabling FM after a previous interrupt and relock occurred.
23	LOC	Loss-of-clock status. Indicates whether a loss-of-clock condition is present when operating in crystal reference, external reference, or dual-controller mode, If LOC = 0, the system clocks are operating normally. If LOC = 1, the system clocks have failed due to a reference failure or a FMPLL failure. If the read of the LOC bit and the loss-of-clock condition occur simultaneously, the bit does not reflect the current loss of clock condition. If a loss-of-clock condition occurs which sets this bit and the clocks later return to normal, this bit will be cleared. A loss of clock condition can only be detected if LOCEN = 1. LOC is always 0 in bypass mode. 0 Clocks are operating normally 1 Clocks are not operating normally.
24	MODE	Clock mode. Determined at reset, this bit indicates which clock mode the system is utilizing. See Chapter 4, “Reset,” for details on how to configure the system clock mode during reset. 0 PLL bypass mode. 1 PLL clock mode.
25	PLLSEL	PLL mode select. Determined at reset, this bit indicates in which mode the FMPLL operates. This bit is cleared in dual-controller and bypass mode. See Chapter 4, “Reset,” for details on how to configure the system clock mode during reset. See Table 11-1 for more information. 0 Dual-controller mode. 1 Crystal reference or external reference mode.
26	PLLREF	PLL clock reference source. Determined at reset, this bit indicates whether the PLL reference source is an external clock or a crystal reference. This bit is cleared in dual controller mode and bypass mode. See Chapter 4, “Reset,” for details on how to configure the system clock mode during reset. 0 External clock reference chosen. 1 Crystal clock reference chosen.

Table 11-6. FMPLL_SYNSR Field Descriptions (continued)

Bits	Name	Description
27	LOCKS	Sticky FMPLL lock status bit. A sticky indication of FMPLL lock status. LOCKS is set by the lock detect circuitry when the FMPLL acquires lock after one of the following: • a system reset • a write to the FMPLL_SYNCR which modifies the MFD bits • the enabling of frequency modulation Whenever the FMPLL loses lock, LOCKS is cleared. LOCKS remains cleared even after the FMPLL relocks, until one of the three previously-stated conditions occurs. Furthermore, if the LOCKS bit is read when the FMPLL simultaneously loses lock, the bit does not reflect the current loss of lock condition. If operating in bypass mode, LOCKS remains cleared after reset. In crystal reference, external reference, and dual-controller mode, LOCKS is set after reset. 0 PLL has lost lock since last system reset, a write to FMPLL_SYNCR to modify the MFD bit field, or frequency modulation enabled. 1 PLL has not lost lock since last system reset, a write to FMPLL_SYNCR to modify the MFD bit field, or frequency modulation enabled.
28	LOCK	PLL lock status bit. Indicates whether the FMPLL has acquired lock. FMPLL lock occurs when the synthesized frequency matches to within approximately 0.75% of the programmed frequency. The FMPLL loses lock when a frequency deviation of greater than approximately 1.5% occurs. If the LOCK bit is read when the FMPLL simultaneously loses lock or acquires lock, the bit does not reflect the current condition of the FMPLL. If operating in bypass mode, LOCK remains cleared after reset. 0 PLL is unlocked. 1 PLL is locked.
29	LOCF	Loss-of-clock flag. This bit provides the interrupt request flag. This is a write 1 to clear (w1c) bit; to clear the flag, the user must write a 1 to the bit. Writing 0 has no effect. Asserting reset will clear the flag. This flag is sticky in the sense that if clocks return to normal after the flag has been set, the bit will remain set until cleared by either writing 1 or asserting reset. 0 Interrupt service not requested 1 Interrupt service requested
30	CALDONE	Calibration complete. Indicates whether the calibration sequence has been completed since the last time modulation was enabled. If CALDONE = 0 then the calibration sequence is either in progress or modulation is disabled. If CALDONE = 1 then the calibration sequence has been completed, and frequency modulation is operating. 0 Calibration not complete. 1 Calibration complete. **Note:** FM relocking does not start until calibration is complete.
31	CALPASS	Calibration passed. Indicates whether the calibration routine was successful. If CALPASS = 1 and CALDONE = 1 then the routine was successful. If CALPASS = 0 and CALDONE = 1, then the routine was unsuccessful. When the calibration routine is initiated the CALPASS is asserted. CALPASS remains asserted until either modulation is disabled by clearing the DEPTH bits in the FMPLL_SYNCR or a failure occurs within the FMPLL calibration sequence. 0 Calibration unsuccessful. 1 Calibration successful. If calibration is unsuccessful, then actual depth is not guaranteed to match the desired depth.

11.4 Functional Description

This section explains clock architecture, clock operation, and clock configuration.

11.4.1 Clock Architecture

This section describes the clocks and clock architecture in the MPC5553/MPC5554 MCU.

11.4.1.1 Overview

The MPC5553/MPC5554 system clocks are generated from one of four FMPLL modes: crystal reference mode, external reference mode, dual-controller (1:1) mode, and bypass mode. See Section 11.1, "Introduction" for information on the different clocking modes available in the MPC5553/MPC5554 FMPLL.

The MPC5553/MPC5554 peripheral IP modules have been designed to allow software to gate the clocks to the non-memory-mapped logic of the modules.

The MPC5553/MPC5554 MCU has three clock output pins that are driven by programmable clock dividers. The clock dividers divide the system clock down by even integer values. The three clock output pins are the following:

- CLKOUT – External address/data bus clock
- MCKO – Nexus auxiliary port clock
- ENGCLK – Engineering clock

The MPC5553/MPC5554 MCU has been designed so that the oscillator clock can be selected as the clock source for the CAN interface in the FlexCAN blocks resulting in very low jitter performance.

Figure 11-1 shows a block diagram of the FMPLL and the system clock architecture.

11.4.1.2 Software Controlled Power Management/Clock Gating

Some of the IP modules on MPC5553/MPC5554 support software controlled power management/clock gating whereby the application software can disable the non-memory-mapped portions of the modules by writing to module disable (MDIS) bits in registers within the modules. The memory-mapped portions of the modules are clocked by the system clock when they are being accessed. The NPC can be configured to disable the MCKO signal when there are no Nexus messages pending. The H7FA Flash array can be disabled by writing to a bit in the Flash register map.

The modules that implemented software controlled power management/clock gating are listed in Table 11-7 along with the registers and bits that disable each module. The software controlled clocks are enabled when the MPC5553/MPC5554 MCU comes out of reset.

Table 11-7. Software Controlled Power Management/Clock Gating Support

Module Name	Register Name	Bit Names
DSPI A[1]	DSPI_A_MCR	MDIS
DSPI B	DSPI_B_MCR	MDIS
DSPI C	DSPI_C_MCR	MDIS
DSPI D	DSPI_D_MCR	MDIS
EBI	EBI_MCR	MDIS

Table 11-7. Software Controlled Power Management/Clock Gating Support

Module Name	Register Name	Bit Names
eTPU Engine A	ETPUECR_1	MDIS
eTPU Engine B	ETPUECR_2	MDIS
FlexCAN A	FLEXCAN_A_MCR	MDIS
FlexCAN B[1]	FLEXCAN_B_MCR	MDIS
FlexCAN C	FLEXCAN_C_MCR	MDIS
EMIOS	EMIOS_MCR	MDIS
ESCI_A	ESCIA_CR2	MDIS
ESCI_B	ESCIB_CR2	MDIS
NPC	NPC_PCR	MCKO_EN, MCKO_GT[2]
Flash Array	FLASH_MCR	STOP[3]

1 Shaded areas indicate that module is only offered on the MPC5554, not on the MPC5553.

2 See Chapter 25, “Nexus Development Interface.”

3 See Chapter 13, “Flash Memory.”

11.4.1.3 Clock Dividers

Each of the CLKOUT, MCKO, and ENGCLK dividers provides a nominal 50% duty cycle clock to an output pin. There is no guaranteed phase relationship between CLKOUT, MCKO, and ENGCLK. ENGCLK is not synchronized to any I/O pins.

11.4.1.3.1 External Bus Clock (CLKOUT)

The external bus clock (CLKOUT) divider can be programmed to divide the system clock by two or four based on the settings of the EBDF bit field in the SIU external clock control register (SIU_ECCR). The reset value of the EBDF selects a CLKOUT frequency of one half of the system clock frequency. The EBI supports gating of the CLKOUT signal when there are no external bus accesses in progress. See the Chapter 6, “System Integration Unit (SIU)” for more information on CLKOUT.

The hold-time for the external bus pins can be changed by writing to the external bus tap select (EBTS) bit in the SIU_ECCR. See Chapter 6, “System Integration Unit (SIU)” for more information.

11.4.1.3.2 Nexus Message Clock (MCKO)

The Nexus message clock (MCKO) divider can be programmed to divide the system clock by two, four or eight based on the MCKO_DIV bit field in the port configuration register (PCR) in the Nexus port controller (NPC). The reset value of the MCKO_DIV selects an MCKO clock frequency one half of the system clock frequency. The MCKO divider is configured by writing to the NPC through the JTAG port. See Chapter 25, “Nexus Development Interface” for more information.

11.4.1.3.3 Engineering Clock (ENGCLK)

The engineering clock (ENGCLK) divider can be programmed to divide the system clock by factors from 2 to 128 in increments of two. The ENGDIV bit field in the SIU_ECCR determines the divide factor. The reset value of ENGDIV selects an ENGCLK frequency of system clock divided by 32.

11.4.1.3.4 FlexCAN_*x* Clock Domains

The FlexCAN modules have two distinct software controlled clock domains. One of the clock domains is always derived from the system clock. This clock domain includes the message buffer logic. The source for the second clock domain can be either the system clock or a direct feed from the oscillator pin EXTAL_EXTCLK. The logic in the second clock domain controls the CAN interface pins. The CLK_SRC bit in the FlexCAN CTRL register selects between the system clock and the oscillator clock as the clock source for the second domain. Selecting the oscillator as the clock source ensures very low jitter on the CAN bus. System software can gate both clocks by writing to the MDIS bit in the FlexCAN MCR register. Figure 11-1 shows the two clock domains in the FlexCAN modules.

See Chapter 22, "FlexCAN2 Controller Area Network" for more information on the FlexCAN modules.

11.4.1.3.5 FEC Clocks

In the MPC5553, the FEC TX_CLK and RX_CLK are inputs. An external source provides the clocks to these pins.

11.4.2 Clock Operation

11.4.2.1 Input Clock Frequency

The FMPLL is designed to operate over an input clock frequency range as determined by the operating mode. The operating ranges for each mode are given in Table 11-8.

Table 11-8. Input Clock Frequency

Mode	Input Frequency Range
Crystal Reference External Reference	8 MHz –20 MHz
Bypass	0 Hz–132MHz
Dual-Controller (1:1)	25 MHz–66 MHz

11.4.2.2 Reduced Frequency Divider (RFD)

The RFD may be used for reducing the FMPLL system clock frequency. The RFD must be programmed to be ≥1 when changing MFD or PREDIV or when enabling frequency modulation.

11.4.2.3 Programmable Frequency Modulation

The FMPLL provides for frequency modulation of the system clock. The modulation is applied as a triangular waveform with modulation depth and rate controlled by fields in the FMPLL_SYNCR. The modulation depth can be set to +/-1% or +/-2% of the system frequency. The modulation rate is dependent on the reference clock frequency.

Complete details for configuring the programmable frequency modulation is given in Section 11.4.3.2, "Programming System Clock Frequency with Frequency Modulation."

11.4.2.4 FMPLL Lock Detection

A pair of counters monitor the reference and feedback clocks to determine when the system has acquired frequency lock. Once the FMPLL has locked, the counters continue to monitor the reference and feedback clocks and will report if/when the FMPLL has lost lock. The FMPLL_SYNCR provides the flexibility to select whether to generate an interrupt, assert system reset, or do nothing in the event that the FMPLL loses lock. See Section 11.3.1.1, "Synthesizer Control Register (FMPLL_SYNCR) for details.

When the frequency modulation is enabled, the loss of lock continues to function as described but with the lock and loss of lock criteria reduced to ensure that false loss of lock conditions are not detected.

In bypass mode, the FMPLL cannot lock since the FMPLL is disabled.

11.4.2.5 FMPLL Loss-of-Lock Conditions

Once the FMPLL acquires lock after reset, the FMPLL_SYNSR[LOCK] and FMPLL_SYNSR[LOCKS] status bits are set. If the MFD is changed or if an unexpected loss of lock condition occurs, the LOCK and LOCKS status bits are negated. While the FMPLL is in an unlocked condition, the system clocks continue to be sourced from the FMPLL as the FMPLL attempts to re-lock. Consequently, during the re-locking process, the system clock frequency is not well defined and may exceed the maximum system frequency thereby violating the system clock timing specifications (when changing MFD, this is avoided by following the procedure detailed in Section 11.4.3, "Clock Configuration"). Because this condition can arise during unexpected loss of lock events, it is recommended to use the loss of lock reset functionality, see Section 11.4.2.5.1, "FMPLL Loss-of-Lock Reset," below. However, LOLRE must be cleared while changing the MFD otherwise a reset will occur.

Once the FMPLL has relocked, the LOCK bit is set. The LOCKS bit remains cleared if the loss of lock was unexpected. The LOCKS bit is set to 1 when the loss of lock was caused by changing the MFD.

11.4.2.5.1 FMPLL Loss-of-Lock Reset

The FMPLL provides the ability to assert reset when a loss of lock condition occurs by programming the FMPLL_SYNCR[LOLRE] bit. Reset is asserted if LOLRE is set and loss of lock occurs. Because the FMPLL_SYNSR[LOCK] and FMPLL_SYNSR[LOCKS] bits are reinitialized after reset, the system reset status register (SIU_RSR) must be read to determine that a loss of lock condition occurred.

To exit reset, the reference must be present and the FMPLL must acquire lock. In bypass mode, the FMPLL cannot lock. Therefore a loss of lock condition cannot occur, and LOLRE has no effect.

11.4.2.5.2 FMPLL Loss-of-Lock Interrupt Request

The FMPLL provides the ability to request an interrupt when a loss of lock condition occurs by programming the FMPLL_SYNCR[LOLIRQ] bit. An interrupt is requested by the FMPLL if LOLIRQ is set and loss of lock occurs.

In bypass mode, the FMPLL cannot lock. Therefore a loss of lock condition cannot occur, and the LOLIRQ bit has no effect.

11.4.2.6 Loss-of-Clock Detection

The FMPLL continuously monitors the reference and feedback clocks. In the event either of the clocks fall below a threshold frequency, the system will report a loss of clock condition. The user may enable a feature to have the FMPLL switch the system clocks to a backup clock in the event of such a failure. Additionally, the user may select to have the system enter reset, assert an interrupt request, or do nothing if/when the FMPLL reports this condition.

11.4.2.6.1 Alternate/Backup Clock Selection

If the user enables loss of clock by setting FMPLL.SYNCR[LOCEN] =1, then the FMPLL will transition system clocks to a backup clock source in the event of a clock failure as per Table 11-9.

If loss of clock is enabled and the reference clock is the source of the failure, the FMPLL will enter self-clock mode (SCM). The exact frequency during self-clock mode operation is indeterminate due to process, voltage, and temperature variation but is guaranteed to be below the maximum system frequency. If the FMPLL clocks have failed, the FMPLL will transition the system clock source to the reference clock.

The FMPLL remains in SCM until the next reset. Note that when the FMPLL is operated in SCM the system frequency is dependent upon the value in RFD[0:2]. The SCM system frequency stated in the *MPC5553/MPC5554 Data Sheet* assumes that the RFD has been programmed to 0x0. If the loss-of-clock condition is due to a FMPLL failure (for example, loss of feedback clock), the FMPLL reference becomes the system clocks source until the next reset, even if the FMPLL regains itself and re-locks.

Table 11-9. Loss of Clock Summary

Clock Mode	System Clock Source before Failure	REFERENCE FAILURE Alternate Clock Selected by LOC Circuitry until Reset	PLL FAILURE Alternate Clock Selected by LOC Circuitry until Reset
PLL	PLL	PLL Self-Clocked Mode	PLL reference
PLL bypass	Ext. Clock(s)	None	NA

A special loss of clock condition occurs when both the reference and the FMPLL fail. The failures may be simultaneous or the FMPLL may fail first. In either case, the reference clock failure takes priority and the FMPLL attempts to operate in SCM. If successful, the FMPLL remains in SCM until the next reset. During SCM, modulation is always disabled. If the FMPLL cannot operate in SCM, the system remains static until the next reset. Both the reference and the FMPLL must be functioning properly to exit reset.

11.4.2.6.2 Loss-of-Clock Reset

When a loss of clock condition is recognized, reset is asserted if the FMPLL_SYNCR[LOCRE] bit is set. The LOCF and LOC bits in FMPLL_SYNSR are cleared after reset, therefore, the SIU_RSR must be read to determine that a loss of clock condition occurred. LOCRE has no effect in bypass mode.

To exit reset in FMPLL mode, the reference must be present and the FMPLL must acquire lock.

11.4.2.6.3 Loss-of-Clock Interrupt Request

When a loss of clock condition is recognized, the FMPLL will request an interrupt if the FMPLL_SYNCR[LOCIRQ] bit is set. The LOCIRQ bit has no effect in bypass mode or if FMPLL_SYNCR[LOCEN] = 0.

11.4.3 Clock Configuration

In crystal reference and external reference clock mode, the default system frequency is determined by the MFD, RFD, and PREDIV reset values. See Section 11.3.1.1, "Synthesizer Control Register (FMPLL_SYNCR)." The frequency multiplier is determined by the RFD, PREDIV, and multiplication frequency divisor (MFD) bits in FMPLL_SYNCR.

Table 11-10 shows the clock-out to clock-in frequency relationships for the possible clock modes.

Table 11-10. Clock-out vs. Clock-in Relationships

Clock Mode	PLL Option
Crystal Reference Mode and External Reference Mode (Frequency modulation disabled)	$F_{sys} = F_{ref} \bullet \frac{(MFD + 4)}{((PREDIV + 1) \times 2^{RFD})}$
Dual Controller (1:1) Mode	$F_{sys} = 2F_{ref}$
Bypass Mode	$F_{sys} = F_{ref}$

NOTES:
F_{sys} = System frequency
F_{ref} = Clock frequency at the EXTAL signal. (See Figure 11-1)
MFD ranges from 0 to 31
RFD ranges from 0 to 7
PREDIV normal reset value is 0. Caution: Programming a PREDIV value such that the ICO operates outside its specified range will cause unpredictable results and the FMPLL will not lock. Refer to the *MPC5553/MPC5554 Data Sheet* for details on the ICO range.

When programming the FMPLL, be sure not to violate the maximum system clocks frequency or max/min ICO frequency specifications. For determining the MFD value, RFD should be assumed zero (that is, divide by 1). This will insure that the FMPLL does not have to synthesize a frequency out of its range. See the *MPC5553/MPC5554 Data Sheet* for more information.

11.4.3.1 Programming System Clock Frequency Without Frequency Modulation

The following steps are required to accommodate the frequency overshoot that may occur when the PREDIV or MFD bits are changed. If frequency modulation is going to be enabled the maximum allowable frequency must be reduced by the programmed ΔF_m.

NOTE

Following these steps will produce immediate changes in supply current, thus the user should ensure that the power supply is sufficiently decoupled with low ESR capacitors.

Here are the steps to program the clock frequency without frequency modulation:

1. Determine the appropriate value for the PREDIV, MFD, and RFD fields in the synthesizer control register (FMPLL_SYNCR). Remember to include the ΔF_m if frequency modulation is to be enabled. Note that the amount of jitter in the system clocks can be minimized by selecting the

maximum MFD factor that can be paired with an RFD factor to provide the desired frequency. The maximum MFD value that can be used is determined by the ICO range. See the *MPC5553/MPC5554 Data Sheet* for the maximum frequency of the ICO.

2. Change the following in FMPLL_SYNCR:
   a) Make sure frequency modulation is disabled (FMPLL_SYNCR[DEPTH] = 00). A change to PREDIV, MFD, or RATE while modulation is enabled will invalidate the previous calibration results.
   b) Clear FMPLL_SYNCR[LOLRE]. If this bit is set, the MCU will go into reset when MFD is written.
   c) Initialize the FMPLL for less than the desired final system frequency:
   — Disable LOLIRQ.
   — Write FMPLL_SYNCR[PREDIV] for the desired final value.
   — Write FMPLL_SYNCR[MFD] for the desired final value.
   — Write the RFD control field for 1 + the desired final RFD value.
3. Wait for the FMPLL to lock by monitoring the FMPLL_SYNSR[LOCK] bit. Refer to Section 11.3.1.1, "Synthesizer Control Register (FMPLL_SYNCR)," for memory synchronization between changing FMPLL_SYNCR[MFD] and monitoring the lock status.
4. Initialize the FMPLL for the desired final system frequency by changing FMPLL_SYNCR[RFD] to its desired final value. Note that the FMPLL will not need to re-lock when only changing the RFD, and that RFD must be programmed to be >1 to protect from overshoot.
5. Re-enable LOLIRQ.

NOTE

This first register write will cause the FMPLL to switch to an initial system frequency which is less than the final one. Keeping the change of frequency to a lower initial value helps minimize the current surge to the external power supply caused by change of frequency. The last step will be to only change the RFD to get to the desired final frequency.

NOTE

Changing the MFD or PREDIV values causes the FMPLL to perform a search for the lock frequency that results in the system clock frequency changing rapidly across the complete frequency range. All MCU peripherals, including the external bus will be subjected to this frequency sweep. Operation of timers and serial communications during this search sequence will produce unpredictable results.

11.4.3.2 Programming System Clock Frequency with Frequency Modulation

In crystal reference and external reference clock modes, the default mode is without frequency modulation enabled. When frequency modulation is enabled, however, three parameters must be set to generate the desired level of modulation: the RATE, DEPTH, and EXP bit fields of the FMPLL_SYNCR. RATE and DEPTH determine the modulation rate and the modulation depth. The EXP field controls the FM calibration routine. Section 11.4.3.3, "FM Calibration Routine," shows how to obtain the values to be programmed for EXP. Figure 11-10 illustrates the effects of the parameters and the modulation waveform built into the modulation hardware. The modulation waveform is always a triangle wave and its shape is not programmable.

Note, the modulation rates given are specific to a reference frequency of 8 MHz.

$F_{mod} = F_{ref} / Q$ (PREDIV + 1) where Q = {40,80} giving modulation rates of 200 kHz and 100 kHz.

NOTE

The following relationship between F_{mod} and modulation rates must be maintained:

$$100 \text{ KHz} \leq F_{mod} \leq 250 \text{ KHz}$$

Therefore, the utilization of a non 8 MHz reference will result in scaled modulation rates.

Here are the steps to program the clock frequency with frequency modulation. These steps ensure proper operation of the calibration routine and prevent frequency overshoot from the sequence:

1. Change the following in FMPLL_SYNCR:
   a) Make sure frequency modulation is disabled (FMPLL_SYNCR[DEPTH] = 00). A change to PREDIV, MFD, or RATE while modulation is enabled will invalidate the previous calibration results.
   b) Clear FMPLL_SYNCR[LOLRE]. If this bit is set, the MCU will go into reset when MFD is written.
   c) Initialize the FMPLL for less than the desired final frequency:
   — Disable LOLIRQ.
   — Write FMPLL_SYNCR[PREDIV] for the desired final value.
   — Write FMPLL_SYNCR[MFD] for the desired final value.
   — Write FMPLL_SYNCR[EXP] for the desired final value.
   — Write FMPLL_SYNCR[RATE] for the desired final value.
   — Write the RFD control field for 1 + the desired final RFD value (RFD must be programmed to be >1 to protect from overshoot).
2. Wait for the FMPLL to lock by monitoring the FMPLL_SYNSR[LOCK] bit. Refer to Section 11.3.1.1, “Synthesizer Control Register (FMPLL_SYNCR),” for memory synchronization between changing FMPLL_SYNCR[MFD] and monitoring the lock status.
3. If using the frequency modulation feature, then:
   a) Enable FM by setting FMPLL_SYNCR[DEPTH] =1 or 2.
   b) Also set FMPLL_SYNCR[RATE] if not done previously in step 2.
4. Calibration starts. After calibration is done, then the FMPLL will re-lock. Wait for the FMPLL to re-lock by monitoring the FMPLL_SYNSR[LOCK] bit.
5. Verify FM calibration completed and was successful by testing the FMPLL_SYNSR[CALDONE] and FMPLL_SYNSR[CALPASS] bitfields.
6. If FM calibration did not complete or was not successful, attempt again by going back to step 1.
7. Initialize the FMPLL for the desired final frequency by changing FMPLL_SYNCR[RFD] to its desired final value.Note that the FMPLL will not need to re-lock when only changing the RFD.
8. Re-enable LOLIRQ.

NOTE

This first register write will cause the FMPLL to switch to an initial frequency which is less than the final one. Keeping the change of frequency to a lower initial value helps minimize the current surge to the external power supply caused by change of frequency. The last step will be to only change the RFD to get to the desired final frequency.

NOTE

Changing the MFD or PREDIV values causes the FMPLL to perform a search for the lock frequency that results in the system clock frequency changing rapidly across the complete frequency range. All MCU peripherals, including the external bus will be subjected to this frequency sweep. Operation of timers and serial communications during this search sequence will produce unpredictable results.

Note that the frequency modulation system is dependent upon several factors: the accuracies of the V_{DDSYN}/V_{SSSYN} voltage, of the crystal oscillator frequency, and of the manufacturing variation.

For example, if a 5% accurate supply voltage is utilized, then a 5% modulation depth error will result. If the crystal oscillator frequency is skewed from 8 MHz the resulting modulation frequency will be proportionally skewed. Finally, the error due to the manufacturing and environment variation alone can cause the frequency modulation depth error to be greater than 20%.

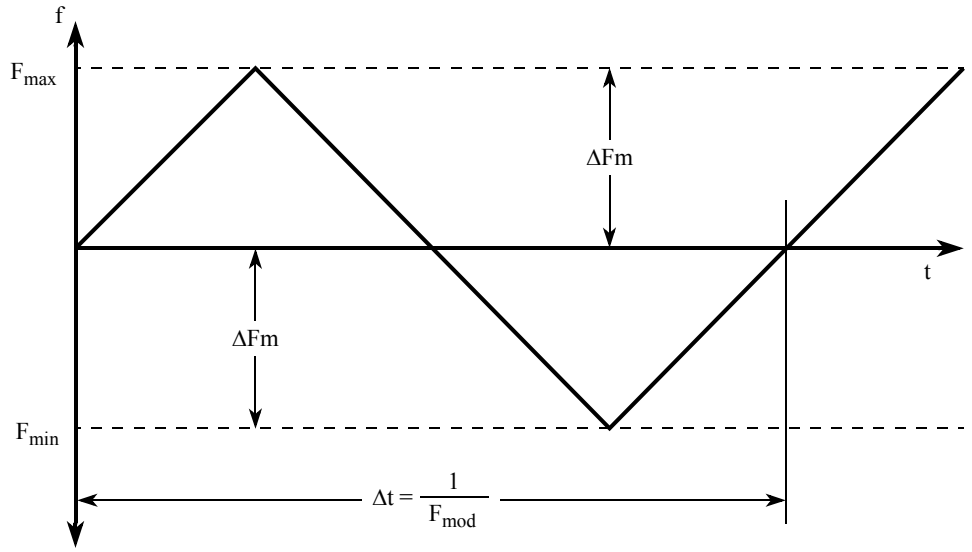


Figure 11-10. Frequency Modulation Waveform

11.4.3.3 FM Calibration Routine

Upon enabling frequency modulation, a new calibration routine is performed. This routine tunes a reference current into the modulation D/A so that the modulation depth (F_{max} and F_{min}) remains within specification.

Entering the FM calibration mode requires the user to program SYNCR[EXP]. The EXP is the expected value of the difference between the reference and feedback counters used in the calibration of the FM equation:

$$EXP = \frac{((MFD + 4) \cdot M \cdot P)}{100}$$

For example, if 80 MHz is the desired final frequency and 8 MHz crystal is used, the final values of MFD=6 and RFD=0 will produce the desired 80 MHz. For a desired frequency modulation with a 1% depth, then EXP is calulated using P = 1, MFD = 6 and M = 480. Refer to Table 11-11 for a complete list of values to be used for the variable (M) based on MFD setting. To obtain a percent modulation (P) of 1%, the EXP field would have to be set at: $EXP = ((6 + 4) \cdot 640 \cdot 1) / 100 = 48$

Rounding this value to the closest integer yields the value of 48 that should be entered into the EXP field for this example.

Table 11-11. Multiplied Factor Dividers with M Values

MFD	M
0-2	960
3-5	640
6-8	480
9-14	320
15-20	240
21-31	160

This routine will correct for process variations, but as temperature can change after the calibration has been performed, variation due to temperature drift is not eliminated. This frequency modulation calibration system is also voltage dependent, so if supply changes after the sequence takes place, error incurred will not be corrected. The calibration system reuses the two counters in the lock detect circuit, the reference and feedback counters. The reference counter is still clocked by the reference clock, but the feedback counter is clocked by the ICO clock.

When the calibration routine is initiated by writing to the DEPTH bits, the CALPASS status bit is immediately set and the CALDONE status bit is immediately cleared.

When calibration is induced, the ICO is given time to settle. Then both the feedback and reference counters start counting. Full ICO clock cycles are counted by the feedback counter during this time to give the initial center frequency count. When the reference counter has counted to the programmed number of reference count cycles, the input to the feedback counter is disabled and the result is placed in the COUNT0 register. The calibration system then enables modulation at programmed ΔFm. The ICO is given time to settle. Both counters are reset and restarted. The feedback counter begins to count full ICO clock cycles again to obtain the delta-frequency count. When the reference counter has counted to the new programmed number of reference count cycles, the feedback counter is stopped again.

The delta-frequency count minus the center frequency count (COUNT0) results in a delta count proportional to the reference current into the modulation D/A. That delta count is subtracted from the expected value given in the EXP field of the FMPLL_SYNCR resulting in an error count. The sign of this error count determines the direction taken by the calibration D/A to update the calibration current. After obtaining the error count for the present iteration, both counters are cleared. The stored count of COUNT0

is preserved while a new feedback count is obtained, and the process to determine the error count is repeated. The calibration system repeats this process eight times, once for each bit of the calibration D/A.

After the last decision is made, the CALDONE bit of the SYNSR is written to a 1. If an error occurs during the calibration routine, then CALPASS is immediately written to a 0. If the routine completed successfully then CALPASS remains a 1.

Figure 11-11 shows a block diagram of the calibration circuitry and its associated registers. Figure 11-12 shows a flow chart showing the steps taken by the calibration circuit.

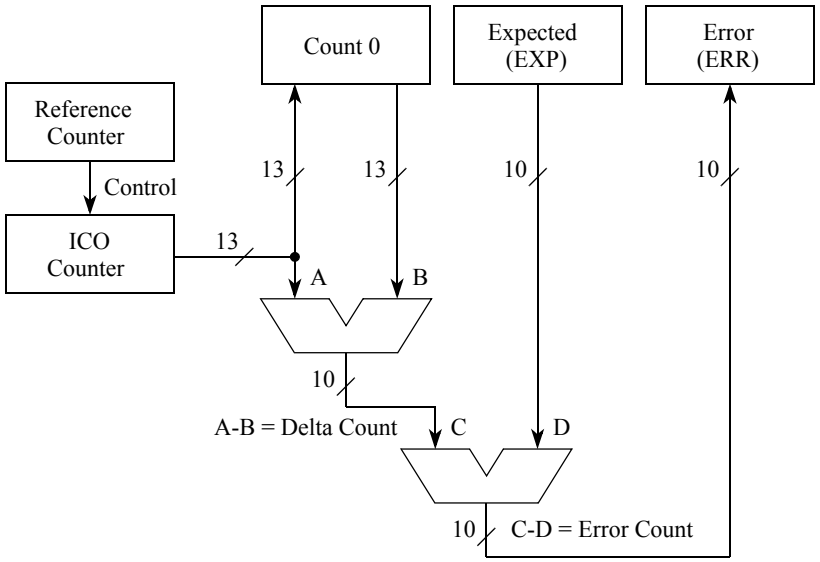


Figure 11-11. FM Auto-Calibration Data Flow

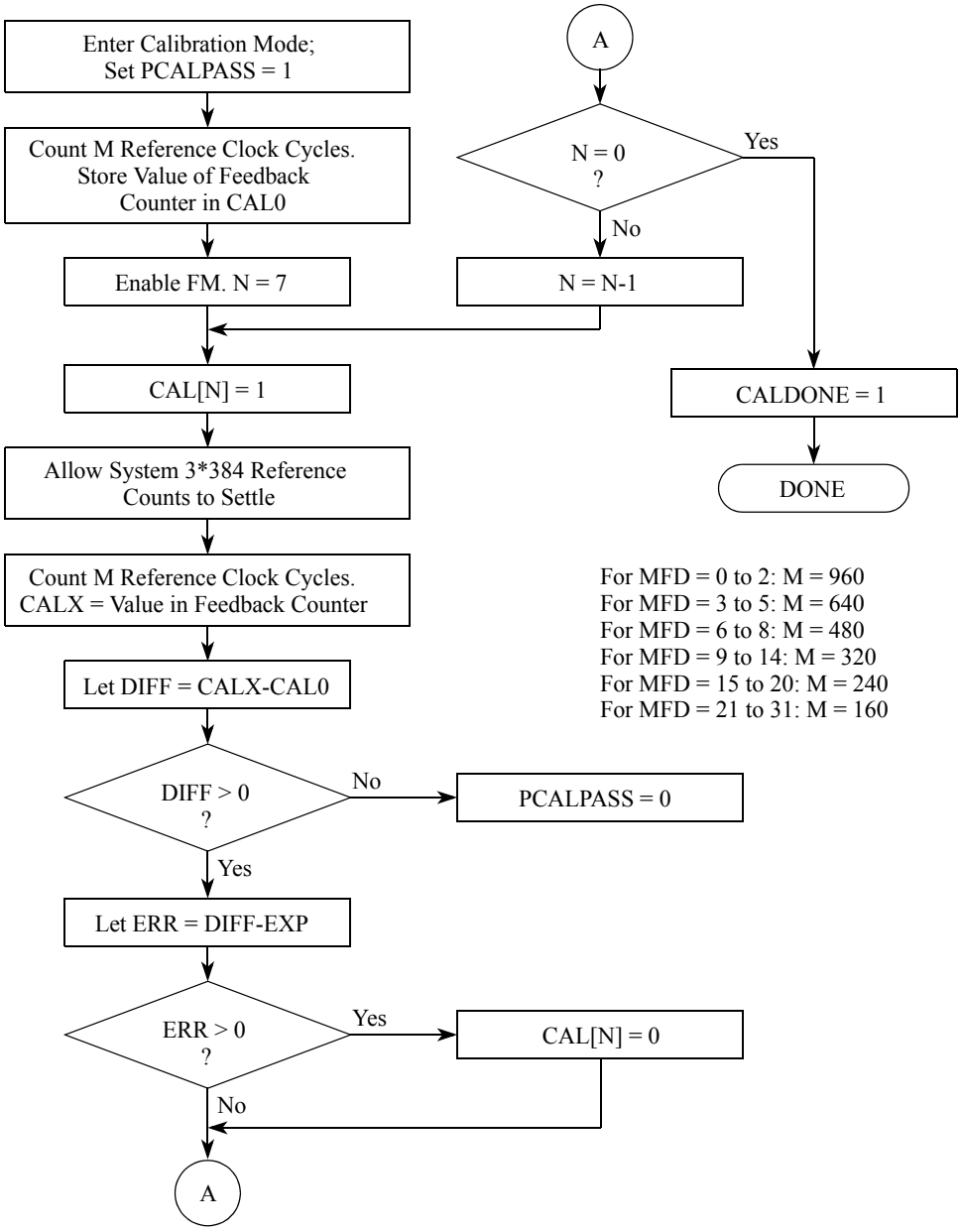


Figure 11-12. FM Auto-Calibration Flow Chart

11.5 Revision History

Substantive Changes since Rev 3.0
Updated Section 11.4.3.3, “FM Calibration Routine.” Section 11.4.3.3, just before table 11-11, gives the formula for EXP as EXP = ((4+4).640.1)/100, should be EXP = ((6+4).480.1)/100 = 48. Changed following line of text to "...yeilds the value of 48..."
Added note to Section 11.3.1.1, “Synthesizer Control Register (FMPLL_SYNCR),” that says “To ensure proper operation across all MPC5500 MCUs, execute an mbar or msync instruction between the write to change the FMPLL_SYNCR[MFD] and the read to check the lock status shown by FMPLL_SYNSR[LOCK]. Furthermore, buffering writes to the FMPLL, as controlled by PBRIDGE_A_OPACR[BW0], must be disabled” Added cross ref to this section from procedures outlined in Section 11.4.3.1, “Programming System Clock Frequency Without Frequency Modulation” and Section 11.4.3.2, “Programming System Clock Frequency with Frequency Modulation.”
Added “Caution: Programming a PREDIV value such that the ICO operates outside its specified range will cause unpredictable results and the FMPLL will not lock. Refer to the *MPC5553/MPC5554 Data Sheet* for details on the ICO range.” to Table 11-10 (Clock-out vs. Clock-in Relationships).

Chapter 12
External Bus Interface (EBI)

12.1 Introduction

This chapter describes the external bus interface (EBI) of the MPC5553/MPC5554, which handles the transfer of information between the internal buses and the memories or peripherals in the external address space and enables an external master to access internal address space. For an overview of how the EBI used in the MPC5553/MPC5554 differs from the EBI used in MPC5xx devices, refer to Section 12.5.6, "Summary of Differences from MPC5xx."

12.1.1 Block Diagram

Figure 12-1 is a block diagram of the EBI. The signals shown are external pins to the MCU. All signals are implemented in the MPC5554 and in the 416 and 324 BGA of the MPC5553 *except where noted*. The MPC5553 208 BGA does not have EBI signals pinned out.

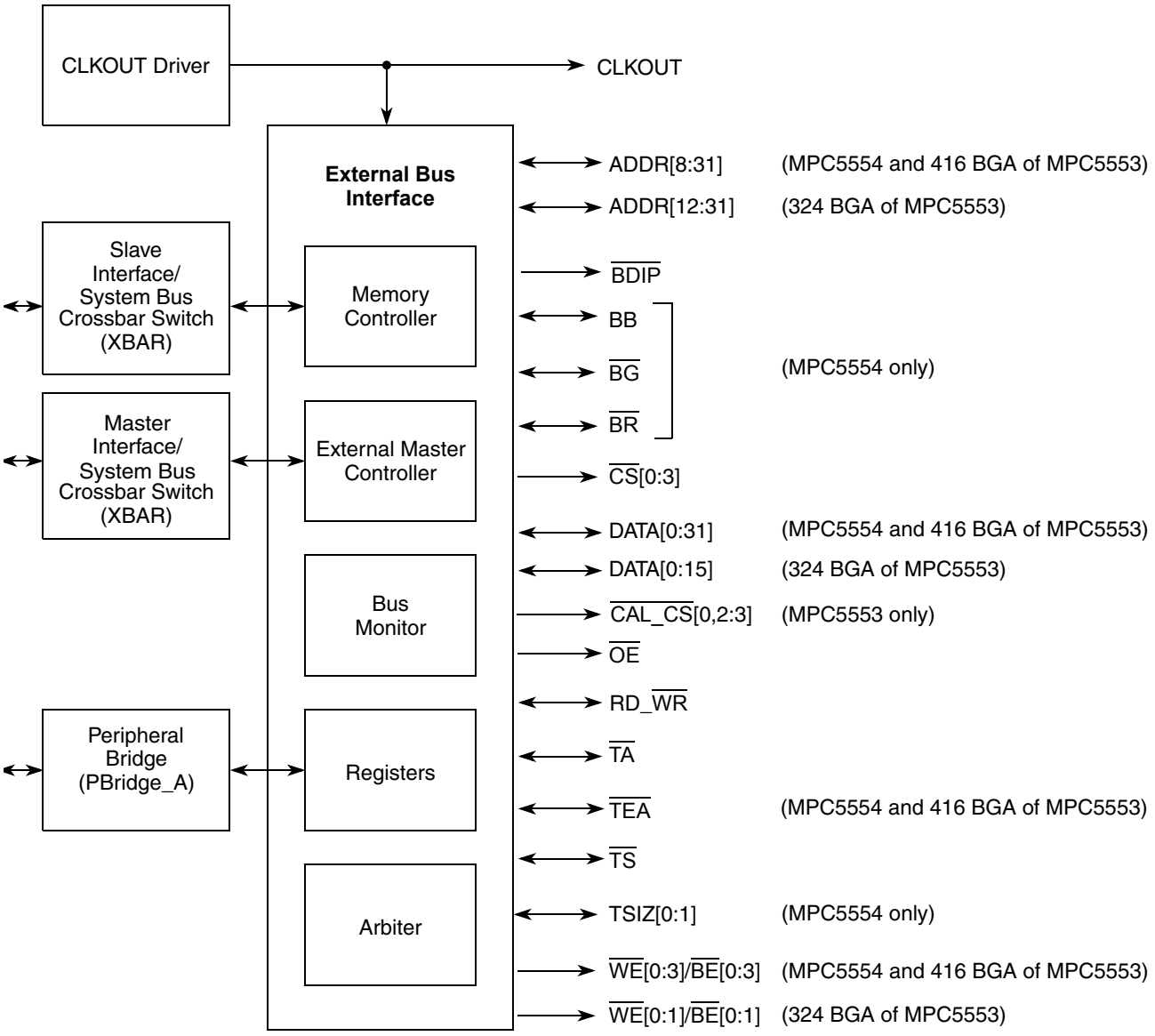


Figure 12-1. EBI Block Diagram

12.1.2 Overview

The EBI includes a memory controller that generates interface signals to support a variety of external memories. This includes single data rate (SDR) burst mode Flash, external SRAM, and asynchronous memories. It supports up to four regions (via chip selects), each with its own programmed attributes.

12.1.3 Features

Features include the following:

- 1.8-3.3 V I/O
- 32-bit address bus with transfer size indication, but
  - 24 bits available in the MPC5554 and in the MPC5553 416 BGA package
  - Only 20 bits available in the MPC5553 324 BGA package
  - No external bus in the MPC5553 208 BGA package
  - Table 12-1 shows the address bus packages supported:

Table 12-1. Address Bus Size in MPC5554 and MPC5553

	MPC5554	MPC5553		
		416	324	208
Bus Size	24 bit	24 bit	20 bit	None

- 32-bit data bus available for both external memory accesses and transactions involving an external master, but
  - 32 bits available in the MPC5554 and in the MPC5553 416 BGA package
  - 16 bits available in the MPC5553 324 BGA package
  - No external bus in the MPC5553 208 BGA package
  - Table 12-2 shows the data bus packages supported:

Table 12-2. Data Bus Size in MPC5554 and MPC5553

	MPC5554	MPC5553		
		416	324	208
Bus Size	32 bit	32 bit	None	None
	16 bit[1]	16 bit[1]	16 bit	None

[1] In both the MPC5554 and in the 416 BGA of the MPC5553, a 16-bit mode is available. See Section 12.1.4.5, "16-Bit Data Bus Mode."

- Support for external master accesses to internal addresses (MPC5554 only)
- Memory controller with support for various memory types:
  - Synchronous burst SDR Flash
  - Asynchronous/legacy Flash
- Burst support (wrapped only)
- Bus monitor
- Port size configuration per chip select (16 or 32 bits)
- Configurable wait states
- Four chip select ($\overline{CS}$[0:3]) signals; but the MPC5553 has no $\overline{CS}$ signals in the 208 BGA package.
- Support for dynamic calibration with up to three calibration chip selects ($\overline{CAL\_CS}$[0] and $\overline{CAL\_CS}$[2:3])(MPC5553 only)

- Write/byte enable ($\overline{WE}$[0:3]/$\overline{BE}$[0:3]) signals
  - The MPC5554 has four $\overline{WE}$/$\overline{BE}$ signals ($\overline{WE}$[0:3]/$\overline{BE}$[0:3])
  - The MPC5553 has the following $\overline{WE}$/$\overline{BE}$ signals depending on the package:
    - 416 BGA: four $\overline{WE}$/$\overline{BE}$ signals ($\overline{WE}$[0:3]/$\overline{BE}$[0:3])
    - 324 BGA: two $\overline{WE}$/$\overline{BE}$ signals ($\overline{WE}$[0:1]/$\overline{BE}$[0:1])
    - 208 BGA: no $\overline{WE}$/$\overline{BE}$ signals
- Configurable bus speed modes (1/2 or 1/4 of system clock frequency)
- Module disable modes for power savings
- Optional automatic CLKOUT gating to save power and reduce EMI (not available on 208 BGA of MPC5553)
- Compatible with MPC5xx external bus (See Section 12.4.1.18, "Compatible with MPC5xx External Bus (with Some Limitations).")

12.1.4 Modes of Operation

The mode of the EBI is determined by the MDIS and EXTM bits in the EBI_MCR. See Section 12.3.1.3, "EBI Module Configuration Register (EBI_MCR)" for details. Configurable bus speed modes and debug mode are modes that the MCU may enter, in parallel to the EBI being configured in one of its module-specific modes.

12.1.4.1 Single Master Mode

In single master mode, the EBI responds to internal requests matching one of its regions, but ignores all externally-initiated bus requests. The MCU is the only master allowed to initiate transactions on the external bus in this mode; therefore, it acts as a parked master and does not have to arbitrate for the bus before starting each cycle. The $\overline{BR}$, $\overline{BG}$, and $\overline{BB}$ signals are not used by the EBI in this mode, and are available for use in an alternate function by another module of the MCU. Single master mode is entered when EXTM = 0 and MDIS = 0 in the EBI_MCR.

12.1.4.2 External Master Mode

When the MPC5554 is in external master mode, the EBI responds to internal requests matching one of its regions, and also to external master accesses to internal address space. In this mode, the $\overline{BR}$, $\overline{BG}$, and $\overline{BB}$ signals are all used by the EBI to handle arbitration between the MCU and an external master. External master mode is entered when EXTM = 1 and MDIS = 0 in the EBI_MCR register.

External master mode support is limited in the MPC5553. See Section 12.5.5, "Dual-MCU Operation with Reduced Pinout MCUs."

External master mode operation is described in Section 12.4.2.10, "Bus Operation in External Master Mode."

12.1.4.3 Module Disable Mode

The module disable mode is used for MCU power management. The clock to the non-memory mapped logic in the EBI is stopped while in module disable mode. Requests (other than to memory-mapped logic) must not be made to the EBI while it is in module disable mode, even if the clocks have not yet been shut off. In this case, the behavior is undefined. Module disable mode is entered when MDIS = 1 in the EBI_MCR.

12.1.4.4 Configurable Bus Speed Modes

In configurable bus speed modes, the external CLKOUT frequency is divided down from the internal system clock. The EBI behavior remains dictated by the mode of the EBI, except that the EBI drives and samples signals at the scaled CLKOUT rather than the internal system clock. This mode is selected by writing the external clock control register in the system integration module (SIU_ECCR). The configurable bus speed modes supports both 1/2 or 1/4 speed modes, meaning that the external CLKOUT frequency is scaled down (by 2 or 4) compared with that of the internal system clock, which is unchanged.

NOTE

In a multi-master system (where the PLL is in dual-controller mode) only 1/2 speed mode is supported.

12.1.4.5 16-Bit Data Bus Mode

For MCUs that have only 16 data bus signals pinned out, or for systems where the use of a different multiplexed function (e.g. GPIO) is desired on 16 of the 32 data pins, the EBI supports a 16-bit data bus mode. In this mode, DATA[0:15] are the only data signals used by the EBI.

For EBI-mastered accesses, the operation in 16-bit data bus mode (EBI_MCR[DBM] = 1, EBI_BR*n*[PS] = x) is similar to a chip select access to a 16-bit port in 32-bit data bus mode (EBI_MCR[DBM] = 0, EBI_BR*n*[PS] = 1), except for the case of an EBI-mastered non-chip select access of exactly 32-bit size.

External master accesses and EBI-mastered non-chip select accesses of exactly 32-bit size are supported via a two (16-bit) beat burst for both reads and writes. See Section 12.4.2.11, "Non-Chip-Select Burst in 16-bit Data Bus Mode." Non-chip select transfers of non-32-bit size are supported in standard non-burst fashion.

16-bit data bus mode is entered when EBI_MCR[DBM] = 1. Note that DBM = 0 out of reset.

12.1.4.6 Debug Mode

When the MCU is in debug mode, the EBI behavior is unaffected and remains dictated by the mode of the EBI.

12.2 External Signal Description

Table 12-3 alphabetically lists the external signals used by the EBI.

Table 12-3. Signal Properties

Name	I/O Type	Function	Pull[1]	MPC5553 Package	MPC5554
ADDR[8:11]	I/O	Address Bus	—	416	Yes
ADDR[12:31]	I/O	Address Bus	—	416, 324	
$\overline{\text{BB}}$	I/O	Bus Busy	Up	None	Yes
$\overline{\text{BDIP}}$	Output	Burst Data in Progress	Up	416, 324	Yes
$\overline{\text{BG}}$	I/O	Bus Grant	Up	None	Yes
$\overline{\text{BR}}$	I/O	Bus Request	Up	None	Yes

Table 12-3. Signal Properties (continued)

Name	I/O Type	Function	Pull[1]	MPC5553 Package	MPC5554
CLKOUT[2]	Output	Clockout	—	416, 324	Yes
$\overline{\text{CS}}$[0:3]	Output	Chip Selects	Up	416, 324	Yes
$\overline{\text{CAL\_CS}}$[0] $\overline{\text{CAL\_CS}}$[2:3]	Output	Calibration Chip Selects	Up	416, 324	No
DATA[0:15]	I/O	Data Bus	—	416, 324	Yes
DATA[16:31]	I/O	Data Bus	—	416	
$\overline{\text{OE}}$	Output	Output Enable	Up	416, 324	Yes
RD_$\overline{\text{WR}}$	I/O	Read_Write	Up	416, 324	Yes
$\overline{\text{TA}}$	I/O	Transfer Acknowledge	Up	416, 324	Yes
$\overline{\text{TEA}}$	I/O	Transfer Error Acknowledge	Up	416	Yes
$\overline{\text{TS}}$	I/O	Transfer Start	Up	416, 324	Yes
TSIZ[0:1]	I/O	Transfer Size	—	None	Yes
$\overline{\text{WE}}$[0:1]/$\overline{\text{BE}}$[0:1]	Output	Write/Byte Enables	Up	416, 324	Yes
$\overline{\text{WE}}$[2:3]/$\overline{\text{BE}}$[2:3]	Output	Write/Byte Enables	Up	416	

1 This column shows which signals require a weak pull-up or pull-down. The EBI module does not contain these pull-up/pull-down devices within the module, but instead are controlled by the pad configuration registers in the System Integration Module (SIU_PCRs).

2 The CLKOUT signal is driven by the FMPLL Module.

12.2.1 Detailed Signal Descriptions

The MPC5554 and the 416 and 324 BGA packages of the MPC5553 have pinned out EBI signals. The 208 BGA package of the MPC5553 does not pin out these signals.

12.2.1.1 Address Lines 8–31 (ADDR[8:31])

The ADDR[8:31] signals specify the physical address of the bus transaction. The 24 address lines correspond to bits 8–31 of the EBI's 32-bit internal address bus. Bits 0–7 are internally driven by the EBI for externally initiated accesses depending on which internal slave is to be accessed. See Section 12.4.2.10.1, "Address Decoding for External Master Accesses," for more details. ADDR[8:31] is driven by the EBI or an external master depending on who owns the external bus.

Note that the 324 package of the MPC5553 uses only ADDR[12:31].

12.2.1.2 Bus Busy ($\overline{\text{BB}}$) — MPC5554 Only

$\overline{\text{BB}}$ is asserted to indicate that the current bus master is using the bus. The $\overline{\text{BB}}$ signal is only used by the EBI when the EBI is in external master mode. In single master mode, the $\overline{\text{BB}}$ signal is never asserted or sampled by the EBI.

When configured for internal arbitration, the EBI asserts $\overline{\text{BB}}$ to indicate that it is currently using the bus. An external master must not begin a transfer until this signal is negated for two cycles. The EBI does not

negate this signal until its transfer is complete. When not driving $\overline{BB}$, the EBI samples this signal to get an indication of when the external master is no longer using the bus ($\overline{BB}$ negated for two cycles).

When configured for external arbitration, the EBI asserts this signal when it is ready to start the transaction after the external arbiter has granted ownership of the bus to the MCU. When not driving $\overline{BB}$, the EBI samples this signal to properly qualify the $\overline{BG}$ line when an external bus transaction is to be executed by the MCU.

12.2.1.3 Burst Data in Progress ($\overline{BDIP}$)

$\overline{BDIP}$ is asserted to indicate that the master is requesting another data beat following the current one.

$\overline{BDIP}$ is driven by the EBI or an external master depending on who owns the external bus. This signal is driven by the EBI on all EBI-mastered external burst cycles, but is only sampled by burst mode memories that have a corresponding pin. See Section 12.4.2.5, "Burst Transfer."

12.2.1.4 Bus Grant ($\overline{BG}$) — MPC5554 Only

$\overline{BG}$ is asserted to grant ownership of the external bus to the requesting master. The $\overline{BG}$ signal is only used by the EBI when the EBI is in external master mode. In single master mode, the $\overline{BG}$ signal is never asserted or sampled by the EBI.

When configured for internal arbitration, $\overline{BG}$ is output only and is asserted by the EBI to indicate that an external master may assume ownership of the bus. The $\overline{BG}$ signal should be qualified by the master requesting the bus in order to ensure it is the bus owner before beginning a bus transaction: Qualified bus grant = $\overline{BG}$ and ~$\overline{BB}$. The EBI negates $\overline{BG}$ following the negation of $\overline{BR}$ if it has an internal request for the external bus pending. Otherwise, it keeps $\overline{BG}$ asserted to park the bus for the external master. The parked external master could then assert $\overline{BB}$ to run subsequent transactions without the normal requirement to assert $\overline{BR}$.

When configured for external arbitration, $\overline{BG}$ is input only and is sampled and qualified (Qualified BG = ~$\overline{BB}$ and $\overline{BG}$) by the EBI when an external bus transaction is to be executed by the MCU.

12.2.1.5 Bus Request ($\overline{BR}$) — MPC5554 Only

$\overline{BR}$ is asserted to request ownership of the external bus. The $\overline{BR}$ signal is only used by the EBI when the EBI is in external master mode. In single master mode, the $\overline{BR}$ signal is never asserted or sampled by the EBI.

When configured for internal arbitration, $\overline{BR}$ is input only and is asserted by an external master when it is requesting the bus.

When configured for external arbitration, $\overline{BR}$ is output only and is asserted by the EBI when it is requesting the bus. The EBI negates $\overline{BR}$ as soon as it is granted the bus and the bus is not busy, provided it has no other internal requests pending. If more requests are pending, the EBI keeps $\overline{BR}$ asserted as long as needed.

12.2.1.6 Clockout (CLKOUT)

CLKOUT is a general-purpose clock output signal to connect to the clock input of SDR external memories and in some cases to the input clock of another MCU in multi-master configurations.

12.2.1.7 Chip Selects 0–3 ($\overline{CS}$[0:3])

$\overline{CS}$x is asserted by the master to indicate that this transaction is targeted for a particular memory bank.

The chip selects are driven by the EBI or an external master depending on who owns the external bus. $\overline{CS}$ is driven in the same clock as the assertion of $\overline{TS}$ and valid address, and is kept valid until the cycle is terminated. See Section 12.4.1.5, "Memory Controller with Support for Various Memory Types" for details on chip select operation.

12.2.1.8 Calibration Chip Selects 0, 2-3 ($\overline{CAL\_CS}$ [0], $\overline{CAL\_CS}$ [2:3]) — MPC5553 Only

$\overline{CAL_CS}$x is asserted by the master to indicate that this transaction is targeted for a particular memory bank on the calibration external bus.

The calibration chip selects are driven only by the EBI. External master accesses on the calibration bus are not supported. In all other aspects, the calibration chip selects behave exactly as the primary chip selects. See Section 12.4.1.5, "Memory Controller with Support for Various Memory Types for details on chip select operation.

12.2.1.9 Data Lines 0–31 (DATA[0:31])

In the 416-pin package of the MPC5553/MPC5554, the DATA[0:31] signals contain the data to be transferred for the current transaction. In the 324-pin package of the MPC5553, DATA[0:15] carry the data.

DATA[0:31] is driven by the EBI when it owns the external bus and it initiates a write transaction to an external device. The EBI also drives DATA[0:31] when an external master owns the external bus and initiates a read transaction to an internal module.

DATA[0:31] is driven by an external device during a read transaction from the EBI. An external master drives DATA[0:31] when it owns the bus and initiates a write transaction to an internal module or shared external memory.

For 8-bit and 16-bit transactions, the byte lanes not selected for the transfer do not supply valid data.

12.2.1.10 Output Enable ($\overline{OE}$)

$\overline{OE}$ is used to indicate when an external memory is permitted to drive back read data. External memories must have their data output buffers off when $\overline{OE}$ is negated. $\overline{OE}$ is only asserted for chip select accesses.

$\overline{OE}$ is driven by the EBI or an external master depending on who owns the external bus. For read cycles, $\overline{OE}$ is asserted one clock after $\overline{TS}$ assertion and held until the termination of the transfer. For write cycles, $\overline{OE}$ is negated throughout the cycle.

12.2.1.11 Read / Write (RD_$\overline{WR}$)

RD_$\overline{WR}$ indicates whether the current transaction is a read access or a write access.

RD_$\overline{WR}$ is driven by the EBI or an external master depending on who owns the external bus. RD_$\overline{WR}$ is driven in the same clock as the assertion of $\overline{TS}$ and valid address, and is kept valid until the cycle is terminated.

12.2.1.12 Transfer Acknowledge ($\overline{TA}$)

$\overline{TA}$ is asserted to indicate that the slave has received the data (and completed the access) for a write cycle, or returned data for a read cycle. If the transaction is a burst read, $\overline{TA}$ is asserted for each one of the

transaction beats. For write transactions, $\overline{TA}$ is only asserted once at access completion, even if more than one write data beat is transferred.

$\overline{TA}$ is driven by the EBI when the access is controlled by the chip selects or when an external master initiated the transaction to an internal module. Otherwise, $\overline{TA}$ is driven by the slave device to which the current transaction was addressed.

See Section 12.4.2.9, "Termination Signals Protocol" for more details.

12.2.1.13 Transfer Error Acknowledge ($\overline{TEA}$)

In the 416-pin package of the MPC5553/MPC5554, $\overline{TEA}$ is asserted by either the EBI or an external device to indicate that an error condition has occurred during the bus cycle. $\overline{TEA}$ assertion terminates the cycle immediately, overriding the value of the $\overline{TA}$ signal.

$\overline{TEA}$ is asserted by the EBI when the internal bus monitor detected a timeout error, or when an external master initiated a transaction to an internal module and an internal error was detected.

The 324 BGA package of the MPC5553 has no $\overline{TEA}$ signal.

See Section 12.4.2.9, "Termination Signals Protocol" for more details.

12.2.1.14 Transfer Start ($\overline{TS}$)

$\overline{TS}$ is asserted by the current bus owner to indicate the start of a transaction on the external bus.

$\overline{TS}$ is driven by the EBI or an external master depending on who owns the external bus. $\overline{TS}$ is only asserted for the first clock cycle of the transaction, and is negated in the successive clock cycles until the end of the transaction.

12.2.1.15 Transfer Size 0–1 (TSIZ[0:1]) — MPC5554 Only

TSIZ[0:1] indicates the size of the requested data transfer.

TSIZ[0:1] is driven by the EBI or an external master depending on who owns the external bus. The TSIZ[0:1] signals may be used with ADDR[30:31] to determine which byte lanes of the data bus are involved in the transfer. For non-burst transfers, the TSIZ[0:1] signals specify the number of bytes starting from the byte location addressed by ADDR[30:31]. In burst transfers, the value of TSIZ[0:1] is always 00.

Table 12-4. TSIZ[0:1] Encoding

Burst Cycle	TSIZ[0:1]	Transfer Size
N	01	Byte
N	10	16-bit
N	11	Reserved
N	00	32-bit
Y	00	Burst

If the SIZEN bit in the EBI_MCR is 1, then TSIZ[0:1] is ignored by the EBI as an input for external master transactions and the size is instead determined by the SIZE field in the EBI_MCR. The SIZEN bit has no effect on the EBI when it is mastering a transaction on the external bus. TSIZ[0:1] is still driven appropriately by the EBI and may or may not be used by the external master depending on the SIZEN

setting for the external master's EBI. See Section 12.3.1.3, "EBI Module Configuration Register (EBI_MCR)."

12.2.1.16 Write/Byte Enables ($\overline{WE}$ / $\overline{BE}$)

Write enables are used to enable program operations to a particular memory. These signals can also be used as byte enables for read and write operation by setting the WEBS bit in the appropriate base register. $\overline{WE}$ / $\overline{BE}$ are only asserted for chip select accesses.

$\overline{WE}$ / $\overline{BE}$ are driven by the EBI or an external master depending on who owns the external bus. See Section 12.4.1.13, "Four Write/Byte Enable (WE/BE) Signals — Only MPC5554 and 416 BGA of MPC5553" for more details on $\overline{WE}$ / $\overline{BE}$ functionality.

The MPC5554 and the 416 BGA package of the MPC5553 use $\overline{WE}$[0:3]/ $\overline{BE}$[0:3]. The 324 BGA of the MPC5553 uses only $\overline{WE}$[0:1]/ $\overline{BE}$[0:1]. The 208 BGA of the MPC5553 has no write/byte enable signals.

12.2.2 Signal Function/Direction by Mode

Depending on the mode of operation, some or all of the EBI external signals may not be used by the EBI. When a signal is configured for non-EBI function in the EBI_MCR, the EBI always negates the signal if the EBI controls the corresponding pad (determined by SIU configuration). Table 12-5 lists the function and direction of the external signals in each of the EBI modes of operation. The clock signals are not included because they are output only (from the FMPLL module) and are not affected by EBI modes. See Section 12.3.1.3, "EBI Module Configuration Register (EBI_MCR)" for details on the EXTM and MDIS bits.

Table 12-5. Signal Function by Mode

Signal Name	Device		Modes		
	MPC5554 Contains the Signal?	MPC5553 Package That Contains the Signal	Module Disable Mode Function (EXTM = X, MDIS = 1)	Single Master Mode Function (Direction) (EXTM = 0, MDIS = 0)	External Master Mode Function (Direction) (EXTM = 1, MDIS = 0)
ADDR[8:11]	Yes	416	non-EBI function	Address bus (Output)	Address bus (I/O)
ADDR[12:31]		416, 324	non-EBI function	Address bus (Output)	Address bus (I/O)
$\overline{BB}$		None	non-EBI function	non-EBI function	Bus Busy (I/O)
$\overline{BDIP}$		416, 324	non-EBI function	Burst Data in Progress (Output)	
$\overline{BG}$		None	non-EBI function	non-EBI function	Bus Grant (I/O)
$\overline{BR}$		None	non-EBI function	non-EBI function	Bus Request (I/O)
$\overline{CS}$[0:3]		416, 324	non-EBI function	Chip Selects (Output)	
DATA[0:15]		416, 324	non-EBI function	Data bus (I/O)	
DATA[16:31]		416	non-EBI function	Data bus (I/O)	
$\overline{OE}$		416, 324	non-EBI function	Output Enable (Output)	
RD_$\overline{WR}$		416, 324	non-EBI function	Read_Write (Output)	Read_Write (I/O)
$\overline{TA}$		416, 324	non-EBI function	Transfer Acknowledge (I/O)	
$\overline{TEA}$		416	non-EBI function	Transfer Error Acknowledge (I/O)	
$\overline{TS}$		416, 324	non-EBI function	Transfer Start (Output)	Transfer Start (I/O)
TSIZ[0:1]		None	non-EBI function	Transfer Size (Output)	Transfer Size (I/O)
$\overline{WE}$[0:]/$\overline{BE}$[0:1]		416, 324	non-EBI function	Write/Byte Enables (Output)	
$\overline{WE}$[2:3]/$\overline{BE}$[2:3]		416	non-EBI function	Write/Byte Enables (Output)	
$\overline{CAL\_CS}$[0] $\overline{CAL\_CS}$[2:3]	No	416	non-EBI function	Chip Selects (Output)	

12.2.3 Signal Pad Configuration by Mode

Depending on the mode of operation, many external signals must have their pads configured to operate as push/pull signals for correct system operation. This configuration is done in the SIU module.

The open drain mode of the pads configuration module is not used for any EBI signals. For a description of how signals are driven by multiple devices in external master mode, see Section 12.4.2.10, "Bus Operation in External Master Mode."

Table 12-6 shows how each EBI signal must have its pad configured prior to operating in each of the EBI modes. See Section 12.3.1.3, "EBI Module Configuration Register (EBI_MCR)" for details on the EXTM and MDIS bits.

Table 12-6. Required EBI Pad Configuration by Mode

Signal Name	MPC5553 Package Type	MPC5554 and MPC5553		
		Module Disable Mode (EXTM = X, MDIS = 1)	Single Master Mode (EXTM = 0, MDIS = 0)	External Master Mode (EXTM = 1, MDIS = 0)
ADDR[8:11]	416	X[1]	Push/Pull	Push/Pull Three-stateable
ADDR[12:31]	416, 324	X	Push/Pull	Push/Pull Three-stateable
$\overline{BB}$	None	X	X	Push/Pull, Three-stateable
$\overline{BDIP}$	416, 324	X	Push/Pull	Push/Pull Three-stateable
$\overline{BG}$	None	X	X	Push/Pull, Three-stateable
$\overline{BR}$	None	X	X	Push/Pull, Three-stateable
$\overline{CS}$[0:3]	416, 324	X	Push/Pull	Push/Pull, Three-stateable
$\overline{CAL\_CS}$[0] $\overline{CAL\_CS}$[2:3]	416	X	Push/Pull	N.A.
DATA[0:15]	416, 324	X	Push/Pull, Three-stateable	
DATA[16:31]	416	X	Push/Pull, Three-stateable	
$\overline{OE}$	416, 324, 208	X	Push/Pull	Push/Pull, Three-stateable
RD_$\overline{WR}$	416, 324	X	Push/Pull	Push/Pull, Three-stateable
$\overline{TA}$	416, 324	X	Push/Pull, Three-stateable	
$\overline{TEA}$	416	X	Push/Pull, Three-stateable	
$\overline{TS}$	416, 324	X	Push/Pull	Push/Pull, Three-stateable

Table 12-6. Required EBI Pad Configuration by Mode (continued)

Signal Name	MPC5553 Package Type	MPC5554 and MPC5553		
		Module Disable Mode (EXTM = X, MDIS = 1)	Single Master Mode (EXTM = 0, MDIS = 0)	External Master Mode (EXTM = 1, MDIS = 0)
TSIZ[0:1]	None	X	Push/Pull	Push/Pull, Three-stateable
$\overline{WE}$[0:]/$\overline{BE}$[0:1]	416, 324	X	Push/Pull	Push/Pull, Three-stateable
$\overline{WE}$[2:3]/$\overline{BE}$[2:3]	416	X	Push/Pull	Push/Pull, Three-stateable

1 'X' indicates the pad configuration is a don't care, because the signal is not used by the EBI in this mode.

12.3 Memory Map/Register Definition

Table 12-7 is a memory map of the EBI registers.

Table 12-7. EBI Memory Map

Address	Register Name	Register Description	Size (bits)
Base (0xC3F8_4000)	EBI_MCR	EBI module configuration register	32
Base + 0x0004	—	Reserved	—
Base + 0x0008	EBI_TESR	EBI transfer error status register	32
Base + 0x000C	EBI_BMCR	EBI bus monitor control register	32
Base + 0x0010	EBI_BR0	EBI base register bank 0	32
Base + 0x0014	EBI_OR0	EBI option register bank 0	32
Base + 0x0018	EBI_BR1	EBI base register bank 1	32
Base + 0x001C	EBI_OR1	EBI option register bank 1	32
Base + 0x0020	EBI_BR2	EBI base register bank 2	32
Base + 0x0024	EBI_OR2	EBI option register bank 2	32
Base + 0x0028	EBI_BR3	EBI base register bank 3	32
Base + 0x002C	EBI_OR3	EBI option register bank 3	32
MPC5553-Only Calibration Registers:			
Base + 0x30 - Base + 0x3C	—	Reserved	—
Base + 0x0040	EBI_CAL_BR0	EBI Calibration Base Register Bank 0	32
Base + 0x0044	EBI_CAL_OR0	EBI Calibration Option Register Bank 0	32
Base + 0x0048	EBI_CAL_BR1	EBI Calibration Base Register Bank 1	32
Base + 0x004C	EBI_CAL_OR1	EBI Calibration Option Register Bank 1	32
Base + 0x0050	EBI_CAL_BR2	EBI Calibration Base Register Bank 2	32
Base + 0x0054	EBI_CAL_OR2	EBI Calibration Option Register Bank 2	32
Base + 0x0058	EBI_CAL_BR3	EBI Calibration Base Register Bank 3	32
Base + 0x005C	EBI_CAL_OR3	EBI Calibration Option Register Bank 3	32

12.3.1 Register Descriptions

12.3.1.1 Writing EBI Registers While a Transaction is in Progress

Other than the exceptions noted below, EBI registers must *not* be written while a transaction to the EBI (from internal or external master) is in progress (or within 2 CLKOUT cycles after a transaction has just completed, to allow internal state machines to go IDLE). In such cases, the behavior is undefined.

Exceptions that can be written while an EBI transaction is in progress are the following:

- All bits in EBI_TESR

- SIZE, SIZEN fields in EBI_MCR

See Section 12.5.1, "Booting from External Memory" for additional information.

12.3.1.2 Separate Input Clock for Registers

The EBI registers are accessed with a clock signal separate from the clock used by the rest of the EBI. In module disable mode, the clock used by the non-register portion of the EBI is disabled to reduce power consumption. The clock signal dedicated to the registers, however, allows access to the registers even while the EBI is in the module disable mode. Flag bits in the EBI transfer error status register (EBI_TESR), however, are set and cleared with the clock used by the non-register portion of the EBI. Consequently, in module disable mode, the EBI_TESR does not have a clock signal and is therefore not writable.

12.3.1.3 EBI Module Configuration Register (EBI_MCR)

The EBI_MCR contains bits that configure various attributes associated with EBI operation.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	SIZEN	SIZE		0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base (0xC3F8_4000)															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ACGE	EXTM	EARB	EARP		0	0	0	0	MDIS	0	0	0	0	0	DBM
W																
Reset	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base (0xC3F8_4000)															

Figure 12-2. EBI Module Configuration Register (EBI_MCR)

Table 12-8. EBI_MCR Field Descriptions

Bits	Name	Description
0–4	—	Reserved.
5	SIZEN	SIZE enable. The SIZEN bit enables the control of transfer size by the SIZE field (as opposed to external TSIZ pins) for external master transactions to internal address space. 0 Transfer size controlled by TSIZ[0:1] pins 1 Transfer size controlled by SIZE field
6–7	SIZE	Transfer size. The SIZE field determines the transfer size of external master transactions to internal address space when SIZEN=1. This field is ignored when SIZEN=0. SIZE encoding: 00 32-bit 01 Byte 10 16-bit 11 Reserved

Table 12-8. EBI_MCR Field Descriptions (continued)

Bits	Name	Description
16	ACGE	Automatic CLKOUT gating enable. Enables the EBI feature of turning off CLKOUT (holding it high) during idle periods in-between external bus accesses. 0 Automatic CLKOUT gating is disabled 1 Automatic CLKOUT gating is enabled
17	EXTM	External master mode. Enables the external master mode of operation when MDIS = 0. When MDIS = 1, the EXTM bit is a don't care, and is treated as 0. In external master mode, an external master on the external bus can access any internal memory-mapped space while the internal e200z6 core is fully operational. When EXTM = 0, only internal masters can access the internal memory space. This bit also determines the functionality of the $\overline{BR}$, $\overline{BG}$, and $\overline{BB}$ signals. **Note:** The SIU PCR registers must configure $\overline{BR}$, $\overline{BG}$, and $\overline{BB}$ for EBI function (as opposed to default GPIO) *prior* to EXTM being set to 1, or erroneous behavior may result. 0 External master mode is inactive (single master mode) 1 External master mode is active **Note:** In the MPC5553, only master/slave systems support the EXTM functionality. Refer to 12.5.5.
18	EARB	External arbitration. See Section 12.4.2.8, "Arbitration," for details on internal and external arbitration. When EXTM = 0, the EARB bit is a don't care, and is treated as 0. 0 Internal arbitration is used. 1 External arbitration is used.
19–20	EARP [0:1]	External arbitration request priority. Defines the priority of an external master's arbitration request (0–2), with 2 being the highest priority level (EARP = 3 is reserved). This field is valid only when EARB = 0 (internal arbitration). The internal masters of the MCU have a fixed priority of 1. By default, internal and external masters have equal priority. See Section 12.4.2.8.2, "Internal Bus Arbiter," for the internal and external priority detailed description. 00 MCU has priority 01 Equal priority, round robin used 10 External master has priority 11 Reserved
21–24	—	Reserved.
25	MDIS	Module disable mode. Allows the clock to be stopped to the non-memory mapped logic in the EBI, effectively putting the EBI in a software controlled power-saving state. See Section 12.1.4.3, "Module Disable Mode," for more information. No external bus accesses can be performed when the EBI is in module disable mode (MDIS = 1). 0 Module disable mode is inactive 1 Module disable mode is active
26–30	—	Reserved.
31	DBM	Data bus mode. Controls whether the EBI is in 32-bit or 16-bit data bus mode. 0 32-bit data bus mode is used 1 16-bit data bus mode is used

12.3.1.4 EBI Transfer Error Status Register (EBI_TESR)

The EBI_TESR contains a bit for each type of transfer error on the external bus. A bit set to logic 1 indicates what type of transfer error occurred since the last time the bits were cleared. Each bit can be cleared by reset or by writing a 1 to it. Writing a 0 has no effect.

This register is not writable in module disable mode due to the use of power saving clock modes.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0008															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TEAF	BMTF
W															w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0008															

Note: w1c means “write 1 to clear” and is explained in the Preface.

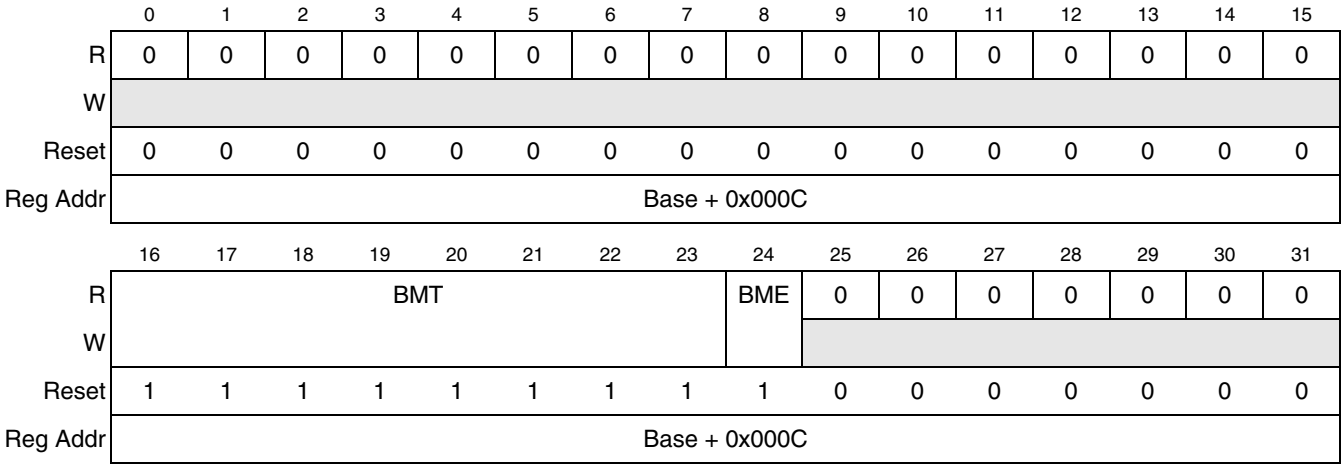
Figure 12-3. EBI Transfer Error Status Register (EBI_TESR)

Table 12-9. EBI_TESR Field Descriptions

Bits	Name	Description
0–29	—	Reserved.
30	TEAF	Transfer error acknowledge flag. Set if the cycle was terminated by an externally generated $\overline{TEA}$ signal. 0 No error 1 External $\overline{TEA}$ occurred This bit can be cleared by writing a 1 to it.
31	BMTF	Bus monitor timeout flag. Set if the cycle was terminated by a bus monitor timeout. 0 No error 1 Bus monitor timeout occurred This bit can be cleared by writing a 1 to it.

12.3.1.5 EBI Bus Monitor Control Register (EBI_BMCR)

The EBI_BMCR controls the timeout period of the bus monitor and whether it is enabled or disabled.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x000C															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BMT								BME	0	0	0	0	0	0	0
W																
Reset	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
Reg Addr	Base + 0x000C															

Figure 12-4. EBI Bus Monitor Control Register (EBI_BMCR)

Table 12-10. EBI_BMCR Field Descriptions

Bits	Name	Description
0–15	—	Reserved.
16–23	BMT [0:7]	Bus monitor timing. Defines the timeout period, in 8 external bus clock resolution, for the Bus Monitor. See Section 12.4.1.7, "Bus Monitor," for more details on bus monitor operation. $\text{Timeout Period} = \frac{2 + (8 \times \text{BMT})}{\text{External Bus Clock Frequency}}$
24	BME	Bus monitor enable. Controls whether the bus monitor is enabled for internal to external bus cycles. Regardless of the BME value, the bus monitor is always disabled for chip select accesses, since these always use internal $\overline{TA}$ and thus have no danger of hanging the system. 0 Disable bus monitor 1 Enable bus monitor (for non-chip select accesses only)
25–31	—	Reserved.

12.3.1.6 EBI Base Registers 0–3 (EBI_BR*n*) and EBI Calibration Base Registers 0–3 (EBI_CAL_BR*n*)

The EBI_BR*n* are used to define the base address and other attributes for the corresponding chip select. The EBI_CAL_BR*n* appear in the MPC5553 only and are used to define the base address and other attributes for the corresponding calibration chip select.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	1	BA												
W																
Reset	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0010 (EBI_BR0); Base + 0x0018 (EBI_BR1); Base + 0x0020 (EBI_BR2); Base + 0x0028 (EBI_BR3); Base + 0x0040 (EBI_CAL_BR0); Base + 0x0048 (EBI_CAL_BR1); Base + 0x0050 (EBI_CAL_BR2); Base + 0x0058 (EBI_CAL_BR3)															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BA	0	0	0	PS	0	0	0	0	BL	WEBS	TBDIP	0	0	BI	V
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
Reg Addr	Base + 0x0010 (EBI_BR0); Base + 0x0018 (EBI_BR1); Base + 0x0020 (EBI_BR2); Base + 0x0028 (EBI_BR3); Base + 0x0040 (EBI_CAL_BR0); Base + 0x0048 (EBI_CAL_BR1); Base + 0x0050 (EBI_CAL_BR2); Base + 0x0058 (EBI_CAL_BR3)															

Figure 12-5. EBI Base Registers 0–3 (EBI_BR*n*) and EBI Calibration Base Registers 0–3 (EBI_CAL_BR*n*)

Table 12-11. EBI_BR*n* and EBI_CAL_BR*n* Field Descriptions

Bits	Name	Description
0–16	BA [0:16]	Base address. Compared to the corresponding unmasked address signals among ADDR[0:16] of the internal address bus to determine if a memory bank controlled by the memory controller is being accessed by an internal bus master. **Note:** The upper 3 bits of the base address (BA) field, EBI_BR*n*[0:2], and EBI_CAL_BR*n*[0:2], are tied to a fixed value of 001. These bits reset to their fixed value.
17–19	—	Reserved.
20	PS	Port size. Determines the data bus width of transactions to this chip select bank.[1] 0 32-bit port 1 16-bit port **Note:** The calibration port size must be 16-bits wide.
21–24	—	Reserved.
25	BL	Burst length. Determines the amount of data transferred in a burst for this chip select, measured in 32-bit words. The number of beats in a burst is automatically determined by the EBI to be 4, 8, or 16 according to the port size so that the burst fetches the number of words chosen by BL. 0 8-word burst length 1 4-word burst length
26	WEBS	Write enable/byte select. Controls the functionality of the $\overline{WE}$[0:3]/$\overline{BE}$[0:3] signals. 0 The $\overline{WE}$[0:3]/$\overline{BE}$[0:3] signals function as $\overline{WE}$[0:3]. 1 The $\overline{WE}$[0:3]/$\overline{BE}$[0:3] signals function as $\overline{BE}$[0:3].

Table 12-11. EBI_BR*n* and EBI_CAL_BR*n* Field Descriptions (continued)

Bits	Name	Description
27	TBDIP	Toggle burst data in progress. Determines how long the $\overline{\text{BDIP}}$ signal is asserted for each data beat in a burst cycle. See Section 12.4.2.5.1, "TBDIP Effect on Burst Transfer," for details. 0 Assert $\overline{\text{BDIP}}$ throughout the burst cycle, regardless of wait state configuration. 1 Only assert $\overline{\text{BDIP}}$ (BSCY + 1) external bus cycles before expecting subsequent burst data beats.
28–29	—	Reserved.
30	BI	Burst inhibit. Determines whether or not burst read accesses are allowed for this chip select bank. 0 Enable burst accesses for this bank. 1 Disable burst accesses for this bank. This is the default value out of reset.
31	V	Valid bit. Indicates that the contents of this base register and option register pair are valid. The appropriate $\overline{\text{CS}}$ signal does not assert unless the corresponding V-bit is set. 0 This bank is not valid. 1 This bank is valid.

[1] In the case where EBI_MCR[DBM] is set for 16-bit data bus mode, the PS bit value is ignored and is always treated as a 1 (16-bit port).

12.3.1.7 EBI Option Registers 0–3 (EBI_OR*n*) and EBI Calibration Option Registers 0-3 (EBI_CAL_OR*n*)

The EBI_OR*n* registers are used to define the address mask and other attributes for the corresponding chip select. The EBI_CAL_OR*n* registers appear in the MPC5553 only and are used to define the address mask and other attributes for the corresponding calibration chip select.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	1	1	1	AM												
W																
Reset	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0014 (EBI_OR0); Base + 0x001C (EBI_OR1); Base + 0x0024 (EBI_OR2); Base + 0x002C (EBI_OR3); Base + 0x0044 (EBI_CAL_OR0); Base + 0x004C (EBI_CAL_OR1); Base + 0x0054 (EBI_CAL_OR2); Base + 0x005C (EBI_CAL_OR3)															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AM	0	0	0	0	0	0	0	SCY				0	BSCY		0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0014 (EBI_OR0); Base + 0x001C (EBI_OR1); Base + 0x0024 (EBI_OR2); Base + 0x002C (EBI_OR3); Base + 0x0044 (EBI_CAL_OR0); Base + 0x004C (EBI_CAL_OR1); Base + 0x0054 (EBI_CAL_OR2); Base + 0x005C (EBI_CAL_OR3)															

Figure 12-6. EBI Option Registers 0–3 (EBI_OR*n*) and EBI Calibration Option Registers

Table 12-12. EBI_OR*n* and EBI_CAL_OR*n* Field Descriptions

Bits	Name	Description
0–16	AM [0:16]	Address mask. Allows masking of any corresponding bits in the associated base register. Masking the address independently allows external devices of different size address ranges to be used. Any clear bit masks the corresponding address bit. Any set bit causes the corresponding address bit to be used in comparison with the address pins. Address mask bits can be set or cleared in any order in the field, allowing a resource to reside in more than one area of the address map. This field can be read or written at any time. **Note:** The upper 3 bits of the address mask (AM) field, EBI_ORx[0:2], are tied to a fixed value of 111. These bits reset to their fixed value.
17–23	—	Reserved.
24–27	SCY [0:3]	Cycle length in clocks. Represents the number of wait states (external bus cycles) inserted after the address phase in the single cycle case, or in the first beat of a burst, when the memory controller handles the external memory access. Values range from 0 to 15. This is the main parameter for determining the length of the cycle. • The total cycle length for the first beat (including the $\overline{TS}$ cycle): (2 + SCY) external clock cycles See Section 12.5.3.1, “Example Wait State Calculation”.
28	—	Reserved.
29–30	BSCY [0:1]	Burst beats length in clocks. This field determines the number of wait states (external bus cycles) inserted in all burst beats except the first, when the memory controller starts handling the external memory access and thus is using SCY[0:3] to determine the length of the first beat. • Total memory access length for each beat: (1 + BSCY) External Clock Cycles • Total cycle length (including the $\overline{TS}$ cycle): (2 + SCY) + [(Number of Beats – 1) x (BSCY + 1)] **Note:** The number of beats (4, 8, 16) is determined by BL and PS bits in the base register. 00 0-clock cycle wait states (1 clock per data beat) 01 1-clock cycle wait states (2 clocks per data beat) 10 2-clock cycle wait states (3 clocks per data beat) 11 3-clock cycle wait states (4 clocks per data beat)

12.4 Functional Description

12.4.1 External Bus Interface Features

12.4.1.1 32-Bit Address Bus with Transfer Size Indication

The transfer size for an external transaction is indicated by the TSIZ[0:1] signals during the clock where address is valid. Valid transaction sizes are 8, 16, and 32 bits. In the MPC5554 and in the 416 BGA package of the MPC5553, only 24 address lines are pinned out externally, but a full 32-bit decode is done internally

to determine the target of the transaction and whether a chip select should be asserted. The 324 BGA package of the MPC5553 has 20 address lines penned out. The 208 package has no external bus.

12.4.1.2 32-Bit Data Bus

The entire 32-bit data bus is available for both external memory accesses and transactions involving an external master in the MPC5554 and in the 416 BGA package of the MPC5553. In the 324 BGA package of the MPC5553, the data bus is 16 bits.

12.4.1.3 16-Bit Data Bus

A 16-bit data bus mode is available via the DBM bit in EBI_MCR. See Section 12.1.4.5, "16-Bit Data Bus Mode."

12.4.1.4 Support for External Master Accesses to Internal Addresses

The EBI allows an external master to access internal address space when the EBI is configured for external master mode in the EBI_MCR. External master operations are described in detail in Section 12.4.2.10, "Bus Operation in External Master Mode."

12.4.1.5 Memory Controller with Support for Various Memory Types

The EBI contains a memory controller that supports a variety of memory types, including synchronous burst mode Flash and external SRAM, and asynchronous/legacy Flash and external SRAM with a compatible interface.

Each $\overline{CS}$ bank is configured via its own pair of base and option registers. Each time an internal to external bus cycle access is requested, the internal address is compared with the base address of each valid base register (with 17 bits having mask). See Figure 12-7. If a match is found, the attributes defined for this bank in its BR and OR are used to control the memory access. If a match is found in more than one bank, the lowest bank matched handles the memory access. For example, bank 0 is selected over bank 1.

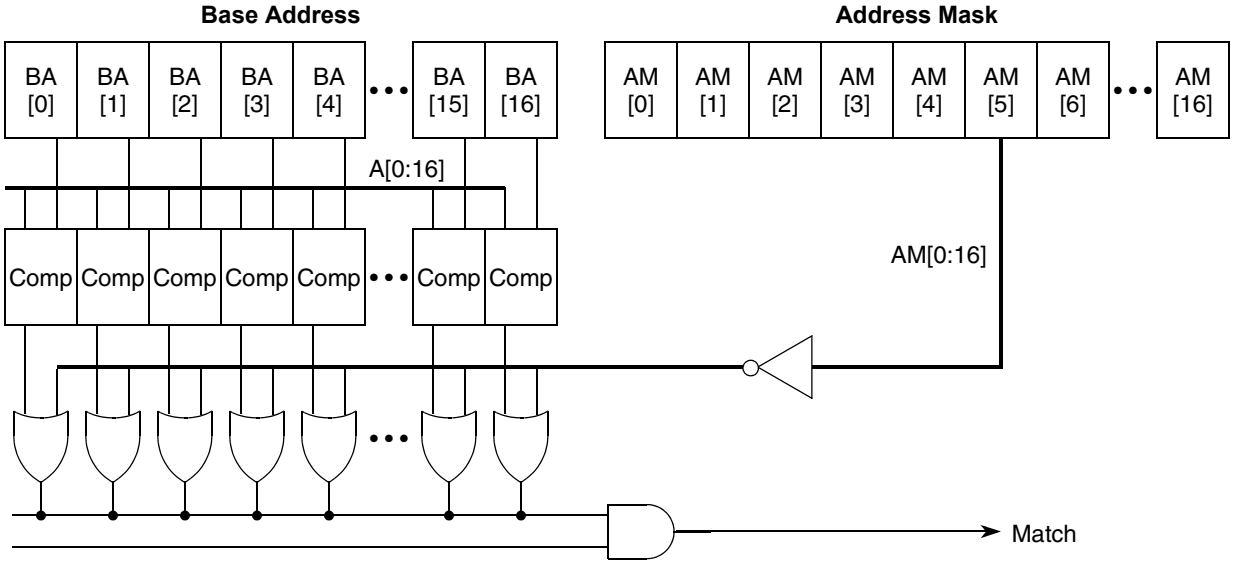


Figure 12-7. Bank Base Address and Match Structure

A match on a valid calibration chip select register overrides a match on any non-calibration chip select register, with CAL_CS0 having the highest priority. Thus the full priority of the chip selects is: CAL_CS0....CAL_CS3, CS0....CS3.

When a match is found on one of the chip select banks, all its attributes (from the appropriate base and option registers) are selected for the functional operation of the external memory access, such as:

- Number of wait states for a single memory access, and for any beat in a burst access
- Burst enable
- Port size for the external accessed device

See Section 12.3.1.6, "EBI Base Registers 0–3 (EBI_BRn) and EBI Calibration Base Registers 0–3 (EBI_CAL_BRn)," and Section 12.3.1.7, "EBI Option Registers 0–3 (EBI_ORn) and EBI Calibration Option Registers 0-3 (EBI_CAL_ORn)," for a full description of all chip select attributes.

When no match is found on any of the chip select banks, the default transfer attributes shown in Table 12-13 are used.

Table 12-13. Default Attributes for Non-Chip Select Transfers

CS Attribute	Default Value	Comment
PS	0	32-bit port size
BL	0	Burst length is don't care since burst is disabled
WEBS	0	Write enables
TBDIP	0	Don't care since burst is disabled
BI	1	Burst inhibited
SCY	0	Don't care since external $\overline{TA}$ is used
BSCY	0	Don't care since external $\overline{TA}$ is used

12.4.1.6 Burst Support (Wrapped Only)

The EBI supports burst read accesses of external burstable memory. To enable bursts to a particular memory region, clear the BI (Burst Inhibit) bit in the appropriate base register. External burst lengths of 4 and 8 words are supported. Burst length is configured for each chip select by using the BL bit in the appropriate base register. See Section 12.4.2.5, "Burst Transfer" for more details.

In 16-bit data bus mode (EBI_MCR[DBM]=1), a special 2-beat burst case is supported for reads and writes for 32-bit non-chip select accesses only. This is to allow 32-bit coherent accesses to another MCU. See Section 12.4.2.11, "Non-Chip-Select Burst in 16-bit Data Bus Mode".

Bursting of accesses that are not controlled by the chip selects is not supported for any other case besides the special case of 32-bit accesses in 16-bit data bus mode.

Burst writes are not supported for any other case besides the special case of 32-bit non-chip select writes in 16-bit data bus mode. Internal requests to write more than 32 bits (such as a cache line) externally are broken up into separate 32-bit or 16-bit external transactions according to the port size. See Section 12.4.2.6, "Small Accesses (Small Port Size and Short Burst Length)" for more detail on these cases.

12.4.1.7 Bus Monitor

When enabled (via the BME bit in the EBI_BMCR), the bus monitor detects when no $\overline{TA}$ assertion is received within a maximum timeout period for non-chip select accesses (that is, accesses that use external $\overline{TA}$). The timeout for the bus monitor is specified by the BMT field in the EBI_BMCR. Each time a timeout error occurs, the BMTF bit is set in the EBI_TESR. The timeout period is measured in external bus (CLKOUT) cycles. Thus the effective real-time period is multiplied (by 2 or 4) when a configurable bus speed mode is used, even though the BMT field itself is unchanged.

12.4.1.8 Port Size Configuration per Chip Select (16 or 32 Bits)

The EBI supports memories with data widths of 16 or 32 bits. The port size for a particular chip select is configured by writing the PS bit in the corresponding base register.

12.4.1.9 Port Size Configuration per Calibration Chip Select (16 Bits)

The port size for calibration must be 16 bits wide.

12.4.1.10 Configurable Wait States

From 0 to 15 wait states can be programmed for any cycle that the memory controller generates, via the SCY bits in the appropriate option register. From zero to three wait states between burst beats can be programmed using the BSCY bits in the appropriate option register.

12.4.1.11 Four Chip Select ($\overline{CS}$[0:3]) Signals

The EBI contains four chip select signals, controlling four independent memory banks. See Section 12.4.1.5, "Memory Controller with Support for Various Memory Types," for more details on chip select bank configuration.

12.4.1.12 Support for Dynamic Calibration with Up to 4 Chip Selects

The EBI contains four calibration chip select signals, controlling four independent memory banks on an optional second external bus for calibration. See Section 12.4.2.12, "Calibration Bus Operation — MPC5553 Only" for more details on using the calibration bus.

12.4.1.13 Four Write/Byte Enable ($\overline{WE}$/$\overline{BE}$) Signals — Only MPC5554 and 416 BGA of MPC5553

In the MPC5554 and in the 416 BGA of the MPC5553, the functionality of the $\overline{WE}$[0:3]/$\overline{BE}$[0:3] signals depends on the value of the WEBS bit in the corresponding base register. Setting WEBS to 1 configures these pins as $\overline{BE}$[0:3], while resetting it to 0 configures them as $\overline{WE}$[0:3]. $\overline{WE}$[0:3] are asserted only during write accesses, while $\overline{BE}$[0:3] is asserted for both read and write accesses. The timing of the $\overline{WE}$[0:3]/$\overline{BE}$[0:3] signals remains the same in either case.

The upper write/byte enable ($\overline{WE0}$/$\overline{BE0}$) indicates that the upper eight bits of the data bus (DATA[0:7]) contain valid data during a write/read cycle. The upper middle write/byte enable ($\overline{WE1}$/$\overline{BE1}$) indicates that the upper middle eight bits of the data bus (DATA[8:15]) contain valid data during a write/read cycle. The lower middle write/byte enable ($\overline{WE2}$/$\overline{BE2}$) indicates that the lower middle eight bits of the data bus (DATA[16:23]) contain valid data during a write/read cycle. The lower write/byte enable ($\overline{WE3}$/$\overline{BE3}$) indicates that the lower eight bits of the data bus (DATA[24:31]) contain valid data during a write/read cycle.

The write/byte enable lines affected in a transaction for a 32-bit port (PS = 0) and a 16-bit port (PS = 1) are shown in Table 12-14. Only big endian byte ordering is supported by the EBI.

Table 12-14. Write/Byte Enable Signals Function -- 416 BGA

Transfer Size	TSIZ[0:1]	Address		32-Bit Port Size				16-Bit Port Size[1]			
		A30	A31	WE0/ BE0	WE1/ BE1	WE2/ BE2	WE3/ BE3	WE0/ BE0	WE1/ BE1	WE2/ BE2	WE3/ BE3
Byte	01	0	0	X	—	—	—	X	—	—	—
	01	0	1	—	X	—	—	—	X	—	—
	01	1	0	—	—	X	—	X	—	—	—
	01	1	1	—	—	—	X		X	—	—
16-bit	10	0	0	X	X	—	—	X	X	—	—
	10	1	0	—	—	X	X	X	X	—	—
32-bit	00	0	0	X	X	X	X	X[2]	X[2]	—	—
Burst	00	0	0	X	X	X	X	X	X	—	—

1 Also applies when DBM=1 for 16-bit data bus mode.

2 This case consists of two 16-bit external transactions, but for both transactions the WE[0:1]/BE[0:1] signals are the only WE/BE signals affected.

NOTE: All areas of the table, both shaded and clear, apply to the 416 BGA package of the MPC5553 and to the MPC5554.

NOTE: “X” indicates that valid data is transferred on these bits.

12.4.1.14 Two Write/Byte Enable (WE/BE) Signals — 324 BGA of MPC5553 Only

In the 324 BGA of the MPC5553, the functionality of the WE[0:1]/BE[0:1] signals depends on the value of the WEBS bit in the corresponding base register. Setting WEBS to 1 configures these pins as BE[0:1], while resetting it to 0 configures them as WE[0:1]. WE[0:1] are asserted only during write accesses, while BE[0:1] is asserted for both read and write accesses. The timing of the WE[0:1]/BE[0:1] signals remains the same in either case.

The upper write/byte enable (WE0/BE0) indicates that the upper eight bits of the data bus (DATA[0:7]) contain valid data during a write/read cycle. The lower write/byte enable (WE1/BE1) indicates that the lower eight bits of the data bus (DATA[8:15]) contain valid data during a write/read cycle.

The write/byte enable lines affected in a transaction are shown below in Table 12-15. Only big endian byte ordering is supported by the EBI.

Table 12-15. Write/Byte Enable Signals Function -- 324 BGA

Transfer Size	TSIZ[0:1]	Address		16-Bit Port Size[1]	
		A30	A31	$\overline{WE}$0/ $\overline{BE}$0	$\overline{WE}$1/ $\overline{BE}$1
Byte	01	0	0	X	—
	01	0	1	—	X
	01	1	0	X	—
	01	1	1	—	X
16-bit	10	0	0	X	X
	10	1	0	X	X
32-bit	00	0	0	X[2]	X[2]
Burst	00	0	0	X	X

1 Also applies when DBM=1 for 16-bit data bus mode.

2 This case consists of two 16-bit external transactions, but for both transactions the $\overline{WE}$[0:1]/$\overline{BE}$[0:1] signals are the only $\overline{WE}$/$\overline{BE}$ signals affected.

NOTE: “X” indicates that valid data is transferred on these bits.

12.4.1.15 Configurable Bus Speed Clock Modes

The EBI supports configurable bus speed clock modes. Refer to Section 12.1.4.4, “Configurable Bus Speed Modes,” for more details on this feature.

12.4.1.16 Stop and Module Disable Modes for Power Savings

See Section 12.1.4, “Modes of Operation,” for a description of the power saving modes.

12.4.1.17 Optional Automatic CLKOUT Gating

The EBI has the ability to hold the external CLKOUT pin high when the EBI’s internal master state machine is idle and no requests are pending. The EBI outputs a signal to the pads logic in the MCU to disable CLKOUT. This feature is disabled out of reset, and can be enabled or disabled by the ACGE bit in the EBI_MCR.

NOTE

This feature must be disabled for multi-master systems. In those cases, one master is getting its clock source from the other master and needs the other master to stay valid continuously.

12.4.1.18 Compatible with MPC5xx External Bus (with Some Limitations)

The EBI is compatible with the external bus of the MPC5xx parts, meaning that it supports most devices supported by the MPC5xx family of parts. However, there are some differences between this EBI and that of the MPC5xx parts that the user needs to be aware of before assuming that an MPC5xx-compatible device works with this EBI. See Section 12.5.6, "Summary of Differences from MPC5xx," for details.

NOTE

Due to testing and complexity concerns, multi-master (or master/slave) operation between an eSys MCU and MPC5xx is not guaranteed.

12.4.2 External Bus Operations

The following sections provide a functional description of the external bus, the bus cycles provided for data transfer operations, bus arbitration, and error conditions.

12.4.2.1 External Clocking

The CLKOUT signal sets the frequency of operation for the bus interface directly. Internally, the MCU uses a phase-locked loop (PLL) circuit to generate a master clock for all of the MCU circuitry (including the EBI) which is phase-locked to the CLKOUT signal. In general, all signals for the EBI are specified with respect to the rising-edge of the CLKOUT signal, and they are guaranteed to be sampled as inputs or changed as outputs with respect to that edge.

12.4.2.2 Reset

Upon detection of internal reset, the EBI immediately terminates all transactions.

12.4.2.3 Basic Transfer Protocol

The basic transfer protocol defines the sequence of actions that must occur on the external bus to perform a complete bus transaction. A simplified scheme of the basic transfer protocol is shown in Figure 12-8.

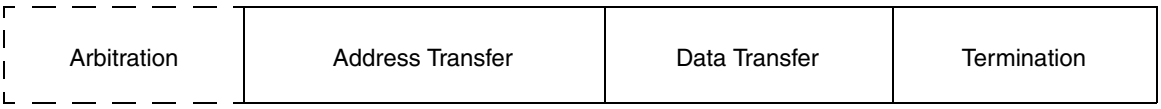


Figure 12-8. Basic Transfer Protocol

The arbitration phase is where bus ownership is requested and granted. This phase is not needed in single master mode because the EBI is the permanent bus owner in this mode. Arbitration is discussed in detail in Section 12.4.2.8, "Arbitration."

The address transfer phase specifies the address for the transaction and the transfer attributes that describe the transaction. The signals related to the address transfer phase are TS, ADDR, CS[0:3], RD_WR, TSIZ[0:1], and BDIP. The address and its related signals (with the exception of TS, BDIP) are driven on the bus with the assertion of the TS signal, and kept valid until the bus master receives TA asserted (the EBI holds them one cycle beyond TA for writes and external TA accesses). Note that for writes with internal TA, RD_WR is not held one cycle past TA.

The data transfer phase performs the transfer of data, from master to slave (in write cycles) or from slave to master (on read cycles), if any is to be transferred. The data phase may transfer a single beat of data (1-4 bytes) for non-burst operations or a 2-beat (special EBI_MCR[DBM]=1 case only), 4-beat, 8-beat, or 16-beat burst of data (2 or 4 bytes per beat depending on port size) when burst is enabled. On a write cycle, the master must not drive write data until after the address transfer phase is complete. This is to avoid electrical contentions when switching between drivers. The master must start driving write data one cycle after the address transfer cycle. The master can stop driving the data bus as soon as it samples the $\overline{TA}$ line asserted on the rising edge of CLKOUT. To facilitate asynchronous write support, the EBI keeps driving valid write data on the data bus until 1 clock after the rising edge where RD_$\overline{WR}$ and $\overline{WE}$ are negated (for chip select accesses only). See Figure 12-14 for an example of write timing. On a read cycle, the master accepts the data bus contents as valid on the rising edge of the CLKOUT in which the $\overline{TA}$ signal is sampled asserted. See Figure 12-10 for an example of read timing.

The termination phase is where the cycle is terminated by the assertion of either $\overline{TA}$ (normal termination) or $\overline{TEA}$ (termination with error). Termination is discussed in detail in Section 12.4.2.9, "Termination Signals Protocol."

12.4.2.4 Single Beat Transfer

The flow and timing diagrams in this section assume that the EBI is configured in single master mode. Therefore, arbitration is not needed and is not shown in these diagrams. Refer to Section 12.4.2.10, "Bus Operation in External Master Mode," to see how the flow and timing diagrams change for external master mode.

12.4.2.4.1 Single Beat Read Flow

The handshakes for a single beat read cycle are illustrated in the following flow and timing diagrams.

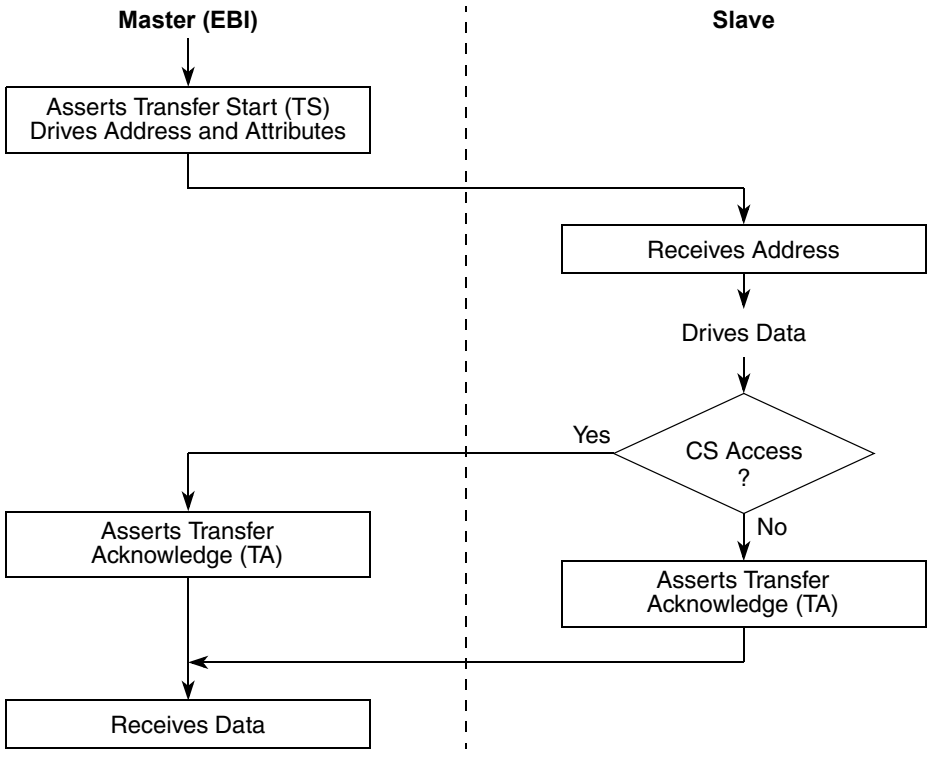


Figure 12-9. Basic Flow Diagram of a Single Beat Read Cycle

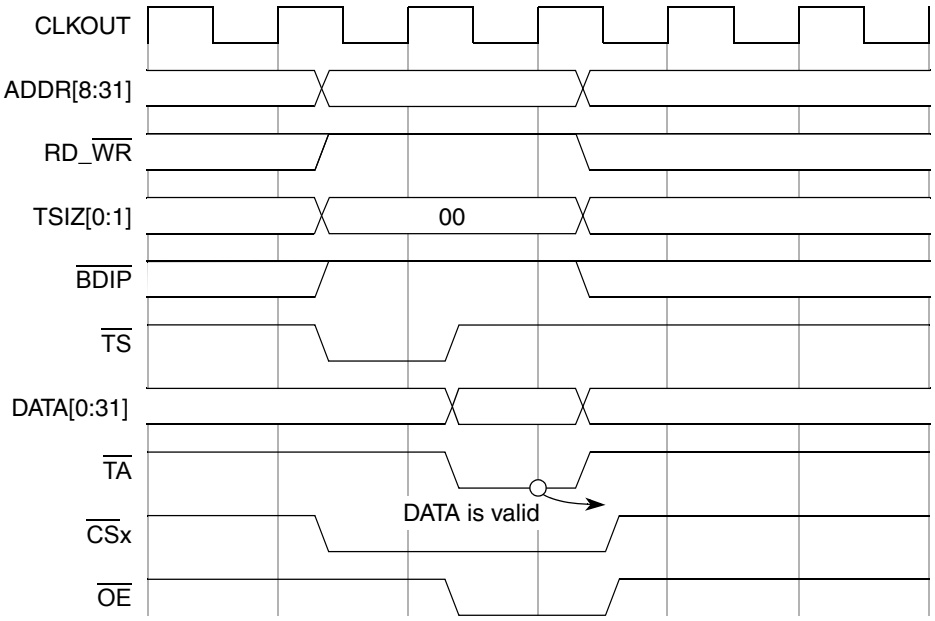


Figure 12-10. Single Beat 32-bit Read Cycle, CS Access, Zero Wait States

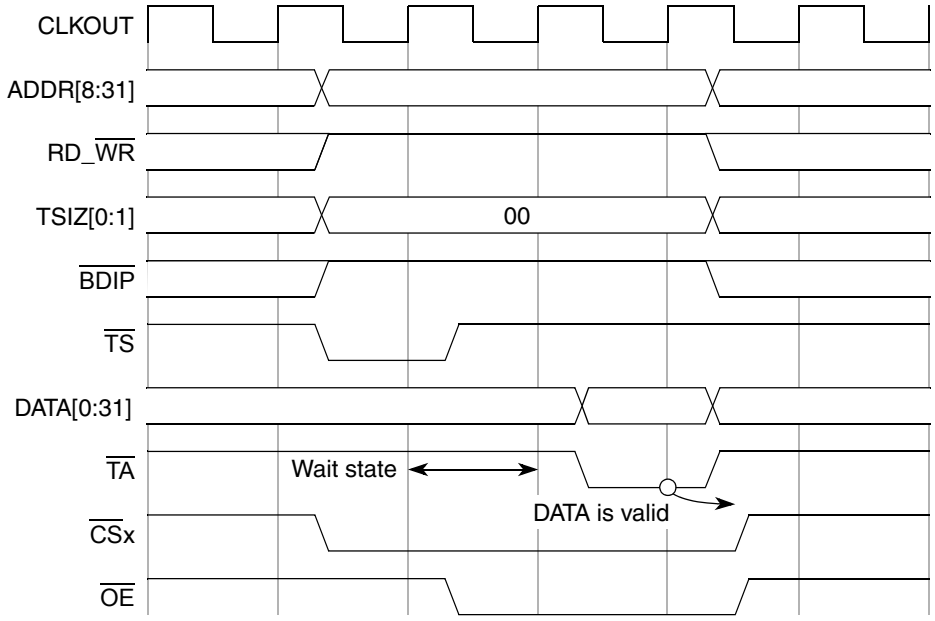
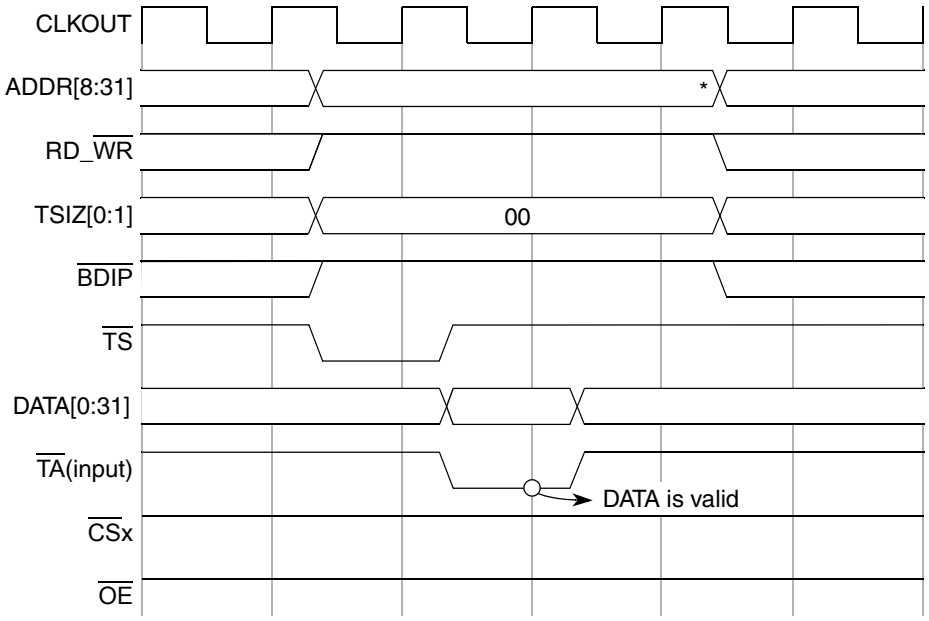


Figure 12-11. Single Beat 32-bit Read Cycle, CS Access, One Wait State



* The EBI drives address and control signals an extra cycle because it uses a latched version of the external TA (1 cycle delayed) to terminate the cycle.

Figure 12-12. Single Beat 32-bit Read Cycle, Non-CS Access, Zero Wait States

12.4.2.4.2 Single Beat Write Flow

The handshakes for a single beat write cycle are illustrated in the following flow and timing diagrams.

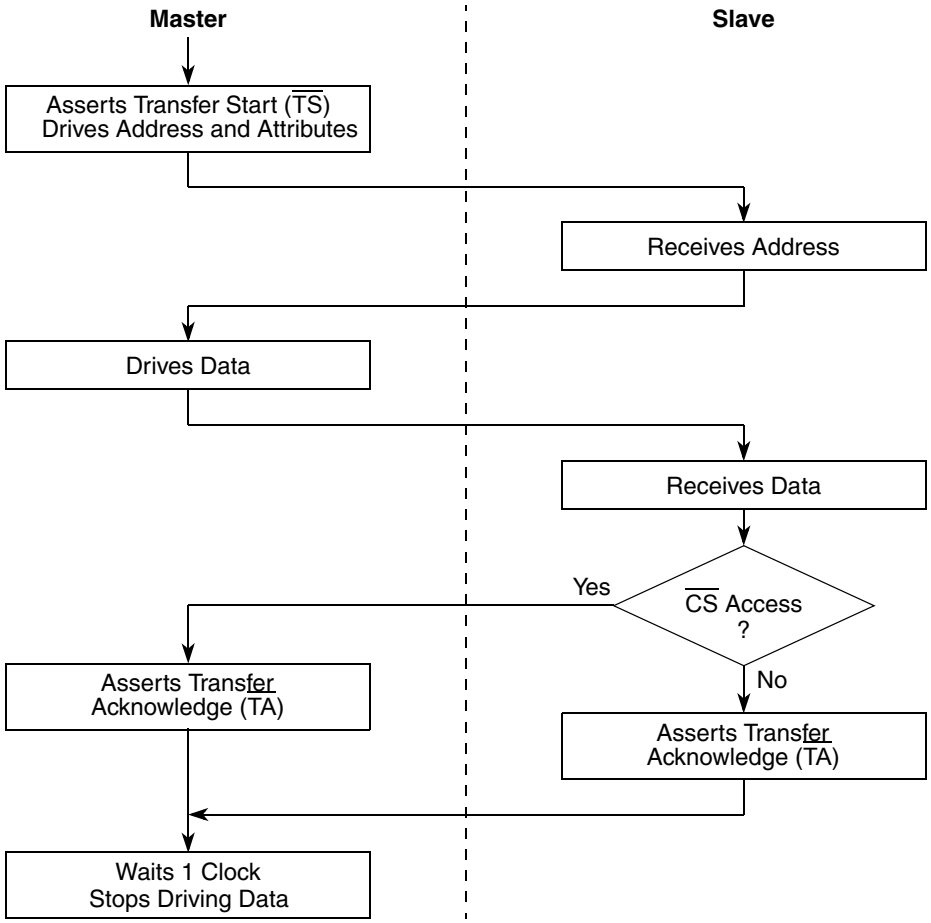


Figure 12-13. Basic Flow Diagram of a Single Beat Write Cycle

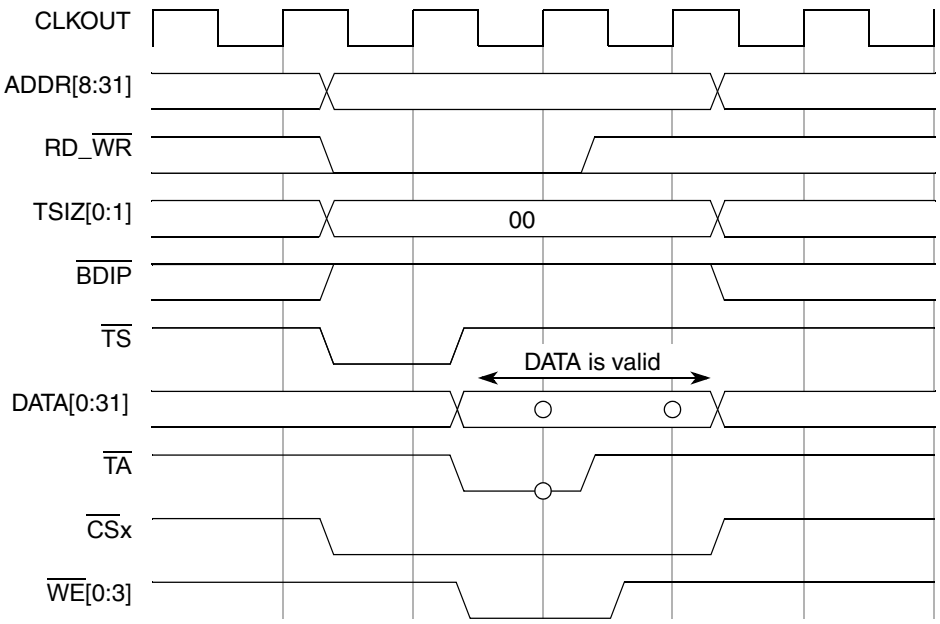


Figure 12-14. Single Beat 32-bit Write Cycle, CS Access, Zero Wait States

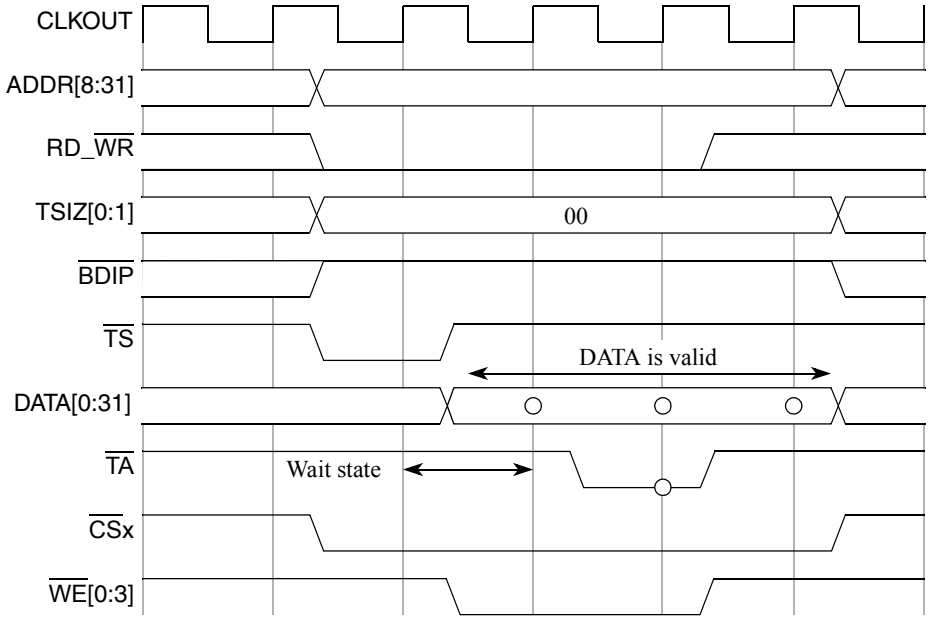
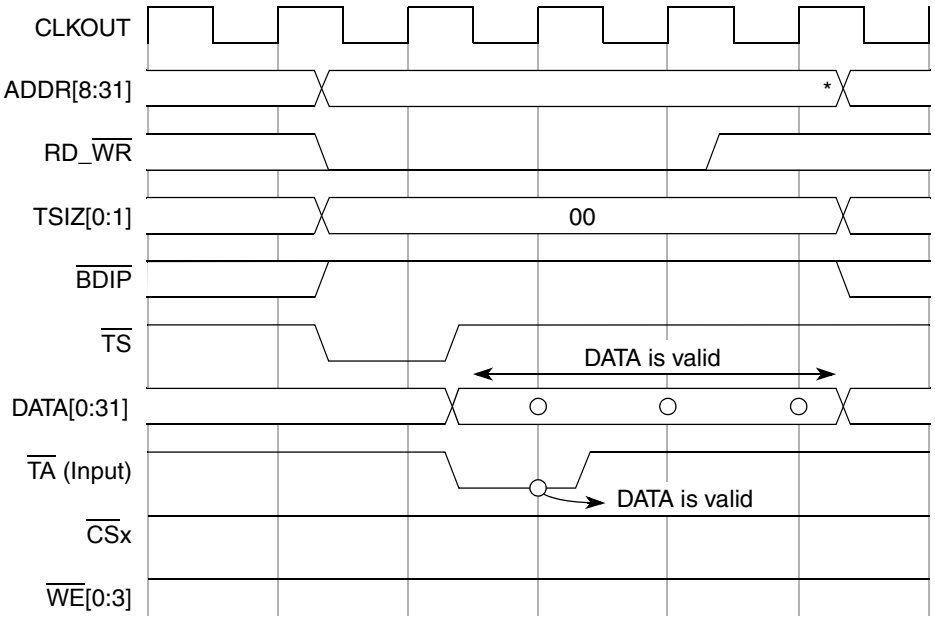


Figure 12-15. Single Beat 32-bit Write Cycle, CS Access, One Wait State



* The EBI drives address and control signals an extra cycle because it uses a latched version of the external TA (1 cycle delayed) to terminate the cycle.

Figure 12-16. Single Beat 32-bit Write Cycle, Non-CS Access, Zero Wait States

12.4.2.4.3 Back-to-Back Accesses

Due to internal bus protocol, one dead cycle is necessary between back-to-back external bus accesses that are not part of a set of small accesses (see Section 12.4.2.6, "Small Accesses (Small Port Size and Short Burst Length)" for small access timing). Besides this dead cycle, in most cases, back-to-back accesses on the external bus do not cause any change in the timing from that shown in the previous diagrams, and the two transactions are independent of each other. The only exceptions to this are as follows:

- Back-to-back accesses where the first access ends with an externally-driven TA or TEA. In these cases, an extra cycle is required between the end of the first access and the TS assertion of the second access. See Section 12.4.2.9, "Termination Signals Protocol," for more details.

The following diagrams show a few examples of back-to-back accesses on the external bus.

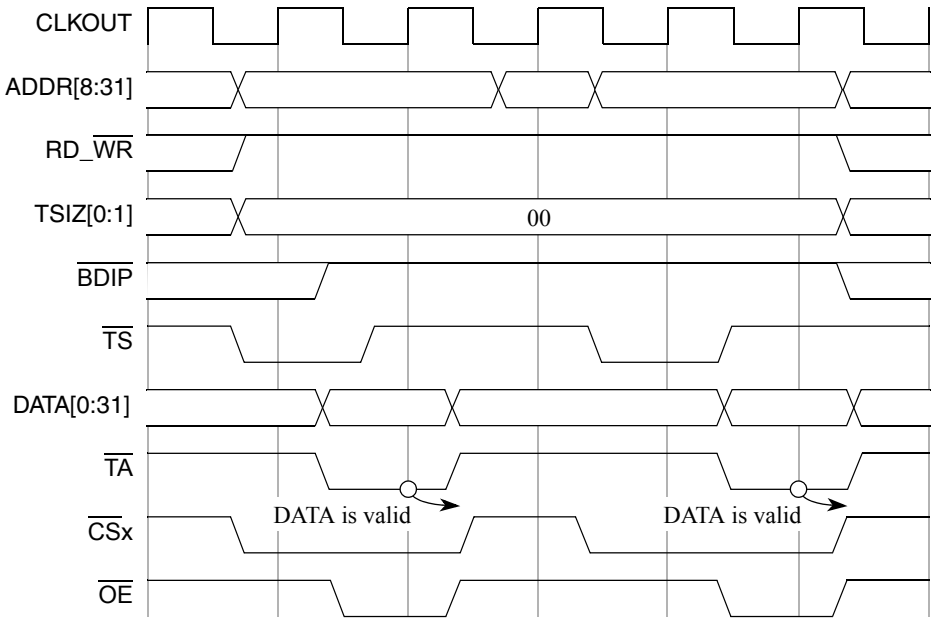


Figure 12-17. Back-to-Back 32-bit Reads to the Same $\overline{CS}$ Bank

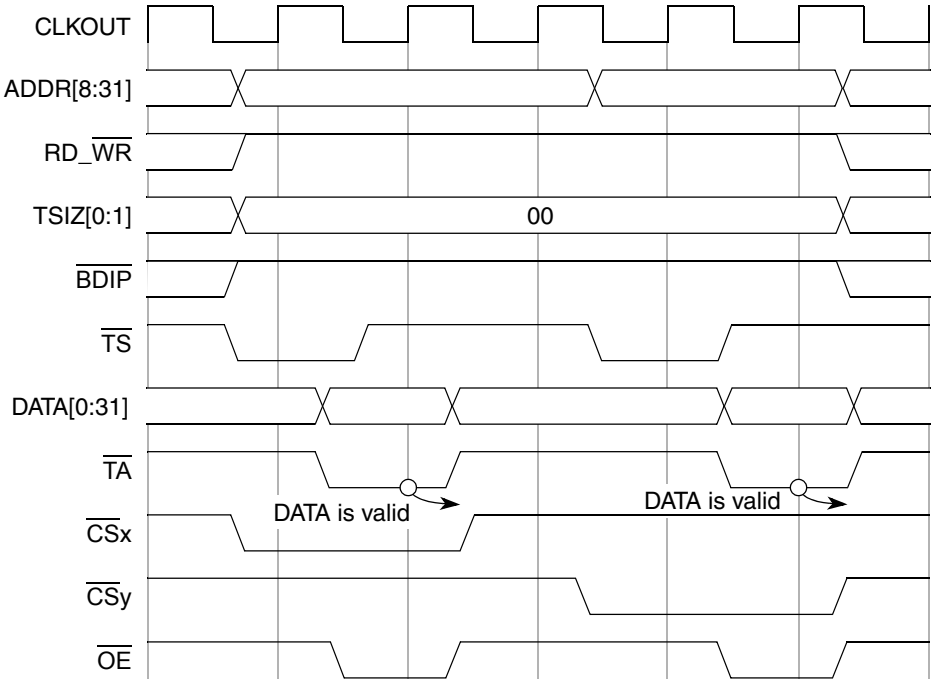


Figure 12-18. Back-to-Back 32-bit Reads to Different $\overline{CS}$ Banks

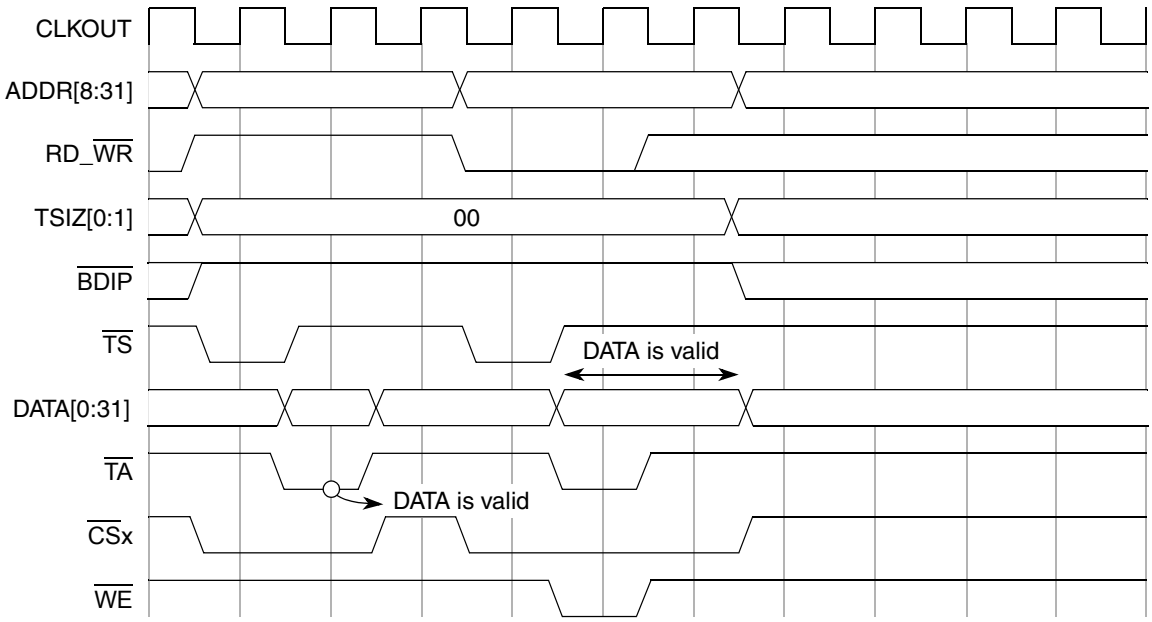


Figure 12-19. Write After Read to the Same $\overline{CS}$ Bank

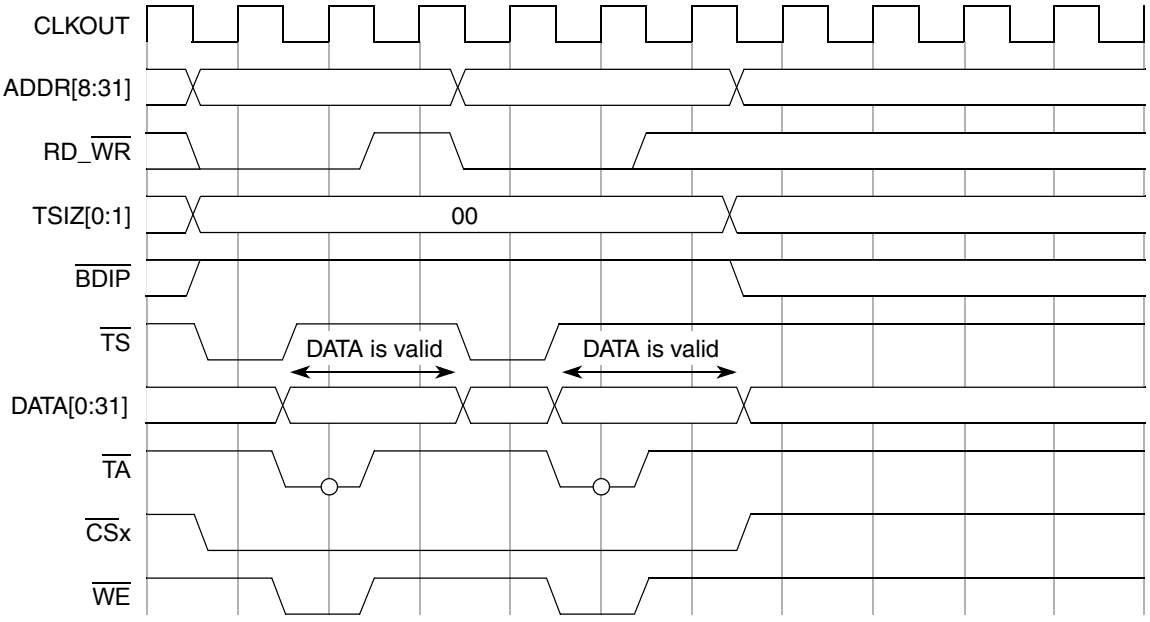


Figure 12-20. Back-to-Back 32-bit Writes to the Same $\overline{CS}$ Bank

12.4.2.5 Burst Transfer

The EBI supports wrapping 32-byte critical-doubleword-first burst transfers. Bursting is supported only for internally-requested cache-line size (32-byte) read accesses to external devices that use the chip selects[1]. Accesses from an external master or to devices operating without a chip select are always single

1. Except for the special case of a 32-bit non-chip select access in 16-bit data bus mode. See Section 12.4.2.11.

beat. If an internal request to the EBI indicates a size of less than 32 bytes, the request is fulfilled by running one or more single-beat external transfers, not by an external burst transfer.

An 8-word wrapping burst reads eight 32-bit words by supplying a starting address that points to one of the words (doubleword aligned) and requiring the memory device to sequentially drive each word on the data bus. The selected slave device must internally increment ADDR[27:29] (also ADDR30 in the case of a 16-bit port size device) of the supplied address for each transfer, until the address reaches an 8-word boundary, and then wrap the address to the beginning of the 8-word boundary. The address and transfer attributes supplied by the EBI remain stable during the transfers, and the EBI terminates each beat transfer by asserting TA. The EBI requires that addresses be aligned to a doubleword boundary on all burst cycles.

Table 12-16 shows the burst order of beats returned for an 8-word burst to a 32-bit port.

Table 12-16. Wrap Bursts Order

Burst Starting Address ADDR[27:28]	Burst Order (Assuming 32-bit Port Size)
00	word0 –> word1 –> word2 –> word3 –> word4 –> word5 –> word6 –> word7
01	word2 –> word3 –> word4 –> word5 –> word6 –> word7 –> word0 –> word1
10	word4 –> word5 –> word6 –> word7 –> word0 –> word1 –> word2 –> word3
11	word6 –> word7 –> word0 –> word1 –> word2 –> word3 –> word4 –> word5

The general case of burst transfers assumes that the external memory has 32-bit port size and 8-word burst length. The EBI can also burst from 16-bit port size memories, taking twice as many external beats to fetch the data as compared to a 32-bit port with the same burst length. The EBI can also burst from 16-bit or 32-bit memories that have a 4-word burst length (BL = 1 in the appropriate base register). In this case, two external 4-word burst transfers (wrapping on 4-word boundary) are performed to fulfill the internal 8-word request. This operation is considered atomic by the EBI, so the EBI does not allow other unrelated master accesses or bus arbitration to intervene between the transfers. For more details and a timing diagram, see Section 12.4.2.6.3, “Small Access Example #3: 32-byte Read to 32-bit Port with BL = 1.”

During burst cycles, the $\overline{\text{BDIP}}$ (burst data in progress) signal is used to indicate the duration of the burst data. During the data phase of a burst read cycle, the EBI receives data from the addressed slave. If the EBI needs more than one data, it asserts the $\overline{\text{BDIP}}$ signal. Upon receiving the data prior to the last data, the EBI negates $\overline{\text{BDIP}}$. Thus, the slave stops driving new data after it receives the negation of $\overline{\text{BDIP}}$ on the rising edge of the clock. Some slave devices have their burst length and timing configurable internally and thus may not support connecting to a $\overline{\text{BDIP}}$ pin. In this case, $\overline{\text{BDIP}}$ is driven by the EBI normally, but the output is ignored by the memory and the burst data behavior is determined by the internal configuration of the EBI and slave device. When the TBDIP bit is set in the appropriate base register, the timing for $\overline{\text{BDIP}}$ is altered. See Section 12.4.2.5.1, “TBDIP Effect on Burst Transfer,” for this timing.

Since burst writes are not supported by the EBI[1], the EBI negates $\overline{\text{BDIP}}$ during write cycles.

1. Except for the special case of a 32-bit non-chip select access in 16-bit data bus mode. See Section 12.4.2.11.

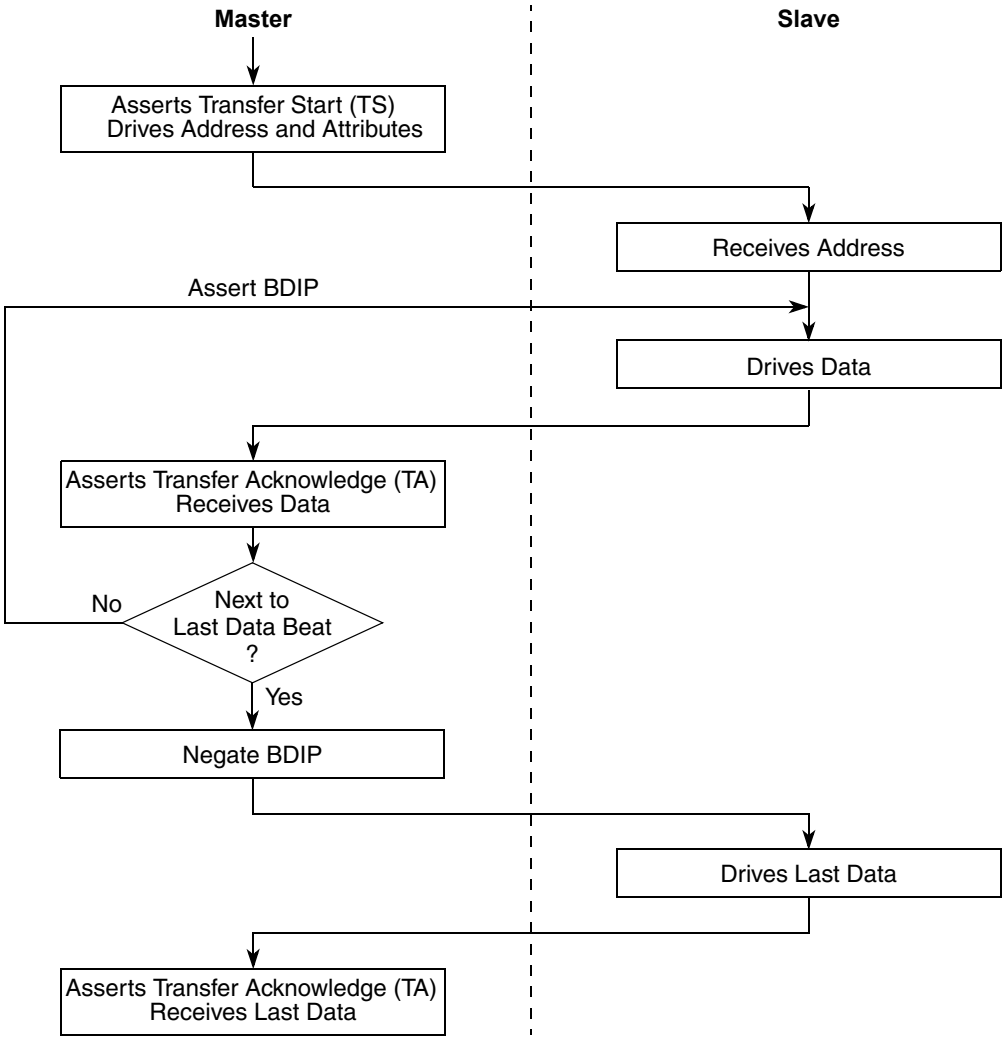


Figure 12-21. Basic Flow Diagram of a Burst Read Cycle

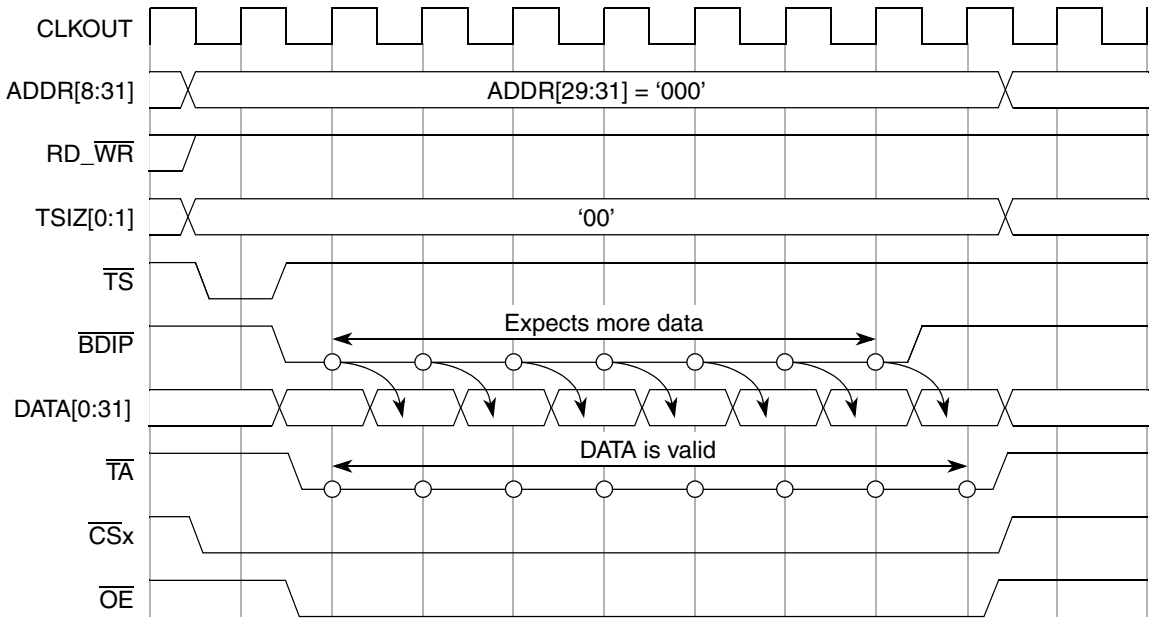


Figure 12-22. Burst 32-bit Read Cycle, Zero Wait States

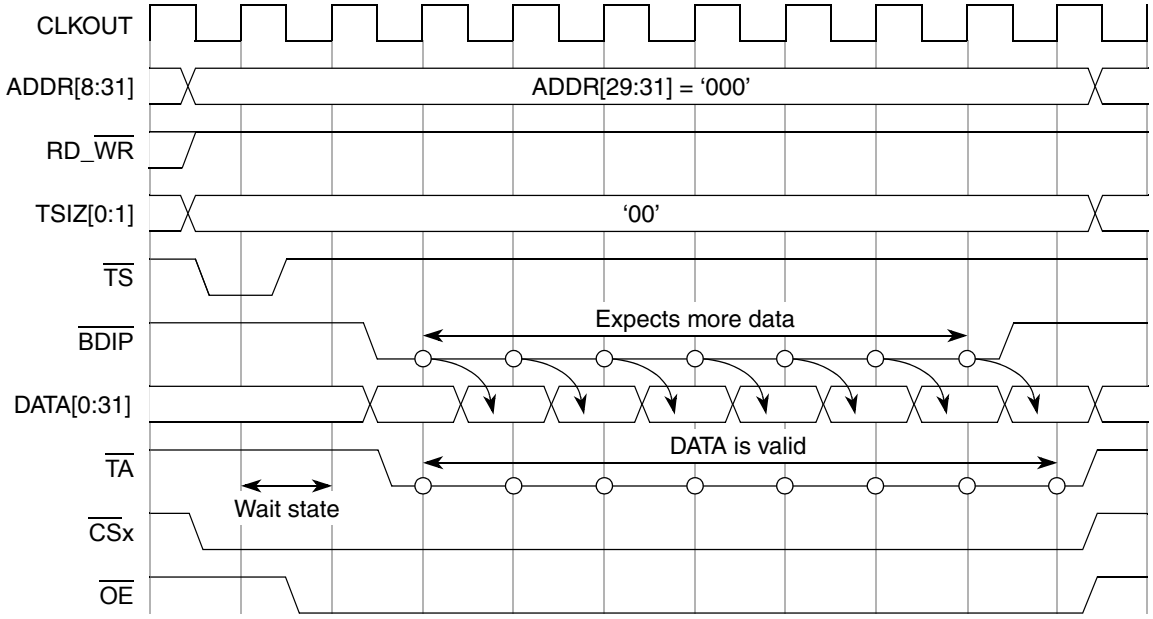


Figure 12-23. Burst 32-bit Read Cycle, One Initial Wait State

12.4.2.5.1 TBDIP Effect on Burst Transfer

Some memories require different timing on the BDIP signal than the default to run burst cycles. Using the default value of TBDIP = 0 in the appropriate EBI base register results in BDIP being asserted (SCY+1) cycles after the address transfer phase, and being held asserted throughout the cycle regardless of the wait

states between beats (BSCY). Figure 12-24 shows an example of the TBDIP = 0 timing for a 4-beat burst with BSCY = 1.

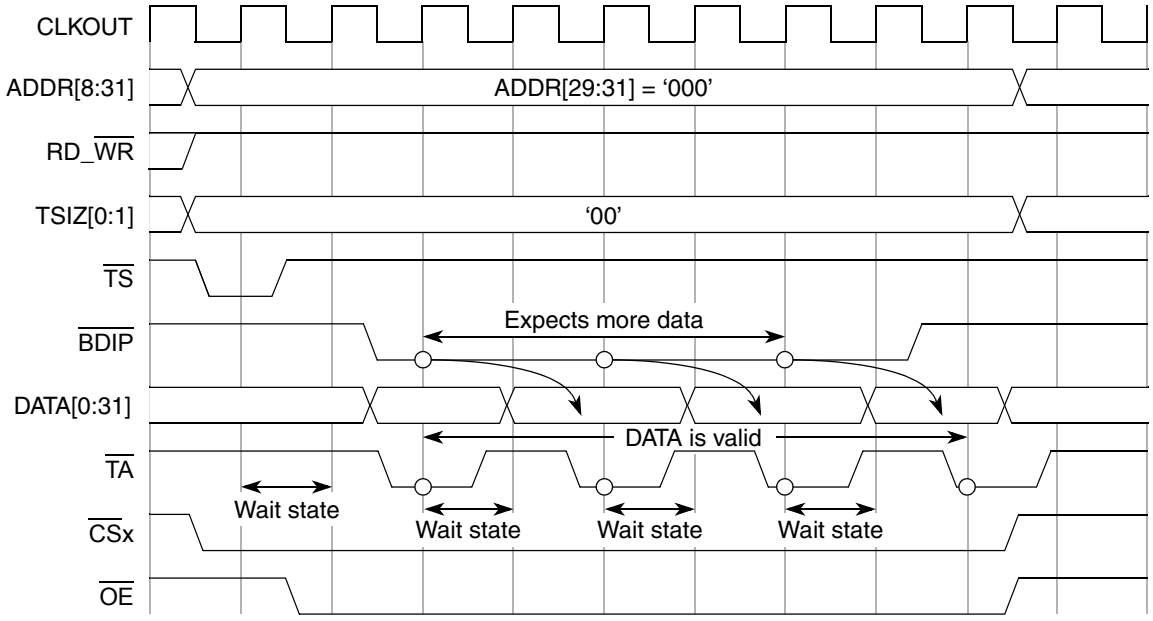


Figure 12-24. Burst 32-bit Read Cycle, One Wait State between Beats, TBDIP = 0

When using TBDIP = 1, the BDIP behavior changes to toggle between every beat when BSCY is a non-zero value. Figure 12-25 shows an example of the TBDIP = 1 timing for the same four-beat burst shown in Figure 12-24.

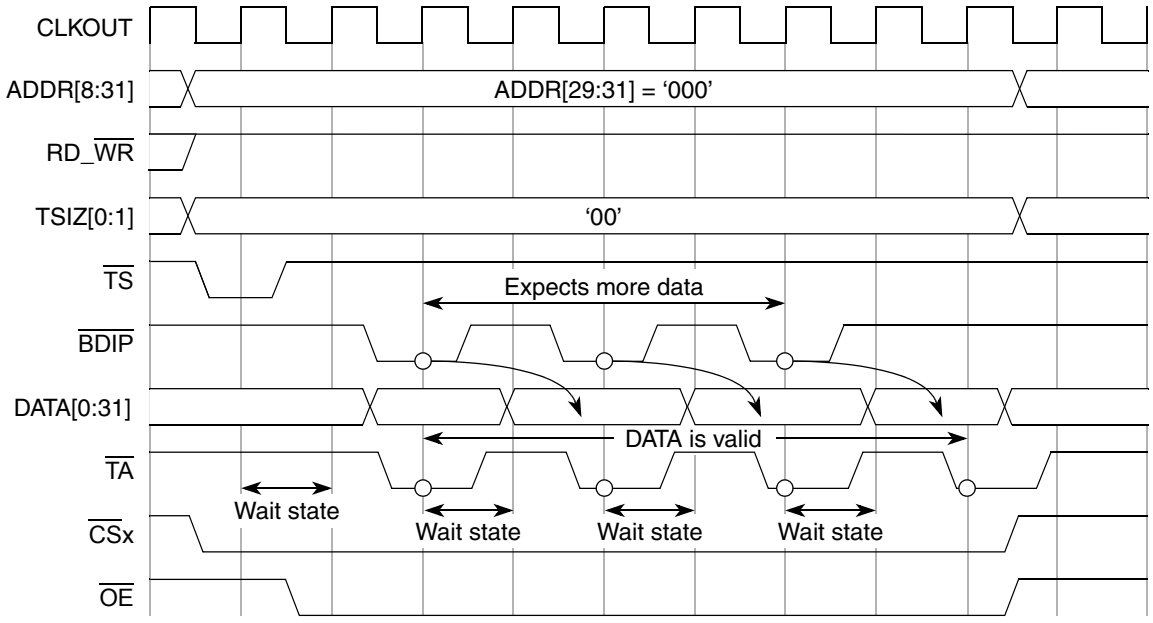


Figure 12-25. Burst 32-bit Read Cycle, One Wait State between Beats, TBDIP = 1

12.4.2.6 Small Accesses (Small Port Size and Short Burst Length)

In this context, a small access refers to an access whose burst length and port size are such that the number of bytes requested by the internal master cannot all be fetched (or written) in one external transaction. This is the case when the base register's burst length bit (EBI_BR*n*[BL]) and port size bit (EBI_BR*n*[PS]) are set such that one of two situations occur:

- Burst accesses are inhibited and the number of bytes requested by the master is greater than the port size (16 or 32 bit) can accommodate in a single access.
- Burst accesses are enabled and the number of bytes requested by the master is greater than the selected burst length (4 words or 8 words).

If this is the case, the EBI initiates multiple transactions until all the requested data is transferred. It should be noted that all the transactions initiated to complete the data transfer are considered as an atomic transaction, so the EBI does not allow other unrelated master accesses to intervene between the transfers. In external master mode, this means that the EBI keeps $\overline{BB}$ asserted and does not grant the bus to another master until the atomic transaction is complete.

Table 12-17 shows all the combinations of burst length, port size, and requested byte count that cause the EBI to run multiple external transactions to fulfill the request.

Table 12-17. Small Access Cases

Byte Count Requested by Internal Master	Burst Length	Port Size	# External Accesses to Fulfill Request
Non-Burstable Chip-Select Banks (BI=1) or Non-Chip-Select Access			
4	1 beat	16-bit	2/1[1]
8	1 beat	32-bit	2
8	1 beat	16-bit	4
32	1 beat	32-bit	8
32	1 beat	16-bit	16
Burstable Chip-Select Banks (BI=0)			
32	4 words	16-bit (8 beats), 32-bit (4 beats)	2

[1] In 32-bit data bus mode (DBM=0 in EBI_MCR), two accesses are performed. In 16-bit data bus mode (DBM=1), one 2-beat burst access is performed and this is not considered a small access case. See Section 12.4.2.11, "Non-Chip-Select Burst in 16-bit Data Bus Mode" for this special DBM=1 case.

In most cases, the timing for small accesses is the same as for normal single-beat and burst accesses, except that multiple back-to-back external transfers are executed for each internal request. These transfers have no additional dead cycles in-between that are not present for back-to-back stand-alone transfers except for the case of writes with an internal request size greater than 64 bits, discussed in Section 12.4.2.6.2, "Small Access Example #2: 32-byte Write with External TA."

The following sections show a few examples of small accesses. The timing for the remaining cases in Table 12-17 can be extrapolated from these and the other timing diagrams in this document.

12.4.2.6.1 Small Access Example #1: 32-bit Write to 16-bit Port

Figure 12-26 shows an example of a 32-bit write to a 16-bit port, requiring two 16-bit external transactions.

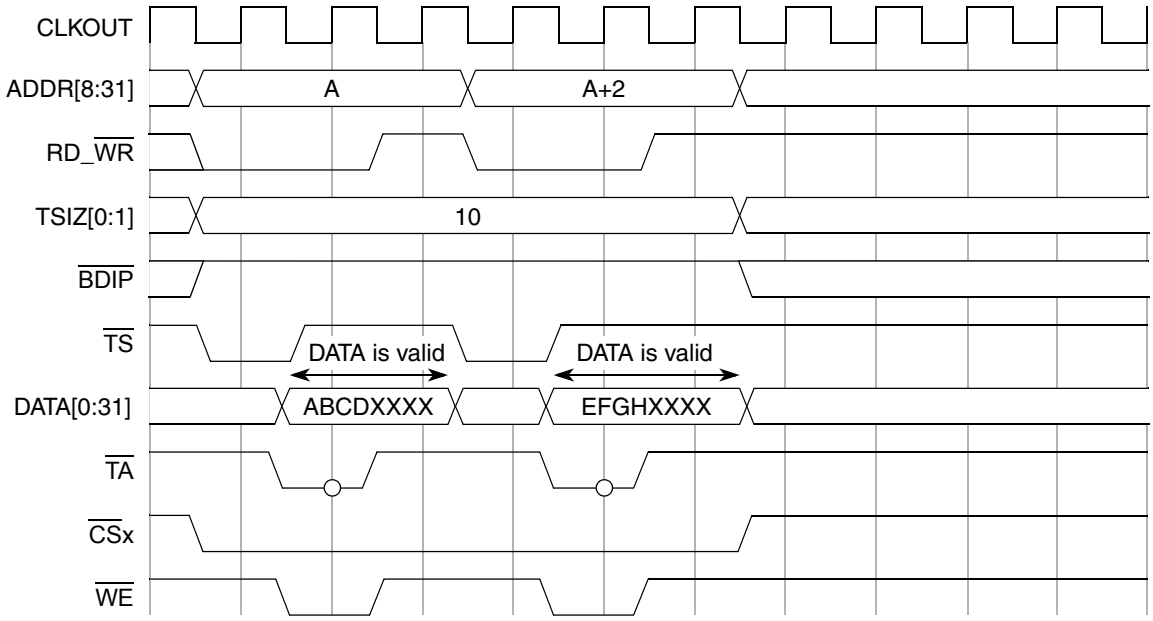
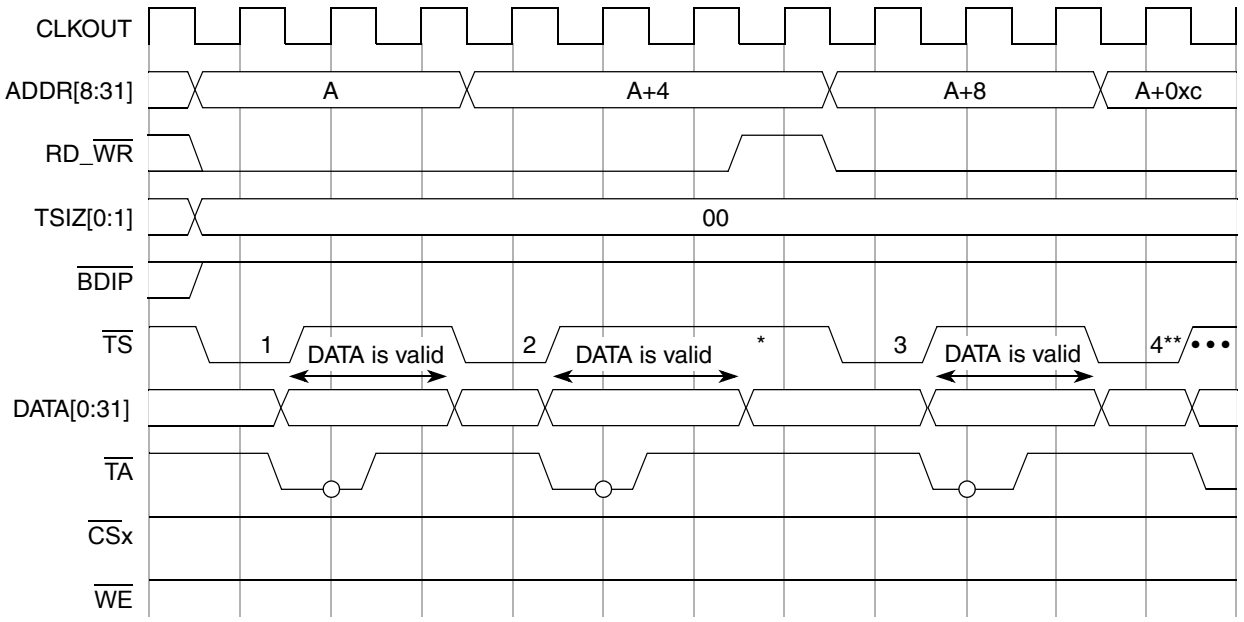


Figure 12-26. Single Beat 32-bit Write Cycle, 16-bit Port Size, Basic Timing

12.4.2.6.2 Small Access Example #2: 32-byte Write with External TA

Figure 12-27 shows an example of a 32-byte write to a non-chip select device, such as an external master, using external TA, requiring eight 32-bit external transactions. Note that due to the use of external TA, RD_WR does not toggle between the accesses unless that access is the end of a 64-bit boundary. In this case, an extra cycle is required between TA and the next TS in order to get the next 64-bits of write data internally and RD_WR negates during this extra cycle.



* This extra cycle is required after accesses 2, 4, and 6 in order to get the next 64-bits of internal write data.

** Four more external accesses (not shown) are required to complete the internal 32-byte request. The timing of these is the same as accesses 1-4 shown in this diagram.

Figure 12-27. 32-Byte Write Cycle with External TA, Basic Timing

12.4.2.6.3 Small Access Example #3: 32-byte Read to 32-bit Port with BL = 1

Figure 12-28 shows an example of a 32-byte read to a 32-bit burst enabled port with burst length of 4 words, requiring two 16-byte external transactions. For this case, the address for the second 4-word burst access is calculated by adding 0x10 to the lower 5 bits of the first address (no carry), and then masking out the lower 4 bits to fix them at zero.

Table 12-18. Examples of 4-word Burst Addresses

1st Address	Lower 5 bits of 1st Address + 0x10 (no carry)	Final 2nd Address (After Masking Lower 4 Bits)
0x000	0x10	0x10
0x008	0x18	0x10
0x010	0x00	0x00
0x018	0x08	0x00
0x020	0x30	0x30
0x028	0x38	0x30
0x030	0x20	0x20
0x038	0x28	0x20

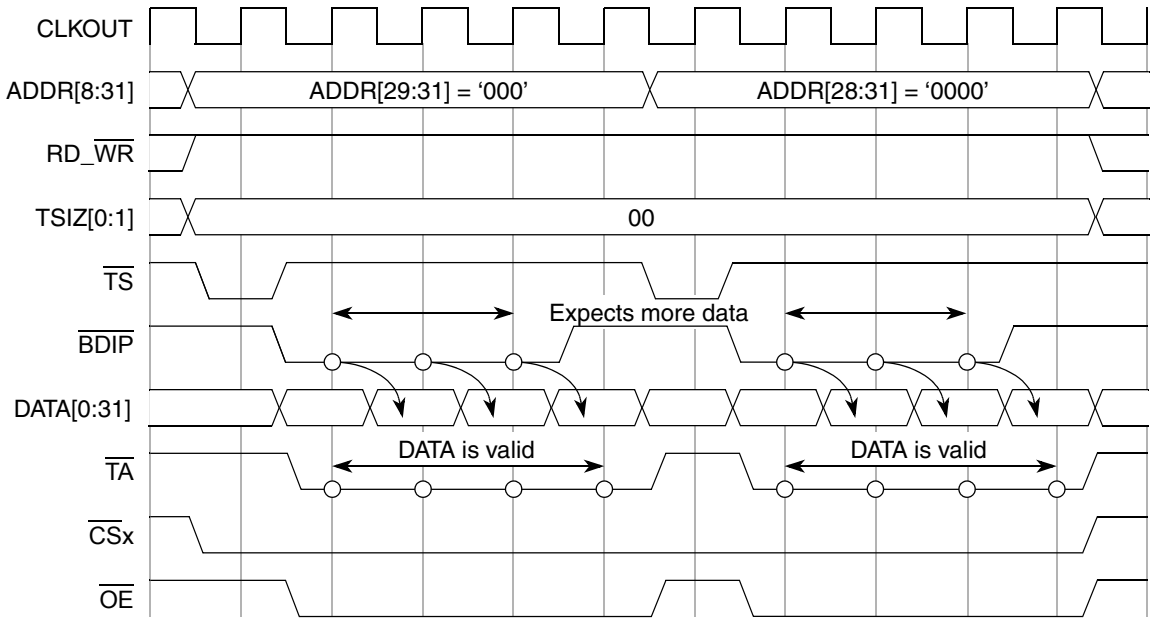


Figure 12-28. 32-Byte Read with Back-to-Back 16-Byte Bursts to 32-bit Port, Zero Wait States

12.4.2.7 Size, Alignment, and Packaging on Transfers

Table 12-19 shows the allowed sizes that an internal or external master can request from the EBI. The behavior of the EBI for request sizes not shown below is undefined. No error signal is asserted for these erroneous cases.

Table 12-19. Transaction Sizes Supported by EBI

No. Bytes (Internal Master)	No. Bytes (External Master)
1	1
2	2
4	4
8	
32	

The EBI supports only natural address alignment:

- Byte access can have any address.
- 16-bit access, address bit 31 must be 0.
- 32-bit access, address bits 30-31 must be 0.
- For burst accesses of any size, address bits 29-31 must be 0.

The EBI does not support misaligned accesses. If a misaligned access to the EBI is attempted by an internal master, the EBI errors the access on the internal bus and does not start the access (nor assert TEA) externally. This means the EBI never generates a misaligned external access, so a multi-master system with two eSys MCUs can never have a misaligned external access. In the erroneous case that an externally-initiated misaligned access does occur, the EBI errors the access (by asserting TEA externally) and does not initiate the access on the internal bus.

The bus requires that the portion of the data bus used for a transfer to/from a particular port size be fixed. A 32-bit port must reside on data bus bits 0-31, and a 16-bit port must reside on bits 0-15.

In the following figures and tables the following convention is adopted:

- The most significant byte of a 32-bit operand is OP0, and OP3 is the least significant byte.
- The two bytes of a 16-bit operand are OP0 (most significant) and OP1, or OP2 (most significant) and OP3, depending on the address of the access.
- The single byte of a byte-length operand is OP0, OP1, OP2, or OP3, depending on the address of the access.

The convention can be seen in Figure 12-29.

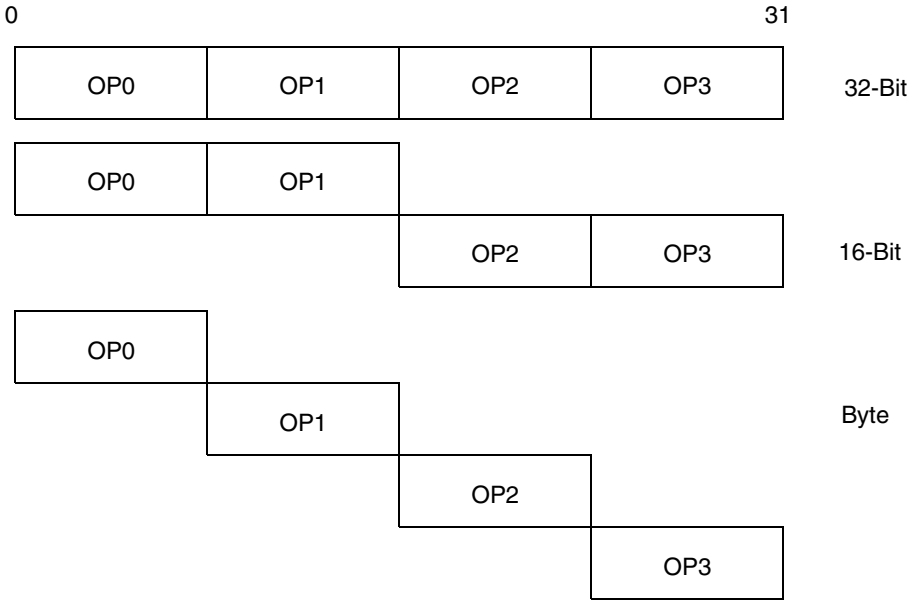


Figure 12-29. Internal Operand Representation

Figure 12-30 shows the device connections on the DATA[0:31] bus.

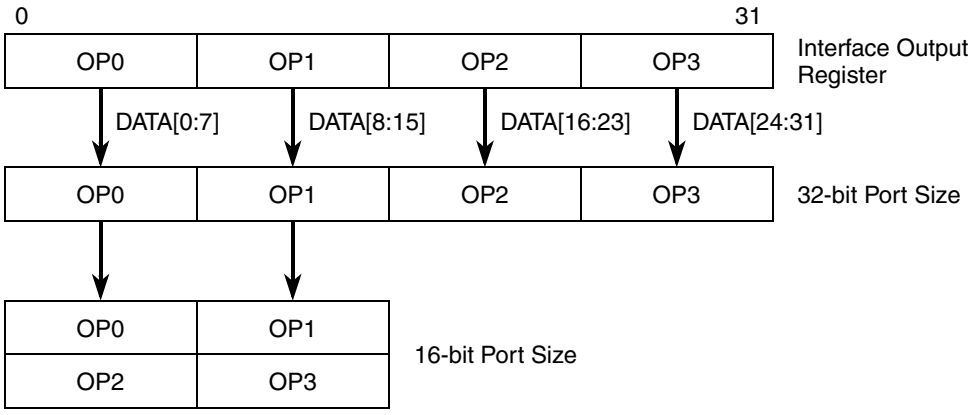


Figure 12-30. Interface to Different Port Size Devices

Table 12-20 lists the bytes required on the data bus for read cycles. The bytes indicated as '—' are not required during that read cycle.

Table 12-20. Data Bus Requirements for Read Cycles

Transfer Size	TSIZ[0:1]	Address		32-Bit Port Size				16-Bit Port Size[1]	
		A30	A31	D0:D7	D8:D15	D16:D23	D24:D31	D0:D7	D8:D15
Byte	01	0	0	OP0	—	—	—	OP0	—
	01	0	1	—	OP1	—	—	—	OP1
	01	1	0	—	—	OP2	—	OP2	—
	01	1	1	—	—	—	OP3	—	OP3
16-bit	10	0	0	OP0	OP1	—	—	OP0	OP1
	10	1	0	—	—	OP2	OP3	OP2	OP3
32-bit	00	0	0	OP0	OP1	OP2	OP3	OP0/OP2[2]	OP1/OP3

1 Also applies when DBM=1 for 16-bit data bus mode.

2 This case consists of two 16-bit external transactions, the first fetching OP0 and OP1, the second fetching OP2 and OP3.

Table 12-21 lists the patterns of the data transfer for write cycles when accesses are initiated by the MCU. The bytes indicated as '—' are not driven during that write cycle.

Table 12-21. Data Bus Contents for Write Cycles

Transfer Size	TSIZ[0:1]	Address		32-Bit Port Size				16-Bit Port Size[1]	
		A30	A31	D0:D7	D8:D15	D16:D23	D24:D31	D0:D7	D8:D15
Byte	01	0	0	OP0	—	—	—	OP0	—
	01	0	1	OP1	OP1	—	—	—	OP1
	01	1	0	OP2	—	OP2	—	OP2	—
	01	1	1	OP3	OP3	—	OP3	—	OP3
16-bit	10	0	0	OP0	OP1	—	—	OP0	OP1
	10	1	0	OP2	OP3	OP2	OP3	OP2	OP3
32-bit	00	0	0	OP0	OP1	OP2	OP3	OP0/OP2[2]	OP1/OP3

1 Also applies when DBM=1 for 16-bit data bus mode.

2 This case consists of two 16-bit external transactions, the first writing OP0 and OP1, the second writing OP2 and OP3.

12.4.2.8 Arbitration

The external bus design provides for a single bus master at any one time, either the MCU or an external device. One of the external devices on the bus has the capability of becoming bus master for the external bus. Bus arbitration may be handled either by an external central bus arbiter or by the internal on-chip arbiter. The arbitration configuration (external or internal) is set via the EARB bit in the EBI_MCR.

Each bus master must have bus request, bus grant, and bus busy signals. The signals are described in detail in Section 12.2.1, "Detailed Signal Descriptions." The device that needs the bus asserts the bus request ($\overline{BR}$) signal. The device then waits for the arbiter to assert the bus grant ($\overline{BG}$) signal. In addition, the new master must sample the bus busy ($\overline{BB}$) signal to ensure that no other master is driving the bus before it can

assert bus busy to assume ownership of the bus. The new master must sample bus busy negated for two cycles before asserting bus busy, to avoid any potential conflicts. Any time the arbiter has taken the bus grant away from the master, and the master wants to execute a new cycle, the master must re-arbitrate before a new cycle can begin. The EBI, however, whether the internal or external arbiter is used, guarantees data coherency for access to a small port size and for decomposed bursts. This means that the EBI does not release the bus before the completion of the transactions which are considered as atomic.

Figure 12-31 describes the basic protocol for bus arbitration.

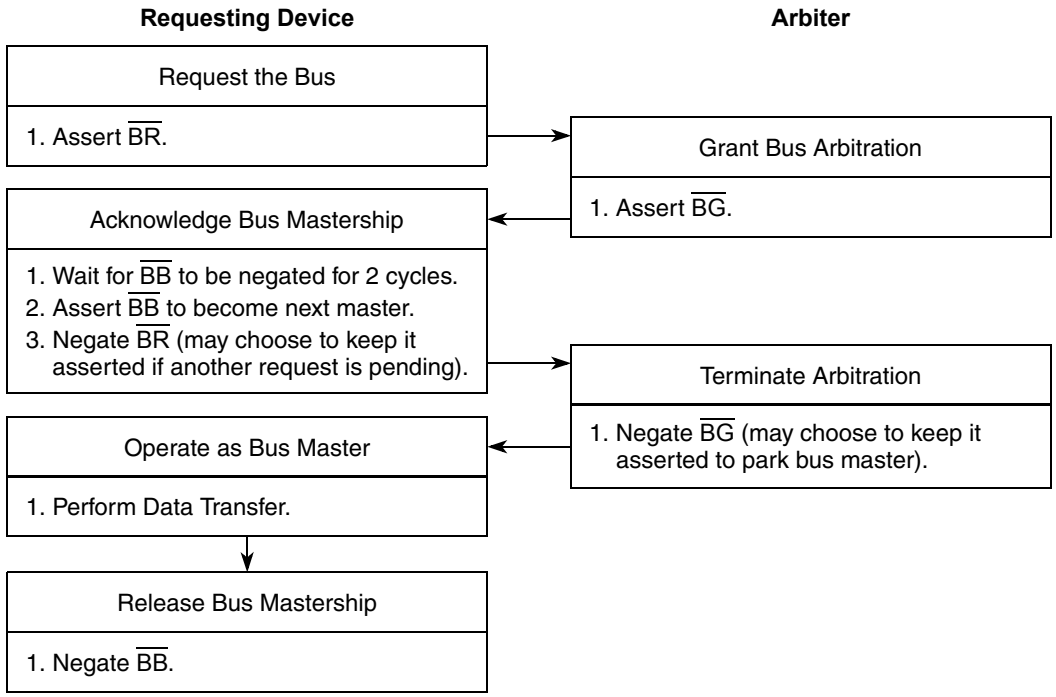


Figure 12-31. Bus Arbitration Flow Chart

12.4.2.8.1 External (or Central) Bus Arbiter

The external arbiter can be either another MCU in a two master system, or a separate central arbiter device. When an MCU is configured to use external arbitration, that MCU asserts BR when it needs ownership of the external bus, and it waits for BG to be asserted from the external arbiter. For timing reasons, a latched (1 cycle delayed) version of BG is used by the EBI in external arbitration mode. This is not a requirement of the protocol. After BG assertion is received and BB is sampled negated for two cycles, the MCU asserts BB and initiates the transaction. An MCU operating under external arbitration may run back-to-back accesses without rearbitrating as long as it is still receiving BG asserted. If BG is negated during a transaction, the MCU must rearbitrate for the bus before the next transaction. The determination of priority between masters is determined entirely by the external arbiter in this mode.

Figure 12-32 shows example timing for the case of two masters connected to a central arbiter. In this case, the BR0 and BR1 signals shown are inputs to the arbiter from the BR pin of each master. The BG0 and BG1 signals are outputs from the arbiter that connect to the BG pin of each master.

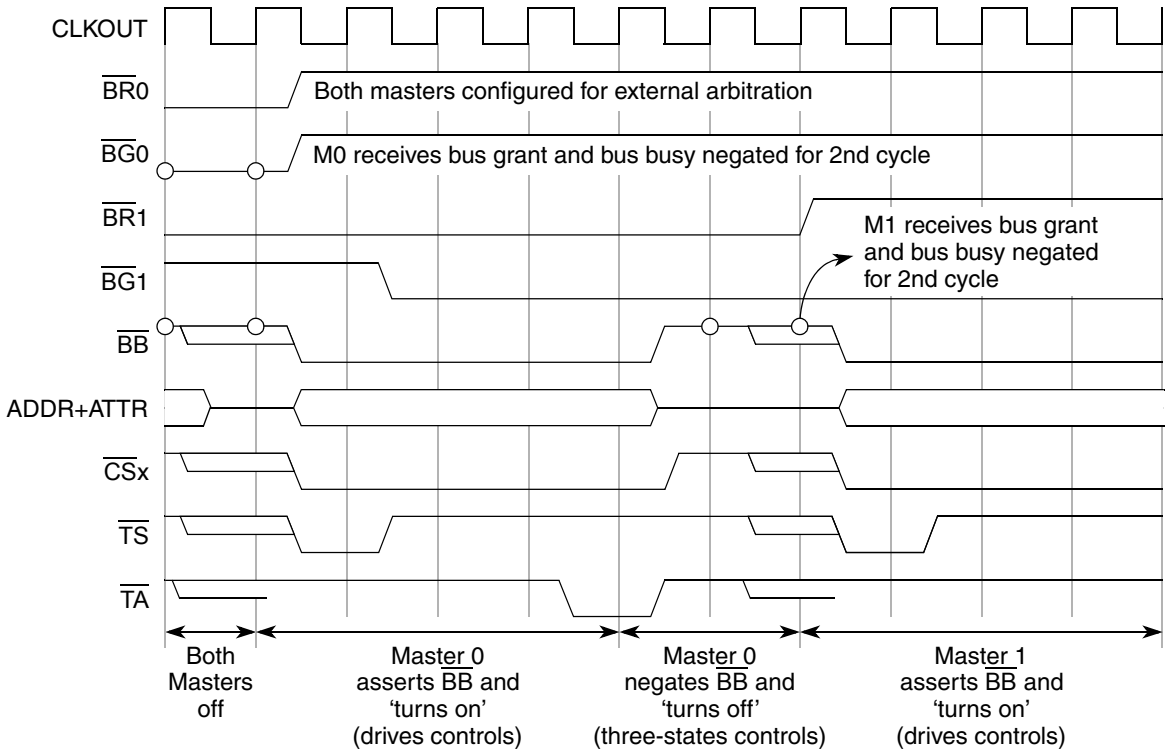


Figure 12-32. Central Arbitration Timing Diagram

12.4.2.8.2 Internal Bus Arbiter

When an MCU is configured to use the internal bus arbiter, that MCU is parked on the bus. The parking feature allows the MCU to skip the bus request phase, and if BB is negated, assert BB, and initiate the transaction without waiting for bus grant from the arbiter. The priority between internal and external masters over the external bus is determined by the EARP field of the EBI_MCR. See Table 12-8 for the EARP field description.

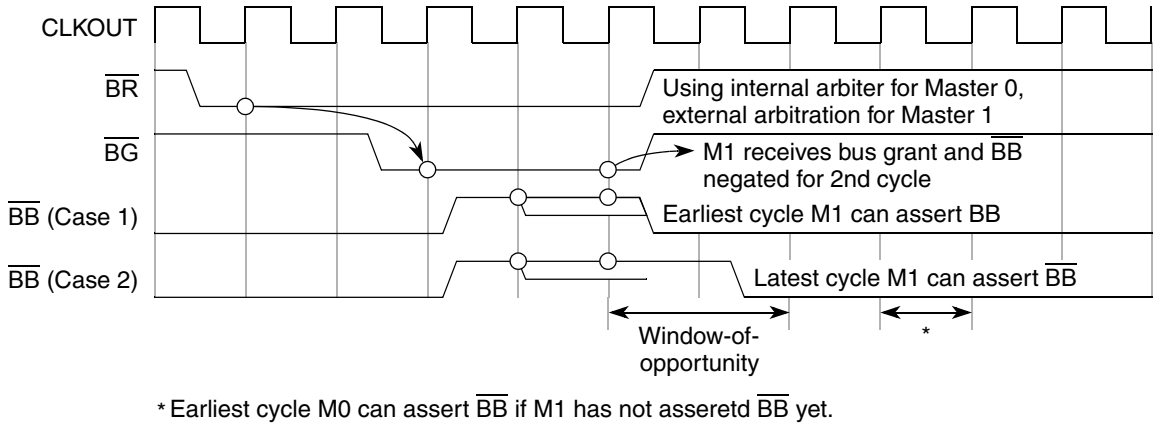
By default, internal and external masters are treated with equal priority, with each having to relinquish the bus after the current transaction if another master is requesting it. If internal and external requests for the bus occur in the same cycle, the internal arbiter grants the bus to the master who least recently used the bus. If no other master is requesting the bus, the bus continues to be granted to the current master, and the current master may start another access without re-arbitrating for the bus.

If the priority field is configured for unequal priority between internal and external masters, then whenever requests are pending from both masters, the one with higher priority is always granted the bus. However, in all cases, a transaction in progress (or that has already been granted, for example MCU bus wait and external bus wait states) is allowed to complete, even when a request from a higher priority master is pending.

There is a minimum of one cycle between the positive edge CLKOUT that a BR assertion is sampled by the EBI and the positive edge CLKOUT where BG is driven out asserted by the EBI. This is to avoid timing problems that would otherwise limit the frequency of operation in external master mode.

The external master is given 2 cycles to start its access after a posed CLKOUT in which bus grant was given to it by the internal arbiter (BG asserted, BB negated for 2 cycles). This means when BG is negated (to take away bus grant from the external master), the EBI does not start an access of its own for 3 cycles (1 extra cycle in order to detect external BB assertion). If the external master jumps on the bus (by asserting

BB) during the 2-cycle window, the EBI detects the BB assertion and delays starting its access until the external master access has completed (BB negated for 2 cycles). Figure 12-33 shows this 2-cycle window of opportunity.



* Earliest cycle M0 can assert BB if M1 has not asseretd BB yet.

Figure 12-33. Internal Arbitration, 2-Cycle Window-of-Opportunity

Figure 12-34 shows example timing for the case of one master using internal arbitration (master 0), while another master is configured for external arbitration (master 1). In this case, the BR signals of each master are connected together, since only master 1 drives BR. The BG signals of each master are also connected together, since only master 0 drives BG. See Figure 12-37 for an example of these connections.

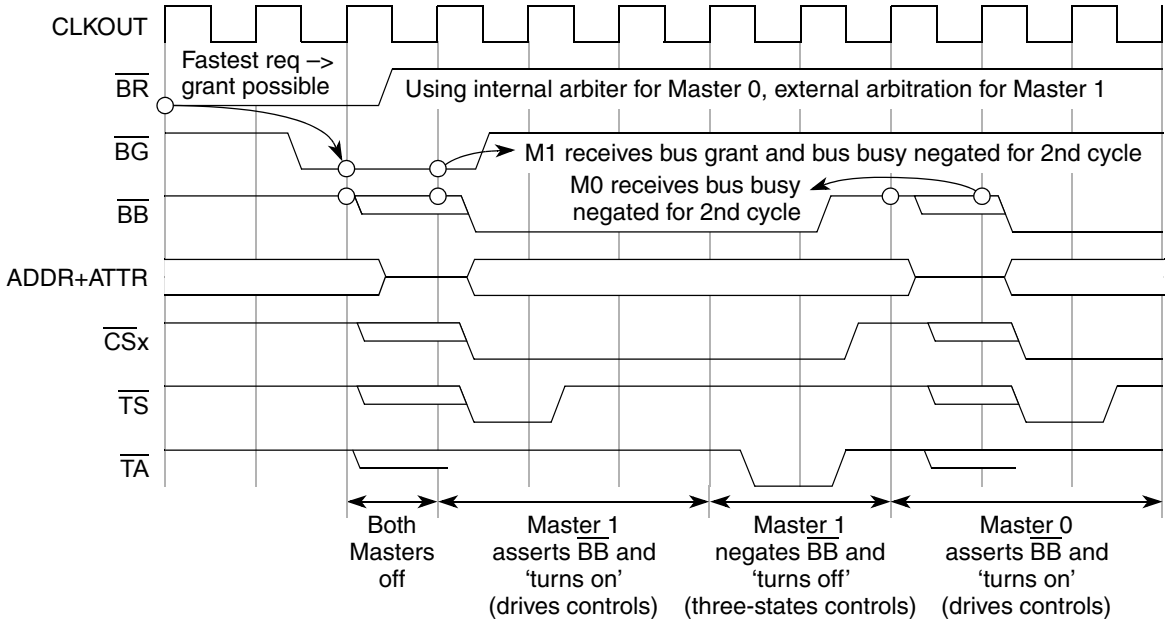


Figure 12-34. Internal/External Arbitration Timing Diagram (EARP = 1)

Table 12-22 shows a description of the states defined for the internal arbiter protocol.

Table 12-22. Internal Arbiter State Descriptions

State	Description	Outputs
MCU Owner Idle	MCU owns bus, but is not currently running a transaction	$\overline{BG}$ = 1, $\overline{BB}$ = hiZ
Ext. Owner	Ext. master owns bus, may or may not be running a transaction	$\overline{BG}$ = 0, $\overline{BB}$ = hiZ
MCU Bus Wait	MCU owns bus for next transaction, waiting for ext. owner to negate $\overline{BB}$ from current transaction in progress	$\overline{BG}$ = 1, $\overline{BB}$ = hiZ
MCU Owner Busy	MCU owns bus, and is currently running a transaction	$\overline{BG}$ = 1, $\overline{BB}$ = 0/1
Ext. Bus Wait	Ext. master owns bus for next transaction, waiting for MCU to negate $\overline{BB}$ from current transaction in progress	$\overline{BG}$ = 0, $\overline{BB}$ = 0/1

Table 12-23 shows the truth table for the internal arbiter protocol.

Table 12-23. Internal Arbiter Truth Table

State	Outputs		Inputs						Next State
	BG	$\overline{BB}$[1]	BR[2] (previous)	$\overline{BB}$[3] (previous)	MCU Internal Request Pending (IRP)[4] (previous)	External has Higher Priority (EHP)[5]	MCU Ext. Transaction in Progress (or starting next cycle) (ETP)[6]	Recent $\overline{BG}$ (RBG)[7]	
MCU Owner Idle	1	hiZ	1	X	0	0	0	X[8]	MCU Owner Idle
MCU Owner Idle	1	hiZ	X	X	0	1	0	X[9]	Ext. Owner
MCU Owner Idle	1	hiZ	0	X	0	X	0	X	Ext. Owner
MCU Owner Idle	1	hiZ	0	X	X	1	0	X	Ext. Owner
MCU Owner Idle	1	hiZ	X	X	1	0	X	X	MCU Owner Busy
MCU Owner Idle	1	hiZ	1	X	1	X	X	X	MCU Owner Busy
MCU Owner Idle	1	hiZ	X	X	X	X	1	X	MCU Owner Busy
Ext. Owner	0	hiZ	X	X	0	X	X[10]	X[11]	Ext. Owner
Ext. Owner	0	hiZ	0	X	X	1	X	X	Ext. Owner
Ext. Owner	0	hiZ	X	X	1	0	X	X	MCU Bus Wait
Ext. Owner	0	hiZ	1	X	1	X	X	X	MCU Bus Wait
MCU Bus Wait	1	hiZ	X	0	X[12]	X	X[10]	X	MCU Bus Wait
MCU Bus Wait	1	hiZ	X	X	X	X	X	1	MCU Bus Wait
MCU Bus Wait	1	hiZ	X	1	X	X	X	0	MCU Owner Busy

Table 12-23. Internal Arbiter Truth Table (continued)

State	Outputs		Inputs						Next State
	BG	$\overline{BB}$[1]	BR[2] (previous)	$\overline{BB}$[3] (previous)	MCU Internal Request Pending (IRP)[4] (previous)	External has Higher Priority (EHP)[5]	MCU Ext. Transaction in Progress (or starting next cycle) (ETP)[6]	Recent $\overline{BG}$ (RBG)[7]	
MCU Owner Busy	1	0/1[13]	1	X	X	X	1	X[8]	MCU Owner Busy
MCU Owner Busy	1	0/1	1	X	1	X	X	X	MCU Owner Busy
MCU Owner Busy	1	0/1	X	X	1	0	X	X	MCU Owner Busy
MCU Owner Busy	1	0/1	0	X	X	1	1	X	Ext. Bus Wait
MCU Owner Busy	1	0/1	0	X	0	X	1	X	Ext. Bus Wait
MCU Owner Busy	1	0/1	0	X	X	1	0	X	Ext. Owner
MCU Owner Busy	1	0/1	0	X	0	X	0	X	Ext. Owner
MCU Owner Busy	1	0/1	1	X	0	X	0	X	MCU Owner Idle
Ext. Bus Wait	0	0/1[13]	X[14]	X	X	X	1	X[8]	Ext. Bus Wait
Ext. Bus Wait	0	0/1	X	X	X	X	0	X	Ext. Owner

1 The Output column for $\overline{BB}$ shows the value EBI drives on $\overline{BB}$ for each state.

2 The Input column for $\overline{BR}$ shows the value driven on $\overline{BR}$ the previous cycle from an external source. The state machine uses the previous clock value to avoid potential speed paths with trying to calculate bus grant based on a late-arriving external $\overline{BR}$ signal.

3 The Input column for $\overline{BB}$ shows the value driven on $\overline{BB}$ the previous cycle from an external source. The state machine uses the previous clock value to ensure adequate switching time between masters driving the same signal and to avoid potential speed paths.

4 This represents an internal EBI signal that indicates whether an internal request for use of the external bus is pending. Once a transaction for a pending request has been started on the external bus, this internal signal is cleared. The state machine uses the previous clock value to avoid potential speed paths with trying to calculate bus grant based on a late-arriving internal request signal.

5 This represents an internal EBI signal that indicates whether the internal MCU (0) or external master (1) currently has higher priority.

6 This represents an internal EBI signal that indicates whether an EBI-mastered transaction on the bus is in progress this cycle or is going to start the next cycle (and thus has already been committed internally).

7 This represents an internal EBI signal that indicates whether the bus was granted to an external master ($\overline{BG}$ = 0, previous $\overline{BB}$ = 1) during the previous 3 cycles.

8 RGB is always low in this state, thus it is ignored in the transition logic.

9 RGB is always low in this state, thus it is ignored in the transition logic.

10 The ETP signal is never asserted in states where it is shown as an ‘X’ for all transitions.

11 RGB is always high in this state, thus it is ignored in the transition logic.

[12] IRP is ignored (treated as 1) in the MCU_WAIT state because the EBI does not optimally support an internal master cancelling its bus request. If IRP is negated in this state, the EBI still grants the internal master the bus as if IRP was still asserted, and a few cycles may be wasted before the external master may be able to grab the bus again (depending on $\overline{BR}$, $\overline{BB}$, etc., according to normal transition logic).

[13] The default $\overline{BB}$ output is 0 for this state. However, anytime the EBI transitions from a state where $\overline{BB}$ = 0 to a state where $\overline{BB}$ = hiZ, there is one external cycle (in this state) where the EBI drives $\overline{BB}$ = 1 to actively negate the pin before letting go to hiZ. In the case where a second granted internal request (IRP = 1, ETP=1) is ready to start just before the transition to the hiZ state would otherwise have occurred (during the $\overline{BB}$ = 1active negate cycle), then $\overline{BB}$ is driven back to 0 to start the next access without ever leaving this state or going to hiZ.

[14] $\overline{BR}$ is ignored (treated as 0) in the EXT_WAIT state because the EBI does not optimally support an external master cancelling its bus request. If $\overline{BR}$ is negated in this state, the EBI still grants the external master the bus as if $\overline{BR}$ was still asserted, and a few cycles may be wasted while the external master ‘window-of-opportunity’ is satisfied before the internal master may be able to grab the bus again (depending on $\overline{BR}$, $\overline{BB}$, etc., according to normal transition logic).

Figure 12-35 shows the internal finite state machine that implements the arbiter protocol.

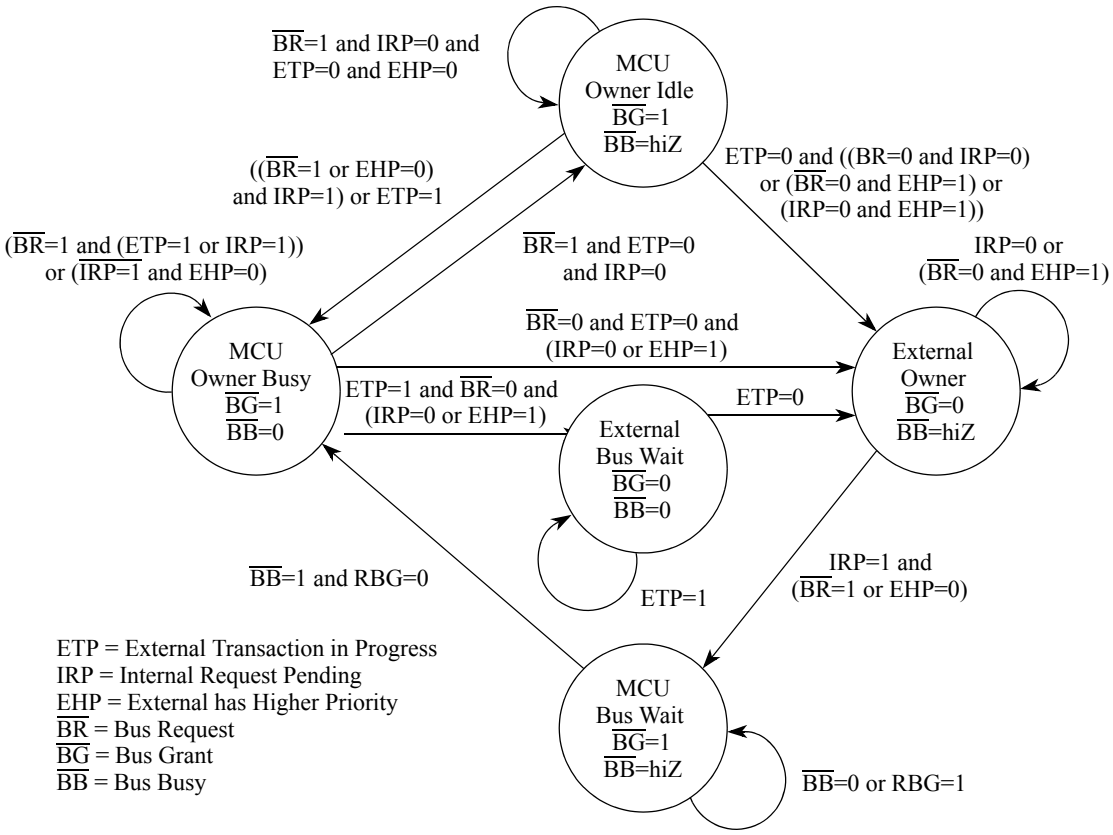


Figure 12-35. Internal Bus Arbitration State Machine

12.4.2.9 Termination Signals Protocol

The termination signals protocol was defined in order to avoid electrical contention on lines that can be driven by various sources. In order to do that, a slave must not drive signals associated with the data transfer until the address phase is completed and it recognizes the address as its own. The slave must disconnect from signals immediately after it acknowledges the cycle and not later than the termination of the next address phase cycle.

For EBI-mastered non-chip select accesses, the EBI requires assertion of TA from an external device to signal that the bus cycle is complete. The EBI uses a latched version of TA (1 cycle delayed) for these accesses to help make timing at high frequencies. This results in the EBI driving the address and control signals 1 cycle longer than required, as seen in Figure 12-36. However, the DATA does not need to be held 1 cycle longer by the slave, because the EBI latches DATA every cycle during non-chip select accesses. During these accesses, the EBI does not drive the TA signal, leaving it up to an external device (or weak internal pull-up) to drive TA.

For EBI-mastered chip select accesses, the EBI drives TA the entire cycle, asserting according to internal wait state counters to terminate the cycle. During idle periods on the external bus, the EBI drives TA negated as long as it is granted the bus; when it no longer owns the bus it lets go of TA. When an external master does a transaction to internal address space, the EBI only drives TA for the cycle it asserts TA to return data and for 1 cycle afterwards to ensure fast negation.

If no device responds by asserting $\overline{TA}$ within the programmed timeout period (BMT in EBI_BMCR) after the EBI initiates the bus cycle, the internal bus monitor (if enabled) asserts $\overline{TEA}$ to terminate the cycle. An external device may also drive $\overline{TEA}$ when it detects an error on an external transaction. $\overline{TEA}$ assertion causes the cycle to terminate and the processor to enter exception processing for the error condition. To properly control termination of a bus cycle for a bus error with external circuitry, $\overline{TEA}$ must be asserted at the same time or before (external) $\overline{TA}$ is asserted. $\overline{TEA}$ must be negated before the second rising edge after it was sampled asserted in order to avoid the detection of an error for the following bus cycle initiated. $\overline{TEA}$ is only driven by the EBI during the cycle where the EBI is asserting $\overline{TEA}$ and the cycle immediately following this assertion (for fast negation). During all other cycles, the EBI relies on a weak internal pull-up to hold $\overline{TEA}$ negated. This allows an external device to assert $\overline{TEA}$ when it needs to indicate an error. External devices must follow the same protocol as the EBI, only driving $\overline{TEA}$ during the assertion cycle and 1 cycle afterwards for negation.

NOTE

In the case where an external master asserts $\overline{TEA}$ to timeout a transaction to an internal address on this MCU, the EBI has no way to terminate the transfer internally. Therefore, any subsequent $\overline{TS}$ assertions by the external master are ignored by the EBI until the original transfer has completed internally and the EBI has returned to an idle state. The expectation is that the internal slaves will always respond with either valid data or an error indication within a reasonable period of time to avoid hanging the system.

When $\overline{TEA}$ is asserted from an external source, the EBI uses a latched version of $\overline{TEA}$ (1 cycle delayed) to help make timing at high frequencies. This means that for any accesses where the EBI drives $\overline{TA}$ (chip select accesses and external master accesses to EBI), a $\overline{TEA}$ assertion that occurs 1 cycle before or during the last $\overline{TA}$ of the access could be ignored by the EBI, since it will have completed the access internally before it detects the latched $\overline{TEA}$ assertion. This means that non-burst chip select accesses with no wait states (SCY = 0) cannot be reliably terminated by external $\overline{TEA}$. If external error termination is required for such a device, the EBI must be configured for SCY ≥ 1.

NOTE

For the cases discussed above where $\overline{TEA}$ could be ignored, this is not guaranteed. For some small access cases (which always use chip select and internally-driven $\overline{TA}$), a $\overline{TEA}$ that occurs 1 cycle before or during the $\overline{TA}$ cycle or for SCY = 0 may in fact lead to terminating the cycle with error. However, proper error termination is not guaranteed for these cases, so $\overline{TEA}$ must always be asserted at least 2 cycles before an internally-driven $\overline{TA}$ cycle for proper error termination.

External $\overline{TEA}$ assertion that occurs during the same cycle that $\overline{TS}$ is asserted by the EBI is always treated as an error (terminating the access) regardless of SCY.

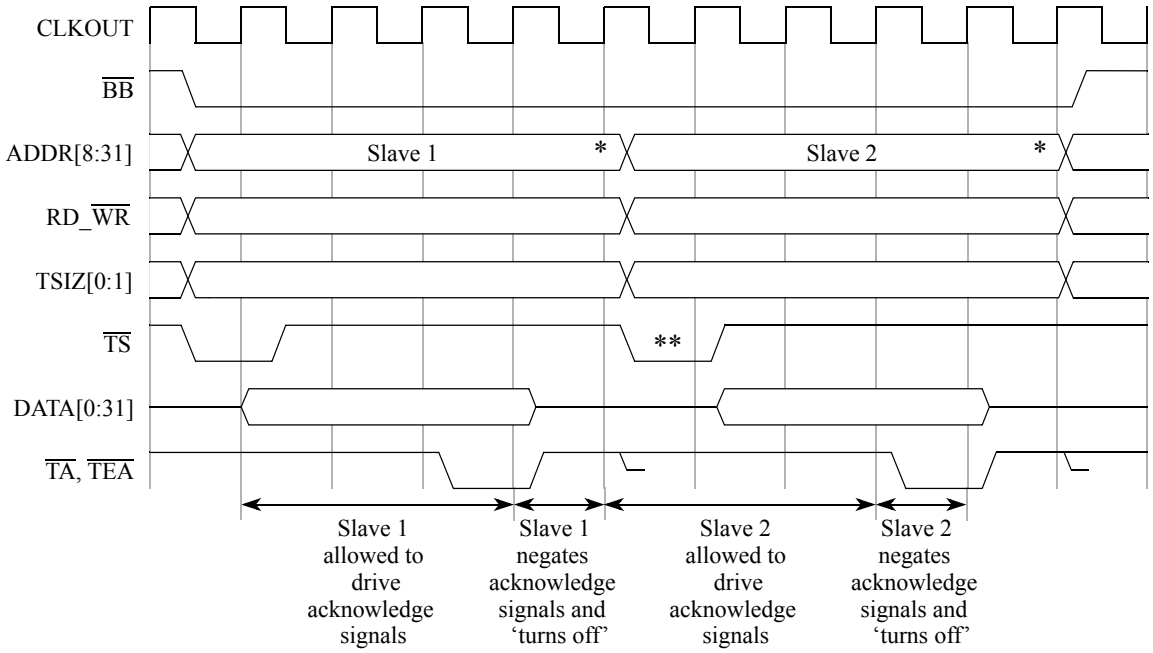
Table 12-24 summarizes how the EBI recognizes the termination signals provided from an external device.

Table 12-24. Termination Signals Protocol

TEA[1]	TA[1]	Action
Negated	Negated	No Termination
Asserted	X	Transfer Error Termination
Negated	Asserted	Normal Transfer Termination

[1] Latched version (1 cycle delayed) used for externally driven $\overline{TEA}$ and $\overline{TA}$.

Figure 12-36 shows an example of the termination signals protocol for back-to-back reads to two different slave devices who properly take turns driving the termination signals. This assumes a system using slave devices that drive termination signals.



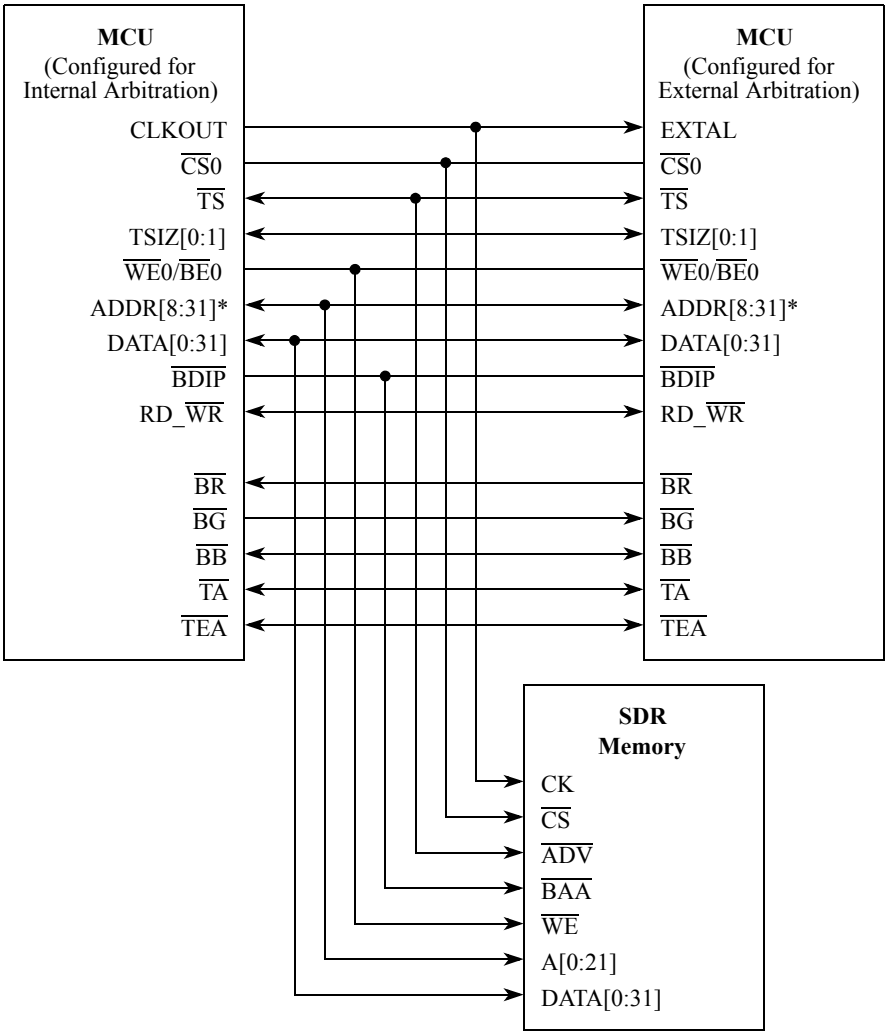
*The EBI drives address and control signals an extra cycle because it uses a latched version of $\overline{TA}$ (1 cycle delayed) to terminate the cycle. An external master is not required to do this.

**This is the earliest that the EBI can start another transfer, in the case of continuing a set of small accesses. For all other cases, an extra cycle is needed before the EBI can start another $\overline{TS}$.

Figure 12-36. Termination Signals Protocol Timing Diagram

12.4.2.10 Bus Operation in External Master Mode

In the MPC5554, external master mode enables an external master to access the internal address space of the MCU. Figure 12-37 shows how to connect an MCU to an external master (2nd MCU) and a shared SDR memory to operate in external master mode. Only master/slave systems are supported in the MPC5553; master to master systems are not supported. Refer to 12.5.5.



* Only ADDR[8:29] are connected to the 32-bit SDR memory.

Figure 12-37. MCU Connected to External Master and SDR Memory

When the external master requires external bus accesses, it takes ownership on the external bus, and the direction of most of the bus signals is inverted, relative to its direction when the MCU owns the bus.

To operate two masters in external master mode, one must be configured for internal arbitration and the other must be configured for external arbitration. Connecting three or more masters together in the same system is not supported by this EBI.

Most of the bidirectional signals shown in Figure 12-37 are only driven by the EBI when the EBI owns the external bus. The only exceptions are the TA and TEA signals (described in Section 12.4.2.9, “Termination Signals Protocol”) and the DATA bus, which are driven by the EBI for external master reads to internal address space. As long as the external master device follows the same protocol for driving signals as this EBI, there is no need to use the open drain mode of the pads configuration module for any EBI pins.

The PowerPC storage reservation protocol is not supported by the EBI. Coherency between multiple masters must be maintained via software techniques, such as event passing.

The EBI does not provide memory controller services to an external master that accesses shared external memories. Each master must properly configure its own memory controller and drive its own chip selects when sharing a memory between two masters.

The EBI does not support burst accesses from an external master; only single accesses of 8, 16, or 32 bits can be performed.[1]

12.4.2.10.1 Address Decoding for External Master Accesses

The EBI allows external masters to access internal address space when the EBI is configured for external master mode. The external address is compared for any external master access, in order to determine if EBI operation is required. Because only 24 address bits are available on the external bus, special decoding logic is required to allow an external master to access on-chip locations whose upper 8 address bits are non-zero. This is done by using the upper 4 external address bits (ADDR[8:11]) as a code to determine whether the access is on-chip and if so, which internal slave it is targeted for.

Below is the address compare sequence:

- External master access to another device — If ADDR[8] = 0, then the access is assumed to be to another device and is ignored by the EBI.
- External master access to valid internal slave — If ADDR[8] = 1, then ADDR[9:11] are checked versus a list of 3-bit codes to determine which internal slave to forward the access to. The upper 8 internal address bits are set appropriately by the EBI according to this 3-bit code, and internal address bits [8:11] are set appropriately to match the internal slave selected.
- External master access to invalid internal slave — If the 3-bit code does not match a valid internal slave, then the EBI responds with a bus error.

Table 12-25 shows the possible 3-bit codes that are associated with various slaves in the MCU, as well as the resulting upper 12 address bits required to appropriately match up with the memory map of each internal slave.

Table 12-25. EBI Internal Slave Address Decoding

Internal Slave	External ADDR[8:11][1]	Internal Addr [0:7][2]	Internal Addr [8:11][3]
- (off-chip)	0b0xxx	—	—
Internal Flash	0b10xx	0b0000_0000	0b00, ADDR[10:11]
	0b1100	0b0100_0000	0b0000
Reserved	0b1101	0b0110_0000	0b0000
Bridge A Peripherals	0b1110	0b1100_0011	0b1111
Bridge B Peripherals	0b1111	0b1111_1111	0b1111

1 Value on upper 4 bits of 24-bit external address bus ADDR[8:31]. ADDR[8] determines whether the access is on or off chip.

2 Value on upper 8 bits of 32-bit internal address bus.

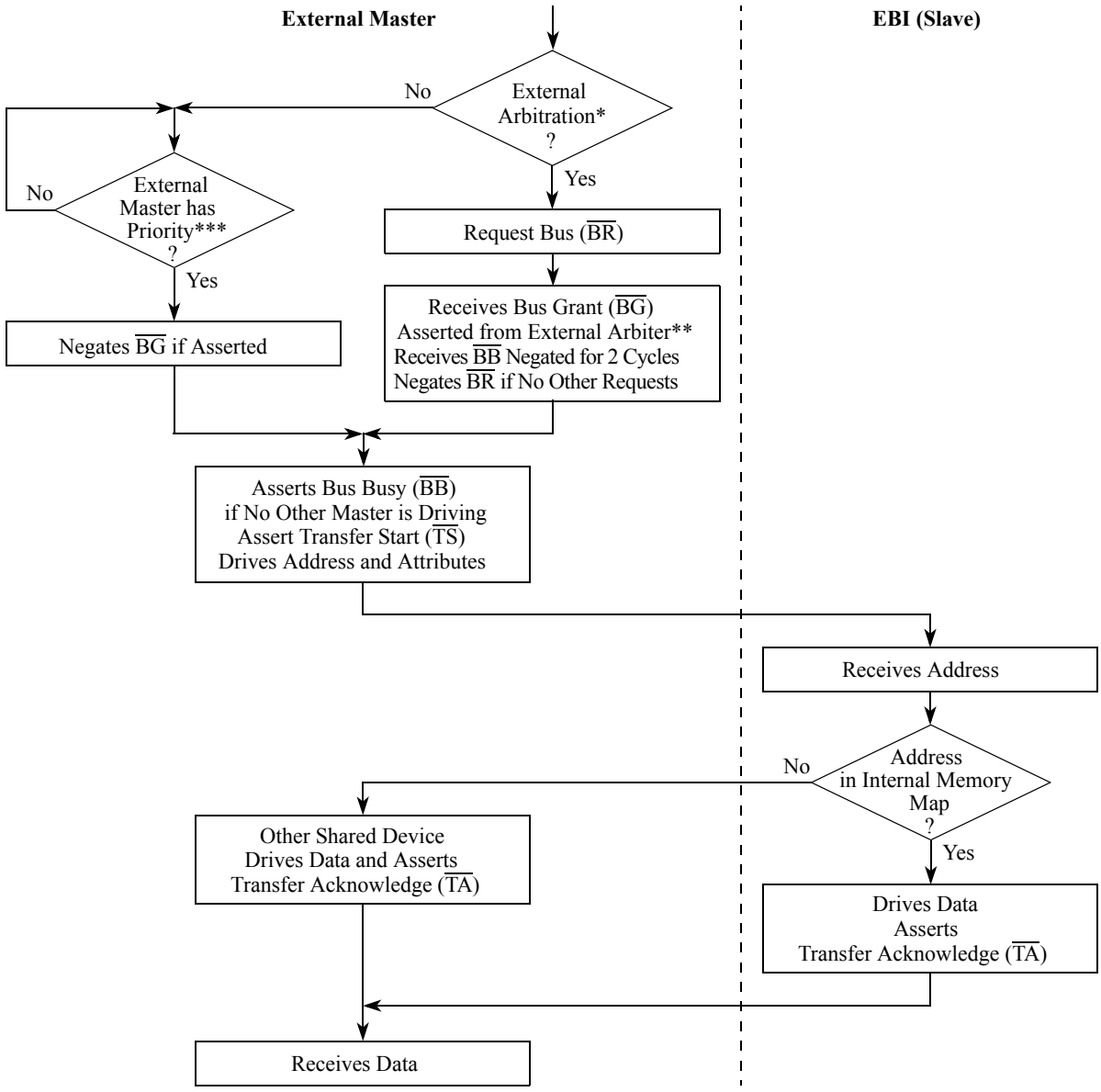
3 Value on bits 8:11 of 32-bit internal address bus.

1. Except for the special case of a 32-bit non-chip select access in 16-bit data bus mode. See Section 12.4.2.11, “Non-Chip-Select Burst in 16-bit Data Bus Mode”.

12.4.2.10.2 Bus Transfers Initiated by an External Master

The external master gets ownership of the bus (see Section 12.4.2.8, "Arbitration") and asserts $\overline{TS}$ in order to initiate an external master access. The access is directed to the internal bus only if the input address matches to the internal address space. The access is terminated with either $\overline{TA}$ or $\overline{TEA}$. If the access was successfully completed, the MCU asserts $\overline{TA}$, and the external master can proceed with another external master access, or relinquish the bus. If an address or data error was detected internally, the MCU asserts $\overline{TEA}$ for one clock.

Figure 12-38 and Figure 12-39 illustrate the basic flow of read and write external master accesses.

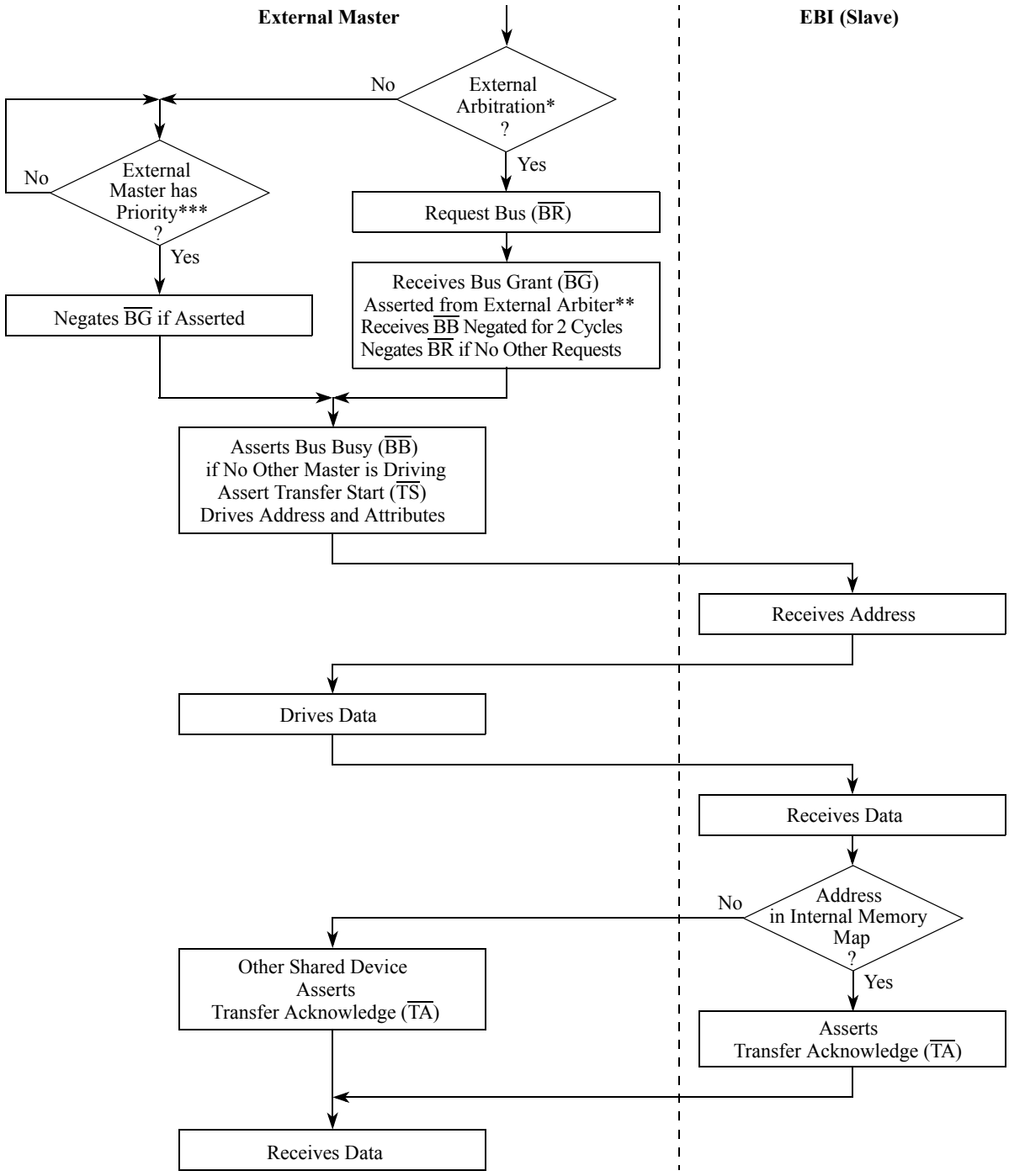


*This refers to whether the external master device is configured for external or internal arbitration.

**External arbiter is the EBI unless a central arbiter device is used.

***Determined by the internal arbiter of the external master device.

Figure 12-38. Basic Flow Diagram of an External Master Read Access



*This refers to whether the external master device is configured for external or internal arbitration.

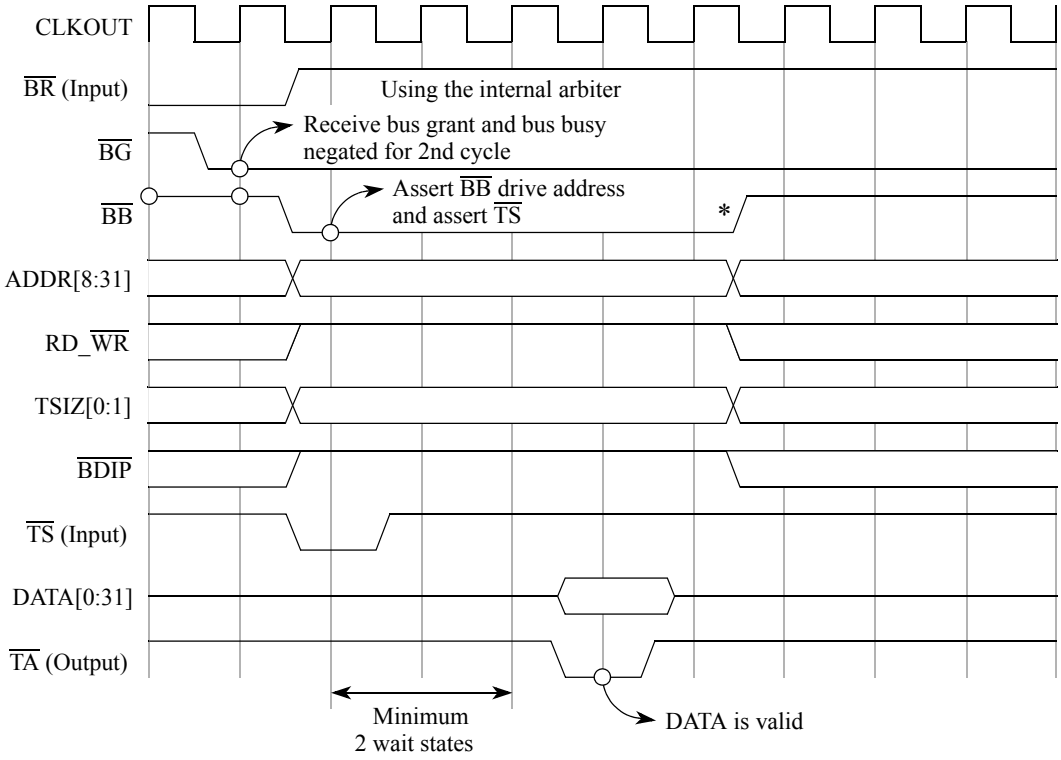
**External arbiter is the EBI unless a central arbiter device is used.

***Determined by the internal arbiter of the external master device.

Figure 12-39. Basic Flow Diagram of an External Master Write Cycle

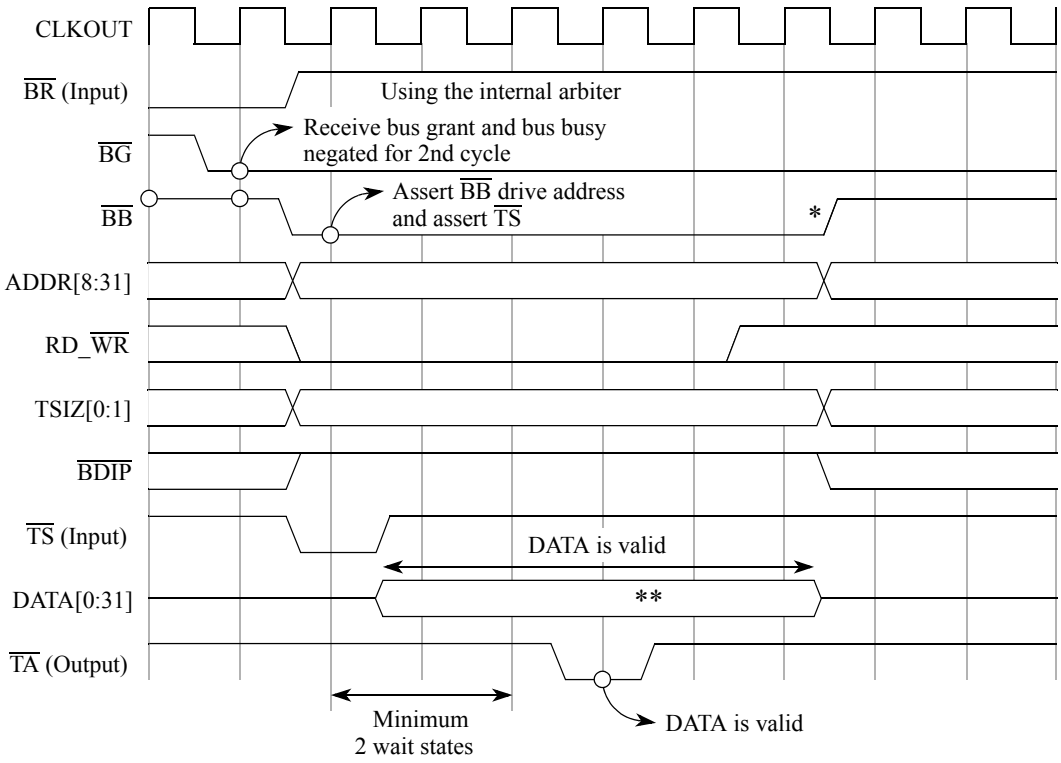
Figure 12-40 and Figure 12-41 describe read and write cycles from an external master accessing internal space in the MCU. Note that the minimal latency for an external master access is 3 clock cycles. The actual

latency of an external to internal cycle varies depending on which internal module is being accessed and how much internal bus traffic is going on at the time of the access.



*If the external master is another MCU with this EBI, then BB and other control pins are turned off as shown due to use of latched TA internally. This extra cycle is not required by the slave EBI.

Figure 12-40. External Master Read from MCU



*If the external master is another MCU with this EBI, then BB and other control pins are turned off as shown due to use of latched TA internally. This extra cycle is not required by the slave EBI.

**If the external master is another MCU with this EBI, then DATA remains valid as shown due to use of latched TA internally. These extra data valid cycles (past TA) are not required by the slave EBI.

Figure 12-41. External Master Write to MCU

12.4.2.10.3 Bus Transfers Initiated by the EBI in External Master Mode

The flow and timing for EBI-mastered transactions in external master mode is identical to that described in earlier sections for single master mode, with the exception that the EBI must now arbitrate for the bus before each transaction. The flow and timing diagrams below show the arbitration sequence added to Figure 12-9 and Figure 12-10 for the basic single beat read case. The remaining cases (writes, bursts, etc.) can be obtained by adding the arbitration sequence to the flow and timing diagrams shown for single master mode in earlier sections. See Section 12.4.2.4, "Single Beat Transfer," and Section 12.4.2.5, "Burst Transfer."

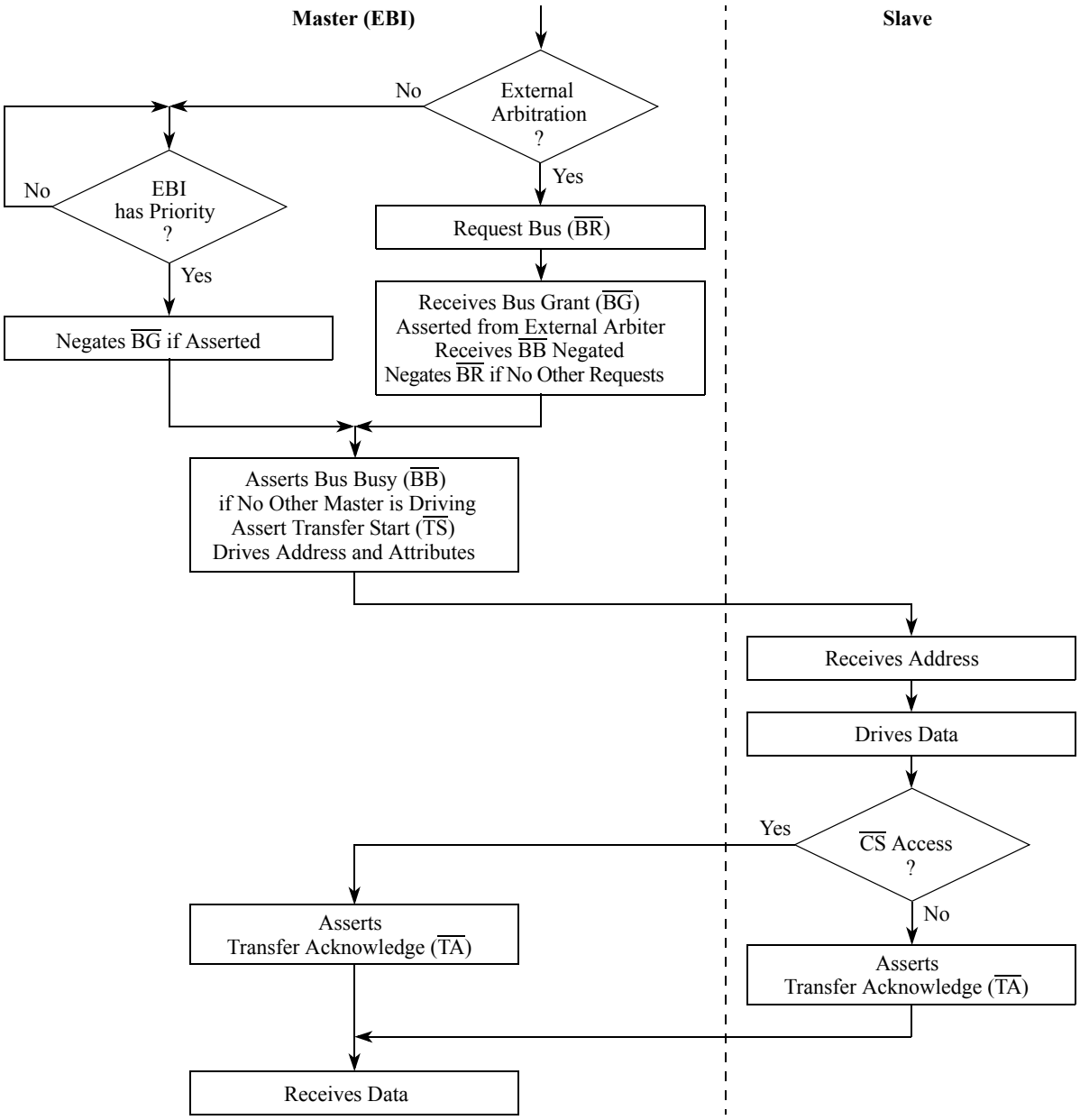


Figure 12-42. Basic Flow Diagram of an EBI Read Access in External Master Mode

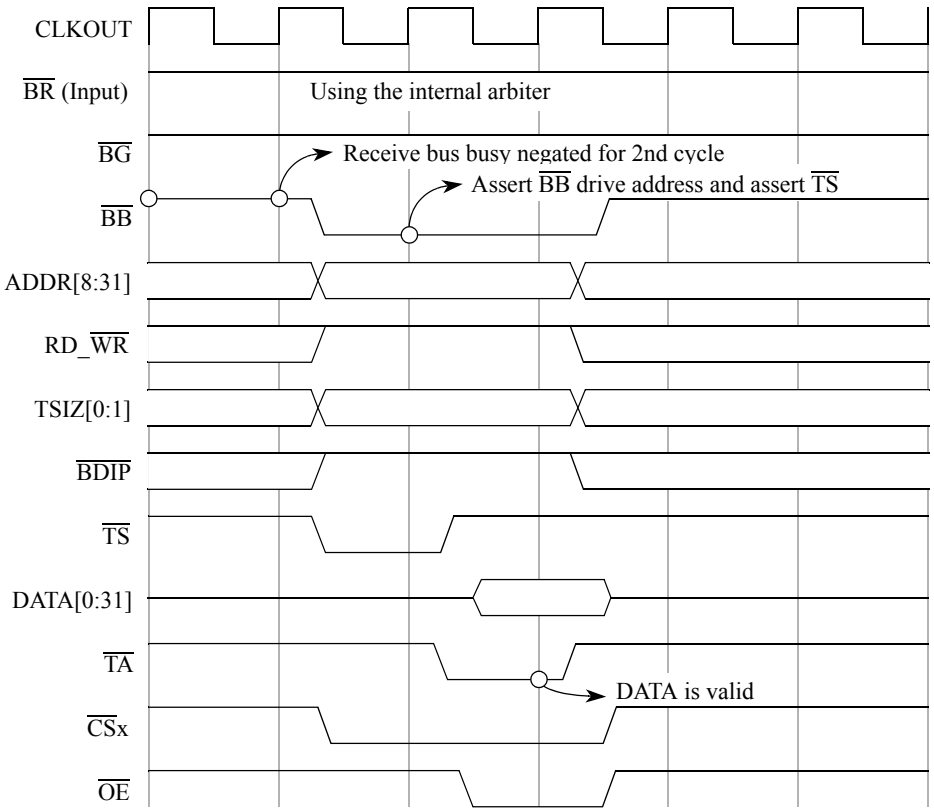


Figure 12-43. Single Beat CS Read Cycle in External Master Mode, Zero Wait States

12.4.2.10.4 Back-to-Back Transfers in External Master Mode

The following timing diagrams show examples of back-to-back accesses in external master mode. In these examples, the reads and writes shown are to a shared external memory, and the EBI is assumed to be configured for internal arbitration while the external master is configured for external arbitration.

Figure 12-44 shows an external master read followed by an MCU read to the same chip select bank. Figure 12-45 shows an MCU read followed by an external master read to a different chip select bank. Figure 12-46 shows an external master read followed by an external master write to a different chip select bank. This case assumes the MCU has no higher priority internal request pending and is able to park the external master on the bus.

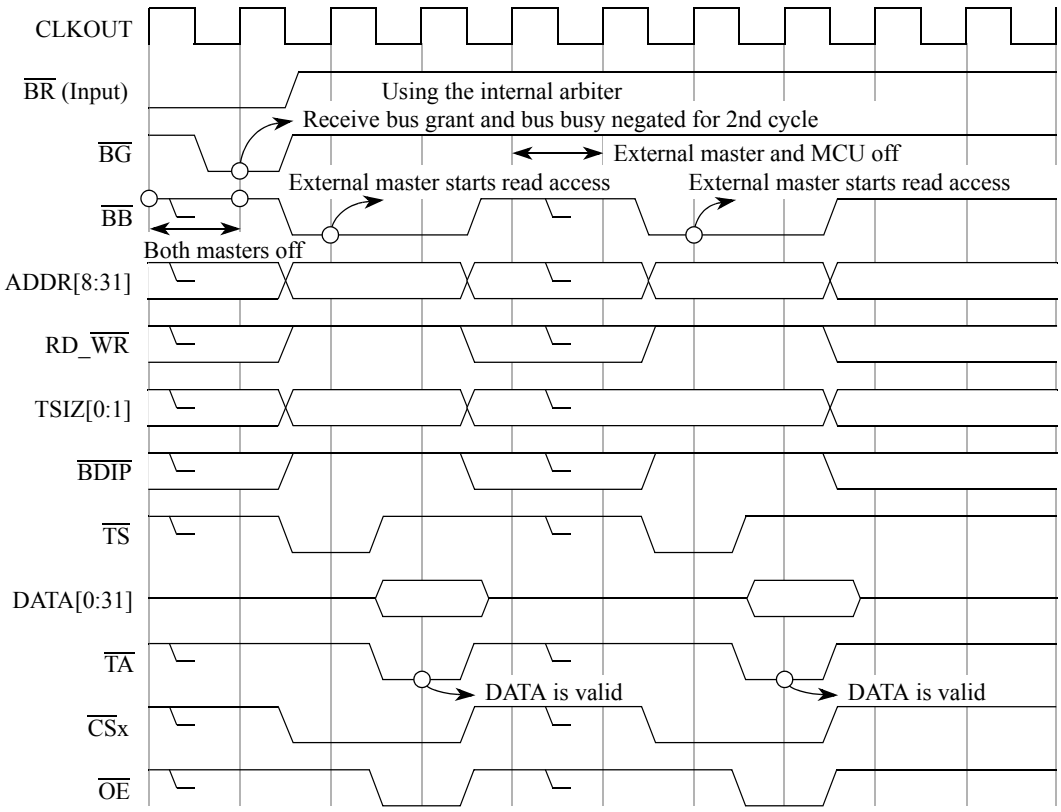


Figure 12-44. External Master Read followed by MCU Read to Same $\overline{CS}$ Bank

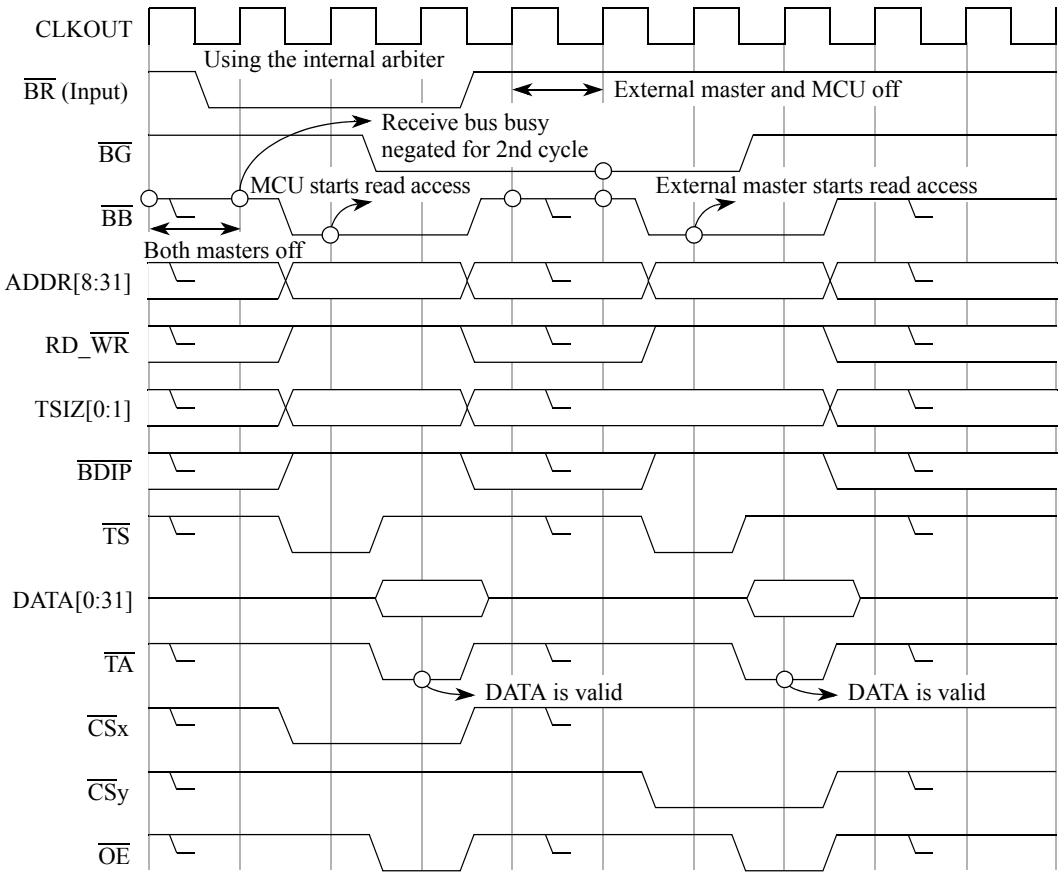


Figure 12-45. MCU Read followed by External Master Read to Different $\overline{CS}$ Bank

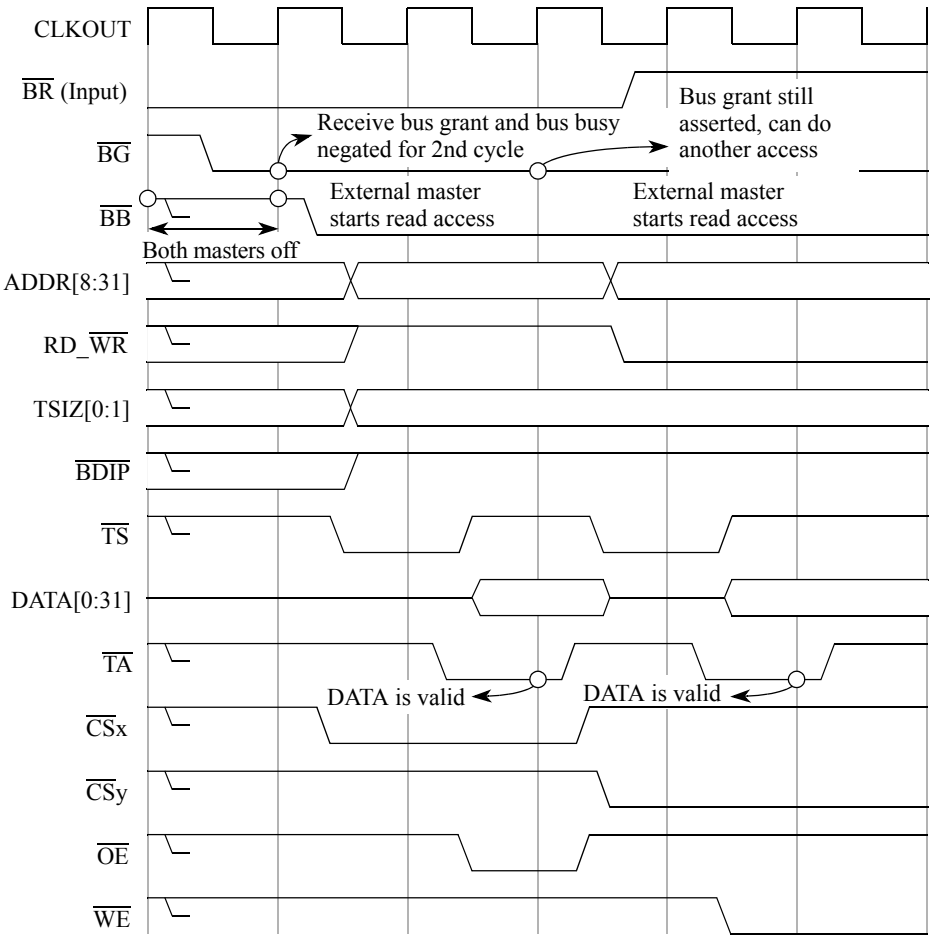


Figure 12-46. External Master Read followed by External Master Write to Different $\overline{CS}$ Bank

12.4.2.11 Non-Chip-Select Burst in 16-bit Data Bus Mode

The timing diagrams in this section apply only to the special case of a non-chip select 32-bit access in 16-bit data bus mode. They specify the behavior for both the EBI-master and EBI-slave, as the external master is expected to be another MCU with this EBI.

For this case, a special 2-beat burst protocol is used for reads and writes, so that the EBI-slave can internally generate one 32-bit read or write access (thus 32-bit coherent), as opposed to two separate 16-bit accesses.

Figure 12-47 shows a 32-bit read from an external master in 16-bit data bus mode.

Figure 12-48 shows a 32-bit write from an external master in 16-bit data bus mode.

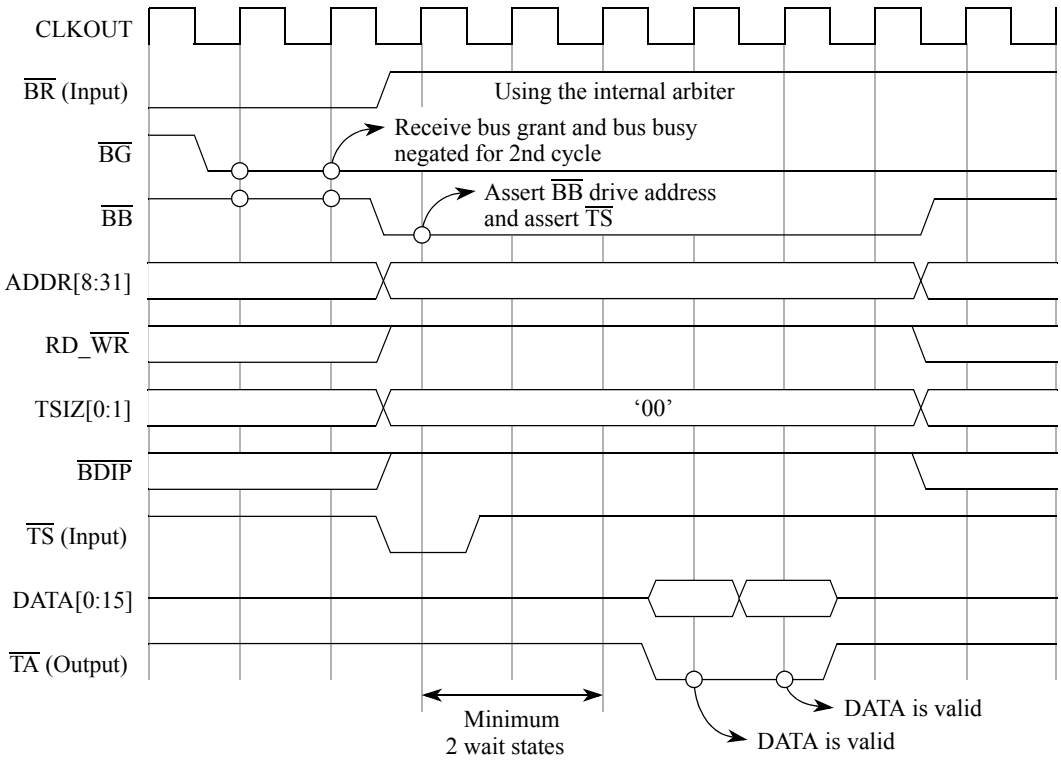


Figure 12-47. External Master 32-bit Read from MCU with DBM=1

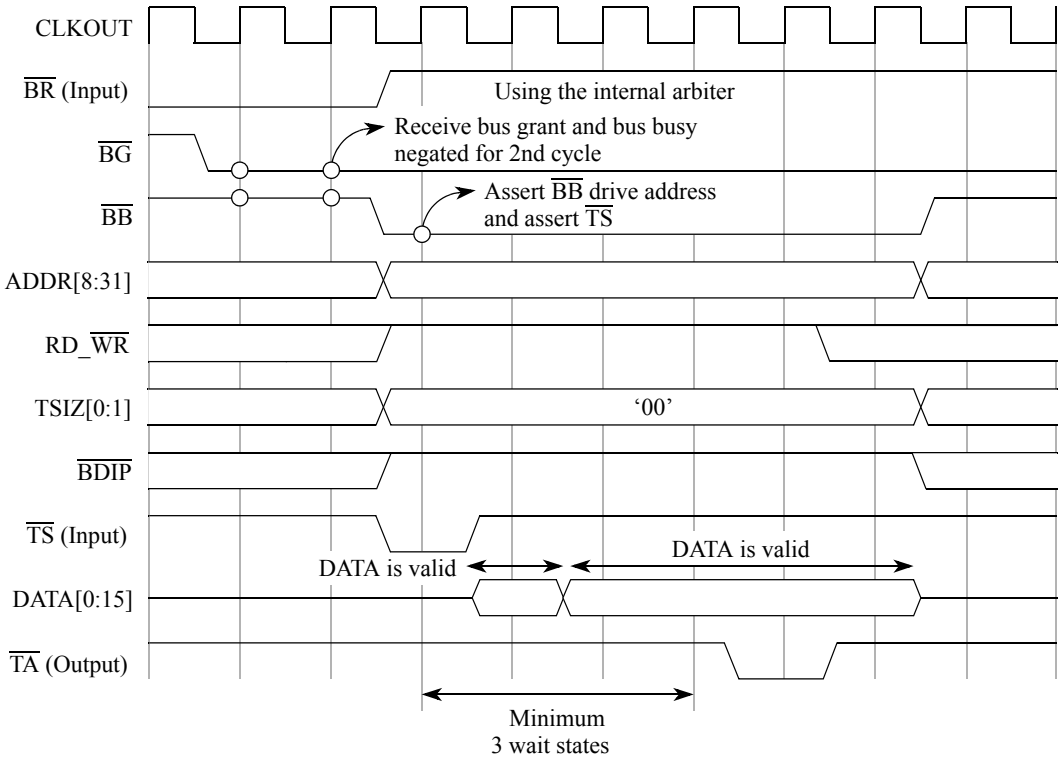


Figure 12-48. External Master 32-bit Write to MCU with DBM=1

12.4.2.12 Calibration Bus Operation — MPC5553 Only

The MPC5553 EBI has a second external bus, intended for calibration use. This bus consists of a second set of the same signals present on the primary external bus, except that arbitration, (and optionally other signals also) are excluded. Both busses are supported by the EBI by using the calibration chip selects to steer accesses to the calibration bus instead of to the primary external bus.

Because the calibration bus has no arbitration signals, the arbitration on the primary bus controls accesses on the calibration bus as well, and no external master accesses can be performed on the calibration bus. Accesses cannot be performed in parallel on both external busses. However, back-to-back accesses can switch from one bus to the other, as determined by the type of chip select each address matches.

The timing diagrams and protocol for the calibration bus is identical to the primary bus, except that some signals are missing on the calibration bus.

There is an inherent dead cycle between a calibration chip select access and a non-calibration access (chip select or non-chip select), just like the one between accesses to two different non-calibration chip selects (described in Section 12.4.2.4.3, “Back-to-Back Accesses”).

Figure 12-49 shows an example of a non-calibration chip select read access followed by a calibration chip select read access. Note that this figure is identical to Figure 12-18, except the $\overline{CS}y$ is replaced by $\overline{CAL\_CS}y$. Timing for other cases on the calibration bus can similarly be derived from other figures in this document (by replacing $\overline{CS}$ with $\overline{CAL\_CS}$).

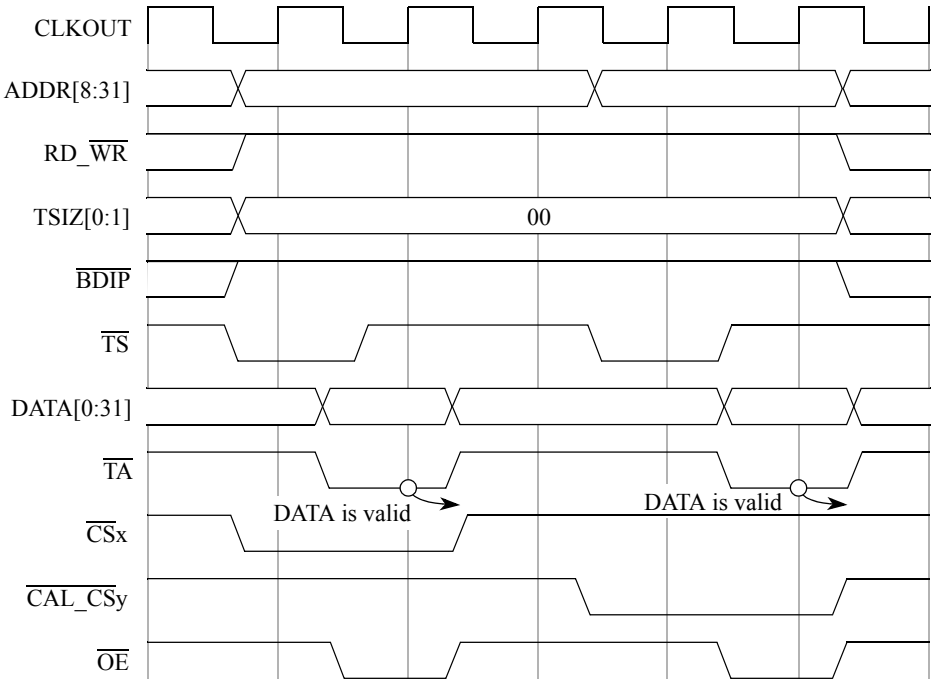


Figure 12-49. Back-to-Back 32-bit Reads to $\overline{CS}$, $\overline{CAL\_CS}$ Banks

12.5 Initialization/Application Information

12.5.1 Booting from External Memory

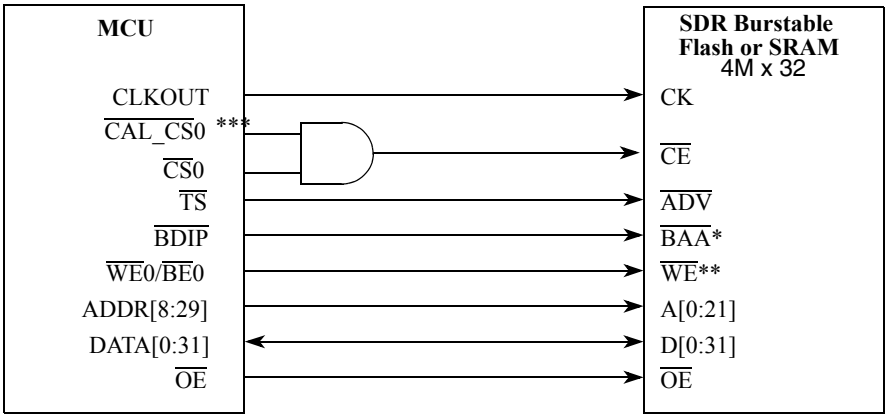
The EBI block does not support booting directly to external memory (i.e. fetching the first instruction after reset externally). The MPC5553 and MPC5554 use an internal boot assist module, which executes after each reset. The BAM code performs basic configuration of the EBI block, allowing for external boot if desired. Refer to Chapter 16, "Boot Assist Module (BAM)" for detail information about the boot modes supported by the MPC5554.

If code in external memory needs to write EBI registers, this must be done in a way that avoids modifying EBI registers while external accesses are being performed, such as the following method:

- Copy the code that is doing the register writes (plus a return branch) to internal SRAM
- Branch to internal SRAM to run this code, ending with a branch back to external flash

12.5.2 Running with SDR (Single Data Rate) Burst Memories

This includes FLASH and external SRAM memories with a compatible burst interface. $\overline{\text{BDIP}}$ is required only for some SDR memories. Figure 12-47 shows a block diagram of an MCU connected to a 32-bit SDR burst memory.



*May or may not be connected, depending on the memory used.
**Flash memories typically use one $\overline{\text{WE}}$ signal as shown, RAMs use 2 or 4 (16-bit or 32-bit).
*** MPC5553 Only

Figure 12-50. MCU Connected to SDR Burst Memory

Refer to Figure 12-22 for an example of the timing of a typical burst read operation to an SDR burst memory. Refer to Figure 12-14 for an example of the timing of a typical single write operation to SDR memory.

12.5.3 Running with Asynchronous Memories

The EBI also supports asychronous memories. In this case, the CLKOUT, $\overline{\text{TS}}$, and $\overline{\text{BDIP}}$ pins are not used by the memory and bursting is not supported. However, the EBI still drives these outputs, and always drives and latches all signals at positive edge CLKOUT (i.e., there is no asynchronous mode for the EBI). The data timing is controlled by setting the SCY bits in the appropriate option register to the proper number

of wait states to work with the access time of the asynchronous memory, just as done for a synchronous memory.

12.5.3.1 Example Wait State Calculation

This example applies to any chip select memory, synchronous or asynchronous.

As an example, say we have a memory with 50ns access time, and we are running the external bus @66MHz (CLKOUT period: 15.2ns). Assume the input data spec for the MCU is 4ns.

number of wait states = (access time) / (CLKOUT period) + (0 or 1) (depending on setup time)

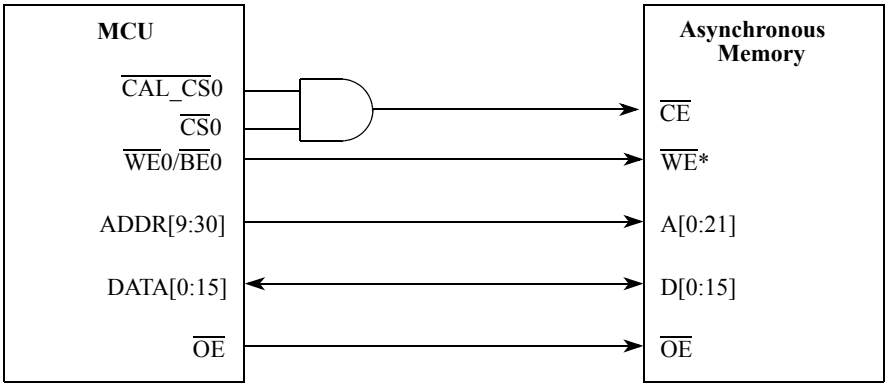
50/15.2 = 3 with 4.4ns remaining (so we need at least three wait states, now check setup time)

15.2-4.4=10.8ns (this is the achieved input data setup time)

Because actual input setup (10.8ns) is greater than the input setup spec (4.0ns), three wait states is sufficient. If the actual input setup was less than 4.0ns, we would have to use four wait states instead.

12.5.3.2 Timing and Connections for Asynchronous Memories

The connections to an asynchronous memory are the same as for a synchronous memory, except that the CLKOUT, TS, and BDIP signals are not used. Figure 12-51 shows a block diagram of an MCU connected to an asynchronous memory.



*Flash memories typically use one WE signal as shown, RAMs use 2 or 4 (16-bit or 32-bit).

Figure 12-51. MCU Connected to Asynchronous Memory

Figure 12-52 shows a timing diagram of a read operation to a 16-bit asynchronous memory using three wait states. Figure 12-53 shows a timing diagram of a write operation to a 16-bit asynchronous memory using three wait states.

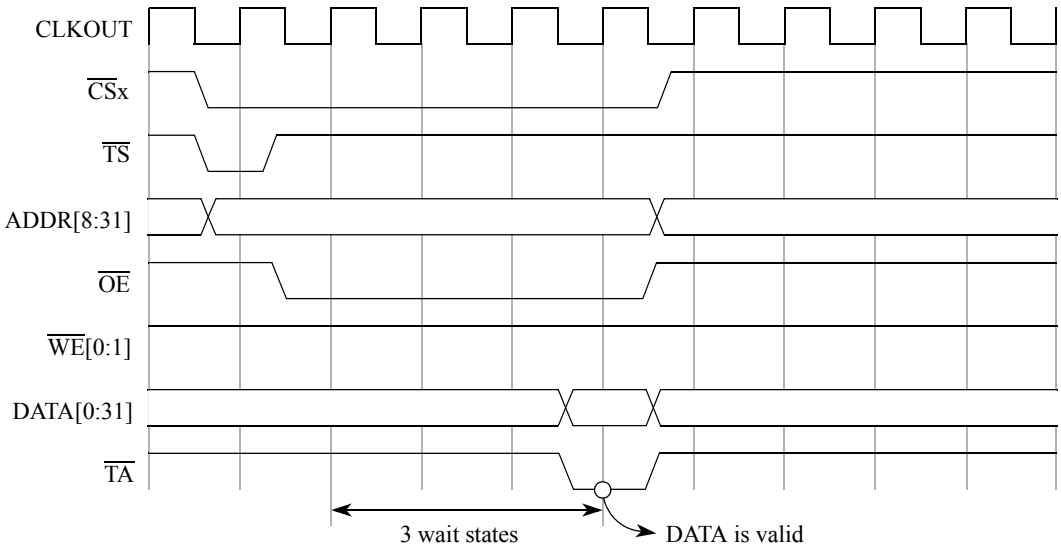


Figure 12-52. Read Operation to Asynchronous Memory, Three Initial Wait States

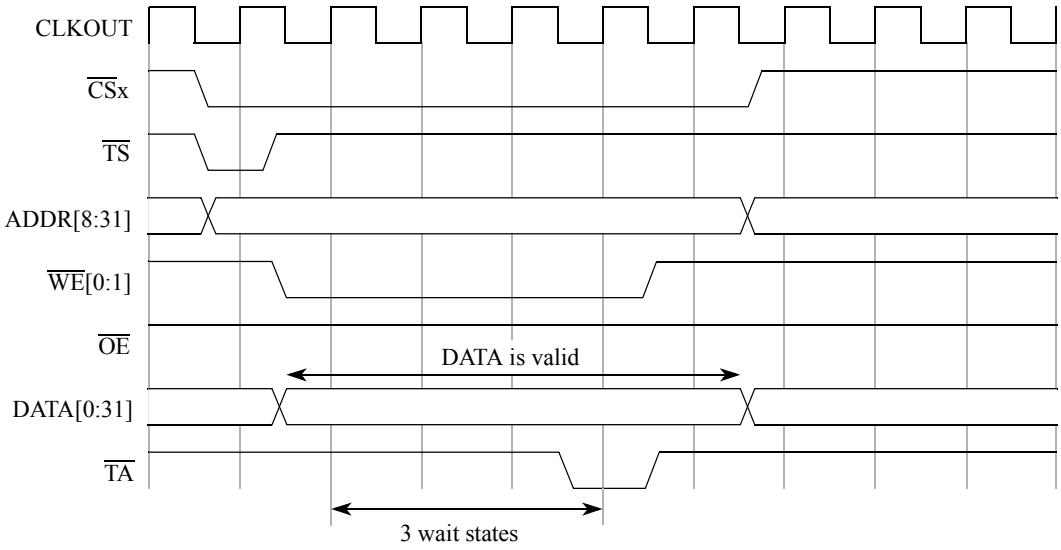
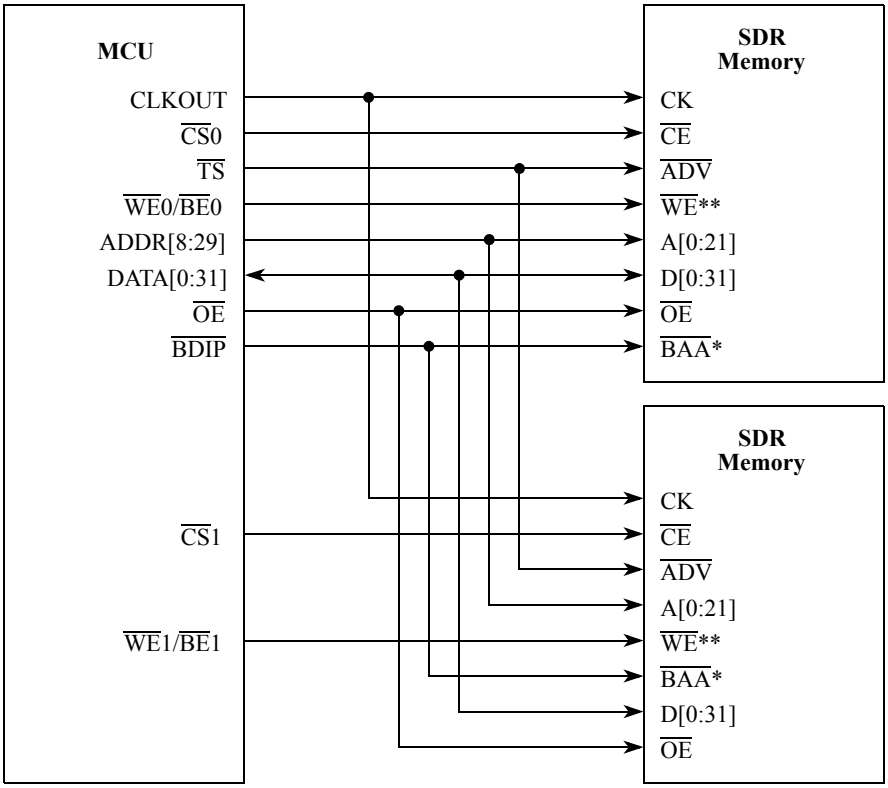


Figure 12-53. Write Operation to Asynchronous Memory, Three Initial Wait States

12.5.4 Connecting an MCU to Multiple Memories

The MCU can be connected to more than one memory at a time.

Figure 12-54 shows an example of how two memories could be connected to one MCU.



*May or may not be connected, depending on the memory used.
**Flash memories typically use one WE signal as shown, RAMs use 2 or 4 (16-bit or 32-bit).

Figure 12-54. MCU Connected to Multiple Memories

12.5.5 Dual-MCU Operation with Reduced Pinout MCUs

Some MCUs with this EBI may not have all the pins described in this document pinned out for a particular package. Some of the most common pins to be removed are DATA[16:31], arbitration pins (BB, BG, BR), and TSIZ[0:1]. This section describes how to configure dual-MCU systems for each of these scenarios. More than one section may apply if the applicable pins are not present on one or both MCUs.

12.5.5.1 Connecting 16-bit MCU to 32-bit MCU (Master/Master or Master/Slave)

This scenario is straightforward. Simply connect DATA[0:15] between both MCUs, and configure both for 16-bit data bus mode operation (DBM=1 in EBI_MCR). Note that 32-bit external memories are not supported in this scenario.

12.5.5.2 Arbitration with No Arb Pins (Master/Slave only)

Without arbitration pins, a dual-master system is impossible, because these is no way for the two masters to take turns driving the external bus without conflicts. However, a master/slave system is possible, as described below.

To implement a master/slave system with an MCU that has no arbitration pins ($\overline{BB}$, $\overline{BG}$, $\overline{BR}$), the user must configure the master MCU for internal arbitration (EARB=0 in EBI_MCR) and the slave MCU for external arbitration (EARB=1). Internally on an MCU with no arbitration pins, the $\overline{BR}$, $\overline{BG}$, and $\overline{BB}$ signals to the EBI will be tied negated. This means that the slave MCU will never receive bus grant asserted, so it will never attempt to start an access on the external bus. The master MCU will never receive bus request or bus busy asserted, so it will maintain ownership of the bus without any arbitration delays. In the erroneous case that the slave MCU executes internal code that attempts to access external address space, that access will never get external and will eventually time-out in the slave MCU.

12.5.5.3 Transfer Size with No TSIZ Pins (Master/Master or Master/Slave)

Since there are no TSIZ pins to communicate transfer size from master MCU to slave MCU, the internal SIZE field of the EBI_MCR must be used on the slave MCU (by setting SIZEN=1 in slave's EBI_MCR). Anytime the master MCU needs to read or write the slave MCU with a different transfer size than the current value of the slave's SIZE field, the master MCU must first write the slave's SIZE field with the correct size for the subsequent transaction.

12.5.5.4 No Transfer Acknowledge ($\overline{TA}$) Pin

If an MCU has no $\overline{TA}$ pin available, this restricts the MCU to chip select accesses only. Non-chip select accesses have no way for the EBI to know which cycle to latch the data. The EBI has no built-in protection to prevent non-chip select accesses in this scenario; it is up to the user to make certain they set up chip selects and external memories correctly to ensure all external accesses fall in a valid chip select region.

12.5.5.5 No Transfer Error ($\overline{TEA}$) Pin

If an MCU has no $\overline{TEA}$ pin available, this eliminates the feature of terminating an access with $\overline{TEA}$. This means if an access times out in the EBI bus monitor, the EBI (master) will still terminate the access early, but there will be no external visibility of this termination, so the slave device might end up driving data much later, when a subsequent access is already underway. Therefore, the EBI bus monitor should be disabled when no $\overline{TEA}$ pin exists.

12.5.5.6 No Burst Data in Progress ($\overline{BDIP}$) Pin

If an MCU has no $\overline{BDIP}$ pin available, this eliminates burst support only if the burstable memory being used requires $\overline{BDIP}$ to burst. Many external memories use a self-timed configurable burst mechanism that does not require a dynamic burst indicator. Check the applicable external memory specification to see if $\overline{BDIP}$ is required in your system.

12.5.6 Summary of Differences from MPC5xx

Below is a summary list of the significant differences between this EBI used in the MPC5553/MPC5554 and that of the MPC5xx parts:

- SETA feature is no longer available
  - Chip select devices cannot use external $\overline{TA}$, instead must use wait state configuration.

- No memory controller support for external masters
  - Must configure each master in multi-master system to drive its own chip selects
- Changes in bit fields:
  - Removed these variable timing attributes from option register: CSNT, ACS, TRLX, EHTR
  - Removed LBDIP base register bit, now late $\overline{\text{BDIP}}$ assertion is default behavior
  - Modified TSIZ[0:1] functionality to only indicate size of current transfer, not give information on ensuing transfers that may be part of the same atomic sequence
  - The BL field of the base register has inverted logic from the MPC56x devices (0 = 8-beat burst on the MPC5500, 1 = 8-beat burst on the MPC56x)
- Removed reservation support on external bus
- Removed address type (AT), write-protect (WP), and dual-mapping features because these functions can be replicated by memory management unit (MMU) in e200z6 core
- Removed support for 8-bit ports
- Removed boot chip select operation
  - On-chip boot assist module (BAM) handles boot (and configuration of EBI registers)
- Open drain mode and pull-up resistors no longer required for multi-master systems, extra cycle needed to switch between masters
- Modified arbitration protocol to require extra cycles when switching between masters
- Added support for 32-bit coherent read and write non-chip select accesses in 16-bit data bus mode
- Misaligned accesses are not supported
- The MPC5553 has calibration features implemented by four calibration chip selects
- Removed support for 3-master systems
- Address decoding for external master accesses uses 4-bit code to determine internal slave instead of straight address decode

12.6 Revision History

Substantive Changes since Rev 3.0
In Section 12.1.2, “Overview,” changed “internal SRAM” t o “external SRAM”.
Changed references to 4 CAL_CS chip selects CAL_CS[0:3] to 3 CAL_CS chip selects [0] and [2:3]

Chapter 13
Flash Memory

13.1 Introduction

This section provides information about the Flash bus interface unit (FBIU) and the Flash memory block of the MPC5553/MPC5554.

13.1.1 Block Diagram

Figure 13-1 shows a block diagram of the Flash memory module. The FBIU is addressed through the system bus while the Flash control and status registers are addressed through the slave (peripheral) bus.

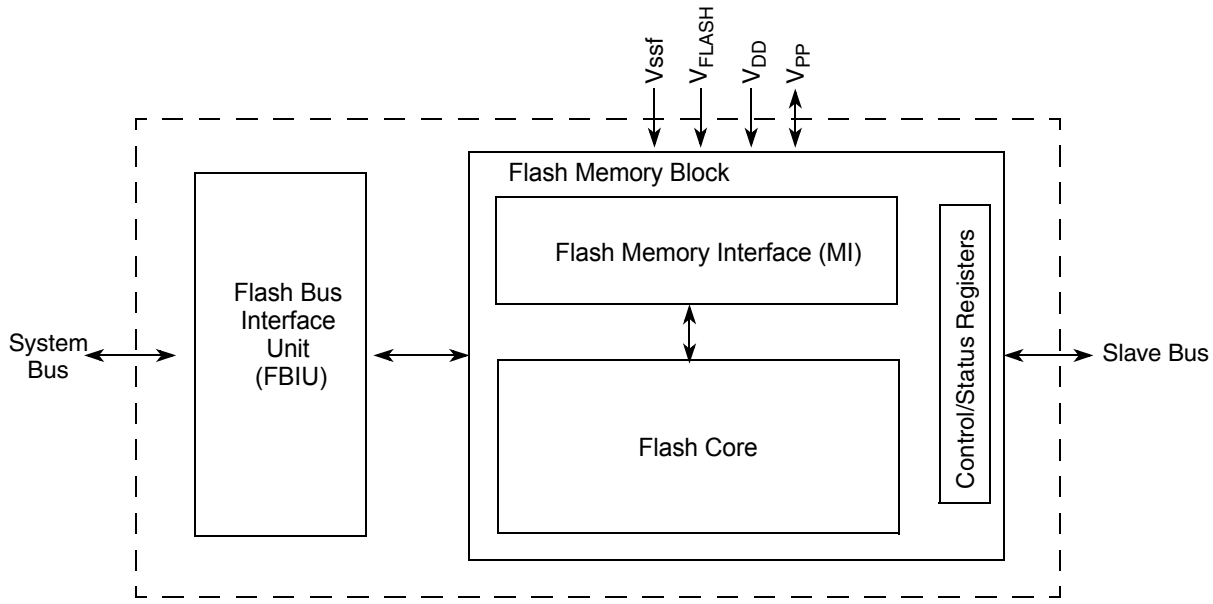


Figure 13-1. Flash System Block Diagram

13.1.2 Overview

The Flash module serves as electrically programmable and erasable non-volatile memory (NVM) that is ideal for program and data storage for single-chip applications allowing for field reprogramming without requiring external programming voltage sources. The module is a solid-state silicon memory device consisting of blocks of single-transistor storage elements.

The MPC5553/MPC5554 Flash contains a Flash bus interface unit (FBIU) and a Flash memory array. The Flash BIU interfaces the system bus to a dedicated Flash memory array controller. The FBIU supports a 64-bit data bus width at the system bus port, and a 256-bit read data interface to Flash memory. If enabled, the Flash BIU contains a two-entry, 256-bit prefetch buffer and a prefetch controller that prefetches sequential lines of data from the Flash array into the buffer. Prefetch buffer hits allow no-wait responses. Normal Flash array accesses are registered in the FBIU and are forwarded to the system bus on the following cycle, incurring at least three wait states (depending on the frequency), with additional wait states being determined by FLASH_BUICR[RWSC] (see Table 13-14).

Prefetch operations can be automatically controlled, and can be restricted to servicing a single bus master. Prefetches can also be restricted to being triggered for instruction or data accesses.

The Flash memory block is arranged as two functional units, the first being the Flash core. The Flash core is composed of arrayed non-volatile storage elements, sense amplifiers, row selects, column selects, charge pumps, ECC logic and redundancy logic. The arrayed storage elements in the Flash core are subdivided into physically separate units referred to as blocks.

The second functional unit of Flash memory is the memory interface (MI). The MI contains the registers and logic that control the operation of the Flash core. The MI is also the interface between the Flash module and the FBIU. The FBIU connects the MPC5553/MPC5554 system bus to the Flash module, and provides all system level customization and configuration functionality.

The Flash array has three address spaces. Low address space (LAS) is 256-Kbytes in size. Mid address space (MAS) is also 256-Kbytes in size. High address space (HAS) is 1.5 Mbyte in size in the MPC5554, and 1.0 MBytes in the MPC5553. Total address space is 2.0 MBytes for the MPC5554 and 1.5 Mbytes for the MPC5553.

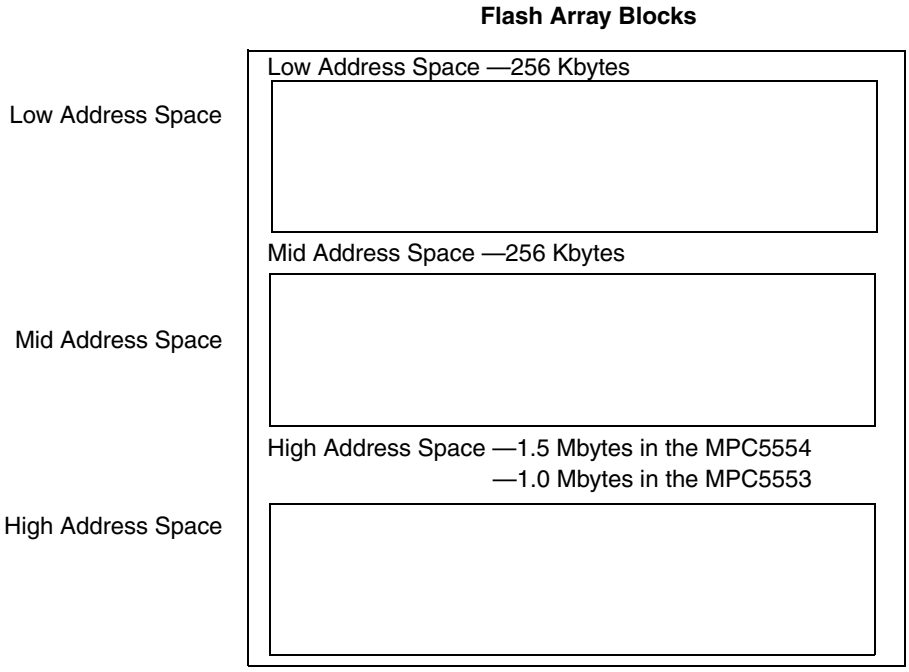


Figure 13-2. Flash Array Diagram

13.1.3 Features

The following list summarizes the key features of the FBIU:

- The FBIU system bus interface supports a 64-bit data bus. Byte, half-word, word, and double-word reads are supported. Only aligned word and double-word writes are supported.
- The FBIU provides configurable read buffering and line prefetch support. Two line read buffers (256 bits wide) and a prefetch controller are used to support single-cycle read responses for hits in the buffers.
- The FBIU provides hardware and software configurable read and write access protections on a per-master basis.
- The FBIU interface to the Flash array controller is pipelined with a depth of 1.
- The FBIU allows configurable access timing.
- The FBIU provides multiple-mapping support and mapping-based block access timing (0–31 additional cycles) allowing for emulation of other memory types.

The Flash memory array has the following features:

- Software programmable block program/erase restriction control for low, mid, and high address spaces.
- Erase of selected blocks.
- ECC with single-bit correction, double-bit detection.
- Page program of 1 to 8 consecutive 32-bit words within a page (recommended minimum is 2 words due to ECC).
- Embedded hardware program and erase algorithm.
- Read while write with multiple partitions.
- Stop mode for low power stand-by.
- Erase suspend, program suspend, and erase-suspended program.
- Automotive Flash that meets automotive endurance and reliability requirements. Shadow information is stored in a non-volatile shadow block.
- Independent program/erase of the shadow block.

13.1.4 Modes of Operation

13.1.4.1 User Mode

User mode is the default operating mode of the Flash memory block. In this mode, it is possible to read, write, program, and erase the Flash. Refer to Section 13.4.2, “Flash Memory Array: User Mode.”

13.1.4.2 Stop Mode

In stop mode (FLASH_MCR[STOP] = 1), all DC current sources in the Flash are disabled. Refer to Section 13.4.3, “Flash Memory Array: Stop Mode.”

13.2 External Signal Description

Table 13-1 shows a list of signals required for Flash.

Table 13-1. Signal Properties

Name	Function	Reset State
V_{FLASH}	Flash read power supply	NA
V_{PP}	Flash program/erase power supply	NA

13.2.1 Voltage for Flash Only (V_{FLASH})

V_{FLASH} is a supply required for reads of the Flash core. This voltage is specified as 3.3V with a tolerance of +/- 0.3V.

13.2.2 Program and Erase Voltage for Flash Only (V_{PP})

V_{PP} is a supply required for program and erase of the Flash core. This voltage is specified as 5V with a tolerance of -0.5V/+0.25V during program and erase operations. V_{PP} is required at all times, even during normal reads of Flash memory. During read operations, V_{PP} can be as high as 5.3V and as low as 3.0V.

13.3 Memory Map/Register Description

The Flash BIU occupies a 512-Mbyte portion of the address space. The actual Flash array is multiply-mapped within this space.

The MPC5553/MPC5554 internal flash has a feature that allows the internal flash timing to be modified to emulate an external memory, hence the name, external emulation mode. The upper 5 address lines are used to provide additional timing control that allows the FBIU response timing on the system bus (which must be controlled in order to provide for timing emulation of alternate memory types). Refer to Figure 13-3.

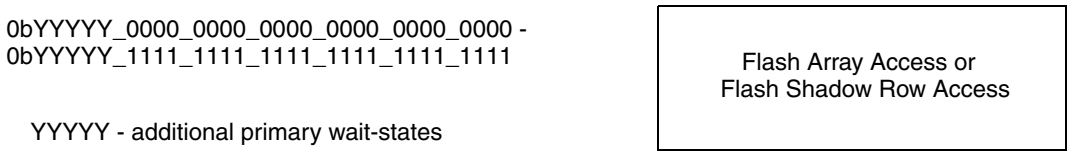


Figure 13-3. Flash BIU Address Scheme

This feature allows calibration parameters to be tested using an external memory; and then in production, the internal flash access timing is modified to match timing of the external memory. The access time of the internal flash is lengthened based on the address range being accessed. To access an area with a slower access time, the address is modified per Table 13-2.

Table 13-2. Internal Flash External Emulation Mode

Address Range		YYYYY	Wait States
0x0000_0000	0x001F_FFFF	00000	0
0x0100_0000	0x011F_FFFF	01000	8

Table 13-2. Internal Flash External Emulation Mode (continued)

Address Range		YYYYY	Wait States
0x0200_0000	0x021F_FFFF	10000	16
0x0300_0000	0x031F_FFFF	11000	24
0x0400_0000	0x041F_FFFF	00001	1
0x0500_0000	0x051F_FFFF	01001	9
0x0600_0000	0x061F_FFFF	10001	17
0x0700_0000	0x071F_FFFF	11001	25
0x0800_0000	0x081F_FFFF	00010	2
0x0900_0000	0x091F_FFFF	01010	10
0x0A00_0000	0x0A1F_FFFF	10010	18
0x0B00_0000	0x0B1F_FFFF	11010	26
0x0C00_0000	0x0C1F_FFFF	00011	3
0x0D00_0000	0x0D1F_FFFF	01011	11
0x0E00_0000	0x0E1F_FFFF	10011	19
0x0F00_0000	0x0F1F_FFFF	11011	27
0x1000_0000	0x101F_FFFF	00100	4
0x1100_0000	0x111F_FFFF	01100	12
0x1200_0000	0x121F_FFFF	10100	20
0x1300_0000	0x131F_FFFF	11100	28
0x1400_0000	0x141F_FFFF	00101	5
0x1500_0000	0x151F_FFFF	01101	13
0x1600_0000	0x161F_FFFF	10101	21
0x1700_0000	0x171F_FFFF	11101	29
0x1800_0000	0x181F_FFFF	00110	6
0x1900_0000	0x191F_FFFF	01110	14
0x1A00_0000	0x1A1F_FFFF	10110	22
0x1B00_0000	0x1B1F_FFFF	11110	30
0x1C00_0000	0x1C1F_FFFF	00111	7
0x1D00_0000	0x1D1F_FFFF	01111	15
0x1E00_0000	0x1E1F_FFFF	10111	23
0x1F00_0000	0x1F1F_FFFF	11111	31

13.3.1 Flash Memory Map

Table 13-3 shows the Flash array memory map and how it is mapped assuming byte addressing.

Base addresses for the MPC5554 and the MPC5553 are the following:

- Shadow base address = 0x00FF_FC00
- Array base address = 0x0000_0000
- Control registers base address = 0xC3F8_8000

Table 13-3. Module Flash Array Memory Map

Byte Address	Use	Access
Shadow base + 0x00_0000– Shadow base + 0x00_03FF	Shadow block space (1024 Bytes)	User
Array Base + 0x00_0000– Array Base + 0x03_FFFF	Low address space (256 Kbytes)	User
Array Base + 0x04_0000– Array Base + 0x07_FFFF	Mid address space (256 Kbytes)	User
Array Base + 0x08_0000 to Array Base + 0x1F_FFFF (MPC5554) or to Array Base + 0x17_FFFF (MPC5553)	High address space (1.5 Mbyte in MPC5554 or 1.0 Mbyte in MPC5553)	User

Table 13-4 shows how the array is partitioned into three address spaces — low, mid, and high — and into partitions and blocks.

Table 13-4. Flash Partitions

Address	Use	Block	Size	Partition
Array Base + 0x00_0000	Low Address Space	L0	16K	1
Array Base + 0x00_4000		L1	48K	
Array Base + 0x01_0000		L2	48K	
Array Base + 0x01_C000		L3	16K	
Array Base + 0x02_0000		L4	64K	2
Array Base + 0x03_0000		L5	64K	
Array Base + 0x04_0000	Mid Address Space	M0	128K	3
Array Base + 0x06_0000		M1	128K	
Array Base + 0x08_0000	High Address Space (MPC5554 and MPC5553)	H0	128K	4
Array Base + 0x0A_0000		H1	128K	
Array Base + 0x0C_0000		H2	128K	5
Array Base + 0x0E_0000		H3	128K	
Array Base + 0x10_0000		H4	128K	6
Array Base + 0x12_0000		H5	128K	
Array Base + 0x14_0000		H6	128K	7
Array Base + 0x16_0000		H7	128K	

Table 13-4. Flash Partitions (continued)

Address	Use	Block	Size	Partition
Array Base + 0x18_0000	High Address Space (MPC5554 Only)	H8[1]	128K	8[1]
Array Base + 0x1A_0000		H9[1]	128K	
Array Base + 0x1C_0000		H10[1]	128K	9[1]
Array Base + 0x1E_0000		H11[1]	128K	
Array Base + 0xFF_FC00	Shadow block space	S	472	All[2]
Array Base+ 0xFF_FDD8	Flash shadow row, serial passcode		8	
Array Base+ 0xFF_FDE0	Flash shadow row, control word		4	
Array Base+ 0xFF_FDE4	For general use		4	
Array Base+ 0xFF_FDE8	Flash shadow row, FLASH_LMLR reset configuration		4	
Array Base+ 0xFF_FDEC	For general use		4	
Array Base+ 0xFF_FDF0	Flash shadow row, FLASH_HLR reset configuration		4	
Array Base+ 0xFF_FDF4	For general use		4	
Array Base+ 0xFF_FDF8	Flash Shadow Row, FLASH_SLMLR reset configuration		4	
Array Base+ 0xFF_FDFC – 0xFF_FFFF	For general use		516	

[1] Not available in the MPC5553; only available in the MPC5554.

[2] The shadow row does not support RWW. See Section 13.4.2.5, “Flash Shadow Block.

Table 13-5 shows the register set for the Flash module.

Table 13-5. Module Register Memory Map

Byte Address	Register Name	Register Description	Size (bits)
Register Base + 0x00	FLASH_MCR	Module configuration register	32
Register Base + 0x04	FLASH_LMLR	Low/mid address space block locking register	32
Register Base + 0x08	FLASH_HLR	High address space block locking register	32
Register Base + 0x0C	FLASH_SLMLR	Secondary low/mid address space block locking register	32
Register Base + 0x10	FLASH_LMSR	Low/mid address space block select register	32
Register Base + 0x14	FLASH_HSR	High address space block select register	32
Register Base + 0x18	FLASH_AR	Address register	32
Register Base + 0x1C	FLASH_BIUCR	Flash bus interface unit control register	32

Table 13-5. Module Register Memory Map (continued)

Byte Address	Register Name	Register Description	Size (bits)
Register Base + 0x20	FLASH_BIUAPR	Flash bus interface unit access protection register	32
Register Base +0x30 to Register Base +0x7FFF	—	Reserved	—

13.3.2 Register Descriptions

The Flash registers are detailed in the following sections.

13.3.2.1 Module Configuration Register (FLASH_MCR)

A number of module configuration register (FLASH_MCR) bits are protected from a write while another bit or set of bits are in a specific state. These locks are discussed in relationship to each bit in this section. Simultaneously writing bits which lock each other out is discussed in Section 13.3.2.1.1, "MCR Simultaneous Register Writes." The MCR is always available to be read except when the Flash module is disabled.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SIZE				0	LAS			0	0	0	MAS
W																
MPC5554 Reset	0	0	0	0	0	1	1	1	0	1	1	0	0	0	0	0
MPC5553 Reset	0	0	0	0	0	1	0	1	0	1	1	0	0	0	0	0
Reg Addr	Base (0xC3F8_8000) + 0x0000															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EER	RWE	1	1	PEAS	DONE	PEG	0	0	STOP	0	PGM	PSUS	ERS	ESUS	EHV
W	w1c	w1c														
MPC5554 Reset	0	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0
MPC5553 Reset	0	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0
Reg Addr	Base (0xC3F8_8000) + 0x0000															

Figure 13-4. Module Configuration Register (FLASH_MCR)

Table 13-6. FLASH_MCR Field Descriptions

Bits	Name	Description
0–3	—	Reserved.

Table 13-6. FLASH_MCR Field Descriptions (continued)

Bits	Name	Description
4–7	SIZE [0:3]	Array space size. Dependent upon the size of the Flash module. All possible values of SIZE and the configuration to which each value corresponds are shown below. SIZE is read only. 0101 Total array size is 1.5 Mbytes (MPC5553) 0111 Total array size is 2 Mbytes (MPC5554)
8	—	Reserved.
9–11	LAS [0:2]	Low address space. Corresponds to the configuration of the low address space. All possible values of LAS and the configuration to which each value corresponds are shown below. LAS is read only. 110 The LAS value of 110 provides two 16-Kbyte blocks, two 48-Kbyte blocks, and two 64-Kbyte blocks. This is the space configuration for both the MPC5553 and the MPC5554.
12–14	—	Reserved.
15	MAS	Mid address space size. Corresponds to the configuration of the mid address space. MAS is read only. The value of the parameter for this device is shown in bold. **Note:** The MAS encoding for the MPC5554 and for the MPC5553 is 0. 0 Two 128-Kbyte blocks are available
16	EER	ECC event error. Provides information on previous reads; if a double bit detection occurred, the EER bit will be set to a 1. This bit must then be cleared, or a reset must occur before this bit will return to a 0 state. This bit may not be set by the user. In the event of a single bit detection and correction, this bit will not be set. If EER is not set, or remains 0, this indicates that all previous reads (from the last reset, or clearing of EER) were correct. Since this bit is an error flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 will have no effect. 0 Reads are occurring normally. 1 An ECC Error occurred during a previous read. **Note:** This bit can be set on speculative prefetches that cause double bit error detection.
17	RWE	Read while write event error. Provides information on previous RWW reads. If a read while write error occurs, this bit will be set to 1. This bit must then be cleared, or a reset must occur before this bit will return to a 0 state. This bit may not be written to a 1 by the user. If RWE is not set, or remains 0, this indicates that all previous RWW reads (from the last reset, or clearing of RWE) were correct. Since this bit is an error flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 will have no effect. 0 Reads are occurring normally. 1 A read while write error occurred during a previous read.
18–19	—	Reserved.
20	PEAS	Program/erase access space. Indicates which space is valid for program and erase operations, either main array space or shadow space. PEAS is read only. 0 Shadow address space is disabled for program/erase and main address space enabled 1 Shadow address space is enabled for program/erase and main address space disabled.

Table 13-6. FLASH_MCR Field Descriptions (continued)

Bits	Name	Description
21	DONE	State machine status. Indicates if the Flash module is performing a high voltage operation. DONE is set to a 1 on termination of the Flash module reset and at the end of program and erase high voltage sequences. 0 Flash is executing a high voltage operation. 1 Flash is not executing a high voltage operation.
22	PEG	Program/erase good. Indicates the completion status of the last Flash program or erase sequence for which high voltage operations were initiated. The value of PEG is updated automatically during the program and erase high voltage operations. Aborting a program/erase high voltage operation will cause PEG to be cleared, indicating the sequence failed. PEG is set to a 1 when the module is reset. PEG is read only. The value of PEG is valid only when PGM = 1 and/or ERS = 1 and after DONE has transitioned from 0 to 1 due to an abort or the completion of a program/erase operation. PEG is valid until PGM/ERS makes a 1 to 0 transition or EHV makes a 0 to 1 transition. The value in PEG is not valid after a 0 to 1 transition of DONE caused by PSUS or ESUS being set to logic 1. A diagram presenting PEG valid times is presented in Figure 13-5. If PGM and ERS are both 1 when DONE makes a qualifying 0 to 1 transition the value of PEG indicates the completion status of the PGM sequence. This happens in an erase-suspended program operation. 0 Program or erase operation failed. 1 Program or erase operation successful.
23–24	—	Reserved.
25	STOP	Stop mode enabled. Puts the Flash into stop mode. Changing the value in STOP from a 0 to a 1 places the Flash module in stop mode. A 1 to 0 transition of STOP returns the Flash module to normal operation. STOP may be written only when PGM and ERS are low. When STOP = 1, only the STOP bit in the MCR can be written. In STOP mode all address spaces, registers, and register bits are deactivated except for the FLASH_MCR[STOP] bit. 0 Flash is not in stop mode; the read state is active. 1 Flash is in stop mode.
26	—	Reserved.
27	PGM	Program. Used to set up Flash for a program operation. A 0 to 1 transition of PGM initiates an Flash program sequence. A 1 to 0 transition of PGM ends the program sequence. PGM can be set only under one of the following conditions: • User mode read (STOP and ERS are low). • Erase suspend[1] (ERS and ESUS are 1) with EHV low. PGM can be cleared by the user only when EHV are low and DONE is high. PGM is cleared on reset. 0 Flash is not executing a program sequence. 1 Flash is executing a program sequence.
28	PSUS	Program suspend. Indicates the Flash module is in program suspend or in the process of entering a suspend state. The Flash module is in program suspend when PSUS = 1 and DONE = 1. PSUS can be set high only when PGM and EHV are high. A 0 to 1 transition of PSUS starts the sequence which sets DONE and places the Flash in program suspend. PSUS can be cleared only when DONE and EHV are high. A 1 to 0 transition of PSUS with EHV = 1 starts the sequence which clears DONE and returns the Flash module to program. The Flash module cannot exit program suspend and clear DONE while EHV is low. PSUS is cleared on reset. 0 Program sequence is not suspended. 1 Program sequence is suspended.

Table 13-6. FLASH_MCR Field Descriptions (continued)

Bits	Name	Description
29	ERS	Erase. Used to set up Flash for an erase operation. A 0 to 1 transition of ERS initiates an Flash erase sequence. A 1 to 0 transition of ERS ends the erase sequence. ERS can be set only in a normal operating mode read (STOP and PGM are low). ERS can be cleared by the user only when ESUS and EHV are low and DONE is high. ERS is cleared on reset. 0 Flash is not executing an erase sequence. 1 Flash is executing an erase sequence.
30	ESUS	Erase suspend. Indicates that the Flash module is in erase suspend or in the process of entering a suspend state. The Flash module is in erase suspend when ESUS = 1 and DONE = 1. ESUS can be set high only when ERS and EHV are high and PGM is low. A 0 to 1 transition of ESUS starts the sequence which sets DONE and places the Flash in erase suspend. ESUS can be cleared only when DONE and EHV are high and PGM is low. A 1 to 0 transition of ESUS with EHV = 1 starts the sequence which clears DONE and returns the Flash module to erase mode. The Flash module cannot exit erase suspend and clear DONE while EHV is low. ESUS is cleared on reset. 0 Erase sequence is not suspended. 1 Erase sequence is suspended.
31	EHV	Enable high voltage. Enables the Flash module for a high voltage program/erase operation. EHV is cleared on reset. EHV must be set after an interlock write to start a program/erase sequence. EHV may be set, initiating a program/erase, after an interlock write under one of the following conditions: • Erase (ERS = 1, ESUS = 0). • Program (ERS = 0, ESUS = 0, PGM = 1, PSUS = 0). • Erase-suspended program (ERS = 1, ESUS = 1, PGM = 1, PSUS = 0). If a program operation is to be initiated while an erase is suspended the user must clear EHV while in erase suspend before setting PGM. In normal operation, a 1 to 0 transition of EHV with DONE high, PSUS and ESUS low terminates the current program/erase high voltage operation. When an operation is aborted[2], there is a 1 to 0 transition of EHV with DONE low and the suspend bit for the current program/erase sequence low. An abort causes the value of PEG to be cleared, indicating a failed program/erase; address locations being operated on by the aborted operation contain indeterminate data after an abort. A suspended operation cannot be aborted. EHV may be written during suspend. EHV must be high for the Flash to exit suspend. EHV may not be written after a suspend bit is set high and before DONE has transitioned high. EHV may not be set low after the current suspend bit is set low and before DONE has transitioned low. 0 Flash is not enabled to perform a high voltage operation. 1 Flash is enabled to perform a high voltage operation.

1 In an erase-suspended program, programming Flash locations in blocks which were being operated on in the erase may corrupt Flash core data. This should be avoided due to reliability implications.

2 Aborting a high voltage operation will leave Flash core addresses in an indeterminate data state. This may be recovered by executing an erase on the affected blocks.

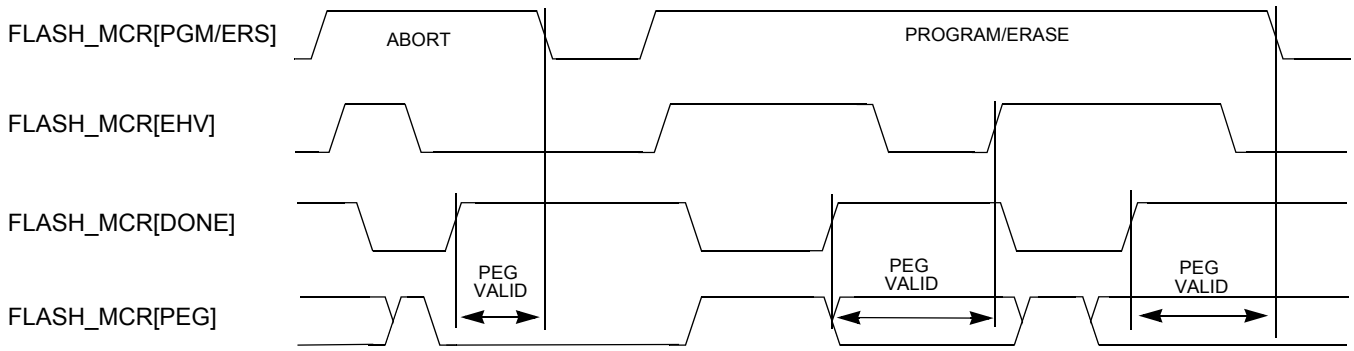


Figure 13-5. PEG Valid Times

13.3.2.1.1 MCR Simultaneous Register Writes

A number of MCR bits are protected against write when another bit or set of bits is in a specific state. These write locks are covered on a bit by bit basis in Section 13.3.2.1, "Module Configuration Register (FLASH_MCR)." The write locks detailed in that section do not consider the effects of trying to write two or more bits simultaneously. The effects of writing bits simultaneously which would put the Flash module in an illegal state are detailed here.

The Flash does not allow the user to write bits simultaneously which would put the device into an illegal state. This is implemented through a priority mechanism among the bits. The bit changing priorities are detailed in Table 13-7.

Table 13-7. MCR Bit Set/Clear Priority Levels

Priority Level	MCR Bits
1	STOP
2	ERS
3	PGM
4	EHV
5	ESUS, PSUS

If the user attempts to write two or more MCR bits simultaneously then only the bit with the highest priority level will be written. Setting two bits with the same priority level is prevented by existing write locks and will not put the Flash in an illegal state.

For example, setting FLASH_MCR[STOP] and FLASH_MCR[PGM] simultaneously results in only FLASH_MCR[STOP] being set. Attempting to clear FLASH_MCR[EHV] while setting FLASH_MCR[PSUS] will result in FLASH_MCR[EHV] being cleared, while FLASH_MCR[PSUS] will remain unaffected.

13.3.2.2 Low/Mid Address Space Block Locking Register (FLASH_LMLR)

The low and mid address block locking register provides a means to protect blocks from being modified. These bits along with bits in the secondary LMLOCK field (FLASH_SLMLR), determine if the block is locked from program or erase. An "OR"' of FLASH_LMLR and FLASH_SLMLR determine the final lock status. See Section 13.3.2.4, "Secondary Low/Mid Address Space Block Locking Register (FLASH_SLMLR)" for more information on FLASH_SLMLR.

NOTE

In the event that blocks are not present (due to configuration or total memory size), the LOCK bits will default to locked, and will not be writable. The reset value will always be 1 (independent of the shadow block), and register writes will have no effect.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	LME	0	0	0	0	0	0	0	0	0	0	SLOCK	MLOCK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1[1]	1[1]	1[1]	1[1]	1[1]
Reg Addr	Base (0xC3F8_8000) + 0x0004															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LLOCK															
W																
Reset	1[1]	1[1]	1[1]	1[1]	1[1]	1[1]	1[1]	1[1]	1[1]	1[1]	1[1]	1[1]	1[1]	1[1]	1[1]	1[1]
Reg Addr	Base (0xC3F8_8000) + 0x0004															

[1] The reset value of these bits is determined by Flash values in the shadow row. An erased array will cause the reset value to be 1

Figure 13-6. Low/Mid Address Space Block Locking Register (FLASH_LMLR)

Table 13-8. FLASH_LMLR Field Descriptions

Bits	Name	Description
0	LME	Low and mid address lock enable. Enables the locking register fields (SLOCK, MLOCK and LLOCK) to be set or cleared by register writes. This bit is a status bit only, and may not be written or cleared, and the reset value is 0. The method to set this bit is to write a password, and if the password matches, the LME bit will be set to reflect the status of enabled, and is enabled until a reset operation occurs. For LME, the password 0xA1A1_1111 must be written to the FLASH_LMLR. 0 Low and mid address locks are disabled, and cannot be modified. 1 Low and mid address locks are enabled and can be written.
1–10	—	Reserved.
11	SLOCK	Shadow lock. Locks the shadow row from programs and erases. The SLOCK bit is not writable if a high voltage operation is suspended. Upon reset, information from the shadow row is loaded into the SLOCK bit. The SLOCK bit may be written as a register. Reset will cause the bits to go back to their shadow row value. The default value of the SLOCK bit (assuming the corresponding shadow row bit is erased) would be locked. SLOCK is not writable unless LME is high. 0 Shadow row is available to receive program and erase pulses. 1 Shadow row is locked for program and erase.

Table 13-8. FLASH_LMLR Field Descriptions (continued)

Bits	Name	Description
12–15	MLOCK [0:3]	Mid address block lock. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. Likewise the lock register is not writable if a high voltage operation is suspended. Upon reset, information from the shadow row is loaded into the block registers. The LOCK bits may be written as a register. Reset will cause the bits to go back to their shadow row value. The default value of the LOCK bits (assuming erased fuses) would be locked. In the event that blocks are not present (due to configuration or total memory size), the LOCK bits will default to locked, and will not be writable. The reset value will always be 1 (independent of the shadow row), and register writes will have no effect. MLOCK is not writable unless LME is high. As an example of how the LOCK bits are used, if a configuration has two 128-Kbyte blocks in the low address space (MCR-LAS = 3'b000), and four 64-Kbyte blocks in the mid address space (MCR-MAS = 1), the first 128-Kbyte block located at address Array Base + 0 will correspond to MLOCK0, the second 128-Kbyte block will correspond to MLOCK1. The first mid address space block (64 Kbytes) located at address Array Base + 40000h will correspond to MLOCK0, the second mid address space block will correspond to MLOCK1, the third mid address space block will correspond to MLOCK2, and the final mid address space block will correspond to MLOCK3.
16–31	LLOCK [0:15]	Low address block lock. These bits have the same description and attributes as MLOCK. As an example of how the LLOCK bits are used, if a configuration has sixteen 16-Kbyte blocks in the low address space (MCR-LAS = 3'b011), the block residing at address Array Base + 0, will correspond to LLOCK0. The next 16-Kbyte block will correspond to LLOCK1, and so on up to LLOCK15.

13.3.2.3 High Address Space Block Locking Register (FLASH_HLR)

The high address space block locking register provides a means to protect blocks from being modified.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HBE	0	0	0	HLOCK											
W																
Reset	0	0	0	0	1[1]	1[1]	1[1]	1[1]	1[1]	1[1]	1[1]	1[1]	1[1]	1[1]	1[1]	1[1]
Reg Addr	Base (0xC3F8_8000) + 0x0008															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HLOCK															
W																
Reset	1[1]	1[1]	1[1]	1[1]	1[1]	1[1]	1[1]	1[1]	1[1]	1[1]	1[1]	1[1]	1[1]	1[1]	1[1]	1[1]
Reg Addr	Base (0xC3F8_8000) + 0x0008															

[1] The reset value of these bits is determined by Flash values in the shadow row. An erased array will cause the reset value to be 1.

Figure 13-7. High Address Space Block Locking Register (FLASH_HLR)

Table 13-9. FLASH_HLR Field Descriptions

Bits	Name	Description
0	HBE	High address lock enable. Enables the locking field (HLOCK) to be set or cleared by register writes. This bit is a status bit only, and may not be written or cleared, and the reset value is 0. The method to set this bit is to provide a password, and if the password matches, the HBE bit will be set to reflect the status of enabled, and is enabled until a reset operation occurs. For HBE, the password 0xB2B2_2222 must be written to FLASH_HLR. 0 High address locks are disabled, and cannot be modified. 1 High address locks are enabled to be written.
1–3	—	Reserved.
4–31	HLOCK [0:27]	High address space block lock. Has the same characteristics as MLOCK. See Section 13.3.2.2, "Low/Mid Address Space Block Locking Register (FLASH_LMLR)" for more information. The block numbering for High Address space starts with HLOCK[0] and continues until all blocks are accounted. HLOCK is not writable unless HBE is set. In the event that blocks are not present (due to configuration or total memory size), the HLOCK bits will default to locked, and will not be writable.

13.3.2.4 Secondary Low/Mid Address Space Block Locking Register (FLASH_SLMLR)

The FLASH_SLMLR provides an alternative means to protect blocks from being modified. These bits along with bits in the LMLOCK field (FLASH_LMLR), determine if the block is locked from program or erase. An "OR" of FLASH_LMLR and FLASH_SLMLR determine the final lock status. See Section 13.3.2.2, "Low/Mid Address Space Block Locking Register (FLASH_LMLR)" for more information on FLASH_LMLR.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SLE	0	0	0	0	0	0	0	0	0	0	SSLOCK	SMLOCK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1[1]	1[1]	1[1]	1[1]	1[1]
Reg Addr	Base (0xC3F8_8000) + 0x000C															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SLLOCK															
W																
Reset	1[1]	1[1]	1[1]	1[1]	1[1]	1[1]	1[1]	1[1]	1[1]	1[1]	1[1]	1[1]	1[1]	1[1]	1[1]	1[1]
Reg Addr	Base (0xC3F8_8000) + 0x000C															

[1] The reset value of these bits is determined by Flash values in the shadow row. An erased array will cause the reset value to be 1

Figure 13-8. Secondary Low/Mid Address Space Block Locking Register (FLASH_SLMLR)

Table 13-10. FLASH_SLMLR Field Descriptions

Bits	Name	Description
0	SLE	Secondary low and mid address lock enable. Enables the secondary lock fields (SSLOCK, SMLOCK, and SLLOCK) to be set or cleared by register writes. This bit is a status bit only, and may not be written or cleared, and the reset value is 0. The method to set this bit is to provide a password, and if the password matches, the SLE bit will be set to reflect the status of enabled, and is enabled until a reset operation occurs. For SLE, the password 0xC3C3_3333 must be written to the FLASH_SLMLR. 0 Secondary low and mid address locks are disabled, and cannot be modified. 1 Secondary low and mid address locks are enabled to be written.
1–10	—	Reserved.
11	SSLOCK	Secondary shadow lock. An alternative method that may be used to lock the shadow row from programs and erases. SSLOCK has the same description as SLOCK in Section 13.3.2.2, “Low/Mid Address Space Block Locking Register (FLASH_LMLR).” SSLOCK is not writable unless SLE is high.
12–15	SMLOCK [0:3]	Secondary mid address block lock. Alternative method that may be used to lock the mid address space blocks from programs and erases. SMLOCK has the same description as MLOCK in section Section 13.3.2.2, “Low/Mid Address Space Block Locking Register (FLASH_LMLR).” SMLOCK is not writable unless SLE is set. In the event that blocks are not present (due to configuration or total memory size), the SMLOCK bits will default to locked, and will not be writable.
16–31	SLLOCK [0:15]	Secondary low address block lock. These bits are an alternative method that may be used to lock the low address space blocks from programs and erases. SLLOCK has the same description as LLOCK in Section 13.3.2.2, “Low/Mid Address Space Block Locking Register (FLASH_LMLR). SLLOCK is not writable unless SLE is high. In the event that blocks are not present (due to configuration or total memory size), the SLLOCK bits will default to locked, and will not be writable.

13.3.2.5 Low/Mid Address Space Block Select Register (FLASH_LMSR)

The FLASH_LMSR provides a means to select blocks to be operated on during erase.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	MSEL			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base (0xC3F8_8000) + 0x0010															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LSEL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base (0xC3F8_8000) + 0x0010															

Figure 13-9. Low/Mid Address Space Block Select Register (FLASH_LMSR)

Table 13-11. FLASH_LMSR Field Descriptions

Bits	Name	Description
0–11	—	Reserved.
12–15	MSEL [0:3]	Mid address space block select. Values in the selected register signify that a block(s) is or is not selected for erase. The reset value for the select registers is 0, or unselected. The blocks must be selected (or unselected) before doing an erase interlock write as part of the erase sequence. The select register is not writable once an interlock write is completed or if a high voltage operation is suspended. In the event that blocks are not present (due to configuration or total memory size), the corresponding SELECT bits will default to unselected, and will not be writable. The reset value will always be 0, and register writes will have no effect. A description of how blocks are numbered is detailed in Section 13.3.2.2, "Low/Mid Address Space Block Locking Register (FLASH_LMLR)." 0b0000 Mid address space blocks are *not* selected for erase 0b0001 One mid address space block is selected for erase 0b0011 Two mid address space blocks are selected for erase
16–31	LSEL [0:15]	Low address space block select. Used to select blocks in the low address space; these have the same description and attributes as the MSEL bits 0b0000 Low address space blocks are *not* selected for erase 0b0001 One low address space block is selected for erase 0b0011 Two low address space blocks are selected for erase 0b0111 Three low address space blocks are selected for erase 0b1111 Four low address space blocks are selected for erase 0b1_1111 Five low address space blocks are selected for erase 0b11_1111 Six low address space blocks are selected for erase

13.3.2.6 High Address Space Block Select Register (FLASH_HSR)

The FLASH_HSR provides a means to select blocks to be operated on.

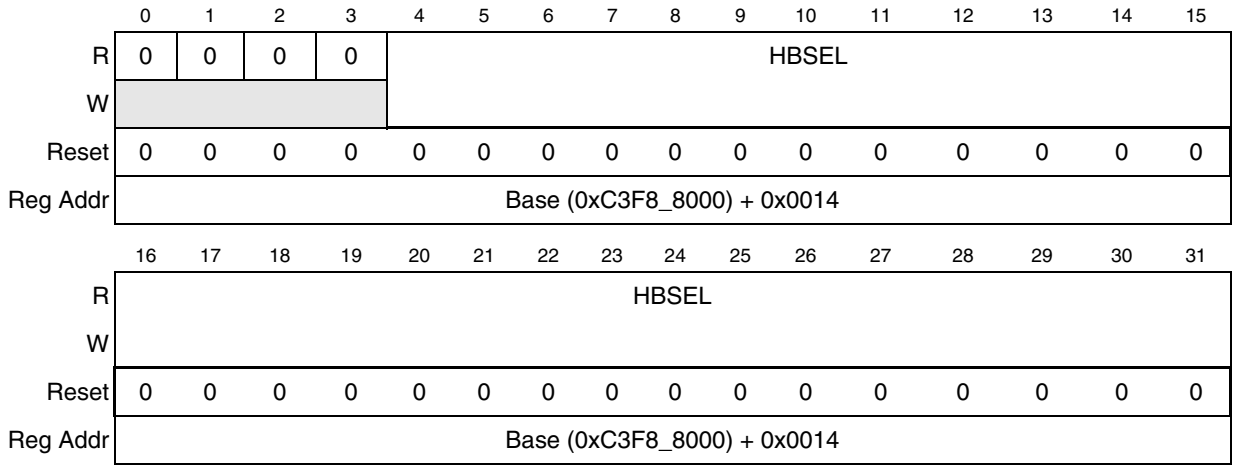


Figure 13-10. High Address Space Block Select Register (FLASH_HSR)

Table 13-12. FLASH_HSR Field Descriptions

Bits	Name	Description
0–3	—	Reserved.
4–31	HBSEL [0:27]	High address space block select. Has the same characteristics as MSEL. For more information see Section 13.3.2.5, "Low/Mid Address Space Block Select Register (FLASH_LMSR)." In both the MPC5553 and the MPC5554: 0b0000 High address space blocks are *not* selected for erase 0b0001 One high address space block is selected for erase 0b0011 Two high address space blocks are selected for erase 0b0111 Three high address space blocks are selected for erase 0b1111 Four high address space blocks are selected for erase 0b1_1111 Five high address space blocks are selected for erase 0b11_1111 Six high address space blocks are selected for erase 0b111_1111 Seven high address space blocks are selected for erase 0b1111_1111 Eight high address space blocks are selected for erase and in the MPC5554 only: 0b1_1111_1111 Nine high address space blocks are selected for erase 0b11_1111_1111 Ten high address space blocks are selected for erase 0b111_1111_1111 Eleven high address space blocks are selected for erase 0b1111_1111_1111 Twelve high address space blocks are selected for erase

13.3.2.7 Address Register (FLASH_AR)

The FLASH_AR provides the first failing address in the event of ECC event error (FLASH_MCR[EER] set), as well as providing the address of a failure that may have occurred in a state machine operation (FLASH_MCR[PEG] cleared). ECC event errors take priority over state machine errors. This is especially valuable in the event of a RWW operation, where the read senses an ECC error and the state machine fails simultaneously. This address is always a double-word address that selects 64 bits.

In normal operating mode, the FLASH_AR is not writable.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	ADDR					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base (0xC3F8_8000) + 0x0018															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ADDR													0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base (0xC3F8_8000) + 0x0018															

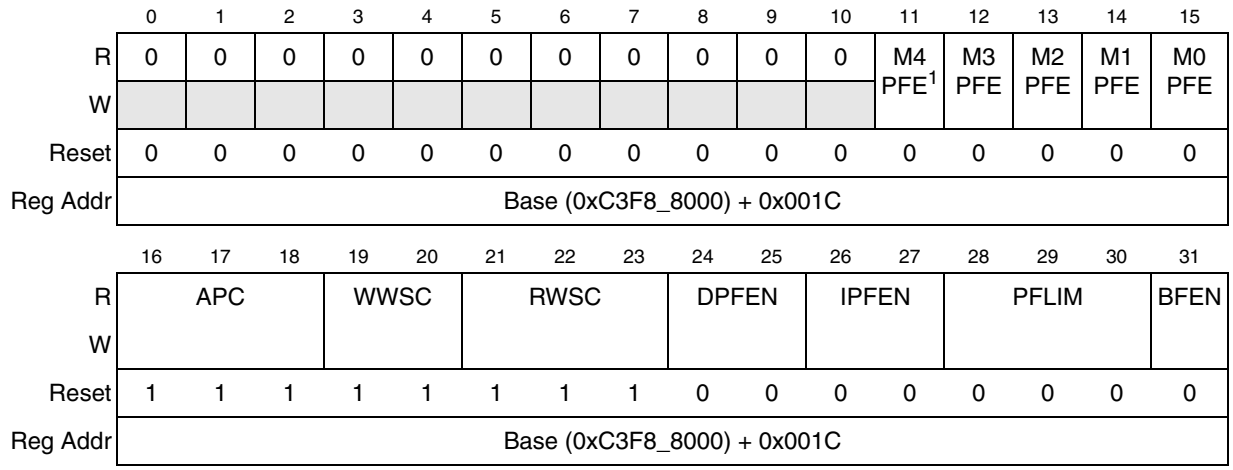
Figure 13-11. Address Register (FLASH_AR)

Table 13-13. FLASH_AR Field Descriptions

Bits	Name	Description
0–9	—	Reserved.
10–28	ADDR [3:21]	Double-word address of first failing address in the event of an ECC error, or the address of a failure occurring during state machine operation.
29–31	ADDR [0:2]	Always read as 0.

13.3.2.8 Flash Bus Interface Unit Control Register (FLASH_BIUCR)

The FLASH_BIUCR is the control register for the set up and control of the Flash interface. This register must not be written while executing from Flash. This register should only be written in a 32-bit write operation.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	M4 PFE[1]	M3 PFE	M2 PFE	M1 PFE	M0 PFE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base (0xC3F8_8000) + 0x001C															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	APC			WWSC		RWSC			DPFEN		IPFEN		PFLIM			BFEN
W																
Reset	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
Reg Addr	Base (0xC3F8_8000) + 0x001C															

Figure 13-12. Flash Bus Interface Unit Control Register (FLASH_BIUCR)

[1] M4PFE is functional only in the MPC5553.

Table 13-14. FLASH_BIUCR Field Descriptions

Bits	Name	Description
0-11	—	Reserved
12–15	M*n*PFE	Master *n* prefetch enable. Used to control whether prefetching may be triggered based on the master ID of a requesting master. These bits are cleared by hardware reset. Refer to Table 7-1. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master These fields are identified as follows: M4PFE = FEC (in MPC5553 only) M3PFE= EBI M2PFE= eDMA M1PFE= Nexus M0PFE= e200z6 core
16–18	APC[1]	Address pipelining control. Used to control the number of cycles between pipelined access requests. This field must be set to a value corresponding to the operating frequency of the system clock. The required settings are documented in Table 13-15. 000 Reserved 001 Access requests require one hold cycle 010 Access requests require two hold cycles ... 110 Access requests require 6 hold cycles 111 No address pipelining
19–20	WWSC [1]	Write wait state control. Used to control the timing for array writes. This field must be set to a value corresponding to the operating frequency of the system clock. The required settings are documented in Table 13-15. 00 Reserved 01 One wait state 10 Two wait states 11 Three wait states

Table 13-14. FLASH_BIUCR Field Descriptions (continued)

Bits	Name	Description
21–23	RWSC [1]	Read wait state control. Used to control the Flash array access time for array reads. This field must be set to a value corresponding to the operating frequency of the system clock. The required settings are documented in Table 13-15. 000 Zero wait states 001 One wait state ... 111 Seven wait states
24–25	DPFEN	Data prefetch enable. Enables or disables prefetching initiated by a data read access. This field is cleared by hardware reset. 00 No prefetching is triggered by a data read access 01 Prefetching may be triggered only by a data burst read access 10 Reserved 11 Prefetching may be triggered by any data read access
26–27	IPFEN	Instruction prefetch enable. Enables or disables prefetching initiated by an instruction read access. This field is cleared by hardware reset. 00 No prefetching is triggered by an instruction read access 01 Prefetching may be triggered only by an instruction burst read access 10 Reserved 11 Prefetching may be triggered by any instruction read access
28–30	PFLIM	Prefetch limit. Controls the prefetch algorithm used by the FBIU prefetch controller. This field defines a limit on the maximum number of sequential prefetches which will be attempted between buffer misses. This field is cleared by hardware reset. 000 No prefetching is performed 001 A single additional line (next sequential) is prefetched on a buffer miss 010 Up to two additional lines may be prefetched following each buffer miss before prefetching is halted. A single additional line (next sequential) is prefetched on a buffer miss, and the next sequential line is prefetched on a buffer hit (if not already present). 011 Up to three additional lines may be prefetched following each buffer miss before prefetching is halted. Only a single additional prefetch is initiated after each buffer hit or miss. 100 Up to four additional lines may be prefetched following each buffer miss before prefetching is halted. Only a single additional prefetch is initiated after each buffer hit or miss. 101 Up to five additional lines may be prefetched following each buffer miss before prefetching is halted. Only a single additional prefetch is initiated after each buffer hit or miss. 110 An unlimited number of additional lines may be prefetched following each buffer miss. Only a single additional prefetch is initiated on each buffer hit or miss. 111 Reserved
31	BFEN	FBIU line read buffers enable. Enables or disables line read buffer hits. It is also used to invalidate the buffers. These bits are cleared by hardware reset. 0 The line read buffers are disabled from satisfying read requests, and all buffer valid bits are cleared. 1 The line read buffers are enabled to satisfy read requests on hits. Buffer valid bits may be set when the buffers are successfully filled. **Note:** Disable prefetching before invalidating the buffers. This includes starting a program or erase operation, or turning on and off the buffers.

[1] APC, WWSC, and RWSC values are determined by the maximum frequency of operation. See Table 13-15.

Table 13-15. FLASH_BIU Settings vs. Frequency of Operation

Maximum Frequency (MHz)	APC	RWSC	WWSC	DPFEN	IPFEN	PFLIM	BFEN
up to and including 82 MHz[1]	0b001	0b001	0b01	0b00, 0b01, or 0b11[2]	0b00, 0b01, or 0b11[2]	0b000-0b110[3]	0b0, 0b1[4]
up to and including 102 MHz[5]	0b001	0b010	0b01	0b00, 0b01, or 0b11[2]	0b00, 0b01, or 0b11[2]	0b000-0b110[3]	0b0, 0b1[4]
up to and including132 MHz[6]	0b010	0b011	0b01	0b00, 0b01, or 0b11[2]	0b00, 0b01, or 0b11[2]	0b000-0b110[3]	0b0, 0b1[4]
Default Setting after Reset	0b111	0b111	0b11	0b00	0b00	0b000	0b0

1 This setting allows for 80 MHz system clock with 2% frequency modulation.

2 For maximum flash performance, this should be set to 0b11.

3 For maximum flash performance, this should be set to 0b110.

4 For maximum flash performance, this should be set to 0b1.

5 This setting allows for 100 MHz system clock with 2% frequency modulation.

6 This setting allows for 128 MHz system clock with 2% frequency modulation.

13.3.2.9 Flash Bus Interface Unit Access Protection Register (FLASH_BIUAPR)

The FLASH_BIUAPR controls access protection for the Flash from masters on the crossbar switch. This register should only be written in a 32-bit write operation.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Reg Addr	Base (0xC3F8_8000) + 0x0020															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	1	1	1	1	1	1	M4AP		M3AP		M2AP		M1AP		M0AP	
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Reg Addr	Base (0xC3F8_8000) + 0x0020															

Figure 13-13. Flash Bus Interface Unit Access Protection Register (FLASH_BIUAPR)

Table 13-16. FLASH_BIUAPR Field Descriptions

Bits	Name	Description
0–31	M*n*AP [0:1]	Master *n* access protection. Controls whether read and write accesses to the Flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. Refer to Table 7-1. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master These fields are identified as follows: M0AP= e200z6 core M1AP= Nexus M2AP= eDMA M3AP= EBI M4AP = FEC (implemented in the MPC5553 only)

13.4 Functional Description

13.4.1 Flash Bus Interface Unit (FBIU)

The Flash BIU interfaces between the system bus and the Flash memory interface unit and generates read and write enables, the Flash array address, write size, and write data as inputs to the Flash memory interface unit (MI). The Flash BIU captures read data from the MI and drives it on the system bus. Up to two lines (1 line is a 256-bit width) of data or instructions are buffered by the Flash BIU. Lines can be prefetched in advance of being requested by the system bus interface, allowing single-cycle read data responses on buffer hits.

Several prefetch control algorithms are available for controlling line read buffer fills. Prefetch triggering can be restricted to instruction accesses only, data accesses only, or can be unrestricted. Prefetch triggering can also be controlled on a per-master basis.

Access protections can be applied on a per-master basis for both reads and writes to support security and privilege mechanisms.

13.4.1.1 FBIU Basic Interface Protocol

The Flash BIU interfaces to the Flash array by driving addresses and read or write enable signals to the Flash memory interface unit. The access time of the Flash is determined by the settings of the wait state control bits in the FLASH_BIUCR, as well as the pipelining of addresses.

The Flash BIU also has the capability of extending the normal system bus access timing by inserting additional primary (initial access) wait states for reads and burst reads. This capability is provided to allow emulation of other memories which have different access time characteristics.

13.4.1.2 FBIU Access Protections

The Flash BIU provides hardware configurable access protections for both read and write cycles from masters. It allows restriction of read and write requests on a per-master basis. The FBIU also supports software configurable access protections. Detection of a protection violation results in an error response from the Flash BIU to the system bus.

13.4.1.3 Flash Read Cycles—Buffer Miss

Read data is normally stored in the least-recently updated line read buffer in parallel with the requested data being forwarded to the system bus.

13.4.1.4 Flash Read Cycles—Buffer Hit

Single clock read responses to the system bus are possible with the Flash BIU when the requested read access is buffered.

13.4.1.5 Flash Access Pipelining

Accesses to the Flash array can be pipelined by driving a subsequent access address and control signals while waiting for the current access to complete. Pipelined access requests are always run to completion and are not aborted by the Flash BIU. Request pipelining allows for improved performance by reducing the access latency seen by the system bus master. Access pipelining can be applied to both read and write cycles by the Flash array.

13.4.1.6 Flash Error Response Operation

The Flash array can terminate a requested access with an error. This can occur due to an uncorrectable ECC error, an access control violation, or because of improper access sequencing during program/erase operations. When an error response is received, the Flash BIU will mark a line read buffer as invalid. An error response can be signaled on read or write operations.

13.4.1.7 FBIU Line Read Buffers and Prefetch Operation

The Flash BIU contains a pair of 256-bit line read buffers which are used to hold data read from the Flash array. Each buffer operates independently and is filled using a single array access. The buffers are used for both prefetch and normal demand fetches.

Prefetch triggering is controllable on a per-master and access-type basis. Bus masters can be enabled or disabled from triggering prefetches, and triggering can be further restricted based on whether a read access is for instruction or data and whether or not it is a burst access. A read access to the Flash BIU can trigger a prefetch to the next sequential line of array data on the cycle following the request. The access address is incremented to the next-higher 32-byte boundary, and A Flash array prefetch is initiated if the data is not already resident in a line read buffer. Prefetched data is loaded into the buffer which is not being used to satisfy the original request.

Buffers can be in one of six states, listed here in prioritized order:

- Invalid—the buffer contains no valid data.
- Used—the buffer contains valid data which has been provided to satisfy a burst type read.
- Valid—the buffer contains valid data which has been provided to satisfy a single type read.
- Prefetched—the buffer contains valid data which has been prefetched to satisfy a potential future access.
- Busy—the buffer is currently being used to satisfy a burst read.
- Busy fill—the buffer has been allocated to receive data from the Flash array, and the array access is still in progress.

Selection of a buffer to fill on a buffer miss is based on this prioritized order beginning with the first item (invalid). Selection of a buffer to fill on a triggered prefetch is based on the buffer which is not being used to satisfy the triggering access.

The consequences of this replacement policy are that buffers are selected for filling on a ‘least recently updated’ basis when prefetching, and on a ‘most recently emptied’ basis for demand fetches (that is, a fetch which is actually satisfying a current system bus access). This policy allows for prefetched data to remain valid when non-prefetch enabled bus masters are granted Flash access.

Several algorithms are available for prefetch control which trade off performance for power. They are described in Section 13.3.2.8, “Flash Bus Interface Unit Control Register (FLASH_BIUCR).” More aggressive prefetching increases power due to the number of wasted (discarded) prefetches, but can increase performance by lowering average read latency.

13.4.1.8 FBIU Instruction/Data Prefetch Triggering

Prefetch triggering can be enabled for instruction reads. Triggering can be enabled for all instruction reads or only for instruction burst reads. Prefetch triggering can be enabled for data reads. Triggering can be enabled for all data reads or only for data burst reads. Prefetches are not triggered by write cycles.

13.4.1.9 FBIU Per-Master Prefetch Triggering

Prefetch triggering can be controlled for individual bus masters. System bus accesses indicate the requesting master.

13.4.1.10 FBIU Buffer Invalidation

The line read buffers can be invalidated under hardware and software control. Buffers are automatically invalidated whenever the buffers are turned on or off, or at the beginning of a program or erase operation.

NOTE

Disable prefetching before invalidating the buffers. This includes starting a program or erase operation, or turning on and off the buffers.

13.4.1.11 Flash Wait-state Emulation

Emulation of other memory array timings are supported by the Flash BIU. This functionality can be useful to maintain the access timing for blocks of memory which were used to overlay Flash blocks for the purpose of system calibration or tuning during code development.

The Flash BIU will insert additional primary wait states according to user-programmable values for primary wait states. When these inputs are non-zero, additional cycles are added to system bus transfers. Normal system bus termination will be extended. In addition, no line read buffer prefetches will be initiated, and buffer hits will be ignored.

13.4.2 Flash Memory Array: User Mode

In user (normal) operating mode the Flash module can be read, written (register writes and interlock writes), programmed, or erased. The following subsections define all actions that can be performed in normal operating mode. The registers mentioned in these sections are detailed in Section 13.3.2, “Register Descriptions.”

13.4.2.1 Flash Read and Write

The default state of the Flash module is read. The main and shadow address space can be read only in the read state. The module configuration register (FLASH_MCR) is always available for read. The Flash module enters the read state on reset. The Flash module is in the read state under four sets of conditions:

- The read state is active when FLASH_MCR[STOP] = 0 (User mode read).
- The read state is active when FLASH_MCR[PGM] = 1 and/or FLASH_MCR[ERS] = 1 and high voltage operation is ongoing (Read while write).

NOTE

Reads done to the partitions being operated on (either erased or programmed) will result in an error and the FLASH_MCR[RWE] bit will be set.

- The read state is active when FLASH_MCR[PGM] = 1 and FLASH_MCR[PSUS] = 1 in the MCR. (Program suspend).
- The read state is active when FLASH_MCR[ERS] = 1 and FLASH_MCR[ESUS] = 1 and FLASH_MCR[PGM] = 0 in the MCR. (Erase suspend).

NOTE

Flash core reads are done through the BIU. In many cases the BIU will do page buffering to allow sequential reads to be done with higher performance. This can create a data coherency issue that must be handled with software. Data coherency can be an issue after a program or erase operation, as well as shadow row operations.

In Flash normal operating mode, registers can be written and the Flash array can be written to do interlock writes.

Reads attempted to invalid locations will result in indeterminate data. Invalid locations occur when addressing is done to blocks that do not exist in non 2^n array sizes.

Interlock writes attempted to invalid locations (due to blocks that do not exist in non 2^n array sizes), will result in an interlock occurring, but attempts to program or erase these blocks will not occur since they are forced to be locked.

See Section 13.3.2.2, “Low/Mid Address Space Block Locking Register (FLASH_LMLR), Section 13.3.2.3, “High Address Space Block Locking Register (FLASH_HLR)” and Section 13.3.2.4, “Secondary Low/Mid Address Space Block Locking Register (FLASH_SLMLR)” for more information.

13.4.2.2 Read While Write (RWW)

The Flash core is divided into partitions. Partitions are always comprised of two or more blocks. Partitions are used to determine read while write (RWW) groupings. While a write (program or erase) is being done within a given partition, a read can be simultaneously executed to any other partition. Partitions are listed in Table 13-4. Each partition in high address space comprises of two 128-Kbyte blocks. Note that the shadow block has unique RWW restrictions described in Section 13.4.2.5, “Flash Shadow Block.”

The Flash core is also divided into blocks to implement independent erase or program protection. The shadow block exists outside the normal address space and is programmed, erased and read independently of the other blocks. The shadow block is included to support systems that require NVM for security or system initialization information.

A software mechanism is provided to independently lock or unlock each block in high-, mid-, and low-address space against program and erase.

13.4.2.3 Flash Programming

Programming changes the value stored in an array bit from logic 1 to logic 0 only. Programming cannot change a stored logic 0 to a logic 1. Addresses in locked/disabled blocks cannot be programmed. The user can program the values in any or all of eight words within a page in a single program sequence. Word addresses are selected using bits 4:2 of the page-bound word.

Whenever a program operation occurs, ECC bits are programmed. ECC is handled on a 64-bit boundary. Thus, if only 1 word in any given 64-bit ECC segment is programmed, the adjoining word (in that segment) should not be programmed because ECC calculation has already completed for that 64-bit segment. Attempts to program the adjoining word will probably result in an operation failure. It is recommended that all programming operations be from 64 bits to 256 bits, and be 64-bit aligned. The programming operation should completely fill selected ECC segments within the page.

The program operation consists of the following sequence of events:

1. Change the value in the FLASH_MCR[PGM] bit from a 0 to a 1.

NOTE

Ensure the block that contains the address to be programmed is unlocked. See Section 13.3.2.2, “Low/Mid Address Space Block Locking Register (FLASH_LMLR), Section 13.3.2.3, “High Address Space Block Locking Register (FLASH_HLR)” and Section 13.3.2.4, “Secondary Low/Mid Address Space Block Locking Register (FLASH_SLMLR)” for more information.

2. Write the first address to be programmed in the Flash module with the program data. This write is referred to as a program data interlock write. An interlock write may be either be an aligned word or double-word.
3. If more than 1 word or double-word is to be programmed, write each additional address in the page with data to be programmed. This is referred to as a program data write. All unwritten data words default to 0xFFFF FFFF.
4. Write a logic 1 to the FLASH_MCR[EHV] bit to start the internal program sequence or skip to step 9 to terminate.
5. Wait until the FLASH_MCR[DONE] bit goes high.
6. Confirm FLASH_MCR[PEG] = 1.
7. Write a logic 0 to the FLASH_MCR[EHV] bit.
8. If more addresses are to be programmed, return to step 2.
9. Write a logic 0 to the FLASH_MCR[PGM] bit to terminate the program sequence.

The program sequence is presented graphically in Figure 13-14. The program suspend operation detailed in Figure 13-14 is discussed in Section 13.4.2.3.2, “Flash Program Suspend/Resume.”

The first write after a program is initiated determines the page address to be programmed. Program may be initiated with the 0 to 1 transition of the FLASH_MCR[PGM] bit or by clearing the FLASH_MCR[EHV] bit at the end of a previous program. This first write is referred to as an interlock write. If the program is not an erase-suspended program, the interlock write determines if the shadow or normal array space will be programmed and causes FLASH_MCR[PEAS] to be set/cleared.

In the case of an erase-suspended program, the value in FLASH_MCR[PEAS], is retained from the erase.

An interlock write must be performed before setting FLASH_MCR[EHV]. The user may terminate a program sequence by clearing FLASH_MCR[PGM] prior to setting FLASH_MCR[EHV].

If multiple writes are done to the same location the data for the last write is used in programming.

While FLASH_MCR[DONE] is low, FLASH_MCR[EHV] is high and FLASH_MCR[PSUS] is low the user may clear FLASH_MCR[EHV], resulting in a program abort. A program abort forces the module to step 8 of the program sequence. An aborted program will result in FLASH_MCR[PEG] being set low, indicating a failed operation. The data space being operated on before the abort will contain indeterminate data. The user may not abort a program sequence while in program suspend.

WARNING

Aborting a program operation will leave the Flash core addresses being programmed in an indeterminate data state. This may be recovered by executing an erase on the affected blocks.

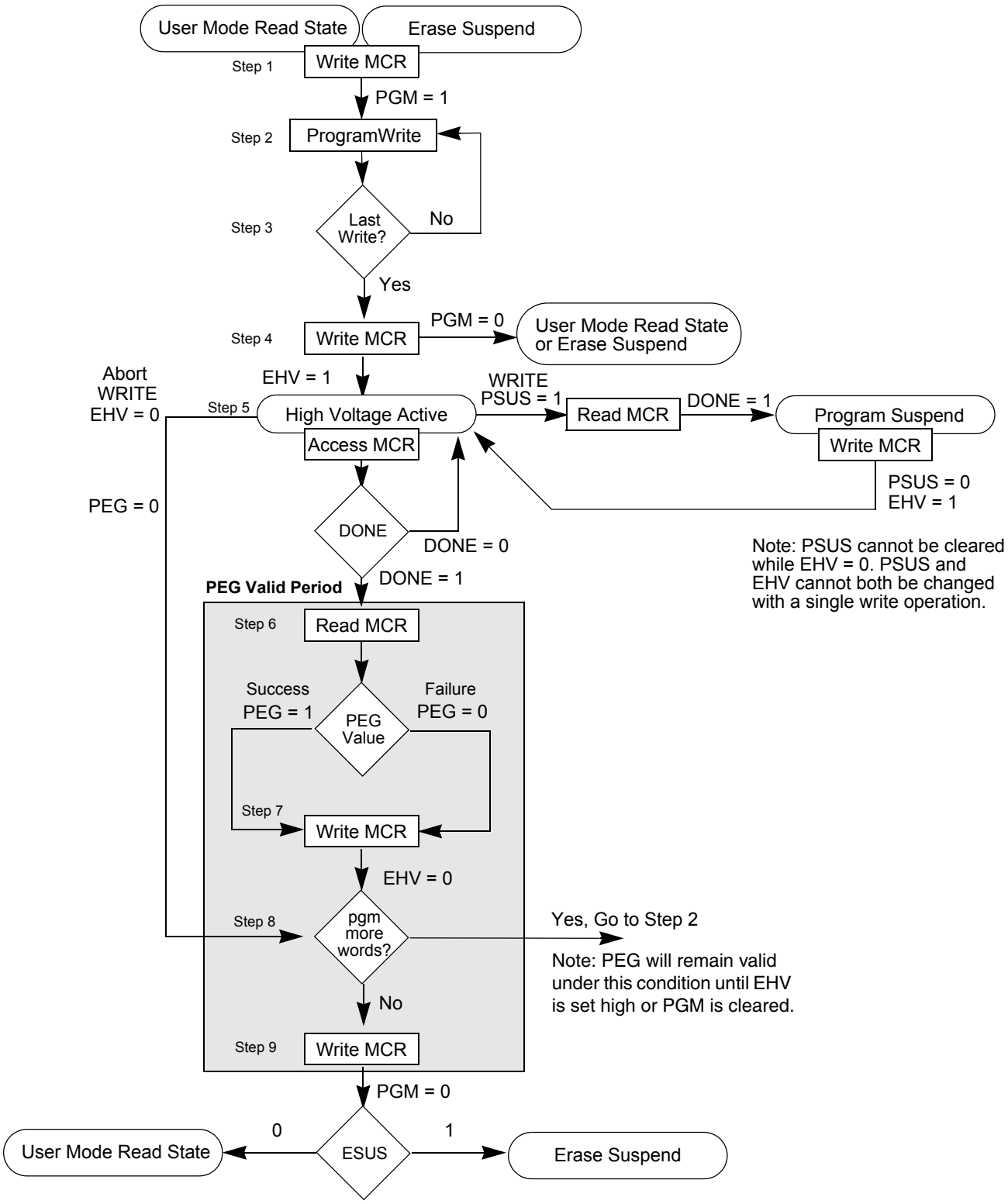


Figure 13-14. Program Sequence

13.4.2.3.1 Software Locking

A software mechanism is provided to independently lock/unlock each high, mid, and low address space against program and erase.

Software Locking is done through the FLASH_LMLR (low/mid address space block locking register), FLASH_SLMLR (secondary low/mid address space block locking register), or FLASH_HLR (high address space block locking register). These can be written through register writes, and can be read through register reads.

When the program/erase operations are enabled through hardware, software locks are enforced through doing register writes.

13.4.2.3.2 Flash Program Suspend/Resume

The program sequence may be suspended to allow read access to the Flash core. It is not possible to erase or program during a program suspend. Interlock writes should not be attempted during program suspend.

A program suspend can be initiated by changing the value of the FLASH_MCR[PSUS] bit from a 0 to a 1. FLASH_MCR[PSUS] can be set high at any time when FLASH_MCR[PGM] and FLASH_MCR[EHV] are high. A 0 to 1 transition of FLASH_MCR[PSUS] causes the Flash module to start the sequence to enter program suspend, which is a read state. The module is not suspended until FLASH_MCR[DONE] = 1. At this time Flash core reads may be attempted. Once suspended, the Flash core may only be read. Reads to the blocks being programmed/erased return indeterminate data.

The program sequence is resumed by writing a logic 0 to FLASH_MCR[PSUS]. FLASH_MCR[EHV] must be set to a 1 before clearing FLASH_MCR[PSUS] to resume operation. When the operation resumes, the Flash module continues the program sequence from one of a set of predefined points. This may extend the time required for the program operation.

13.4.2.4 Flash Erase

Erase changes the value stored in all bits of the selected blocks to logic 1. Locked or disabled blocks cannot be erased. If multiple blocks are selected for erase during an erase sequence, the blocks are erased sequentially starting with the lowest numbered block and terminating with the highest. Aborting an erase operation will leave the Flash core blocks being erased in an indeterminate data state. This can be recovered by executing an erase on the affected blocks.

The erase sequence consists of the following sequence of events:

1. Change the value in the FLASH_MCR[ERS] bit from 0 to a 1.
2. Select the block, or blocks to be erased by writing ones to the appropriate registers in FLASH_LMSR or FLASH_HSR. If the shadow row is to be erased, this step may be skipped, and FLASH_LMSR and FLASH_HSR are ignored. For shadow row erase, see section Section 13.4.2.5, “Flash Shadow Block” for more information.

NOTE

Lock and Select are independent. If a block is selected and locked, no erase will occur. See Section 13.3.2.2, “Low/Mid Address Space Block Locking Register (FLASH_LMLR), Section 13.3.2.3, “High Address Space Block Locking Register (FLASH_HLR)” and Section 13.3.2.4, “Secondary Low/Mid Address Space Block Locking Register (FLASH_SLMLR)” for more information.

3. Write to any address in Flash. This is referred to as an erase interlock write.
4. Write a logic 1 to the FLASH_MCR[EHV] bit to start an internal erase sequence or skip to step 9 to terminate.
5. Wait until the FLASH_MCR[DONE] bit goes high.
6. Confirm FLASH_MCR[PEG] = 1.
7. Write a logic 0 to the FLASH_MCR[EHV] bit.
8. If more blocks are to be erased, return to step 2.
9. Write a logic 0 to the FLASH_MCR[ERS] bit to terminate the erase.

The erase sequence is presented graphically in Figure 13-15. The erase suspend operation detailed in Figure 13-15 is discussed in section Section 13.4.2.4.1, "Flash Erase Suspend/Resume."

After settingFLASH_MCR[ERS], one write, referred to as an interlock write, must be performed before FLASH_MCR[EHV] can be set to a 1. Data words written during erase sequence interlock writes are ignored. The user may terminate the erase sequence by clearing FLASH_MCR[ERS] before setting FLASH_MCR[EHV].

An erase operation may be aborted by clearing FLASH_MCR[EHV] assuming FLASH_MCR[DONE] is low, FLASH_MCR[EHV] is high and FLASH_MCR[ESUS] is low. An erase abort forces the module to step 8 of the erase sequence. An aborted erase will result in FLASH_MCR[PEG] being set low, indicating a failed operation. The blocks being operated on before the abort contain indeterminate data. The user may not abort an erase sequence while in erase suspend.

WARNING

Aborting an erase operation will leave the Flash core blocks being erased in an indeterminate data state. This may be recovered by executing an erase on the affected blocks.

13.4.2.4.1 Flash Erase Suspend/Resume

The erase sequence may be suspended to allow read access to the Flash core. The erase sequence may also be suspended to program (erase-suspended program) the Flash core. A program started during erase suspend can in turn be suspended. Only one erase suspend and one program suspend are allowed at a time during an operation. It is not possible to erase during an erase suspend, or program during a program suspend. During suspend, all reads to Flash core locations targeted for program and blocks targeted for erase return indeterminate data. Programming locations in blocks targeted for erase during erase-suspended program may result in corrupted data.

An erase suspend operation is initiated by setting the FLASH_MCR[ESUS] bit. FLASH_MCR[ESUS] can be set to a 1 at any time when FLASH_MCR[ERS] and FLASH_MCR[EHV] are high and FLASH_MCR[PGM] is low. A 0 to 1 transition of FLASH_MCR[ESUS] causes the Flash module to start the sequence which places it in erase suspend. The user must wait until FLASH_MCR[DONE] = 1 before the module is suspended and further actions are attempted. Once suspended, the array may be read or a program sequence may be initiated (erase-suspended program). Before initiating a program sequence the user must first clear FLASH_MCR[EHV]. If a program sequence is initiated the value of the FLASH_MCR[PEAS] is not reset. These values are fixed at the time of the first interlock of the erase. Flash core reads while FLASH_MCR[ESUS] = 1 from the blocks being erased return indeterminate data.

The erase operation is resumed by clearing the FLASH_MCR[ESUS] bit. The Flash continues the erase sequence from one of a set of predefined points. This can extend the time required for the erase operation.

WARNING

In an erase-suspended program, programming Flash locations in blocks which were being operated on in the erase may corrupt Flash core data.

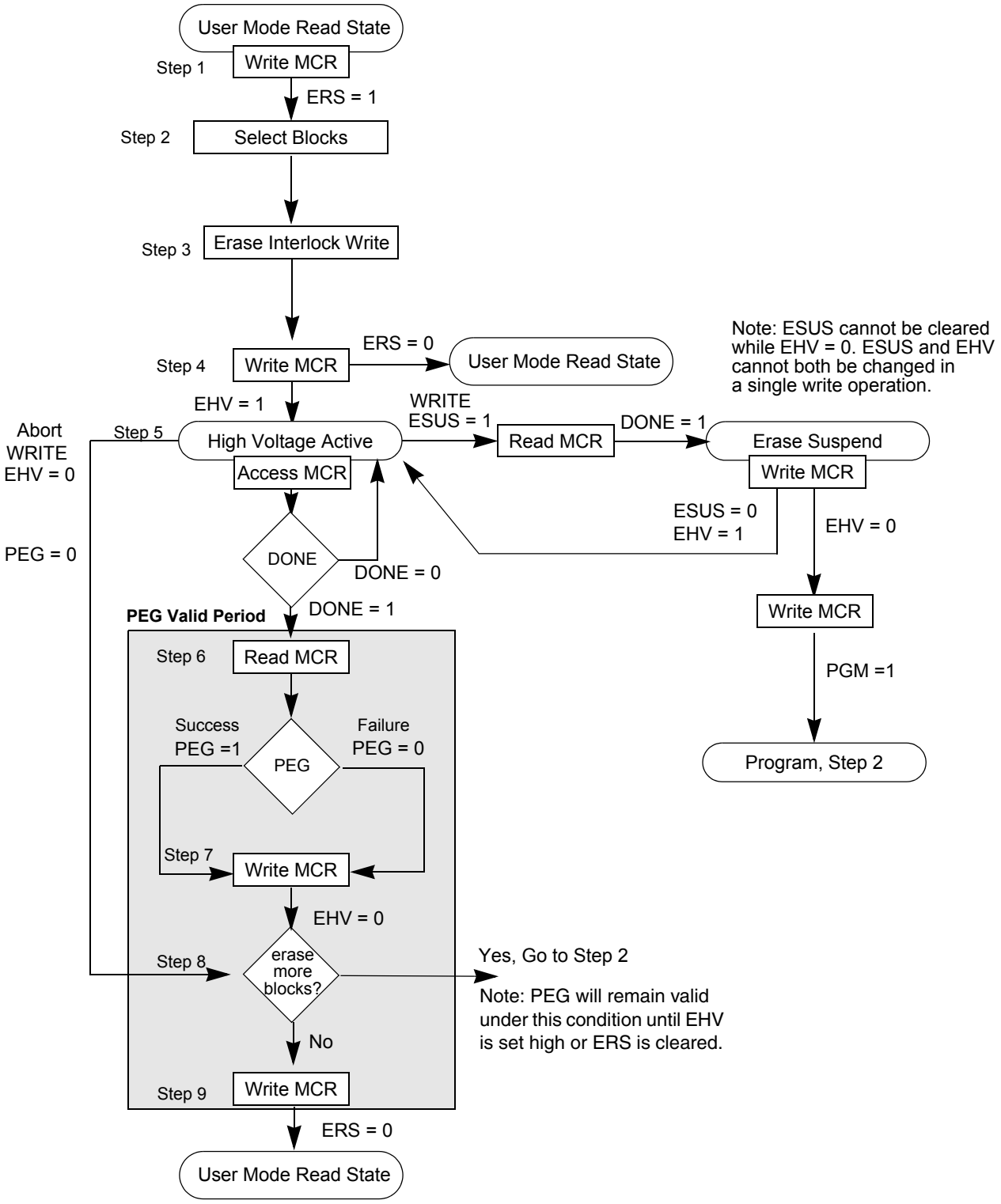


Figure 13-15. Erase Sequence

13.4.2.5 Flash Shadow Block

The Flash shadow block is a memory-mapped block in the Flash memory map. Program and erase of the shadow block are enabled only when FLASH_MCR[PEAS] = 1. Once the user has begun an erase operation on the shadow block, the operation cannot be suspended to program the main address space and vice-versa. The user must terminate the shadow erase operation to program or erase the main address space.

NOTE

If an erase of user space is requested, and a suspend is done with attempts to erase suspend program shadow space, this attempted program will be directed to user space as dictated by the state of FLASH_MCR[PEAS]. Likewise an attempted erase suspended program of user space, while the shadow space is being erased, will be directed to shadow space as dictated by the state of FLASH_MCR[PEAS].

The shadow block cannot utilize the RWW feature. Once an operation is started in the shadow block, a read cannot be done to the shadow block, or any other block Likewise, once an operation is started in a block in low/mid/high address space, a read cannot be done in the shadow block.

The shadow block contains information on how the lock registers are reset. The first and second words can be used for reset configuration words. All other words can be used for user defined functions or other configuration words.

The shadow block may be locked/unlocked against program or erase by using the FLASH_LMLR or FLASH_SLMLR discussed in Section 13.3.2, “Register Descriptions.”

Programming of the shadow row has similar restrictions to programming the array in terms of how ECC is calculated. See Section 13.4.2.3, “Flash Programming” for more information. Only one program is allowed per 64 bit ECC segment between erases. Erase of the shadow row is done similarly as an array erase. See section Section 13.4.2.4, “Flash Erase” for more information.

13.4.2.6 Censorship

Censorship logic disables access to internal Flash based on the censorship control word value and the BOOTCFG[0:1] bits in the SIU_RSR. This prevents modification of the FLASH_BIUAPR bitfields associated with all masters except the core based on the censorship control word value, the BOOTCFG[0:1] bits in the SIU_RSR, and the EXTM bit in the EBI_MCR. Also, censorship logic sets the boot default value to external-with-external-master access disabled based on the value of the censorship control word and a TCU input signal.

13.4.2.6.1 Censorship Control Word

The censorship control word is a 32-bit value located at the base address of the shadow row plus 0x1E0. The Flash module latches the value of the control word prior to the negation of system reset. Censorship logic uses the value latched in the Flash module to disable access to internal Flash, disable the NDI, prevent modification of the FLASH_BIUAPR bitfields, and/or set the boot default value .

13.4.2.6.2 Flash Disable

Censorship logic disables read and write access to internal Flash according to the logic presented in Table 13-17.

Table 13-17 shows the encoding of the BOOTCFG signals in conjunction with the value stored in the Censorship word in the shadow row of internal flash memory. The table also shows: the name of the boot mode; whether the internal flash memory is enabled or disabled; whether the Nexus port is enabled or disabled; whether the password downloaded in serial boot mode is compared with a fixed 'public' password or compared to a user programmable Flash password.

Table 13-17. Flash Access Disable Logic

BOOTCFG[1] [0:1]	Censorship Control 0x00FF_FDE0 (Upper Half)	Serial Boot Control 0x00FF_FDE2 (Lower Half)	Boot Mode Name	Internal Flash State	Nexus State[2]	Serial Password
00	!0x55AA	Don't care	Internal - Censored	Enabled	Disabled	Flash
	0x55AA		Internal - Public	Enabled	Enabled	Public
01	Don't care	0x55AA	Serial - Flash Password	Enabled	Disabled	Flash
		!0x55AA	Serial - Public Password	Disabled	Enabled	Public
10	!0x55AA	Don't care	External – No Arbitration - Censored	Disabled	Enabled	Public
	0x55AA		External – No Arbitration - Public	Enabled	Enabled	Public
11	!0x55AA	Don't care	External – External Arbitration - Censored	Disabled	Enabled	Public
	0x55AA		External – External Arbitration - Public	Enabled	Enabled	Public
'!' = 'NOT' Meaning any value other than the value specified						

1 BOOTCFG[0:1] bits are located in the SIU_RSR.

2 The Nexus port controller is held in reset when in censored mode.

The FBIU returns a bus error if an access is attempted while Flash access is disabled. Flash access is any read, write or execute access.

13.4.2.6.3 FLASH_BIUAPR Modification

Censorship logic prevents modification of the access protection register (FLASH_BIUAPR) bit fields associated with all masters except the core according to the logic presented in Table 13-18.

Table 13-18. PFBAPR Modification Logic

BOOTCFG[1]		Censorship Control Word		EXTM[2]	PFBAPR Bitfields Writable
[0]	[1]	Upper Half	Lower Half		
0	0	0x55AA	0xXXXX	0	Yes
0	0	!0x55AA	0xXXXX	0	Yes
1	0	0x55AA	0xXXXX	0	Yes
1	0	!0x55AA	0xXXXX	0	Yes
1	1	0x55AA	0xXXXX	0	Yes
1	1	!0x55AA	0xXXXX	0	Yes

Table 13-18. PFBAPR Modification Logic (continued)

BOOTCFG[1]		Censorship Control Word		EXTM[2]	PFBAPR Bitfields Writable
[0]	[1]	Upper Half	Lower Half		
0	1	0xXXXX	0x55AA	0	Yes
0	1	0xXXXX	!0x55AA	0	Yes
0	0	0x55AA	0xXXXX	1	Yes
0	0	!0x55AA	0xXXXX	1	No
1	0	0x55AA	0xXXXX	1	Yes
1	0	!0x55AA	0xXXXX	1	No
1	1	0x55AA	0xXXXX	1	Yes
1	1	!0x55AA	0xXXXX	1	No
0	1	0xXXXX	0x55AA	1	No
0	1	0xXXXX	!0x55AA	1	No

[1] BOOTCFG[0:1] bits are located in the SIU_RSR.

[2] EXTM bit is located in the EBI_MCR.

13.4.2.6.4 External Boot Default

The SIU latches the boot default value in the SIU_RSR BOOTCFG[0:1] bits if and only if $\overline{\text{RSTCFG}}$ is negated. Censorship logic sets the boot default value before the SIU latches the value to external-with-external-master access disabled (EXTM=0) if the lower half of the censorship control word equals 0xFFFF or 0x0000. Otherwise, censorship logic sets the boot default value to internal Flash.

13.4.3 Flash Memory Array: Stop Mode

Stop mode is entered by setting the FLASH_MCR[STOP] bit. The FLASH_MCR[STOP] bit cannot be written when FLASH_MCR[PGM] = 1 or FLASH_MCR[ERS] = 1. In stop mode all DC current sources in the Flash module are disabled. Stop mode is exited by clearing the FLASH_MCR[STOP] bit.

Accessing the Flash memory array when STOP is asserted results in an error response from the Flash BIU to the system bus. Memory array accesses must not be attempted until the Flash transitions out of stop mode.

13.4.4 Flash Memory Array: Reset

A reset is the highest priority operation for the Flash and terminates all other operations.

The Flash uses reset to initialize register and status bits to their default reset values. If the Flash is executing a program or erase operation and a reset is issued, the operation will be aborted and the Flash will disable the high voltage logic without damage to the high voltage circuits. Reset aborts all operations and forces the Flash into normal operating mode ready to receive accesses. FLASH_MCR[DONE] will be set to 1 at the exit of reset.

After reset is negated, register accesses can be performed, although it should be noted that registers that require updating from shadow information, or other inputs, cannot read updated values until Flash exits reset. FLASH_MCR[DONE] may be polled to determine if reset has been exited.

13.5 Revision History

Substantive Changes since Rev 3.0
Added note about disabling prefetching before invalidating the buffers. Added to BFEN bit and a note to Section 13.4.1.10, “FBIU Buffer Invalidation.”

Chapter 14
Fast Ethernet Controller (FEC)

14.1 Introduction

This fast ethernet control chapter of the MPC5553/MPC5554 Reference Manual provides a feature-set overview, a functional block diagram, and transceiver connection information for both the 10 and 100 Mbps MII (media independent interface), as well as the 7-wire serial interface. Additionally, detailed descriptions of operation and the programming model are included.

NOTE

The information in this chapter applies only to the MPC5553 device. The MPC5554 device does not have an FEC block.

14.1.1 Block Diagram

The block diagram of the FEC is shown below. The FEC is implemented with a combination of hardware and microcode. The off-chip (Ethernet) interfaces are compliant with industry and IEEE® 802.3 standards.

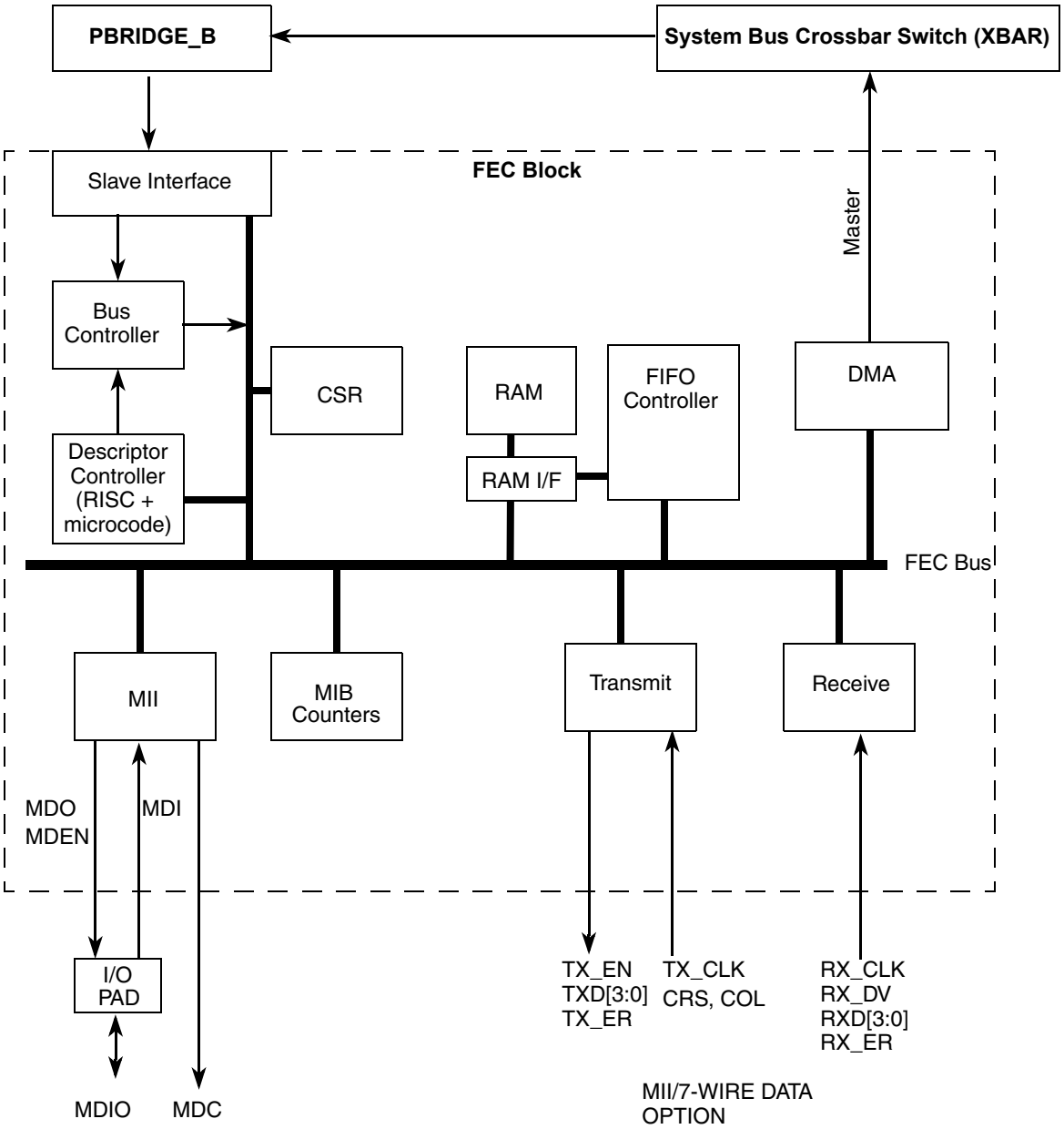


Figure 14-1. FEC Block Diagram

14.1.2 Overview

The Ethernet media access controller (MAC) is designed to support both 10 and 100 Mbps Ethernet/IEEE® 802.3 networks. An external transceiver interface and transceiver function are required to complete the interface to the media. The FEC supports three different standard MAC-PHY (physical) interfaces for connection to an external Ethernet transceiver. The FEC supports the 10/100 Mbps MII and the 10 Mbps-only 7-wire interface, which uses a subset of the MII signals.

The descriptor controller is a RISC-based controller that provides the following functions in the FEC:

- Initialization (those internal registers not initialized by the user or hardware)
- High level control of the DMA channels (initiating DMA transfers)
- Interpreting buffer descriptors
- Address recognition for receive frames
- Random number generation for transmit collision backoff timer

NOTE

DMA references in this section refer to the FEC's DMA engine. This DMA engine is for the transfer of FEC data only, and is not related to the DMA controller described in Chapter 9.

The RAM is the focal point of all data flow in the fast Ethernet controller and is divided into transmit and receive FIFOs. The FIFO boundaries are programmable using the FRSR register. User data flows to/from the DMA block from/to the receive/transmit FIFOs. Transmit data flows from the transmit FIFO into the transmit block and receive data flows from the receive block into the receive FIFO.

The user controls the FEC by writing, through the SIF (slave interface) module, into control registers located in each block. The CSR (control and status register) block provides global control (e.g. Ethernet reset and enable) and interrupt handling registers.

The MII block provides a serial channel for control/status communication with the external physical layer device (transceiver). This serial channel consists of the MDC (management data clock) and MDIO (management data input/output) lines of the MII interface.

The DMA block provides multiple channels allowing transmit data, transmit descriptor, receive data, and receive descriptor accesses to run independently.

The transmit and receive blocks provide the Ethernet MAC functionality (with some assist from microcode).

The message information block (MIB) maintains counters for a variety of network events and statistics. It is not necessary for operation of the FEC but provides valuable counters for network management. The counters supported are the RMON (RFC 1757) Ethernet Statistics groupand some of the IEEE® 802.3 counters.

14.1.3 Features

The FEC incorporates the following features:

- Support for three different Ethernet physical interfaces:
  - 100-Mbps IEEE® 802.3 MII
  - 10-Mbps IEEE® 802.3 MII
  - 10-Mbps 7-wire interface (industry standard)
- Built-in FIFO and DMA controller

- IEEE® 802.3 MAC (compliant with IEEE® 802.3 1998 edition)
- Programmable max frame length supports IEEE® 802.1 VLAN tags and priority
- IEEE® 802.3 full duplex flow control
- Support for full-duplex operation (200 Mbps throughput) with a system clock rate of 100 MHz using the external TX_CLK or RX_CLK
- Support for half-duplex operation (100 Mbps throughput) with a system clock rate of 50 MHz using the external TX_CLK or RX_CLK
- Retransmission from transmit FIFO following a collision (no system bus utilization)
- Automatic internal flushing of the receive FIFO for runts (collision fragments) and address recognition rejects (no system bus utilization)
- Address recognition
  - Frames with broadcast address may be always accepted or always rejected
  - Exact match for single 48-bit individual (unicast) address
  - Hash (64-bit hash) check of individual (unicast) addresses
  - Hash (64-bit hash) check of group (multicast) addresses
  - Promiscuous mode
- RMON and IEEE® statistics
- Interrupts for network activity and error conditions

14.2 Modes of Operation

The primary operational modes are described in this section.

14.2.1 Full and Half Duplex Operation

Full duplex mode is intended for use on point-to-point links between switches or end node to switch. Half duplex mode is used in connections between an end node and a repeater or between repeaters. Selection of the duplex mode is controlled by TCR[FDEN].

When configured for full duplex mode, flow control may be enabled. Refer to the TCR[RFC_PAUSE] and TCR[TFC_PAUSE] bits, the RCR[FCE] bit, and Section 14.4.10, “Full Duplex Flow Control,” for more details.

Throughputs of 200 Mbps in full duplex operations and 100 Mbps in half-duplex operations can be attained.

14.2.2 Interface Options

The following interface options are supported. A detailed discussion of the interface configurations is provided in Section 14.4.5, “Network Interface Options”.

14.2.2.1 10 Mbps and 100 Mbps MII Interface

MII is the media independent interface defined by the IEEE® 802.3 standard for 10/100 Mbps operation. The MAC-PHY interface may be configured to operate in MII mode by asserting RCR[MII_MODE].

The speed of operation is determined by the TX_CLK and RX_CLK signals which are driven by the external transceiver. The transceiver will either auto-negotiate the speed or it may be controlled by software via the serial management interface (MDC/MDIO signals) to the transceiver. Refer to the MMFR

and MSCR register descriptions as well as the section on the MII for a description of how to read and write registers in the transceiver via this interface.

14.2.2.2 10 Mpbs 7-Wire Interface Operation

The FEC supports a 7-wire interface as used by many 10 Mbps ethernet transceivers. The RCR[MII_MODE] bit controls this functionality. If this bit is deasserted, the MII mode is disabled and the 10 Mbps, 7-wire mode is enabled.

14.2.3 Address Recognition Options

The address options supported are promiscuous, broadcast reject, individual address (hash or exact match), and multicast hash match. Address recognition options are discussed in detail in Section 14.4.8, "Ethernet Address Recognition".

14.2.4 Internal Loopback

Internal loopback mode is selected via RCR[LOOP]. Loopback mode is discussed in detail in Section 14.4.13, "Internal and External Loopback."

14.3 Programming Model

This section gives an overview of the registers, followed by a description of the buffers.

The FEC is programmed by a combination of control/status registers (CSRs) and buffer descriptors. The CSRs are used for mode control and to extract global status information. The descriptors are used to pass data buffers and related buffer information between the hardware and software.

14.3.1 Top Level Module Memory Map

The FEC implementation requires a 1-Kbyte memory map space. This is divided into two sections of 512 bytes each. The first is used for control/status registers. The second contains event/statistic counters held in the MIB block. Table 14-1 defines the top level memory map. All accesses to and from the FEC memory map must be via 32-bit accesses. There is no support for accesses other than 32-bit.

Table 14-1. Module Memory Map

Address	Function
FFF4_C000 (Base Address) – FFF4_C1FF	Control/Status Registers
FFF4_C200 – FFF4_C3FF	MIB Block Counters

14.3.2 Detailed Memory Map (Control/Status Registers)

Table 14-2 shows the FEC register memory map with each register address, name, and a brief description. The base address of the FEC registers is 0xFFF4_C000.

NOTE

Some memory locations are not documented. The actual FEC memory map is from 0xFFF4_C000 - 0xFFF4_C5FF. Also, some bits in otherwise documented registers are not documented. These memory locations and bits are not needed for the FEC software driver. They are used mainly by the FEC subblocks for the FEC operation and happen to be visible through the slave interface.

Errant writes to these locations can corrupt FEC operation. Because the FEC is a system bus master, errant writes also can result in the corruption of any memory mapped location in the system. However, even errant writes to documented FEC memory locations can cause the same corruption.

Table 14-2. FEC Register Memory Map

Address Offset (Base +)	Name	Width[1]	Description
0x0004	EIR	32	Interrupt Event Register
0x0008	EIMR	32	Interrupt Mask Register
0x0010	RDAR	32	Receive Descriptor Active Register
0x0014	TDAR	32	Transmit Descriptor Active Register
0x0024	ECR	32	Ethernet Control Register
0x0040	MMFR	32	MII Management Frame Register
0x0044	MSCR	32	MII Speed Control Register
0x0064	MIBC	32	MIB Control/Status Register
0x0084	RCR	32	Receive Control Register
0x00C4	TCR	32	Transmit Control Register
0x00E4	PALR	32	MAC Address Low Register
0x00E8	PAUR	32	MAC Address Upper Register + Type Field
0x00EC	OPD	32	Opcode + Pause Duration Fields
0x0118	IAUR	32	Upper 32 bits of Individual Hash Table
0x011C	IALR	32	Lower 32 Bits of Individual Hash Table
0x0120	GAUR	32	Upper 32 bits of Group Hash Table
0x0124	GALR	32	Lower 32 bits of Group Hash Table
0x0144	TFWR	32	Transmit FIFO Watermark
0x014C	FRBR	32	FIFO Receive Bound Register

Table 14-2. FEC Register Memory Map (continued)

Address Offset (Base +)	Name	Width[1]	Description
0x0150	FRSR	32	FIFO Receive FIFO Start Registers
0x0180	ERDSR	32	Pointer to Receive Descriptor Ring
0x0184	ETDSR	32	Pointer to Transmit Descriptor Ring
0x0188	EMRBR	32	Maximum Receive Buffer Size

1 All accesses to and from the FEC memory map must be via 32-bit accesses. There is no support for accesses other than 32-bit.

14.3.3 MIB Block Counters Memory Map

Table 14-3 defines the MIB Counters memory map which defines the locations in the MIB RAM space where hardware-maintained counters reside. These fall in the 0xFFF4_C200 – 0xFFF4_C3FF address offset range. The counters are divided into two groups.

- RMON counters are included which cover the Ethernet Statistics counters defined in RFC 1757. In addition to the counters defined in the Ethernet Statistics group, a counter is included to count truncated frames as the FEC only supports frame lengths up to 2047 bytes. The RMON counters are implemented independently for transmit and receive to insure accurate network statistics when operating in full duplex mode.
- IEEE® counters are included which support the Mandatory and Recommended counter packages defined in section 5 of ANSI/IEEE® Std. 802.3 (1998 edition). The IEEE® Basic Package objects are supported by the FEC but do not require counters in the MIB block. In addition, some of the recommended package objects which are supported do not require MIB counters. Counters for transmit and receive full duplex flow control frames are included as well.

Table 14-3. MIB Counters Memory Map

Address Offset [1] (Base +)	Mnemonic	Description
0x0200	RMON_T_DROP	Count of frames not counted correctly
0x0204	RMON_T_PACKETS	RMON Tx packet count
0x0208	RMON_T_BC_PKT	RMON Tx Broadcast Packets
0x020C	RMON_T_MC_PKT	RMON Tx Multicast Packets
0x0210	RMON_T_CRC_ALIGN	RMON Tx Packets w CRC/Align error
0x0214	RMON_T_UNDERSIZE	RMON Tx Packets < 64 bytes, good crc
0x0218	RMON_T_OVERSIZE	RMON Tx Packets > MAX_FL bytes, good crc
0x021C	RMON_T_FRAG	RMON Tx Packets < 64 bytes, bad crc
0x0220	RMON_T_JAB	RMON Tx Packets > MAX_FL bytes, bad crc
0x0224	RMON_T_COL	RMON Tx collision count
0x0228	RMON_T_P64	RMON Tx 64 byte packets

Table 14-3. MIB Counters Memory Map (continued)

Address Offset [1] (Base +)	Mnemonic	Description
0x022C	RMON_T_P65TO127	RMON Tx 65 to 127 byte packets
0x0230	RMON_T_P128TO255	RMON Tx 128 to 255 byte packets
0x0234	RMON_T_P256TO511	RMON Tx 256 to 511 byte packets
0x0238	RMON_T_P512TO1023	RMON Tx 512 to 1023 byte packets
0x023C	RMON_T_P1024TO2047	RMON Tx 1024 to 2047 byte packets
0x0240	RMON_T_P_GTE2048	RMON Tx packets w > 2048 bytes
0x0244	RMON_T_OCTETS	RMON Tx Octets
0x0248	IEEE_T_DROP	Count of frames not counted correctly
0x024C	IEEE_T_FRAME_OK	Frames Transmitted OK
0x0250	IEEE_T_1COL	Frames Transmitted with Single Collision
0x0254	IEEE_T_MCOL	Frames Transmitted with Multiple Collisions
0x0258	IEEE_T_DEF	Frames Transmitted after Deferral Delay
0x025C	IEEE_T_LCOL	Frames Transmitted with Late Collision
0x0260	IEEE_T_EXCOL	Frames Transmitted with Excessive Collisions
0x0264	IEEE_T_MACERR	Frames Transmitted with Tx FIFO Underrun
0x0268	IEEE_T_CSERR	Frames Transmitted with Carrier Sense Error
0x026C	IEEE_T_SQE	Frames Transmitted with SQE Error
0x0270	IEEE_T_FDXFC	Flow Control Pause frames transmitted
0x0274	IEEE_T_OCTETS_OK	Octet count for Frames Transmitted w/o Error
0x0280	RMON_R_DROP	Count of frames not counted correctly
0x0284	RMON_R_PACKETS	RMON Rx packet count
0x0288	RMON_R_BC_PKT	RMON Rx Broadcast Packets
0x028C	RMON_R_MC_PKT	RMON Rx Multicast Packets
0x0290	RMON_R_CRC_ALIGN	RMON Rx Packets w CRC/Align error
0x0294	RMON_R_UNDERSIZE	RMON Rx Packets < 64 bytes, good crc
0x0298	RMON_R_OVERSIZE	RMON Rx Packets > MAX_FL bytes, good crc
0x029C	RMON_R_FRAG	RMON Rx Packets < 64 bytes, bad crc
0x02A0	RMON_R_JAB	RMON Rx Packets > MAX_FL bytes, bad crc
0x02A4	—	Reserved
0x02A8	RMON_R_P64	RMON Rx 64 byte packets
0x02AC	RMON_R_P65TO127	RMON Rx 65 to 127 byte packets

Table 14-3. MIB Counters Memory Map (continued)

Address Offset [1] (Base +)	Mnemonic	Description
0x02B0	RMON_R_P128TO255	RMON Rx 128 to 255 byte packets
0x02B4	RMON_R_P256TO511	RMON Rx 256 to 511 byte packets
0x02B8	RMON_R_P512TO1023	RMON Rx 512 to 1023 byte packets
0x02BC	RMON_R_P1024TO2047	RMON Rx 1024 to 2047 byte packets
0x02C0	RMON_R_P_GTE2048	RMON Rx packets w > 2048 bytes
0x02C4	RMON_R_OCTETS	RMON Rx Octets
0x02C8	IEEE_R_DROP	Count of frames not counted correctly
0x02CC	IEEE_R_FRAME_OK	Frames Received OK
0x02D0	IEEE_R_CRC	Frames Received with CRC Error
0x02D4	IEEE_R_ALIGN	Frames Received with Alignment Error
0x02D8	IEEE_R_MACERR	Receive Fifo Overflow count
0x02DC	IEEE_R_FDXFC	Flow Control Pause frames received
0x02E0	IEEE_R_OCTETS_OK	Octet count for Frames Rcvd w/o Error

[1] All accesses to and from the FEC memory map must be via 32-bit accesses. There is no support for accesses other than 32-bit.

14.3.4 Registers

14.3.4.1 FEC Burst Optimization Master Control Register (FBOMCR) (MPC5553 Only)

Although *not* an FEC register, the FEC burst optimization master control register (FBOMCR) controls FEC burst optimization behavior on the system bus, hence it is described below. FEC registers are described in Section 14.3.4.2.1, "Ethernet Interrupt Event Register (EIR)" through Section 14.3.4.3.4, "Receive Buffer Size Register (EMRBR)."

In order to increase throughput, the FEC interface to the system bus can accumulate read requests or writes to burst those transfers on the system bus. The FBOMCR determines the XBAR ports for which this bursting is enabled, as well as whether the bursting is for reads, writes, or both. FBOMCR also controls how errors for writes are handled. The FBOMCR address is 0xFFF4_0024, which is the ECSM base address 0xFFF4_0000 plus the offset of 0x0024.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	FXS BE0	FXS BE1	FXS BE2	0	0	0	FXS BE6	FXS BE7	RBEN	WBEN	ACCERR	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Address	ECSM Base Address: 0xFFF4_0000 + 0x0024 = 0xFFF4_0024															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Address	ECSM Base Address: 0xFFF4_0000 + 0x0024 = 0xFFF4_0024															

Figure 14-2. FEC Burst Optimization Master Control Register (FBOMCR)

Table 14-4. FBOMCR Field Descriptions

Bits	Name	Description
0-7	FXSBE*n* [0:7]	FXSBE – FEC XBAR slave burst enable. FXSBE*n* enables bursting by the FEC interface to the XBAR slave port controlled by by that respective FXSBE*n* bit. If FXSBE*n* is asserted, then that XBAR slave port enabled by the bit can accept the bursts allowed by RBEN and WBEN. Otherwise, the FEC interface will not burst to the XBAR slave port controlled by that respective FXSBE*n* bit. Read bursts from that XBAR slave port are enabled by RBEN. Write bursts to that XBAR slave port are enabled by WBEN. FXSBE*n* assignments to XBAR slave ports: FXSBE0 = Flash FXSBE1 = EBI FXSBE2 = Internal SRAM FXSBE6 = Peripheral bridge A FXSBE7 = Peripheral bridge B
8	RBEN	Global read burst enable from XBAR slave port designated by FXSBE*n* 0 = Read bursting from all XBAR slave ports is disabled. 1 = Read bursting is enabled from any XBAR slave port whose FXSBE*n* bit is asserted.
9	WBEN	Global write burst enable to XBAR slave port designated by FXSBE*n* 0 = Write bursting to all XBAR slave ports is disabled. 1 = Write bursting is enabled to any XBAR slave port whose FXSBE*n* bit is asserted.
10	ACCERR	Accumulate error - This bit determines whether an error response for the first half of the write burst is accumulated to the second half of the write burst or discarded. In order to complete the burst, the FEC interface to the system bus responds by indicating that the first half of the burst completed without error before it actually writes the data so that it can fetch the second half of the write data from the FIFO. When actually written onto the system bus, the first half of the write burst can have an error. Because this half initially responded without an error to the FIFO, the error is discarded or accumulated with the error response for the second half of the burst. 0 Any error to the first half of the write burst is discarded. 1 Any actual error response to the first half of the write burst is accumulated in the second half's response. In other words, an error response to the first half will be seen in the response to the second half, even if the second half does not error.
11-31	—	Reserved, should be cleared.

14.3.4.2 FEC Registers

The following sections describe each FEC register in detail. The base address of these registers is 0xFFF4_C000.

14.3.4.2.1 Ethernet Interrupt Event Register (EIR)

When an event occurs that sets a bit in the EIR, an interrupt will be generated if the corresponding bit in the interrupt mask register (EIMR) is also set. The bit in the EIR is cleared if a one is written to that bit position; writing zero has no effect. This register is cleared upon hardware reset.

These interrupts can be divided into operational interrupts, transceiver/network error interrupts, and internal error interrupts. Interrupts which may occur in normal operation are GRA, TXF, TXB, RXF, RXB, and MII. Interrupts resulting from errors/problems detected in the network or transceiver are HBERR, BABR, BABT, LC, and RL. Interrupts resulting from internal errors are HBERR and UN.

Some of the error interrupts are independently counted in the MIB block counters. Software may choose to mask off these interrupts since these errors will be visible to network management via the MIB counters:

- HBERR - IEEE_T_SQE
- BABR - RMON_R_OVERSIZE (good CRC), RMON_R_JAB (bad CRC)
- BABT - RMON_T_OVERSIZE (good CRC), RMON_T_JAB (bad CRC)
- LATE_COL - IEEE_T_LCOL
- COL_RETRY_LIM - IEEE_T_EXCOL
- XFIFO_UN - IEEET_MACERR

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HBERR	BABR	BABT	GRA	TXF	TXB	RXF	RXB	MII	EBERR	LC	RL	UN	0	0	0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Address	Base (0xFFF4_C000) + 0x0004															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Address	Base (0xFFF4_C000) + 0x0004															

[1] "w1c" signifies the bit is cleared by writing 1 to it.

Figure 14-3. Ethernet Interrupt Event Register (EIR)

Table 14-5. EIR Field Descriptions

Bits	Name	Description
0	HBERR	Heartbeat error. This interrupt indicates that HBC is set in the TCR register and that the COL input was not asserted within the Heartbeat window following a transmission.
1	BABR	Babbling receive error. This bit indicates a frame was received with length in excess of RCR[MAX_FL] bytes.

Table 14-5. EIR Field Descriptions (continued)

Bits	Name	Description
2	BABT	Babbling transmit error. This bit indicates that the transmitted frame length has exceeded RCR[MAX_FL] bytes. This condition is usually caused by a frame that is too long being placed into the transmit data buffers. Truncation does not occur.
3	GRA	Graceful stop complete. This interrupt will be asserted for one of three reasons. Graceful stop means that the transmitter is put into a pause state after completion of the frame currently being transmitted. 1) A graceful stop, which was initiated by the setting of the TCR[GTS] bit is now complete. 2) A graceful stop, which was initiated by the setting of the TCR[TFC_PAUSE] bit is now complete. 3) A graceful stop, which was initiated by the reception of a valid full duplex flow control “pause” frame is now complete. Refer to the “Full Duplex Flow Control” section of the Functional Description chapter.
4	TXF	Transmit frame interrupt. This bit indicates that a frame has been transmitted and that the last corresponding buffer descriptor has been updated.
5	TXB	Transmit buffer interrupt. This bit indicates that a transmit buffer descriptor has been updated.
6	RXF	Receive frame interrupt. This bit indicates that a frame has been received and that the last corresponding buffer descriptor has been updated.
7	RXB	Receive buffer interrupt. This bit indicates that a receive buffer descriptor has been updated that was not the last in the frame.
8	MII	MII interrupt. This bit indicates that the MII has completed the data transfer requested.
9	EBERR	Ethernet bus error. This bit indicates that a system bus error occurred when a DMA transaction was underway. When the EBERR bit is set, ECR[ETHER_EN] will be cleared, halting frame processing by the FEC. When this occurs software will need to insure that the FIFO controller and DMA are also soft reset.
10	LC	Late collison. This bit indicates that a collision occurred beyond the collision window (slot time) in half duplex mode. The frame is truncated with a bad CRC and the remainder of the frame is discarded.
11	RL	Collision retry limit. This bit indicates that a collision occurred on each of 16 successive attempts to transmit the frame. The frame is discarded without being transmitted and transmission of the next frame will commence. Can only occur in half duplex mode.
12	UN	Transmit FIFO underrun. This bit indicates that the transmit FIFO became empty before the complete frame was transmitted. A bad CRC is appended to the frame fragment and the remainder of the frame is discarded.
13-31	—	Reserved, should be cleared.

14.3.4.2.2 Ethernet Interrupt Mask Register (EIMR)

The EIMR register controls which interrupt events are allowed to generate actual interrupts. All implemented bits in this CSR are read/write. This register is cleared upon a hardware reset. If the corresponding bits in both the EIR and EIMR registers are set, the interrupt will be signalled to the CPU. The interrupt signal will remain asserted until a 1 is written to the EIR bit (write 1 to clear) or a 0 is written to the EIMR bit.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HBERRM	BABRM	BABTM	GRAM	TXFM	TXBM	RXFM	RXBM	MIIM	EBERRM	LCM	RLM	UNM	0	0	0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Address	Base + 0x0008															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Address	Base + 0x0008															

1 "w1c" signifies the bit is cleared by writing 1 to it.

Figure 14-4. Interrupt Mask Register (EIMR)

Table 14-6. EIMR Field Descriptions

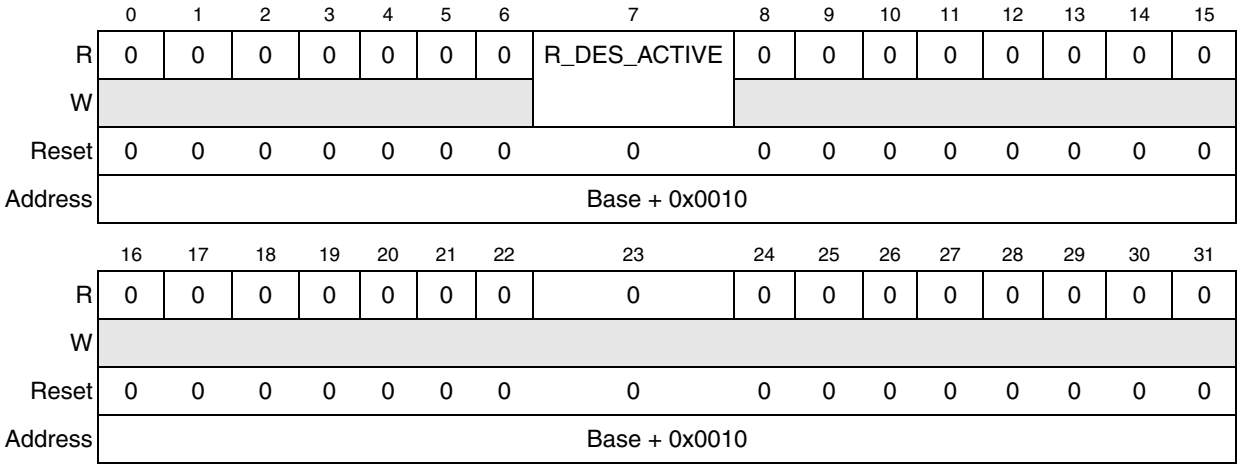
Bits	Name	Description
0-12	See Figure 17-6 and Table 14-5 .	Interrupt mask. Each bit corresponds to an interrupt source defined by the EIR register. The corresponding EIMR bit determines whether an interrupt condition can generate an interrupt. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
13-31	—	Reserved, should be cleared.

14.3.4.2.3 Receive Descriptor Active Register (RDAR)

RDAR is a command register, written by the user, that indicates that the receive descriptor ring has been updated (empty receive buffers have been produced by the driver with the empty bit set).

Whenever the register is written, the R_DES_ACTIVE bit is set. This is independent of the data actually written by the user. When set, the FEC will poll the receive descriptor ring and process receive frames (provided ECR[ETHER_EN] is also set). Once the FEC polls a receive descriptor whose empty bit is not set, then the FEC will clear R_DES_ACTIVE and cease receive descriptor ring polling until the user sets the bit again, signifying that additional descriptors have been placed into the receive descriptor ring.

The RDAR register is cleared at reset and when ECR[ETHER_EN] is cleared.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	R_DES_ACTIVE	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Address	Base + 0x0010															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Address	Base + 0x0010															

Figure 14-5. Receive Descriptor Active Register (RDAR)

Table 14-7. RDAR Field Descriptions

Bits	Name	Description
0–6	—	Reserved, should be cleared.
7	R_DES_ACTIVE	Set to one when this register is written, regardless of the value written. Cleared by the FEC device whenever no additional “empty” descriptors remain in the receive ring. Also cleared when ECR[ETHER_EN] is cleared.
8–31	—	Reserved, should be cleared.

14.3.4.2.4 Transmit Descriptor Active Register (TDAR)

The TDAR is a command register that should be written by the user to indicate that the transmit descriptor ring has been updated (transmit buffers have been produced by the driver with the ready bit set in the buffer descriptor).

Whenever the register is written, the X_DES_ACTIVE bit is set. This value is independent of the data actually written by the user. When set, the FEC will poll the transmit descriptor ring and process transmit frames (provided ECR[ETHER_EN] is also set). Once the FEC polls a transmit descriptor whose ready bit is not set, then the FEC will clear X_DES_ACTIVE and cease transmit descriptor ring polling until the user sets the bit again, signifying additional descriptors have been placed into the transmit descriptor ring.

The TDAR register is cleared at reset, when ECR[ETHER_EN] is cleared, or when ECR[RESET] is set.

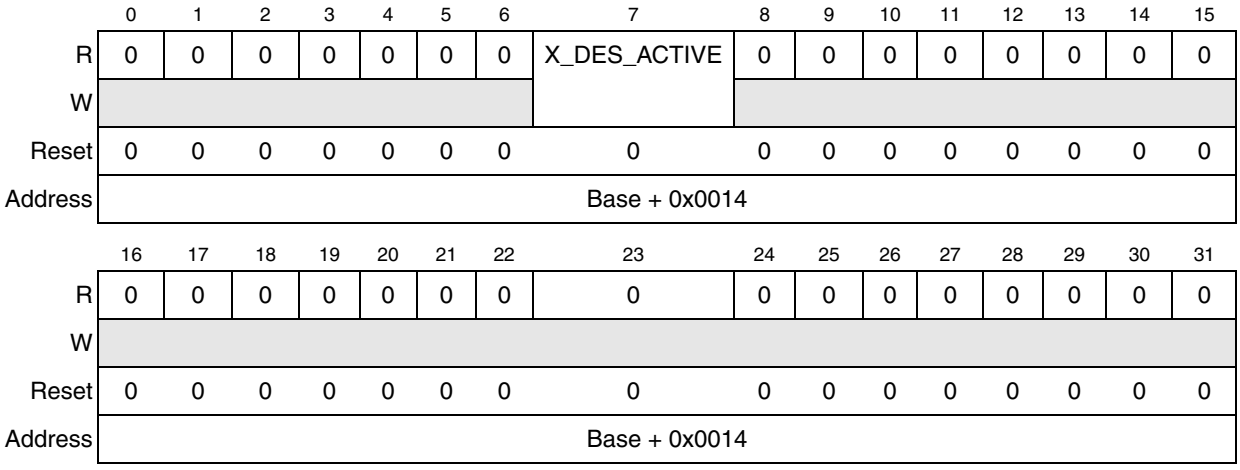


Figure 14-6. Transmit Descriptor Active Register (TDAR)

Table 14-8. TDAR Field Descriptions

Bits	Name	Description
0–6	—	Reserved, should be cleared.
7	X_DES_ACTIVE	Set to one when this register is written, regardless of the value written. Cleared by the FEC device whenever no additional “ready” descriptors remain in the transmit ring. Also cleared when ECR[ETHER_EN] is cleared.
8–31	—	Reserved, should be cleared.

14.3.4.2.5 Ethernet Control Register (ECR)

ECR is a read/write user register, though both fields in this register may be altered by hardware as well. The ECR is used to enable/disable the FEC.

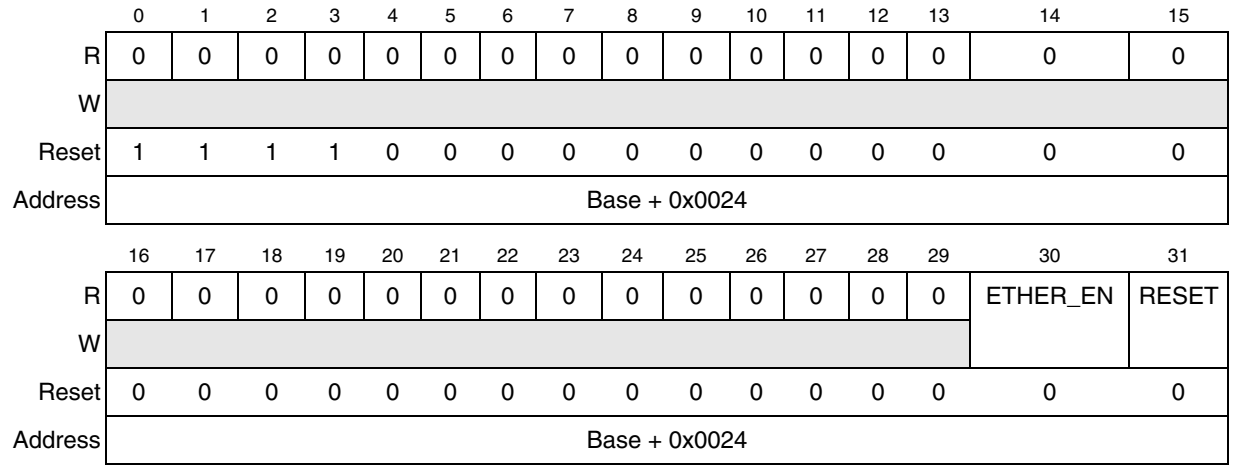


Figure 14-7. Ethernet Control Register (ECR)

Table 14-9. ECR Field Descriptions

Bits	Name	Description
0–29	—	Reserved.
30	ETHER_EN	When this bit is set, the FEC is enabled, and reception and transmission are possible. When this bit is cleared, reception is immediately stopped and transmission is stopped after a bad CRC is appended to any currently transmitted frame. The buffer descriptors for an aborted transmit frame are not updated after clearing this bit. When ETHER_EN is deasserted, the DMA, buffer descriptor, and FIFO control logic are reset, including the buffer descriptor and FIFO pointers. The ETHER_EN bit is altered by hardware under the following conditions: • ECR[RESET] is set by software, in which case ETHER_EN will be cleared • An error condition causes the EIR[EBERR] bit to set, in which case ETHER_EN will be cleared
31	RESET	When this bit is set, the equivalent of a hardware reset is performed but it is local to the FEC. ETHER_EN is cleared and all other FEC registers take their reset values. Also, any transmission/reception currently in progress is abruptly aborted. This bit is automatically cleared by hardware during the reset sequence. The reset sequence takes approximately 8 system clock cycles after RESET is written with a 1.

14.3.4.2.6 MII Management Frame Register (MMFR)

The MMFR is accessed by the user and does not reset to a defined value. The MMFR register is used to communicate with the attached MII compatible PHY devices, providing read/write access to their MII registers. Performing a write to the MMFR will cause a management frame to be sourced unless the MSCR has been programmed to 0. In the case of writing to MMFR when MSCR = 0, if the MSCR register is then written to a non-zero value, an MII frame will be generated with the data previously written to the MMFR. This allows MMFR and MSCR to be programmed in either order if MSCR is currently zero.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ST		OP		PA					RA					TA	
Reset	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U
Address	Base + 0x0040															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DATA															
Reset	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U
Address	Base + 0x0040															

[1] "U" signifies a bit that is uninitialized. Refer to the Preface of the book.

Figure 14-8. MII Management Frame Register (MMFR)

Table 14-10. MMFR Field Descriptions

Bit	Name	Description
0–1	ST	Start of frame delimiter. These bits must be programmed to 01 for a valid MII management frame.
2–3	OP	Operation code. This field must be programmed to 10 (read) or 01 (write) to generate a valid MII management frame. A value of 11 will produce “read” frame operation while a value of 00 will produce “write” frame operation, but these frames will not be MII compliant.
4–8	PA	PHY address. This field specifies one of up to 32 attached PHY devices.
9–13	RA	Register address. This field specifies one of up to 32 registers within the specified PHY device.
14–15	TA	Turn around. This field must be programmed to 10 to generate a valid MII management frame.
16–31	DATA	Management frame data. This is the field for data to be written to or read from the PHY register.

To perform a read or write operation on the MII management interface, the MMFR register must be written by the user. To generate a valid read or write management frame, the ST field must be written with a 01 pattern, and the TA field must be written with a 10. If other patterns are written to these fields, a frame will be generated but will not comply with the IEEE® 802.3 MII definition.

To generate an IEEE® 802.3-compliant MII management interface write frame (write to a PHY register), the user must write {01 01 PHYAD REGAD 10 DATA} to the MMFR register. Writing this pattern will cause the control logic to shift out the data in the MMFR register following a preamble generated by the control state machine. During this time the contents of the MMFR register will be altered as the contents are serially shifted and will be unpredictable if read by the user. Once the write management frame operation has completed, the MII interrupt will be generated. At this time the contents of the MMFR register will match the original value written.

To generate an MII management interface read frame (read a PHY register) the user must write {01 10 PHYAD REGAD 10 XXXX} to the MMFR register (the content of the DATA field is a don’t care). Writing this pattern will cause the control logic to shift out the data in the MMFR register following a preamble generated by the control state machine. During this time the contents of the MMFR register will be altered as the contents are serially shifted, and will be unpredictable if read by the user. Once the read management frame operation has completed, the MII interrupt will be generated. At this time the contents of the MMFR register will match the original value written except for the DATA field whose contents have been replaced by the value read from the PHY register.

If the MMFR register is written while frame generation is in progress, the frame contents will be altered. Software should software should poll the EIR[MII] bit or use the EIR[MII] bit to generate an interrupt to avoid writing to the MMFR register while frame generation is in progress.

14.3.4.2.7 MII Speed Control Register (MSCR)

The MSCR provides control of the MII clock (MDC signal) frequency, allows a preamble drop on the MII management frame, and provides observability (intended for manufacturing test) of an internal counter used in generating the MDC clock signal.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Address	Base + 0x0044															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	DIS_PREAMBLE	MII_SPEED						0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Address	Base + 0x0044															

Figure 14-9. MII Speed Control Register (MSCR)

Table 14-11. MSCR Field Descriptions

Bits	Name	Description
0–23	—	Reserved, should be cleared.
24	DIS_PREAMBLE	Asserting this bit will cause preamble (32 1's) not to be prepended to the MII management frame. The MII standard allows the preamble to be dropped if the attached PHY devices does not require it.
25–30	MII_SPEED	MII_SPEED controls the frequency of the MII management interface clock (MDC) relative to the system clock. A value of 0 in this field will "turn off" the MDC and leave it in low voltage state. Any non-zero value will result in the MDC frequency of 1/(MII_SPEED * 4) of the system clock frequency.
31	—	Reserved, should be cleared.

The MII_SPEED field must be programmed with a value to provide an MDC frequency of less than or equal to 2.5 MHz to be compliant with the IEEE® 802.3 MII specification. The MII_SPEED must be set to a non-zero value in order to source a read or write management frame. After the management frame is complete the MSCR register may optionally be set to zero to turn off the MDC. The MDC generated will have a 50% duty cycle except when MII_SPEED is changed during operation (change will take effect following either a rising or falling edge of MDC).

If the system clock is 50 MHz, programming this register to 0x0000_0005 will result in an MDC frequency of 50 MHz * 1/20 = 2.5 MHz. A table showing optimum values for MII_SPEED as a function of system clock frequency is provided below.

Table 14-12. Programming Examples for MSCR

System Clock Frequency	MII_SPEED (field in reg)	MDC frequency
50 MHz	0x5	2.5 MHz
66 MHz	0x7	2.36 MHz
80 MHz	0x8	2.5 MHz

Table 14-12. Programming Examples for MSCR (continued)

System Clock Frequency	MII_SPEED (field in reg)	MDC frequency
100 MHz	0xA	2.5 MHz
132 MHz	0xD	2.5 MHz

14.3.4.2.8 MIB Control Register (MIBC)

The MIBC is a read/write register used to provide control of and to observe the state of the MIB block. This register is accessed by user software if there is a need to disable the MIB block operation. For example, in order to clear all MIB counters in RAM the user should disable the MIB block, then clear all the MIB RAM locations, then enable the MIB block. The MIB_DISABLE bit is reset to 1. See Table 14-3 for the locations of the MIB counters.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MIB_DISABLE	MIB_IDLE	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Address	Base + 0x0064															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Address	Base + 0x0064															

Figure 14-10. MIB Control Register (MIBC)

Table 14-13. MIBC Field Descriptions

Bits	Name	Description
0	MIB_DISABLE	A read/write control bit. If set, the MIB logic will halt and not update any MIB counters.
1	MIB_IDLE	A read-only status bit. If set the MIB block is not currently updating any MIB counters.
2–31	—	Reserved.

14.3.4.2.9 Receive Control Register (RCR)

The RCR is programmed by the user. The RCR controls the operational mode of the receive block and should be written only when ECR[ETHER_EN] = 0 (initialization time).

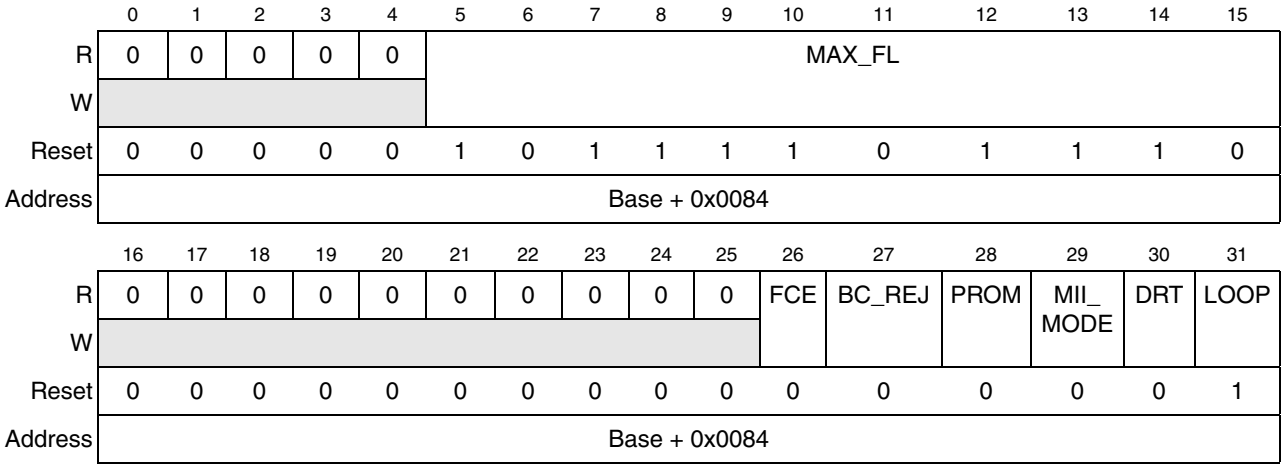


Figure 14-11. Receive Control Register (RCR)

Table 14-14. RCR Field Descriptions

Bits	Name	Description
0–4	—	Reserved, should be cleared.
5–15	MAX_FL	Maximum frame length. Resets to decimal 1518. Length is measured starting at DA and includes the CRC at the end of the frame. Transmit frames longer than MAX_FL will cause the BABT interrupt to occur. Receive frames longer than MAX_FL will cause the BABR interrupt to occur and will set the LG bit in the end of frame receive buffer descriptor. The recommended default value to be programmed by the user is 1518 or 1522 (if VLAN Tags are supported).
16–25	—	Reserved, should be cleared.
26	FCE	Flow control enable. If asserted, the receiver will detect PAUSE frames. Upon PAUSE frame detection, the transmitter will stop transmitting data frames for a given duration.
27	BC_REJ	Broadcast frame reject. If asserted, frames with DA (destination address) = FF_FF_FF_FF_FF_FF will be rejected unless the PROM bit is set. If both BC_REJ and PROM = 1, then frames with broadcast DA will be accepted and the M (MISS) bit will be set in the receive buffer descriptor.
28	PROM	Promiscuous mode. All frames are accepted regardless of address matching.
29	MII_MODE	Media independent interface mode. Selects external interface mode. Setting this bit to one selects MII mode, setting this bit equal to zero selects 7-wire mode (used only for serial 10 Mbps). This bit controls the interface mode for both transmit and receive blocks.

Table 14-14. RCR Field Descriptions (continued)

Bits	Name	Description
30	DRT	Disable receive on transmit. 0 Receive path operates independently of transmit (use for full duplex or to monitor transmit activity in half duplex mode). 1 Disable reception of frames while transmitting (normally used for half duplex mode).
31	LOOP	Internal loopback. If set, transmitted frames are looped back internal to the device and the transmit output signals are not asserted. The system clock is substituted for the TX_CLK when LOOP is asserted. DRT must be set to zero when asserting LOOP.

14.3.4.2.10 Transmit Control Register (TCR)

The TCR is read/write and is written by the user to configure the transmit block. This register is cleared at system reset. Bits 29 and 30 should be modified only when ECR[ETHER_EN] = 0.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Address	Base + 0x00C4															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	RFC_ PAUSE	TFC_ PAUSE	FDEN	HBC	GTS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Address	Base + 0x00C4															

Figure 14-12. Transmit Control Register (TCR)

Table 14-15. TCR Field Descriptions

Bits	Name	Description
0–26	—	Reserved, should be cleared.
27	RFC_PAUSE	Receive frame control pause. This read-only status bit will be asserted when a full duplex flow control pause frame has been received and the transmitter is paused for the duration defined in this pause frame. This bit will automatically clear when the pause duration is complete.

Table 14-15. TCR Field Descriptions (continued)

Bits	Name	Description
28	TFC_PAUSE	Transmit frame control pause. Transmits a PAUSE frame when asserted. When this bit is set, the MAC will stop transmission of data frames after the current transmission is complete. At this time, the GRA interrupt in the EIR register will be asserted. With transmission of data frames stopped, the MAC will transmit a MAC Control PAUSE frame. Next, the MAC will clear the TFC_PAUSE bit and resume transmitting data frames. Note that if the transmitter is paused due to user assertion of GTS or reception of a PAUSE frame, the MAC may still transmit a MAC Control PAUSE frame.
29	FDEN	Full duplex enable. If set, frames are transmitted independent of carrier sense and collision inputs. This bit should only be modified when ETHER_EN is deasserted.
30	HBC	Heartbeat control. If set, the heartbeat check is performed following end of transmission and the HB bit in the status register will be set if the collision input does not assert within the heartbeat window. This bit should only be modified when ETHER_EN is deasserted.
31	GTS	Graceful transmit stop. When this bit is set, the MAC will stop transmission after any frame that is currently being transmitted is complete and the GRA interrupt in the EIR register will be asserted. If frame transmission is not currently underway, the GRA interrupt will be asserted immediately. Once transmission has completed, a “restart” can be accomplished by clearing the GTS bit. The next frame in the transmit FIFO will then be transmitted. If an early collision occurs during transmission when GTS = 1, transmission will stop after the collision. The frame will be transmitted again once GTS is cleared. Note that there may be old frames in the transmit FIFO that will be transmitted when GTS is reasserted. To avoid this deassert ECR[ETHER_EN] following the GRA interrupt.

14.3.4.2.11 Physical Address Low Register (PALR)

The PALR is written by the user. This register contains the lower 32 bits (bytes 0,1,2,3) of the 48-bit MAC address used in the address recognition process to compare with the DA (destination address) field of receive frames with an individual DA. In addition, this register is used in bytes 0 through 3 of the 6-byte source address field when transmitting PAUSE frames. This register is not reset and must be initialized by the user.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PADDR1															
W																
Reset	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U
Address	Base + 0x00E4															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PADDR1															
W																
Reset	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U
Address	Base + 0x00E4															

1 "U" signifies a bit that is uninitialized. Refer to the Preface of the book.

Figure 14-13. Physical Address Low Register (PALR)

Table 14-16. PALR Field Descriptions

Bits	Name	Description
0–31	PADDR1	Bytes 0 (bits 0:7), 1 (bits 8:15), 2 (bits 16:23) and 3 (bits 24:31) of the 6-byte individual address to be used for exact match, and the Source Address field in PAUSE frames.

14.3.4.2.12 Physical Address Upper Register (PAUR)

The PAUR is written by the user. This register contains the upper 16 bits (bytes 4 and 5) of the 48-bit MAC address used in the address recognition process to compare with the DA (destination address) field of receive frames with an individual DA. In addition, this register is used in bytes 4 and 5 of the 6-byte Source Address field when transmitting PAUSE frames. Bits 16:31 of PAUR contain a constant TYPE field (0x8808) used for transmission of PAUSE frames.This register is not reset, and bits 0:15 must be initialized by the user. Refer to Section 14.4.10, "Full Duplex Flow Control" for information on using the TYPE field.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PADDR2															
W																
Reset	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U
Address	Base + 0x00E8															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TYPE															
W																
Reset	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0
Address	Base + 0x00E8															

1 "U" signifies a bit that is uninitialized. Refer to the Preface of the book.

Figure 14-14. Physical Address Upper Register (PAUR)

Table 14-17. PAUR Field Descriptions

Bits	Name	Description
0–15	PADDR2	Bytes 4 (bits 0:7) and 5 (bits 8:15) of the 6-byte individual address to be used for exact match, and the Source Address field in PAUSE frames.
16–31	TYPE	The type field is used in PAUSE frames. These bits are a constant, 0x8808.

14.3.4.2.13 Opcode/Pause Duration Register (OPD)

The OPD is read/write accessible. This register contains the 16-bit OPCODE and 16-bit pause duration (PAUSE_DUR) fields used in transmission of a PAUSE frame. The OPCODE field is a constant value, 0x0001. When another node detects a PAUSE frame, that node will pause transmission for the duration specified in the pause duration field. This register is not reset and must be initialized by the user. Refer to Section 14.4.10, "Full Duplex Flow Control" for information on using the OPD register.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OPCODE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Address	Base + 0x00EC															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PAUSE_DUR															
W																
Reset	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U
Address	Base + 0x00EC															

[1] "U" signifies a bit that is uninitialized. Refer to the Preface of the book.

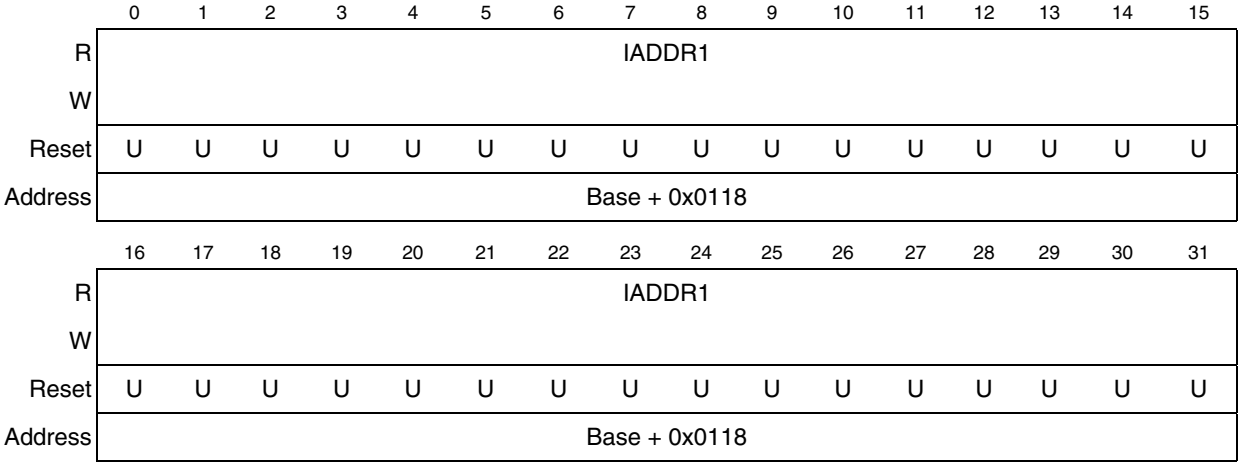
Figure 14-15. Opcode/Pause Duration Register (OPD)

Table 14-18. OPD Field Descriptions

Bits	Name	Description
0–15	OPCODE	Opcode field used in PAUSE frames. These bits are a constant, 0x0001.
16–31	PAUSE_DUR	Pause duration field used in PAUSE frames.

14.3.4.2.14 Descriptor Individual Upper Address Register (IAUR)

The IAUR is written by the user. This register contains the upper 32 bits of the 64-bit individual address hash table used in the address recognition process to check for possible match with the DA field of receive frames with an individual DA. This register is not reset and must be initialized by the user.



1 "U" signifies a bit that is uninitialized. Refer to the Preface of the book.

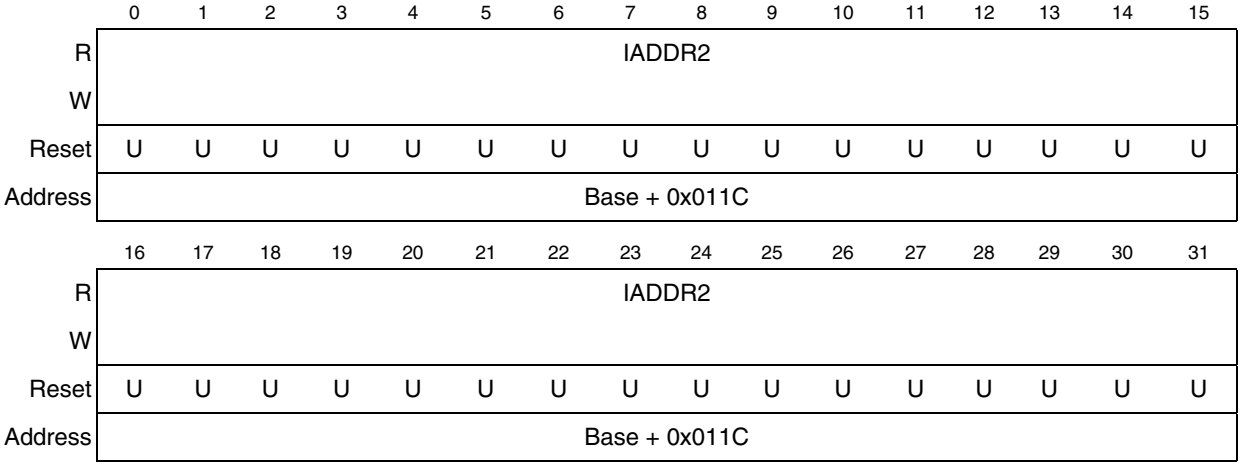
Figure 14-16. Descriptor Individual Upper Address Register (IAUR)

Table 14-19. IAUR Field Descriptions

Bits	Name	Descriptions
0–31	IADDR1	The upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a unicast address. Bit 31 of IADDR1 contains hash index bit 63. Bit 0 of IADDR1 contains hash index bit 32.

14.3.4.2.15 Descriptor Individual Lower Address (IALR)

The IALR register is written by the user. This register contains the lower 32 bits of the 64-bit individual address hash table used in the address recognition process to check for possible match with the DA field of receive frames with an individual DA. This register is not reset and must be initialized by the user.



1 "U" signifies a bit that is uninitialized. Refer to the Preface of the book.

Figure 14-17. Descriptor Individual Lower Address Register (IALR)

Table 14-20. IALR Field Descriptions

Bits	Name	Description
0–31	IADDR2	The lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a unicast address. Bit 31 of IADDR2 contains hash index bit 31. Bit 0 of IADDR2 contains hash index bit 0.

14.3.4.2.16 Descriptor Group Upper Address (GAUR)

The GAUR is written by the user. This register contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. This register must be initialized by the user.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	GADDR1															
W																
Reset	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U
Address	Base + 0x0120															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	GADDR1															
W																
Reset	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U
Address	Base + 0x0120															

1 "U" signifies a bit that is uninitialized. Refer to the Preface of the book.

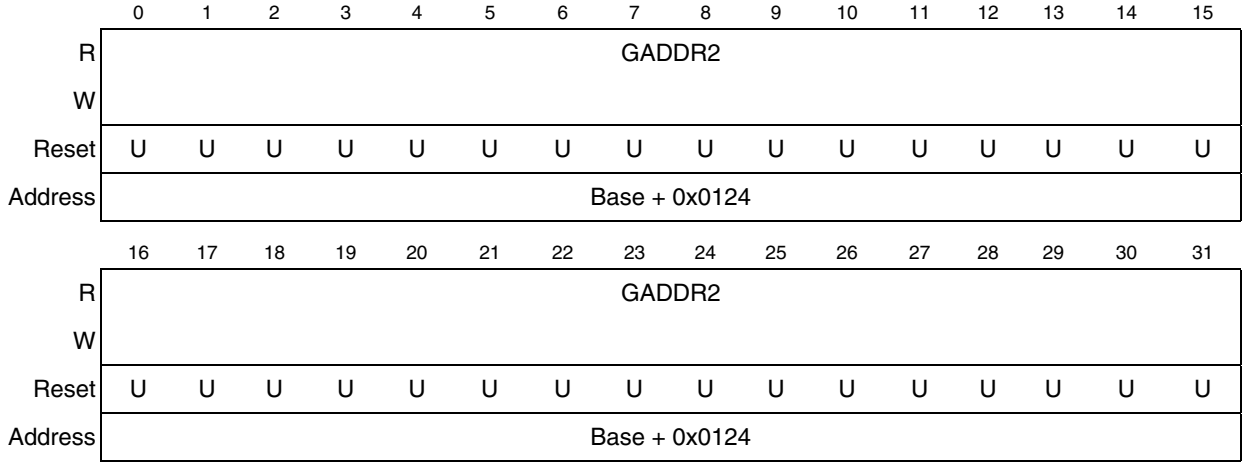
Figure 14-18. Descriptor Group Upper Address Register (GAUR)

Table 14-21. GAUR Field Descriptions

Bits	Name	Description
0–31	GADDR1	The GADDR1 register contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. Bit 31 of GADDR1 contains hash index bit 63. Bit 0 of GADDR1 contains hash index bit 32.

14.3.4.2.17 Descriptor Group Lower Address (GALR)

The GALR register is written by the user. This register contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. This register must be initialized by the user.



1 "U" signifies a bit that is uninitialized. Refer to the Preface of the book.

Figure 14-19. Descriptor Group Lower Address Register (GALR)

Table 14-22. GALR Field Descriptions

Bits	Name	Description
0–31	GADDR2	The GADDR2 register contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. Bit 31 of GADDR2 contains hash index bit 31. Bit 0 of GADDR2 contains hash index bit 0.

14.3.4.2.18 FIFO Transmit FIFO Watermark Register (TFWR)

The TFWR is a 32-bit read/write register with one 2-bit field programmed by the user to control the amount of data required in the transmit FIFO before transmission of a frame can begin. This allows the user to minimize transmit latency (TFWR = 0x) or allow for larger bus access latency (TFWR = 11) due to contention for the system bus. Setting the watermark to a high value will minimize the risk of transmit FIFO underrun due to contention for the system bus. The byte counts associated with the TFWR field may need to be modified to match a given system requirement (worst case bus access latency by the transmit data DMA channel).

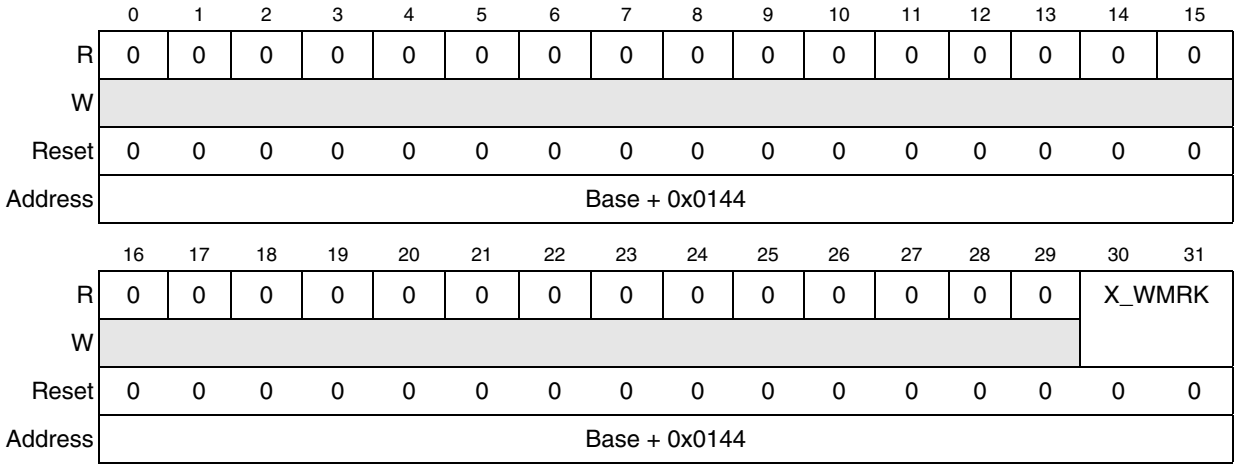


Figure 14-20. FIFO Transmit FIFO Watermark Register (TFWR)

Table 14-23. TFWR Field Descriptions

Bits	Name	Descriptions
0–29	—	Reserved, should be cleared.
30–31	X_WMRK	Number of bytes written to transmit FIFO before transmission of a frame begins 0x 64 bytes written 10 128 bytes written 11 192 bytes written

14.3.4.3 FIFO Receive Bound Register (FRBR)

The FRBR is a 32-bit register with one 8-bit field that the user can read to determine the upper address bound of the FIFO RAM. Drivers can use this value, along with the FRSR register,to appropriately divide the available FIFO RAM between the transmit and receive data paths.

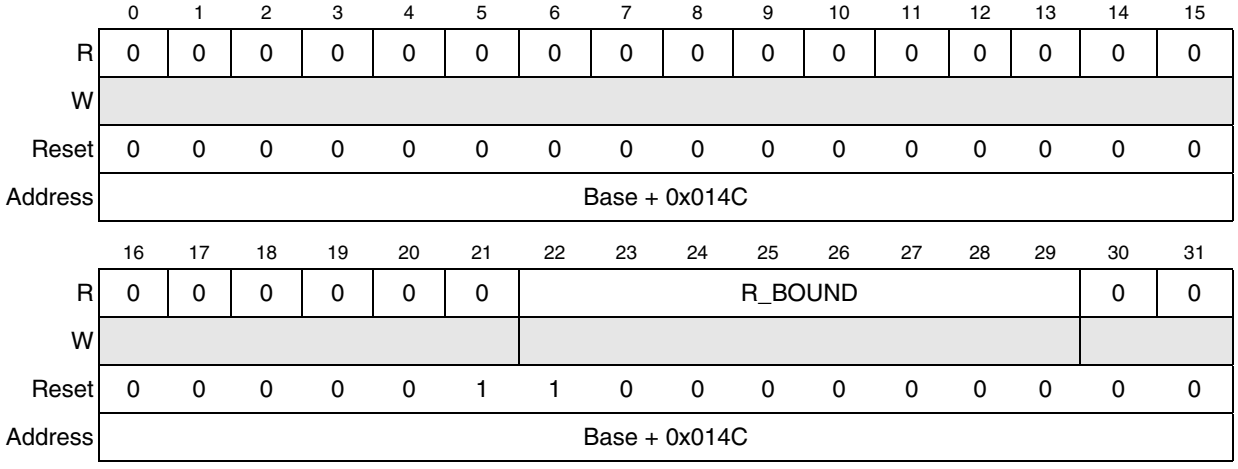


Figure 14-21. FIFO Receive Bound Register (FRBR)

Table 14-24. FRBR Field Descriptions

Bits	Name	Descriptions
0–21	—	Reserved, read as 0 (except bit 10, which is read as 1).
22–29	R_BOUND	Read-only. Highest valid FIFO RAM address.
30–31	—	Reserved, should be cleared.

14.3.4.3.1 FIFO Receive Start Register (FRSR)

The FRSR is a 32-bit register with one 8-bit field programmed by the user to indicate the starting address of the receive FIFO. FRSR marks the boundary between the transmit and receive FIFOs. The transmit FIFO uses addresses from the start of the FIFO to the location four bytes before the address programmed into the FRSR. The receive FIFO uses addresses from FRSR to FRBR inclusive.

The FRSR register is initialized by hardware at reset. FRSR only needs to be written to change the default value.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Address	Base + 0x0150															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	R_FSTART								0	0
W																
Reset	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0
Address	Base + 0x0150															

Figure 14-22. FIFO Receive Start Register (FRSR)

Table 14-25. FRSR Field Descriptions

Bits	Name	Descriptions
0–21	—	Reserved, read as 0 (except bit 10, which is read as 1).
22–29	R_FSTART	Address of first receive FIFO location. Acts as delimiter between receive and transmit FIFOs.
30–31	—	Reserved, read as 0.

14.3.4.3.2 Receive Descriptor Ring Start (ERDSR)

The ERDSR is written by the user. It provides a pointer to the start of the circular receive buffer descriptor queue in external memory. This pointer must be 32-bit aligned; however, it is recommended it be made 128-bit aligned (evenly divisible by 16).

This register is not reset and must be initialized by the user prior to operation.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	R_DES_START															
W																
Reset	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U
Address	Base + 0x0180															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	R_DES_START														0	0
W																
Reset	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U
Address	Base + 0x0180															

[1] "U" signifies a bit that is uninitialized. Refer to the Preface of the book.

Figure 14-23. Receive Descriptor Ring Start Register (ERDSR)

Table 14-26. ERDSR Field Descriptions

Bits	Name	Descriptions
0–29	R_DES_START	Pointer to start of receive buffer descriptor queue.
30–31	—	Reserved, should be cleared.

14.3.4.3.3 Transmit Buffer Descriptor Ring Start (ETDSR)

The ETDSR is written by the user. It provides a pointer to the start of the circular transmit buffer descriptor queue in external memory. This pointer must be 32-bit aligned; however, it is recommended it be made 128-bit aligned (evenly divisible by 16). Bits 30 and 31 should be written to 0 by the user. Non-zero values in these two bit positions are ignored by the hardware.

This register is not reset and must be initialized by the user prior to operation.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	X_DES_START															
W																
Reset	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U
Address	Base + 0x0184															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	X_DES_START														0	0
W																
Reset	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U
Address	Base + 0x0184															

1 "U" signifies a bit that is uninitialized. Refer to the Preface of the book.

Figure 14-24. Transmit Buffer Descriptor Ring Start Register (ETDSR)

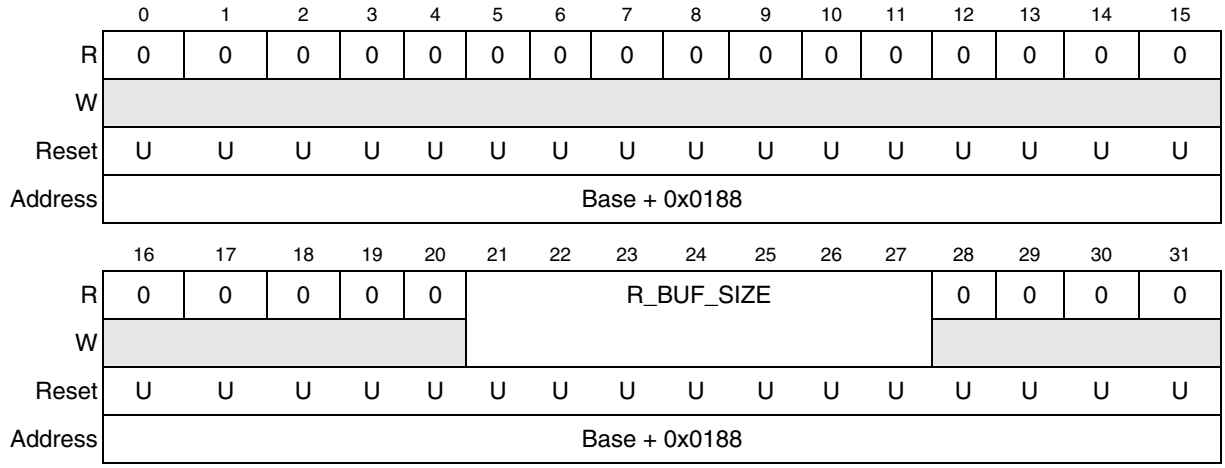
Table 14-27. ETDSR Field Descriptions

Bits	Name	Descriptions
0–29	X_DES_START	Pointer to start of transmit buffer descriptor queue.
30–31	—	Reserved, should be cleared.

14.3.4.3.4 Receive Buffer Size Register (EMRBR)

The EMRBR is a 32-bit register with one 7-bit field programmed by the user. The EMRBR register dictates the maximum size of all receive buffers. Note that because receive frames will be truncated at 2K-1 bytes, only bits 21–27 are used. This value should take into consideration that the receive CRC is always written into the last receive buffer. To allow one maximum size frame per buffer, EMRBR must be set to RCR[MAX_FL] or larger. The EMRBR must be evenly divisible by 16. To insure this, bits 28-31 are forced low. To minimize bus utilization (descriptor fetches) it is recommended that EMRBR be greater than or equal to 256 bytes.

The EMRBR register does not reset, and must be initialized by the user.



[1] "U" signifies a bit that is uninitialized. Refer to the Preface of the book.

Figure 14-25. Receive Buffer Size Register (EMRBR)

Table 14-28. EMRBR Field Descriptions

Bits	Name	Descriptions
0–20	—	Reserved, should be written to 0 by the host processor.
21–27	R_BUF_SIZE	Receive buffer size.
28–31	—	Reserved, should be written to 0 by the host processor.

14.4 Functional Description

This section describes the operation of the FEC, beginning with the hardware and software initialization sequence, then the software (Ethernet driver) interface for transmitting and receiving frames.

Following the software initialization and operation sections are sections providing a detailed description of the functions of the FEC.

14.4.1 Initialization Sequence

This section describes which registers are reset due to hardware reset, which are reset by the FEC RISC, and what locations the user must initialize prior to enabling the FEC.

14.4.1.1 Hardware Controlled Initialization

In the FEC, registers and control logic that generate interrupts are reset by hardware. A hardware reset deasserts output signals and resets general configuration bits.

Other registers reset when the ECR[ETHER_EN] bit is cleared. ECR[ETHER_EN] is deasserted by a hard reset or may be deasserted by software to halt operation. By deasserting ECR[ETHER_EN], the configuration control registers such as the TCR and RCR will not be reset, but the entire data path will be reset.

Table 14-29. ECR[ETHER_EN] De-Assertion Effect on FEC

Register/Machine	Reset Value
XMIT block	Transmission is aborted (bad CRC appended)
RECV block	Receive activity is aborted
DMA block	All DMA activity is terminated
RDAR	Cleared
TDAR	Cleared
Descriptor Controller block	Halt operation

14.4.2 User Initialization (Prior to Asserting ECR[ETHER_EN])

The user needs to initialize portions of the FEC prior to setting the ECR[ETHER_EN] bit. The exact values will depend on the particular application. The sequence is not important.

Ethernet MAC registers requiring initialization are defined in Table 14-30.

Table 14-30. User Initialization (Before ECR[ETHER_EN])

Description
Initialize EIMR
Clear EIR (write 0xFFFF_FFFF)
TFWR (optional)
IALR / IAUR
GAUR / GALR
PALR / PAUR (only needed for full duplex flow control)
OPD (only needed for full duplex flow control)
RCR
TCR
MSCR (optional)
Clear MIB_RAM (locations Base + 0x0200 – 0x02FC)

FEC FIFO/DMA registers that require initialization are defined in Table 14-31.

Table 14-31. FEC User Initialization (Before ECR[ETHER_EN])

Description
Initialize FRSR (optional)
Initialize EMRBR
Initialize ERDSR
Initialize ETDSR
Initialize (Empty) Transmit Descriptor ring
Initialize (Empty) Receive Descriptor ring

14.4.3 Microcontroller Initialization

In the FEC, the descriptor control RISC initializes some registers after ECR[ETHER_EN] is asserted. After the microcontroller initialization sequence is complete, the hardware is ready for operation.

Table 14-32 shows microcontroller initialization operations.

Table 14-32. Microcontroller Initialization

Description
Initialize BackOff Random Number Seed
Activate Receiver
Activate Transmitter
Clear Transmit FIFO
Clear Receive FIFO
Initialize Transmit Ring Pointer
Initialize Receive Ring Pointer
Initialize FIFO Count Registers

14.4.4 User Initialization (After Asserting ECR[ETHER_EN])

After asserting ECR[ETHER_EN], the user can set up the buffer/frame descriptors and write to the TDAR and RDAR. Refer to Section 14.5, "Buffer Descriptors" for more details.

14.4.5 Network Interface Options

The FEC supports both an MII interface for 10/100 Mbps Ethernet and a 7-wire serial interface for 10 Mbps Ethernet. The interface mode is selected by the RCR[MII_MODE] bit. In MII mode (RCR[MII_MODE] = 1), there are 18 signals defined by the IEEE® 802.3 standard and supported by the EMAC. These signals are shown in Table 14-33 below.

Table 14-33. MII Mode

Signal Description	EMAC Signal
Transmit Clock	TX_CLK
Transmit Enable	TX_EN
Transmit Data	TXD[3:0]
Transmit Error	TX_ER
Collision	COL
Carrier Sense	CRS
Receive Clock	RX_CLK
Receive Data Valid	RX_DV
Receive Data	RXD[3:0]
Receive Error	RX_ER

Table 14-33. MII Mode (continued)

Signal Description	EMAC Signal
Management Data Clock	MDC
Management Data Input/Output	MDIO

The 7-wire serial mode interface (RCR[MII_MODE] = 0) operates in what is generally referred to as the "AMD" mode. 7-wire mode connections to the external transceiver are shown in Table 14-34.

Table 14-34. 7-Wire Mode Configuration

Signal Description	FEC Signal
Transmit Clock	TX_CLK
Transmit Enable	TX_EN
Transmit Data	TXD0
Collision	COL
Receive Clock	RX_CLK
Receive Data Valid	RX_DV
Receive Data	RXD0

14.4.6 FEC Frame Transmission

The Ethernet transmitter is designed to work with almost no intervention from software. Once ECR[ETHER_EN] is asserted and data appears in the transmit FIFO, the Ethernet MAC is able to transmit onto the network.

When the transmit FIFO fills to the watermark (defined by the TFWR), the MAC transmit logic will assert TX_EN and start transmitting the preamble (PA) sequence, the start frame delimiter (SFD), and then the frame information from the FIFO. However, the controller defers the transmission if the network is busy (CRS asserts). Before transmitting, the controller waits for carrier sense to become inactive, then determines if carrier sense stays inactive for 60 bit times. If so, the transmission begins after waiting an additional 36 bit times (96 bit times after carrier sense originally became inactive). See Section 14.4.14.1, "Transmission Errors" for more details.

If a collision occurs during transmission of the frame (half duplex mode), the Ethernet controller follows the specified backoff procedures and attempts to retransmit the frame until the retry limit is reached. The transmit FIFO stores at least the first 64 bytes of the transmit frame, so that they do not have to be retrieved from system memory in case of a collision. This improves bus utilization and latency in case immediate retransmission is necessary.

When all the frame data has been transmitted, the FCS (frame check sequence or 32-bit cyclic redundancy check, CRC) bytes are appended if the TC bit is set in the transmit frame control word. If the ABC bit is set in the transmit frame control word, a bad CRC will be appended to the frame data regardless of the TC bit value. Following the transmission of the CRC, the Ethernet controller writes the frame status information to the MIB block. Short frames are automatically padded by the transmit logic (if the TC bit in the transmit buffer descriptor for the end of frame buffer = 1).

Both buffer (TXB) and frame (TFINT) interrupts may be generated as determined by the settings in the EIMR.

The transmit error interrupts are HBERR, BABT, LATE_COL, COL_RETRY_LIM, and XFIFO_UN. If the transmit frame length exceeds MAX_FL bytes, the BABT interrupt will be asserted but the entire frame will be transmitted (no truncation).

To pause transmission, set the GTS (graceful transmit stop) bit in the TCR register. When the TCR[GTS] is set, the FEC transmitter stops immediately if transmission is not in progress; otherwise, it continues transmission until the current frame either finishes or terminates with a collision. After the transmitter has stopped, the GRA (graceful stop complete) interrupt is asserted. If TCR[GTS] is cleared, the FEC resumes transmission with the next frame.

The Ethernet controller transmits bytes least significant bit first.

14.4.7 FEC Frame Reception

The FEC receiver is designed to work with almost no intervention from the host and can perform address recognition, CRC checking, short frame checking, and maximum frame length checking.

When the driver enables the FEC receiver by asserting ECR[ETHER_EN], it will immediately start processing receive frames. When RXDV asserts, the receiver will first check for a valid PA/SFD header. If the PA/SFD is valid, it will be stripped and the frame will be processed by the receiver. If a valid PA/SFD is not found, the frame will be ignored.

In serial mode, the first 16 bit times of RXD0 following assertion of RXDV are ignored. Following the first 16 bit times the data sequence is checked for alternating 1/0s. If a 11 or 00 data sequence is detected during bit times 17 to 21, the remainder of the frame is ignored. After bit time 21, the data sequence is monitored for a valid SFD (11). If a 00 is detected, the frame is rejected. When a 11 is detected, the PA/SFD sequence is complete.

In MII mode, the receiver checks for at least one byte matching the SFD. Zero or more PA bytes may occur, but if a 00 bit sequence is detected prior to the SFD byte, the frame is ignored.

After the first 6 bytes of the frame have been received, the FEC performs address recognition on the frame.

Once a collision window (64 bytes) of data has been received and if address recognition has not rejected the frame, the receive FIFO is signalled that the frame is “accepted” and may be passed on to the DMA. If the frame is a runt (due to collision) or is rejected by address recognition, the receive FIFO is notified to “reject” the frame. Thus, no collision fragments are presented to the user except late collisions, which indicate serious LAN problems.

During reception, the Ethernet controller checks for various error conditions and once the entire frame is written into the FIFO, a 32-bit frame status word is written into the FIFO. This status word contains the M, BC, MC, LG, NO, CR, OV and TR status bits, and the frame length. See Section 14.4.14.2, “Reception Errors” for more details.

Receive buffer (RXB) and frame interrupts (RFINT) may be generated if enabled by the EIMR register. A receive error interrupt is babbling receiver error (BABR). Receive frames are not truncated if they exceed the max frame length (MAX_FL); however, the BABR interrupt will occur and the LG bit in the receive buffer descriptor (RxBD) will be set. See Section 14.5.2, “Ethernet Receive Buffer Descriptor (RxBD)” for more details.

When the receive frame is complete, the FEC sets the L-bit in the RxBD, writes the other frame status bits into the RxBD, and clears the E-bit. The Ethernet controller next generates a maskable interrupt (RFINT bit in EIR, maskable by RFIEN bit in EIMR), indicating that a frame has been received and is in memory. The Ethernet controller then waits for a new frame.

The Ethernet controller receives serial data LSB first.

14.4.8 Ethernet Address Recognition

The FEC filters the received frames based on the type of destination address (DA) — individual (unicast), group (multicast), or broadcast (all-ones group address). The difference between an individual address and a group address is determined by the I/G bit in the destination address field. A flowchart for address recognition on received frames is illustrated in the figures below.

Address recognition is accomplished through the use of the receive block and microcode running on the microcontroller. The flowchart shown in Figure 14-26 illustrates the address recognition decisions made by the receive block, while Figure 14-27 illustrates the decisions made by the microcontroller.

If the DA is a broadcast address and broadcast reject (RCR[BC_REJ]) is deasserted, then the frame will be accepted unconditionally, as shown in Figure 14-26. Otherwise, if the DA is not a broadcast address, then the microcontroller runs the address recognition subroutine, as shown in Figure 14-27.

If the DA is a group (multicast) address and flow control is disabled, then the microcontroller will perform a group hash table lookup using the 64-entry hash table programmed in GAUR and GALR. If a hash match occurs, the receiver accepts the frame.

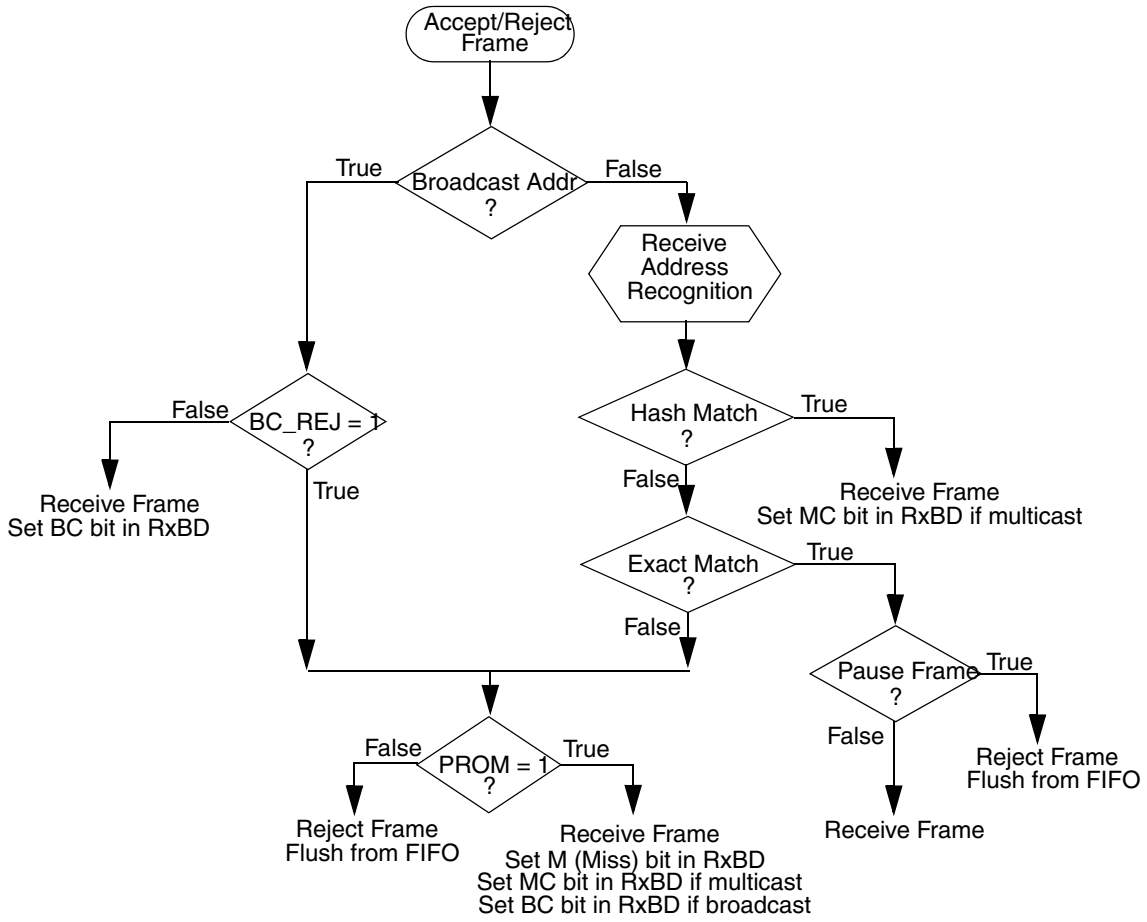
If flow control is enabled, the microcontroller will do an exact address match check between the DA and the designated PAUSE DA (01:80:C2:00:00:01). If the receive block determines that the received frame is a valid PAUSE frame, then the frame will be rejected. Note the receiver will detect a PAUSE frame with the DA field set to either the designated PAUSE DA or the unicast physical address.

If the DA is the individual (unicast) address, the microcontroller performs an individual exact match comparison between the DA and 48-bit physical address that the user programs in the PALR and PAUR registers. If an exact match occurs, the frame is accepted; otherwise, the microcontroller does an individual hash table lookup using the 64-entry hash table programmed in registers, IAUR and IALR. In the case of an individual hash match, the frame is accepted. Again, the receiver will accept or reject the frame based on PAUSE frame detection, shown in Figure 14-26.

If neither a hash match (group or individual), nor an exact match (group or individual) occur, then if promiscuous mode is enabled (RCR[PROM] = 1), then the frame will be accepted and the MISS bit in the receive buffer descriptor is set; otherwise, the frame will be rejected.

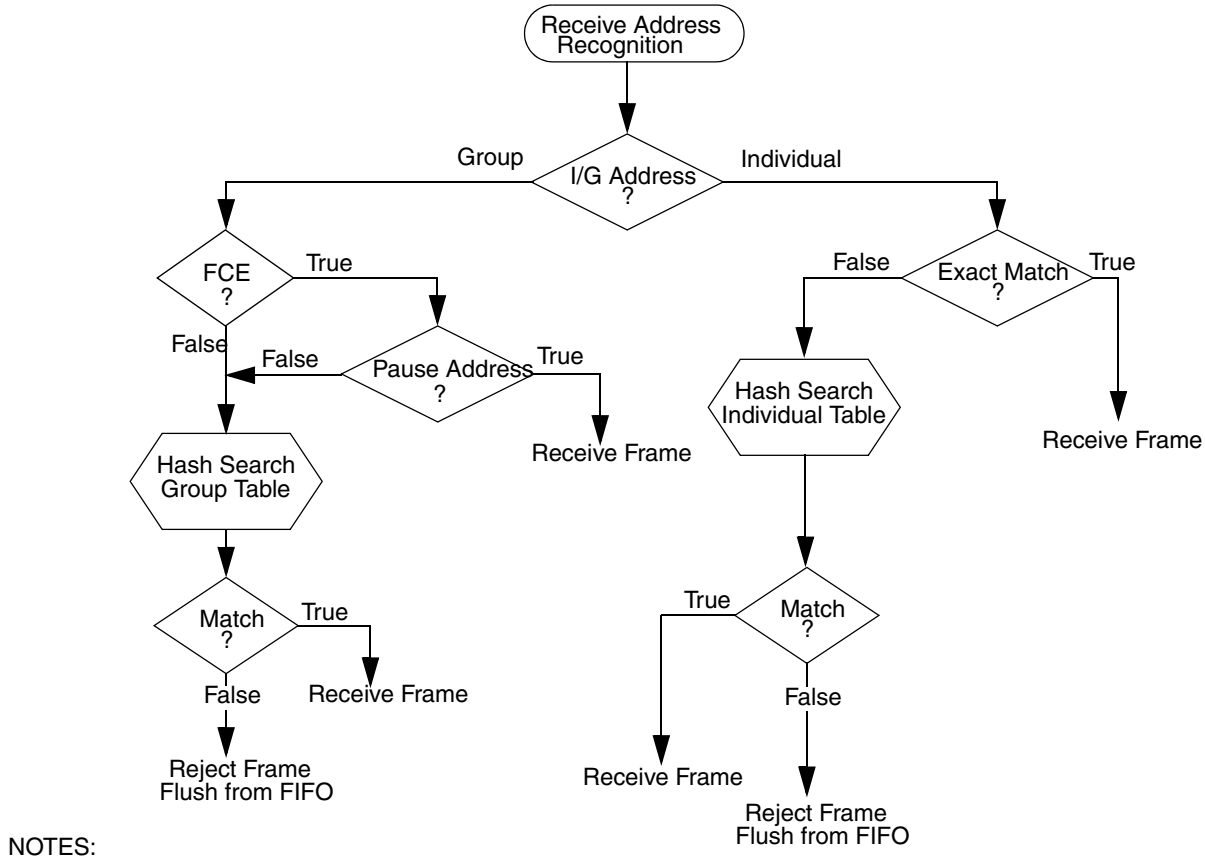
Similarly, if the DA is a broadcast address, broadcast reject (RCR[BC_REJ]) is asserted, and promiscuous mode is enabled, then the frame will be accepted and the MISS bit in the receive buffer descriptor is set; otherwise, the frame will be rejected.

In general, when a frame is rejected, it is flushed from the FIFO.



NOTES:
BC_REJ - field in RCR register (BroadCast REJect)
PROM - field in RCR register (PROMiscous mode)
Pause Frame - valid Pause frame received

Figure 14-26. Ethernet Address Recognition—Receive Block Decisions



NOTES:
FCE - field in RCR register (Flow Control Enable)
I/G - Individual/Group bit in Destination Address (least significant bit in first byte received in MAC frame)

Figure 14-27. Ethernet Address Recognition—Microcode Decisions

14.4.9 Hash Algorithm

The hash table algorithm used in the group and individual hash filtering operates as follows. The 48-bit destination address is mapped into one of 64 bits, which are represented by 64 bits stored in GAUR, GALR (group address hash match) or IAUR, IALR (individual address hash match). This mapping is performed by passing the 48-bit address through the on-chip 32-bit CRC generator and selecting the 6 most significant bits of the CRC-encoded result to generate a number between 0 and 63. The MSB of the CRC result selects GAUR (MSB = 1) or GALR (MSB = 0). The least significant 5 bits of the hash result select the bit within the selected register. If the CRC generator selects a bit that is set in the hash table, the frame is accepted; otherwise, it is rejected.

For example, if eight group addresses are stored in the hash table and random group addresses are received, the hash table prevents roughly 56/64 (or 87.5%) of the group address frames from reaching memory. Those that do reach memory must be further filtered by the processor to determine if they truly contain one of the eight desired addresses.

The effectiveness of the hash table declines as the number of addresses increases.

The hash table registers must be initialized by the user. The CRC32 polynomial to use in computing the hash is:

$$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^{8} + X^{7} + X^{5} + X^{4} + X^{2} + X + 1$$

A table of example Destination Addresses and corresponding hash values is included below for reference.

Table 14-35. Destination Address to 6-Bit Hash

48-bit DA	6-bit Hash (in hex)	Hash Decimal Value
65:ff:ff:ff:ff:ff	0x0	0
55:ff:ff:ff:ff:ff	0x1	1
15:ff:ff:ff:ff:ff	0x2	2
35:ff:ff:ff:ff:ff	0x3	3
B5:ff:ff:ff:ff:ff	0x4	4
95:ff:ff:ff:ff:ff	0x5	5
D5:ff:ff:ff:ff:ff	0x6	6
F5:ff:ff:ff:ff:ff	0x7	7
DB:ff:ff:ff:ff:ff	0x8	8
FB:ff:ff:ff:ff:ff	0x9	9
BB:ff:ff:ff:ff:ff	0xA	10
8B:ff:ff:ff:ff:ff	0xB	11
0B:ff:ff:ff:ff:ff	0xC	12
3B:ff:ff:ff:ff:ff	0xD	13
7B:ff:ff:ff:ff:ff	0xE	14
5B:ff:ff:ff:ff:ff	0xF	15
27:ff:ff:ff:ff:ff	0x10	16
07:ff:ff:ff:ff:ff	0x11	17
57:ff:ff:ff:ff:ff	0x12	18
77:ff:ff:ff:ff:ff	0x13	19
F7:ff:ff:ff:ff:ff	0x14	20
C7:ff:ff:ff:ff:ff	0x15	21
97:ff:ff:ff:ff:ff	0x16	22
A7:ff:ff:ff:ff:ff	0x17	23
99:ff:ff:ff:ff:ff	0x18	24
B9:ff:ff:ff:ff:ff	0x19	25
F9:ff:ff:ff:ff:ff	0x1A	26

Table 14-35. Destination Address to 6-Bit Hash (continued)

48-bit DA	6-bit Hash (in hex)	Hash Decimal Value
C9:ff:ff:ff:ff:ff	0x1B	27
59:ff:ff:ff:ff:ff	0x1C	28
79:ff:ff:ff:ff:ff	0x1D	29
29:ff:ff:ff:ff:ff	0x1E	30
19:ff:ff:ff:ff:ff	0x1F	31
D1:ff:ff:ff:ff:ff	0x20	32
F1:ff:ff:ff:ff:ff	0x21	33
B1:ff:ff:ff:ff:ff	0x22	34
91:ff:ff:ff:ff:ff	0x23	35
11:ff:ff:ff:ff:ff	0x24	36
31:ff:ff:ff:ff:ff	0x25	37
71:ff:ff:ff:ff:ff	0x26	38
51:ff:ff:ff:ff:ff	0x27	39
7F:ff:ff:ff:ff:ff	0x28	40
4F:ff:ff:ff:ff:ff	0x29	41
1F:ff:ff:ff:ff:ff	0x2A	42
3F:ff:ff:ff:ff:ff	0x2B	43
BF:ff:ff:ff:ff:ff	0x2C	44
9F:ff:ff:ff:ff:ff	0x2D	45
DF:ff:ff:ff:ff:ff	0x2E	46
EF:ff:ff:ff:ff:ff	0x2F	47
93:ff:ff:ff:ff:ff	0x30	48
B3:ff:ff:ff:ff:ff	0x31	49
F3:ff:ff:ff:ff:ff	0x32	50
D3:ff:ff:ff:ff:ff	0x33	51
53:ff:ff:ff:ff:ff	0x34	52
73:ff:ff:ff:ff:ff	0x35	53
23:ff:ff:ff:ff:ff	0x36	54
13:ff:ff:ff:ff:ff	0x37	55
3D:ff:ff:ff:ff:ff	0x38	56
0D:ff:ff:ff:ff:ff	0x39	57
5D:ff:ff:ff:ff:ff	0x3A	58

Table 14-35. Destination Address to 6-Bit Hash (continued)

48-bit DA	6-bit Hash (in hex)	Hash Decimal Value
7D:ff:ff:ff:ff:ff	0x3B	59
FD:ff:ff:ff:ff:ff	0x3C	60
DD:ff:ff:ff:ff:ff	0x3D	61
9D:ff:ff:ff:ff:ff	0x3E	62
BD:ff:ff:ff:ff:ff	0x3F	63

14.4.10 Full Duplex Flow Control

Full-duplex flow control allows the user to transmit pause frames and to detect received pause frames. Upon detection of a pause frame, MAC data frame transmission stops for a given pause duration.

To enable pause frame detection, the FEC must operate in full-duplex mode (TCR[FDEN] asserted) and flow control enable (RCR[FCE]) must be asserted. The FEC detects a pause frame when the fields of the incoming frame match the pause frame specifications, as shown in the table below. In addition, the receive status associated with the frame should indicate that the frame is valid.

Table 14-36. PAUSE Frame Field Specification

48-bit Destination Address	0x0180_C200_0001 or Physical Address
48-bit Source Address	Any
16-bit TYPE	0x8808
16-bit OPCODE	0x0001
16-bit PAUSE_DUR	0x0000 to 0xFFFF

Pause frame detection is performed by the receiver and microcontroller modules. The microcontroller runs an address recognition subroutine to detect the specified pause frame destination address, while the receiver detects the TYPE and OPCODE pause frame fields. On detection of a pause frame, TCR[GTS] is asserted by the FEC internally. When transmission has paused, the EIR[GRA] interrupt is asserted and the pause timer begins to increment. Note that the pause timer makes use of the transmit backoff timer hardware, which is used for tracking the appropriate collision backoff time in half-duplex mode. The pause timer increments once every slot time, until OPD[PAUSE_DUR] slot times have expired. On OPD[PAUSE_DUR] expiration, TCR[GTS] is deasserted allowing MAC data frame transmission to resume. Note that the receive flow control pause (TCR[RFC_PAUSE]) status bit is asserted while the transmitter is paused due to reception of a pause frame.

To transmit a pause frame, the FEC must operate in full-duplex mode and the user must assert flow control pause (TCR[TFC_PAUSE]). On assertion of transmit flow control pause (TCR[TFC_PAUSE]), the transmitter asserts TCR[GTS] internally. When the transmission of data frames stops, the EIR[GRA] (graceful stop complete) interrupt asserts. Following EIR[GRA] assertion, the pause frame is transmitted. On completion of pause frame transmission, flow control pause (TCR[TFC_PAUSE]) and TCR[GTS] are deasserted internally.

The user must specify the desired pause duration in the OPD register.

Note that when the transmitter is paused due to receiver/microcontroller pause frame detection, transmit flow control pause (TCR[TFC_PAUSE]) still may be asserted and will cause the transmission of a single pause frame. In this case, the EIR[GRA] interrupt will not be asserted.

14.4.11 Inter-Packet Gap (IPG) Time

The minimum inter-packet gap time for back-to-back transmission is 96 bit times. After completing a transmission or after the backoff algorithm completes, the transmitter waits for carrier sense to be negated before starting its 96 bit time IPG counter. Frame transmission may begin 96 bit times after carrier sense is negated if it stays negated for at least 60 bit times. If carrier sense asserts during the last 36 bit times, it will be ignored and a collision will occur.

The receiver receives back-to-back frames with a minimum spacing of at least 28 bit times. If an inter-packet gap between receive frames is less than 28 bit times, the following frame may be discarded by the receiver.

14.4.12 Collision Handling

If a collision occurs during frame transmission, the Ethernet controller will continue the transmission for at least 32 bit times, transmitting a JAM pattern consisting of 32 ones. If the collision occurs during the preamble sequence, the JAM pattern will be sent after the end of the preamble sequence.

If a collision occurs within 512 bit times, the retry process is initiated. The transmitter waits a random number of slot times. A slot time is 512 bit times. If a collision occurs after 512 bit times, then no retransmission is performed and the end of frame buffer is closed with a late collision (LC) error indication.

14.4.13 Internal and External Loopback

Both internal and external loopback are supported by the Ethernet controller. In loopback mode, both of the FIFOs are used and the FEC actually operates in a full-duplex fashion. Both internal and external loopback are configured using combinations of the LOOP and DRT bits in the RCR register and the FDEN bit in the TCR register.

For both internal and external loopback set FDEN = 1.

For internal loopback set RCR[LOOP] = 1 and RCR[DRT] = 0. TX_EN and TX_ER will not assert during internal loopback. During internal loopback, the transmit/receive data rate is higher than in normal operation because the internal system clock is used by the transmit and receive blocks instead of the clocks from the external transceiver. This will cause an increase in the required system bus bandwidth for transmit and receive data being DMA'd to/from external memory. It may be necessary to pace the frames on the transmit side and/or limit the size of the frames to prevent transmit FIFO underrun and receive FIFO overflow.

For external loopback set RCR[LOOP] = 0, RCR[DRT] = 0 and configure the external transceiver for loopback.

14.4.14 Ethernet Error-Handling Procedure

The Ethernet controller reports frame reception and transmission error conditions using the FEC RxBDs, the EIR register, and the MIB block counters.

14.4.14.1 Transmission Errors

14.4.14.1.1 Transmitter Underrun

If this error occurs, the FEC sends 32 bits that ensure a CRC error and stops transmitting. All remaining buffers for that frame are then flushed and closed. The UN bit is set in the EIR. The FEC will then continue to the next transmit buffer descriptor and begin transmitting the next frame.

The “UN” interrupt will be asserted if enabled in the EIMR register.

14.4.14.1.2 Retransmission Attempts Limit Expired

When this error occurs, the FEC terminates transmission. All remaining buffers for that frame are flushed and closed, and the RL bit is set in the EIR. The FEC will then continue to the next transmit buffer descriptor and begin transmitting the next frame.

The “RL” interrupt will be asserted if enabled in the EIMR register.

14.4.14.1.3 Late Collision

When a collision occurs after the slot time (512 bits starting at the preamble), the FEC terminates transmission. All remaining buffers for that frame are flushed and closed, and the LC bit is set in the EIR register. The FEC will then continue to the next transmit buffer descriptor and begin transmitting the next frame.

The “LC” interrupt will be asserted if enabled in the EIMR register.

14.4.14.1.4 Heartbeat

Some transceivers have a self-test feature called ‘heartbeat’ or ‘signal quality error.’ To signify a good self-test, the transceiver indicates a collision to the FEC within 4 microseconds after completion of a frame transmitted by the Ethernet controller. This indication of a collision does not imply a real collision error on the network, but is rather an indication that the transceiver still seems to be functioning properly. This is called the heartbeat condition.

If the HBC bit is set in the TCR register and the heartbeat condition is not detected by the FEC after a frame transmission, then a heartbeat error occurs. When this error occurs, the FEC closes the buffer, sets the HB bit in the EIR register, and generates the HBERR interrupt if it is enabled.

14.4.14.2 Reception Errors

14.4.14.2.1 Overrun Error

If the receive block has data to put into the receive FIFO and the receive FIFO is full, the FEC sets the OV bit in the RxBD. All subsequent data in the frame will be discarded and subsequent frames may also be discarded until the receive FIFO is serviced by the DMA and space is made available. At this point the receive frame/status word is written into the FIFO with the OV bit set. This frame must be discarded by the driver.

14.4.14.2.2 Non-Octet Error (Dribbling Bits)

The Ethernet controller handles up to seven dribbling bits when the receive frame terminates past an non-octet aligned boundary. Dribbling bits are not used in the CRC calculation. If there is a CRC error,

then the frame non-octet aligned (NO) error is reported in the RxBD. If there is no CRC error, then no error is reported.

14.4.14.2.3 CRC Error

When a CRC error occurs with no dribble bits, the FEC closes the buffer and sets the CR bit in the RxBD. CRC checking cannot be disabled, but the CRC error can be ignored if checking is not required.

14.4.14.2.4 Frame Length Violation

When the receive frame length exceeds MAX_FL bytes the BABR interrupt will be generated, and the LG bit in the end of frame RxBD will be set. The frame is not truncated unless the frame length exceeds 2047 bytes).

14.4.14.2.5 Truncation

When the receive frame length exceeds 2047 bytes the frame is truncated, and the TR bit is set in the RxBD.

14.5 Buffer Descriptors

This section provides a description of the operation of the driver/DMA via the buffer descriptors. It is followed by a detailed description of the receive and transmit descriptor fields.

14.5.1 Driver/DMA Operation with Buffer Descriptors

The data for the FEC frames must reside in memory external to the FEC. The data for a frame is placed in one or more buffers. Associated with each buffer is a buffer descriptor (BD) which contains a starting address (pointer), data length, and status/control information (which contains the current state for the buffer). To permit maximum user flexibility, the BDs are also located in external memory and are read in by the FEC DMA engine.

Software “produces” buffers by allocating/initializing memory and initializing buffer descriptors. Setting the RxBD[E] or TxBD[R] bit “produces” the buffer. Software writing to either the TDAR or RDAR tells the FEC that a buffer has been placed in external memory for the transmit or receive data traffic, respectively. The hardware reads the BDs and “consumes” the buffers after they have been produced. After the data DMA is complete and the buffer descriptor status bits have been written by the DMA engine, the RxBD[E] or TxBD[R] bit will be cleared by hardware to signal that the buffer has been “consumed.” Software may poll the BDs to detect when the buffers have been consumed or may rely on the buffer/frame interrupts. These buffers may then be processed by the driver and returned to the free list.

The ECR[ETHER_EN] signal operates as a reset to the BD/DMA logic. When ECR[ETHER_EN] is deasserted the DMA engine BD pointers are reset to point to the starting transmit and receive BDs. The buffer descriptors are not initialized by hardware during reset. At least one transmit and receive buffer descriptor must be initialized by software before the ECR[ETHER_EN] bit is set.

The buffer descriptors operate as two separate rings. ERDSR defines the starting address for receive BDs and ETDSR defines the starting address for transmit BDs. The last buffer descriptor in each ring is defined by the wrap (W) bit. When set, W indicates that the next descriptor in the ring is at the location pointed to by ERDSR and ETDSR for the receive and transmit rings, respectively. Buffer descriptor rings must start on a 32-bit boundary; however, it is recommended they are made 128-bit aligned.

14.5.1.1 Driver/DMA Operation with Transmit BDs

Typically a transmit frame will be divided between multiple buffers. An example is to have an application payload in one buffer, TCP header in a second buffer, IP header in a third buffer, Ethernet/IEEE® 802.3 header in a fourth buffer. The Ethernet MAC does not prepend the Ethernet header (destination address, source address, length/type fields), so this must be provided by the driver in one of the transmit buffers. The Ethernet MAC can append the Ethernet CRC to the frame. Whether the CRC is appended by the MAC or by the driver is determined by the TC bit in the transmit BD which must be set by the driver.

The driver (TxBD software producer) should set up Tx BDs in such a way that a complete transmit frame is given to the hardware at once. If a transmit frame consists of three buffers, the BDs should be initialized with pointer, length and control (W, L, TC, ABC) and then the TxBD[R] bits should be set = 1 in reverse order (3rd, 2nd, 1st BD) to insure that the complete frame is ready in memory before the DMA begins. If the TxBDs are set up in order, the DMA Controller could DMA the first BD before the 2nd was made available, potentially causing a transmit FIFO underrun.

In the FEC, the DMA is notified by the driver that new transmit frames are available by writing to the TDAR register. When this register is written to (data value is not significant) the FEC RISC will tell the DMA to read the next transmit BD in the ring. Once started, the RISC + DMA will continue to read and interpret transmit BDs in order and DMA the associated buffers, until a transmit BD is encountered with the R bit = 0. At this point the FEC will poll this BD one more time. If the R bit = 0 the second time, then the RISC will stop the transmit descriptor read process until software sets up another transmit frame and writes to TDAR.

When the DMA of each transmit buffer is complete, the DMA writes back to the BD to clear the R bit, indicating that the hardware consumer is finished with the buffer.

14.5.1.2 Driver/DMA Operation with Receive BDs

Unlike transmit, the length of the receive frame is unknown by the driver ahead of time. Therefore the driver must set a variable to define the length of all receive buffers. In the FEC, this variable is written to the EMRBR register.

The driver (RxBD software producer) should set up some number of “empty” buffers for the Ethernet by initializing the address field and the E and W bits of the associated receive BDs. The hardware (receive DMA) will consume these buffers by filling them with data as frames are received and clearing the E bit and writing to the L (1 indicates last buffer in frame) bit, the frame status bits (if L = 1) and the length field.

If a receive frame spans multiple receive buffers, the L bit is only set for the last buffer in the frame. For non-last buffers, the length field in the receive BD will be written by the DMA (at the same time the E bit is cleared) with the default receive buffer length value. For end of frame buffers the receive BD will be written with L = 1 and information written to the status bits (M, BC, MC, LG, NO, CR, OV, TR). Some of the status bits are error indicators which, if set, indicate the receive frame should be discarded and not given to higher layers. The frame status/length information is written into the receive FIFO following the end of the frame (as a single 32-bit word) by the receive logic. The length field for the end of frame buffer will be written with the length of the entire frame, not just the length of the last buffer.

For simplicity the driver may assign the default receive buffer length to be large enough to contain an entire frame, keeping in mind that a malfunction on the network or out of spec implementation could result in giant frames. Frames of 2K (2048) bytes or larger are truncated by the FEC at 2047 bytes so software is guaranteed never to see a receive frame larger than 2047 bytes.

Similar to transmit, the FEC will poll the receive descriptor ring after the driver sets up receive BDs and writes to the RDAR register. As frames are received the FEC will fill receive buffers and update the associated BDs, then read the next BD in the receive descriptor ring. If the FEC reads a receive BD and

finds the E bit = 0, it will poll this BD once more. If the BD = 0 a second time the FEC will stop reading receive BDs until the driver writes to RDAR.

14.5.2 Ethernet Receive Buffer Descriptor (RxBD)

In the RxBD, the user initializes the E and W bits in the first word and the pointer in second word. When the buffer has been DMA'd, the Ethernet controller will modify the E, L, M, BC, MC, LG, NO, CR, OV, and TR bits and write the length of the used portion of the buffer in the first word. The M, BC, MC, LG, NO, CR, OV and TR bits in the first word of the buffer descriptor are only modified by the Ethernet controller when the L bit is set.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	E	RO1	W	RO2	L	—	—	M	BC	MC	LG	NO	—	CR	OV	TR
Offset + 2	Data Length															
Offset + 4	Tx Data Buffer Pointer - A [0:15]															
Offset + 6	Tx Data Buffer Pointer - A [16:31]															

Figure 14-28. Receive Buffer Descriptor (RxBD)

Table 14-37. Receive Buffer Descriptor Field Definitions

Halfword	Location	Field Name	Description
Offset + 0	Bit 0	E	Empty. Written by the FEC (=0) and user (=1). 0 The data buffer associated with this BD has been filled with received data, or data reception has been aborted due to an error condition. The status and length fields have been updated as required. 1 The data buffer associated with this BD is empty, or reception is currently in progress.
Offset + 0	Bit 1	RO1	Receive software ownership. This field is reserved for use by software. This read/write bit will not be modified by hardware, nor will its value affect hardware.
Offset + 0	Bit 2	W	Wrap. Written by user. 0 The next buffer descriptor is found in the consecutive location 1 The next buffer descriptor is found at the location defined in ERDSR.
Offset + 0	Bit 3	RO2	Receive software ownership. This field is reserved for use by software. This read/write bit will not be modified by hardware, nor will its value affect hardware.
Offset + 0	Bit 4	L	Last in frame. Written by the FEC. 0 The buffer is not the last in a frame. 1 The buffer is the last in a frame.
Offset + 0	Bits 5-6	—	Reserved.

Table 14-37. Receive Buffer Descriptor Field Definitions (continued)

Halfword	Location	Field Name	Description
Offset + 0	Bit 7	M	Miss. Written by the FEC. This bit is set by the FEC for frames that were accepted in promiscuous mode, but were flagged as a “miss” by the internal address recognition. Thus, while in promiscuous mode, the user can use the M-bit to quickly determine whether the frame was destined to this station. This bit is valid only if the L-bit is set and the PROM bit is set. 0 The frame was received because of an address recognition hit. 1 The frame was received because of promiscuous mode.
Offset + 0	Bit 8	BC	Will be set if the DA is broadcast (FF-FF-FF-FF-FF-FF).
Offset + 0	Bit 9	MC	Will be set if the DA is multicast and not BC.
Offset + 0	Bit 10	LG	Rx frame length violation. Written by the FEC. A frame length greater than RCR[MAX_FL] was recognized. This bit is valid only if the L-bit is set. The receive data is not altered in any way unless the length exceeds 2047 bytes.
Offset + 0	Bit 11	NO	Receive non-octet aligned frame. Written by the FEC. A frame that contained a number of bits not divisible by 8 was received, and the CRC check that occurred at the preceding byte boundary generated an error. This bit is valid only if the L-bit is set. If this bit is set the CR bit will not be set.
Offset + 0	Bit 12	—	Reserved.
Offset + 0	Bit 13	CR	Receive CRC error. Written by the FEC. This frame contains a CRC error and is an integral number of octets in length. This bit is valid only if the L-bit is set.
Offset + 0	Bit 14	OV	Overrun. Written by the FEC. A receive FIFO overrun occurred during frame reception. If this bit is set, the other status bits, M, LG, NO, CR, and CL lose their normal meaning and will be zero. This bit is valid only if the L-bit is set.
Offset + 0	Bit 15	TR	Will be set if the receive frame is truncated (frame length > 2047 bytes). If the TR bit is set the frame should be discarded and the other error bits should be ignored as they may be incorrect.
Offset + 2	Bits [0:15]	Data Length	Data length. Written by the FEC. Data length is the number of 8-bit data groups (octets) written by the FEC into this BD’s data buffer if L = 0 (the value will be equal to EMRBR), or the length of the frame including CRC if L = 1. It is written by the FEC once as the BD is closed.
0ffset + 4	Bits [0:15]	A[0:15]]	RX data buffer pointer, bits [0:15][1]
Offset + 6	Bits [0:15]	A[16:31]	RX data buffer pointer, bits [16:31]

[1] The receive buffer pointer, which contains the address of the associated data buffer, must always be evenly divisible by 16. The buffer must reside in memory external to the FEC. This value is never modified by the Ethernet controller.

NOTE

Whenever the software driver sets an E bit in one or more receive descriptors, the driver should follow that with a write to RDAR.

14.5.3 Ethernet Transmit Buffer Descriptor (TxBD)

Data is presented to the FEC for transmission by arranging it in buffers referenced by the channel's TxBDs. The Ethernet controller confirms transmission by clearing the ready bit (R bit) when DMA of the buffer is complete. In the TxBD the user initializes the R, W, L, and TC bits and the length (in bytes) in the first word, and the buffer pointer in the second word.

The FEC will set the R bit = 0 in the first word of the BD when the buffer has been DMA'd. Status bits for the buffer/frame are not included in the transmit buffer descriptors. Transmit frame status is indicated via individual interrupt bits (error conditions) and in statistic counters in the MIB block. See Section 14.3.3, "MIB Block Counters Memory Map" for more details.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	R	TO1	W	TO2	L	TC	ABC	—	—	—	—	—	—	—	—	—
Offset + 2	Data Length															
Offset + 4	Tx Data Buffer Pointer - A [0:15]															
Offset + 6	Tx Data Buffer Pointer - A [16:31]															

Figure 14-29. Transmit Buffer Descriptor (TxBD)

Table 14-38. Transmit Buffer Descriptor Field Definitions

Halfword	Location	Field Name	Description
Offset + 0	Bit 0	R	Ready. Written by the FEC and the user. 0 The data buffer associated with this BD is not ready for transmission. The user is free to manipulate this BD or its associated data buffer. The FEC clears this bit after the buffer has been transmitted or after an error condition is encountered. 1 The data buffer, which has been prepared for transmission by the user, has not been transmitted or is currently being transmitted. No fields of this BD may be written by the user once this bit is set.
Offset + 0	Bit 1	TO1	Transmit software ownership. This field is reserved for software use. This read/write bit will not be modified by hardware, nor will its value affect hardware.
Offset + 0	Bit 2	W	Wrap. Written by user. 0 The next buffer descriptor is found in the consecutive location 1 The next buffer descriptor is found at the location defined in ETDSR.

Table 14-38. Transmit Buffer Descriptor Field Definitions (continued)

Halfword	Location	Field Name	Description
Offset + 0	Bit 3	TO2	Transmit software ownership. This field is reserved for use by software. This read/write bit will not be modified by hardware, nor will its value affect hardware.
Offset + 0	Bit 4	L	Last in frame. Written by user. 0 The buffer is not the last in the transmit frame. 1 The buffer is the last in the transmit frame.
Offset + 0	Bit 5	TC	Tx CRC. Written by user (only valid if L = 1). 0 End transmission immediately after the last data byte. 1 Transmit the CRC sequence after the last data byte.
Offset + 0	Bit 6	ABC	Append bad CRC. Written by user (only valid if L = 1). 0 No effect 1 Transmit the CRC sequence inverted after the last data byte (regardless of TC value).
Offset + 0	Bits [7:15]	—	Reserved.
Offset + 2	Bits [0:15]	Data Length	Data length, written by user. Data length is the number of octets the FEC should transmit from this BD's data buffer. It is never modified by the FEC. Bits [0:10] are used by the DMA engine, bits[11:15] are ignored.
Offset + 4	Bits [0:15]	A[0:15]	Tx data buffer pointer, bits [0:15][1]
Offset + 6	Bits [0:15]	A[16:31]	Tx data buffer pointer, bits [16:31].

[1] The transmit buffer pointer, which contains the address of the associated data buffer, must always be evenly divisible by 4. The buffer must reside in memory external to the FEC. This value is never modified by the Ethernet controller.

NOTE

Once the software driver has set up the buffers for a frame, it should set up the corresponding BDs. The last step in setting up the BDs for a transmit frame should be to set the R bit in the first BD for the frame. The driver should follow that with a write to TDAR which will trigger the FEC to poll the next BD in the ring.

14.6 Revision History

Substantive Changes since Rev 3.0
Table 14-2 - changed MDATA to MMFR as well as name of register to MII Management Frame Regsiter.
Changed last sentence of Section 14.3.4.2.6, “MII Management Frame Register (MMFR),” to read “Software should software should poll the EIR[MII] bit or use the EIR[MII] bit to generate an interrupt to avoid writing to the MMFR register while frame generation is in progress.”
In Section 14.3.4.3.3, “Transmit Buffer Descriptor Ring Start (ETDSR),” changed title and first sentence to ETDSR (was ETSDR).
Figure 14-8 - changed DATA to be R/W and not Read-only.
Added “All accesses to and from the FEC memory map must be via 32-bit accesses. There is no support for accesses other than 32-bit.” to Section 14.3.1, “Top Level Module Memory Map,” Table 14-2 (as a footnote), and Table 14-3 (as a footnote).
Corrected Offset + 6 range in both transmit and receive buffer descriptor to 16:31. It previously said 0:15.

Chapter 15
Internal Static RAM (SRAM)

15.1 Introduction

15.1.1 Block Diagram

The internal SRAM block diagram is shown in Figure 15-1

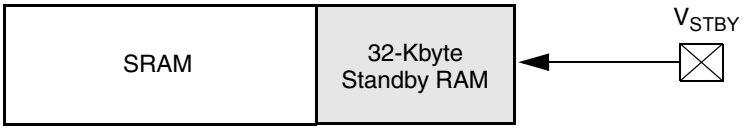


Figure 15-1. Internal SRAM Block Diagram

15.1.2 Overview

The SRAM provides 64 Kbytes of general-purpose system SRAM. The first 32-Kbyte block of the SRAM is powered by its own power supply pin for standby operation.

15.1.3 Features

The SRAM controller includes the following features:

- Supports read/write accesses mapped to the SRAM memory from any master
- 32-Kbyte block powered by separate supply for standby operation
- Byte, halfword, word, and doubleword addressable
- ECC performs single bit correction, double bit error detection

15.2 External Signal Description

The only external signal used by the SRAM is the V_{STBY} RAM power supply. If not used, V_{STBY} is tied toV_{SS}.

15.3 Memory Map/Register Definition

The SRAM occupies 64 Kbytes of address space. See Table 1-3.

Table 15-1 shows the SRAM memory map.

Table 15-1. SRAM Memory Map

Address	Register Name	Register Description	Size
Base (0x4000_0000)	—	32-Kbyte RAM, Powered by V_{STBY}	32 Kbytes
Base + 0x8000	—	32-Kbyte RAM	32 Kbytes

15.3.1 Register Descriptions

The internal SRAM has no registers. Registers associated with the SRAM ECC are located in the ECSM. See Section 8.2.1, "Register Descriptions."

15.4 Functional Description

The RAM BIU generates a 72-bit code word based upon a 64-bit data write. The ECC scheme will correct all single bit corrections, and flag all double-bit errors. Some bit errors greater than 2 bits will be flagged as multiple bit errors. The codeword of 72'b0 and 72'b1 will cause a multi-bit error. Detected multiple bit errors will assert an error indication with the bus cycle, as well as setting a flag.

15.4.1 SRAM ECC Mechanism

The ECC is calculated for each 64-bits of data. For example, for a byte write:

1. The 64-bit word (double word-aligned) is read, which causes a check of ECC on all 64-bits. If a correctable error is detected, it will be corrected prior to merging in the write data. If a non-correctable error occurs during the read portion of the write operation, then the write will not be performed.
2. The byte data is merged and the ECC is generated for the new 64-bit data value.
3. The data and ECC bits are written back.

In the case of a 64-bit write, the 64-bit word is not read for the merge operation. Instead, the ECC is generated for the 64-bit word data then both data and ECC bits are written. Because the ECC bits will contain random data after power on, the 64-bit write mechanism is used to initialize the SRAM and insure that the ECC bits are valid. See Section 15.5, "Initialization/Application Information."

15.4.2 Access Timing

The system bus is a two-stage pipelined bus, which makes the timing of any access dependent on the access during the previous clock. Table 15-2 shows the wait states for accesses, column Current is the access being measured, and column Previous is the RAM access during the previous clock.

Table 15-2. Wait States During RAM Access

Current	Previous	Waits
Read	Idle	1
	Pipelined Read	
	Burst Read	
	64-bit Write	2
	8/16/32-bit Write	0 (if reading from the same address)
		1 (if reading from a different address)
Pipelined Read	Read	0
Burst Read	idle	1,0,0,0

Table 15-2. Wait States During RAM Access (continued)

Current	Previous	Waits
8/16/32-bit Write	idle	1
	Read	
	Pipelined 8/16/32-bit write	2
	64-bit write	
	8/16/32-bit write	0 (if writing to the same addrerss)
Pipelined 8/16/32-bit Write	8/16/32-bit Write	0
64-bit Write	idle	0
	64-bit Write	
	Read	
64-bit Burst Write	idle	0,0,0,0
	64-bit Write	
	Read	

15.4.3 Reset Operation

When a reset event asserts while an access to system memory is in progress, the access will either complete successfully, or will not occur, depending on the cycle at which the reset occurs. Any data stored during such an access will be the intended data, and no other address locations will be accessed or changed. If the system RAM is cached, dirty cache lines may not be completely written to memory unless the region is set for write through mode.

15.5 Initialization/Application Information

In order to use the SRAM, it is essential for the ECC check bits to be initialized after power on. A 64-bit cache inhibited write to each location in SRAM should be used to initialize the SRAM array as part of the application initialization code. The write transfer must be 64 bits in size, otherwise the write transfer will generate a read / modify / write operation which will check the ECC value upon the read. See Section 15.4.1, “SRAM ECC Mechanism.”

NOTE

The SRAM must be initialized, even if the application does not use ECC reporting.

15.5.1 Example Code

For proper initialization, a 64-bit write must be made to all SRAM locations. The PowerPC BookE instruction set provides the store multiple word (**stmw)** instruction to implement 64-bit writes. The **stmw** instruction concatenates two 32-bit registers for use as a single 64-bit write. To insure that the writes are

64-bit, the writes must be made on 64-bit word aligned boundaries, and an even number of registers should be specified.

The following example code illustrates the use of the **stmw** instruction to initialize the SRAM ECC bits.

```
init_L2RAM:

lis      r11,0x4000        # base address of the L2SRAM, 64-bit word aligned
ori      r11,r11,0         # not needed for this address but could be forothers

li       r12,512           # loop counter to get all of L2SRAM;
                           # 64k/4 bytes/32 GPRs = 512
mtctr    r12

init_l2ram_loop:
stmw     r0,0(r11)         # write all 32 GPRs to L2SRAM
addi     r11,r11,128       # inc the ram ptr; 32 GPRs * 4 bytes = 128
bdnz     init_l2ram_loop   # loop for 64k of L2SRAM

blr                        # done
```

15.6 Revision History

Substantive Changes since Rev 3.0
Added Section 15.4.3, “Reset Operation.”

Chapter 16
Boot Assist Module (BAM)

16.1 Introduction

This chapter describes the boot assist module (BAM).

16.1.1 Block Diagram

Figure 16-1 is a block diagram of the BAM.

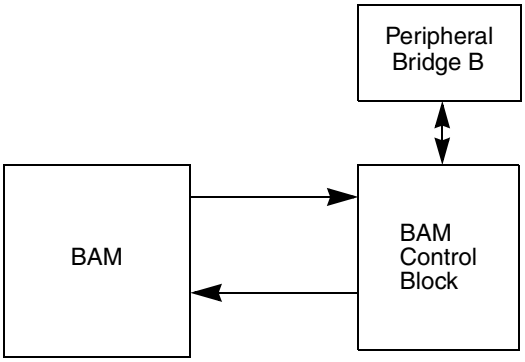


Figure 16-1. BAM Block Diagram

16.1.2 Overview

The MPC5553/MPC5554 BAM contains the MCU boot program code, identical for all eSys MCUs with an e200z6 core. The BAM control block is connected to peripheral bridge B and occupies the last 16 Kbytes of the MCU memory space. The BAM program supports four different booting modes: from internal Flash, from external memory without bus arbitration, from external memory with bus arbitration, serial boot via SCI or CAN interfaces. The BAM program is executed by the e200z6 core just after the MCU reset. Depending on the boot mode, the program initializes appropriate minimum MCU resources to start user code execution.

16.1.3 Features

The BAM program provides:

- Initial e200z6 core MMU setup with minimum address translation for all internal MCU resources and external memory address space
- Location and detection of user boot code
- Automatic switch to serial boot mode if internal or external Flash is blank or invalid
- User programmable 64-bit password protection for serial boot mode
- Booting user code from internal Flash module, from external memory without arbitration and from external memory with arbitration
- Serial boot by loading user program via CAN bus or eSCI to the internal SRAM
- Censorship protection for internal flash module

- An option to enable the e200z6 core watchdog timer
- An option to configure the external data bus to 16- or 32-bits wide (416 PBGA package only)

16.1.4 Modes of Operation

16.1.4.1 Normal Mode

In normal operation the BAM responds to all read requests within its address space. The BAM program is executed following the negation of reset.

16.1.4.2 Debug Mode

The BAM program is not executed when the MCU comes out of reset in OnCE debug mode. The user should provide the required MCU initialization using the development tool before accessing the MCU resources.

16.1.4.3 Internal Boot Mode

This mode of operation is intended for systems that boot from internal Flash memory. The internal Flash is used for all code and all boot configuration data. Once the BAM program has completed the boot process, user code may enable the external bus interface if required.

16.1.4.4 External Boot Modes

This mode of operation is intended for systems that have user code and configuration information in an external memory device connected to the external bus. The bus arbitration can be enabled to allow a boot option for multiprocessor systems.

Note that external boot mode should not be chosen for devices that do not have an external bus.

16.1.4.5 Serial Boot Mode

This mode of operation is intended to load a user program into internal SRAM using either the eSCI or CAN serial interface, then to execute that program. The program can then be used to control the download of data and erasing/programming of the internal or external Flash memory.

16.2 Memory Map/Register Definition

The BAM occupies 16 Kbytes of memory space, 0xFFFF_C000 to 0xFFFF_FFFF. The actual code size of the BAM program is less than 4 Kbytes and starts at 0xFFFF_F000, repeating itself down every 4 Kbytes in the BAM address space. The CPU starts the BAM program execution at its reset vector from address 0xFFFF_FFFC. Table 16-1 shows the BAM address map.

Table 16-1. BAM Memory Map

Address	Description
0xFFFF_C000 - 0xFFFF_CFFF	BAM Program Mirrored
0xFFFF_D000 - 0xFFFF_DFFF	BAM Program Mirrored

Table 16-1. BAM Memory Map (continued)

0xFFFF_E000 - 0xFFFF_EFFF	BAM Program Mirrored
0xFFFF_F000 - 0xFFFF_FFFF	BAM Program

16.3 Functional Description

16.3.1 BAM Program Resources

The BAM program uses/initializes following MCU resources:

- The BOOTCFG field in the reset status register (SIU_RSR) to determine the boot option.
- The location and value of the reset configuration half word (RCHW) to determine the location of boot code and the boot configuration options. Refer to Chapter 4, "Reset" for information about the RCHW.
- The DISNEX bit in the SIU_CCR to determine if the Nexus port is enabled.
- The MMU to allow core access to the MCU internal resources and external bus.
- The EBI registers and external bus pads, when performing external boot modes.
- The CAN_A, eSCI_A and their pads, when performing serial boot mode.
- The eDMA during serial boot mode.

16.3.2 BAM Program Operation

BAM is accessed by the MCU core after the negation of $\overline{\text{RSTOUT}}$, before user code starts.

First, the BAM program configures e200z6 core MMU to allow access to all MCU internal resources and external memory space, according the Table 16-2. This MMU setup remains the same for internal Flash Boot mode.

Table 16-2. MMU Configuration for Internal Flash Boot

TLB Entry	Region	Logical Base Address	Physical Base Address	Size	Attributes
0	Peripheral Bridge B and BAM	0xFFF0_0000	0xFFF0_0000	1 Mbyte	Cache inhibited Guarded Big Endian Global PID
1	Internal Flash	0x0000_0000	0x0000_0000	16 Mbytes	Cache enabled Not guarded Big Endian Global PID
2	EBI	0x2000_0000	0x0000_0000	16 Mbytes	Cache enabled Not guarded Big Endian Global PID
3	Internal SRAM	0x4000_0000	0x4000_0000	256 Kbytes	Cache inhibited Not guarded Big Endian Global PID
4	Peripheral Bridge A	0xC3F0_0000	0xC3F0_0000	1 Mbyte	Cache inhibited Not Guarded[1] Big Endian Global PID

[1] For future compatibility, configure peripheral bridge A as guarded.

The MMU regions are mapped with logical address the same as physical address except for the external bus interface (EBI). The logical EBI address space is mapped to physical addresses of the internal Flash memory. This allows a code, written to run from external memory, to be executed from internal Flash

Then the BAM program reads the status of the two BOOTCFG pins from the reset status register (SIU_RSR) and the appropriate boot sequence is started as shown in the Table 16-3.

Depending on the values stored in the censorship word and serial boot control word in the shadow row of internal Flash memory, the internal Flash memory can be enabled or disabled, the Nexus port can be enabled or disabled, the password received in serial boot mode is compared with a fixed public password or compared to a user programmable password in the internal Flash memory. The Table 16-3 summarizes all these possibilities.

Table 16-3. Boot Modes

BOOTCFG [0:1]	Censorship Control 0x00FF_FDE0	Serial Boot Control 0x00FF_FDE2	Boot Mode Name	Internal Flash State	Nexus State	Serial Password
00	!0x55AA	Don't care	Internal—Censored	Enabled	Disabled	Flash
	0x55AA		Internal—Public	Enabled	Enabled	Public

Note: '!' = 'NOT', meaning any value other than the value specified. Values 0x0000 and 0xFFFF should not be used.

Table 16-3. Boot Modes (continued)

01	Don't care	0x55AA	Serial—Flash Password	Enabled	Disabled	Flash
		!0x55AA	Serial—Public Password	Disabled	Enabled	Public
10	!0x55AA	Don't care	External—No Arbitration—Censored	Disabled	Enabled	Public
	0x55AA		External—No Arbitration—Public	Enabled	Enabled	Public
11	!0x55AA	Don't care	External—External Arbitration —Censored	Disabled	Enabled	Public
	0x55AA		External—External Arbitration —Public	Enabled	Enabled	Public

Note: '!' = 'NOT', meaning any value other than the value specified. Values 0x0000 and 0xFFFF should not be used.

The censorship word is a 32-bit word of data stored in the shadow row of internal Flash memory. This memory location is read and interpreted by hardware as part of the boot process and is used in conjunction with the BOOTCFG pins to enable/disable the internal Flash memory and the Nexus interface. The memory address of the censorship word is 0x00FF_FDE0. The censorship word consists of two fields: censorship control and serial boot control. The censorship word is programmed during manufacturing to be 0x55AA_55AA. This results in a device that is not censored and uses a Flash-based password for serial boot mode.

Censorship Word at 0x00FF_FDE0

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	0	1	0	1	0	1	1	0	1	0	1	0	1	0
Censorship Control - showing an uncensored part (factory default)															

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	1	0	1	0	1	0	1	1	0	1	0	1	0	1	0
Serial Boot Control - showing the use of the Flash based password (factory default)															

Figure 16-2. Censorship Word

The BAM program uses the state of the DISNEX bit to determine whether the serial password received in serial boot mode should be compared to a public password (fixed value of the 0xFEED_FACE_CAFE_BEEF) or needs to be compared to a Flash password - 64 bits data, stored in the shadow row of internal Flash at address 0x00FF_FDD8.

Flash Password at 0x00FF_FDD8

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	1	1	1	1	1	0	1	1	1	0	1	0	1	0
Serial Boot Password (0x00FF_FDD8) - 0xFEED (Factory Default)															

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
1	1	1	1	1	0	1	0	1	1	0	0	1	1	1	0
Serial Boot Password (0x00FF_FDDA) - 0xFACE (Factory Default)															

32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
1	1	0	0	1	0	1	0	1	1	1	1	1	1	1	0
Serial Boot Password (0x00FF_FDDC) - 0xCAFE (Factory Default)															

48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
1	0	1	1	1	1	1	0	1	1	1	0	1	1	1	1
Serial Boot Password (0x00FF_FDDE) - 0xBEEF (Factory Default)															

Figure 16-3. Serial Boot Flash Password

The BAM program continues to make specific initialization in one of the four boot modes.

16.3.2.1 Internal Boot Mode Flow

When the BAM software detects internal Flash boot mode, it sets up a bus error exception handler because it will be accessing Flash memory locations that may be corrupted and cause a bus error. Then the BAM program tries to find a valid RCHW in six predefined locations. If a valid RCHW is found, the BAM program enables the e200z6 watchdog timer with the RCHW[WTE] bit. If a valid RCHW is not found, the BAM program proceeds to the serial boot mode.

16.3.2.1.1 Finding Reset Configuration Half Word

The BAM searches the internal Flash memory for a valid reset configuration half word (RCHW). A valid RCHW is a 16-bit value that contains a fixed 8-bit boot identifier and some configuration bits (see Section 4.4.3.5.1, “Reset Configuration Half Word Definition”). The RCHW is expected to be the first half word in one of the low address space Flash blocks as shown in Table 16-4.

Table 16-4. Low Address Space (LAS) Block Memory Addresses

Block	Address
0	0x0000_0000
1	0x0000_4000
2	0x0001_0000
3	0x0001_C000
4	0x0002_0000
5	0x0003_0000

BOOT_BLOCK_ADDRESS is the first address from Table 16-4, where the BAM program finds a valid RCHW.

If the BAM program does find a valid RCHW, the watchdog is enabled with the RCHW[WTE] bit, the BAM program fetches the reset vector from the address of the BOOT_BLOCK_ADDRESS + 0x4, and branches to the reset boot vector. A user application should have a valid instruction at the reset boot vector address.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
A16	A17	A18	A19	A20	A21	A22	A23	A24	A25	A26	A27	A28	A29	A30	A31

BOOT_BLOCK_ADDRESS + 0x0000_0004

Figure 16-4. Reset Boot Vector

The watchdog timeout is set to 2.5×2^{17} system clock periods if the watchdog is enabled.

16.3.2.2 External Boot Modes Flow.

The external boot mode is used to boot a user application from an external asynchronous memory that is connected to the MCU external bus; it is controlled by $\overline{CS0}$.

16.3.2.2.1 External Boot MMU Configuration

As shown in Table 16-5 , the BAM program sets up the two MMU regions differently than in internal Flash boot mode. The internal Flash logical address space is mapped to the physical addresses of the EBI.

Table 16-5. MMU Configuration for an External Boot

TLB Entry	Region	Logical Base Address	Physical Base Address	Size	Attributes
1	Internal Flash Memory	0x0000_0000	0x2000_0000	16 Mbytes	• Cache enabled • Not guarded • Big Endian • Global PID
2	EBI	0x2000_0000	0x2000_0000	16 Mbytes	• Cache enabled • Not guarded • Big Endian • Global PID

This allows a code, written to run from internal Flash memory, to be executed from the external memory.

16.3.2.2.2 Single Bus Master or Multiple Bus Masters

External boot mode has two options for booting:

- External boot with no arbitration — This option is a single master system where the MCU is the only bus master in the system and therefore does not need to consider arbitration of the external bus.
- External boot with external arbitration — This option is where there is another bus master on the external bus and arbitration of the bus is handled external to the MCU.

These two modes are selected based on the state of the two BOOTCFG pins.

In a multiple master system where both are booting from the same external bus memory, one boots in external boot with no arbitration mode while the other boots in external boot with external arbitration mode.

The configuration of the EBI is different for the two modes.

16.3.2.2.3 External Boot—Single Master with no Arbitration EBI Configuration

The BAM program configures:

1. Chip select $\overline{CS}$0 region as a 16-bit port with a base address of 0x2000_0000, no burst, 15 wait states, 8 Mbyte size.
2. EBI for no external master (clear EXTM bit).
3. Enables the EBI for normal operation.
4. Configures the following I/O pins as bus signals: address signals[8:31]; data[0:15]; $\overline{WE}$0; $\overline{OE}$; $\overline{TS}$; $\overline{CS}$0. Data[16:31] is also configured if RCHW[PS0] = 0. See for more information.

16.3.2.2.4 External Boot with External Arbitration EBI Configuration

In the external boot mode with external arbitration the BAM program also does the following:

1. Sets the EXTM bit, enabling the EBI for external master operation.
2. Configures EBI for external arbitration (sets the EARB bit).
3. Configures the additional I/O signals $\overline{BB}$, $\overline{BG}$, $\overline{BR}$ for bus function . See Table 16-6.

Table 16-6. External Bus Interface Configuration

Pins	Reset	Serial Boot Mode[1] or Internal Boot Mode	External Boot with no Arbitration[2] (Single Master Mode)		External Boot with External Arbitration [2] (Multi Master Mode)	
	Function	Function	Function	PCR	Function	PCR
ADDR[8:31]	GPIO	GPIO	ADDR[8:31]	0x0440	ADDR[8:31]	0X0440
DATA[16:31]	GPIO	GPIO	GPIO[3]	Default [3]	GPIO [3]	Default [3]
DATA[0:15]	GPIO	GPIO	DATA[0:15]	0X0440	DATA[0:15]	0X0440
BB	GPIO	GPIO	GPIO	Default	BB	0X0443
BG	GPIO	GPIO	GPIO	Default	BG	0X0443
BR	GPIO	GPIO	GPIO	Default	BR	0X0443
TSIZ[0:1]	GPIO	GPIO	GPIO	Default	GPIO	Default
TEA	GPIO	GPIO	GPIO	Default	GPIO	Default
CS0	GPIO	GPIO	CS0	0X0443	CS0	0X0443
WE0_BE0	GPIO	GPIO	WE0	0X0443	WE0_BE0	0X0443
OE	GPIO	GPIO	OE	0X0443	OE	0X0443
TS	GPIO	GPIO	TS	0X0443	TS	0X0443
TA	GPIO	GPIO	GPIO	Default	GPIO	Default
RD_WR	GPIO	GPIO	GPIO	Default	GPIO	Default
CS[1:3]	GPIO	GPIO	GPIO	Default	GPIO	Default
BDIP	GPIO	GPIO	GPIO	Default	GPIO	Default
WE[1:3]_ BE[1:3]	GPIO	GPIO	GPIO	Default	GPIO	Default

[1] This column is for serial boot mode only when entered directly using the BOOTCFG signals (See note 2).

[2] If serial boot is entered indirectly from either external boot mode because a valid RCHW was not found, the EBI remains configured according to these columns.

[3] If the BAM reads a valid RCHW with the PS0 bit clear, data[16:31] are reconfigured from GPIO to data bus signals by writing 0x0440 to the PCRs.

16.3.2.2.5 Reset Configuration Half Word Read

The BAM program checks for a valid reset configuration half word (RCHW, see Figure 16-4) at the first location in external memory, i.e address 0x2000_0000.

If the BAM program fails to find a valid RCHW, it assumes the external memory does not contain a user application and switches to serial boot mode.

If the BAM program does find a valid RCHW, it configures data pins and $\overline{CS}0$ port size according to the RCHW[PS0] bit and the e200z6 core watchdog according to the RCHW[WTE] bit. The watchdog timeout is set to 2.5×2^{17} system clock periods. Then the BAM program reads the reset vector (Figure 16-4) from the address 0x2000_0004 and branches to that reset vector address, starting user program execution.

16.3.2.3 Serial Boot Mode Operation

In this mode of operation, the CAN_A and the eSCI_A GPIO signals are reconfigured, unused message buffers in CAN_A are used as scratch pad RAM, the MMU is setup; the watchdog is enabled. No exceptions are used.

16.3.2.3.1 Serial Boot Mode MMU and EBI Configuration

The BAM program sets up the MPC5553/MPC5554 MMU for all peripheral and memory regions in one of two different modes and sets up the EBI in one of three different modes; depending on how serial boot mode was entered.

If serial boot mode is entered directly by choosing the mode with the BOOTCFG signals, or was entered indirectly from internal boot mode because no valid RCHW was found, then the MMU is configured the same way as for internal boot mode. See Table 16-3 for more information. The EBI is disabled and all bus pins function as GPIO.

If serial boot mode is entered indirectly from either external boot/single master or external boot/multimaster/external arbitration because no valid RCHW was found, then the MMU and EBI are configured the same way as for one of the external boot modes with a 16-bit data bus. See Table 16-5 for more information.

16.3.2.3.2 CAN and eSCI Configuration

In serial boot mode, the BAM program configures CAN_A and eSCI_A to receive messages. The CNRX_A signal and the RXD_A signals are configured as inputs to the CAN and eSCI modules. The CNTX_A signal is configured as an output from the CAN module. The TXD_A signal of the eSCI_A remains configured as GPIO input. The BAM program writes the e200z6 core timebase registers (TB) to 0x0000_0000_0000_0000 and enables the e200z6 core watchdog timer to use the system clock and to cause a reset after a time-out period of 3×2^{28} system clock cycles. (See Table 16-7 for examples of time out periods.)

The CAN controller is configured to operate at a baud rate equal to the system clock frequency divided by 60 with one message buffer (MB) using the standard 11-bit identifier format detailed in the CAN 2.0A specification. If the PLL is enabled out of reset, the default system clock is 1.5 times the crystal frequency.

(See Chapter 11, "Frequency Modulated Phase Locked Loop (FMPLL) and System Clocks," for more information.) So with the PLL enabled, the baud rate is equal to the crystal frequency divided by 40. (See Table 16-7 for examples of baud rates)

The BAM ignores the following errors:

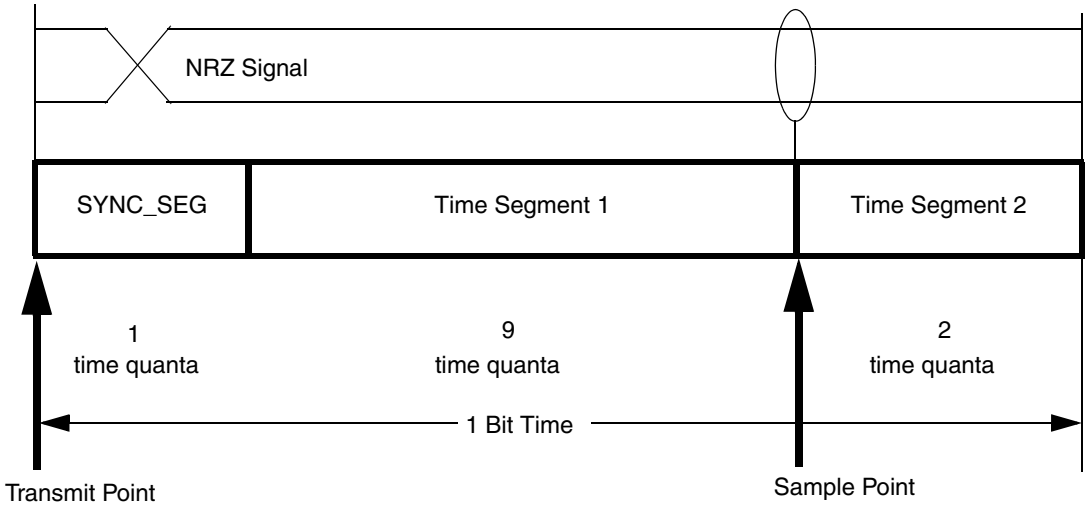
- Bit1 errors
- Bit0 errors
- Acknowledge errors
- Cyclic redundancy code errors
- Form errors
- Stuffing errors
- TX error counter errors
- Rx error counter errors

All data received is assumed to be good and is echoed out on the CNTX_A signal.

NOTE

It is the responsibility of the host computer to compare the 'echoes' with the sent data and restart the process if an error is detected.

See Figure 16-5 for details of CAN bit timing.



1 time Quanta = 5 system clock periods = 3 1/3 crystal clock periods (with PLL enabled)

Figure 16-5. CAN Bit Timing

The eSCI is configured for 1 start bit, 8 data bits, no parity and 1 stop bit and to operate at a baud rate equal to the system clock divided by 1250. See Table 16-7 for examples of baud rates.

The BAM ignores the following eSCI errors:

- Overrun errors
- Noise errors
- Framing errors
- Parity errors

All data received is assumed to be good and is echoed out on the TXD signal. It is the responsibility of the host computer to compare the echoes with the sent data and restart the process if an error is detected.

Table 16-7. Serial Boot Mode—Baud Rate and Watchdog Summary

Crystal Frequency (MHz)	System Clock Frequency (MHz)	SCI Baud Rate (baud)	CAN Baud Rate (baud)	Watchdog Timeout period (seconds)
f_{xtal}	$f_{sys}=1.5 * f_{xtal}$	f_{sys} / 1250	f_{sys} / 60	$2.5 * 2^{27} / f_{sys}$
8	12	9600	200K	67.1
12	18	14400	300K	44.7
16	24	19200	400K	33.6
20	30	24000	500K	26.8

Upon reception of either a valid CAN message with an ID equal to 0x011 and containing 8 bytes of data or a valid eSCI message, the BAM moves to one of two serial boot submodes: either CAN serial boot mode or eSCI serial boot mode.

In CAN serial boot mode, the eSCI_A signal RXD_A reverts to GPIO input. The ensuing download protocol is assumed to be all on the CAN bus; eSCI messages are ignored.

In eSCI serial boot mode, the CAN_A signals CNRX_A and CNTX_A revert to GPIO inputs and the TXD_A signal is configured as an output. The ensuing download protocol is assumed to be on the eSCI bus and CAN messages are ignored.

Table 16-8. CAN/eSCI Reset Configuration for CAN/eSCI Boot

Pins	Reset Function	Initial Serial Boot Mode	Serial Boot Mode after a valid CAN message received	Serial Boot Mode after a valid eSCI message received
CNTX_A	GPIO	CNTX_A	CNTX_A	GPIO
CNRX_A	GPIO	CNRX_A	CNRX_A	GPIO
TXD_A	GPIO	GPIO	GPIO	TXD_A
RXD_A	GPIO	RXD_A	GPIO	RXD_A

Table 16-9. CAN/eSCI Reset Pin Configuration

Pins	I/O	Weak Pull-Up State	Hysteresis	Driver Configuration	Slew Rate	Input Buffer Enable
CNTX_A / TXD_A	Output	Enabled/Up	—	Push/Pull	Medium	N
CNRX_A / RXD_A	Input	Enabled/Up	Y	—	—	—
GPIO	Input	Enabled/Up	Y	—	—	—

16.3.2.3.3 CAN Serial Boot Mode Download Protocol

The download protocol follows 4 steps:

1. Download 64-bit password
2. Download start address and size of download

3. Download data
4. Execute code from start address

Each step must complete before the next step starts.

1. Download 64-bit password

   The host computer must send a CAN message with ID = 0x011 and containing the 64-bit serial download password. CAN messages with other IDs or fewer bytes of data are ignored. When a valid message has been received, the BAM transmits a CAN message using ID = 0x001 and containing the data received. The host should not send a second CAN message until the echo of the first message has been received. A CAN message sent before the echo is received is ignored.

   The received 64-bit password is checked for validity. It is checked to ensure that none of the 4 x 16-bit half words are 0x0000 or 0xFFFF. These are considered illegal passwords. A password must have at least one 0 and one 1 in each half word lane to be considered legal.

   The BAM program then checks the censorship status of the MCU by checking the DISNEX bit in the SIU_CCR. If Nexus is disabled, the MCU is considered to be censored and the password is compared with a password stored in the shadow row in internal Flash memory.

   If Nexus is enabled, the MCU is considered to be not censored or booting from external Flash and the password is compared to the fixed value = 0xFEED_FACE_CAFE_BEEF.

   If the password fails any of these validity tests, the MCU stops responding to all stimulus. To repeat boot operation the MCU needs to be reset by external reset or by watchdog. If the password is valid, the BAM program refreshes the e200z6 watchdog timer and the next step in the protocol can be performed.

2. Download start address and size of download

   The host computer must send a CAN message with ID = 0x012 and containing a 32 bit address in internal SRAM, indicating where the following data should be stored in the memory map of the MCU; and a 32 bit number indicating how many bytes of data are to be received and stored in memory before switching to execute the code just loaded. The start address is assumed to be on a word boundary (4 bytes), therefore the least significant 2 bits of the address are ignored. CAN messages with other IDs or fewer bytes of data are ignored. When a valid message has been received, the BAM transmits a CAN message using ID = 0x002 and containing the data received. The host should not send a another CAN message until the echo of the previous message has been received by the host. A CAN message sent before the echo is received is ignored.

3. Download data

   The host computer must send a succession of CAN messages with ID = 0x013 (The data length is variable) and containing raw binary data. Each byte of data received is stored in the MCU's memory, starting at the address specified in the previous protocol step and incrementing through memory until the number of bytes of data received and stored in memory matches the number specified in the previous protocol step. CAN messages with other IDs are ignored. When a valid message has been received, the BAM transmits a CAN message using ID = 0x003 and containing the data received. The host should not send another CAN message until the echo of the previous message has been received by the host. A CAN message sent before the echo is received is ignored.

NOTE

Internal SRAM is protected by 64 bit wide error correction coding hardware (ECC). This means that any write to uninitialized internal SRAM must be 64 bits wide, otherwise an ECC error occurs. Therefore the BAM buffers downloaded data until 8 bytes have been received then does a single 64 bit wide write. Only internal SRAM supports 64 bit writes therefore attempting to download data to other RAM apart from internal SRAM causes errors. If the start address of the downloaded data is not on an 8 byte boundary, the BAM writes 0x00 to the memory locations from the preceeding 8 byte boundary to the start address (maximum 4 bytes). The BAM also writes 0x00 to all memory locations from the last byte of data downloaded to the following 8 byte boundary (maximum 7 bytes)

4. Execute code

   The BAM waits for the last CAN message transmission to complete. Then the CAN controller is disabled. CNTX_A and CNRX_A revert to GPIO inputs. Then the BAM switches execution to the downloaded code by branching to the first address in which code is stored, as specified in step 2 of the protocol.

NOTE

The code that is downloaded and executed must periodically refresh the e200z6 watchdog timer or change the timeout period to a value that does not cause resets during normal operation.

Table 16-10. CAN Serial Boot Mode Download Protocol

Protocol Step	Host Sent Message	MCU Response Message	Action
1	CAN ID 0x011 + 64-bit password	CAN ID 0x001 + 64-bit password	Password checked for validity and compared against stored password. e200z6 Watchdog timer is refreshed if the password check is successful
2	CAN ID 0x012 + 32-bit store address + 32-bit number of bytes	CAN ID 0x002 + 32-bit store address + 32-bit number of bytes	Load address and size of download are stored for future use
3	CAN ID 0x013 + 8 to 64 bits of raw binary data	CAN ID 0x003 + 8 to 64 bits of raw binary data	Each byte of data received is store in MCU memory, starting at the address specified in the previous step and incrementing until the amount of data received and stored, matched the size as specified in the previous step.
4	None	None	The BAM program returns I/O pins and CAN module to their reset state, then branches to the first address the data was stored to (As specified in step 2)

16.3.2.3.4 eSCI Serial Boot Mode Protocol

The download protocol follows four steps:

1. Download 64-bit password
2. Download start address and size of download
3. Download data
4. Execute code from start address

Each step must complete before the next step starts. The eSCI operates in half duplex mode where the host sends a byte of data, then waits for the echo back from the MCU before proceeding with the next byte. Bytes sent from the host before the previous echo from the MCU is received, are ignored.

1. Download 64-bit password

   The first 8 bytes of eSCI data the host computer sends must contain the 64-bit serial download password. For each valid eSCI message received, the BAM transmits the same data on the eSCI_A TXD_A signal.

   The received 64-bit password is checked for validity. It is checked to ensure that none of the 4 x 16-bit half words are 0x0000 or 0xFFFF, which are considered illegal passwords. A password must have at least one 0 and one 1 in each half word lane to be considered legal.

   The BAM program then checks the censorship status of the MCU by checking the DISNEX bit in the SIU_CCR. If Nexus is disabled, the MCU is considered to be censored and the password is compared with a password stored in the shadow row in internal Flash memory.

   If Nexus is enabled, the MCU is considered to be not censored or is booting from external Flash and the password is compared to the fixed value of 0xFEED_FACE_CAFE_BEEF.

   If the password fails any of these validity tests, the MCU stops responding to all stimulus. To repeat the boot operation the only options are to assert the RESET signal or wait for watchdog reset the MCU. If the password is valid, the BAM refreshes the e200z6 watchdog timer and the next step in the protocol can be performed.

2. Download start address and size of download

   The next 8 bytes of eSCI data the host computer sends must contain a 32-bit address in internal SRAM, indicating where the following data should be stored in the memory map of the MCU; and a 32-bit number indicating how many bytes of data are to be received and stored in memory before switching to execute the code just loaded. The start address is assumed to be on a word boundary (4 bytes), therefore the least significant 2 bits of the address are ignored. For each valid eSCI message received, the BAM transmits the same data on the eSCI_A TXD_A signal.

3. Download data

   The host computer must then send a succession of eSCI messages, each containing raw binary data. Each byte of data received is stored in the MCU's memory, starting at the address specified in the previous protocol step and incrementing through memory until the number of bytes of data received and stored in memory matches the number specified in the previous protocol step. For each valid eSCI message received, the BAM transmits the same data on the eSCI_A TXD_A signal.

NOTE

Internal SRAM is protected by 64 bit wide error correction coding hardware (ECC). This means that any write to uninitialized internal SRAM must be 64 bits wide, otherwise an ECC error occurs. Therefore the BAM buffers downloaded data until 8 bytes have been received then does a single 64 bit wide write. Only internal SRAM supports 64 bit writes therefore attempting to download data to other RAM apart from internal SRAM causes errors. If the start address of the downloaded data is not on an 8 byte boundary, the BAM writes 0x00 to the memory locations from the preceeding 8 byte boundary to the start address (maximum 4 bytes). The BAM also writes 0x00 to all memory locations from the last byte of data downloaded to the following 8 byte boundary (maximum 7 bytes).

4. Execute code

   The BAM waits for the last eSCI message transmission to complete and then the eSCI is disabled. TXD_A and RXD_A revert to general-purpose inputs. The BAM switches execution to the downloaded code by branching to the first address in which code was stored, as specified in step 2 of the protocol.

NOTE

The code that is downloaded and executed must periodically refresh the e200z6 watchdog timer or change the timeout period to a value that does not cause resets during normal operation.

Table 16-11. eSCI Serial Boot Mode Download Protocol

Protocol Step	Host Sent Message	BAM Response Message	Action
1	64-bit password MSB first	64-bit password	Password checked for validity and compared against stored password. e200z6 watchdog timer is refreshed if the password check is successful
2	32-bit store address + 32-bit number of bytes MSB first	32-bit store address + 32-bit number of bytes	Load address and size of download are stored for future use
3	8 bits of raw binary data	8 bits of raw binary data	Each byte of data received is store in MCU memory, starting at the address specified in the previous step and incrementing until the amount of data received and stored, matched the size as specified in the previous step.
4	None	None	The BAM returns I/O pins and the eSCI module to their reset state, with the exception that ESCI_A_CR2[MDIS] is asserted rather than negated. Then it branches to the first address the data was stored to (as specified in step 2)

16.3.3 Interrupts

No interrupts are generated by or are enabled by the BAM.

16.4 Revision History

Substantive Changes since Rev 3.0
• Section 16.3.2.1.1, “Finding Reset Configuration Half Word,” changed 2^{18} to be 2.5×2^{17} • Section 16.3.2.2.5, “Reset Configuration Half Word Read,” changed 2^{18} to be 2.5×2^{17} • Table 16-7 changed $3^* 2^{28}/ f_{sys}$ to be $2.5 * 2^{27}/ f_{sys}$

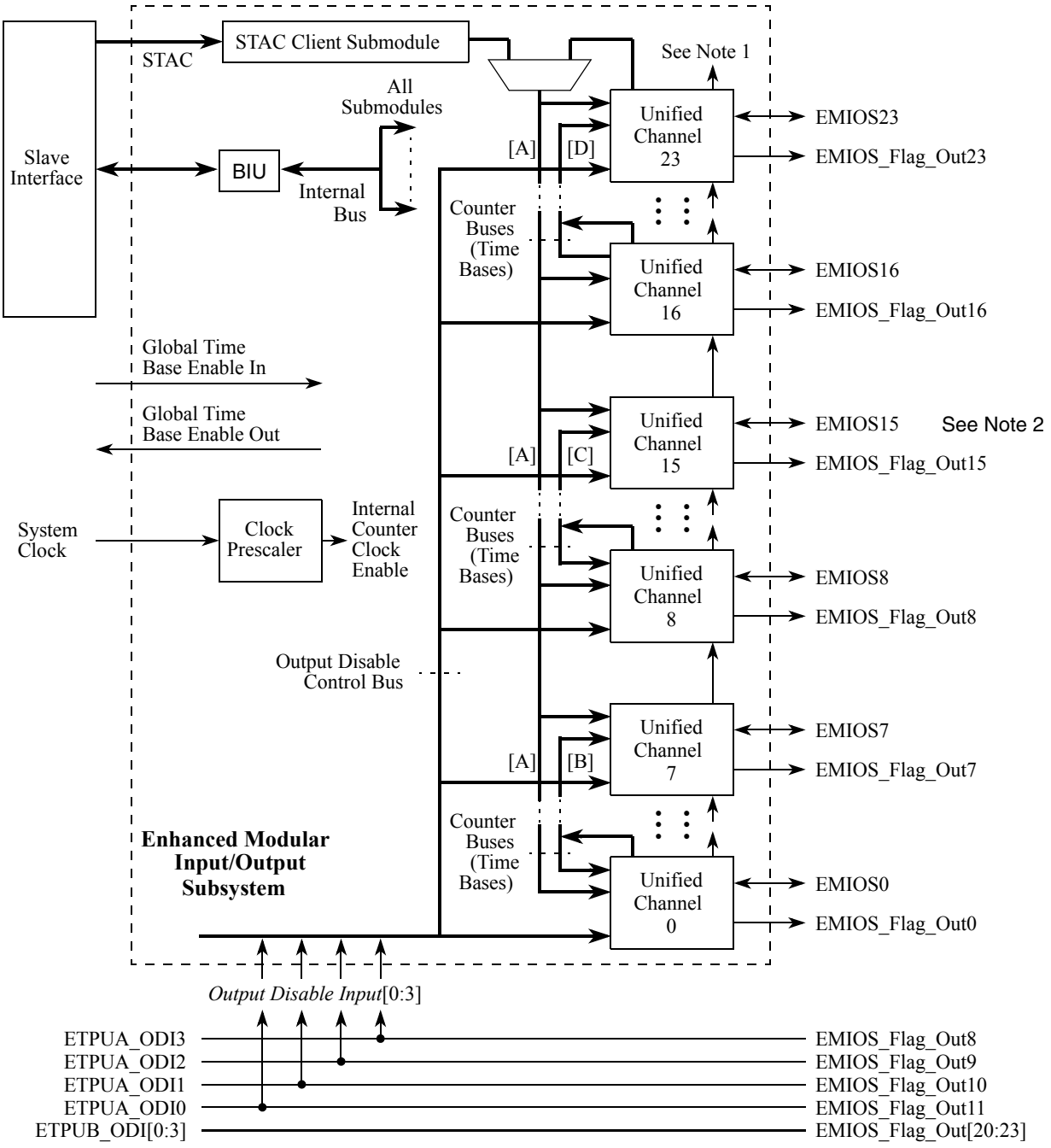
Chapter 17
Enhanced Modular Input/Output Subsystem (eMIOS)

17.1 Introduction

This chapter describes the enhanced modular input/output subsystem (eMIOS) of the MPC5553/MPC5554, which provides functionality to generate or measure timed events.

17.1.1 Block Diagram

Figure 17-1 shows the block diagram of the eMIOS.



Note 1: Connection between UC[n-1] and UC*n* necessary to implement QDEC mode.

Note 2: On channels 12–15, there is no input from EMIOS[12:15], but only from the DSPI module.

Figure 17-1. eMIOS Block Diagram

17.1.2 Overview

The eMIOS builds on the MIOS concept by using a unified channel module that provides a superset of the functionality of all the individual MIOS channels, while providing a consistent user interface. This allows more flexibility as each unified channel can be programmed for different functions.

17.1.3 Features

- 24 unified channels
- Unified channels features
  - 24-bit registers for captured/match values
  - 24-bit internal counter
  - Internal prescaler
  - Dedicated output pin for buffer direction control
  - Selectable time base
  - Can generate its own time base
- Four 24-bit wide counter buses
  - Counter bus A can be driven by unified channel 23 or by the STAC bus.
  - Counter bus B, C, and D are driven by unified channels 0, 8, and 16, respectively.
  - Counter bus A can be shared among all unified channels. UCs 0 to 7, 8 to 15, and 16 to 23 can share counter buses B, C, and D, respectively.
- One global prescaler
- Shared time bases through the counter buses
- Synchronization among internal and external time bases
- Shadow FLAG register
- State of module can be frozen for debug purposes
- DMA request capability for some channels
- Motor control capability

17.1.4 Modes of Operation

17.1.4.1 eMIOS Modes

The eMIOS operates in one of the modes described below:

- User mode
  This is the normal operating mode. When EMIOS_MCR[FRZ] = 0, and EMIOS_CCR[FREN] = 0, the eMIOS is in user mode.
- Debug mode
  Debug mode is individually programmed for each channel. When entering this mode, the UC registers' contents are frozen, but remain available for read and write access through the slave interface. After leaving debug mode, all counters that were frozen upon debug mode entry will resume at the point where they were frozen.

  In debug mode, all clocks are running and all registers are accessible; thus, this mode is not intended for power saving, but for use during software debugging.

- Freeze mode
  Freeze mode enables the eMIOS to freeze the registers of the unified channels when debug mode is requested at the MCU level. While in freeze mode, the eMIOS continues to operate to allow the MCU access to the unified channels' registers. The unified channel will remain frozen until the EMIOS_MCR[FRZ] bit is written to zero, the MCU exits debug mode, or a unified channel's EMIOS_CCR[FREN] bit is cleared.

17.1.4.2 Unified Channel Modes

The unified channels can be configured to operate in the following modes:

Table 17-1. Unified Channel Modes

Mode	MPC5554	MPC5553
General purpose input/output	Yes	Yes
Single action input capture	Yes	Yes
Single action output compare	Yes	Yes
Input pulse width measurement	Yes	Yes
Input period measurement	Yes	Yes
Double action output compare	Yes	Yes
Pulse/edge accumulation	Yes	Yes
Pulse/edge counting	Yes	Yes
Quadrature decode	Yes	Yes
Windowed programmable time accumulation	Yes	Yes
Modulus counter, normal	Yes	Yes
Modulus counter, buffered	No	Yes
Output pulse width and frequency modulation, normal	Yes	Yes
Output pulse width and frequency modulation, buffered	No	Yes
Center aligned output pulse width modulation with dead time insertion, normal	Yes	Yes
Center aligned output pulse width modulation with dead time insertion, buffered	No	Yes
Output pulse width modulation, normal	Yes	Yes
Output pulse width modulation, buffered	No	Yes

These modes are described in Section 17.4, "Functional Description."

17.2 External Signal Description

17.2.1 Overview

Each unified channel has one input and one output signal connected to the channel's I/O pin. Refer to the SIU, eTPU, and DSPI sections for details about the connection to pads and other modules.

NOTE

On channels 12-15, input can be from DSPI, but cannot be from eMIOS[12:15] because these are not pinned out. See Figure 2-8 and Figure 2-9).

The internal *output disable input* signals 0-3 (refer to Table 17-3) are provided to implement the output disable feature needed for motor control. They are connected to EMIOS_Flag_Out signals according to Section 17.2.1.2, "Output Disable Input—eMIOS Output Disable Input Signals."

17.2.1.1 External Signals

When configured as an input, EMIOS*n* is synchronized and filtered by the programmable input filter (PIF). The output of the PIF is then used by the channel logic and is available to be read by the MCU through the UCIN bit of the EMIOS_CSR*n*. When configured as an output, EMIOS*n* is a registered output and is available for reading by the MCU through the UCOUT bit of the EMIOS_CSR*n*.

Table 17-2. External Signals

Signal	Direction	Function	Reset State
EMIOS[0:11, 16:23]	Input	eMIOS Unified Channel n input	—
EMIOS[12:15]	Input	From DSPI	—
EMIOS[0:23]	Output	eMIOS Unified Channel n output	0 / Hi-Z[1]

[1] A value of 0 refers to the reset value of the signal. Hi-Z refers to the state of the external pin if a tri-state output buffer is controlled by the corresponding eMIOS signal.

17.2.1.2 Output Disable Input—eMIOS Output Disable Input Signals

Output disable inputs to both the eMIOS and the eTPU modules are connected to EMIOS_Flag_Out*n* signals according to Table 17-3.

Table 17-3. eMIOS Output Disable Input Signals

eMIOS Channel[1]	eMIOS Output Disable Input Signal[2]	eTPU Output Disable Input Signal[3]
EMIOS_Flag_Out8	*output disable input 3*	ETPUA_ODI3
EMIOS_Flag_Out9	*output disable input 2*	ETPUA_ODI2
EMIOS_Flag_Out10	*output disable input 1*	ETPUA_ODI1
EMIOS_Flag_Out11	*output disable input 0*	ETPUA_ODI0
EMIOS_Flag_Out20	—	ETPUB_ODI0
EMIOS_Flag_Out21	—	ETPUB_ODI1
EMIOS_Flag_Out22	—	ETPUB_ODI2
EMIOS_Flag_Out23	—	ETPUB_ODI3

1 All other EMIOS_Flag_Out*n* output signals are not connected.

2 Each of the four internal eMIOS *output disable input* signals can be programmed to disable the output of any eMIOS channel if that channel has selected output disable capability by the setting of its EMIOS_CCR*n*[ODIS] bit, and by specifying the output disable input in its EMIOS_CCRn[ODISSL] field.

3 ETPU*x*_ODI*y* input signals disable outputs for eTPU engine *x*, channels (*y**8) through (*y**8+7). Refer to the ETPU chapter for more details.

17.3 Memory Map/Register Definition

Addresses of unified channel (UC) registers are specified as offsets from the channel's base address, otherwise the eMIOS base address is used as reference.

The overall address map organization is shown in Table 17-4. Table 17-5 describes the unified channel registers. All registers are cleared on reset.

Table 17-4. eMIOS Memory Map

Address	Register Name	Register Description	Size (bits)
Base (0xC3FA_0000)	EMIOS_MCR	Module Configuration Register	32
Base + 0x004	EMIOS_GFR	Global Flag Register	32
Base + 0x008	EMIOS_OUDR	Output Update Disable Register	32
Base + 0x00C– Base + 0x01F	—	Reserved	—
Base + 0x020	UC0	Unified Channel 0 Registers	256
Base + 0x040	UC1	Unified Channel 1 Registers	256
Base + 0x060	UC2	Unified Channel 2 Registers	256
Base + 0x080	UC3	Unified Channel 3 Registers	256
Base + 0x0A0	UC4	Unified Channel 4 Registers	256
Base + 0x0C0	UC5	Unified Channel 5 Registers	256
Base + 0x0E0	UC6	Unified Channel 6 Registers	256
Base + 0x100	UC7	Unified Channel 7 Registers	256
Base + 0x120	UC8	Unified Channel 8 Registers	256
Base + 0x140	UC9	Unified Channel 9 Registers	256
Base + 0x160	UC10	Unified Channel 10 Registers	256
Base + 0x180	UC11	Unified Channel 11 Registers	256
Base + 0x1A0	UC12	Unified Channel 12 Registers	256
Base + 0x1C0	UC13	Unified Channel 13 Registers	256
Base + 0x1E0	UC14	Unified Channel 14 Registers	256
Base + 0x200	UC15	Unified Channel 15 Registers	256
Base + 0x220	UC16	Unified Channel 16 Registers	256
Base + 0x240	UC17	Unified Channel 17 Registers	256
Base + 0x260	UC18	Unified Channel 18 Registers	256
Base + 0x280	UC19	Unified Channel 19 Registers	256
Base + 0x2A0	UC20	Unified Channel 20 Registers	256
Base + 0x2C0	UC21	Unified Channel 21 Registers	256

Table 17-4. eMIOS Memory Map (continued)

Address	Register Name	Register Description	Size (bits)
Base + 0x2E0	UC22	Unified Channel 22 Registers	256
Base + 0x300	UC23	Unified Channel 23 Registers	256

Table 17-5. UC Memory Map

Address	Register Name	Register Description	Size (bits)
UC*n* Base + 0x00	EMIOS_CADR*n*	Channel A Data Register	32
UC*n* Base + 0x04	EMIOS_CBDR*n*	Channel B Data Register	32
UC*n* Base + 0x08	EMIOS_CCNTR*n*	Channel Counter Register	32
UC*n* Base + 0x0C	EMIOS_CCR*n*	Channel Control Register	32
UC*n* Base + 0x10	EMIOS_CSR*n*	Channel Status Register	32
UC*n* Base + 0x14– UC*n* Base + 0x1F	—	Reserved	—

17.3.1 Register Description

All registers are 32-bit wide. This section illustrates the eMIOS with 24 unified channels supporting 24-bit wide data.

17.3.1.1 eMIOS Module Configuration Register (EMIOS_MCR)

EMIOS_MCR contains global control bits for the eMIOS module.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MDIS	FRZ	GTBE	ETB	GPREN	0	0	0	0	0	0	SRV			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x00															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	GPRE								0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x00															

Figure 17-2. eMIOS Module Configuration Register (EMIOS_MCR)

Table 17-6. EMIOS_MCR Field Descriptions

Bits	Name	Description
0	—	Reserved. This bit is readable/writable, but has no effect.
1	MDIS	Module disable. Puts the eMIOS in low power mode. The MDIS bit is used to stop the clock of the module, except the access to registers EMIOS_MCR and EMIOS_OUDR. 0 Clock is running 1 Enter low power mode
2	FRZ	Freeze. Enables the eMIOS to freeze the registers of the unified channels when debug mode is requested at MCU level. Each unified channel should have FREN bit set in order to enter freeze mode. While in freeze mode, the eMIOS continues to operate to allow the MCU access to the unified channels registers. The unified channel will remain frozen until the FRZ bit is written to zero or the MCU exits debug mode or the unified channel FREN bit is cleared. 0 Allows unified channels to continue to operate when device enters debug mode and the EMIOS_CCR*n*[FREN] bit is set 1 Stops unified channels operation when in debug mode and the EMIOS_CCR*n*[FREN] bit is set
3	GTBE[1]	Global time base enable. Used to export a global time base enable from the module and provide a method to start time bases of several modules simultaneously. 0 Global time base enable out signal negated 1 Global time base enable out signal asserted **Note:** The global time base enable input signal controls the internal counters. When asserted, internal counters are enabled. When negated, internal counters disabled.
4	ETB	External time base. Selects the time base source that drives counter bus[A]. 0 Unified channel 23 drives counter bus[A] 1 STAC drives counter bus[A] **Note:** If ETB is set to select STAC as the counter bus[A] source, the GTBE must be set to enable the STAC to counter bus[A]. See Section 17.4.2, “STAC Client Submodule” and the shared time and angle clock (STAC) bus interface section and the STAC bus configuration register (ETPU_REDCR) section of the eTPU chapter for more information about the STAC.
5	GPREN	Global prescaler enable. Enables the prescaler counter. 0 Prescaler disabled (no clock) and prescaler counter is cleared 1 Prescaler enabled
6–11	—	Reserved.
12–15	SRV [0:3]	Server time slot. Selects the address of a specific STAC server to which the STAC client submodule is assigned (refer to Section 17.4.2, “STAC Client Submodule,” for details) 0000 – eTPU engine A, TCR1 0001 – eTPU engine B, TCR1 0010 – eTPU engine A, TCR2 0011 – eTPU engine B, TCR2 0100–1111 reserved
16–23	GPRE [0:7]	Global prescaler. Selects the clock divider value for the global prescaler, as shown in Table 17-7.
24–31	—	Reserved.

[1] The GTBE signal is an inter-module signal, not an external pin on the device.

Table 17-7. Global Prescaler Clock Divider

GPRE[0:7]	Divide Ratio
00000000	1
00000001	2
. . . .	. . . .
11111111	256

17.3.1.2 eMIOS Global Flag Register (EMIOS_GFR)

The EMIOS_GFR is a read-only register that groups the FLAG bits from all channels. This organization improves interrupt handling on simpler devices. These bits are mirrors of the FLAG bits of each channel register (EMIOS_CSR) and flag bits in those channel registers cannot be cleared by accessing this ‘mirror’ register.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	F23	F22	F21	F20	F19	F18	F17	F16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x04															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	F15	F14	F13	F12	F11	F10	F9	F8	F7	F6	F5	F4	F3	F2	F1	F0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x04															

Figure 17-3. eMIOS Global Flag Register (EMIOS_GFR)

17.3.1.3 eMIOS Output Update Disable Register (EMIOS_OUDR)

The EMIOS_OUDR serves to disable transfers from the A2 to the A1 channel registers and from the B2 to the B1 channel registers when values are written to these registers, and the channel is running in modulus counter (MC) mode or an output mode.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	OU23	OU22	OU21	OU20	OU19	OU18	OU17	OU16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x08															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OU15	OU14	OU13	OU12	OU11	OU10	OU9	OU8	OU7	OU6	OU5	OU4	OU3	OU2	OU1	OU0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x08															

Figure 17-4. eMIOS Output Update Disable Register (EMIOS_OUDR)

Table 17-8. EMIOS_OUDR Field Descriptions

Bits	Name	Description
0–7	—	Reserved.
8–31	OU*n*	Channel *n* output update disable. When running in MC mode or an output mode, values are written to registers A2 and B2. OU*n* bits are used to disable transfers from registers A2 to A1 and B2 to B1. Each bit controls one channel. 0 Transfer enabled. Depending on the operating mode, transfer may occur immediately or in the next period. Unless stated otherwise, transfer occurs immediately. 1 Transfers disabled

17.3.1.4 eMIOS Channel A Data Register (EMIOS_CADR*n*)

Depending on the mode of operation, internal registers A1 or A2, used for matches and captures, can be assigned to address EMIOS_CADR*n*. Both A1 and A2 are cleared by reset. Table 17-9 summarizes the EMIOS_CADR*n* writing and reading accesses for all operating modes. For more information see section Section 17.4.4.4, "Modes of Operation of the Unified Channels."

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	A							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	UC*n* Base + 0x00															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	A															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	UC*n* Base + 0x00															

Figure 17-5. eMIOS Channel A Data Register (EMIOS_CADR*n*)

17.3.1.5 eMIOS Channel B Data Register (EMIOS_CBDR*n*)

Depending on the mode of operation, internal registers B1 or B2 can be assigned to address EMIOS_CBDR*n*. Both B1 and B2 are cleared by reset. Table 17-9 summarizes the EMIOS_CBDR*n* writing and reading accesses for all operating modes. For more information see section Section 17.4.4.4, "Modes of Operation of the Unified Channels."

NOTE

The EMIOS_CBDR*n* must not be read speculatively. For future compatibility, the TLB entry covering the EMIOS_CBDR*n* must be configured to be guarded.

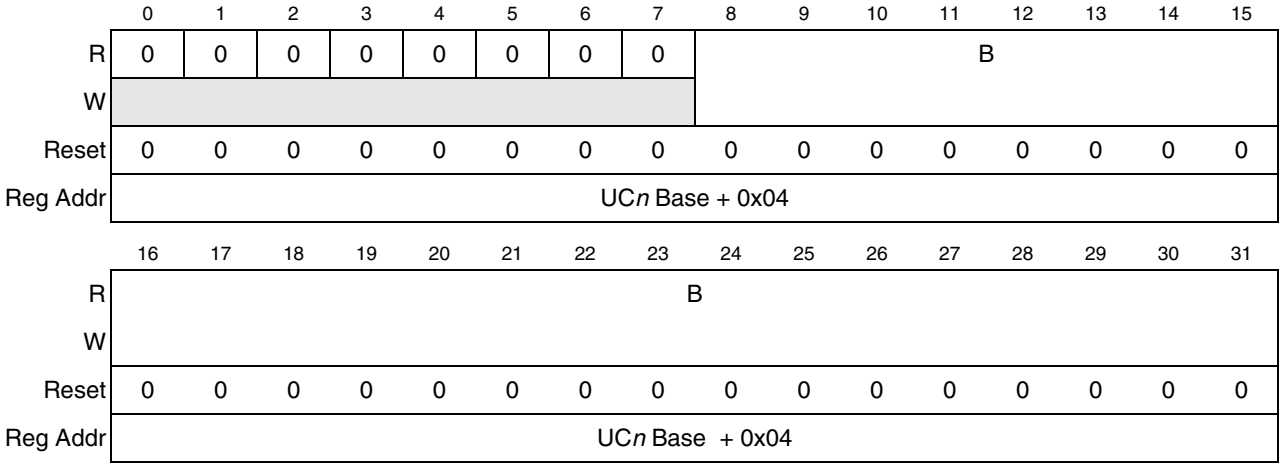


Figure 17-6. eMIOS Channel B Data Register (EMIOS_CBDR*n*)

Table 17-9. EMIOS_CADR*n* and EMIOS_CBDR*n* Value Assignments

Operating Mode	Register Access			
	Write	Read	Write	Read
GPIO	A1, A2	A1	B1,B2	B1
SAIC[1]	—	A2	B2	B2
SAOC[1]	A2	A1	B2	B2
IPWM	—	A2	—	B1
IPM	—	A2	—	B1
DAOC	A2	A1	B2	B1
PEA	A1	A2	—	B1
PEC[1]	A1	A1	B1	B1
QDEC[1]	A1	A1	B2	B2

Table 17-9. EMIOS_CADR*n* and EMIOS_CBDR*n* Value Assignments (continued)

WPTA	A1	A1	B1	B1
MC – Normal[1]	A2	A1	B2	B2
MC – Buffered	A2	A1	B2	B2
OPWFM – Normal	A2	A1	B2	B1
OPWFM – Buffered	A2	A1	B2	B1
OPWMC – Normal	A2	A1	B2	B1
OPWMC – Buffered	A2	A1	B2	B1
OPWM – Normal	A2	A1	B2	B1
OPWM – Buffered	A2	A1	B2	B1

1 In these modes, the register EMIOS_CBDR*n* is not used, but B2 can be accessed.

17.3.1.6 eMIOS Channel Counter Register (EMIOS_CCNTR*n*)

The EMIOS_CCNTR*n* contains the value of the internal counter. When GPIO mode is selected or the channel is frozen, the EMIOS_CCNTR*n* is readable and writable. For all others modes, the EMIOS_CCNTR*n* is a read-only register. When entering some operating modes, this register is automatically cleared (refer to section Section 17.4.4.4, "Modes of Operation of the Unified Channels," for details).

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	C							
W[1]																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	UC*n* Base + 0x08															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	C															
W[1]																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	UC*n* Base + 0x08															

1 In GPIO mode or freeze action, this register is writable.

Figure 17-7. eMIOS Channel Counter Register (EMIOS_CCNTR*n*)

17.3.1.7 eMIOS Channel Control Register (EMIOS_CCR*n*)

The eMIOS_CCR*n* enables the setting of several control parameters for a unified channel. Among these controls are the setting of a channel prescaler, channel mode selection, input trigger sensitivity and filtering, interrupt and DMA request enabling, and output mode control.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FREN	ODIS	ODISSL		UCPRE		UCPREN	DMA	0	IF				FCK	FEN	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	UC*n* Base + 0x0C															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	BSL		EDSEL	EDPOL	MODE						
W			FOR CMA	FOR CMB												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	UC*n* Base + 0x0C															

Figure 17-8. eMIOS Channel Control Register (EMIOS_CCR*n*)

Table 17-10. EMIOS_CCR*n* Field Description

Bits	Name	Description
0	FREN	Freeze enable. If set and validated by FRZ bit in EMIOS_MCR, freezes all registers values when in debug mode, allowing the MCU to perform debug functions. 0 Normal operation 1 Freeze UC registers values
1	ODIS	Output disable. Allows output disable in any output mode except GPIO. 0 The output pin operates normally 1 If the selected *output disable input* signal is asserted, the output pin goes to the complement of EDPOL for OPWFM, OPWFMB, and OPWMB modes, but the unified channel continues to operate normally; that is, it continues to produce FLAG and matches. When the selected *output disable input* signal is negated, the output pin operates normally.
2–3	ODISSL [0:1]	Output disable select. Selects one of the four *output disable input* signals. 00 *output disable input* 0 01 *output disable input* 1 10 *output disable input* 2 11 *output disable input* 3
4–5	UCPRE [0:1]	Prescaler. Selects the clock divider value for the unified channel internal prescaler, as shown below.

UCPRE[0:1]	Divide Ratio
00	1
01	2
10	3
11	4

Table 17-10. EMIOS_CCR*n* Field Description (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>6</td><td>UCPREN</td><td>Prescaler enable. Enables the prescaler counter.
0 Prescaler disabled (no clock) and prescaler counter is loaded with UCPREvalue
1 Prescaler enabled</td></tr>
<tr><td>7</td><td>DMA</td><td>Direct memory access. Selects if the FLAG generation will be used as an interrupt or as a DMA request.
0 FLAG assigned to Interrupt request
1 FLAG assigned to DMA request
Not all eMIOS channels support DMA, as shown below. The DMA bit should not be changed from its default value of 0 for any channel that does not support DMA.
<table>
<tr><th>eMIOS Channel</th><th>DMA = 0</th><th>DMA = 1</th></tr>
<tr><td>0</td><td>Interrupt</td><td>DMA request</td></tr>
<tr><td>1</td><td>Interrupt</td><td>DMA request</td></tr>
<tr><td>2</td><td>Interrupt</td><td>DMA request</td></tr>
<tr><td>3</td><td>Interrupt</td><td>DMA request</td></tr>
<tr><td>4</td><td>Interrupt</td><td>DMA request</td></tr>
<tr><td>5</td><td>Interrupt</td><td>Reserved</td></tr>
<tr><td>6</td><td>Interrupt</td><td>DMA request</td></tr>
<tr><td>7</td><td>Interrupt</td><td>DMA request</td></tr>
<tr><td>8</td><td>Interrupt</td><td>DMA request</td></tr>
<tr><td>9</td><td>Interrupt</td><td>DMA request</td></tr>
<tr><td>10</td><td>Interrupt</td><td>DMA request</td></tr>
<tr><td>11</td><td>Interrupt</td><td>DMA request</td></tr>
<tr><td>12</td><td>Interrupt</td><td>Reserved</td></tr>
<tr><td>13</td><td>Interrupt</td><td>Reserved</td></tr>
<tr><td>14</td><td>Interrupt</td><td>Reserved</td></tr>
<tr><td>15</td><td>Interrupt</td><td>Reserved</td></tr>
<tr><td>16</td><td>Interrupt</td><td>DMA request</td></tr>
<tr><td>17</td><td>Interrupt</td><td>DMA request</td></tr>
<tr><td>18</td><td>Interrupt</td><td>DMA request</td></tr>
<tr><td>19</td><td>Interrupt</td><td>DMA request</td></tr>
<tr><td>20</td><td>Interrupt</td><td>Reserved</td></tr>
<tr><td>21</td><td>Interrupt</td><td>Reserved</td></tr>
<tr><td>22</td><td>Interrupt</td><td>Reserved</td></tr>
<tr><td>23</td><td>Interrupt</td><td>Reserved</td></tr>
</table>
</td></tr>
</table>

Table 17-10. EMIOS_CCR*n* Field Description (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>8</td><td>—</td><td>Reserved.</td></tr>
<tr><td>9–12</td><td>IF
[0:3]</td><td>Input filter. Controls the programmable input filter, selecting the minimum input pulse width that can pass through the filter, as shown below. For output modes, these bits have no meaning.
<table>
<tr><th>IF[0:3][1]</th><th>Minimum input pulse width [filter clock periods]</th></tr>
<tr><td>0000</td><td>Bypassed[2]</td></tr>
<tr><td>0001</td><td>2 filter clock periods</td></tr>
<tr><td>0010</td><td>4 filter clock periods</td></tr>
<tr><td>0100</td><td>8 filter clock periods</td></tr>
<tr><td>1000</td><td>16 filter clock periods</td></tr>
<tr><td>all others</td><td>Reserved</td></tr>
</table>
1 Filter latency is 3 clock cycles.
2 The input signal is synchronized before arriving at the digital filter.</td></tr>
<tr><td>13</td><td>FCK</td><td>Filter clock select. Selects the clock source for the programmable input filter.
0 Prescaled clock
1 Main clock</td></tr>
<tr><td>14</td><td>FEN</td><td>FLAG enable. Allows the unified channel FLAG bit to generate an interrupt signal or a DMA request signal (The type of signal to be generated is defined by the DMA bit).
0 Disable (FLAG does not generate an interrupt or DMA request)
1 Enable (FLAG will generate an interrupt or DMA request)</td></tr>
<tr><td>15–17</td><td>—</td><td>Reserved.</td></tr>
<tr><td>18</td><td>FORCMA</td><td>Force match A. For output modes, the FORCMA bit is equivalent to a successful comparison on comparator A (except that the FLAG bit is not set). This bit is cleared by reset and is always read as zero. This bit is valid for every output operating mode which uses comparator A, otherwise it has no effect.
0 Has no effect
1 Force a match at comparator A
For input modes, the FORCMA bit is not used and writing to it has no effect.</td></tr>
<tr><td>19</td><td>FORCMB</td><td>Force match B. For output modes, the FORCMB bit is equivalent to a successful comparison on comparator B (except that the FLAG bit is not set). This bit is cleared by reset and is always read as zero. This bit is valid for every output operating mode which uses comparator B, otherwise it has no effect.
0 Has no effect
1 Force a match at comparator B
For input modes, the FORCMB bit is not used and writing to it has no effect.</td></tr>
<tr><td>20</td><td>—</td><td>Reserved.</td></tr>
</table>

Table 17-10. EMIOS_CCR*n* Field Description (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>21–22</td><td>BSL
[0:1]</td><td>Bus select. Used to select either one of the counter buses or the internal counter to be used by the unified channel.

<table><tr><th>BSL[0:1]</th><th>Selected Bus</th></tr><tr><td>00</td><td>All channels: counter bus[A]</td></tr><tr><td>01</td><td>Channels 0 to 7: counter bus[B]
Channels 8 to 15: counter bus[C]
Channels 16 to 23: counter bus[D]</td></tr><tr><td>10</td><td>Reserved</td></tr><tr><td>11</td><td>All channels: internal counter (see Note)</td></tr></table>
Note: In certain modes the internal counter is used internally and therefore cannot be used as the channel time base.</td></tr>
<tr><td>23</td><td>EDSEL</td><td>Edge selection bit. For input modes, the EDSEL bit selects whether the internal counter is triggered by both edges of a pulse or just by a single edge as defined by the EDPOL bit. When not shown in the mode of operation description, this bit has no effect.
0 Single edge triggering defined by the EDPOL bit
1 Both edges triggering

For GPIO input mode, the EDSEL bit selects if a FLAG can be generated.
0 A FLAG is generated as defined by the EDPOL bit
1 No FLAG is generated

For SAOC mode, the EDSEL bit selects the behavior of the output flip-flop at each match.
0 The EDPOL value is transferred to the output flip-flop
1 The output flip-flop is toggled</td></tr>
</table>

Table 17-10. EMIOS_CCR*n* Field Description (continued)

Bits	Name	Description
24	EDPOL	Edge polarity. For input modes (except QDEC and WPTA mode), the EDPOL bit asserts which edge triggers either the internal counter or an input capture or a FLAG. When not shown in the mode of operation description, this bit has no affect. 0 Trigger on a falling edge 1 Trigger on a rising edge For WPTA mode, the internal counter is used as a time accumulator and counts up when the input gating signal has the same polarity of EDPOL bit. 0 Counting occurs when the input gating signal is low 1 Counting occurs when the input gating signal is high For QDEC (MODE[6] cleared), the EDPOL bit selects the count direction according to *direction* signal (UC*n* input). 0 Counts down when UC*n* is asserted 1 Counts up when UC*n* is asserted NOTE: UC[n-1] EDPOL bit selects which edge clocks the internal counter of UC*n* 0 Trigger on a falling edge 1 Trigger on a rising edge For QDEC (MODE[6] set), the EDPOL bit selects the count direction according to the phase difference. 0 Internal counter decrements if phase_A is ahead phase_B signal 1 Internal counter increments if phase_A is ahead phase_B signal NOTE: In order to operate properly, EDPOL bit must contain the same value in UC*n* and UC[n-1] For output modes, the EDPOL bit is used to select the logic level on the output pin. 0 A match on comparator A clears the output flip-flop, while a match on comparator B sets it 1 A match on comparator A sets the output flip-flop, while a match on comparator B clears it
25–31	MODE [0:6]	Mode selection. Selects the mode of operation of the unified channel, as shown in Table 17-11.

Table 17-11. Unified Channel MODE Bits

MODE0:6]	Unified Channel Mode of Operation
0000000	General purpose input/output mode (input)
0000001	General purpose input/output mode (output)
0000010	Single action input capture
0000011	Single action output compare
0000100	Input pulse width measurement
0000101	Input period measurement
0000110	Double action output compare (with FLAG set on the second match)
0000111	Double action output compare (with FLAG set on both match)

Table 17-11. Unified Channel MODE Bits (continued)

MODE0:6]	Unified Channel Mode of Operation
0001000	Pulse/edge accumulation (continuous)
0001001	Pulse/edge accumulation (single shot)
0001010	Pulse/edge counting (continuous)
0001011	Pulse/edge counting (single shot)
0001100	Quadrature decode (for count and direction encoders type)
0001101	Quadrature decode (for phase_A and phase_B encoders type)
0001110	Windowed programmable time accumulation
0001111	Reserved
0010000	Modulus counter (up counter, internal clock source)
0010001	Modulus counter (up counter, external clock source)
0010010– 0010011	Reserved
0010100	Modulus counter (up/down counter, no change in counter direction upon match of input counter and register B1, internal clock source)
0010101	Modulus counter (up/down counter, no change in counter direction upon match of input counter and register B1, external clock source)
0010110	Modulus counter (up/down counter, change in counter direction upon match of input counter and register B1 and sets the FLAG, internal clock source)
0010111	Modulus counter (up/down counter, change in counter direction upon match of input counter and register B1 and sets the FLAG, external clock source)
0011000	Output pulse width and frequency modulation (FLAG set at match of internal counter and comparator B, immediate update)
0011001	Output pulse width and frequency modulation (FLAG set at match of internal counter and comparator B, next period update)
0011010	Output pulse width and frequency modulation (FLAG set at match of internal counter and comparator A or comparator B, immediate update)
0011011	Output pulse width and frequency modulation (FLAG set at match of internal counter and comparator A or comparator B, next period update)
0011100	Center aligned output pulse width modulation (FLAG set in trailing edge, trailing edge dead-time)
0011101	Center aligned output pulse width modulation (FLAG set in trailing edge, leading edge dead-time)

Table 17-11. Unified Channel MODE Bits (continued)

MODE0:6]	Unified Channel Mode of Operation
0011110	Center aligned output pulse width modulation (FLAG set in both edges, trailing edge dead-time)
0011111	Center aligned output pulse width modulation (FLAG set in both edges, leading edge dead-time)
0100000	Output pulse width modulation (FLAG set at match of internal counter and comparator B, immediate update)
0100001	Output pulse width modulation (FLAG set at match of internal counter and comparator B, next period update)
0100010	Output pulse width modulation (FLAG set at match of internal counter and comparator A or comparator B, immediate update)
0100011	Output pulse width modulation (FLAG set at match of internal counter and comparator A or comparator B, next period update)
1100100– 1111111	Reserved
1010000	Modulus up counter, buffered, internal clock
1010001	Modulus up counter, buffered, external clock
1010010– 1010001	Reserved
1010100	Modulus up/down counter, buffered (FLAG set on one event, internal clock)
1010101	Modulus up/down counter, buffered (FLAG set on one event, external clock)
1010110	Modulus up/down counter, buffered (FLAG set on both events, internal clock)
1010111	Modulus up/down counter, buffered (FLAG set on both events, external clock)
1011000	Output pulse width and frequency modulation, buffered (FLAG set at match of internal counter and comparator B)
1011001	Reserved
1011010	Output pulse width and frequency modulation, buffered (FLAG set at match of internal counter and comparator A or comparator B)
1011011	Reserved
1011100	Center aligned output pulse width modulation, buffered (FLAG set on trailing edge, trailing edge dead-time)
1011101	Center aligned output pulse width modulation, buffered (FLAG set on trailing edge, leading edge dead-time)
1011110	Center aligned output pulse width modulation, buffered (FLAG set on both edges, trailing edge dead-time)
1011111	Center aligned output pulse width modulation, buffered (FLAG set on both edges, leading edge dead-time)

Table 17-11. Unified Channel MODE Bits (continued)

MODE0:6]	Unified Channel Mode of Operation
1100000	Output pulse width modulation, buffered (FLAG set on second match)
1100001	Reserved
1100010	Output pulse width modulation, buffered (FLAG set on both matches)

17.3.1.8 eMIOS Channel Status Register (EMIOS_CSR*n*)

EMIOS_CSR*n* reflects the status of the UC input/output signals and the overflow condition of the internal counter, as well as the occurrence of a trigger event.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OVR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	UC*n* Base + 0x10															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OVFL	0	0	0	0	0	0	0	0	0	0	0	0	UCIN	UCOUT	FLAG
W	w1c															w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	UC*n* Base + 0x10															

Figure 17-9. eMIOS Channel Status Register (EMIOS_CSR*n*)

Table 17-12. EMIOS_CSR*n* Field Descriptions

Bits	Name	Description
0	OVR	Overrun. Indicates that FLAG generation occurred when the FLAG bit was already set. This bit can be cleared by writing a 1 to it or by clearing the FLAG bit. 0 Overrun has not occurred 1 Overrun has occurred
1–15	—	Reserved.
16	OVFL	Overflow. Indicates that an overflow has occurred in the internal counter. OVFL is cleared by writing a 1 to it. 0 No overflow 1 An overflow had occurred
17–28	—	Reserved.
29	UCIN	Unified channel input pin. Reflects the input pin state after being filtered and synchronized.

Table 17-12. EMIOS_CSR*n* Field Descriptions (continued)

Bits	Name	Description
30	UCOUT	Unified channel output pin. The UCOUT bit reflects the output pin state.
31	FLAG	FLAG. Set when an input capture or a match event in the comparators occurred. This bit is cleared by writing a 1 to it. 0 FLAG cleared 1 FLAG set event has occurred **Note:** When EMIOS_CCR[DMA] bit is set, the FLAG bit is cleared by the eDMA controller.

17.4 Functional Description

The eMIOS provides independent channels (UC) that can be configured and accessed by the MPC5553/MPC5554. Four time bases can be shared by the channels through four counter buses and each unified channel can generate its own time base. Optionally, the counter A bus can be driven by an external time base from the eTPU imported through the STAC interface.

NOTE

Counter bus A can be driven by unified channel 23 or by the STAC bus. Counter bus B, C, and D are driven by unified channels 0, 8, and 16, respectively. Counter bus A can be shared among all unified channels. UCs 0 to 7, 8 to 15, and 16 to 23 can share counter buses B, C, and D, respectively.

The following four components of the MPC5553/MPC5554 eMIOS are discussed below:

- Bus interface unit
- STAC client submodule
- Global clock prescaler
- Unified channels and their modes of operation

17.4.1 Bus Interface Unit (BIU)

The bus interface unit provides the interface between the internal bus and the slave interface, allowing communication among all submodules and the slave interface.

The BIU allows 8-, 16-, and 32-bit accesses. They are performed over a 32-bit data bus in a single cycle clock.

17.4.1.1 Effect of Freeze on the BIU

When the FRZ bit in the EMIOS_MCR is set and the module is in debug mode, the operation of the BIU is not affected.

17.4.2 STAC Client Submodule

The shared time and angle count (STAC) bus provides access to one external time base, imported from the STAC bus to the eMIOS unified channels. The eTPU module's time bases and angle count can be exported and/or imported through the STAC client submodule interface. Time bases and/or angle information of either eTPU engine can be exported to the other eTPU engine and to the eMIOS module, which is only a

STAC client. There are restrictions on engine export/import targets: one engine cannot export from or import to itself, nor can it import time base and/or angle count if in angle mode.

The MPC5553/MPC5554 STAC server identification assignment is shown in Table 17-13. The time slot assignment is fixed, so only time bases running at system clock ÷ 4 or slower can be integrally exported. The STAC client submodule runs with the system clock, and its time slot timing is synchronized with the eTPU timing on reset. The time slot sequence is 0-1-2-3, such that they are alternated between engines 1 and 2.

Table 17-13. STAC Client Submodule Server Slot Assignment

Engine	Time Base	Server ID
1	TCR1	0
1	TCR2	2
2	TCR1	1
2	TCR2	3

Figure 17-10 provides a block diagram for the STAC client submodule.

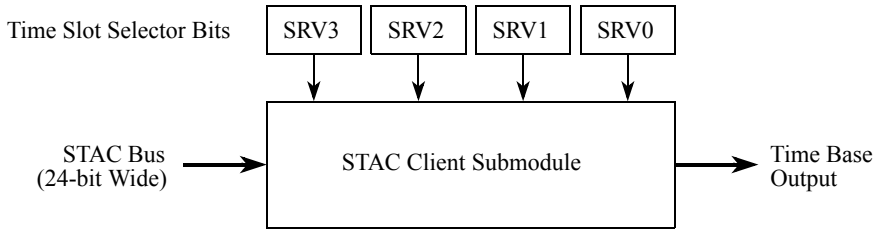
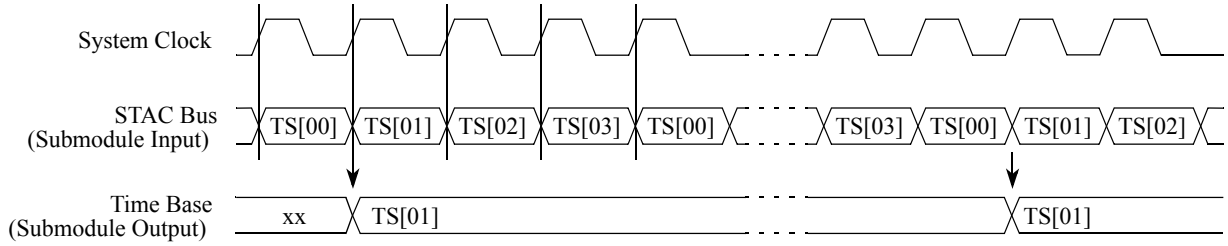


Figure 17-10. STAC Client Submodule Block Diagram

Bits SRV[0:3] in register EMIOS_MCR, selects the desired time slot of the STAC bus to be output. Figure 17-11 shows a timing diagram for the STAC client submodule.



Note: In this case, SRV bits were set to capture TS[01].

Figure 17-11. Timing Diagram for the STAC Bus and STAC Client Submodule Output

Every time the selected time slot changes, the STAC Client Submodule output is updated.

17.4.2.1 Effect of Freeze on the STAC Client Submodule

When the FRZ bit in the EMIOS_MCR is set and the module is in debug mode, the operation of the STAC client submodule submodule is not affected; that is, there is no freeze function in this submodule.

17.4.3 Global Clock Prescaler Submodule (GCP)

The GCP divides the system clock to generate a clock for the clock prescalers of the unified channels. The system clock is prescaled by the value defined in Table 17-7 according to the GPRE[0:7] bits in the EMIOS_MCR. The output is clocked every time the counter overflows. Counting is enabled by setting EMIOS_MCR[GPREN]. The counter can be stopped at any time by clearing this bit, thereby stopping the internal counter in all the unified channels.

17.4.3.1 Effect of Freeze on the GCP

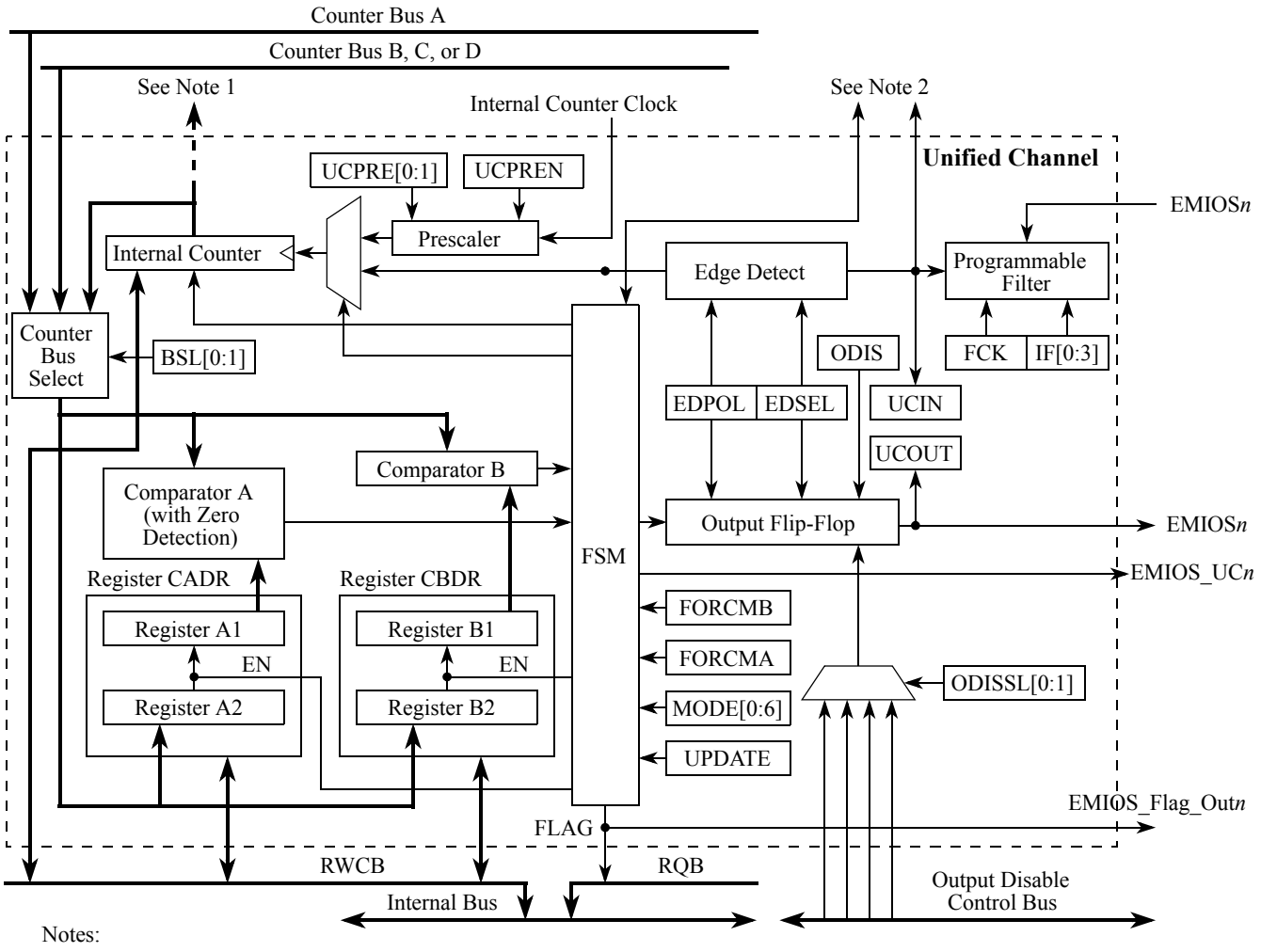
When the FRZ bit in the EMIOS_MCR is set and the module is in debug mode, the operation of GCP submodule is not affected; that is, there is no freeze function in this submodule.

17.4.4 Unified Channel (UC)

Figure 17-12 shows the unified channel block diagram. Each unified channel consists of the following:

- Counter bus selector that selects the time base to be used by the channel for all timing functions
- Programmable clock prescaler
- Two double buffered data registers A and B that allow up to two input capture and/or output compare events to occur before software intervention is needed.
- Two comparators (equal only) A and B that compare the selected counter bus with the value in the data registers
- Internal counter that can be used as a local time base or to count input events
- Programmable input filter that ensures that only valid pin transitions are received by a channel
- Programmable input edge detector that detects rising, falling, or both edges
- Output flip-flop that holds the logic level to be applied to the output pin
- eMIOS status and control registers
- Output disable input selector that selects the output disable input signal to be used as the unified channel output disable
- Control state machine (FSM)

The major components and functions of the MPC5553/MPC5554 unified channels are discussed in Section 17.4.4.1, “Programmable Input Filter (PIF) through Section 17.4.4.4, “Modes of Operation of the Unified Channels.”



Notes:

1. Counter bus A can be driven by either the STAC bus or channel 23. Refer to EMIOS_MCR[ETB]. Channel 0 drives counter bus B, channel 8 drives counter bus C and channel 16 drives counter bus D. Counter bus B can be selected as the counter for channels 0-7, counter bus C for channels 8-15, and counter bus D for channels 16-23. Refer to Figure 16-1 and EMIOS_CCR*n*[BS].
2. Goes to the finite state machine of the UC[n-1]. These signals are used for QDEC mode.

Figure 17-12. Unified Channel Block Diagram

17.4.4.1 Programmable Input Filter (PIF)

The PIF ensures that only valid input pin transitions are received by the unified channel edge detector. A block diagram of the PIF is shown in Figure 17-13.

The PIF is a 5-bit programmable up counter that is incremented by the selected clock source, according to bits IF[0:3] in EMIOS_CCR*n*. The clock source is selected by the EMIOS_CCR*n*[FCK] bit.

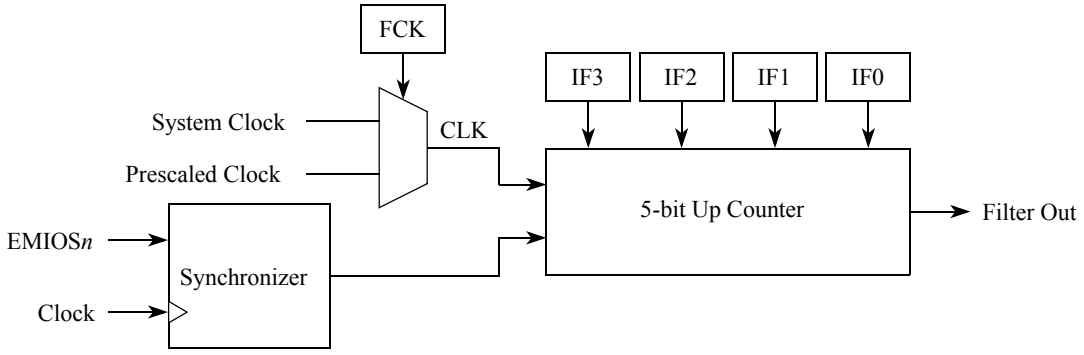


Figure 17-13. Programmable Input Filter Submodule Diagram

The input signal is synchronized by the system clock. When a state change occurs in this signal, the 5-bit counter starts counting up. As long as the new state is stable on the pin, the counter continues incrementing. If a counter overflows occurs, the new pin value is validated. In this case, it is transmitted as a pulse edge to the edge detector. If the opposite edge appears on the pin before validation (overflow), the counter is reset. At the next synchronized pin transition, the counter starts counting again. Any pulse that is shorter than a full range of the masked counter is regarded as a glitch, and it is not passed on to the edge detector. A timing diagram of the input filter is shown in Figure 17-14.

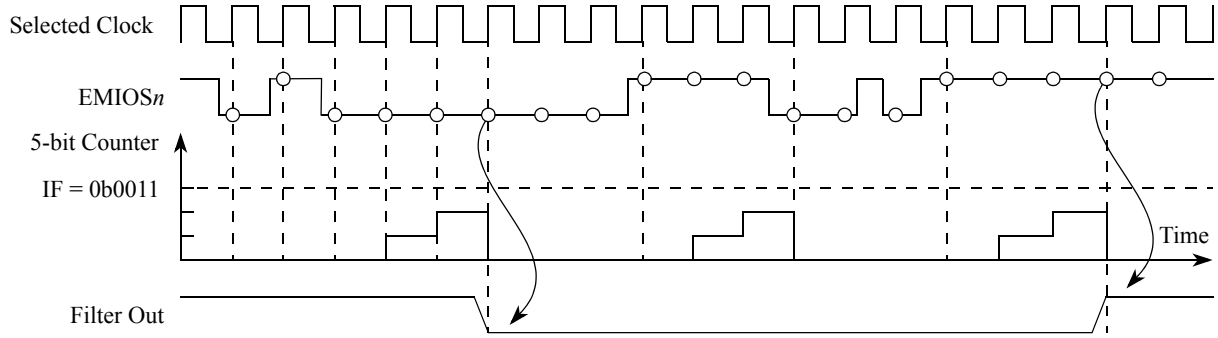


Figure 17-14. Programmable Input Filter Example

17.4.4.2 Clock Prescaler (CP)

A unified channel has a clock prescaler (CP) that divides the global clock prescaler (refer to Section 17.4.3, "Global Clock Prescaler Submodule (GCP)") output signal to generate a clock enable for the internal counter of the unified channel. It is a programmable 2-bit down counter. The global clock prescaler submodule (GCP) output signal is prescaled by the value defined in Table 17-10 according to the UCPRE[0:1] bits in the EMIOS_CCR*n*. The output is clocked every time the counter reaches zero. Counting is enabled by setting the UCPREN bit in the EMIOS_CCR*n*. The counter can be stopped at any time by clearing this bit, thereby stopping the internal counter in the unified channel.

17.4.4.3 Effect of Freeze on the Unified Channel

When in debug mode and the EMIOS_MCR[FRZ] bit and the EMIOS_CCR*n*[FREN] bit are both set, the internal counter and the unified channel's capture and compare functions are halted. The UC is frozen in its current state.

During freeze, all registers are accessible. When the unified channel is operating in an output mode, the force match functions remain available, allowing the software to force the output to the desired level.

Note that for input modes, any input events that may occur while the channel is frozen are ignored.

When exiting debug mode or freeze enable bit is cleared (FRZ in the EMIOS_MCR or FREN in the EMIOS_CCR*n*) the channel actions resume.

17.4.4.4 Modes of Operation of the Unified Channels

The mode of operation of a unified channel is determined by the mode select bits MODE[0:6] in the EMIOS_CCR*n*. See Table 17-11 for details.

When entering an output mode (except for GPIO mode), the output flip-flop is set to the complement of the EDPOL bit in the EMIOS_CCR*n*.

Because the internal counter EMIOS_CCNTR*n* continues to run in all modes (except for GPIO mode), it is possible to use this counter as the UC time base unless it (the internal counter) is a required resource in the operation of the selected mode.

To provide smooth waveform generation while allowing A and B registers to be changed on the fly, the double-buffered modes MCB, OPWFMB, OPWMB, and OPWMCB are provided (beginning at Section 17.4.4.4.15, "Modulus Counter, Buffered Mode (MCB) (MPC5553 Only)"). In these modes the A and B registers are double buffered. Descriptions of the double-buffered modes are presented separately, because there are several basic differences from the single-buffered MC, OPWFM, OPWM, and OPWMC modes.

Section 17.4.4.4.2, "Single Action Input Capture Mode (SAIC)" through Section 17.4.4.4.18, "Output Pulse Width Modulation, Buffered Mode (OPWMB) (MPC5553 Only)" below explain in detail the unified channels' modes of operation.

17.4.4.4.1 General Purpose Input/Output Mode (GPIO)

In GPIO mode, all input capture and output compare functions of the UC are disabled, the internal counter (EMIOS_CCNTRn register) is cleared and disabled. All control bits remain accessible. In order to prepare the UC for a new operating mode, writing to registers EMIOS_CADR*n* or EMIOS_CBDR*n* stores the same value in registers A1/A2 or B1/B2, respectively.

MODE[6] bit selects between input (MODE[6] = 0) and output (MODE[6] = 1) modes.

It is required that when changing MODE[0:6], the application software goes to GPIO mode first in order to reset the UC's internal functions properly. Failure to do this can lead to invalid and unexpected output compares and input capture results, or can cause the FLAGs to be set incorrectly.

In GPIO input mode, the FLAG generation is determined according to EDPOL and EDSEL bits and the input pin status can be determined by reading the UCIN bit.

In GPIO output mode, the unified channel is used as a single output port pin and the value of the EDPOL bit is permanently transferred to the output flip-flop.

NOTE

The GPIO modes provided in the eMIOS are particularly useful as interim modes when certain other eMIOS modes are being dynamically configured and enabled or disabled during the execution of the application. For normal GPIO function on the eMIOS pins, it is recommended that the SIU be used to configure those pins as system GPIO. See Section 6.2.1.3, "General-Purpose I/O Pins (GPIO[0:210])".

17.4.4.4.2 Single Action Input Capture Mode (SAIC)

In SAIC mode, when a triggering event occurs on the input pin, the value on the selected time base is captured into register A2. At the same time, the FLAG bit is set to indicate that an input capture has occurred. Register EMIOS_CADR*n* returns the value of register A2.

The input capture is triggered by a rising, falling or either edges in the input pin, as configured by EDPOL and EDSEL bits in EMIOS_CCR*n*.

Figure 17-15 shows how the unified channel can be used for input capture.

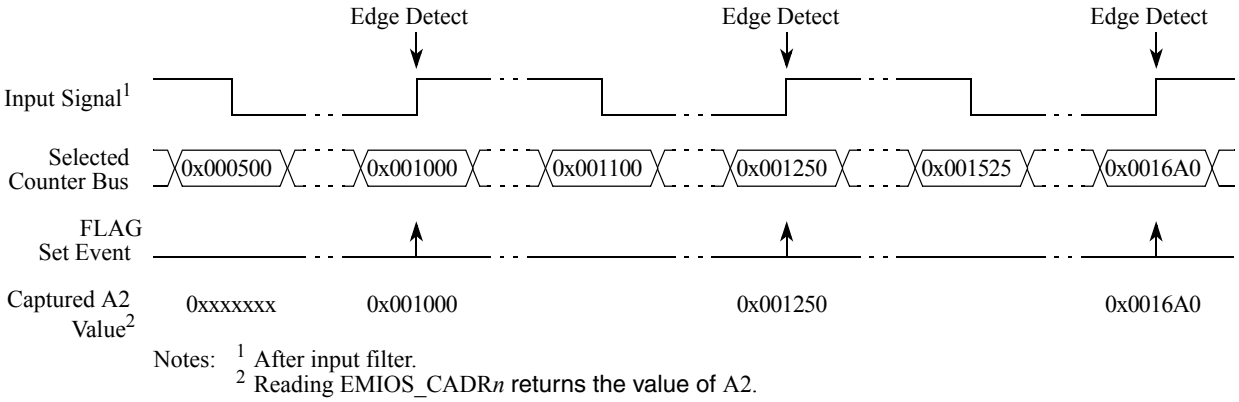


Figure 17-15. Single Action Input Capture Example

17.4.4.4.3 Single Action Output Compare Mode (SAOC)

In SAOC mode a match value is loaded in register A2 and then transferred to register A1 to be compared with the selected time base. When a match occurs, the EDSEL bit selects if the output flip-flop is toggled or if the value in EDPOL is transferred to it. At the same time, the FLAG bit is set to indicate that the output compare match has occurred. Writing to register EMIOS_CADR*n* stores the value in register A2 and reading to register EMIOS_CADR*n* returns the value of register A1.

An output compare match can be simulated in software by setting the FORCMA bit in EMIOS_CCR*n*. In this case, the FLAG bit is not set.

Figure 17-16 and Figure 17-17 show how the unified channel can be used to perform a single output compare with EDPOL value being transferred to the output flip-flop and toggling the output flip-flop at each match, respectively.

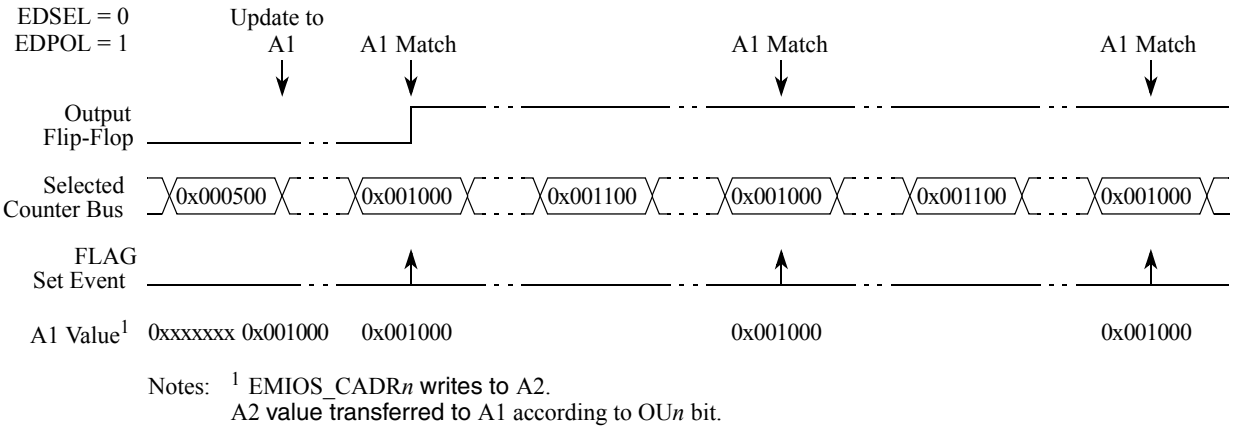


Figure 17-16. SAOC Example with EDPOL Value Transferred to the Output Flip-flop

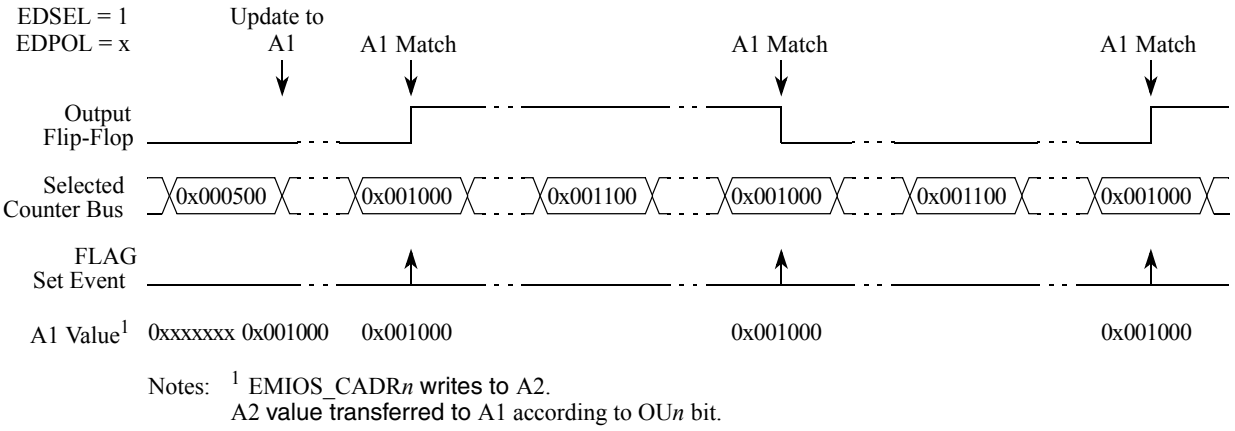


Figure 17-17. SAOC Example Toggling the Output Flip-flop

17.4.4.4.4 Input Pulse Width Measurement Mode (IPWM)

The IPWM mode allows the measurement of the width of a positive or negative pulse by capturing the leading edge on register B1 and the trailing edge on register A2. Successive captures are done on consecutive edges of opposite polarity. The leading edge sensitivity (that is, pulse polarity) is selected by EDPOL bit in the EMIOS_CCR*n*. Registers EMIOS_CADR*n* and EMIOS_CBDR*n* return the values in register A2 and B1, respectively.

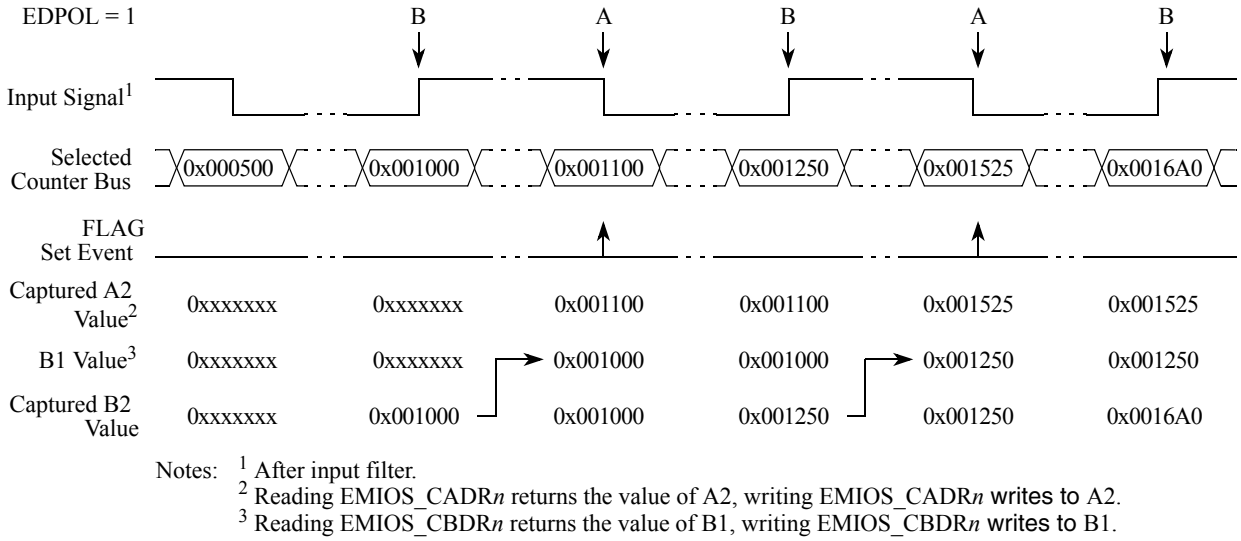
The capture function of register A2 remains disabled until the first leading edge triggers the first input capture on register B2. When this leading edge is detected, the count value of the selected time base is latched into register B2; the FLAG bit is not set. When the trailing edge is detected, the count value of the selected time base is latched into register A2 and, at the same time, the FLAG bit is set and the content of register B2 is transferred to register B1.

If subsequent input capture events occur while the corresponding FLAG bit is set, registers A2 and B1 will be updated with the latest captured values and the FLAG will remain set. Registers EMIOS_CADR*n* and EMIOS_CBDR*n* return the value in registers A2 and B1, respectively.

In order to guarantee coherent access, reading EMIOS_CADR*n* disables transfers between B2 and B1 until reading EMIOS_CBDR*n*. After that, transfer is re-enabled.

The input pulse width is calculated by subtracting the value in B1 from A2.

Figure 17-18 shows how the unified channel can be used for input pulse width measurement.



Notes: [1] After input filter.
[2] Reading EMIOS_CADR*n* returns the value of A2, writing EMIOS_CADR*n* writes to A2.
[3] Reading EMIOS_CBDR*n* returns the value of B1, writing EMIOS_CBDR*n* writes to B1.

Figure 17-18. Input Pulse Width Measurement Example

17.4.4.4.5 Input Period Measurement Mode (IPM)

The IPM mode allows the measurement of the period of an input signal by capturing two consecutive rising edges or two consecutive falling edges. Successive input captures are done on consecutive edges of the same polarity. The edge polarity is defined by the EDPOL bit in the EMIOS_CCR*n*.

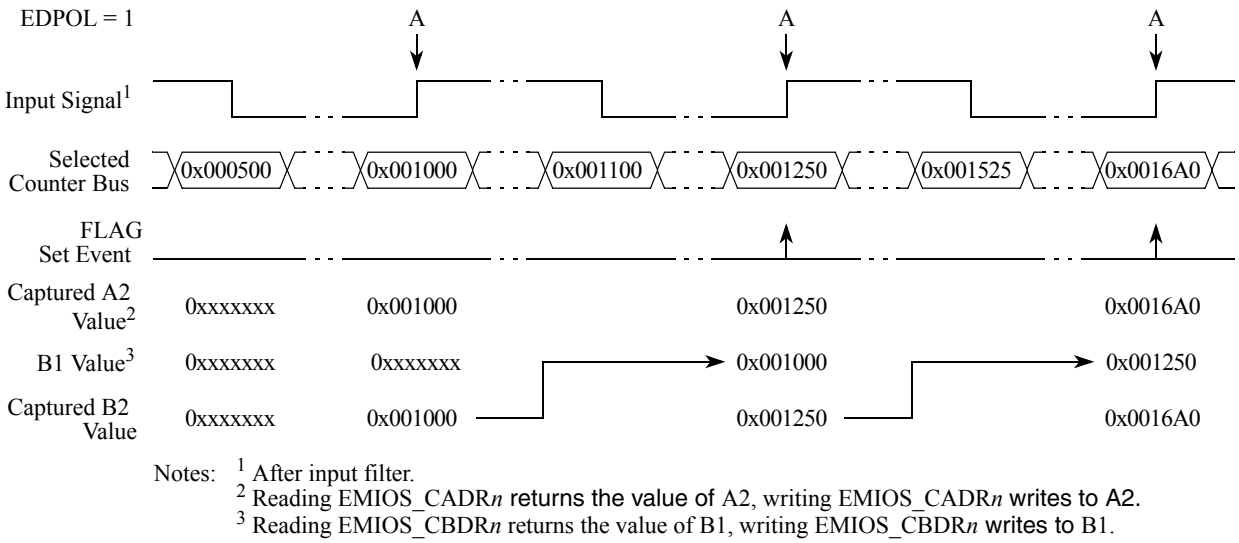
When the first edge of selected polarity is detected, the selected time base is latched into the registers A2 and B2, and the data previously held in register B2 is transferred to register B1. On this first capture the FLAG line is not set, and the values in registers B1 is meaningless. On the second and subsequent captures, the FLAG line is set and data in register B2 is transferred to register B1.

When the second edge of the same polarity is detected, the counter bus value is latched into registers A2 and B2, the data previously held in register B2 is transferred to data register B1, and the FLAG bit is set to indicate the start and end points of a complete period have been captured. This sequence of events is repeated for each subsequent capture. Registers EMIOS_CADR*n* and EMIOS_CBDR*n* return the values in register A2 and B1, respectively.

In order to guarantee coherent access, reading EMIOS_CADR*n* disables transfers between B2 and B1 until reading EMIOS_CBDR*n* register, then any pending transfer is re-enabled.

The input pulse period is calculated by subtracting the value in B1 from A2.

Figure 17-19 shows how the unified channel can be used for input period measurement.



Notes: [1] After input filter.
[2] Reading EMIOS_CADR*n* returns the value of A2, writing EMIOS_CADR*n* writes to A2.
[3] Reading EMIOS_CBDR*n* returns the value of B1, writing EMIOS_CBDR*n* writes to B1.

Figure 17-19. Input Period Measurement Example

17.4.4.4.6 Double Action Output Compare Mode (DAOC)

In the DAOC mode the leading and trailing edges of the variable pulse width output are generated by matches occurring on comparators A and B, respectively.

When the DAOC mode is first selected (coming from GPIO mode) both comparators are disabled. Comparators A and B are enabled by updating registers A1 and B1 respectively and remain enabled until a match occurs on that comparator, when it is disabled again. In order to update registers A1 and B1, a write to A2 and B2 must occur and the EMIOS_CCR*n*[ODIS] bit must be cleared.

The output flip-flop is set to the value of EMIOS_CCR*n*[EDPOL] when a match occurs on comparator A and to the complement of EDPOL when a match occurs on comparator B.

MODE[6] controls if the EMIOS_CSR*n*[FLAG] is set on both matches or just on the second match (see Table 17-11 for details).

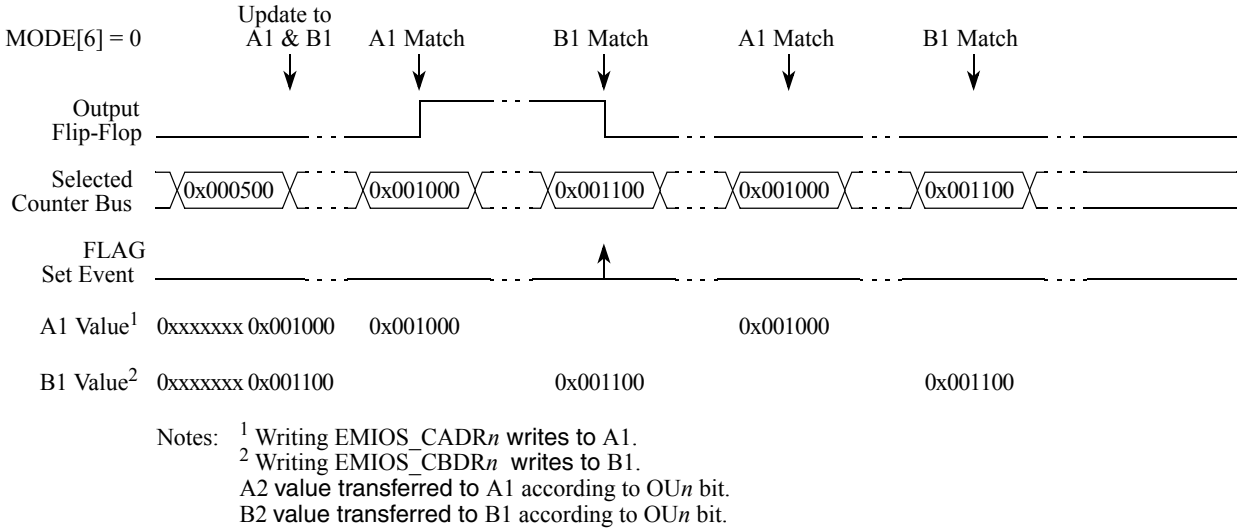
If subsequent enabled output compares occur on registers A1 and B1, pulses will continue to be generated, regardless of the state of the FLAG bit.

At any time, the EMIOS_CCR*n*[FORCMA] and EMIOS_CCR*n*[FORCMB] bits allow the software to force the output flip-flop to the level corresponding to a comparison event in comparator A or B, respectively. Note that the FLAG bit is not affected by these forced operations.

NOTE

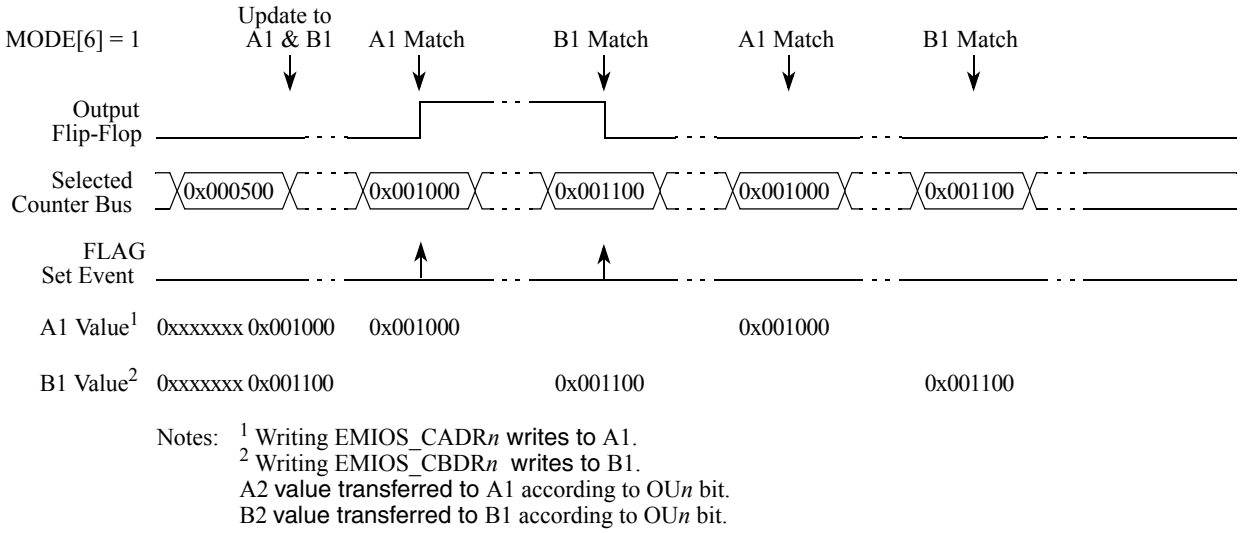
If both registers (A1 and B1) are loaded with the same value, the unified channel behaves as if a single match on comparator B had occurred; that is, the output flip-flop will be set to the complement of EDPOL bit and the FLAG bit is set.

Figure 17-20 and Figure 17-21 show how the unified channel can be used to generate a single output pulse with FLAG bit being set on the second match or on both matches, respectively.



Notes: [1] Writing EMIOS_CADR*n* writes to A1.
[2] Writing EMIOS_CBDR*n* writes to B1.
A2 value transferred to A1 according to OU*n* bit.
B2 value transferred to B1 according to OU*n* bit.

Figure 17-20. Double Action Output Compare with FLAG Set on the Second Match



Notes: [1] Writing EMIOS_CADR*n* writes to A1.
[2] Writing EMIOS_CBDR*n* writes to B1.
A2 value transferred to A1 according to OU*n* bit.
B2 value transferred to B1 according to OU*n* bit.

Figure 17-21. Double Action Output Compare with FLAG Set on Both Matches

17.4.4.4.7 Pulse/Edge Accumulation Mode (PEA)

The PEA mode returns the time taken to detect a desired number of input events. MODE[6] bit selects between continuous or single shot operation.

After writing to register A1, the internal counter is cleared on the first input event, ready to start counting input events and the selected timebase is latched into register B2. On the match between the internal counter and register A1, a counter bus capture is triggered to register A2 and B2. The data previously held in register B2 is transferred to register B1 and the FLAG bit is set to indicate that an event has occurred. The desired time interval can be determined subtracting register B1 from A2. Registers EMIOS_CADR*n* and EMIOS_CBDR*n* return the values in register A2 and B1, respectively.

In order to guarantee coherent access, reading EMIOS_CADR*n* disables transfers between B2 and B1 until reading EMIOS_CBDR*n* register, then any pending transfer is re-enabled.

Triggering of the counter clock (input event) is done by a rising or falling edge or both edges on the input pin. The polarity of the triggering edge is selected by the EDSEL and EDPOL bits in EMIOS_CCR*n*.

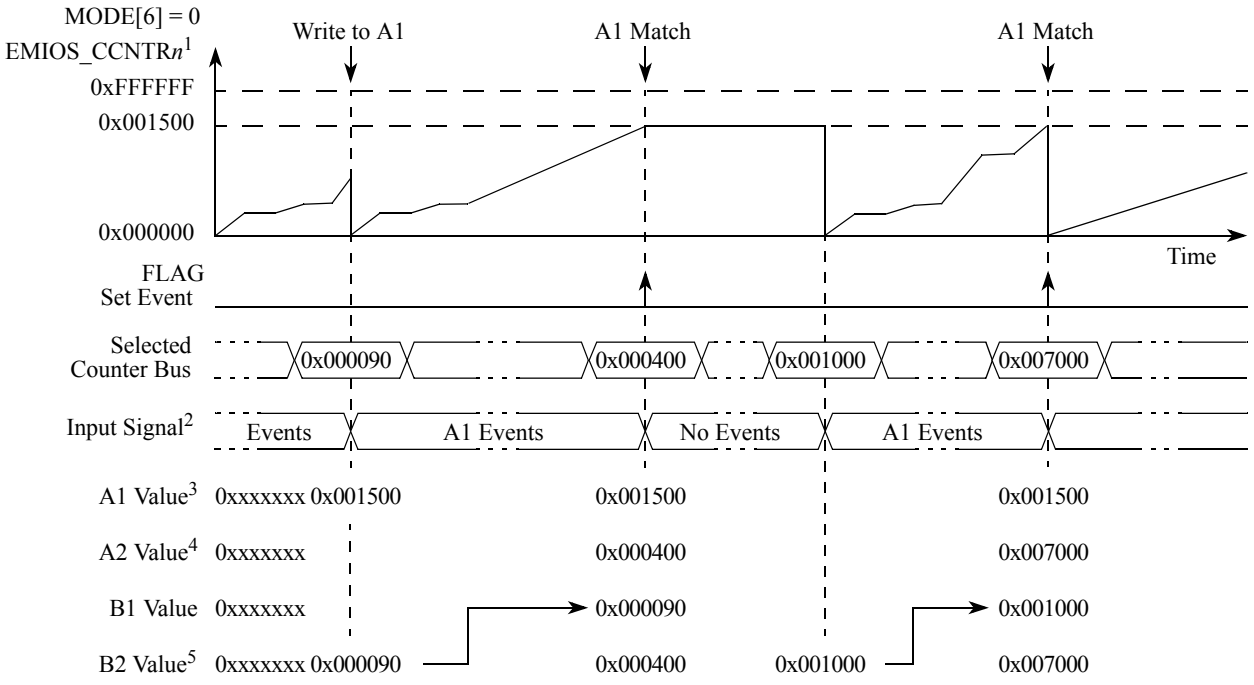
For continuous operating mode (MODE[6] cleared), the counter is cleared on the next input event after a FLAG generation and continues to operate as described above.

For single shot operation (MODE[6] set), the counter is not cleared or incremented after a FLAG generation, until a new writing operation to register A is performed.

NOTE

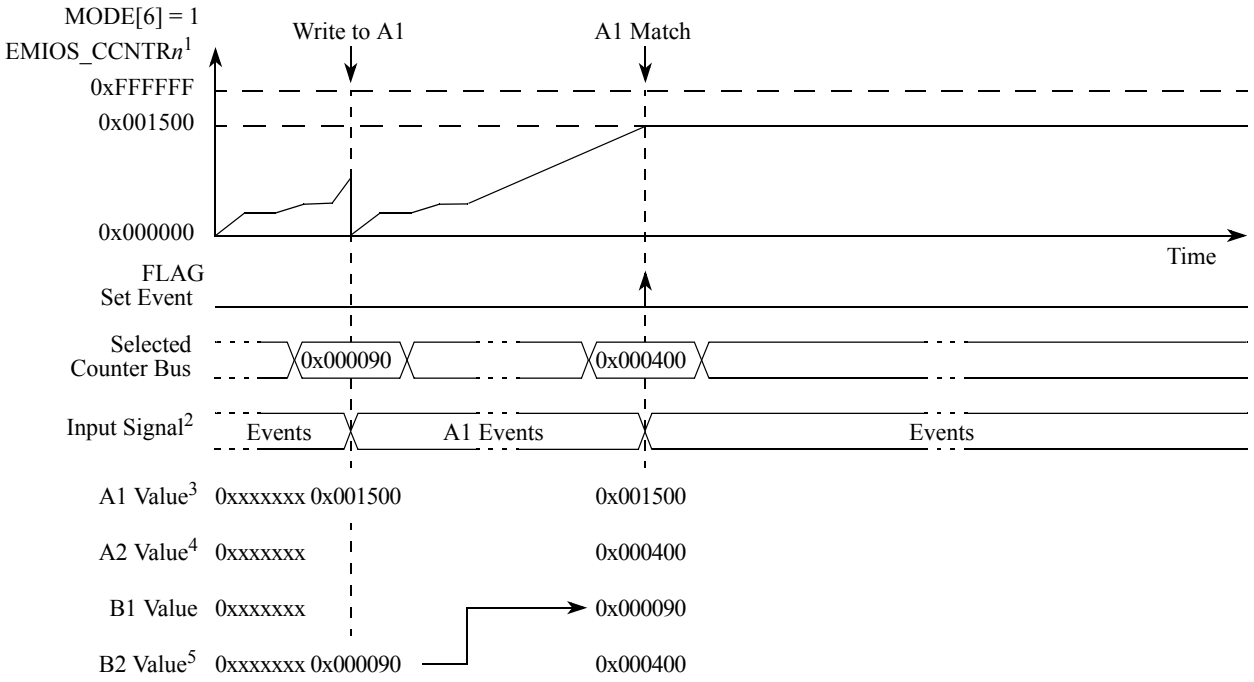
The FORCMA and FORCMB bits have no effect when the unified channel is configured for PEA mode.

Figure 17-22 and Figure 17-23 show how the unified channel can be used for continuous and single shot pulse/edge accumulation mode.



Notes: [1] Cleared on the first input event after writing to register A1.
[2] After input filter.
[3] Writing EMIOS_CADR*n* writes to A1.
[4] Reading EMIOS_CADR*n* returns the value of A2 .
[5] Reading EMIOS_CBDR*n* returns the value of B1.

Figure 17-22. Pulse/Edge Accumulation Continuous Mode Example



Notes: [1] Cleared on the first input event after writing to register A1.
[2] After input filter.
[3] Writing EMIOS_CADR*n* writes to A1.
[4] Reading EMIOS_CADR*n* returns the value of A2 .
[5] Reading EMIOS_CBDR*n* returns the value of B1.

Figure 17-23. Pulse/Edge Accumulation Single-shot Mode Example

17.4.4.4.8 Pulse/Edge Counting Mode (PEC)

The PEC mode returns the amount of pulses or edges detected on the input for a desired time window. MODE[6] bit selects between continuous or single shot operation.

Triggering of the internal counter is done by a rising or falling edge or both edges on the input signal. The polarity and the triggering edge is selected by EDSEL and EDPOL bits in EMIOS_CCR*n*.

Register A1 holds the start time and register B1 holds the stop time for the time window. After writing to register A1, when a match occur between comparator A and the selected timebase, the internal counter is cleared and it is ready to start counting input events. When the time base matches comparator B1, the internal counter is disabled and the FLAG bit is set. Reading the EMIOS_CCNTR*n* returns the amount of detected pulses.

For continuous operation (MODE[6] cleared), the next match between comparator A and the selected time base clears the internal counter and counting is enabled again. In order to guarantee the accuracy when reading EMIOS_CCNTR*n* after the flag is set, the software must check if the time base value is out of the time interval defined by registers A1 and B1.

For single shot operation (MODE[6] set), the next match between comparator A and the selected time base has no effect, until a new write to register A is performed.

NOTE

The FORCMA and FORCMB bits have no effect when the unified channel is configured for PEC mode.

Figure 17-24 and Figure 17-25 show how the unified channel can be used for continuous or single shot pulse/edge counting mode.

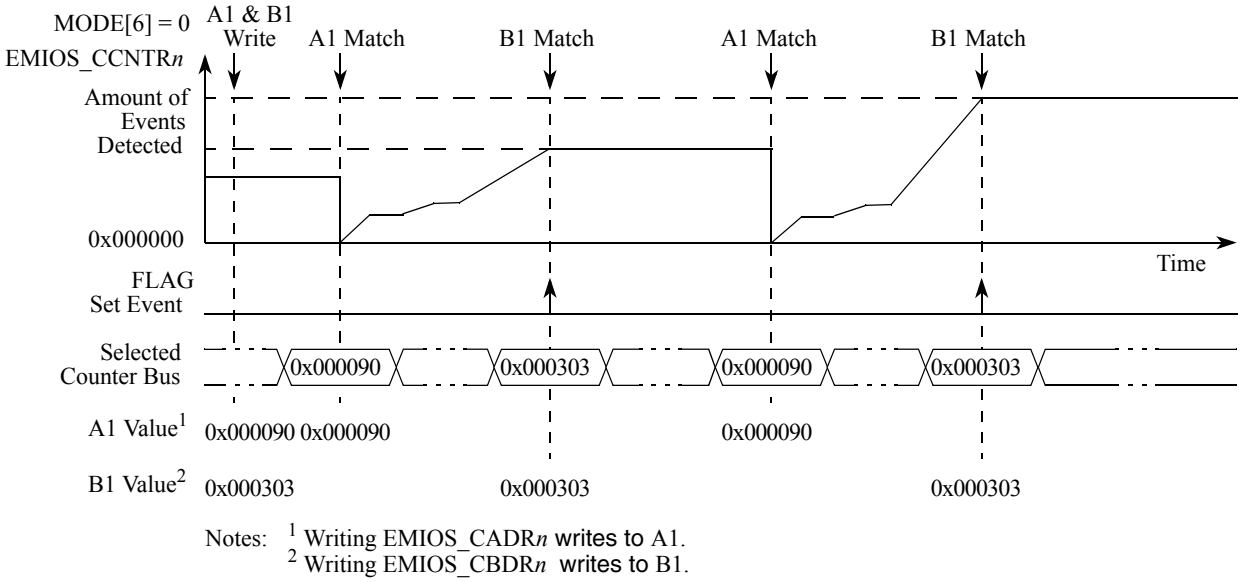


Figure 17-24. Pulse/Edge Counting Continuous Mode Example

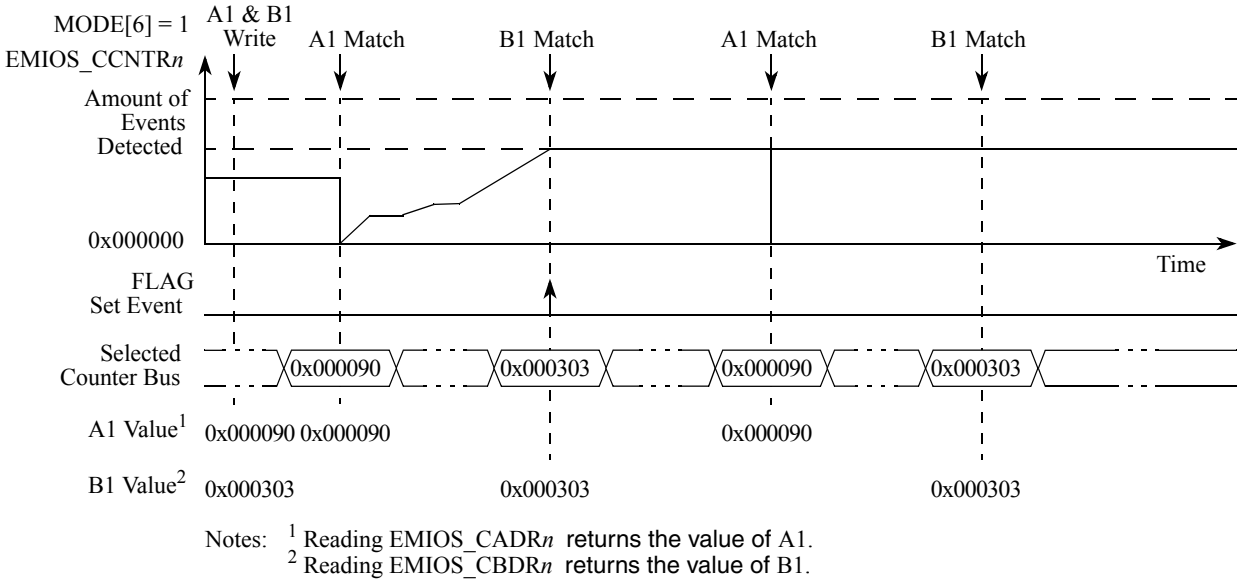


Figure 17-25. Pulse/Edge Counting Single-Shot Mode Example

17.4.4.4.9 Quadrature Decode Mode (QDEC)

Quadrature decode mode uses UC*n* operating in QDEC mode and the programmable input filter (PIF) from UC[n-1]. Note that UC[n-1] can be configured, at the same time, to an operation mode that does not use I/O pins, such as MC mode (modulus counter). The connection among the UCs is circular; that is, when UC0 is running in QDEC mode, the programmable input filter from UC23 is being used.

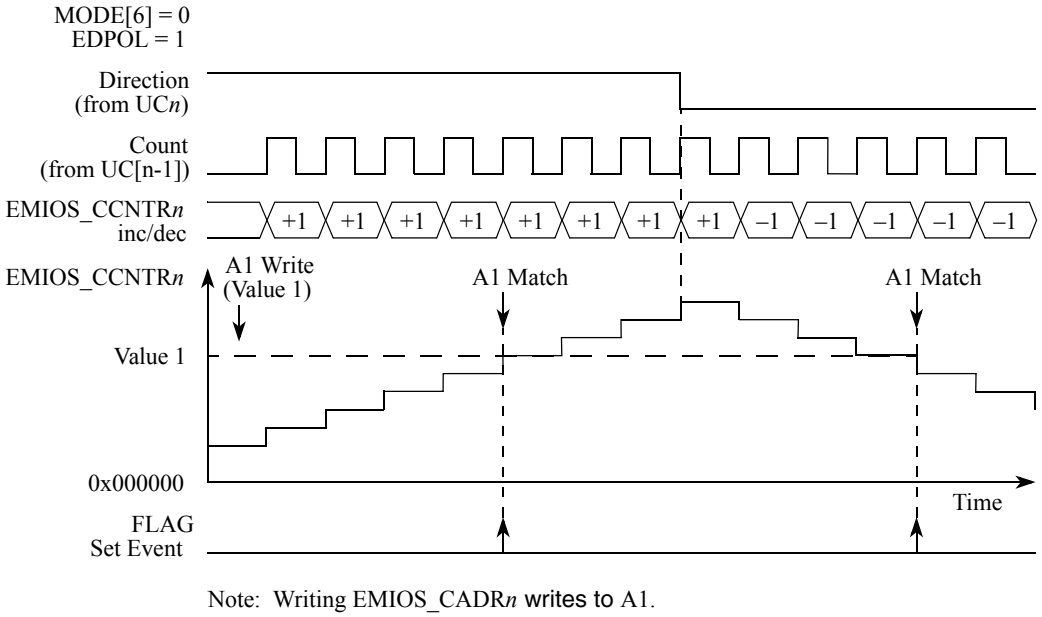
This mode generates a FLAG every time the internal counter matches A1 register. The internal counter is automatically selected and is not cleared when entering this mode.

MODE[6] bit selects which type of encoder will be used: count and direction encoder or phase_A and phase_B encoders.

When operating with count and direction encoder (MODE[6] cleared), UC*n* input pin must be connected to the direction signal and UC[n-1] input pin must be connected to the count signal of the quadrature encoder. UC*n* EDPOL bit selects count direction according to direction signal and UC[n-1] EDPOL bit selects if the internal counter is clocked by the rising or falling edge of the count signal.

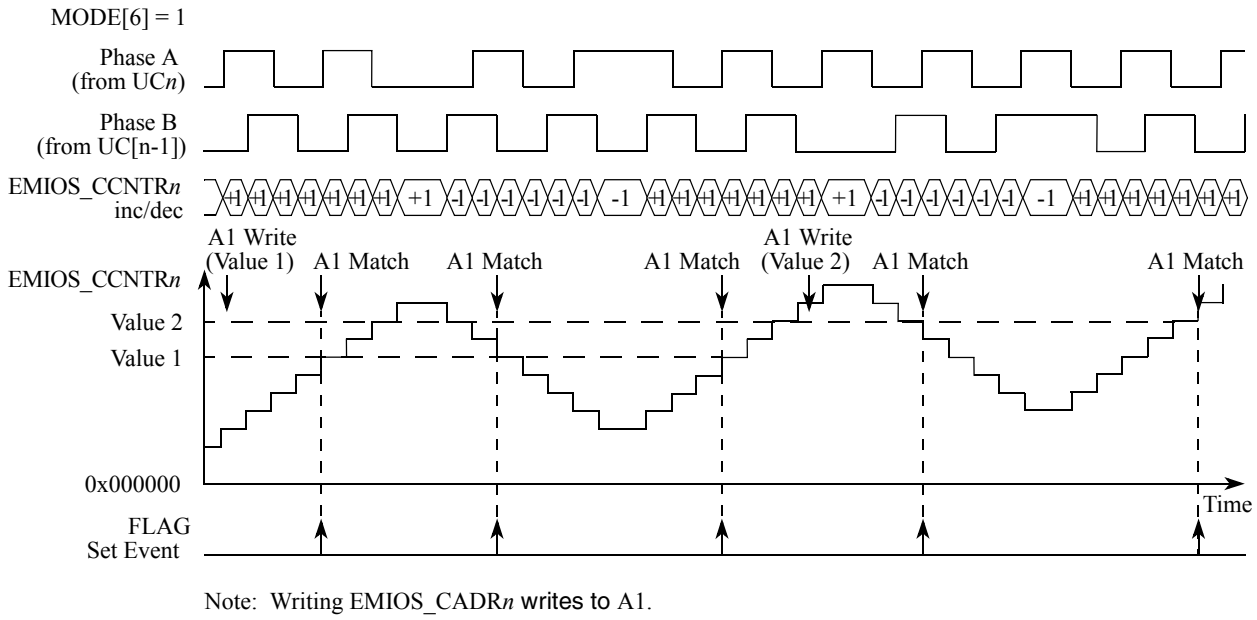
When operating with phase_A and phase_B encoder (MODE[6] set), UC*n* input pin must be connected to the phase_A signal and UC[n-1] input pin must be connected to the phase_B signal of the quadrature encoder. EDPOL bit selects the count direction according to the phase difference between phase_A and phase_B signals.

Figure 17-26 and Figure 17-27 show two unified channels configured to quadrature decode mode for count and direction encoder and phase_A and phase_B encoders, respectively.



Note: Writing EMIOS_CADR*n* writes to A1.

Figure 17-26. Quadrature Decode Mode Example with Count and Direction Encoder



Note: Writing EMIOS_CADR*n* writes to A1.

Figure 17-27. Quadrature Decode Mode Example with Phase_A and Phase_B Encoder

17.4.4.4.10 Windowed Programmable Time Accumulation Mode (WPTA)

The WPTA mode accumulates the sum of the total high time or low time of an input signal over a programmable interval (time window).

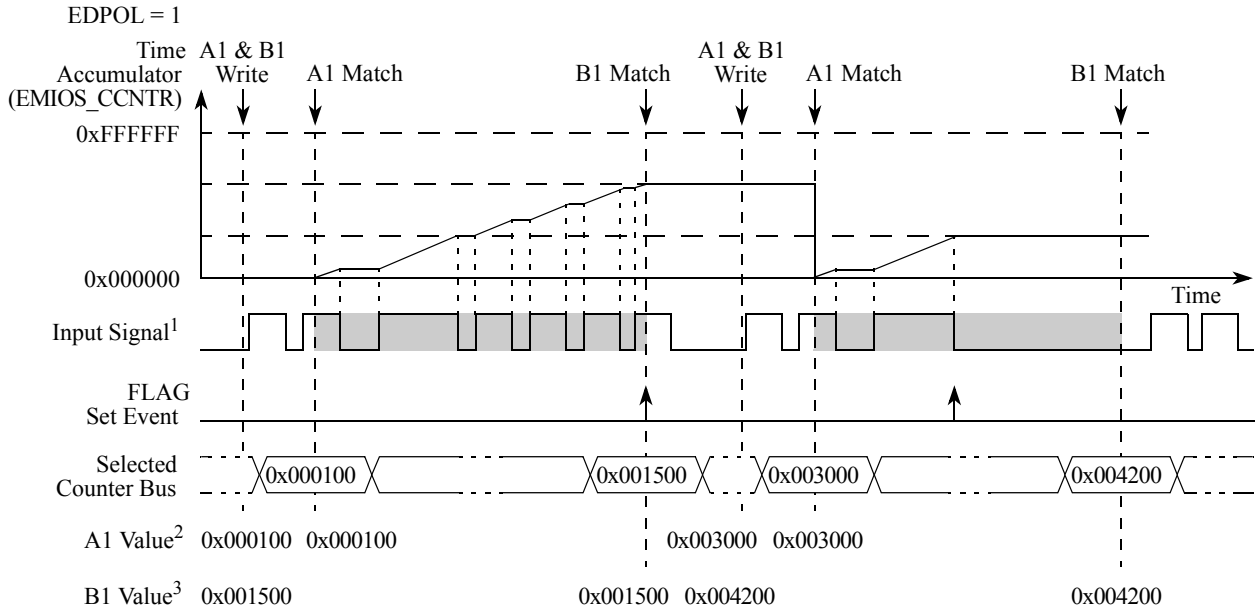
The prescaler bits UCPRE[0:1] in EMIOS_CCR*n* define the increment rate of the internal counter.

Register A1 holds the start time and register B1 holds the stop time of the programmable time interval. When a match occurs between register A and the selected timebase, the internal counter is cleared and it is ready to start counting. The internal counter is used as a time accumulator; that is, it counts up when the input signal has the same polarity of EDPOL bit in EMIOS_CCR*n* and does not count otherwise. When a match occurs in comparator B, the internal counter is disabled regardless of the input signal polarity and the FLAG bit is set. Reading EMIOS_CCNTR*n* returns the high or low time of the input signal.

NOTE

The FORCMA and FORCMB bits have no effect when the unified channel is configured for WPTA mode.

Figure 17-28 shows how the unified channel can be used to accumulate high time.



Notes: [1] After input filter.
[2] Writing EMIOS_CADR*n* writes to A1.
[3] Writing EMIOS_CBDR*n* writes to B1.

Figure 17-28. Windowed Programmable Time Accumulation Example

17.4.4.4.11 Modulus Counter Mode (MC)

The MC mode can be used to provide a time base for a counter bus or as a general purpose timer.

MODE[6] bit selects internal or external clock source when cleared or set, respectively. When external clock is selected, the input signal pin is used as the source and the triggering polarity edge is selected by the EDPOL and EDSEL in the EMIOS_CCR*n*.

The internal counter counts up from the current value until it matches the value in register A1. Register B1 is cleared and is not accessible to the MCU. MODE[4] bit selects up mode or up/down mode, when cleared or set, respectively.

When in up count mode, a match between the internal counter and register A1 sets the FLAG and clears the internal counter.

When in up/down count mode, a match between the internal counter and register A1 sets the FLAG and changes the counter direction from increment to decrement. A match between register B1 and the internal counter changes the counter direction from decrement to increment and sets the FLAG only if MODE[5] bit is set.

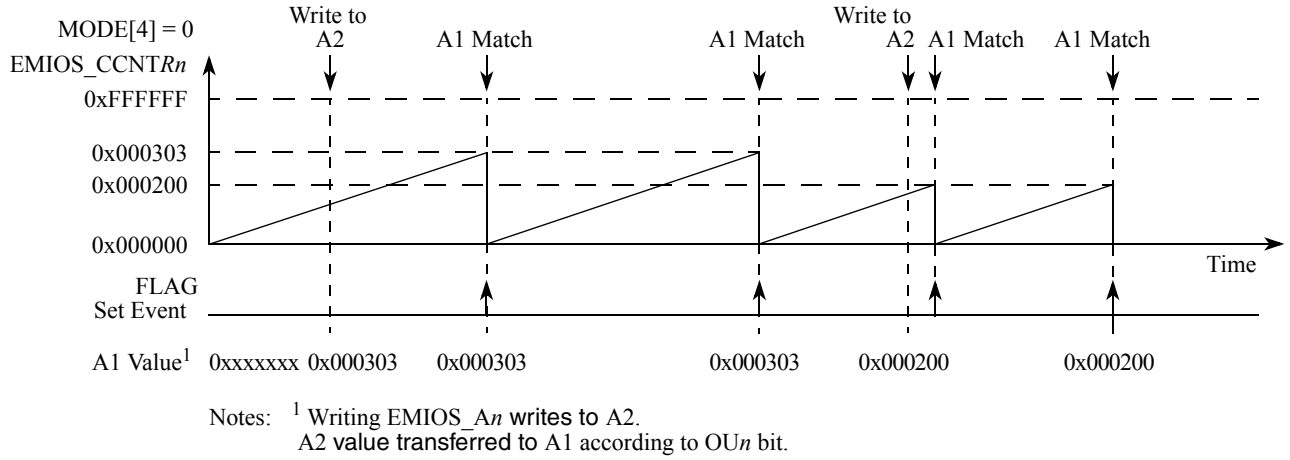
NOTE

The FORCMA and FORCMB bits have no effect when the unified channel is configured for MC mode.

NOTE

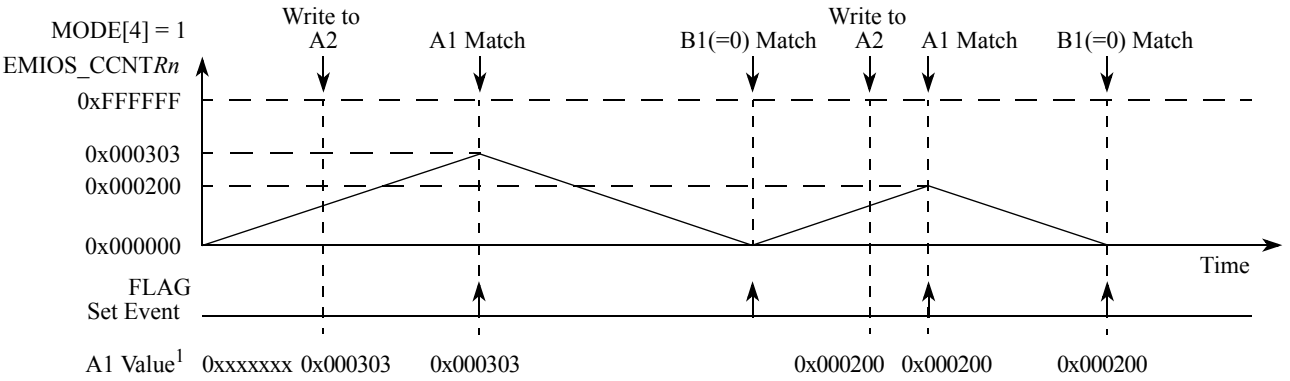
Any update to the A register will take place immediately, regardless of the current state of the counter and whether the counter is in up mode, or up/down mode.

Figure 17-29 and Figure 17-30 shows how the unified channel can be used as modulus counter in up mode and up/down mode, respectively.



Notes: [1] Writing EMIOS_A*n* writes to A2.
A2 value transferred to A1 according to OU*n* bit.

Figure 17-29. Modulus Counter Up Mode Example



Notes: [1] Writing EMIOS_A*n* writes to A2.
A2 value transferred to A1 according to OU*n* bit.

Figure 17-30. Modulus Counter Up/Down Mode Example

17.4.4.4.12 Output Pulse Width and Frequency Modulation Mode (OPWFM)

In this mode, register A1 contains the duty cycle and register B1 contains the period of the output signal. MODE[6] bit controls the transfer from register B2 to B1, which can be done either immediately (MODE[6] cleared), providing the fastest change in the duty cycle, or at every match of register A1 (MODE[6] set).

The internal counter is automatically selected as a time base, therefore the BSL[0:1] bits in register EMIOS_CCR*n* have no meaning. The output flip-flop's active state is the complement of EDPOL bit. The output flip-flop is active during the duty cycle (from the start of the cycle until a match occurs in comparator A). After the match in comparator A the output flip-flop is in the inactive state (the value of EDPOL) until the next cycle starts. When a match on comparator A occurs, the output flip-flop is set to the value of the EDPOL bit. When a match occurs on comparator B, the output flip-flop is set to the complement of the EDPOL bit and the internal counter is cleared.

FLAG can be generated at match B, when MODE[5] is cleared, or in both matches, when MODE[5] is set.

At any time, the FORCMA and FORCMB bits allow the software to force the output flip-flop to the level corresponding to a match on A or B respectively. Also, FORCMB clears the internal counter. Note that the FLAG bit is not set by the FORCMA or FORCMB operations.

If subsequent comparisons occur on comparators A and B, the PWFM pulses continue to be output, regardless of the state of the FLAG bit.

In order to achieve 0% duty cycle, both registers A1 and B1 must be set to the same value. When a simultaneous match occurs on comparators A and B, the output flip-flop is set at every period to the value of EDPOL bit.

To temporarily change from the curent duty cycle to 0% and then return to the current duty cycle, the sequence is the following:

1. If not currently stored, store value of register A.
2. Set A=B.
3. If immediate 0% duty cycle is desired, set FORCA=1.
4. To return to the previous duty cycle, restore register A with its former value.

100% duty cycle is possible by writing 0x000000 to register A. When a match occurs, the output flip-flop is set at every period to the complement of EDPOL bit. The transfer from register B2 to B1 is still controlled by MODE[6] bit.

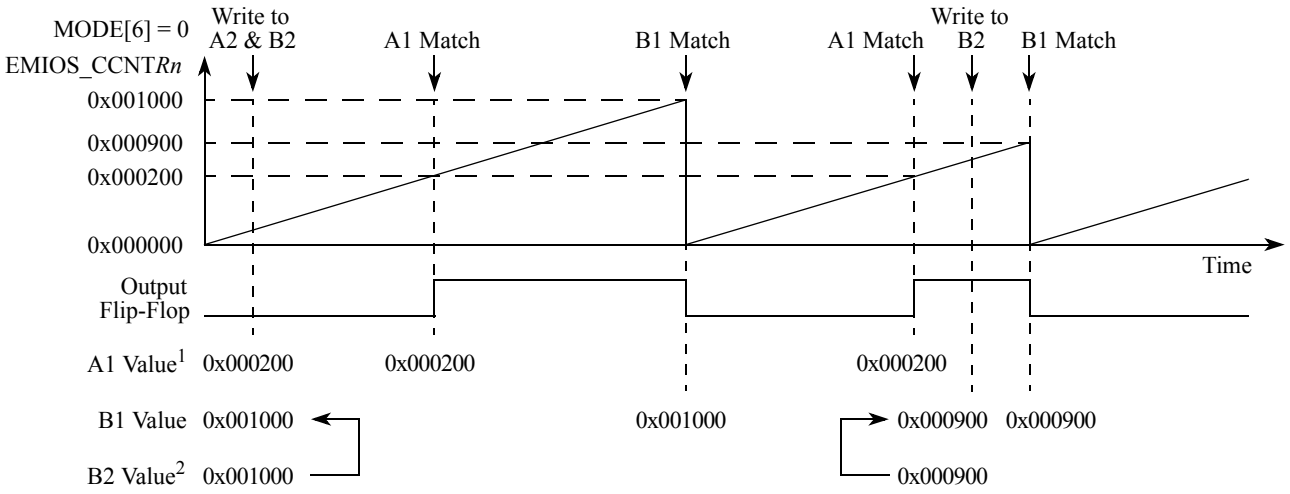
To temporarily change from the current duty cycle to 100% and then return to the current duty cycle, the sequence is the following:

1. If not currently stored, store value of register A.
2. Set A=0.
3. If immediate 100% duty cycle is desired, set FORCB=1.
4. To return to the previous duty cycle, restore register A with its former value.

NOTE

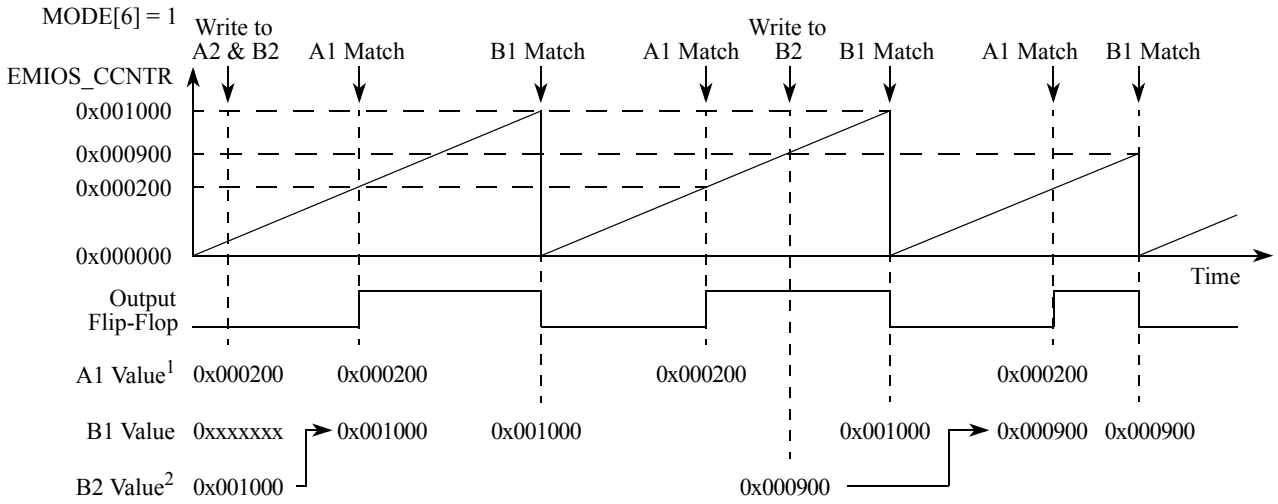
Updates to the A register will always occur immediately. If next period update is selected via the mode[6] bit, only the B register update is delayed until the next period.

Figure 17-31 shows the unified channel running in OPFWM mode with immediate register update and Figure 17-32 shows the unified channel running in OPFWM mode with next period update PFWM mode. In both figures EDPOL = 1, so the output is low during the duty cyle. Table 17-14 has additional illustrative examples.



Notes: [1] Writing EMIOS_A*n* writes to A2.
[2] Writing EMIOS_B*n* writes to B2.
A2 value transferred to A1 according to OU*n* bit.
B2 value transferred to B1 according to OU*n* bit.

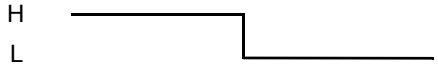
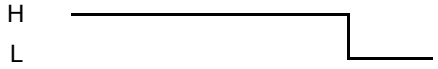
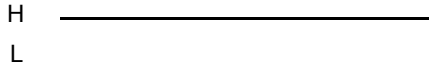
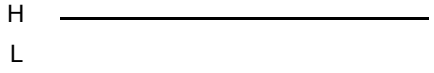
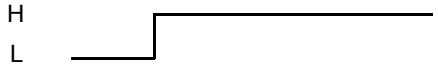
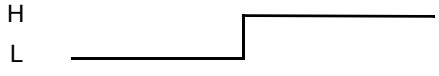
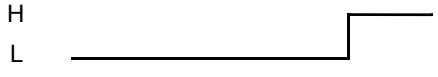
Figure 17-31. OPWFM with Immediate Update



Notes: [1] Writing EMIOS_A*n* writes to A2.
[2] Writing EMIOS_B*n* writes to B2.
A2 value transferred to A1 according to OU*n* bit.
B2 value transferred to B1 according to OU*n* bit.

Figure 17-32. OPWFM with Next Period Update

Table 17-14. Examples of Output Waveforms

EDPOL	Duty Cycle	A (decimal)	B (decimal)	Waveform
0 (active high ouput)	0%	1000	1000	
	25%	250	1000	
	50%	500	1000	
	75%	750	1000	
	100%	0	1000	
1 (active low ouput)	0%	1000	1000	
	25%	250	1000	
	50%	500	1000	
	75%	750	1000	
	100%	0	1000	

17.4.4.4.13 Center Aligned Output Pulse Width Modulation with Dead-time Mode (OPWMC)

This operating mode generates a center aligned PWM with dead time insertion in the leading or trailing edge.

The selected counter bus must be running an up/down time base, as shown in Figure 17-30. BSL[0:1] bits select the time base. Register A1 contains the ideal duty cycle for the PWM signal and is compared with the selected time base. Register B1 contains the dead time value and is compared with the internal counter. For a leading edge dead time insertion, the output PWM duty cycle is equal to the difference between register A1 and register B1, and for a trailing edge dead time insertion, the output PWM duty cycle is equal

to the sum of register A1 and register B1. MODE[6] bit selects between trailing and leading dead time insertion, respectively.

NOTE

It is recommended that the internal prescaler of the OPWMCB channel be set to the same value as the MCB channel prescaler, and the prescalers should also be synchronized. This allows the A1 and B1 registers to represent the same time scale for duty cycle and dead time insertion.

When operating with leading edge dead time insertion, the first match between A1 and the selected time base clears the internal counter and switches the selected time base to the internal counter. When a match occurs between register B1 and the selected time base, the output flip-flop is set to the value of the EDPOL bit and the time base is switched to the selected counter bus. In the next match between register A1 and the selected time base, the output flip-flop is set to the complement of the EDPOL bit. This sequence repeats continuously.

When operating with trailing edge dead time insertion, the first match between A1 and the selected time base sets the output flip-flop to the value of the EDPOL bit. In the next match between register A1 and the selected time base, the internal counter is cleared and the selected time base is switched to the internal counter. When a match occurs between register B1 and the selected time base, the output flip-flop is set to the complement of the EDPOL bit and the time base is switched to the selected counter bus. This sequence repeats continuously.

FLAG can be generated in the trailing edge of the output PWM signal when MODE[5] is cleared, or in both edges, when MODE[5] is set.

At any time, the FORCMA or FORCMB bits are equivalent to a successful comparison on comparator A or B with the exception that the FLAG bit is not set.

NOTE

When in freeze mode, the FORCMA or FORCMB bits only allow the software to force the output flip-flop to the level corresponding of a match on A or B respectively.

If subsequent matches occur on comparators A and B, the PWM pulses continue to be generated, regardless of the state of the FLAG bit.

In order to achieve a duty cycle of 100%, both registers A1 and B1 must be set to the same value. When a simultaneous match occurs between the selected time base and registers A1 and B1, the output flip-flop is set at every period to the value of EDPOL bit and the selected time base switches to the selected counter bus, allowing a new cycle to begin at any time, as previously described. 0% duty cycle is possible by writing 0x000000 to register A. When a match occurs, the output flip-flop is set at every period to the complement of EDPOL bit and the selected time base switches to the selected counter bus, allowing a new cycle to begin at any time, as previously described. In both cases, FLAG is generated regardless of MODE[5] bit.

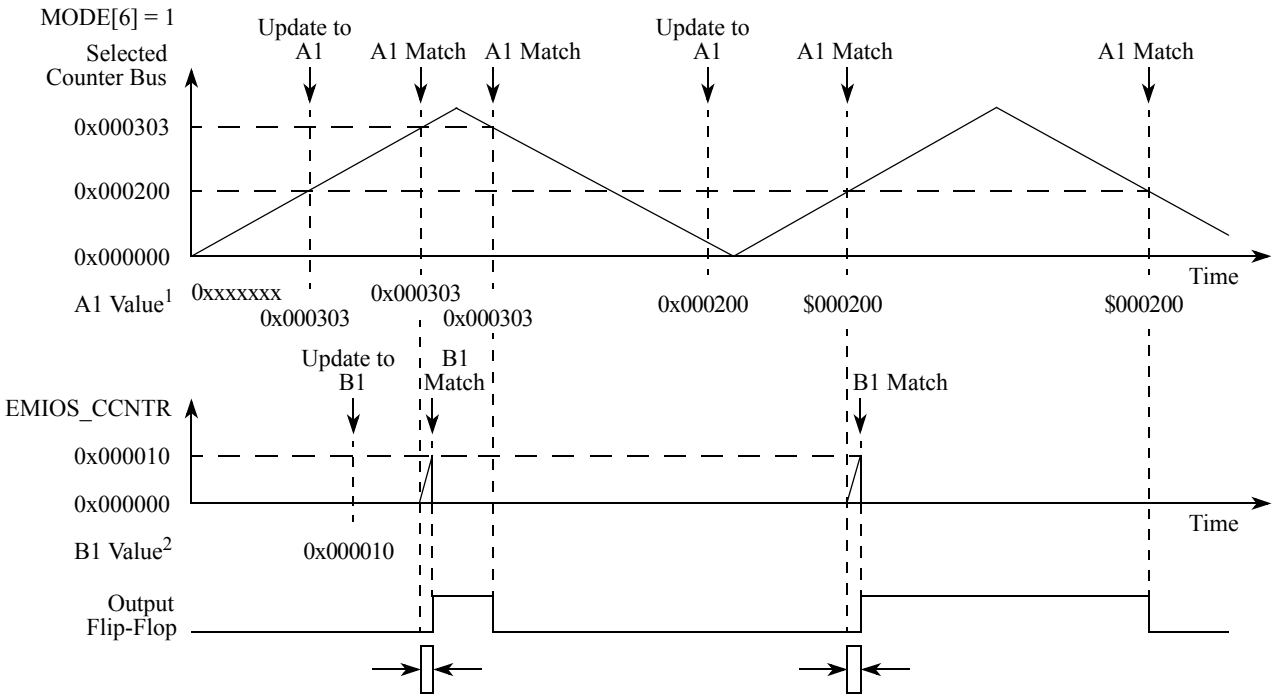
NOTE

If A1 and B1 are set to the 0x000000, a 0% duty cycle waveform is produced.

NOTE

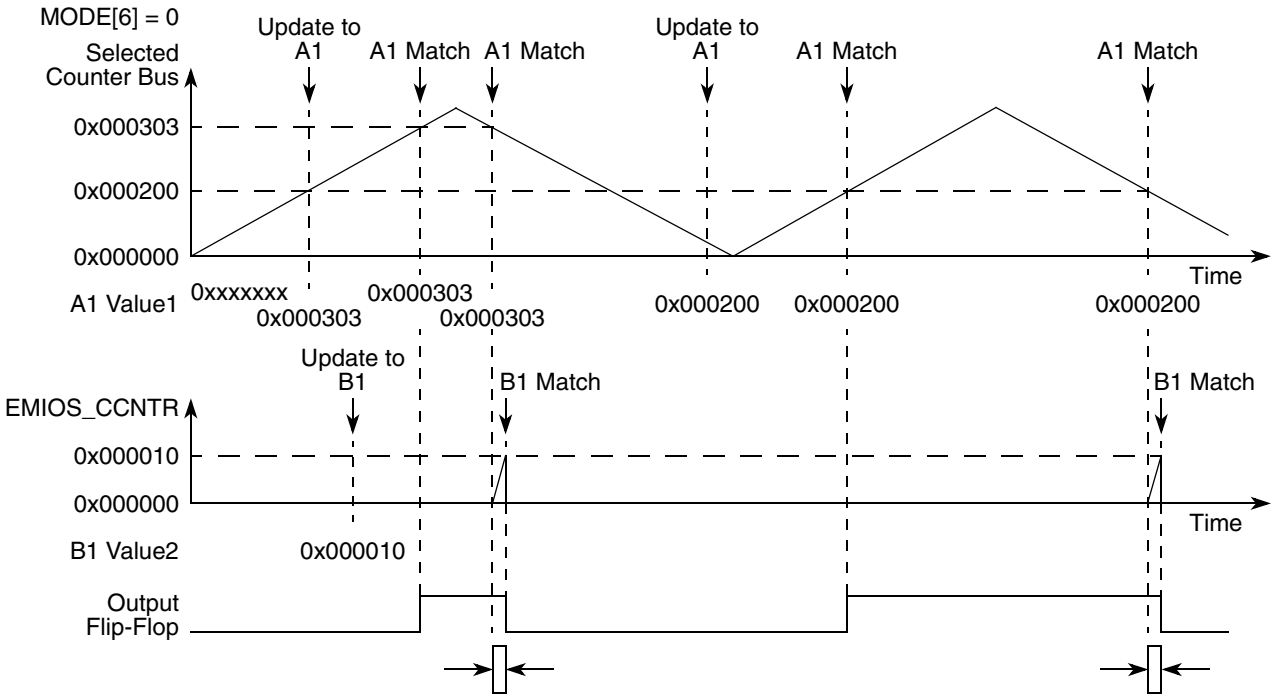
Any updates to the A or B register will take place immediately.

Figure 17-33 and Figure 17-34 show the unified channel running in OPWMC with leading and trailing dead time, respectively.



Notes: [1] Writing EMIOS_A*n* writes to A2.
[2] Writing EMIOS_B*n* writes to B1.
A2 value transferred to A1 according to OU*n* bit.
B2 value transferred to B1 according to OU*n* bit.

Figure 17-33. Output PWMC with Leading Dead-time Insertion



Notes: 1 Writing EMIOS_A*n* writes to A2.
2 Writing EMIOS_B*n* writes to B1.
A2 value transferred to A1 according to OU*n* bit.
B2 value transferred to B1 according to OU*n* bit.

Figure 17-34. Output PWMC with Trailing Dead-time Insertion

17.4.4.4.14 Output Pulse Width Modulation Mode (OPWM)

Registers A1 and B1 define the leading and trailing edges of the PWM output pulse, respectively. MODE[6] bit controls the transfer from register B2 to B1, which can be done either immediately (MODE[6] cleared), providing the fastest change in the duty cycle, or at every match of register A1 (MODE[6] set).

The value loaded in register A1 is compared with the value on the selected time base. When a match on comparator A occurs, the output flip-flop is set to the value of the EDPOL bit. When a match occurs on comparator B, the output flip-flop is set to the complement of the EDPOL bit.

FLAG can be generated at match B, when MODE[5] is cleared, or in both matches, when MODE[5] is set.

At any time, the FORCMA and FORCMB bits allow the software to force the output flip-flop to the level corresponding to a match on A or B respectively. Note that FLAG bit is not set by the FORCMA and FORCMB operations.

If subsequent matches occur on comparators A and B, the PWM pulses continue to be generated, regardless of the state of the FLAG bit.

In order to achieve 0% duty cycle, both registers A1 and B1 must be set to the same value. When a simultaneous match on comparators A and B occur, the output flip-flop is set at every period to the value of EDPOL bit. 0% duty cycle is possible by writing 0x000000 to register A. When a match occurs, the output flip-flop is set at every period to the complement of EDPOL bit. The transfer from register B2 to B1 is still controlled by MODE[6] bit.

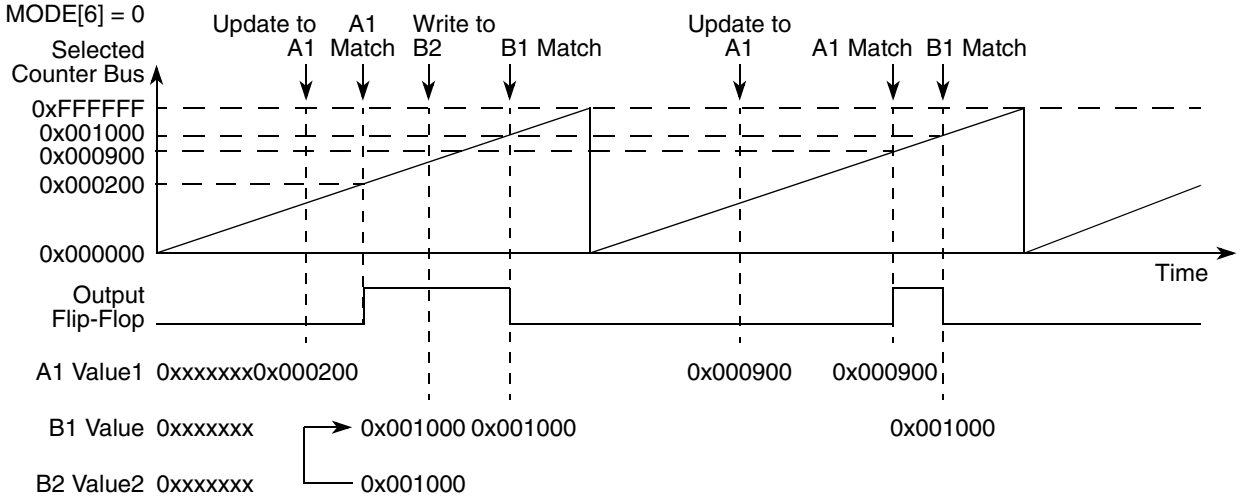
NOTE

If A1 and B1 are set to the 0x000000, a 100% duty cycle waveform is produced.

NOTE

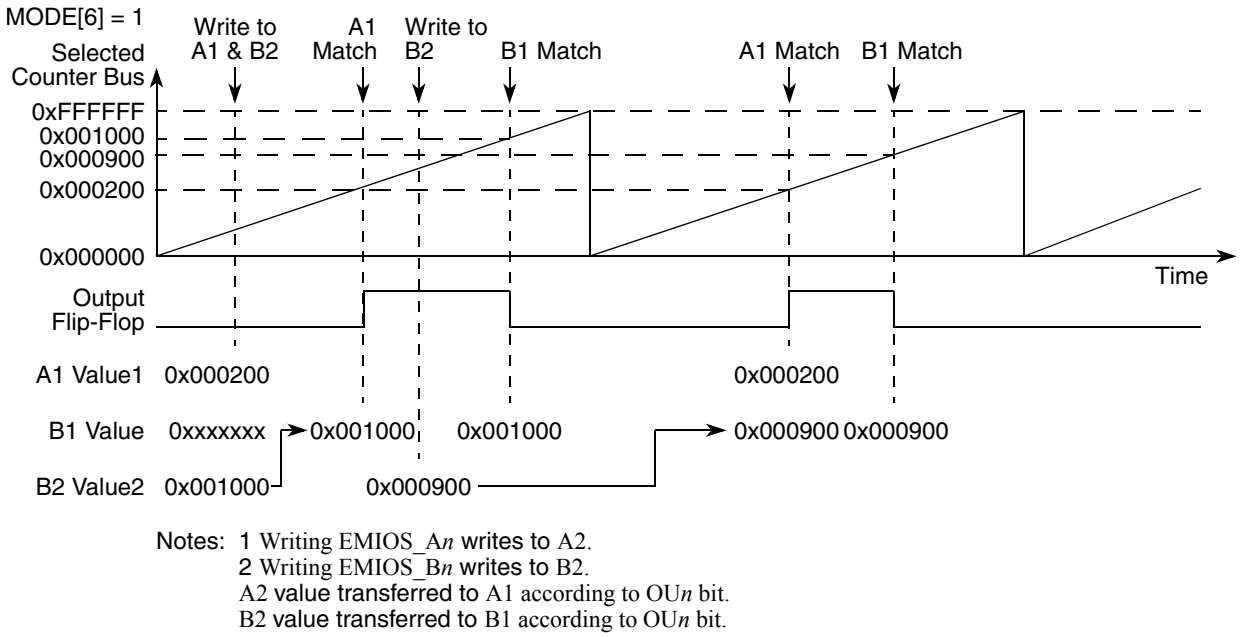
Updates to the A register will always occur immediately. If next period update is selected via the mode[6] bit, only the B register update is delayed until the next period.

Figure 17-35 and Figure 17-36 show the unified channel running in OPWM with immediate update and next period update, respectively.



Notes: 1 Writing EMIOS_A*n* writes to A2.
2 Writing EMIOS_B*n* writes to B2.
A2 value transferred to A1 according to OU*n* bit.
B2 value transferred to B1 according to OU*n* bit.

Figure 17-35. Output PWM with Immediate Update



Notes: 1 Writing EMIOS_A*n* writes to A2.
2 Writing EMIOS_B*n* writes to B2.
A2 value transferred to A1 according to OU*n* bit.
B2 value transferred to B1 according to OU*n* bit.

Figure 17-36. Output PWM with Next Period Update

17.4.4.4.15 Modulus Counter, Buffered Mode (MCB) (MPC5553 Only)

The MCB mode provides a time base which can be shared with other channels through the internal counter buses. Register A1 is double buffered, thus allowing smooth transitions between cycles when changing the A2 register value on the fly. The A1 register is updated at the cycle boundary, which is defined as when the internal counter reaches the value one. Note that the internal counter values are within a range from one up to register A1 value in MCB mode.

The MODE[6] bit selects the internal clock source if clear or external if set. When an external clock is selected, the channel input pin is used as the channel clock source. The active edge of this clock is defined by EDPOL and EDSEL bits in the EMIOS_CCR channel register.

When entering the MCB mode, if up counter is selected (MODE[4] = 0), the internal counter starts counting up from its current value to until an A1 match occurs. On the next system clock cycle after an A1 match occurs, the internal counter is set to one and the counter continues counting up. If up/down mode is selected (MODE[4] = 1), the counter changes direction at the A1 match and counts down until it reaches one and is then set to count up again. In this mode B1 is set to one and cannot be changed, as it is used to generate a match to switch from down count to up count.

Note that versus the MC mode, the MCB mode counts between one and the A1 register value. The counter cycle period in up count mode is equal to the A1 value. In up/down counter mode the period is defined by the formula: $(2 \times A1) - 2$.

Figure 17-37 illustrates the counter cycle for several A1 values. Register A1 is loaded with the A2 value at the cycle boundary. Thus any value written to A2 within cycle (*n*) will be updated to A1 at the next cycle boundary, and therefore will be used on cycle (*n*+1). The cycle boundary between cycle (*n*) and cycle (*n*+1) is defined as the first clock cycle of cycle (*n*+1). Note that flags are set when A1 matches occur.

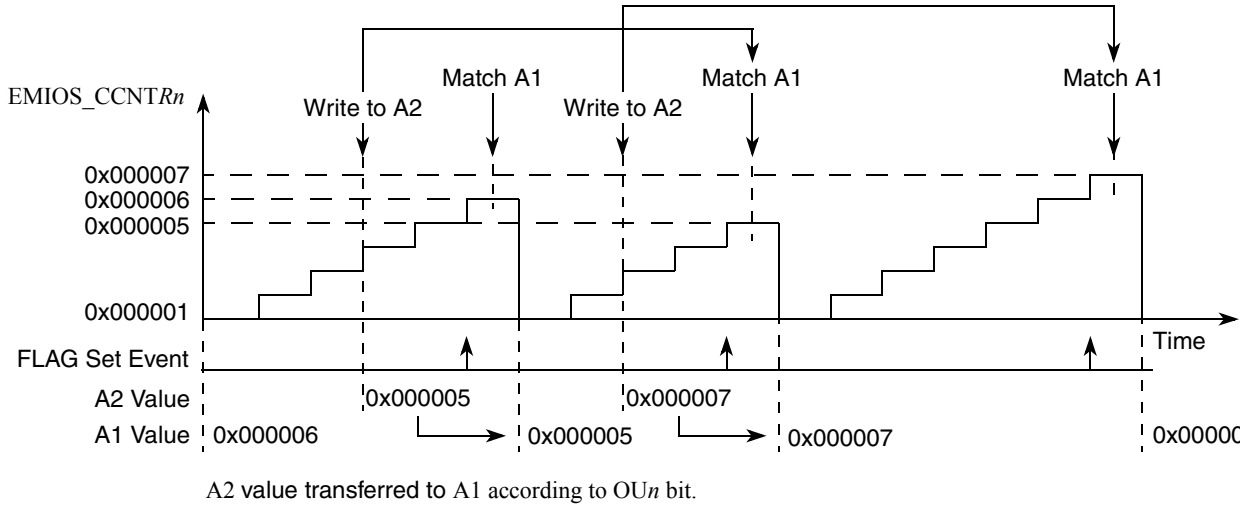


Figure 17-37. eMIOS MCB Mode Example — Up Operation

NOTE

If a prescaler greater than 1 is used, there are several system clock cycles between when the flag is asserted and the counter is set to one. This should be considered when the A value is changed in every cycle, because A1 is updated on the cycle boundary, which is after the flag is set.

Figure 17-38 illustrates the MCB up/down counter mode. The A1 register is updated at the cycle boundary. If A2 is written in cycle (*n*), this new value will be used in cycle (*n*+1) for the next A1 match.

Flags are generated only at an A1 match if MODE[5] is 0. If MODE[5] is 1, flags are also generated at the cycle boundary.

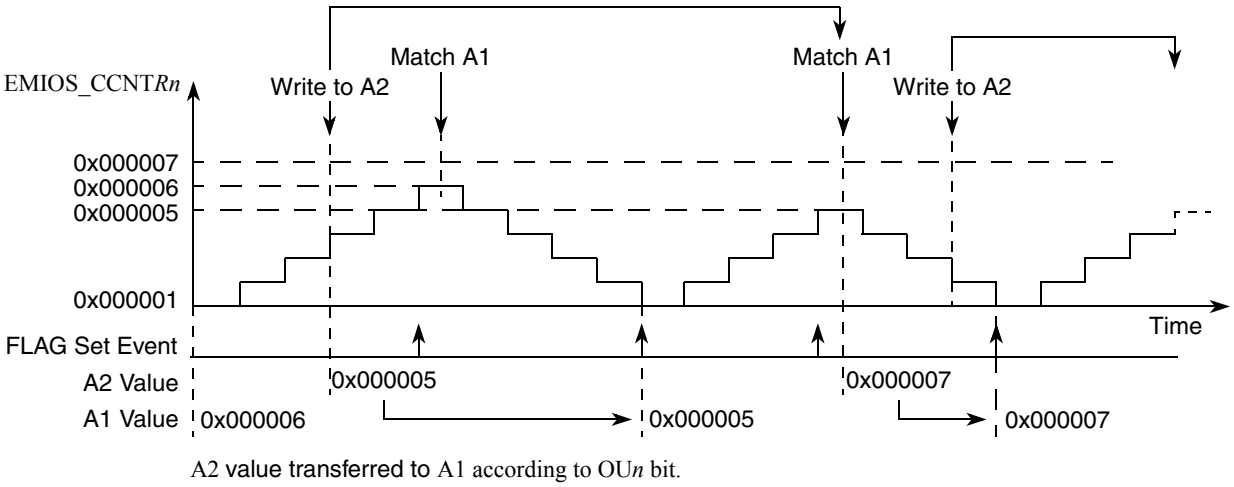
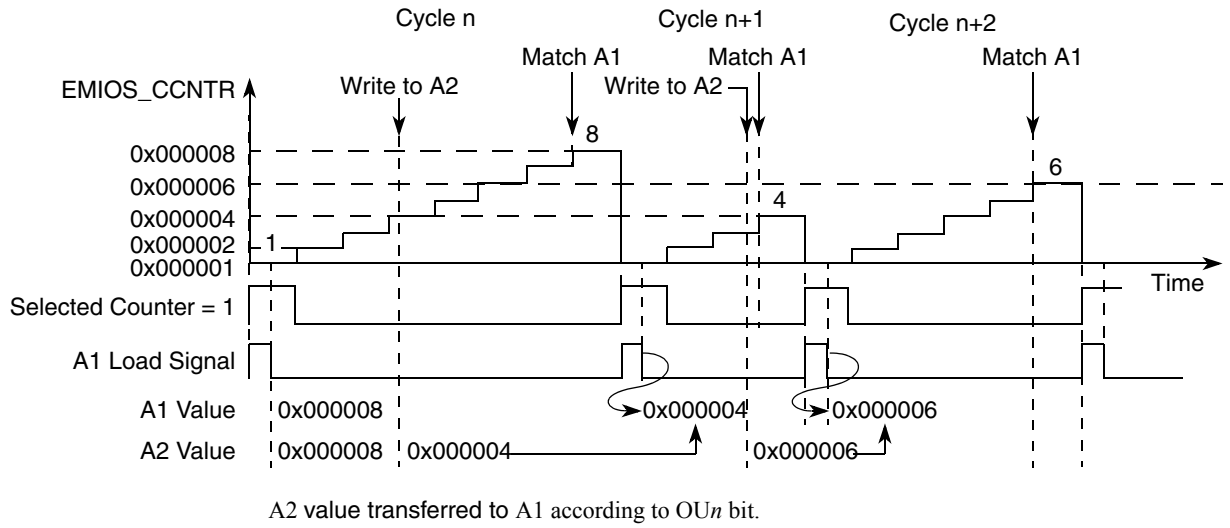


Figure 17-38. eMIOS MCB Mode Example — Up/Down Operation

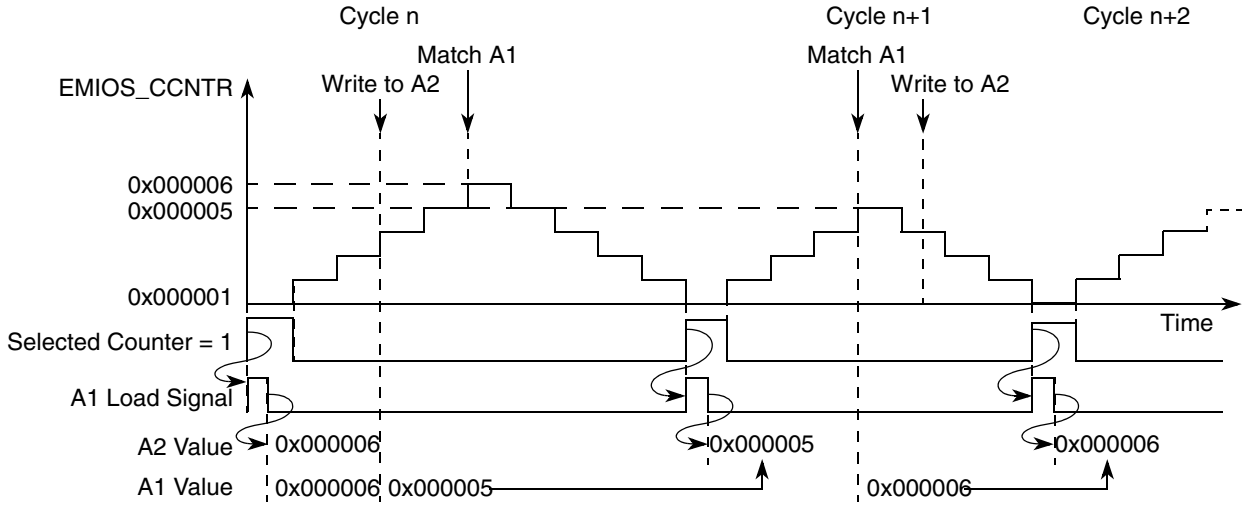
Figure 17-39 provides a more detailed illustration of the A1 update process in up counter mode. The A1 load signal is generated based on the detection of the internal counter reaching one, and has the duration of one system clock cycle. Note that during the load pulse A1 still holds its previous value. It is actually updated at the second system clock cycle.



A2 value transferred to A1 according to OU*n* bit.

Figure 17-39. eMIOS MCB Mode Example — Up Operation A1 Register Update

Figure 17-40 illustrates the A1 register update process in up/down counter mode. Note that A2 can be written at any time within cycle (*n*) in order to be used in cycle (*n*+1). Thus A1 receives the new value at the next cycle boundary. The EMIOS_OUDR[n] bits can be used to disable the update of A1 register.



A2 value transferred to A1 according to OU*n* bit (the transfer is triggered by the A1 load signal)

Figure 17-40. eMIOS MCB Mode Example — Up/Down Operation A1 Register Update

17.4.4.4.16 Output Pulse Width and Frequency Modulation, Buffered Mode (OPWFMB) (MPC5553 Only)

This mode generates waveforms with variable duty cycle and frequency. The internal channel counter is automatically selected as the time base, A1 sets the duty cycle and B1 determines the frequency. Both A1 and B1 are double buffered to allow smooth signal generation when changing the register values on the fly. 0% and 100% duty cycles are supported.

In order to provide smooth and consistent channel operation, this mode differs substantially from the OPWFM mode. The main differences are in how A1 and B1 are updated, the delay from the A1 match to the output flip-flop transition, and the range of the internal counter which ranges from 1 up to B1 value.

When a match on comparator A occurs, the output register is set to the value of EDPOL. When a match on comparator B occurs, the output register is set to the complement of EDPOL. A B1 match also causes the internal counter to transition to 1, thus re-starting the counter cycle.

Figure 17-41 shows an example of OPWFMB mode operation. Note that the output flip-flop transition occurs when the A1 or B1 match signal is negated, as detected by the negative edge of the A1 and B1 match signals. For example, if register A1 is set to 0x000004, the output flip-flop transitions 4 counter periods after the cycle starts, plus one system clock cycle. Note that in the example shown in Figure 17-41 the prescaler ratio is set to two (refer to Section 17.5.3, "Time Base Generation).

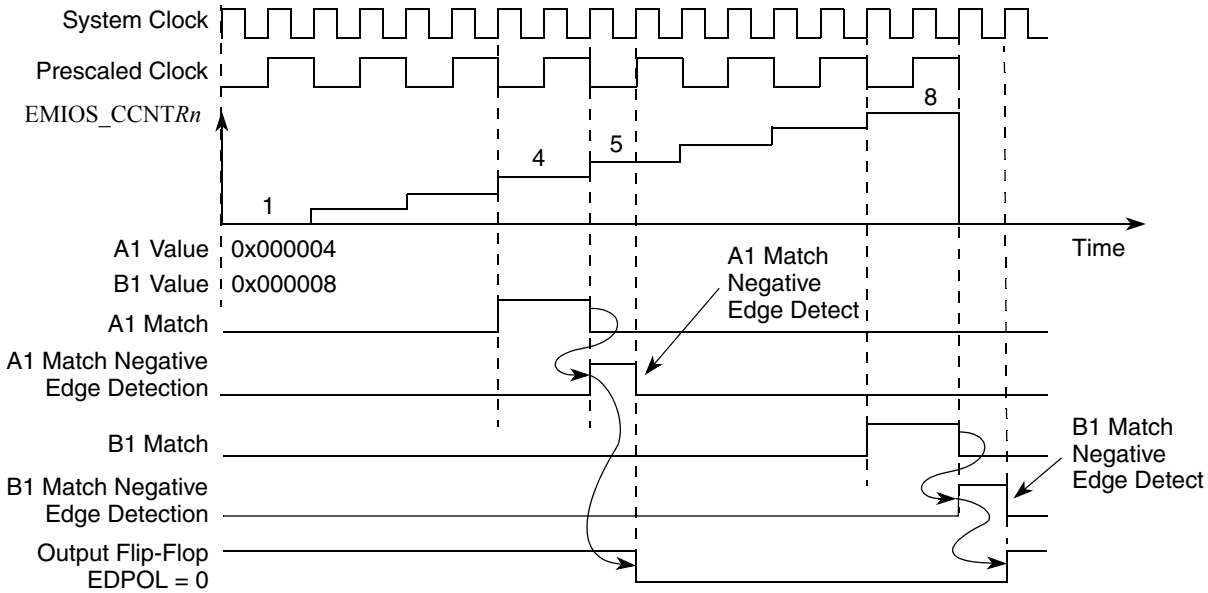


Figure 17-41. eMIOS OPWFMB Mode Example — A1/B1 Match to Output Register Delay

Figure 17-42 shows the generated output signal if A1 is 0. Since the counter does not reach zero in this mode, the channel internal logic infers a match as if A1 = 1, with the difference that in this case the positive edge of the match signal is used to trigger the output flip-flop transition instead of the positive edge that is used when A1 = 1. Note that the A1 positive edge match signal from cycle (*n*+1) occurs at the same time as the B1 match negative edge from cycle (*n*). This allows the use of the A1 match positive edge to mask the B1 match negative edge when they occur at the same time. The result is that no transition occurs on the output flip-flop, and a 0% duty cycle is generated.

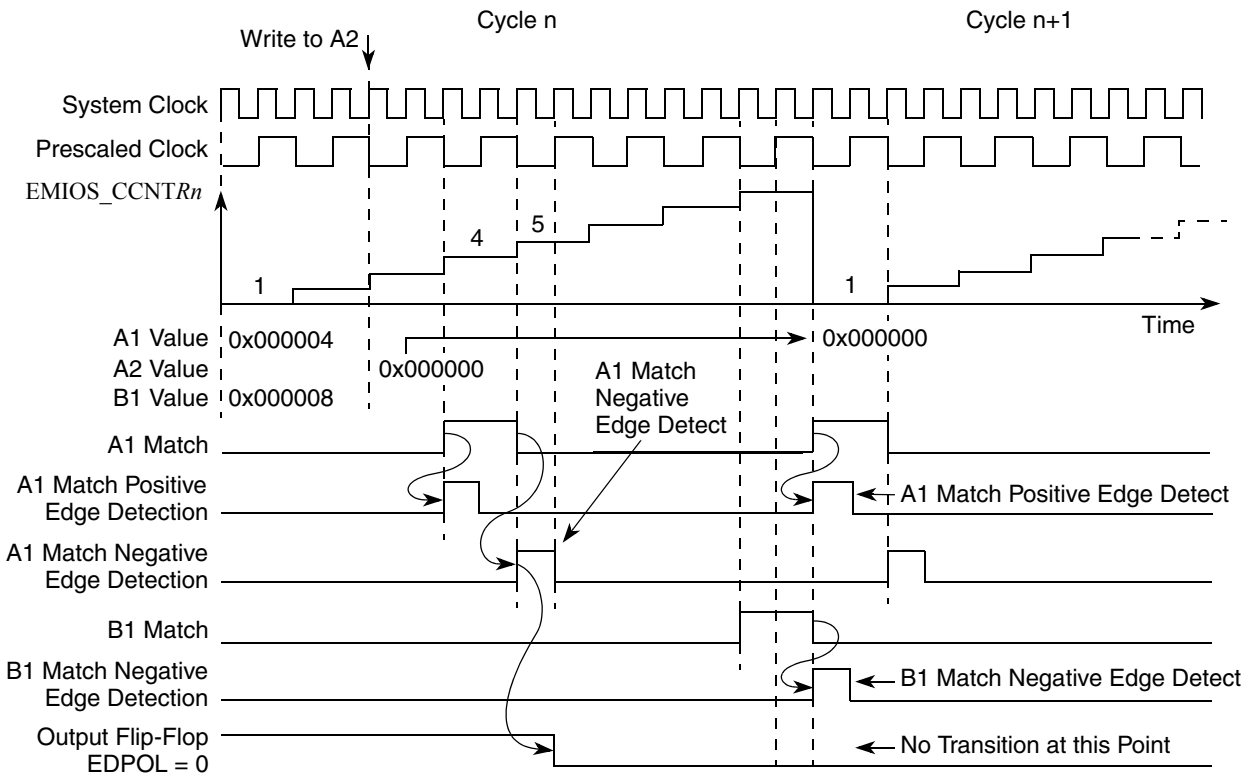


Figure 17-42. eMIOS OPWFMB Mode Example — A1 = 0 (0% Duty Cycle)

Figure 17-43 shows the timing for the A1 and B1 loading. A1 and B1 use the same signal to trigger a load, which is generated based on the selected counter reaching one. This event is defined as the cycle boundary. The load signal pulse has the duration of one system clock cycle and occurs at the first system clock period of every cycle of the counter. If A2 and B2 are written within cycle (*n*), their values are loaded into A1 and B1, respectively, at the first clock of cycle (*n*+1). The update disable bits, EMIOS_OUDR, can be used to control the update of these registers, thus allowing the delay of A1 and B1 update for synchronization purposes.

During the load pulse A1 still holds its old value, which is updated on the following system clock cycle. During the A1 load pulse, an internal by-pass allows the use of A2 instead of A1 for matches if A2 is either 0 or 1, thus allowing matches to be generated even when A1 is being loaded. This approach allows a uniform channel operation for any A2 value, including 1 and 0.

In Figure 17-43 it is assumed that the channel and global prescalers are set to one, meaning that the channel internal counter transition at every system clock cycle. FLAGs can be generated only on B1 matches when MODE[5] is cleared, or on both A1 and B1 matches when MODE[5] is set. Since B1 FLAG occurs at the cycle boundary, this flag can be used to indicate that A2 or B2 data written on cycle (*n*) were loaded to A1 or B1, respectively, thus generating matches in cycle (*n*+1).

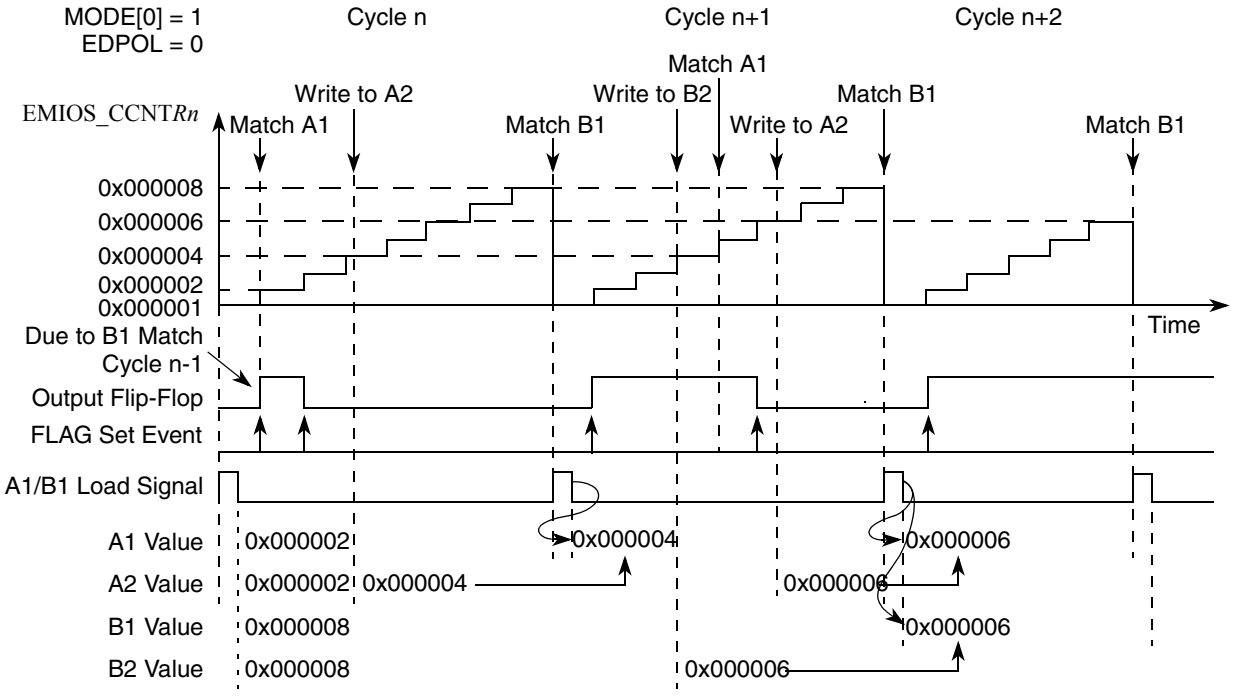


Figure 17-43. eMIOS OPWFMB Mode Example — A1/B1 Updates and Flags

Figure 17-44 shows the operation of the output disable feature in OPWFMB mode. Unlike OPWFM mode, the output disable forces the channel output flip-flop to the EDPOL bit value. This functionality targets applications that use active high signals and a high to low transition at A1 match. For such cases EDPOL should be 0.

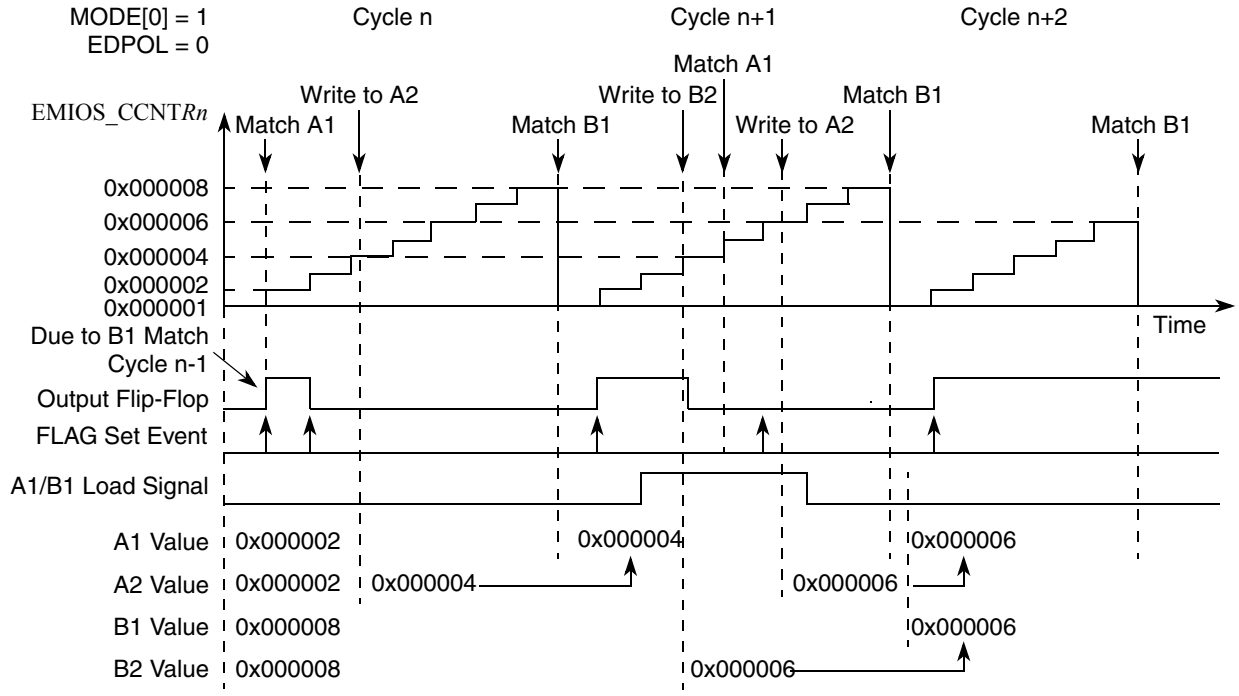


Figure 17-44. eMIOS OPWFMB Mode Example — Active Output Disable

Note that the output disable has a synchronous operation, meaning that the assertion of the output disable input signal causes the channel output flip-flop to transition to EDPOL at the next system clock cycle. If the output disable input is negated, the output flip-flop transitions at the following A1 or B1 match.

In Figure 17-44 it is assumed that the output disable input is enabled and selected for the channel (refer to Section 17.3.1.7, "eMIOS Channel Control Register (EMIOS_CCRn)," for a detailed description of the ODIS and ODISSL bits and selection of the output disable inputs).

The FORCMA and FORCMB bits allow the software to force the output flip-flop to the level corresponding to a match on comparators A or B respectively. Similar to a B1 match, FORCMB clears the internal counter. The FLAG bit is not set when the FORCMA or FORCMB bits are set.

Figure 17-45 illustrates the generation of 100% and 0% duty cycle signals. It is assumed that EDPOL = 0 and the prescaler ratio is 1. Initially A1 = 0x000008 and B1 = 0x000008. In this case, a B1 match has precedence over an A1 match, thus the output flip-flop is set to the complement of EDPOL. This cycle corresponds to a 100% duty cycle signal. The same output signal can be generated for any A1 value greater than or equal to B1.

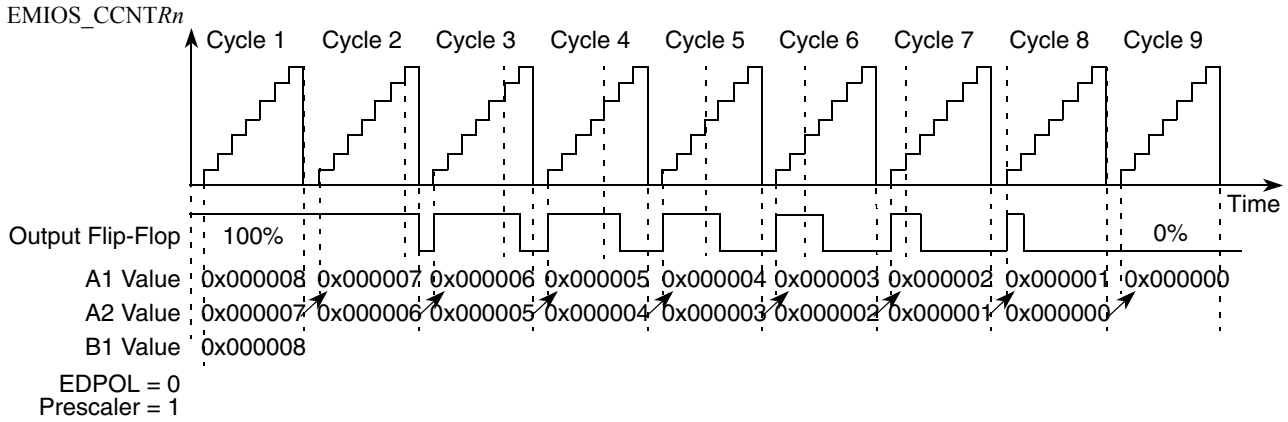


Figure 17-45. eMIOS OPWFMB Mode Example — 100% to 0% Duty Cycle

A 0% duty cycle signal is generated if A1 = 0 as shown in Figure 17-45 cycle 9. In this case the B1 = 0x000008 match from cycle 8 occurs at the same time as the A1 = 0x000000 match from cycle 9. Refer to Figure 17-42 for a description of A1 and B1 match generation for a case where A1 match has precedence over B1 match and the output signal transitions to EDPOL.

17.4.4.4.17 Center Aligned Output Pulse Width Modulation, Buffered Mode (OPWMCB) (MPC5553 Only)

This mode generates a center aligned PWM with dead time insertion on the leading or trailing edge. A1 and B1 registers are double buffered to allow smooth output signal generation when changing A2 or B2 values on the fly.

The selected counter bus for a channel configured to OPWMCB mode must be another channel running in MCB up/down counter mode (refer to Section 17.4.4.4.15, "Modulus Counter, Buffered Mode (MCB) (MPC5553 Only)"). Register A1 contains the ideal duty cycle for the PWM signal and is compared with the selected time base. Register B1 contains the dead time value and is compared against the internal counter. For a leading edge dead time insertion, the output PWM duty cycle is equal to the difference between register A1 and register B1, and for a trailing edge dead time insertion, the output PWM duty

cycle is equal to the sum of register A1 and register B1. The MODE[6] bit selects between trailing and leading dead time insertion, respectively.

NOTE

It is recommended that the internal prescaler of the OPWMCB channel be set to the same value as the MCB channel prescaler, and the prescalers should also be synchronized. This allows the A1 and B1 registers to represent the same time scale for duty cycle and dead time insertion.

Figure 17-46 illustrates loading of the A1 and B1 registers, which occurs when the selected counter bus reaches the value one. This counter value defines the cycle boundary. Values written to A2 or B2 within cycle (*n*) are loaded into A1 or B1 registers and are used to generate matches in cycle (n+1).

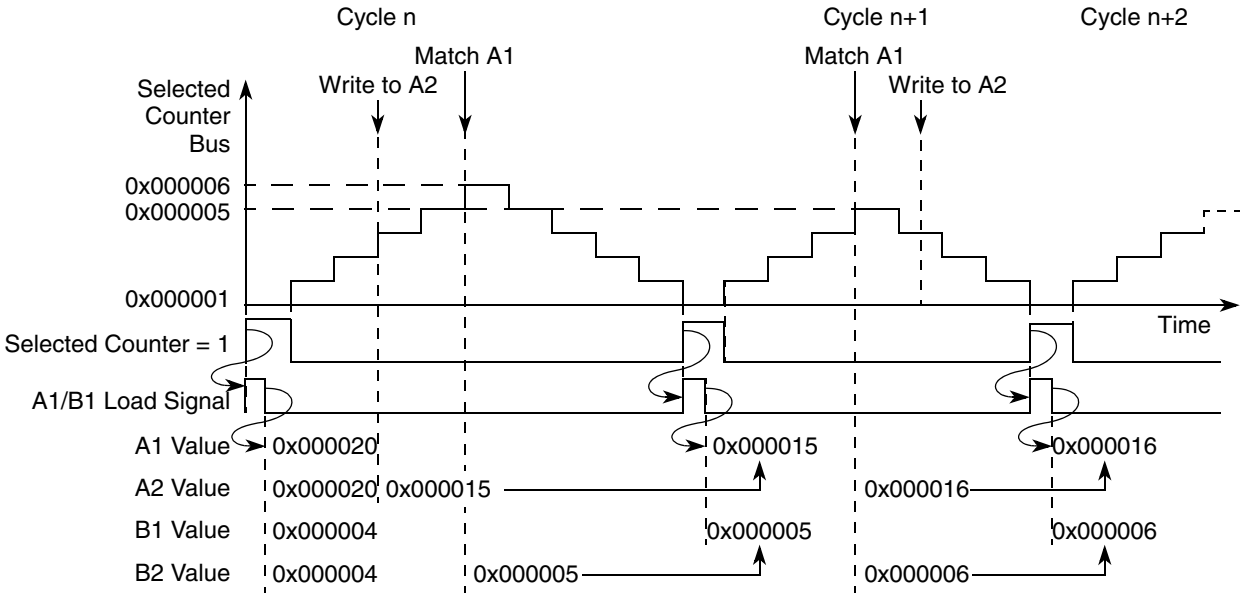


Figure 17-46. eMIOS OPWMCB Mode Example — A1/B1 Register Loading

The EMIOS_OUDR[n] bit can be used to disable the A1 and B1 updates, thus allowing the loading of these registers to be synchronized with the load of A1 or B1 registers in others channels. Note that by using the update disable bit, the A1 and B1 registers can be updated in the same counter cycle.

In this mode A1 matches set the internal counter to one. When operating with leading edge dead time insertion, the first A1 match resets the internal counter to 0x000001. When a match occurs between register B1 and the internal time base, the output flip-flop is set to the value of the EDPOL bit. In the following match between A1 and the selected time base, the output flip-flop is set to the complement of the EDPOL bit. This sequence repeats continuously. Figure 17-47 shows two cycles of a center aligned PWM signal. Note that both A1 and B1 register values are changing within the same cycle, which allows the duty cycle and dead time values to be changed at simultaneously.

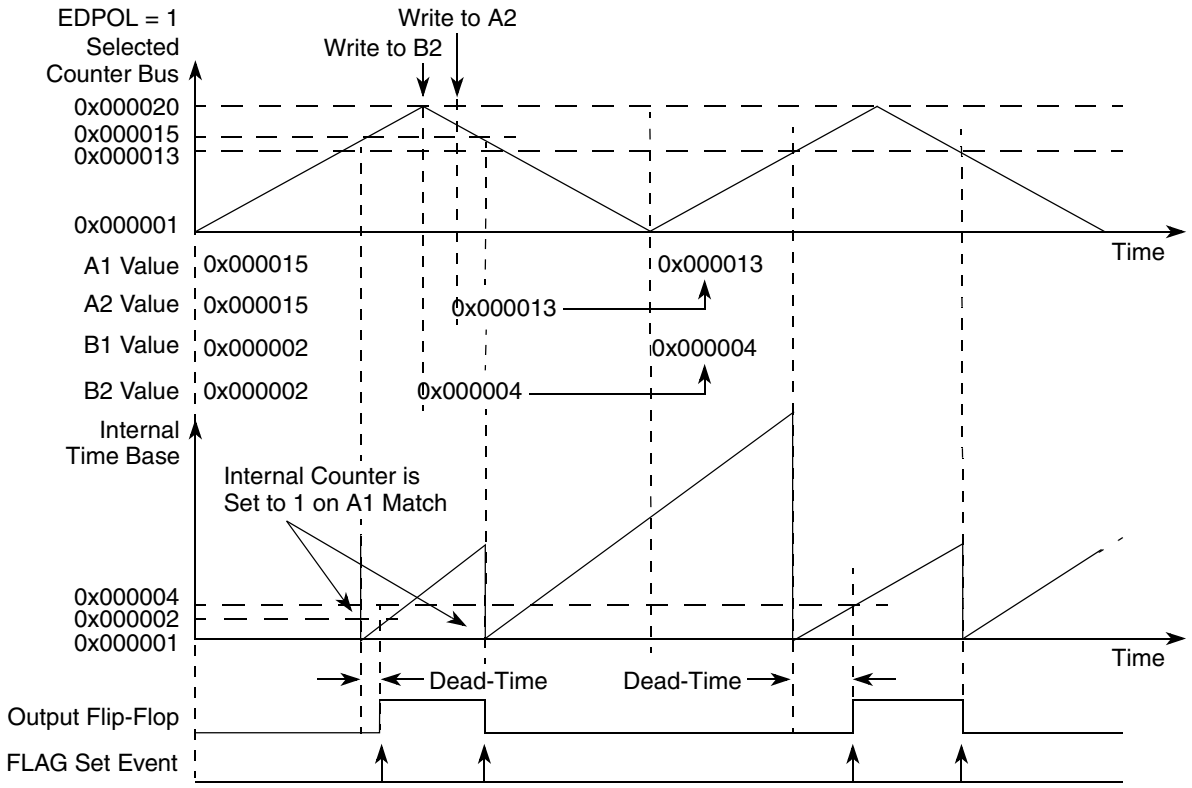


Figure 17-47. eMIOS PWMCB Mode Example — Lead Dead Time Insertion

As shown in Figure 17-48, when operating with trailing edge dead time insertion the first match between A1 and the selected time base sets the output flip-flop to the value of the EDPOL bit and resets the internal counter to 0x000001. In the second match between register A1 and the selected time base, the internal counter is reset to 0x000001 and B1 matches are enabled. When the match between register B1 and the selected time base occurs the output flip-flop is set to the complement of the EDPOL bit. This sequence repeats continuously.

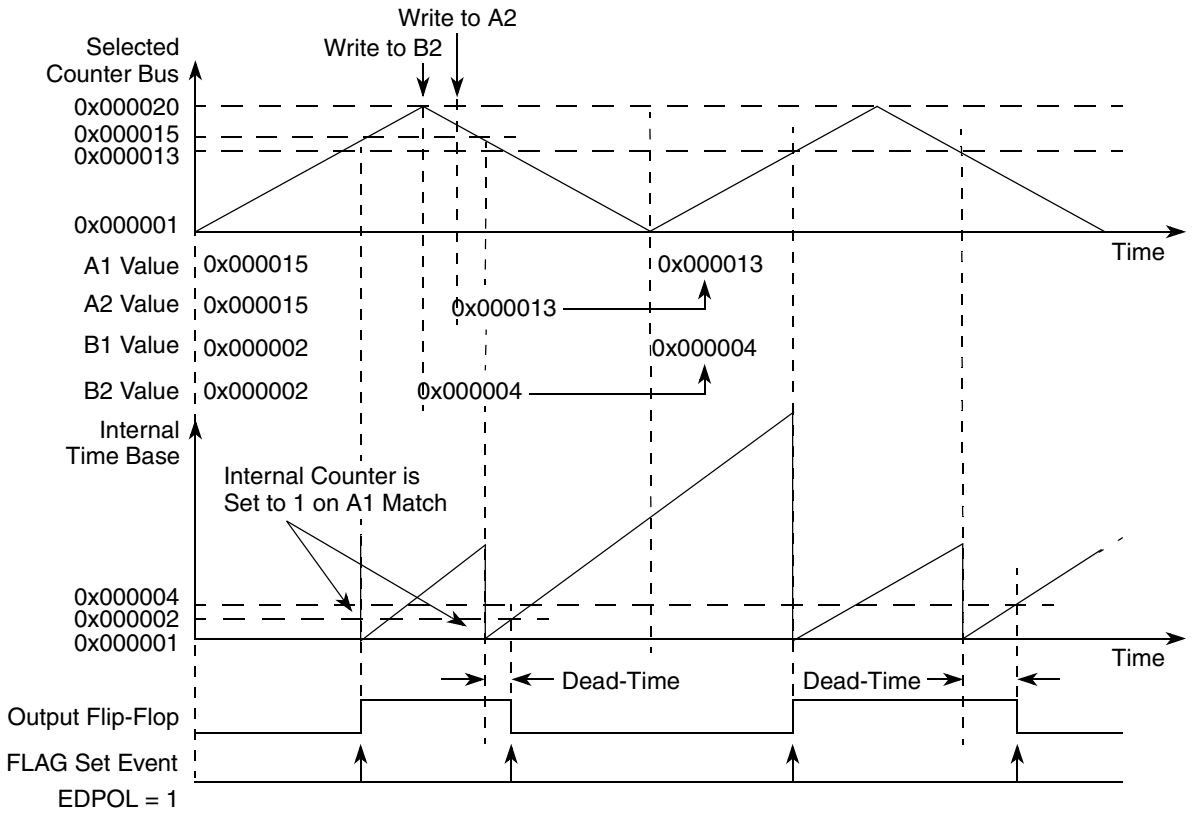


Figure 17-48. eMIOS PWMCB Mode Example — Trailing Dead Time Insertion

FLAG can be generated in the trailing edge of the output PWM signal when MODE[5] is cleared, or on both edges when MODE[5] is set. If subsequent matches occur on A and B, the PWM pulses continue to be generated, regardless of the state of the FLAG bit.

NOTE

In OPWMCB mode, FORCMA and FORCMB do not have the same behavior as a regular match. Instead they force the output flip-flop to a constant value which depends upon the selected dead time insertion mode, lead or trail and the value of the EDPOL bit.

FORCMA has different behaviors depending on the selected dead time insertion mode. In leading dead time insertion mode, writing one to FORCMA sets the output flip-flop to the compliment of EDPOL. In trailing dead time insertion mode, the output flip-flop is forced to the value of EDPOL.

If FORCMB is set, the output flip-flop value depends on the selected dead time insertion mode. In leading dead time insertion mode, FORCMB sets the output flip-flop to the value of EDPOL. In trailing dead time insertion mode, the output flip-flop is forced to the compliment of EDPOL.

NOTE

Setting the FORCMA bit does not reset the internal time base to 0x000001 as a regular A1 match does. FORCMA and FORCMB have the same behavior even in freeze or normal mode regarding the output flip-flop transition.

The FLAG bit is not set in the case of the FORCMA, FORCMB or both bits being set at the same time.

When FORCMA and FORCMB are both set, the output flip-flop is set to the compliment of the EDPOL bit. This is equivalent to FORCMA having precedence over FORCMB when lead dead time insertion is selected and FORCMB having precedence over FORCMA when trailing dead time insertion is selected.

Duty cycles from 0% to 100% can be generated by setting appropriate A1 and B1 values relative to the period of the external time base. Setting A1 = 1 generates a 100% duty cycle waveform. If A1 > period ÷ 2, where period refers to the selected counter bus period, then a 0% duty cycle is produced. Assuming EDPOL is one and OPWMCB mode with trailing dead time insertion mode is selected, 100% duty cycle signals can be generated if B1 occurs at or after the cycle boundary (external counter = 1).

NOTE

A special case occurs when A1 is set to the external counter bus period ÷ 2, which is the maximum value of the external counter. In this case the output flip-flop is constantly set to the EDPOL bit value.

Internal channel logic prevents matches from one cycle to propagate to the next cycle. In trailing dead time insertion mode, a B1 match from cycle (*n*) could eventually cross the cycle boundary and occur in cycle (n+1). In this case the B1 match is masked out and does not cause the output flip-flop to transition. Therefore matches in cycle (n+1) are not affected by the late B1 matches from cycle (*n*).

Figure 17-49 shows a 100% duty cycle output signal generated by setting A1 = 4 and B1 = 3. In this case the trailing edge is positioned at the boundary of cycle (n+1), which is actually considered to belong to cycle (n+2) and therefore does not cause the output flip-flip to transition.

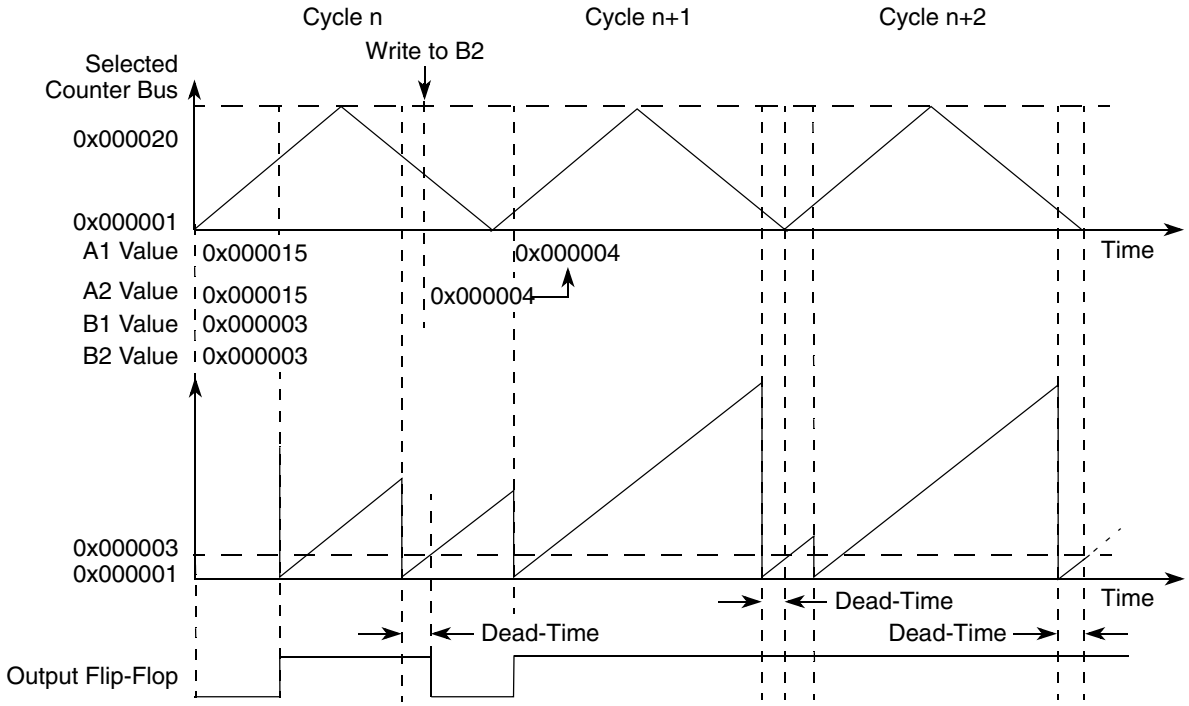


Figure 17-49. eMIOS PWMCB Mode Example — 100% Duty Cycle (A1 = 4, B1 = 3)

The output disable input, if enabled, causes the output flip-flop to transition to the compliment of EDPOL. This allows to the channel output flip-flop to be forced to a safety state. The internal channel matches continue to occur in this case, thus generating flags. When the output disable is negated, the channel output flip-flop is again controlled by A1 and B1 matches. This process is synchronous, meaning that the output channel pin transitions only occur on system clock edges.

It is important to note that, like in OPWMB and OPWFMB modes, the match signal used to set or clear the channel output flip-flop is generated on the negation of the channel comparator output signal which compares the selected time base with A1 or B1. Refer to Figure 17-41, which illustrates the delay from matches to output flip-flop transition in OPWFMB mode.

17.4.4.4.18 Output Pulse Width Modulation, Buffered Mode (OPWMB) (MPC5553 Only)

OPWMB mode is used to generate pulses with programmable leading and trailing edge placement. An external counter is selected from one of the counter buses. The A1 register value defines the first edge and B1 defines the second edge. The output signal polarity is defined by the EDPOL bit. If EDPOL is zero, a negative edge occurs when A1 matches the selected counter bus and a positive edge occurs when B1 matches the selected counter bus.

The A1 and B1 registers are double buffered and updated from A2 and B2, respectively, at the cycle boundary. The load operation is similar to the OPWFMB mode. Refer to Figure 17-43 for more information on A1 and B1 register updates.

Flags are generated at B1 matches when MODE[5] is cleared, or on both A1 and B1 matches when MODE[5] is set. If subsequent matches occur on comparators A and B, the PWM pulses continue to be generated regardless of the state of the FLAG bit.

The FORCMA and FORCMB bits allow software to force the output flip-flop to the level corresponding to a match on A1 or B1 respectively. FLAG is not set by the FORCMA and FORCMB operations.

The following rules apply to the OPWMB mode:

- B1 matches have precedence over A1 matches if they occur at the same time within the same counter cycle.
- A1 = 0 match from cycle (*n*) has precedence over a B1 match from cycle (n-1).
- A1 matches are masked if they occur after a B1 match within the same cycle.
- Values written to A2 or B2 on cycle (*n*) are loaded to A1 or B1 at the following cycle boundary (assuming EMIOS_OUDR[*n*] is not asserted). Thus the new values will be used for A1 and B1 matches in cycle (*n*+1).

Figure 17-50 illustrates operation in OPWMB mode with A1/B1 matches and the transition of the channel output flip-flop. In this example EDPOL is zero.

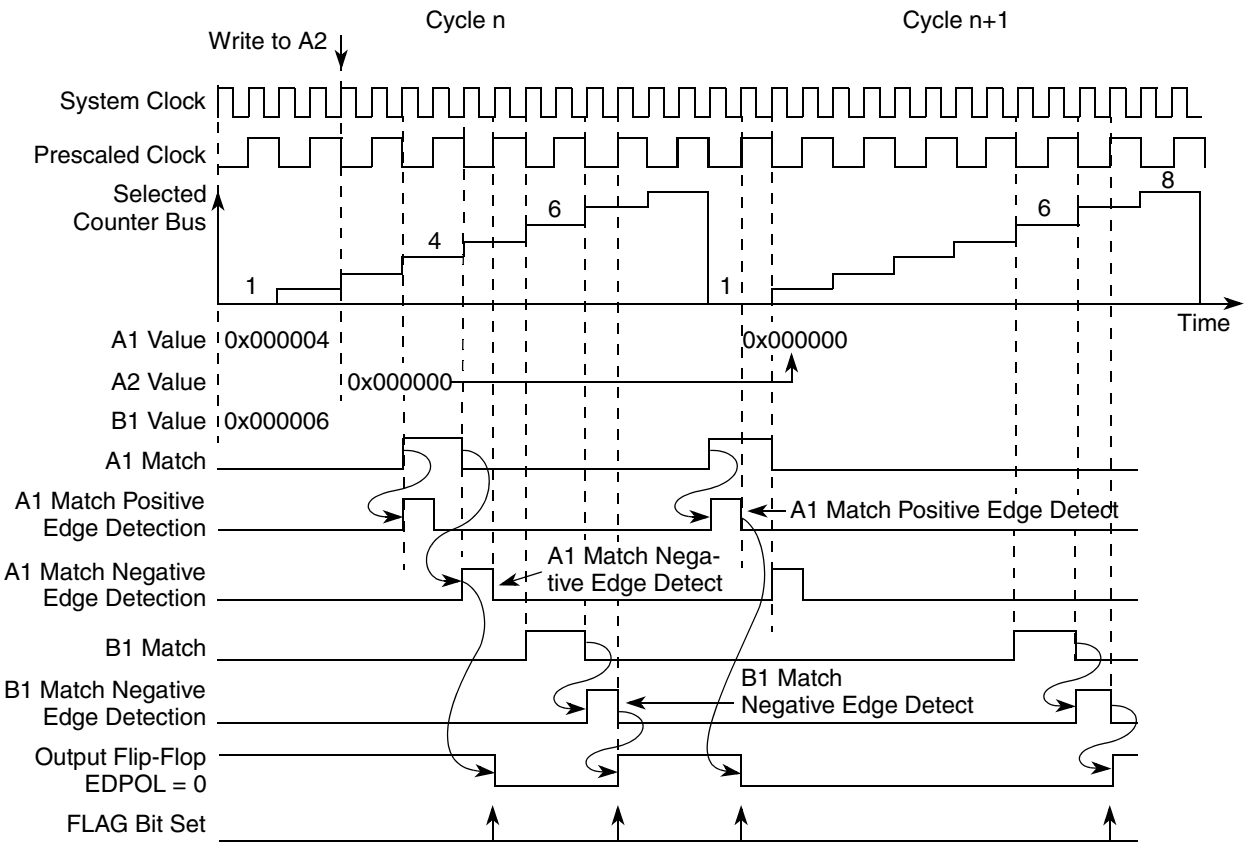


Figure 17-50. eMIOS OPWMB Mode Example — Matches and Flags

Note that the output flip-flop transitions are based on the negative edges of the A1 and B1 match signals. Figure 17-50 shows the value of A1 being set to zero in cycle (n+1). In this case the match positive edge is used instead of the negative edge to transition the output flip-flop.

Figure 17-51 illustrates the channel operation for 0% duty cycle. Note that the A1 match signal positive edge occurs at the same time as the B1 = 8 signal negative edge. In this case the A1 match has precedence over the B1 match, causing the output flip-flop to remain at the EDPOL value, thus generating a 0% duty cycle.

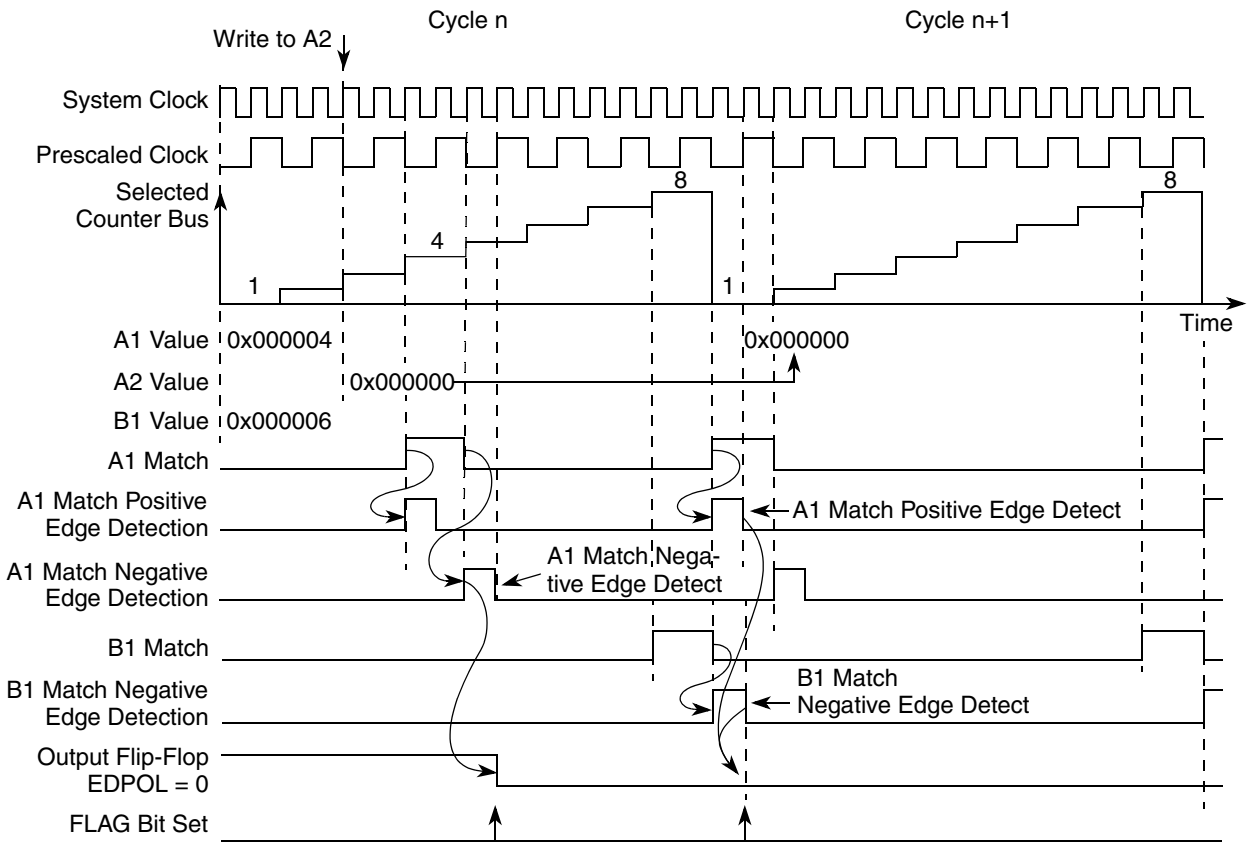


Figure 17-51. eMIOS OPWMB Mode Example — 0% Duty Cycle

Figure 17-52 shows the operation of the OPWMB mode with the output disable signal asserted. The output disable forces a transition in the output flip-flop to the EDPOL bit value. After the output disable is negated, the output flip-flop is allowed to transition at the next A1 or B1 match. The output disable does not modify the flag bit behavior. Note that there is one system clock delay between the assertion of the output disable signal and the transition of the output flip-flop.

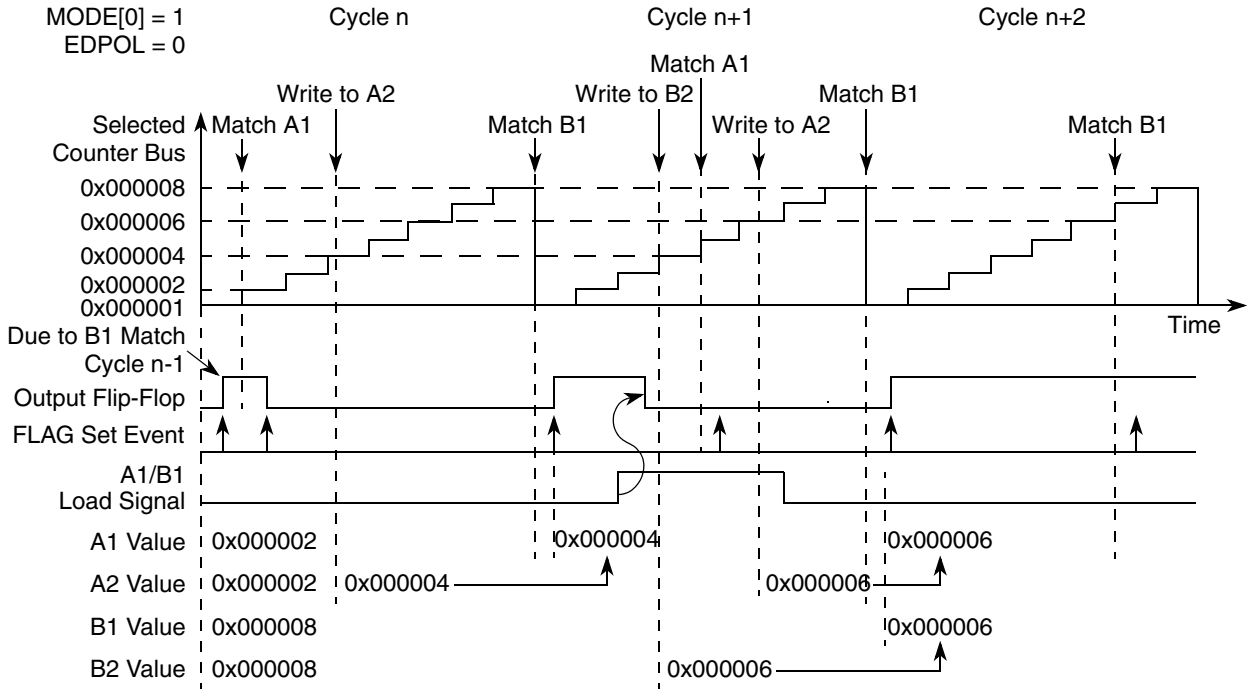


Figure 17-52. eMIOS OPWMB Mode Example — Active Output Disable

Figure 17-53 shows a waveform changing from 100% to 0% duty cycle. In this case EDPOL is zero and B1 is set to the same value as the period of the selected external time base.

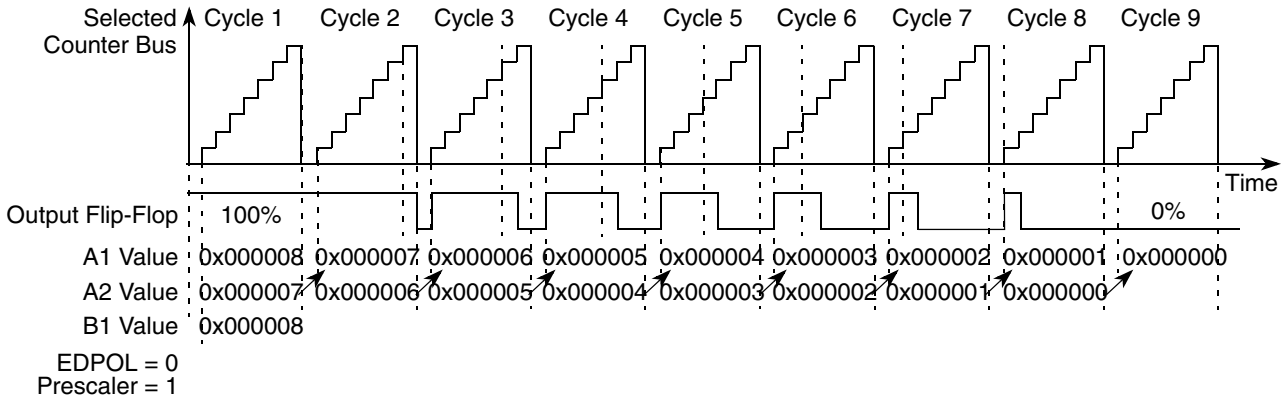


Figure 17-53. eMIOS OPWMB Mode Example — 100% to 0% Duty Cycle

In Figure 17-53 if B1 is set to a value lower than 0x000008 it is not possible to achieve 0% duty cycle by only changing A1 register value. Since B1 matches have precedence over A1 matches, the output flip-flop transitions to the compliment of EDPOL at B1 matches. In this example, if B1 = 0x000009, a B1 match does not occur, and thus a 0% duty cycle signal is generated.

17.5 Initialization / Application Information

Upon reset all of the unified channels of the eMIOS default to general purpose inputs (GPIO input mode).

17.5.1 Considerations on Changing a UC Mode

Before changing an operating mode, the UC must be programmed to GPIO mode, and EMIOS_CADR*n* and EMIOS_CBDR*n* must be updated with the correct values for the next operating mode. Then the EMIOS_CCR*n* can be written with the new operating mode. If a UC is changed from one mode to another without performing this procedure, the first operating cycle of the selected time base is unpredictable.

NOTE

When interrupts are enabled and an interrupt is generated, the FLAG bits should be cleared before exiting the interrupt service routine.

17.5.2 Generating Correlated Output Signals

Correlated output signals can be generated by all output operating modes. Bits ODIS*n* can be used to control the update of these output signals.

In order to guarantee that the internal counters of correlated channels are incremented in the same clock cycle, the internal prescalers must be set up before enabling the global prescaler. If the internal prescalers are set after enabling the global prescaler, the internal counters may increment in the same ratio, but at a different clock cycle.

When an output disable condition occurs, the software interrupt routine must service the output channels before servicing the channels running SAIC. This procedure avoid glitches in the output pins.

17.5.3 Time Base Generation

For all channel operation modes that generate a time base (MC, OPWFM, OPWM, MCB, OPWFMB and OPWMB), the clock prescaler can use several ratios calculated as:

$$\text{Ratio} = (\text{GPRE} + 1) \times (\text{UCPRE} + 1)$$

The prescaled clocks in Figure 17-55, Figure 17-56, and Figure 17-57 illustrate this ratio. For example, if the ratio is 1, the prescaled clock is high and continuously enables the internal counter (EMIOS_CCNTR*n*) (Figure 17-55); if the ratio is 3, then it pulses every 3 clock cycles (Figure 17-56) and the internal counter increments every 3 clock cycles; if the ratio is 9, it pulses every 9 clock cycles, etc. This high pulse enables the EMIOS_CCNTR*n* to increment as long as no other conditions disable this counter. The match signal is generated by pulsing every time the internal counter matches the programmed match value. Note that for the same programmed match value, the period is shorter when using a prescaler ratio greater than one.

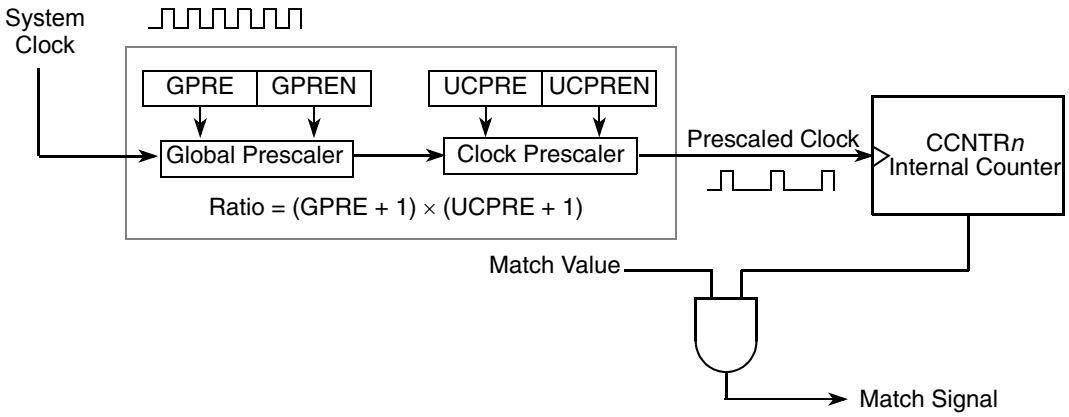
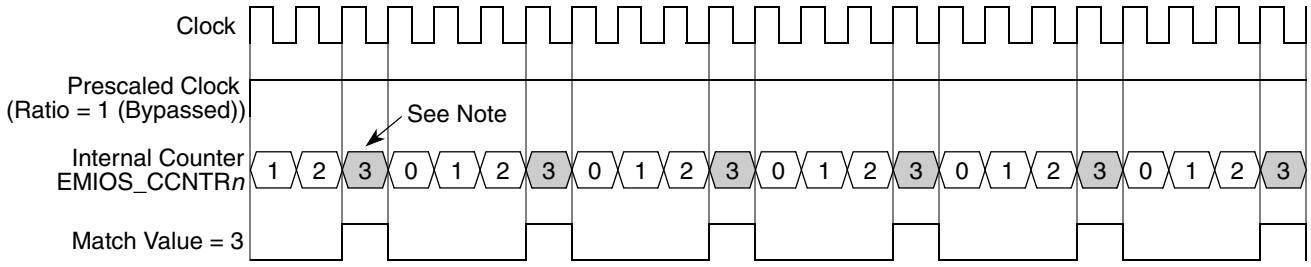
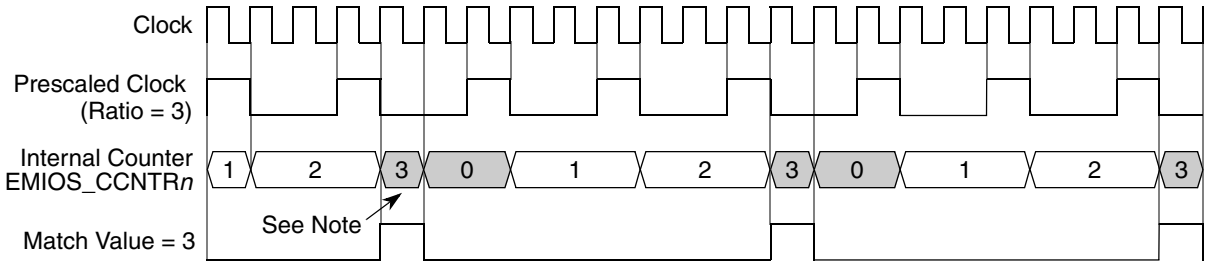


Figure 17-54. eMIOS Time Base Generation Block Diagram



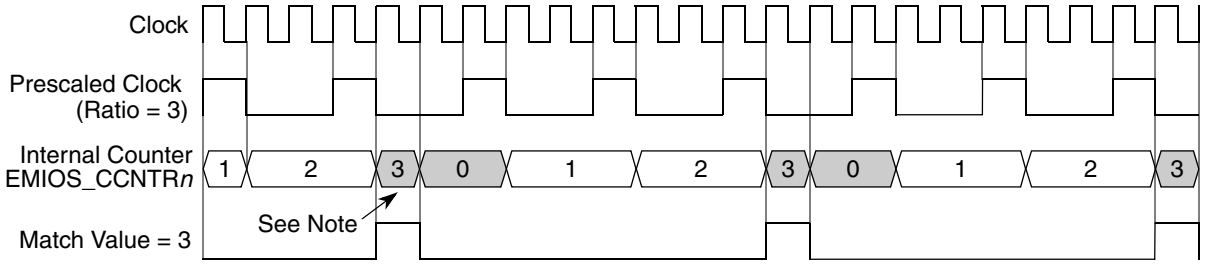
NOTE: The period of the time base includes the match value. When a match occurs, the first clock cycle is used to clear the internal counter, starting another period

Figure 17-55. eMIOS Time Base Example — Fastest Prescaler Ratio



NOTE: The period of the time base does not include the match value. When a match occurs, the first clock cycle is used to clear the internal counter, starting another period

Figure 17-56. eMIOS Time Base Example — Prescale Ratio = 3, Match Value = 3



NOTE: The period of the time base does not include the match value. When a match occurs, the first clock cycle is used to clear the internal counter, starting another period

Figure 17-57. eMIOS Time Base Example — Prescale Ratio = 2, Match Value = 5

17.6 Revision History

Substantive Changes since Rev 3.0
• Corrected footnote in MC, OPWM, OPWFM, and OPWMC mode figures. • Added note regarding FORCMA and FORMCB bits having no effect in some modes. • Added note regarding immediate update of A register in OPW and OPWFM modes. • Added noteregarding immediate update of A register in MC mode. • Added note regarding immediate update of A and B registers in OPWMC mode.
Added note to BSL bit definition “In certain modes the internal counter is used internally and therefore cannot be used as the channel time base.”
Added Section 17.4.4.4.1, “General Purpose Input/Output Mode (GPIO).”

Chapter 18
Enhanced Time Processing Unit (eTPU)

18.1 Introduction

The enhanced time processing unit (eTPU) is a new timing unit featured on the MPC5553/MPC5554 microcontroller that operates in parallel with the MPC5553/MPC5554 core (CPU). The eTPU does the following:

- Executes programs independently from the host core
- Detects and precisely records timing of input events
- Generates complex output waveforms
- Is controlled by the core without a requirement for real-time host processing

The host core setup and service times for each input and output event are greatly minimized. The MPC5554 contains two eTPUs, and the MPC5553 contains one.

The eTPU improves the performance of the MPC5553/MPC5554 by providing high resolution timing:

- eTPU dedicated channels that include two match and two capture registers, as opposed to the previous generation TPUs which only had one of each register
- eTPU engines that are optimized with specific instructions to service channel hardware
- The fast instruction execution rate of the eTPU engine that reduces service time

Because responding to hardware service requests is primarily done by the eTPU engine, the host is free to handle higher level operations.

18.1.1 The MPC5553/MPC5554 eTPU Implementation

For more detailed information regarding the eTPU module and compiler, refer to the *Enhanced Time Processing (eTPU) Reference Manual.* The MPC5553/MPC5554 devices contain a specific implementation of the eTPU's full functionality. This chapter will focus only on an eTPU overview and those details that are different than the full instantiation of the module. These differences include the following:

- 3 (MPC5554) or 2.5 (MPC5553) Kbytes of shared data memory (SDM). This memory is alternately referred to as eTPU shared parameter (data) RAM (SPRAM).
- 16 Kbytes (MPC5554) or 12 Kbytes (MPC5553) of shared code memory (SCM).
- For the MPC5553, only one eTPU engine: eTPU A in the eTPU reference manual. Ignore any references to eTPU B.
- The eTPU debug interface is built into the MPC5553/MPC5554's debug module. Refer to Section 10.2.1 of the eTPU reference manual for details on eTPU debug.
- Data transfer requests are implemented as a single DMA request to the MPC5553/MPC5554's DMA controller. All 32 channels' data transfer request signals are logically OR'd to produce the single DMA request.
- I/O channel pairs may be shared on a common pin. The output buffer enable (OBE) is not used in the MPC5553/MPC5554. The outputs are enabled in the SIU; refer to Chapter 6, "System Integration Unit (SIU)."

Because of the above differences between the MPC5553/MPC5554's implementation of the eTPU and the full eTPU, full register bit descriptions are included within this chapter as well as in the *Enhanced Time Processing (eTPU) Reference Manual.*

18.1.2 Block Diagram

Figure 18-1 shows a top-level eTPU block diagram. It displays the MPC5554's dual eTPU engine configuration.

NOTE

The MPC5553 has a single eTPU engine configuration, and the MPC5554 has two.

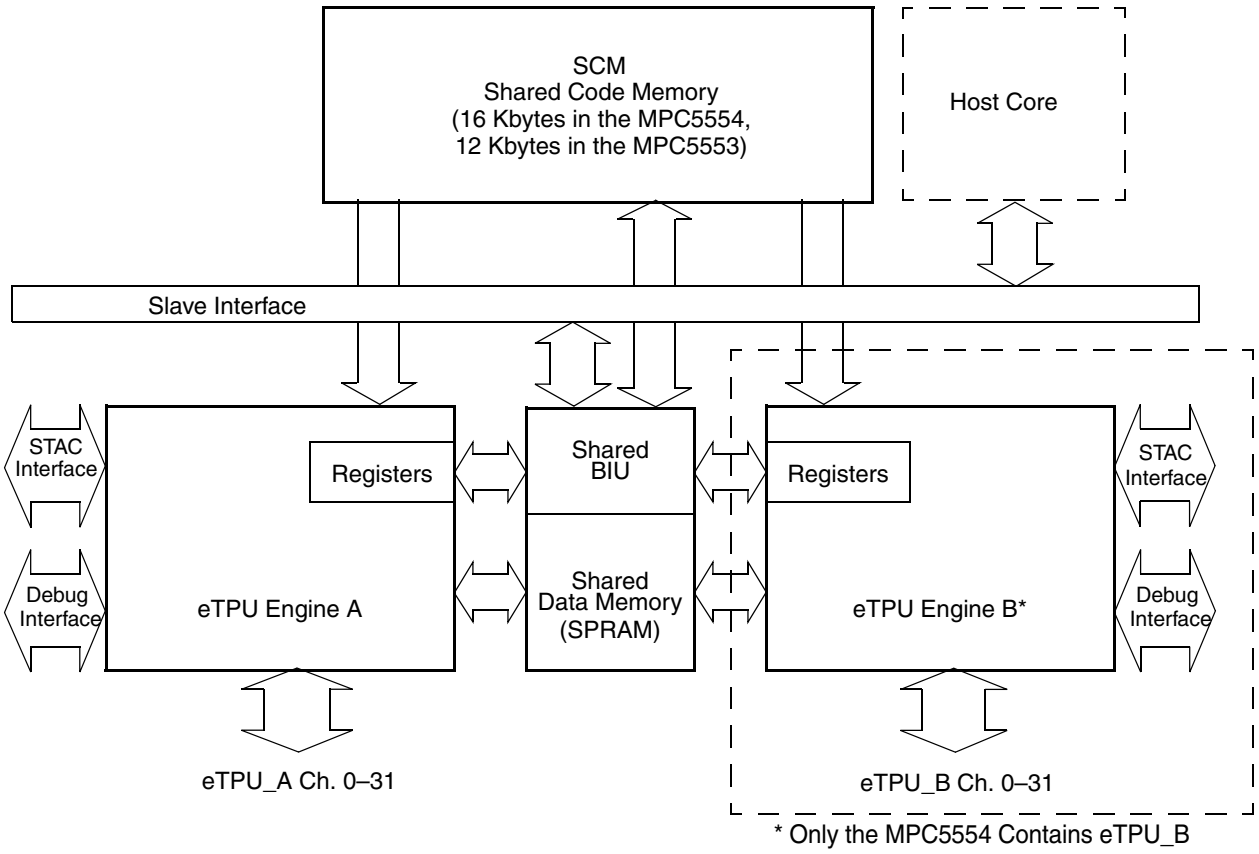


Figure 18-1. eTPU Block Diagram

Figure 18-2 shows the block diagram for the eTPU engine.

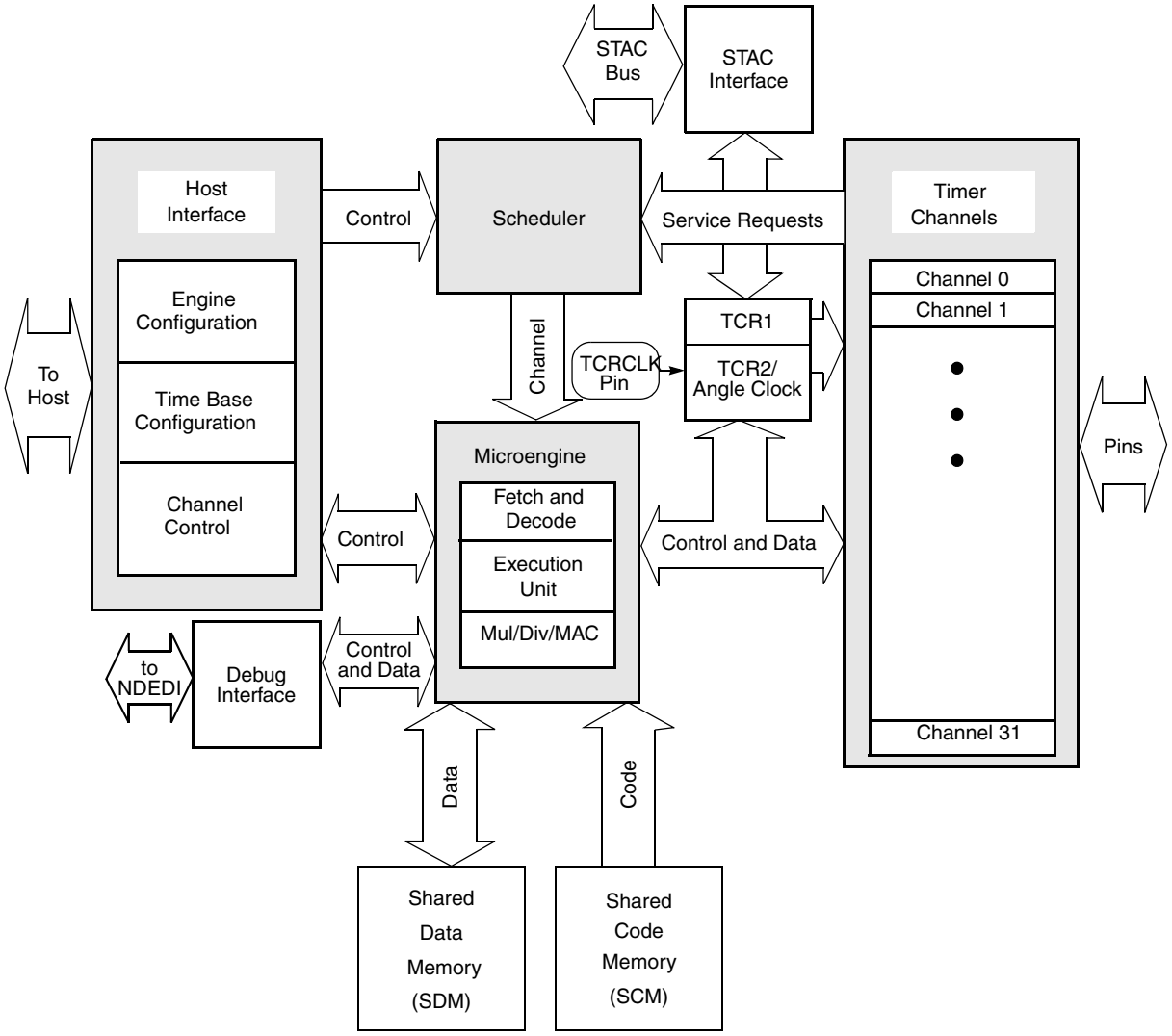


Figure 18-2. eTPU Engine Block Diagram

18.1.3 eTPU Operation Overview

The eTPU is a real-time microprocessed subsystem. Therefore it runs microengine code from instruction memory (SCM) to handle specific events and accesses data memory (SDM) for parameters, work data, and application state information. Events may originate from I/O channels (due to pin transitions and/or time base matches), MPC5553/MPC5554 core requests, or inter-channel requests. Events that call for local eTPU processing activate the microengine by issuing a service request. The service request microcode may send an interrupt to the MPC5553/MPC5554 core, but the core cannot be directly interrupted by I/O channel events.

Each channel is associated with a function that defines its behavior. A function is a software entity consisting of a set of microengine routines, called threads, that respond to eTPU service requests. Function routines, which reside in the SCM, are also responsible for channel configuration. A function may be

assigned to several channels, but a channel can only be associated with one function at a given moment. The eTPU has the capability to change the function assigned to a channel if reconfigured by the MPC5553/MPC5554 core. The association between functions and channels is defined by the MPC5553/MPC5554 core.

The eTPU hardware supplies resource sharing features that support concurrency:

- A hardware scheduler dispatches the service request microengine routines based on a set of priorities defined by the MPC5553/MPC5554's core. Each channel has its own unique priority assignment that primarily depends on CPU assignment. The channel's number is an inherent property also used to determine priority.
- A service request routine cannot be interrupted by another service request until it ends, that is, until an end instruction is executed. This sequence of uninterrupted instruction execution is called a thread. A thread may be interrupted only by resetting the entire eTPU module.
- Channel-specific contexts (registers and flags) are automatically switched between the end of a thread and the beginning of the next one.
- SDM arbitration, a dual-parameter coherency controller, and semaphores can be used to ensure coherent access to eTPU data shared by both eTPU engines and the MPC5553/MPC5554 core.

18.1.3.1 eTPU Engine

The eTPU engine processes input pin transitions and generates output pin waveforms. These events are triggered by eTPU timers (time bases) that are driven by a system clock to give absolute time control or by an asyncronous counter such as an angle clock that may be tracking the angle of a rotating shaft.

Each eTPU engine consists of the following blocks: 32 independent timer channels, a task scheduler, a host interface, and a microprocessor (hereinafter called a microengine) that has dedicated hardware for input signal processing and output signal generation over the 32 I/O channels. Each channel can also choose between two 24-bit counter registers for a time base.

The microengines fetch microinstructions from shared code memory (SCM). eTPU application parameters and global and local variables, referred to as work data, are held in 32-bit shared data memory (SDM), which is also used for passing information between the MPC5553/MPC5554's core and both (or one) microengines. The bus interface unit (BIU) allows the MPC5553/MPC5554's core to access eTPU registers, SDM, and SCM.

The blocks of an eTPU engine are duplicated in a dual eTPU configuration. eTPU engines A and B are often referred to as eTPU A and eTPU B in this document.

18.1.3.1.1 Time Bases

Each eTPU engine has two 24-bit count registers TCR1 and TCR2 that provide reference time bases for all match and input capture events. Prescalers for both time bases are controlled by the MPC5553/MPC5554 core through bit fields in the eTPU engine configuration registers.

The values for each of TCR1 and TCR2 counter registers can be independently derived from the system clock or from an external input via the TCRCLK pin. In addition, the TCR2 time base can be derived from special angle-clock hardware that enables implementing angle-based functions. This feature is added to support advanced angle-based engine control applications.

The TCRs may also drive or be driven by an eMIOS time base through the shared time and counter (STAC) bus, or they may be written by eTPU function software.

18.1.3.1.2 eTPU Timer Channels

Each eTPU engine has 32 identical, independent channels. Each channel corresponds to an input/output signal pair. Every channel has access to two 24-bit counter registers, TCR1 and TCR2.

Each channel consists of event logic which supports a total of four events, two capture and two match events. The event logic contains two 24-bit capture registers and two 24-bit match registers. The match registers are compared to a selected TCR by greater-than-or-equal-to and equal-only comparators. The match and compare register pairs enable many combinations of single and double-action functions.

The channel configuration can be changed by the microengine. Each channel can perform double capture, double match or a variety of other capture-match combinations. Service requests may be generated on one or both of the match events and/or on one of the capture events.

Digital filters that have different filtering modes are provided for the input signals.

Every channel can use any time base or angle counter for either match or capture operation. For example, a match on TCR1 can capture the value of TCR2. The channels can request service from the microengine due to recognized pin transitions (input events) or time base matches.

Every eTPU channel may be configured with the following combinations:

- Single input capture, no match (TPU3 functionality)
- Single input capture with single match time-out (TPU3 functionality)
- Single input capture with double match time-out with several double match submodes
- Double input capture with single or double match time-out with several double match submodes
- Single output match (TPU3 functionality)
- Double output match with several double match submodes
- Input-dependent output generation

The double match functionality has various combinations for generation of service request and determining pin actions.

18.1.3.1.3 Host Interface

The engine's host interface allows the MPC5553/MPC5554 core to control the operation of the eTPU. In order for the eTPU to start operation, the MPC5553/MPC5554 core must initialize the eTPU by writing to the appropriate host interface registers to assign a function and priority to each channel. In addition, the MPC5553/MPC5554 core writes to the host service request and channel configuration registers to further define operation for each initialized channel.

NOTE

The host transfers the code image for the eTPU microcode to the SCM, then the host enables eTPU access to the SCM (which also disables host access).

18.1.3.1.4 Shared Data Memory (SDM)

The SDM works as data RAM that can be accessed by the MPC5553/MPC5554 core and up to two eTPU engines. This memory is used for either:

- Information transfer between the MPC5553/MPC5554 core and the eTPU
- Data storage for the eTPU microcode program
- Communication between the two eTPU engines

The SDM width is 32 bits, and is accessible by the host in any of the three formats: byte, 16-bit, or 32-bit. The eTPU can access the SDM's full 32 bits, lower 24 bits or upper byte (8-bit).

The host can also access the SDM space mirrored in an alternate area with parameter sign extension (PSE). PSE allows for 24-bit data to be accessed as 32 bit sign-extended data without using the MPC5553/MPC5554's bandwidth to extend the data.

Parameter signal extension accesses differ from the usual host accesses to the original SDM area as follows:

- Writes are effective only to the lower 3 bytes of a word: the word's most significant byte (byte address) is kept unaltered in SDM.

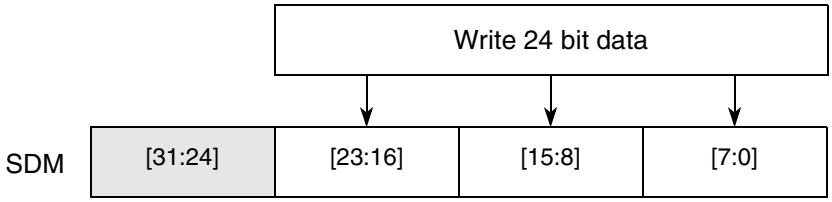


Figure 18-3. SDM Write

NOTE

For the most significant byte, it should be recalled that the word format is big endian, as in the default PowerPC word format.

- Reads return the lower 3 bytes of a word sign-extended to 32 bits, that is: the most significant bit of the word's second most significant byte (byte addresses) is copied in all 8 bits of the most significant read byte.

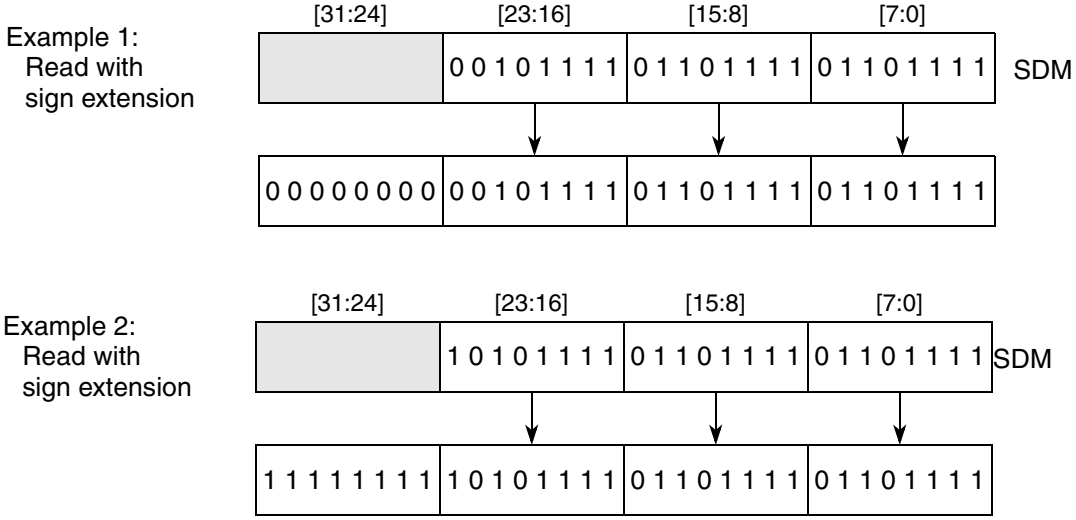


Figure 18-4. PSE Accesses

Each eTPU channel can be associated with a variable number of parameters located in the SDM, according to its selected function. In addition, the SDM can be fully shared between two eTPU engines, enabling communication between them. Each function may require a different number of parameters. During eTPU initialization the host has to program channel base addresses, allocating proper parameters for each channel according to its selected function.

In the host address space each parameter occupies four bytes (32 bits). eTPU usage of the upper byte is achieved by having a 32-bit Preload (P) register that can access the upper byte, the lower 24 bits, or all the 32 bits. The microcode can switch between access sizes at any time.

Each function may require a different number of parameters. During the eTPU initialization the host has to program channel base addresses, allocating proper parameters for each channel according to its selected function.

18.1.3.1.5 Task Scheduler

As mentioned in Section 18.1.3, "eTPU Operation Overview" every channel function is composed of one or more threads, and threads cannot be interrupted by host or channel events, such as channel servicing. The function of the task scheduler, therefore, is to recognize and prioritize the channels needing service and grant execution time to each channel. The time given to an individual thread for execution or service is called a time slot. The duration of a time slot is determined by the number of instructions executed in the thread plus SDM wait-states received, and varies in length. Although several channels may request service at the same time, the function threads must be executed serially.

At any time, an arbitrary number of channels can require service. The channel logic, eTPU microcode, or the host application notifies the scheduler by issuing a service request.

Out of reset, all channels are disabled. The MPC5553/MPC5554 core makes a channel active by assigning it one of three priorities: high, middle, or low. The scheduler determines the order in which channels are serviced based on channel number and assigned priority. The priority mechanism, implemented in hardware, ensures that all requesting channels are serviced.

18.1.3.1.6 Microengine

The eTPU microengine is a simple RISC implementation that performs each instruction in a microcycle of two system clocks, while pre-fetching the next instruction through an instruction pipeline. Instruction execution time is constant for the arithmetic logic unit (ALU) unless it gets wait states from SDM arbitration.

Microcode is stored in shared code memory (SCM) that is 32 bits wide. The microengine instruction set provides basic arithmetic and logic operations, flow control (jumps and subroutine calls), SDM access, and channel configuration and control. The instruction formats are defined in such a way that allow particular combinations of two or three of these operations with unconflicting resources to be executed in parallel in the same microcycle, thus improving performance.

The microengine also has an independent multiply/divide/MAC unit that performs these complex operations in parallel with other microengine instructions.

Channel functionality is integrated to the instruction set through channel control operations and conditional branch operations, which support jumps/calls on channel-specific conditions. This allows quick and terse channel configuration and control code, contributing to reduced service time.

18.1.3.1.7 Dual eTPU Engine System (MPC5554 Only)

The MPC5554 eTPU implementation includes two eTPU engines sharing SDM and the same code in the SCM.

The two eTPU engines share the bus interface unit (BIU) and the shared data memory (SDM). This allows the MPC5554 core to communicate with the eTPU and also provides a means of communication between the eTPU engines. The shared BIU includes coherency logic which supports dual parameter (8 bytes) coherency in transfers between the host and eTPU, using a temporary parameter area within the SDM.

18.1.3.2 Debug Interface

Nexus level 3 debug support is available through the eTPU Nexus development interface (NDEDI). Refer to Chapter 25, "Nexus Development Interface."

18.1.4 Features

The eTPU includes these distinctive features:

- Up to 32 channels for each eTPU engine: each channel is associated with an I/O signal pair
  - Enhanced input digital filters on the input pins for improved noise immunity. The eTPU digital filter can use two samples, three samples, or work in continuous mode.
  - Orthogonal channels, except for channel 0: each channel can perform any time function. Each time function can be assigned to more than one channel at a given time, so each signal can have any functionality. Channel 0 has the same capabilities of the others, but can also work with special angle counter logic (see below).
  - A link service request allows activation of a channel thread by request of another channel, even between eTPU engines.
  - A host service request allows activation of a channel thread by the MPC5553/MPC5554 core request.
  - Each channel has an event mechanism that supports single and double action functionality in various combinations. It includes two 24-bit capture registers, two 24-bit match registers, 24-bit greater-equal or equal-only comparator.
- Two independent 24-bit time bases for channel synchronization
  - The first time base may be clocked by the system clock with programmable prescaler division from 2 to 512 (in steps of 2), or by the output of the second time base prescaler.
  - The first time base can also be clocked by an external signal with programmable prescaler divisions of 1 to 256.
  - The second time base may be clocked by an external signal with programmable prescaler divisions from 1 to 64 or by the system clock divided by 8.
  - The second time base counter can work as an angle counter, enabling angle-based applications to match angle instead of time.
  - The second time base can alternatively be used as a pulse accumulator gated by an external signal.
  - Either time base can be written or read by either eTPU engine at any time.
  - Either time base can be read, but not written, by the host.
  - Both time bases can be exported or imported from engine to engine through the STAC (shared time and counter) bus.

NOTE

An engine cannot export/import to/from itself. An engine cannot import a time base and/or angle count if it is in angle mode.

- Event-triggered RISC processor (microengine)
  - 2-stage pipeline implementation (fetch and execution), with separate instruction memory (SCM) and data memory (SDM).
  - Two-system-clock microcycle fixed-length instruction execution for the ALU.
  - 16 Kbytes (MPC5554) or 12 Kbytes (MPC5553) of shared code memory (SCM).

  - Interleaved SCM access in dual-engine eTPU (MPC5554) avoids contention in time for instruction memory.
  - 3 (MPC5554) or 2.5 (MPC5553) Kbytes of shared data memory (SDM) with interleaved access in dual (MPC5554) eTPU engine avoids contention for data memory.
  - Instruction set with embedded channel support, including specialized channel control subinstructions and conditional branching on channel-specific flags.
  - Channel-oriented addressing: channel-bound address mode with host configured channel base address allows the same function to operate independently on different channels.
  - Channel-bound data address space of up to 128 32-bit parameters (512 bytes).
  - Global parameter address mode allows access to common channel data of up to 256 32-bit parameters (1024 bytes).
  - Support for indirect and stacked data access schemes.
  - Parallel execution of: data access, ALU, channel control and flow control subinstructions in selected combinations.
  - 32-bit microengine registers and 24-bit resolution ALU, with 1 microcycle addition and subtraction, absolute value, bitwise logical operations on 24-bit, 16-bit, or byte operands: single bit manipulation, shift operations, sign extension and conditional execution.
  - Additional 24-bit multiply/MAC/divide unit which supports all signed/unsigned/ multiply/MAC combinations, and unsigned 24-bit divide. The MAC/divide unit works in parallel with the regular microcode commands.
- Resource sharing features resolve channel contention for common use of channel registers, memory and microengine time
  - Hardware scheduler works as a ‘task management’ unit, dispatching event service routines by predefined, host-configured priority.
  - Automatic channel context switch when a ‘task switch’ occurs; that is, one function thread ends and another begins to service a request from another channel. Channel-specific registers, flags and parameter base address are automatically loaded for the next serviced channel.
  - Individual channel priority setting in three levels: high, middle, and low.
  - Scheduler priority scheme allows calculation of worst case latency for event servicing and ensures servicing of all channels by preventing permanent blockage.
  - SDM shared between host core and both eTPU engines, supporting channel-channel or host-channel communication.
  - Hardware implementation of four semaphores allows for resource arbitration between channels in both eTPU engines.
  - Hardware semaphores are directly supported by the microengine instruction set.
  - Dual-parameter coherency hardware support allows coherent (to host) access to 2 parameters by microengines in back-to-back accesses.
  - Coherent dual-parameter controller allows coherent (to microengines) accesses to two parameters by the host.
- Test and development support features
  - Nexus level 3 debug support through the eTPU Nexus block (NDEDI)
  - Software breakpoints
  - SCM (code memory) continuous signature-check built-in code integrity test multiple input signature calculator (MISC): runs concurrently with eTPU normal operation

18.2 Modes of Operation

The eTPU is capable of working in the following modes.

18.2.1 User Configuration Mode

By having access to the shared code memory (SCM), the core has the ability to program the eTPU cores with time functions.

18.2.2 User Mode

In user mode the core does not access the eTPU shared code memory, and pre-defined eTPU functions are used.

18.2.3 Debug Mode

The core debugs eTPU code, accessing special trace/debug features via Nexus interface:

- Hardware breakpoint/watchpoint setting
- Access to internal registers
- Single-step execution
- Forced instruction execution
- Software breakpoint insertion and removal

18.2.4 Module Disable Mode

eTPU engine clocks are stopped through a register write to ETPU_ECR bit MDIS, saving power. Input sampling stops. eTPU engines can be in stop mode independently. Module disable mode stops only the engine clock, so that the shared BIU and global channel registers can be accessed, and interrupts and DMA requests can be cleared and enabled/disabled. An engine only enters module disable mode when any currently running thread is finished.

These modes are loosely selected: there is no unique register field or signals to choose between them. Some features of one mode can be used with features of other modes.

18.2.5 eTPU Mode Selection

User and user configuration are the production operating modes, and differ from each other only in access to SCM.

Module disable mode is entered by setting ETPU_ECR[MDIS]. eTPU engines can be individually stopped (there is one ETPU_ECR for each engine).

18.3 External Signal Description

18.3.1 Overview

There are 69 external signals associated with each eTPU engine: 32 channel input signals, 32 channel output signals, 4 output disable signals, and a TCRCLK clock input, totalling 138 in a dual engine system, or 69 in a single engine system.

18.3.2 Detailed Signal Description

18.3.2.1 Output and Input Channel Signals

The channel signal connections for eTPU engine A (in both the MPC5553 and the MPC5554) and eTPU engine B (only in the MPC5554) are described in Table 18-1 and Table 18-2, respectively. Each eTPU channel has an input and output associated with it. In Table 18-1 and Table 18-2 this is represented by the Input/output column. The eTPU channels can be connected to external pins or wired internally to other peripheral devices. In the MPC5553/MPC5554, some of the eTPU channels are connected to pins. The pin connections are represented by the Pin Number column in Table 18-1 and Table 18-2. To the right of the Pin Number column is the eTPU Channel Connections column that shows the channel number that corresponds to each input or output pin. Many of these pins are multipurpose, that is they are multiplexed. Table 18-1 and Table 18-2 shows the other non-eTPU signals listed in the Signals with Which eTPU Signal is Shared column.

To reduce the number of pins required by the MPC5553/MPC5554's eTPU while still maintaining the eTPU's functionality, the eTPU is also internally wired to the DSPI (20.1, "Introduction"). The DSPI connections are shown in the column labeled DSPI Serial Channel Connections in Table 18-1 and Table 18-2. The eTPU microcode may be programmed to set the output level of an eTPU channel in one of two manners:

- By forcing the logic level to a specified value
- By specifying the logic level output action when a match or transition event occurs

Every eTPU channel input has a digital filter. This filter is designed to filter out noise pulses that have width less than a specified value. This prevents small noise glitches from being recognized by the transition detect logic. Any pulses wider than the specified filter width will be passed to the channel transition detect logic.

Table 18-1. eTPU A Channel Connection Table (both MPC5553 and MPC5554)

eTPU Channel Number	I/O	Pin Number	eTPU Channel Connections	DSPI Serial Channel Connections	eTPU A Signal	Signals with Which eTPU Signal is Shared:
0-9	I	N3 M4–M1[1] L4–L1[1] K4	0 1–4 5–8 9	not connected DSPI_C[4:13]	eTPU_A[0:9]	eTPU_A[12:21] (output only) GPIO[114:123]
	O	AF15, AE15, AC16, AD15, AF16, AE16, AD16, AF17, AC17, AE17	0–9			eMIOS[0:9] GPIO[179:188]
10–11	I	K3–K2[1]	10–11	not connected	eTPU_A[10:11]	eTPU_A[22:23] (output only) GPIO[124:125]
	O			DSPI_C[14:15]		
12–15	I	K1 J4–J2[1]	12 13–15	not connected DSPI_C[0:3]	eTPU_A[12:15]	GPIO[126:129]
	O	N3 M4–M2[1]	12 13–15			eTPU_A[0:3] GPIO[114:117]

Table 18-1. eTPU A Channel Connection Table (both MPC5553 and MPC5554) (continued)

eTPU Channel Number	I/O	Pin Number	eTPU Channel Connections	DSPI Serial Channel Connections	eTPU A Signal	Signals with Which eTPU Signal is Shared:
16–19	I	J1 H4–H2[1]	16 17–19	not connected	eTPU_A[16:19]	GPIO[130:133]
				DSPI_B[7:4][1] DSPI_D[5:2][1]		
	O	M1 L4–L2[1]	16 17–19			eTPU_A[4:7] GPIO[118:121]
20–21	I	H1 G4	20 21	not connected	eTPU_A[20:21]	IRQ[8:9] (input only) GPIO[134:135]
				DSPI_B[3:2][1] DSPI_D[1:0][1]		
	O	L1 K4	20 21			eTPU_A[8:9] GPIO[122:123]
22–23	I	G2 G1	22 23	not connected	eTPU_A[22:23]	IRQ[10:11] (input only) GPIO[136:137]
	O	K3 K2	22 23			eTPU_A[10:11] GPIO[124:125]
24–27	I	—	not connected	DSPI_B[13:10][1]	eTPU_A[24:27]	not connected
	O	F1, G3 F3, F2	24, 25 26, 27	DSPI_B[13:10][1] DSPI_D[15:12][1]	eTPU_A[24:27]	IRQ[12:15] (input only) GPIO[138:141]
28–29	I	—	not connected	DSPI_B[9:8][1]	eTPU_A[28:29]	not connected
	O	E1 E2	28 29	DSPI_B[9:8][1] DSPI_D[11:10][1]		GPIO[142:143]
30–31	I	D1 D2	30 31	not connected	eTPU_A[30:31]	GPIO[144:145]
	O					

[1] The channel numbers for some of the DSPI channels connections are reversed, for example if eTPU_A[16:19] is mapped to DSPI_B[7:4], then eTPU_A16 is connected to DSPI_B7, eTPU_A17 is connected to DSPI_B6,..., and eTPU_A19 is connected to DSPI_B4

Table 18-2. eTPU B Channel Connection Table (MPC5554 Only)

eTPU Channel Number	I/O	Pin Number	eTPU Channel Connections	DSPI Serial Channel Connections	eTPU B Signal	Signals with Which eTPU Signal is Shared:
0–7	I	M25, M24, L26, L25, L24, K26, L23, K25	0–7	not connected	eTPU_B[0:7]	eTPU_B[16:23] (output only) GPIO[147:154]
				DSPI_A[15:8][1]		
	O	AE19, AD19, AF20, AE20, AF21, AC19, AD20, AF21	0–7			eMIOS[16:23] GPIO[195:202]
8–15	I	K24, J26, K23, J25, J24, H26, H25, G26	8–15	not connected	eTPU_B[8:15]	eTPU_B[24:31] (output only) GPIO[155:162]
	O			DSPI_A[7:0][1]		

Table 18-2. eTPU B Channel Connection Table (MPC5554 Only)

eTPU Channel Number	I/O	Pin Number	eTPU Channel Connections	DSPI Serial Channel Connections	eTPU B Signal	Signals with Which eTPU Signal is Shared:
16–31	I O	D16, D17, A17, C16, A18, B17, C17, D18, A19, B18, C18, A20, B19, D19, C19, B20	16–31	not connected	eTPU_B[16:31]	GPIO[163:178]

1 The channel numbers for some of the DSPI channels connections are reversed, for example if eTPU_B[0:7] is mapped to DSPI_A[15:8], then eTPU_B[0] is connected to DSPI_A[15], eTPU_B[1] is connected to DSPI_A[14],..., eTPU_B[7] is connected to DSPI_A[8].

18.3.2.2 Time Base Clock Signal (TCRCLK[A:B])

The TCRCLK[A:B] input signals are used to control the TCR1 and TCR2 time bases for eTPU A and eTPU B.

NOTE

Throughout this document, TCRCLKA and TCRCLKB are referred to generically as TCRCLK.

There is one independent TCRCLK input for each engine. Table 18-3 shows the TCRCLK pin connections. For pulse accumulator operations TCRCLK can be used as a gate for a counter based on the system clock divided by eight. For angle operations TCRCLK can be used to get the tooth transition indications in angle mode. Further details can be found in the eTPU reference manual's Section 5.9 and 5.10.

Table 18-3. TCRCLK Signals

Signal Name	Pin Connection	Other Signals Muxed on Same Pin
TCRCLKA	N4	IRQ7 (input only) GPIO113
TCRCLKB[1]	M23	IRQ6 (input only) GPIO146

1 TCRCLKB is implemented only in the MPC5554.

18.3.2.3 Channel Output Disable Signals

Each eTPU engine has four input signals that are used to force the outputs of a group of 8 channels to an inactive level. These signals originate from the eMIOS. When an output disable signal is active, all the 8 channels assigned to the disable signal that have their ODIS bits set to 1 in ETPU_C*n*CR register have their outputs forced to the opposite of the value specified in the ETPU_C*n*CR[OPOL] bit. Therefore, individual channels can be selected to be affected by the output disable signals, as well as their disabling forced polarity.

The output disable channel groups are defined in Table 18-4.

Table 18-4. Output Disable Channel Groups

eMIOS Channel	Engine	eTPU Channels Disabled
11	A	0–7
10		8–15
9		16–23
8		24–31
20	B	0–7
21		8–15
22		16–23
23		24–31

18.4 Memory Map/Register Definition

18.4.1 Memory Map

The eTPU system simplified memory map is shown in Table 18-5. The base address for the eTPU module is listed as BASE. Each of the register areas shown may have their own reserved address areas.

Table 18-5 shows a detailed memory map.

Table 18-5. eTPU High-Level Memory Map

Address	Register Description
Base– Base + 0x0_001F	eTPU system module configuration registers
Base + 0x0_0020– Base + 0x0_002F	eTPU A time base registers
Base + 0x0_0030– Base + 0x0_003F	Reserved
Base + 0x0_0040– Base + 0x0_004F	eTPU B time base registers
Base + 0x0_0050– Base + 0x0_01FF	Reserved
Base + 0x0_0200– Base + 0x0_02FF	eTPU[A:B] global channel registers
Base + 0x0_0300– Base + 0x0_03FF	Reserved
Base + 0x0_0400– Base + 0x0_07FF	eTPU A channel registers
Base + 0x0_0800– Base + 0x0_0BFF	eTPU B channel registers

Table 18-5. eTPU High-Level Memory Map (continued)

Address	Register Description
Base + 0x0_0C00– Base + 0x0_7FFF	Reserved
Base + 0x0_8000– Base + 0x0_8BFF	SDM (3 Kbytes)
Base + 0x0_8C00– Base + 0x0_BFFF	Reserved
Base + 0x0_C000– Base + 0x0_CBFF	SDM PSE mirror[1] (3 Kbytes)
Base + 0xCC00– Base + 0xFFFF	Reserved
Base + 0x1_0000– Base + 0x1_3FFF	SCM (16 Kbytes — MPC5554) (12 Kbytes — MPC5553)
Base + 0x1_4000– Base + 0x1_FFFF	Not writable Reads the return value of ETPU_SCMOFFDATAR register.

1 Parameter Sign Extension access area. See the eTPU reference manual.

18.4.2 Register Description

Table 18-6 shows the eTPU registers and their locations, without examples or explanation of how the fields are used. For a complete description of these registers users should refer to the *Enhanced Time Processing Unit (eTPU) Reference Manual*. The features are explained in detail there.

Table 18-6. Detailed Memory Map

Address	Register Name	Register Description	Size (bits)
Base (0xC3FC_0000)	ETPU_MCR	eTPU module configuration register	32
Base + 0x0_0004	ETPU_CDCR	eTPU coherent dual-parameter controller register	32
Base + 0x0_0008	—	Reserved	—
Base + 0x0_000C	ETPU_MISCCMPR	eTPU MISC compare register	32
Base + 0x0_0010	ETPU_SCMOFFDATAR	eTPU SCM off-range data register	32
Base + 0x0_0014	ETPU_ECR_A	eTPU A engine configuration register	32
Base + 0x0_0018	ETPU_ECR_B[1]	eTPU B engine configuration register	32
Base + 0x0_001C	—	Reserved	—
Base + 0x0_0020	ETPU_TBCR_A	eTPU A time base configuration register	32
Base + 0x0_0024	ETPU_TB1R_A	eTPU A time base 1	32
Base + 0x0_0028	ETPU_TB2R_A	eTPU A time base 2	32
Base + 0x0_002C	ETPU_REDCR_A	eTPU A STAC bus interface configuration register	32
Base + 0x0_0030– Base + 0x0_003F	—	Reserved	—

Table 18-6. Detailed Memory Map (continued)

Address	Register Name	Register Description	Size (bits)
Base + 0x0_0040	ETPU_TBCR_B[1]	eTPU B time base configuration register	32
Base + 0x0_0044	ETPU_TB1R_B[1]	eTPU B time base 1	32
Base + 0x0_0048	ETPU_TB2R_B[1]	eTPU B time base 2	32
Base + 0x0_004C	ETPU_REDCR_B[1]	eTPU B STAC bus interface configuration register	32
Base + 0x0_0050– Base + 0x0_01FF	—	Reserved	—
Base + 0x0_0200	ETPU_CISR_A	eTPU A channel interrupt status register	32
Base + 0x0_0204	ETPU_CISR_B[1]	eTPU B channel interrupt status register	32
Base + 0x0_0208	—	Reserved	—
Base + 0x0_020C	—	Reserved	—
Base + 0x0_0210	ETPU_CDTRSR_A	eTPU A channel data transfer request status register	32
Base + 0x0_0214	ETPU_CDTRSR_B[1]	eTPU B channel data transfer request status register	32
Base + 0x0_0218	—	Reserved	—
Base + 0x0_021C	—	Reserved	—
Base + 0x0_0220	ETPU_CIOSR_A	eTPU A channel interrupt overflow status register	32
Base + 0x0_0224	ETPU_CIOSR_B[1]	eTPU B channel interrupt overflow status register	32
Base + 0x0_0228	—	Reserved	—
Base + 0x0_022C	—	Reserved	—
Base + 0x0_0230	ETPU_CDTROSR_A	eTPU A channel data transfer request overflow status register	32
Base + 0x0_0234	ETPU_CDTROSR_B[1]	eTPU B channel data transfer request overflow status register	32
Base + 0x0_0238	—	Reserved	—
Base + 0x0_023C	—	Reserved	—
Base + 0x0_0240	ETPU_CIER_A	eTPU A channel interrupt enable register	32
Base + 0x0_0244	ETPU_CIER_B[1]	eTPU B channel interrupt enable register	32
Base + 0x0_0248	—	Reserved	—
Base + 0x0_024C	—	Reserved	—
Base + 0x0_0250	ETPU_CDTRER_A	eTPU A channel data transfer request enable register	32
Base + 0x0_0254	ETPU_CDTRER_B[1]	eTPU B channel data transfer request enable register	32
Base + 0x0_0258– Base + 0x0_027F	—	Reserved	—
Base + 0x0_0280	ETPU_CPSSR_A	eTPU A channel pending service status register	32
Base + 0x0_0284	ETPU_CPSSR_B[1]	eTPU B channel pending service status register	32
Base + 0x0_0288	—	Reserved	—

Table 18-6. Detailed Memory Map (continued)

Address	Register Name	Register Description	Size (bits)
Base + 0x0_028C	—	Reserved	—
Base + 0x0_0290	ETPU_CSSR_A	eTPU A channel service status register	32
Base + 0x0_0294	ETPU_CSSR_B[1]	eTPU B channel service status register	32
Base + 0x0_0298– Base + 0x0_03FF	—	Reserved	—
Base + 0x0_0400	ETPU_C0CR_A	eTPU A channel 0 configuration register	32
Base + 0x0_0404	ETPU_C0SCR_A	eTPU A channel 0 status and control register	32
Base + 0x0_0408	ETPU_C0HSRR_A	eTPU A channel 0 host service request register	32
Base + 0x0_040C	—	Reserved	—
Base + 0x0_0410	ETPU_C1CR_A	eTPU A channel 1 configuration register	32
Base + 0x0_0414	ETPU_C1SCR_A	eTPU A channel 1 status and control register	32
Base + 0x0_0418	ETPU_C1HSRR_A	eTPU A channel 1 host service request register	32
Base + 0x0_041C	—	Reserved	—
. . .	. . .	. . .	. . .
Base + 0x0_05F0	ETPU_C31CR_A	eTPU A channel 31 configuration register	32
Base + 0x0_05F4	ETPU_C31SCR_A	eTPU A channel 31 status and control register	32
Base + 0x0_05F8	ETPU_C31HSRR_A	eTPU A channel 31 host service request register	32
Base + 0x0_05FC– Base + 0x0_07FF	—	Reserved	—
Base + 0x0_0800	ETPU_C0CR_B[1]	eTPU B channel 0 configuration register	32
Base + 0x0_0804	ETPU_C0SCR_B[1]	eTPU B channel 0 status and control register	32
Base + 0x0_0808	ETPU_C0HSRR_B[1]	eTPU B channel 0 host service request register	32
Base + 0x0_080C	—	Reserved	—
Base + 0x0_0810	ETPU_C1CR_B[1]	eTPU B channel 1 configuration register	32
Base + 0x0_0814	ETPU_C1SCR_B[1]	eTPU B channel 1 status and control register	32
Base + 0x0_0818	ETPU_C1HSRR_B[1]	eTPU B channel 1 host service request register	32
Base + 0x0_081C	—	Reserved	—
. . .	. . .	. . .	. . .
Base + 0x0_09F0	ETPU_C31CR_B[1]	eTPU B channel 31 configuration register	32
Base + 0x0_09F4	ETPU_C31SCR_B[1]	eTPU B channel 31 status and control register	32
Base + 0x0_09F8	ETPU_C31HSRR_B[1]	eTPU B Channel 31 host service request register	32

Table 18-6. Detailed Memory Map (continued)

Address	Register Name	Register Description	Size (bits)
Base + 0x0_09FC– Base + 0x0_7FFF	—	Reserved	—
Base + 0x0_8000– Base + 0x0_8BFF	—	3 Kbytes shared data memory (parameter RAM)	3 Kbytes
Base + 0x0_8C00– Base + 0x0_BFFF	—	Reserved	—
Base + 0x0_C000– Base + 0x0_CBFF	—	3 Kbytes SDM PSE mirror[2]	3 Kbytes
Base + 0x0_CC00– Base + 0x0_FFFF	—	Reserved	—
Base + 0x1_0000– Base + 0x1_2FFF	SCM	Shared code memory[3]	12 Kbytes (MPC5553) 16 Kbytes (MPC5554)
Base + 0x1_3000– Base + 0x1_FFFF	—	Reserved	—

1 The register at this address is available only on the MPC5554, not on the MPC5553.

2 Parameter sign extension access area. See the eTPU reference manual.

3 SCM access is only available under certain conditions when ETPU_MCR[VIS] = 1. The SCM can only be written in 32-bit accesses.

18.4.2.1 System Configuration Registers

18.4.2.1.1 eTPU Module Configuration Register (ETPU_MCR)

This register is global to both eTPU engines, and resides in the shared BIU. ETPU_MCR gathers global configuration and status in the eTPU system, including global exception. It is also used for configuring the SCM (shared code memory) operation and test.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	MGEA	MGEB	ILFA	ILFB	0	0	0	SCMSIZE				
W	GEC															
Reset	0	0	0	0	0	0	0	0	0	0	0	SCMSIZE				
Reg Addr	Base + 0x0_0000															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	SCMMISF	SCMMISEN	0	0	VIS	0	0	0	0	0	GTBE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0_0000															

Figure 18-5. ETPU_MCR Register

Table 18-7. ETPU_MCR Bit Field Descriptions

Bits	Name	Description
0	GEC	Global exception clear. Negates global exception request and clears global exception status bits MGEA, MGEB, ILFA, ILFB and SCMMISF. A read will always return 0. Writes have the following effect: 0 Keep global exception request and status bits ILFA, ILFB, MGEA, MGEB, and SCMMISF as is. 1 Negate global exception, clear status bits ILFA, ILFB, MGEA, MGEB, and SCMMISF. GEC works the same way with either one or both engines in stop mode.
1–3	—	Reserved.
4	MGEA	Microcode global exception engine A. Indicates that a global exception was asserted by microcode executed on the respective engine. The determination of the reason why the global exception was asserted is application dependent: it can be coded in an SDM status parameter, for instance. This bit is cleared by writing 1 to GEC. 0 No microcode-requested global exception pending. 1 Global exception requested by microcode is pending.
5	MGEB	Microcode global exception engine B. Indicates that a global exception was asserted by microcode executed on the respective engine. The determination of the reason why the global exception was asserted is application dependent: it can be coded in an SDM status parameter, for instance. This bit is cleared by writing 1 to GEC. 0 No microcode requested global exception pending. 1 Global exception requested by microcode is pending.
6	ILFA	Illegal instruction flag eTPU A. Set by the microengine to indicate that an illegal instruction was decoded in engine A. This bit is cleared by host writing 1 to GEC. For more information about illegal instructions, see Section 9.6 in the eTPU reference manual. 0 Illegal Instruction not detected. 1 Illegal Instruction detected by eTPU A.
7	ILFB	Illegal instruction flag eTPU B. Set by the microengine to indicate that an illegal instruction was decoded in engine B. This bit is cleared by host writing 1 to GEC. For more details, refer to the eTPU reference manual. 0 Illegal Instruction not detected. 1 Illegal Instruction detected by eTPU B.
8–10	—	Reserved.
11–15	SCMSIZE [0:4]	SCM size. Holds the number of 2 Kbyte SCM Blocks minus 1. This value is MCU-dependent.
16–20	—	Reserved.
21	SCMMISF	SCM MISC Flag. Set by the SCM MISC (multiple input signature calculator) logic to indicate that the calculated signature does not match the expected value, at the end of a MISC iteration. For more details, refer to the eTPU reference manual for more details. 0 Signature mismatch not detected. 1 MISC has read entire SCM array and the expected signature in ETPU_MISCCMPR does not match the value calculated. This bit is automatically cleared when SCMMISEN changes from 0 to 1, or when global exception is cleared by writing 1 to GEC.

Table 18-7. ETPU_MCR Bit Field Descriptions (continued)

Bits	Name	Description
22	SCMMISEN	SCM MISC enable. Used for enabling/disabling the operation of the MISC logic. SCMMISEN is readable and writable at any time. The MISC logic will only operate when this bit is set to 1. When the bit is reset the MISC address counter is set to the initial SCM address. When enabled, the MISC will continuously cycle through the SCM addresses, reading each and calculating a CRC. In order to save power, the MISC can be disabled by clearing the SCMMISEN bit. For more details, refer to the eTPU reference manual. 0 MISC operation disabled. The MISC logic is reset to its initial state. 1 MISC operation enabled. (Toggling to 1 clears the SCMMISF bit) SCMMISEN is cleared automatically when MISC logic detects an error; that is, when SCMMISF transitions from 0 to 1, disabling the MISC operation.
23 – 24	—	Reserved.
25	VIS	SCM visibility. Determines SCM visibility to the slave bus interface and resets the MISC state (but SCMMISEN keeps its value). 0 SCM is not visible to the slave bus. Accessing SCM address space issues a bus error. 1 SCM is visible to the slave bus. The MISC state is reset. This bit is write protected when any of the engines are not in halt or stop states. When VIS=1, the ETPU_ECR MDIS bits are write protected, and only 32-bit aligned SCM writes are supported. The value written to SCM is unpredictable if other transfer sizes are used.
26 – 30	—	Reserved.
31	GTBE	Global time base enable. Enables time bases in both engines, allowing them to be started synchronously. An assertion of GTBE also starts the eMIOS time base[1]. This enables the eTPU time bases and the eMIOS time base to all start synchronously. 1 time bases in both eTPU engines and eMIOS are enabled to run. 0 time bases in both engines are disabled to run. **Note:** When GTBE is turned off with Angle Mode enabled, the EAC must be reinitialized before GTBE is turned on again.

[1] The eMIOS also has an GTBE bit. Assertion of either the eMIOS or eTPU GTBE bit starts time bases for the eMIOS and eTPU, see the eTPU reference manual.

18.4.2.1.2 eTPU Coherent Dual-Parameter Controller Register (ETPU_CDCR)

ETPU_CDCR configures and controls dual-parameter coherent transfers. For more information, refer to the eTPU reference manual.

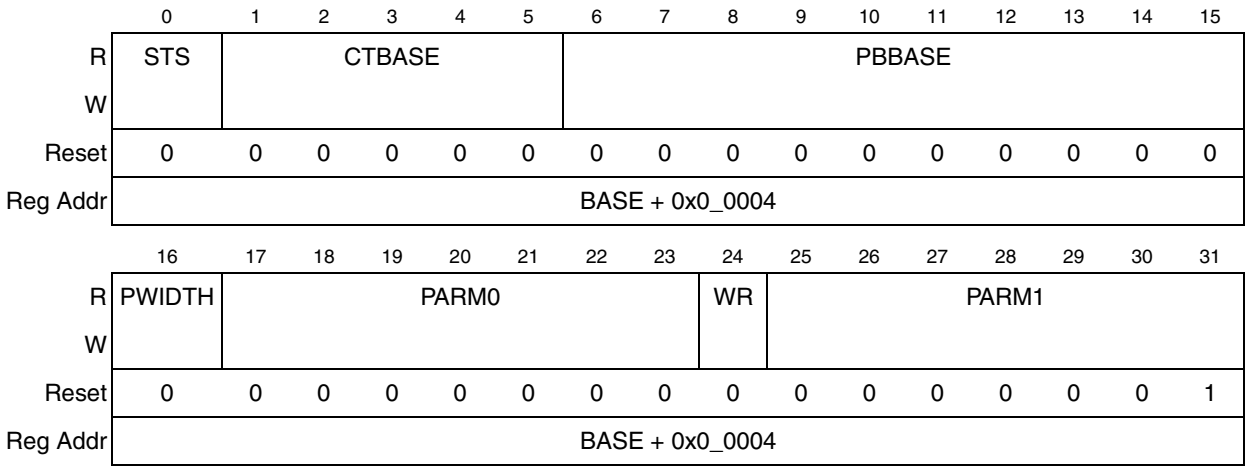


Figure 18-6. eTPU Coherent Dual-Parameter Controller Register (ETPU_CDCR)

Table 18-8. ETPU_CDCR Field Descriptions

Bits	Name	Description
0	STS	Start. Set by the host in order to start the data transfer between the parameter buffer pointed by PBBASE and the target addresses selected by the concatenation of fields CTBASE and PARM0/1. The host receives wait-states until the data transfer is complete. Coherency logic resets STS once the data transfer is complete. For more information, refer to the eTPU reference manual. 0 (Write) does not start a coherent transfer. 1 (Write) starts a coherent transfer.
1–5	CTBASE [0:4]	Channel transfer base. This field concatenates with fields PARM0/PARM1 to determine the absolute offset (from the SDM base) of the parameters to be transferred: Parameter 0 address = {CTBASE, PARM0} × 4 + SDM base Parameter 1 address = {CTBASE, PARM1} × 4 + SDM base
6–15	PBBASE [0:9]	Parameter buffer base address. Points to the base address of the parameter buffer location, with granularity of 2 parameters (8 bytes). The host (byte) address of the first parameter in the buffer is PBBASE × 8 + SDM Base Address.
16	PWIDTH	Parameter width selection. Selects the width of the parameters to be transferred between the PB and the target address. 0 Transfer 24-bit parameters. The upper byte remains unchanged in the destination address. 1 Transfer 32-bit parameters. All 32 bits of the parameters are written in the destination address.
17–23	PARM0 [0:6]	Channel parameter number 0. This field in concatenation with CTBASE[3:0] determine the address offset (from the SDM base address) of the parameter which is the destination or source (defined by WR) of the coherent transfer. The SDM address offset of the parameter is {CTBASE, PARM0}*4.Note that PARM0 allows non-contiguous parameters to be transferred coherently[1].

Table 18-8. ETPU_CDCR Field Descriptions (continued)

Bits	Name	Description
24	WR	Read/Write selection. This bit selects the direction of the coherent data transfer. 0 Read operation. Data transfer is from the selected parameter RAM address to the PB. 1 Write operation. Data transfer is from the PB to the selected parameter RAM address.
25 – 31	PARM1 [0:6]	Channel parameter number 1. This field in concatenation with CTBASE[3:0] determines the address offset (from the SDM base) of the parameter which is the destination or source (defined by WR) of the coherent transfer. The SDM address offset of the parameter is {CTBASE, PARM1}*4.Note that PARM1 allows non-contiguous parameters to be transferred coherently[1].

[1] The parameter pointed by {CTBASE, PARM0} is the first transferred.

18.4.2.1.3 eTPU MISC Compare Register (ETPU_MISCCMPR)

The multiple input signature calculator compare register (ETPU_MISCCMPR) holds the 32-bit signature expected from the whole shared code memory (SCM) array. This register must be written by the host with the 32-bit word to be compared against the calculated signature at the end of the MISC cycle. This register is global to both eTPU engines. For more details, refer to the eTPU reference manual.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EMISCCMP															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0_000C															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EMISCCMP															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0_000C															

Figure 18-7. eTPU MISC Compare Register (ETPU_MISCCMPR)

Table 18-9. ETPU_MISCCMPR Field Descriptions

Bits	Name	Description
0 – 31	EMISCCMP[0:31]	Expected multiple input signature calculator compare register value. For more information, refer to the eTPU reference manual.

18.4.2.1.4 eTPU SCM Off-Range Data Register (ETPU_SCMOFFDATAR)

ETPU_SCMOFFDATAR holds the 32-bit value returned when the SCM array is accessed at non implemented addresses, either by the host or by the microengine. This register can be written by the host with the 32-bit instruction to be executed by the microengine to recover from runaway code. This register is global to both ETPU engines.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ETPUSCMOFFDATA[0:15]															
W																
Reset	1	1	1	1	0	0	1	1	0	1	1	1	0	1	1	1
Reg Addr	Base + 0x0_0010															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ETPUSCMOFFDATA[16:31]															
W																
Reset	0	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1
Reg Addr	Base + 0x0_0010															

Figure 18-8. eTPU SCM Off-Range Data Register (ETPU_SCMOFFDATAR)

Table 18-10. ETPU_SCMOFFDATAR Field Descriptions

Bits	Name	Description
0 – 31	ETPUSCMOFFDATAR	SCM Off-range read data value.

18.4.2.1.5 eTPU Engine Configuration Register (ETPU_ECR)

Each engine has its own ETPU_ECR. The ETPU_ECR holds configuration and status fields that are programmed independently in each engine.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FEND	MDIS	0	STF	0	0	0	0	HLTF	0	0	0	0	FPSCK		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	eTPU A: Base + 0x0_0014 / eTPU B: Base + 0x0_0018															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CDFC		0	0	0	0	0	0	0	0	0	ETB				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	eTPU A: Base + 0x0_0014 / eTPU B: Base + 0x0_0018															

Figure 18-9. eTPU Engine Configuration Register (ETPU_ECR)

Table 18-11. ETPU_ECR Field Descriptions

Bits	Name	Description
0	FEND	Force end. Assertion terminates any current running thread as if an END instruction have been executed. For more information, refer to the eTPU reference manual. 0 Normal operation. 1 Terminates current thread. This bit is self-negating.
1	MDIS	Module disable internal stop. This is the low power stop bit. When MDIS is set, the engine shuts down its internal clocks. TCR1 and TCR2 cease to increment, and input sampling stops. The engine asserts the stop flag (STF) bit to indicate that it has stopped. However, the BIU continues to run, and the host can access all registers except for the channel registers[1] and writes to time base registers. More information on channel registers may be found in Section 18.4.2.4, "Channel Configuration and Control Registers." After MDIS is set, even before STF asserts, data read from the channel registers is not meaningful, a Bus Error is issued, and writes are unpredictable. When the MDIS bit is asserted while the microcode is executing, the eTPU will stop when the thread is complete. 0 eTPU engine runs. 1 Commands engine to stop its clocks. Stop completes on the next system clock after the stop condition is valid. The MDIS bit is write-protected when ETPU_MCR[VIS]=1. **Note:** Once MDIS has been switched from 1 to 0 or vice-versa, do not switch its value again until STF is written to.
2	—	Reserved.
3	STF	Stop flag bit. Each engine asserts its stop flag (STF) to indicate that it has stopped. Only then the host can assume that the engine has actually stopped. The eTPU system is fully stopped when the STF bits of both eTPU engines are asserted. The engine only stops when any ongoing thread is complete in this case. 0 The engine is operating. 1 The engine has stopped (after the local MDIS bit has been asserted, or after the STAC bus stop line has been asserted). Summarizing engine stop conditions, which STF reflects: STF_A := (after stop completed) MDIS_A STF_B := (after stop completed) MDIS_B STF_A and STF_B mean STF bit from engine A and STF bit from engine B respectively.
4–7	—	Reserved.
8	HLTF	Halt mode flag. If eTPU engine entered halt state, this flag is asserted. The flag remains asserted while the microengine is in halt state, even during a single-step or forced instruction execution. Refer to the eTPU reference manual for further details about entering halt mode. 0 eTPU engine is not halted. 1 eTPU engine is halted
9–12	—	Reserved.

Table 18-11. ETPU_ECR Field Descriptions (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>13–15</td><td>FPSCK
[0:2]</td><td>Filter prescaler clock control. Controls the prescaling of the clocks used in digital filters for the channel input signals and TCRCLK input. The following table illustrates filter prescaler clock control.
<table>
<tr><th>Filter Control</th><th>Sample on System Clock Divided by:</th></tr>
<tr><td>000</td><td>2</td></tr>
<tr><td>001</td><td>4</td></tr>
<tr><td>010</td><td>8</td></tr>
<tr><td>011</td><td>16</td></tr>
<tr><td>100</td><td>32</td></tr>
<tr><td>101</td><td>64</td></tr>
<tr><td>110</td><td>128</td></tr>
<tr><td>111</td><td>256</td></tr>
</table>
Filtering can be controlled independently by the engine, but all input digital filters in the same engine have same clock prescaling. For more details, refer to the eTPU reference manual.</td></tr>
<tr><td>16–17</td><td>CDFC
[0:1]</td><td>Channel digital filter control. Select a digital filtering mode for the channels when configured as inputs for improved noise immunity. Channel digital filter control is illustrated in the following table.
<table>
<tr><th>CDFC</th><th>Selected Digital Filter</th></tr>
<tr><td>00</td><td>TPU2/3 two sample mode: Using the filter clock which is the system clock divided by (2, 4, 8,..., 256) as a sampling clock (selected by FPSCK field in ETPU_ECR), comparing two consecutive samples which agree with each other sets the input signal state. This is the default reset state.</td></tr>
<tr><td>01</td><td>Reserved.</td></tr>
<tr><td>10</td><td>eTPU three sample mode: Similar to the TPU2/3 two sample mode, but comparing three consecutive samples which agree with each other sets the input signal state.</td></tr>
<tr><td>11</td><td>eTPU continuous mode: Signal needs to be stable for the whole filter clock period. This mode compares all the values at the rate of system clock divided by two, between two consecutive filter clock pulses. If all the values agree with each other, input signal state is updated.</td></tr>
</table>
The eTPU has three digital filtering modes for the channels which provide programmable trade-off between signal latency and noise immunity. For more information on filtering, refer to the eTPU reference manual. Changing CDFC during eTPU normal input channel operation is not recommended since it changes the behavior of the transition detection logic while executing its operation.</td></tr>
</table>

Table 18-11. ETPU_ECR Field Descriptions (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>18–26</td><td>—</td><td>Reserved.</td></tr>
<tr><td>27–31</td><td>ETB
[0:4]</td><td>Entry table base. Determines the location of the microcode entry table for the eTPU functions in SCM. More information about entry points is located in the eTPU reference manual. The following table shows the entry table base address options.
<table>
<tr><th>ETB</th><th>Entry Table Base Address for CPU Host Address (byte format)</th><th>Entry Table Base Address for Microcode Address (word format)</th></tr>
<tr><td>00000</td><td>0x0_0000</td><td>0x0_0000</td></tr>
<tr><td>00001</td><td>0x0_0800</td><td>0x0_0200</td></tr>
<tr><td>00010</td><td>0x0_1000</td><td>0x0_400</td></tr>
<tr><td>.
.
.
.</td><td>.
.
.
.</td><td>.
.
.
.</td></tr>
<tr><td>11110</td><td>0x0_F000</td><td>0x0_3C00</td></tr>
<tr><td>11111</td><td>0x0_F800</td><td>0x0_3E00</td></tr>
</table>
</td></tr>
</table>

[1] The time base registers can still be read in stop mode, but writes are ineffective and a bus error is issued. Global channel registers and SDM can be accessed normally.

18.4.2.2 Time Base Registers

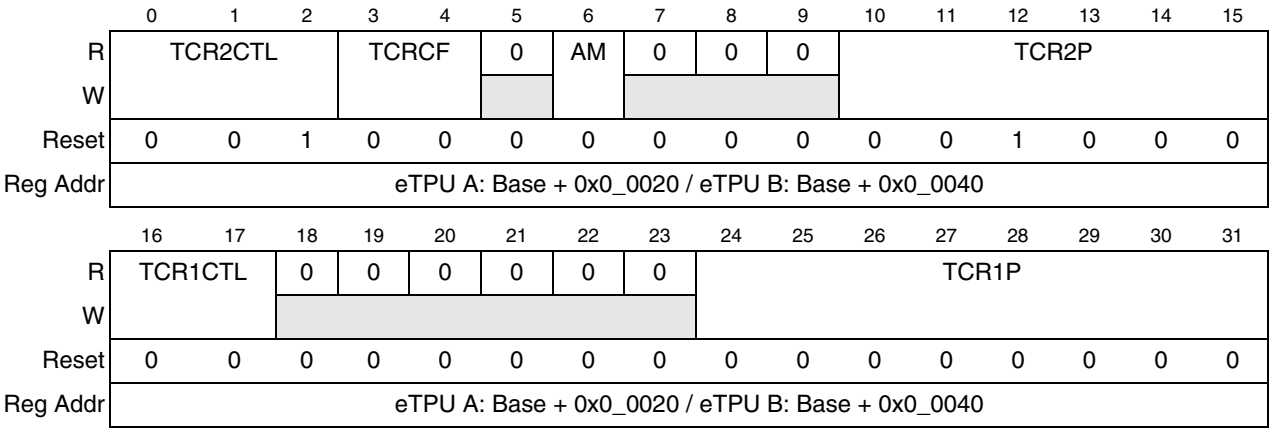
Time base registers allow the configuration and visibility of internally-generated time bases TCR1 and TCR2. There is one of each of these registers for each eTPU engine.

NOTE

Writes to this register issue a bus error and are ineffective when MDIS=1.
Reads are always allowed.

18.4.2.2.1 eTPU Time Base Configuration Register (ETPU_TBCR)

This register configures several time base options.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TCR2CTL			TCRCF		0	AM	0	0	0	TCR2P					
W																
Reset	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0
Reg Addr	eTPU A: Base + 0x0_0020 / eTPU B: Base + 0x0_0040															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TCR1CTL		0	0	0	0	0	0	TCR1P							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	eTPU A: Base + 0x0_0020 / eTPU B: Base + 0x0_0040															

Figure 18-10. eTPU Time Base Configuration Register (ETPU_TBCR)

NOTE

The MPC5554 has two eTPU engines, where the MPC5553 only has one. So for MPC5554, there are 2 TCRCLK signals, one for each eTPU engine: TCRCLKA and TCRCLKB

Table 18-12. ETPU_TBCR Field Descriptions

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>0–2</td><td>TCR2CTL</td><td>TCR2 clock/gate control. Part of the TCR2 clocking system. These bits determine the clock source for TCR2 before the prescaler. TCR2 can count on any detected edge of the TCRCLK signal or use it for gating system clock divided by 8. After reset, TCRCLK signal rising edge is selected. TCR2 can also be clocked by an internal peripheral timebase signal or the system clock divided by 8. TCR2CTL also determines the TCRCLK edge selected for angle tooth detection in angle mode. Refer to the eTPU User's Manual for more information. TCR2 clock sources are listed in the following table.
<table>
<tr><th>TCR2CTL</th><th>AM=0
(TCR2 Clock)</th><th>AM=1
(Angle Tooth Detection)</th></tr>
<tr><td>000</td><td>Gated DIV8 clock (system clock / 8). When the external TCRCLK signal is low, the DIV8 clock is blocked, preventing it from incrementing the TCR2 prescaler. When the external TCRCLK signal is high, TCR2 prescaler is incremented at the frequency of the system clock divided by 8.</td><td>See Note[1]</td></tr>
<tr><td>001</td><td>Rise transition on TCRCLK signal increments TCR2 prescaler.</td><td>Rising Edge</td></tr>
<tr><td>010</td><td>Fall transition on TCRCLK signal increments TCR2 prescaler.</td><td>Falling Edge</td></tr>
<tr><td>011</td><td>Rise or fall transition on TCRCLK signal increments TCR2 prescaler.</td><td>Rising or Falling Edge</td></tr>
<tr><td>100</td><td>DIV8 clock (system clock / 8)</td><td rowspan="4">See Note[1]</td></tr>
<tr><td>101</td><td>Peripheral Timebase clock source</td></tr>
<tr><td>110</td><td>Reserved</td></tr>
<tr><td>111</td><td>TCR2CTL shuts down TCR2 clocking, except on Angle Mode. TCR2 can also change as STAC client.</td></tr>
</table>
[1] These selections must not be used with AM=1 (Angle Mode).</td></tr>
</table>

Table 18-12. ETPU_TBCR Field Descriptions (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>3–4</td><td>TCRCF</td><td>TCRCLK signal filter control. Controls the TCRCLK digital filter determining whether the TCRCLK signal input (after a synchronizer) is filtered with the same filter clock as the channel input signals or uses the system clock divided by 2, and also whether the TCRCLK digital filter works in integrator mode or two sample mode. The following table describes TCRCLK filter clock/mode.
<table>
<tr><th>TCRCF</th><th>Filter Input</th><th>Filter Mode</th></tr>
<tr><td>00</td><td>system clock divided by 2</td><td>two sample</td></tr>
<tr><td>01</td><td>filter clock of the channels</td><td>two sample</td></tr>
<tr><td>10</td><td>system clock divided by 2</td><td>integration</td></tr>
<tr><td>11</td><td>filter clock of the channels</td><td>integration</td></tr>
</table>
For more information, refer to the eTPU User's Manual.</td></tr>
<tr><td>5</td><td>—</td><td>Reserved.</td></tr>
<tr><td>6</td><td>AM</td><td>Angle mode selection. When the AM bit is set (MPC5554: and neither TCR1 nor TCR2 are STAC interface clients), the EAC (eTPU Angle Clock) hardware provides angle information to the channels using the TCR2 bus. When the AM bit is cleared (non-angle mode), EAC operation is disabled, and its internal registers can be used as general purpose registers.
0 EAC operation is disabled.
1 TCR2 works in angle mode; (MPC5554: if TCR2 is not a STAC client, the EAC works and stores tooth counter and angle tick counter data in TCR2.)
(MPC5554: If TCR1 or TCR2 is a STAC bus client, EAC operation is forbidden. Therefore, if AM is set, the angle logic will not work properly.)
Note: AM must not be changed when ETPU_MCR[GTBE] = 1.
Note: Changing AM may cause expurious transition detections on channel 0, depending on the channel mode and state.
For more information, refer to the eTPU User's Manual.</td></tr>
<tr><td>7–9</td><td>—</td><td>Reserved.</td></tr>
<tr><td>10–15</td><td>TCR2P</td><td>Timer count register 2 prescaler control. Part of the TCR2 clocking system. TCR2 is clocked from the output of a prescaler. The prescaler divides its input by (TCR2P+1) allowing frequency divisions from 1 to 64. The prescaler input is the system clock divided by 8 (in gated or non-gated clock mode) or Internal Timebase input, or TCRCLK filtered input. This field has no effect on TCCR2 in Angle Mode. For more information on TCR2, refer to the eTPU User's Manual.</td></tr>
</table>

Table 18-12. ETPU_TBCR Field Descriptions (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>16–23</td><td>TCR1CTL</td><td>TCR1 clock/gate control. Part of the TCR1 clocking system. It determines the clock source for TCR1. TCR1 can count on detected rising edge of the TCRCLK signal or the system clock divided by 2. After reset TCRCLK signal is selected. The following table shows the selection of the TCR1 clock source.
<table>
<tr><th>TCR1CTL</th><th>TCR1 Clock</th></tr>
<tr><td>00</td><td>selects TCRCLK as clock source for the TCR1 prescaler (must not be use in Angle Mode)</td></tr>
<tr><td>01</td><td>reserved</td></tr>
<tr><td>10</td><td>selects system clock divided by 2 as clock source for the TCR1 prescaler</td></tr>
<tr><td>11</td><td>TCR1CTL shuts down TCR1 clock. TCR1 can still change if STAC client.</td></tr>
</table>
For more information on the TCR1 clocking system, refer to the eTPU User's Manual.</td></tr>
<tr><td>24–31</td><td>TCR1P</td><td>Timer count register 1 prescaler control. Clocked from the output of a prescaler. The input to the prescaler is the internal eTPU system clock divided by 2 or the output of TCRCLK filter, or Peripheral Timebase input. The prescaler divides this input by (TCR1P+1) allowing frequency divisions from 1 up to 256.</td></tr>
</table>

18.4.2.2.2 eTPU Time Base 1 (TCR1) Visibility Register (ETPU_TB1R)

This register provides visibility of the TCR1 time base for core host read access. This register is read-only. The value of the TCR1 time base shown can be driven by the TCR1 counter or imported, depending on the configuration set in ETPU_REDCR. For more information, refer to the eTPU reference manual.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	TCR1							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	eTPU A: Base + 0x0_0024 / eTPU B: Base + 0x0_0044															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TCR1															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	eTPU A: Base + 0x0_0024 / eTPU B: Base + 0x0_0044															

Figure 18-11. eTPU Time Base 1 (TCR1) Visibility Register (ETPU_TB1R)

Table 18-13. ETPU_TB1R Field Descriptions

Bits	Name	Description
0–7	—	Reserved.
8–31	TCR1 [0:23]	TCR1 value. Used on matches and captures. For more information, see the eTPU reference manual.

18.4.2.2.3 eTPU Time Base 2 (TCR2) Visibility Register (ETPU_TB2R)

This register provides visibility of the TCR2 time base for core host read access. This register is read-only. The value of the TCR2 time base shown can be driven by the TCR2 counter, the angle mode logic, or imported from the STAC interface, depending on angle mode (an engine cannot import when in angle mode) and STAC interface configurations set in registers ETPU_TBCR and ETPU_REDCR. For more information on time bases, refer to the eTPU reference manual.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	TCR2							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	eTPU A: BASE + 0x0_0028 / eTPU B: BASE + 0x0_0048															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TCR2															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	eTPU A: BASE + 0x0_0028 / eTPU B: BASE + 0x0_0048															

Figure 18-12. eTPU Time Base 2 (TCR2) Visibility Register (ETPU_TB2R)

Table 18-14. ETPU_TB2R Bit Field Descriptions

Bits	Name	Description
0–7	—	Reserved.
8–31	TCR2 [0:23]	TCR2 value. Used on matches and captures. For information on TCR2, refer to the eTPU reference manual.

18.4.2.2.4 STAC Bus Configuration Register (ETPU_REDCR)

This register configures the eTPU STAC bus interface module and operation. For more information on the STAC interface, refer to the eTPU reference manual.

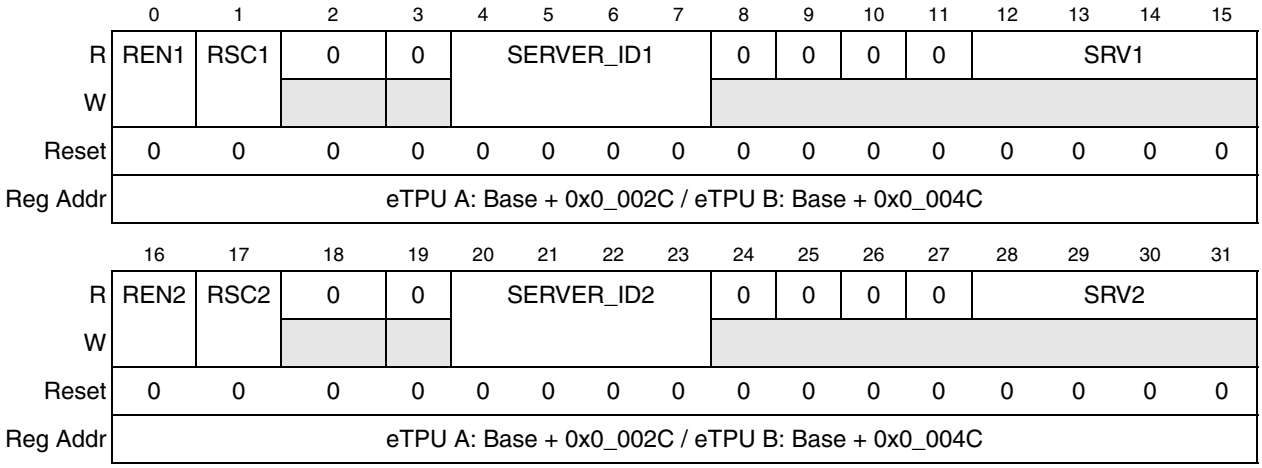


Figure 18-13. STAC Bus Configuration Register (ETPU_REDCR)

Table 18-15. ETPU_REDCR Field Descriptions

Bits	Name	Description
0	REN1	TCR1 resource[1] client/server operation enable. Enables or disables client/server operation for the eTPU STAC interface. REN1 enables TCR1. 0 Server/client operation for resource 1 is disabled. 1 Server/client operation for resource 1 is enabled.
1	RSC1	TCR1 resource server/client assignment. Selects the eTPU data resource assignment to be used as a server or client. RSC1 selects the functionality of TCR1. For server mode, external plugging determines the unique server address assigned to each TCR. For a client mode, the SRV1 field determines the server address to which the client listens. 0 Resource client operation. 1 Resource server operation.
2–3	—	Reserved.
4–7	SERVER_ID 1	STAC bus address for TCR1 as a server. For more information on the STAC interface, refer to the eTPU reference manual.
8–11	—	Reserved.
12–15	SRV1 [0:3]	TCR1 resource server. Selects the address of the specific STAC Server the local TCR1 will listen to when configured as a STAC client. For more information on the STAC interface, refer to the eTPU reference manual.
16	REN2	TCR2 resource[1] client/server operation enable. Enables or disables client/server operation for eTPU slave resources. REN2 enables TCR2 slave bus operations. 1 Server/client operation for resource 2 is enabled. 0 Server/client operation for resource 2 is disabled.
17	RSC2	TCR2[2] resource server/client assignment. Selects the eTPU data resource assignment to be used as a server or client. RSC2 selects the functionality of TCR2. For server mode, external plugging determines the unique server address assigned to each TCR. For a client mode, the SRV2 field determines the Server address to which the client listens. 0 Resource Client operation. 1 Resource Server operation.
18–19	—	Reserved.

Table 18-15. ETPU_REDCR Field Descriptions (continued)

Bits	Name	Description
20–23	SERVER_ID 2	STAC bus address for TCR2 as a server.
24–27	—	Reserved.
28–31	SRV2 [0:3]	TCR2 resource server. Selects the address of the specific STAC server the local TCR2 listens to when configured as a STAC Client. For more information on the STAC interface, refer to the eTPU reference manual.

[1] Resource identifies any parameter that changes in time and can be exported / imported from other device. For the eTPU, a resource can be TCR1 or TCR2 (either time or angle values).

[2] When TCR2 is configured as a STAC bus client (REN2 = 1, RSC2 = 0) the angle clock hardware must be disabled (ETPU_TBCR[AM] = 0).

18.4.2.3 Global Channel Registers

The registers in this section group, by type, the interrupt status and enable bits from all the channels. This organization eases management of all channels or groups of channels by a single interrupt handler routine. These bits are mirrored by the individual channel registers.

18.4.2.3.1 eTPU Channel Interrupt Status Register (ETPU_CISR)

Host interrupt status from all channels are grouped in ETPU_CISR. The bits are mirrored by the channels' status/control registers. For more information, refer to Section 18.4.2.4.3, "eTPU Channel n Status Control Register (ETPU_CnSCR)," and the eTPU reference manual.

NOTE

The host core must write 1 to clear an interrupt status bit.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CIS31	CIS30	CIS29	CIS28	CIS27	CIS26	CIS25	CIS24	CIS23	CIS22	CIS21	CIS20	CIS19	CIS18	CIS17	CIS16
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	eTPU A: Base + 0x0_0200 / eTPU B: Base + 0x0_0204															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CIS15	CIS14	CIS13	CIS12	CIS11	CIS10	CIS9	CIS8	CIS7	CIS6	CIS5	CIS4	CIS3	CIS2	CIS1	CIS0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	eTPU A: Base + 0x0_0200 / eTPU B: Base + 0x0_0204															

Figure 18-14. eTPU Channel Interrupt Status Register (ETPU_CISR)

Table 18-16. ETPU_CISR Field Descriptions

Bits	Name	Description
0–31	CIS*n*	Channel *n* interrupt status. 0 indicates that channel *n* has no pending interrupt to the host core. 1 indicates that channel *n* has a pending interrupt to the host core. To clear a status bit, the host must write 1 to it. For details about interrupts refer to the eTPU reference manual.

18.4.2.3.2 eTPU Channel Data Transfer Request Status Register (ETPU_CDTRSR)

Data transfer request status from all channels are grouped in ETPU_CDTRSR. The bits are mirrored by the channels' status/control registers. For more information on data transfers and channel control registers, see the eTPU reference manual.

In the MPC5554, eTPU A channels [0:2,12:15,28:29] and eTPU B channels [0:3,12:15,28:31] are connected to the DMA; in the MPC5553, eTPU channels [0:2, 14:15] are DMA connected. The data transfer request lines that are not connected to the DMA controller are left disconnected and do not generate transfer requests, even if their request status bits are asserted in registers ETPU_CDTRSR and ETPU_C*n*SCR. Channels that are not connected may still have their status bits (DTRS*n*) cleared by writing a 1 to the appropriate field .

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DTRS 31	DTRS 30	DTRS 29	DTRS 28	DTRS 27	DTRS 26	DTRS 25	DTRS 24	DTRS 23	DTRS 22	DTRS 21	DTRS 20	DTRS 19	DTRS 18	DTRS 17	DTRS 16
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	eTPU A: Base + 0x0_0210 / eTPU B: Base + 0x0_0214															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DTRS 15	DTRS 14	DTRS 13	DTRS 12	DTRS 11	DTRS 10	DTRS 9	DTRS 8	DTRS 7	DTRS 6	DTRS 5	DTRS 4	DTRS 3	DTRS 2	DTRS 1	DTRS 0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	eTPU A: Base + 0x0_0210 / eTPU B: Base + 0x0_0214															

Figure 18-15. eTPU Channel Data Transfer Request Status Register (ETPU_CDTRSR)

Table 18-17. ETPU_CDTRSR Field Descriptions

Bits	Name	Description
0–31	DTRS*n*	Channel *n* data transfer request status. 0 Indicates that channel *n* has no pending data transfer request. 1 Indicates that channel *n* has a pending data transfer request. To clear a status bit, the host must write 1 to it. For details about data transfer requests refer to the eTPU reference manual.

18.4.2.3.3 eTPU Channel Interrupt Overflow Status Register (ETPU_CIOSR)

An interrupt overflow occurs when an interrupt is issued for a channel when the previous interrupt status bit for the same channel has not been cleared. Interrupt overflow status from all channels are grouped in ETPU_CIOSR. The bits are mirrored by the channels' status/control registers. For information about channel status registers and overflow, refer to Section 18.4.2.4.3, "eTPU Channel n Status Control Register (ETPU_CnSCR)," and the eTPU reference manual.

NOTE

The host must write 1 to clear an interrupt overflow status bit.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CIOS 31	CIOS 30	CIOS 29	CIOS 28	CIOS 27	CIOS 26	CIOS 25	CIOS 24	CIOS 23	CIOS 22	CIOS 21	CIOS 20	CIOS 19	CIOS 18	CIOS 17	CIOS 16
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	eTPU A: Base + 0x0_0220 / eTPU B: Base + 0x0_0224															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CIOS 15	CIOS 14	CIOS 13	CIOS 12	CIOS 11	CIOS 10	CIOS 9	CIOS 8	CIOS 7	CIOS 6	CIOS 5	CIOS 4	CIOS 3	CIOS 2	CIOS 1	CIOS 0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	eTPU A: Base + 0x0_0220 / eTPU B: Base + 0x0_0224															

Figure 18-16. eTPU Channel Interrupt Overflow Status Register (ETPU_CIOSR)

Table 18-18. ETPU_CIOSR Field Descriptions

Bits	Name	Description
0–31	CIOS*n*	Channel *n* interrupt overflow status. 0 indicates that no interrupt overflow occurred in the channel. 1 indicates that an interrupt overflow occurred in the channel. To clear a status bit, the host must write 1 to it. For details about interrupts refer to the eTPU reference manual.

18.4.2.3.4 eTPU Channel Data Transfer Request Overflow Status Register (ETPU_CDTROSR)

Data transfer request overflow status from all channels are grouped in ETPU_CDTROSR. The bits are mirrored by the channels' status/control registers. For more information on channel status registers and data transfer request overflow, refer to Section 18.4.2.4.3, "eTPU Channel n Status Control Register (ETPU_CnSCR)," and the eTPU reference manual.

NOTE

The host must write 1 to clear a data transfer request overflow status bit.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DTROS 31	DTROS 30	DTROS 29	DTROS 28	DTROS 27	DTROS 26	DTROS 25	DTROS 24	DTROS 23	DTROS 22	DTROS 21	DTROS 20	DTROS 19	DTROS 18	DTROS 17	DTROS 16
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	eTPU A: BASE + 0x0_0230 / eTPU B: BASE + 0x0_0234															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DTROS 15	DTROS 14	DTROS 13	DTROS 12	DTROS 11	DTROS 10	DTROS 9	DTROS 8	DTROS 7	DTROS 6	DTROS 5	DTROS 4	DTROS 3	DTROS 2	DTROS 1	DTROS 0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	eTPU A: BASE + 0x0_0230 / eTPU B: BASE + 0x0_0234															

Figure 18-17. eTPU Channel Data Transfer Request Overflow Status Register (ETPU_CDTROSR)

Table 18-19. ETPU_CDTROSR Field Descriptions

Bits	Name	Description
0–31	DTROS*n*	Channel *n* data transfer request overflow status. 0 indicates that no data transfer request overflow occurred in the channel 1 indicates that a data transfer request overflow occurred in the channel. To clear a status bit, the host must write 1 to it. For details about data transfer request overflow, refer to the eTPU reference manual.

18.4.2.3.5 eTPU Channel Interrupt Enable Register (ETPU_CIER)

The host interrupt enable bits for all 32 channels are grouped in ETPU_CIER. The bits are mirrored by the channel configuration registers. For more information on channel configuration registers and interrupt enable, refer to Section 18.4.2.4.2, "eTPU Channel n Configuration Register (ETPU_CnCR)," and the eTPU reference manual.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CIE 31	CIE 30	CIE 29	CIE 28	CIE 27	CIE 26	CIE 25	CIE 24	CIE 23	CIE 22	CIE 21	CIE 20	CIE 19	CIE 18	CIE 17	CIE 16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	eTPU A: Base + 0x0_0240 / eTPU B: Base + 0x0_0244															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CIE 15	CIE 14	CIE 13	CIE 12	CIE 11	CIE 10	CIE 9	CIE 8	CIE 7	CIE 6	CIE 5	CIE 4	CIE 3	CIE 2	CIE 1	CIE 0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	eTPU A: Base + 0x0_0240 / eTPU B: Base + 0x0_0244															

Figure 18-18. eTPU Channel Interrupt Enable Register (ETPU_CIER)

Table 18-20. ETPU_CIER Field Descriptions

Bits	Name	Description
0–31	CIE*n*	Channel *n* interrupt enable. Enable the eTPU channels to interrupt the MPC5553/MPC5554 core. 0 Interrupt disabled for channel *n*. 1 Interrupt enabled for channel *n* For details about interrupts refer to the eTPU reference manual.

18.4.2.3.6 eTPU Channel Data Transfer Request Enable Register (ETPU_CDTRER)

Data transfer request enable status bits from all channels are grouped in ETPU_CDTRER. The bits are mirrored in the channels' configuration registers. For more on configuration registers and data transfer request enable, refer to Section 18.4.2.4.2, "eTPU Channel n Configuration Register (ETPU_CnCR)," and the eTPU reference manual.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DTRE 31	DTRE 30	DTRE 29	DTRE 28	DTRE 27	DTRE 26	DTRE 25	DTRE 24	DTRE 23	DTRE 22	DTRE 21	DTRE 20	DTRE 19	DTRE 18	DTRE 17	DTRE 16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	eTPU A: Base + 0x0_0250 / eTPU B: Base + 0x0_0254															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DTRE 15	DTRE 14	DTRE 13	DTRE 12	DTRE 11	DTRE 10	DTRE 9	DTRE 8	DTRE 7	DTRE 6	DTRE 5	DTRE 4	DTRE 3	DTRE 2	DTRE 1	DTRE 0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	eTPU A: Base + 0x0_0250 / eTPU B: Base + 0x0_0254															

Figure 18-19. eTPU Channel Data Transfer Request Enable Register (ETPU_CDTRER)

Table 18-21. ETPU_CDTRER Field Descriptions

Bits	Name	Description
0–31	DTRE*n*	Channel *n* data transfer request enable. Enable data transfer requests for their respective channels. 0 Data transfer request disabled for channel *n*. 1 Data transfer request enabled for channel *n*. For details about interrupts refer to the eTPU reference manual.

18.4.2.3.7 eTPU Channel Pending Service Status Register (ETPU_CPSSR)

ETPU_CPSSR is a read-only register that holds the status of the pending channel service requests. For information on channel service requests, refer to the eTPU reference manual.

NOTE

More than one source may be requesting service when a channel's service request bit is set.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SR31	SR30	SR29	SR28	SR27	SR26	SR25	SR24	SR23	SR22	SR21	SR20	SR19	SR18	SR17	SR16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	eTPU A: Base + 0x0_0280 / eTPU B: Base + 0x0_0284															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SR15	SR14	SR13	SR12	SR11	SR10	SR9	SR8	SR7	SR6	SR5	SR4	SR3	SR2	SR1	SR0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	eTPU A: Base + 0x0_0280 / eTPU B: Base + 0x0_0284															

Figure 18-20. eTPU Channel Pending Service Status Register (ETPU_CPSSR)

Table 18-22. ETPU_CPSSR Bit Field Descriptions

Bits	Name	Description
0–31	SR*n*	Pending service request *n*. Indicates a pending service request for channel *n*. The SR status for the pending request is negated at the time slot transition for the respective service thread. 0 no service request pending for channel *n* 1 pending service request for channel *n*

NOTE

The pending service status bit for a channel is set when a service request is pending, even if the Channel is disabled (CPR*n* = 0).

18.4.2.3.8 eTPU Channel Service Status Register (ETPU_CSSR)

ETPU_CSSR holds the current channel service status on whether it is being serviced or not. Only one bit may be asserted in this register at a given time. When no channel is being serviced the register read value is 0x0000_0000. ETPU_CSSR is a read-only register. The register can be read during normal eTPU operation for monitoring the scheduler activity. For more information on channels being serviced, refer to the eTPU reference manual.

NOTE

The ETPU_CSSR is not an absolute indication of channel status. If more than one source is requesting service, the asserted status bit only indicates that one of the requests has been granted.

NOTE

Channel service status does not always reflect decoding of the CHAN register, since the CHAN register can be changed by the service thread microcode.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SS31	SS30	SS29	SS28	SS27	SS26	SS25	SS24	SS23	SS22	SS21	SS20	SS19	SS18	SS17	SS16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	eTPU A: Base + 0x0_0290 / eTPU B: Base + 0x0_0294															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SS15	SS14	SS13	SS12	SS11	SS10	SS9	SS8	SS7	SS6	SS5	SS4	SS3	SS2	SS1	SS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	eTPU A: Base + 0x0_0290 / eTPU B: Base + 0x0_0294															

Figure 18-21. ETPU_CSSR Register

Table 18-23. ETPU_CSSR Field Descriptions

Bits	Name	Description
0–31	SS*n*	Service status *n*. Indicates that channel *n* is currently being serviced. It is updated at the 1st microcycle of a time slot transition. 0 channel *n* is not currently being serviced 1 channel *n* is currently being serviced Refer to the eTPU reference manual for more information on time slot transitions.

18.4.2.4 Channel Configuration and Control Registers

Each channel, for both eTPU engines, has a group of three registers used to control, configure and check status of that channel as shown in Table 18-24.

Table 18-24. Channel Registers Structure

Channel Offset	Register Name
0x00	eTPU channel configuration register (ETPU_C*n*CR)
0x04	eTPU channel status/control register[1] (ETPU_C*n*SCR)
0x08	eTPU channel host service request register (ETPU_C*n*HSRR)
0x0C	Reserved

[1] In the MPC5554, eTPU A channels [0:2,12:15,28:29] and eTPU B channels [0:3,12:15,28:31] are connected to the DMA; in the MPC5553, eTPU A channels [0:2, 14:15] are connected. The data transfer request lines that are not connected to the DMA controller are left disconnected and do not generate interrupt requests, even if their request status bits assert in registers ETPU_CDTRSR and ETPU_C*n*SCR

18.4.2.4.1 Channel Registers Layout

One contiguous area is used to map all channel registers of each eTPU engine as shown in Table 18-25.

Table 18-25. eTPU Channel Register Map

Address	Registers Structure
Base + 0x0_0400	eTPU A channel 0 register structure
Base + 0x0_0410	eTPU A channel 1 register structure
Base + 0x0_0420	eTPU A channel 2 register structure
Base + 0x0_0430– Base + 0x0_05D0	. . .
Base + 0x0_05E0	eTPU A channel 30 register structure
Base + 0x0_05F0	eTPU A channel 31 register structure
Base + 0x0_0600– Base + 0x0_07FF	Reserved
Base + 0x0_0800	eTPU B channel 0 register structure
Base + 0x0_0810	eTPU B channel 1 register structure
Base + 0x0_0820	eTPU B channel 2 register structure
Base + 0x0_0430– Base + 0x0_05D0	. . .
Base + 0x0_09E0	eTPU B channel 30 register structure
Base + 0x0_09F0	eTPU B channel 31 register structure
Base + 0x0_0A0 – Base + 0x0_0BFF	Reserved

There are 64 structures defined, one for each available channel in the eTPU System (32 for each engine). The base address for the structure presented can be calculated by using the following equation:

Channel_Register_Structure_Base_Address =
ETPU_Engine_Channel_Base + (channel_number * 0x0_0010)

where:

ETPU_Engine_Channel_Base = ETPU_Base + (0x0_0400 for engine A or 0x0_0800 for engine B).

18.4.2.4.2 eTPU Channel *n* Configuration Register (ETPU_C*n*CR)

The ETPU_C*n*CR is a collection of the configuration bits related to an individual channel. Some of these bits are mirrored from the global channel registers.

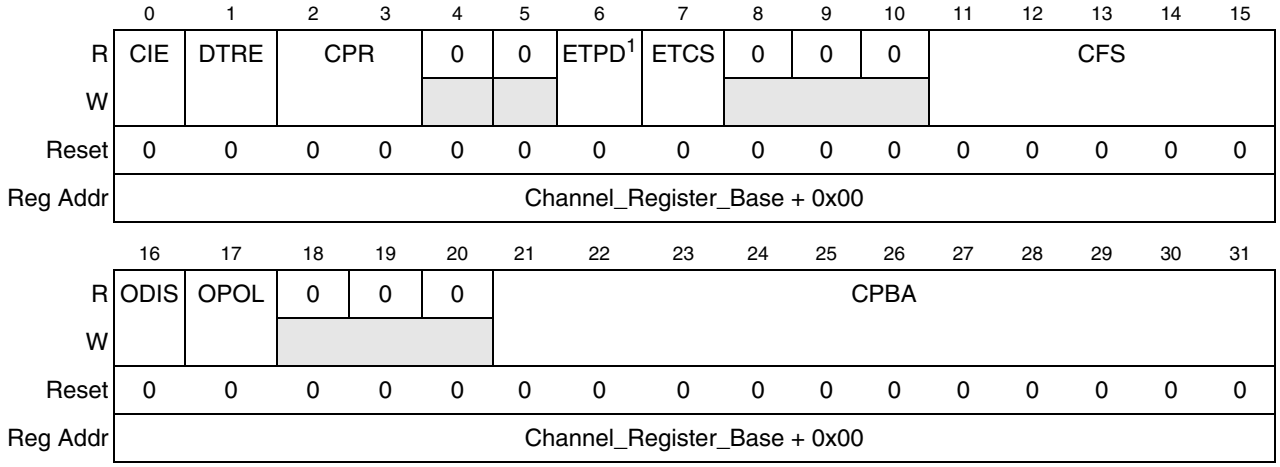


Figure 18-22. eTPU Channel *n* Configuration Register (ETPU_C*n*CR)

1 ETPD is only offered in the MPC5553.

Table 18-26. ETPU_C*n*CR Field Descriptions

Bits	Name	Description
0	CIE	Channel interrupt enable. This bit is mirrored from the ETPU_CIER 0 Disable interrupt for this channel.For more information, refer to the eTPU reference manual. 1 Enable interrupt for this channel.
1	DTRE	Channel data transfer request enable. This bit is mirrored from the ETPU_CDTRER. 0 Disable data transfer request for this channel.Refer to the eTPU reference manual for more information. 1 Enable data transfer request for this channel.

Table 18-26. ETPU_C*n*CR Field Descriptions (continued)

Bits	Name	Description
2–3	CPR [0:1]	Channel priority. Defines the priority level for the channel. The priority level is used by the hardware scheduler. The values for CPR[1:0] and corresponding levels are shown in the table below. **CPR** / **Priority** 00 Disabled 01 Low 10 Middle 11 High For more information on the hardware scheduler, refer to the eTPU reference manual.
4–5	—	Reserved.
6	ETPD	This bit selects which channel signal, input or output, is used in the entry point selection. The ETPD value has to be compatible with the function chosen for the channel, selected in the field CFS. For details about entry table and condition encoding schemes, refer to the eTPU reference manual. The ETPD bit is only present in the MPC5553. 1 = Use PSTO for entry point selection. 0 = Use PSTI for entry point selection.
7	ETCS	Entry table condition select. Determines the channel condition encoding scheme that selects the entry point to be taken in an entry table. The ETCS value has to be compatible with the function chosen for the channel, selected in ETPU_C*n*CR[CFS]. Two condition encoding schemes are available. 1 Select alternate entry table condition encoding scheme. 0 Select standard entry table condition encoding scheme.For details about entry table and condition encoding schemes, refer to the eTPU reference manual.
8–10	—	Reserved.
11–15	CFS [0:4]	Channel function select. Defines the function to be performed by the channel. The function assigned to the channel has to be compatible with the channel condition encoding scheme, selected by ETPU_C*n*CR[ETCS]. For more information about functions, refer to the eTPU reference manual.
16	ODIS	Output disable. Enables the channel to have its output forced to the value opposite to OPOL when the output disable input signal corresponding to the channel group that it belongs is active. 0 Turns off the output disable feature for the channel.For more information on output disable, refer to the eTPU reference manual. 1 Turns on the output disable feature for the channel
17	OPOL	Output polarity. Determines the output signal polarity. The activation of the output disable signal forces, when enabled by ETPU_C*n*CR[ODIS], the channel output signal to the opposite of this polarity. 0 Output active low (output disable drives output to high) 1 Output active high (output disable drives output to low)

Table 18-26. ETPU_C*n*CR Field Descriptions (continued)

Bits	Name	Description
18–20	—	Reserved.
21–31	CPBA [0:10]	Channel *n* parameter base address. The value of this field multiplied by 8 specifies the SDM parameter base host (byte) address for channel *n* (2-parameter granularity). The formula for calculating the absolute channel parameter base (byte) address, as seen by the host, is eTPU_Base + 0x8000 + CPBA*8. The SDM is mirrored in the parameter sign extension (PSE) area. The formula to calculate the absolute channel parameter base (byte) address in the PSE area is eTPU_Base + 0xC000 + CPBA*8. For more information on SDM addresses, refer to the eTPU reference manual.

18.4.2.4.3 eTPU Channel *n* Status Control Register (ETPU_C*n*SCR)

ETPU_C*n*SCR is a collection of the interrupt status bits of the channel, and also the function mode definition (read-write). Bits CIS, CIOS, DTRS, and DTROS for each channel can also be accessed from ETPU_CISR, ETPU_CIOSR, ETPU_CDTRSR, and ETPU_CDTROSR respectively. For more information on the three previously mentioned registers, refer to the eTPU reference manual.

NOTE

The MPC5553/MPC5554 core must write 1 to clear a status bit.

NOTE

In the MPC5554, eTPU A channels [0:2,12:15,28:29] and eTPU B channels [0:3,12:15,28:31] are connected to the DMA; in the MPC5553, eTPU A channels [0:2, 14:15] are DMA connected. The data transfer request lines that are not connected to the DMA controller are left disconnected and do not generate transfer requests, even if their request status bits assert in registers ETPU_CDTRSR and ETPU_C*n*SCR

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CIS	CIOS	0	0	0	0	0	0	DTRS	DTROS	0	0	0	0	0	0
W	w1c	w1c							w1c	w1c						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Channel_Register_Base + 0x04															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IPS	OPS	0	0	0	0	0	0	0	0	0	0	0	0	FM	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Channel_Register_Base + 0x04															

Figure 18-23. eTPU Channel *n* Status Control Register (ETPU_C*n*SCR)

Table 18-27. ETPU_C*n*SCR Field Descriptions

Bits	Name	Description
0	CIS	Channel interrupt status. 0 Channel has no pending interrupt to the MPC5553/MPC5554 core. 1 Channel has a pending interrupt to the MPC5553/MPC5554 core. CIS is mirrored in the ETPU_CISR. For more information on ETPU_CISR and interrupts, see Section 18.4.2.3.1, “eTPU Channel Interrupt Status Register (ETPU_CISR),” and the eTPU reference manual. The core must write 1 to clear CIS.
0	CIC	Channel interrupt clear. 0 Keep interrupt status bit unaltered. 1 Clear interrupt status bit. CIS and CIC are mirrored in the ETPU_CISR. For more information on ETPU_CISR and interrupts, see Section 18.4.2.3.1, “eTPU Channel Interrupt Status Register (ETPU_CISR),” and eTPU reference manual.
1	CIOS	Channel interrupt overflow status. 0 Interrupt overflow negated for this channel 1 Interrupt overflow asserted for this channel CIOS is mirrored in the ETPU_CIOSR. For more information on the ETPU_CIOSR and interrupt overflow, see Section 18.4.2.3.3, “eTPU Channel Interrupt Overflow Status Register (ETPU_CIOSR).” and the eTPU reference manual. The core must write 1 to clear CIOS.
2–7	—	Reserved.
8	DTRS	Data transfer request status. 0 Channel has no pending data transfer request. 1 Channel has a pending data transfer request. DTRS is mirrored in the ETPU_CISR. For more information on the ETPU_CISR and data transfer, see Section 18.4.2.3.2, “eTPU Channel Data Transfer Request Status Register (ETPU_CDTRSR).” and the eTPU reference manual. The core must write 1 to clear DTRS.
9	DTROS	Data transfer request overflow status. 0 Data transfer request overflow negated for this channel. 1 Data transfer request overflow asserted for this channel. DTROS is mirrored in the ETPU_CDTROSR. See Section 18.4.2.3.4, “eTPU Channel Data Transfer Request Overflow Status Register (ETPU_CDTROSR).” and the eTPU reference manual for more information on ETPU_CDTROSR and data transfer overflows. The core must write 1 to clear DTROS.
10–15	—	Reserved.
16	IPS	Channel input pin state. Shows the current value of the filtered channel input signal state
17	OPS	Channel output pin state. Shows the current value driven in the channel output signal, including the effect of the external output disable feature. If the channel input and output signals are connected to the same pad, OPS reflects the value driven to the pad . This is not necessarily the actual pad value, which drives the value in the IPS bit.

Table 18-27. ETPU_C*n*SCR Field Descriptions (continued)

Bits	Name	Description
18–29	—	Reserved.
30–31	FM [0:1]	Channel function mode.[1] Each function may use this field for specific configuration. These bits can be tested by microengine code.

[1] These bits are equivalent to the TPU/TPU2/TPU3 host sequence (HSQ) bits.

18.4.2.4.4 eTPU Channel *n* Host Service Request Register (ETPU_C*n*HSRR)

ETPU_C*n*HSRR is used by the MPC5553/MPC5554 core to issue service requests to the channel.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Channel_Register_Base + 0x08															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	HSR		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Channel_Register_Base + 0x08															

Figure 18-24. eTPU Channel *n* Host Service Request Register (ETPU_CnHSRR)

Table 18-28. ETPU_C*n*HSRR Field Descriptions

Bits	Name	Description
0–28	—	Reserved.
29–31	HSR [0:2]	Host service request. Used by the host core to request service to the channel • HSR = 000: no host service request pending • HSR > 000: function-dependent host service request pending. The HSR value turns to 000 automatically at the end of microengine service for that channel. The host should write HSR > 0 only when HSR = 0. Writing HSR = 000 withdraws a pending request if the scheduler has not started to resolve the entry point, however once the scheduler starts resolving, the service thread will not be aborted.

18.5 Functional Description

Refer to the eTPU User's Manual for information regarding the functional description of the eTPU module.

18.6 Initialization/Application Information

After initial power-on reset, the eTPU remains in an idle state (except when debug is asserted on power-on reset—in this case, the microengines awakens in halt state). In addition, the SCM should be initialized with the eTPU application prior to configuring the eTPU.

18.7 Revision History

Substantive Changes since Rev 3.0
Added Table 18-12, the bit field descriptions of ETPU_TBCR along with the Note above it.

Chapter 19
Enhanced Queued Analog-to-Digital Converter (eQADC)

19.1 Introduction

The enhanced queued analog-to-digital converter (eQADC) of the MPC5553/MPC5554 provides accurate and fast conversions for a wide range of applications. The eQADC provides a parallel interface to two on-chip analog-to-digital converters (ADCs), and a single master to single slave serial interface to an off-chip external device. The two on-chip ADCs are architected to allow access to all the analog channels.

19.1.1 Block Diagram

Figure 19-1 shows the primary components inside the eQADC.

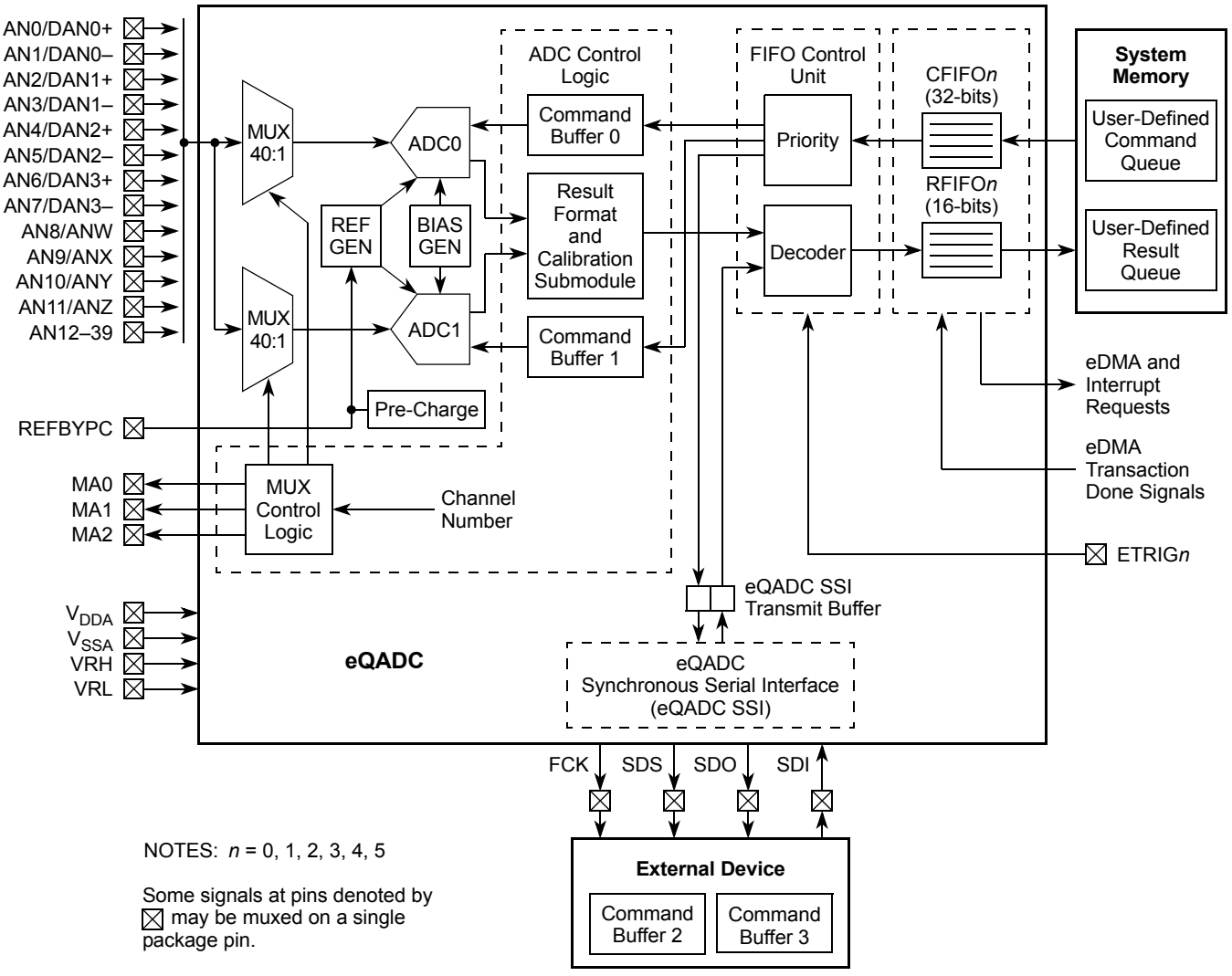


Figure 19-1. Simplified eQADC Block Diagram

19.1.2 Overview

The eQADC transfers commands from multiple command FIFOs (CFIFOs) to the on-chip ADCs or to the external device. The module can also in parallel (independently of the CFIFOs) receive data from the on-chip ADCs or from an off-chip external device into multiple result FIFOs (RFIFOs). The eQADC supports software and external hardware triggers from other modules to initiate transfers of commands from the CFIFOs to the on-chip ADCs or to the external device. (Refer to Section 6.4.5.1, "eQADC External Trigger Input Multiplexing.") It also monitors the fullness of CFIFOs and RFIFOs, which may result in either underflow or overflow conditions. A CFIFO underflow occurs when the CFIFO is in the TRIGGERED state and it becomes empty. An RFIFO overflow occurs when an RFIFO is full and more data is ready to be moved to the RFIFO by the host CPU or by eDMA. Accordingly, the eQADC generates eDMA or interrupt requests to control data movement between the FIFOs and the system memory, which is external to the eQADC.

The eQADC consists of the FIFO control unit which controls the CFIFOs and the RFIFOs, two ADCs with associated control logic, and the eQADC synchronous serial interface (eQADC SSI) which allows communication with an external device. There are 6 CFIFOs and 6 RFIFOs, each with 4 entries.

The FIFO control unit performs the following functions:

- Prioritizes the CFIFOs to determine what CFIFOs will have their commands transferred
- Supports software and hardware triggers to start command transfers from a particular CFIFO
- Decodes command data from the CFIFOs, and accordingly, sends these commands to one of the two on-chip ADCs or to the external device
- Decodes result data from on-chip ADCs or from the external device, and transfers data to the appropriate RFIFO

The ADC control logic manages the execution of commands bound for on-chip ADCs from the CFIFOs and with the RFIFOs via the result format and calibration submodule. The ADC control logic performs the following functions:

- Buffers command data for execution
- Decodes command data and accordingly generates control signals for the two on-chip ADCs
- Formats and calibrates conversion result data coming from the on-chip ADCs
- Generates the internal multiplexer control signals and the select signals used by the external multiplexers

The eQADC SSI allows for a full duplex, synchronous, serial communication between the eQADC and an external device.

Figure 19-1 also depicts data flow through the eQADC. Commands are contained in system memory in a user-defined queue data structure. Command data is moved from the user-defined command queue to the CFIFOs by either the host CPU or by the eDMA. Once a CFIFO is triggered and becomes the highest priority, CFIFO command data is transferred from the CFIFO to the on chip ADCs, or to the external device. The ADC executes the command, and the result is moved through the result format and calibration submodule and to the RFIFO. The RFIFO target is specified by a field in the command that initiated the conversion. Data from the external device bypasses the result format and calibration submodule and is moved directly to its specified RFIFO. When data is stored in an RFIFO, data is moved from the RFIFO by the host CPU or by the eDMA to a data structure in system memory depicted in Figure 19-1 as a user-defined result queue.

For users familiar with the QADC, the eQADC system upgrades the functionality provided by that module. Refer to Section 19.5.7, "eQADC versus QADC," for a comparison between the eQADC and QADC.

19.1.3 Features

The eQADC includes these distinctive features:

- Two independent on-chip RSD cyclic ADCs
  - 12 Bit AD Resolution.
  - Targets up to 10 bit accuracy at 400 kilosamples per second (ADC_CLK=6MHz) and 8 bit accuracy at 800 kilosamples per second (ADC_CLK=12MHz) for differential conversions.
  - Differential conversions (range -2.5V to +2.5V).
  - Single-ended signal range from 0 to 5V.
  - Sample times of 2 (default), 8, 64, or 128 ADC clock cycles.
  - Provides sample time stamp information when requested.
  - Parallel interface to eQADC CFIFOs and RFIFOs.
  - Supports both right-justified unsigned and signed formats for conversion results.
  - The REFBYPC pin allows process-independent low variation biasing current.
- Optional automatic application of ADC calibration constants
  - Provision of reference voltages (25%VREF[1] and 75%VREF) for ADC calibration purposes
- 40 input channels available to the two on-chip ADCs
- Four pairs of differential analog input channels
- Full duplex synchronous serial interface to an external device
  - A free-running clock is provided for use by the external device.
  - Supports a 26-bit message length.
  - Transmits a null message when there are no triggered CFIFOs with commands bound for external command buffers, or when there are triggered CFIFOs with commands bound for external command buffers but the external command buffers are full.
- Priority-based CFIFOs
  - Supports six CFIFOs with fixed priority. The lower the CFIFO number, the higher its priority. When commands of distinct CFIFOs are bound for the same ADC, the higher priority CFIFO is always served first.
  - Supports software and several hardware trigger modes to arm a particular CFIFO.
  - Generates interrupt when command coherency is not achieved.
- External hardware triggers
  - Supports rising edge, falling edge, high level and low level triggers
  - Supports configurable digital filter
- Supports four external 8-to-1 muxes that can expand the input channel number from 40 to 68
- Upgrades the functionality provided by the QADC

19.1.4 Modes of Operation

This section describes the operation modes of the eQADC.

19.1.4.1 Normal Mode

This is the default operational mode when the eQADC is not in background debug or stop mode.

1. VREF=VRH-VRL.

19.1.4.2 Debug Mode

Upon a debug mode entry request, eQADC behavior will vary according to the status of the DBG field in Section 19.3.2.1, "eQADC Module Configuration Register (EQADC_MCR)." If DBG is programmed to 0b00, the debug mode entry request is ignored. If DBG is programmed to 0b10 or to 0b11, the eQADC will enter debug mode. In case the eQADC SSI is enabled, the free running clock (FCK) output to external device will not stop when DBG is programmed to 0b11, but FCK will stop in low phase, when DBG is programmed to 0b10.

During debug mode, the eQADC will not transfer commands from any CFIFOs, no null messages will be transmitted to the external device, no data will be returned to any RFIFO, no hardware trigger event will be captured, and all eQADC registers can be accessed as in normal mode. Access to eQADC registers implies that CFIFOs can still be triggered using software triggers, because no scheme is implemented to write-protect registers during debug mode. eDMA and interrupt requests continue to be generated as in normal mode.

If at the time the debug mode entry request is detected, there are commands in the ADC that were already under execution, these commands will be completed but the generated results, if any, will not be sent to the RFIFOs until debug mode is exited. Commands whose execution has not started will not be executed until debug mode is exited. The clock associated with an on-chip ADC stops, during its low phase, after the ADC ceases executing commands. The time base counter will only stop after all on-chip ADCs cease executing commands.

When exiting debug mode, the eQADC relies on the FIFO control unit and on the CFIFO status to determine the next command entry to transfer.

The eQADC internal behavior after the debug mode entry request is detected differs depending on the status of command transfers.

- No command transfer is in progress.
  The eQADC immediately halts future command transfers from any CFIFO.

  If a null message is being transmitted, eQADC will complete the serial transmission before halting future command transfers. If valid data (conversion result or data read from an ADC register) is received by the result format and calibration submodule at the end of transmission, this data will not be sent to an RFIFO until debug mode is exited.

  If the null message transmission is aborted, the eQADC will complete the abort procedure before halting future command transfers from any CFIFO. The message of the CFIFO that caused the abort of the previous serial transmission will only be transmitted after debug mode is exited.
- Command transfer is in progress.
  eQADC will complete the transfer and update CFIFO status before halting future command transfers from any CFIFO.

  Command transfers to the external device are considered completed when the serial transmission of the command is completed. If valid data (conversion result or data read from an ADC register) is received at the end of a serial transmission, it will not be sent to an RFIFO until debug mode is exited. The CFIFO status bits will still be updated after the completion of the serial transmission, therefore, after debug mode entry request is detected, the eQADC status bits will only stop changing several system clock cycles after the on-going serial transmission completes.

  If the command message transmission is aborted, the eQADC will complete the abort procedure before halting future command transfers from any CFIFO. The message of the CFIFO that caused the abort of the previous serial transmission will only be transmitted after debug mode is exited.
- Command/null message transfer through serial interface was aborted but next serial transmission did not start.

If the debug mode entry request is detected between the time a previous serial transmission was aborted and the start of the next transmission, the eQADC will complete the abort procedure before halting future command transfers from any CFIFO. The message of the CFIFO that caused the abort of the previous serial transmission will only be transmitted after debug mode is exited.

19.1.4.3 Stop Mode

Upon a stop mode entry request detection, the eQADC progressively halts its operations until it reaches a static, stable state from which it can recover when returning to normal mode. The eQADC then asserts an acknowledge signal, indicating that it is static and that the clock input can be stopped. In stop mode, the free running clock (FCK) output to external device will stop during its low phase if the eQADC SSI is enabled, and no hardware trigger events will be captured. No capturing of hardware trigger events means that — as long as the system clock is running — CFIFOs can still be triggered using software triggers because no scheme is implemented to write-protect registers during stop mode.

If at the time the stop mode entry request is detected, there are commands in the ADC that were already under execution, these commands will be completed but the generated results, if any, will not be sent to the RFIFOs until stop mode is exited. Commands whose execution has not started will not be executed until stop mode is exited.

After these remaining commands are executed, the clock input to the ADCs is stopped. The time base counter will stop after all on-chip ADCs cease executing commands and then the stop acknowledge signal is asserted. When exiting stop mode, the eQADC relies on the CFIFO operation modes and on the CFIFO status to determine the next command entry to transfer.

The eQADC internal behavior after the stop mode entry request is detected differs depending on the status of the command transfer.

- No command transfer is in progres.s
  The eQADC immediately halts future command transfers from any CFIFO.

  If a null message is being transmitted, eQADC will complete the transmission before halting future command transfers. If valid data (conversion result or data read from an ADC register) is received at the end of the transmission, it will not be sent to an RFIFO until stop mode is exited.

  If the null message transmission is aborted, the eQADC will complete the abort procedure before halting future command transfers from any CFIFO. The message of the CFIFO that caused the abort of the previous serial transmission will only be transmitted after stop mode is exited.
- Command transfer is in progress.
  The eQADC will complete the transfer and update CFIFO status before halting future command transfers from any CFIFO.

  Command transfers to the external device are considered completed when the serial transmission of the command is completed. If valid data (conversion result or data read from an ADC register) is received at the end of a serial transmission, it will not be sent to an RFIFO until stop mode is exited. The CFIFO status bits will still be updated after the completion of the serial transmission, therefore, after stop mode entry request is detected, the eQADC status bits will only stop changing several system clock cycles after the on-going serial transmission completes.

  If the command message transmission is aborted, the eQADC will complete the abort procedure before halting future command transfers from any CFIFO. The message of the CFIFO that caused the abort of the previous serial transmission will only be transmitted after stop mode is exited.
- Command/null message transfer through serial interface was aborted but next serial transmission did not start.

If the stop mode entry request is detected between the time a previous serial transmission was aborted and the start of the next transmission, the eQADC will complete the abort procedure before halting future command transfers from any CFIFO. The message of the CFIFO that caused the abort of the previous serial transmission will only be transferred after stop mode is exited.

19.2 External Signals

The following is a list of external signals. These signals are external to the eQADC module, but may or may not be physical pins. See Chapter 2, "Signal Description" for a complete list of all physical pins and signals.

Table 19-1. eQADC External Signals

Function	Description	I/O Type	Status During Reset[1]	Status After Reset[2]	Type	Package
AN0 DAN0+	Single Ended Analog Input 0 Positive Terminal Differential Input	I I	I / —	AN0/ —	Analog	416 324 208
AN1 DAN0-	Single Ended Analog Input 1 Negative Terminal Differential Input	I I	I / —	AN1/ —	Analog	416 324 208
AN2 DAN1+	Single Ended Analog Input 2 Positive Terminal Differential Input	I I	I / —	AN2 / —	Analog	416 324 208
AN3 DAN1-	Single Ended Analog Input 3 Negative Terminal Differential Input	I I	I / —	AN3 / —	Analog	416 324 208
AN4 DAN2+	Single Ended Analog Input 4 Positive Terminal Differential Input	I I	I / —	AN4/ —	Analog	416 324 208
AN5 DAN2-	Single Ended Analog Input 5 Negative Terminal Differential Input	I I	I / —	AN5 / —	Analog	416 324 208
AN6 DAN3+	Single Ended Analog Input 6 Positive Terminal Differential Input	I I	I / —	AN6 / —	Analog	416 324 208
AN7 DAN3-	Single Ended Analog Input 7 Negative Terminal Differential Input	I I	I / —	AN7 / —	Analog	416 324 208
AN8 ANW	Single Ended Analog Input 8 External Multiplexed Analog Input W	I I	I / —	AN8/ —	Analog	416 324 208
AN9 ANX	Single Ended Analog Input 9 External Multiplexed Analog Input X	I I	I / —	AN9 / —	Analog	416 324 208

Table 19-1. eQADC External Signals (continued)

Function	Description	I/O Type	Status During Reset[1]	Status After Reset[2]	Type	Package
AN10 ANY	Single Ended Analog Input 10 External Multiplexed Analog Input Y	I I	I / —	AN10/ —	Analog	416 324 208
AN11 ANZ	Single Ended Analog Input 11 External Multiplexed Analog Input Z	I I	I / —	AN11 / —	Analog	416 324 208
AN12 MA0 $\overline{\text{SDS}}$	Single Ended Analog Input 12 Mux Address 0 eQADC SSI Serial Data Select	I O O	I / —	AN12/ —	Analog/ Digital/ Digital	416 324 208
AN13 MA1 SDO	Single Ended Analog Input 13 Mux Address 1 eQADC SSI Serial Data Out	I O O	I / —	AN13/ —	Analog/ Digital/ Digital	416 324 208
AN14 MA2 SDI	Single Ended Analog Input 14 Mux Address 2 eQADC SSI Serial Data In	I O I	I / —	AN14/ —	Analog/ Digital/ Digital	416 324 208
AN15 FCK	Single Ended Analog Input 15 eQADC Free Running Clock	I O	I / —	AN15/ —	Analog/ Digital	416 324 208
AN16	Single Ended Analog Input 16	I	I / —	AN16 / —	Analog	416 324
AN[17:19]	Single Ended Analog Input 17-19	I	I / —	AN[17:19]/ —	Analog	416 324 208
AN20	Single Ended Analog Input	I	I / —	AN20/ —	Analog	416 324
AN21	Single Ended Analog Input	I	I / —	AN21/ —	Analog	416 324 208
AN[22:25]	Single Ended Analog Input	I	I / —	AN[22:25]/ —	Analog	416 324 208
AN26	Single Ended Analog Input	I	I / —	AN26/ —	Analog	416 324
AN[27:28]	Single Ended Analog Input	I	I / —	AN[27:28]/ —	Analog	416 324 208
AN29	Single Ended Analog Input	I	I / —	AN29/ —	Analog	416 324

Table 19-1. eQADC External Signals (continued)

Function	Description	I/O Type	Status During Reset[1]	Status After Reset[2]	Type	Package
AN[30:32]	Single Ended Analog Input	I	I / —	AN[30:32]/ —	Analog	416 324 208
AN33	Single Ended Analog Input	I	I / —	AN33 / —	Analog	416 324
AN[34:35]	Single Ended Analog Input	I	I / —	AN[34:35]/ —	Analog	416 324 208
AN36	Single Ended Analog Input	I	I / —	AN36 / —	Analog	416 324
AN[37:39]	Single Ended Analog Input 37-39	I	I / —	AN[37:39] / —	Analog	416 324 208
ETRIG0/ GPIO111	External trigger for CFIFO0, CFIFO2, and CFIFO4/ GPIO	I I/O	— / Up	— / Up	Digital	416
ETRIG1/ GPIO112	External trigger for CFIFO1, CFIFO3, and CFIFO5/ GPIO	I I/O	— / Up	— / Up	Digital	416
Power Supplies						
VRH	Voltage Reference High	I	— / —	VRH	Power	416 324 208
VRL	Voltage Reference Low	I	— / —	VRL	Power	416 324 208
REFBYPC	Reference Bypass Capacitor Input	I	— / —	REFBYPC	Power	416 324 208
V_{DDA}	Analog Positive Power Supply	I	—	—	Power	416 324 208
V_{SSA}	Analog Negative Power Supply	I	—	—	Power	416 324 208

1 Terminology is O — output, I — input, Up — weak pull up enabled, Down — weak pull down enabled, Low — output driven low, High — output driven high. A dash on the left side of the slash denotes that both the input and output buffers for the pin are off. A dash on the right side of the slash denotes that there is no weak pull up/down enabled on the pin. The signal name to the left or right of the slash indicates the pin is enabled.

2 Function after reset of GPI is general-purpose input. A dash on the left side of the slash denotes that both the input and output buffers for the pin are off. A dash on the right side of the slash denotes that there is no weak pull up/down enabled on the pin.

19.2.1 Detailed Signal Descriptions

19.2.1.1 Single-ended Analog Input/Differential Analog Input Positive Terminal (AN0/DAN0+)

AN0 is a single-ended analog inputs to the two on-chip ADCs. DAN0+ is the positive terminal of the differential analog input DAN0 (DAN0+–DAN0–).

19.2.1.2 Single-ended Analog Input/Differential Analog Input Negative Terminal (AN1/DAN0–)

AN1 is a single-ended analog inputs to the two on-chip ADCs. DAN0– is the negative terminal of the differential analog input DAN0 (DAN0+–DAN0–).

19.2.1.3 Single-ended Analog Input/Differential Analog Input Positive Terminal (AN2/DAN1+)

AN2 is a single-ended analog inputs to the two on-chip ADCs. DAN1+ is the positive terminal of the differential analog input DAN1 (DAN1+–DAN1–).

19.2.1.4 Single-ended Analog Input/Differential Analog Input Negative Terminal (AN3/DAN1–)

AN3 is a single-ended analog inputs to the two on-chip ADCs. DAN1– is the negative terminal of the differential analog input DAN1 (DAN1+–DAN1–).

19.2.1.5 Single-ended Analog Input/Differential Analog Input Positive Terminal (AN4/DAN2+)

AN4 is a single-ended analog inputs to the two on-chip ADCs. DAN2+ is the positive terminal of the differential analog input DAN2 (DAN2+–DAN2–).

19.2.1.6 Single-ended Analog Input/Differential Analog Input Negative Terminal (AN5/DAN2–)

AN5 is a single-ended analog inputs to the two on-chip ADCs. DAN2– is the negative terminal of the differential analog input DAN2 (DAN2+–DAN2–).

19.2.1.7 Single-ended Analog Input/Differential Analog Input Positive Terminal (AN6/DAN3+)

AN6 is a single-ended analog inputs to the two on-chip ADCs. DAN3+ is the positive terminal of the differential analog input DAN3 (DAN3+–DAN3–).

19.2.1.8 Single-ended Analog Input/Differential Analog Input Negative Terminal (AN7/DAN3–)

AN7 is a single-ended analog inputs to the two on-chip ADCs. DAN3– is the negative terminal of the differential analog input DAN3 (DAN3+–DAN3–).

19.2.1.9 Single-ended Analog Input/ Single-ended Analog Input from External Multiplexers (AN8/ANW)

AN8 is a single-ended analog inputs to the two on-chip ADCs. ANW is a single-ended analog input to one of the on-chip ADCs in external multiplexed mode.

19.2.1.10 Single-ended Analog Input/ Single-ended Analog Input from External Multiplexers (AN9/ANX)

AN9 is a single-ended analog inputs to the two on-chip ADCs. ANX is a single-ended analog input to one of the on-chip ADCs in external multiplexed mode.

19.2.1.11 Single-ended Analog Input/ Single-ended Analog Input from External Multiplexers (AN10/ANY)

AN10 is a single-ended analog inputs to the two on-chip ADCs. ANY is a single-ended analog input to one of the on-chip ADCs in external multiplexed mode.

19.2.1.12 Single-ended Analog Input/ Single-ended Analog Input from External Multiplexers (AN11/ANZ)

AN11 is a single-ended analog inputs to the two on-chip ADCs. ANZ is a single-ended analog input to one of the on-chip ADCs in external multiplexed mode.

19.2.1.13 Single-ended Analog Input (AN[12:14]/MA[0:2]/SD*x*)

AN12 through AN14 are single-ended analog inputs to the two on-chip ADCs. MA0, MA1, and MA2 combined form a select signal associated with external multiplexers. Serial data strobe, input, and output for the eQADC serial synchronous interface are also multiplexed here.

NOTE

Performance of the analog channels AN12, AN13, AN14, and AN15 may have slightly reduced analog to digital conversion accuracy when compared to AN[0:11] and AN[16:3] because they are powered by V_{DDEH9} and can be used as digital pins for the synchronous serial interface (SSI) to external ADCs or used as the multiplexor digital outputs (MA[0:2]).

Attempts to convert the input voltage applied to AN12, AN13, AN14, and AN15 while a non-eQADC function is selected will result in an undefined conversion result.

19.2.1.13.1 eQADC SSI Serial Data Select ($\overline{\text{SDS}}$)

$\overline{\text{SDS}}$ is the serial data select output that is muxed with AN12 and MA0. It indicates to the external (slave) device when it can latch incoming serial data, when it can output its own serial data, and when it must abort a data transmission. $\overline{\text{SDS}}$ corresponds to the chip select signal in a conventional SPI interface.

These pins are configured by setting the pad configuration register, SIU_PCR215 (See Chapter 6, “System Integration Unit (SIU)”).

19.2.1.13.2 eQADC SSI Serial Data Out (SDO)

SDO is the serial data output signal to the external (slave) device. It is muxed with AN13 and MA1. These pins are configured by setting the pad configuration register, SIU_PCR216 (See Chapter 6, "System Integration Unit (SIU)").

19.2.1.13.3 eQADC SSI Serial Data In (SDI)

SDI is the serial data input signal from the external (slave) device. It is muxed with AN14 and MA2. These pins are configured by setting the pad configuration register, SIU_PCR217 (See Chapter 6, "System Integration Unit (SIU)").

19.2.1.14 Single-ended Analog Input (AN15) / eQADC SSI Free-running Clock (FCK)

FCK is a free-running clock signal for synchronizing transmissions between the eQADC (master) and the external (slave) device. AN15 is a single-ended analog input to the two on-chip ADCs. These pins are configured by setting the pad configuration register, SIU_PCR218 (See Chapter 6, "System Integration Unit (SIU)").

19.2.1.15 Single-ended Analog Input (AN[16:39])

AN16 through AN39 are single-ended analog inputs to the two on-chip ADCs.

19.2.1.16 External Triggers (ETRIG[0:1])

The external trigger signals are for hardware triggering. The eQADC can detect rising edge, falling edge, high level, and low level on each of the external trigger signals. The eQADC also supports configurable digital filters for these external trigger signals.

The eQADC external triggers input pins can be connected to the eTPU, the eMIOS, or an external signal. The source is selected by configuring the eQADC trigger source in the SIU_ETISR register. See Section 6.3.1.15, "eQADC Trigger Input Select Register (SIU_ETISR)."

ETRIG0 is the external trigger for CFIFO0, CFIFO2, and CFIFO4, and ETRIG1 serves as the external trigger for CFIFO1, CFIFO3, and CFIFO5.

19.2.1.17 Voltage Reference High and Voltage Reference Low (VRH, VRL)

VRH and VRL are voltage references for the ADCs. VRH is the highest voltage reference, while VRL is the lowest voltage reference.

19.2.1.18 Power Supplies for Analog Components (V_{DDA}, V_{SSA})

V_{DDA} is the positive power supply pin for the ADCs and V_{SSA} is the negative power supply pin for the ADCs. Refer to electrical specifications.

19.2.1.19 Reference Bypass Capacitor (REFBYPC)

The REFBYPC pin is used to connect an external bias capacitor between the REFBYPC pin and VRL. The value of this capacitor should be 100nF. This bypass capacitor is used to provide a stable reference voltage for the ADC.

19.3 Memory Map/Register Definition

This section provides memory maps and detailed descriptions of all registers. Data written to or read from reserved areas of the memory map is undefined.

19.3.1 eQADC Memory Map

This section provides memory maps for the eQADC.

Table 19-2. eQADC Memory Map

Address	Register Name	Register Description	Size (bits)
Base (0xFFF8_0000)	EQADC_MCR	EQADC module configuration register	32
Base + 0x004	—	Reserved	—
Base + 0x008	EQADC_NMSFR	eQADC null message send format register	32
Base + 0x00C	EQADC_ETDFR	eQADC external trigger digital filter register	32
Base + 0x010	EQADC_CFPR0	eQADC command FIFO push register 0	32
Base + 0x014	EQADC_CFPR1	eQADC command FIFO push register 1	32
Base + 0x018	EQADC_CFPR2	eQADC command FIFO push register 2	32
Base + 0x01C	EQADC_CFPR3	eQADC command FIFO push register 3	32
Base + 0x020	EQADC_CFPR4	eQADC command FIFO push register 4	32
Base + 0x024	EQADC_CFPR5	eQADC command FIFO push register 5	32
Base + 0x028	—	Reserved	—
Base + 0x02C	—	Reserved	—
Base + 0x030	EQADC_RFPR0	eQADC result FIFO pop register 0	32
Base + 0x034	EQADC_RFPR1	eQADC result FIFO pop register 1	32
Base + 0x038	EQADC_RFPR2	eQADC result FIFO pop register 2	32
Base + 0x03C	EQADC_RFPR3	eQADC result FIFO pop register 3	32
Base + 0x040	EQADC_RFPR4	eQADC result FIFO pop register 4	32
Base + 0x044	EQADC_RFPR5	eQADC result FIFO pop register 5	32
Base + 0x048	—	Reserved	—
Base + 0x04C	—	Reserved	—
Base + 0x050	EQADC_CFCR0	eQADC command FIFO control register 0	16
Base + 0x052	EQADC_CFCR1	eQADC command FIFO control register 1	16
Base + 0x054	EQADC_CFCR2	eQADC command FIFO control register 2	16
Base + 0x056	EQADC_CFCR3	eQADC command FIFO control register 3	16
Base + 0x058	EQADC_CFCR4	eQADC command FIFO control register 4	16
Base + 0x05A	EQADC_CFCR5	eQADC command FIFO control register 5	16
Base + 0x05C	—	Reserved	—

Table 19-2. eQADC Memory Map (continued)

Address	Register Name	Register Description	Size (bits)
Base + 0x060	EQADC_IDCR0	eQADC interrupt and eDMA control register 0	16
Base + 0x062	EQADC_IDCR1	eQADC interrupt and eDMA control register 1	16
Base + 0x064	EQADC_IDCR2	eQADC interrupt and eDMA control register 2	16
Base + 0x066	EQADC_IDCR3	eQADC interrupt and eDMA control register 3	16
Base + 0x068	EQADC_IDCR4	eQADC interrupt and eDMA control register 4	16
Base + 0x06A	EQADC_IDCR5	eQADC interrupt and eDMA control register 5	16
Base + 0x06C	—	Reserved	—
Base + 0x070	EQADC_FISR0	eQADC FIFO and interrupt status register 0	32
Base + 0x074	EQADC_FISR1	eQADC FIFO and interrupt status register 1	32
Base + 0x078	EQADC_FISR2	eQADC FIFO and interrupt status register 2	32
Base + 0x07C	EQADC_FISR3	eQADC FIFO and interrupt status register 3	32
Base + 0x080	EQADC_FISR4	eQADC FIFO and interrupt status register 4	32
Base + 0x084	EQADC_FISR5	eQADC FIFO and interrupt status register 5	32
Base + 0x088	—	Reserved	—
Base + 0x08C	—	Reserved	—
Base + 0x090	EQADC_CFTCR0	eQADC command FIFO transfer counter register 0	16
Base + 0x092	EQADC_CFTCR1	eQADC command FIFO transfer counter register 1	16
Base + 0x094	EQADC_CFTCR2	eQADC command FIFO transfer counter register 2	16
Base + 0x096	EQADC_CFTCR3	eQADC command FIFO transfer counter register 3	16
Base + 0x098	EQADC_CFTCR4	eQADC command FIFO transfer counter register 4	16
Base + 0x09A	EQADC_CFTCR5	eQADC command FIFO transfer counter register 5	16
Base + 0x09C	—	Reserved	—
Base + 0x0A0	EQADC_CFSSR0	eQADC command FIFO status snapshot register 0	32
Base + 0x0A4	EQADC_CFSSR1	eQADC command FIFO status snapshot register 1	32
Base + 0x0A8	EQADC_CFSSR2	eQADC command FIFO status snapshot register 2	32
Base + 0x0AC	EQADC_CFSR	eQADC command FIFO status register	32
Base + 0x0B0	—	Reserved	—
Base + 0x0B4	EQADC_SSICR	eQADC synchronous serial interface control register	32
Base + 0x0B8	EQADC_SSIRDR	eQADC synchronous serial interface receive data register	32
Base + 0x0BC– Base + 0x0FC	—	Reserved	—
Base + 0x100– Base + 0x10C	EQADC_CF0R*n*	eQADC CFIFO0 registers 0–3	32

Table 19-2. eQADC Memory Map (continued)

Address	Register Name	Register Description	Size (bits)
Base + 0x110– Base + 0x13C	—	Reserved	—
Base + 0x140– Base + 0x14C	EQADC_CF1R*n*	eQADC CFIFO1 registers 0–3	32
Base + 0x150– Base + 0x17C	—	Reserved	—
Base + 0x180– Base + 0x18C	EQADC_CF2R*n*	eQADC CFIFO2 registers 0–3	32
Base + 0x190– Base + 0x1BC	—	Reserved	—
Base + 0x1C0– Base + 0x1CC	EQADC_CF3R*n*	eQADC CFIFO3 registers 0–3	32
Base + 0x1D0– Base + 0x1FC	—	Reserved	—
Base + 0x200– Base + 0x20C	EQADC_CF4R*n*	eQADC CFIFO4 registers 0–3	32
Base + 0x210– Base + 0x23C	—	Reserved	—
Base + 0x240– Base + 0x24C	EQADC_CF5R*n*	eQADC CFIFO5 registers 0–3	32
Base + 0x250– Base + 0x2FC	—	Reserved	—
Base + 0x300– Base + 0x30C	EQADC_RF0R*n*	eQADC RFIFO0 registers 0–3	32
Base + 0x310– Base + 0x33C	—	Reserved	—
Base + 0x340– Base + 0x34C	EQADC_RF1R*n*	eQADC RFIFO1 registers 0–3	32
Base + 0x350– Base + 0x37C	—	Reserved	—
Base + 0x380– Base + 0x38C	EQADC_RF2R*n*	eQADC RFIFO2 registers 0–3	32
Base + 0x390– Base + 0x3BC	—	Reserved	—
Base + 0x3C0– Base + 0x3CC	EQADC_RF3R*n*	eQADC RFIFO3 registers 0–3	32
Base + 0x3D0– Base + 0x3FC	—	Reserved	—
Base + 0x400– Base + 0x40C	EQADC_RF4R*n*	eQADC RFIFO4 registers 0–3	32

Table 19-2. eQADC Memory Map (continued)

Address	Register Name	Register Description	Size (bits)
Base + 0x410– Base + 0x43C	—	Reserved	—
Base + 0x440– Base + 0x44C	EQADC_RF5R*n*	eQADC RFIFO5 registers 0–3	32
Base + 0x450– Base + 0x7FC	—	Reserved	—

19.3.2 eQADC Register Descriptions

19.3.2.1 eQADC Module Configuration Register (EQADC_MCR)

The EQADC_MCR contains bits used to control how the eQADC responds to a debug mode entry request, and to enable the eQADC SSI interface.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base+ 0x000															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	ESSIE		0	DBG	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base+ 0x000															

Figure 19-2. eQADC Module Configuration Register (EQADC_MCR)

Table 19-3. EQADC_MCR Field Descriptions

Bits	Name	Description
0–26	—	Reserved.
27–28	ESSIE [0:1]	eQADC synchronous serial interface enable. Defines the eQADC synchronous serial interface operation. 00 eQADC SSI is disabled 01 Reserved 10 eQADC SSI is enabled, FCK is free running, and serial transmissions are disabled 11 eQADC SSI is enabled, FCK is free running, and serial transmissions are enabled

Table 19-3. EQADC_MCR Field Descriptions

Bits	Name	Description
29	—	Reserved.
30–31	DBG [0:1]	Debug enable. Defines the eQADC response to a debug mode entry request. 00 Do not enter debug mode 01 Reserved 10 Enter debug mode. If the eQADC SSI is enabled, FCK stops while the eQADC is in debug mode. 11 Enter debug mode. If the eQADC SSI is enabled, FCK is free running while the eQADC is in debug mode

NOTE

Disabling the eQADC SSI (0b00 write to ESSIE) or serial transmissions from the eQADC SSI (0b10 write to ESSIE) while a serial transmission is in progress results in the abort of that transmission.

NOTE

When disabling the eQADC SSI, the FCK will not stop until it reaches its low phase.

19.3.2.2 eQADC Null Message Send Format Register (EQADC_NMSFR)

The EQADC_NMSFR defines the format of the null message sent to the external device.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	NMF									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x008															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NMF															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x008															

Figure 19-3. eQADC Null Message Send Format Register (EQADC_NMSFR)

Table 19-4. EQADC_NMSFR Field Descriptions

Bits	Name	Description
0–5	—	Reserved.
6–31	NMF [0:25]	Null message format. Contains the programmable null message send value for the eQADC. The value written to this register will be sent as a null message when serial transmissions from the eQADC SSI are enabled (ESSIE field is configured to 0b11 in EQADC_MCR (Section 19.3.2.1)) and either • there are no triggered CFIFOs with commands bound for external command buffers, or; • there are triggered CFIFOs with commands bound for external command buffers but the external command buffers are full. Refer to Section for more information on the format of a null message.

NOTE

The eQADC null message send format register only affects how the eQADC sends a null message, but it has no control on how the eQADC detects a null message on receiving data. The eQADC detects a null message by decoding the MESSAGE_TAG field on the receive data. Refer to Table 19-34 for more information on the MESSAGE_TAG field.

NOTE

Writing to the eQADC null message send format register while serial transmissions are enabled is not recommended (See EQADC_MCR[ESSIE] field in Section 19.3.2.1).

19.3.2.3 eQADC External Trigger Digital Filter Register (EQADC_ETDFR)

The EQADC_ETDFR is used to set the minimum time a signal must be held in a logic state on the CFIFO triggers inputs to be recognized as an edge or level gated trigger. The digital filter length field specifies the minimum number of system clocks that must be counted by the digital filter counter to recognize a logic state change.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x00C															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	DFL			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x00C															

Figure 19-4. eQADC External Trigger Digital Filter Register (EQADC_ETDFR)

Table 19-5. EQADC_ETDFR Field Description Table

Bits	Name	Description
0–27	—	Reserved.
28–31	DFL [0:3]	Digital filter length. Specifies the minimum number of system clocks that must be counted by the digital filter counter to recognize a logic state change. The count specifies the sample period of the digital filter which is calculated according to the following equation: $FilterPeriod = (SystemClockPeriod \times 2^{DFL}) + 1(SystemClockPeriod)$ Minimum clock counts for which an ETRIG signal needs to be stable to be passed through the filter are shown in Table 19-6. Refer to Section 19.4.3.4, "External Trigger Event Detection," for more information on the digital filter. **Note:** The DFL field must only be written when the MODE*n* of all CFIFOs are configured to disabled.

Table 19-6. Minimum Required Time to Valid ETRIG

DFL[0:3]	Minimum Clock Count	Minimum Time (ns) (System Clock = 120MHz)
0b0000	2	16.67
0b0001	3	25.00
0b0010	5	41.67
0b0011	9	75.00
0b0100	17	141.67
0b0101	33	275.00
0b0110	65	541.67
0b0111	129	1075.00
0b1000	257	2141.67
0b1001	513	4275.00
0b1010	1025	8541.67
0b1011	2049	17075.00
0b1100	4097	34141.67
0b1101	8193	68275.00
0b1110	16385	136541.67
0b1111	32769	273075.00

19.3.2.4 eQADC CFIFO Push Registers 0–5 (EQADC_CFPR*n*)

The EQADC_CFPRs provide a mechanism to fill the CFIFOs with command messages from the command queues. Refer to Section 19.4.3, "eQADC Command FIFOs," for more information on the CFIFOs and to Section 19.4.1.2, "Message Format in eQADC," for a description on command message formats.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	CF_PUSH*n*															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x010 (EQADC_CFPR0); Base + 0x014 (EQADC_CFPR1); Base + 0x018 (EQADC_CFPR2); Base + 0x01C (EQADC_CFPR3); Base + 0x020 (EQADC_CFPR4); Base + 0x024 (EQADC_CFPR5)															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	CF_PUSH*n*															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x010 (EQADC_CFPR0); Base + 0x014 (EQADC_CFPR1); Base + 0x018 (EQADC_CFPR2); Base + 0x01C (EQADC_CFPR3); Base + 0x020 (EQADC_CFPR4); Base + 0x024 (EQADC_CFPR5)															

Figure 19-5. eQADC CFIFO Push Registers (EQADC_CFPR*n*)

Table 19-7. EQADC_CFPR*n* Field Description

Bits	Name	Description
0–31	CF_PUSH*n*	CFIFO push data *n*. When CFIFO*n* is not full, writing to the whole word or any bytes of EQADC_CFPR*n* will push the 32-bit CF_PUSH*n* value into CFIFO*n*. Writing to the CF_PUSH*n* field also increments the corresponding CFCTR*n* value by one in Section 19.3.2.8, “eQADC FIFO and Interrupt Status Registers 0–5 (EQADC_FISRn).” When the CFIFO*n* is full, the eQADC ignores any write to the CF_PUSH*n*. Reading the EQADC_CFPR*n* always returns 0. **Note:** Only whole words must be written to EQADC_CFPR. Writing half-words or bytes to EQADC_CFPR will still push the whole 32-bit CF_PUSH field into the corresponding CFIFO, but undefined data will fill the areas of CF_PUSH that were not specifically designated as target locations for the write.

19.3.2.5 eQADC Result FIFO Pop Registers 0–5 (EQADC_RFPR*n*)

The eQADC_RFPRs provide a mechanism to retrieve data from RFIFOs.

NOTE

The EQADC_RFPR*n* must not be read speculatively. For future compatibility, the TLB entry covering the EQADC_RFPR*n* must be configured to be guarded.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x030 (EQADC_RFPR0); Base + 0x034 (EQADC_RFPR1); Base + 0x038 (EQADC_RFPR2); Base + 0x03C (EQADC_RFPR3); Base + 0x040 (EQADC_RFPR4); Base + 0x044 (EQADC_RFPR5)															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RF_POP*n*															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x030 (EQADC_RFPR0); Base + 0x034 (EQADC_RFPR1); Base + 0x038 (EQADC_RFPR2); Base + 0x03C (EQADC_RFPR3); Base + 0x040 (EQADC_RFPR4); Base + 0x044 (EQADC_RFPR5)															

Figure 19-6. eQADC RFIFO Pop Registers (EQADC_RFPR*n*)

Table 19-8. EQADC_RFPR*n* Field Description

Bits	Name	Description
0–15	—	Reserved.
16–31	RF_POP*n* [0:15]	Result FIFO pop data *n*. When RFIFO*n* is not empty, the RF_POP*n* contains the next unread entry value of RFIFO*n*. Reading the whole word, a half-word, or any bytes of EQADC_RFPR*n* will pop one entry from RFIFO*n*, and the corresponding RFCTR*n* value will be decremented by 1 (See Section 19.3.2.8*)*. When the RFIFO*n* is empty, any read on EQADC_RFPR*n* returns undefined data value and does not decrement the RFCTR*n* value. Writing to EQADC_RFPR*n* has no effect.

19.3.2.6 eQADC CFIFO Control Registers 0–5 (EQADC_CFCR*n*)

The eQADC_CFCRs contain bits that affect CFIFOs. These bits specify the CFIFO operation mode and can invalidate all of the CFIFO contents.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	MODE*n*				0	0	0	0
W						SSE*n*	CFINV*n*									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	EQADC_BASE+0x050 (EQADC_CFCR0); EQADC_BASE+0x052 (EQADC_CFCR1); EQADC_BASE+0x054 (EQADC_CFCR2) EQADC_BASE+0x056 (EQADC_CFCR3); EQADC_BASE+0x058 (EQADC_CFCR4); EQADC_BASE+0x05A (EQADC_CFCR5)															

Figure 19-7. eQADC CFIFO Control Registers (EQADC_CFCR*n*)

Table 19-9. EQADC_CFCR*n* Field Descriptions

Bits	Name	Description
0–4	—	Reserved.
5	SSE*n*	CFIFO single-scan enable bit *n*. Used to set the SSS*n* bit, as described in Section 19.3.2.8. Writing a 1 to SSE*n* will set the SSS*n* if the CFIFO is in single-scan mode. When SSS*n* is already asserted, writing a 1 to SSE*n* has no effect. If the CFIFO is in continuous-scan mode or is disabled, writing a 1 to SSE*n* will not set SSS*n*. Writing a 0 to SSE*n* has no effect. SSE*n* always is read as 0. 0 No effect. 1 Set the SSS*n* bit.
6	CFINV*n*	CFIFO invalidate bit *n*. Causes the eQADC to invalidate all entries of CFIFO*n*. Writing a 1 to CFINV*n* will reset the value of CFCTR*n* in the EQADC_FISR register (refer to Section 19.3.2.8, “eQADC FIFO and Interrupt Status Registers 0–5 (EQADC_FISRn).” Writing a 1 to CFINV*n* also resets the push next data pointer, transfer next data pointer to the first entry of CFIFO*n* in Figure 19-35. CFINV*n* always is read as 0. Writing a 0 has no effect. 0 No effect. 1 Invalidate all of the entries in the corresponding CFIFO. **Note:** Writing CFINV*n* only invalidates commands stored in CFIFO*n*; previously transferred commands that are waiting for execution, that is commands stored in the ADC command buffers, will still be executed, and results generated by them will be stored in the appropriate RFIFO. **Note:** CFINV*n* must not be written unless the MODE*n* is configured to disabled, and CFIFO status is IDLE.
7	—	Reserved.
8–11	MODE*n* [0:3]	CFIFO operation mode *n*. Selects the CFIFO operation mode for CFIFO*n*. Refer to Section 19.4.3.5, “CFIFO Scan Trigger Modes,” for more information on CFIFO trigger mode. **Note:** If MODE*n* is not disabled, it must not be changed to any other mode besides disabled. If MODE*n* is disabled and the CFIFO status is IDLE, MODE*n* can be changed to any other mode.
12–15	—	Reserved.

Table 19-10. CFIFO Operation Mode Table

MODE*n*[0:3]	CFIFO Operation Mode
0b0000	Disabled
0b0001	Software trigger, single scan
0b0010	Low level gated external trigger, single scan
0b0011	High level gated external trigger, single scan
0b0100	Falling edge external trigger, single scan
0b0101	Rising edge external trigger, single scan
0b0110	Falling or rising edge external trigger, single scan
0b0111–0b1000	Reserved
0b1001	Software trigger, continuous scan

Table 19-10. CFIFO Operation Mode Table (continued)

MODE*n*[0:3]	CFIFO Operation Mode
0b1010	Low level gated external trigger, continuous scan
0b1011	High level gated external trigger, continuous scan
0b1100	Falling edge external trigger, continuous scan
0b1101	Rising edge external trigger, continuous scan
0b1110	Falling or rising edge external trigger, continuous scan
0b1111	Reserved

19.3.2.7 eQADC Interrupt and eDMA Control Registers 0–5 (EQADC_IDCR*n*)

The eQADC_IDCRs contain bits to enable the generation of interrupt or eDMA requests when the corresponding flag bits are set in EQADC_FISRn (Section 19.3.2.8).

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	NCIE*n*	TORIE*n*	PIE*n*	EOQIE*n*	CFUIE*n*	0	CFFE*n*	CFFS*n*	0	0	0	0	RFOIE*n*	0	RFDE*n*	RFDS*n*
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	EQADC_BASE+0x060 (EQADC_IDCR0); EQADC_BASE+0x062 (EQADC_IDCR1); EQADC_BASE+0x064 (EQADC_IDCR2) EQADC_BASE+0x066 (EQADC_IDCR3); EQADC_BASE+0x068 (EQADC_IDCR4); EQADC_BASE+0x06A (EQADC_IDCR5)															

Figure 19-8. eQADC Interrupt and eDMA Control Registers (EQADC_IDCR*n*)

Table 19-11. EQADC_IDCR*n* Field Descriptions

Bits	Name	Description
0	NCIE*n*	Non-coherency interrupt enable *n*. Enables the eQADC to generate an interrupt request when the corresponding NCF*n*, described in Section 19.3.2.8, is asserted. 0 Disable non-coherency interrupt request 1 Enable non-coherency interrupt request
1	TORIE*n*	Trigger overrun interrupt enable *n*. Enables the eQADC to generate an interrupt request when the corresponding TORF*n* (described in Section 19.3.2.8) is asserted. Apart from generating an independent interrupt request for a CFIFO*n* trigger overrun event, the eQADC also provides a combined interrupt at which the result FIFO overflow interrupt, the command FIFO underflow interrupt, and the command FIFO trigger overrun interrupt requests of all CFIFOs are ORed. When RFOIE*n*, CFUIE*n*, and TORIE*n* are all asserted, this combined interrupt request is asserted whenever one of the following 18 flags becomes asserted: RFOF*n*, CFUF*n*, and TORF*n* (assuming that all interrupts are enabled). See Section 19.4.7, “eQADC eDMA/Interrupt Request,” for details. 0 Disable trigger overrun interrupt request 1 Enable trigger overrun interrupt request
2	PIE*n*	Pause interrupt enable *n*. Enables the eQADC to generate an interrupt request when the corresponding PFx in EQADC_FISRn (See Section 19.3.2.8) is asserted. 0 Disable pause interrupt request 1 Enable pause interrupt request

Table 19-11. EQADC_IDCR*n* Field Descriptions (continued)

Bits	Name	Description
3	EOQIE*n*	End-of-queue interrupt enable *n*. Enables the eQADC to generate an interrupt request when the corresponding EOQF*n* in EQADC_FISRn (See Section 19.3.2.8) is asserted. 0 Disable end of queue interrupt request. 1 Enable end of queue interrupt request.
4	CFUIE*n*	CFIFO underflow interrupt enable *n*. Enables the eQADC to generate an interrupt request when the corresponding CFUF*n* in EQADC_FISRn (See Section 19.3.2.8) is asserted. Apart from generating an independent interrupt request for a CFIFO*n* underflow event, the eQADC also provides a combined interrupt at which the result FIFO overflow interrupt, the command FIFO underflow interrupt, and the command FIFO trigger overrun interrupt requests of all CFIFOs are ORed. When RFOIE*n*, CFUIE*n*, and TORIE*n* are all asserted, this combined interrupt request is asserted whenever one of the following 18 flags becomes asserted: RFOF*n*, CFUF*n*, and TORF*n* (assuming that all interrupts are enabled). See Section 19.4.7, “eQADC eDMA/Interrupt Request,” for details. 0 Disable underflow interrupt request 1 Enable underflow interrupt request
5	—	Reserved.
6	CFFE*n*	CFIFO fill enable *n*. Enables the eQADC to generate an interrupt request (CFFS*n* is asserted) or eDMA request (CFFS*n* is negated) when CFFF*n* in EQADC_FISRn (Section 19.3.2.8) is asserted. 0 Disable CFIFO fill eDMA or interrupt request 1 Enable CFIFO fill eDMA or interrupt request **Note:** CFFE*n* must not be negated while an eDMA transaction is in progress.
7	CFFS*n*	CFIFO fill select *n*. Selects if an eDMA or interrupt request is generated when CFFF*n* in EQADC_FISRn (See Section 19.3.2.8) is asserted. If CFFE*n* is asserted, the eQADC generates an interrupt request when CFFS*n* is negated, or it generates an eDMA request if CFFS*n* is asserted. 0 Generate interrupt request to move data from the system memory to CFIFO*n*. 1 Generate eDMA request to move data from the system memory to CFIFO*n*. **Note:** CFFS*n* must not be negated while an eDMA transaction is in progress.
8–11	—	Reserved.
12	RFOIE*n*	RFIFO overflow interrupt enable *n*. Enables the eQADC to generate an interrupt request when the corresponding RFOF*n* in EQADC_FISRn (See Section 19.3.2.8) is asserted. Apart from generating an independent interrupt request for an RFIFO*n* overflow event, the eQADC also provides a combined interrupt at which the result FIFO overflow Interrupt, the command FIFO underflow interrupt, and the command FIFO trigger overrun interrupt requests of all CFIFOs are ORed. When RFOIE*n*, CFUIE*n,* and TORIE*n* are all asserted, this combined interrupt request is asserted whenever one of the following 18 flags becomes asserted: RFOF*n,* CFUF*n*, and TORF*n* (assuming that all interrupts are enabled). See Section 19.4.7, “eQADC eDMA/Interrupt Request,” for details. 0 Disable overflow interrupt request 1 Enable overflow Interrupt request
13	—	Reserved.

Table 19-11. EQADC_IDCR*n* Field Descriptions (continued)

Bits	Name	Description
14	RFDE*n*	RFIFO drain enable *n*. Enables the eQADC to generate an interrupt request (RFDS*n* is asserted) or eDMA request (RFDS*n* is negated) when RFDF*n* in EQADC_FISRn (See Section 19.3.2.8) is asserted. 0 Disable RFIFO drain eDMA or interrupt request 1 Enable RFIFO drain eDMA or interrupt request **Note:** RFDE*n* must not be negated while an eDMA transaction is in progress.
15	RFDS*n*	RFIFO drain select *n*. Selects if an eDMA or interrupt request is generated when RFDF*n* in EQADC_FISRn (See Section 19.3.2.8) is asserted. If RFDE*n* is asserted, the eQADC generates an interrupt request when RFDS*n* is negated, or it generates an eDMA request when RFDS*n* is asserted. 0 Generate interrupt request to move data from RFIF*n* to the system memory 1 Generate eDMA request to move data from RFIFO*n* to the system memory **Note:** RFDS*n* must not be negated while an eDMA transaction is in progress.

19.3.2.8 eQADC FIFO and Interrupt Status Registers 0–5 (EQADC_FISR*n*)

The EQADC_FISRs contain flag and status bits for each CFIFO and RFIFO pair. Writing 1 to a flag bit clears it. Writing 0 has no effect. Status bits are read only. These bits indicate the status of the FIFO itself.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	NCF*n*	TORF*n*	PF*n*	EOQF*n*	CFUF*n*	SSS*n*	CFFF*n*	0	0	0	0	0	RFOF*n*	0	RFDF*n*	0
W	w1c	w1c	w1c	w1c	w1c		w1c						w1c		w1c	
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x070 (EQADC_FISR0); Base + 0x074 (EQADC_FISR1); Base + 0x078 (EQADC_FISR2); Base + 0x07C (EQADC_FISR3); Base + 0x080 (EQADC_FISR4); Base + 0x084 (EQADC_FISR5)															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CFCTR*n*				TNXTPTR*n*				RFCTR*n*				POPNXTPTR*n*			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x070 (EQADC_FISR0); Base + 0x074 (EQADC_FISR1); Base + 0x078 (EQADC_FISR2); Base + 0x07C (EQADC_FISR3); Base + 0x080 (EQADC_FISR4); Base + 0x084 (EQADC_FISR5)															

Figure 19-9. eQADC FIFO and Interrupt Status Registers (EQADC_FISR*n*)

Table 19-12. EQADC_FISR*n* Field Descriptions

Bits	Name	Description
0	NCF*n*	Non-coherency flag *n*. NCF*n* is set whenever a command sequence being transferred through CFIFO*n* becomes non-coherent. If NCIE*n* in EQADC_IDCRn (See Section 19.3.2.7) and NCF*n* are asserted, an interrupt request will be generated. Writing a 1 clears NCF*n*. Writing a 0 has no effect. More for information on non-coherency refer to Section 19.4.3.6.5, "Command Sequence Non-Coherency Detection." 0 Command sequence being transferred by CFIFO*n* is coherent 1 Command sequence being transferred by CFIFO*n* became non-coherent **Note:** Non-coherency means that a command in the command FIFO was not immediately executed, but delayed. This may occur if the command is pre-empted, where a higher priority queue is triggered and has a competing conversion command for the same converter.
1	TORF*n*	Trigger overrun flag for CFIFO *n*. TORF*n* is set when trigger overrun occurs for the specified CFIFO in edge or level trigger mode. Trigger overrun occurs when an already triggered CFIFO receives an additional trigger. When EQADC_IDCRn[TORIE*n]* is set (See Section 19.3.2.7) and TORF*n* are asserted, an interrupt request will be generated. Apart from generating an independent interrupt request for a CFIFO*n* trigger overrun event, the eQADC also provides a combined interrupt at which the result FIFO overflow interrupt, the command FIFO underflow interrupt, and the command FIFO trigger overrun Interrupt requests of all CFIFOs are ORed. When RFOIE*n*, CFUIE*n*, and TORIE*n* are all asserted, this combined interrupt request is asserted whenever one of the following 18 flags becomes asserted: RFOF*n*, CFUF*n*, and TORF*n* (assuming that all interrupts are enabled). See Section 19.4.7, "eQADC eDMA/Interrupt Request," for details. Write 1 to clear the TORF*n* bit. Writing 0 has no effect. 0 No trigger overrun occurred 1 Trigger overrun occurred **Note:** The trigger overrun flag will not set for CFIFOs configured for software trigger mode.

Table 19-12. EQADC_FISR*n* Field Descriptions (continued)

Bits	Name	Description
2	PF*n*	Pause flag *n*. PF behavior changes according to the CFIFO trigger mode. • In edge trigger mode, PF*n* is set when the eQADC completes the transfer of an entry with an asserted pause bit from CFIFO*n*. • In level trigger mode, when CFIFO*n* is in the TRIGGERED state, PF*n* is set when CFIFO status changes from TRIGGERED due to the detection of a closed gate. An interrupt routine, generated due to the asserted PF, can be used to verify if a complete scan of the user-defined command queue was performed. If a closed gate is detected while no command transfers are taking place, it will have immediate effect on the CFIFO status. If a closed gate is detected while a command transfer to an on-chip ADC is taking place, it will only affect the CFIFO status when the transfer completes. If a closed gate is detected during the serial transmission of a command to the external device, it will have no effect on the CFIFO status until the transmission completes. The transfer of entries bound for the on-chip ADCs is considered completed when they are stored in the appropriate ADC command buffer. The transfer of entries bound for the external device is considered completed when the serial transmission of the entry is completed. In software trigger mode, PF*n* will never become asserted. If PIE*n* (See Section 19.3.2.7) and PF*n* are asserted, an interrupt will be generated. Writing a 1 clears the PF*n*. Writing a 0 has no effect. Refer to Section 19.4.3.6.3, “Pause Status,” for more information on pause flag. 0 Entry with asserted pause bit was not transferred from CFIFO*n* (CFIFO in edge trigger mode), or CFIFO status did not change from the TRIGGERED state due to detection of a closed gate (CFIFO in level trigger mode). 1 Entry with asserted pause bit was transferred from CFIFO*n* (CFIFO in edge trigger mode), or CFIFO status changes from the TRIGGERED state due to detection of a closed gate (CFIFO in level trigger mode). **Note:** In edge trigger mode, an asserted PF*n* only implies that the eQADC has finished transferring a command with an asserted pause bit from CFIFO*n*. It does not imply that result data for the current command and for all previously transferred commands has been returned to the appropriate RFIFO. **Note:** In software or level trigger mode, when the eQADC completes the transfer of an entry from CFIFO*n* with an asserted pause bit, PF*n* will not be set and transfer of commands will continue without pausing.
3	EOQF*n*	End-of-queue flag *n*. Indicates that an entry with an asserted EOQ bit was transferred from CFIFO*n* to the on-chip ADCs or to the external device. See Section 19.4.1.2, “Message Format in eQADC,” for details about command message formats. When the eQADC completes the transfer of an entry with an asserted EOQ bit from CFIFO*n*, EOQF*n* will be set. The transfer of entries bound for the on-chip ADCs is considered completed when they are stored in the appropriate command buffer. The transfer of entries bound for the external device is considered completed when the serial transmission of the entry is completed. If the EOQIE*n* bit (See 19.3.2.7) and EOQF*n* are asserted, an interrupt will be generated. Writing a 1 clears the EOQF*n* bit. Writing a 0 has no effect. Refer to Section 19.4.3.6.2, “Command Queue Completion Status,” for more information on end-of-queue flag. 0 Entry with asserted EOQ bit was not transferred from CFIFO*n* 1 Entry with asserted EOQ bit was transferred from CFIFO*n* **Note:** An asserted EOQF*n* only implies that the eQADC has finished transferring a command with an asserted EOQ bit from CFIFO*n*. It does not imply that result data for the current command and for all previously transferred commands has been returned to the appropriate RFIFO.

Table 19-12. EQADC_FISR*n* Field Descriptions (continued)

Bits	Name	Description
4	CFUF*n*	CFIFO underflow flag *n*. Indicates an underflow event on CFIFO*n*. CFUF*n* is set when CFIFO*n* is in the TRIGGERED state and it becomes empty. No commands will be transferred from an underflowing CFIFO, nor will command transfers from lower priority CFIFOs be blocked. When CFUIE*n* (see Section 19.3.2.7) and CFUF*n* are both asserted, the eQADC generates an interrupt request. Apart from generating an independent interrupt request for a CFIFO*n* underflow event, the eQADC also provides a combined interrupt at which the result FIFO overflow interrupt, the command FIFO underflow interrupt, and the command FIFO trigger overrun interrupt requests of all CFIFOs are ORed. When RFOIE*n*, CFUIE*n*, and TORIE*n* are all asserted, this combined interrupt request is asserted whenever one of the following 18 flags becomes asserted: RFOF*n*, CFUF*n*, and TORF*n* (assuming that all interrupts are enabled). See Section 19.4.7, "eQADC eDMA/Interrupt Request," for details. Writing a 1 clears CFUF*n*. Writing a 0 has no effect. 0 No CFIFO underflow event occurred 1 A CFIFO underflow event occurred
5	SSS*n*	CFIFO single-scan status bit *n*. When asserted, enables the detection of trigger events for CFIFOs programmed into single-scan level- or edge-trigger mode, and works as trigger for CFIFOs programmed into single-scan software-trigger mode. Refer to Section 19.4.3.5.2, "Single-Scan Mode," for further details. The SSS*n* bit is set by writing a 1 to the SSE*n* bit (see Section 19.3.2.6). The eQADC clears the SSS*n* bit when a command with an asserted EOQ bit is transferred from a CFIFO in single-scan mode, when a CFIFO is in single-scan level-trigger mode and its status changes from the TRIGGERED state due to the detection of a closed gate, or when the value of the CFIFO operation mode MODE*n* (see Section 19.3.2.6) is changed to disabled. Writing to SSS*n* has no effect. SSS*n* has no effect in continuous-scan or in disabled mode. 0 CFIFO in single-scan level- or edge-trigger mode will ignore trigger events, or CFIFO in single-scan software-trigger mode is not triggered. 1 CFIFO in single-scan level- or edge-trigger mode will detect a trigger event, or CFIFO in single-scan software-trigger mode is triggered.
6	CFFF*n*	CFIFO fill flag *n*. CFFF*n* is set when the CFIFO*n* is not full. When CFFE*n* (see Section 19.3.2.7) and CFFF*n* are both asserted, an interrupt or an eDMA request will be generated depending on the status of the CFFS*n* bit. When CFFS*n* is negated (interrupt requests selected), software clears CFFF*n* by writing a 1 to it. Writing a 0 has no effect. When CFFS*n* is asserted (eDMA requests selected), CFFF*n* is automatically cleared by the eQADC when the CFIFO becomes full. 0 CFIFO*n* is full. 1 CFIFO*n* is not full. **Note:** When generation of interrupt requests is selected (CFFS*n*=0), CFFF*n* must only be cleared in the ISR after the CFIFO*n* push register is accessed. **Note:** CFFF*n* should not be cleared when CFFSn is asserted (eDMA requests selected).
7–11	—	Reserved.

Table 19-12. EQADC_FISR*n* Field Descriptions (continued)

Bits	Name	Description
12	RFOF*n*	RFIFO overflow flag *n*. Indicates an overflow event on RFIFO*n*. RFOF*n* is set when RFIFO*n* is already full, and a new data is received from the on-chip ADCs or from the external device. The RFIFO*n* will not overwrite older data in the RFIFO, and the new data will be ignored. When RFOIE*n* (see Section 19.3.2.7) and RFOF*n* are both asserted, the eQADC generates an interrupt request. Apart from generating an independent interrupt request for an RFIFO*n* overflow event, the eQADC also provides a combined interrupt at which the result FIFO overflow interrupt, the command FIFO underflow interrupt, and the command FIFO trigger overrun interrupt requests of all CFIFOs are ORed. When RFOIE*n*, CFUIE*n*, and TORIE*n* are all asserted, this combined interrupt request is asserted whenever one of the following 18 flags becomes asserted: RFOF*n*, CFUF*n*, and TORF*n* (assuming that all interrupts are enabled). See Section 19.4.7, “eQADC eDMA/Interrupt Request,” for details. Write 1 to clear RFOF*n*. Writing a 0 has no effect. 0 No RFIFO overflow event occurred. 1 An RFIFO overflow event occurred.
13	—	Reserved.
14	RFDF*n*	RFIFO drain flag *n*. Indicates if RFIFO*n* has valid entries that can be drained or not. RFDF*n* is set when the RFIFO*n* has at least one valid entry in it. When RFDE*n* (see Section 19.3.2.7) and RFDF*n* are both asserted, an interrupt or an eDMA request will be generated depending on the status of the RFDS*n* bit. When RFDS*n* is negated (interrupt requests selected), software clears RFDF*n* by writing a 1 to it. Writing a 0 has no effect. When RFDS*n* is asserted (eDMA requests selected), RFDF*n* is automatically cleared by the eQADC when the RFIFO becomes empty. 0 RFIFO*n* is empty. 1 RFIFO*n* has at least one valid entry. **Note:** In the interrupt service routine, RFDF must be cleared only after the RFIFO*n* pop register is read. **Note:** RFDF*n* should not be cleared when RFDS*n* is asserted (eDMA requests selected).
15	—	Reserved.
16–19	CFCTR*n* [0:3]	CFIFO*n* entry counter. Indicates the number of commands stored in the CFIFO*n*. When the eQADC completes transferring a piece of new data from the CFIFO*n*, it decrements CFCTR*n* by 1. Writing a word or any bytes to the corresponding CFIFO Push Register (see Section 19.3.2.4) increments CFCTR*n* by 1. Writing any value to CFCTR*n* has no effect.
20–23	TNXTPTR*n* [0:3]	CFIFO*n* transfer next pointer. Indicates the index of the next entry to be removed from CFIFO*n* when it completes a transfer. When TNXTPTR*n* is 0, it points to the entry with the smallest memory-mapped address inside CFIFO*n*. TNXTPTR*n* is only updated when a command transfer is completed. If the maximum index number (CFIFO depth minus 1) is reached, TNXTPTR*n* is wrapped to 0, else, it is incremented by 1. For details refer to Section 19.4.3.1, “CFIFO Basic Functionality.” Writing any value to TNXTPTR*n* has no effect.

Table 19-12. EQADC_FISR*n* Field Descriptions (continued)

Bits	Name	Description
24–27	RFCTR*n* [0:3]	RFIFO*n* entry counter. Indicates the number of data items stored in the RFIFO*n*. When the eQADC stores a piece of new data into RFIFO*n,* it increments RFCTR*n* by 1. Reading the whole word, half-word or any bytes of the corresponding Result FIFO pop register (see Section 19.3.2.5) decrements RFCTR*n* by 1. Writing any value to RFCTR*n* itself has no effect.
28–31	POPNXTPTR*n* [0:3]	RFIFO*n* pop next pointer. Indicates the index of the entry that will be returned when EQADC_RFPR*n* is read. When POPNXTPTR*n* is 0, it points to the entry with the smallest memory-mapped address inside RFIFO*n*. POPNXTPTR*n* is updated when EQADC_RFPR*n* is read. If the maximum index number (RFIFO depth minus 1) is reached, POPNXTPTR*n* is wrapped to 0, else, it is incremented by 1. For details refer to Section 19.4.4.1, “RFIFO Basic Functionality.” Writing any value to POPNXTPTR*n* has no effect.

19.3.2.9 eQADC CFIFO Transfer Counter Registers 0–5 (EQADC_CFTCR*n*)

The EQADC_CFTCRs record the number of commands transferred from a CFIFO. The EQADC_CFTCR supports the monitoring of command transfers from a CFIFO.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	TC_CF*n*										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	EQADC_BASE+0x090 (EQADC_CFTCR0); EQADC_BASE+0x092 (EQADC_CFTCR1); EQADC_BASE+0x094 (EQADC_CFTCR2); EQADC_BASE+0x096 (EQADC_CFTCR3); EQADC_BASE+0x098 (EQADC_CFTCR4); EQADC_BASE+0x09A (EQADC_CFTCR5);															

Figure 19-10. eQADC CFIFO Transfer Counter Registers (EQADC_CFTCR*n*)

Table 19-13. EQADC_CFTCR*n* Field Descriptions

Bits	Name	Description
0–4	—	Reserved.
5–15	TC_CF*n* [0:10]	Transfer counter for CFIFO*n*. TC_CF*n* counts the number of commands that have been completely transferred from CFIFO*n*. TC_CF*n*=2, for example, signifies that two commands have been transferred. The transfer of entries bound for the on-chip ADCs is considered completed when they are stored in the appropriate command buffer. The transfer of entries bound for an external device is considered completed when the serial transmission of the entry is completed. The eQADC increments the TC_CF*n* value by 1 after a command is transferred. TC_CF*n* resets to 0 after eQADC completes transferring a command with an asserted EOQ bit. Writing any value to TC_CF*n* sets the counter to that written value. **Note:** If CFIFO*n* is in the TRIGGERED state when its MODE*n* field is programmed to disabled, the exact number of entries transferred from the CFIFO until that point (TC_CF*n*) is only known after the CFIFO status changes to IDLE, as indicated by CFS*n*. For details refer to Section 19.4.3.5.1, “Disabled Mode.”

19.3.2.10 eQADC CFIFO Status Snapshot Registers 0–2 (EQADC_CFSSR*n*)

The eQADC_CFSSRs contain status fields to track the operation status of each CFIFO and the transfer counter of the last CFIFO to initiate a command transfer to the internal ADCs and the external command buffers. EQADC_CFSSR0–1 are related to the on-chip ADC command buffers (buffers 0 and 1) while EQADC_CFSSR2 is related to the external command buffers (buffers 2 and 3). All fields of a particular EQADC_CFSSR are captured at the beginning of a command transfer to the buffer associated with that register.

Note that captured status register values are associated with a previous command transfer. This means that the eQADC_CFSSR registers capture the status registers before the status registers change, because of the transfer of the current command that is about to be popped from the CFIFO. The EQADC_CFSSRs are read only. Writing to the EQADC_CFSSRs has no effect.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CFS0_T0		CFS1_T0		CFS2_T0		CFS3_T0		CFS4_T0		CFS5_T0		0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0A0 (EQADC_CFSSR0)															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	LCFT0				TC_LCFT0										
W																
Reset	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0A0 (EQADC_CFSSR0)															

Figure 19-11. eQADC CFIFO Status Snapshot Register 0 (EQADC_CFSSR0)

Table 19-14. EQADC_CFSSR0 Field Descriptions

Bits	Name	Description
0–11	CFS*n*_T0 [0:1]	CFIFO status at transfer to ADC*n* command buffer. Indicates the CFIFO*n* status at the time a command transfer to ADC*n* command buffer is initiated. CFS*n*_T0 is a copy of the corresponding CFS*n* in EQADC_CFSR (see Section 19.3.2.11) captured at the time a command transfer to buffer*n* is initiated.
12–16	—	Reserved.

Table 19-14. EQADC_CFSSR0 Field Descriptions (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>17–20</td><td>LCFT0
[0:3]</td><td>Last CFIFO to transfer to ADCn command buffer. Holds the CFIFO number of last CFIFO to have initiated a command transfer to ADCn command buffer. LCFT0 has the following values:
<table>
<tr><th>LCFT0[0:3]</th><th>LCFT0 Meaning</th></tr>
<tr><td>0b0000</td><td>Last command was transferred from CFIFO0</td></tr>
<tr><td>0b0001</td><td>Last command was transferred from CFIFO1</td></tr>
<tr><td>0b0010</td><td>Last command was transferred from CFIFO2</td></tr>
<tr><td>0b0011</td><td>Last command was transferred from CFIFO3</td></tr>
<tr><td>0b0100</td><td>Last command was transferred from CFIFO4</td></tr>
<tr><td>0b0101</td><td>Last command was transferred from CFIFO5</td></tr>
<tr><td>0b0110–0b1110</td><td>Reserved</td></tr>
<tr><td>0b1111</td><td>No command was transferred to ADCn command buffer</td></tr>
</table></td></tr>
<tr><td>21–31</td><td>TC_LCFT0
[0:10]</td><td>Transfer counter for last CFIFO to transfer commands to ADCn command buffer. Indicates the number of commands which have been completely transferred from CFIFOn when a command transfer from CFIFOn to ADCn command buffer is initiated. TC_LCFT0 is a copy of the corresponding TC_CFn in EQADC_CFTCRn (see Section 19.3.2.9) captured at the time a command transfer from CFIFOn to ADCn command buffer is initiated. This field has no meaning when LCFT0 is 0b1111.</td></tr>
</table>

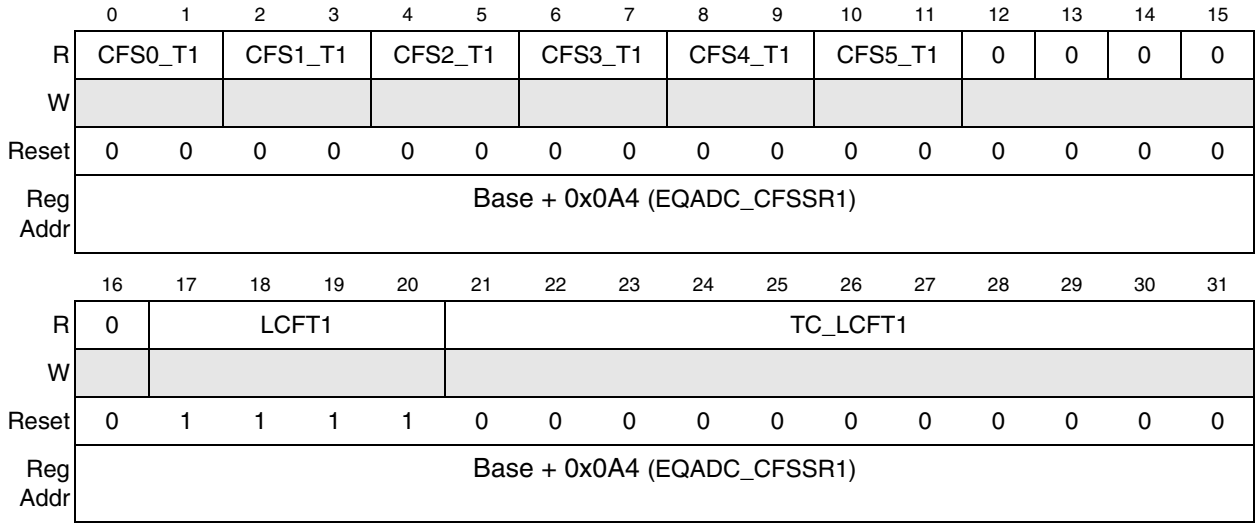


Figure 19-12. eQADC CFIFO Status Snapshot Register 1 (EQADC_CFSSR1)

Table 19-15. EQADC_CFSSR1 Field Descriptions

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>0–11</td><td>CFSn_T1
[0:1]</td><td>CFIFO status at transfer to ADCn command buffer. Indicates the CFIFOn status at the time a command transfer to ADCn command buffer is initiated. CFSn_T1 is a copy of the corresponding CFSn in EQADC_CFSR (see Section 19.3.2.11) captured at the time a command transfer to buffern is initiated.</td></tr>
<tr><td>12–16</td><td>—</td><td>Reserved.</td></tr>
<tr><td>17–20</td><td>LCFT1
[0:3]</td><td>Last CFIFO to transfer to ADCn command buffer. Holds the CFIFO number of last CFIFO to have initiated a command transfer to ADCn command buffer. LCFT1 has the following values:
<table>
<tr><th>LCFT1[0:3]</th><th>LCFT1 Meaning</th></tr>
<tr><td>0b0000</td><td>Last command was transferred from CFIFO0</td></tr>
<tr><td>0b0001</td><td>Last command was transferred from CFIFO1</td></tr>
<tr><td>0b0010</td><td>Last command was transferred from CFIFO2</td></tr>
<tr><td>0b0011</td><td>Last command was transferred from CFIFO3</td></tr>
<tr><td>0b0100</td><td>Last command was transferred from CFIFO4</td></tr>
<tr><td>0b0101</td><td>Last command was transferred from CFIFO5</td></tr>
<tr><td>0b0110–0b1110</td><td>Reserved</td></tr>
<tr><td>0b1111</td><td>No command was transferred to ADCn command buffer</td></tr>
</table></td></tr>
<tr><td>21–31</td><td>TC_LCFT1
[0:10]</td><td>Transfer counter for last CFIFO to transfer commands to ADCn command buffer. Indicates the number of commands which have been completely transferred from CFIFOn when a command transfer from CFIFOn to ADCn command buffer is initiated. TC_LCFT1 is a copy of the corresponding TC_CFn in EQADC_CFTCRn (see Section 19.3.2.9) captured at the time a command transfer from CFIFOn to ADCn command buffer is initiated. This field has no meaning when LCFT1 is 0b1111.</td></tr>
</table>

The third eQADC CFIFO status snapshot register is displayed in Figure 19-13.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CFS0_TSSI		CFS1_TSSI		CFS2_TSSI		CFS3_TSSI		CFS4_TSSI		CFS5_TSSI		0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0A8 (EQADC_CFSSR2)															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ENI	LCFTSSI				TC_LCFTSSI										
W																
Reset	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0A8 (EQADC_CFSSR2)															

Figure 19-13. eQADC CFIFO Status Snapshot Register 2 (EQADC_CFSSR2)

Table 19-16. EQADC_CFSSR2 Field Descriptions

Bits	Name	Description
0–11	CFS*n*_TSSI [0:1]	CFIFO Status at Transfer through the eQADC SSI. Indicates the CFIFO*n* status at the time a serial transmission through the eQADC SSI is initiated. CFS*n*_TSSI is a copy of the corresponding CFS*n* in EQADC_CFSR (see Section 19.3.2.11) captured at the time a serial transmission through the eQADC SSI is initiated.
12–15	—	Reserved.
16	ENI	External command buffer number Indicator. Indicates to which external command buffer the last command was transmitted. 0 Last command was transferred to command buffer 2. 1 Last command was transferred to command buffer 3.

Table 19-16. EQADC_CFSSR2 Field Descriptions (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>17–20</td><td>LCFTSSI [0:3]</td><td>Last CFIFO to transfer commands through the eQADC SSI. Holds the CFIFO number of last CFIFO to have initiated a command transfer to an external command buffer through the eQADC SSI. LCFTSSI does not indicate the transmission of null messages. LCFTSSI has the following values:
<table>
<tr><th>LCFTSSI[0:3]</th><th>LCFTSSI Meaning</th></tr>
<tr><td>0b0000</td><td>Last command was transferred from CFIFO0</td></tr>
<tr><td>0b0001</td><td>Last command was transferred from CFIFO1</td></tr>
<tr><td>0b0010</td><td>Last command was transferred from CFIFO2</td></tr>
<tr><td>0b0011</td><td>Last command was transferred from CFIFO3</td></tr>
<tr><td>0b0100</td><td>Last command was transferred from CFIFO4</td></tr>
<tr><td>0b0101</td><td>Last command was transferred from CFIFO5</td></tr>
<tr><td>0b0110 – 0b1110</td><td>Reserved</td></tr>
<tr><td>0b1111</td><td>No command was transferred to an external command buffer</td></tr>
</table></td></tr>
<tr><td>21–31</td><td>TC_LCFTSSI[0:10]</td><td>Transfer counter for last CFIFO to transfer commands through eQADC SSI. Indicates the number of commands which have been completely transferred from a particular CFIFO at the time a command transfer from that CFIFO to an external command buffer is initiated. TC_LCFTSSI is a copy of the corresponding TC_CFn in EQADC_CFTCRn (see Section 19.3.2.9) captured at the time a command transfer to an external command buffer is initiated. This field has no meaning when LCFTSSI is 0b1111.</td></tr>
</table>

19.3.2.11 eQADC CFIFO Status Register (EQADC_CFSR)

The EQADC_CFSR contains the current CFIFO status. The EQADC_CFSRs are read only. Writing to the EQADC_CFSR has no effect.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CFS0		CFS1		CFS2		CFS3		CFS4		CFS5		0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0AC															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0AC															

Figure 19-14. eQADC CFIFO Status Register (EQADC_CFSR)

Table 19-17. EQADC_CFSR Field Descriptions

Bits	Name	Description
0–11	CFS*n* [0:1]	CFIFO status. Indicates the current status of CFIFO*n*. Refer to Table 19-18 for more information on CFIFO status.
12–31	—	Reserved.

Table 19-18. Current CFIFO Status

CFIFO Status	Field Value	Explanation
IDLE	0b00	• CFIFO is disabled. • CFIFO is in single-scan edge or level trigger mode and does not have EQADC_FISRn[SSS] asserted. • eQADC completed the transfer of the last entry of the user defined command queue in single-scan mode.
Reserved	0b01	Not applicable.
WAITING FOR TRIGGER	0b10	• CFIFO mode is modified to continuous-scan edge or level trigger mode. • CFIFO mode is modified to single-scan edge or level trigger mode and EQADC_FISRn[SSS] is asserted. • CFIFO mode is modified to single-scan software trigger mode and EQADC_FISRn[SSS] is negated. • CFIFO is paused. • eQADC transferred the last entry of the queue in continuous-scan edge trigger mode.
TRIGGERED	0b11	CFIFO is triggered

19.3.2.12 eQADC SSI Control Register (EQADC_SSICR)

The EQADC_SSICR configures the SSI submodule.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0B4															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	MDT			0	0	0	0	BR			
W																
Reset	0	0	0	0	0	1	1	1	0	0	0	0	1	1	1	1
Reg Addr	Base + 0x0B4															

Figure 19-15. eQADC SSI Control Register (EQADC_SSICR)

Table 19-19. EQADC_SSICR Field Descriptions

Bits	Name	Description
0–20	—	Reserved.
21–23	MDT [0:2]	Minimum delay after transmission. Defines the minimum delay after transmission time (t_{MDT}) expressed in serial clock (FCK) periods. t_{MDT} is the minimum time $\overline{SDS}$ should be kept negated between two consecutive serial transmissions. Table 19-20 lists the minimum delay after transfer time according to how MDT is set. The MDT field must only be written when the serial transmissions from the eQADC SSI are disabled - See EQADC_MCR[ESSIE] field in Section 19.3.2.1.
24–27	—	Reserved.
28–31	BR [0:3]	Baud rate. Selects system clock divide factor as shown in Table 19-21. The baud clock is calculated by dividing the system clock by the clock divide factor specified with the BR field. **Note:** The BR field must only be written when the eQADC SSI is disabled - See EQADC_MCR[ESSIE] field in Section 19.3.2.1.

Table 19-20. Minimum Delay After Transmission (t_{MDT}) Time

MDT	t_{MDT} (FCK period)
0b000	1
0b001	2
0b010	3
0b011	4
0b100	5
0b101	6
0b110	7
0b111	8

Table 19-21. System Clock Divide Factor for Baud Clock

BR[0:3]	System Clock Divide Factor [1]
0b0000	2
0b0001	3
0b0010	4
0b0011	5
0b0100	6
0b0101	7
0b0110	8
0b0111	9
0b1000	10

Table 19-21. System Clock Divide Factor for Baud Clock (continued)

BR[0:3]	System Clock Divide Factor [1]
0b1001	11
0b1010	12
0b1011	13
0b1100	14
0b1101	15
0b1110	16
0b1111	17

1 If the system clock is divided by a odd number then the serial clock will have a duty cycle different from 50%.

19.3.2.13 eQADC SSI Receive Data Register (EQADC_SSIRDR)

The eQADC SSI receive data register (EQADC_SSIRDR) records the last message received from the external device.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RDV	0	0	0	0	0	R_DATA									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
Reg Addr	Base + 0x0B8															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	R_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0B8															

Figure 19-16. eQADC SSI Receive Data Register (EQADC_SSIRDR)

Table 19-22. EQADC_SSIRDR Field Descriptions

Bits	Name	Description
0	RDV	Receive data valid. Indicates if the last received data is valid. This bit is cleared automatically whenever the EQADC_SSIRDR is read. Writes have no effect. 0 Receive data is not valid. 1 Receive data is valid.
1–5	—	Reserved.
6–31	R_DATA [0:25]	eQADC receive DATA. Contains the last result message that was shifted in. Writes to the R_DATA have no effect. Messages that were not completely received due to a transmission abort will not be copied into EQADC_SSIRDR.

19.3.2.14 eQADC CFIFO Registers (EQADC_CF[0–5]R*n*)

EQADC_CF[0–5]R*n* provide visibility of the contents of a CFIFO for debugging purposes. Each CFIFO has four registers that are uniquely mapped to its four 32-bit entries. Refer to Section 19.4.3, "eQADC Command FIFOs," for more information on CFIFOs. These registers are read only. Data written to these registers is ignored.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CFIFO[0–5]_DATA*n*															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	CFIFO0: Base + 0x100 (CF0R0); Base + 0x104 (CF0R1); Base + 0x108 (CF0R2); Base + 0x10C (CF0R3) CFIFO1:Base + 0x140 (CF1R0); Base + 0x144 (CF1R1); Base + 0x148 (CF1R2);Base + 0x14C (CF1R3) CFIFO2: Base + 0x180 (CF2R0); Base + 0x184 (CF2R1); Base + 0x188 (CF2R2); Base + 0x18C (CF2R3) CFIFO3: Base + 0x1C0 (CF3R0); Base + 0x1C4 (CF3R1); Base + 0x1C8 (CF3R2); Base + 0x1CC (CF3R3) CFIFO4: Base + 0x200 (CF4R0); Base + 0x204 (CF4R1); Base + 0x208 (CF4R2); Base + 0x20C (CF4R3) CFIFO5: Base + 0x240 (CF5R0); Base + 0x244 (CF5R1); Base + 0x248 (CF5R2); Base + 0x24C (CF5R3)															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CFIFO[0–5]_DATA*n*															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	CFIFO0: Base + 0x100 (CF0R0); Base + 0x104 (CF0R1); Base + 0x108 (CF0R2); Base + 0x10C (CF0R3) CFIFO1:Base + 0x140 (CF1R0); Base + 0x144 (CF1R1); Base + 0x148 (CF1R2);Base + 0x14C (CF1R3) CFIFO2: Base + 0x180 (CF2R0); Base + 0x184 (CF2R1); Base + 0x188 (CF2R2); Base + 0x18C (CF2R3) CFIFO3: Base + 0x1C0 (CF3R0); Base + 0x1C4 (CF3R1); Base + 0x1C8 (CF3R2); Base + 0x1CC (CF3R3) CFIFO4: Base + 0x200 (CF4R0); Base + 0x204 (CF4R1); Base + 0x208 (CF4R2); Base + 0x20C (CF4R3) CFIFO5: Base + 0x240 (CF5R0); Base + 0x244 (CF5R1); Base + 0x248 (CF5R2); Base + 0x24C (CF5R3)															

Figure 19-17. eQADC CFIF0[0–5] Registers (EQADC_CF[0–5]R*n*)

Table 19-23. EQADC_CF[0–5]R*n* Field Descriptions

Bits	Name	Description
0–31	CFIFO[0–5]_DATA*n* [0:31]	CFIFO[0–5]_data*n*. Returns the value stored within the entry of CFIFO[0–5]. Each CFIFO is composed of four 32-bit entries, with register 0 being mapped to the entry with the smallest memory mapped address.

19.3.2.15 eQADC RFIFO Registers (EQADC_RF[0–5]R*n*)

EQADC_RF[0–5]R*n* provide visibility of the contents of a RFIFO for debugging purposes. Each RFIFO has four registers which are uniquely mapped to its four 16-bit entries. Refer to Section 19.4.4, "Result FIFOs," for more information on RFIFOs. These registers are read only. Data written to these registers is ignored.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	RFIFO0: Base + 0x300 (RF0R0); Base + 0x304 (RF0R1); Base + 0x308 (RF0R2); Base+0x30C (RF0R3) RFIFO1: Base + 0x340 (RF1R0); Base + 0x344 (RF1R1); Base + 0x348 (RF1R2); Base + 0x34C (RF1R3) RFIFO2: Base + 0x380 (RF2R0); Base + 0x384 (RF2R1); Base + 0x388 (RF2R2); Base + 0x38C (RF2R3) RFIFO3: Base + 0x3C0 (RF3R0); Base + 0x3C4 (RF3R1); Base + 0x3C8 (RF3R2); Base + 0x3CC (RF3R3) RFIFO4: Base + 0x400 (RF4R0); Base + 0x404 (RF4R1); Base + 0x408 (RF4R2); Base + 0x40C (RF4R3) RFIFO5: Base + 0x440 (RF5R0); Base + 0x444 (RF5R1); Base + 0x448 (RF5R2); Base + 0x44C (RF5R3)															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RFIFO[0–5]_DATA*n*															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	RFIFO0: Base + 0x300 (RF0R0); Base + 0x304 (RF0R1); Base + 0x308 (RF0R2); Base+0x30C (RF0R3) RFIFO1: Base + 0x340 (RF1R0); Base + 0x344 (RF1R1); Base + 0x348 (RF1R2); Base + 0x34C (RF1R3) RFIFO2: Base + 0x380 (RF2R0); Base + 0x384 (RF2R1); Base + 0x388 (RF2R2); Base + 0x38C (RF2R3) RFIFO3: Base + 0x3C0 (RF3R0); Base + 0x3C4 (RF3R1); Base + 0x3C8 (RF3R2); Base + 0x3CC (RF3R3) RFIFO4: Base + 0x400 (RF4R0); Base + 0x404 (RF4R1); Base + 0x408 (RF4R2); Base + 0x40C (RF4R3) RFIFO5: Base + 0x440 (RF5R0); Base + 0x444 (RF5R1); Base + 0x448 (RF5R2); Base + 0x44C (RF5R3)															

Figure 19-18. eQADC RFIFO*n* Registers (EQADC_RF[0–5]R*n*)

Table 19-24. EQADC_RF[0–5]R*n* Field Descriptions

Bits	Name	Description
0–31	RFIFO[0–5]_DATA*n* [0:15]	RFIFO[0–5] data *n*. Returns the value stored within the entry of RFIFO[0–5]. Each RFIFO is composed of four 16-bit entries, with register 0 being mapped to the entry with the smallest memory mapped address.

19.3.3 On-Chip ADC Registers

This section describes a list of registers that control on-chip ADC operation. The ADC registers are not part of the CPU accessible memory map. These registers can only be accessed indirectly through configuration commands. There are five non memory mapped registers per ADC, five for ADC0 and five for ADC1. The address, usage, and access privilege of each register is shown in Table 19-25 and Table 19-26. Data written to or read from reserved areas of the memory map is undefined.

Their assigned addresses are the values used to set the ADC_REG_ADDRESS field of the read/write configuration commands bound for the on-chip ADCs. These are half-word addresses. Further, the following restrictions apply when accessing these registers:

- Registers ADC0_CR, ADC0_GCCR, and ADC0_OCCR can only be accessed by configuration commands sent to the ADC0 command buffer.
- Registers ADC1_CR, ADC1_GCCR, and ADC1_OCCR can only be accessed by configuration commands sent to the ADC1 command buffer.
- Registers ADC_TSCR and ADC_TBCR can be accessed by configuration commands sent to the ADC0 command buffer or to the ADC1 command buffer. A data write to ADC_TSCR through a configuration command sent to the ADC0 command buffer will write the same memory location

as when writing to it through a configuration command sent to the ADC1 command buffer. The same is valid for ADC_TBCR.

NOTE

Simultaneous write accesses from the ADC0 and ADC1 command buffers to ADC_TSCR or to ADC_TBCR are not allowed.

Table 19-25. ADC0 Registers

ADC0 Register Address	Use	Access
0x00	ADC0 Address 0x00 is used for conversion command messages.	
0x01	ADC0 Control Register (ADC0_CR)	Write/Read
0x02	ADC Time Stamp Control Register (ADC_TSCR) [1]	Write/Read
0x03	ADC Time Base Counter Register (ADC_TBCR) [1]	Write/Read
0x04	ADC0 Gain Calibration Constant Register (ADC0_GCCR)	Write/Read
0x05	ADC0 Offset Calibration Constant Register (ADC0_OCCR)	Write/Read
0x06–0xFF	Reserved	—

[1] This register is also accessible by configuration commands sent to the ADC1 command buffer.

Table 19-26. ADC1 Registers

ADC1 Register Address	Use	Access
0x00	ADC1 Address 0x00 is used for conversion command messages.	
0x01	ADC1 Control Register (ADC1_CR)	Write/Read
0x02	ADC Time Stamp Control Register (ADC_TSCR) [1]	Write/Read
0x03	ADC Time Base Counter Register (ADC_TBCR) [1]	Write/Read
0x04	ADC1 Gain Calibration Constant Register (ADC1_GCCR)	Write/Read
0x05	ADC1 Offset Calibration Constant Register (ADC1_OCCR)	Write/Read
0x06–0xFF	Reserved	—

[1] This register is also accessible by configuration commands sent to the ADC0 command buffer.

19.3.3.1 ADC*n* Control Registers (ADC0_CR and ADC1_CR)

The ADC*n* control registers (ADC*n*_CR) are used to configure the on-chip ADCs.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ADC0_EN	0	0	0	ADC0_EMUX	0	0	0	0	0	0	ADC0_CLK_PS				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
Reg Addr	0x01															

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ADC1_EN	0	0	0	ADC1_EMUX	0	0	0	0	0	0	ADC1_CLK_PS				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
Reg Addr	0x01															

Figure 19-19. ADC*n* Control Registers (ADC0_CR and ADC1_CR)

Table 19-27. ADC*n*_CR Field Descriptions

Bits	Name	Description
0	ADC*n*_EN	ADC*n* enable. Enables ADC*n* to perform A/D conversions. Refer to Section 19.4.5.1, "Enabling and Disabling the on-chip ADCs," for details. 0 ADC is disabled. Clock supply to ADC0/1 is stopped. 1 ADC is enabled and ready to perform A/D conversions. **Note:** The bias generator circuit inside the ADC ceases functioning when both ADC0_EN and ADC1_EN bits are negated. **Note:** Conversion commands sent to a disabled ADC are ignored by the ADC control hardware. **Note:** When the ADC*n*_EN status is changed from asserted to negated, the ADC clock will not stop until it reaches its low phase.
1–3	—	Reserved.
4	ADC*n*_EMUX	ADC*n* external multiplexer enable. When ADC*n*_EMUX is asserted, the MA pins will output digital values according to the number of the external channel being converted for selecting external multiplexer inputs. Refer to Section 19.4.6, "Internal/External Multiplexing," for a detailed description about how ADC*n*_EMUX affects channel number decoding. 0 External multiplexer disabled; no external multiplexer channels can be selected. 1 External multiplexer enabled; external multiplexer channels can be selected. **Note:** Both ADC*n*_EMUX bits must not be asserted at the same time. **Note:** The ADC*n*_EMUX bit must only be written when the ADC*n*_EN bit is negated. ADC*n*_EMUX can be set during the same write cycle used to set ADC*n*_EN.
5–10	—	Reserved.
11–15	ADC*n*_CLK_PS [0:4]	ADC*n* clock prescaler. The ADC*n*_CLK_PS field controls the system clock divide factor for the ADC*n* clock as in Table 19-28. See Section 19.4.5.2, "ADC Clock and Conversion Speed," for details about how to set ADC0/1_CLK_PS. The ADC*n*_CLK_PS field must only be written when the ADC*n*_EN bit is negated. This field can be configured during the same write cycle used to set ADC*n*_EN.

Table 19-28. System Clock Divide Factor for ADC Clock

ADC*n*_CLK_PS[0:4]	System Clock Divide Factor
0b00000	2
0b00001	4
0b00010	6
0b00011	8
0b00100	10
0b00101	12
0b00110	14
0b00111	16
0b01000	18
0b01001	20
0b01010	22
0b01011	24
0b01100	26
0b01101	28
0b01110	30
0b01111	32
0b10000	34
0b10001	36
0b10010	38
0b10011	40
0b10100	42
0b10101	44
0b10110	46
0b10111	48
0b11000	50
0b11001	52
0b11010	54
0b11011	56
0b11100	58
0b11101	60
0b11110	62
0b11111	64

19.3.3.2 ADC Time Stamp Control Register (ADC_TSCR)

The ADC_TSCR contains a system clock divide factor used in the making of the time base counter clock. It determines at what frequency the time base counter will run. ADC_TSCR can be accessed by configuration commands sent to ADC0 or to ADC1. A data write to ADC_TSCR through a configuration command sent to ADC0 will write the same memory location as when writing to it through a configuration command sent to ADC1.

NOTE

Simultaneous write accesses from ADC0 and ADC1 to ADC_TSCR are not allowed.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	TBC_CLK_PS			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	0x02															

Figure 19-20. ADC Time Stamp Control Register (ADC_TSCR)

Table 19-29. ADC_TSCR Field Descriptions

Bits	Name	Description
0–11	—	Reserved.
12–15	TBC_CLK_PS [0:3]	Time base counter clock prescaler. Contains the system clock divide factor for the time base counter. It controls the accuracy of the time stamp. The prescaler is disabled when TBC_CLK_PS is set to 0b0000.

Table 19-30. Clock Divide Factor for Time Stamp

TBC_CLK_PS[0:3]	System Clock Divide Factor	Clock to Time Stamp Counter for a 120 MHz System Clock (MHz)
0b0000	Disabled	Disabled
0b0001	1	120
0b0010	2	60
0b0011	4	30
0b0100	6	20
0b0101	8	15
0b0110	10	12
0b0111	12	10
0b1000	16	7.5
0b1001	32	3.75
0b1010	64	1.88

Table 19-30. Clock Divide Factor for Time Stamp (continued)

TBC_CLK_PS[0:3]	System Clock Divide Factor	Clock to Time Stamp Counter for a 120 MHz System Clock (MHz)
0b1011	128	0.94
0b1100	256	0.47
0b1101	512	0.23
0b1110 - 0b1111	Reserved	—

NOTE

If TBC_CLK_PS is not set to disabled, it must not be changed to any other value besides disabled. If TBC_CLK_PS is set to disabled it can be changed to any other value.

19.3.3.3 ADC Time Base Counter Registers (ADC_TBCR)

The ADC_TBCR contains the current value of the time base counter. ADC_TBCR can be accessed by configuration commands sent to ADC0 or to ADC1. A data write to ADC_TBCR through a configuration command sent to ADC0 will write the same memory location as when writing to it through a configuration command sent to ADC1.

NOTE

Simultaneous write accesses from ADC0 and ADC1 to ADC_TBCR are not allowed.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TBC_VALUE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	0x03															

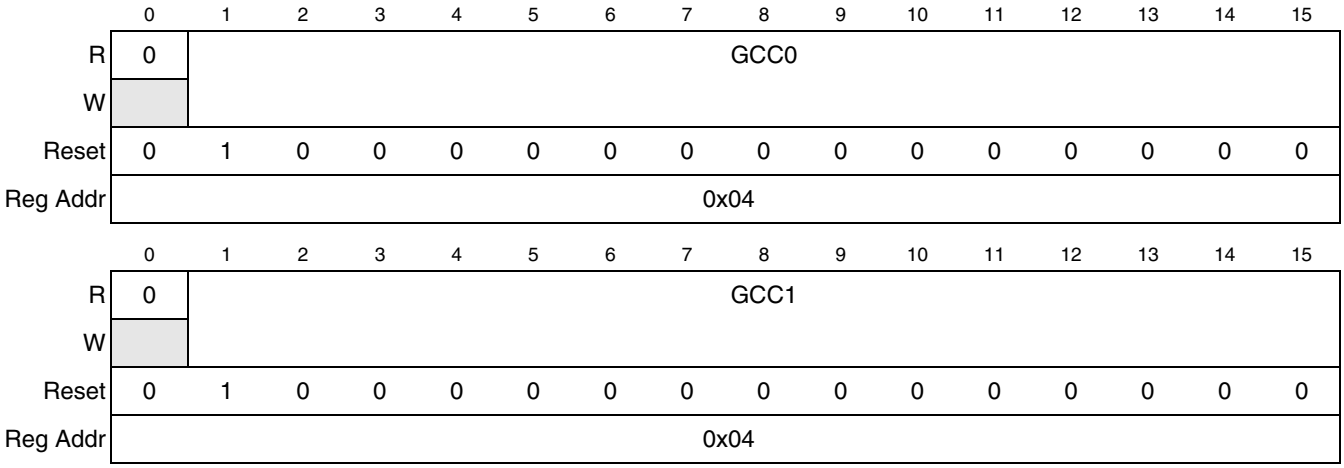
Figure 19-21. ADC Time Base Counter Register (ADC_TBCR)

Table 19-31. ADC_TBCR Field Descriptions

Bits	Name	Description
0–15	TBC_VALUE [0:15]	Time base counter VALUE. Contains the current value of the time base counter. Reading TBC_VALUE returns the current value of time base counter. Writes to TBC_VALUE register load the written data to the counter. The time base counter counts from 0x0000 to 0xFFFF and wraps when reaching 0xFFFF.

19.3.3.4 ADC*n* Gain Calibration Constant Registers (ADC0_GCCR and ADC1_GCCR)

The ADC*n*_GCCR contains the gain calibration constant used to fine-tune the ADC*n* conversion results. Refer to Section 19.4.5.4, "ADC Calibration Feature," for details about the calibration scheme used in the eQADC.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	GCC0														
W																
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	0x04															

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	GCC1														
W																
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	0x04															

Figure 19-22. ADC*n* Gain Calibration Constant Registers (ADC*n*_GCCR)

Table 19-32. ADC*n*_GCCR Field Descriptions

Bits	Name	Description
0	—	Reserved.
1–15	GCC*n* [0:14]	ADC*n* gain calibration constant. Contains the gain calibration constant used to fine-tune ADC*n* conversion results. It is a unsigned 15-bit fixed pointed value. The gain calibration constant is an unsigned fixed point number expressed in the *GCC_INT.GCC_FRAC* binary format. The integer part of the gain constant (GCC_INT) contains a single binary digit while its fractional part (GCC_FRAC) contains 14 digits. For details about the GCC data format refer to Section 19.4.5.4.2, "MAC Unit and Operand Data Format."

19.3.3.5 ADC*n* Offset Calibration Constant Registers (ADC0_OCCR and ADC1_OCCR)

The ADC*n*_OCCR contains the offset calibration constant used to fine-tune of ADC0/1 conversion results. The offset constant is a signed 14-bit integer value. Refer to Section 19.4.5.4, "ADC Calibration Feature," for details about the calibration scheme used in the eQADC.

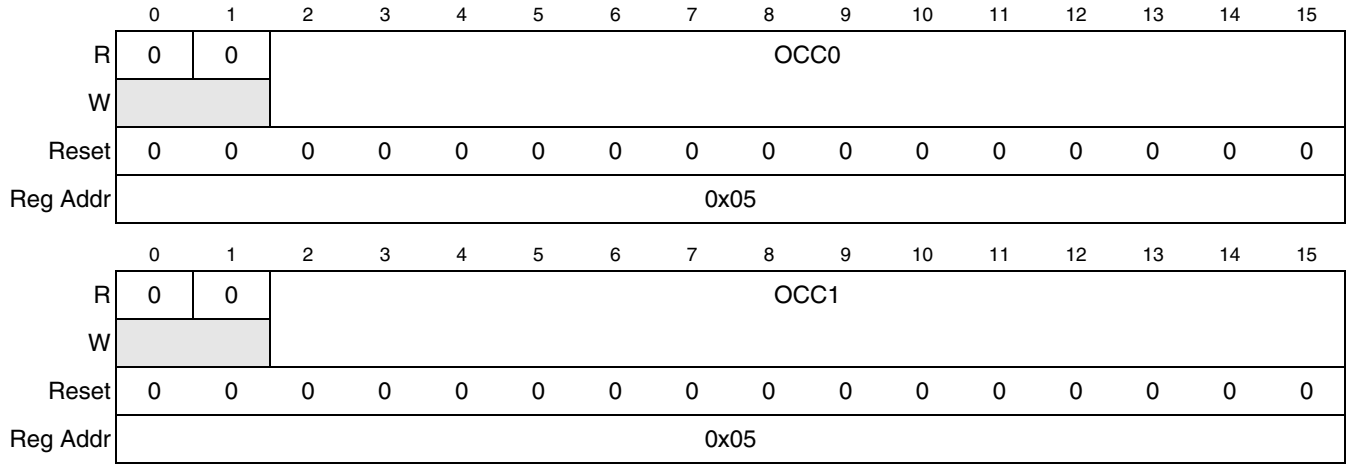


Figure 19-23. ADC*n* Offset Calibration Constant Registers (ADC*n*_OCCR)

Table 19-33. ADC*n*_OCCR Field Descriptions

Bits	Name	Description
0–1	—	Reserved.
2–15	OCC*n* [0:13]	ADC*n* offset calibration constant. Contains the offset calibration constant used to fine-tune ADC*n* conversion results. Negative values should be expressed using the two's complement representation.

19.4 Functional Description

The eQADC provides a parallel interface to two on-chip ADCs, and a single master to single slave serial interface to an off-chip external device. The two on-chip ADCs are architected to allow access to all the analog channels.

Initially, command data is contained in system memory in a user defined data queue structure. Command data is moved between the user-defined queues and CFIFOs by the host CPU or by the eDMA which responds to interrupt and eDMA requests generated by the eQADC. The eQADC supports software and hardware triggers from other modules or external pins to initiate transfers of commands from the multiple CFIFOs to the on-chip ADCs or to the external device.

CFIFOs can be configured to be in single-scan or continuous-scan mode. When a CFIFO is configured to be in single-scan mode, the eQADC scans the user-defined command queue one time. The eQADC stops transferring commands from the triggered CFIFO after detecting the EOQ bit set in the last transfer. After an EOQ bit is detected, software involvement is required to rearm the CFIFO so that it can detect new trigger events.

When a CFIFO is configured for continuous-scan mode, the whole user command queue is scanned multiple times. After the detection of an asserted EOQ bit in the last command transfer, command transfers can continue or not depending on the mode of operation of the CFIFO.

The eQADC can also in parallel and independently of the CFIFOs receive data from the on-chip ADCs or from off-chip external device into multiple RFIFOs. Result data is moved from the RFIFOs to the user-defined result queues in system memory by the host CPU or by the eDMA.

19.4.1 Data Flow in the eQADC

Figure 19-24 shows how command data flows inside the eQADC system. A command message is the predefined format in which command data is stored in the user-defined command queues. A command message has 32 bits and is composed of two parts: a CFIFO header and an ADC command. Command messages are moved from the user command queues to the CFIFOs by the host CPU or by the eDMA as they respond to interrupt and eDMA requests generated by the eQADC. The eQADC generates these requests whenever a CFIFO is not full. The FIFO control unit will only transfer the command part of the command message to the selected ADC. Information in the CFIFO header together with the upper bit of the ADC command is used by the FIFO control unit to arbitrate which triggered CFIFO will be transferring the next command. Because command transfer through the serial interface can take significantly more time than a parallel transfer to the on-chip ADCs, command transfers for on-chip ADCs occur concurrently with the transfers through the serial interface. Commands sent to the ADCs are executed in a first-in-first-out (FIFO) basis and three types of results can be expected: data read from an ADC register, a conversion result, or a time stamp. The order at which ADC commands sent to the external device are executed, and the type of results that can be expected depends on the architecture of that device with the exception of unsolicited data like null messages for example.

NOTE

While the eQADC pops commands out from a CFIFO, it also is checking the number of entries in the CFIFO and generating requests to fill it. The process of pushing and popping commands to and from a CFIFO can occur simultaneously.

The FIFO control unit expects all incoming results to be shaped in a pre-defined result message format. Figure 19-25 shows how result data flows inside the eQADC system. Results generated on the on-chip ADCs are formatted into result messages inside the result format and calibration submodule. Results returning from the external device are already formatted into result messages and therefore bypass the result format and calibration submodule located inside the eQADC. A result message is composed of an RFIFO header and an ADC Result. The FIFO control unit decodes the information contained in the RFIFO header to determine the RFIFO to which the ADC result should be sent. Once in an RFIFO, the ADC result is moved to the corresponding user result queue by the host CPU or by the eDMA as they respond to interrupt and eDMA requests generated by the eQADC. The eQADC generates these requests whenever an RFIFO has at least one entry.

NOTE

While conversion results are returned, the eQADC is checking the number of entries in the RFIFO and generating requests to empty it. The process of pushing and popping ADC results to and from an RFIFO can occur simultaneously.

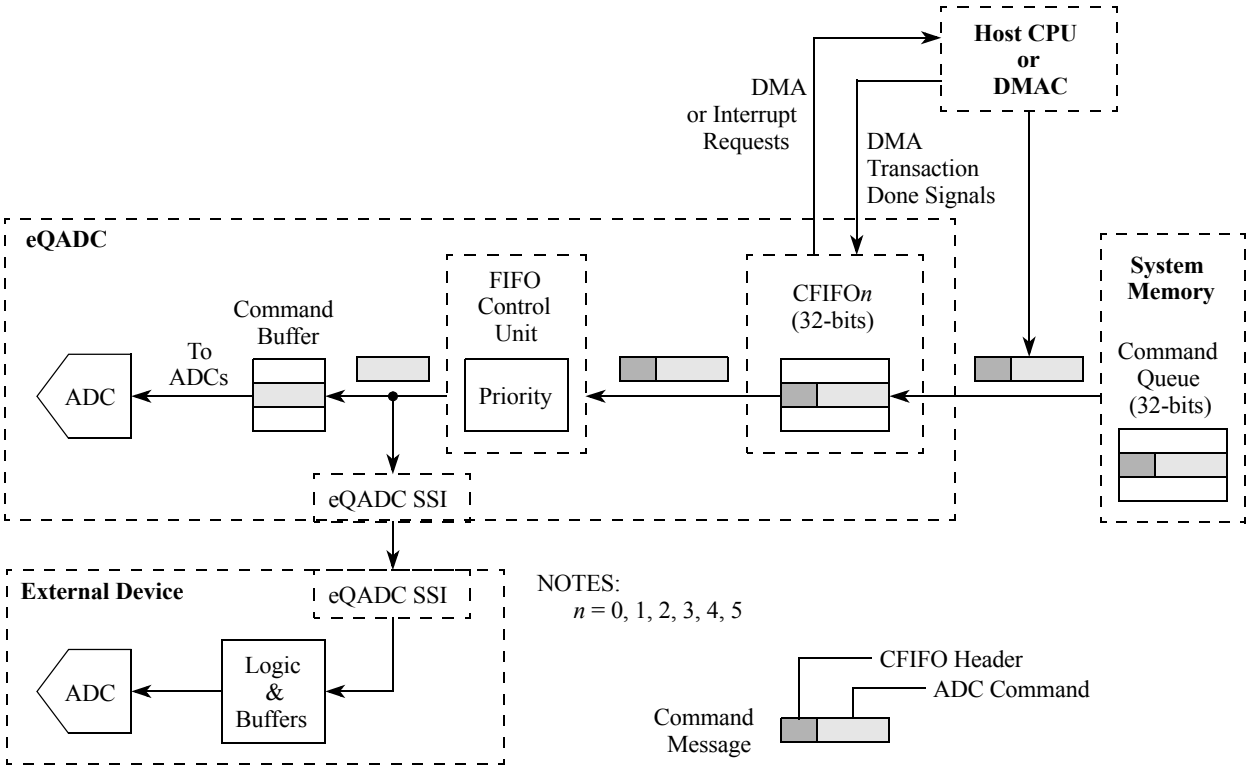


Figure 19-24. Command Flow During eQADC Operation

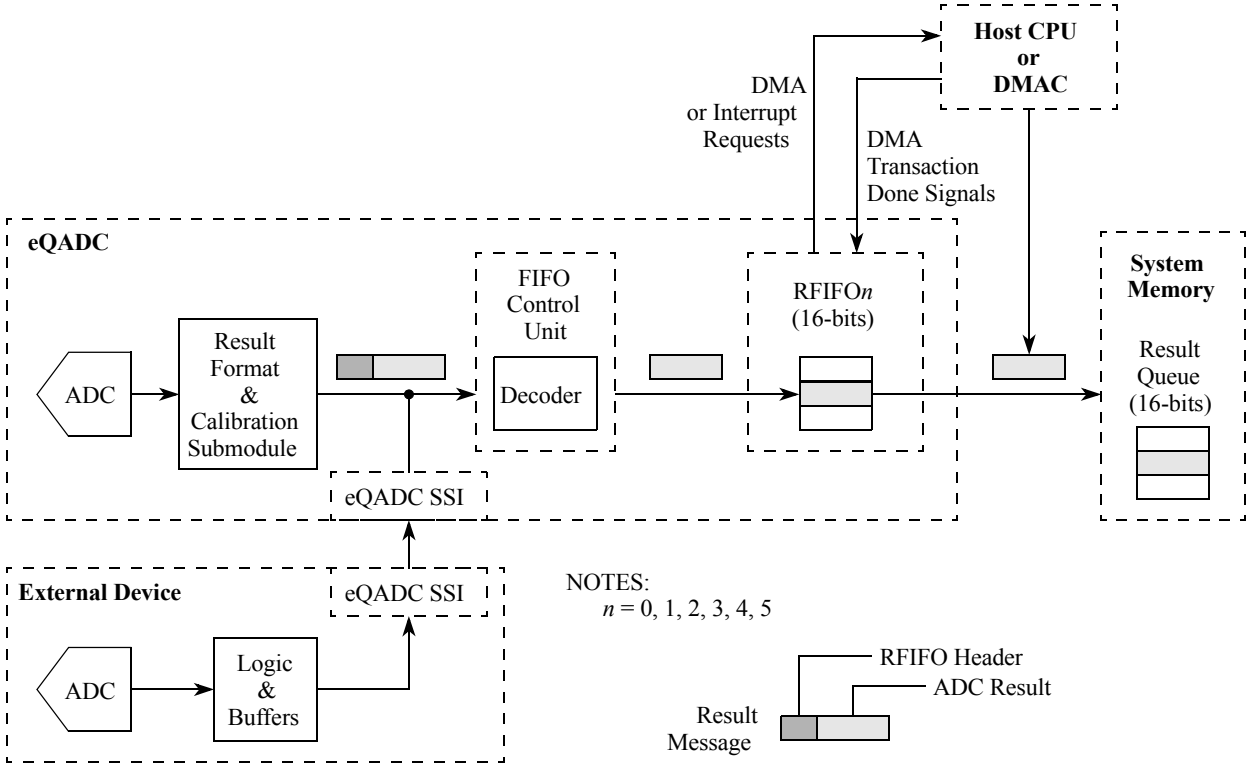


Figure 19-25. Result Flow During eQADC Operation

19.4.1.1 Assumptions/Requirements Regarding the External Device

The external device exchanges command and result data with the eQADC through the eQADC SSI interface. This section explains the minimum requirements an external device has to meet to properly interface with the eQADC. Some assumptions about the architecture of the external device are also described.

19.4.1.1.1 eQADC SSI Protocol Support

The external device must fully support the eQADC SSI protocol as specified in Section 19.4.8, "eQADC Synchronous Serial Interface (SSI) Submodule," section of this document. Support for the abort feature is optional. When aborts are not supported, all command messages bound for an external command buffer must have the ABORT_ST bit negated - see Section , " Command Message Format for External Device Operation."

19.4.1.1.2 Number of Command Buffers and Result Buffers

The external device should have a minimum of one and a maximum of two command buffers to store command data sent from the eQADC. If more than two command buffers are implemented in the external device, they are not recognized by the eQADC as valid destinations for commands. In this document, the two valid external command buffers are referred to as command buffer 2 and command buffer 3 (the two on-chip ADCs being command buffer 0 and 1). The external device decides to which external command buffer a command should go by decoding the upper bit (BN bit) of the ADC command - see Section , " Command Message Format for External Device Operation." An external device that only implements one command buffer can ignore the BN bit.

The limit of two command buffers does not limit the number of result buffers in the slave device.

19.4.1.1.3 Command Execution and Result Return

Commands sent to a specific external command buffer should be executed in the order they were received.

Results generated by the execution of commands in an external command buffer should be returned in the order that the command buffer received these commands.

19.4.1.1.4 Null and Result Messages

The external device must be capable of correctly processing null messages as specified in the Section 19.3.2.2, "eQADC Null Message Send Format Register (EQADC_NMSFR)."

In case no valid result data is available to be sent to the eQADC, the external device must send data in the format specified in Section , " Null Message Format for External Device Operation."

In case valid result data is available to sent to the eQADC, the external device must send data in the format specified in Section , " Result Message Format for External Device Operation."

The BUSY0/1 fields of all messages sent from the external device to the eQADC must be correctly encoded according to the latest information on the fullness state of the command buffers. For example, if external command buffer 2 is empty before the end of the current serial transmission and if at the end of this transmission the external device receives a command to command buffer 2, then the BUSY0 field, that is to be sent to the eQADC on the next serial transmission, should be encoded assuming that the external command buffer has one entry.

19.4.1.2 Message Format in eQADC

This section explains the command and result message formats used for on-chip ADC operation and for external device operation.

A command message is the pre-defined format at which command data is stored in the user command queues. A command message has 32 bits and is composed of two parts: a CFIFO header and an ADC command. The size of the CFIFO header is fixed to 6 bits, and it works as inputs to the FIFO control unit. The header controls when a command queue ends, when it pauses, if commands are sent to internal or external buffers, and if it can abort a serial data transmission. Information contained in the CFIFO header, together with the upper bit of the ADC command, is used by the FIFO control unit to arbitrate which triggered CFIFO will transfer the next command. ADC commands are encoded inside the least significant 26 bits of the command message.

A result message is composed of an RFIFO header and an ADC result. The FIFO control unit decodes the information contained in the RFIFO header to determine the RFIFO to which the ADC result should be sent. An ADC result is always 16 bits long.

19.4.1.2.1 Message Formats for On-Chip ADC Operation

This section describes the command/result message formats used for on-chip ADC operation.

NOTE

Although this subsection describes how the command and result messages are formatted to communicate with the on-chip ADCs, nothing prevents the programmer from using a different format when communicating with an external device through the serial interface. Refer to Section 19.4.1.2.2, "Message Formats for External Device Operation." Apart from the BN bit, the ADC command of a command message can be formatted to communicate to an arbitrary external device provided that the device returns an RFIFO header in the format expected by the eQADC. When the FIFO control unit receives return data message, it decodes the message tag field and stores the 16-bit data into the corresponding RFIFO.

Conversion Command Message Format for On-Chip ADC Operation

Figure 19-26 describes the command message format for conversion commands when interfacing with the on-chip ADCs. A conversion result is always returned for conversion commands and time stamp information can be optionally requested. The lower byte of conversion commands is always set to 0 to distinguish it from configuration commands.

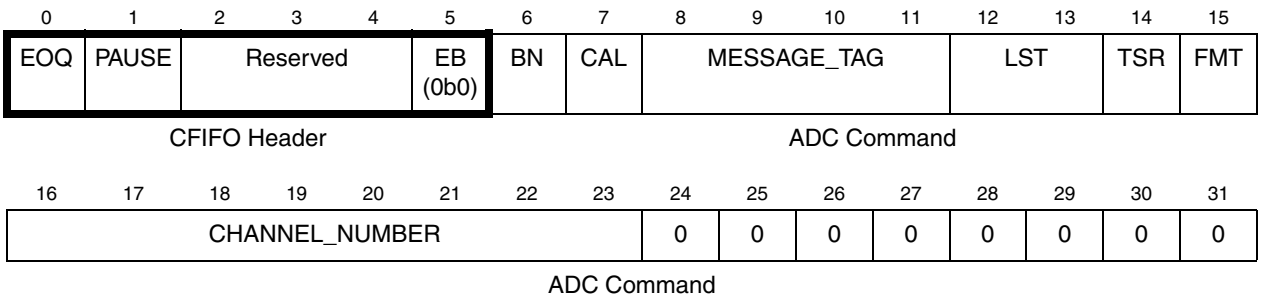


Figure 19-26. Conversion Command Message Format for On-Chip ADC Operation

Table 19-34. On-Chip ADC Field Descriptions: Conversion Command Message Format

Bits	Name	Description
0	EOQ	End-of-queue. Asserted in the last command of a command queue to indicate to the eQADC that a scan of the queue is completed. EOQ instructs the eQADC to reset its current CFIFO transfer counter value (TC_CF) to 0. Depending on the CFIFO mode of operation, the CFIFO status will also change upon the detection of an asserted EOQ bit on the last transferred command. See Section 19.4.3.5, "CFIFO Scan Trigger Modes," for details. 0 Not the last entry of the command queue. 1 Last entry of the command queue. **Note:** If both the pause and EOQ bits are asserted in the same command message the respective flags are set, but the CFIFO status changes as if only the EOQ bit were asserted.
1	PAUSE	Pause. Allows software to create sub-queues within a command queue. When the eQADC completes the transfer of a command with an asserted pause bit, the CFIFO enters the WAITING FOR TRIGGER state. Refer to Section 19.4.3.6.1, "CFIFO Operation Status," for a description of the state transitions. The pause bit is only valid when CFIFO operation mode is configured to single or continuous-scan edge trigger mode. 0 Do not enter WAITING FOR TRIGGER state after transfer of the current command message. 1 Enter WAITING FOR TRIGGER state after transfer of the current command message. **Note:** If both the pause and EOQ bits are asserted in the same command message the respective flags are set, but the CFIFO status changes as if only the EOQ bit were asserted.
2–4	—	Reserved.
5	EB	External buffer bit. A negated EB bit indicates that the command is sent to an on chip ADC. 0 Command is sent to an internal buffer. 1 Command is sent to an external buffer.
6	BN	Buffer number. Indicates which ADC the message will be sent to. ADCs 1 and 0 can either be internal or external depending on the EB bit setting. 0 Message sent to ADC 0. 1 Message sent to ADC 1.
7	CAL	Calibration. Indicates if the returning conversion result must be calibrated. 0 Do not calibrate conversion result. 1 Calibrate conversion result.

Table 19-34. On-Chip ADC Field Descriptions: Conversion Command Message Format (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>8–11</td><td>MESSAGE_TAG [0:3]</td><td>MESSAGE_TAG field. Allows the eQADC to separate returning results into different RFIFOs. When the eQADC transfers a command, the MESSAGE_TAG is included as part of the command. Eventually the external device/on-chip ADC returns the result with the same MESSAGE_TAG. The eQADC separates incoming messages into different RFIFOs by decoding the MESSAGE_TAG of the incoming data.
<table>
<tr><th>MESSAGE_TAG[0:3]</th><th>MESSAGE_TAG Meaning</th></tr>
<tr><td>0b0000</td><td>Result is sent to RFIFO 0</td></tr>
<tr><td>0b0001</td><td>Result is sent to RFIFO 1</td></tr>
<tr><td>0b0010</td><td>Result is sent to RFIFO 2</td></tr>
<tr><td>0b0011</td><td>Result is sent to RFIFO 3</td></tr>
<tr><td>0b0100</td><td>Result is sent to RFIFO 4</td></tr>
<tr><td>0b0101</td><td>Result is sent to RFIFO 5</td></tr>
<tr><td>0b0110–0b0111</td><td>Reserved</td></tr>
<tr><td>0b1000</td><td>Null message received</td></tr>
<tr><td>0b1001</td><td>Reserved for customer use. [1]</td></tr>
<tr><td>0b1010</td><td>Reserved for customer use. [1]</td></tr>
<tr><td>0b1011–0b1111</td><td>Reserved</td></tr>
</table>
[1] These messages are treated as null messages. Therefore, they must obey the format for incoming null messages and return valid BUSY0/1 fields. Refer to Section , “ Null Message Format for External Device Operation.”</td></tr>
<tr><td>12–13</td><td>LST [0:1]</td><td>Long sampling time. These two bits determine the duration of the sampling time in ADC clock cycles.
Note: For external mux mode, 64 or 128 sampling cycles is recommended.
<table>
<tr><th>LST[0:1]</th><th>Sampling cycles (ADC Clock Cycles)</th></tr>
<tr><td>0b00</td><td>2</td></tr>
<tr><td>0b01</td><td>8</td></tr>
<tr><td>0b10</td><td>64</td></tr>
<tr><td>0b11</td><td>128</td></tr>
</table></td></tr>
<tr><td>14</td><td>TSR</td><td>Time stamp request. TSR indicates the request for a time stamp. When TSR is asserted, the on-chip ADC control logic returns a time stamp for the current conversion command after the conversion result is sent to the RFIFOs. See Section 19.4.5.3, “Time Stamp Feature,” for details.
0 Return conversion result only.
1 Return conversion time stamp after the conversion result.</td></tr>
</table>

Table 19-34. On-Chip ADC Field Descriptions: Conversion Command Message Format (continued)

Bits	Name	Description
15	FMT	Conversion data format. FMT specifies to the eQADC how to format the 12-bit conversion data returned by the ADCs into the 16-bit format which is sent to the RFIFOs. See Section , " ADC Result Format for On-Chip ADC Operation," for details. 0 Right justified unsigned. 1 Right justified signed.
16–23	CHANNEL_ NUMBER [0:7]	Channel number. Selects the analog input channel. The software programs this field with the channel number corresponding to the analog input pin to be sampled and converted. See Section 19.4.6.1, "Channel Assignment," for details.
24–31	—	Reserved.

Write Configuration Command Message Format for On-Chip ADC Operation

Figure 19-27 describes the command message format for a write configuration command when interfacing with the on-chip ADCs. A write configuration command is used to set the control registers of the on-chip ADCs. No conversion data will be returned for a write configuration command. Write configuration commands are differentiated from read configuration commands by a negated R/W bit.

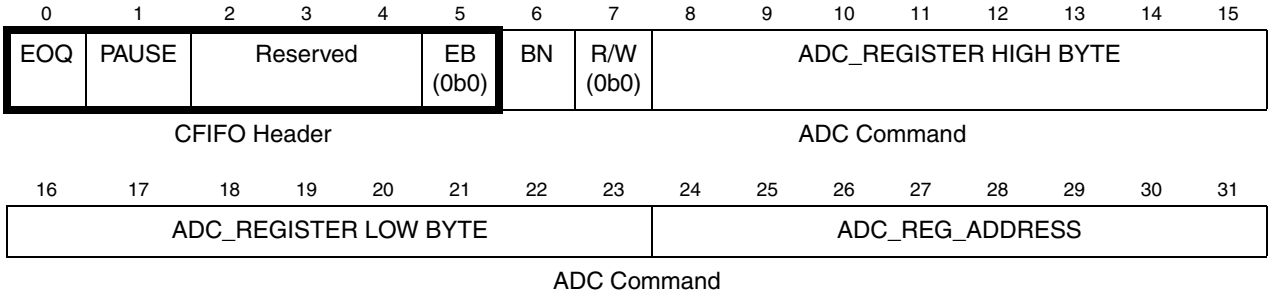


Figure 19-27. Write Configuration Command Message Format for On-chip ADC Operation

Table 19-35. On-Chip ADC Field Descriptions: Write Configuration

Bits	Name	Description
0	EOQ	End-of-queue. Asserted in the last command of a command queue to indicate to the eQADC that a scan of the queue is completed. EOQ instructs the eQADC to reset its current CFIFO transfer counter value (TC_CF) to 0. Depending on the CFIFO mode of operation, the CFIFO status will also change upon the detection of an asserted EOQ bit on the last transferred command. See Section 19.4.3.5, "CFIFO Scan Trigger Modes," for details. 0 Not the last entry of the command queue. 1 Last entry of the command queue. **Note:** If both the pause and EOQ bits are asserted in the same command message the respective flags are set, but the CFIFO status changes as if only the EOQ bit were asserted.
1	PAUSE	Pause bit. Allows software to create sub-queues within a command queue. When the eQADC completes the transfer of a command with an asserted pause bit, the CFIFO enters the WAITING FOR TRIGGER state. Refer to Section 19.4.3.6.1, "CFIFO Operation Status," for a description of the state transitions. The pause bit is only valid when CFIFO operation mode is configured to single or continuous-scan edge trigger mode. 0 Do not enter WAITING FOR TRIGGER state after transfer of the current command message. 1 Enter WAITING FOR TRIGGER state after transfer of the current command message. **Note:** If both the pause and EOQ bits are asserted in the same command message, the respective flags are set, but the CFIFO status changes as if only the EOQ bit were asserted.
2–4	—	Reserved.
5	EB	External buffer bit. This bit should always be cleared for messages sent to an on-chip ADC. 0 Command is sent to an internal command buffer. 1 Command is sent to an external command buffer.
6	BN	Buffer number. Indicates which buffer the message will be stored in. Buffers 1 and 0 can either be internal or external depending on the EB bit setting. 0 Message stored in buffer 0. 1 Message stored in buffer 1.
7	R/W	Read/write. A negated R/W indicates a write configuration command. 0 Write 1 Read
8–15	ADC_ REGISTER_ HIGH_BYTE [0:7]	ADC register high byte. The value to be written into the most significant 8 bits of control/configuration register when the R/W bit is negated.
16–23	ADC_ REGISTER_ LOW_BYTE [0:7]	ADC register low byte. The value to be written into the least significant 8 bits of a control/configuration register when the R/W bit is negated.
24–31	ADC_REG_ ADDRESS [0:7]	ADC register address. Selects a register on the ADC register set to be written or read. Only half-word addresses can be used. See Table 19-25.

Read Configuration Command Message Format for On-Chip ADC Operation

Figure 19-28 describes the command message format for a read configuration command when interfacing with the on-chip ADCs. A read configuration command is used to read the contents of the on-chip ADC registers which are only accessible via command messages. Read configuration commands are differentiated from write configuration commands by an asserted R/W bit.

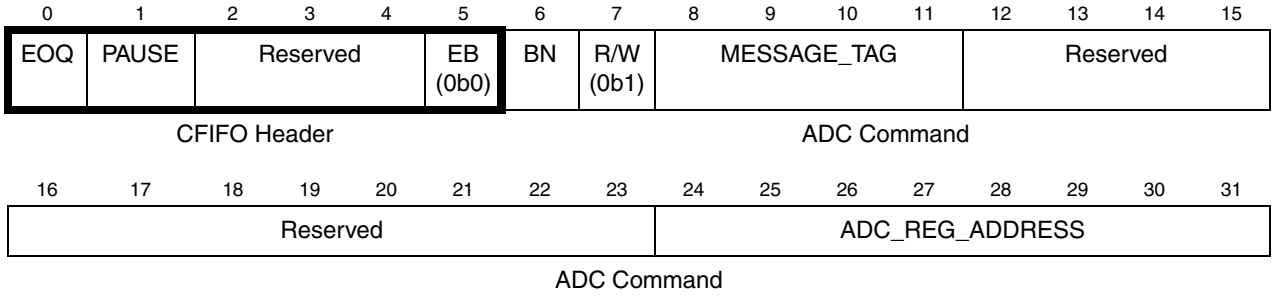


Figure 19-28. Read Configuration Command Message Format for On-Chip ADC Operation

Table 19-36. On-Chip ADC Field Descriptions: Read Configuration

Bits	Name	Description
0	EOQ	End-of-queue. Asserted in the last command of a command queue to indicate to the eQADC that a scan of the queue is completed. EOQ instructs the eQADC to reset its current CFIFO transfer counter value (TC_CF) to 0. Depending on the CFIFO mode of operation, the CFIFO status will also change upon the detection of an asserted EOQ bit on the last transferred command. See Section 19.4.3.5, "CFIFO Scan Trigger Modes," for details. 0 Not the last entry of the command queue. 1 Last entry of the command queue. **Note:** If both the pause and EOQ bits are asserted in the same command message the respective flags are set, but the CFIFO status changes as if only the EOQ bit were asserted.
1	PAUSE	Pause bit. Allows software to create sub-queues within a command queue. When the eQADC completes the transfer of a command with an asserted pause bit, the CFIFO enters the WAITING FOR TRIGGER state. Refer to Section 19.4.3.6.1, "CFIFO Operation Status," for a description of the state transitions. The pause bit is only valid when CFIFO operation mode is configured to single or continuous-scan edge trigger mode. 0 Do not enter WAITING FOR TRIGGER state after transfer of the current command message. 1 Enter WAITING FOR TRIGGER state after transfer of the current command message. **Note:** If both the pause and EOQ bits are asserted in the same command message the respective flags are set, but the CFIFO status changes as if only the EOQ bit were asserted.
2–4	—	Reserved.
5	EB	External buffer bit. This bit should always be cleared for messages sent to an on-chip ADC. 0 Command is sent to an internal command buffer. 1 Command is sent to an external command buffer.

Table 19-36. On-Chip ADC Field Descriptions: Read Configuration (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>6</td><td>BN</td><td>Buffer number. Indicates which buffer the message will be stored in. Buffers 1 and 0 can either be internal or external depending on the EB bit setting.
0 Message stored in buffer 0.
1 Message stored in buffer 1.</td></tr>
<tr><td>7</td><td>R/W</td><td>Read/write. An asserted R/W bit indicates a read configuration command.
0 Write
1 Read</td></tr>
<tr><td>8–11</td><td>MESSAGE_TAG [0:3]</td><td>MESSAGE_TAG field. Allows the eQADC to separate returning results into different RFIFOs. When the eQADC transfers a command, the MESSAGE_TAG is included as part of the command. Eventually the external device/on-chip ADC returns the result with the same MESSAGE_TAG. The eQADC separates incoming messages into different RFIFOs by decoding the MESSAGE_TAG of the incoming data.
<table>
<tr><th>MESSAGE_TAG[0:3]</th><th>MESSAGE_TAG Meaning</th></tr>
<tr><td>0b0000</td><td>Result is sent to RFIFO 0</td></tr>
<tr><td>0b0001</td><td>Result is sent to RFIFO 1</td></tr>
<tr><td>0b0010</td><td>Result is sent to RFIFO 2</td></tr>
<tr><td>0b0011</td><td>Result is sent to RFIFO 3</td></tr>
<tr><td>0b0100</td><td>Result is sent to RFIFO 4</td></tr>
<tr><td>0b0101</td><td>Result is sent to RFIFO 5</td></tr>
<tr><td>0b0110–0b0111</td><td>Reserved</td></tr>
<tr><td>0b1000</td><td>Null message received</td></tr>
<tr><td>0b1001</td><td>Reserved for customer use. [1]</td></tr>
<tr><td>0b1010</td><td>Reserved for customer use. [1]</td></tr>
<tr><td>0b1011–0b1111</td><td>Reserved</td></tr>
</table>
[1] These messages are treated as null messages. Therefore, they must obey the format for incoming null messages and return valid BUSY0/1 fields. Refer to Section , “ Null Message Format for External Device Operation.”</td></tr>
<tr><td>12–23</td><td>—</td><td>Reserved.</td></tr>
<tr><td>24–31</td><td>ADC_REG_ ADDRESS [0:7]</td><td>ADC register address. Selects a register on the ADC register set to be written or read. Only half-word addresses can be used. See Table 19-25.</td></tr>
</table>

ADC Result Format for On-Chip ADC Operation

When the FIFO control unit receives a return data message, it decodes the MESSAGE_TAG field and stores the 16-bit data into the appropriate RFIFO. This section describes the ADC result portion of the result message returned by the on-chip ADCs.

The 16-bit data stored in the RFIFOs can be the following:

- Data read from an ADC register with a read configuration command. In this case, the stored 16-bit data corresponds to the contents of the ADC register that was read.
- A time stamp. In this case, the stored 16-bit data is the value of the time base counter latched when the eQADC detects the end of the analog input voltage sampling. For details see Section 19.4.5.3, “Time Stamp Feature.”

- A conversion result. In this case, the stored 16-bit data contains a right justified 14-bit result data. The conversion result can be calibrated or not depending on the status of CAL bit in the command that requested the conversion. When the CAL bit is negated, this 14-bit data is obtained by executing a 2-bit left-shift on the 12-bit data received from the ADC. When the CAL bit is asserted, this 14-bit data is the result of the calculations performed in the EQADC MAC unit using the12-bit data received from the ADC and the calibration constants GCC and OCC (See Section 19.4.5.4, "ADC Calibration Feature"). Then, this 14-bit data is further formatted into a 16-bit format according to the status of the FMT bit in the conversion command. When FMT is asserted, the 14-bit result data is reformatted to look as if it was measured against an imaginary ground at VREF/2 (the msb (most significant bit) bit of the 14-bit result is inverted), and is sign-extended to a 16-bit format as in Figure 19-29. When FMT is negated, the eQADC zero-extends the 14-bit result data to a 16-bit format as in Figure 19-30. Correspondence between the analog voltage in a channel and the calculated digital values is shown in Table 19-39.

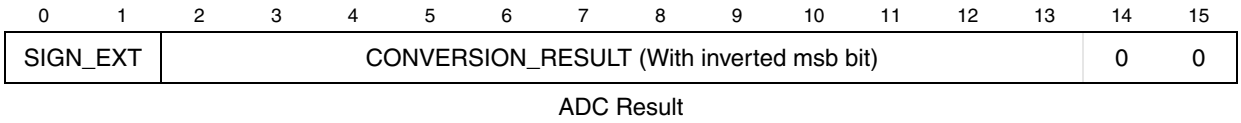


Figure 19-29. ADC Result Format when FMT = 1 (Right Justified Signed)—On-Chip ADC Operation

Table 19-37. ADC Result Format when FMT = 1 Field Descriptions

Bits	Name	Description
0–1	SIGN_EXT [0:1]	Sign extension. Only has meaning when FMT is asserted. SIGN_EXT is 0b00 when CONVERSION_RESULT is positive, and 0b11 when CONVERSION_RESULT is negative.
2–15	CONVERSION_RESULT [0:13]	Conversion result. A digital value corresponding to the analog input voltage in a channel when the conversion command was initiated. The two's complement representation is used to express negative values.

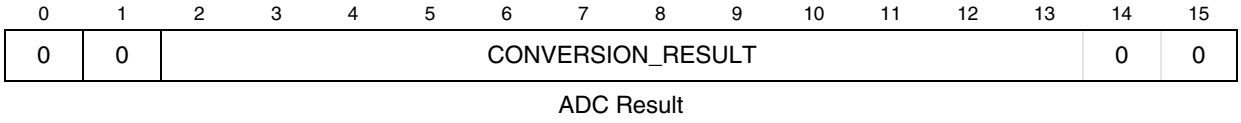


Figure 19-30. ADC Result Format when FMT = 0 (Right Justified Unsigned)—On-Chip ADC Operation

Table 19-38. ADC Result Format when FMT = 0 Field Descriptions

Bits	Name	Description
0–1	SIGN_EXT [0:1]	Sign extension. Only has meaning when FMT is asserted. SIGN_EXT is 0b00 when CONVERSION_RESULT is positive, and 0b11 when CONVERSION_RESULT is negative.
2–15	CONVERSION_RESULT [0:13]	Conversion result. A digital value corresponding to the analog input voltage in a channel when the conversion command was initiated.

Table 19-39. Correspondence between Analog Voltages and Digital Values[1, 2]

	Voltage Level on Channel (V)	Corresponding 12-bit Conversion Result Returned by the ADC	16-bit Result Sent to RFIFOs (FMT=0) [3]	16-bit Result Sent to RFIFOs (FMT=1) [3]
Single-Ended Conversions	5.12	0xFFF	0x3FFC	0x1FFC
	5.12 – lsb	0xFFF	0x3FFC	0x1FFC
	...	...	...	...
	2.56	0x800	0x2000	0x0000
	...	...	...	...
	1 lsb	0x001	0x0004	0xE004
	0	0x000	0x0000	0xE000
Differential Conversions	2.56	0xFFF	0x3FFC	0x1FFC
	2.56 – lsb	0xFFF	0x3FFC	0x1FFC
	...	...	...	...
	0	0x800	0x2000	0x0000
	...	...	...	...
	-2.56 + lsb	0x001	0x0004	0xE004
	-2.56	0x000	0x0000	0xE000

1 $V_{REF}=V_{RH}-V_{RL}=5.12V$. Resulting in one 12-bit count (lsb) =1.25mV.

2 The two's complement representation is used to express negative values.

3 Assuming uncalibrated conversion results.

19.4.1.2.2 Message Formats for External Device Operation

This section describes the command messages, data messages, and null messages formats used for external device operation.

Command Message Format for External Device Operation

Figure 19-31 describes the command message format for external device operation. Command message formats for on-chip operation and for external device operation share the same CFIFO header format. However, there are no limitations regarding the format an ADC Command used to communicate to an arbitrary external device. Only the upper bit of an ADC Command has a fixed format (BN field) to indicate to the FIFO control unit/external device to which external command buffer the corresponding command should be sent. The remaining 25 bits can be anything decodable by the external device. Only the ADC command portion of a command message is transferred to the external device.

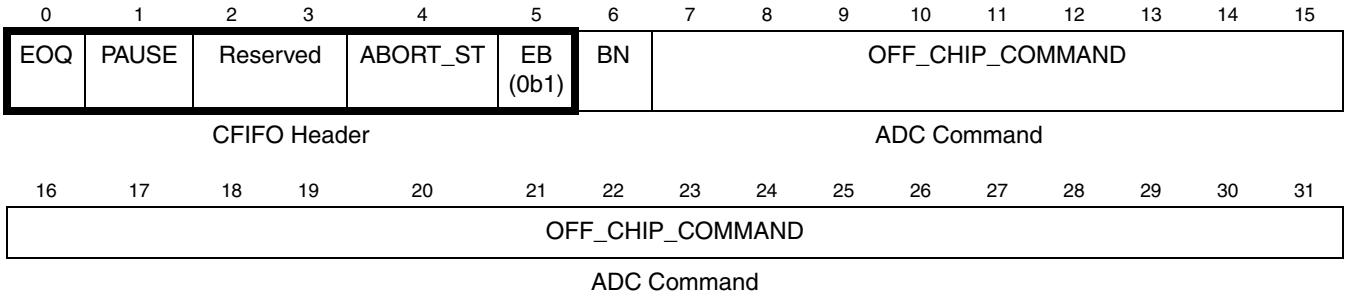


Figure 19-31. Command Message Format for External Device Operation

Table 19-40. On-Chip ADC Field Descriptions: External Device Operation

Bits	Name	Description
0	EOQ	End-of-queue. Asserted in the last command of a command queue to indicate to the eQADC that a scan of the queue is completed. EOQ instructs the eQADC to reset its current CFIFO transfer counter value (TC_CF) to 0. Depending on the CFIFO mode of operation, the CFIFO status will also change upon the detection of an asserted EOQ bit on the last transferred command. See Section 19.4.3.5, "CFIFO Scan Trigger Modes," for details. 0 Not the last entry of the command queue. 1 Last entry of the command queue. **Note:** If both the pause and EOQ bits are asserted in the same command message the respective flags are set, but the CFIFO status changes as if only the EOQ bit were asserted.
1	PAUSE	Pause bit. Allows software to create sub-queues within a command queue. When the eQADC completes the transfer of a command with an asserted pause bit, the CFIFO enters the WAITING FOR TRIGGER state. Refer to Section 19.4.3.6.1, "CFIFO Operation Status," for a description of the state transitions. The pause bit is only valid when CFIFO operation mode is configured to single or continuous-scan edge trigger mode. 0 Do not enter WAITING FOR TRIGGER state after transfer of the current command message. 1 Enter WAITING FOR TRIGGER state after transfer of the current command message. **Note:** If both the pause and EOQ bits are asserted in the same command message the respective flags are set, but the CFIFO status changes as if only the EOQ bit were asserted.
2–3	—	Reserved.
4	ABORT_ST	ABORT serial transmission. Indicates whether an on-going serial transmission should be aborted or not. All CFIFOs can abort null message transmissions when triggered but only CFIFO0 can abort command transmissions of lower priority CFIFOs. For more on serial transmission aborts see Section 19.4.3.2, "CFIFO Prioritization and Command Transfer." 0 Do not abort current serial transmission. 1 Abort current serial transmission.
5	EB	External buffer. This bit should always be set for messages sent to an external ADC. 0 Command is sent to an internal command buffer. 1 Command is sent to an external command buffer.

Table 19-40. On-Chip ADC Field Descriptions: External Device Operation (continued)

Bits	Name	Description
6	BN	Refer to Section , " Conversion Command Message Format for On-Chip ADC Operation."
7–31	OFF_CHIP_ COMMAND [0:24]	OFF-CHIP COMMAND Field. The OFF_CHIP_COMMAND field can be anything decodable by the external device. It is 25 bits long and it is transferred together with the BN bit to the external device when the CFIFO is triggered. Refer to Section , " Conversion Command Message Format for On-Chip ADC Operation," for a description of the command message used when interfacing with the on-chip ADCs.

Result Message Format for External Device Operation

Data is returned from the ADCs in the form of result messages. A result message is composed of an RFIFO header and an ADC result. The FIFO control unit decodes the information contained in the RFIFO header and sends the contents of the ADC result to the appropriate RFIFO. Only data stored on the ADC_RESULT field is stored in the RFIFOs/result queues. The ADC result of any received message with a null data message tag will be ignored. The format of a result message returned from the external device is shown in Figure 19-32. It is 26 bits long, and is composed of a MESSAGE_TAG field, information about the status of the buffers (BUSY fields), and result data. The BUSY fields are needed to inform the eQADC about when it is appropriate to transfer commands to the external command buffers.

Figure 19-32. Result Message Format for External Device Operation

Table 19-41. Result Message Format for External Device Operation

Bits	Name	Description
6–7	—	Reserved.
8–11	MESSAGE_TAG [0:3]	MESSAGE_TAG Field. Refer to Section , " Conversion Command Message Format for On-Chip ADC Operation."

Table 19-41. Result Message Format for External Device Operation (continued)

Bits	Name	Description
12–15	BUSY*n* [0:1]	BUSY status. The BUSY fields indicate if the external device can receive more commands. Table 19-42 shows how these two bits are encoded. When an external device cannot accept any more new commands, it must set BUSY*n* to a value indicating "Do not send commands" in the returning message. The BUSY fields of values 0b10 and 0b10 can be freely encoded by the external device to allow visibility of the status of the external command buffers for debug. As an example, they could indicate the number of entries in an external command buffer.
16–31	ADC_RESULT [0:15]	ADC RESULT Field. The result data received from the external device or on-chip ADC. This can be the result of a conversion command, data requested via a read configuration command, or time stamp value. The ADC_RESULT of any incoming message with a null message tag will be ignored. When the MESSAGE_TAG is for an RFIFO, the eQADC extracts the 16-bit ADC_RESULT from the raw message and stores it into the appropriate RFIFO.

Table 19-42. Command BUFFER*n* BUSY Status[1]

BUSY*n*[0:1]	Meaning
0b00	Send available commands—command buffer is empty
0b01	Send available commands
0b10	Send available commands
0b11	Do not send commands

[1] After reset, the eQADC always assumes that the external command buffers are full and cannot receive commands.

Null Message Format for External Device Operation

Null messages are only transferred through the serial interface to allow results and unsolicited control data, like the status of the external command buffers, to return when there are no more commands pending to transfer. Null messages are only transmitted when serial transmissions from the eQADC SSI are enabled (see ESSIE field in Section 19.3.2.1, "eQADC Module Configuration Register (EQADC_MCR),"), and when one of the following conditions apply:

1. There are no triggered CFIFOs with commands bound for external command buffers.
2. There are triggered CFIFOs with commands bound for external command buffers but the external buffers are full. The eQADC detected returning BUSY*n* fields indicating "Do not send commands."

Figure 19-33 illustrates the null message send format. When the eQADC transfers a null message, it directly shifts out the 26-bit data content inside the Section 19.3.2.2, "eQADC Null Message Send Format Register (EQADC_NMSFR)." The register must be programmed with the null message send format of the external device.

Figure 19-34 illustrates the null message receive format. It has the same fields found in a result message with the exception that the ADC result is not used. Refer to Section , " Result Message Format for External Device Operation," for more information. The MESSAGE_TAG field must be set to the null message tag (0b1000). The eQADC does not store into an RFIFO any incoming message with a null message tag.

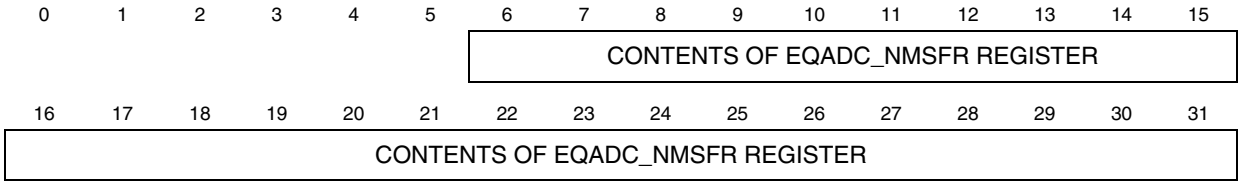


Figure 19-33. Null Message Send Format for External Device Operation

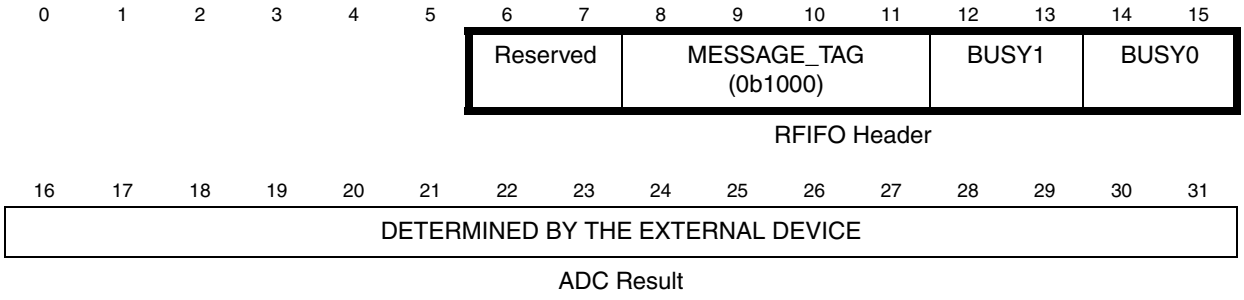


Figure 19-34. Null Message Receive Format for External Device Operation

Table 19-43. Null Message Receive Format for External Device Operation

Bits	Name	Description
6–7	—	Reserved.
8–11	MESSAGE_TAG[0:3]	MESSAGE_TAG field. Refer to Section , “ Conversion Command Message Format for On-Chip ADC Operation.”
12–15	BUSY*n* [0:1]	BUSY status. Refer to Section , “ Result Message Format for External Device Operation.”
16–31	—	Determined by the external device.

19.4.2 Command/Result Queues

The command and result queues are actually part of the eQADC system although they are not hardware implemented inside the eQADC. Instead command and result queues are user-defined queues located in system memory. Each command queue entry is a 32-bit command message.The last entry of a command queue has the EOQ bit asserted to indicate that it is the last entry of the queue. The result queue entry is a 16-bit data item.

See Section 19.1.4, “Modes of Operation,” for a description of the message formats and their flow in eQADC.

Refer to Section 19.5.5, “Command Queue and Result Queue Usage,” for examples of how command queues and result queues can be used.

19.4.3 eQADC Command FIFOs

19.4.3.1 CFIFO Basic Functionality

There are six prioritized CFIFOs located in the eQADC. Each CFIFO is four entries deep, and each CFIFO entry is 32 bits long. A CFIFO serves as a temporary storage location for the command messages stored in the command queues in system memory. When a CFIFO is not full, the eQADC sets the corresponding CFFF bit in Section 19.3.2.8, "eQADC FIFO and Interrupt Status Registers 0–5 (EQADC_FISRn)." If CFFE is asserted as in Section 19.3.2.7, "eQADC Interrupt and eDMA Control Registers 0–5 (EQADC_IDCRn)," the eQADC generates requests for more commands from a command queue. An interrupt request, served by the host CPU, is generated when CFFS is negated, and a eDMA request, served by the eDMA, is generated when CFFS is asserted. The host CPU or the eDMA respond to these requests by writing to the Section 19.3.2.4, "eQADC CFIFO Push Registers 0–5 (EQADC_CFPRn)," to fill the CFIFO.

NOTE

Only whole words must be written to EQADC_CFPR. Writing half-words or bytes to EQADC_CFPR will still push the whole 32-bit CF_PUSH field into the corresponding CFIFO, but undefined data will fill the areas of CF_PUSH that were not specifically designated as target locations for writing.

Figure 19-35 describes the important components in the CFIFO. Each CFIFO is implemented as a circular set of registers to avoid the need to move all entries at each push/pop operation. The push next data pointer points to the next available CFIFO location for storing data written into the eQADC command FIFO push register. The transfer next data pointer points to the next entry to be removed from CFIFO*n* when it completes a transfer. The CFIFO transfer counter control logic counts the number of entries in the CFIFO and generates eDMA or interrupt requests to fill the CFIFO. TNXTPTR in Section 19.3.2.8, "eQADC FIFO and Interrupt Status Registers 0–5 (EQADC_FISRn)," indicates the index of the entry that is currently being addressed by the transfer next data pointer, and CFCTR, in the same register, provides the number of entries stored in the CFIFO.

Using TNXTPTR and CFCTR, the absolute addresses for the entries indicated by the transfer next data pointer and by the push next data pointer can be calculated using the following formulas:

```
Transfer Next Data Pointer Address = CFIFOn_BASE_ADDRESS + TNXTPTRn*4

Push Next Data Pointer Address = CFIFOn_BASE_ADDRESS +
[(TNXTPTRn+CFCTRn) mod CFIFO_DEPTH] * 4
```

where

- *a mod b* returns the remainder of the division of *a* by *b*.
- CFIFO*n*_BASE_ADDRESS is the smallest memory mapped address allocated to a CFIFO*n* entry.
- CFIFO_DEPTH is the number of entries contained in a CFIFO - four in this implementation.

When CFS*n* in Section 19.3.2.11, "eQADC CFIFO Status Register (EQADC_CFSR)," is in the TRIGGERED state, the eQADC generates the proper control signals for the transfer of the entry pointed by transfer next data pointer. CFUF*n* in Section 19.3.2.8, "eQADC FIFO and Interrupt Status Registers 0–5 (EQADC_FISRn)," is set when a CFIFO*n* underflow event occurs. A CFIFO underflow occurs when the CFIFO is in the TRIGGERED state and it becomes empty. No commands will be transferred from an underflowing CFIFO, nor will command transfers from lower priority CFIFOs be blocked. CFIFO*n* is empty when the transfer next data pointer *n* equals the push next data pointer *n* and CFCTR*n* is 0. CFIFO*n* is full when the transfer next data pointer *n* equals the push next data pointer *n* and CFCTR*n* is not 0.

When the eQADC completes the transfer of an entry from CFIFO*n*: the transferred entry is popped from CFIFO*n*, the CFIFO counter CFCTR in the Section 19.3.2.8, "eQADC FIFO and Interrupt Status Registers 0–5 (EQADC_FISRn)," is decremented by 1, and transfer next data pointer *n* is incremented by 1 (or wrapped around) to point to the next entry in the CFIFO. The transfer of entries bound for the on-chip ADCs is considered completed when they are stored in the appropriate ADC command buffer. The transfer of entries bound for the external device is considered completed when the serial transmission of the entry is completed.

When the EQADC_CFPR*n* is written and CFIFO*n* is not full, the CFIFO counter CFCTR*n* is incremented by 1, and the push next data pointer *n* then is incremented by 1 (or wrapped around) to point to the next entry in the CFIFO.

When the EQADC_CFPR*n* is written but CFIFO*n* is full, the eQADC will not increment the counter value and will not overwrite any entry in CFIFO*n*.

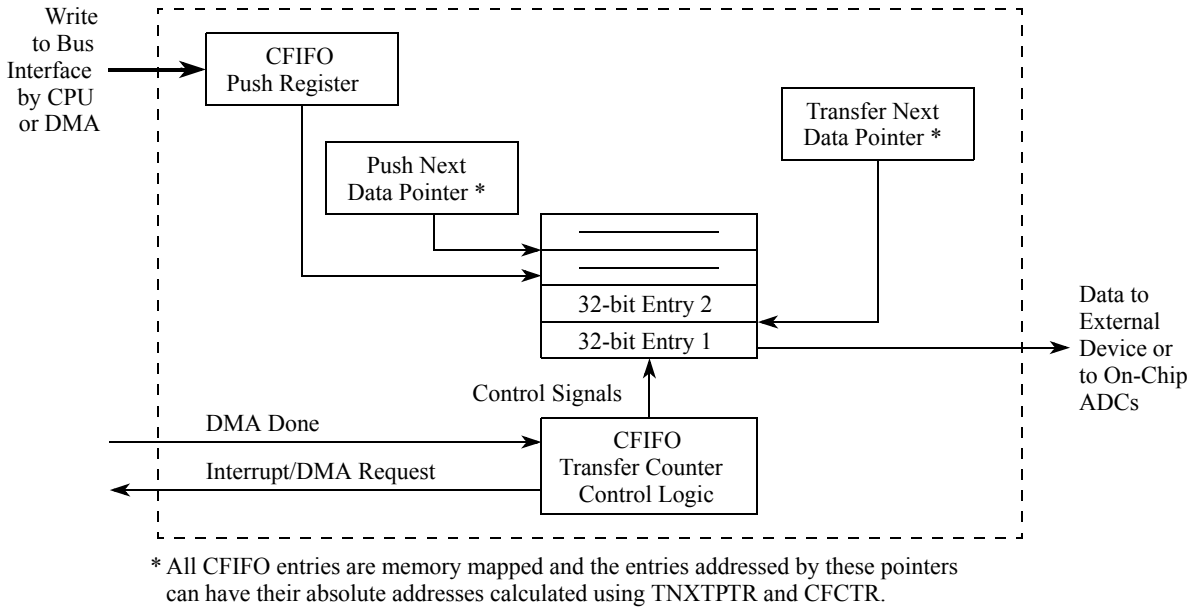


Figure 19-35. CFIFO Diagram

The detailed behavior of the push next data pointer and transfer next data pointer is described in the example shown in Figure 19-36 where a CFIFO with 16 entries is shown for clarity of explanation, the actual hardware implementation has only four entries. In this example, CFIFO*n* with 16 entries is shown in sequence after pushing and transferring entries.

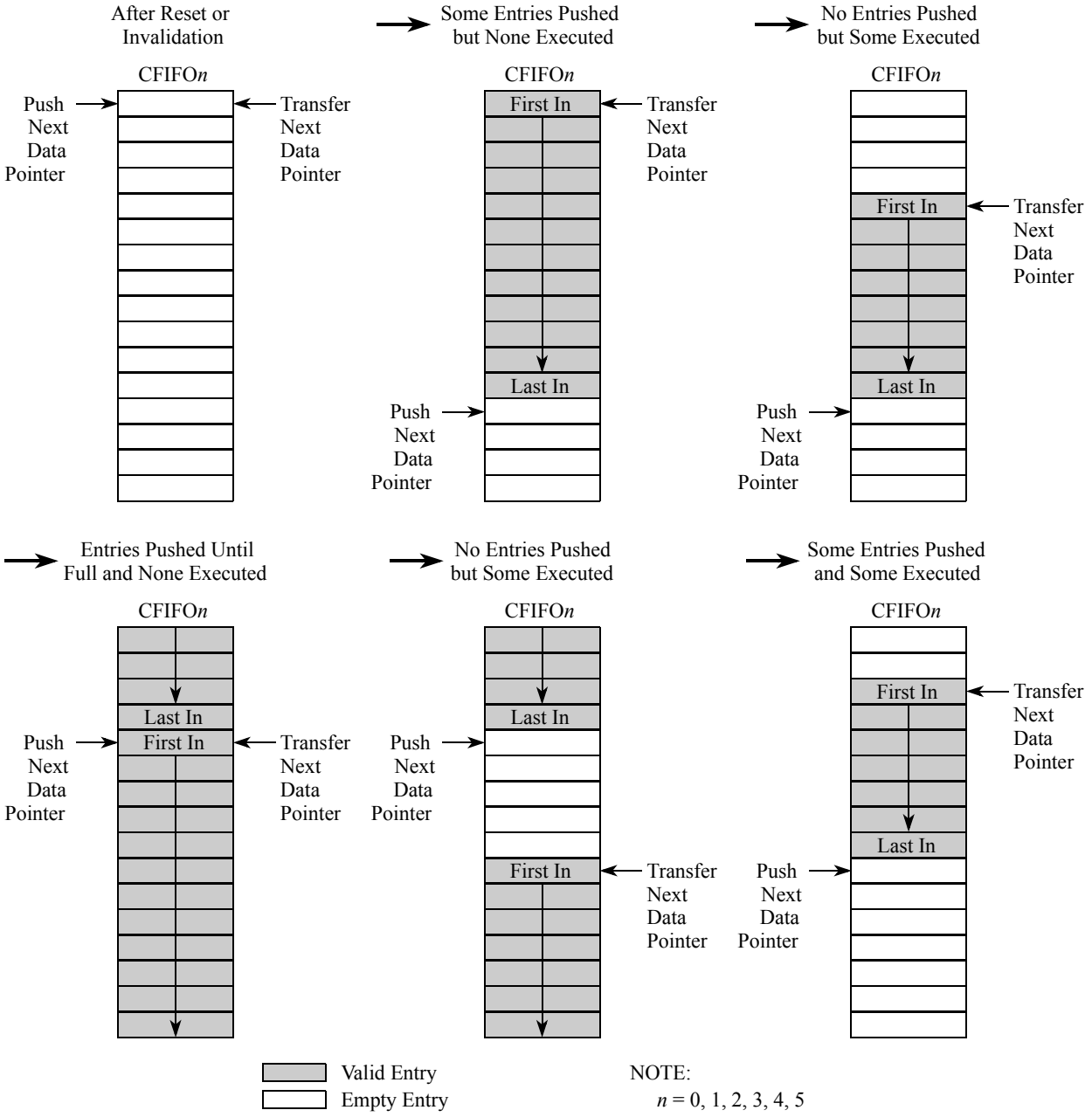


Figure 19-36. CFIFO Entry Pointer Example

19.4.3.2 CFIFO Prioritization and Command Transfer

The CFIFO priority is fixed according to the CFIFO number. A CFIFO with a smaller number has a higher priority. When commands of distinct CFIFOs are bound for the same destination (the same on-chip ADC), the higher priority CFIFO is always served first. A triggered, not-underflowing CFIFO will start the transfer of its commands when the following occur:

- Its commands are bound for an internal command buffer that is not full, and it is the highest priority triggered CFIFO sending commands to that buffer.

- Its commands are bound for an external command buffer that is not full, and it is the highest priority triggered CFIFO sending commands to an external buffer that is not full.

A triggered CFIFO with commands bound for a certain command buffer consecutively transfers its commands to the buffer until one of the following occurs:

- An asserted end of queue bit is reached.
- An asserted pause bit is encountered and the CFIFO is configured for edge trigger mode.
- CFIFO is configured for level trigger mode and a closed gate is detected.
- In case its commands are bound for an internal command buffer, a higher priority CFIFO that uses the same internal buffer is triggered.
- In case its commands are bound for an external command buffer, a higher priority CFIFO that uses an external buffer is triggered.

The prioritization logic of the eQADC, depicted in Figure 19-37, is composed of three independent submodules: one that prioritizes CFIFOs with commands bound for ADC0, another that prioritizes CFIFOs with commands for ADC1, and a last one that prioritizes CFIFOs with commands for external command buffer 2 and buffer 3. As these three submodules are independent, simultaneous commands to ADC0, to ADC1, and to eQADC SSI transmit buffer are allowed. The hardware identifies the destination of a command by decoding the EB and BN bits in the command message (see Section 19.4.1.2, "Message Format in eQADC," for details).

NOTE

Triggered but empty CFIFOs, underflowing CFIFOs, are not considered for prioritization. No data from these CFIFOs will be sent to either of the on-chip ADCs or to either of the external command buffers, nor will they stop lower priority CFIFOs from transferring commands.

Whenever ADC0 is able to receive new commands, the prioritization submodule selects the highest-priority triggered CFIFO with a command bound for ADC0, and sends it to the ADC. In case ADC0 is able to receive new entries but there are no triggered CFIFOs with commands bound for it, nothing is sent. The submodule prioritizing ADC1 usage behaves in the same way.

When the eQADC SSI is enabled and ready to start serial transmissions, the submodule prioritizing eQADC SSI usage writes command or null messages into the eQADC SSI transmit buffer, data written to the eQADC SSI transmit buffer is subsequently transmitted to the external device through the eQADC SSI link. The submodule writes commands to the eQADC SSI transmit buffer when there are triggered CFIFOs with commands bound for not-full external command buffers. The command written to the transmit buffer belongs to the highest priority CFIFO sending commands to an external buffer that is not full. This implies that a lower priority CFIFO can have its commands sent if a higher priority CFIFO cannot send its commands due to a full command buffer. The submodule writes null messages to the eQADC SSI transmit buffer when there are no triggered CFIFOs with commands bound for external command buffers, or when there are triggered CFIFOs with commands bound for external buffers but the external buffers are full. The eQADC monitors the status of the external buffers by decoding the BUSY fields of the incoming result messages from the external device (see Section , " Result Message Format for External Device Operation," for details).

NOTE

When a lower priority CFIFO is served first because a higher priority CFIFO cannot send its commands due to a full external command buffer, there is a possibility that command transfers from the lower priority CFIFO will be interrupted and the CFIFO will become non-coherent, when the higher priority CFIFO again becomes ready to send commands. Whether the lower priority CFIFO becomes non-coherent or not depends on the rate at which commands on the external ADCs are executed, on the rate at which commands are transmitted to the external command buffers, and on the depth of those buffers.

Once a serial transmission is started, the submodule monitors triggered CFIFOs and manages the abort of serial transmissions. In case a null message is being transmitted, the serial transmission is aborted when all of the following conditions are met:

- A not-underflowing CFIFO in the TRIGGERED state has commands bound for an external command buffer that is not full, and it is the highest priority CFIFO sending commands to an external buffer that is not full.
- The ABORT_ST bit of the command to be transmitted is asserted.
- The 26th bit of the currently transmitting null message has not being shifted out.

The command from the CFIFO is then written into eQADC SSI transmit buffer, allowing for a new serial transmission to initiate.

In case a command is being transmitted, the serial transmission is aborted when all following conditions are met:

- CFIFO0 is in the TRIGGERED state, is not underflowing, and its current command is bound for an external command buffer that is not full.
- The ABORT_ST bit of the command to be transmitted is asserted.
- The 26th bit of the currently transmitting command has not being shifted out.

The command from CFIFO0 is then written into eQADC SSI transmit buffer, allowing for a new serial transmission to initiate.

NOTE

The aborted command is not popped from the preempted CFIFO and will be retransmitted as soon as its CFIFO becomes the highest priority CFIFO sending commands to an unfilled external command buffer.

After a serial transmission is completed, the eQADC prioritizes the CFIFOs and schedules a command or a null message to be sent in the next serial transmission. After the data for the next transmission has been defined and scheduled, the eQADC can, under certain conditions, stretch the $\overline{\text{SDS}}$ negation time in order to allow the schedule of new data for that transmission. This occurs when the eQADC acknowledges that the status of a higher-priority CFIFO has changed to the TRIGGERED state and attempts to schedule that CFIFO command before $\overline{\text{SDS}}$ is asserted. Only commands of CFIFOs that have the ABORT_ST bit asserted can be scheduled in this manner. Under such conditions:

1. A CFIFO0 command is scheduled for the next transmission independently of the type of data that was previously scheduled. The time during which $\overline{\text{SDS}}$ is negated is stretched in order to allow the eQADC to load the CFIFO0 command and start its transmission.

2. CFIFO1-5 commands are only scheduled for the next transmission if the previously scheduled data was a null message. The time during which $\overline{SDS}$ is negated is stretched in order to allow the eQADC to load that command and start its transmission. However, if the previously scheduled data was a command, no rescheduling occurs and the next transmission starts without delays.

If a CFIFO becomes triggered while $\overline{SDS}$ is negated, but the eQADC only attempts to reschedule that CFIFO command after $\overline{SDS}$ is asserted, then the current transmission is aborted depending on if the conditions for that are met or not.

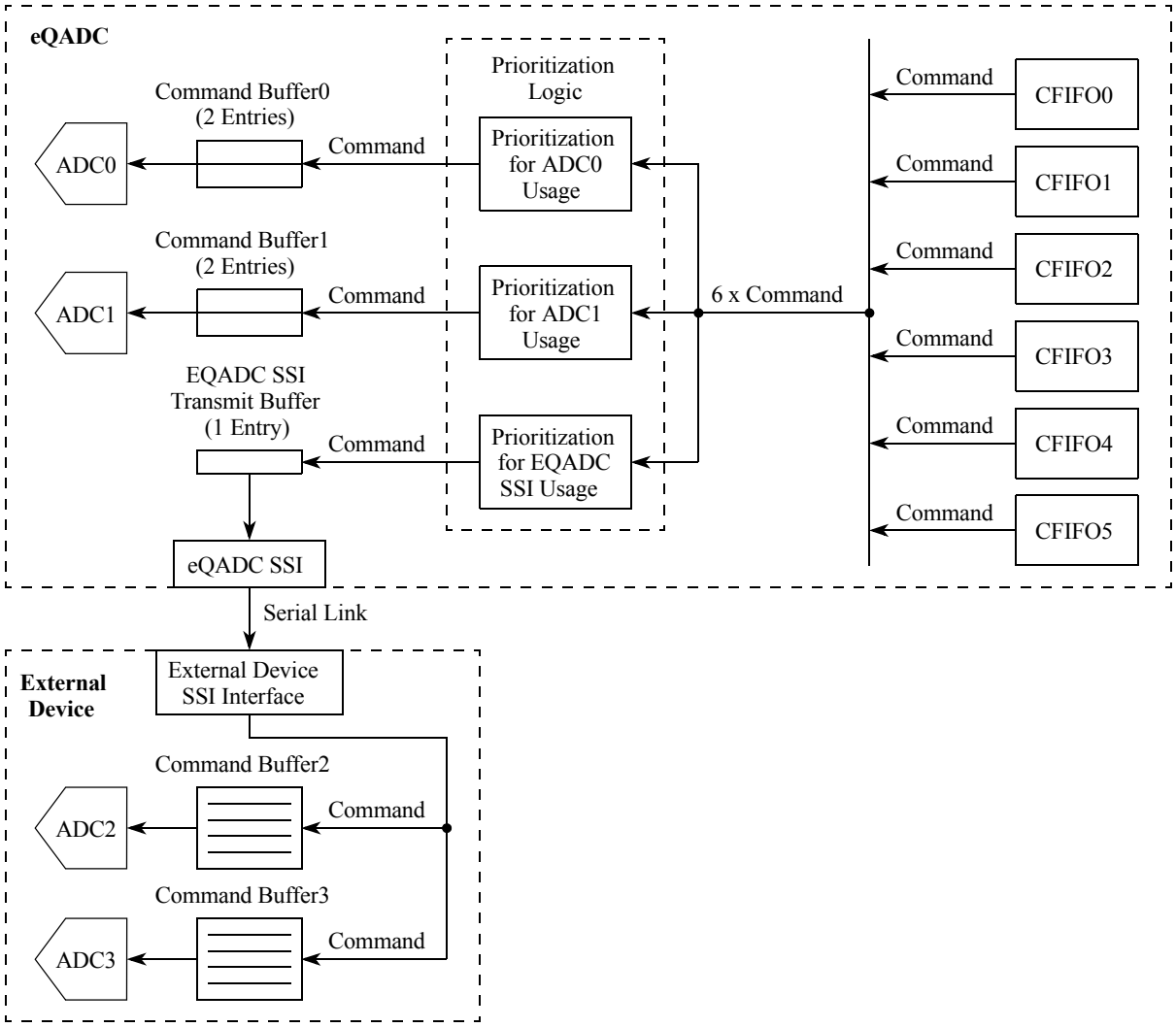


Figure 19-37. CFIFO Prioritization Logic

19.4.3.3 External Trigger from eTPU or eMIOS Channels

The six eQADC external trigger inputs can be connected to either an external pin, an eTPU channel, or an eMIOS channel. The input source for each eQADC external trigger is individually specified in the eQADC trigger input select register (SIU_ETISR) in the SIU block.

The eQADC trigger numbers specified by SIU_ETISR[TSEL(0-5)] correspond to CFIFO numbers 0-5. To calculate the CFIFO number that each trigger is connected to, divide the eDMA channel number by 2.

A complete description of the eTPU and eMIOS trigger function and configuration is found in Section 6.4.5.1, "eQADC External Trigger Input Multiplexing."

19.4.3.4 External Trigger Event Detection

The digital filter length field in Section 19.3.2.3, "eQADC External Trigger Digital Filter Register (EQADC_ETDFR)," specifies the minimum number of system clocks that the external trigger signals 0 and 1 must be held at a logic level to be recognized as valid. All ETRIG signals are filtered. A counter for each queue trigger is implemented to detect a transition between logic levels. The counter counts at the system clock rate. The corresponding counter is cleared and restarted each time the signal transitions between logic levels. When the corresponding counter matches the value specified by the digital filter length field in Section 19.3.2.3, "eQADC External Trigger Digital Filter Register (EQADC_ETDFR)," the eQADC considers the ETRIG logic level to be valid and passes that new logic level to the rest of the eQADC.

The filter is only for filtering the ETRIG signal. Logic after the filter checks for transitions between filtered values, such as for detecting the transition from a filtered logic level zero to a filtered logic level one in rising edge external trigger mode. The eQADC can detect rising edge, falling edge, or level gated external triggers. The digital filter will always be active independently of the status of the MODE*n* field in Section 19.3.2.6, "eQADC CFIFO Control Registers 0–5 (EQADC_CFCRn)," but the edge, level detection logic is only active when MODE*n* is set to a value different from disabled, and in case MODE*n* is set to single scan mode, when the SSS bit is asserted. Note that the time necessary for a external trigger event to result into a CFIFO status change is not solely determined by the DFL field in the Section 19.3.2.3, "eQADC External Trigger Digital Filter Register (EQADC_ETDFR)." After being synchronized to the system clock and filtered, a trigger event is checked against the CFIFO trigger mode. Only then, after a valid trigger event is detected, the eQADC accordingly changes the CFIFO status. Refer to Figure 19-38 for an example.

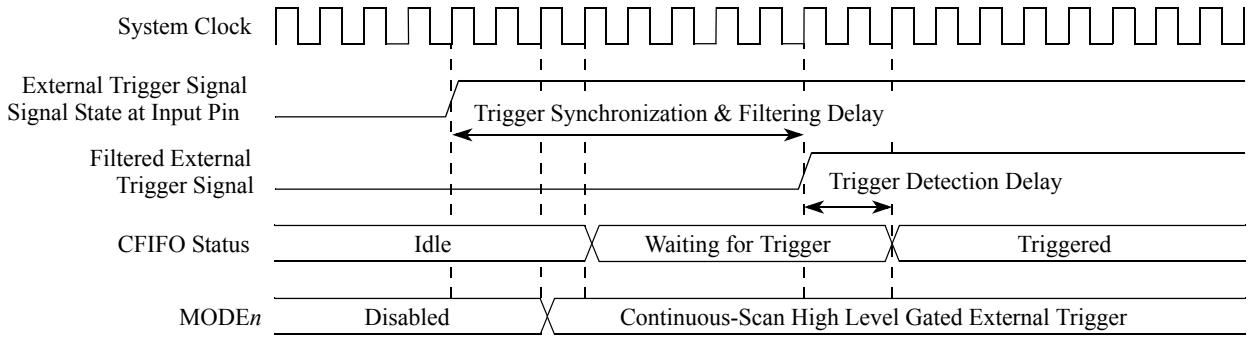


Figure 19-38. ETRIG Event Propagation Example

19.4.3.5 CFIFO Scan Trigger Modes

The eQADC supports two different scan modes, single-scan and continuous-scan. Refer to Table 19-44 for a summary of these two scan modes. When a CFIFO is triggered, the eQADC scan mode determines whether the eQADC will stop command transfers from a CFIFO, and wait for software intervention to rearm the CFIFO to detect new trigger events, upon detection of an asserted EOQ bit in the last transfer. Refer to Section 19.4.1.2, "Message Format in eQADC," for details about command formats.

CFIFOs can be configured in single-scan or continuous-scan mode. When a CFIFO is configured in single-scan mode, the eQADC scans the command queue one time. The eQADC stops future command transfers from the triggered CFIFO after detecting the EOQ bit set in the last transfer. After a EOQ bit is detected, software involvement is required to rearm the CFIFO so that it can detect new trigger events.

When a CFIFO is configured for continuous-scan mode, no software involvement is necessary to rearm the CFIFO to detect new trigger events after an asserted EOQ is detected. In continuous-scan mode the whole command queue is scanned multiple times.

The eQADC also supports different triggering mechanisms for each scan mode. The eQADC will not transfer commands from a CFIFO until the CFIFO is triggered. The combination of scan modes and triggering mechanisms allows the support of different requirements for scanning input channels. The scan mode and trigger mechanism are configured by programming the MODE*n* field in Section 19.3.2.6, "eQADC CFIFO Control Registers 0–5 (EQADC_CFCRn)."

Enabled CFIFOs can be triggered by software or external trigger events. The elapsed time from detecting a trigger to transferring a command is a function of clock frequency, trigger synchronization, trigger filtering, programmable trigger events, command transfer, CFIFO prioritization, ADC availability, etc. Fast and predictable transfers can be achieved by ensuring that the CFIFO is not underflowing and that the target ADC can accept commands when the CFIFO is triggered.

19.4.3.5.1 Disabled Mode

The MODE*n* field in Section 19.3.2.6, "eQADC CFIFO Control Registers 0–5 (EQADC_CFCRn)," for all of the CFIFOs can be changed from any other mode to disabled at any time. No trigger event can initiate command transfers from a CFIFO which has its MODE field programmed to disabled.

NOTE

If MODE*n* is not disabled, it must not be changed to any other mode besides disabled. If MODE*n* is disabled and the CFIFO status is IDLE, MODE*n* can be changed to any other mode.

If MODE*n* is changed to disabled:

- The CFIFO execution status will change to IDLE. The timing of this change depends on whether a command is being transferred or not:
  - When no command transfer is in progress, the eQADC switches the CFIFO to IDLE status immediately.
  - When a command transfer to an on-chip ADC is in progress, the eQADC will complete the transfer, update TC_CF, and switch CFIFO status to IDLE. Command transfers to the internal ADCs are considered completed when a command is written to the relevant buffer.
  - When a command transfer to an external command buffer is in progress, the eQADC will abort the transfer and switch CFIFO status to IDLE. If the eQADC cannot abort the transfer, that is when the 26th bit of the serial message has being already shifted out, the eQADC will complete the transfer, update TC_CF and then switch CFIFO status to IDLE.
- The CFIFOs are not invalidated automatically. The CFIFO still can be invalidated by writing a 1 to the CFINV*n* bit (see Section 19.3.2.6). Certify that CFS has changed to IDLE before setting CFINV*n*.
- The TC_CF*n* value also is not reset automatically, but it can be reset by writing 0 to it.
- The EQADC_FISRn[SSS] bit (see Section 19.3.2.8) is negated. The SSS bit can be set even if a 1 is written to the EQADC_CFCR[SSE] bit (see Section 19.3.2.6) in the same write that the MODE*n* field is changed to a value other than disabled.
- The trigger detection hardware is reset. If MODE*n* is changed from disabled to an edge trigger mode, a new edge, matching that edge trigger mode, is needed to trigger the command transfers from the CFIFO.

NOTE

CFIFO fill requests, which generated when CFFF is asserted, are not automatically halted when MODE*n* is changed to disabled. CFIFO fill requests will still be generated until EQADC_IDCRn[CFFE] bit is cleared (see Section 19.3.2.7).

19.4.3.5.2 Single-Scan Mode

In single-scan mode, a single pass through a sequence of command messages in the user-defined command queue is performed.

In single-scan software trigger mode, the CFIFO is triggered by an asserted single-scan status bit, EQADC_FISRn[SSS] (see Section 19.3.2.8). The SSS bit is set by writing 1 to the single-scan enable bit, EQADC_CFCRn[SSE] (see Section 19.3.2.6).

In single-scan edge- or level-trigger mode, the respective triggers are only detected when the SSS bit is asserted. When the SSS bit is negated, all trigger events for that CFIFO are ignored. Writing a 1 to the SSE bit can be done during the same write cycle that the CFIFO operation mode is configured.

Only the eQADC can clear the SSS bit. Once SSS is asserted, it remains asserted until the eQADC completes the command queue scan, or the CFIFO operation mode, EQADC_CFCRn[MODE*n*] (see Section 19.3.2.6) is changed to disabled. The SSS*n* bit will be negated while MODE*n* is disabled.

Single-Scan Software Trigger

When single-scan software trigger mode is selected, the CFIFO is triggered by an asserted SSS bit. The SSS bit is asserted by writing 1 to the SSE bit. Writing to SSE while SSS is already asserted will not have any effect on the state of the SSS bit, nor will it cause a trigger overrun event.

The CFIFO commands start to be transferred when the CFIFO becomes the highest priority CFIFO using an available on-chip ADC or an external command buffer that is not full. When an asserted EOQ bit is encountered, the eQADC will clear the SSS bit. Setting the SSS bit is required for the eQADC to start the next scan of the queue.

The pause bit has no effect in single-scan software trigger mode.

Single-Scan Edge Trigger

When SSS is asserted and an edge triggered mode is selected for a CFIFO, an appropriate edge on the associated trigger signal causes the CFIFO to become triggered. For example, if rising-edge trigger mode is selected, the CFIFO becomes triggered when a rising edge is sensed on the trigger signal. The CFIFO commands start to be transferred when the CFIFO becomes the highest priority CFIFO using an available on-chip ADC, or an external command buffer that is not full.

When an asserted EOQ bit is encountered, the eQADC clears SSS and stops command transfers from the CFIFO. An asserted SSS bit and a subsequent edge trigger event are required to start the next scan for the CFIFO. When an asserted pause bit is encountered, the eQADC stops command transfers from the CFIFO, but SSS remains set. Another edge trigger event is required for command transfers to continue. A trigger overrun happens when the CFIFO is in a TRIGGERED state and an edge trigger event is detected.

Single-Scan Level Trigger

When SSS is asserted and a level gated trigger mode is selected, the input level on the associated trigger signal puts the CFIFO in a TRIGGERED state. When the CFIFO is set to high-level gated trigger mode, a high level signal opens the gate, and a low level closes the gate. When the CFIFO is set to low-level gated trigger mode, a low level signal opens the gate, and a high level closes the gate. If the corresponding level is already present, setting the SSS bit triggers the CFIFO. The CFIFO commands start to be transferred

when the CFIFO becomes the highest priority CFIFO using an available on-chip ADC or an external command buffer that is not full.

The eQADC clears the SSS bit and stops transferring commands from a triggered CFIFO when an asserted EOQ bit is encountered or when CFIFO status changes from triggered due to the detection of a closed gate. If a closed gate is detected while no command transfers are taking place and the CFIFO status is triggered, the CFIFO status is immediately changed to IDLE, the SSS bit is negated, and the PF flag is asserted. If a closed gate is detected during the serial transmission of a command to the external device, it will have no effect on the CFIFO status until the transmission completes. Once the transmission is completed, the TC_CF counter is updated, the SSS bit is negated, the PF flag is asserted, and the CFIFO status is changed to IDLE. An asserted SSS bit and a level trigger are required to restart the CFIFO. Command transfers will restart from the point they have stopped.

If the gate closes and opens during the same serial transmission of a command to the external device, it will have no effect on the CFIFO status or on the PF flag, but the TORF flag will become asserted as was exemplified in Figure 19-40. Therefore, closing the gate for a period less than a serial transmission time interval does not guarantee that the closure will affect command transfers from a CFIFO.

The pause bit has no effect in single-scan level-trigger mode.

19.4.3.5.3 Continuous-Scan Mode

In continuous-scan mode, multiple passes looping through a sequence of command messages in a command queue are executed. When a CFIFO is programmed for a continuous-scan mode, the EQADC_CFCRn[SSE] (see Section 19.3.2.6) does not have any effect.

Continuous-Scan Software Trigger

When a CFIFO is programmed to continuous-scan software trigger mode, the CFIFO is triggered immediately. The CFIFO commands start to be transferred when the CFIFO becomes the highest priority CFIFO using an available on-chip ADC or an external command buffer that is not full. When a CFIFO is programmed to run in continuous-scan software trigger mode, the eQADC will not halt transfers from the CFIFO until the CFIFO operation mode is modified to disabled or a higher priority CFIFO preempts it. Although command transfers will not stop upon detection of an asserted EOQ bit, the EOQF is set and, if enabled, an EOQ interrupt request is generated.

The pause bit has no effect in continuous-scan software trigger mode.

Continuous-Scan Edge Trigger

When rising, falling, or either edge trigger mode is selected for a CFIFO, a corresponding edge on the associated ETRIG signal places the CFIFO in a TRIGGERED state. The CFIFO commands start to be transferred when the CFIFO becomes the highest priority CFIFO using an available on-chip ADC or an external command buffer that is not full.

When an EOQ or a pause is encountered, the eQADC halts command transfers from the CFIFO and, if enabled, the appropriate interrupt requests are generated. Another edge trigger event is required to resume command transfers but no software involvement is required to rearm the CFIFO in order to detect such event.

A trigger overrun happens when the CFIFO is already in a TRIGGERED state and a new edge trigger event is detected.

Continuous-Scan Level Trigger

When high or low level gated trigger mode is selected, the input level on the associated trigger signal places the CFIFO in a TRIGGERED state. When high-level gated trigger is selected, a high-level signal opens the gate, and a low level closes the gate. The CFIFO commands start to be transferred when the

CFIFO becomes the highest priority CFIFO using an available on-chip ADC or an external buffer that is not full. Although command transfers will not stop upon detection of an asserted EOQ bit at the end of a command transfer, the EOQF is asserted and, if enabled, an EOQ interrupt request is generated.

The eQADC stops transferring commands from a triggered CFIFO when CFIFO status changes from triggered due to the detection of a closed gate. If a closed gate is detected while no command transfers are taking place and the CFIFO status is TRIGGERED, the CFIFO status is immediately changed to waiting for trigger and the PF flag is asserted. If a closed gate is detected during the serial transmission of a command to the external device, it will have no effect on the CFIFO status until the transmission completes. Once the transmission is completed, the TC_CF counter is updated, the PF flag is asserted, and the CFIFO status is changed to waiting for trigger. Command transfers will restart as the gate opens.

If the gate closes and opens during the same serial transmission of a command to the external device, it will have no effect on the CFIFO status or on the PF flag, but the TORF flag will become asserted as was exemplified in Figure 19-40. Therefore, closing the gate for a period less than a serial transmission time interval does not guarantee that the closure will affect command transfers from a CFIFO.

The pause bit has no effect in continuous-scan level-trigger mode.

19.4.3.5.4 CFIFO Scan Trigger Mode Start/Stop Summary

Table 19-44 summarizes the start and stop conditions of command transfers from CFIFOs for all of the single-scan and continuous-scan trigger modes.

Table 19-44. CFIFO Scan Trigger Mode—Command Transfer Start/Stop Summary

Trigger Mode	Requires Asserted SSS to Recognize Trigger Events?	Command Transfer Start/Restart Condition	Stop on asserted EOQ bit[1]?	Stop on asserted Pause bit[2]?	Other Command Transfer Stop Condition[3] [4]
Single Scan Software	Not Applicable	Asserted SSS bit.	Yes	No	None.
Single Scan Edge	Yes	A corresponding edge occurs when the SSS bit is asserted.	Yes	Yes	None.
Single Scan Level	Yes	Gate is opened when the SSS bit is asserted.	Yes	No	The eQADC also stops transfers from the CFIFO when CFIFO status changes from triggered due to the detection of a closed gate.[5]
Continuous Scan Software	No	CFIFO starts automatically after being configured into this mode.	No	No	None.
Continuous Scan Edge	No	A corresponding edge occurs.	Yes	Yes	None.
Continuous Scan Level	No	Gate is opened.	No	No	The eQADC also stops transfers from the CFIFO when CFIFO status changes from triggered due to the detection of a closed gate.[5]

1 Refer to Section 19.4.3.6.2, "Command Queue Completion Status," for more information on EOQ.

2 Refer to Section 19.4.3.6.3, "Pause Status," for more information on pause.

3 The eQADC always stops command transfers from a CFIFO when the CFIFO operation mode is disabled.

4 The eQADC always stops command transfers from a CFIFO when a higher priority CFIFO is triggered. Refer to Section 19.4.3.2, "CFIFO Prioritization and Command Transfer," for information on CFIFO priority.

5 If a closed gate is detected while no command transfers are taking place, it will have immediate effect on the CFIFO status. If a closed gate is detected during the serial transmission of a command to the external device, it will have no effect on the CFIFO status until the transmission completes.

19.4.3.6 CFIFO and Trigger Status

19.4.3.6.1 CFIFO Operation Status

Each CFIFO has its own CFIFO status field. CFIFO status (CFS) can be read from EQADC_CFSSR (see Section 19.3.2.11, "eQADC CFIFO Status Register (EQADC_CFSR)." Figure 19-39 and Table 19-45 indicate the CFIFO status switching condition. Refer to Table 19-18 for the meaning of each CFIFO operation status. The last CFIFO to transfer a command to an on-chip ADC can be read from the LCFT*n* (*n*=0,1) fields (see Section 19.3.2.10, "eQADC CFIFO Status Snapshot Registers 0–2 (EQADC_CFSSRn)." The last CFIFO to transfer a command to a specific external command buffer can be identified by reading the EQADC_CFSSR*n*[LCFTSSI] and EQADC_CFSSR*n*[ENI] fields (see Section 19.3.2.10, "eQADC CFIFO Status Snapshot Registers 0–2 (EQADC_CFSSRn)."

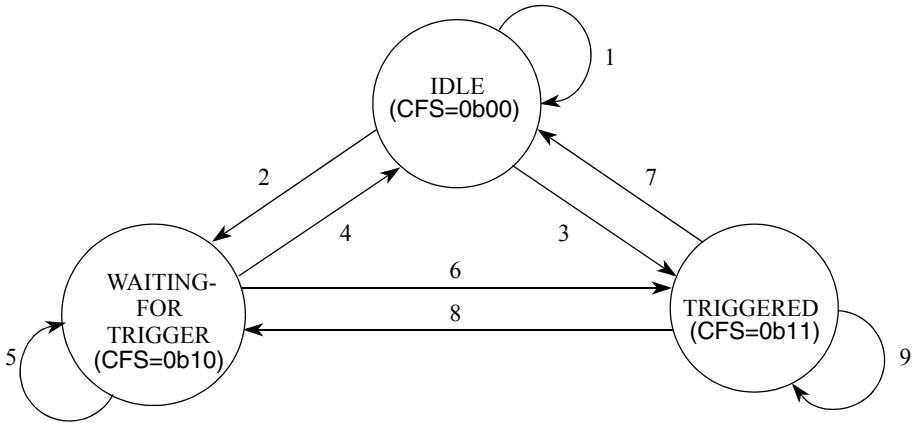


Figure 19-39. State Machine of CFIFO Status

Table 19-45. Command FIFO Status Switching Condition

No.	From Current CFIFO Status (CFS)	To New CFIFO Status (CFS)	Status Switching Condition
1	IDLE (00)	IDLE (0b00)	• CFIFO mode is programmed to disabled, OR • CFIFO mode is programmed to single-scan edge or level trigger mode and SSS is negated.
2		WAITING FOR TRIGGER (0b10)	• CFIFO mode is programmed to continuous-scan edge or level trigger mode, OR • CFIFO mode is programmed to single-scan edge or level trigger mode and SSS is asserted, OR • CFIFO mode is programmed to single-scan software trigger mode.
3		TRIGGERED (0b11)	• CFIFO mode is programmed to continuous-scan software trigger mode
4	WAITING FOR TRIGGER (10)	IDLE (0b00)	• CFIFO mode is modified to disabled mode.
5		WAITING FOR TRIGGER (0b10)	• No trigger occurred.
6		TRIGGERED (0b11)	• Appropriate edge or level trigger occurred, OR • CFIFO mode is programmed to single-scan software trigger mode and SSS bit is asserted.

Table 19-45. Command FIFO Status Switching Condition (continued)

No.	From Current CFIFO Status (CFS)	To New CFIFO Status (CFS)	Status Switching Condition
7	TRIGGERED (11)	IDLE (0b00)	• CFIFO in single-scan mode, eQADC detects the EOQ bit asserted at end of command transfer, and CFIFO mode is not modified to disabled, OR • CFIFO, in single-scan level trigger mode, and the gate closes while no commands are being transferred from the CFIFO, and CFIFO mode is not modified to disabled, OR • CFIFO, in single-scan level trigger mode, and eQADC detects a closed gated at end of command transfer, and CFIFO mode is not modified to disabled, OR • CFIFO mode is modified to disabled mode and CFIFO was not transferring commands. • CFIFO mode is modified to disabled mode while CFIFO was transferring commands, and CFIFO completes or aborts the transfer.
8		WAITING FOR TRIGGER (0b10)	• CFIFO in single or continuous-scan edge trigger mode, eQADC detects the pause bit asserted at the end of command transfer, the EOQ bit in the same command is negated, and CFIFO mode is not modified to disabled, OR • CFIFO in continuous-scan edge trigger mode, eQADC detects the EOQ bit asserted at the end of command transfer, and CFIFO mode is not modified to disabled, OR • CFIFO, in continuous-scan level trigger mode, and the gate closes while no commands are being transferred from the CFIFO, and CFIFO mode is not modified to disabled, OR • CFIFO, in continuous-scan level trigger mode, and eQADC detects a closed gated at end of command transfer, and CFIFO mode is not modified to disabled.
9		TRIGGERED (0b11)	• No event to switch to IDLE or WAITING FOR TRIGGER status has happened.

19.4.3.6.2 Command Queue Completion Status

The end of queue flag, EQADC_FISRn[EOQF] (see Section 19.3.2.8) is asserted when the eQADC completes the transfer of a CFIFO entry with an asserted EOQ bit. Software sets the EOQ bit in the last command message of a user-defined command queue to indicate that this entry is the end of the queue. See Section 19.4.1.2, "Message Format in eQADC," for information on command message formats. The transfer of entries bound for the on-chip ADCs is considered completed when they are stored in the appropriate command buffer. The transfer of entries bound for the external device is considered completed when the serial transmission of the entry is completed.

The command with a EOQ bit asserted is valid and will be transferred. When EQADC_CFCRn[EOQIE] (see Section 19.3.2.6) and EQADC_FISRn[EOQF] are asserted, the eQADC will generate an end of queue interrupt request.

In single-scan modes, command transfers from the corresponding CFIFO will cease when the eQADC completes the transfer of a entry with an asserted EOQ. Software involvement is required to rearm the CFIFO so that it can detect new trigger events.

NOTE

An asserted EOQF*n* only implies that the eQADC has finished transferring a command with an asserted EOQ bit from CFIFO*n*. It does not imply that result data for the current command and for all previously transferred commands has been returned to the appropriate RFIFO.

19.4.3.6.3 Pause Status

In edge trigger mode, when the eQADC completes the transfer of a CFIFO entry with an asserted pause bit, the eQADC will stop future command transfers from the CFIFO and set EQADC_FISRn[PF] (see Section 19.3.2.8). Refer to Section 19.4.1.2, "Message Format in eQADC," for information on command message formats. The eQADC ignores the pause bit in command messages in any software level trigger mode. The eQADC sets the PF flag upon detection of an asserted pause bit only in single or continuous-scan edge trigger mode. When the PF flag is set for a CFIFO in single-scan edge trigger mode, the EQADC_FISRn[SSS] bit will not be cleared (see Section 19.3.2.8).

In level trigger mode, the definition of the PF flag has been redefined. In level trigger mode, when CFIFO*n* is in TRIGGERED status, PF*n* is set when the CFIFO status changes from TRIGGERED due to detection of a closed gate. The pause flag interrupt routine can be used to verify if the a complete scan of the command queue was performed. If a closed gate is detected while no command transfers are taking place, it will have immediate effect on the CFIFO status. If a closed gate is detected during the serial transmission of a command to the external device, it will have no effect on the CFIFO status until the transmission completes.

When EQADC_CFCR[PIE] (see Section 19.3.2.6) and EQADC_FISRn[PF] are asserted, the eQADC will generate a pause interrupt request.

NOTE

In edge trigger mode, an asserted PF*n* only implies that the eQADC finished transferring a command with an asserted pause bit from CFIFO*n*. It does not imply that result data for the current command and for all previously transferred commands has been returned to the appropriate RFIFO.

NOTE

In software or level trigger mode, when the eQADC completes the transfer of an entry from CFIFO*n* with an asserted pause bit, PF*n* will not be set and command transfers will continues without pausing.

19.4.3.6.4 Trigger Overrun Status

When a CFIFO is configured for edge- or level-trigger mode and is in a TRIGGERED state, an additional trigger occurring for the same CFIFO results in a trigger overrun. The trigger overrun bit for the corresponding CFIFO will be set (EQADC_FISRn[TORF*n*] = 1, see Section 19.3.2.8). When EQADC_CFCRn[TORIE] (see Section 19.3.2.6) and EQADC_FISRn[TORF] are asserted, the eQADC generates a trigger overrun interrupt request.

For CFIFOs configured for level-trigger mode, a trigger overrun event is only detected when the gate closes and reopens during a single serial command transmission as shown in Figure 19-40.

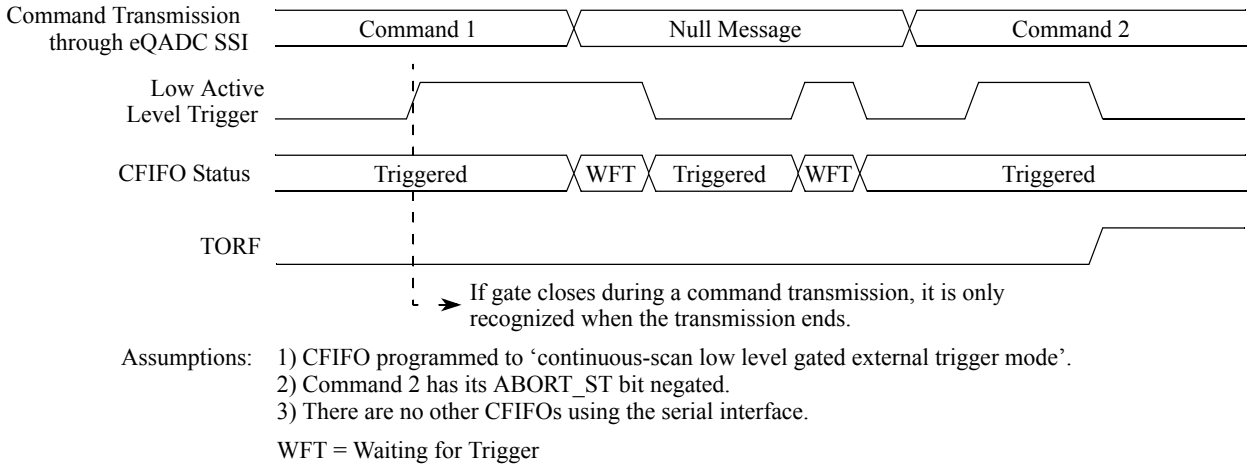


Figure 19-40. Trigger Overrun on Level-Trigger Mode CFIFOs

NOTE

The trigger overrun flag will not set for CFIFOs configured for software trigger mode.

19.4.3.6.5 Command Sequence Non-Coherency Detection

The eQADC provides a mechanism to indicate if a command sequence has been completely executed without interruptions. A command sequence is defined as a group of consecutive commands bound for the same ADC and it is expected to be executed without interruptions. A command sequence is coherent if its commands are executed in order without interruptions. Because commands are stored in the ADC's command buffers before being executed in the eQADC, a command sequence is coherent if, while it is transferring commands to an on-chip ADC command buffer, the buffer is only fed with commands from that sequence without ever becoming empty.

A command sequence starts when:

- A CFIFO in TRIGGERED state transfers its first command to an on-chip ADC.
- The CFIFO is constantly transferring commands and the previous command sequence ended.
- The CFIFO resumes command transfers after being interrupted.

And a command sequence ended when:

- An asserted EOQ bit is detected on the last transferred command.
- CFIFO is in edge-trigger mode and asserted pause bit is detected on the last transferred command.
- The ADC to which the next command is bound is different from the ADC to which the last command was transferred.

Figure 19-41 shows examples of how the eQADC would detect command sequences when transferring commands from a CFIFO. The smallest possible command sequence can have a single command as shown in example 3 of Figure 19-41.

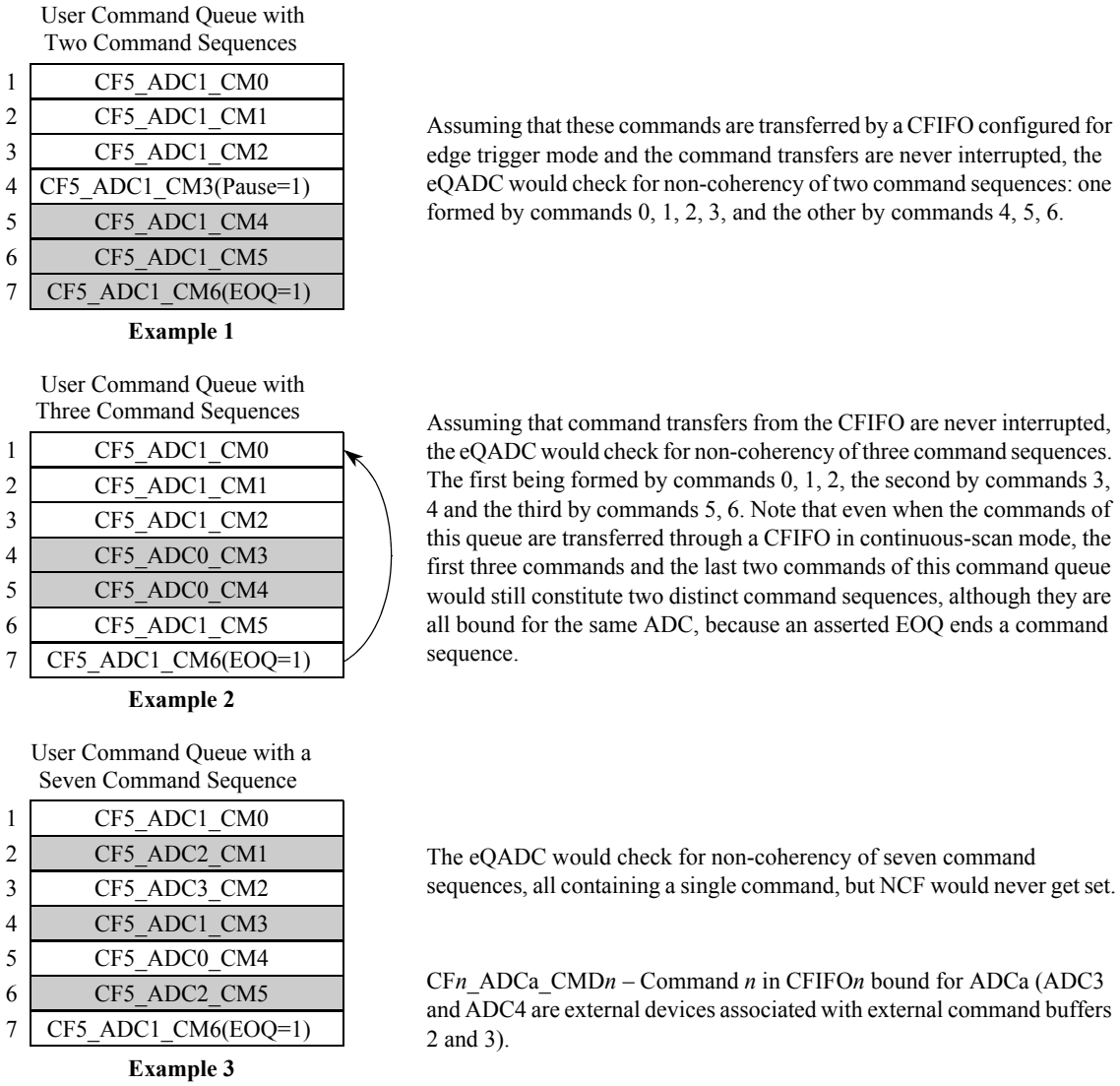


Figure 19-41. Command Sequence Examples

The NCF flag is used to indicate command sequence non-coherency. When the NCF*n* flag is asserted, it indicates that the command sequence being transferred through CFIFO*n* became non-coherent. The NCF flag only becomes asserted for CFIFOs in a TRIGGERED state.

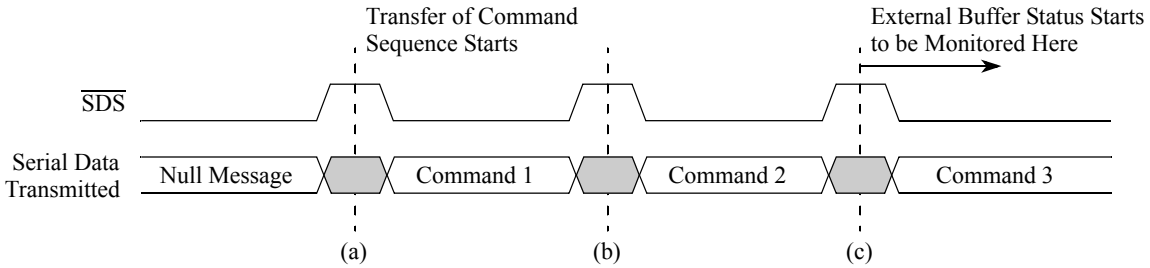
A command sequence is non-coherent when, after transferring the first command of a sequence from a CFIFO to a buffer, it cannot successively send all the other commands of the sequence before any of the following conditions are true:

- The CFIFO through which commands are being transferred is pre-empted by a higher priority CFIFO which sends commands to the same ADC. The NCF flag becomes asserted immediately after the first command transfer from the pre-empting CFIFO, that is the higher priority CFIFO, to the ADC in use is completed. See Figure 19-43.
- The external command buffer in use becomes empty. (Only the fullness of external buffers is monitored because the fill rate for internal ADC buffers is many times faster than the drain rate, and each has a dedicated priority engine.) This case happens when different CFIFOs attempt to use different external command buffers and the higher priority CFIFO bars the lower priority one from

sending new commands to its buffer—see Figure 19-44. An external command buffer is considered empty when the corresponding BUSY field in the last result message received from external device is encoded as “Send available commands - buffer is empty”. Refer to Section , “ Result Message Format for External Device Operation.” The NCF flag becomes asserted immediately after the eQADC detects that the external buffer in use becomes empty.

NOTE

After the transfer of a command sequence to an external command buffer starts, the eQADC ignores, for non-coherency detection purposes, the BUSY fields captured at the end of the first serial transmission. Thereafter, all BUSY fields captured at the end of consecutive serial transmissions are used to check the fullness of that external command buffer. This is done because the eQADC only updates its external ADC command buffer status record when it receives a serial message, resulting that the record kept by the eQADC is always outdated by, at least, the length of one serial transmission. This prevents a CFIFO from immediately becoming non-coherent when it starts transferring commands to an empty external command buffer. Refer to Figure 19-42 for an example.



Assumptions: 1) The CFIFO starts sending commands to an external command buffer when triggered.
2) Execution of a command on the external device takes longer than the time to complete three serial transmissions.

Figure 19-42. External Command Buffer Status Detection at Command Sequence Transfer Start

Table 19-46. External Buffer Status

Capture Point at eQADC	Buffer Status at External Device	Buffer Status as Captured by the eQADC	Used for NCF detection on the eQADC?
(a)	EMPTY	EMPTY	Don’t care
(b)	1 ENTRY	EMPTY	No
(c)	2 ENTRY	1 ENTRY	Yes

Once a command sequence starts to be transferred, the eQADC will check for the command sequence coherency until the command sequence ends or until one of the conditions below becomes true:

- The command sequence became non-coherent.
- The CFIFO status changed from the TRIGGERED state.
- The CFIFO had underflow.

NOTE

The NCF flag still becomes asserted if an external command buffer empty event is detected at the same time the eQADC stops checking for the coherency of a command sequence.

Once command transfers restart/continue, the non-coherency hardware will behave as if the command sequence started from that point. Figure 19-45 depicts how the non-coherency hardware will behave when a non-coherency event is detected.

NOTE

If MODE*n* is changed to disabled while a CFIFO is transferring commands, the NCF flag for that CFIFO will not become asserted.

NOTE

When the eQADC enters debug or stop mode while a command sequence is being executed, the NCF will become asserted if an empty external command buffer is detected after debug/stop mode is exited.

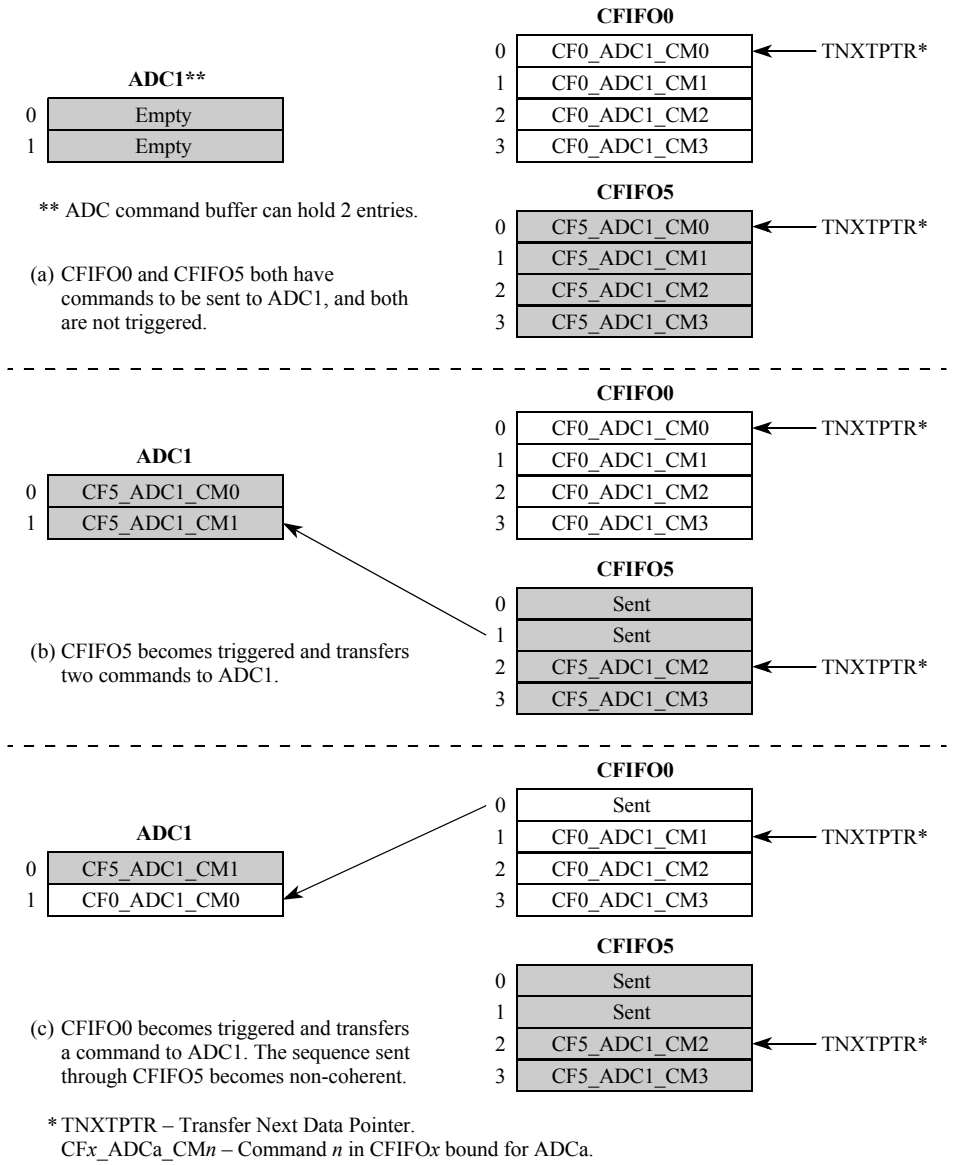


Figure 19-43. Non-Coherency Event When Different CFIFOs Use the Same Buffer

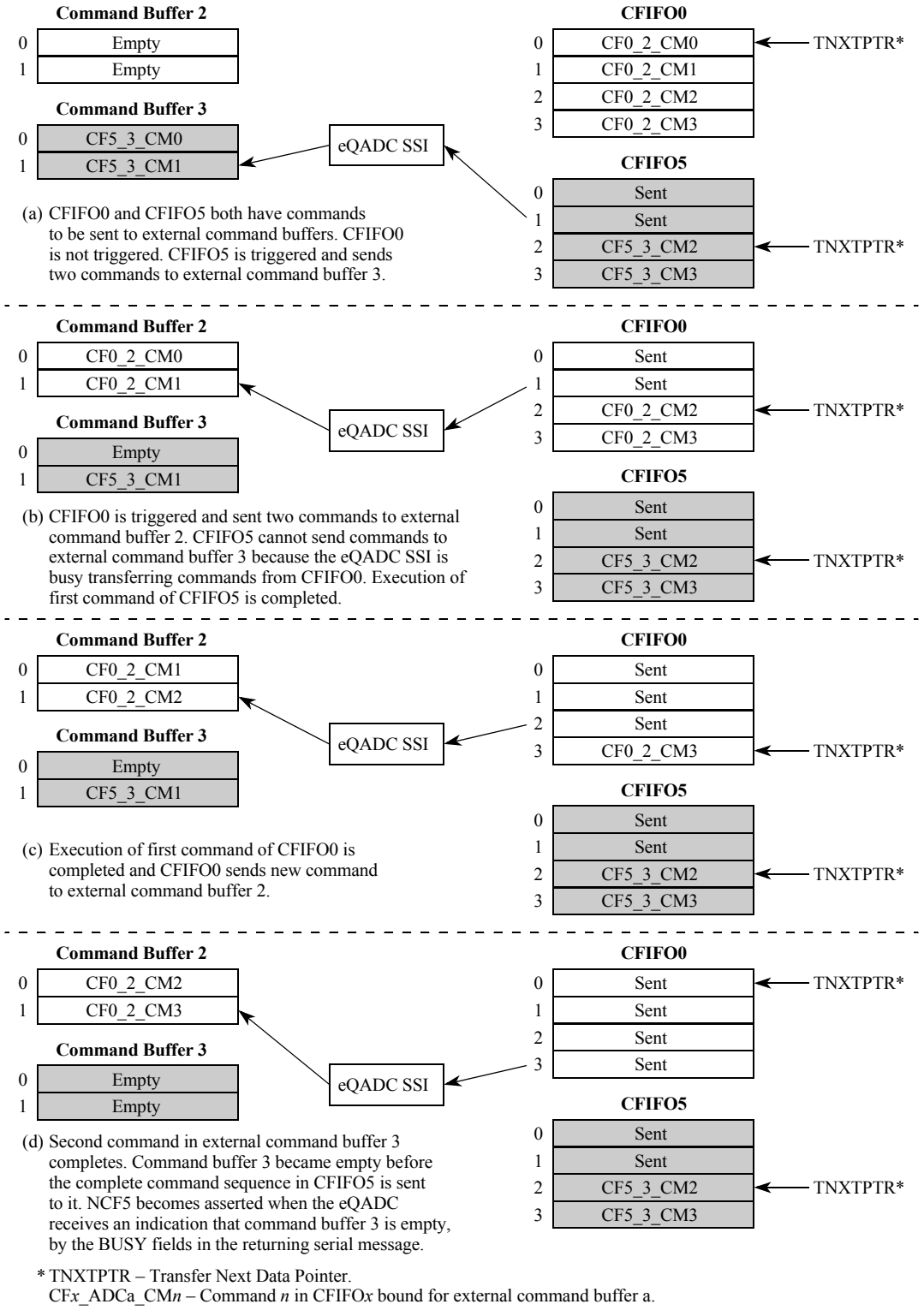


Figure 19-44. Non-Coherency Event When Different CFIFOs Are Using Different External Command Buffers

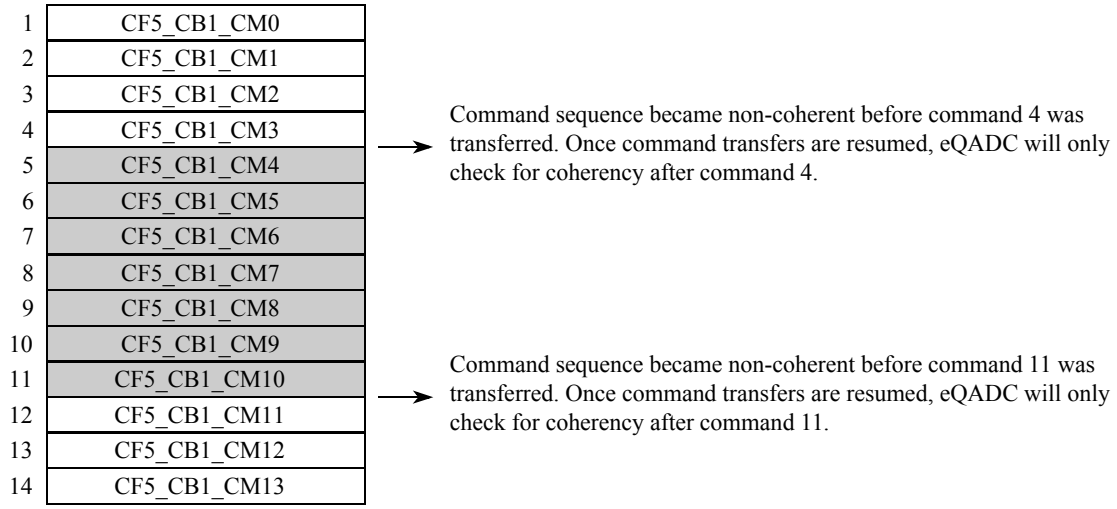


Figure 19-45. Non-coherency Detection When Transfers From A Command Sequence Are Interrupted

19.4.4 Result FIFOs

19.4.4.1 RFIFO Basic Functionality

There are six RFIFOs located in the eQADC. Each RFIFO is four entries deep, and each RFIFO entry is 16 bits long. Each RFIFO serves as a temporary storage location for the one of the result queues allocated in system memory. All result data is saved in the RFIFOs before being moved into the system result queues. When an RFIFO is not empty, the eQADC sets the corresponding EQADC_FISRn[RFDF] (see Section 19.3.2.8). If EQADC_IDCR*n*[RFDE] is asserted (see Section 19.3.2.7), the eQADC generates a request so that the RFIFO entry is moved to a result queue. An interrupt request, served by the host CPU, is generated when EQADC_IDCR*n*[RFDS] is negated, and an eDMA request, served by the eDMA, is generated when RFDS is asserted. The host CPU or the eDMA responds to these requests by reading EQADC_RFPRn (see Section 19.3.2.5) to retrieve data from the RFIFO.

NOTE

Reading a word, half-word, or any bytes from EQADC_RFPR*n* will pop an entry from RFIFO*n*,and the RFCTR*n* field will be decremented by 1.

The eDMA controller should be configured to read a single result (16-bit data) from the RFIFO pop registers for every asserted eDMA request it acknowledges. Refer to Section 19.5.2, "EQADC/eDMA Controller Interface" for eDMA controller configuration guidelines.

Figure 19-46 describes the important components in the RFIFO. Each RFIFO is implemented as a circular set of registers to avoid the need to move all entries at each push/pop operation. The pop next data pointer always points to the next RFIFO message to be retrieved from the RFIFO when reading eQADC_RFPR. The receive next data pointer points to the next available RFIFO location for storing the next incoming message from the on-chip ADCs or from the external device. The RFIFO counter logic counts the number of entries in RFIFO and generates interrupt or eDMA requests to drain the RFIFO.

EQADC_FISRn[POPNXTPTR] (see Section 19.3.2.8) indicates which entry is currently being addressed by the pop next data pointer, and EQADC_FISRn[RFCTR] provides the number of entries stored in the

RFIFO. Using POPNXTPTR and RFCTR, the absolute addresses for pop next data pointer and receive next data pointer can be calculated using the following formulas:

```
Pop Next Data Pointer Address= RFIFOn_BASE_ADDRESS + POPNXTPTRn*4

Receive Next Data Pointer Address = RFIFOn_BASE_ADDRESS +
[(POPNXTPTRn + RFCTRn) mod RFIFO_DEPTH] * 4
```

where

- *a mod b* returns the remainder of the division of *a* by *b*.
- RFIFO*n*_BASE_ADDRESS is the smallest memory mapped address allocated to an RFIFO*n* entry.
- RFIFO_DEPTH is the number of entries contained in a RFIFO - four in this implementation.

When a new message arrives and RFIFO*n* is not full, the eQADC copies its contents into the entry pointed by receive next data pointer. The RFIFO counter EQADC_FISR*n*[RFCTR*n*] (see Section 19.3.2.8) is incremented by 1, and the receive next data pointer *n* is also incremented by 1 (or wrapped around) to point to the next empty entry in RFIFO*n*. However, if the RFIFO*n* is full, the eQADC sets the EQADC_FISR*n*[RFOF] (see Section 19.3.2.8). The RFIFO*n* will not overwrite the older data in the RFIFO, the new data will be ignored, and the receive next data pointer *n* is not incremented or wrapped around. RFIFO*n* is full when the receive next data pointer *n* equals the pop next data pointer *n* and RFCTR*n* is not 0. RFIFO*n* is empty when the receive next data pointer *n* equals the pop next data pointer *n* and RFCTR*n* is 0.

When the eQADC RFIFO pop register *n* is read and the RFIFO*n* is not empty, the RFIFO counter RFCTR*n* is decremented by 1, and the pop next data pointer is incremented by 1 (or wrapped around) to point to the next RFIFO entry.

When the eQADC RFIFO pop register *n* is read and RFIFO*n* is empty, eQADC will not decrement the counter value and the pop next data pointer *n* will not be updated. The read value will be undefined.

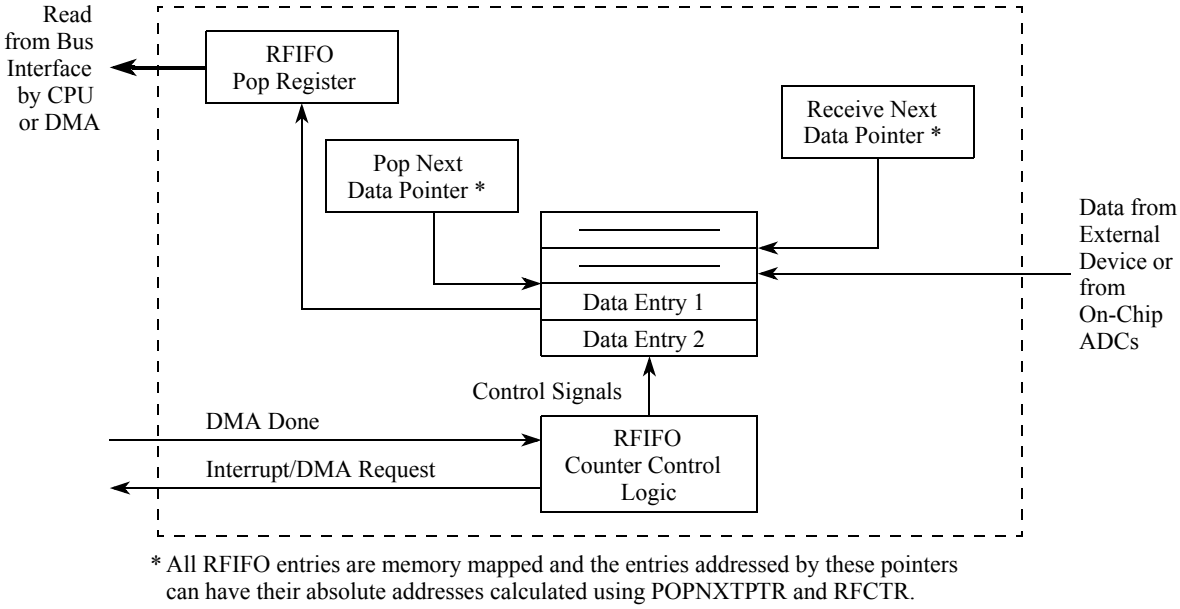


Figure 19-46. RFIFO Diagram

The detailed behavior of the pop next data pointer and receive next data pointer is described in the example shown in Figure 19-47 where an RFIFO with 16 entries is shown for clarity of explanation, the actual

hardware implementation has only four entries. In this example, RFIFO*n* with 16 entries is shown in sequence after popping or receiving entries.

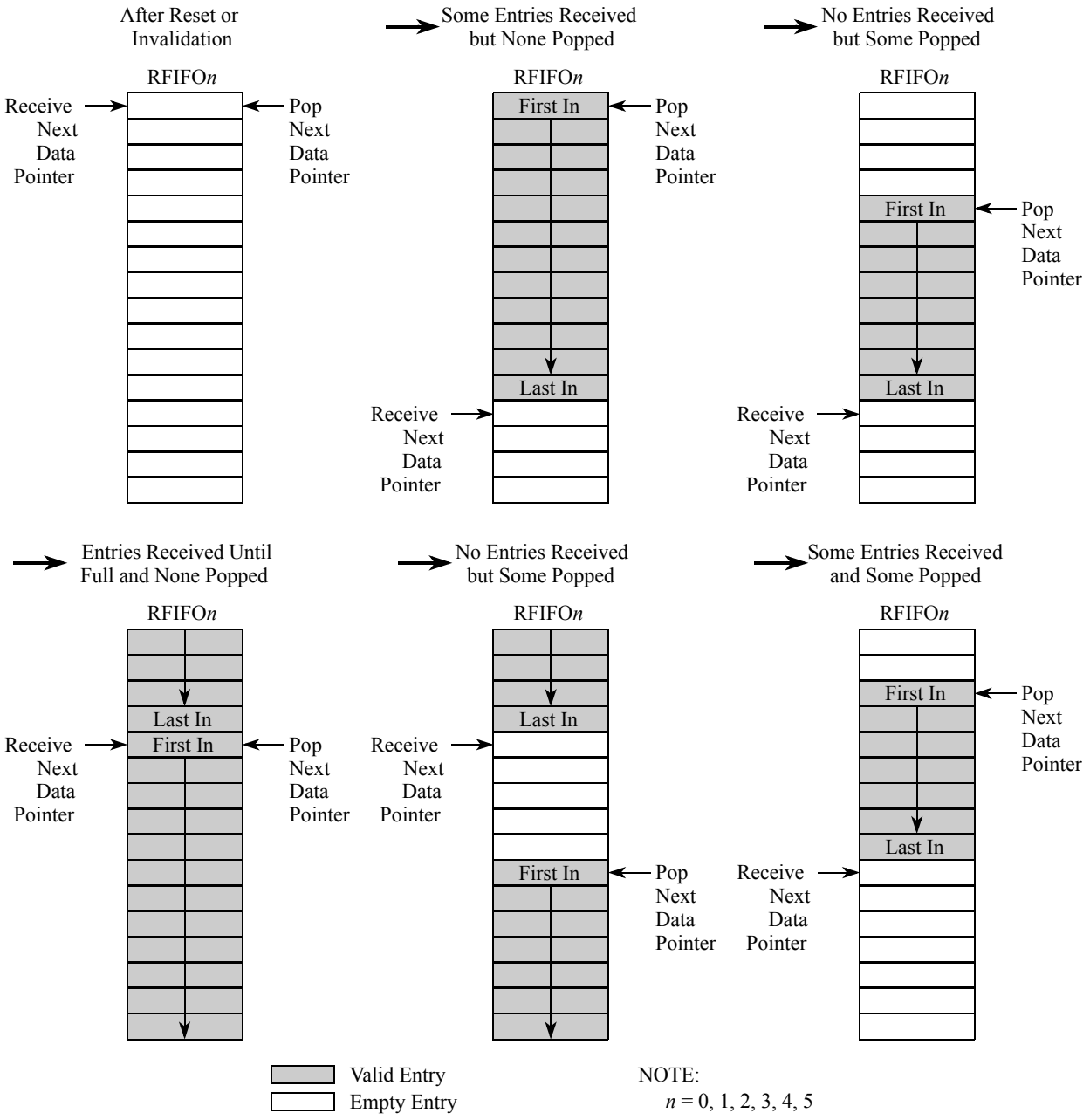


Figure 19-47. RFIFO Entry Pointer Example

19.4.4.2 Distributing Result Data into RFIFOs

Data to be moved into the RFIFOs can come from three sources: from ADC0, from ADC1, or from the external device. All result data comes with a MESSAGE_TAG field defining what should be done with the received data. The FIFO control unit decodes the MESSAGE_TAG field and:

- Stores the 16-bit data into the appropriate RFIFO if the MESSAGE_TAG indicates a valid RFIFO number or

- Ignores the data in case of a null or “reserved for customer use” MESSAGE_TAG

In general, received data is moved into RFIFOs as they become available, while an exception happens when multiple results from different sources become available at the same time. In that case, result data from ADC0 is processed first, result data from ADC1 is only processed after all ADC0 data is processed, and result data from the external device is only processed after all data from ADC0/1 is processed.

When time-stamped results return from the on-chip ADCs, the conversion result and the time stamp are always moved to the RFIFOs in consecutive clock cycles in order to guarantee they are always stored in consecutive RFIFO entries.

19.4.5 On-Chip ADC Configuration and Control

19.4.5.1 Enabling and Disabling the on-chip ADCs

The on-chip ADCs have an enable bit (ADC0_CR[ADC0_EN] and ADC1_CR[ADC1_EN], see Section 19.3.3.1) which allows the enabling of the ADCs only when necessary. When the enable bit for an ADC is negated, the clock input to that ADC is stopped. The ADCs are disabled out of reset - ADC0/1_EN bits are negated - to allow for their safe configuration. The ADC must only be configured when its enable bit is negated. Once the enable bit of an ADC is asserted, clock input is started, and the bias generator circuit is turned on. When the enable bits of both ADCs are negated, the bias circuit generator is stopped.

NOTE

Conversion commands sent to a disabled ADC are ignored by the ADC control hardware.

NOTE

NOTE An 8ms wait time from V_{DDA} power up to enabling ADC is required to pre-charge the external 100nf capacitor on REFBYPC. This time must be guaranteed by crystal startup time plus reset duration or the user. The ADC internal bias generator circuit will start up after 10us upon VRH/VRL power up and produces a stable/required bias current to the pre-charge circuit, but the current to the other analog circuits are disabled until ADCs are enabled. As soon as the ADCs are enabled, the bias currents to other analog circuits will be ready.

Because of previous design versions, the EQADC will always wait 120 ADC clocks before issuing the first conversion command following the enabling of one of on-chip ADCs, or the exiting of stop mode. There are two independent counters checking for this delay: one clocked by ADC0_CLK and another by ADC1_CLK. Conversion commands can start to be executed whenever one of these counters completes counting 120 ADC clocks.

19.4.5.2 ADC Clock and Conversion Speed

The clock input to the ADCs is defined by setting the ADC0_CR[ADC0_CLK_PS] and ADC1_CR[ADC1_CLK_PS] fields (see Section 19.3.3.1) The ADC0/1_CLK_PS field selects the clock divide factor by which the system clock will be divided as showed in Table 19-28. The ADC clock frequency is calculated as below and it must not exceed 12 MHz.

$$ADCClockFrequency = \frac{SystemClockFrequency(MHz)}{SystemClockDivideFactor}; (ADCClockFrequency \leq 12MHz)$$

Figure 19-48 depicts how the ADC clocks for ADC0 and ADC1 are generated.

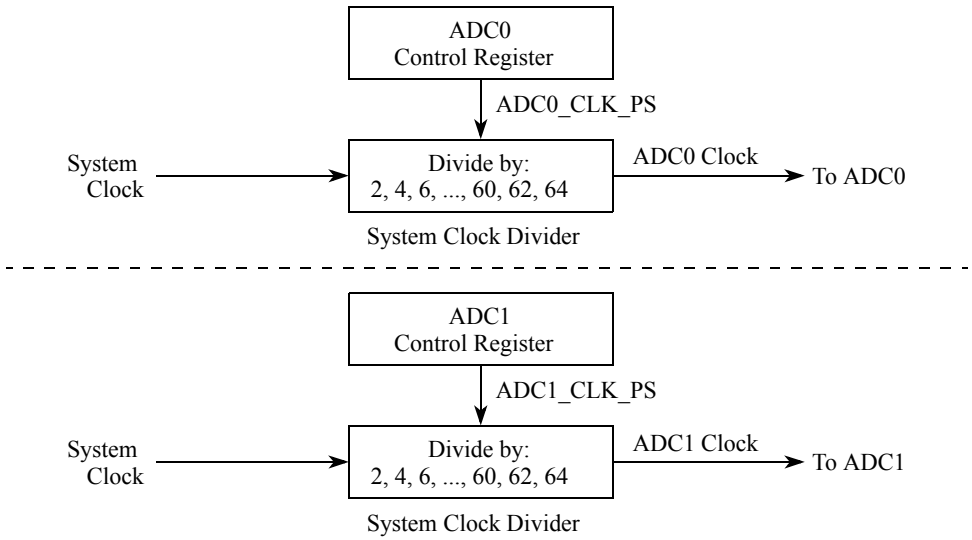


Figure 19-48. ADC0/1 Clock Generation

The ADC conversion speed (in kilosamples per second – ksamp/s)is calculated by the following formula. The number of sampling cycles is determined by the LST bits in the command message — see Section , " Conversion Command Message Format for On-Chip ADC Operation," — and it can take one of the following values: 2, 8, 64, or 128 ADC clock cycles. The number of AD conversion cycles is 13 for differential conversions and 14 for single-ended conversions. The maximum conversion speed is achieved when the ADC Clock frequency is set to its maximum (12Mhz) and the number of sampling cycles set to its minimum (2 cycles). The maximum conversion speed for differential and single-ended conversions are 800ksamp/s and 750ksamp/s, respectively.

$$ADCConversionSpeed = \frac{ADCClockFrequency(MHz)}{(NumberOfSamplingCycles + NumberOfADConversionCycles)}$$

Table 19-47 shows an example of how the ADC0/1_CLK_PS can be set when using a 120 MHz system clock and the corresponding conversion speeds for all possible ADC clock frequencies. The table also shows that according to the system clock frequency, certain clock divide factors are invalid (2, 4, 6, 8 clock divide factors in the example) since their use would result in a ADC clock frequency higher than the maximum one supported by the ADC. ADC clock frequency must not exceed 12 Mhz.

Table 19-47. ADC Clock Configuration Example (System Clock Frequency = 120 MHz)

ADC0/1_CLK_PS[0:4]	System Clock Divide Factor	ADC Clock in MHz (System Clock = 120MHz)	Differential Conversion Speed with Default Sampling Time (13 + 2 cycles) in ksamp/s	Single-Ended Conversion Speed with Default Sampling Time (14 + 2 cycles) in ksamp/s
0b00000	2	N/A	N/A	N/A
0b00001	4	N/A	N/A	N/A
0b00010	6	N/A	N/A	N/A
0b00011	8	N/A	N/A	N/A
0b00100	10	12.0	800	750
0b00101	12	10.0	667	625
0b00110	14	8.57	571	536
0b00111	16	7.5	500	469
0b01000	18	6.67	444	417
0b01001	20	6.0	400	375
0b01010	22	5.45	364	341
0b01011	24	5.0	333	313
0b01100	26	4.62	308	288
0b01101	28	4.29	286	268
0b01110	30	4.0	267	250
0b01111	32	3.75	250	234
0b10000	34	3.53	235	221
0b10001	36	3.33	222	208
0b10010	38	3.16	211	197
0b10011	40	3.0	200	188
0b10100	42	2.86	190	179
0b10101	44	2.73	182	170
0b10110	46	2.61	174	163
0b10111	48	2.5	167	156
0b11000	50	2.4	160	150
0b11001	52	2.31	154	144
0b11010	54	2.22	148	139
0b11011	56	2.14	143	134
0b11100	58	2.07	138	129

Table 19-47. ADC Clock Configuration Example (continued)
(System Clock Frequency = 120 MHz)

ADC0/1_CLK_PS[0:4]	System Clock Divide Factor	ADC Clock in MHz (System Clock = 120MHz)	Differential Conversion Speed with Default Sampling Time (13 + 2 cycles) in ksamp/s	Single-Ended Conversion Speed with Default Sampling Time (14 + 2 cycles) in ksamp/s
0b11101	60	2.0	133	125
0b11110	62	1.94	129	121
0b11111	64	1.88	125	117

19.4.5.3 Time Stamp Feature

The on-chip ADCs can provide a time stamp for the conversions they execute. A time stamp is the value of the time base counter latched when the eQADC detects the end of the analog input voltage sampling. A time stamp for a conversion command is requested by setting the TSR bit in the corresponding command. When TSR is negated, that is a time stamp is not requested, the ADC returns a single result message containing the conversion result. When TSR is asserted, that is a time stamp is requested, the ADC returns two result messages; one containing the conversion result, and another containing the time stamp for that conversion. The result messages are sent in this order to the RFIFOs and both messages are sent to the same RFIFO as specified in the MESSAGE_TAG field of the executed conversion command.

The time base counter is a 16-bit up counter and wraps after reaching 0xFFFF. It is disabled after reset and it is enabled according to the setting of ADC_TSCR[TBC_CLK_PS] field (see Section 19.3.3.2). TBC_CLK_PS defines if the counter is enabled or disabled, and, if enabled, at what frequency it is incremented. The time stamps are returned regardless of whether the time base counter is enabled or disabled. The time base counter can be reset by writing 0x0000 to the ADC_TBCR (Section 19.3.3.3) with a write configuration command.

19.4.5.4 ADC Calibration Feature

19.4.5.4.1 Calibration Overview

The eQADC provides a calibration scheme to remove the effects of gain and offset errors from the results generated by the on-chip ADCs. Only results generated by the on-chip ADCs are calibrated. The results generated by ADCs on the external device are directly sent to RFIFOs unchanged. The main component of calibration hardware is a multiply-and-accumulate (MAC) unit, one per on-chip ADC, that is used to calculate the following transfer function which relates a calibrated result to a raw, uncalibrated one.

```
CAL_RES = GCC * RAW_RES + OCC + 2;
```

where:

- CAL_RES is the calibrated result corresponding the input voltage V_i.
- GCC is the gain calibration constant.
- RAW_RES is the raw, uncalibrated result corresponding to an specific input voltage V_i.
- OCC is the offset calibration constant.
- The addition of two reduces the maximum quantization error of the ADC. See Section 19.5.6.3, “Quantization Error Reduction During Calibration.”

Calibration constants GCC and OCC are determined by taking two samples of known reference voltages and using these samples to calculate their values. For details and an example about how to calculate the calibration constants and use them in result calibration refer to Section 19.5.6, "ADC Result Calibration." Once calculated, GCC is stored in ADC0_GCCR and ADC1_GCCR (see Section 19.3.3.4) and OCC in ADC0_OCCR and ADC1_OCCR (see Section 19.3.3.5) from where their values are fed to the MAC unit. Since the analog characteristics of each on-chip ADC differs, each ADC has an independent pair of calibration constants.

A conversion result is calibrated according to the status of CAL bit in the command that initiated the conversion. If the CAL bit is asserted, the eQADC will automatically calculate the calibrated result before sending the result to the appropriate RFIFO. If the CAL bit is negated, the result is not calibrated, it bypasses the calibration hardware, and is directly sent to the appropriate RFIFO.

19.4.5.4.2 MAC Unit and Operand Data Format

The MAC unit diagram is shown in Figure 19-49. Each on-chip ADC has a separate MAC unit to calibrate its conversion results.

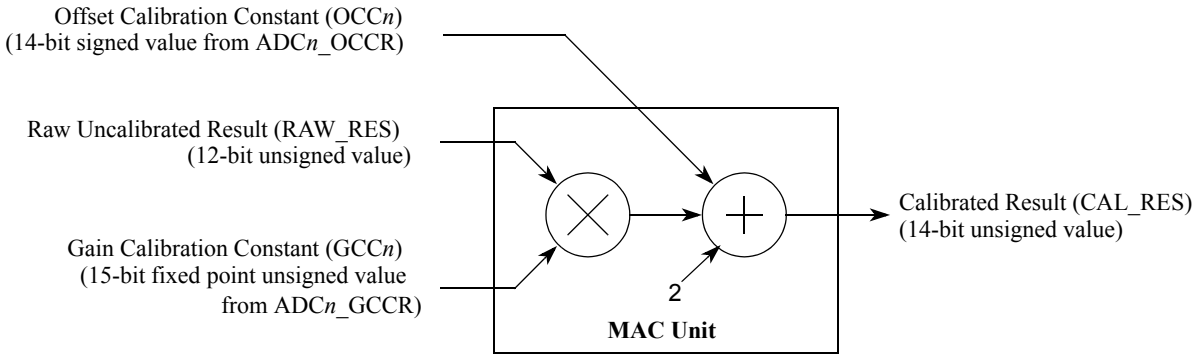


Figure 19-49. MAC Unit Diagram

The OCC*n* operand is a 14-bit signed value and it is the upper 14 bits of the value stored in ADC0_OCCR and ADC1_OCCR. The RAW_RES operand is the raw uncalibrated result, and it is a direct output from the on-chip ADCs.

The GCC*n* operand is a 15-bit fixed point unsigned value, and it is the upper 15 bits of the value stored in ADC0_GCCR and ADC1_GCCR. The GCC is expressed in the *GCC_INT.GCC_FRAC* binary format. The integer part of the GCC (GCC_INT = GCC[1]) contains a single binary digit while its fractional part (GCC_FRAC = GCC[2:15]) contains 14 bits. See Figure 19-50 for more information. The gain constant equivalent decimal value ranges from 0 to 1.999938..., as shown in Table 19-49. Two is always added to the MAC output: see Section 19.5.6.3, "Quantization Error Reduction During Calibration. CAL_RES output is the calibrated result, and it is a 14-bit unsigned value. CAL_RES is truncated to 0x3FFF, in case of a overflow, and to 0x0000, in case of an underflow.

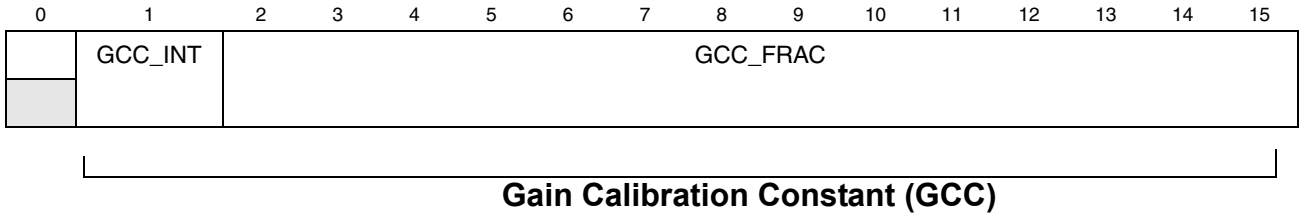


Figure 19-50. Gain Calibration Constant Format

Table 19-48. Gain Calibration Constant Format Field Descriptions

Bits	Name	Description
0	—	Reserved
1	GCC_INT [0]	Integer part of the gain calibration constant for ADC*n*. GCC_INT is the integer part of the gain calibration constant (GCC) for ADC0/1.
2–15	GCC_FRAC [1:14]	Fractional part of the gain calibration constant for ADC*n*. GCC_FRAC is the fractional part of the gain calibration constant (GCC) for ADC*n*. GCC_FRAC can expresses decimal values ranging from 0 to 0.999938...

Table 19-49. Correspondence between Binary and Decimal Representations of the Gain Constant

Gain Constant (GCC_INT.GCC_FRAC binary format)	Corresponding Decimal Value
0.0000_0000_0000_00	0
...	...
0.1000_0000_0000_00	0.5
...	...
0.1111_1111_1111_11	0.999938...
1.0000_0000_0000_00	1
...	...
1.1100_0000_0000_00	1.75
...	...
1.1111_1111_1111_11	1.999938...

19.4.5.5 ADC Control Logic Overview and Command Execution

Figure 19-51 shows the basic logic blocks involved in the ADC control and how they interact. CFIFOs/RFIFOs interact with ADC command/result message return logic through the FIFO control unit. The EB and BN bits in the command message uniquely identify the ADC to which a command should be sent. The FIFO control unit decodes these bits and sends the ADC command to the proper ADC. Other blocks of logic are the result format and calibration submodule, the time stamp logic, and the MUX control logic.

The result format and calibration submodule formats the returning data into result messages and sends them to the RFIFOs. The returning data can be data read from an ADC register, a conversion result, or a time stamp. The formatting and calibration of conversion results also take place inside this submodule.

The time stamp logic latches the value of the time base counter when detecting the end of the analog input voltage sampling, and sends it to the result format and calibration submodule as time stamp information.

The MUX control logic generates the proper MUX control signals and, when the ADC0/1_EMUX bits are asserted, the MA signals based on the channel numbers extracted from the ADC Command.

ADC commands are stored in the ADC command buffers (2 entries) as they come in and they are executed on a first-in-first-out basis. After the execution of a command in ENTRY1 finishes, all commands are shifted one entry. After the shift, ENTRY0 is always empty and ready to receive a new command. Execution of configuration commands only starts when they reach ENTRY1. Consecutive conversion commands are pipelined, and their execution can start while in ENTRY0. This is explained below.

A/D conversion accuracy can be affected by the settling time of the input channel multiplexers. Some time is required for the channel multiplexer's internal capacitances to settle after the channel number is changed. If the time prior to and during sampling is not long enough to permit this settling, then the voltage on the sample capacitors will not accurately represent the voltage to be read. This is a problem in particular when external muxes are used.

To maximize settling time, when a conversion command is in buffer ENTRY1 and another conversion command is identified in ENTRY0, then the channel number of ENTRY0 is sent to the *MUX control logic* half an ADC clock before the start of the sampling phase of the command in ENTRY0. This pipelining of sample and settling phase is shown in Figure 19-52(b).

This provides more accurate sampling, which is specially important for applications that require high conversion speeds, i.e., with the ADC running at maximum clock frequency and with the analog input voltage sampling time set to a minimum (2 ADC clock cycles). In this case the short sampling time may not allow the multiplexers to completely settle. The second advantage of pipelining conversion commands is to provide equal conversion intervals even though the sample time increases on second and subsequent conversions. See Figure 19-52. This is important for any digital signal process application.

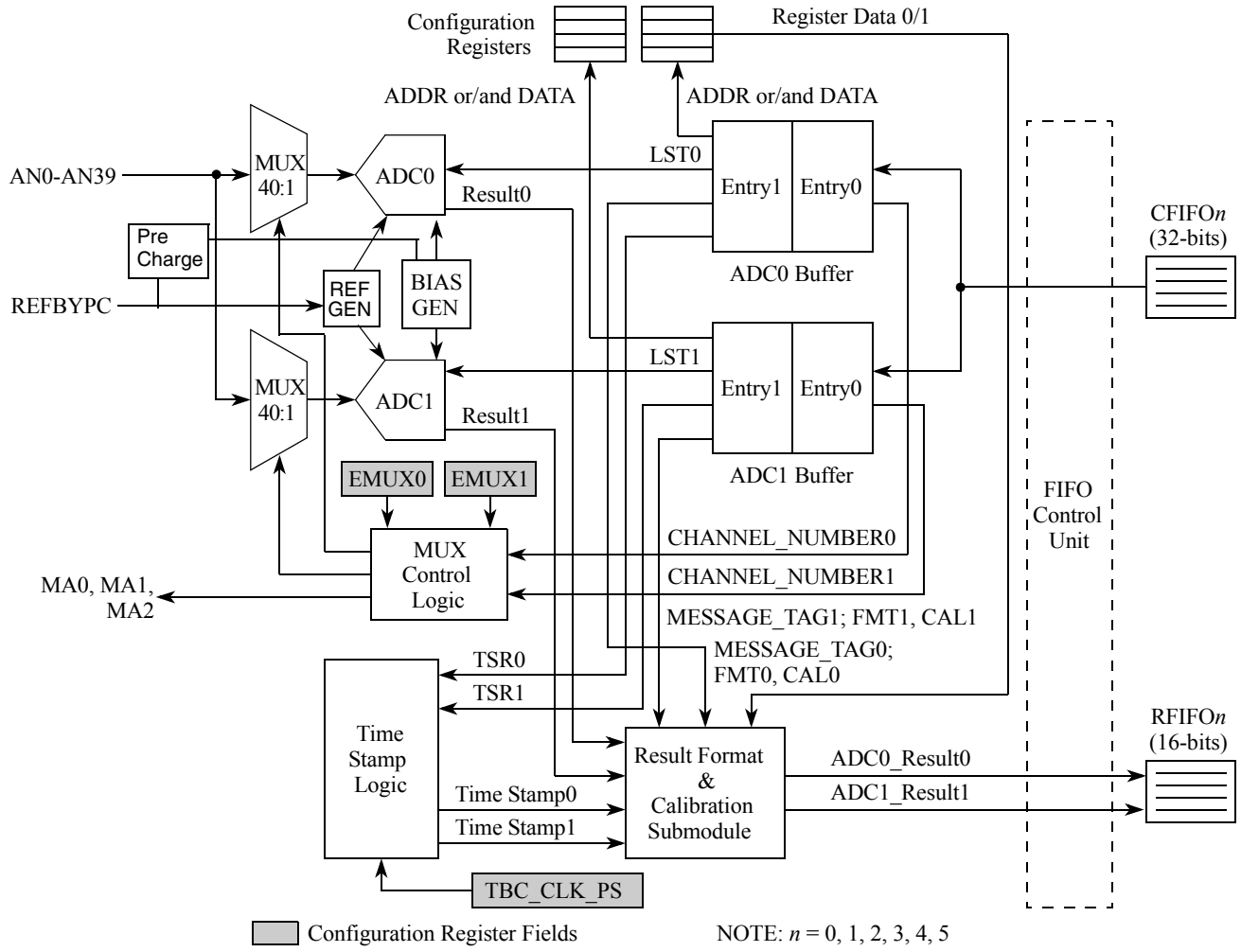


Figure 19-51. On-Chip ADC Control Scheme

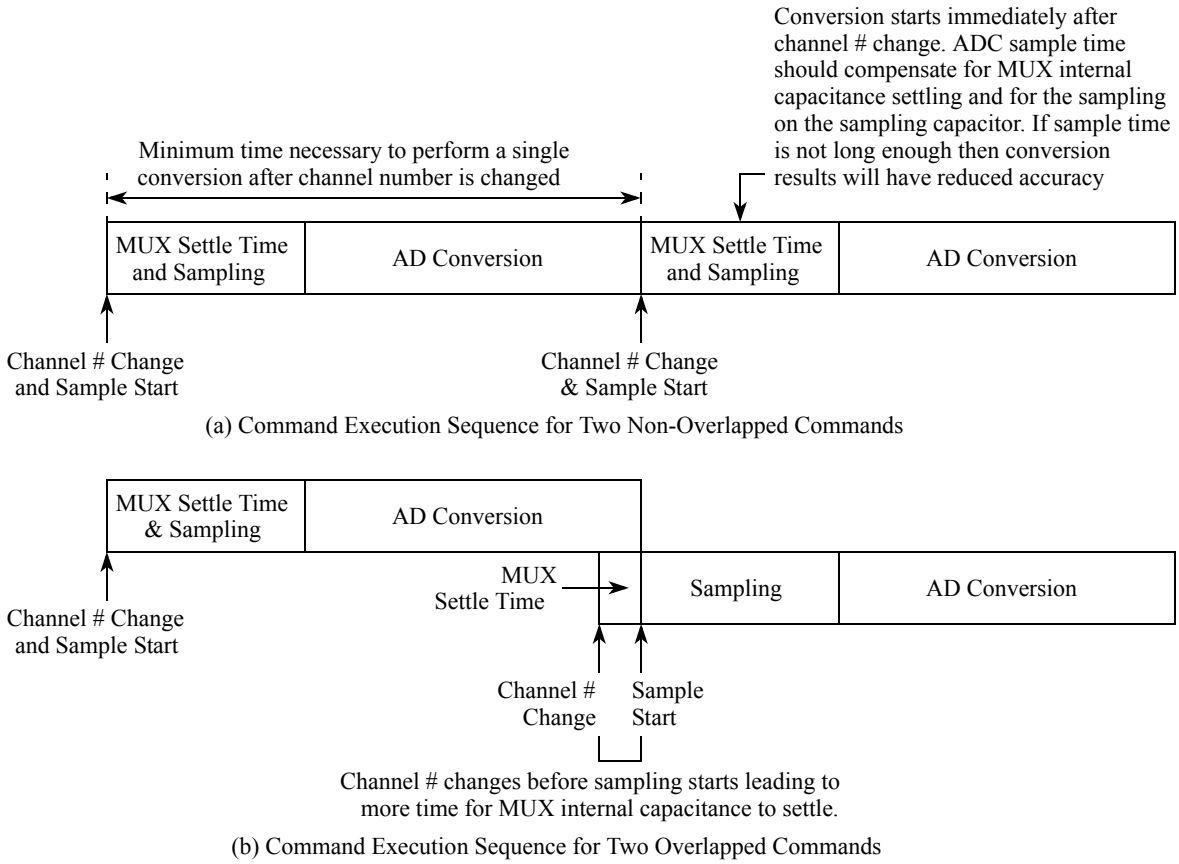


Figure 19-52. Overlapping Consecutive Conversion Commands

19.4.6 Internal/External Multiplexing

19.4.6.1 Channel Assignment

The internal analog multiplexers select one of the 40 analog input pins for conversion, based on the CHANNEL_NUMBER field of a Command Message. The analog input pin channel number assignments and the pin definitions vary depending on how the ADC0/1_EMUX are configured. Allowed combinations of ADC0/1_EMUX bits are shown in Table 19-50 together with references to tables indicating how CHANNEL_NUMBER field of each conversion command must be set to avoid channel selection conflicts.

During differential conversions the analog multiplexer passes differential signals to both the positive and negative terminals of the ADC. The differential conversions can only be initiated on four channels: DAN0, DAN1, DAN2, and DAN3. Refer to Table 19-51 and Figure 19-52 for the channel numbers used to select differential conversions.

Table 19-50. ADC*n*_EMUX Bits Combinations

ADC0_EMUX	ADC1_EMUX	CHANNEL_NUMBER should be set as in	
		ADC0	ADC1
0	0	Refer to Table 19-51	Refer to Table 19-51
0	1	Refer to Table 19-51	Refer to Figure 19-52
1	0	Refer to Figure 19-52	Refer to Table 19-51
1	1	Reserved [1]	

[1] ADC0_EMUX and ADC1_EMUX must not be asserted at the same time.

Table 19-51 shows the channel number assignments for the non-multiplexed mode. The 40 single-ended channels and 4 differential pairs are shared between the two ADCs.

Table 19-51. Non-multiplexed Channel Assignments[1]

Input Pins			Channel Number in CHANNEL_NUMBER Field	
Analog Pin Name	Other Functions	Conversion Type	Binary	Decimal
AN0 to AN39		Single-ended	0000_0000 to 0010_0111	0 to 39
VRH		Single-ended	0010_1000	40
VRL		Single-ended	0010_1001	41
	(VRH - VRL)/2 see footnote[2]	Single-ended	0010_1010	42
	75% x (VRH - VRL)	Single-ended	0010_1011	43
	25% x (VRH - VRL)	Single-ended	0010_1100	44
Reserved			0010_1101 to 0101_1111	45 to 95
DAN0+ and DAN0- DAN1+ and DAN1- DAN2+ and DAN2- DAN3+ and DAN3-		Differential Differential Differential Differential	0110_0000 0110_0001 0110_0010 0110_0011	96 97 98 99
Reserved			0110_0100 to 1111_1111	100 to 255

[1] The two on-chip ADCs can access the same analog input pins but simultaneous conversions are not allowed. Also, when one ADC is performing a differential conversion on a pair of pins, the other ADC must not access either of these two pins as single-ended channels.

[2] This equation only applies before calibration. After calibration, the 50% reference point will actually return approximately 20mV lower than the expected 50% of the difference between the High Reference Voltage (VRH) and the Low Reference Voltage (VRL). For calibration of the ADC only the 25% and 75% points should be used as described in Section 19.5.6.1, "MAC Configuration Procedure"

Figure 19-52 shows the channel number assignments for multiplexed mode. The ADC with the ADC*n*_EMUX bit asserted can access 4 differential pairs, 39 single-ended, and, at most, 32 externally multiplexed channels. Refer to Section 19.4.6.2, "External Multiplexing," for a detailed explanation about how external multiplexing can be achieved.

Table 19-52. Multiplexed Channel Assignments[1]

Input Pins			Channel Number in CHANNEL_NUMBER Field	
Analog Pin Name	**Other Functions**	**Conversion Type**	**Binary**	**Decimal**
AN0 to AN7		Single-ended	0000_0000 to 0000_0111	0 to 7
Reserved			0000_1000 to 0000_1011	8 to 11
AN12 to AN39		Single-ended	0000_1100 to 0010_0111	12 to 39
VRH		Single-ended	0010_1000	40
VRL		Single-ended	0010_1001	41
	(VRH–VRL)/2	Single-ended	0010_1010	42
	75% x (VRH–VRL)	Single-ended	0010_1011	43
	25% x (VRH–VRL)	Single-ended	0010_1100	44
Reserved			0010_1101 to 0011_1111	45 to 63
ANW ANX ANY ANZ	— — — —	Single-ended Single-ended Single-ended Single-ended	0100_0xxx 0100_1xxx 0101_0xxx 0101_1xxx	64 to 71 72 to 79 80 to 87 88 to 95
DAN0+ and DAN0- DAN1+ and DAN1- DAN2+ and DAN2- DAN3+ and DAN3-		Differential Differential Differential Differential	0110_0000 0110_0001 0110_0010 0110_0011	96 97 98 99
Reserved			0011_0100 to 1111_1111	100 to 255

[1]The two on-chip ADCs can access the same analog input pins but simultaneous conversions are not allowed. Also, when one ADC is performing a differential conversion on a pair of pins, the other ADC must not access either of these two pins as single-ended channels.

19.4.6.2 External Multiplexing

The eQADC can use from one to four external multiplexers to expand the number of analog signals that may be converted. Up to 32 analog channels can be converted through external multiplexer selection. The externally multiplexed channels are automatically selected by the CHANNEL_NUMBER field of a command message, in the same way done with internally multiplexed channels. The software selects the external multiplexed mode by setting the ADC0/1_EMUX bit in either ADC0_CR or ADC1_CR depending on which ADC will perform the conversion. Figure 19-52 shows the channel number assignments for the multiplexed mode. There are 4 differential pairs, 40 single-ended, and, at most, 32 externally multiplexed channels that can be selected. Only one ADC can have its ADC0/1_EMUX bit asserted at a time.

Figure 19-53 shows the maximum configuration of four external multiplexer chips connected to the eQADC. The external multiplexer chip selects one of eight analog inputs and connects it to a single analog output, which is fed to a specific input of the eQADC. The eQADC provides three multiplexed address signals, MA0, MA1, and MA2, to select one of eight inputs. These three multiplexed address signals are connected to all four external multiplexer chips. The analog output of the four multiplex chips are each connected to four separate eQADC inputs, ANW, ANX, ANY, and ANZ. The MA pins correspond to the

three least significant bits of the channel number that selects ANW, ANX, ANY, and ANZ with MA0 being the most significant bit - See Table 19-53.

Table 19-53. Encoding of MA Pins[1]

Channel Number selecting ANW, ANX, ANY, ANZ (decimal)				MA0	MA1	MA2
ANW	ANX	ANY	ANZ			
64	72	80	88	0	0	0
65	73	81	89	0	0	1
66	74	82	90	0	1	0
67	75	83	91	0	1	1
68	76	84	92	1	0	0
69	77	85	93	1	0	1
70	78	86	94	1	1	0
71	79	87	95	1	1	1

[1] 0 means pin is driven LOW and 1 that pin is driven HIGH.

When the external multiplexed mode is selected for either ADC, the eQADC automatically creates the MA output signals from CHANNEL_NUMBER field of a command message. The eQADC also converts the proper input channel (ANW, ANX, ANY, and ANZ) by interpreting the CHANNEL_NUMBER field. As a result, up to 32 externally multiplexed channels appear to the conversion queues as directly connected signals.

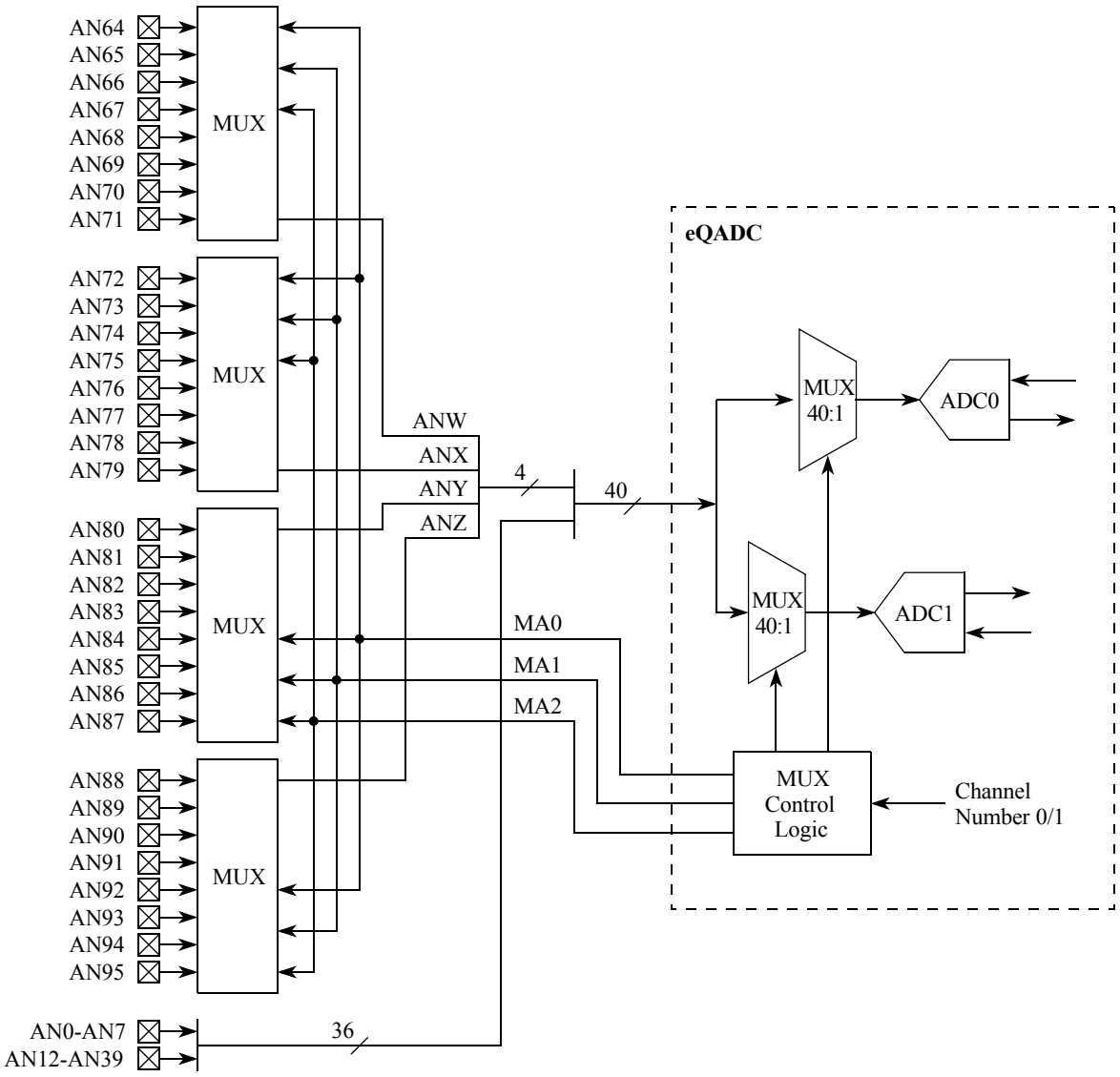


Figure 19-53. Example of External Multiplexing

19.4.7 eQADC eDMA/Interrupt Request

Table 19-54 lists methods to generate interrupt requests in the eQADC queuing control and triggering control. The eDMA/interrupt request select bits and the eDMA/interrupt enable bits are described in Section 19.3.2.7, "eQADC Interrupt and eDMA Control Registers 0–5 (EQADC_IDCRn)," and the interrupt flag bits are described in Section 19.3.2.8, "eQADC FIFO and Interrupt Status Registers 0–5 (EQADC_FISRn)." Table 19-54 depicts all interrupts and eDMA requests generated by the eQADC.

Table 19-54. eQADC FIFO Interrupt Summary[1]

Interrupt	Condition	Clearing Mechanism
Non Coherency Interrupt	NCIE*n* = 1 NCF*n* = 1	Clear NCF*n* bit by writing a 1 to the bit.
Trigger Overrun Interrupt[2]	TORIE*n* = 1 TORF*n* =1	Clear TORF*n* bit by writing a 1 to the bit.
Pause Interrupt	PIE*n* = 1 PF*n* =1	Clear PF*n* bit by writing a 1 to the bit.
End of Queue Interrupt	EOQIE*n* = 1 EOQF*n* = 1	Clear EOQF*n* bit by writing a 1 to the bit.
Command FIFO Underflow Interrupt[2]	CFUIE*n* = 1 CFUF*n* = 1	Clear CFUF*n* bit by writing a 1 to the bit.
Command FIFO Fill Interrupt	CFFE*n* = 1 CFFS*n* = 0 CFFF*n* = 1	Clear CFFF*n* bit by writing a 1 to the bit.
Result FIFO Overflow Interrupt[2]	RFOIE*n* = 1 RFOF*n* = 1	Clear RFOF*n* bit by writing a 1 to the bit.
Result FIFO Drain Interrupt	RFDE*n* = 1 RFDS*n* = 0 RFDF*n* = 1	Clear RFDF*n* bit by writing a 1 to the bit.

1 For details refer to Section 19.3.2.8, “eQADC FIFO and Interrupt Status Registers 0–5 (EQADC_FISRn),” and Section 19.3.2.7, “eQADC Interrupt and eDMA Control Registers 0–5 (EQADC_IDCRn).”

2 Apart from generating an independent interrupt request for when a RFIFO overflow interrupt, a CFIFO underflow interrupt, and a CFIFO trigger overrun interrupt occurs, the eQADC also provides a combined interrupt request at which these requests from ALL CFIFOs are ORed. Refer to Figure 19-54 for details.

Table 19-55 describes a list of methods to generate eDMA requests by the eQADC.

Table 19-55. eQADC FIFO eDMA Summary[1]

eDMA Request	Condition	Clearing Mechanism
Result FIFO Drain eDMA Request	RFDE*n* = 1 RFDS*n* = 1 RFDF*n* = 1	The eQADC automatically clears the RFDF*n* when RFIFO*n* becomes empty. Writing 1 to the RFDF*n* bit is not allowed while RDFS = 1.
Command FIFO Fill eDMA Request	CFFE*n* = 1 CFFS*n* = 1 CFFF*n* = 1	The eQADC automatically clears the CFFF*n* when CFIFO*n* becomes full. Writing 1 to the CFFF*n* bit is not allowed while CFDS = 1.

1 For details refer to Section 19.3.2.8, “eQADC FIFO and Interrupt Status Registers 0–5 (EQADC_FISRn),” and Section 19.3.2.7, “eQADC Interrupt and eDMA Control Registers 0–5 (EQADC_IDCRn).”

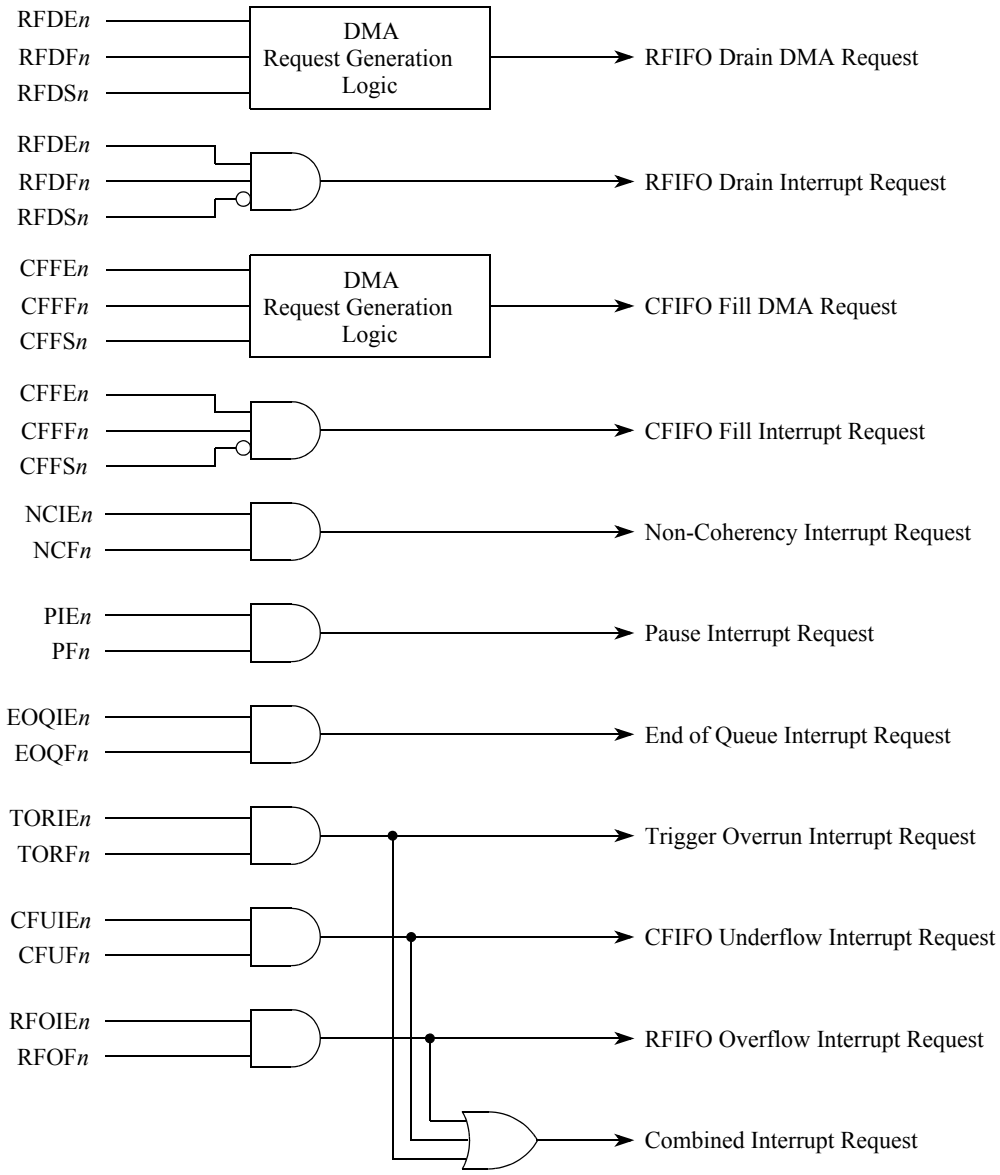


Figure 19-54. eQADC eDMA and Interrupt Requests

19.4.8 eQADC Synchronous Serial Interface (SSI) Submodule

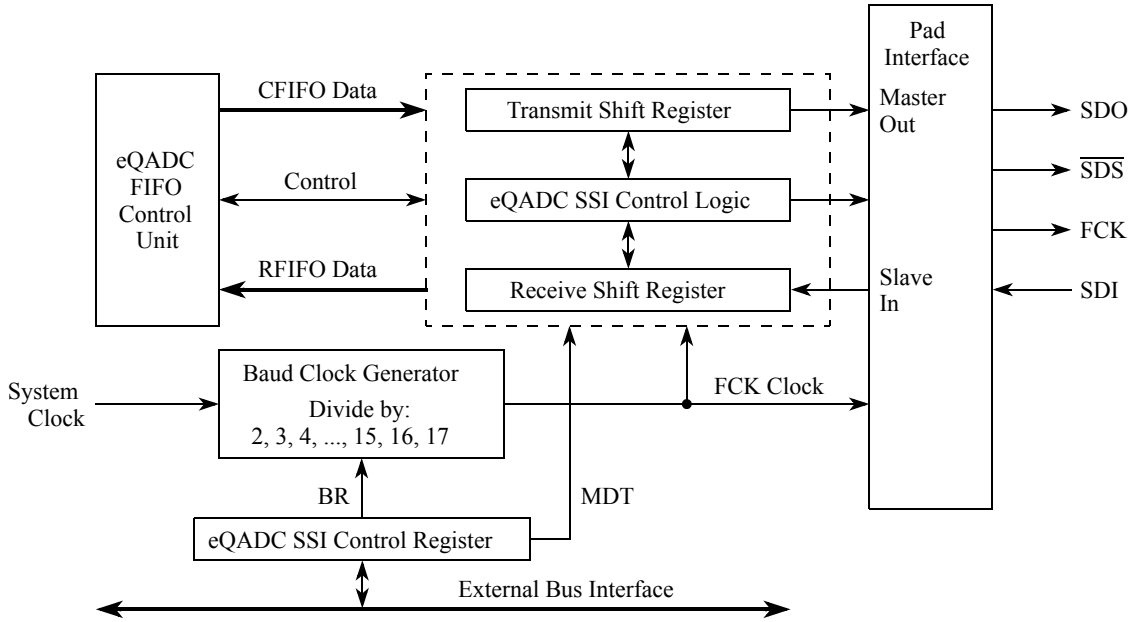


Figure 19-55. eQADC Synchronous Serial Interface Block Diagram

The eQADC SSI protocol allows for a full duplex, synchronous, serial communication between the eQADC and a single external device. Figure 19-55 shows the different components inside the eQADC SSI. The eQADC SSI submodule on the eQADC is always configured as a master. The eQADC SSI has four associated port pins:

- Free running clock (FCK)
- Serial data select ($\overline{\text{SDS}}$)
- Serial data in (SDI)
- Serial data out (SDO)

The FCK clock signal times the shifting and sampling of the two serial data signals and it is free running between transmissions, allowing it to be used as the clock for the external device. The $\overline{\text{SDS}}$ signal will be asserted to indicate the start of a transmission, and negated to indicate the end or the abort of a transmission. SDI is the master serial data input and SDO the master serial data output.

The eQADC SSI submodule is enabled by setting the EQADC_MCR[ESSIE] (see Section 19.3.2.1). When enabled, the eQADC SSI can be optionally capable of starting serial transmissions. When serial transmissions are disabled (ESSIE set to 0b10), no data will be transmitted to the external device but FCK will be free-running. This operation mode permits the control of the timing of the first serial transmission, and can be used to avoid the transmission of data to an unstable external device, for example, a device that is not fully reset. This mode of operation is specially important for the reset procedure of an external device that uses the FCK as its main clock.

The main elements of the eQADC SSI are the shift registers. The 26-bit transmit shift register in the master and 26-bit receive shift register in the slave are linked by the SDO pin. In a similar way, the 26-bit transmit shift register in the slave and 26-bit receive shift register in the master are linked by the SDI pin. See Figure 19-56. When a data transmission operation is performed, data in the transmit registers is serially shifted twenty-six bit positions into the receive registers by the FCK clock from the master; data is exchanged between the master and the slave. Data in the master transmit shift register in the beginning of

a transmission operation becomes the output data for the slave, and data in the master receive shift register after a transmission operation is the input data from the slave.

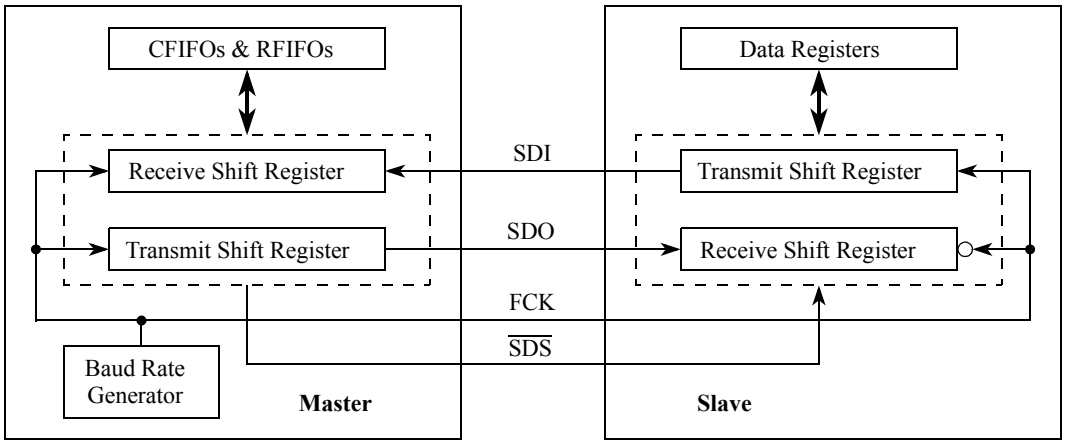


Figure 19-56. Full Duplex Pin Connection

19.4.8.1 eQADC SSI Data Transmission Protocol

Figure 19-57 shows the timing of an eQADC SSI transmission operation. The main characteristics of this protocol are the following:

- FCK is free running, it does not stop between data transmissions. FCK will be driven low:
  - When the serial interface is disabled
  - In stop/debug mode
  - Immediately after reset
- Frame size is fixed to 26 bits.
- Msb bit is always transmitted first.
- Master drives data on the positive edge of FCK and latches incoming data on the next positive edge of FCK.
- Slave drives data on the positive edge of FCK and latches incoming data on the negative edge of FCK.

Master initiates a data transmission by driving $\overline{SDS}$ low, and its msb bit on SDO on the positive edge of FCK. Once an asserted $\overline{SDS}$ is detected, the slave shifts its data out, one bit at a time, on every FCK positive edge. Both the master and the slave drive new data on the serial lines on every FCK positive edge. This process continues until all the initial 26-bits in the master shift register are moved into the slave shift register. t_{DT} is the delay between two consecutive serial transmissions, time during which $\overline{SDS}$ is negated. When ready to start of the next transmission, the slave must drive the msb bit of the message on every positive edge of FCK regardless of the state of the $\overline{SDS}$ signal. On the next positive edge, the second bit of the message is conditionally driven according to if an asserted $\overline{SDS}$ was detected by the slave on the preceding FCK negative edge. This is an important requisite since the $\overline{SDS}$ and the FCK are not synchronous. The $\overline{SDS}$ signal is not generated by FCK, rather both are generated by the system clock, so that it is not guaranteed that FCK edges will precede $\overline{SDS}$ edges. While $\overline{SDS}$ is negated, the slave continuously drives its msb bit on every positive edge of FCK until it detects an asserted $\overline{SDS}$ on the immediately next FCK negative edge. See Figure 19-58 for three situations showing how the slave should behave according to when $\overline{SDS}$ is asserted.

NOTE

On the master, the FCK is not used as a clock. Although, the eQADC SSI behavior is described in terms of the FCK positive and negative edges, all eQADC SSI related signals (SDI, SDS, SDO, and FCK) are synchronized by the system clock on the master side. There are no restrictions regarding the use of the FCK as a clock on the slave device.

19.4.8.1.1 Abort Feature

The master indicates it is aborting the current transfer by negating $\overline{SDS}$ before the whole data frame has being shifted out, that is the 26th bit of data being transferred has not being shifted out. The eQADC ignores the incompletely received message. The eQADC re-sends the aborted message whenever the corresponding CFIFO becomes again the highest priority CFIFO with commands bound for an external command buffer that is not full. Refer to Section 19.4.3.2, "CFIFO Prioritization and Command Transfer," for more information on aborts and CFIFO priority.

19.4.8.2 Baud Clock Generation

As shown in Figure 19-55, the baud clock generator divides the system clock to produce the baud clock. The EQADC_SSICR[BR] field (see Section 19.3.2.12) selects the system clock divide factor as in Table 19-21. [1]

$$\text{BaudClockFrequency} = \frac{\text{SystemClockFrequency(MHz)}}{\text{SystemClockDivideFactor}}$$

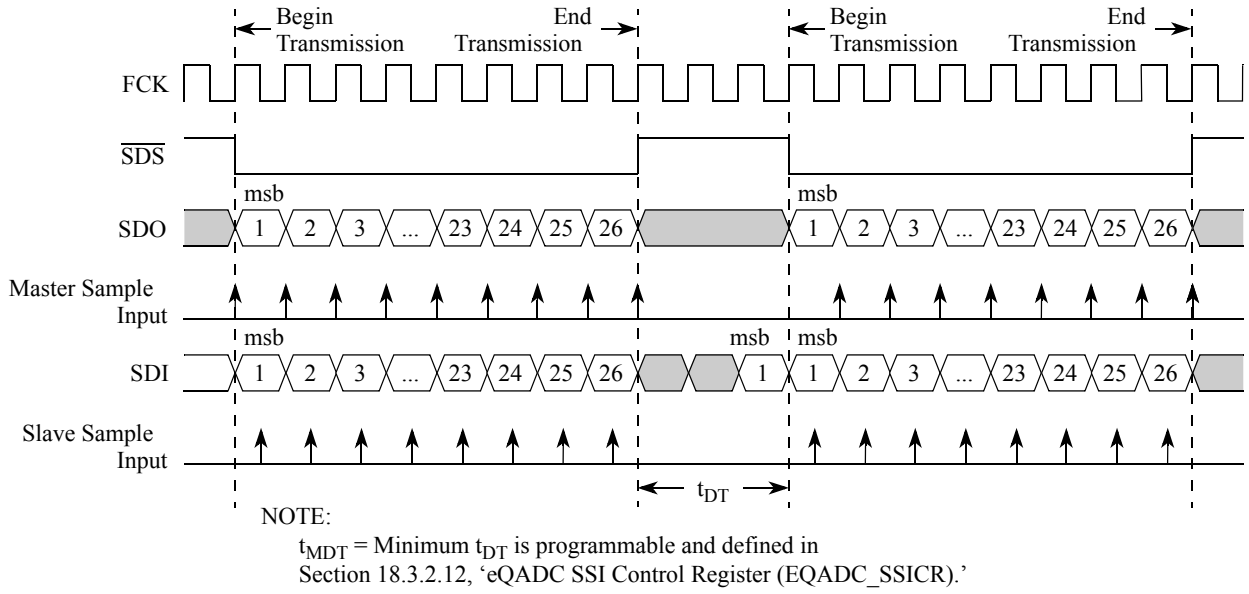


Figure 19-57. Synchronous Serial Interface Protocol Timing

1.Maximum FCK frequency is highly dependable on track delays, master pad delays, and slave pad delays.

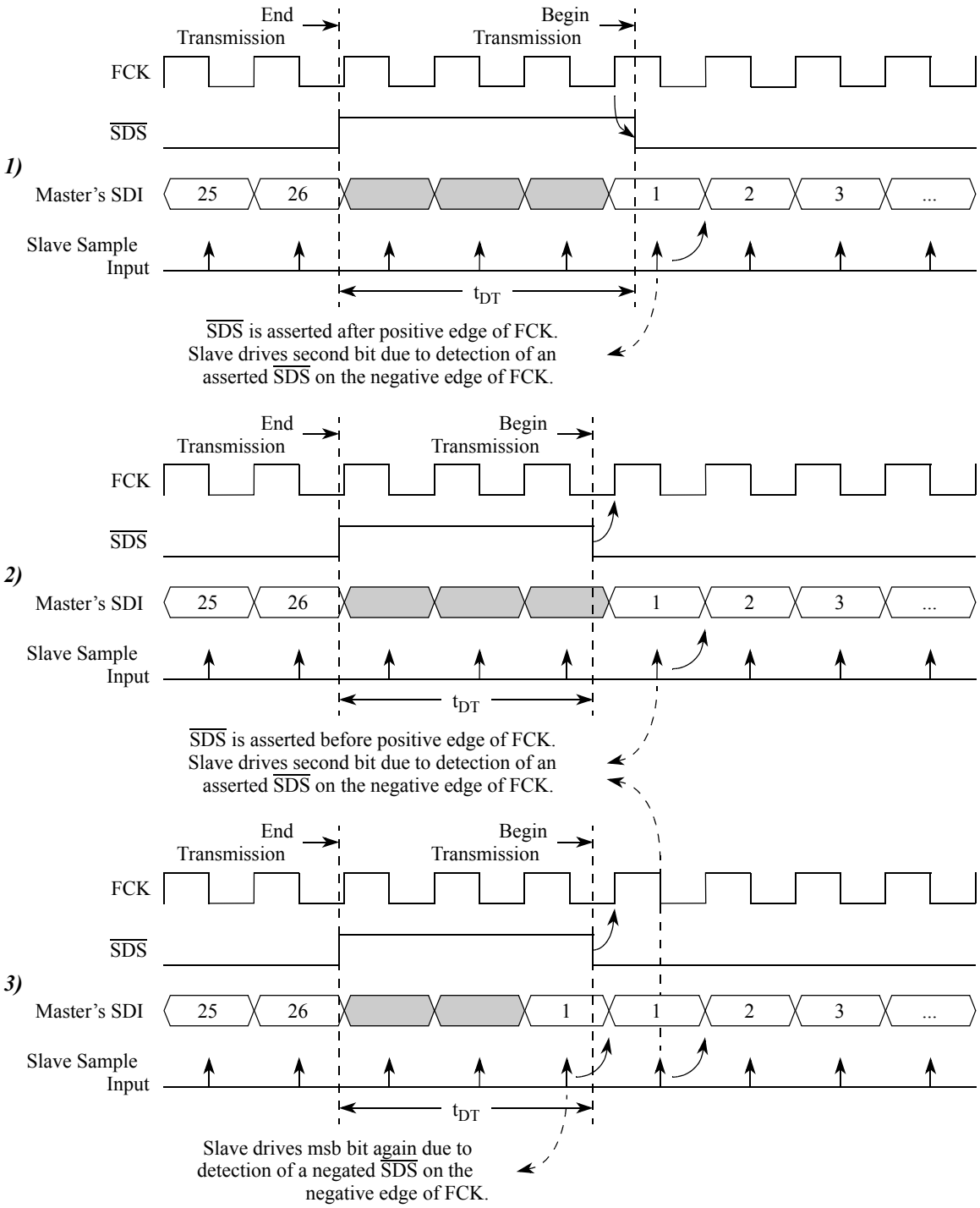


Figure 19-58. Slave Driving the msb and Consecutive Bits in a Data Transmission

19.4.9 Analog Submodule

19.4.9.1 Reference Bypass

The reference bypass capacitor (REFBYPC) signal requires a 100 nF capacitor connected to VRL to filter noise on the internal reference used by the ADC.

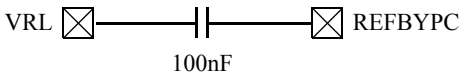


Figure 19-59. Reference Bypass Circuit

19.4.9.2 Analog-to-Digital Converter (ADC)

19.4.9.2.1 ADC Architecture

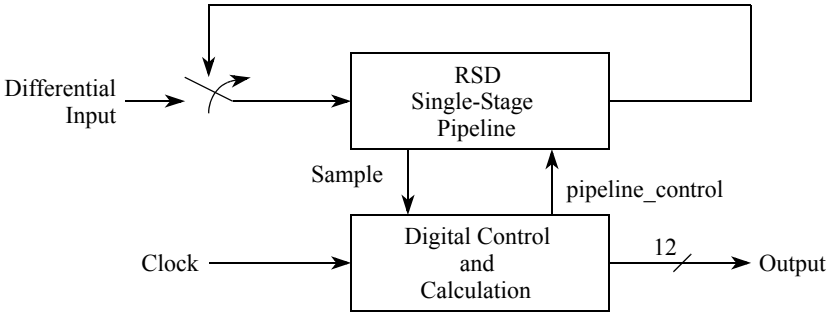


Figure 19-60. RSD ADC Block Diagram

The redundant signed digit (RSD) cyclic ADC consists of two main portions, the analog RSD stage, and the digital control and calculation module, as shown in Figure 19-60. To begin an analog-to-digital conversion, a differential input is passed into the analog RSD stage. The signal is passed through the RSD stage, and then from the RSD stage output, back to its input to be passed again. To complete a 12-bit conversion, the signal must pass through the RSD stage 12 times. Each time an input signal is read into the RSD stage, a digital sample is taken by the digital control/calculation module. The digital control/calculation module uses this sample to tell the analog module how to condition the signal. The digital module also saves each successive sample and adds them according to the RSD algorithm at the end of the entire conversion cycle.

19.4.9.2.2 RSD Overview

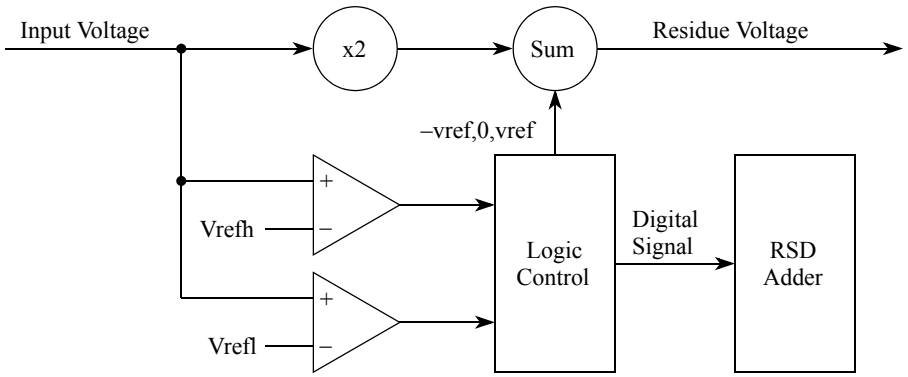


Figure 19-61. RSD Stage Block Diagram

On each pass through the RSD stage, the input signal will be multiplied by exactly two, and summed with either –vref, 0, or vref, depending on the logic control. The logic control will determine –vref, 0, or vref depending on the two comparator inputs. As the logic control sets the summing operation, it also sends a digital value to the RSD adder. Each time an analog signal passes through the RSD single-stage, a digital value is collected by the RSD adder. At the end of an entire AD conversion cycle, the RSD adder uses these collected values to calculate the 12-bit digital output.

Figure 19-62 shows the transfer function for the RSD stage. Note how the digital value (AB) is dependent on the two comparator inputs.

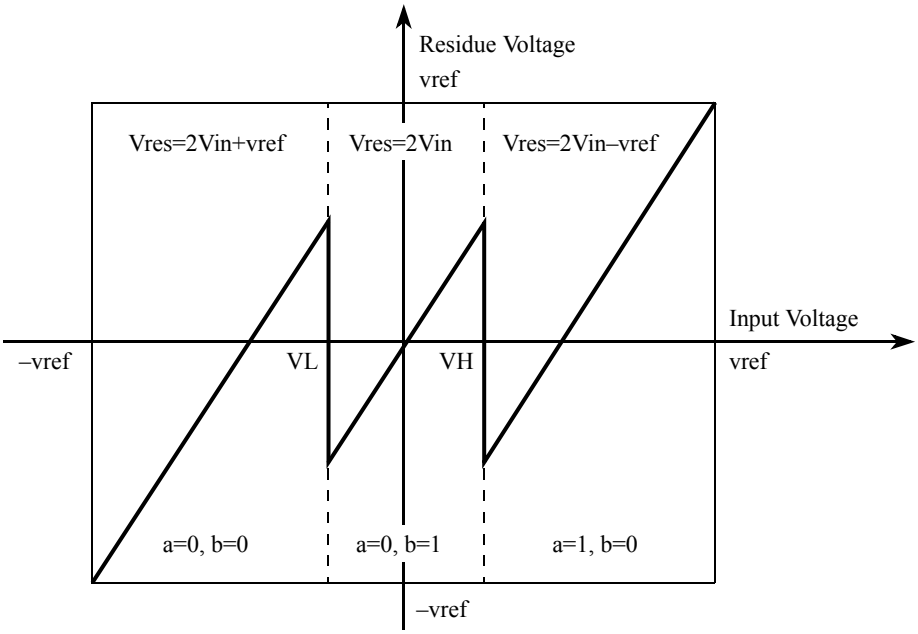


Figure 19-62. RSD Stage Transfer Function

In each pass through the RSD stage, the residue will be sent back to be the new input, and the digital signals, a and b, will be stored. For the 12-bit ADC, the input signal is sampled during the input phase, and after each of the 12 passes through the RSD stage. Thus, 13 total a and b values are collected. Upon

collecting all these values, they will be added according to the RSD algorithm to create the 12-bit digital representation of the original analog input. The bits are added in the following manner:

19.4.9.2.3 RSD Adder

The array, s1 to s12,will be the digital output of the RSD ADC with s1 being the msb and s12 being the lsb (least significant bit).

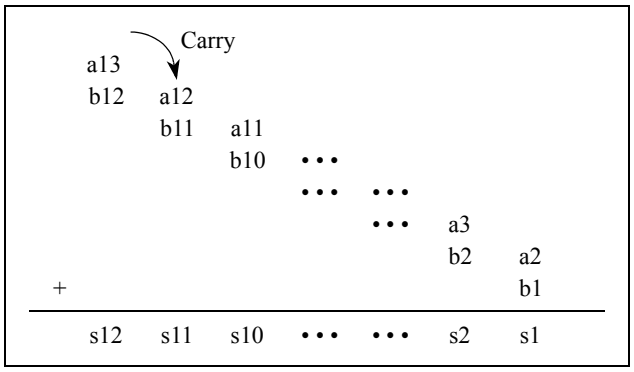


Figure 19-63. RSD Adder

19.5 Initialization/Application Information

19.5.1 Multiple Queues Control Setup Example

This section provides an example of how to configure multiple user command queues. Table 19-56 describes how each queue can be used for a different application. Also documented in this section are general guidelines on how to initialize the on-chip ADCs and the external device, and how to configure the command queues and the eQADC.

Table 19-56. Example Applications of Each Command Queue

Command Queue Number	Queue Type	Running Speed	Number of Contiguous Conversions	Example
0	Very fast burst time-based queue	every 2 μs for 200 μs; pause for 300 μs and then repeat	2	Injector current profiling
1	Fast hardware-triggered queue	every 900 μs	3	Current sensing of PWM controlled actuators
2	Fast repetitive time-based queue	every 2 ms	8	Throttle position
3	Software-triggered queue	every 3.9 ms	3	Command triggered by software strategy
4	Repetitive angle-based queue	every 625 us	7	Airflow read every 30 degrees at 8000 RPM
5	Slow repetitive time-based queue	every 100 ms	10	Temperature sensors

19.5.1.1 Initialization of On-Chip ADCs/External Device

The following steps provide an example of configuring the eQADC to initialize the on-chip ADCs and the external device. In this example, commands will be sent through CFIFO0.

1. Load all required configuration commands in the RAM in such way that they form a queue; this data structure will be referred below as Queue0. Figure 19-64 shows an example of a command queue able to configure the on-chip ADCs and external device at the same time.
2. Configure Section 19.3.2.2, "eQADC Null Message Send Format Register (EQADC_NMSFR)."
3. Configure Section 19.3.2.12, "eQADC SSI Control Register (EQADC_SSICR)," to communicate with the external device.
4. Enable the eQADC SSI by programming the ESSIE field the Section 19.3.2.1, "eQADC Module Configuration Register (EQADC_MCR)."
   a) Write 0b10 to ESSIE field to enable the eQADC SSI. FCK is free running but serial transmissions are not started.
   b) Wait until the external device becomes stable after reset.
   c) Write 0b11 to ESSIE field to enable the eQADC SSI to start serial transmissions.
5. Configure the eDMA to transfer data from Queue0 to CFIFO0 in the eQADC.
6. Configure Section 19.3.2.7, "eQADC Interrupt and eDMA Control Registers 0–5 (EQADC_IDCRn)."
   a) Set CFFS0 to configure the eQADC to generate an eDMA request to load commands from Queue0 to the CFIFO0.
   b) Set CFFE0 to enable the eQADC to generate an eDMA request to transfer commands from Queue0 to CFIFO0; Command transfers from the RAM to the CFIFO0 will start immediately.
   c) Set EOQIE0 to enable the eQADC to generate an interrupt after transferring all of the commands of Queue0 through CFIFO0.
7. Configure Section 19.3.2.6, "eQADC CFIFO Control Registers 0–5 (EQADC_CFCRn)."
   a) Write 0b0001 to the MODE0 field in eQADC_CFCR0 to program CFIFO0 for software single-scan mode.
   b) Write 1 to SSE0 to assert SSS0 and trigger CFIFO0.
8. Because CFIFO0 is in single-scan software mode and it is also the highest priority CFIFO, the eQADC starts to transfer configuration commands to the on-chip ADCs and to the external device.
9. When all of the configuration commands have been transferred, EQADC_FISRn[CF0])(see Section 19.3.2.8) will be set. The eQADC generates a end of queue interrupt. The initialization procedure is complete.

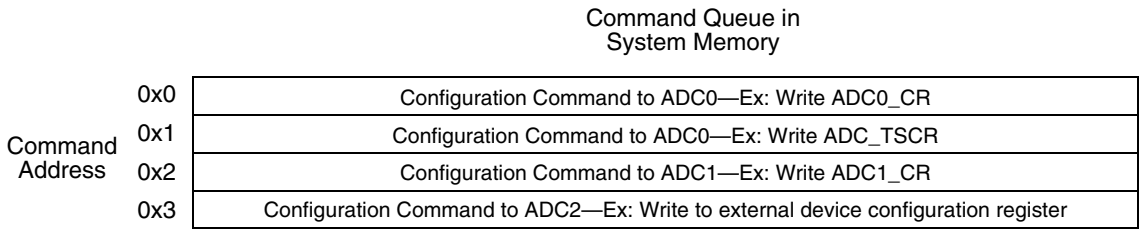


Figure 19-64. Example of a Command Queue Configuring the On-Chip ADCs/External Device

19.5.1.2 Configuring eQADC for Applications

This section provides an example based on the applications in Table 19-56. The example describes how to configure multiple command queues to be used for those applications and provides a step-by-step procedure to configure the eQADC and the associated command queue structures. In the example, the "Fast hardware-triggered command queue," described on the second row of Table 19-56, will have its commands transferred to ADC1; the conversion commands will be executed by ADC1. The generated results will be returned to RFIFO3 before being transferred to the result queues in the RAM by the eDMA.

NOTE

There is no fixed relationship between CFIFOs and RFIFOs with the same number. The results of commands being transferred through CFIFO1 can be returned to any RFIFO, regardless of its number. The destination of a result is determined by the MESSAGE_TAG field of the command that requested the result. See Section 19.4.1.2, "Message Format in eQADC," for details.

Step One: Set up the command queues and result queues.

1. Load the RAM with configuration and conversion commands. Table 19-57 is an example of how command queue 1 commands should be set.
   a) Each trigger event will cause four commands to be executed. When the eQADC detects the pause bit asserted, it will wait for another trigger to restart transferring commands from the CFIFO.
   b) At the end of the command queue, the "EOQ" bit is asserted as shown in Table 19-57.
   c) Results will be returned to RFIFO3 as specified in the MESSAGE_TAG field of commands.
2. Reserve memory space for storing results.

Table 19-57. Example of Command Queue Commands[1]

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	EOQ	PAUSE	RESERVED		ABORT_ST	EB (0b1)	BN	CAL	MESSAGE TAG		ADC COMMAND																					
CMD1	0	0	0		0	0	1	0	0b0011		Conversion Command																					
CMD2	0	0	0		0	0	1	0	0b0011		Conversion Command																					
CMD3	0	0	0		0	0	1	0	0b0011		Conversion Command																					
CMD4	0	1	0		0	0	1	0	0b0011 [2]		Configure peripheral device for next conversion sequence																					
CMD5	0	0	0		0	0	1	0	0b0011		Conversion Command																					
CMD6	0	0	0		0	0	1	0	0b0011		Conversion Command																					
CMD7	0	0	0		0	0	1	0	0b0011		Conversion Command																					
CMD8	0	1	0		0	0	1	0	0b0011[2]		Configure peripheral device for next conversion sequence																					
≈									etc.....																						≈	
	CFIFO Header						ADC Command																									

Table 19-57. Example of Command Queue Commands[1]

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	EOQ	PAUSE	RESERVED		ABORT_ST	EB (0b1)	BN	CAL	MESSAGE TAG												ADC COMMAND											
CMDEOQ	1	0	0		0	0	1	0	0b0011												EOQ Message											
	CFIFO Header																				ADC Command											

[1] Fields LST, TSR, FMT, and CHANNEL_NUMBER are not shown for clarity. See Section , " Conversion Command Message Format for On-Chip ADC Operation," for details.

[2] MESSAGE_TAG field is only defined for read configuration commands.

Step Two: Configure the eDMA to handle data transfers between the command/result queues in RAM and the CFIFOs/RFIFOs in the eQADC.

1. For transferring, set the source address of the eDMA TCD*n* to point to the start address of command queue 1. Set the destination address of the eDMA to point to EQADC_CFPR1. Refer to Section 19.3.2.4, "eQADC CFIFO Push Registers 0–5 (EQADC_CFPRn)."
2. For receiving, set the source address of the eDMA TCD*n* to point to EQADC_RFPR3. Refer to Section 19.3.2.5, "eQADC Result FIFO Pop Registers 0–5 (EQADC_RFPRn)." Set the destination address of the eDMA to point to the starting address of result queue 1.

Step Three: Configure the eQADC control registers.

3. Configure Section 19.3.2.7, "eQADC Interrupt and eDMA Control Registers 0–5 (EQADC_IDCRn)."
   a) Set EOQIE1 to enable the End of Queue Interrupt request.
   b) Set CFFS1 and RFDS3 to configure the eQADC to generate eDMA requests to push commands into CFIFO1 and to pop result data from RFIF03.
   c) Set CFINV1 to invalidate the contents of CFIFO1.
   d) Set RFDE3 and CFFE1 to enable the eQADC to generate eDMA requests. Command transfers from the RAM to the CFIFO1 will start immediately.
   e) Set RFOIE3 to indicate if RFIFO3 overflows.
   f) Set CFUIE1 to indicate if CFIFO1 underflows.
4. Configure MODE1 to continuous-scan rising edge external trigger mode in Section 19.3.2.6, "eQADC CFIFO Control Registers 0–5 (EQADC_CFCRn)."

Step Four: Command transfer to ADCs and result data reception.

When an external rising edge event occurs for CFIFO1, the eQADC automatically will begin transferring commands from CFIFO1 when it becomes the highest priority CFIFO trying to send commands to ADC1. The received results will be placed in RFIFO3 and then moved to result queue 1 by the eDMA.

19.5.2 EQADC/eDMA Controller Interface

This section provides an overview of the EQADC/eDMA interface and general guidelines about how the eDMA should be configured in order for it to correctly transfer data between the queues in system memory and the EQADC FIFOs.

19.5.2.1 Command Queue/CFIFO Transfers

In transfers involving command queues and CFIFOs, the eDMA moves data from a queued source to a single destination as shown in Figure 19-65. The location of the data to be moved is indicated by the source address, and the final destination for that data, by the destination address. The eDMA has transfer control descriptors (TCDs) containing these addresses and other parameters used in the control of data transfers (See Section 9.3.1.16, "Transfer Control Descriptor (TCD)" for more information). For every eDMA request issued by the EQADC, the eDMA must be configured to transfer a single command (32-bit data) from the command queue, pointed to by the source address, to the CFIFO push register, pointed to by the destination address. After the service of an eDMA request is completed, the source address has to be updated to point to the next valid command. The destination address remains unchanged. When the last command of a queue is transferred one of the following actions is recommended. Refer to Chapter 9, "Enhanced Direct Memory Access (eDMA)" for details about how this functionality is supported.

- The corresponding eDMA channel should be disabled. This might be desirable for CFIFOs in single scan mode.
- The source address should be updated to pointed to a valid command which can be the first command in the queue that has just been transferred (cyclic queue), or the first command of any other command queue. This is desirable for CFIFOs in continuous scan mode, or in some cases, for CFIFOs in single scan mode.

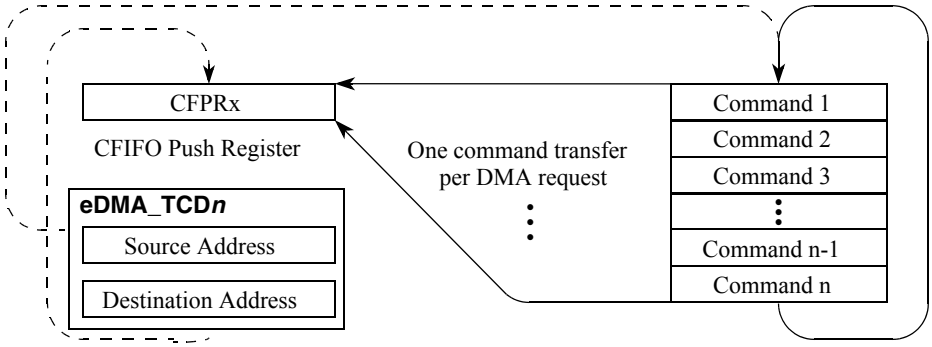


Figure 19-65. Command Queue/CFIFO Interface

19.5.2.2 Receive Queue/RFIFO Transfers

In transfers involving receive queues and RFIFOs, the eDMA controller moves data from a single source to a queue destination as shown in Figure 19-66. The location of the data to be moved is indicated by the source address, and the final destination for that data, by the destination address. For every eDMA request issued by the EQADC, the eDMA controller has to be configured to transfer a single result (16-bit data), pointed to by the source address, from the RFIFO pop register to the receive queue, pointed to by the destination address. After the service of an eDMA request is completed, the destination address has to be updated to point to the location where the next 16-bit result will be stored. The source address remains unchanged. When the last expected result is written to the receive queue, one of the following actions is recommended. Refer to Chapter 9, "Enhanced Direct Memory Access (eDMA)" for details about how this functionality is supported.

- The corresponding eDMA channel should be disabled.
- The destination address should be updated pointed to the next location where new coming results are stored, which can be the first entry of the current receive queue (cyclic queue), or the beginning of a new receive queue.

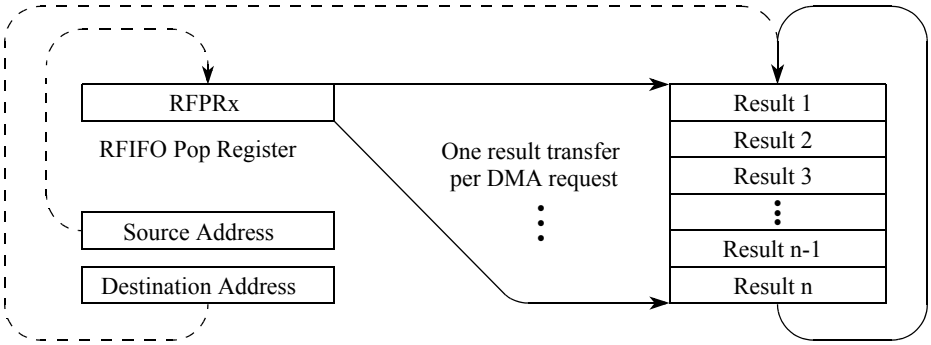


Figure 19-66. Receive Queue/RFIFO Interface

19.5.3 Sending Immediate Command Setup Example

In the eQADC, there is no immediate command register for sending a command immediately after writing to that register. However, a CFIFO can be configured to perform the same function as an immediate command register. The following steps illustrate how to configure CFIFO5 as an immediate command CFIFO. This eliminates the use of the eDMA. The results will be returned to RFIFO5.

1. Configure the Section 19.3.2.7, "eQADC Interrupt and eDMA Control Registers 0–5 (EQADC_IDCRn)."
   a) Clear CFIFO fill enable5 (CFFE5 = 0) in EQADC_IDCR5.
   b) Clear CFIFO underflow interrupt enable5 (CFUIE5 = 0) in EQADC_IDCR2.
   c) Clear RFDS5 to configure the eQADC to generate interrupt requests to pop result data from RFIF05.
   d) Set RFIFO drain enable5 (RFDE5 = 1) in EQADC_IDCR5.
2. Configure the Section 19.3.2.6, "eQADC CFIFO Control Registers 0–5 (EQADC_CFCRn)."
   a) Write 1 to CFINV5 in EQADC_CFCR5. This will invalidate the contents of CFIFO5.
   b) Set MODE5 to continuous-scan software trigger mode in EQADC_CFCR5.
3. To transfer a command, write it to the eQADC CFIFO push register 5 (EQADC_CFPR5) with message tag = 0b0101. Refer to Section 19.3.2.4, "eQADC CFIFO Push Registers 0–5 (EQADC_CFPRn)."
4. Up to 4 commands can be queued in CFIFO5. Check the CFCTR5 status in EQADC_FISR5 before pushing another command to avoid overflowing the CFIFO. Refer to Section 19.3.2.8, "eQADC FIFO and Interrupt Status Registers 0–5 (EQADC_FISRn)."
5. When the eQADC receives a conversion result for RFIFO5, it generates an interrupt request. RFIFO pop register 5 (EQADC_RFPR5) can be popped to read the result. Refer to Section 19.3.2.5, "eQADC Result FIFO Pop Registers 0–5 (EQADC_RFPRn)."

19.5.4 Modifying Queues

More command queues may be needed than the six supported by the eQADC. These additional command queues can be supported by interrupting command transfers from a configured CFIFO, even if it is triggered and transferring, modifying the corresponding command queue in the RAM or associating another command queue to it, and restarting the CFIFO. More details on disabling a CFIFO are described in Section 19.4.3.5.1, "Disabled Mode."

1. Determine the resumption conditions when later resuming the scan of the command queue at the point before it was modified.
   a) Change EQADC_CFCRn[MODE*n*] (see Section 19.3.2.6) to disabled. Refer to Section 19.4.3.5.1, "Disabled Mode," for a description of what happens when MODE*n* is changed to disabled.
   b) Poll EQADC_CFSR[CFS*n*] until it becomes IDLE (see Section 19.3.2.11).
   c) Read and save EQADC_CFTCRn[TC_CF*n*] (see Section 19.3.2.9) for later resuming the scan of the queue. The TC_CF*n* provides the point of resumption.
   d) Since all result data may not have being stored in the appropriate RFIFO at the time MODE*n* is changed to disable, wait for all expected results to be stored in the RFIFO/result queue before reconfiguring the eDMA to work with the modified result queue. The number of results that must return can be estimated from the TC_CF*n* value obtained above.
2. Disable the eDMA from responding to the eDMA request generated by EQADC_FISRn[CFFF*n*] and EQADC_FISRn[RFDF*n*] (see Section 19.3.2.8).
3. Write "0x0000" to the TC_CF*n* field.
4. Load the new configuration and conversion commands into RAM. Configure the eDMA to support the new command/result queue, but do not configure it yet to respond to eDMA requests from CFIFO*n*/RFIFO*n*.
5. If necessary, modify the EQADC_IDCRn registers (see Section 19.3.2.7) to suit the modified command queue.
6. Write 1 to EQADC_CFCRn[CFINV*n*] (see Section 19.3.2.6) to invalidate the entries of CFIFO*n*.
7. Configure the eDMA to respond to eDMA requests generated by CFFF*n* and RFDF*n*.
8. Change MODE*n* to the modified CFIFO operation mode. Write 1 to SSE*n* to trigger CFIFO*n* if MODE*n* is software trigger.

19.5.5 Command Queue and Result Queue Usage

Figure 19-67 is an example of command queue and result queue usage. It shows the command queue 0 commands requesting results that will be stored in result queue 0 and result queue 1, and command queue 1 commands requesting results that will be stored only in result queue 1. Some command messages request data to be returned from the on-chip ADC/external device, but some only configure them and do not request returning data. When a command queue contains both write and read commands like command queue 0, the command queue and result queue entries will not be aligned, in Figure 19-67, the result for the second command of command queue 0 is the first entry of result queue 0. The figure also shows that command queue and result queue entries can also become unaligned even if all commands in a command queue request data as command queue 1. Command queue 1 entries became unaligned to result queue 1 entries because a result requested by the forth command queue 0 command was sent to result queue 1. This happens because the system can be configured so that several command queues can have results sent to a single result queue.

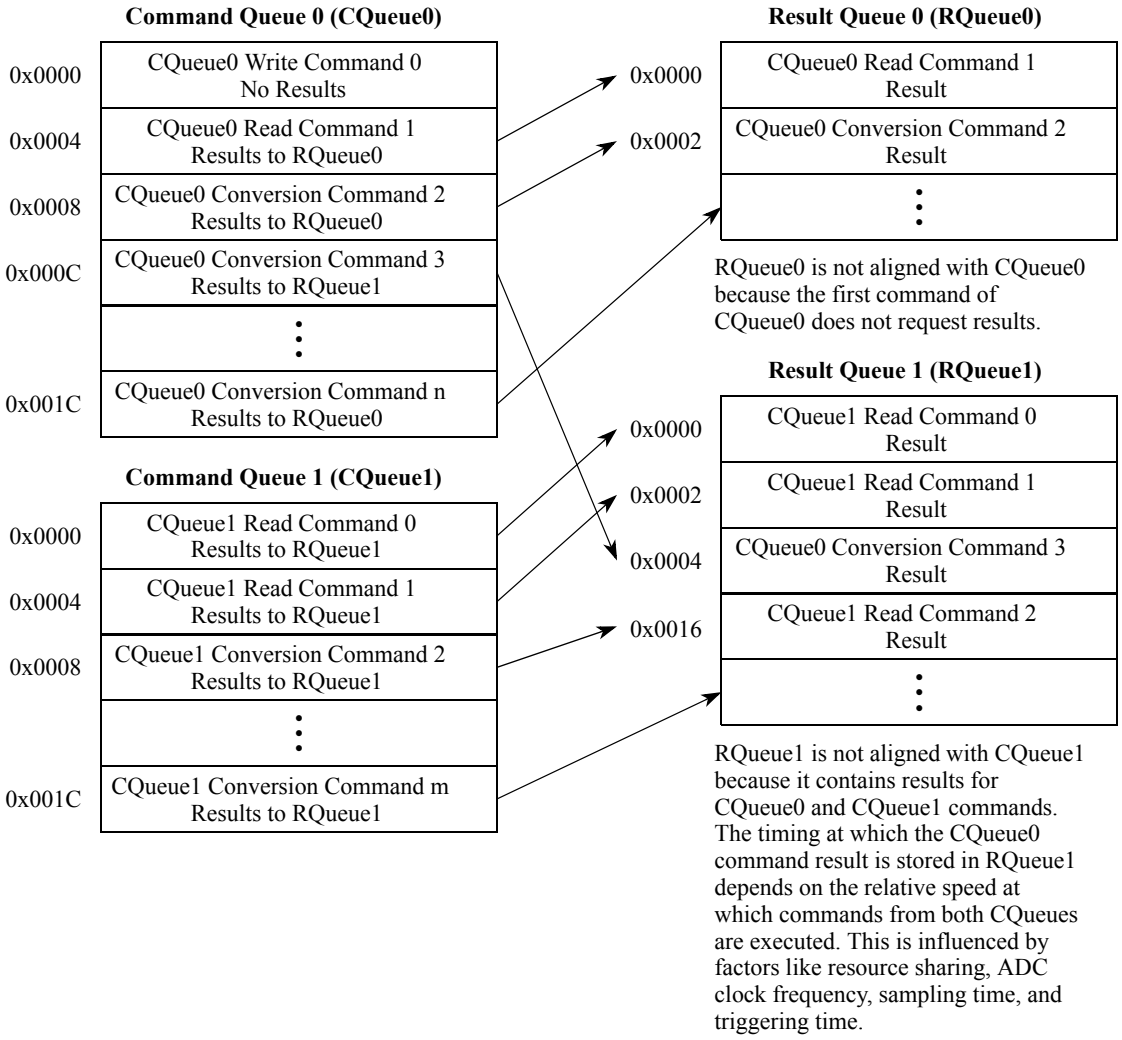


Figure 19-67. eQADC Command and Result Queues

19.5.6 ADC Result Calibration

The ADC result calibration process consists of two steps: determining the gain and offset calibration constants, and calibrating the raw results generated by the on-chip ADCs by solving the following equation discussed in Section 19.4.5.4.1, "Calibration Overview."

$$CAL\_RES = GCC * RAW\_RES + OCC + 2; \quad (5.5.a)$$

The calibration constants GCC and OCC can be calculated from equation (5.5.a) provided that two pairs of expected (CAL_RES) and measured (RAW_RES) result values are available for two different input voltages. Most likely calibration points to be used are 25% VREF[1] and 75% VREF since they are far apart but not too close to the end points of the full input voltage range. This allows for calculations of more representative calibration constants. The eQADC provides these voltages via channel numbers 43 and 44. The raw, uncalibrated results for these input voltages are obtained by converting these channels with conversion commands that have the CAL bit negated.

1.VREF=V_{RH}-V_{RL}

The transfer equations for when sampling these reference voltages are:

$$CAL\_RES_{75\%VREF} = GCC * RAW\_RES_{75\%VREF} + OCC + 2; \quad (5.5.b)$$

$$CAL\_RES_{25\%VREF} = GCC * RAW\_RES_{25\%VREF} + OCC + 2; \quad (5.5.c)$$

Thus;

$$GCC = (CAL\_RES_{75\%VREF} - CAL\_RES_{25\%VREF}) / (RAW\_RES_{75\%VREF} - RAW\_RES_{25\%VREF}); \quad (5.5.d)$$

$$OCC = CAL\_RES_{75\%VREF} - GCC*RAW\_RES_{75\%VREF} - 2; \quad (5.5.e)$$

or

$$OCC = CAL\_RES_{25\%VREF} - GCC*RAW\_RES_{25\%VREF} - 2; \quad (5.5.f)$$

After being calculated, the GCC and OCC values must be written to ADC0_GCCR and ADC1_GCCR registers (see Section 19.3.3.4) and the ADC0_OCCR and ADC1_OCCR registers (see Section 19.3.3.5) using write configuration commands.

The eQADC will automatically calibrate the results, according to equation (5.5.a), of every conversion command that has its CAL bit asserted using the GCC and OCC values stored in the ADC calibration registers.

19.5.6.1 MAC Configuration Procedure

The following steps illustrate how to configure the calibration hardware, that is, determining the values of the gain and offset calibration constants, and the writing these constants to the calibration registers. This procedure should be performed for both ADC0 and ADC1.

1. Convert channel 44 with a command that has its CAL bit negated and obtain the raw, uncalibrated result for 25%VREF ($RAW\_RES_{25\%VREF}$).
2. Convert channel 43 with a command that has its CAL bit negated and obtain the raw, uncalibrated result for 75%VREF ($RAW\_RES_{75\%VREF}$).
3. Because the expected values for the conversion of these voltages are known ($CAL\_RES_{25\%VREF}$ and $CAL\_RES_{75\%VREF}$), GCC and OCC values can be calculated from equations (5.5.d) and (5.5.e) using these values, and the results determined in steps 1 and 2.
4. Reformat GCC and OCC to the proper data formats as specified in Section 19.4.5.4.2, “MAC Unit and Operand Data Format.” GCC is an unsigned 15-bit fixed point value and OCC is a signed 14-bit value.
5. Write the GCC value to ADCn gain calibration registers (see Section 19.3.3.4) and the OCC value to ADCn offset calibration constant registers (see Section 19.3.3.5) using write configuration commands.

19.5.6.2 Example Calculation of Calibration Constants

The raw results obtained when sampling reference voltages 25%VREF and 75%VREF were, respectively, 3798 and 11592. The results that should have been obtained from the conversion of these reference voltages are, respectively, 4096 and 12288. Therefore, using equations (5.5.d) and (5.5.e), the gain and offset calibration constants are:

```
GCC=(12288-4096)/(11592-3798) = 1.05106492-> 1.05102539[1] = 0x4344
    OCC=12288-1.05106492*11592 -2 = 102.06-> 102 = 0x0066
```

1. This calculation is rounded down due to binary approximation.

Table 19-58 shows, for this particular case, examples of how the result values change according to GCC and OCC when result calibration is executed (CAL=1) and when it is not (CAL=0).

Table 19-58. Calibration Example

Input Voltage	Raw result (CAL=0)		Calibrated result (CAL=1)	
	Hexadecimal	Decimal	Hexadecimal	Decimal
25% VREF	0x0ED6	3798	0x1000	4095.794
75% VREF	0x2D48	11592	0x3000	12287.486

19.5.6.3 Quantization Error Reduction During Calibration

Figure 19-68 shows how the ADC transfer curve changes due to the addition of two to the MAC output during the calibration - see MAC output equation in Section 19.4.5.4, "ADC Calibration Feature". The maximum absolute quantization error is reduced by half leading to an increase in accuracy.

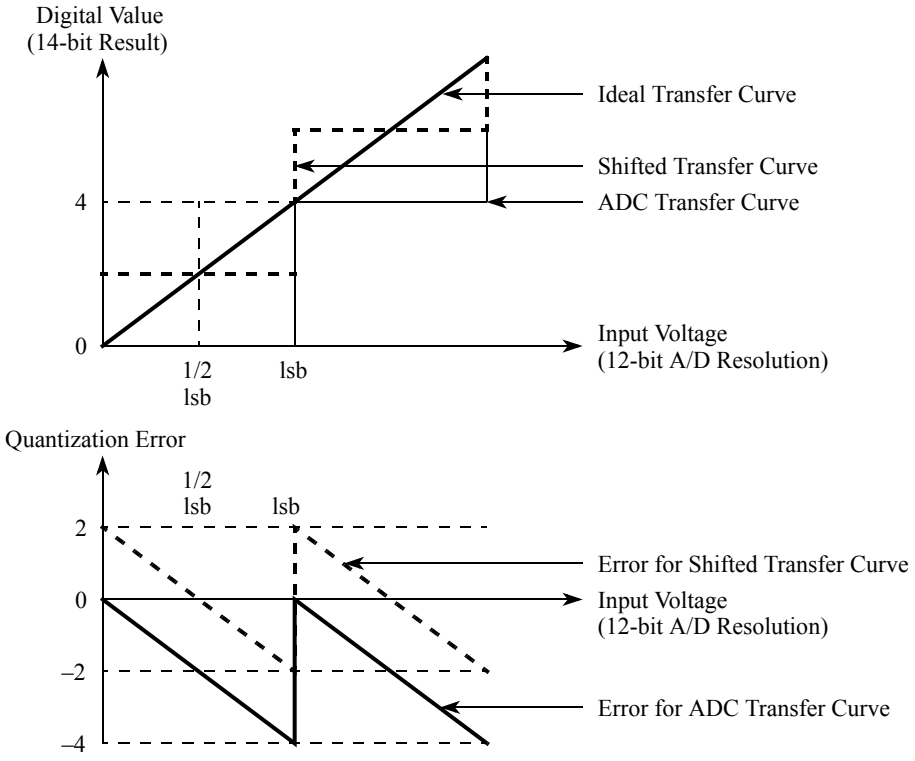


Figure 19-68. Quantization Error Reduction During Calibration

19.5.7 eQADC versus QADC

This section describes how the eQADC upgrades the QADC functionality. The section also provides a comparison between the eQADC and QADC in terms of their functionality. This section targets users familiar with terminology in QADC. Figure 19-69 is an overview of a QADC. Figure 19-70 is an overview of the eQADC system.

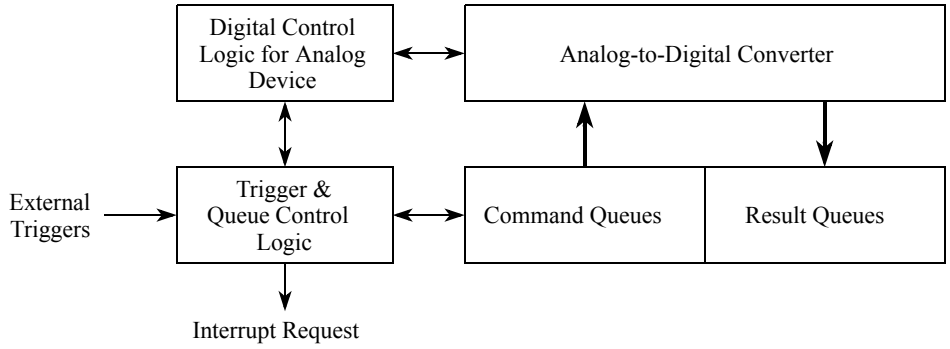


Figure 19-69. QADC Overview

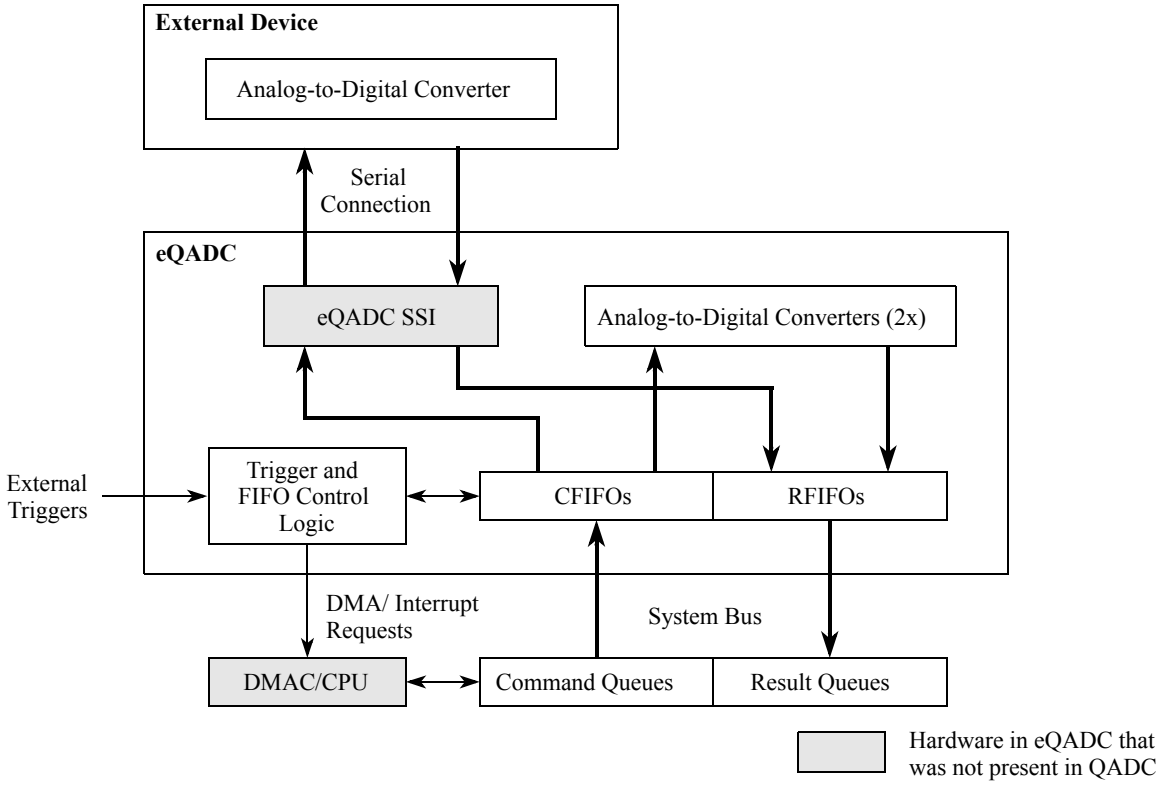


Figure 19-70. eQADC System Overview

The eQADC system consists of four parts: queues in system memory, the eQADC, on-chip ADCs, and an external device. As compared with the QADC, the eQADC system requires two pieces of extra hardware.

1. An eDMA or an MCU is required to move data between the eQADC's FIFOs and queues in the system memory.
2. A serial interface [eQADC synchronous serial interface (SSI)]is implemented to transmit and receive data between the eQADC and the external device.

Because there are only FIFOs inside the eQADC, much of the terminology or use of the register names, register contents, and signals of the eQADC involve FIFO instead of queue. These register names, register contents, and signals are functionally equivalent to the queue counterparts in the QADC. Table 19-59 lists how the eQADC register, register contents, and signals are related to QADC.

Table 19-59. Terminology Comparison between QADC and eQADC

QADC Terminology	eQADC Terminology	Function
CCW	Command Message	In the QADC, the hardware only executes conversion command words. In the eQADC, not all commands are conversion commands; some are configuration commands.
Queue Trigger	CFIFO Trigger	In the QADC, a trigger event is required to start the execution of a queue. In the eQADC, a trigger event is required to start command transfers from a CFIFO. When a CFIFO is triggered and transferring, commands are continuously moved from command queues to CFIFOs. Thus, the trigger event initiates the “execution of a queue” indirectly.
Command Word Pointer Queue *n* (CWPQ*n*)	Counter Value of Commands Transferred from Command FIFO*n* (TC_CF*n*)	In the QADC, CWPQ*n* allows the last executed command on queue *n* to be determined. In the eQADC, the TC_CF*n* value allows the last transferred command on command queue *n* to be determined.
Queue Pause Bit (P)	CFIFO Pause Bit	In the QADC, detecting a pause bit in the CCW will pause the queue execution. In the eQADC, detecting a pause bit in the command will pause command transfers from a CFIFO.
Queue Operation Mode (MQ*n*)	CFIFO Operation Mode (MODE*n*)	The eQADC supports all queue operation modes in the QADC except operation modes related to a periodic timer. A timer elsewhere in the system can provide the same functionality if it is connected to ETRIG*n*.
Queue Status (QS)	CFIFO Status (CFS*n*)	In the QADC, the queue status is read to check whether a queue is idle, active, paused, suspended, or trigger pending. In the eQADC, the CFIFO status is read to check whether a queue is IDLE, WAITING FOR TRIGGER (idle or paused in QADC), or triggered (suspended or trigger pending in QADC).

The eQADC and QADC also have similar procedures for the configuration or execution of applications. Table 19-60 shows the steps required for the QADC versus the steps required for the eQADC system.

Table 19-60. Usage Comparison between QADC and eQADC System

Procedure	QADC	eQADC System
Analog Control Configuration	Configure analog device by writing to the QADCs.	Program configuration commands into command queues.
Prepare Scan Sequence	Program scan commands into command queues.	Program scan commands into command queues.
Queue Control Configuration	Write to the QADC control registers.	Write to the eQADC control registers.
Data Transferred between Queues and Buffers	Not Required.	Program the eDMA or the CPU to handle the data transfer.
Serial Interface Configuration	Not Required.	Write to the eQADC SSI registers.
Queue Execution	Require software or external trigger events to start queue execution.	Require software or external trigger events to start command transfers from a CFIFO.

19.6 Revision History

Substantive Changes since Rev 3.0
Changed Section 19.4.9.1, “Reference Bypass,” to say only “The reference bypass capacitor (REFBYPC) signal requires a 100 nF capacitor connected to VRL to filter noise on the internal reference used by the ADC.” Also changed Figure 19-59 to just show the bypass cap. Changed section title and figure title from “Bias Generator” to “Reference Bypasst”.
Added footnote to Table 19-51: “This equation only applies before calibration. After calibration, the 50% reference point will actually return approximately 20mV lower than the expected 50% of the difference between the High Reference Voltage (VRH) and the Low Reference Voltage (VRL). For calibration of the ADC only the 25% and 75% points should be used as described in Section 19.5.6.1, “MAC Configuration Procedure”

Chapter 20
Deserial Serial Peripheral Interface (DSPI)

20.1 Introduction

This chapter describes the deserial serial peripheral interface (DSPI), which provides a synchronous serial bus for communication between the MPC5553/MPC5554 and an external peripheral device.

20.1.1 Block Diagram

A block diagram of the DSPI is shown in Figure 20-1.

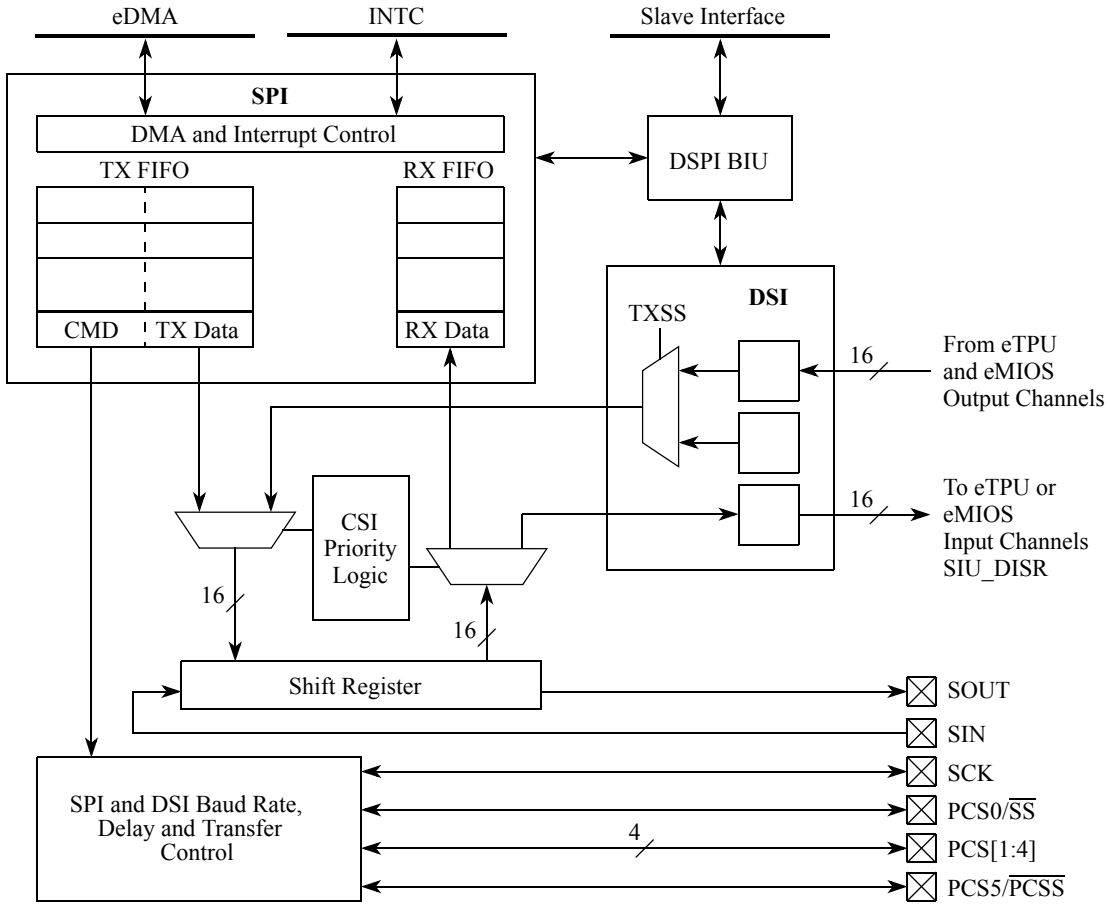


Figure 20-1. DSPI Block Diagram

20.1.2 Overview

The DSPI supports pin count reduction through serialization and deserialization of eTPU channels, eMIOS channels, and memory-mapped registers. Incoming serial data may be used to trigger external interrupt requests through DSPI deserialized output connections to the SIU. The channels and register content are transmitted using an SPI-like protocol. There are four identical DSPI modules (DSPI_A, DSPI_B, DSPI_C, and DSPI_D) on the MPC5554, and three DSPI modules on the MPC5553 (DSPI_B, DSPI_C, and DSPI_D). The DSPI has three configurations:

- Serial peripheral interface (SPI) configuration where the DSPI operates as an SPI with support for queues.
- Deserial serial interface (DSI) configuration where the DSPI serializes eTPU and eMIOS output channels and deserializes the received data by placing it on the eTPU and eMIOS input channels and as inputs to the external interrupt request submodule of the SIU.
- Combined serial interface (CSI) configuration where the DSPI operates in both SPI and DSI configurations interleaving DSI frames with SPI frames, giving priority to SPI frames.

For queued operations the SPI queues reside in internal SRAM which is external to the DSPI. Data transfers between the queues and the DSPI FIFOs are accomplished through the use of the eDMA controller or through host software. Figure 20-2 shows a DSPI with external queues in internal SRAM.

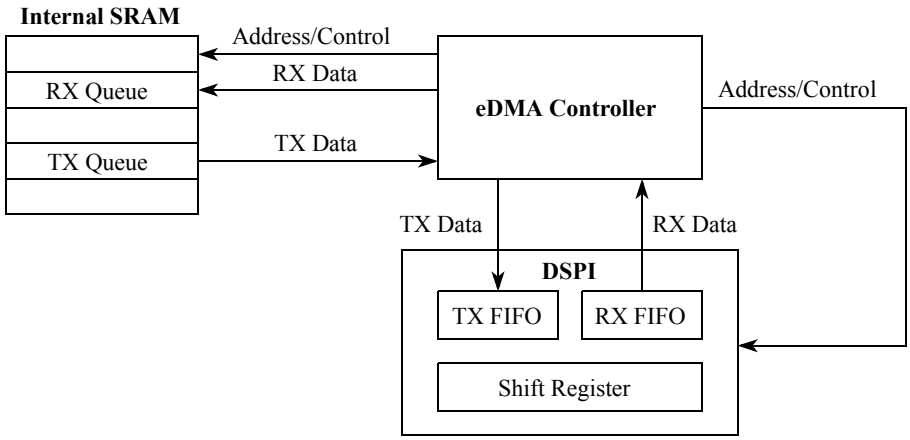


Figure 20-2. DSPI with Queues and eDMA

20.1.3 Features

The DSPI supports these SPI features:

- Full-duplex, three-wire synchronous transfers
- Master and slave mode
- Buffered transmit and receive operation using the TX and RX FIFOs, with depths of four entries
- Visibility into TX and RX FIFOs for ease of debugging
- FIFO bypass mode for low-latency updates to SPI queues
- Programmable transfer attributes on a per-frame basis
  - — Eight clock and transfer attribute registers
  - — Serial clock with programmable polarity and phase
  - — Programmable delays
    - – PCS to SCK delay
    - – SCK to PCS delay
    - – Delay between frames
  - — Programmable serial frame size of 4 to 16 bits, expandable with software control
  - — Continuously held chip select capability
- Six peripheral chip selects, expandable to 64 with external demultiplexer
- Deglitching support for up to 32 peripheral chip selects with external demultiplexer
- Two DMA conditions for SPI queues residing in RAM or Flash
  - — TX FIFO is not full (TFFF)
  - — RX FIFO is not empty (RFDF)
- Six interrupt conditions
  - – End of queue reached (EOQF)
  - – TX FIFO is not full (TFFF)
  - – Transfer of current frame complete (TCF)

  - RX FIFO is not empty (RFDF)
  - FIFO overrun (attempt to transmit with an empty TX FIFO or serial frame received while RX FIFO is full) (RFOF)
  - FIFO under flow (slave only and SPI mode, the slave is asked to transfer data when the TX FIFO is empty) in the MPC5553 only (TFUF)
- Modified SPI transfer formats for communication with slower peripheral devices
- Continuous serial communications clock (SCK)

When configured for DSI or CSI operation, the DSPI supports pin reduction through serialization and deserialization.

- Serialized data sources
  — eTPU_A, eTPU_B (for the MPC5554 only), and eMIOS output channels
  — Memory-mapped register in the DSPI
- Deserialized data destinations
  — eTPU_A and eMIOS input channels
  — SIU external interrupt request inputs
  — Memory-mapped register in the DSPI
- Transfer initiation conditions
  - Continuous
  - Edge sensitive hardware trigger
  - Change in data
- Support for parallel and serial chaining of DSPI modules
- Pin serialization/deserialization with interleaved SPI frames for control and diagnostics

20.1.4 Modes of Operation

The DSPI has four modes of operation. These modes can be divided into two categories; module-specific modes such as master, slave, and module disable modes, and an MCU-specific mode (debug mode).

The module-specific modes are entered by host software writing to a register. The MCU-specific mode is controlled by signals external to the DSPI. The MCU-specific mode is a mode that the entire MPC5553/MPC5554 may enter, in parallel to the DSPI being in one of its module-specific modes.

20.1.4.1 Master Mode

Master mode allows the DSPI to initiate and control serial communication. In this mode the SCK, PCS*n* and SOUT signals are controlled by the DSPI and configured as outputs. For more information, see Section 20.4.1.1, “Master Mode.”

20.1.4.2 Slave Mode

Slave mode allows the DSPI to communicate with SPI/DSI bus masters. In this mode the DSPI responds to externally controlled serial transfers. The DSPI cannot initiate serial transfers in slave mode. In slave mode, the SCK signal and the PCS0/$\overline{SS}$ signal are configured as inputs and provided by a bus master. PCS0/$\overline{SS}$ must be configured as input and pulled high. If the internal pull up is being used then the appropriate bits in the relevant SIU_PCR must be set (SIU_PCR [WPE=1], [WPS=1]). For more information, see Section 20.4.1.2, “Slave Mode.”

20.1.4.3 Module Disable Mode

The module disable mode is used for MCU power management. The clock to the non-memory mapped logic in the DSPI is stopped while in module disable mode. The DSPI enters the module disable mode when the MDIS bit in DSPI*x*_MCR is set. For more information, see Section 20.4.1.3, "Module Disable Mode."

20.1.4.4 Debug Mode

Debug mode is used for system development and debugging. If the device enters debug mode while the FRZ bit in the DSPI*x*_MCR is set, the DSPI halts operation on the next frame boundary. If the device enters debug mode while the FRZ bit is negated, the DSPI behavior is unaffected and remains dictated by the module-specific mode and configuration of the DSPI. For more information, see Section 20.4.1.4, "Debug Mode."

20.2 External Signal Description

20.2.1 Signal Overview

Table 20-1 lists off-chip DSPI signals.

Table 20-1. Signal Properties

Name	I/O Type	Function	
		Master Mode	Slave Mode
PCS0/$\overline{SS}$	Output / Input	Peripheral chip select 0	Slave select
PCS[1:3]	Output	Peripheral chip select 1–3	Unused[1]
PCS4/$\overline{MTRIG}$	Output	Peripheral chip select 4	Master Trigger
PCS5/$\overline{PCSS}$	Output	Peripheral chip select 5 / Peripheral chip select strobe	Unused[1]
SIN	Input	Serial data in	Serial data in
SOUT	Output	Serial data out	Serial data out
SCK	Output / Input	Serial clock (output)	Serial clock (input)

[1]In the SIU the user can select alternate pin functions for the MPC5553/MPC5554.

20.2.2 Signal Names and Descriptions

20.2.2.1 Peripheral Chip Select / Slave Select (PCS0/$\overline{SS}$)

In master mode, the PCS0 signal is a peripheral chip select output that selects the slave device to which the current transmission is intended.

In slave mode, the $\overline{SS}$ signal is a slave select input signal that allows an SPI master to select the DSPI as the target for transmission. PCS0/$\overline{SS}$ must be configured as input and pulled high. If the internal pull up is being used then the appropriate bits in the relevant SIU_PCR must be set (SIU_PCR [WPE=1], [WPS=1]).

The IBE and OBE bits in the corresponding SIU_PCR need to be set appropriately for all PCSx0 pins when the DSPI chip select/slave select primary function is selected for that pin. When the pin is to be used in DSPI master mode as a chip select output, then the OBE bit should be set. When the pin is to be used in DSPI slave mode as a slave select input, then the IBE bit should be set. See Section 6.3.1.12, "Pad Configuration Registers (SIU_PCR) for more information.

20.2.2.2 Peripheral Chip Selects 1–3 (PCS[1:3])

PCS[1:3] are peripheral chip select output signals in master mode. In slave mode these signals are not used.

20.2.2.3 Peripheral Chip Select 4 / Master Trigger (PCS4/$\overline{\text{MTRIG}}$)

PCS4 is a peripheral chip select output signal in master mode. In slave mode this signal is a master trigger.

20.2.2.4 Peripheral Chip Select 5 / Peripheral Chip Select Strobe (PCS5/$\overline{\text{PCSS}}$)

PCS5 is a peripheral chip select output signal. When the DSPI is in master mode and PCSSE bit in the DSPI*x*_MCR is negated, this signal is used to select which slave device the current transfer is intended for.

$\overline{\text{PCSS}}$ provides a strobe signal that can be used with an external logic device for deglitching of the PCS signals. When the DSPI is in master mode and the PCSSE bit in the DSPI*x*_MCR is set, the $\overline{\text{PCSS}}$ provides the appropriate timing for the decoding of the PCS[0:4] signals which prevents glitches from occurring.

PCS5/ $\overline{\text{PCSS}}$ is not used in slave mode.

20.2.2.5 Serial Input (SIN)

SIN is a serial data input signal.

20.2.2.6 Serial Output (SOUT)

SOUT is a serial data output signal.

20.2.2.7 Serial Clock (SCK)

SCK is a serial communication clock signal. In master mode, the DSPI generates the SCK. In slave mode, SCK is an input from an external bus master.

20.3 Memory Map/Register Definition

20.3.1 Memory Map

Table 20-2 shows the DSPI memory map.

Table 20-2. DSPI Detailed Memory Map

Address	Register Name	Register Description	Size (bits)
Base: 0xFFF9_0000 (DSPI A) 0xFFF9_4000 (DSPI B) 0xFFF9_8000 (DSPI C) 0xFFF9_C000 (DSPI D)	DSPI*x*_MCR	DSPI module configuration register	32
Base + 0x4	—	Reserved	—
Base + 0x8	DSPI*x*_TCR	DSPI transfer count register	32
Base + 0xC	DSPI*x*_CTAR0	DSPI clock and transfer attributes register 0	32
Base + 0x10	DSPI*x*_CTAR1	DSPI clock and transfer attributes register 1	32
Base + 0x14	DSPI*x*_CTAR2	DSPI clock and transfer attributes register 2	32
Base + 0x18	DSPI*x*_CTAR3	DSPI clock and transfer attributes register 3	32
Base + 0x1C	DSPI*x*_CTAR4	DSPI clock and transfer attributes register 4	32
Base + 0x20	DSPI*x*_CTAR5	DSPI clock and transfer attributes register 5	32
Base + 0x24	DSPI*x*_CTAR6	DSPI clock and transfer attributes register 6	32
Base + 0x28	DSPI*x*_CTAR7	DSPI clock and transfer attributes register 7	32
Base + 0x2C	DSPI*x*_SR	DSPI status register	32
Base + 0x30	DSPI*x*_RSER	DSPI DMA/interrupt request select and enable register	32
Base + 0x34	DSPI*x*_PUSHR	DSPI push TX FIFO register	32
Base + 0x38	DSPI*x*_POPR	DSPI pop RX FIFO register	32
Base + 0x3C	DSPI*x*_TXFR0	DSPI transmit FIFO register 0	32
Base + 0x40	DSPI*x*_TXFR1	DSPI transmit FIFO register 1	32
Base + 0x44	DSPI*x*_TXFR2	DSPI transmit FIFO register 2	32
Base + 0x48	DSPI*x*_TXFR3	DSPI transmit FIFO register 3	32
Base + 0x4C– Base + 0x78	—	Reserved	—
Base + 0x7C	DSPI*x*_RXFR0	DSPI receive FIFO register 0	32
Base + 0x80	DSPI*x*_RXFR1	DSPI receive FIFO register 1	32
Base + 0x84	DSPI*x*_RXFR2	DSPI receive FIFO register 2	32
Base + 0x88	DSPI*x*_RXFR3	DSPI receive FIFO register 3	32
Base + 0x8C– Base + 0xB8	—	Reserved	—
Base + 0xBC	DSPI*x*_DSICR	DSPI DSI configuration register	32
Base + 0xC0	DSPI*x*_SDR	DSPI DSI serialization data register	32
Base + 0xC4	DSPI*x*_ASDR	DSPI DSI alternate serialization data register	32

Table 20-2. DSPI Detailed Memory Map (continued)

Address	Register Name	Register Description	Size (bits)
Base + 0xC8	DSPI*x*_COMPR	DSPI DSI transmit comparison register	32
Base + 0xCC	DSPI*x*_DDR	DSPI DSI deserialization data register	32

20.3.2 Register Descriptions

20.3.2.1 DSPI Module Configuration Register (DSPI*x*_MCR)

The DSPI*x*_MCR contains bits which configure various attributes associated with DSPI operation. The HALT and MDIS bits can be changed at any time but will only take effect on the next frame boundary. Only the HALT and MDIS bits in the DSPI*x*_MCR may be changed while the DSPI is running.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MSTR	CONT_SCKE	DCONF		FRZ	MTFE	PCSSE	ROOE	0	0	PCSIS5	PCSIS4	PCSIS3	PCSIS2	PCSIS1	PCSIS0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	0	MDIS	DIS_TXF	DIS_RXF	CLR_TXF	CLR_RXF	SMPL_PT		0	0	0	0	0	0	0	HALT
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Reg Addr	Base															

Figure 20-3. DSPI Module Configuration Register (DSPI*x*_MCR)

Table 20-3. DSPI*x*_MCR Field Descriptions

Bits	Name	Description
0	MSTR	Master/slave mode select. Configures the DSPI for either master mode or slave mode. 0 DSPI is in slave mode 1 DSPI is in master mode
1	CONT_SCKE	Continuous SCK enable. Enables the serial communication clock (SCK) to run continuously. See Section 20.4.8, “Continuous Serial Communications Clock,” for details. 0 Continuous SCK disabled 1 Continuous SCK enabled

Table 20-3. DSPI*x*_MCR Field Descriptions (continued)

Bits	Name	Description
2–3	DCONF [0:1]	DSPI configuration. Selects between the three different configurations of the DSPI. The table below lists the DCONF values for the various configurations. **DCONF** / **Configuration** 00 — SPI 01 — DSI 10 — CSI 11 — Reserved
4	FRZ	Freeze. Enables the DSPI transfers to be stopped on the next frame boundary when the MPC5553/MPC5554 enters debug mode. 0 Do not halt serial transfers 1 Halt serial transfers
5	MTFE	Modified timing format enable. Enables a modified transfer format to be used. See Section 20.4.7.4, “Modified SPI/DSI Transfer Format (MTFE = 1, CPHA = 1),” for more information. 0 Modified SPI transfer format disabled 1 Modified SPI transfer format enabled
6	PCSSE	Peripheral chip select strobe enable. Enables the PCS5/$\overline{\text{PCSS}}$ to operate as an PCS strobe output signal. See Section 20.4.6.5, “Peripheral Chip Select Strobe Enable (PCSS),” for more information. 0 PCS5/$\overline{\text{PCSS}}$ is used as the Peripheral chip select 5 signal 1 PCS5/$\overline{\text{PCSS}}$ is used as an active-low PCS strobe signal
7	ROOE	Receive FIFO overflow overwrite enable. Enables an RX FIFO overflow condition to either ignore the incoming serial data or to overwrite existing data. If the RX FIFO is full and new data is received, the data from the transfer that generated the overflow is either ignored or shifted in to the shift register. If the ROOE bit is asserted, the incoming data is shifted in to the shift register. If the ROOE bit is negated, the incoming data is ignored. See Section 20.4.9.6, “Receive FIFO Overflow Interrupt Request (RFOF),” for more information. 0 Incoming data is ignored 1 Incoming data is shifted in to the shift register
8–9	—	Reserved, but implemented. These bits are writable, but have no effect.
10–15	PCSIS*n*	Peripheral chip select inactive state. Determines the inactive state of the PCS*n* signal. PCS0/$\overline{\text{SS}}$ must be configured as inactive high for slave mode operation. 0 The inactive state of PCS*n* is low 1 The inactive state of PCS*n* is high
16	—	Reserved.
17	MDIS	Module disable. Allows the clock to be stopped to the non-memory mapped logic in the DSPI effectively putting the DSPI in a software controlled power-saving state. See Section 20.4.10, “Power Saving Features for more information.” The reset value of the MDIS bit is parameterized, with a default reset value of 0. 0 Enable DSPI clocks 1 Allow external logic to disable DSPI clocks

Table 20-3. DSPI*x*_MCR Field Descriptions (continued)

Bits	Name	Description
18	DIS_TXF	Disable transmit FIFO. Provides a mechanism to disable the TX FIFO. When the TX FIFO is disabled, the transmit part of the DSPI operates as a simplified double-buffered SPI. See Section 20.4.3.3, "FIFO Disable Operation for details." 0 TX FIFO is enabled 1 TX FIFO is disabled
19	DIS_RXF	Disable receive FIFO. Provides a mechanism to disable the RX FIFO. When the RX FIFO is disabled, the receive part of the DSPI operates as a simplified double-buffered SPI. See Section 20.4.3.3, "FIFO Disable Operation for details." 0 RX FIFO is enabled 1 RX FIFO is disabled
20	CLR_TXF	Clear TX FIFO. Flushes the TX FIFO. Writing a 1 to CLR_TXF clears the TX FIFO counter. The CLR_TXF bit is always read as zero. 0 Do not clear the TX FIFO counter 1 Clear the TX FIFO counter
21	CLR_RXF	Clear RX FIFO. Flushes the RX FIFO. Writing a 1 to CLR_RXF clears the RX counter. The CLR_RXF bit is always read as zero. 0 Do not clear the RX FIFO counter 1 Clear the RX FIFO counter
22–23	SMPL_PT [0:1]	Sample point. Allows the host software to select when the DSPI master samples SIN in modified transfer format. Figure 20-38 shows where the master can sample the SIN pin. The table below lists the various delayed sample points. SMPL_PT: 00 → Number of system clock cycles between odd-numbered edge of SCK and sampling of SIN. = 0 01 = 1 10 = 2 11 = Reserved
24–30	—	Reserved.
31	HALT	Halt. Provides a mechanism for software to start and stop DSPI transfers. See Section 20.4.2, "Start and Stop of DSPI Transfers," for details on the operation of this bit. 0 Start transfers 1 Stop transfers

20.3.2.2 DSPI Transfer Count Register (DSPI*x*_TCR)

The DSPI*x*_TCR contains a counter that indicates the number of SPI transfers made. The transfer counter is intended to assist in queue management. The user must not write to the DSPI*x*_TCR while the DSPI is running.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SPI_TCNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x8															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x8															

Figure 20-4. DSPI Transfer Count Register (DSPI*x*_TCR)

Table 20-4. DSPI*x*_TCR Field Descriptions

Bits	Name	Description
0–15	SPI_TCNT [0:15]	SPI transfer counter. Counts the number of SPI transfers the DSPI makes. The SPI_TCNT field is incremented every time the last bit of an SPI frame is transmitted. A value written to SPI_TCNT presets the counter to that value. SPI_TCNT is reset to zero at the beginning of the frame when the CTCNT field is set in the executing SPI command. The transfer counter 'wraps around,' incrementing the counter past 65535 resets the counter to zero.
16–31	—	Reserved.

20.3.2.3 DSPI Clock and Transfer Attributes Registers 0–7 (DSPI*x*_CTAR*n*)

The MPC5553/MPC5554 DSPI modules each contain eight clock and transfer attribute registers (DSPI*x*_CTAR*n*) which are used to define different transfer attribute configurations. Each DSPI*x*_CTAR controls:

- Frame size
- Baud rate and transfer delay values
- Clock phase
- Clock polarity
- MSB/LSB first

DSPI*x*_CTARs support compatibility with the QSPI module in the MPC5xx family of MCUs. See Section 20.5.4, "MPC5xx QSPI Compatibility with the DSPI," for a discussion on DSPI/QSPI compatibility. At the initiation of an SPI or DSI transfer, control logic selects the DSPI*x*_CTAR that contains the transfer's attributes. The user must not write to the DSPI*x*_CTARs while the DSPI is running.

In master mode, the DSPI*x*_CTAR*n* registers define combinations of transfer attributes such as frame size, clock phase and polarity, data bit ordering, baud rate, and various delays. In slave mode, a subset of the bit fields in the DSPI*x*_CTAR0 and DSPI*x*_CTAR1 registers are used to set the slave transfer attributes. See the individual bit descriptions for details on which bits are used in slave modes.

When the DSPI is configured as an SPI master, the CTAS field in the command portion of the TX FIFO entry selects which of the DSPI*x*_CTAR registers is used on a per-frame basis. When the DSPI is configured as an SPI bus slave, the DSPI*x*_CTAR0 register is used.

When the DSPI is configured as a DSI master, the DSICTAS field in the DSPI DSI configuration register (DSPI*x*_DSICR) selects which of the DSPI*x*_CTAR register is used. For more information on the DSPI*x*_DSICR see Section 20.3.2.10, "DSPI DSI Configuration Register (DSPIx_DSICR)." When the DSPI is configured as a DSI bus slave, the DSPI*x*_CTAR1 register is used.

In CSI configuration, the transfer attributes are selected based on whether the current frame is SPI data or DSI data. SPI transfers in CSI configuration follow the protocol described for SPI configuration, and DSI transfers in CSI configuration follow the protocol described for DSI configuration. CSI configuration is only valid in conjunction with master mode. See Section 20.4.5, "Combined Serial Interface (CSI) Configuration" for more details.

.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DBR	FMSZ				CPOL	CPHA	LSBFE	PCSSCK		PASC		PDT		PBR	
Reset	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0xC (DSPI*x*_CTAR0); 0x10 (DSPI*x*_CTAR1); 0x14 (DSPI*x*_CTAR2); 0x18 (DSPI*x*_CTAR3); 0x1C (DSPI*x*_CTAR4); 0x20 (DSPI*x*_CTAR5); 0x24 (DSPI*x*_CTAR6); 0x28 (DSPI*x*_CTAR7)															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CSSCK				ASC				DT				BR			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0xC (DSPI*x*_CTAR0); 0x10 (DSPI*x*_CTAR1); 0x14 (DSPI*x*_CTAR2); 0x18 (DSPI*x*_CTAR3); 0x1C (DSPI*x*_CTAR4); 0x20 (DSPI*x*_CTAR5); 0x24 (DSPI*x*_CTAR6); 0x28 (DSPI*x*_CTAR7)															

Figure 20-5. DSPI Clock and Transfer Attributes Registers 0–7 (DSPI*x*_CTARn)

Table 20-5. DSPI*x*_CTAR*n* Field Description

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>0</td><td>DBR</td><td>Double baud rate.
The DBR bit doubles the effective baud rate of the serial communications clock (SCK). This field is only used in master mode. It effectively halves the baud rate division ratio supporting faster frequencies and odd division ratios for the serial communications clock (SCK). When the DBR bit is set, the duty cycle of the serial communications clock (SCK) depends on the value in the baud rate prescaler and the clock phase bit as listed below. See the BR field below and Section 20.4.6.1, “Baud Rate Generator” for details on how to compute the baud rate. If the overall baud rate is divide by two or divide by three of the system clock then the continuous SCK enable or the modified timing format enable bits must not be set.
0 The baud rate is computed normally with a 50/50 duty cycle
1 Baud rate is doubled with the duty cycle depending on the baud rate prescaler
<table>
<tr><th>DBR</th><th>CPHA</th><th>PBR</th><th>SCK Duty Cycle</th></tr>
<tr><td>0</td><td>any</td><td>any</td><td>50/50</td></tr>
<tr><td>1</td><td>0</td><td>00</td><td>50/50</td></tr>
<tr><td>1</td><td>0</td><td>01</td><td>33/66</td></tr>
<tr><td>1</td><td>0</td><td>10</td><td>40/60</td></tr>
<tr><td>1</td><td>0</td><td>11</td><td>43/57</td></tr>
<tr><td>1</td><td>1</td><td>00</td><td>50/50</td></tr>
<tr><td>1</td><td>1</td><td>01</td><td>66/33</td></tr>
<tr><td>1</td><td>1</td><td>10</td><td>60/40</td></tr>
<tr><td>1</td><td>1</td><td>11</td><td>57/43</td></tr>
</table>
</td></tr>
<tr><td>1–4</td><td>FMSZ
[0:3]</td><td>FMSZ. Selects the number of bits transferred per frame. The FMSZ field is used in master mode and slave mode. The table below lists the frame sizes.
<table>
<tr><th>FMSZ</th><th>Framesize</th><th>FMSZ</th><th>Framesize</th></tr>
<tr><td>0000</td><td>Reserved</td><td>1000</td><td>9</td></tr>
<tr><td>0001</td><td>Reserved</td><td>1001</td><td>10</td></tr>
<tr><td>0010</td><td>Reserved</td><td>1010</td><td>11</td></tr>
<tr><td>0011</td><td>4</td><td>1011</td><td>12</td></tr>
<tr><td>0100</td><td>5</td><td>1100</td><td>13</td></tr>
<tr><td>0101</td><td>6</td><td>1101</td><td>14</td></tr>
<tr><td>0110</td><td>7</td><td>1110</td><td>15</td></tr>
<tr><td>0111</td><td>8</td><td>1111</td><td>16</td></tr>
</table>
</td></tr>
</table>

Table 20-5. DSPI*x*_CTAR*n* Field Description (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>5</td><td>CPOL</td><td>Clock polarity. Selects the inactive state of the serial communications clock (SCK). This bit is used in both master and slave mode. For successful communication between serial devices, the devices must have identical clock polarities. When the continuous selection format is selected (CONT = 1 or DCONT = 1), switching between clock polarities without stopping the DSPI can cause errors in the transfer due to the peripheral device interpreting the switch of clock polarity as a valid clock edge. For more information on continuous selection format, refer to Section 20.4.7.5, “Continuous Selection Format.”
0 The inactive state value of SCK is low
1 The inactive state value of SCK is high</td></tr>
<tr><td>6</td><td>CPHA</td><td>Clock phase. Selects which edge of SCK causes data to change and which edge causes data to be captured. This bit is used in both master and slave mode. For successful communication between serial devices, the devices must have identical clock phase settings.
0 Data is captured on the leading edge of SCK and changed on the following edge
1 Data is changed on the leading edge of SCK and captured on the following edge</td></tr>
<tr><td>7</td><td>LSBFE</td><td>LSB first enable. Selects if the LSB or MSB of the frame is transferred first. This bit is only used in master mode.
0 Data is transferred MSB first
1 Data is transferred LSB first</td></tr>
<tr><td>8–9</td><td>PCSSCK [0:1]</td><td>PCS to SCK delay prescaler. Selects the prescaler value for the delay between assertion of PCS and the first edge of the SCK. This field is only used in master mode. The table below lists the prescaler values. The description for bitfeild CSSCK in Table 20-5 details how to compute the PCS to SCK delay.
<table>
<tr><th>PCSSCK</th><th>PCS to SCK Delay Prescaler Value</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>3</td></tr>
<tr><td>10</td><td>5</td></tr>
<tr><td>11</td><td>7</td></tr>
</table></td></tr>
<tr><td>10–11</td><td>PASC [0:1]</td><td>After SCK delay prescaler. Selects the prescaler value for the delay between the last edge of SCK and the negation of PCS. This field is only used in master mode. The table below lists the prescaler values. The description for bitfeild ASC in Table 20-5 details how to compute the after SCK delay.
<table>
<tr><th>PASC</th><th>After SCK Delay Prescaler Value</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>3</td></tr>
<tr><td>10</td><td>5</td></tr>
<tr><td>11</td><td>7</td></tr>
</table></td></tr>
</table>

Table 20-5. DSPI*x*_CTAR*n* Field Description (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>12–13</td><td>PDT
[0:1]</td><td>Delay after transfer prescaler. The PDT field selects the prescaler value for the delay between the negation of the PCS signal at the end of a frame and the assertion of PCS at the beginning of the next frame. The PDT field is only used in master mode. The table below lists the prescaler values. The description for bitfield DT in Table 20-5 details how to compute the delay after transfer.
<table>
<tr><th>PDT</th><th>Delay after Transfer Prescaler Value</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>3</td></tr>
<tr><td>10</td><td>5</td></tr>
<tr><td>11</td><td>7</td></tr>
</table></td></tr>
<tr><td>14–15</td><td>PBR
[0:1]</td><td>Baud rate prescaler. Selects the prescaler value for the baud rate. This field is only used in master mode. The baud rate is the frequency of the serial communications clock (SCK). The system clock is divided by the prescaler value before the baud rate selection takes place. The baud rate prescaler values are listed in the table below. The description for Section 20.4.6.1, “Baud Rate Generator” details how to compute the baud rate.
<table>
<tr><th>PBR</th><th>Baud Rate Prescaler Value</th></tr>
<tr><td>00</td><td>2</td></tr>
<tr><td>01</td><td>3</td></tr>
<tr><td>10</td><td>5</td></tr>
<tr><td>11</td><td>7</td></tr>
</table></td></tr>
</table>

Table 20-5. DSPI*x*_CTAR*n* Field Description (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>16–19</td><td>CSSCK
[0:3]</td><td>PCS to SCK delay scaler. Selects the scaler value for the PCS to SCK delay. This field is only used in master mode. The PCS to SCK delay is the delay between the assertion of PCS and the first edge of the SCK. The table below lists the scaler values.
<table>
<tr><th>CSSCK</th><th>PCS to SCK Delay Scaler Value</th><th>CSSCK</th><th>PCS to SCK Delay Scaler Value</th></tr>
<tr><td>0000</td><td>2</td><td>1000</td><td>512</td></tr>
<tr><td>0001</td><td>4</td><td>1001</td><td>1024</td></tr>
<tr><td>0010</td><td>8</td><td>1010</td><td>2048</td></tr>
<tr><td>0011</td><td>16</td><td>1011</td><td>4096</td></tr>
<tr><td>0100</td><td>32</td><td>1100</td><td>8192</td></tr>
<tr><td>0101</td><td>64</td><td>1101</td><td>16384</td></tr>
<tr><td>0110</td><td>128</td><td>1110</td><td>32768</td></tr>
<tr><td>0111</td><td>256</td><td>1111</td><td>65536</td></tr>
</table>
The PCS to SCK delay is a multiple of the system clock period and it is computed according to the following equation:
$t_{CSC} = \frac{1}{f_{SYS}} \times \text{PCSSCK Prescaler value} \times \text{CSSCK Scaler value}$
Note: See Section 20.4.6.2, “PCS to SCK Delay (tCSC),” for more details.</td></tr>
</table>

Table 20-5. DSPI*x*_CTAR*n* Field Description (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>20-23</td><td>ASC
[0:3]</td><td>After SCK delay scaler. Selects the scaler value for the After SCK delay. This field is only used in master mode. The after SCK delay is the delay between the last edge of SCK and the negation of PCS. The table below lists the scaler values.
<table>
<tr><th>ASC</th><th>After SCK Delay Scaler Value</th><th>ASC</th><th>After SCK Delay Scaler Value</th></tr>
<tr><td>0000</td><td>2</td><td>1000</td><td>512</td></tr>
<tr><td>0001</td><td>4</td><td>1001</td><td>1024</td></tr>
<tr><td>0010</td><td>8</td><td>1010</td><td>2048</td></tr>
<tr><td>0011</td><td>16</td><td>1011</td><td>4096</td></tr>
<tr><td>0100</td><td>32</td><td>1100</td><td>8192</td></tr>
<tr><td>0101</td><td>64</td><td>1101</td><td>16384</td></tr>
<tr><td>0110</td><td>128</td><td>1110</td><td>32768</td></tr>
<tr><td>0111</td><td>256</td><td>1111</td><td>65536</td></tr>
</table>
The after SCK delay is a multiple of the system clock period, and it is computed according to the following equation:
$$t_{ASC} = \frac{1}{t_{SYS}} \times \text{PASC Prescaler value} \times \text{ASC Scaler value}$$
Note: See Section 20.4.6.3, “After SCK Delay (tASC),” for more details.</td></tr>
</table>

Table 20-5. DSPI*x*_CTAR*n* Field Description (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>24–27</td><td>DT
[0:3]</td><td>Delay after transfer scaler. The DT field selects the delay after transfer scaler. This field is only used in master mode. The delay after transfer is the time between the negation of the PCS signal at the end of a frame and the assertion of PCS at the beginning of the next frame. The table below lists the scaler values.
<table>
<tr><th>DT</th><th>Delay after Transfer Scaler Value</th><th>DT</th><th>Delay after Transfer Scaler Value</th></tr>
<tr><td>0000</td><td>2</td><td>1000</td><td>512</td></tr>
<tr><td>0001</td><td>4</td><td>1001</td><td>1024</td></tr>
<tr><td>0010</td><td>8</td><td>1010</td><td>2048</td></tr>
<tr><td>0011</td><td>16</td><td>1011</td><td>4096</td></tr>
<tr><td>0100</td><td>32</td><td>1100</td><td>8192</td></tr>
<tr><td>0101</td><td>64</td><td>1101</td><td>16384</td></tr>
<tr><td>0110</td><td>128</td><td>1110</td><td>32768</td></tr>
<tr><td>0111</td><td>256</td><td>1111</td><td>65536</td></tr>
</table>
The delay after transfer is a multiple of the system clock period and it is computed according to the following equation:
$$t_{DT} = \frac{1}{f_{SYS}} \times \text{PDT Prescaler value} \times \text{DT Scaler value}$$
Note: See Section 20.4.6.4, “Delay after Transfer (tDT),” for more details</td></tr>
</table>

Table 20-5. DSPI*x*_CTAR*n* Field Description (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>28–31</td><td>BR
[0:3]</td><td>Baud rate scaler. Selects the scaler value for the baud rate. This field is only used in master mode. The pre-scaled system clock is divided by the baud rate scaler to generate the frequency of the SCK. The table below lists the baud rate scaler values.
<table>
<tr><th>BR</th><th>Baud Rate Scaler Value</th><th>BR</th><th>Baud Rate Scaler Value</th></tr>
<tr><td>0000</td><td>2</td><td>1000</td><td>256</td></tr>
<tr><td>0001</td><td>4</td><td>1001</td><td>512</td></tr>
<tr><td>0010</td><td>6</td><td>1010</td><td>1024</td></tr>
<tr><td>0011</td><td>8</td><td>1011</td><td>2048</td></tr>
<tr><td>0100</td><td>16</td><td>1100</td><td>4096</td></tr>
<tr><td>0101</td><td>32</td><td>1101</td><td>8192</td></tr>
<tr><td>0110</td><td>64</td><td>1110</td><td>16384</td></tr>
<tr><td>0111</td><td>128</td><td>1111</td><td>32768</td></tr>
</table>
The baud rate is computed according to the following equation:
$$\text{SCK baud rate} = \frac{f_{SYS}}{\text{PBRPrescalerValue}} \times \frac{1 + \text{DBR}}{\text{BRScalerValue}}$$
Note: See Section 20.4.6.1, “Baud Rate Generator,” for more details.</td></tr>
</table>

20.3.2.4 DSPI Status Register (DSPI*x*_SR)

The DSPI*x*_SR contains status and flag bits. The bits reflect the status of the DSPI and indicate the occurrence of events that can generate interrupt or DMA requests. Software can clear flag bits in the DSPI*x*_SR by writing a 1 to it. Writing a 0 to a flag bit has no effect.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TCF	TXRXS	0	EOQF	TFUF	0	TFFF	0	0	0	0	0	RFOF	0	RFDF	0
W	w1c	w1c		w1c	w1c		w1c						w1c		w1c	
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x2C															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXCTR				TXNXTPTR				RXCTR				POPNXTPTR			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x2C															

Figure 20-6. DSPI Status Register (DSPI*x*_SR)

Table 20-6. DSPI*x*_SR Field Descriptions

Bits	Name	Description
0	TCF	Transfer complete flag. Indicates that all bits in a frame have been shifted out. The TCF bit is set after the last incoming databit is sampled, but before the tASC delay starts. Refer to Section 20.4.7.1, "Classic SPI Transfer Format (CPHA = 0)" for details. The TCF bit is cleared by writing 1 to it. 0 Transfer not complete 1 Transfer complete
1	TXRXS	TX and RX status. Reflects the status of the DSPI. See Section 20.4.2, "Start and Stop of DSPI Transfers" for information on what causes this bit to be negated or asserted. TXRXS is cleared by writing 1 to it. 0 TX and RX operations are disabled (DSPI is in STOPPED state) 1 TX and RX operations are enabled (DSPI is in RUNNING state)
2	—	Reserved.
3	EOQF	End of queue flag. Indicates that transmission in progress is the last entry in a queue. The EOQF bit is set when the TX FIFO entry has the EOQ bit set in the command halfword and after the last incoming databit is sampled, but before the tASC delay starts. Refer to Section 20.4.7.1, "Classic SPI Transfer Format (CPHA = 0)" for details. The EOQF bit is cleared by writing 1 to it. When the EOQF bit is set, the TXRXS bit is automatically cleared. 0 EOQ is not set in the executing command 1 EOQ bit is set in the executing SPI command **Note:** EOQF does not function in slave mode.
4	TFUF	Transmit FIFO underflow flag. Indicates that an underflow condition in the TX FIFO has occurred. The transmit underflow condition is detected only for DSPI modules operating in slave mode and SPI configuration. The TFUF bit is set when the TX FIFO of a DSPI operating in SPI slave mode is empty, and a transfer is initiated by an external SPI master. The TFUF bit is cleared by writing 1 to it. 0 TX FIFO underflow has not occurred 1 TX FIFO underflow has occurred
5	—	Reserved.
6	TFFF	Transmit FIFO fill flag: indicates that the TX FIFO can be filled. Provides a method for the DSPI to request more entries to be added to the TX FIFO. The TFFF bit is set while the TX FIFO is not full. The TFFF bit can be cleared by writing 1 to it or by an acknowledgement from the eDMA controller when the TX FIFO is full. 0 TX FIFO is full 1 TX FIFO is not full
7–11	—	Reserved.
12	RFOF	Receive FIFO overflow flag. Indicates that an overflow condition in the RX FIFO has occurred. The bit is set when the RX FIFO and shift register are full and a transfer is initiated. The bit is cleared by writing 1 to it. 0 RX FIFO overflow has not occurred 1 RX FIFO overflow has occurred
13	—	Reserved.

Table 20-6. DSPI*x*_SR Field Descriptions (continued)

Bits	Name	Description
14	RFDF	Receive FIFO drain flag: indicates that the RX FIFO can be drained. Provides a method for the DSPI to request that entries be removed from the RX FIFO. The bit is set while the RX FIFO is not empty. The RFDF bit can be cleared by writing 1 to it or by an acknowledgement from the eDMA controller when the RX FIFO is empty. 0 RX FIFO is empty 1 RX FIFO is not empty **Note:** In the interrupt service routine, RFDF must be cleared only after the DSPIx_POPR register is read.
15	—	Reserved.
16–19	TXCTR [0:3]	TX FIFO counter. Indicates the number of valid entries in the TX FIFO. The TXCTR is incremented every time the DSPI _PUSHR is written. The TXCTR is decremented every time an SPI command is executed and the SPI data is transferred to the shift register.
20–23	TXNXTPTR [0:3]	Transmit next pointer. Indicates which TX FIFO Entry will be transmitted during the next transfer. The TXNXTPTR field is updated every time SPI data is transferred from the TX FIFO to the shift register. See Section 20.4.3.4, “Transmit First In First Out (TX FIFO) Buffering Mechanism” for more details.
24–27	RXCTR [0:3]	RX FIFO counter. Indicates the number of entries in the RX FIFO. The RXCTR is decremented every time the DSPI _POPR is read. The RXCTR is incremented after the last incoming databit is sampled, but before the tASC delay starts. Refer to Section 20.4.7.1, “Classic SPI Transfer Format (CPHA = 0)” for details.
28–31	POPNXTPTR [0:3]	Pop next pointer. Contains a pointer to the RX FIFO entry that will be returned when the DSPI*x*_POPR is read. The POPNXTPTR is updated when the DSPI*x*_POPR is read. See Section 20.4.3.5, “Receive First In First Out (RX FIFO) Buffering Mechanism” for more details.

20.3.2.5 DSPI DMA/Interrupt Request Select and Enable Register (DSPI*x*_RSER)

The DSPI*x*_RSER serves two purposes. It enables flag bits in the DSPI*x*_SR to generate DMA requests or interrupt requests. The DSPI*x*_RSER also selects the type of request to be generated. See the individual bit descriptions for information on the types of requests the bits support. The user must not write to the DSPI*x*_RSER while the DSPI is running.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TCF_RE	0	0	EOQF_RE	TFUF_RE	0	TFFF_RE	TFFF_DIRS	0	0	0	0	RFOF_RE	0	RFDF_RE	RFDF_DIRS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x30															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x30															

Figure 20-7. DSPI DMA/Interrupt Request Select and Enable Register (DSPI*x*_RSER)

Table 20-7. DSPI*x*_RSER Field Descriptions

Bits	Name	Description
0	TCF_RE	Transmission complete request enable. Enables TCF flag in the DSPI*x*_SR to generate an interrupt request. 0 TCF interrupt requests are disabled 1 TCF interrupt requests are enabled
1–2	—	Reserved.
3	EOQF_RE	DSPI finished request enable. Enables the EOQF flag in the DSPI*x*_SR to generate an interrupt request. 0 EOQF interrupt requests are disabled 1 EOQF interrupt requests are enabled
4	TFUF_RE	Transmit FIFO underflow request enable.The TFUF_RE bit enables the TFUF flag in the DSPI*x*_SR to generate an interrupt request. 0 TFUF interrupt requests are disabled 1 TFUF interrupt requests are enabled
5	—	Reserved.
6	TFFF_RE	Transmit FIFO fill request enable. Enables the TFFF flag in the DSPI*x*_SR to generate a request. The TFFF_DIRS bit selects between generating an interrupt request or a DMA requests. 0 TFFF interrupt requests or DMA requests are disabled 1 TFFF interrupt requests or DMA requests are enabled
7	TFFF_DIRS	Transmit FIFO fill DMA or interrupt request select. Selects between generating a DMA request or an interrupt request. When the TFFF flag bit in the DSPI*x*_SR is set, and the TFFF_RE bit in the DSPI*x*_RSER is set, this bit selects between generating an interrupt request or a DMA request. 0 Interrupt request will be generated 1 DMA request will be generated
8–11	—	Reserved.

Table 20-7. DSPI*x*_RSER Field Descriptions

Bits	Name	Description
12	RFOF_RE	Receive FIFO overflow request enable. Enables the RFOF flag in the DSPI*x*_SR to generate an interrupt requests. 0 RFOF interrupt requests are disabled 1 RFOF interrupt requests are enabled
13	—	Reserved.
14	RFDF_RE	Receive FIFO drain request enable. Enables the RFDF flag in the DSPI*x*_SR to generate a request. The RFDF_DIRS bit selects between generating an interrupt request or a DMA request. 0 RFDF interrupt requests or DMA requests are disabled 1 RFDF interrupt requests or DMA requests are enabled
15	RFDF_DIRS	Receive FIFO drain DMA or interrupt request select. Selects between generating a DMA request or an interrupt request. When the RFDF flag bit in the DSPI*x*_SR is set, and the RFDF_RE bit in the DSPI*x*_RSER is set, the RFDF_DIRS bit selects between generating an interrupt request or a DMA request. 0 Interrupt request will be generated 1 DMA request will be generated
16–31	—	Reserved.

20.3.2.6 DSPI PUSH TX FIFO Register (DSPI*x*_PUSHR)

The DSPI*x*_PUSHR provides a means to write to the TX FIFO. Data written to this register is transferred to the TX FIFO. See Section 20.4.3.4, "Transmit First In First Out (TX FIFO) Buffering Mechanism," for more information. Write accesses of 8- or 16-bits to the DSPI*x*_PUSHR will transfer 32 bits to the TX FIFO.

NOTE

Only the TXDATA field is used for DSPI slaves.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CONT	CTAS			EOQ	CTCNT	0	0	0	0	PCS5	PCS4	PCS3	PCS2	PCS1	PCS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x34															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x34															

Figure 20-8. DSPI PUSH TX FIFO Register (DSPI*x*_PUSHR)

Table 20-8. DSPI*x*_PUSHR Field Descriptions

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>0</td><td>CONT</td><td>Continuous peripheral chip select enable.Selects a continuous selection format. The bit is used in SPI master mode. The bit enables the selected PCS signals to remain asserted between transfers. See Section 20.4.7.5, “Continuous Selection Format,” for more information.
0 Return peripheral chip select signals to their inactive state between transfers
1 Keep peripheral chip select signals asserted between transfers</td></tr>
<tr><td>1–3</td><td>CTAS
[0:2]</td><td>Clock and transfer attributes select. Selects which of the DSPIx_CTARs is used to set the transfer attributes for the associated SPI frame. In SPI slave mode DSPIx_CTAR0 is used. The table below shows how the CTAS values map to the DSPIx_CTARs. There are eight DSPIx_CTARs in the MPC5553/MPC5554 DSPI implementation.
Note: The field is only used in SPI master mode.

<table>
<tr><th>CTAS</th><th>Use Clock and Transfer Attributes from</th></tr>
<tr><td>000</td><td>DSPIx_CTAR0</td></tr>
<tr><td>001</td><td>DSPIx_CTAR1</td></tr>
<tr><td>010</td><td>DSPIx_CTAR2</td></tr>
<tr><td>011</td><td>DSPIx_CTAR3</td></tr>
<tr><td>100</td><td>DSPIx_CTAR4</td></tr>
<tr><td>101</td><td>DSPIx_CTAR5</td></tr>
<tr><td>110</td><td>DSPIx_CTAR6</td></tr>
<tr><td>111</td><td>DSPIx_CTAR7</td></tr>
</table></td></tr>
<tr><td>4</td><td>EOQ</td><td>End of queue. Provides a means for host software to signal to the DSPI that the current SPI transfer is the last in a queue. At the end of the transfer the EOQF bit in the DSPIx_SR is set.
0 The SPI data is not the last data to transfer
1 The SPI data is the last data to transfer
Note: This bitfield is used only in SPI master mode</td></tr>
<tr><td>5</td><td>CTCNT</td><td>Clear SPI_TCNT. Provides a means for host software to clear the SPI transfer counter. The CTCNT bit clears the SPI_TCNT field in the DSPIx_TCR. The SPI_TCNT field is cleared before transmission of the current SPI frame begins.
0 Do not clear SPI_TCNT field in the DSPIx_TCR
1 Clear SPI_TCNT field in the DSPIx_TCR
Note: This bitfield is used only in SPI master mode</td></tr>
<tr><td>6–7</td><td>—</td><td>Reserved.</td></tr>
<tr><td>8–9</td><td>—</td><td>Reserved, but implemented. These bits are writable, but have no effect.</td></tr>
</table>

Table 20-8. DSPI*x*_PUSHR Field Descriptions

Bits	Name	Description
10–15	PCS*n*	Peripheral chip select *n*. Selects which PCS signals will be asserted for the transfer. 0 Negate the PCS*n* signal 1 Assert the PCS*n* signal **Note:** This bitfield is only used in SPI master mode
16–31	TXDATA [0:15]	Transmit data. Holds SPI data to be transferred according to the associated SPI command. **Note:** TXDATA is used in slave mode.

20.3.2.7 DSPI POP RX FIFO Register (DSPI*x*_POPR)

The DSPI*x*_POPR provides a means to read the RX FIFO. See Section 20.4.3.5, "Receive First In First Out (RX FIFO) Buffering Mechanism" for a description of the RX FIFO operations. Eight or sixteen bit read accesses to the DSPI*x*_POPR will read from the RX FIFO and update the counter and pointer.

NOTE

The DSPI*x*_POPR must not be read speculatively. For future compatibility, the TLB entry covering the DSPI*x*_POPR must be configured to be guarded.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x38															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x38															

Figure 20-9. DSPI POP RX FIFO Register (DSPI*x*_POPR)

Table 20-9. DSPI*x*_POPR Field Descriptions

Bits	Name	Description
0–15	—	Reserved, should be cleared.
16–31	RXDATA [0:15]	Received data. The RXDATA field contains the SPI data from the RX FIFO entry pointed to by the pop next data pointer (POPNXTPTR).

20.3.2.8 DSPI Transmit FIFO Registers 0–3 (DSPI*x*_TXFR*n*)

The DSPI*x*_TXFR*n* registers provide visibility into the TX FIFO for debugging purposes. Each register is an entry in the TX FIFO. The registers are read-only and cannot be modified. Reading the DSPI*x*_TXFR*n* registers does not alter the state of the TX FIFO. The MPC5553/MPC5554 uses four registers to implement the TX FIFO, that is DSPI*x*_TXFR0–DSPI*x*_TXFR3 are used.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXCMD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x3C (DSPI*x*_TXFR0); 0x40 (DSPI*x*_TXFR1); 0x44 (DSPI*x*_TXFR2); 0x48 (DSPI*x*_TXFR3)															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x3C (DSPI*x*_TXFR0); 0x40 (DSPI*x*_TXFR1); 0x44 (DSPI*x*_TXFR2); 0x48 (DSPI*x*_TXFR3)															

Figure 20-10. DSPI Transmit FIFO Register 0–3 (DSPI*x*_TXFR*n*)

Table 20-10. DSPI*x*_TXFR*n* Field Descriptions

Bits	Name	Description
0–15	TXCMD [0:15]	Transmit command. Contains the command that sets the transfer attributes for the SPI data. See Section 20.3.2.6, “DSPI PUSH TX FIFO Register (DSPIx_PUSHR),” for details on the command field.
16–31	TXDATA [0:15]	Transmit data. Contains the SPI data to be shifted out.

20.3.2.9 DSPI Receive FIFO Registers 0–3 (DSPI*x*_RXFR*n*)

The DSPI*x*_RXFR*n* registers provide visibility into the RX FIFO for debugging purposes. Each register is an entry in the RX FIFO. The DSPI*x*_RXFR registers are read-only. Reading the DSPI*x*_RXFR*n* registers does not alter the state of the RX FIFO. The MPC5553/MPC5554 uses four registers to implement the RX FIFO, that is DSPI*x*_RXFR0–DSPI*x*_RXFR3 are used.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x7C (DSPI*x*_RXFR0); 0x80 (DSPI*x*_RXFR1); 0x84 (DSPI*x*_RXFR2); 0x88 (DSPI*x*_RXFR3)															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x7C (DSPI*x*_RXFR0); 0x80 (DSPI*x*_RXFR1); 0x84 (DSPI*x*_RXFR2); 0x88 (DSPI*x*_RXFR3)															

Figure 20-11. DSPI Receive FIFO Registers 0–3 (DSPI*x*_RXFR*n*)

Table 20-11. DSPI*x*_RXFR*n* Field Description

Bits	Name	Description
0–15	—	Reserved, should be cleared.
16–31	RXDATA [15:0]	Receive data. Contains the received SPI data.

20.3.2.10 DSPI DSI Configuration Register (DSPI*x*_DSICR)

The DSPI*x*_DSICR selects various attributes associated with DSI and CSI configurations. The user must not write to the DSPI*x*_DSICR while the DSPI is running.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MTOE	0	MTOCNT						0	0	0	0	TXSS	TPOL	TRRE	CID
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0xBC															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DCONT	DSICTAS			0	0	0	0	0	0	DPCS 5	DPCS 4	DPCS 3	DPCS 2	DPCS 1	DPCS 0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0xBC															

Figure 20-12. DSPI DSI Configuration Register (DSPI*x*_DSICR)

Table 20-12. DSPI*x*_DSICR Field Descriptions

Bits	Name	Description
0	MTOE	Multiple transfer operation enable. Enables multiple DSPIs to be connected in a parallel or serial configuration. See Section 20.4.4.7, "Multiple Transfer Operation (MTO)," for more information. 0 Multiple transfer operation disabled 1 Multiple transfer operation enabled
1	—	Reserved.
2–7	MTOCNT [0:5]	Multiple transfer operation count. Selects number of bits to be shifted out during a transfer in multiple transfer operation. The field sets the number of SCK cycles that the bus master will generate to complete the transfer. The number of SCK cycles used will be one more than the value in the MTOCNT field. The number of SCK cycles defined by MTOCNT must be equal to or greater than the frame size.
8–11	—	Reserved.
12	TXSS	Transmit data source select. Selects the source of data to be serialized. The source can be either data from host software written to the DSPI DSI alternate serialization data register (DSPI*x*_ASDR), or parallel output pin states latched into the DSPI DSI serialization data register (DSPI*x*_SDR). 0 Source of serialized data is the DSPI*x*_SDR 1 Source of serialized data is the DSPI*x*_ASDR
13	TPOL	Trigger polarity. Selects the active edge of the internal hardware trigger input signal (*ht*). The bit selects which edge will initiate a transfer in the DSI configuration. See Section 20.4.4.5, "DSI Transfer Initiation Control," for more information. 0 Falling edge will initiate a transfer 1 Rising edge will initiate a transfer
14	TRRE	Trigger reception enable. Enables the DSPI to initiate a transfer when an external trigger signal is received. The bit is only valid in DSI configuration. See Section 20.4.4.5, "DSI Transfer Initiation Control," for more information. 0 Trigger signal reception disabled 1 Trigger signal reception enabled
15	CID	Change in data transfer enable. Enables a change in serialization data to initiate a transfer. The bit is used in master mode in DSI and CSI configurations to control when to initiate transfers. When the CID bit is set, serialization is initiated when the current DSI data differs from the previous DSI data shifted out. The DSPI*x*_COMPR is compared with the DSPI*x*_SDR or DSPI*x*_ASDR to detect a change in data. Refer to Section 20.4.4.5, "DSI Transfer Initiation Control," for more information. 0 Change in data transfer operation disabled 1 Change in data transfer operation enabled
16	DCONT	DSI continuous peripheral chip select enable. Enables the PCS signals to remain asserted between transfers. The DCONT bit only affects the PCS signals in DSI master mode. See Section 20.4.7.5, "Continuous Selection Format," for details. 0 Return peripheral chip select signals to their inactive state after transfer is complete 1 Keep peripheral chip select signals asserted after transfer is complete

Table 20-12. DSPI*x*_DSICR Field Descriptions (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>17–19</td><td>DSICTAS
[0:2]</td><td>DSI clock and transfer attributes select. The DSICTAS field selects which of the DSPIx_CTARs is used to provide transfer attributes in DSI configuration. The DSICTAS field is used in DSI master mode. In DSI slave mode, the DSPIx_CTAR1 is always selected. The table below shows how the DSICTAS values map to the DSPIx_CTARs.
<table>
<tr><th>DSICTAS</th><th>DSI Clock and Transfer Attributes Controlled by</th></tr>
<tr><td>000</td><td>DSPIx_CTAR0</td></tr>
<tr><td>001</td><td>DSPIx_CTAR1</td></tr>
<tr><td>010</td><td>DSPIx_CTAR2</td></tr>
<tr><td>011</td><td>DSPIx_CTAR3</td></tr>
<tr><td>100</td><td>DSPIx_CTAR4</td></tr>
<tr><td>101</td><td>DSPIx_CTAR5</td></tr>
<tr><td>110</td><td>DSPIx_CTAR6</td></tr>
<tr><td>111</td><td>DSPIx_CTAR7</td></tr>
</table></td></tr>
<tr><td>20–23</td><td>—</td><td>Reserved.</td></tr>
<tr><td>24–25</td><td>—</td><td>Reserved, but implemented. These bits are writable, but have no effect.</td></tr>
<tr><td>26–31</td><td>DPCSn</td><td>DSI peripheral chip select n. The DPCS bits select which of the PCS signals to assert during a DSI transfer. The DPCS bits only control the assertions of the PCS signals in DSI master mode.
0 Negate PCSn
1 Assert PCSn</td></tr>
</table>

20.3.2.11 DSPI DSI Serialization Data Register (DSPI*x*_SDR)

The DSPI*x*_SDR contains the signal states of the parallel input signals from the eTPU or the eMIOS. The pin states of the parallel input signals are latched into the DSPI*x*_SDR on the rising edge of every system clock. The DSPI*x*_SDR is read-only. When the TXSS bit in the DSPI*x*_DSICR is negated, the data in the DSPI*x*_SDR is the source of the serialized data.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0xC0															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SER_DATA [15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0xC0															

Figure 20-13. DSPI DSI Serialization Data Register (DSPI*x*_SDR)

Table 20-13. DSPI*x*_SDR Field Description

Bits	Name	Description
0–15	—	Reserved.
16–31	SER_DATA [15:0]	Serialized data. The SER_DATA field contains the signal states of the parallel input signals. SER_DATA [15:0] maps to DSPI serialization inputs IN[15:0]. Refer to Section 20.4.4.6, “DSPI Connections to eTPU_A, eTPU_B, EMIOS and SIU.”

20.3.2.12 DSPI DSI Alternate Serialization Data Register (DSPI*x*_ASDR)

The DSPI*x*_ASDR provides a means for host software to write the data to be serialized. When the TXSS bit in the DSPI*x*_DSICR is set, the data in the DSPI*x*_ASDR is the source of the serialized data. Writes to the DSPI*x*_ASDR take effect on the next frame boundary.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0xC4															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ASER_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0xC4															

Figure 20-14. DSPI DSI Alternate Serialization Data Register (DSPI*x*_ASDR)

Table 20-14. DSPI*x*_ASDR Field Description

Bits	Name	Description
0–15	—	Reserved.
16–31	ASER_DATA [0:15]	Alternate serialized data. The ASER_DATA field holds the alternate data to be serialized.

20.3.2.13 DSPI DSI Transmit Comparison Register (DSPI*x*_COMPR)

The DSPI*x*_COMPR holds a copy of the last transmitted DSI data. The DSPI*x*_COMPR is read-only. DSI data is transferred to this register as it is loaded into the TX shift register.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0xC8															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COMP_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0xC8															

Figure 20-15. DSPI DSI Transmit Comparison Register (DSPI*x*_COMPR)

Table 20-15. DSPI*x*_COMPR Field Description

Bits	Name	Description
0–15	—	Reserved.
16–31	COMP_DATA [0:15]	Compare data. The COMP_DATA field holds the last serialized DSI data.

20.3.2.14 DSPI DSI Deserialization Data Register (DSPI*x*_DDR)

The DSPI*x*_DDR holds the signal states for the parallel output signals. The DSPI*x*_DDR is read-only and it is memory mapped so that host software can read the incoming DSI frames.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0xCC															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DESER_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0xCC															

Figure 20-16. DSPI Deserialization Data Register (DSPI*x*_DDR)

Table 20-16. DSPI*x*_DDR Field Description

Bits	Name	Description
0–15	—	Reserved.
16–31	DESER_DATA	Deserialized data. Holds deserialized data which is presented as signal states to the parallel output signals.

20.4 Functional Description

The DSPI supports full-duplex, synchronous serial communications between the MPC5553/MPC5554 and peripheral devices. The DSPI can also be used to reduce the number of pins required for I/O by serializing and deserializing up to 16 parallel input/output signals from the eTPU and eMIOS. All communications are through an SPI-like protocol.

The DSPI has three configurations:

- SPI configuration in which the DSPI operates as a basic SPI or a queued SPI.
- DSI configuration in which the DSPI serializes and deserializes parallel input/output signals or bits from memory mapped registers.
- CSI configuration in which the DSPI combines the functionality of the SPI and DSI configurations.

The DCONF field in the DSPI*x*_MCR register determines the DSPI configuration. See Table 20-3 for the DSPI configuration values.

The DSPI*x*_CTAR0–DSPI*x*_CTAR7 registers hold clock and transfer attributes. The manner in which a CTAR is selected depends on the DSPI configuration (SPI, DSI, or CSI). The SPI configuration can select which CTAR to use on a frame by frame basis by setting the CTAS field in the DSPI*x*_PUSHR. The DSI configuration statically selects which CTAR to use. In CSI configuration, priority logic determines if SPI data or DSI data is transferred. The type of data transferred (whether DSI or SPI) dictates which CTAR the CSI configuration will use. See Section 20.3.2.3, “DSPI Clock and Transfer Attributes Registers 0–7 (DSPIx_CTARn),” for information on DSPI*x*_CTAR fields.

The 16-bit shift register in the master and the 16-bit shift register in the slave are linked by the SOUT and SIN signals to form a distributed 32-bit register. When a data transfer operation is performed, data is

serially shifted a pre-determined number of bit positions. Because the registers are linked, data is exchanged between the master and the slave; the data that was in the master's shift register is now in the shift register of the slave, and vice versa. At the end of a transfer, the TCF bit in the DSPI*x*_SR is set to indicate a completed transfer. Figure 20-17 illustrates how master and slave data is exchanged.

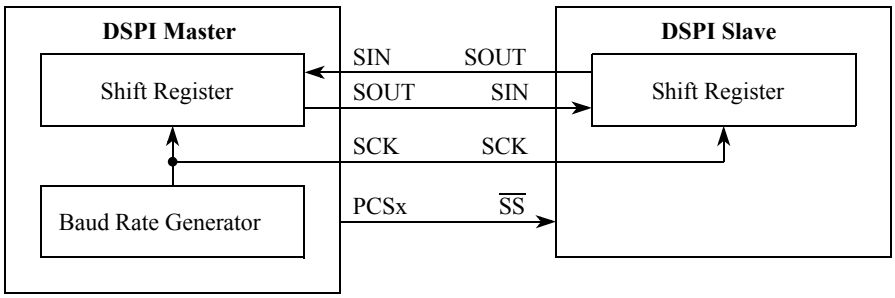


Figure 20-17. SPI and DSI Serial Protocol Overview

The DSPI has six peripheral chip select (PCS) signals that are be used to select which of the slaves to communicate with.

Transfer protocols and timing properties are shared by the three DSPI configurations; these properties are described independently of the configuration in Section 20.4.7, "Transfer Formats." The transfer rate and delay settings are described in section Section 20.4.6, "DSPI Baud Rate and Clock Delay Generation."

See Section 20.4.10, "Power Saving Features" for information on the power-saving features of the DSPI.

20.4.1 Modes of Operation

The MPC5553/MPC5554 DSPIs have four available distinct modes:

- Master mode
- Slave mode
- Module disable mode
- Debug mode

Master, slave, and module disable modes are module-specific modes while debug mode is a MPC5553/MPC5554-specific mode.

The module-specific modes are determined by bits in the DSPI*x*_MCR. Debug mode is a mode that the entire MPC5553/MPC5554 can enter in parallel with the DSPI being configured in one of its module-specific modes.

20.4.1.1 Master Mode

In master mode the DSPI can initiate communications with peripheral devices. The DSPI operates as bus master when the MSTR bit in the DSPI*x*_MCR is set. The serial communications clock (SCK) is controlled by the master DSPI. All three DSPI configurations are valid in master mode.

In SPI configuration, master mode transfer attributes are controlled by the SPI command in the current TX FIFO entry. The CTAS field in the SPI command selects which of the eight DSPI*x*_CTARs will be used to set the transfer attributes. Transfer attribute control is on a frame by frame basis. See Section 20.4.3, "Serial Peripheral Interface (SPI) Configuration" for more details.

In DSI configuration, master mode transfer attributes are controlled by the DSPI*x*_DSCIR. A detailed description of the DSPI*x*_DSCIR is located in Section 20.3.2.10, "DSPI DSI Configuration Register

(DSPIx_DSICR)." The DSISCTAS field in the DSPI*x*_DSICR selects which of the DSPI*x*_CTARs will be used to set the transfer attributes. Transfer attributes are set up during initialization and must not be changed between frames. See Section 20.4.4, "Deserial Serial Interface (DSI) Configuration," for more details.

The CSI configuration is only available in master mode. In CSI configuration, the DSI data is transferred using DSI configuration transfer attributes and SPI data is transferred using the SPI configuration transfer attributes. In order for the bus slave to distinguish between DSI and SPI frames, the transfer attributes for the two types of frames must utilize different peripheral chip select signals. See Section 20.4.5, "Combined Serial Interface (CSI) Configuration," for details.

20.4.1.2 Slave Mode

In slave mode the DSPI responds to transfers initiated by an SPI master. The DSPI operates as bus slave when the MSTR bit in the DSPI*x*_MCR is negated. The DSPI slave is selected by a bus master by having the slave's $\overline{SS}$ asserted. In slave mode the SCK is provided by the bus master. All transfer attributes are controlled by the bus master but clock polarity, clock phase and numbers of bits to transfer must still be configured in the DSPI slave for proper communications.

The SPI and DSI configurations are valid in slave mode. CSI configuration is not available in slave mode. In SPI slave mode the slave transfer attributes are set in the DSPI*x*_CTAR0. In DSI slave mode the slave transfer attributes are set in the DSPI*x*_CTAR1. In slave mode, for both SPI and DSI configurations, data is transferred MSB first. The LSBFE field of the associated CTAR is ignored.

20.4.1.3 Module Disable Mode

The module disable mode is used for MCU power management. The clock to the non-memory mapped logic in the DSPI is stopped while in module disable mode. The DSPI enters the module disable mode when the MDIS bit in DSPI*x*_MCR is set. See Section 20.4.10, "Power Saving Features," for more details on the module disable mode.

20.4.1.4 Debug Mode

The debug mode is used for system development and debugging. If the MPC5553/MPC5554 enters debug mode while the FRZ bit in the DSPI*x*_MCR is set, the DSPI stops all serial transfers and enters a stopped state. If the MPC5553/MPC5554 enters debug mode while the FRZ bit is negated, the DSPI behavior is unaffected and remains dictated by the module-specific mode and configuration of the DSPI. The DSPI enters debug mode when a debug request is asserted by an external controller. See Figure 20-18 for a state diagram.

20.4.2 Start and Stop of DSPI Transfers

The DSPI has two operating states; STOPPED and RUNNING. The states are independent of DSPI configuration. The default state of the DSPI is STOPPED. In the STOPPED state no serial transfers are initiated in master mode and no transfers are responded to in slave mode. The STOPPED state is also a safe state for writing the various configuration registers of the DSPI without causing undetermined results. The TXRXS bit in the DSPI*x*_SR is negated in this state. In the RUNNING state, serial transfers take place. The TXRXS bit in the DSPI*x*_SR is asserted in the RUNNING state. Figure 20-18 shows a state diagram of the start and stop mechanism. The transitions are described in Table 20-17.

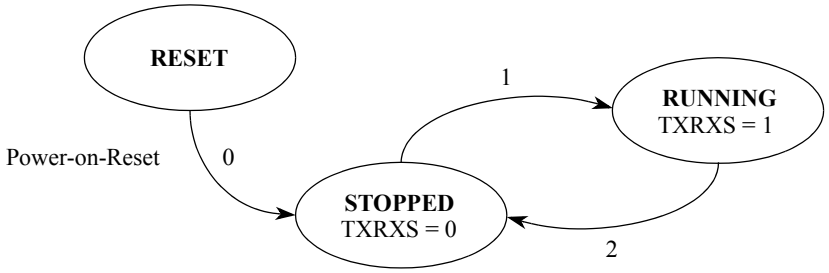


Figure 20-18. DSPI Start and Stop State Diagram

Table 20-17. State Transitions for Start and Stop of DSPI Transfers

Transition #	Current State	Next State	Description
0	RESET	STOPPED	Generic power-on-reset transition
1	STOPPED	RUNNING	The DSPI is started (DSPI transitions to RUNNING) when all of the following conditions are true: • EOQF bit is clear • Debug mode is unselected or the FRZ bit is clear • HALT bit is clear
2	RUNNING	STOPPED	The DSPI stops (transitions from RUNNING to STOPPED) after the current frame for any one of the following conditions: • EOQF bit is set • Debug mode is selected and the FRZ bit is set • HALT bit is set

State transitions from RUNNING to STOPPED occur on the next frame boundary if a transfer is in progress, or on the next system clock cycle if no transfers are in progress.

20.4.3 Serial Peripheral Interface (SPI) Configuration

The SPI configuration transfers data serially using a shift register and a selection of programmable transfer attributes. The DSPI is in SPI configuration when the DCONF field in the DSPI*x*_MCR is 0b00. The SPI frames can be from 4 to 16 bits long. The data to be transmitted can come from queues stored in RAM external to the DSPI. Host software or an eDMA controller can transfer the SPI data from the queues to a first-in first-out (FIFO) buffer. The received data is stored in entries in the receive FIFO (RX FIFO) buffer. Host software or an eDMA controller transfers the received data from the RX FIFO to memory external to the DSPI. The FIFO buffer operations are described in Section 20.4.3.4, "Transmit First In First Out (TX FIFO) Buffering Mechanism," and Section 20.4.3.5, "Receive First In First Out (RX FIFO) Buffering Mechanism." The interrupt and DMA request conditions are described in Section 20.4.9, "Interrupts/DMA Requests."

The SPI configuration supports two module-specific modes; master mode and slave mode. The FIFO operations are similar for the master mode and slave mode. The main difference is that in master mode the DSPI initiates and controls the transfer according to the fields in the SPI command field of the TX FIFO entry. In slave mode the DSPI only responds to transfers initiated by a bus master external to the DSPI and the SPI command field of the TX FIFO entry is ignored.

20.4.3.1 SPI Master Mode

In SPI master mode the DSPI initiates the serial transfers by controlling the serial communications clock (SCK) and the peripheral chip select (PCS) signals. The SPI command field in the executing TX FIFO entry determines which CTARs will be used to set the transfer attributes and which PCS signal to assert. The command field also contains various bits that help with queue management and transfer protocol. See Section 20.3.2.6, "DSPI PUSH TX FIFO Register (DSPIx_PUSHR)," for details on the SPI command fields. The data field in the executing TX FIFO entry is loaded into the shift register and shifted out on the serial out (SOUT) pin. In SPI master mode, each SPI frame to be transmitted has a command associated with it allowing for transfer attribute control on a frame by frame basis.

20.4.3.2 SPI Slave Mode

In SPI slave mode the DSPI responds to transfers initiated by an SPI bus master. The DSPI does not initiate transfers. Certain transfer attributes such as clock polarity, clock phase and frame size must be set for successful communication with an SPI master. The SPI slave mode transfer attributes are set in the DSPI*x*_CTAR0.

20.4.3.3 FIFO Disable Operation

The FIFO disable mechanisms allow SPI transfers without using the TX FIFO or RX FIFO. The DSPI operates as a double-buffered simplified SPI when the FIFOs are disabled. The TX and RX FIFOs are disabled separately. The TX FIFO is disabled by writing a 1 to the DIS_TXF bit in the DSPI*x*_MCR. The RX FIFO is disabled by writing a 1 to the DIS_RXF bit in the DSPI*x*_MCR.

The FIFO disable mechanisms are transparent to the user and to host software; transmit data and commands are written to the DSPI*x*_PUSHR and received data is read from the DSPI*x*_POPR. When the TX FIFO is disabled, the TFFF, TFUF, and TXCTR fields in DSPI*x*_SR behave as if there is a one-entry FIFO but the contents of the DSPI*x*_TXFRs and TXNXTPTR are undefined. When the RX FIFO is disabled, the RFDF, RFOF, and RXCTR fields in the DSPI*x*_SR behave as if there is a one-entry FIFO but the contents of the DSPI*x*_RXFRs and POPNXTPTR are undefined.

The TX and RX FIFOs should be disabled only if the application's operating mode requires the FIFO to be disabled. A FIFO must be disabled before it is accessed. Failure to disable a FIFO prior to a first FIFO access is not supported, and may result in incorrect results.

20.4.3.4 Transmit First In First Out (TX FIFO) Buffering Mechanism

The TX FIFO functions as a buffer of SPI data and SPI commands for transmission. The TX FIFO holds four entries, each consisting of a command field and a data field. SPI commands and data are added to the TX FIFO by writing to the DSPI push TX FIFO register (DSPI*x*_PUSHR). For more information on DSPI*x*_PUSHR, refer to Section 20.3.2.6, "DSPI PUSH TX FIFO Register (DSPIx_PUSHR)." TX FIFO entries can only be removed from the TX FIFO by being shifted out or by flushing the TX FIFO.

The TX FIFO counter field (TXCTR) in the DSPI status register (DSPI*x*_SR) indicates the number of valid entries in the TX FIFO. The TXCTR is updated every time the DSPI _PUSHR is written or SPI data is transferred into the shift register from the TX FIFO. For more information on DSPI*x*_SR, refer to Section 20.3.2.4, "DSPI Status Register (DSPIx_SR)."

The TXNXTPTR field indicates which TX FIFO entry will be transmitted during the next transfer. The TXNXTPTR contains the positive offset from DSPI*x*_TXFR0 in number of 32-bit registers. For example, TXNXTPTR equal to two means that the DSPI*x*_TXFR2 contains the SPI data and command for the next

transfer. The TXNXTPTR field is incremented every time SPI data is transferred from the TX FIFO to the shift register.

20.4.3.4.1 Filling the TX FIFO

Host software or the eDMA controller can add (push) entries to the TX FIFO by writing to the DSPI*x*_PUSHR. When the TX FIFO is not full, the TX FIFO fill flag (TFFF) in the DSPI*x*_SR is set. The TFFF bit is cleared when the TX FIFO is full and the eDMA controller indicates that a write to DSPI*x*_PUSHR is complete or alternatively by host software writing a 1 to the TFFF in the DSPI*x*_SR. The TFFF can generate a DMA request or an interrupt request. See Section 20.4.9.2, "Transmit FIFO Fill Interrupt or DMA Request (TFFF)," for details.

The DSPI ignores attempts to push data to a full TX FIFO; that is, the state of the TX FIFO is unchanged. No error condition is indicated.

20.4.3.4.2 Draining the TX FIFO

The TX FIFO entries are removed (drained) by shifting SPI data out through the shift register. Entries are transferred from the TX FIFO to the shift register and shifted out as long as there are valid entries in the TX FIFO. Every time an entry is transferred from the TX FIFO to the shift register, the TX FIFO counter is decremented by one. At the end of a transfer, the TCF bit in the DSPI*x*_SR is set to indicate the completion of a transfer. The TX FIFO is flushed by writing a 1 to the CLR_TXF bit in DSPI*x*_MCR.

If an external SPI bus master initiates a transfer with a DSPI slave while the slave's DSPI TX FIFO is empty, the transmit FIFO underflow flag (TFUF) in the slave's DSPI*x*_SR is set. See Section 20.4.9.4, "Transmit FIFO Underflow Interrupt Request (TFUF),"for details.

20.4.3.5 Receive First In First Out (RX FIFO) Buffering Mechanism

The RX FIFO functions as a buffer for data received on the SIN pin. The RX FIFO holds four received SPI data frames. SPI data is added to the RX FIFO at the completion of a transfer when the received data in the shift register is transferred into the RX FIFO. SPI data is removed (popped) from the RX FIFO by reading the DSPI*x*_POPR register. RX FIFO entries can only be removed from the RX FIFO by reading the DSPI*x*_POPR or by flushing the RX FIFO. For more information on the DSPI*x*_POPR, refer to Section 20.3.2.7, "DSPI POP RX FIFO Register (DSPIx_POPR)."

The RX FIFO counter field (RXCTR) in the DSPI status register (DSPI*x*_SR) indicates the number of valid entries in the RX FIFO. The RXCTR is updated every time the DSPI _POPR is read or SPI data is copied from the shift register to the RX FIFO.

The POPNXTPTR field in the DSPI*x*_SR points to the RX FIFO entry that is returned when the DSPI*x*_POPR is read. The POPNXTPTR contains the positive, 32-bit word offset from DSPI*x*_RXFR0. For example, POPNXTPTR equal to two means that the DSPI*x*_RXFR2 contains the received SPI data that will be returned when DSPI*x*_POPR is read. The POPNXTPTR field is incremented every time the DSPI*x*_POPR is read. POPNXTPTR rolls over every four frames on the MPC5553/MPC5554.

20.4.3.5.1 Filling the RX FIFO

The RX FIFO is filled with the received SPI data from the shift register. While the RX FIFO is not full, SPI frames from the shift register are transferred to the RX FIFO. Every time an SPI frame is transferred to the RX FIFO the RX FIFO counter is incremented by one.

If the RX FIFO and shift register are full and a transfer is initiated, the RFOF bit in the DSPI*x*_SR is asserted indicating an overflow condition. Depending on the state of the ROOE bit in the DSPI*x*_MCR, the data from the transfer that generated the overflow is either ignored or shifted in to the shift register. If

the ROOE bit is asserted, the incoming data is shifted in to the shift register. If the ROOE bit is negated, the incoming data is ignored.

20.4.3.5.2 Draining the RX FIFO

Host software or the eDMA can remove (pop) entries from the RX FIFO by reading the DSPI*x*_POPR. For more information on DSPI*x*_POPR, refer to Section 20.3.2.7, "DSPI POP RX FIFO Register (DSPIx_POPR)." A read of the DSPI*x*_POPR decrements the RX FIFO counter by one. Attempts to pop data from an empty RX FIFO are ignored, the RX FIFO counter remains unchanged. The data returned from reading an empty RX FIFO is undetermined.

When the RX FIFO is not empty, the RX FIFO drain flag (RFDF) in the DSPI*x*_SR is set. The RFDF bit is cleared when the RX_FIFO is empty and the eDMA controller indicates that a read from DSPI*x*_POPR is complete; alternatively the RFDF bit can be cleared by the host writing a 1 to it.

20.4.4 Deserial Serial Interface (DSI) Configuration

The DSI configuration supports pin count reduction by serializing parallel input signals or register bits and shifting them out in an SPI-like protocol. The timing and transfer protocol is described in Section 20.4.7, "Transfer Formats." The received serial frames are converted to a parallel form (deserialized) and placed on the parallel output signals or in a register. The various features of the DSI configuration are set in the DSPI*x*_DSICR. For more information on the DSPI*x*_DSICR, refer to Section 20.3.2.10, "DSPI DSI Configuration Register (DSPIx_DSICR)." The DSPI is in DSI configuration when the DCONF field in the DSPI*x*_MCR is 0b01.

The DSI frames can be from 4 to 16 bits long. With multiple transfer operation (MTO), the DSPI supports serial chaining of DSPI modules within the MPC5553/MPC5554 to create DSI frames consisting of concatenated bits from multiple DSPIs. The DSPI also supports parallel chaining allowing several DSPIs and off-chip SPI devices to share the same serial communications clock (SCK) and peripheral chip select (PCS) signals. See Section 20.4.4.7, "Multiple Transfer Operation (MTO)," for details on the serial and parallel chaining support.

20.4.4.1 DSI Master Mode

In DSI master mode the DSPI initiates and controls the DSI transfers. The DSI master has four different conditions that can initiate a transfer:

- Continuous
- Change in data
- Trigger signal
- Trigger signal combined with a change in data

The four transfer initiation conditions are described in Section 20.4.4.5, "DSI Transfer Initiation Control." Transfer attributes are set during initialization. The DSICTAS field in the DSPI*x*_DSICR determines which of the DSPI*x*_CTARs will control the transfer attributes.

20.4.4.2 DSI Slave Mode

In DSI slave mode the DSPI responds to transfers initiated by an SPI or DSI bus master. In this mode the DSPI does not initiate DSI transfers. Certain transfer attributes such as clock polarity and phase must be set for successful communication with a DSI master. The DSI slave mode transfer attributes are set in the DSPI*x*_CTAR1.

If the CID bit in the DSPI*x*_DSICR is set and the data in the DSPI*x*_COMPR differs from the selected source of the serialized data, the slave DSPI will assert the MTRIG signal. If the slave's internal hardware trigger signal is asserted and the TRRE is set, the slave DSPI asserts MTRIG. These features are included to support chaining of several DSPI. Details about the MTRIG signal are found in Section 20.4.4.7, "Multiple Transfer Operation (MTO)."

20.4.4.3 DSI Serialization

In the DSI configuration, 4 to 16 bits can be serialized using two different sources. The TXSS bit in the DSPI*x*_DSICR selects between the DSPI*x*_SDR and DSPI*x*_ASDR as the source of serialized data. See Section 20.3.2.11, "DSPI DSI Serialization Data Register (DSPIx_SDR)," and Section 20.3.2.12, "DSPI DSI Alternate Serialization Data Register (DSPIx_ASDR)," for more details. The DSPI*x*_SDR holds the latest parallel input signal values which is sampled at every rising edge of the system clock. The DSPI*x*_ASDR is written by host software and used as an alternate source of serialized data.

A copy of the last DSI frame shifted out of the shift register is stored in the DSPI*x*_COMPR. This register provides added visibility for debugging and it serves as a reference for transfer initiation control. Figure 20-19 shows the DSI serialization logic. Section 20.3.2.13, "DSPI DSI Transmit Comparison Register (DSPIx_COMPR)," contains details on the DSPI*x*_COMPR.

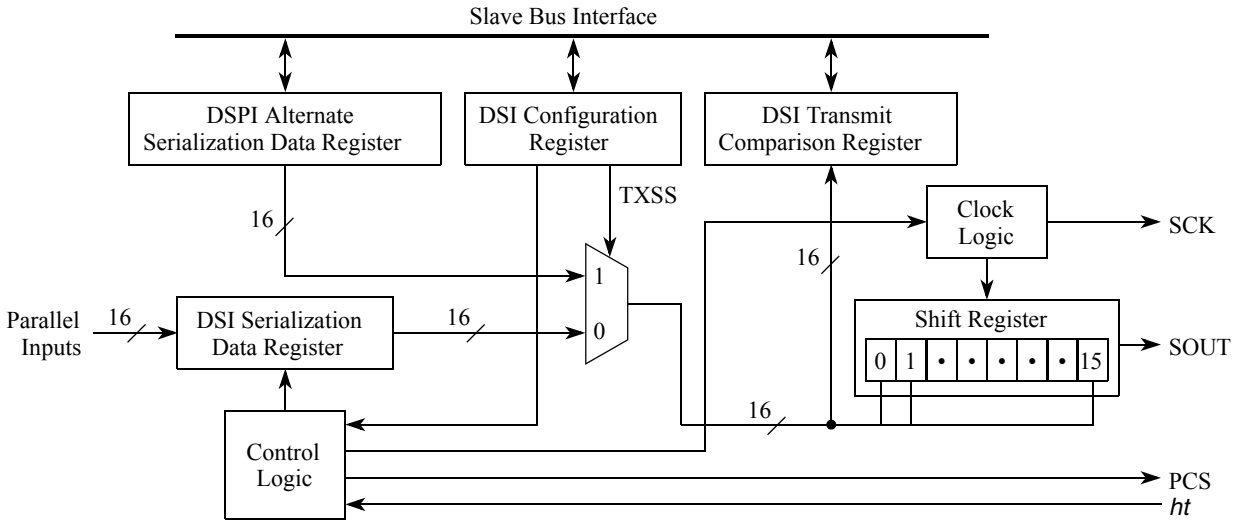


Figure 20-19. DSI Serialization Diagram

20.4.4.4 DSI Deserialization

When all bits in a DSI frame have been shifted in, the frame is copied to the DSPI*x*_DDR. This register presents the deserialized data as parallel output signal values. The DSPI*x*_DDR is memory mapped to allow host software to read the deserialized data directly. Figure 20-20 shows the DSI deserialization logic. for more information on the DSPI*x*_DDR, refer to Section 20.3.2.14, "DSPI DSI Deserialization Data Register (DSPIx_DDR)."

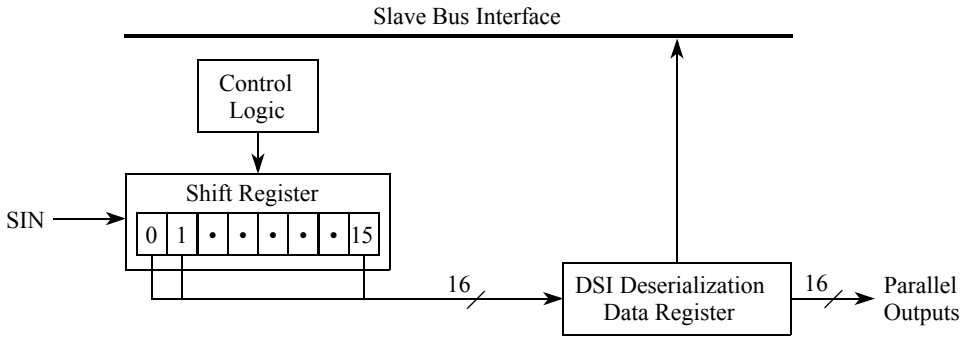


Figure 20-20. DSI Deserialization Diagram

20.4.4.5 DSI Transfer Initiation Control

Data transfers for a master DSPI in DSI configuration are initiated by a condition. When chaining DSPIs, the master and all slaves must be configured for the transfer initiation. The transfer initiation conditions are selected by the TRRE and CID bits in the DSPI*x*_DSICR. Table 20-18 lists the four transfer initiation conditions.

Table 20-18. DSI Data Transfer Initiation Control

DSPI*x*_DSICR Bits		
TRRE	**CID**	**Type of Transfer Initiation Control**
0	0	Continuous
0	1	Change in Data
1	0	Triggered
1	1	Triggered or Change in Data

20.4.4.5.1 Continuous Control

For continuous control, the initiation of a transfer is based on the baud rate at which data is transferred between the DSPI and the external device. The baud rate is set in the DSPI*x*_CTAR selected by the DSICTAS field in the DSPI*x*_DSICR. A new DSI frame shifts out when the previous transfer cycle has completed and the delay after transfer (t_{DT}) has elapsed.

20.4.4.5.2 Change In Data Control

For change in data control, a transfer is initiated when the data to be serialized has changed since the transfer of the last DSI frame. A copy of the previously transferred DSI data is stored in the DSPI*x*_COMPR. When the data in the DSPI*x*_SDR or the DSPI*x*_ASDR is different from the data in the DSPI*x*_COMPR, a new DSI frame is transmitted. The TXSS bit in the DSPI*x*_DSICR selects which register the DSPI*x*_COMPR is compared to. The $\overline{\text{MTRIG}}$ output signal is asserted every time a change in data is detected.

20.4.4.5.3 Triggered Control

For triggered control, initiation of a transfer is controlled by the internal hardware trigger signal (*ht*). The TPOL bit in the DSPI*x*_DSICR selects the active edge of *ht*. For *ht* to have any affect, the TRRE bit in the DSPI*x*_DSICR must be set.

20.4.4.5.4 Triggered or Change In Data Control

For triggered or change in data control, initiation of a transfer is controlled by the *ht* signal or by the detection of a change in data to be serialized.

20.4.4.6 DSPI Connections to eTPU_A, eTPU_B, EMIOS and SIU

The four (MPC5554) or three (MPC5553) DSPI blocks connect to the input and output channels of the eTPUs and the EMIOS. The MPC5554 connects to eTPU_A, eTPU_B, EMIOS, and SIU. The MPC5553 connects to eTPU_A, EMIOS, and SIU. Some of the DSPI outputs connect to the external interrupt input multiplexing subblock in the SIU. See Section 6.4.3, "External Interrupt," and Section 6.3.1.17, "DSPI Input Select Register (SIU_DISR)," for details on how the DSPI deserialized outputs can be used to trigger external interrupt requests.

20.4.4.6.1 DSPI_A Connectivity (MPC5554 Only)

The DSPI_A connects to the eTPU_B as shown in Figure 20-21. The DSPI_A is provided only in the MPC5554.

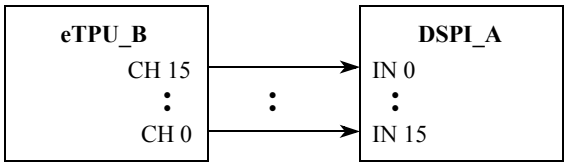


Figure 20-21. DSPI_A Connectivity

Table 20-19 lists the DSPI_A connections.

Table 20-19. DSPI_A Connectivity Table

DSPI_A IN[*n*]	Connected to:
0	eTPU_B Output Channel 15
1	eTPU_B Output Channel 14
2	eTPU_B Output Channel 13
3	eTPU_B Output Channel 12
4	eTPU_B Output Channel 11
5	eTPU_B Output Channel 10
6	eTPU_B Output Channel 9
7	eTPU_B Output Channel 8
8	eTPU_B Output Channel 7
9	eTPU_B Output Channel 6
10	eTPU_B Output Channel 5
11	eTPU_B Output Channel 4
12	eTPU_B Output Channel 3
13	eTPU_B Output Channel 2

DSPI_A OUT[*n*]	Connected to:
0	N/C[1]
1	N/C
2	N/C
3	N/C
4	N/C
5	N/C
6	N/C
7	N/C
8	N/C
9	N/C
10	N/C
11	N/C
12	N/C
13	N/C

Table 20-19. DSPI_A Connectivity Table (continued)

DSPI_A IN[*n*]	Connected to:
14	eTPU_B Output Channel 1
15	eTPU_B Output Channel 0

DSPI_A OUT[*n*]	Connected to:
14	N/C
15	N/C

[1]Not Connected

20.4.4.6.2 DSPI_B Connectivity (MPC5554 and MPC5553)

The DSPI_B connects to the EMIOS, eTPU_A, and SIU as shown in Figure 20-22.

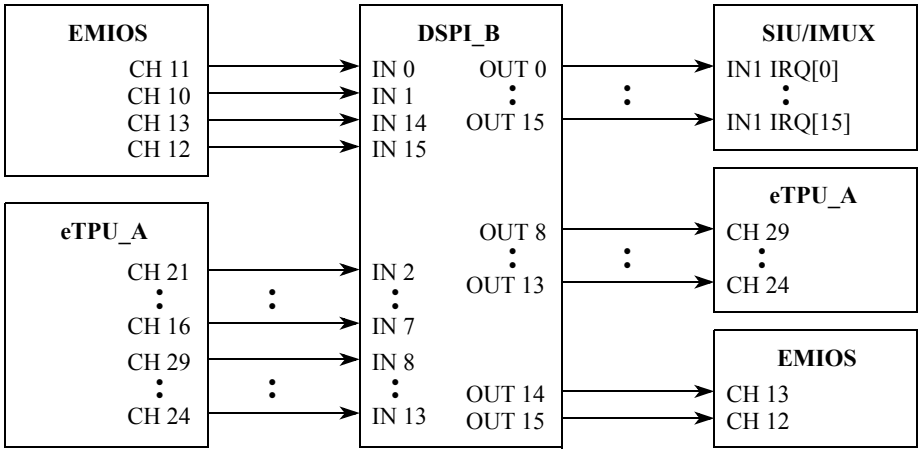


Figure 20-22. DSPI_B Connectivity

Table 20-20 lists the DSPI_B connections.

Table 20-20. DSPI_B Connectivity Table

DSPI_B IN[*n*]	Connected to:
0	EMIOS Output Channel 11
1	EMIOS Output Channel 10
2	eTPU_A Output Channel 21
3	eTPU_A Output Channel 20
4	eTPU_A Output Channel 19
5	eTPU_A Output Channel 18
6	eTPU_A Output Channel 17
7	eTPU_A Output Channel 16
8	eTPU_A Output Channel 29
9	eTPU_A Output Channel 28
10	eTPU_A Output Channel 27

DSPI_B OUT[*n*]	Connected to:
0	Input 1 on IMUX for External IRQ[0]
1	Input 1 on IMUX for External IRQ[1]
2	Input 1 on IMUX for External IRQ[2]
3	Input 1 on IMUX for External IRQ[3]
4	Input 1 on IMUX for External IRQ[4]
5	Input 1 on IMUX for External IRQ[5]
6	Input 1 on IMUX for External IRQ[6]
7	Input 1 on IMUX for External IRQ[7]
8	eTPU_A Input Channel 29, Input 1 on IMUX for External IRQ[8]
9	eTPU_A Input Channel 28, Input 1 on IMUX for External IRQ[9]
10	eTPU_A Input Channel 27, Input 1 on IMUX for External IRQ[10]

Table 20-20. DSPI_B Connectivity Table (continued)

DSPI_B IN[*n*]	Connected to:
11	eTPU_A Output Channel 26
12	eTPU_A Output Channel 25
13	eTPU_A Output Channel 24
14	EMIOS Output Channel 13
15	EMIOS Output Channel 12

DSPI_B OUT[*n*]	Connected to:
11	eTPU_A Input Channel 26, Input 1 on IMUX for External IRQ[11]
12	eTPU_A Input Channel 25, Input 1 on IMUX for External IRQ[12]
13	eTPU_A Input Channel 24, Input 1 on IMUX for External IRQ[13]
14	EMIOS Input Channel 13, Input 1 on IMUX for External IRQ[14]
15	EMIOS Input Channel 12, Input 1 on IMUX for External IRQ[15]

20.4.4.6.3 DSPI_C Connectivity (MPC5554 and MPC5553)

The DSPI_C connects to eTPU_A and SIU as shown in Figure 20-23.

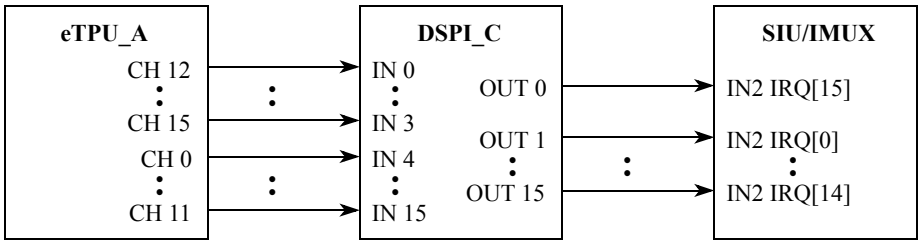


Figure 20-23. DSPI_C Connectivity

Table 20-21 lists the DSPI_C connections.

Table 20-21. DSPI_C Connectivity Table

DSPI_C IN[*n*]	Connected to:
0	eTPU_A Output Channel 12
1	eTPU_A Output Channel 13
2	eTPU_A Output Channel 14
3	eTPU_A Output Channel 15
4	eTPU_A Output Channel 0
5	eTPU_A Output Channel 1
6	eTPU_A Output Channel 2
7	eTPU_A Output Channel 3
8	eTPU_A Output Channel 4
9	eTPU_A Output Channel 5
10	eTPU_A Output Channel 6
11	eTPU_A Output Channel 7

DSPI_C OUT[*n*]	Connected to:
0	Input 2 on IMUX for External IRQ[15]
1	Input 2 on IMUX for External IRQ[0]
2	Input 2 on IMUX for External IRQ[1]
3	Input 2 on IMUX for External IRQ[2]
4	Input 2 on IMUX for External IRQ[3]
5	Input 2 on IMUX for External IRQ[4]
6	Input 2 on IMUX for External IRQ[5]
7	Input 2 on IMUX for External IRQ[6]
8	Input 2 on IMUX for External IRQ[7]
9	Input 2 on IMUX for External IRQ[8]
10	Input 2 on IMUX for External IRQ[9]
11	Input 2 on IMUX for External IRQ[10]

Table 20-21. DSPI_C Connectivity Table (continued)

DSPI_C IN[*n*]	Connected to:
12	eTPU_A Output Channel 8
13	eTPU_A Output Channel 9
14	eTPU_A Output Channel 10
15	eTPU_A Output Channel 11

DSPI_C OUT[*n*]	Connected to:
12	Input 2 on IMUX for External IRQ[11]
13	Input 2 on IMUX for External IRQ[12]
14	Input 2 on IMUX for External IRQ[13]
15	Input 2 on IMUX for External IRQ[14]

20.4.4.6.4 DSPI_D Connectivity (MPC5554 and MPC5553)

The DSPI_D connects to the eTPU_A, EMIOS and SIU as shown in Figure 20-24.

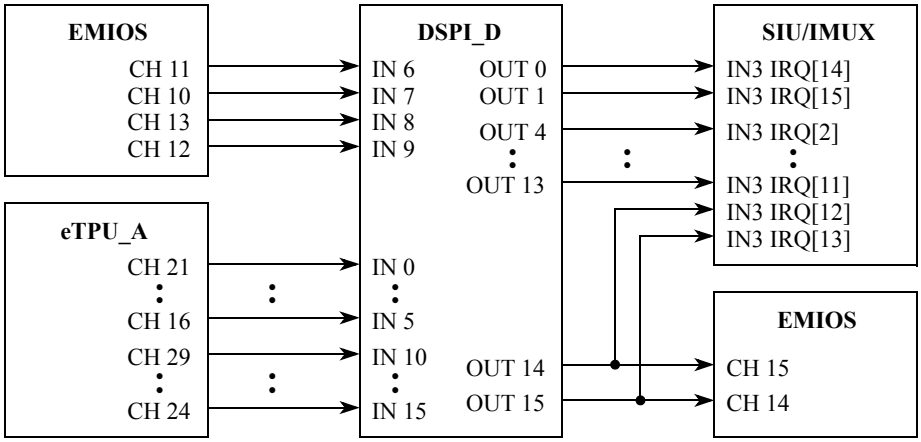


Figure 20-24. DSPI_D Connectivity

Table 20-22 lists the DSPI_D connections.

Table 20-22. DSPI_D Connectivity Table

DSPI_D IN[*n*]	Connected to:
0	eTPU_A Output Channel 21
1	eTPU_A Output Channel 20
2	eTPU_A Output Channel 19
3	eTPU_A Output Channel 18
4	eTPU_A Output Channel 17
5	eTPU_A Output Channel 16
6	EMIOS Output Channel 11
7	EMIOS Output Channel 10
8	EMIOS Output Channel 13
9	EMIOS Output Channel 12
10	eTPU_A Output Channel 29
11	eTPU_A Output Channel 28

DSPI_D OUT[*n*]	Connected to:
0	Input 3 on IMUX for External IRQ[14]
1	Input 3 on IMUX for External IRQ15]
2	N/C
3	N/C
4	Input 3 on IMUX for External IRQ[2]
5	Input 3 on IMUX for External IRQ[3]
6	Input 3 on IMUX for External IRQ[4]
7	Input 3 on IMUX for External IRQ[5]
8	Input 3 on IMUX for External IRQ[6]
9	Input 3 on IMUX for External IRQ[7]
10	Input 3 on IMUX for External IRQ[8]
11	Input 3 on IMUX for External IRQ[9]

Table 20-22. DSPI_D Connectivity Table (continued)

DSPI_D IN[*n*]	Connected to:
12	eTPU_A Output Channel 27
13	eTPU_A Output Channel 26
14	eTPU_A Output Channel 25
15	eTPU_A Output Channel 24

DSPI_D OUT[*n*]	Connected to:
12	Input 3 on IMUX for External IRQ[10]
13	Input 3 on IMUX for External IRQ[11]
14	EMIOS Input Channel 15, Input 3 on IMUX for External IRQ[12]
15	EMIOS Input Channel 14, Input 3 on IMUX for External IRQ[13]

20.4.4.7 Multiple Transfer Operation (MTO)

In DSI configuration the MTO feature allows for multiple DSPIs within the MPC5553/MPC5554 to be chained together in a parallel or serial configuration. The parallel chaining allows multiple DSPIs internal to the MPC5553/MPC5554 and multiple SPI devices external to the MPC5553/MPC5554 to share SCK and PCS signals thereby saving pins. The serial chaining allows bits from multiple DSPIs to be concatenated into a single DSI frame. MTO is enabled by setting the MTOE bit in the DSPI*x*_DSICR.

In parallel and serial chaining there is one bus master and multiple bus slaves. The bus master initiates and controls the transfers, but the DSPI slaves generate trigger signals for the bus DSPI master when an internal condition in the slave warrants a transfer. The DSPI slaves also propagate triggers from other slaves to the master. When a DSPI slave detects a trigger signal on its *ht* input, the slave generates a trigger signal on the MTRIG output.

The SIU_DISR must be configured to use serial or parallel chaining.

20.4.4.7.1 Internal Muxing/SIU Support for Serial and Parallel Chaining

To support MTO, each DSPI in MPC5553/MPC5554 has multiplexers on the SIN, SS, SCK, and *ht* inputs. The internal multiplexers are controlled by registers in the SIU block.

Figure 20-25 (MPC5553) shows DSPI_B and four of the multiplexers in the IMUX subblock of the SIU. The SOUT, MTRIG, SCK and PCS0 outputs from the other two DSPIs connect to the multiplexers on the DSPI_B inputs. DSPI_C and DSPI_D have similar multiplexers on their inputs.

Figure 20-26 (MPC5554) shows DSPI_A and four of the multiplexers in the IMUX subblock of the SIU. The SOUT, MTRIG, SCK and PCS0 outputs from the other three DSPIs connect to the multiplexers on the DSPI_A inputs. DSPI_B, DSPI_C and DSPI_D have similar multiplexers on their inputs.

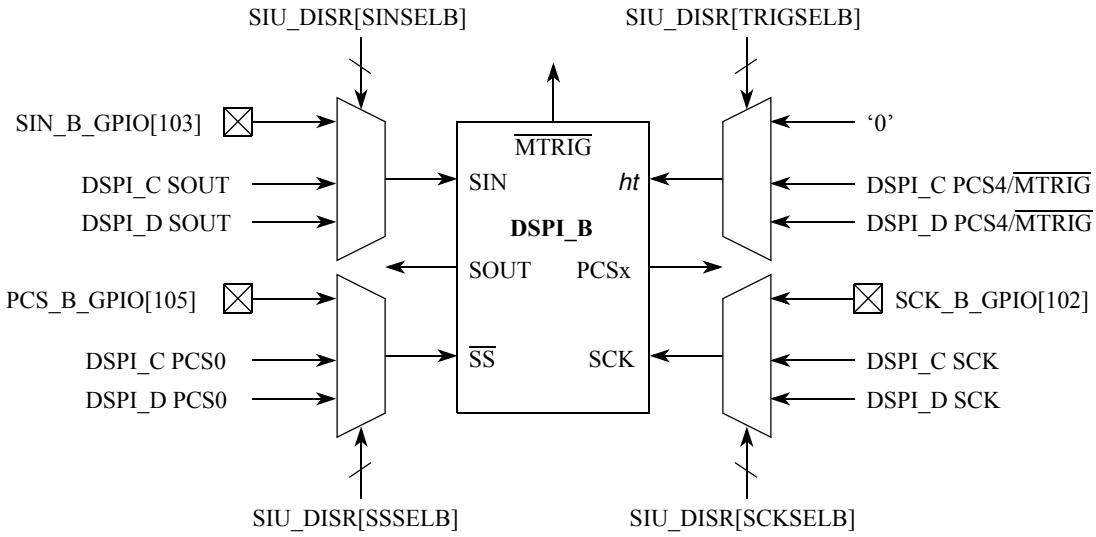


Figure 20-25. DSPI Input Select Muxes — MPC5553

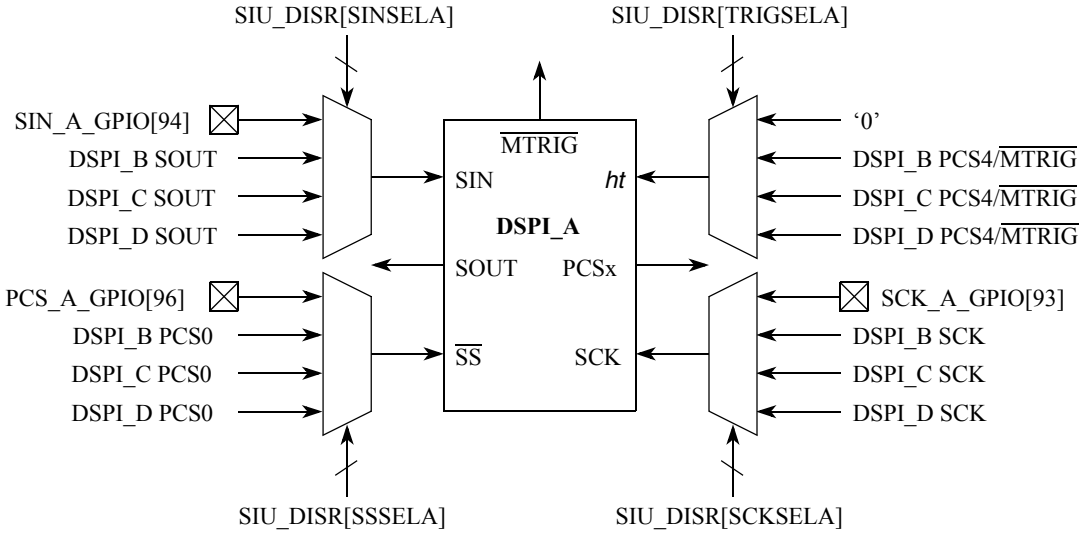


Figure 20-26. DSPI Input Select Muxes — MPC5554

The source for the SIN input of a DSPI can be a pin or the SOUT of any of the other three (for the MPC5554) DSPIS or two (for the MPC5553) DSPIs. The source for the SS input of a DSPI can be a pin or the PCS0 signal from any of the other DSPIs. The source for the SCK input of a DSPI can be a pin or the SCK output of any of the other DSPIs. The source for the hardware trigger (*ht*) input can be the MTRIG signal from any of the other DSPIs. The DSPI input select register (SIU_DSR) selects the source for each DSPI SIN, SS, SCK, and *ht* signal individually.

20.4.4.7.2 Parallel Chaining

Parallel chaining allows the PCS and SCK signals from a master DSPI to be shared by internal slave DSPIs and external slave SPI devices. Signal sharing reduces DSPI pin utilization. An example of a parallel chain is shown in Figure 20-27 (for the MPC5554) or Figure 20-28 (for the MPC5553).

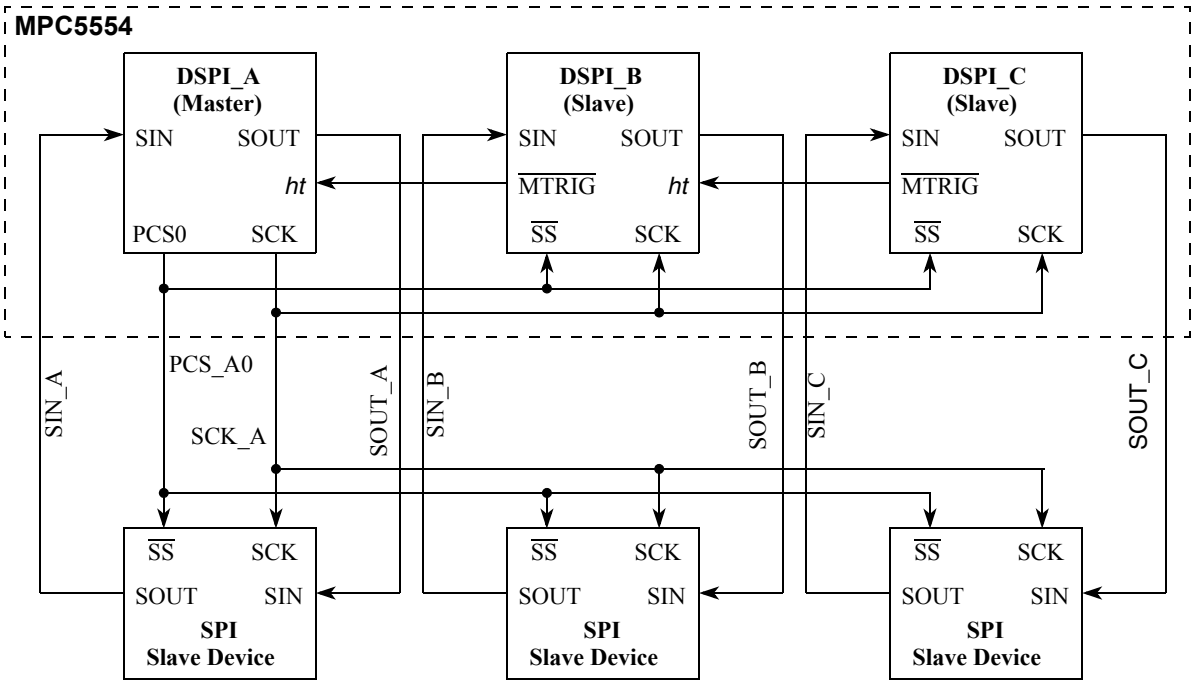


Figure 20-27. Example of Parallel Chaining of DSPIs in the MPC5554

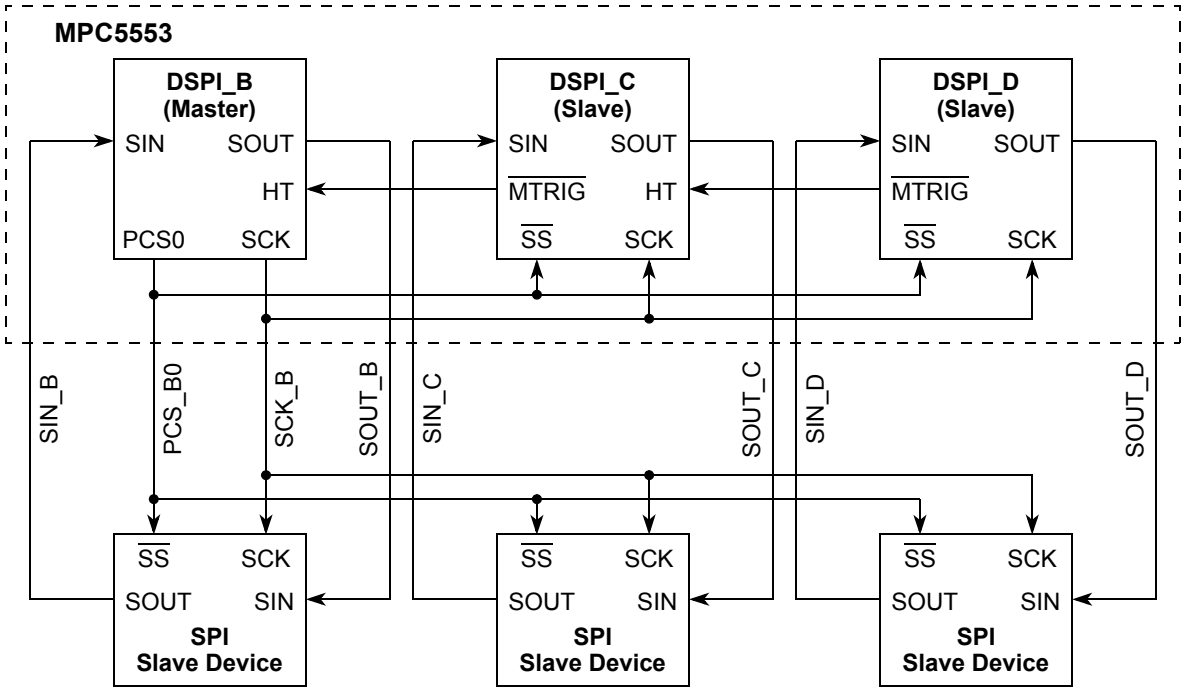


Figure 20-28. Example of Parallel Chaining of DSPIs in the MPC5553

In the parallel chaining example, the SOUT and SIN of the three DSPIs connect to separate external SPI devices. All internal and external SPI modules share PCS and SCK signals. In the MPC5554, DSPI_A controls and initiates all transfers, but the DSPI slaves each have a trigger output signal $\overline{\text{MTRIG}}$ that indicates to DSPI_A that a trigger condition has occurred in the DSPI slaves. In the MPC5553, it is DSPI_B that controls and initiates all transfers, but the DSPI slaves each have a trigger output signal $\overline{\text{MTRIG}}$ that indicates to DSPI_B that a trigger condition has occurred in the DSPI slaves.

When the slave DSPI has a change in data to be serialized, it asserts the $\overline{\text{MTRIG}}$ signal that propagates to DSPI_A (MPC5554)/DSPI_B (MPC5553) which initiates the transfer. In the MPC5554, DSPI_B propagates trigger signals from DSPI_C to DSPI_A. In the MPC5554, DSPI_C propagates trigger signals from DSPI_D to DSPI_B.

The MTOCNT field in the DSPI*x*_DSICR must be written with the number of bits to be transferred. In parallel chaining the number written to MTOCNT must match the FMSZ field in the selected DSPI*x*_CTAR.

20.4.4.7.3 Serial Chaining

Serial chaining allows transfers of DSI frames consisting of concatenated bits from multiple DSPIs. The concatenated frames can be from 8 to 64 bits long. Figures 20-29 and 20-30 show an example of how the modules can be connected in the MPC5554/MPC5553, respectively.

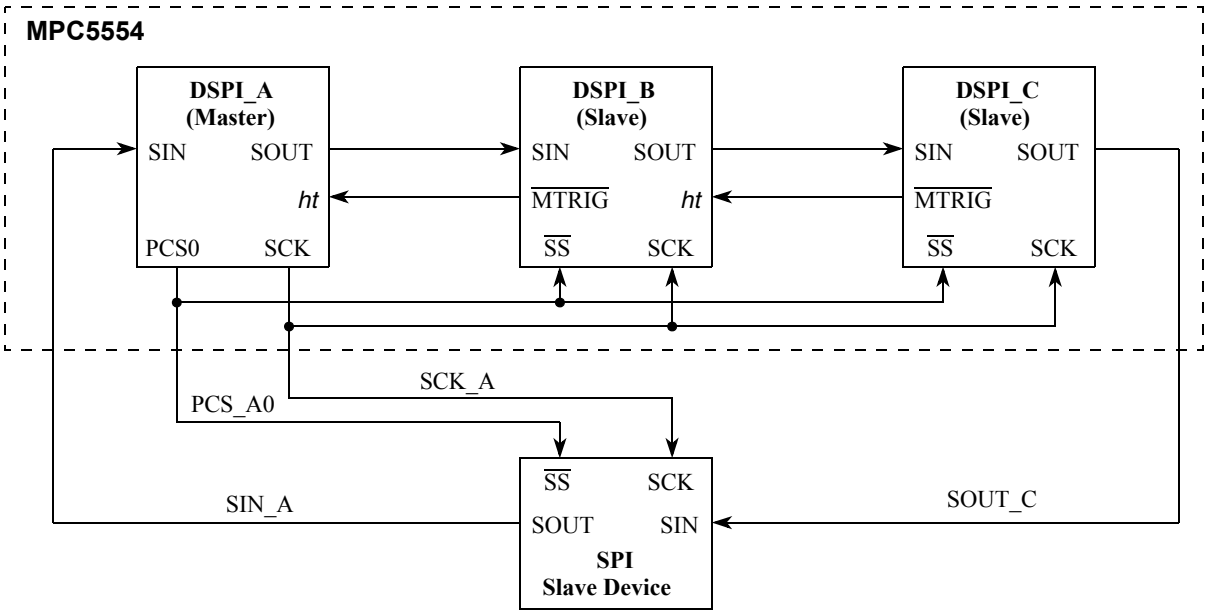


Figure 20-29. Example of Serial Chaining of DSPIs in the MPC5554

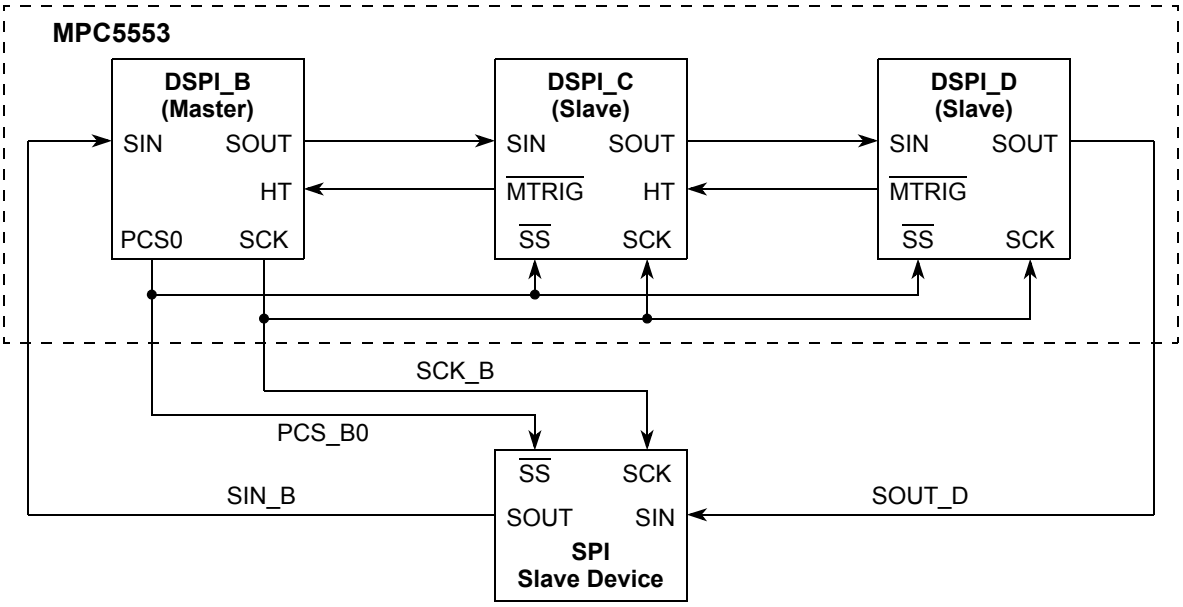


Figure 20-30. Example of Serial Chaining of DSPIs in the MPC5553

In the MPC5554 (master), the SOUT of DSPI_A is connected to the SIN of DSPI_B (slave), and in the MPC5553 (master), the SOUT of DSPI_B is connected to the SIN of DSPI_C (slave). In the MPC5554, the SOUT of the DSPI_B (slave) is connected to the SIN input of the DSPI_C and so on (slave). In the MPC5553, the SOUT of the DSPI_C (slave) is connected to the SIN input of the DSPI_D and so on (slave). The SOUT of the last on-chip DSPI slave is connected to the SIN of the external SPI slave. The SOUT of the external SPI slave is connected to the SIN of DSPI_A master (MPC5554)/DSPI_B master (MPC5553).

The MPC5554 DSPI_A master and the MPC5553 DSPI_B master control and initiate all transfers, but the slave DSPIs use the trigger output signal MTRIG to indicate to the DSPI_A (MPC5554) or DSPI_B

(MPC5553) master that a trigger condition has occurred. When an on-chip DSPI slave has a change in data to be serialized it can assert the MTRIG signal to the DSPI master which initiates the transfer. When a DSPI slave has its *ht* signal asserted it will assert its MTRIG signal thereby propagating trigger signals from other DSPI slaves to the DSPI master.

The MTOCNT field in the DSPI*x*_DSICR must be written with the total number of bits to be transferred. The MTOCNT field must equal the sum of all FMSZ fields in the selected DSPI*x*_CTARs for the DSPI master and all DSPI slaves. For example if one 16-bit DSI frame is created by concatenating 8 bits from the DSPI master, and 4 bits from each of the DSPI slaves in Figure 20-29, the DSPI master's frame size must be set to eight in the FMSZ field, and the DSPI slaves' frame size must be set to four. The largest DSI frame supported by the MTOCNT field is 64 bits (MPC5554) or 48 bits (MPC5553). Any number of DSPIs can be connected together to concatenate DSI frames, as long as each DSPI transfers a minimum of 4 bits and a maximum of 16 bits and the total size of the concatenated frame is less than or equal to 64 bits long (MPC5554) or 48 bits (MPC5553).

20.4.5 Combined Serial Interface (CSI) Configuration

In master mode, the CSI configuration of the DSPI is used to support SPI and DSI functions on a frame by frame basis. CSI configuration allows interleaving of DSI data frames from the parallel input signals (from the eTPU or eMIOS) with SPI commands and data from the TX FIFO. The data returned from the bus slave is either used to drive the parallel output signals (to the eTPU or eMIOS) or is stored in the RX FIFO. CSI configuration allows serialized data and configuration or diagnostic data to be transferred to a slave device using only one serial link. The DSPI is in CSI configuration when the DCONF field in the DSPI*x*_MCR is 0b10. Figure 20-31 shows an example of how a DSPI can be used with a deserializing peripheral that supports SPI control for control and diagnostic frames.

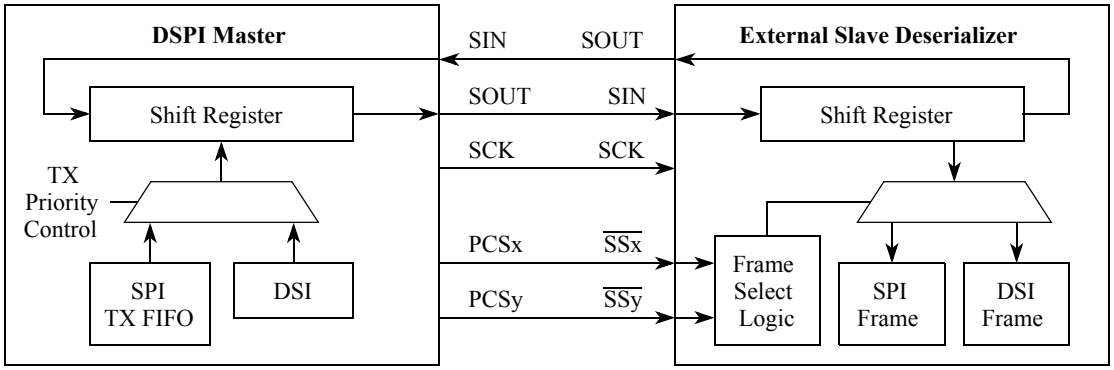


Figure 20-31. Example of System using DSPI in CSI Configuration

In CSI configuration the DSPI transfers DSI data based on Section 20.4.4.5, "DSI Transfer Initiation Control." When there are SPI commands in the TX FIFO, the SPI data has priority over the DSI frames. When the TX FIFO is empty, DSI transfer resumes.

Two peripheral chip select signals indicate whether DSI data or SPI data is transmitted. The user must configure the DSPI so that the two CTARs associated with DSI data and SPI data assert different peripheral chip select signals denoted in the figure as PCSx and PCSy. The CSI configuration is only supported in master mode.

Data returned from the external slave while a DSI frame is transferred is placed on the parallel output signals. Data returned from the external slave while an SPI frame is transferred is moved to the RX FIFO. The TX FIFO and RX FIFO are fully functional in CSI mode.

20.4.5.1 CSI Serialization

Serialization in the CSI configuration is similar to serialization in DSI configuration. The transfer attributes for SPI frames are determined by the DSPI*x*_CTAR selected by the CTAS field in the SPI command halfword. The transfer attributes for the DSI frames are determined by the DSPI*x*_CTAR selected by the DSICTAS field in the DSPI*x*_DSICR. Figure 20-32 shows the CSI serialization logic.

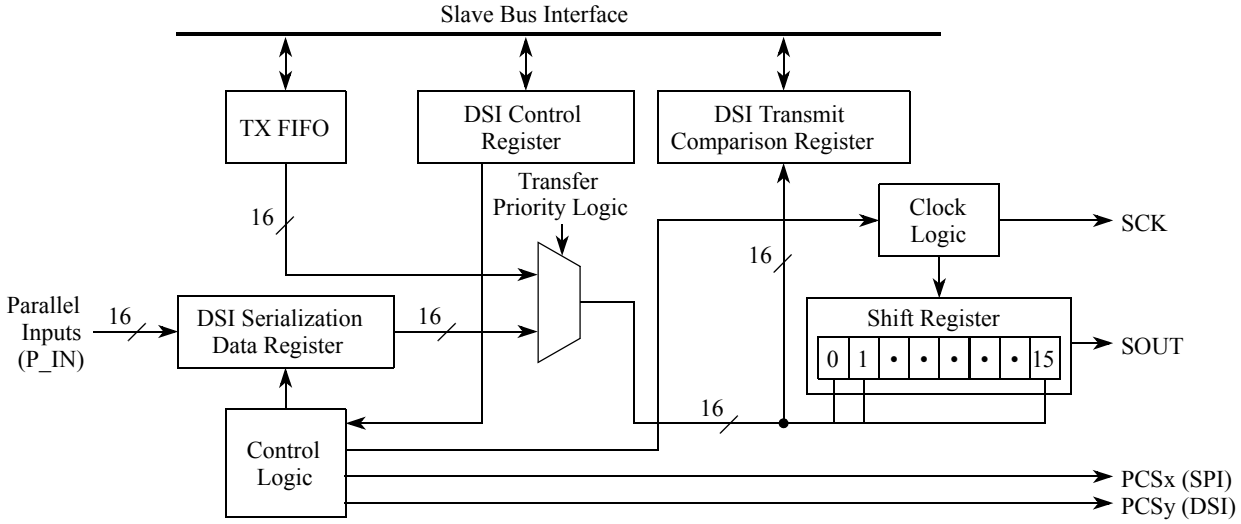


Figure 20-32. CSI Serialization Diagram

The parallel inputs signal states are latched into the DSPI*x*_SDR on the rising edge of every system clock and serialized based on the transfer initiation control settings in the DSPI*x*_DSICR. For more information on the DSPI*x*_SDR, refer to Section 20.3.2.11, "DSPI DSI Serialization Data Register (DSPIx_SDR)." SPI frames written to the TX FIFO have priority over DSI data from the DSPI*x*_SDR and are transferred at the next frame boundary. A copy of the most recently transferred DSI frame is stored in the DSPI*x*_COMPR. The transfer priority logic selects the source of the serialized data and asserts the appropriate $\overline{CS}$ signal.

20.4.5.2 CSI Deserialization

The deserialized frames in CSI configuration go into the DSPI*x*_SDR or the RX FIFO based on the transfer priority logic. When DSI frames are transferred the returned frames are deserialized and latched into the DSPI*x*_DDR. When SPI frames are transferred the returned frames are deserialized and written to the RX FIFO. Figure 20-33 shows the CSI deserialization logic.

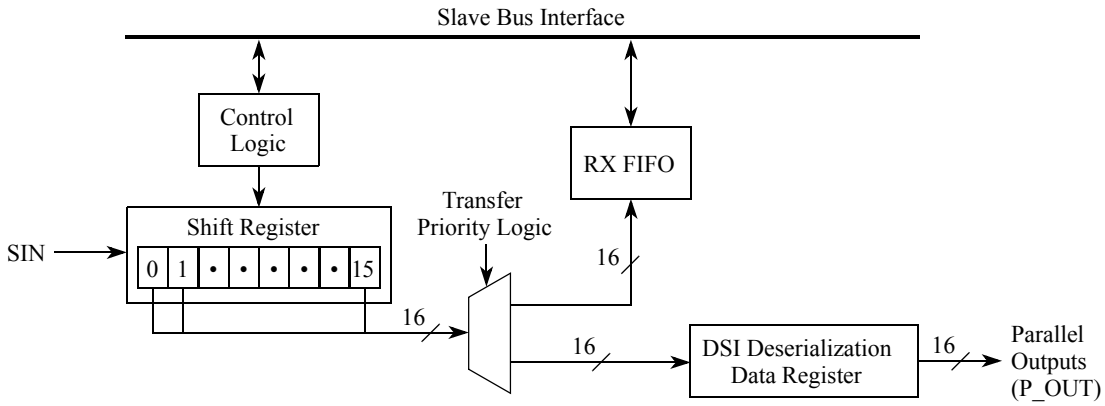


Figure 20-33. CSI Deserialization Diagram

20.4.6 DSPI Baud Rate and Clock Delay Generation

The SCK frequency and the delay values for serial transfer are generated by dividing the system clock frequency by a prescaler and a scaler with the option of doubling the baud rate. Figure 20-34 shows conceptually how the SCK signal is generated.

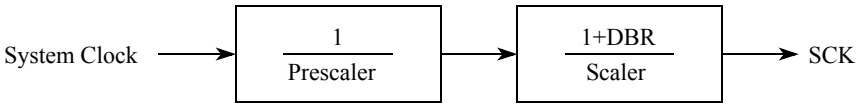


Figure 20-34. Communications Clock Prescalers and Scalers

20.4.6.1 Baud Rate Generator

The baud rate is the frequency of the serial communication clock (SCK). The system clock is divided by a baud rate prescaler (defined by DSPI*x*_CTAR[PBR]) and baud rate scaler (defined by DSPI*x*_CTAR[BR]) to produce SCK with the possibility of doubling the baud rate. The DBR, PBR, and BR fields in the DSPI*x*_CTARs select the frequency of SCK using the following formula:

$$\text{SCK baud rate} = \frac{f_{SYS}}{\text{PBRPrescalerValue}} \times \frac{1 + \text{DBR}}{\text{BRScalerValue}}$$

Table 20-23 shows an example of a computed baud rate.

Table 20-23. Baud Rate Computation Example

f_{SYS}	PBR	Prescaler Value	BR	Scaler Value	DBR Value	Baud Rate
100 MHz	0b00	2	0b0000	2	0	25 Mb/s
20 MHz	0b00	2	0b0000	2	1	10 Mb/s

20.4.6.2 PCS to SCK Delay (t_{CSC})

The PCS to SCK delay is the length of time from assertion of the PCS signal to the first SCK edge. See Figure 20-36 for an illustration of the PCS to SCK delay. The PCSSCK and CSSCK fields in the

DSPI*x*_CTAR*n* registers select the PCS to SCK delay, and the relationship is expressed by the following formula:

$$t_{CSC} = \frac{1}{f_{SYS}} \times PCSSCK \times CSSCK$$

Table 20-24 shows an example of the computed PCS to SCK delay.

Table 20-24. PCS to SCK Delay Computation Example

PCSSCK	Prescaler Value	CSSCK	Scaler Value	f_{SYS}	PCS to SCK Delay
0b01	3	0b0100	32	100 MHz	0.96 µs

20.4.6.3 After SCK Delay (t_{ASC})

The after SCK delay is the length of time between the last edge of SCK and the negation of PCS. See Figure 20-36 and Figure 20-37 for illustrations of the after SCK delay. The PASC and ASC fields in the DSPI*x*_CTAR*n* registers select the after SCK delay. The relationship between these variables is given in the following formula:

$$t_{ASC} = \frac{1}{f_{SYS}} \times PASC \times ASC$$

Table 20-25 shows an example of the computed after SCK delay.

Table 20-25. After SCK Delay Computation Example

PASC	Prescaler Value	ASC	Scaler Value	Fsys	After SCK Delay
0b01	3	0b0100	32	100 MHz	0.96 us

20.4.6.4 Delay after Transfer (t_{DT})

The delay after transfer is the length of time between negation of the PCS signal for a frame and the assertion of the PCS signal for the next frame. See Figure 20-36 for an illustration of the delay after transfer. The PDT and DT fields in the DSPI*x*_CTAR*n* registers select the delay after transfer. The following formula expresses the PDT/DT/delay after transfer relationship:

$$t_{DT} = \frac{1}{f_{SYS}} \times PDT \times DT$$

Table 20-26 shows an example of the computed delay after transfer.

Table 20-26. Delay after Transfer Computation Example

PDT	Prescaler Value	DT	Scaler Value	f_{SYS}	Delay after Transfer
0b01	3	0b1110	32768	100 MHz	0.98 ms

20.4.6.5 Peripheral Chip Select Strobe Enable ($\overline{\text{PCSS}}$)

The $\overline{\text{PCSS}}$ signal provides a delay to allow the PCS signals to settle after transitioning thereby avoiding glitches. When the DSPI is in master mode and PCSSE bit is set in the DSPI*x*_MCR, $\overline{\text{PCSS}}$ provides a signal for an external demultiplexer to decode the PCS[0:4] signals into as many as 32 glitch-free PCS signals. Figure 20-35 shows the timing of the $\overline{\text{PCSS}}$ signal relative to PCS signals.

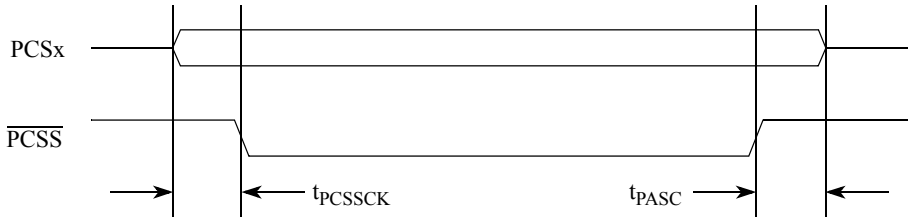


Figure 20-35. Peripheral Chip Select Strobe Timing

The delay between the assertion of the PCS signals and the assertion of $\overline{\text{PCSS}}$ is selected by the PCSSCK field in the DSPI*x*_CTAR based on the following formula:

$$tPCSSCK = \frac{1}{f_{SYS}} \times PCSSCK$$

At the end of the transfer the delay between $\overline{\text{PCSS}}$ negation and PCS negation is selected by the PASC field in the DSPI*x*_CTAR based on the following formula:

$$tPASC = \frac{1}{f_{SYS}} \times PASC$$

Table 20-27 shows an example of the computed t_{PCSSCK} delay.

Table 20-27. Peripheral Chip Select Strobe Assert Computation Example

PCSSCK	Prescaler	f_{SYS}	Delay before Transfer
0b11	7	100 MHz	70.0 ns

Table 20-28 shows an example of the computed the t_{PASC} delay

.

Table 20-28. Peripheral Chip Select Strobe Negate Computation Example

PASC	Prescaler	f_{SYS}	Delay after Transfer
0b11	7	100 MHz	70.0 ns

20.4.7 Transfer Formats

The SPI serial communication is controlled by the serial communications clock (SCK) signal and the PCS signals. The SCK signal provided by the master device synchronizes shifting and sampling of the data by the SIN and SOUT pins. The PCS signals serve as enable signals for the slave devices.

When the DSPI is the bus master, the CPOL and CPHA bits in the DSPI clock and transfer attributes registers (DSPI*x*_CTAR*n*) select the polarity and phase of the serial clock, SCK. The polarity bit selects

the idle state of the SCK. The clock phase bit selects if the data on SOUT is valid before or on the first SCK edge.

When the DSPI is the bus slave, CPOL and CPHA bits in the DSPI*x*_CTAR0 (SPI slave mode) or DSPI*x*_CTAR1 (DSI slave mode) select the polarity and phase of the serial clock. Even though the bus slave does not control the SCK signal, clock polarity, clock phase and number of bits to transfer must be identical for the master device and the slave device to ensure proper transmission.

The DSPI supports four different transfer formats:

- Classic SPI with CPHA = 0
- Classic SPI with CPHA = 1
- Modified transfer format with CPHA = 0
- Modified transfer format with CPHA = 1

A modified transfer format is supported to allow for high-speed communication with peripherals that require longer setup times. The DSPI can sample the incoming data later than halfway through the cycle to give the peripheral more setup time. The MTFE bit in the DSPI*x*_MCR selects between classic SPI format and modified transfer format. The classic SPI formats are described in Section 20.4.7.1, "Classic SPI Transfer Format (CPHA = 0)" and Section 20.4.7.2, "Classic SPI Transfer Format (CPHA = 1)." The modified transfer formats are described in Section 20.4.7.3, "Modified SPI/DSI Transfer Format (MTFE = 1, CPHA = 0)" and Section 20.4.7.4, "Modified SPI/DSI Transfer Format (MTFE = 1, CPHA = 1)."

In the SPI and DSI configurations, the DSPI provides the option of keeping the PCS signals asserted between frames. See Section 20.4.7.5, "Continuous Selection Format" for details.

20.4.7.1 Classic SPI Transfer Format (CPHA = 0)

The transfer format shown in Figure 20-36 is used to communicate with peripheral SPI slave devices where the first data bit is available on the first clock edge. In this format, the master and slave sample their SIN pins on the odd-numbered SCK edges and change the data on their SOUT pins on the even-numbered SCK edges.

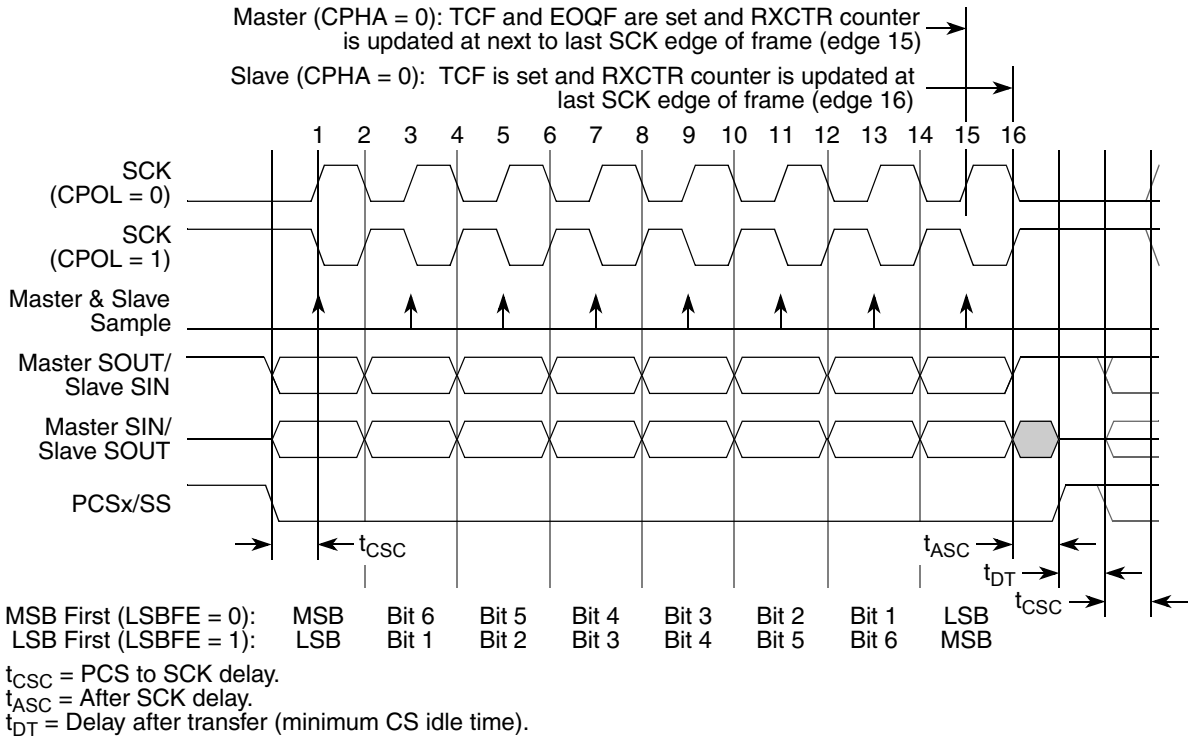


Figure 20-36. DSPI Transfer Timing Diagram (MTFE = 0, CPHA = 0, FMSZ = 8)

The master initiates the transfer by placing its first data bit on the SOUT pin and asserting the appropriate peripheral chip select signals to the slave device. The slave responds by placing its first data bit on its SOUT pin. After the t_{CSC} delay has elapsed, the master outputs the first edge of SCK. This is the edge used by the master and slave devices to sample the first input data bit on their serial data input signals. At the second edge of the SCK the master and slave devices place their second data bit on their serial data output signals. For the rest of the frame the master and the slave sample their SIN pins on the odd-numbered clock edges and changes the data on their SOUT pins on the even-numbered clock edges. After the last clock edge occurs a delay of t_{ASC} is inserted before the master negates the PCS signals. A delay of t_{DT} is inserted before a new frame transfer can be initiated by the master.

For the CPHA =0 conditon of the master, TCF and EOQF are set and the RXCTR counter is updated at the next to last serial clock edge of the frame (edge 15) of Figure 20-36.

For the CPHA=0 condition of the slave, TCF is set and the RXCTR counter is updated at the last serial clock edge of the frame (edge 16) of Figure 20-36.

20.4.7.2 Classic SPI Transfer Format (CPHA = 1)

This transfer format shown in Figure 20-37 is used to communicate with peripheral SPI slave devices that require the first SCK edge before the first data bit becomes available on the slave SOUT pin. In this format the master and slave devices change the data on their SOUT pins on the odd-numbered SCK edges and sample the data on their SIN pins on the even-numbered SCK edges.

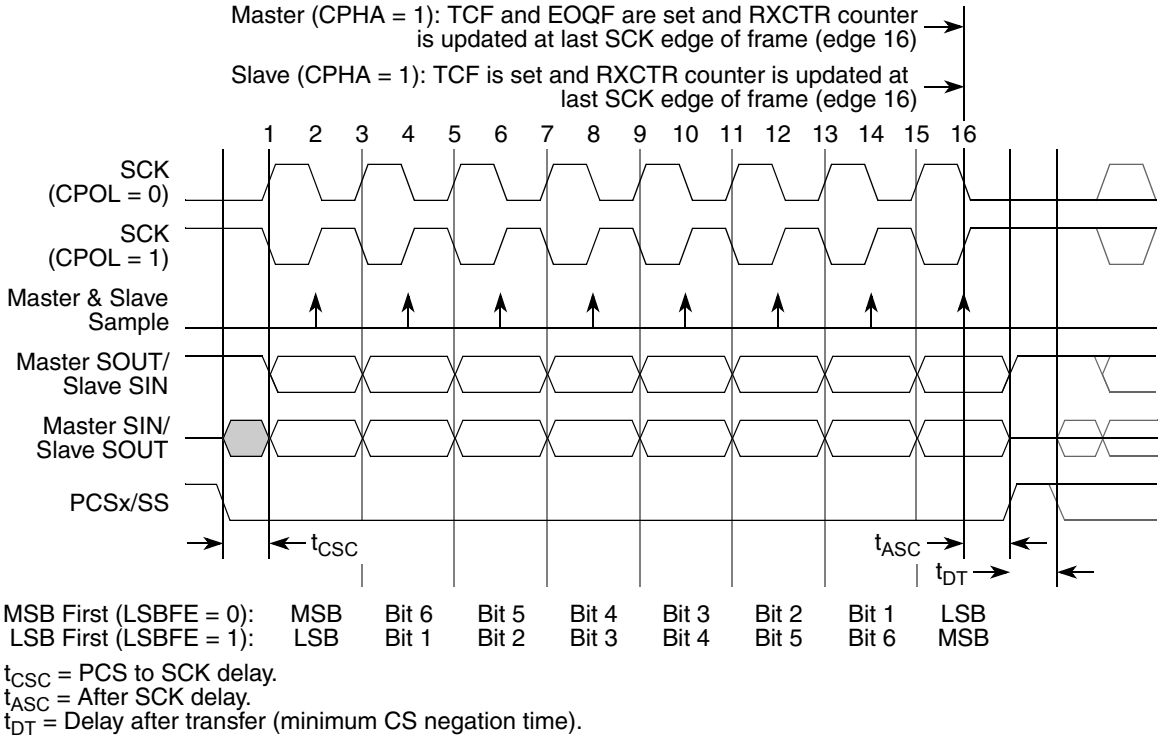


Figure 20-37. DSPI Transfer Timing Diagram (MTFE = 0, CPHA = 1, FMSZ = 8)

The master initiates the transfer by asserting the PCS signal to the slave. After the t_{CSC} delay has elapsed, the master generates the first SCK edge and at the same time places valid data on the master SOUT pin. The slave responds to the first SCK edge by placing its first data bit on its slave SOUT pin.

At the second edge of the SCK the master and slave sample their SIN pins. For the rest of the frame the master and the slave change the data on their SOUT pins on the odd-numbered clock edges and sample their SIN pins on the even-numbered clock edges. After the last clock edge occurs a delay of t_{ASC} is inserted before the master negates the PCS signal. A delay of t_{DT} is inserted before a new frame transfer can be initiated by the master.

For CPHA=1 the master EOQF and TCF and slave TCF are set at the last serial clock edge (edge 16) of Figure 20-37. For CPHA=1 the master and slave RXCTR counters are updated on the same clock edge.

20.4.7.3 Modified SPI/DSI Transfer Format (MTFE = 1, CPHA = 0)

In this modified transfer format both the master and the slave sample later in the SCK period than in classic SPI mode to allow for delays in device pads and board traces. These delays become a more significant fraction of the SCK period as the SCK period decreases with increasing baud rates.

NOTE

For correct operation of the modified transfer format, the user must thoroughly analyze the SPI link timing budget.

The master and the slave place data on the SOUT pins at the assertion of the PCS signal. After the PCS to SCK delay has elapsed the first SCK edge is generated. The slave samples the master SOUT signal on every odd numbered SCK edge. The slave also places new data on the slave SOUT on every odd numbered clock edge.

The master places its second data bit on the SOUT line one system clock after odd numbered SCK edge. The point where the master samples the slave SOUT is selected by writing to the SMPL_PT field in the DSPI*x*_MCR. Table 20-29 lists the number of system clock cycles between the active edge of SCK and the master sample point for different values of the SMPL_PT bit field. The master sample point can be delayed by one or two system clock cycles.

Table 20-29. Delayed Master Sample Point

SMPL_PT	Number of System Clock Cycles between Odd-numbered Edge of SCK and Sampling of SIN
00	0
01	1
10	2
11	Reserved

Figure 20-38 shows the modified transfer format for CPHA = 0. Only the condition where CPOL = 0 is illustrated. The delayed master sample points are indicated with a lighter shaded arrow.

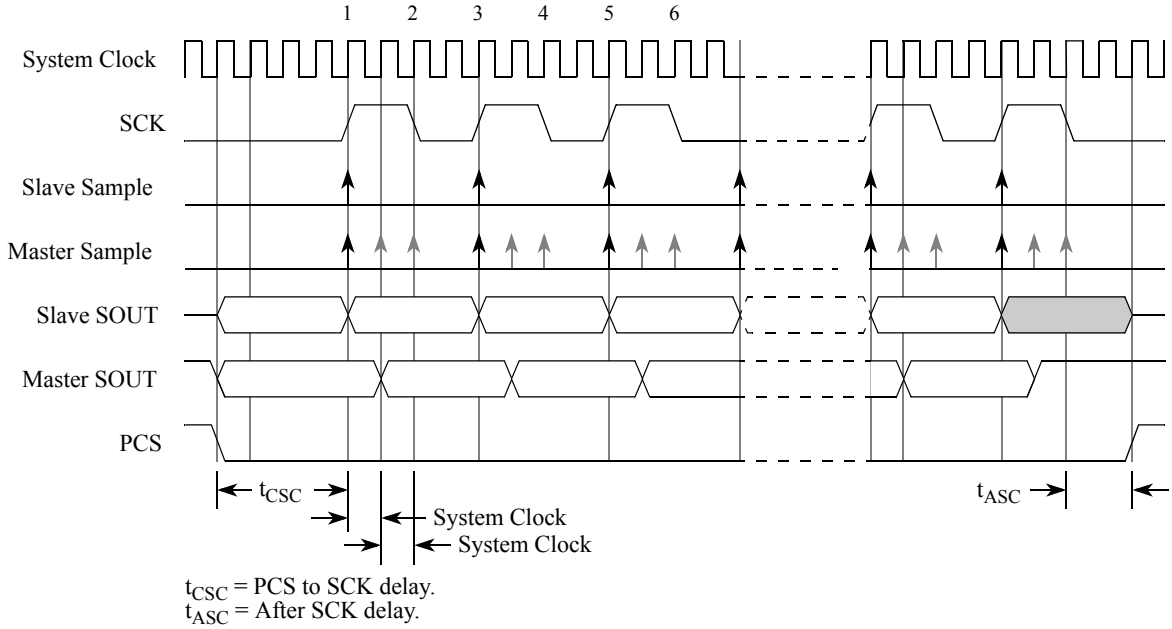


Figure 20-38. DSPI Modified Transfer Format (MTFE = 1, CPHA = 0, Fsck = Fsys/4)

20.4.7.4 Modified SPI/DSI Transfer Format (MTFE = 1, CPHA = 1)

Figure 20-39 shows the modified transfer format for CPHA = 1. Only the condition where CPOL = 0 is described. At the start of a transfer the DSPI asserts the PCS signal to the slave device. After the PCS to SCK delay has elapsed the master and the slave put data on their SOUT pins at the first edge of SCK. The slave samples the master SOUT signal on the even numbered edges of SCK. The master samples the slave SOUT signal on the odd numbered SCK edges starting with the 3rd SCK edge. The slave samples the last bit on the last edge of the SCK. The master samples the last slave SOUT bit one half SCK cycle after the last edge of SCK. No clock edge will be visible on the master SCK pin during the sampling of the last bit. The SCK to PCS delay must be greater or equal to half of the SCK period.

NOTE

For correct operation of the modified transfer format, the user must thoroughly analyze the SPI link timing budget.

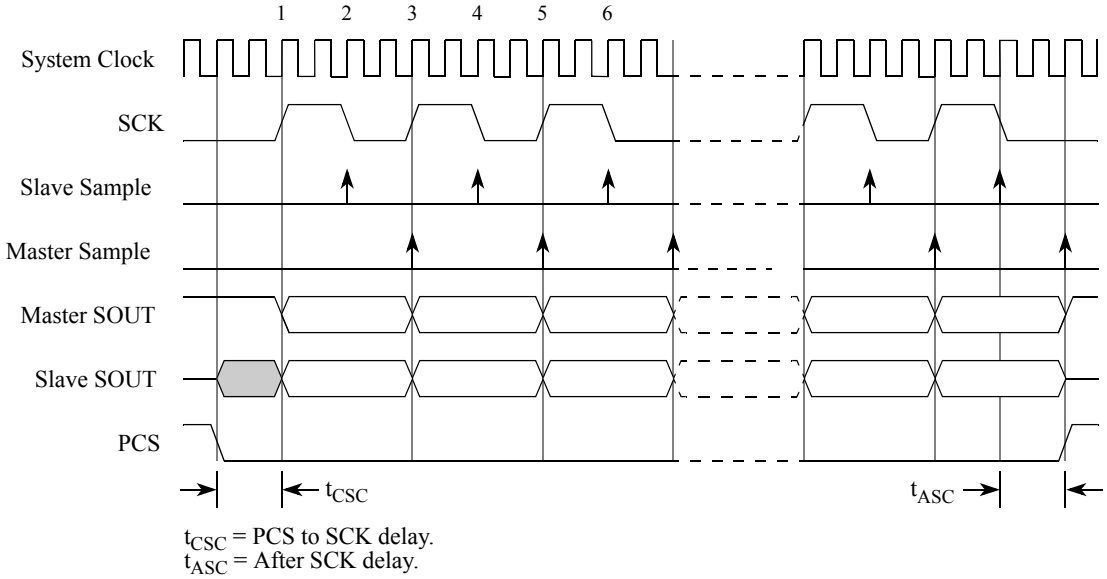


Figure 20-39. DSPI Modified Transfer Format (MTFE = 1, CPHA = 1, Fsck = Fsys/4)

20.4.7.5 Continuous Selection Format

Some peripherals must be deselected between every transfer. Other peripherals must remain selected between several sequential serial transfers. The continuous selection format provides the flexibility to handle both cases. The continuous selection format is enabled for the SPI configuration by setting the CONT bit in the SPI command. Continuous selection is enabled for the DSI configuration by setting the DCONT bit in the DSPI*x*_DSICR. The behavior of the PCS signals in the two configurations is identical so only SPI configuration will be described.

When the CONT bit = 0, the DSPI drives the asserted chip select signals to their idle states in between frames. The idle states of the chip select signals are selected by the PCSIS field in the DSPI*x*_MCR. Figure 20-40 shows the timing diagram for two four-bit transfers with CPHA = 1 and CONT = 0.

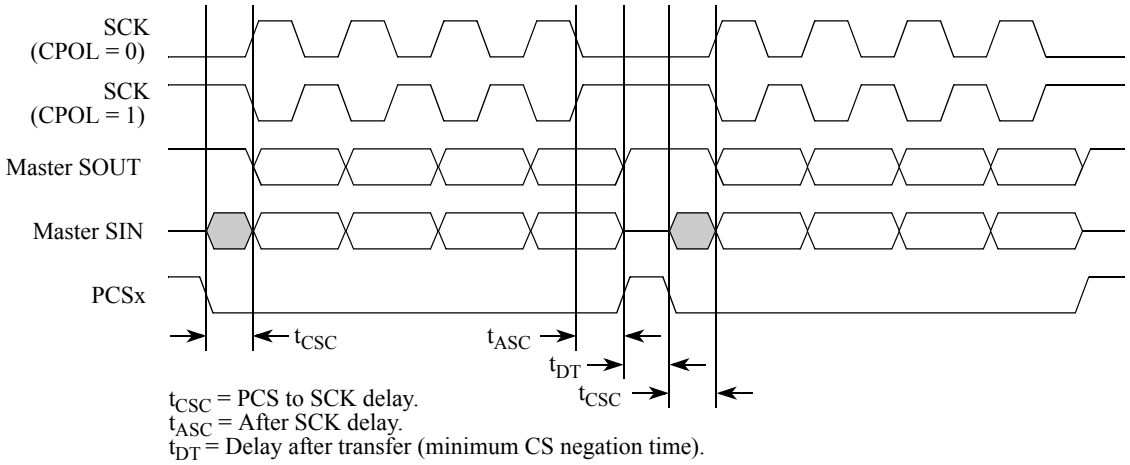


Figure 20-40. Example of Non-Continuous Format (CPHA=1, CONT=0)

When the CONT = 1 and the PCS signal for the next transfer is the same as for the current transfer, the PCS signal remains asserted for the duration of the two transfers. The delay between transfers (t_{DT}) is not inserted between the transfers. Figure 20-41 shows the timing diagram for two four-bit transfers with CPHA = 1 and CONT = 1.

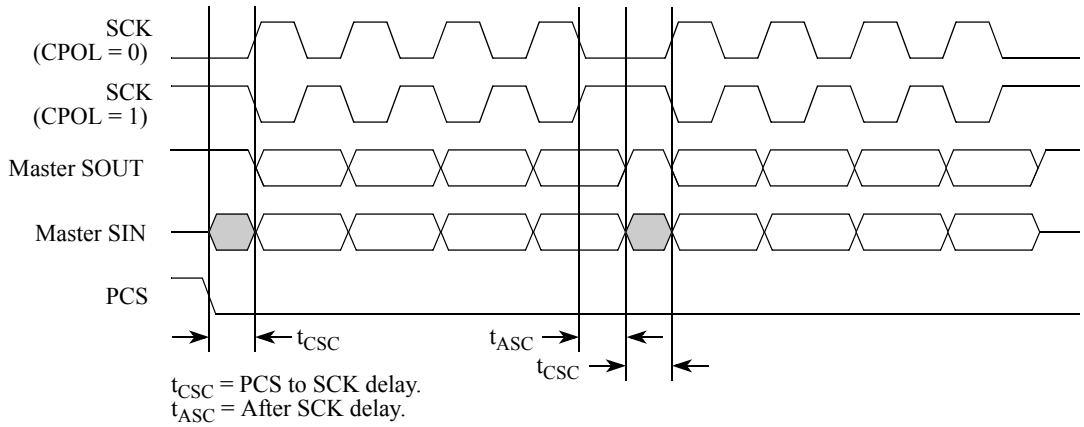


Figure 20-41. Example of Continuous Transfer (CPHA = 1, CONT = 1)

In Figure 20-41, note that the period length at the start of the next transfer is the sum of t_{ASC} and t_{CSC}; i.e., it does not include a half-clock period. The default settings for these provide a total of four system clocks. In many situations, t_{ASC} and t_{CSC} must be increased if a full half-clock period is required.

Switching CTARs between frames while using continuous selection can cause errors in the transfer. The PCS signal must be negated before CTAR is switched.

When the CONT bit = 1 and the PCS signals for the next transfer are different from the present transfer, the PCS signals behave as if the CONT bit was not set.

20.4.7.6 Clock Polarity Switching between DSPI Transfers

If it is desired to switch polarity between non-continuous DSPI frames, the edge generated by the change in the idle state of the clock occurs one system clock before the assertion of the chip select for the next frame. In Figure 20-42, time ‘A’ shows the one clock interval. Time ‘B’ is user programmable from a

minimum of 2 system clocks. Refer to Section 20.3.2.3, “DSPI Clock and Transfer Attributes Registers 0–7 (DSPIx_CTARn).”

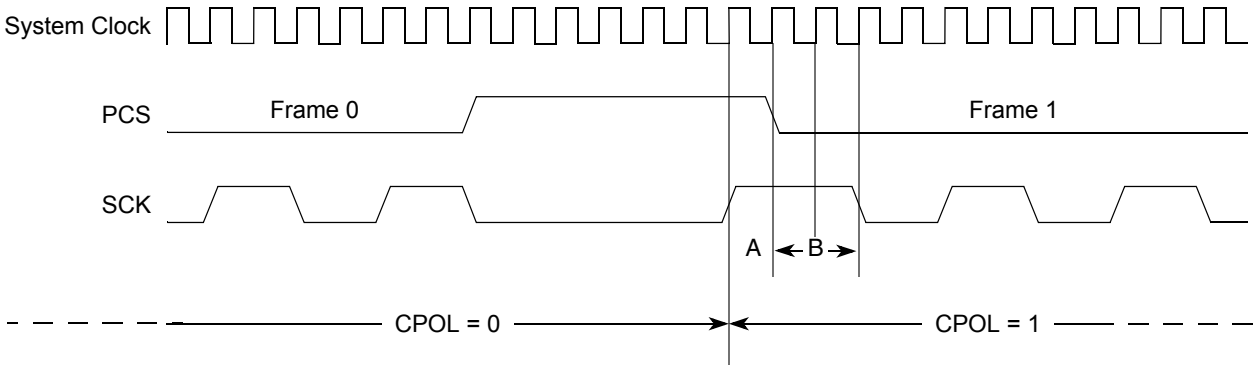


Figure 20-42. Polarity Switching between Frames

20.4.8 Continuous Serial Communications Clock

The DSPI provides the option of generating a continuous SCK signal for slave peripherals that require a continuous clock.

Continuous SCK is enabled by setting the CONT_SCKE bit in the DSPI*x*_MCR. Continuous SCK is valid in all configurations.

Continuous SCK is only supported for CPHA = 1. Setting CPHA = 0 will be ignored if the CONT_SCKE bit is set. Continuous SCK is supported for modified transfer format.

Clock and transfer attributes for the continuous SCK mode are set according to the following rules:

- When the DSPI is in SPI configuration, CTAR0 shall be used initially. At the start of each SPI frame transfer, the CTAR specified by the CTAS for the frame shall be used.
- When the DSPI is in DSI configuration, the CTAR specified by the DSICTAS field shall be used at all times.
- When the DSPI is in CSI configuration, the CTAR selected by the DSICTAS field shall be used initially. At the start of an SPI frame transfer, the CTAR specified by the CTAS value for the frame shall be used. At the start of a DSI frame transfer, the CTAR specified by the DSICTAS field shall be used.
- In all configurations, the currently selected CTAR shall remain in use until the start of a frame with a different CTAR specified, or the continuous SCK mode is terminated.

It is recommended that the baud rate is the same for all transfers made while using the continuous SCK. Switching clock polarity between frames while using continuous SCK can cause errors in the transfer. Continuous SCK operation is not guaranteed if the DSPI is put into module disable mode.

Enabling continuous SCK disables the PCS to SCK delay and the After SCK delay. The delay after transfer is fixed at one SCK cycle. Figure 20-43 shows timing diagram for continuous SCK format with continuous selection disabled.

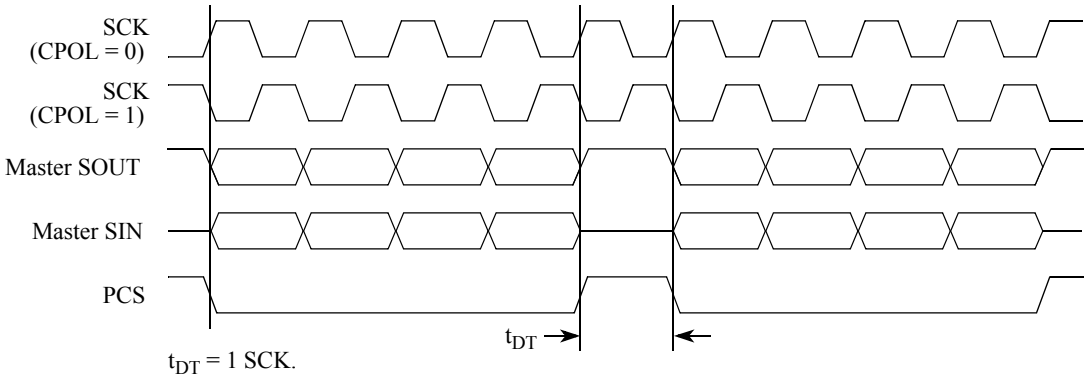


Figure 20-43. Continuous SCK Timing Diagram (CONT=0)

If the CONT bit in the TX FIFO entry is set or the DCONT in the DSPI*x*_DSICR is set, PCS remains asserted between the transfers when the PCS signal for the next transfer is the same as for the current transfer. Figure 20-44 shows timing diagram for continuous SCK format with continuous selection enabled.

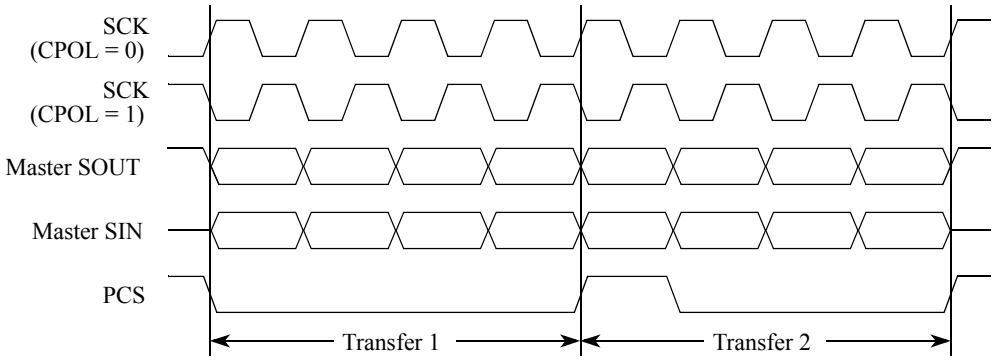


Figure 20-44. Continuous SCK Timing Diagram (CONT=1)

20.4.9 Interrupts/DMA Requests

The DSPI has five conditions which can only generate interrupt requests and two conditions that can generate interrupt or DMA request. Table 20-30 lists the six conditions.

Table 20-30. Interrupt and DMA Request Conditions

Condition	Flag	Interrupt	DMA
End of transfer queue has been reached (EOQ)	EOQF	X	
TX FIFO is not full	TFFF	X	X
Current frame transfer is complete	TCF	X	
TX FIFO underflow has occurred	TFUF	X	
RX FIFO is not empty	RFDF	X	X
RX FIFO overflow has occurred	RFOF	X	
A FIFO overrun has occurred[1]	TFUF OR RFOF	X	

[1] The FIFO Overrun condition is created by OR-ing the TFUF and RFOF flags together.

Each condition has a flag bit and a request enable bit. The flag bits are described in the Section 20.3.2.4, "DSPI Status Register (DSPIx_SR)" and the request enable bits are described in the Section 20.3.2.5, "DSPI DMA/Interrupt Request Select and Enable Register (DSPIx_RSER)." The TX FIFO fill flag (TFFF) and RX FIFO drain flag (RFDF) generate interrupt requests or DMA requests depending on the TFFF_DIRS and RFDF_DIRS bits in the DSPI*x*_RSER.

20.4.9.1 End of Queue Interrupt Request (EOQF)

The end of queue equest indicates that the end of a transmit queue is reached. The end of queue request is generated when the EOQ bit in the executing SPI command is asserted and the EOQF_RE bit in the DSPI*x*_RSER is asserted. See the EOQ bit description in Section 20.3.2.4, "DSPI Status Register (DSPIx_SR)." Refer to Figure 20-36 and Figure 20-37 that illustrate when EOQF is set.

20.4.9.2 Transmit FIFO Fill Interrupt or DMA Request (TFFF)

The transmit FIFO fill request indicates that the TX FIFO is not full. The transmit FIFO fill request is generated when the number of entries in the TX FIFO is less than the maximum number of possible entries, and the TFFF_RE bit in the DSPI*x*_RSER is asserted. The TFFF_DIRS bit in the DSPI*x*_RSER selects whether a DMA request or an interrupt request is generated.

20.4.9.3 Transfer Complete Interrupt Request (TCF)

The transfer complete request indicates the end of the transfer of a serial frame. The transfer complete request is generated at the end of each frame transfer when the TCF_RE bit is set in the DSPI*x*_RSER. See the TCF bit description in Section 20.3.2.4, "DSPI Status Register (DSPIx_SR)." Refer to Figure 20-36 and Figure 20-37 that illustrate when TCF is set.

20.4.9.4 Transmit FIFO Underflow Interrupt Request (TFUF)

The transmit FIFO underflow request indicates that an underflow condition in the TX FIFO has occurred. The transmit underflow condition is detected only for DSPI modules operating in slave mode and SPI configuration. The TFUF bit is set when the TX FIFO of a DSPI operating in slave mode and SPI configuration is empty, and a transfer is initiated from an external SPI master. If the TFUF bit is set while the TFUF_RE bit in the DSPI*x*_RSER is asserted, an interrupt request is generated.

20.4.9.5 Receive FIFO Drain Interrupt or DMA Request (RFDF)

The receive FIFO drain request indicates that the RX FIFO is not empty. The receive FIFO drain request is generated when the number of entries in the RX FIFO is not zero, and the RFDF_RE bit in the DSPI*x*_RSER is asserted. The RFDF_DIRS bit in the DSPI*x*_RSER selects whether a DMA request or an interrupt request is generated.

20.4.9.6 Receive FIFO Overflow Interrupt Request (RFOF)

The receive FIFO overflow request indicates that an overflow condition in the RX FIFO has occurred. A receive FIFO overflow request is generated when RX FIFO and shift register are full and a transfer is initiated. The RFOF_RE bit in the DSPI*x*_RSER must be set for the interrupt request to be generated.

Depending on the state of the ROOE bit in the DSPI*x*_MCR, the data from the transfer that generated the overflow is either ignored or shifted in to the shift register. If the ROOE bit is set, the incoming data is shifted in to the shift register. If the ROOE bit is negated, the incoming data is ignored.

20.4.9.7 FIFO Overrun Request (TFUF) or (RFOF)

The FIFO overrun request indicates that at least one of the FIFOs in the DSPI has exceeded its capacity. The FIFO overrun request is generated by logically OR'ing together the RX FIFO overflow and TX FIFO underflow signals.

20.4.10 Power Saving Features

The DSPI supports two power-saving strategies:

- Module disable mode—clock gating of non-memory mapped logic
- Clock gating of slave interface signals and clock to memory-mapped logic

20.4.10.1 Module Disable Mode

Module disable mode is a module-specific mode that the DSPI can enter to save power. Host software can initiate the module disable mode by writing a 1 to the MDIS bit in the DSPI*x*_MCR. In module disable mode, the DSPI is in a dormant state, but the memory mapped registers are still accessible. Certain read or write operations have a different affect when the DSPI is in the module disable mode. Reading the RX FIFO pop register will not change the state of the RX FIFO. Likewise, writing to the TX FIFO push register will not change the state of the TX FIFO. Clearing either of the FIFOs will not have any affect in the module disable mode. Changes to the DIS_TXF and DIS_RXF fields of the DSPI*x*_MCR will not have any affect in the module disable mode. In the module disable mode, all status bits and register flags in the DSPI will return the correct values when read, but writing to them will have no affect. Writing to the DSPI*x*_TCR during module disable mode will not have any affect. Interrupt and DMA request signals cannot be cleared while in the module disable mode.

20.4.10.2 Slave Interface Signal Gating

The DSPI's module enable signal is used to gate slave interface signals such as address, byte enable, read/write and data. This prevents toggling slave interface signals from consuming power unless the DSPI is accessed.

20.5 Initialization/Application Information

20.5.1 How to Change Queues

DSPI queues are not part of the DSPI module, but the DSPI includes features in support of queue management. Queues are primarily supported in SPI configuration. This section presents an example of how to change queues for the DSPI.

1. The last command word from a queue is executed. The EOQ bit in the command word is set to indicate to the DSPI that this is the last entry in the queue.
2. At the end of the transfer, corresponding to the command word with EOQ set is sampled, the EOQ flag (EOQF) in the DSPI*x*_SR is set.

3. The setting of the EOQF flag will disable both serial transmission, and serial reception of data, putting the DSPI in the STOPPED state. The TXRXS bit is negated to indicate the STOPPED state.
4. The eDMA will continue to fill TX FIFO until it is full or step 5 occurs.
5. Disable DSPI DMA transfers by disabling the DMA enable request for the DMA channel assigned to TX FIFO and RX FIFO. This is done by clearing the corresponding DMA enable request bits in the eDMA controller.
6. Ensure all received data in RX FIFO has been transferred to memory receive queue by reading the RXCNT in DSPI*x*_SR or by checking RFDF in the DSPI*x*_SR after each read operation of the DSPI*x*_POPR.
7. Modify DMA descriptor of TX and RX channels for “new” queues.
8. Flush TX FIFO by writing a 1 to the CLR_TXF bit in the DSPI*x*_MCR, Flush RX FIFO by writing a 1 to the CLR_RXF bit in the DSPI*x*_MCR.
9. Clear transfer count either by setting CTCNT bit in the command word of the first entry in the new queue or via CPU writing directly to SPI_TCNT field in the DSPI*x*_TCR.
10. Enable DMA channel by enabling the DMA enable request for the DMA channel assigned to the DSPI TX FIFO, and RX FIFO by setting the corresponding DMA set enable request bit.
11. Enable serial transmission and serial reception of data by clearing the EOQF bit.

20.5.2 Baud Rate Settings

Table 20-31 shows the baud rate that is generated based on the combination of the baud rate prescaler PBR and the baud rate scaler BR in the DSPI*x*_CTARs. The values calculated assume a 100 MHz system frequency.

Table 20-31. Baud Rate Values

		Baud Rate Divider Prescaler Values (DSPI_CTAR[PBR])			
		2	3	5	7
Baud Rate Scaler Values (DSPI_CTAR[BR])	2	25.0MHz	16.7MHz	10.0MHz	7.14MHz
	4	12.5MHz	8.33MHz	5.00MHz	3.57MHz
	6	8.33MHz	5.56MHz	3.33MHz	2.38MHz
	8	6.25MHz	4.17MHz	2.50MHz	1.79MHz
	16	3.12MHz	2.08MHz	1.25MHz	893kHz
	32	1.56MHz	1.04MHz	625kHz	446kHz
	64	781kHz	521kHz	312kHz	223kHz
	128	391kHz	260kHz	156kHz	112kHz
	256	195kHz	130kHz	78.1kHz	55.8kHz
	512	97.7kHz	65.1kHz	39.1kHz	27.9kHz
	1024	48.8kHz	32.6kHz	19.5kHz	14.0kHz
	2048	24.4kHz	16.3kHz	9.77kHz	6.98kHz
	4096	12.2kHz	8.14kHz	4.88kHz	3.49kHz
	8192	6.10kHz	4.07kHz	2.44kHz	1.74kHz
	16384	3.05kHz	2.04kHz	1.22kHz	872Hz
	32768	1.53kHz	1.02kHz	610Hz	436Hz

20.5.3 Delay Settings

Table 20-32 shows the values for the delay after transfer (t_{DT}) and CS to SCK delay (t_{CSC}) that can be generated based on the prescaler values and the scaler values set in the DSPI*x*_CTARs. The values calculated assume a 100 MHz system frequency.

Table 20-32. Delay Values

Delay Scaler Values (DSPI_CTAR[DT])	Delay Prescaler Values (DSPI_CTAR[PBR])			
	1	3	5	7
2	20.0 ns	60.0 ns	100.0 ns	140.0 ns
4	40.0 ns	120.0 ns	200.0 ns	280.0 ns
8	80.0 ns	240.0 ns	400.0 ns	560.0 ns
16	160.0 ns	480.0 ns	800.0 ns	1.1 µs
32	320.0 ns	960.0 ns	1.6 µs	2.2 µs
64	640.0 ns	1.9 µs	3.2 µs	4.5 µs
128	1.3 µs	3.8 µs	6.4 µs	9.0 µs
256	2.6 µs	7.7 µs	12.8 µs	17.9 µs
512	5.1 µs	15.4 µs	25.6 µs	35.8 µs
1024	10.2 µs	30.7 µs	51.2 µs	71.7 µs
2048	20.5 µs	61.4 µs	102.4 µs	143.4 µs
4096	41.0 µs	122.9 µs	204.8 µs	286.7 µs
8192	81.9 µs	245.8 µs	409.6 µs	573.4 µs
16384	163.8 µs	491.5 µs	819.2 µs	1.1 ms
32768	327.7 µs	983.0 µs	1.6 ms	2.3 ms
65536	655.4 µs	2.0 ms	3.3 ms	4.6 ms

20.5.4 MPC5xx QSPI Compatibility with the DSPI

Table 20-33 shows the translation of commands written to the TX FIFO command halfword with commands written to the command RAM of the MPC5xx family QSPI. The table illustrates how to configure the DSPI*x*_CTARs to match the default cases for the possible combinations of the MPC5xx family control bits in its command RAM. The defaults for the MPC5xx family are based on a system clock of 40MHz. All delay variables below will generate the same delay, or as close as possible, from the DSPI 100MHz system clock that an MPC5xx family part would generate from its 40MHz system clock. For other system clock frequencies, the customer can recompute the values using the information presented in Section 20.5.3, "Delay Settings."

For BITSE = 0 --> 8 bits per transfer

For DT = 0 --> 0.425µs delay: For this value, the closest value in the DSPI is 0.480µs

For DSCK = 0 --> 1/2 SCK period: For this value, the value for the DSPI is 20ns

Table 20-33. MPC5xx QSPI Compatibility with the DSPI

MPC5xx Family Control Bits DSPI Corresponding Control Bits						Corresponding DSPI*x*_CTAR Register Configuration					
BITSE	CTAS[0]	DT	CTAS[1]	DSCK	CTAS[2]	DSPI*x*_CTAR*x*	FMSZ	PDT	DT	PCSSCK	CSSCK
0		0		0		0	0111	10	0011	00	0000
0		0		1		1	0111	10	0011	User	User
0		1		0		2	0111	User[1]	User	00	0000
0		1		1		3	0111	User	User	User	User
1		0		0		4	User	10	0011	00	0000
1		0		1		5	User	10	0011	User	User
1		1		0		6	User	User	User	00	0000
1		1		1		7	User	User	User	User	User

[1]Selected by user

20.5.5 Calculation of FIFO Pointer Addresses

The user has complete visibility of the TX and RX FIFO contents through the FIFO registers, and valid entries can be identified through a memory mapped pointer and a memory mapped counter for each FIFO. The pointer to the first-in entry in each FIFO is memory mapped. For the TX FIFO the first-in pointer is the transmit next pointer (TXNXTPTR). For the RX FIFO the first-in pointer is the pop next pointer (POPNXTPTR). Figure 20-45 illustrates the concept of first-in and last-in FIFO entries along with the FIFO counter. The TX FIFO is chosen for the illustration, but the concepts carry over to the RX FIFO. See Section 20.4.3.4, "Transmit First In First Out (TX FIFO) Buffering Mechanism," and Section 20.4.3.5, "Receive First In First Out (RX FIFO) Buffering Mechanism," for details on the FIFO operation.

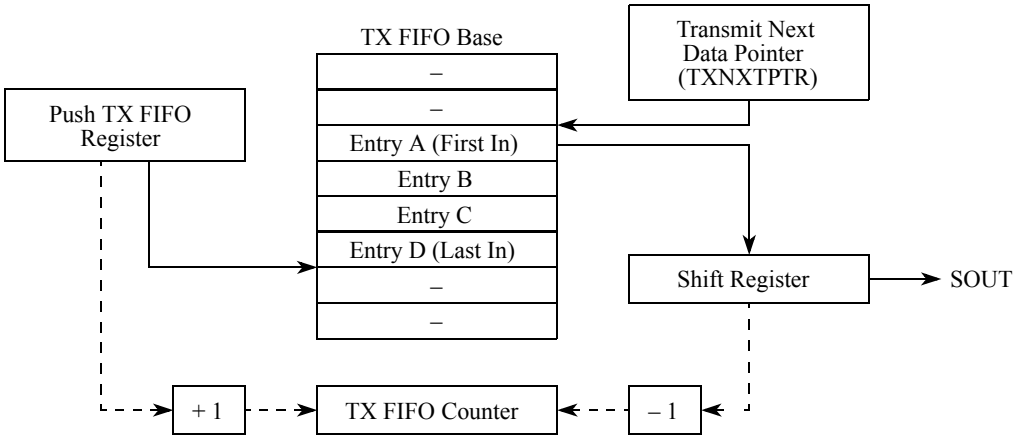


Figure 20-45. TX FIFO Pointers and Counter

20.5.5.1 Address Calculation for the First-in Entry and Last-in Entry in the TX FIFO

The memory address of the first-in entry in the TX FIFO is computed by the following equation:

First-in entry address = TXFIFO base + 4 (TXNXTPTR)

The memory address of the last-in entry in the TX FIFO is computed by the following equation:

Last-in entry address = TX FIFO base + 4*[(TXCTR + TXNXTPTR - 1) modulo TX FIFO depth]

where:

TX FIFO base: base address of TX FIFO

TXCTR: TX FIFO counter

TXNXTPTR: transmit next pointer

TX FIFO depth: transmit FIFO depth, implementation specific

20.5.5.2 Address Calculation for the First-in Entry and Last-in Entry in the RX FIFO

The memory address of the first-in entry in the RX FIFO is computed by the following equation:

First-in entry address = TX FIFO base + 4*(POPNXTPTR)

The memory address of the last-in entry in the RX FIFO is computed by the following equation:

Last-in entry address = RX FIFO base + 4*[(RXCTR + POPNXTPTR - 1) modulo RX FIFO depth]

RX FIFO base: base address of RX FIFO

RXCTR: RX FIFO counter

POPNXTPTR: pop next pointer

RX FIFO depth: Receive FIFO depth, implementation specific

20.6 Revision History

Substantive Changes since Rev 3.0
No changes.

Chapter 21
Enhanced Serial Communication Interface (eSCI)

21.1 Introduction

This section gives an overview of the MPC5553/MPC5554's eSCI module, and presents a block diagram, its features and its modes of operation.

21.1.1 Block Diagram

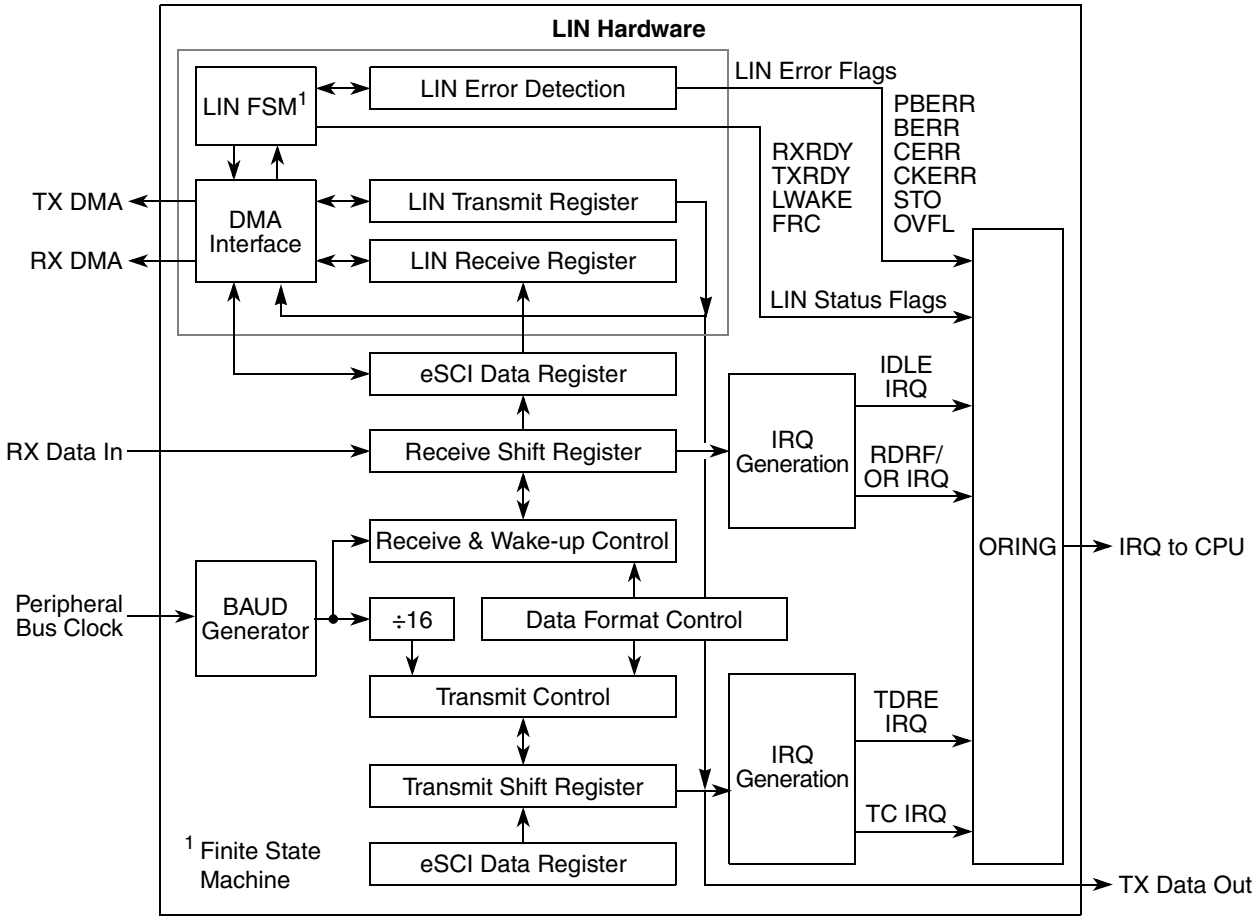


Figure 21-1. eSCI Block Diagram

21.1.2 Overview

The eSCI allows asynchronous serial communications with peripheral devices and other CPUs. The eSCI has special features which allow the eSCI to operate as a LIN bus master, complying with the LIN 2.0 specification.

Each of the eSCI modules can be independently disabled by writing to the module disable (MDIS) bit in the module's control register 2 (ESCI*x*_CR2). Disabling the module turns off the clock to the module, although some of the module registers may be accessed by the core via the slave bus. The MDIS bit is intended to be used when the module is not required in the application.

21.1.3 Features

The eSCI includes these distinctive features:

- Full-duplex operation
- Standard mark/space non-return-to-zero (NRZ) format
- Configurable baud rate
- Programmable 8-bit or 9-bit data format
- LIN master node support
- Configurable CRC detection for LIN
- Separately enabled transmitter and receiver
- Programmable transmitter output parity
- Two receiver wake-up methods:
  - Idle line wake-up
  - Address mark wake-up
- Interrupt-driven operation
- Receiver framing error detection
- Hardware parity checking
- 1/16 bit-time noise detection
- Two-channel DMA interface

21.1.4 Modes of Operation

The eSCI functions the same in normal, special, and emulation modes. It has a low-power module disable mode.

21.2 External Signal Description

This section provides a description of all module signals external to the MCU.

21.2.1 Overview

Each eSCI module has two I/O signals connected to the external MCU pins. These signals are summarized in Table 21-1 and described in more detail in the following sections.

Table 21-1. eSCI Signals

Signal Name[1]	I/O	Description
RXD*x*	I	eSCI Receive
TXD*x*	O	eSCI Transmit

1 *x* indicates eSCI module A or B

21.2.2 Detailed Signal Description

21.2.2.1 SCI Transmit (TXD*x*)

This pin serves as transmit data output of eSCI.

21.2.2.2 SCI Receive Pin (RXD*x*)

This pin serves as receive data input of the eSCI.

21.3 Memory Map/Register Definition

21.3.1 Overview

This section provides a detailed description of all memory and registers.

21.3.2 Module Memory Map

The memory map for the eSCI module is given below in Table 21-2. The address offset is listed for each register. The total address for each register is the sum of the base address for the eSCI module (ESCI*x*_base) and the address offset for each register. There are two eSCI modules on the MPC5553/MPC5554: the base is 0xFFFB_0000 for eSCIA and 0xFFFB_4000 for eSCIB.

Table 21-2. Module Memory Map

Address	Register Name	Register Description	Size (bits)
Base 0xFFFB_0000 (A) 0xFFFB_4000 (B)	ESCI*x*_CR1	eSCI control register 1	32
Base + 0x04	ESCI*x*_CR2	eSCI control register 2	16
Base + 0x06	ESCI*x*_DR	eSCI data register	16

Table 21-2. Module Memory Map (continued)

Address	Register Name	Register Description	Size (bits)
Base + 0x08	ESCI*x*_SR	eSCI status register	32
Base + 0x0C	ESCI*x*_LCR	LIN control register	32
Base + 0x10	ESCI*x*_LTR	LIN transmit register	32
Base + 0x14	ESCI*x*_LRR	LIN receive register	32
Base + 0x18	ESCI*x*_LPR	LIN cyclic redundancy check polynomial register	32

21.3.3 Register Definition

This section consists of register descriptions in address order. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

21.3.3.1 eSCI Control Register 1 (ESCI*x*_CR1)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	SBR 0	SBR 1	SBR 2	SBR 3	SBR 4	SBR 5	SBR 6	SBR 7	SBR 8	SBR 9	SBR 10	SBR 11	SBR 12
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
Reg Addr	Base + 0x0															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LOOPS	0	RSRC	M	WAKE	ILT	PE	PT	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0															

Figure 21-2. eSCI Control Register 1 (ESCI*x*_CR1)

Table 21-3. ESCI*x*_CR1 Field Descriptions

Bits	Name	Description
0–2	—	Reserved.
3–15	SBR*n*	SCI baud rate. Used by the counter to determine the baud rate of the eSCI. The formula for calculating the baud rate is: $\text{SCI baud rate} = \frac{\text{eSCI system clock}}{16 \times \text{BR}}$ where BR is the content of the eSCI control register 1 (ESCI*x*_CR1), bits SBR0–SBR12. SBR0–SBR12 can contain a value from 1 to 8191. Also refer to the ESCIx_LCR[WU] bit description on page 21-13.

Table 21-3. ESCI*x*_CR1 Field Descriptions

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>16</td><td>LOOPS</td><td>Loop select. Enables loop operation. In loop operation, the RXD pin is disconnected from the eSCI and the transmitter output is internally connected to the receiver input. Both the transmitter and the receiver must be enabled to use the loop function.
0 Normal operation enabled, loop operation disabled
1 Loop operation enabled
Note: The receiver input is determined by the RSRC bit.</td></tr>
<tr><td>17</td><td>—</td><td>Reserved.</td></tr>
<tr><td>18</td><td>RSRC</td><td>Receiver source. When LOOPS = 1, the RSRC bit determines the source for the receiver shift register input.
0 Receiver input internally connected to transmitter output
1 Receiver input connected externally to transmitter
The table below shows how LOOPS and RSRC determine the loop function of the eSCI.
<table>
<tr><th>LOOPS</th><th>RSRC</th><th>Function</th></tr>
<tr><td>0</td><td>x</td><td>Normal operation</td></tr>
<tr><td>1</td><td>0</td><td>Loop mode with RXD input internally connected to TXD output</td></tr>
<tr><td>1</td><td>1</td><td>Single-wire mode with RXD input connected to TXD</td></tr>
</table>
</td></tr>
<tr><td>19</td><td>M</td><td>Data format mode. Determines whether data characters are 8 or 9 bits long.
0 1 start bit, 8 data bits, 1 stop bit
1 1 start bit, 9 data bits, 1 stop bit</td></tr>
<tr><td>20</td><td>WAKE</td><td>Wake-up condition. Determines which condition wakes up the eSCI: a logic 1 (address mark) in the most significant bit position of a received data character or an idle condition on the RXD.
0 Idle line wake-up
1 Address mark wake-up
Note: This is not a wake-up out of a power-save mode, it refers solely to the receiver standby mode.</td></tr>
<tr><td>21</td><td>ILT</td><td>Idle line type. Determines when the receiver starts counting logic 1s as idle character bits. The counting begins either after the start bit or after the stop bit. If the count begins after the start bit, then a string of logic 1s preceding the stop bit may cause false recognition of an idle character. Beginning the count after the stop bit avoids false idle character recognition, but requires properly synchronized transmissions.
0 Idle character bit count begins after start bit
1 Idle character bit count begins after stop bit</td></tr>
<tr><td>22</td><td>PE</td><td>Parity enable. Enables the parity function. When enabled, the parity function inserts a parity bit in the most significant bit position of the transmitted word. During reception, the received parity bit will be verified in the most significant bit position. The received parity bit will not be masked out.
0 Parity function disabled
1 Parity function enabled</td></tr>
</table>

Table 21-3. ESCI*x*_CR1 Field Descriptions

Bits	Name	Description
23	PT	Parity type. Determines whether the eSCI generates and checks for even parity or odd parity. With even parity, an even number of 1s clears the parity bit and an odd number of 1s sets the parity bit. With odd parity, an odd number of 1s clears the parity bit and an even number of 1s sets the parity bit. 0 Even parity 1 Odd parity
24	TIE	Transmitter interrupt enable. Enables the transmit data register empty flag ESCI*x*_SR[TDRE] to generate interrupt requests. The interrupt is suppressed in TX DMA mode. 0 TDRE interrupt requests disabled 1 TDRE interrupt requests enabled
25	TCIE	Transmission complete interrupt enable. Enables the transmission complete flag ESCIx_SR[TC] to generate interrupt requests. The interrupt is suppressed in TX DMA mode. 0 TC interrupt requests disabled 1 TC interrupt requests enabled
26	RIE	Receiver full interrupt enable. Enables the receive data register full flag ESCIx_SR[RDRF] and the overrun flag ESCIx_SR[OR] to generate interrupt requests. The interrupt is suppressed in RX DMA mode. 0 RDRF and OR interrupt requests disabled 1 RDRF and OR interrupt requests enabled
27	ILIE	Idle line interrupt enable. Enables the idle line flag ESCIx_SR[IDLE] to generate interrupt requests. 0 IDLE interrupt requests disabled 1 IDLE interrupt requests enabled
28	TE	Transmitter enable. Enables the eSCI transmitter and configures the TXD pin as being controlled by the eSCI. The TE bit can be used to queue an idle preamble. 0 Transmitter disabled 1 Transmitter enabled
29	RE	Receiver enable. Enables the eSCI receiver. 0 Receiver disabled 1 Receiver enabled
30	RWU	Receiver wake-up. Standby state. 0 Normal operation. 1 RWU enables the wake-up function and inhibits further receiver interrupt requests. Normally, hardware wakes the receiver by automatically clearing RWU.
31	SBK	Send break. Toggling SBK sends one break character (see the description of ESCI*x*_CR2[BRK13] for break character length). Toggling implies clearing the SBK bit before the break character has finished transmitting. As long as SBK is set, the transmitter continues to send complete break characters. 0 No break characters 1 Transmit break characters

NOTES

After reset, the baud rate generator is disabled until the TE bit or the RE bit is set for the first time.

The baud rate generator is disabled when SBR0–SBR12 = 0x0.

Normally the baud rate should be written with a single write. If 8-bit writes are used, writing to ESCI*x*_CR1[0–7] has no effect without writing to ESCI*x*_CR1[8–15], since writing to ESCI*x*_CR1[0–7] puts the data in a temporary location until ESCI*x*_CR1[8–15] is written to.

During reception, when parity is enabled, the received parity bit will appear in the data register.

21.3.3.2 eSCI Control Register 2 (ESCI*x*_CR2)

NOTE

DMA requests are negated when in module disable mode.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MDIS	FBR	BSTP	IEBERR	RXDMA	TXDMA	BRK13	0	BESM 13	SBSTP	0	0	ORIE	NFIE	FEIE	PFIE
W																
Reset	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x4															

Figure 21-3. eSCI Control Register 2 (ESCIx_CR2)

Table 21-4. ESCI*x*_CR2 Field Description

Bits	Name	Description
0	MDIS	Module disable. By default the module is enabled, but can be disabled by writing a 1 to this bit. DMA requests are negated if the device is in module disable mode. 0 Module enabled 1 Module disabled
1	FBR	Fast bit error detection. Handles bit error detection on a per bit basis. If this is not enabled, bit errors will be detected on a per byte basis.
2	BSTP	Bit error/physical bus error stop. Causes DMA TX requests to be suppressed, as long as the bit error and physical bus error flags are not cleared. This stops further DMA writes, which would otherwise cause data bytes to be interpreted as LIN header information.
3	IEBERR	Enable bit error interrupt. Generates an interrupt, when a LIN bit error is detected. For a list of interrupt enables and flags, see Table 21-21.
4	RXDMA	Activate RX DMA channel. If this bit is enabled and the eSCI has received data, it will raise a DMA RX request.
5	TXDMA	Activate TX DMA channel. Whenever the eSCI is able to transmit data, it will raise a DMA TX request.

Table 21-4. ESCI*x*_CR2 Field Description (continued)

Bits	Name	Description
6	BRK13	Break transmit character length. Determines whether the transmit break character is 10/11 or 13/14 bits long. The detection of a framing error is not affected by this bit. **Break Length:** BRK13 = 0: ESCI*x*_CR1[M] = 0 → 10; ESCI*x*_CR1[M] = 1 → 11 BRK13 = 1: ESCI*x*_CR1[M] = 0 → 13; ESCI*x*_CR1[M] = 1 → 14 0 Break Character is 10 or 11 bits long 1 Break character is 13 or 14 bits long **Note:** LIN 2.0 now requires that a break character is always 13 bits long, so this bit should always be set to 1. The eSCI will work with BRK13=0, but it will violate LIN 2.0.
7	—	Reserved. This bit is readable/writable, but has no effect on the operation of the eSCI module.
8	BESM13	Bit error sample mode, bit 13. Determines when to sample the incoming bit in order to detect a bit error. (This is only relevant when FBR is set.) 0 Sample at RT clock 9 1 Sample at RT clock 13 (see Section 21.4.5.3, “Data Sampling”)
9	SBSTP	SCI bit error stop. Stops the SCI when a bit error is asserted. This allows to stop driving the LIN bus quickly after a bit error has been detected. 0 Byte is completely transmitted 1 Byte is partially transmitted
10–11	—	Reserved.
12	ORIE	Overrun error interrupt enable. Generates an interrupt, when a frame error is detected. For a list of interrupt enables and flags, see Table 21-21.
13	NFIE	Noise flag interrupt enable. Generates an interrupt, when noise flag is set. For a list of interrupt enables and flags, see Table 21-21.
14	FEIE	Frame error interrupt enable. Generates an interrupt, when a frame error is detected. For a list of interrupt enables and flags, see Table 21-21.
15	PFIE	Parity flag interrupt enable. Generates an interrupt, when parity flag is set. For a list of interrupt enables and flags, see Table 21-21.

21.3.3.3 eSCI Data Register (ESCI*x*_DR)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	R8	T8	0	0	0	0	0	0	R7	R6	R5	R4	R3	R2	R1	R0
W									T7	T6	T5	T4	T3	T2	T1	T0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x6															

Figure 21-4. eSCI Data Register (ESCI*x*_DR)

Table 21-5. ESCI*x*_DR Field Description

Bits	Name	Description
0	R8	Received bit 8. R8 is the ninth data bit received when the eSCI is configured for 9-bit data format (M = 1).
1	T8	Transmit bit 8. T8 is the ninth data bit transmitted when the eSCI is configured for 9-bit data format (M = 1). **Note:** If the value of T8 is the same as in the previous transmission, T8 does not have to be rewritten.The same value is transmitted until T8 is rewritten.
2–7	—	Reserved.
8–15	R7–R0 / T7–T0	Received bits/transmit bits 7–0 for 9-bit or 8-bit formats. Bits 7–0 from SCI communication may be read from ESCI*x*_DR[8–15] (provided that SCI communication was successful). Writing to ESCI*x*_DR [8–15] provides bits 7–0 for SCI transmission.

NOTES

In 8-bit data format, only bits 8–15 of ESCI*x*_DR need to be accessed.

When transmitting in 9-bit data format and using 8-bit write instructions, write first to ESCI*x*_DR[0–7], then ESCI*x*_DR[8–15]. For 9-bit transmissions, a single write may also be used.

ESCI*x*_DR should not be used in LIN mode, writes to this register are blocked in LIN mode.

Even if parity generation/checking is enabled via ESCIx_CR[PE], the parity bit will not be masked out.

21.3.3.4 eSCI Status Register (ESCI*x*_SR)

The ESCI*x*_SR indicates the current status. The status flags can be polled, and some can also be used to generate interrupts. All bits in ESCI*x*_SR except for RAF are cleared by writing 1 to them.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF	0	0	0	BERR	0	0	0	RAF
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c				w1c				
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x8															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXRDY	TXRDY	LWAKE	STO	PBERR	CERR	CKERR	FRC	0	0	0	0	0	0	0	OVFL
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c								w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x8															

Figure 21-5. eSCI Status Register (ESCI*x*_SR)

Table 21-6. ESCI*x*_SR Field Descriptions

Bits	Name	Description
0	TDRE	Transmit data register empty flag. TDRE is set when the transmit shift register receives a byte from the eSCI data register. When TDRE is 1, the data register (ESCI*x*_DR) is empty and can receive a new value to transmit. Clear TDRE by writing 1 to it. 0 No byte transferred to transmit shift register 1 Byte transferred to transmit shift register; transmit data register empty
1	TC	Transmit complete flag. TC is set low when there is a transmission in progress or when a preamble or break character is loaded. TC is set high when the TDRE flag is set and no data, preamble, or break character is being transmitted. When TC is set, the TXD out signal becomes idle (logic 1). After the device is switched on (by clearing the MDIS bit, see Section 21.3.3.2, "eSCI Control Register 2 (ESCIx_CR2)," a preamble is transmitted; if no byte is written to the the SCI data register then the completion of the preamble can be monitored using the TC flag. Clear TC by writing 1 to it. 0 Transmission in progress 1 No transmission in progress. Indicates that TXD out is idle.
2	RDRF	Receive data register full flag. RDRF is set when the data in the receive shift register transfers to the eSCI data register. Clear RDRF by writing 1 to it. 0 Data not available in eSCI data register 1 Received data available in eSCI data register
3	IDLE	Idle line flag. IDLE is set when 10 consecutive logic 1s (if M = 0) or 11 consecutive logic 1s (if M = 1) appear on the receiver input. Once the IDLE flag is cleared, a valid frame must again set the RDRF flag before an idle condition can set the IDLE flag. Clear IDLE by writing 1 to it. 0 Receiver input is either active now or has never become active since the IDLE flag was last cleared 1 Receiver input has become idle **Note:** When the receiver wake-up bit (RWU) is set, an idle line condition does not set the IDLE flag.

Table 21-6. ESCI*x*_SR Field Descriptions (continued)

Bits	Name	Description
4	OR	Overrun flag. OR is set when software fails to read the eSCI data register before the receive shift register receives the next frame. The OR bit is set immediately after the stop bit has been completely received for the second frame. The data in the shift register is lost, but the data already in the eSCI data registers is not affected. Clear OR by writing 1 to it. 0 No overrun 1 Overrun
5	NF	Noise flag. NF is set when the eSCI detects noise on the receiver input. NF bit is set during the same cycle as the RDRF flag but does not get set in the case of an overrun. Clear NF by writing 1 to it. 0 No noise 1 Noise
6	FE	Framing error flag. FE is set when a logic 0 is accepted as the stop bit. FE bit is set during the same cycle as the RDRF flag but does not get set in the case of an overrun. Clear FE by writing 1 to it. 0 No framing error 1 Framing error
7	PF	Parity error flag. PF is set when the parity enable bit, PE, is set and the parity of the received data does not match its parity bit. Clear PE by writing 1 to it. 0 No parity error 1 Parity error
8–10	—	Reserved, should be cleared.
11	BERR	Bit error. Indicates a bit on the bus did not match the transmitted bit. If FBR = 0, checking happens after a complete byte has been transmitted and received again. If FBR = 1, checking happens bit by bit. This bit is only used for LIN mode. BERR is also set if an unrequested byte is received (i.e. a byte that is not part of an RX frame) that is not recognized as a wake-up flag. (Because the data on the RX line does not match the idle state that was assigned to the TX line.) Clear BERR by writing 1 to it. A bit error causes the LIN finite state machine (FSM) to reset unless ESCI*x*_LCR[LDBG] is set.
12–14	—	Reserved.
15	RAF	Receiver active flag. RAF is set when the receiver detects a logic 0 during the RT1 time period of the start bit search. RAF is cleared when the receiver detects an idle character. 0 No reception in progress. 1 Reception in progress.
16	RXRDY	The eSCI has received LIN data. This bit is set when the ESCI*x*_LCR receives a byte. Clear RXRDY by writing it with 1.
17	TXRDY	The LIN FSM can accept another write to ESCI*x*_LTR. This bit is set when the ESCI*x*_LTR register becomes free. Clear TXRDY by writing it with 1.
18	LWAKE	Received LIN wake-up signal. A LIN slave has sent a wake-up signal on the bus. When this signal is detected, the LIN FSM will reset. If the setup of a frame had already started, it therefore must be repeated. LWAKE will also be set if ESCI receives a LIN 2.0 wake-up signal (in which the baud rate is lower than 32K baud). See the WU bit.

Table 21-6. ESCI*x*_SR Field Descriptions (continued)

Bits	Name	Description
19	STO	Slave time out. Represents a NO_RESPONSE_ERROR. This is set if a slave does not complete a frame within the specified maximum frame length. For LIN 1.3 the following formula is used: $TFRAME\_MAX = (10 \times NDATA + 44) \times 1.4$
20	PBERR	Physical bus error. No valid message can be generated on the bus. This is set if, after the start of a byte transmission, the input remains unchanged for 31 cycles. This will reset the LIN FSM.
21	CERR	CRC error. The CRC pattern received with an extended frame was not correct. 0 No error 1 CRC error
22	CKERR	Checksum error. Checksum error on a received frame.
23	FRC	Frame complete. LIN frame completely transmitted. All LIN data bytes received.
24–30	—	Reserved.
31	OVFL	ESCI*x*_LRR overflow. The LIN receive register has not been read before a new data byte, CRC, or checksum byte has been received from the LIN bus. Set when the condition is detected, and cleared by writing 1 to it.

21.3.3.5 LIN Control Register (ESCI*x*_LCR)

ESCI*x*_LCR can be written only when there are no ongoing transmissions.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	LRES	0	WUD0	WUD1	LDBG	DSF	PRTY	LIN	RXIE	TXIE	WUIE	STIE	PBIE	CIE	CKIE	FCIE
W		WU														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0xC															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	OFIE	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0xC															

Figure 21-6. LIN Control Register (ESCI*x*_LCR)

Table 21-7. ESCI*x*_LCR Field Descriptions

Bits	Name	Description
0	LRES	LIN resynchronize. Causes the LIN protocol engine to return to start state. This happens automatically after bit errors, but software may force a return to start state manually via this bit. The bit first must be set then cleared, so that the protocol engine is operational again.
1	WU	LIN bus wake-up. Generates a wake-up signal on the LIN bus. This must be set before a transmission, if the bus is in sleep mode. This bit will auto-clear, so a read from this bit will always return 0. According to LIN 2.0, generating a valid wake-up character requires programming the SCI baud rate to a range of 32K baud down to 1.6K baud.
2–3	WUD [0:1]	Wake-up delimiter time. Determines how long the LIN engine waits after generating a wake-up signal, before starting a new frame. The eSCI will not set ESCIx_SR[TXRDY] before this time expires. Note that in addition to this delimiter time, the CPU and the eSCI will require some setup time to start a new transmission and typically there is an additional bit time delay. The table below shows how the values for WUD0 and WUD1 affect the delimiter time.
4	LDBG	LIN debug mode. Prevents the LIN FSM from automatically resetting, after an exception (bit error, physical bus error, wake-up flag) has been received. This is for debug purposes only.
5	DSF	Double stop flags. When a bit error has been detected, this will add an additional stop flag to the byte in which the error occurred.
6	PRTY	Activating parity generation. Generate the two parity bits in the LIN header.
7	LIN	LIN mode. Switch device into LIN mode. 0 LIN disabled 1 LIN enabled
8	RXIE	LIN RXREG ready interrupt enable. Generates an Interrupt when new data is available in the LIN RXREG. For a list of interrupt enables and flags, see Table 21-21.
9	TXIE	LIN TXREG ready interrupt enable. Generates an Interrupt when new data can be written to the LIN TXREG. For a list of interrupt enables and flags, see Table 21-21.
10	WUIE	RX wake-up interrupt enable. Generates an Interrupt when a wake-up flag from a LIN slave has been received. For a list of interrupt enables and flags, see Table 21-21.
11	STIE	Slave timeout error interrupt enable. Generates an Interrupt when the slave response is too slow. For a list of interrupt enables and flags, see Table 21-21.
12	PBIE	Physical bus error interrupt enable. Generates an Interrupt when no valid message can be generated on the bus. For a list of interrupt enables and flags, see Table 21-21.
13	CIE	CRC error interrupt enable. Generates an Interrupt when a CRC error on a received extended frame is detected. For a list of interrupt enables and flags, see Table 21-21.

WUD0	WUD1	Bit Times
0	0	4
0	1	8
1	0	32
1	1	64

Table 21-7. ESCI*x*_LCR Field Descriptions (continued)

Bits	Name	Description
14	CKIE	Checksum error interrupt enable. Generates an Interrupt on a detected checksum error. For a list of interrupt enables and flags, see Table 21-21.
15	FCIE	Frame complete interrupt enable. Generates an Interrupt after complete transmission of a TX frame, or after the last byte of an RX frame is received. (The complete frame includes all header, data, CRC and checksum bytes as applicable.) For a list of interrupt enables and flags, see Table 21-21.
16–22	—	Reserved.
23	OFIE	Overflow interrupt enable. Generates an Interrupt when a data byte in the ESCIx_LRR has not been read before the next data byte is received. For a list of interrupt enables and flags, see Table 21-21.
24–31	—	Reserved.

21.3.3.6 LIN Transmit Register (ESCI*x*_LTR)

ESCI*x*_LTR can be written to only when TXRDY is set. The first byte written to the register selects the transmit address, the second byte determines the frame length, the third and fourth byte set various frame options and determine the timeout counter. Header parity will be automatically generated if the ESCI*x*_LCR[PRTY] bit is set. For TX frames, the fourth byte (bits T7–T0) is skipped, since the timeout function does not apply. All following bytes are data bytes for the frame. CRC and checksum bytes will be automatically appended when the appropriate options are selected.

When a bit error is detected, an interrupt is set and the transmission aborted. The register can only be written again once the interrupt is cleared. Afterwards a new frame starts, and the first byte needs to contain a header again.

Additionally it is possible to flush the ESCI*x*_LTR by setting the ESCI*x*_LCR[LRES] bit.

NOTE

Not all values written to the ESCI*x*_LTR will generate valid LIN frames. The values are determined according to the LIN specification.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R									0	0	0	0	0	0	0	0
W	P1/ L7/ HDCHK/ T7/ D7	P0/ L6/ CSUM/ T6/ D6	ID5/ L5/ CRC/ T5/ D5	ID4/ L4/ TX/ T4/ D4	ID3/ L3/ T11/ T3/ D3	ID2/ L2/ T10/ T2/ D2	ID1/ L1/ T9/ T1/ D1	ID0/ L0/ T8/ T0/ D0								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x10															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x10															

Figure 21-7. LIN Transmit Register (ESCI*x*_LTR)

	0	1	2	3	4	5	6	7
R								
1st Write (Table 21-8) W	P[1:0]		ID[5:0]					
2nd Write (Table 21-9) W	L[7:0]							
3rd Write (Table 21-10) W	HDCHK	CSUM	CRC	TX ($\overline{RX}$)	T[11:8]			
4th Write (Table 21-11) W	T[7:0]							
5th Write (Table 21-12) W	D[7:0]							
Reset	0	0	0	0	0	0	0	0
Reg Addr	eSCI *x* Base + 0x0010							

Figure 21-8. LIN Transmit Register (ESCIx_LTR) Alternate Diagram

Table 21-8. ESCI*x*_LTR First Byte Field Description

Bits	Name	Description
0–1	P*n*	Parity bit *n*. When parity generation is enabled (ESCI*x*_LCR[PRTY] = 1), the parity bits are generated automatically. Otherwise they must be provided in this field.
2–7	ID*n*[1]	Header bit *n*. The LIN address, for LIN 1.x standard frames the length bits must be set appropriately (see the table below), extended frames are recognized by their specific patterns.
8–31	—	Reserved.

ID5	ID4	data bytes
0	0	2
0	1	2
1	0	4
1	1	8

[1] The values 3C, 3D, 3E and 3F of the ID-field (ID0-5) indicate command and extended frames.Refer to LIN Specification Package Revision 2.0.

Table 21-9. ESCI*x*_LTR Second Byte Field Descriptions

Bits	Name	Description
0–7	L*n*	Length bit *n*. Defines the length of the frame (0 to 255 data bytes). This information is needed by the LIN state machine in order to insert the checksum or CRC pattern as required. LIN 1.x slaves will only accept frames with 2, 4, or 8 data bytes.
8–31	—	Reserved.

Table 21-10. ESCI*x*_LTR Third Byte Field Descriptions

Bits	Name	Description
0	HDCHK	Header checksum enable. Include the header fields into the mod 256 checksum of the standard frames.
1	CSUM	Checksum enable. Append a checksum byte to the end of a TX frame. Verify the checksum byte of a RX frame.
2	CRC	CRC enable. Append two CRC bytes to the end of a TX frame. Verify the two CRC bytes of a RX frame are correct. If both CSUM and CRC bits are set, the LIN FSM will first append the CRC bytes, then the checksum byte, and will expect them in this order, as well. If HDCHK is set, the CRC calculation will include header and data bytes, otherwise just the data bytes. CRC bytes are not part of the LIN standard; they are normal data bytes and belong to a higher-level protocol.
3	TX	Transmit direction. Indicates a TX frame; that is, the eSCI will transmit data to a slave. Otherwise, an RX frame is assumed, and the eSCI only transmits the header. The data bytes are received from the slave. 0 RX frame 1 TX frame

Table 21-10. ESCI*x*_LTR Third Byte Field Descriptions (continued)

Bits	Name	Description
4–7	T*n*	Timeout bit *n*. Sets the counter to determine a NO_RESPONSE_ERROR, if the frame is a read access to a LIN slave. Following LIN standard rev 1.3, the value $(10 \times N_{DATA} + 45) \times 1.4$ is recommended. For transmissions, this counter has to be set to 0. The timeout bits 7–0 will not be written on a TX frame. For TX frames, the fourth byte written to the LIN transmit register (ESCI*x*_LTR) is the first data byte, for RX frames it contains timeout bits 7–0.The time is specified in multiples of bit times. The timeout period starts with the transmission of the LIN break character.
8–31	—	Reserved.

Table 21-11. ESCI*x*_LTR Rx Frame Fourth Byte Field Descriptions

Bits	Name	Description
0–7	T*n*	Timeout bit *n*. Sets the counter to determine a NO_RESPONSE_ERROR, if the frame is a read access to a LIN slave. Following LIN standard rev 1.3, the value $(10 \times N_{DATA} + 45) \times 1.4$ is recommended. For transmissions, this counter has to be set to 0. The timeout bits 7–0 will not be written on a TX frame. For TX frames, the fourth byte written to the LIN transmit register (ESCI*x*_LTR) is the first data byte. For RX frames, it contains timeout bits 7–0.The time is specified in multiples of bit times. The timeout period starts with the transmission of the LIN break character.
8–31	—	Reserved.

Table 21-12. ESCI*x*_LTR Tx Frame Fourth+ Byte/ Rx Frame Fifth+ Byte Field Description

Bits	Name	Description
0–7	D*n*	Data bits for transmission.
8–31	—	Reserved.

21.3.3.7 LIN Receive Register (ESCI*x*_LRR)

ESCI*x*_LRR can be ready only when ESCI*x*_SR[RXRDY] is set.

NOTE

Application software must ensure that ESCI*x*_LRR be read before new data or checksum bytes or CRCs are received from the LIN bus.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x14															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x14															

Figure 21-9. LIN Receive Register (ESCI*x*_LRR)

Table 21-13. ESCI*x*_LRR Field Descriptions

Bits	Name	Description
0–7	D*n*	Data bit *n*. Provides received data bytes from RX frames. Data is only valid when the ESCI*x*_SR[RXRDY] flag is set. CRC and checksum information will not be available in the ESCI*x*_LRR unless they are treated as data. It is possible to treat CRC and checksum bytes as data by deactivating the CSUM respectively CRC control bits in the ESCI*x*_LTR; however, then CRC and CSUM checking has to be performed by software. Data bytes must be read from the ESCI*x*_LRR (by CPU or DMA) before any new bytes (including CRC or checksum) are received from the LIN bus otherwise the data byte is lost and OVFL is set. **Note:** The data must be collected and the LIN frame finished (including CRC and checksum if applicable) before a wake-up character can be sent.
8–31	—	Reserved.

21.3.3.8 LIN CRC Polynomial Register (ESCI*x*_LPR)

ESCI*x*_LPR*n* can be written when there are no ongoing transmissions.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
W																
Reset	1	1	0	0	0	1	0	1	1	0	0	1	1	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x18															

Figure 21-10. LIN CRC Polynomial Register (ESCI*x*_LPR)

Table 21-14. ESCI*x*_LPR Field Description

Bits	Name	Description
0–15	P*n*	Polynomial bit x^n. Bits P15–P0 are used to define the LIN polynomial - standard is $x^{15} + x^{14} + x^{10} + x^8 + x^7 + x^4 + x^3 + 1$ (the polynomial used for the CAN protocol).
16–31	—	Reserved.

21.4 Functional Description

21.4.1 Overview

This section provides a complete functional description of the eSCI module, detailing the operation of the design from the end user perspective in a number of subsections.

Figure 21-11 shows the structure of the eSCI module. The eSCI allows full duplex, asynchronous, NRZ serial communication between the CPU and remote devices, including other CPUs. The eSCI transmitter and receiver operate independently, although they use the same baud rate generator. The CPU monitors the status of the eSCI, writes the data to be transmitted, and processes received data.

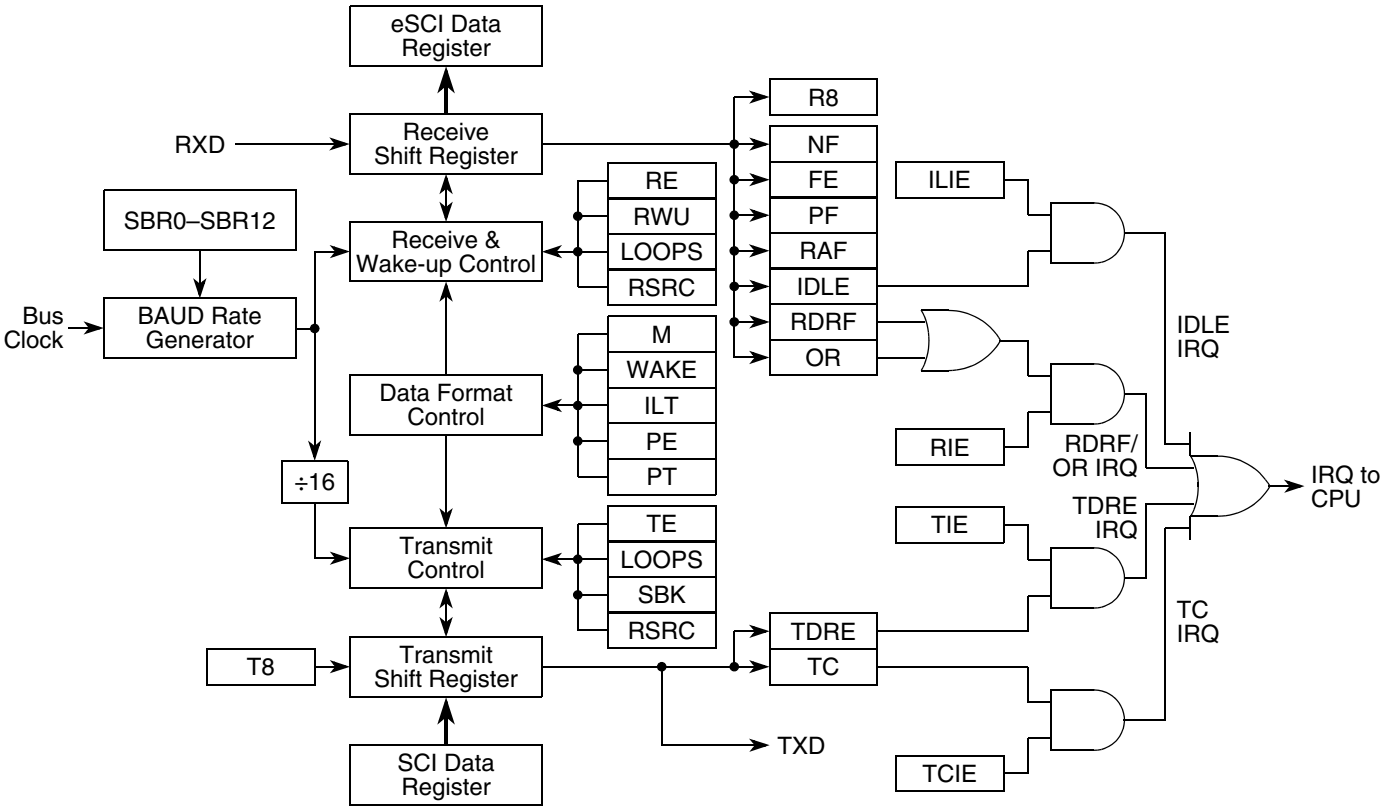


Figure 21-11. eSCI Operation Block Diagram

21.4.2 Data Format

The eSCI uses the standard NRZ mark/space data format. Each data character is contained in a frame that includes a start bit, eight or nine data bits, and a stop bit. Clearing the M bit in eSCI control register 1

configures the eSCI for 8-bit data characters. A frame with eight data bits has a total of 10 bits. Setting the M bit configures the eSCI for 9-bit data characters. A frame with nine data bits has a total of 11 bits.

When the eSCI is configured for 9-bit data characters, the ninth data bit is the T8 bit in the eSCI data register (ESCI*x*_DR). It remains unchanged after transmission and can be used repeatedly without rewriting it. A frame with nine data bits has a total of 11 bits.

The two different data formats are illustrated in Figure 21-12. Table 21-15 and Table 21-16 show the number of each type of bit in 8-bit data format and 9-bit data format, respectively.

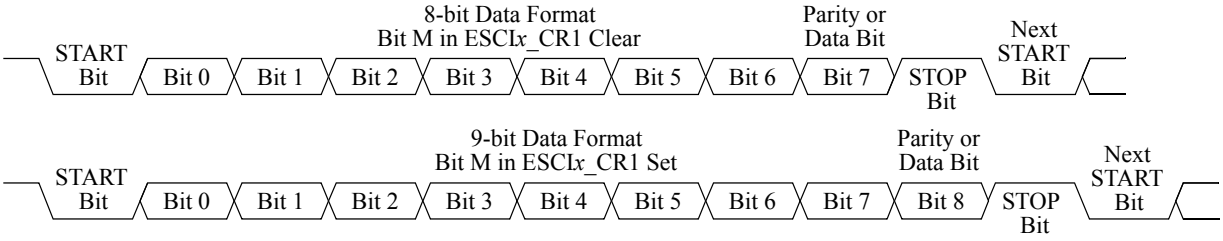


Figure 21-12. eSCI Data Formats

Table 21-15. Example of 8-bit Data Formats

Start Bit	Data Bits	Address Bits	Parity Bits	Stop Bit
1	8	0	0	1
1	7	0	1	1
1	7	1[1]	0	1

[1]The address bit identifies the frame as an address character. See Section 21.4.5.6, “Receiver Wake-up.”

Table 21-16. Example of 9-Bit Data Formats

Start Bit	Data Bits	Address Bits	Parity Bits	Stop Bit
1	9	0	0	1
1	8	0	1	1
1	8	1[1]	0	1

[1]The address bit identifies the frame as an address character. See Section 21.4.5.6, “Receiver Wake-up.”

21.4.3 Baud Rate Generation

A 13-bit modulus counter in the baud rate generator derives the baud rate for both the receiver and the transmitter. The value, 1 to 8191, written to the SBR0–SBR12 bits determines the system clock divider. The SBR bits are in the eSCI control register 1 (ESCI*x*_CR1). The baud rate clock is synchronized with the system clock and drives the receiver. The baud rate clock divided by 16 drives the transmitter. The receiver has an acquisition rate of 16 samples per bit time.

Baud rate generation is subject to one source of error:

- Integer division of the system clock may not give the exact target frequency.

Table 21-17 lists some examples of achieving target baud rates with a system clock frequency of 128 MHz.

$$\text{SCI baud rate} = \frac{\text{System clock}}{16 \times \text{ESCIx\_CR1[SBR]}}$$

Table 21-17. Baud Rates (Example: System Clock = 128 Mhz)

Bits SBR[0:12]	Receiver Clock (Hz)	Transmitter Clock (Hz)	Target Baud Rate	Error (%)
0x0023	3,657,143	228,571	230,400	–0.79
0x0045	1,855,072	115,942	115,200	+0.64
0x008B	920,863	57,554	57,600	–0.01
0x00D0	615,385	38,462	38,400	+0.16
0x01A1	306,954	19,185	19,200	–0.08
0x022C	230,216	14,388	14,400	–0.08
0x0341	153,661	9,604	9600	+.04
0x0683	76,785	4,799	4800	–0.02
0x0D05	38,404	2,400.2	2400	+.01
0x1A0A	19,202	1,200.1	1200	+.01

21.4.4 Transmitter

Figure 21-13 illustrates the features of the eSCI transmitter.

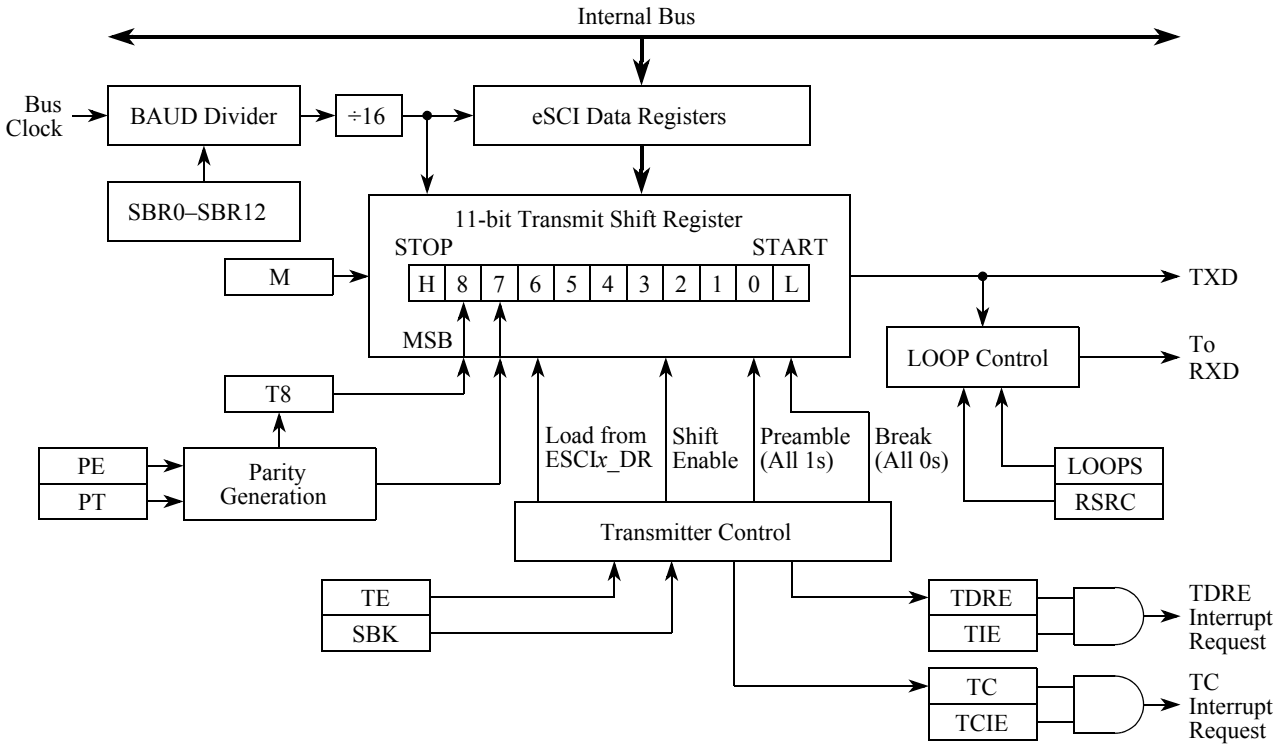


Figure 21-13. eSCI Transmitter Block Diagram

21.4.4.1 Transmitter Character Length

The eSCI transmitter can accommodate either 8-bit or 9-bit data characters. The state of the M bit in eSCI control register 1 (ESCI*x*_CR1) determines the length of data characters. When transmitting 9-bit data, bit T8 in the eSCI data register (ESCI*x*_DR) is the ninth bit (bit 8).

21.4.4.2 Character Transmission

To transmit data, the MCU writes the data bits to the eSCI data register (ESCI*x*_DR), which in turn are transferred to the transmit shift register. The transmit shift register then shifts a frame out through the TXD signal, after it has prefaced them with a start bit and appended them with a stop bit. The eSCI data register (ESCI*x*_DR) is the buffer (write-only during transmit) between the internal data bus and the transmit shift register.

The eSCI also sets a flag, the transmit data register empty flag (TDRE), every time it transfers data from the buffer (ESCI*x*_DR) to the transmit shift register. The transmit driver routine may respond to this flag by writing another byte to the transmitter buffer (ESCI*x*_DR), while the shift register is still shifting out the first byte.

To initiate an eSCI transmission:

1. Configure the eSCI:
   a) Turn on the module by clearing ESCI*x*_CR2[MDIS] if this bit is set.
   b) Select a baud rate. Write this value to the eSCI control register 1 (ESCI*x*_CR1) to start the baud rate generator. Remember that the baud rate generator is disabled when the ESCI*x*_CR1[SBR] field is zero. When using 8-bit writes, writes to the ESCI*x*_CR1[0–7] have no effect without also writing to ESCI*x*_CR1[8–15].
   c) Write to ESCIx_CR1 to configure word length, parity, and other configuration bits (LOOPS, RSRC, M, WAKE, ILT, PE, PT).
   d) Enable the transmitter, interrupts, receive, and wake-up as required, by writing to the ESCI*x*_CR1 register bits (TIE, TCIE, RIE, ILIE, TE, RE, RWU, SBK). A preamble or idle character will now be shifted out of the transmitter shift register.
2. Transmit procedure for each byte:
   a) Poll the TDRE flag by reading the ESCI*x*_SR or responding to the TDRE interrupt. Keep in mind that the TDRE bit resets to 1.
   b) If the TDRE flag is set, write the data to be transmitted to ESCI*x*_DR, where the ninth bit is written to the T8 bit in ESCI*x*_DR if the eSCI is in 9-bit data format. A new transmission will not result until the TDRE flag has been cleared.
3. Repeat step 2 for each subsequent transmission.

NOTE

The TDRE flag is set when the shift register is loaded with the next data to be transmitted from ESCI*x*_DR, which occurs approximately half-way through the stop bit of the previous frame. Specifically, this transfer occurs 9/16ths of a bit time AFTER the start of the stop bit of the previous frame.

Toggling the TE bit from 0 to 1 automatically loads the transmit shift register with a preamble of 10 logic 1s (if M = 0) or 11 logic 1s (if M = 1). After the preamble shifts out, control logic transfers the data from the eSCI data register into the transmit shift register. A logic 0 start bit automatically goes into the least

significant bit position of the transmit shift register. A logic 1 stop bit goes into the most significant bit position.

The eSCI hardware supports odd or even parity. When parity is enabled, the most significant bit (Msb) of the data character is the parity bit.

The transmit data register empty flag, TDRE, in the eSCI status register (ESCI*x*_SR) becomes set when the eSCI data register transfers a byte to the transmit shift register. The TDRE flag indicates that the eSCI data register can accept new data from the internal data bus. If the transmit interrupt enable bit, TIE, in eSCI control register 1 (ESCI*x*_CR1) is also set, the TDRE flag generates a transmitter interrupt request.

When the transmit shift register is not transmitting a frame, the TXD output goes to the idle condition, logic 1. If at any time software clears the TE bit in eSCI control register 1 (ESCI*x*_CR1), the transmitter enable signal goes low and the TXD output goes idle.

If software clears TE while a transmission is in progress (ESCI*x*_CR1[TC] = 0), the frame in the transmit shift register continues to shift out. To avoid accidentally cutting off the last frame in a message, always wait for TDRE to go high after the last frame before clearing TE.

To separate messages with preambles with minimum idle line time, use the following sequence between messages:

1. Write the last byte of the first message to ESCI*x*_DR.
2. Wait for the TDRE flag to go high, indicating the transfer of the last frame to the transmit shift register.
3. Queue a preamble by clearing and then setting the TE bit.
4. Write the first byte of the second message to ESCI*x*_DR.

21.4.4.3 Break Characters

Setting the break bit, SBK, in eSCI control register 1 (ESCI*x*_CR1) loads the transmit shift register with a break character. A break character contains all logic 0s and has no start, stop, or parity bit. Break character length depends on the M bit in the eSCI control register 1 (ESCI*x*_CR1) and on the BRK13 bit in the eSCI control register 2 (ESCI*x*_CR2). As long as SBK is set, the transmitter logic continuously loads break characters into the transmit shift register. After software clears the SBK bit, the shift register finishes transmitting the last break character and then transmits at least one logic 1. The automatic logic 1 at the end of a break character guarantees the recognition of the start bit of the next frame.

NOTE

LIN 2.0 now requires that a break character is always 13 bits long, so the BRK13 bit should always be set to 1. The eSCI will work with BRK13=0, but it will violate LIN 2.0.

The eSCI recognizes a break character when a start bit is followed by eight or nine logic 0 data bits and a logic 0 where the stop bit should be. Receiving a break character has the following effects on eSCI registers:

- Sets the framing error flag, FE.
- Sets the receive data register full flag, RDRF.
- Clears the eSCI data register (ESCI*x*_DR).
- May set the overrun flag, OR, noise flag, NF, parity error flag, PF, or the receiver active flag, RAF. For more details, see Section 21.3.3.4, “eSCI Status Register (ESCIx_SR).”

21.4.4.4 Idle Characters

An idle character contains all logic 1s and has no start, stop, or parity bit. Idle character length depends on the M bit in eSCI control register 1 (ESCI*x*_CR1). The preamble is a synchronizing idle character that begins the first transmission initiated after toggling the TE bit from 0 to 1.

If the TE bit is cleared during a transmission, the TXD output becomes idle after completion of the transmission in progress. Clearing and then setting the TE bit during a transmission queues an idle character to be sent after the frame currently being transmitted.

NOTE

When queueing an idle character, return the TE bit to logic 1 before the stop bit of the current frame shifts out through the TXD output. Setting the TE bit after the stop bit shifts out through the TXD output causes data previously written to the eSCI data register to be lost. Toggle the TE bit for a queued idle character while the TDRE flag is set and immediately before writing the next byte to the eSCI data register.

21.4.4.5 Fast Bit Error Detection in LIN Mode

Fast bit error detection has been designed to allow flagging of LIN bit errors while they occur, rather than flagging them after a byte transmission has completed. In order to use this feature, it is assumed a physical interface connects to the LIN bus as shown in Figure 21-14.

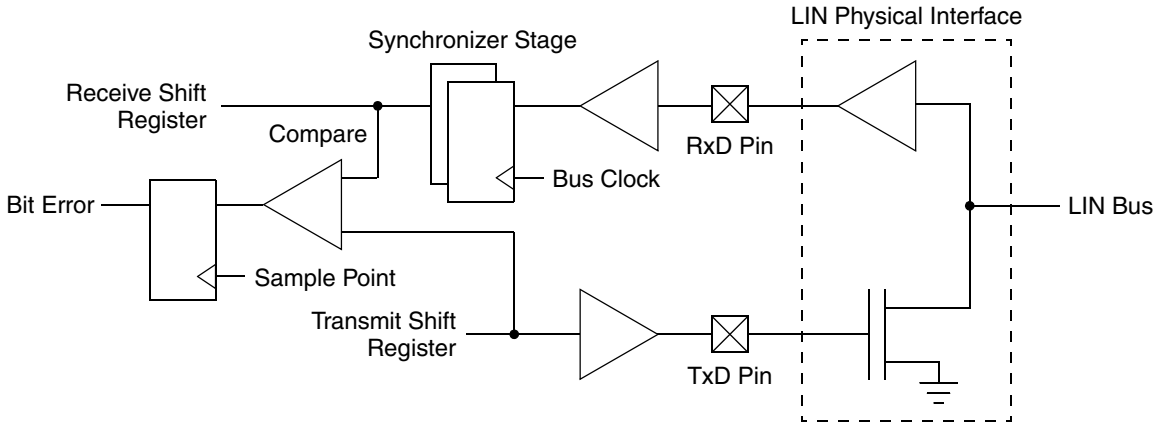


Figure 21-14. Fast Bit Error Detection on a LIN Bus

If fast bit error detection is enabled (FBR = 1), the eSCI will compare the transmitted and the received data stream when the transmitter is active (not idle). Once a mismatch between the transmitted data and the received data is detected the following actions are performed:

- The LIN frame is aborted (provided LDBG=0).
- The bit error flag BERR will be set.
- If SBSTP is 0, the remainder of the byte will be transmitted normally.
- If SBSTP is 1, the remaining bits in the byte after the error bit are transmitted as 1s (idle).

To adjust to different bus loads the sample point at which the incoming bit is compared to the one which was transmitted can be selected with the BESM13 bit (see Figure 21-15). If set, the comparison will be performed at RT clock 13, otherwise at RT clock 9 (also see Section 21.4.5.3, “Data Sampling.”).

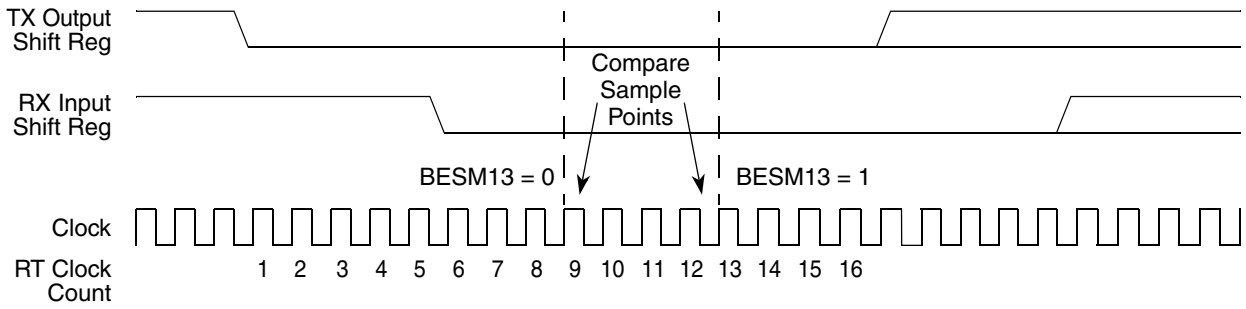


Figure 21-15. Fast Bit Error Detection Timing Diagram

21.4.5 Receiver

Figure 21-16 illustrates the eSCI receiver.

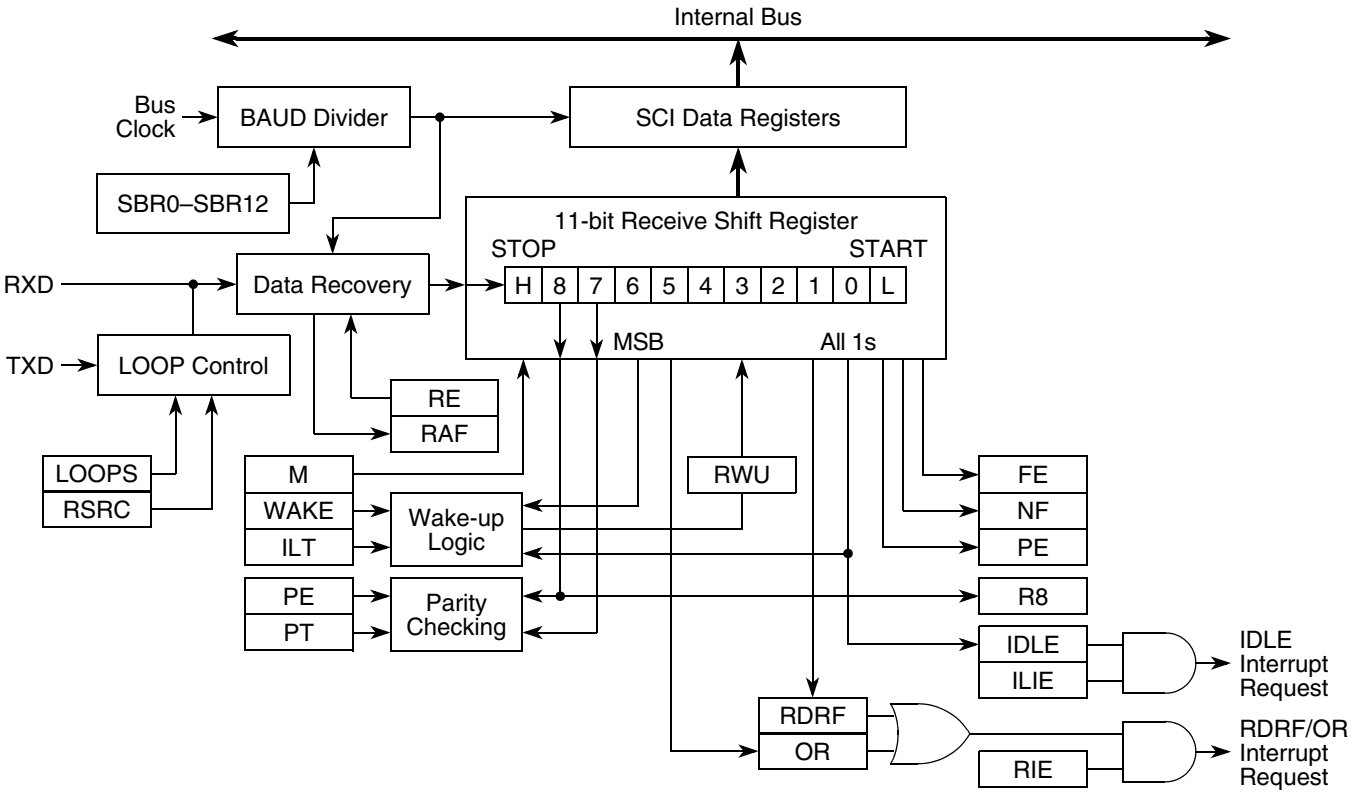


Figure 21-16. eSCI Receiver Block Diagram

21.4.5.1 Receiver Character Length

The eSCI receiver can accommodate either 8-bit or 9-bit data characters. The state of the M bit in eSCI control register 1 (ESCI*x*_CR1) determines the length of data characters. When receiving 9-bit data, bit R8 in the eSCI data register (ESCI*x*_DR) is the ninth bit (bit 8).

21.4.5.2 Character Reception

During an eSCI reception, the receive shift register shifts a frame in from the RXD input signal. The eSCI data register is the buffer (read-only during receive) between the internal data bus and the receive shift register.

After a complete frame shifts into the receive shift register, the data portion of the frame transfers to the eSCI data register. The receive data register full flag, RDRF, in eSCI status register (ESCI*x*_SR) is then set, indicating that the received byte can be read. If the receive interrupt enable bit, RIE, in eSCI control register 1 (ESCI*x*_CR1) is also set, the RDRF flag generates an RDRF interrupt request.

21.4.5.3 Data Sampling

The receiver uses a sampling clock to sample the RXD input signal at the 16 times the baud-rate frequency. This sampling clock is called the RT clock. To adjust for baud rate mismatch, the RT clock (see Figure 21-17) is re-synchronized:

- After every start bit.
- After the receiver detects a data bit change from logic 1 to logic 0. This data bit change is detected when a majority of data samples return a valid logic 1 and a majority of the next data samples return a valid logic 0. Data samples are taken at RT8, RT9, and RT10, as shown in Figure 21-17.

To locate the start bit, eSCI data recovery logic performs an asynchronous search for a logic 0 preceded by three logic 1s. When the falling edge of a possible start bit occurs, the RT clock begins to count to 16.

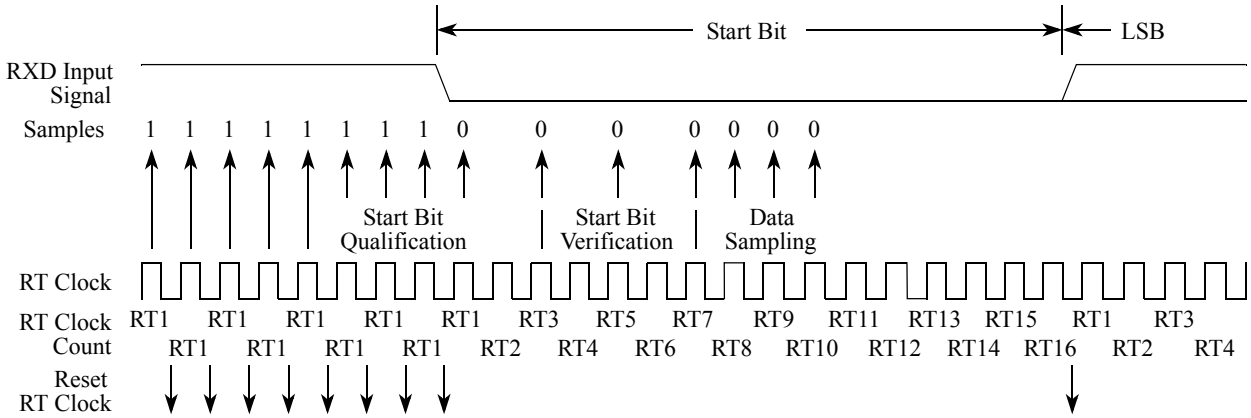


Figure 21-17. Receiver Data Sampling

To verify the start bit and to detect noise, the eSCI data recovery logic takes samples at RT3, RT5, and RT7. Table 21-18 summarizes the results of the start bit verification samples.

Table 21-18. Start Bit Verification

RT3, RT5, and RT7 Samples	Start Bit Verification	Noise Flag
000	Yes	0
001	Yes	1
010	Yes	1
011	No	0
100	Yes	1

Table 21-18. Start Bit Verification (continued)

RT3, RT5, and RT7 Samples	Start Bit Verification	Noise Flag
101	No	0
110	No	0
111	No	0

If start bit verification is not successful, the RT clock is reset and a new search for a start bit begins.

To determine the value of a data bit and to detect noise, eSCI recovery logic takes samples at RT8, RT9, and RT10. Table 21-19 summarizes the results of the data bit samples.

Table 21-19. Data Bit Recovery

RT8, RT9, and RT10 Samples	Data Bit Determination	Noise Flag
000	0	0
001	0	1
010	0	1
011	1	1
100	0	1
101	1	1
110	1	1
111	1	0

NOTE

The RT8, RT9, and RT10 samples do not affect start bit verification. If any or all of the RT8, RT9, and RT10 start bit samples are logic 1s following a successful start bit verification, the noise flag (NF) is set.

To verify a stop bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. Table 21-20 summarizes the results of the stop bit samples.

Table 21-20. Stop Bit Recovery

RT8, RT9, and RT10 Samples	Framing Error Flag	Noise Flag
000	1	0
001	1	1
010	1	1
011	0	1
100	1	1
101	0	1
110	0	1
111	0	0

In Figure 21-18 the verification samples RT3 and RT5 determine that the first low detected was noise and not the beginning of a start bit. The RT clock is reset and the start bit search begins again. The noise flag is not set because the noise occurred before the start bit was found.

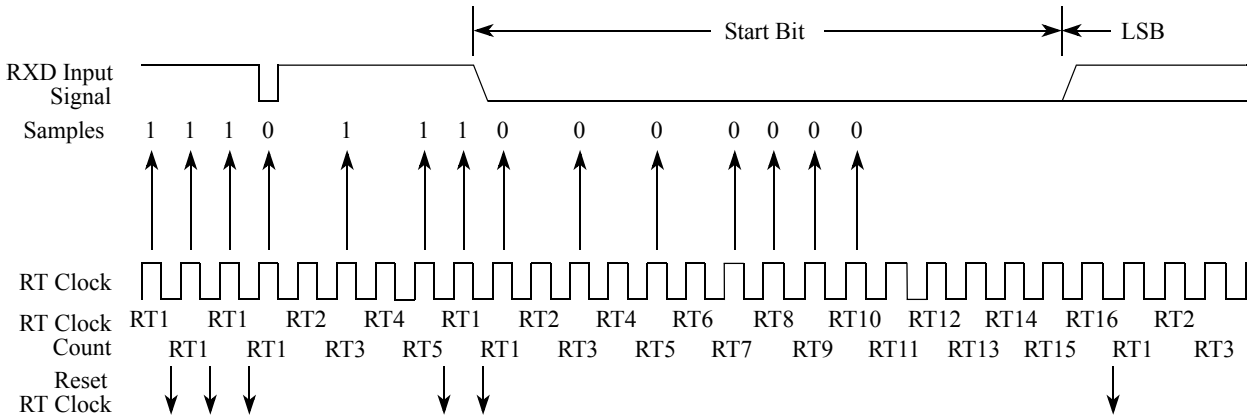


Figure 21-18. Start Bit Search Example

21.4.5.4 Framing Errors

If the data recovery logic sets the framing error flag, ESCI*x*_SR[FE], it does not detect a logic 1 where the stop bit should be in an incoming frame. A break character also sets the FE flag because a break character has no stop bit. The FE flag is set at the same time that the RDRF flag is set.

21.4.5.5 Baud Rate Tolerance

A transmitting device may be operating at a baud rate below or above the receiver baud rate. Accumulated bit time misalignment can cause one of the three stop bit data samples (RT8, RT9, and RT10) to fall outside the actual stop bit. A noise error will occur if the RT8, RT9, and RT10 samples are not all the same logical values. A framing error will occur if the receiver clock is misaligned in such a way that the majority of the RT8, RT9, and RT10 stop bit samples are a logic zero.

As the receiver samples an incoming frame and re-synchronizes the RT clock on any valid falling edge within the frame. Re-synchronization within frames will correct a misalignment between transmitter bit times and receiver bit times.

21.4.5.5.1 Slow Data Tolerance

Figure 21-19 shows how much a slow received frame can be misaligned without causing a noise error or a framing error. The slow stop bit begins at RT8 instead of RT1 but arrives in time for the stop bit data samples at RT8, RT9, and RT10.

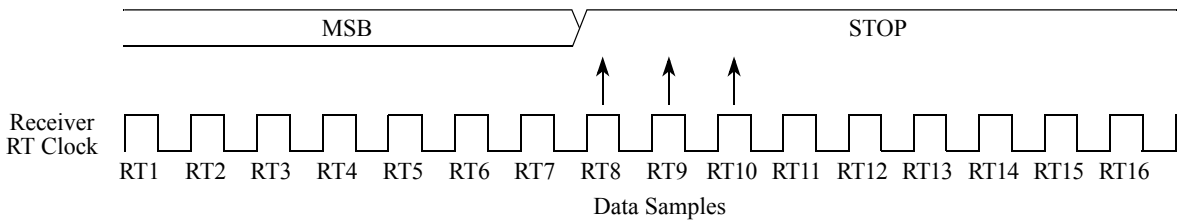


Figure 21-19. Slow Data

For an 8-bit data character, data sampling of the stop bit takes the receiver RT clock 151 clock cycles, as is shown below:

$$9 \text{ bit times} \times 16 \text{ RT cycles} + 7 \text{ RT cycles} = 151 \text{ RT cycles}$$

With the misaligned character shown in Figure 21-19, the receiver counts 151 RT cycles at the point when the count of the transmitting device is 9 bit times x 16 RT cycles = 147 RT cycles.

The maximum percent difference between the receiver count and the transmitter count of a slow 8-bit data character with no errors is 4.63%, as is shown below:

$$\frac{151 - 144}{151} \times 100 = 4.63\%$$

For a 9-bit data character, data sampling of the stop bit takes the receiver 167 RT cycles, as is shown below:

$$10 \text{ bit times} \times 16 \text{ RT cycles} + 7 \text{ RT cycles} = 167 \text{ RT cycles}$$

With the misaligned character shown in Figure 21-19, the receiver counts 167 RT cycles at the point when the count of the transmitting device is 10 bit times x 16 RT cycles cycles = 160 RT cycles.

The maximum percent difference between the receiver count and the transmitter count of a slow 9-bit character with no errors is 4.19%, as is shown below:

$$\frac{167 - 160}{167} \times 100 = 4.19\%$$

21.4.5.5.2 Fast Data Tolerance

Figure 21-20 shows how much a fast received frame can be misaligned. The fast stop bit ends at RT10 instead of RT16 but is still sampled at RT8, RT9, and RT10.

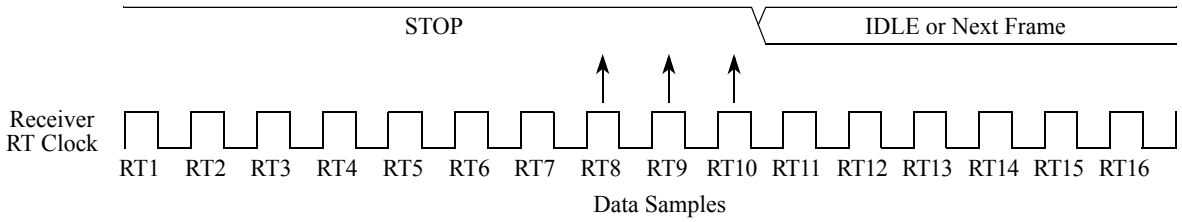


Figure 21-20. Fast Data

For an 8-bit data character, data sampling of the stop bit takes the receiver 154 RT cycles, as is shown below:

$$9 \text{ bit times} \times 16 \text{ RT cycles} + 10 \text{ RT cycles} = 154 \text{ RT cycles}$$

With the misaligned character shown in Figure 21-20, the receiver counts 154 RT cycles at the point when the count of the transmitting device is 10 bit times x 16 RT cycles = 160 RT cycles.

The maximum percent difference between the receiver count and the transmitter count of a fast 8-bit character with no errors is 3.40%, as is shown below:

$$\frac{160 - 154}{160} \times 100 = 3.40\%$$

For a 9-bit data character, data sampling of the stop bit takes the receiver 170 RT cycles, as shown below:

$$10 \text{ bit times} \times 16 \text{ RT cycles} + 10 \text{ RT cycles} = 170 \text{ RT cycles}$$

With the misaligned character shown in Figure 21-20, the receiver counts 170 RT cycles at the point when the count of the transmitting device is 11 bit times x 16 RT cycles = 176 RT cycles.

The maximum percent difference between the receiver count and the transmitter count of a fast 9-bit character with no errors is 3.40%, as is shown below:

$$\frac{176 - 170}{176} \times 100 = 3.40\%$$

21.4.5.6 Receiver Wake-up

The receiver can be put into a standby state, which enables the eSCI to ignore transmissions intended only for other receivers in multiple-receiver systems. Setting the receiver wake-up bit, ESCI*x*_CR1[RWU], in eSCI control register 1 (ESCI*x*_CR1) puts the receiver into standby state during which receiver interrupts are disabled. The eSCI will still load the received data into the ESCI*x*_DR, but it will not set the receive data register full, RDRF, flag.

The transmitting device can address messages to selected receivers by including addressing information (address bits) in the initial frame or frames of each message. See section Section 21.4.2, "Data Format," for an example of address bits.

The WAKE bit in eSCI control register 1 (ESCI*x*_CR1) determines how the eSCI is brought out of the standby state to process an incoming message. The WAKE bit enables either idle line wake-up or address mark wake-up.

21.4.5.6.1 Idle Input Line Wake-up (WAKE = 0)

Using the receiver idle input line wake-up method allows an idle condition on the RXD signal clears the ESCI*x*_CR1[RWU] bit and wakes up the eSCI. The initial frame or frames of every message contain addressing information. All receivers evaluate the addressing information, and receivers for which the message is addressed process the frames that follow. Any receiver for which a message is not addressed can set its RWU bit and return to the standby state. The RWU bit remains set and the receiver remains on standby until another idle character appears on the RXD signal.

Idle line wake-up requires that messages be separated by at least one idle character and that no message contains idle characters.

The idle character that wakes a receiver does not set the receiver idle bit, ESCI*x*_SR[IDLE], or the receive data register full flag, RDRF.

The idle line type bit, ESCI*x*_CR1[ILT], determines whether the receiver begins counting logic 1s as idle character bits after the start bit or after the stop bit.

21.4.5.6.2 Address Mark Wake-up (WAKE = 1)

Using the address mark wake-up method allows a logic 1 in the most significant bit (msb) position of a frame to clear the RWU bit and wake-up the eSCI. The logic 1 in the msb position marks a frame as an address frame that contains addressing information. All receivers evaluate the addressing information, and the receivers for which the message is addressed process the frames that follow. Any receiver for which a

message is not addressed can set its RWU bit and return to the standby state. The RWU bit remains set and the receiver remains on standby until another address frame appears on the RXD signal.

The logic 1 msb of an address frame clears the receiver's RWU bit before the stop bit is received and sets the RDRF flag.

Address mark wake-up allows messages to contain idle characters but requires that the msb be reserved for use in address frames.

NOTE

With the WAKE bit clear, setting the RWU bit after the RXD signal has been idle can cause the receiver to wake-up immediately.

21.4.6 Single-Wire Operation

Normally, the eSCI uses two pins for transmitting and receiving. In single-wire operation, the RXD pin is disconnected from the eSCI. The eSCI uses the TXD pin for both receiving and transmitting.

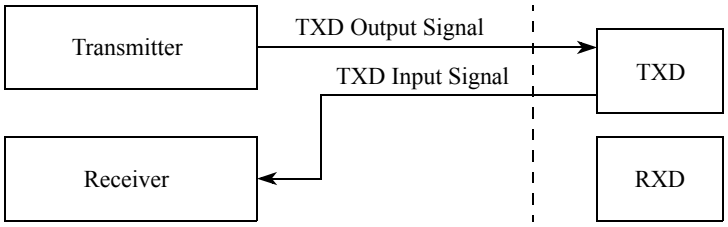


Figure 21-21. Single-Wire Operation (LOOPS = 1, RSRC = 1)

Enable single-wire operation by setting the LOOPS bit and the receiver source bit, RSRC, in eSCI control register 1 (ESCI*x*_CR1). Setting the LOOPS bit disables the path from the RXD signal to the receiver. Setting the RSRC bit connects the receiver input to the output of the TXD pin driver.

During reception, both the transmitter and receiver must be enabled (TE = 1 and RE = 1). The SIU_PCR89[PA] and SIU_PCR91[PA] bits must be set to select the TXD function for the relevant eSCI module, and the TXD pin should be set for open drain operation (SIU_PCR*nn*[ODE] = 1). Weak pull-up may optionally be enabled if the external transmitting device is also open drain. See Section 6.3.1.12, "Pad Configuration Registers (SIU_PCR)".

During transmission, the transmitter must be enabled (TE=1); the receiver may be enabled or disabled. If the receiver is enabled (RE=1), transmissions will be echoed back on the receiver. Set or clear open drain output enable depending on desired operation.

21.4.7 Loop Operation

In loop operation the transmitter output goes to the receiver input. The RXD signal is disconnected from the eSCI.

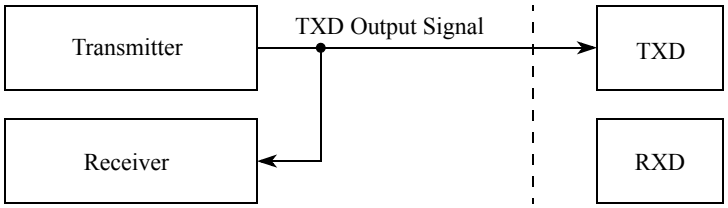


Figure 21-22. Loop Operation (LOOPS = 1, RSRC = 0)

Enable loop operation by setting the LOOPS bit and clearing the RSRC bit in eSCI control register 1 (ESCI*x*_CR1). Setting the LOOPS bit disables the path from the RXD signal to the receiver. Clearing the RSRC bit connects the transmitter output to the receiver input. Both the transmitter and receiver must be enabled (TE = 1 and RE = 1).

21.4.8 Modes of Operation

21.4.8.1 Run Mode

This is the normal mode of operation.

NOTE

The eSCI does not support a freeze mode. If the device is being operated in debug mode, the eSCI will continue to shift data if the e200z6 core asserts a freeze.

21.4.8.2 Disabling the eSCI

The module disable bit (ESCIx_CR2[MDIS]) in the eSCI control register 2 can be used to turn off the eSCI. This will save power by stopping the eSCI core from being clocked.By default the eSCI is enabled (ESCIx_CR2[MDIS]=0).

21.4.9 Interrupt Operation

21.4.9.1 Interrupt Sources

There are several interrupt sources that can generate an eSCI interrupt to the CPU. They are listed with details and descriptions in Table 21-21.

Table 21-21. eSCI Interrupt Flags, Sources, Mask Bits, and Descriptions

Interrupt Source	Flag	Source	Local Enable	Description
Transmitter	TDRE	ESCI*x*_SR[0]	TIE	Indicates that a byte was transferred from ESCIx_DR to the transmit shift register.
Transmitter	TC	ESCI*x*_SR[1]	TCIE	Indicates that a transmit is complete.
Receiver	RDRF	ESCI*x*_SR[2]	RIE	Indicates that received data is available in the eSCI data register.
Receiver	IDLE	ESCI*x*_SR[3]	ILIE	Indicates that receiver input has become idle.

Table 21-21. eSCI Interrupt Flags, Sources, Mask Bits, and Descriptions (continued)

Interrupt Source	Flag	Source	Local Enable	Description
Receiver	OR	ESCI*x*_SR[4]	ORIE	Indicates that an overrun condition has occurred.
Receiver	NF	ESCI*x*_SR[5]	NFIE	Detect noise error on receiver input.
Receiver	FE	ESCI*x*_SR[6]	FEIE	Framing error has occurred.
Receiver	PF	ESCI*x*_SR[7]	PFIE	Parity of received data does not match parity bit; parity error has occurred.
LIN	BERR	ESCI*x*_SR[11]	IEBERR	Detected a bit error, only valid in LIN mode.
LIN	RXRDY	ESCI*x*_SR[16]	RXIE	Indicates LIN hardware has received a data byte.
LIN	TXRDY	ESCI*x*_SR[17]	TXIE	Indicates LIN hardware can accept a control or data byte.
LIN	LWAKE	ESCI*x*_SR[18]	WUIE	A wake-up character has been received from a LIN frame.
LIN	STO	ESCI*x*_SR[19]	STIE	The response of the slave has been too slow (slave timeout).
LIN	PBERR	ESCI*x*_SR[20]	PBIE	Physical bus error detected.
LIN	CERR	ESCI*x*_SR[21]	CIE	CRC error detected.
LIN	CKERR	ESCI*x*_SR[22]	CKIE	Checksum error detected.
LIN	FRC	ESCI*x*_SR[23]	FCIE	LIN frame completed.
LIN	OVFL	ESCI*x*_SR[31]	OFIE	ESCIx_LRR overflow.

The eSCI only originates interrupt requests. The following sections describe how the eSCI generates a request and how the MCU acknowledges that request. The eSCI only has a single interrupt line (eSCI interrupt signal, active high operation) and all the following interrupts, when generated, are ORed together and issued through that port.

21.4.9.2 Interrupt Flags

21.4.9.2.1 TDRE Description

The transmit data register empty (TDRE) interrupt is set high by the eSCI when the transmit shift register receives data, 8 or 9 bits, from the eSCI data register, ESCI*x*_DR. A TDRE interrupt indicates that the transmit data register (ESCI*x*_DR) is empty and that a new data can be written to the ESCI*x*_DR for transmission. The TDRE bit is cleared by writing a one to the TDRE bit location in the ESCI*x*_SR.

21.4.9.2.2 TC Description

The transmit complete (TC) interrupt is set by the eSCI when a transmission has completed. A TC interrupt indicates that there is no transmission in progress. TC is set high when the TDRE flag is set and no data, preamble, or break character is being transmitted. When TC is set, the TXD pin becomes idle (logic 1). The TC bit is cleared by writing a one to the TC bit location in the ESCI*x*_SR.

21.4.9.2.3 RDRF Description

The receive data register full (RDRF) interrupt is set when the data in the receive shift register transfers to the eSCI data register. An RDRF interrupt indicates that the received data has been transferred to the eSCI data register and that the received data can now be read by the MCU. The RDRF bit is cleared by writing a one to the RDRF bit location in the ESCI*x*_SR.

21.4.9.2.4 OR Description

The overrun (OR) interrupt is set when software fails to read the eSCI data register before the receive shift register receives the next frame. The newly acquired data in the shift register is lost in this case, but the data already in the eSCI data registers is not affected.The OR bit is cleared by writing a one to the OR bit location in the ESCI*x*_SR.

21.4.9.2.5 IDLE Description

The idle line (IDLE) interrupt is set when 10 consecutive logic 1s (if M = 0) or 11 consecutive logic 1s (if M = 1) appear on the receiver input. Once the IDLE is cleared, a valid frame must again set the RDRF flag before an idle condition can set the IDLE flag. The IDLE bit is cleared by writing a one to the IDLE bit location in the ESCI*x*_SR.

21.4.9.2.6 PF Description

The interrupt is set when the parity of the received data is not correct. PF is cleared by writing it with 1.

21.4.9.2.7 FE Description

The interrupt is set when the stop bit is read as a 0; which violates the SCI protocol. FE is cleared by writing it with 1.

21.4.9.2.8 NF Description

The NF interrupt is set when the eSCI detects noise on the receiver input.

21.4.9.2.9 BERR Description

While the eSCI is in LIN mode, the bit error (BERR) flag is set when one or more bits in the last transmitted byte is not read back with the same value. The BERR flag is cleared by writing a 1 to the bit. A bit error will cause the LIN FSM to reset. The BERR flag is cleared by writing a 1 to the bit.

21.4.9.2.10 RXRDY Description

While in LIN mode, the receiver ready (RXRDY) flag is set when the eSCI receives a valid data byte in an RX frame. RXRDY will not be set for bytes which the receiver obtains by reading back the data which the LIN finite state machine (FSM) has sent out. The RXRDY flag is cleared by writing a 1 to the bit.

21.4.9.2.11 TXRDY Description

While in LIN mode, the transmitter ready (TXRDY) flag is set when the eSCI can accept a control or data byte. The TXRDY flag is cleared by writing a 1 to the bit.

21.4.9.2.12 LWAKE Description

The LIN wake-up (LWAKE) flag is set when the LIN hardware receives a wake-up character sent by one of the LIN slaves. This occurs only when the LIN bus is in sleep mode. The LWAKE flag is cleared by writing a 1 to the bit.

21.4.9.2.13 STO Description

The slave timeout (STO) flag is set during an RX frame when the LIN slave has not transmitted all requested data bytes before the specified timeout period. The STO flag is cleared by writing a 1 to the bit.

21.4.9.2.14 PBERR Description

If the RXD input remains stuck at a fixed value for 15 cycles after a transmission has started, the LIN hardware sets the physical bus error (PBERR) flag. The PBERR flag is cleared by writing a 1 to the bit.

21.4.9.2.15 CERR Description

If an RX frame has the CRC checking flag set and the two CRC bytes do not match the calculated CRC pattern, the CRC error (CERR) flag is set. The CERR flag is cleared by writing a 1 to the bit.

21.4.9.2.16 CKERR Description

If an RX frame has the checksum checking flag set and the last byte does not match the calculated checksum, the checksum error (CKERR) flag is set. The CKERR flag is cleared by writing a 1 to the bit.

21.4.9.2.17 FRC Description

The frame complete (FRC) flag is set after the last byte of a TX frame is sent out, or after the last byte of an RX frame is received. The FRC flag is cleared by writing a 1 to the bit.

NOTE

The last byte of a TX frame being sent or an RX frame being received indicates that the checksum comparison has taken place.

NOTE

The FRC flag is used to indicate to the CPU that the next frame can be set up. However, it should be noted that it might be set before the DMA controller has transferred the last byte from the eSCI to system memory. The FRC flag should not be used if the intention is to process data. Instead, the appropriate interrupt of the DMA controller should be used.

21.4.9.2.18 OVFL Description

The overflow (OVFL) flag is set when a byte is received in the ESCI*x*_LRR before the previous byte is read. Since the system is responsible for reading the register before the next byte arrives, this condition indicates a problem with CPU load. The OVFL flag is cleared by writing a 1 to the bit.

21.4.10 Using the LIN Hardware

The eSCI provides special support for the LIN protocol. It can be used to automate most tasks of a LIN master. In conjunction with the DMA interface it is possible to transmit entire frames (or sequences of

frames) and receive data from LIN slaves without any CPU intervention. There is no special support for LIN slave mode. If required, LIN slave mode may be implemented in software.

A LIN frame consists of a break character (10 or 13 bits), a sync field, an ID field, *n* data fields (*n* could be 0) and a checksum field. The data and checksum bytes are either provided by the LIN master (TX frame) or by the LIN slave (RX frame). The header fields will always be generated by the LIN master.

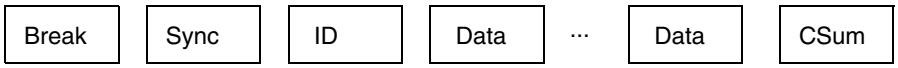


Figure 21-23. Typical LIN frame

The LIN hardware is highly configurable. This configurability allows the eSCI's LIN hardware to generate frames for LIN slaves from all revisions of the LIN standard. The settings are adjusted according to the capabilities of the slave device.

In order to activate the LIN hardware, the LIN mode bit in the ESCI*x*_LCR needs to be set. Other settings, such as double stop flags after bit errors and automatic parity bit generation, are also available for use in LIN mode.

The eSCI settings must be made according to the LIN specification. The eSCI must be configured for 2-wire operation (2 wires connected to the LIN transceiver) with 8 data bytes and no parity. Normally a 13-bit break is used, but the eSCI can also be configured for 10-bit breaks as required by the application.

21.4.10.1 Features of the LIN Hardware

The eSCI's LIN hardware has several features to support different revisions of the LIN slaves. The ESCI*x*_LTR can be configured to include or not include header bits in the checksum on a frame by frame basis. This feature supports LIN slaves with different LIN revisions. The LIN control register allows the user to decide whether the parity bits in the ID field should be calculated automatically and whether double stop flags should be inserted after a bit error. The BRK13 bit in ESCI*x*_CR2 decides whether to generate 10 or 13 bit break characters.

NOTE

LIN 2.0 now requires that a break character is always 13 bits long, so the BRK13 bit should always be set to 1. The eSCI will work with BRK13=0, but it will violate LIN 2.0.

The application software can decide to turn off the checksum generation/verification on a per frame basis and handle that function on its own. The application software can also decide to let the LIN hardware append two CRC bytes (Figure 21-24). The CRC bytes are not part of the LIN standard, but could be part of the application layer, that is they would be treated as data bytes by the LIN protocol. This can be useful when very long frames are transmitted. By default the CRC polynomial used is the same polynomial as for the CAN protocol.

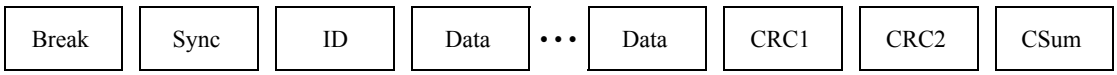


Figure 21-24. LIN Frame with CRC bytes

It is possible to force a resync of the LIN FSM, with the LRES bit in the LIN control register. However, under normal circumstances, the LIN hardware will automatically abort a frame after detecting a bit error.

21.4.10.2 Generating a TX Frame

The following procedure describes how a basic TX frame is generated.

The frame is controlled via the LIN transmit register (ESCI*x*_LTR). Initially, the application software will need to check the TXRDY bit (either using an interrupt, the TX DMA interface, or by polling the LIN status register). If TXRDY is set, the register is writable. Before each write, TXRDY must be checked (though this step is performed automatically in DMA mode). The first write to the ESCI*x*_LTR must contain the LIN ID field. The next write to ESCI*x*_LTR specifies the length of the frame (0 to 255 Bytes). The third write to ESCI*x*_LTR contains the control byte (frame direction, checksum/CRC settings). Note that timeout bits are not included in TX frames, since they only refer to LIN slaves. The three previously mentioned writes to the ESCI*x*_LTR specify the LIN frame data. Once the LIN frame data is specified, the eSCI LIN hardware starts to generate a LIN frame.

First, the eSCI transmits a break field. The sync field is transmitted next. The third field is the ID field. After these three fields have been broadcast, the ESCIx_LTR accepts data bytes; the LIN hardware transmits these data bytes as soon as they are available and can be sent out. After the last step the LIN hardware automatically appends the checksum field.

It is possible to set up a DMA channel to handle all the tasks required to send a TX frame. (See Figure 21-25 for more information.) For this operation, the TX DMA channel must be activated by setting the ESCI*x*_CR2[TXDMA] bit. The control information for the LIN frame (ID, message length, TX/RX type, timeout, etc.) and the data bytes are stored at an appropriate memory location. The DMA controller is then set up to transfer this block of memory to a location (the ESCI*x*_LTR). After transmission is complete, either the DMA controller or the LIN hardware can generate an interrupt to the CPU.

NOTE

In contrast to the standard software implementation where each byte transmission requires several interrupts, the DMA controller and eSCI handle communication, bit error and physical bus error checking, checksum, and CRC generation (checking on the RX side).

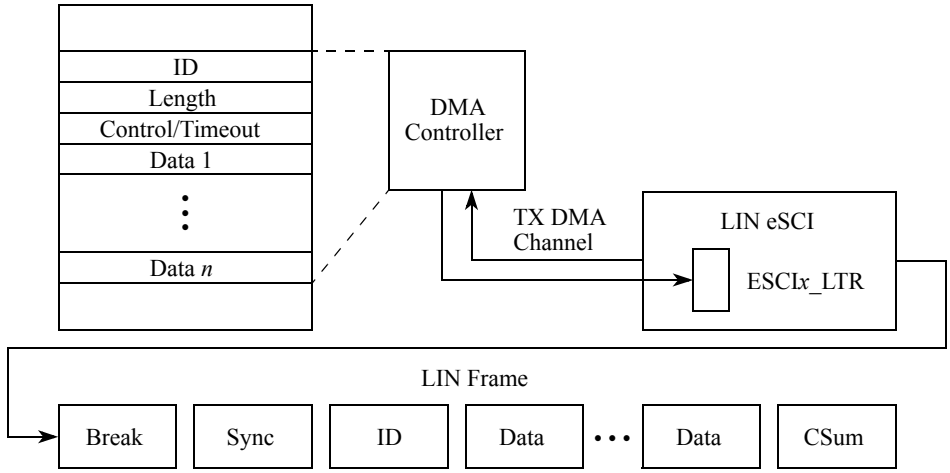


Figure 21-25. DMA Transfer of a TX frame

21.4.10.3 Generating an RX Frame

For RX frames the header information is provided by the LIN master. The data, CRC and checksum bytes (as enabled) are provided by the LIN slave. The LIN master verifies CRC and checksum bytes transmitted by the slave.

For a RX frame, control information must be written to the ESCI*x*_LTR in the same manner as for the TX frames. Additionally the timeout bits, which define the time to complete the entire frame, must be written. Then the ESCI*x*_SR[RXRDY] bit must be checked (either with an interrupt, RX DMA interface, or by polling) to detect incoming data bytes. The checksum byte normally does not appear in the ESCI*x*_LRR, instead the LIN hardware will verify the checksum and issue an interrupt, if the checksum value is not correct.

Two DMA channels can be used when executing a RX frame: one to transfer the header/control information from a memory location to the ESCI*x*_LTR, and one to transfer the incoming data bytes from the ESCI*x*_LRR to a table in memory. See Figure 21-26 for more information. After the last byte from the RX frame has been stored, the DMA controller can indicate completion to the CPU.

NOTE

It is also possible to setup a whole sequence of RX and TX frames, and generate a single event at the end of that sequence.

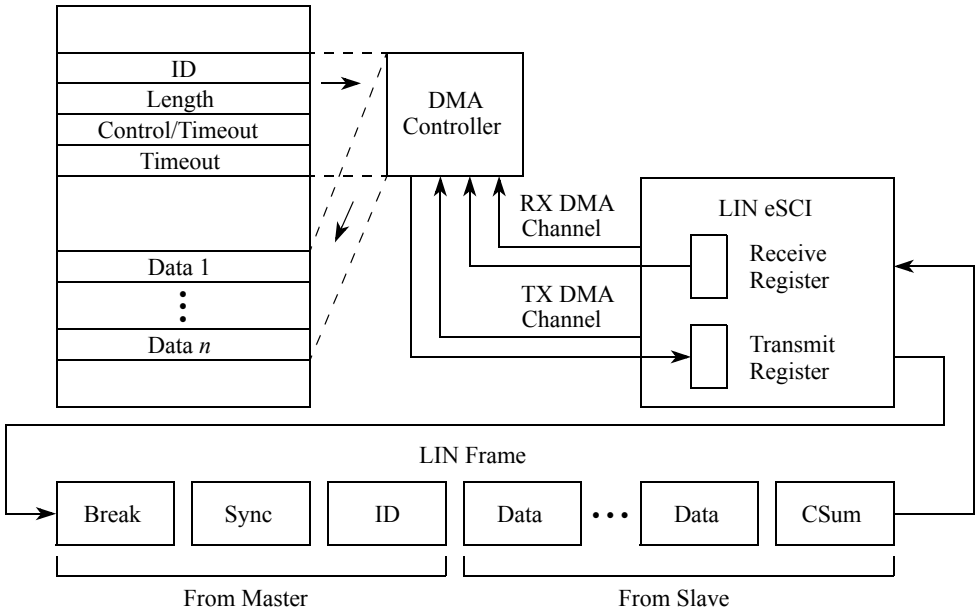


Figure 21-26. DMA Transfer of a RX frame

21.4.10.4 LIN Error Handling

The LIN hardware can detect several error conditions of the LIN protocol. LIN hardware will receive every byte that was transmitted, and compare it with the intended values. If there is a mismatch, a bit error is issued, and the LIN FSM will return to its start state.

For a RX frame the LIN hardware can detect a slave timeout error. The exact slave timeout error value can be set via the timeout bits in the ESCI*x*_LTR. If the frame is not complete within the number of clock cycles specified in the register, the LIN FSM will return to its start state, and the STO interrupt is issued.

The LIN protocol supports a sleep mode. After 25,000 bus cycles of inactivity the bus is assumed to be in sleep mode. Normally entering sleep mode can be avoided, if the LIN master is regularly creating some bus activity. Otherwise the timeout state needs to be detected by the application software, for example by setting a timer.

Both LIN masters and LIN slaves can cause the bus to exit sleep mode by sending a break signal. The LIN hardware will generate such a break, when WU bit in the LIN control register is written. After transmitting this break the LIN hardware will not send out data (that is not raise the TXRDY flag) before the wake-up delimiter period has expired. This period can be selected by setting the WUD bits in the LIN control register.

Break signals sent by a LIN slave are received by the LIN hardware, and so indicated by setting the WAKE flag in the LIN status register.

A physical bus error (LIN bus is permanently stuck at a fixed value) will set several error flags. If the input is permanently low, the eSCI will set the framing error (FE) flag in the eSCI status register. If the RXD input remains stuck at a fixed value for 15 cycles, after a transmission has started, the LIN hardware will set the PBERR flag in the LIN status register. In addition a bit error may be generated.

21.4.10.5 LIN Setup

Since the eSCI is for general-purpose use, some of the settings are not applicable for LIN operation. The following setup applies for most applications, regardless of which kind of LIN slave is addressed:

a) The module is enabled by writing the ESCI*x*_CR2[MDIS] bit to 0.
b) Both transmitter and receiver are enabled (ESCI*x*_CR1[TE] = 1, ESCI*x*_CR1[RE] = 1).
c) The data format bit ESCI*x*_CR1[M], is set to 0 (8 data bits), and the parity is disabled (PE = 0).
d) ESCI*x*_CR1[TIE], ESCI*x*_CR1[TCIE], ESCI*x*_CR1[RIE] interrupt enable bits should be inactive. Instead, the LIN interrupts should be used.
e) Switch eSCI to LIN mode (ESCI*x*_LCR[LIN] = 1).
f) The LIN standard requires that the break character always be 13 bits long (ESCI*x*_CR2[BRK13] = 1). The eSCI will work with BRK13=0, but it will violate LIN 2.0.
g) Normally, bit errors should cause the LIN FSM to reset, stop driving the bus immediately, and stop further DMA requests until the BERR flag has been cleared. Set ESCI*x*_LCR[LDBG] = 0, ESCI*x*_CR2[SBSTP] = 1, and ESCI*x*_CR2[BSTP] = 1 to accomplish these functions.
h) Fast bit error detection provides superior error checking, so ESCI*x*_CR2[FBR] should be set; normally it will be used with ESCI*x*_CR2[BESM13] = 1.
i) If available, a pulldown should be enabled on the RX input. (Thus if the transceiver fails, the RX pin will not float).
j) The error indicators NF, FE, BERR, STO, PBERR, CERR, CKERR, and OVFL should be enabled.
k) Initially a wake-up character may need to be transmitted on the LIN bus, so that the LIN slaves activate.

Other settings like baud rate, length of break character etc., depend on the LIN slaves to which the eSCI is connected.

21.5 Revision History

Substantive Changes since Rev 3.0
No changes.

Chapter 22
FlexCAN2 Controller Area Network

22.1 Introduction

The MPC5554 MCU contains three controller area network (FlexCAN2) modules; the MPC5553 contains two FlexCAN2 modules. Each FlexCAN2 module is a communication controller implementing the CAN protocol according to CAN Specification version 2.0B and ISO Standard 11898. Each FlexCAN2 module contains a 1024-byte embedded memory, capable of storing 64 message buffers (MBs). The respective functions are described in subsequent sections.

22.1.1 Block Diagram

A general block diagram is shown in Figure 22-1, which describes the main submodules implemented in the FlexCAN2 module, including an embedded RAM for up to 64 message buffers.

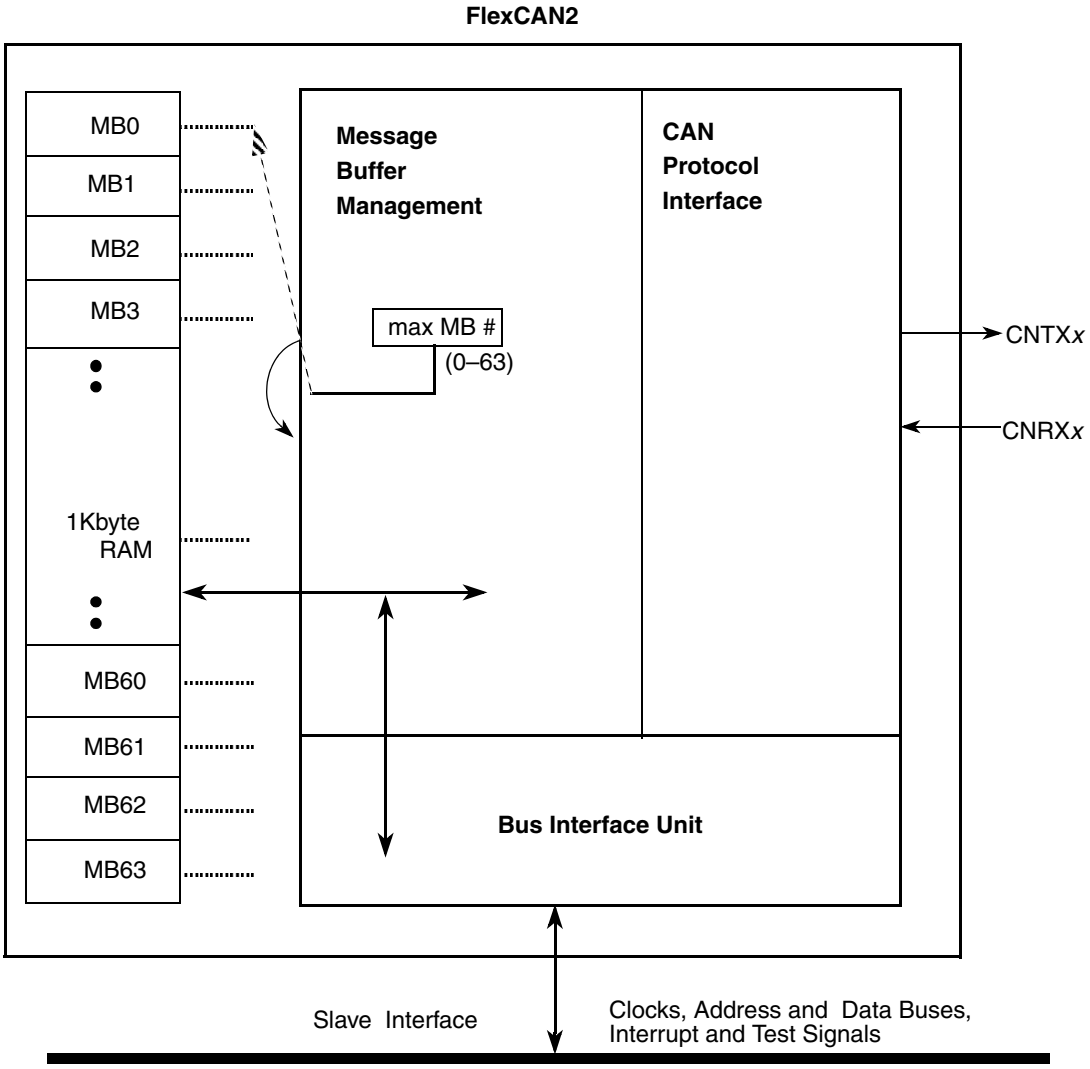


Figure 22-1. FlexCAN2 Block Diagram

22.1.2 Overview

The CAN protocol was designed primarily, but not exclusively, to be used as a vehicle serial data bus, meeting the specific requirements of this field: real-time processing, reliable operation in the EMI environment of a vehicle, cost-effectiveness and required bandwidth. The FlexCAN2 module is a full implementation of the CAN protocol specification, Version 2.0 B, which supports both standard and extended message frames. Sixty-four message buffers (MBs) are stored in an embedded 1024-byte RAM dedicated to the FlexCAN2 module.

The CAN protocol interface (CPI) manages the serial communication on the CAN bus, requesting RAM access for receiving and transmitting message frames, validating received messages and performing error handling. The message buffer management (MBM) handles message buffer selection for reception and transmission, taking care of arbitration and ID matching algorithms. The bus interface unit (BIU) controls the access to and from the internal interface bus, in order to establish connection to the CPU and to any other modules. Clocks, address and data buses, interrupt outputs and test signals are accessed through the bus interface unit.

22.1.3 Features

The FlexCAN2 module includes these distinctive features:

- Based on and includes all existing features of the Freescale TouCAN module
- Full implementation of the CAN protocol specification, version 2.0B
  - — Standard data and remote frames
  - — Extended data and remote frames
  - — Data length of 0 to 8 bytes
  - — Programmable bit rate up to 1 Mb/sec
- Content-related addressing
- 64 flexible message buffers of 0 to 8 bytes data length
- Each MB configurable as RX or TX, all supporting standard and extended messages
- Includes 1024 bytes of RAM used for MB storage
- Programmable clock source to the CAN protocol interface, either system clock or oscillator clock
- Listen-only mode capability
- Programmable loop-back mode supporting self-test operation
- Three programmable mask registers
- Programmable transmit-first scheme: lowest ID or lowest buffer number
- Time stamp based on 16-bit free-running timer
- Global network time, synchronized by a specific message
- Maskable interrupts
- Independent of the transmission medium (an external transceiver is assumed)
- Multi master concept
- High immunity to EMI
- Short latency time due to an arbitration scheme for high-priority messages

22.1.4 Modes of Operation

The MPC5553/MPC5554 supports four FlexCAN functional modes: normal, freeze, listen-only and loop-back. Just one of the low power modes—module disabled—is supported.

22.1.4.1 Normal Mode

In normal mode, the module operates receiving and/or transmitting message frames, errors are handled normally and all the CAN protocol functions are enabled. In the MPC5553/MPC5554, there is no distinction between user and supervisor modes.

22.1.4.2 Freeze Mode

Freeze mode is entered when the FRZ bit in the module configuration register (CAN*x*_MCR) is asserted while the HALT bit in the CAN*x*_MCR is set or debug mode is requested by the NPC. In freeze mode no transmission or reception of frames is done, and synchronization with the CAN bus is lost. See Section 22.4.6.1, "Freeze Mode," for more information.

22.1.4.3 Listen-Only Mode

The module enters this mode when the LOM bit in the CAN*x*_CR is asserted. In this mode, FlexCAN operates in a CAN error passive mode, freezing all error counters and receiving messages without sending acknowledgments.

22.1.4.4 Loop-Back Mode

The module enters this mode when the LPB bit in the CAN*x*_CR is asserted. In this mode, FlexCAN performs an internal loop back that can be used for self test operation. The bit stream output of the transmitter is internally fed back to the receiver input. The CAN receive input pin (CNRX*x*) is ignored and the transmit output (CNTX*x*) goes to the recessive state (logic 1). FlexCAN behaves as it normally does when transmitting, and treats its own transmitted message as a message received from a remote node. In this mode, FlexCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field to ensure proper reception of its own message. Both transmit and receive interrupts are generated.

22.1.4.5 Module Disabled Mode

This low power mode is entered when the MDIS bit in the CAN_MCR is asserted. When disabled, the module shuts down the clocks to the CAN protocol interface and message buffer management submodules. Exit from this mode is done by negating the CAN_MCR[MDIS] bit. See Section 22.4.6.2, "Module Disabled Mode," for more information.

22.2 External Signal Description

22.2.1 Overview

The FlexCAN2 module has two I/O signals connected to the external MCU pins. These signals are summarized in Table 22-1 and described in more detail in the next sub-sections.

Table 22-1. FlexCAN2 Signals

Signal Name[1]	Direction	Description
CNRX*x*	I	CAN receive
CNTX*x*	O	CAN transmit

[1] In the MPC5554, *x* indicates FlexCAN2 module A, B or C, whereas in the MPC5553, *x* indicates FlexCAN2 module A and C.

22.2.2 Detailed Signal Description

22.2.2.1 CNRX*x*

This pin is the receive pin to the CAN bus transceiver. The dominant state is represented by logic level 0. The recessive state is represented by logic level 1.

22.2.2.2 CNTX*x*

This pin is the transmit pin to the CAN bus transceiver. The dominant state is represented by logic level 0. The recessive state is represented by logic level 1.

22.3 Memory Map/Register Definition

This section describes the registers and data structures in the FlexCAN2 module. The addresses presented here are relative to the base address of the module.

The address space occupied by FlexCAN2 is continuous: 128 bytes for registers starting at the module base address, extra space for MB storage, and 1024 bytes for 64 MBs.

22.3.1 Memory Map

The complete memory map for a FlexCAN2 module with its 64 MBs is shown in Table 22-2. Except for the base addresses, the three (MPC5554) or two (MPC5553) FlexCAN2 modules have identical memory maps. Each individual register is identified by its complete name and the corresponding mnemonic.

Table 22-2. Module Memory Map

Address	Register Name	Register Description	Size (bits)
Base = 0xFFFC_0000 (FlexCAN A)[1] Base = 0xFFFC_4000 (FlexCAN B)[1] Base = 0xFFFC_8000 (FlexCAN C)[1]	CAN*x*_MCR	Module configuration register	32
Base + 0x0004	CAN*x*_CR	Control register	32
Base + 0x0008	CAN*x*_TIMER	Free running timer	32
Base + 0x000C	—	Reserved	—
Base + 0x0010	CAN*x*_RXGMASK	RX global mask	32
Base + 0x0014	CAN*x*_RX14MASK	RX buffer 14 mask	32
Base + 0x0018	CAN*x*_RX15MASK	RX buffer 15 mask	32
Base + 0x001C	CAN*x*_ECR	Error counter register	32
Base + 0x0020	CAN*x*_ESR	Error and status register	32
Base + 0x0024	CAN*x*_IMRH	Interrupt masks high register	32
Base + 0x0028	CAN*x*_IMRL	Interrupt masks low register	32
Base + 0x002C	CAN*x*_IFRH	Interrupt flags high register	32
Base + 0x0030	CAN*x*_IFRL	Interrupt flags low register	32
Base + 0x0034– Base + 0x005F	—	Reserved	—

Table 22-2. Module Memory Map (continued)

Base + 0x0060– Base + 0x007F	—	Reserved	—
Base + 0x0080– Base + 0x017F	MB0–MB15	Message buffers 0–15	128 bits per MB
Base + 0x0180– Base + 0x027F	MB16–MB31	Message buffers 16–31	128 bits per MB
Base + 0x0280– Base + 0x047F	MB32–MB63	Message buffers 32–63	128 bits per MB

[1] The MPC5554 has FlexCAN2 modules A, B, and C, whereas the MPC5553 only has FlexCAN2 modules A and C.

The FlexCAN2 module stores CAN messages for transmission and reception using a message buffer structure. Each individual MB is formed by 16 bytes mapped in memory as described in Table 22-3. The FlexCAN2 module can manage up to 64 message buffers. Table 22-3 shows a standard/extended message buffer (MB0) memory map, using 16 bytes (0x80–0x8F) total space.

Table 22-3. Message Buffer MB0 Memory Mapping

Address Offset	MB Field
0x80	Control and Status (C/S)
0x84	Identifier Field
0x88–0x8F	Data fields 0–7 (1 byte each)

NOTE

Reading the C/S word of a message buffer (the first word of each MB) will lock it, preventing it from receiving further messages until it is unlocked either by reading another MB or by reading the timer.

22.3.2 Message Buffer Structure

The message buffer structure used by the FlexCAN2 module is represented in Figure 22-2. Both extended and standard frames (29-bit identifier and 11-bit identifier, respectively) used in the CAN specification (version 2.0 Part B) are represented.

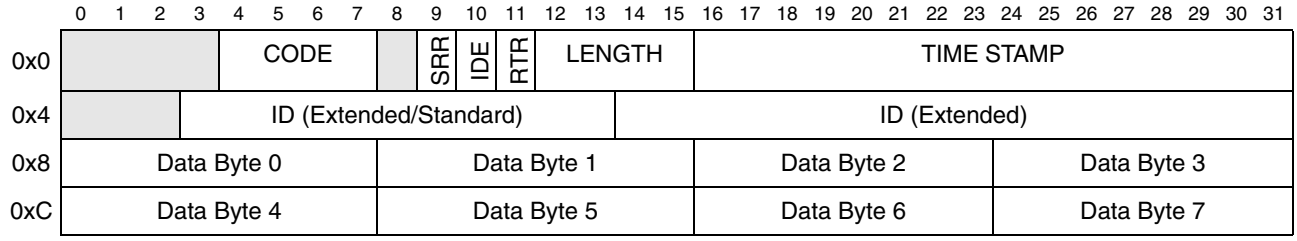


Figure 22-2. Message Buffer Structure

Table 22-4. Message Buffer Field Descriptions

Name	Description
CODE	Message buffer code. This 4-bit field can be accessed (read or write) by the CPU and by the FlexCAN2 module itself, as part of the message buffer matching and arbitration process. The encoding is shown in Table 22-5 and Table 22-6. See Section 22.4, "Functional Description," for additional information.
SRR	Substitute remote request. Fixed recessive bit, used only in extended format. It must be set to '1' by the user for transmission (TX Buffers) and will be stored with the value received on the CAN bus for RX receiving buffers. It can be received as either recessive or dominant. If FlexCAN2 receives this bit as dominant, then it is interpreted as arbitration loss. 0 Dominant is not a valid value for transmission in extended format frames 1 Recessive value is compulsory for transmission in extended format frames
IDE	ID extended bit. This bit identifies whether the frame format is standard or extended. 0 Frame format is standard 1 Frame format is extended
RTR	Remote transmission request. This bit is used for requesting transmissions of a data frame. If FlexCAN2 transmits this bit as '1' (recessive) and receives it as '0' (dominant), it is interpreted as arbitration loss. If this bit is transmitted as '0' (dominant), then if it is received as '1' (recessive), the FlexCAN2 module treats it as bit error. If the value received matches the value transmitted, it is considered as a successful bit transmission. 0 Indicates the current MB has a data frame to be transmitted 1 Indicates the current MB has a remote frame to be transmitted
LENGTH	Length of data in bytes. This 4-bit field is the length (in bytes) of the RX or TX data, which is located in offset 0x8 through 0xF of the MB space (see Figure 22-2). In reception, this field is written by the FlexCAN2 module, copied from the DLC (data length code) field of the received frame. In transmission, this field is written by the CPU and corresponds to the DLC field value of the frame to be transmitted. When RTR = 1, the Frame to be transmitted is a remote frame and does not include the data field, regardless of the length field.
TIME STAMP	Free-running counter time stamp. This 16-bit field is a copy of the free-running timer, captured for Tx and Rx frames at the time when the beginning of the Identifier field appears on the CAN bus.
ID	Frame identifier. In standard frame format, only the 11 most significant bits (28 to 18) are used for frame identification in both receive and transmit cases. The 18 least significant bits are ignored. In extended frame format, all bits are used for frame identification in both receive and transmit cases.
DATA	Data field. Up to eight bytes can be used for a data frame. For RX frames, the data is stored as it is received from the CAN bus. For TX frames, the CPU prepares the data field to be transmitted within the frame.

Table 22-5. Message Buffer Code for RX buffers

RX Code before RX New Frame	Description	RX Code after RX New Frame	Comment
0000	NOT ACTIVE: MB is not active.	—	MB does not participate in the matching process.
0100	EMPTY: MB is active and empty.	0010	MB participates in the matching process. When a frame is received successfully, the code is automatically updated to FULL.

Table 22-5. Message Buffer Code for RX buffers (continued)

RX Code before RX New Frame	Description	RX Code after RX New Frame	Comment
0010	FULL: MB is full.	0010	The act of reading the C/S word followed by unlocking the MB does not make the code return to EMPTY. It remains FULL. If a new frame is written to the MB after the C/S word was read and the MB was unlocked, the code still remains FULL.
		0110	If the MB is FULL and a new frame is overwritten to this MB before the CPU had time to read it, the code is automatically updated to OVERRUN. Refer to Section 22.4.3.1, “Matching Process for details about overrun behavior.
0110	OVERRUN: A frame was overwritten into a full buffer.	0010	If the code indicates OVERRUN but the CPU reads the C/S word and then unlocks the MB, when a new frame is written to the MB the code returns to FULL.
		0110	If the code already indicates OVERRUN, and yet another new frame must be written, the MB will be overwritten again, and the code will remain OVERRUN. Refer to Section 22.4.3.1, “Matching Process for details about overrun behavior.
0XY1[1]	BUSY: FlexCAN is updating the contents of the MB. The CPU must not access the MB.	0010	An EMPTY buffer was written with a new frame (XY was 01).
		0110	A FULL/OVERRUN buffer was overwritten (XY was 11).

[1] Note that for TX MBs (see Table 22-6), the BUSY bit should be ignored upon read.

Table 22-6. Message Buffer Code for TX buffers

RTR	Initial TX Code	Code after Successful Transmission	Description
X	1000	—	INACTIVE: MB does not participate in the arbitration process.
0	1100	1000	Transmit data frame unconditionally once. After transmission, the MB automatically returns to the INACTIVE state..
1	1100	0100	Transmit remote frame unconditionally once. After transmission, the MB automatically becomes and RX MB with the same ID.

Table 22-6. Message Buffer Code for TX buffers (continued)

RTR	Initial TX Code	Code after Successful Transmission	Description
0	1010	1010	Transmit a data frame whenever a remote request frame with the same ID is received. This MB participates simultaneously in both the matching and arbitration processes. The matching process compares the ID of the incoming remote request frame with the ID of the MB. If a match occurs this MB is allowed to participate in the current arbitration process and the CODE field is automatically updated to '1110' to allow the MB to participate in future arbitration runs. When the frame is eventually transmitted successfully, the Code automatically returns to '1010' to restart the process again.
0	1110	1010	the MBM as a result of match to a remote request frame. The data frame will be transmitted unconditionally once and then the code will automatically return to '1010'. The CPU can also write this code with the same effect.

22.3.3 Register Descriptions

The FlexCAN2 registers are described in this section. Note that there are three (or two in the MPC5553) separate, identical FlexCAN2 modules. Each register in the following sections is denoted with an '*x*' that represents the specified module, A, B, or C.

22.3.3.1 Module Configuration Register (CAN*x*_MCR)

CAN*x*_MCR defines global system configurations, such as the module operation mode and maximum message buffer configuration. Most of the fields in this register can be accessed at any time, except the MAXMB field, which should only be changed while the module is in freeze mode.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MDIS	FRZ	0	HALT	NOT RDY	0	SOFT RST	FRZ ACK	0	0	0	MDIS ACK	0	0	0	0
W																
Reset	0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0
Reg Addr	Base + 0x0000															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	MAXMB					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
Reg Addr	Base + 0x0000															

Figure 22-3. Module Configuration Register (CAN*x*_MCR)

Table 22-7. CAN*x*_MCR Field Descriptions

Bits	Name	Description
0	MDIS	Module disable. Controls whether FlexCAN2 is enabled or not. When disabled, FlexCAN2 shuts down the clock to the CAN protocol interface and message buffer management submodules. This is the only bit in CAN*x*_MCR not affected by soft reset. See Section 22.4.6.2, "Module Disabled Mode," for more information. 0 Enable the FlexCAN2 module 1 Disable the FlexCAN2 module
1	FRZ	Freeze enable. Specifies the FlexCAN2 behavior when the HALT bit in the CAN*x*_MCR is set or when debug mode is requested at MCU level. When FRZ is asserted, FlexCAN2 is enabled to enter freeze mode. Negation of this bit field causes FlexCAN2 to exit from freeze mode. 0 Not enabled to enter freeze mode 1 Enabled to enter freeze mode
2	—	Reserved.
3	HALT	Halt FlexCAN. Assertion of this bit puts the FlexCAN2 module into freeze mode if FRZ is asserted. The CPU should clear it after initializing the message buffers and CAN*x*_CR. If FRZ is set, no reception or transmission is performed by FlexCAN2 before this bit is cleared. While in freeze mode, the CPU has write access to the CAN*x*_ECR, that is otherwise read-only. Freeze mode cannot be entered while FlexCAN2 is disabled. See Section 22.4.6.1, "Freeze Mode," for more information. 0 No freeze mode request. 1 Enters freeze mode if the FRZ bit is asserted.
4	NOTRDY	FlexCAN2 not ready. Indicates that FlexCAN2 is either disabled or in freeze mode. It is negated once FlexCAN2 has exited these modes. 0 FlexCAN2 module is either in normal mode, listen-only mode or loop-back mode 1 FlexCAN2 module is either disabled or freeze mode
5	—	Reserved.

Table 22-7. CAN*x*_MCR Field Descriptions (continued)

Bits	Name	Description
6	SOFTRST	Soft reset. When asserted, FlexCAN2 resets its internal state machines and some of the memory-mapped registers. The following registers are affected by soft reset: • CAN*x*_MCR (except the MDIS bit), • CAN*x*_TIMER, • CAN*x*_ECR, • CAN*x*_ESR, • CAN*x*_IMRL, • CAN*x*_IMRH, • CAN*x*_IFRL, • CAN*x*_IFRH. Configuration registers that control the interface to the CAN bus are not affected by soft reset. The following registers are unaffected: • CANx_CR • CAN*x*_RXGMASK • CAN*x*_RX14MASK • CAN*x*_RX15MASK • all Message buffers The SOFTRST bit can be asserted directly by the CPU when it writes to the CAN*x*_MCR, but it is also asserted when global soft reset is requested at MCU level. Because soft reset is synchronous and has to follow a request/acknowledge procedure across clock domains, it may take some time to fully propagate its effect. The SOFTRST bit remains asserted while reset is pending, and is automatically negated when reset completes. Therefore, software can poll this bit to know when the soft reset has completed. 0 No reset request 1 Resets values in registers indicated above.
7	FRZACK	Freeze mode acknowledge. Indicates that FlexCAN2 is in freeze mode and its prescaler is stopped. The freeze mode request cannot be granted until current transmission and reception processes have finished. Therefore the software can poll the FRZACK bit to know when FlexCAN2 has actually entered freeze mode. If freeze mode request is negated, then this bit is negated once the FlexCAN2 prescaler is running again. If freeze mode is requested while FlexCAN2 is disabled, then the FRZACK bit will only be set when the low power mode is exited. See Section 22.4.6.1, “Freeze Mode,” for more information. 0 FlexCAN2 not in freeze mode, prescaler running 1 FlexCAN2 in freeze mode, prescaler stopped
8–10	—	Reserved.
11	MDISACK	Low power mode acknowledge. Indicates whether FlexCAN2 is disabled. This cannot be performed until all current transmission and reception processes have finished, so the CPU can poll the MDISACK bit to know when FlexCAN2 has actually been disabled. See Section 22.4.6.2, “Module Disabled Mode,” for more information. 0 FlexCAN2 not disabled 1 FlexCAN2 is disabled
12–25	—	Reserved.
26–31	MAXMB [0:5]	Maximum number of message buffers. This 6-bit field defines the maximum number of message buffers of the FlexCAN2 module. The reset value (0x0F) is equivalent to 16 MB configuration. This field should be changed only while the module is in freeze mode. Maximum MBs in use = MAXMB + 1 **Note:** MAXMB has to be programmed with a value smaller or equal to the number of available message buffers, otherwise FlexCAN2 will not transmit or receive frames.

22.3.3.2 Control Register (CAN*x*_CR)

CAN*x*_CR is defined for specific FlexCAN2 control features related to the CAN bus, such as bit-rate, programmable sampling point within an RX bit, loop-back mode, listen-only mode, bus off recovery behavior, and interrupt enabling (for example, bus-off, error). It also determines the division factor for the clock prescaler. Most of the fields in this register should only be changed while the module is disabled or in freeze mode. Exceptions are the BOFFMSK, ERRMSK, and BOFFREC bits, which can be accessed at any time. Note that CANx_CR is unaffected by soft reset (which occurs when CAN_MCR[SOFTRST] is asserted).

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PRESDIV								RJW		PSEG1			PSEG2		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0004															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BOFF MSK	ERR MSK	CLK_ SRC	LPB	0	0	0	0	SMP	BOFF REC	TSYN	LBUF	LOM	PROPSEG		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0004															

[1] CAN*x*_CR is unaffected by soft reset (which occurs when CAN_MCR[SOFTRST] is asserted).

Figure 22-4. Control Register (CAN*x*_CR)

Table 22-8. CAN*x*_CR Field Descriptions

Bits	Name	Description
0–7	PRESDIV [0:7]	Prescaler division factor. Defines the ratio between the CPI clock frequency and the serial clock (SCK) frequency. The SCK period defines the time quantum of the CAN protocol. For the reset value, the SCK frequency is equal to the CPI clock frequency. The maximum value of this register is 0xFF, that gives a minimum SCK frequency equal to the CPI clock frequency divided by 256. For more information, refer to Section 22.4.5.4, "Protocol Timing." $\text{S-clock frequency} = \frac{\text{CPI clock frequency}}{\text{PRESDIV} + 1}$
8–9	RJW [0:1]	Resync jump width. Defines the maximum number of time quanta[1] that a bit time can be changed by one re-synchronization. The valid programmable values are 0–3. $\text{Resync Jump Width} = \text{RJW} + 1$
10–12	PSEG1 [0:2]	Phase segment 1. Defines the length of phase buffer segment 1 in the bit time. The valid programmable values are 0–7. $\text{Phase Buffer Segment 1} = (\text{PSEG1} + 1) \times \text{Time Quanta}$

Table 22-8. CAN*x*_CR Field Descriptions

Bits	Name	Description
13–15	PSEG2 [0:2]	Phase segment 2. Defines the length of phase buffer segment 2 in the bit time. The valid programmable values are 1–7. Phase Buffer Segment 2 = (PSEG2 + 1) × Time Quanta
16	BOFFMSK	Bus off mask. Provides a mask for the bus off interrupt. 0 Bus off interrupt disabled 1 Bus off interrupt enabled
17	ERRMSK	Error mask. Provides a mask for the error interrupt. 0 Error interrupt disabled 1 Error interrupt enabled
18	CLK_SRC	CAN engine clock source. Selects the clock source to the CAN Protocol Interface (CPI) to be either the system clock (driven by the PLL) or the crystal oscillator clock. The selected clock is the one fed to the prescaler to generate the serial clock (SCK). In order to guarantee reliable operation, this bit should only be changed while the module is disabled. 0 = The CAN engine clock source is the oscillator clock 1 = The CAN engine clock source is the system clock
19	LPB	Loop back. Configures FlexCAN2 to operate in loop-back mode. See Section 22.1.4, “Modes of Operation” for information about this operating mode. 0 Loop back disabled 1 Loop back enabled
20–23	—	Reserved.
24	SMP	Sampling mode. Defines the sampling mode of each bit in the receiving messages (RX). 0 Just one sample is used to determine the RX bit value 1 Three samples are used to determine the value of the received bit: the regular one (sample point) and 2 preceding samples, a majority rule is used
25	BOFFREC	Bus off recovery mode. Defines how FlexCAN2 recovers from bus off state. If this bit is negated, automatic recovering from bus off state occurs according to the CAN Specification 2.0B. If the bit is asserted, automatic recovering from bus off is disabled and the module remains in bus off state until the bit is negated by the user. If the negation occurs before 128 sequences of 11 recessive bits are detected on the CAN bus, then bus off recovery happens as if the BOFFREC bit had never been asserted. If the negation occurs after 128 sequences of 11 recessive bits occurred, then FlexCAN2 will re-synchronize to the bus by waiting for 11 recessive bits before joining the bus. After negation, the BOFFREC bit can be re-asserted again during bus off, but it will only be effective the next time the module enters bus off. If BOFFREC was negated when the module entered bus off, asserting it during bus off will not be effective for the current bus off recovery. 0 Automatic recovering from bus off state enabled, according to CAN Spec 2.0 part B 1 Automatic recovering from bus off state disabled
26	TSYN	Timer sync mode. Enables a mechanism that resets the free-running timer each time a message is received in message buffer 0. This feature provides means to synchronize multiple FlexCAN2 stations with a special SYNC message (that is, global network time). 0 Timer sync feature disabled 1 Timer sync feature enabled **Note:** There is a possibility of 4–5 ticks count skew between the different FlexCAN2 stations that would operate in this mode.

Table 22-8. CAN*x*_CR Field Descriptions

Bits	Name	Description
27	LBUF	Lowest buffer transmitted first. This bit defines the ordering mechanism for message buffer transmission. 0 Buffer with lowest ID is transmitted first 1 Lowest number buffer is transmitted first
28	LOM	Listen-only mode. Configures FlexCAN2 to operate in listen-only mode. In this mode, the FlexCAN2 module receives messages without giving any acknowledge. It is not possible to transmit any message in this mode. 0 FlexCAN2 module is in normal active operation, listen only mode is deactivated 1 FlexCAN2 module is in listen only mode operation
29–31	PROPSEG [0:2]	Propagation segment. Defines the length of the propagation segment in the bit time. The valid programmable values are 0–7. Propagation Segment Time = (PROPSEG + 1) × Time Quanta Time Quantum = one S clock period

1 One time quantum is equal to the S clock period.

22.3.3.3 Free Running Timer (CAN*x*_TIMER)

CAN*x*_TIMER represents a 16-bit free running counter that can be read and written by the CPU. The timer starts from 0x0000 after Reset, counts linearly to 0xFFFF, and wraps around.

The timer is clocked by the FlexCAN2 bit-clock (which defines the baud rate on the CAN bus). During a message transmission/reception, it increments by one for each bit that is received or transmitted. When there is no message on the bus, it counts using the previously programmed baud rate. During freeze mode, the timer is not incremented.

The timer value is captured at the beginning of the identifier field of any frame on the CAN bus. This captured value is written into the TIME STAMP entry in a message buffer after a successful reception or transmission of a message.

Writing to the timer is an indirect operation. The data is first written to an auxiliary register and then an internal request/acknowledge procedure across clock domains is executed. All this is transparent to the user, except for the fact that the data will take some time to be actually written to the register. If desired, software can poll the register to discover when the data was actually written.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0008															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TIMER															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0008															

Figure 22-5. Free Running Timer (CAN*x*_TIMER)

22.3.3.4 RX Mask Registers

These registers are used as acceptance masks for received frame ID. Three masks are defined: a global mask, used for RX buffers 0–13 and 16–63, and two extra masks dedicated for buffers 14 and 15. The meaning of each mask bit is the following:

- Mask bit = 0: the corresponding incoming ID bit is “don’t care.”
- Mask bit = 1: the corresponding ID bit is checked against the incoming ID bit, to see if a match exists.

Note that these masks are used both for standard and extended ID formats. The value of mask registers should not be changed while in normal operation. Locked frames which had matched a MB through a mask may be transferred into the MB (upon release) but may no longer match. Table 22-9 shows some examples of ID masking for standard and extended message buffers.

Table 22-9. Mask Examples for Standard/Extended Message Buffers

	Base ID ID28.....ID18	IDE	Extended ID ID17.....ID0	Match
MB2 ID	1 1 1 1 1 1 1 1 0 0 0	0		
MB3 ID	1 1 1 1 1 1 1 1 0 0 0	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	
MB4 ID	0 0 0 0 0 0 1 1 1 1 1	0		
MB5 ID	0 0 0 0 0 0 1 1 1 0 1	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	
MB14 ID	1 1 1 1 1 1 1 1 0 0 0	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	
RX Global Mask	1 1 1 1 1 1 1 1 1 1 0		1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1	
RX Msg in[1]	1 1 1 1 1 1 1 1 0 0 1	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	3
RX Msg in[2]	1 1 1 1 1 1 1 1 0 0 1	0		2
RX Msg in[3]	1 1 1 1 1 1 1 1 0 0 1	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0	
RX Msg in[4]	0 1 1 1 1 1 1 1 0 0 0	0		
RX Msg in[5]	0 1 1 1 1 1 1 1 0 0 0	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	14
RX 14 Mask	0 1 1 1 1 1 1 1 1 1 1		1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0	

Table 22-9. Mask Examples for Standard/Extended Message Buffers (continued)

	Base ID ID28.....ID18	IDE	Extended ID ID17.....ID0	Match
RX Msg in[6]	1 0 1 1 1 1 1 1 0 0 0	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	
RX Msg in[7]	0 1 1 1 1 1 1 1 0 0 0	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	14

1 Match for Extended Format (MB3).
2 Match for Standard Format. (MB2).
3 Mismatch for MB3 because of ID0.
4 Mismatch for MB2 because of ID28.
5 Mismatch for MB3 because of ID28, Match for MB14 (uses RX14MASK).
6 Mismatch for MB14 because of ID27 (uses RX14MASK).
7 Match for MB14 (uses RX14MASK).

22.3.3.4.1 RX Global Mask (CAN*x*_RXGMASK)

The RX global mask bits are applied to all RX identifiers excluding RX buffers 14–15, that have their specific RX mask registers. Access to this register is unrestricted. Note that CANx_RXGMASK is unaffected by soft reset (which occurs when CAN_MCR[SOFTRST] is asserted).

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	MI28	MI27	MI26	MI25	MI24	MI23	MI22	MI21	MI20	MI19	MI18	MI17	MI16
W																
Reset[1]	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
Reg Addr	Base + 0x0010 (CANx_RXGMASK), Base + 0x0014 (CANx_RX14MASK), Base + 0x0018 (CANx_RX15MASK)															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MI15	MI14	MI13	MI12	MI11	MI10	MI9	MI8	MI7	MI6	MI5	MI4	MI3	MI2	MI1	MI0
W																
Reset[1]	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Reg Addr	Base + 0x0010 (CANx_RXGMASK), Base + 0x0014 (CANx_RX14MASK), Base + 0x0018 (CANx_RX15MASK)															

1 CAN*x*_RXGMASK is unaffected by soft reset (which occurs when CAN_MCR[SOFTRST] is asserted).

Figure 22-6. RX Global Mask Register (CAN*x*_RXGMASK)

Table 22-10. CAN*x*_RXGMASK Field Descriptions

Bits	Names	Description
0–2	—	Reserved, should be cleared.
3–13	MI*n*	Standard ID mask bits. These bits are the same mask bits for the standard and extended formats.
14–31	MI*n*	Extended ID mask bits. These bits are used to mask comparison only in extended format.

22.3.3.4.2 RX 14 Mask (CAN*x*_RX14MASK)

The CAN*x*_RX14MASK register has the same structure as the RX global mask register and is used to mask message buffer 14. Access to this register is unrestricted. Note that CANx_RX14MASK is unaffected by soft reset (which occurs when CAN_MCR[SOFTRST] is asserted).

- Address offset: 0x14
- Reset value: 0x1FFF_FFFF

22.3.3.4.3 RX 15 Mask (CAN*x*_RX15MASK)

The CAN*x*_RX15MASK register has the same structure as the RX global mask register and is used to mask message buffer 15. Access to this register is unrestricted. Note that CANx_RX15MASK is unaffected by soft reset (which occurs when CAN_MCR[SOFTRST] is asserted).

- Address offset: 0x18
- Reset value: 0x1FFF_FFFF

22.3.3.5 Error Counter Register (CAN*x*_ECR)

CAN*x*_ECR has two 8-bit fields reflecting the value of two FlexCAN2 error counters: the transmit error counter (TXECTR field) and receive error counter (RXECTR field). The rules for increasing and decreasing these counters are described in the CAN protocol and are completely implemented in the FlexCAN2 module. Both counters are read only except in freeze mode, where they can be written by the CPU.

Writing to the CAN*x*_ECR while in freeze mode is an indirect operation. The data is first written to an auxiliary register and then an internal request/acknowledge procedure across clock domains is executed. All this is transparent to the user, except for the fact that the data will take some time to be actually written to the register. If desired, software can poll the register to discover when the data was actually written.

FlexCAN2 responds to any bus state as described in the protocol: transmitting, for example, an ‘error active’ or ‘error passive’ flag, delaying its transmission start time (‘error passive’), and avoiding any influence on the bus when in the bus off state. The following are the basic rules for FlexCAN2 bus state transitions:

- If the value of TXECTR or RXECTR increases to be greater than or equal to 128, the FLTCONF field in the CAN*x*_ESR is updated to reflect the ‘error passive’ state.
- If the FlexCAN2 state is ‘error passive,’ and either TXECTR or RXECTR decrements to a value less than or equal to 127 while the other already satisfies this condition, the FLTCONF field in the CAN*x*_ESR is updated to reflect the ‘error active’ state.
- If the value of TXECTR increases to be greater than 255, the FLTCONF field in the CAN*x*_ESR is updated to reflect the bus off state, and an interrupt may be issued. The value of TXECTR is then reset to zero.
- If FlexCAN2 is in the bus off state, then TXECTR is cascaded together with another internal counter to count the 128th occurrences of 11 consecutive recessive bits on the bus. Hence, TXECTR is reset to zero and counts in a manner where the internal counter counts 11 such bits and then wraps around while incrementing the TXECTR. When TXECTR reaches the value of 128, the FLTCONF field in CAN*x*_ESR is updated to be ‘error active’ and both error counters are reset to zero. At any instance of dominant bit following a stream of less than 11 consecutive recessive bits, the internal counter resets itself to zero without affecting the TXECTR value.
- If during system start-up, only one node is operating, then its TXECTR increases in each message it is trying to transmit, as a result of acknowledge errors (indicated by the ACKERR bit in

CAN*x*_ESR). After the transition to the ‘error passive’ state, the TXECTR does not increment anymore by acknowledge errors. Therefore the device never goes to the bus off state.

- If the RXECTR increases to a value greater than 127, it is not incremented further, even if more errors are detected while being a receiver. At the next successful message reception, the counter is set to a value between 119 and 127 to resume to ‘error active’ state.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x001C															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXECTR								TXECTR							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x001C															

Figure 22-7. Error Counter Register (CAN*x*_ECR)

22.3.3.6 Error and Status Register (CAN*x*_ESR)

CAN*x*_ESR reflects various error conditions, some general status of the device, and it is the source of two interrupts to the CPU. The reported error conditions (bits 16–21) are those that occurred since the last time the CPU read this register. The CPU read action clears BIT1ERR, BIT0ERR, ACKERR, CRCERR, FRMERR, and STFERR. TXWRN, RXWRN, IDLE, TXRX, FLTCONF, BOFFINT, and ERRINT are status bits.

Most bits in this register are read-only, except BOFFINT and ERRINT, which are interrupt flags that can be cleared by writing 1 to them (writing 0 has no effect). See Section 22.4.7, “Interrupts,” for more details.

NOTE

A read clears BIT1ERR, BIT0ERR, ACKERR, CRCERR, FRMERR, and STFERR, therefore these bits must not be read speculatively. For future compatibility, the TLB entry covering the CAN*x*_ESR must be configured to be guarded.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0020															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BIT1 ERR	BIT0 ERR	ACK ERR	CRC ERR	FRM ERR	STF ERR	TX WRN	RX WRN	IDLE	TXRX	FLTCONF		0	BOFF INT	ERR INT	0
W														w1c	w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0020															

Figure 22-8. Error and Status Register (CAN*x*_ESR)

Table 22-11. CAN*x*_ESR Field Descriptions

Bits	Name	Description
0–15	—	Reserved.
16	BIT1ERR	Bit 1 error. Indicates when an inconsistency occurs between the transmitted and the received message in a bit. A read clears BIT1ERR. 0 No such occurrence 1 At least one bit sent as recessive is received as dominant **Note:** This bit is not set by a transmitter in case of arbitration field or ACK slot, or in case of a node sending a passive error flag that detects dominant bits.
17	BIT0ERR	Bit 0 error. Indicates when an inconsistency occurs between the transmitted and the received message in a bit. A read clears BIT0ERR. 0 No such occurrence 1 At least one bit sent as dominant is received as recessive
18	ACKERR	Acknowledge error. Indicates that an acknowledge error has been detected by the transmitter node; that is, a dominant bit has not been detected during the ACK SLOT. A read clears ACKERR. 0 No such occurrence 1 An ACK error occurred since last read of this register
19	CRCERR	Cyclic redundancy code error. Indicates that a CRC error has been detected by the receiver node; that is, the calculated CRC is different from the received. A read clears CRCERR. 0 No such occurrence 1 A CRC error occurred since last read of this register.
20	FRMERR	Form error. Indicates that a form error has been detected by the receiver node; that is, a fixed-form bit field contains at least one illegal bit. A read clears FRMERR. 0 No such occurrence 1 A form error occurred since last read of this register
21	STFERR	Stuffing error. Indicates that a stuffing error has been detected. A read clears STFERR. 0 No such occurrence. 1 A stuffing error occurred since last read of this register.

Table 22-11. CAN*x*_ESR Field Descriptions (continued)

Bits	Name	Description
22	TXWRN	TX error counter. This status bit indicates that repetitive errors are occurring during message transmission. 0 No such occurrence 1 TXECTR ≥ 96
23	RXWRN	RX error counter. This status bit indicates when repetitive errors are occurring during messages reception. 0 No such occurrence 1 RXECTR ≥ 96
24	IDLE	CAN bus IDLE state. This status bit indicates when CAN bus is in IDLE state. 0 No such occurrence 1 CAN bus is now IDLE
25	TXRX	Current FlexCAN2 status (transmitting/receiving). This status bit indicates if FlexCAN2 is transmitting or receiving a message when the CAN bus is not in IDLE state. This bit has no meaning when IDLE is asserted. 0 FlexCAN2 is receiving a message (IDLE = 0) 1 FlexCAN2 is transmitting a message (IDLE = 0)
26–27	FLTCONF [0:1]	Fault confinement state. This status bit indicates the confinement state of the FlexCAN2 module. If the LOM bit in the CAN*x*_CR is asserted, the FLTCONF field will indicate “Error Passive”. Since the CAN*x*_CR is not affected by soft reset, the FLTCONF field will not be affected by soft reset if the LOM bit is asserted. 00 Error active 01 Error passive 1X Bus off
28	—	Reserved.
29	BOFFINT	Bus off interrupt. This status bit is set when FlexCAN2 is in the bus off state. If CAN*x*_CR[BOFFMSK] is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. Writing 0 has no effect. 0 No such occurrence 1 FlexCAN2 module is in ‘Bus Off’ state
30	ERRINT	Error interrupt. This status bit indicates that at least one of the error bits (bits 16-21) is set. If CAN*x*_CR[ERRMSK] is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. Writing 0 has no effect. 0 No such occurrence 1 Indicates setting of any error bit in the CAN*x*_ESR
31	—	Reserved.

22.3.3.7 Interrupt Masks High Register (ICAN*x*_IMRH)

CAN*x*_IMRH allows any number of a range of 32 message buffer interrupts to be enabled or disabled. It contains one interrupt mask bit per buffer, enabling the CPU to determine which buffer generates an interrupt after a successful transmission or reception (that is, when the corresponding IFRH bit is set).

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	BUF 63M	BUF 62M	BUF 61M	BUF 60M	BUF 59M	BUF 58M	BUF 57M	BUF 56M	BUF 55M	BUF 54M	BUF 53M	BUF 52M	BUF 51M	BUF 50M	BUF 49M	BUF 48M
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0024															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	BUF 47M	BUF 46M	BUF 45M	BUF 44M	BUF 43M	BUF 42M	BUF 41M	BUF 40M	BUF 39M	BUF 38M	BUF 37M	BUF 36M	BUF 35M	BUF 34M	BUF 33M	BUF 32M
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0024															

Figure 22-9. Interrupt Masks High Register (CAN*x*_IMRH)

Table 22-12. CAN*x*_IMRH Field Descriptions

Bits	Name	Description
0–31	BUF*n*M	Message buffer *n* mask. Enables or disables the respective FlexCAN2 message buffer (MB63 to MB32) Interrupt. 0 The corresponding buffer Interrupt is disabled 1 The corresponding buffer Interrupt is enabled **Note:** Setting or clearing a bit in the IMRH register can assert or negate an interrupt request, respectively.

22.3.3.8 Interrupt Masks Low Register (CAN*x*_IMRL)

CAN*x*_IMRL allows enabling or disabling any number of a range of 32 message buffer interrupts. It contains one interrupt mask bit per buffer, enabling the CPU to determine which buffer generates an interrupt after a successful transmission or reception (that is, when the corresponding IFRL bit is set).

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	BUF 31M	BUF 30M	BUF 29M	BUF 28M	BUF 27M	BUF 26M	BUF 25M	BUF 24M	BUF 23M	BUF 22M	BUF 21M	BUF 20M	BUF 19M	BUF 18M	BUF 17M	BUF 16M
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0028															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	BUF 15M	BUF 14M	BUF 13M	BUF 12M	BUF 11M	BUF 10M	BUF 09M	BUF 08M	BUF 07M	BUF 06M	BUF 05M	BUF 04M	BUF 03M	BUF 02M	BUF 01M	BUF 00M
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0028															

Figure 22-10. Interrupt Mask Low Register (CAN*x*_IMRL)

Table 22-13. CAN*x*_IMRL Field Descriptions

Bits	Name	Description
0–31	BUF*n*M	Message buffer *n* mask. Enables or disables the respective FlexCAN2 message buffer (MB31 to MB0) Interrupt. 0 The corresponding buffer Interrupt is disabled 1 The corresponding buffer Interrupt is enabled **Note:** Setting or clearing a bit in the IMRL register can assert or negate an interrupt request, respectively.

22.3.3.9 Interrupt Flags High Register (CAN*x*_IFRH)

CAN*x*_IFRH defines the flags for 32 message buffer interrupts. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the corresponding IFRH bit. If the corresponding IMRH bit is set, an interrupt will be generated. The interrupt flag may be cleared by writing it to 1. Writing 0 has no effect.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BUF 63I	BUF 62I	BUF 61I	BUF 60I	BUF 59I	BUF 58I	BUF 57I	BUF 56I	BUF 55I	BUF 54I	BUF 53I	BUF 52I	BUF 51I	BUF 50I	BUF 49I	BUF 48I
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x002C															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BUF 47I	BUF 46I	BUF 45I	BUF 44I	BUF 43I	BUF 42I	BUF 41I	BUF 40I	BUF 39I	BUF 38I	BUF 37I	BUF 36I	BUF 35I	BUF 34I	BUF 33I	BUF 32I
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x002C															

Figure 22-11. Interrupt Flags High Register (CAN*x*_IFRH)

Table 22-14. CAN*x*_IFRH Field Descriptions

Bits	Name	Description
0–31	BUF*n*I	Message buffer *n* interrupt. Each bit represents the respective FlexCAN2 message buffer (MB63–MB32) interrupt. Write 1 to clear. 0 No such occurrence 1 The corresponding buffer has successfully completed transmission or reception.

22.3.3.10 Interrupt Flags Low Register (CAN*x*_IFRL)

CAN*x*_IFRL defines the flags for 32 message buffer interrupts. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the corresponding IFRL bit. If the corresponding IMRL bit is set, an interrupt will be generated. The interrupt flag may be cleared by writing it to 1. Writing 0 has no effect.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BUF 31I	BUF 30I	BUF 29I	BUF 28I	BUF 27I	BUF 26I	BUF 25	BUF 24I	BUF 23I	BUF 22I	BUF 21I	BUF 20I	BUF 19I	BUF 18I	BUF 17I	BUF 16I
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0030															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BUF 15I	BUF 14I	BUF 13I	BUF 12I	BUF 11I	BUF 10I	BUF 09I	BUF 08I	BUF 07I	BUF 06I	BUF 05I	BUF 04I	BUF 03I	BUF 02I	BUF 01I	BUF 00I
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x0030															

Figure 22-12. Interrupt Flags Low Register (CAN*x*_IFRL)

Table 22-15. CAN*x*_IFRL Field Descriptions

Bits	Name	Description
0–31	BUF*n*I	Message buffer *n* interrupt. Each bit represents the respective FlexCAN2 message buffer (MB31 to MB0) interrupt. Write 1 to clear. 0 No such occurrence 1 The corresponding buffer has successfully completed transmission or reception.

22.4 Functional Description

22.4.1 Overview

The FlexCAN2 module is a CAN protocol engine with a very flexible message buffer configuration scheme. The module can have up to 64 message buffers, any of which can be assigned as either a TX buffer or an RX buffer. Each message buffer has an assigned interrupt flag to indicate successful completion of transmission or reception.

22.4.2 Transmit Process

The CPU prepares a message buffer for transmission by executing the following steps:

- Write the CODE field of the control and status word to keep the TX MB inactive (code = 1000).
- Write the ID word.
- Write the DATA bytes.
- Write the LENGTH and CODE fields of the control and status word to activate the TX MB.

The first and last steps are mandatory.

22.4.2.1 Arbitration process

This process selects which will be the next MB to be transmitted. All MBs programmed as transmit buffers will be scanned to find the lowest ID[1] or the lowest MB number, depending on the LBUF bit in the CAN*x*_CR. The selected MB will be transferred to an internal serial message buffer (SMB), which is not user accessible, and then transmitted.

The arbitration process is triggered in the following events:

- During the CRC field of the CAN frame
- During the error delimiter field of the CAN frame
- During Intermission, if the winner MB defined in a previous arbitration was deactivated, or if there was no MB to transmit, but the CPU wrote to the C/S word of any MB after the previous arbitration finished
- When MBM is in idle or bus off state and the CPU writes to the C/S word of any MB
- Upon leaving freeze mode

When the arbitration is over, and there is a winner MB for transmission, the frame is transferred to the SMB for transmission. This is called ‘move out.’ After move out, the CAN transmit machine will start to transmit the frame according to the CAN protocol rules. FlexCAN2 transmits up to eight data bytes, even if the data length code (DLC) value is bigger.

At the end of a successful transmission, the value of the free running timer at the beginning of the identifier field is written into the TIME STAMP field in the MB, the CODE field in the control and status word of the MB is updated, a status flag is set in CAN*x*_IFRL or CAN*x*_IFRH, and an MB interrupt is generated if allowed by the corresponding interrupt mask register bit.

22.4.3 Receive Process

The CPU prepares a message buffer for frame reception by executing the following steps:

1. If LBUF is negated, the arbitration considers not only the ID, but also the RTR and IDE bits placed inside the ID at the same positions they are transmitted in the CAN frame.

- Write the CODE field of the control and status word to keep the RX MB inactive (CODE = 0000).
- Write the ID word.
- Write the CODE field of the control and status word to mark the MB as ‘receive active and empty.’

The first and last steps are mandatory.

22.4.3.1 Matching Process

After a MB is marked as ‘RX active and empty,’ it will participate in the internal matching process, which takes place every time the receiver receives an error free frame. In this process, all active RX buffers compare their ID value to the newly received one, and if a match occurs, the frame is transferred (move in) to the first (that is, lowest entry) matching MB. The value of the free running timer is written into the TIME STAMP field in the MB. The ID field, the DATA field (8 bytes at most), and the LENGTH field are stored, the CODE field is updated, and a status flag is set in CAN*x*_IFRL or CAN*x*_IFRH, and an interrupt is generated if the corresponding interrupt mask is enabled in CAN*x*_IMRL/H.

The CPU should read an RX frame from its MB in the following way:

- Control and status word (mandatory, activates internal lock for this buffer)
- ID (optional, needed only if a mask was used)
- DATA field words
- Free running timer (optional, releases internal lock)

Reading the free running timer is not mandatory. If not executed, the MB remains locked, unless the CPU starts reading another MB. Note that only a single MB is locked at a time. The only mandatory CPU read operation is of the control and status word, to assure data coherency. If the BUSY bit is set in the CODE field, then the CPU should defer the access to the MB until this bit is negated.

The CPU should synchronize to frame reception by the status flag bit for the specific MB in one of the CANx_IFRH and CANx_IFRL registers and not by the control and status word code field of that MB. Polling the CODE field does not work because once a frame was received and the CPU services the MB (by reading the C/S word followed by unlocking the MB), the CODE field will not return to EMPTY. It will remain FULL, as explained in Table 22-5. If the CPU tries to workaround this behavior by writing to the C/S word to force an EMPTY code after reading the MB, the MB is actually deactivated from any currently ongoing matching process. As a result, a newly received frame matching the ID of that MB may be lost. In summary: never do polling by reading directly the C/S word of the MBs. Instead, read the CANx_IFRH and CANx_IFRL registers.

Note that the received ID field is always stored in the matching MB, thus the contents of the ID field in a MB can change if the match was due to mask.

22.4.3.2 Self Received Frames

FlexCAN2 receives frames transmitted by itself if there exists an RX matching MB, but only if an ACK is generated by an external node or if loop-back mode is enabled.

22.4.4 Message Buffer Handling

In order to maintain data coherency and FlexCAN2 proper operation, the CPU must obey the rules described in Section 22.4.2, “Transmit Process,” and Section 22.4.3, “Receive Process.” Any form of CPU accessing a MB structure within FlexCAN2 other than those specified can cause FlexCAN2 to behave in an unpredictable way.

Deactivation of a message buffer is a CPU action that causes that MB to be excluded from FlexCAN2 transmit or receive processes during the current match/arbitration round. Any CPU write access to a control and status word of the MB structure deactivates that MB, excluding it from the current RX/TX process. However, deactivation is not permanent. The MB that was deactivated during the current match/arbitration round will be available for transmission or reception in the next round.

The purpose of deactivation is data coherency. The match/arbitration process scans the MBs to decide which MB to transmit or receive. If the CPU updates the MB in the middle of a match or arbitration process, the data of that MB may no longer be coherent, therefore that MB is deactivated.

Match and arbitration are one-pass processes. After scanning all MBs, a winner is determined. If MBs are changed after they are scanned, no re-evaluation is done to determine a new match/winner; and a frame may be lost because the winner may have been deactivated. If two RX MBs have a matching ID to a received frame, then it is not guaranteed reception if the user deactivated the matching MB after FlexCAN2 has scanned the second.

22.4.4.1 Notes on TX Message Buffer Deactivation

There is a point in time until which the deactivation of a TX MB causes it not to be transmitted (end of move out). After this point, it is transmitted but no interrupt is issued and the CODE field is not updated.

If a TX MB containing the lowest ID (or lowest buffer if LBUF is set) is deactivated after FlexCAN2 has scanned it while in arbitration process, then FlexCAN2 can transmit a MB with ID that may not be the lowest at the time.

22.4.4.2 Notes on RX Message Buffer Deactivation

If the deactivation occurs during move in, the move operation is aborted and no interrupt is issued, but the MB contains mixed data from two different frames.

In case the CPU writes data into RX MB data words while it is being moved in, the move operation is aborted and no interrupt will be issued, but the control/status word may be changed to reflect FULL or OVRN. This action should be avoided.

22.4.4.3 Data Coherency Mechanisms

The FlexCAN2 module has a mechanism to assure data coherency in both receive and transmit processes. The mechanism includes a lock status for MBs and two internal storage areas, called serial message buffers (SMB), to buffer frame transfers within FlexCAN. The details of the mechanism are the following:

- CPU reading a control and status word of an MB triggers a lock for that MB; that is, a new RX frame which matches this MB, cannot be written into this MB.
- In order to release a locked MB, the CPU should either lock another MB (by reading its control and status word), or globally release any locked MB (by reading the free-running timer).
- If while a MB is locked, an RX frame with a matching ID is received, then it cannot be stored within that MB and it remains in the SMB. There is no indication in the CAN*x*_ESR of that situation.
- If while a MB is locked, two or more RX frames with matching ID are received, then the last received one is kept within the SMB, while all preceding ones are lost. There is no indication that the preceding ones were lost in the CAN*x*_ESR.
- If a locked MB is released, and there exists a matching frame within the SMB, this frame is then transferred to the matching MB.

- If the CPU reads a RX MB while it is being transferred into (from) SMB, then the BUSY bit is set in the CODE field of the control and status word. In order to assure data coherency, the CPU should wait until this bit is negated before further reading from that MB. Note that in this case such MB is not locked.
- If the CPU deactivates a locked RX MB, then its lock status is negated, but no data is transferred into that MB.

22.4.5 CAN Protocol Related Features

22.4.5.1 Remote Frames

A remote frame is a special kind of frame. The user can program a MB to be a request remote frame by writing the MB as transmit with the RTR bit set to 1. After the remote request frame is transmitted successfully, the MB becomes a receive message buffer, with the same ID as before.

When a remote request frame is received by FlexCAN, its ID is compared to the IDs of the transmit message buffers with the CODE field ‘1010’. If there is a matching ID, then this MB frame will be transmitted. Note that if the matching MB has the RTR bit set, then FlexCAN2 will transmit a remote frame as a response.

A received remote request frame is not stored in a receive buffer. It is only used to trigger a transmission of a frame in response. The mask registers are not used in remote frame matching, and all ID bits (except RTR) of the incoming received frame should match.

In the case that a remote request frame was received and matched a MB, this message buffer immediately enters the internal arbitration process, but is considered as normal TX MB, with no higher priority. The data length of this frame is independent of the DLC field in the remote frame that initiated its transmission.

22.4.5.2 Overload Frames

FlexCAN2 will transmit overload frames due to detection of following conditions on CAN bus:

- Detection of a dominant bit in the first/second bit of intermission
- Detection of a dominant bit at the 7th bit (last) of end of frame field (RX frames)
- Detection of a dominant bit at the 8th bit (last) of error frame delimiter or overload frame delimiter

22.4.5.3 Time Stamp

The value of the free running timer is sampled at the beginning of the identifier field on the CAN bus, and is stored at the end of ‘move in’ in the TIME STAMP field, providing network behavior with respect to time.

Note that the free running timer can be reset upon a specific frame reception, enabling network time synchronization. Refer to TSYN description in Section 22.3.3.2, “Control Register (CANx_CR).”

22.4.5.4 Protocol Timing

The clock source to the CAN protocol interface (CPI) can be either the system clock or a direct feed from the oscillator pin EXTAL. The clock source is selected by the CLK_SRC bit in the CAN_CR. The clock is fed to the prescaler to generate the serial clock (SCK).

The FlexCAN2 module supports a variety of means to setup bit timing parameters that are required by the CAN protocol. The CAN*x*_CR has various fields used to control bit timing parameters: PRESDIV, PROPSEG, PSEG1, PSEG2 and RJW. See Section 22.3.3.2, "Control Register (CANx_CR)."

The PRESDIV field controls a prescaler that generates the serial clock (SCK), whose period defines the 'time quantum' used to compose the CAN waveform. A time quantum is the atomic unit of time handled by FlexCAN.

$$f_{Tq} = \frac{f_{CANCLK}}{\text{Prescaler Value}}$$

A bit time is subdivided into three segments[1] (reference Figure 22-13 and Table 22-16):

- SYNCSEG: This segment has a fixed length of one time quantum. Signal edges are expected to happen within this section.
- Time segment 1: This segment includes the propagation segment and the phase segment 1 of the CAN standard. It can be programmed by setting the PROPSEG and the PSEG1 fields of the CTRL register so that their sum (plus 2) is in the range of 4 to 16 time quanta.
- Time segment 2: This segment represents the phase segment 2 of the CAN standard. It can be programmed by setting the PSEG2 field of the CTRL register (plus 1) to be 2 to 8 time quanta long.

$$\text{Bit Rate} = \frac{f_{Tq}}{(\text{Number of Time Quanta})}$$

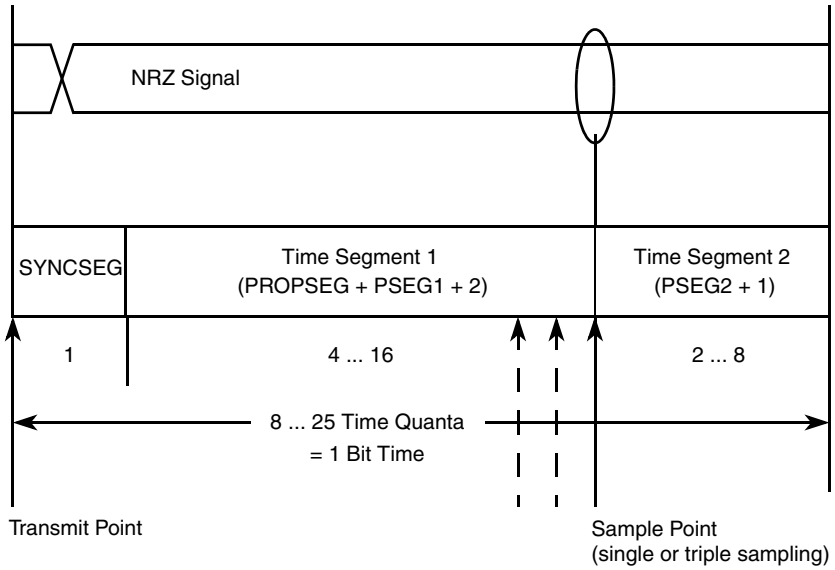


Figure 22-13. Segments within the Bit Time

Table 22-16. Time Segment Syntax

Syntax	Description
SYNCSEG	System expects transitions to occur on the bus during this period.

1. For further explanation of the underlying concepts please refer to ISO/DIS 11519–1, Section 10.3. Reference also the Bosch CAN 2.0A/B protocol specification dated September 1991 for bit timing.

Table 22-16. Time Segment Syntax

Transmit Point	A node in transmit mode transfers a new value to the CAN bus at this point.
Sample Point	A node in receive mode samples the bus at this point. If the three samples per bit option is selected, then this point marks the position of the third sample.

Table 22-17 gives an overview of the CAN compliant segment settings and the related parameter values.

NOTE

It is the user's responsibility to ensure the bit time settings are in compliance with the CAN standard.

Table 22-17. CAN Standard Compliant Bit Time Segment Settings

Time Segment 1	Time Segment 2	Resynchronization Jump Width
5 .. 10	2	1 .. 2
4 .. 11	3	1 .. 3
5 .. 12	4	1 .. 4
6 .. 13	5	1 .. 4
7 .. 14	6	1 .. 4
8 .. 15	7	1 .. 4
9 .. 16	8	1 .. 4

22.4.5.5 Arbitration and Matching Timing

During normal transmission or reception of frames, the arbitration, match, move in and move out processes are executed during certain time windows inside the CAN frame, as shown in Figure 22-14. When doing matching and arbitration, FlexCAN2 needs to span the whole message buffer memory during the available time slot. In order to have sufficient time to do that, the following restrictions must be observed:

- A valid CAN bit timing must be programmed, as indicated in Figure 22-14.
- The system clock frequency cannot be smaller than the oscillator clock frequency, i.e. the PLL cannot be programmed to divide down the oscillator clock.
- There must be a minimum ratio of 16 between the system clock frequency and the CAN bit rate.

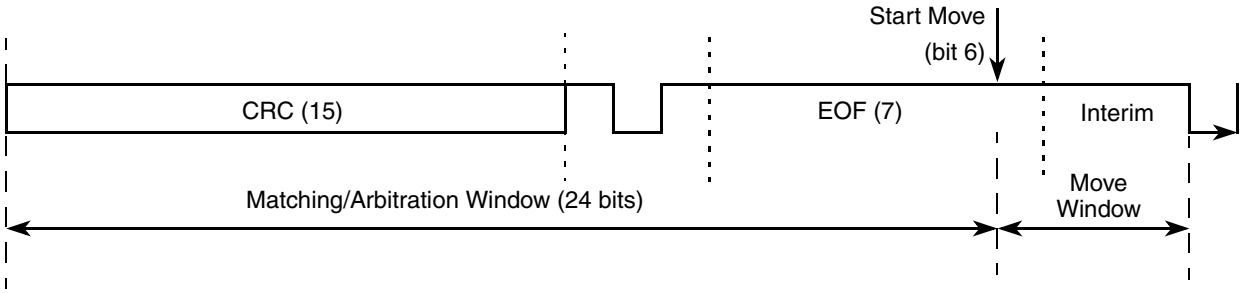


Figure 22-14. Arbitration, Match and Move Time Windows

22.4.6 Modes of Operation Details

22.4.6.1 Freeze Mode

This mode is entered by asserting the HALT bit in the CAN*x*_MCR or when the MCU is put into debug mode. In both cases it is also necessary that the FRZ bit is asserted in the CAN*x*_MCR. When freeze mode is requested during transmission or reception, FlexCAN2 does the following:

- Waits to be in either intermission, passive error, bus off or idle state
- Waits for all internal activities like move in or move out to finish
- Ignores the RX input pin and drives the TX pin as recessive
- Stops the prescaler, thus halting all CAN protocol activities
- Grants write access to the CAN*x*_ECR, which is read-only in other modes
- Sets the NOTRDY and FRZACK bits in CAN*x*_MCR

After requesting freeze mode, the user must wait for the FRZACK bit to be asserted in CAN*x*_MCR before executing any other action, otherwise FlexCAN2 can operate in an unpredictable way. In freeze mode, all memory mapped registers are accessible.

Exiting freeze mode is done in one of the following ways:

- CPU negates the FRZ bit in the CAN*x*_MCR.
- The MCU exits debug mode and/or the HALT bit is negated.

Once out of freeze mode, FlexCAN2 tries to re-synchronize to the CAN bus by waiting for 11 consecutive recessive bits.

22.4.6.2 Module Disabled Mode

This low power mode is entered when the CAN*x*_MCR[MDIS] bit is asserted. If the module is disabled during freeze mode, it shuts down the clocks to the CPI and MBM sub-modules, sets the CAN*x*_MCR[MDISACK] bit and negates the CAN*x*_MCR[FRZACK] bit. If the module is disabled during transmission or reception, FlexCAN2 does the following:

- Waits to be in either idle or bus off state, or else waits for the third bit of intermission and then checks it to be recessive
- Waits for all internal activities like move in or move out to finish
- Ignores its RX input pin and drives its TX pin as recessive
- Shuts down the clocks to the CPI and MBM sub-modules
- Sets the NOTRDY and MDISACK bits in CAN*x*_MCR

The bus interface unit continues to operate, enabling the CPU to access memory mapped registers except the free running timer, the CAN*x*_ECR and the message buffers, which cannot be accessed when the module is disabled. Exiting from this mode is done by negating the CAN*x*_MCR[MDIS] bit, which will resume the clocks and negate the CAN*x*_MCR[MDISACK] bit.

22.4.7 Interrupts

The module can generate interrupts from 20 interrupt sources (16 interrupts due to message buffers, two interrupts due to bus off and error conditions and two interrupts for the OR'd MB16–MB31 and MB32–63).

Each of the 64 message buffers can be an interrupt source, if its corresponding CAN*x*_IMRH or CAN*x*_IMRL bit is set. There is no distinction between TX and RX interrupts for a particular buffer, under

the assumption that the buffer is initialized for either transmission or reception. Each of the buffers has assigned a flag bit in the CAN*x*_IFRH or CAN*x*_IFRL registers. The bit is set when the corresponding buffer completes a successful transmission/reception and is cleared when the CPU writes it to 1.

A combined interrupt for each of two MB groups, MB16–MB31 and MB32–MB63, is also generated by an OR of all the interrupt sources from the associated MBs. This interrupt gets generated when any of the MBs generates an interrupt. In this case the CPU must read the CAN*x*_IFRH and CAN*x*_IFRL registers to determine which MB caused the interrupt.

The other two interrupt sources (bus off and error) generate interrupts like the MB interrupt sources, and can be read from CAN*x*_ESR. The bus off and error interrupt mask bits are located in the CAN*x*_CR.

22.4.8 Bus Interface

The CPU access to FlexCAN2 registers are subject to the following rules:

- Read and write access to unimplemented or reserved address space also results in access error. Any access to unimplemented MB locations results in access error.
- For a FlexCAN2 configuration that uses less than the total number of MBs and MAXMB is set accordingly, the remaining MB space can be used as general-purpose RAM space. Byte, word and long word accesses are allowed to the unused MB space.

22.5 Initialization/Application Information

This section provides instructions for initializing the FlexCAN2 module.

22.5.1 FlexCAN2 Initialization Sequence

The FlexCAN2 module can be reset in three ways:

- MCU-level hard reset, which resets all memory mapped registers asynchronously
- MCU-level soft reset, which resets some of the memory mapped registers synchronously (refer to Table 22-2 to see what registers are affected by soft reset)
- SOFTRST bit in CAN*x*_MCR, which has the same effect as the MCU level soft reset

Soft reset is synchronous and has to follow an internal request/acknowledge procedure across clock domains. Therefore, it may take some time to fully propagate its effects. The SOFTRST bit remains asserted while soft reset is pending, so software can poll this bit to know when the reset has completed.

After the module is enabled (CAN*x*_MCR[MDIS] bit negated), FlexCAN2 should be put into freeze mode before doing any configuration. In freeze mode, FlexCAN2 is un-synchronized to the CAN bus, the HALT and FRZ bits in CAN*x*_MCR are set, the internal state machines are disabled and the FRZACK and NOTRDY bits in the CAN*x*_MCR are set. The CNTX pin is in recessive state and FlexCAN2 does not initiate frame transmission nor receives any frames from the CAN bus. Note that the message buffer contents are not affected by reset, so they are not automatically initialized.

For any configuration change/initialization, it is required that FlexCAN2 is put into freeze mode (see Section 22.4.6.1, “Freeze Mode). The following is a generic initialization sequence applicable for the FlexCAN2 module:

- Initialize CANx_CR.
  - Determine bit timing parameters: PROPSEG, PSEG1, PSEG2, RJW.
  - Determine the bit rate by programming the PRESDIV field.
  - Determine internal arbitration mode (LBUF bit).

- Initialize message buffers.
  - The control and status word of all message buffers may be written either as active or inactive.
  - Other entries in each message buffer should be initialized as required.
- Initialize CANx_RXGMASK, CANx_RX14MASK, and CANx_RX15MASK registers for acceptance mask as needed.
- Set required mask bits in CAN*x*_IMRH and CAN*x*_IMRL registers (for all MBs interrupts), and in CAN*x*_CR (for bus off and error interrupts).
- Negate the CAN*x*_MCR[HALT] bit.

Starting with this last event, FlexCAN2 attempts to synchronize with the CAN bus.

22.5.2 FlexCAN2 Addressing and RAM Size

There are 1024 bytes of RAM for a maximum of 64 message buffers. The user can program the maximum number of message buffers (MBs) using the MAXMB field in the CAN*x*_MCR. For this 1024-byte RAM configuration, MAXMB can be any number from 0–63.

22.6 Revision History

Substantive Changes since Rev 3.0
No changes.

Chapter 23
Voltage Regulator Controller (VRC) and POR Module

23.1 Introduction

The voltage regulator controller (VRC) and POR module contains circuitry to control regulation of the external 1.5-V supply used by the device. It also contains power-on reset (POR) circuits for the 1.5-V supply, V_{DDSYN} and the VDDE supply that powers the $\overline{\text{RESET}}$ pad.

23.1.1 Block Diagram

The block diagram of the VRC and POR module is shown in Figure 23-1. The diagram represents the various submodules as implemented on the MPC5553/MPC5554.

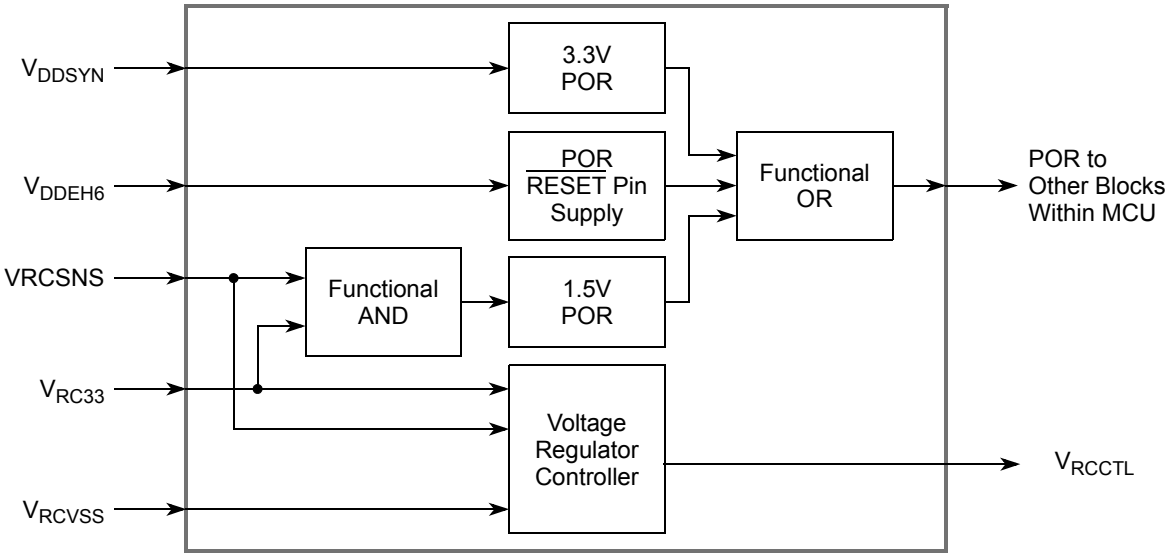


Figure 23-1. Voltage Regulator Controller and POR Blocks

23.2 External Signal Description

Table 23-1 provides an overview of VRC signals.

Table 23-1. Voltage Regulator Controller and POR Block External Signals

Signal	Type	Signal Level	Description
V_{RC33}	Supply pin	3.3V	Regulator supply input
V_{DDSYN}	Supply pin	3.3V	FMPLL supply input
V_{DDEH6}	Supply pin	3.3/5.0V	$\overline{\text{RESET}}$ pin supply input
V_{RCVSS}	Supply pin	0V	Regulator supply ground
VRCSNS	1.5-V Sense	1.5V	1.5-V Sense used by VRC. Pad connected to V_{DD} plane in package—not a package ball.
V_{RCCTL}	Current output	—	Regulator control output

23.2.1 Detailed Signal Description

The following paragraphs provide descriptions of signals coming into and going out of the VRC.

23.2.1.1 V_{RC33}

3.3V VRC supply input.

23.2.1.2 V_{DDSYN}

3.3V supply input for FMPLL.

23.2.1.3 V_{DDEH6}

Power supply input for padring segment that contains the $\overline{\text{RESET}}$ pad.

23.2.1.4 V_{RCVSS}

3.3V VRC ground supply.

23.2.1.5 VRCSNS

1.5V sense from external 1.5-V supply output of NPN transistor. This input is monitored by the VRC to determine current value for V_{RCCTL}. VRCSNS is a pad on the die that is connected to a V_{DD} plane inside the package. It is not a package ball.

23.2.1.6 V_{RCCTL}

The V_{RCCTL} sources base current to the external bypass transistor. The V_{RCCTL} signal is used with internal and external transistors to provide V_{DD}, which is the MCU's 1.5V power supply.

23.2.1.7 V_{DD}

Internal 1.5V supply input.

23.3 Memory Map/Register Definition

The VRC and POR module has no memory-mapped registers.

23.4 Functional Description

The VRC portion of the module contains a voltage regulator controller, and the POR portion contains circuits to monitor the voltage levels of the 1.5V and V_{DDSYN} supplies as well as circuits to monitor the supply that powers the $\overline{\text{RESET}}$ pad. The PORs indicate whether each monitored supply is above a specified voltage threshold. These PORs are used to ensure that the device is correctly powered up during a power-on reset. The MPC5553/MPC5554 resets the device if any of the supplies are below the specified minimum.

23.4.1 Voltage Regulator Controller

The VRC circuit provides a control current that can be used with an external NPN transistor and an external resistor to provide the 1.5V V_{DD} supply. The control current is output on the V_{RCCTL} pin. The voltage regulator controller begins to turn the pass transistor on slowly while the 3.3V POR still is asserted. The pass transistor will be completely turned on when the 3.3V POR negates.

NOTE

The voltage regulator controller will keep the 1.5V supply in regulation as long as V_{RC33} is in regulation. If more protection is desired, the customer may also supply an external 1.5V low voltage reset circuit.

If the on chip voltage regulator controller is not used, an external 1.5V power supply must be used. To avoid a power sequencing requirement when an external power supply is used, external 3.3V must power V_{RC33} while the V_{RCCTL} pad is unconnected. In this case the internal 1.5V POR will remain enabled. If the V_{RC33} is not powered, the device is subject to power sequencing requirements for the 1.5V and 3.3V or $\overline{\text{RESET}}$ power supplies (See Section 23.5.3, “Power Sequencing”). This is necessary to ensure that the 1.5V power supply is high enough for internal logic to operate properly during power-up.

23.4.2 POR Circuits

The individual POR circuits will negate whenever the supply they are monitoring is below a specified threshold. The entire device will be in power-on reset if any of these supplies are below the values specified in the *MPC5553/MPC5554 Microcontroller Hardware Specifications*.

Power-on reset will assert as soon as possible after the voltage level of the POR power supplies begins to rise. Each POR will negate *before* its power supply rises into its specified range. Each POR will assert *after* its power supply drops below its specified range. Power-on reset will remain asserted until all of the POR power supplies have dropped below the minimum POR threshold. The behavior for each POR during power sequencing is shown in Figure 23-2.

Before the 3.3V POR circuit asserts when ramping up or after it negates when ramping down, the device can exit POR but still be in system reset. In this case, MDO[0] will be driving high. Also in this case, though, no clocks will be toggling. If the 3.3V POR circuit is asserted, the device will behave as if in POR even if the 1.5V and $\overline{\text{RESET}}$ power POR circuits have not yet asserted when ramping up or have negated when ramping down.

NOTE

The PORs for each power supply are not intended to indicate that the power supply has dropped below the specified voltage range for the device. The user must monitor the power supplies externally and assert $\overline{\text{RESET}}$ to provide this precision of monitoring.

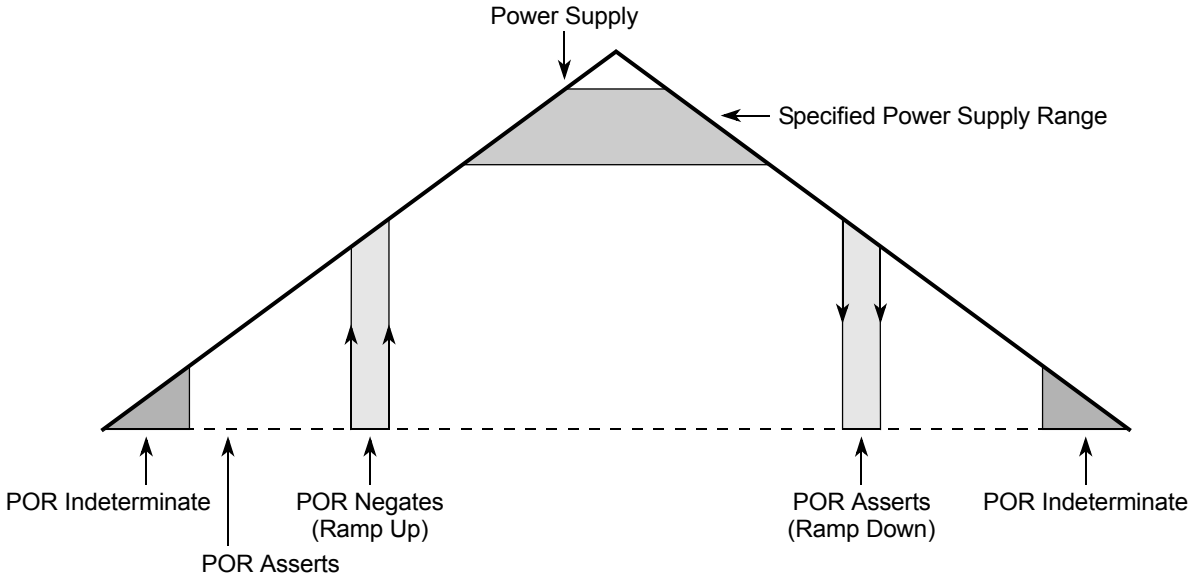


Figure 23-2. Regions POR is Asserted

23.4.2.1 1.5V POR Circuit

The 1.5V POR circuit monitors the voltage on the VRCSNS pad. The 1.5V POR will function if the V_{RC33} pad is powered. If the user does not power V_{RC33} to the specified voltage, the 1.5V POR will be disabled and the user must follow the specified power sequence.

23.4.2.2 3.3V POR Circuit

The 3.3V POR circuit is used to ensure that V_{DDSYN} is high enough that the FMPLL will begin to operate properly.

23.4.2.3 $\overline{\text{RESET}}$ Power POR Circuit

The $\overline{\text{RESET}}$ power POR circuit, which monitors the power supply that is powering the $\overline{\text{RESET}}$ pin, is used to ensure that the supply that powers the $\overline{\text{RESET}}$ pin is high enough that the state of the input will propagate reliably. The power supply monitored by this POR can go as high as 5.5V.

23.5 Initialization/Application Information

23.5.1 Voltage Regulator Example

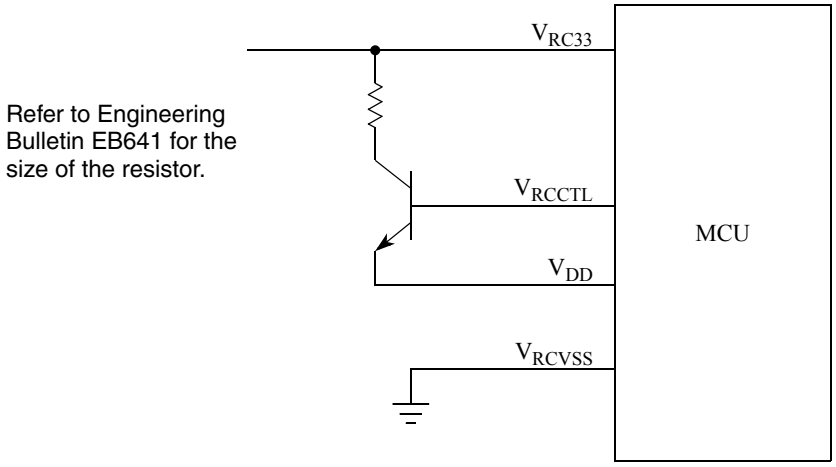


Figure 23-3. Voltage Regulator Controller Hookup

NOTE

The figure above should not be used as an application board design reference. See Engineering Bulletin EB641: Power Supplies on the MPC5500.

23.5.2 Recommended Power Transistors

Freescale recommends the use of the following NPN transistors with the on-chip Voltage Regulator Controller: ON Semiconductor™ BCP68T1 and Phillips Semiconductor™ BCP68.

Refer to the *MPC5553/MPC5554 Microcontroller Hardware Specifications* for information on recommended operating characteristics.

23.5.3 Power Sequencing

Power sequencing between the 1.5V power supply and V_{DDSYN} or the $\overline{\text{RESET}}$ power supplies is required if the user provides an external 1.5V power supply and ties V_{RC33} to ground. To avoid this power sequencing requirement, the user should power up V_{RC33} within the specified operating range, even if not using the on chip voltage regulator controller. Refer to Section 23.5.3.1, "Power-Up Sequence If V_{RC33} Grounded" and Section 23.5.3.2, "Power-Down Sequence If V_{RC33} Grounded."

Another power sequencing requirement is that V_{DD33} must be of sufficient voltage before POR negates so that the values on certain pins are treated as 1s when POR does negate. Refer to Section 23.5.3.3, "Input Value of Pins During POR Dependent on V_{DD33}."

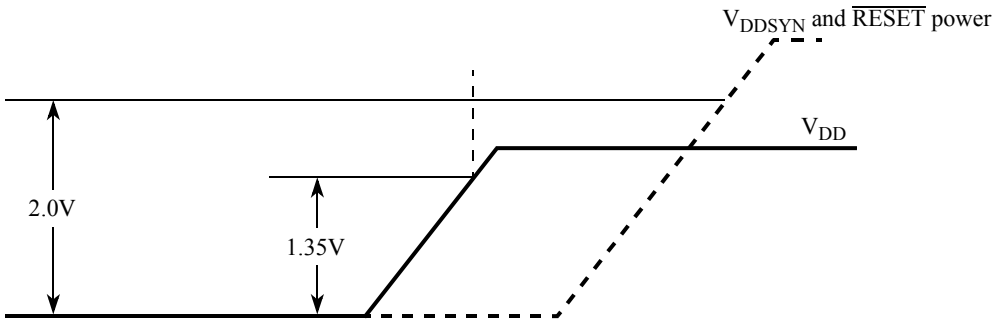
Although there is no power sequencing required between V_{RC33} and V_{DDSYN}, during power up, in order for the VRC staged turn-on to operate within specification, V_{RC33} must not lead V_{DDSYN} by more than 600 mV or lag by more than 100 mV. Higher spikes in the emitter current of the pass transistor will occur if V_{RC33} leads or lags V_{DDSYN} by more than those amounts. The value of that higher spike in current depends on the board power supply circuitry and the amount of board level capacitance.

When powering down, V_{RC33} and V_{DDSYN} do not have a delta requirement to each other because the bypass capacitors internal and external to the SoC already are charged.

When not powering up or down, V_{RC33} and V_{DDSYN} do not have a delta requirement to each other for the VRC to operate within specification.

23.5.3.1 Power-Up Sequence If V_{RC33} Grounded

In this case, the 1.5V V_{DD} supply must rise to 1.35V before the 3.3V V_{DDSYN} and the $\overline{\text{RESET}}$ power supplies rise above 2.0V. This is to insure that digital logic in the PLL on the 1.5V supply will not begin to operate below the specified operation range lower limit of 1.35V. Since the internal 1.5V POR is disabled, the internal 3.3V POR or the $\overline{\text{RESET}}$ power POR must be depended on to hold the device in reset. Since they may negate as low as 2.0V, it is necessary for V_{DD} to be within spec before the 3.3V POR and the $\overline{\text{RESET}}$ power POR negate.



NOTE: V_{DD} must reach 1.35V before V_{DDSYN} and $\overline{\text{RESET}}$ reach 2.0V.

Figure 23-4. Power-Up Sequence, V_{RC33} Grounded

23.5.3.2 Power-Down Sequence If V_{RC33} Grounded

In this case, the only requirement is that if V_{DD} falls below its operating range, V_{DDSYN} or the $\overline{\text{RESET}}$ power must fall below 2.0V before V_{DD} is allowed to rise back into its operating range. This is to insure that digital 1.5V logic that is only reset by ORed_POR, which may have been affected by the 1.5V supply falling below spec, will be reset properly.

23.5.3.3 Input Value of Pins During POR Dependent on V_{DD33}

In order to avoid accidentally selecting the bypass clock because PLLCFG[0:1] and $\overline{\text{RSTCFG}}$ were not treated as 1s when POR negates (refer to Section 23.5.3.4, "Pin Values after Negation of POR"), V_{DD33} must not lag V_{DDSYN} and the $\overline{\text{RESET}}$ pin power when powering the device by more than the V_{DD33}_LAG specification in Table 5 of the *MPC5554 Microcontroller Hardware Specifications*. V_{DD33} individually can lag either V_{DDSYN} or the $\overline{\text{RESET}}$ pin power by more than the V_{DD33}_LAG specification. The V_{DD33}_LAG specification applies regardless of whether V_{RC33} is powered. The V_{DD33}_LAG specification only applies during power up. V_{DD33} has no lead or lag requirements when powering down.

23.5.3.4 Pin Values after Negation of POR

Depending on the final PLL mode required, the PLLCFG[0:1] and $\overline{\text{RSTCFG}}$ pins must have the values shown in Table 23-2 after POR negates. See application note AN2613, "MPC5554 Minimum Board Configuration" for one example of the external configuration circuit.

Table 23-2. Values after POR Negation

Final PLL Mode	$\overline{\text{RSTCFG}}$	PLLCFG0	PLLCFG1
Crystal Reference (Using $\overline{\text{RSTCFG}}$ to select Crystal Reference as the default)	1	—	—
Crystal Reference (Using $\overline{\text{RSTCFG}}$ to *not* select Crystal Reference as the default)	—	1	—
External Reference	0	1	1
Dual-Controller	—	1	—

NOTE

After POR negates, $\overline{\text{RSTCFG}}$ and PLLCFG[0:1] can be changed to their final value, but must avoid switching through the 0, 0, 0 state on these pins.

23.6 Revision History

Substantive Changes since Rev 3.0
Changed V_{RCVSS} from 3.3V to 0V in Table 23-1
added "The V_{DD33}_LAG specification applies regardless of whether or not V_{RC33} is powered." to Section 23.5.3.3, "Input Value of Pins During POR Dependent on V_{DD33}."

Chapter 24
IEEE 1149.1 Test Access Port Controller (JTAGC)

24.1 Introduction

The JTAG port of the MPC5553/MPC5554 consists of four inputs and one output. These pins include JTAG compliance select (JCOMP), test data input (TDI), test data output (TDO), test mode select (TMS), and test clock input (TCK). TDI, TDO, TMS, and TCK are compliant with the IEEE 1149.1-2001 standard and are shared with the NDI through the test access port (TAP) interface.

24.1.1 Block Diagram

Figure 24-1 is a block diagram of the JTAG Controller (JTAGC).

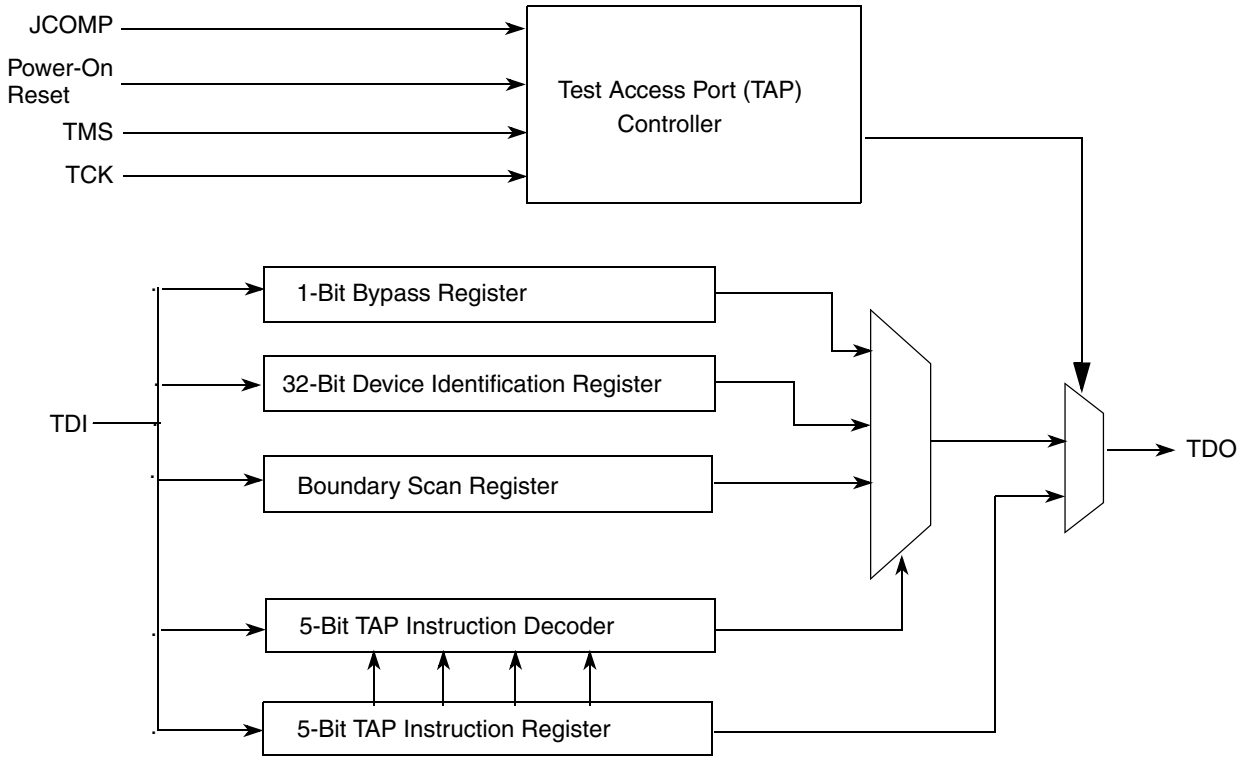


Figure 24-1. JTAG Controller Block Diagram

24.1.2 Overview

The JTAGC provides the means to test chip functionality and connectivity while remaining transparent to system logic when not in test mode. Testing is performed via a boundary scan technique, as defined in the IEEE 1149.1-2001 standard. In addition, instructions can be executed that allow the Test Access Port (TAP) to be shared with other modules on the MCU. All data input to and output from the JTAGC is communicated in serial format.

24.1.3 Features

The JTAGC is compliant with the IEEE 1149.1-2001 standard, and supports the following features:

- IEEE 1149.1-2001 Test Access Port (TAP) interface.
  — 4 pins (TDI, TMS, TCK, and TDO), See Section 24.2, "External Signal Description."
- A JCOMP input that provides the ability to share the TAP.
- A 5-bit instruction register that supports several IEEE 1149.1-2001 defined instructions, as well as several public and private MPC5553/MPC5554 specific instructions.
- Four test data registers: a bypass register, a boundary scan register, and a device identification register. The size of the boundary scan register is 464 bits.
- A TAP controller state machine that controls the operation of the data registers, instruction register and associated circuitry.

24.1.4 Modes of Operation

The JTAGC uses JCOMP and a power-on reset indication as its primary reset signals. Several IEEE 1149.1-2001 defined test modes are supported, as well as a bypass mode.

24.1.4.1 Reset

The JTAGC is placed in reset when the TAP controller state machine is in the TEST-LOGIC-RESET state. The TEST-LOGIC-RESET state is entered upon the assertion of the power-on reset signal, negation of JCOMP, or through TAP controller state machine transitions controlled by TMS. Asserting power-on reset or negating JCOMP results in asynchronous entry into the reset state. While in reset, the following actions occur:

- The TAP controller is forced into the test-logic-reset state, thereby disabling the test logic and allowing normal operation of the on-chip system logic to continue unhindered.
- The instruction register is loaded with the IDCODE instruction.

In addition, execution of certain instructions can result in assertion of the internal system reset. These instructions include EXTEST, CLAMP, and HIGHZ.

24.1.4.2 IEEE 1149.1-2001 Defined Test Modes

The JTAGC supports several IEEE 1149.1-2001 defined test modes. The test mode is selected by loading the appropriate instruction into the instruction register while the JTAGC is enabled. Supported test instructions include EXTEST, HIGHZ, CLAMP, SAMPLE and SAMPLE/PRELOAD. Each instruction defines the set of data registers that may operate and interact with the on-chip system logic while the instruction is current. Only one test data register path is enabled to shift data between TDI and TDO for each instruction.

The boundary scan register is enabled for serial access between TDI and TDO when the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions are active. The single-bit bypass register shift stage is enabled for serial access between TDI and TDO when the HIGHZ, CLAMP or reserved instructions are active. The functionality of each test mode is explained in more detail in Section 24.4.4, “JTAGC Instructions.”

24.1.4.3 Bypass Mode

When no test operation is required, the BYPASS instruction can be loaded to place the JTAGC into bypass mode. While in bypass mode, the single-bit bypass shift register is used to provide a minimum-length serial path to shift data between TDI and TDO.

24.1.4.4 TAP Sharing Mode

On the MPC5553/MPC5554, there are four selectable auxiliary TAP controllers that share the TAP with the JTAGC. Selectable TAP controllers include the Nexus port controller (NPC), e200 OnCE, eTPU Nexus, and eDMA Nexus. The instructions required to grant ownership of the TAP to the auxiliary TAP controllers are ACCESS_AUX_TAP_NPC, ACCESS_AUX_TAP_ONCE, ACCESS_AUX_TAP_eTPUN3, and ACCESS_AUX_TAP_DMAN3. Instruction opcodes for each instruction are shown in Table 24-3.

When the access instruction for an auxiliary TAP is loaded, control of the JTAG pins is transferred to the selected TAP controller. Any data input via TDI and TMS is passed to the selected TAP controller, and any TDO output from the selected TAP controller is sent back to the JTAGC to be output on the pins. The

JTAGC regains control of the JTAG port during the UPDATE-DR state if the PAUSE-DR state was entered. Auxiliary TAP controllers are held in RUN-TEST/IDLE while they are inactive.

For more information on the TAP controllers see Chapter 25, "Nexus Development Interface."

24.2 External Signal Description

24.2.1 Overview

The JTAGC consists of five signals that connect to off-chip development tools and allow access to test support functions. The JTAGC signals are outlined in Table 24-1.

Table 24-1. JTAG Signal Properties

Name	I/O	Function	Reset State	Pull[1]
TCK	I	Test Clock	—	Down
TDI	I	Test Data In	—	Up
TDO	O	Test Data Out	High Z[2]	Down[2]
TMS	I	Test Mode Select	—	Up
JCOMP	I	JTAG Compliancy	—	Down

1 The pull is not implemented in this module. Pull-up/pull-down devices are implemented in the pads.

2 TDO output buffer enable is negated when JTAGC is not in the Shift-IR or Shift-DR states. A weak pull-down may be implemented on TDO.

24.3 Register Definition

This section provides a detailed description of the JTAGC registers accessible through the TAP interface, including data registers and the instruction register. Individual bit-level descriptions and reset states of each register are included. These registers are not memory-mapped and can only be accessed through the TAP.

24.3.1 Register Descriptions

The JTAGC registers are described in this section.

24.3.1.1 Instruction Register

The JTAGC uses a 5-bit instruction register as shown in Figure 24-2. The instruction register allows instructions to be loaded into the module to select the test to be performed or the test data register to be accessed or both. Instructions are shifted in through TDI while the TAP controller is in the Shift-IR state, and latched on the falling edge of TCK in the Update-IR state. The latched instruction value can only be changed in the Update-IR and Test-Logic-Reset TAP controller states. Synchronous entry into the test-logic-reset state results in the IDCODE instruction being loaded on the falling edge of TCK. Asynchronous entry into the test-logic-reset state results in asynchronous loading of the IDCODE instruction. During the capture-IR TAP controller state, the instruction shift register is loaded with the value 0b10101, making this value the register's read value when the TAP controller is sequenced into the Shift-IR state.

	4	3	2	1	0
R	1	0	1	0	1
W	Instruction Code				
Reset	0	0	0	0	1

Figure 24-2. 5-Bit Instruction Register

24.3.1.2 Bypass Register

The bypass register is a single-bit shift register path selected for serial data transfer between TDI and TDO when the BYPASS, CLAMP, HIGHZ or reserve instructions are active. After entry into the capture-DR state, the single-bit shift register is set to a logic 0. Therefore, the first bit shifted out after selecting the bypass register is always a logic 0.

24.3.1.3 Device Identification Register

The device identification register, shown in Figure 24-3, allows the part revision number, design center, part identification number, and manufacturer identity code to be determined through the TAP. The device identification register is selected for serial data transfer between TDI and TDO when the IDCODE instruction is active. Entry into the capture-DR state while the device identification register is selected loads the IDCODE into the shift register to be shifted out on TDO in the Shift-DR state. No action occurs in the update-DR state.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PRN				DC						PIN					
W																
Reset MPC5553	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	1
Reset MPC5554	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PIN				MIC											1
W																
Reset MPC5553	0	0	1	1	0	0	0	0	0	0	0	1	1	1	0	1
Reset MPC5554	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1

Figure 24-3. Device Identification Register

Table 24-2. Device Identification Register Field Descriptions

Bits	Name	Description
0–3	PRN	Part revision number. Contains the revision number of the device. This field changes with each revision of the device or module.
4–9	DC	Design center. Indicates the Freescale design center. For both the MPC5554 and MPC5553, this value is 0x20.
10–19	PIN	Part identification number. Contains the part number of the device. For the MPC5554, this value is 0x0, for the MPC5553 this value is 0x53.
20–30	MIC	Manufacturer identity code. Contains the reduced Joint Electron Device Engineering Council (JEDEC) ID for Freescale, 0xE.
31	—	IDCODE register ID. Identifies this register as the device identification register and not the bypass register. Always set to 1.

24.3.1.4 Boundary Scan Register

The boundary scan register is connected between TDI and TDO when the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions are active. It is used to capture input pin data, force fixed values on output pins, and select a logic value and direction for bidirectional pins. Each bit of the boundary scan register represents a separate boundary scan register cell, as described in the IEEE 1149.1-2001 standard and discussed in Section 24.4.5, "Boundary Scan." The size of the boundary scan register is 464 bits for the MPC5554, and 392 bits for the MPC5553.

24.4 Functional Description

24.4.1 JTAGC Reset Configuration

While in reset, the TAP controller is forced into the test-logic-reset state, thus disabling the test logic and allowing normal operation of the on-chip system logic. In addition, the instruction register is loaded with the IDCODE instruction.

24.4.2 IEEE 1149.1-2001 (JTAG) Test Access Port

The JTAGC uses the IEEE 1149.1-2001 TAP for accessing registers. This port can be shared with other TAP controllers on the MCU. Ownership of the port is determined by the value of the JCOMP signal and the currently loaded instruction. For more detail on TAP sharing via JTAGC instructions refer to Section 24.4.4.2, "ACCESS_AUX_TAP_x Instructions."

Data is shifted between TDI and TDO though the selected register starting with the least significant bit, as illustrated in Figure 24-4. This applies for the instruction register, test data registers, and the bypass register.

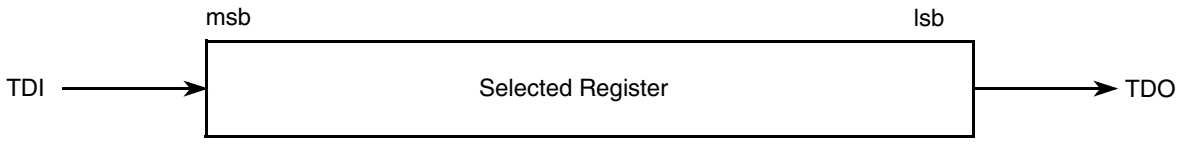
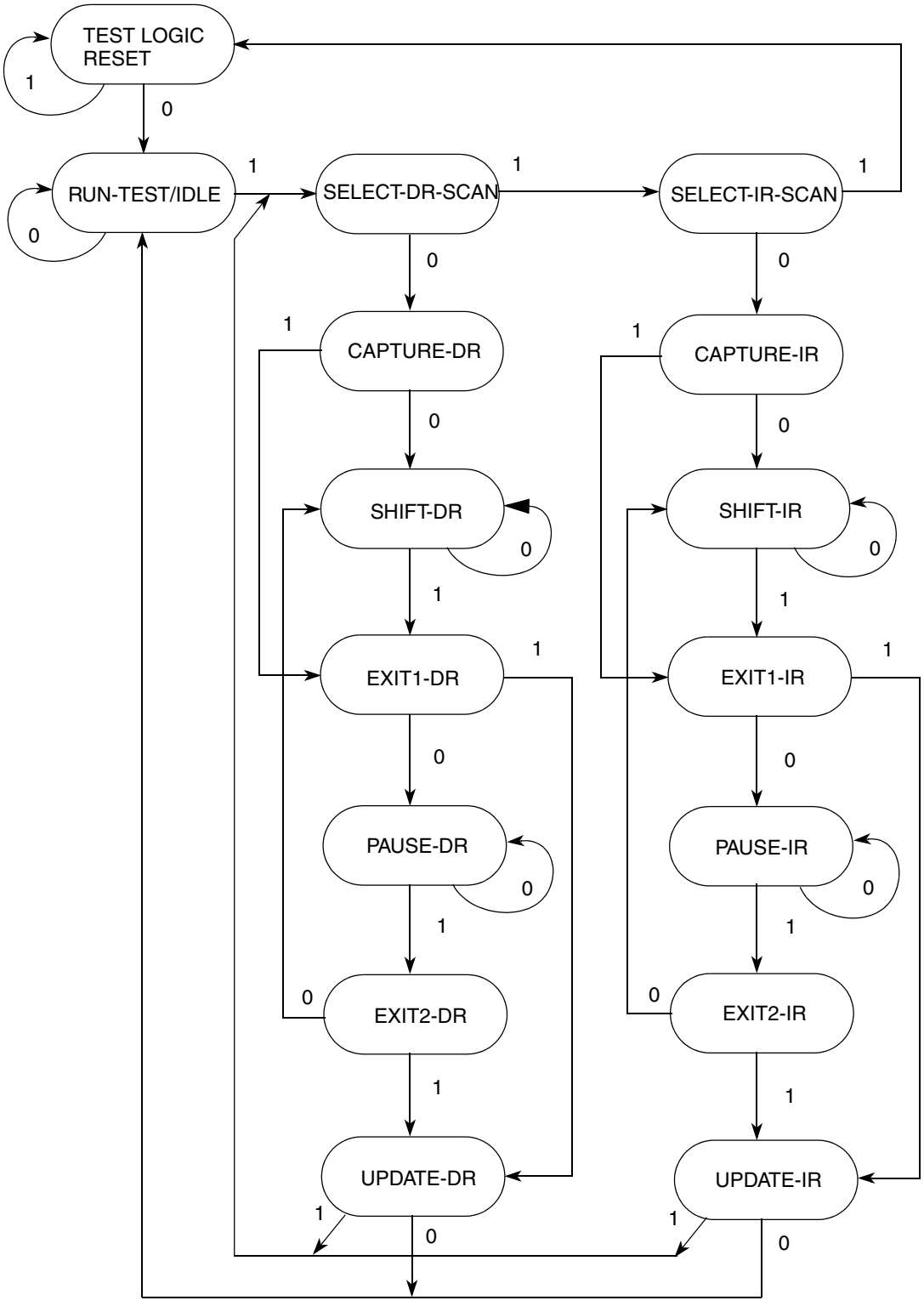


Figure 24-4. Shifting Data Through a Register

24.4.3 TAP Controller State Machine

The TAP controller is a synchronous state machine that interprets the sequence of logical values on the TMS pin. Figure 24-5 shows the machine's states. The value shown next to each state is the value of the TMS signal sampled on the rising edge of the TCK signal. As Figure 24-5 shows, holding TMS at logic 1 while clocking TCK through a sufficient number of rising edges also causes the state machine to enter the test-logic-reset state.



NOTE: The value shown adjacent to each state transition in this figure represents the value of TMS at the time of a rising edge of TCK.

Figure 24-5. IEEE 1149.1-2001 TAP Controller Finite State Machine

24.4.3.1 Enabling the TAP Controller

The JTAGC TAP controller is enabled by setting JCOMP to a logic 1 value.

24.4.3.2 Selecting an IEEE 1149.1-2001 Register

Access to the JTAGC data registers is achieved by loading the instruction register with any of the JTAGC instructions while the JTAGC is enabled. Instructions are shifted in via the select-ir-scan path and loaded in the update-IR state. At this point, all data register access is performed via the select-dr-scan path.

The select-dr-scan path is used to read or write the register data by shifting in the data (lsb first) during the shift-DR state. When reading a register, the register value is loaded into the IEEE 1149.1-2001 shifter during the capture-DR state. When writing a register, the value is loaded from the IEEE 1149.1-2001 shifter to the register during the update-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated once the required number of bits have been acquired.

24.4.4 JTAGC Instructions

The JTAGC implements the IEEE 1149.1-2001 defined instructions listed in Table 24-3. This section gives an overview of each instruction, refer to the IEEE 1149.1-2001 standard for more details.

Table 24-3. JTAG Instructions

Instruction	Code[4:0]	Instruction Summary
IDCODE	00001	Selects device identification register for shift
SAMPLE/PRELOAD	00010	Selects boundary scan register for shifting, sampling, and preloading without disturbing functional operation
SAMPLE	00011	Selects boundary scan register for shifting and sampling without disturbing functional operation
EXTEST	00100	Selects boundary scan register while applying preloaded values to output pins and asserting functional reset
HIGHZ	01001	Selects bypass register while three-stating all output pins and asserting functional reset
CLAMP	01100	Selects bypass register while applying preloaded values to output pins and asserting functional reset
ACCESS_AUX_TAP_NPC	10000	Grants the Nexus port controller (NPC) ownership of the TAP
ACCESS_AUX_TAP_ONCE	10001	Grants the Nexus e200z6 core interface (NZ6C3) ownership of the TAP
ACCESS_AUX_TAP_eTPUN3	10010	Grants the Nexus dual-eTPU development interface (NDEDI) ownership of the TAP
ACCESS_AUX_TAP_DMAN3	10011	Grants the Nexus crossbar DMA interface (NXDM) ownership of the TAP
BYPASS	11111	Selects bypass register for data operations
Factory Debug Reserved[1]	00101, 00110, 01010	Intended for factory debug only
Reserved[2]	All Other Codes	Decoded to select bypass register

[1] Intended for factory debug, and not customer use

[2] Freescale reserves the right to change the decoding of reserved instruction codes in the future

24.4.4.1 BYPASS Instruction

BYPASS selects the bypass register, creating a single-bit shift register path between TDI and TDO. BYPASS enhances test efficiency by reducing the overall shift path when no test operation of the MCU is required. This allows more rapid movement of test data to and from other components on a board that are required to perform test functions. While the BYPASS instruction is active the system logic operates normally.

24.4.4.2 ACCESS_AUX_TAP_x Instructions

The ACCESS_AUX_TAP_x instructions allow the Nexus modules on the MCU to take control of the TAP. When this instruction is loaded, control of the TAP pins is transferred to the selected auxiliary TAP controller. Any data input via TDI and TMS is passed to the selected TAP controller, and any TDO output from the selected TAP controller is sent back to the JTAGC to be output on the pins. The JTAGC regains control of the JTAG port during the UPDATE-DR state if the PAUSE-DR state was entered. Auxiliary TAP controllers are held in RUN-TEST/IDLE while they are inactive.

24.4.4.3 CLAMP Instruction

CLAMP allows the state of signals driven from MCU pins to be determined from the boundary scan register while the bypass register is selected as the serial path between TDI and TDO. CLAMP enhances test efficiency by reducing the overall shift path to a single bit (the bypass register) while conducting an EXTEST type of instruction through the boundary scan register. CLAMP also asserts the internal system reset for the MCU to force a predictable internal state.

24.4.4.4 EXTEST — External Test Instruction

EXTEST selects the boundary scan register as the shift path between TDI and TDO. It allows testing of off-chip circuitry and board-level interconnections by driving preloaded data contained in the boundary scan register onto the system output pins. Typically, the preloaded data is loaded into the boundary scan register using the SAMPLE/PRELOAD instruction before the selection of EXTEST. EXTEST asserts the internal system reset for the MCU to force a predictable internal state while performing external boundary scan operations.

24.4.4.5 HIGHZ Instruction

HIGHZ selects the bypass register as the shift path between TDI and TDO. While HIGHZ is active, all output drivers are placed in an inactive drive state (for example, high impedance). HIGHZ also asserts the internal system reset for the MCU to force a predictable internal state.

24.4.4.6 IDCODE Instruction

IDCODE selects the 32-bit device identification register as the shift path between TDI and TDO. This instruction allows interrogation of the MCU to determine its version number and other part identification data. IDCODE is the instruction placed into the instruction register when the JTAGC is reset.

24.4.4.7 SAMPLE Instruction

The SAMPLE instruction obtains a sample of the system data and control signals present at the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising edge of TCK in the capture-DR state when the SAMPLE instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the Shift-DR state. There is no defined action in the update-DR state. Both the data capture and the shift operation are transparent to system operation.

24.4.4.8 SAMPLE/PRELOAD Instruction

The SAMPLE/PRELOAD instruction has two functions:

- First, the SAMPLE portion of the instruction obtains a sample of the system data and control signals present at the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising edge of TCK in the capture-DR state when the SAMPLE/PRELOAD instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the shift-DR state. Both the data capture and the shift operation are transparent to system operation.
- Secondly, the PRELOAD portion of the instruction initializes the boundary scan register cells before selecting the EXTEST or CLAMP instructions to perform boundary scan tests. This is achieved by shifting in initialization data to the boundary scan register during the shift-DR state. The initialization data is transferred to the parallel outputs of the boundary scan register cells on the falling edge of TCK in the update-DR state. The data is applied to the external output pins by the EXTEST or CLAMP instruction. System operation is not affected.

24.4.5 Boundary Scan

The boundary scan technique allows signals at component boundaries to be controlled and observed through the shift-register stage associated with each pad. Each stage is part of a larger boundary scan register cell, and cells for each pad are interconnected serially to form a shift-register chain around the border of the design. The boundary scan register consists of this shift-register chain, and is connected between TDI and TDO when the EXTEST, SAMPLE, or SAMPLE/PRELOAD instructions are loaded. The shift-register chain contains a serial input and serial output, as well as clock and control signals.

24.5 Initialization/Application Information

The test logic is a static logic design, and TCK can be stopped in either a high or low state without loss of data. However, the system clock is not synchronized to TCK internally. Any mixed operation using both the test logic and the system functional logic requires external synchronization.

To initialize the JTAGC module and enable access to registers, the following sequence is required:

1. Set the JCOMP signal to logic 1, thereby enabling the JTAGC TAP controller.
2. Load the appropriate instruction for the test or action to be performed.

24.6 Revision History

Substantive Changes since Rev 3.0
No changes.

Chapter 25
Nexus Development Interface

25.1 Introduction

The MPC5553/MPC5554 microcontroller contains multiple Nexus clients that communicate over a single IEEE®-ISTO 5001™-2003 Nexus class 3 combined JTAG IEEE® 1149.1/auxiliary out interface. Combined, all of the Nexus clients are referred to as the Nexus development interface (NDI). Class 3 Nexus allows for program, data, and ownership trace of the microcontroller execution without access to the external data and address buses.

This chapter is organized in the following manner:

- The chapter opens with sections that provide a high level view of the Nexus development interface: Section 25.1, "Introduction" through Section 25.8, "NPC Initialization/Application Information."

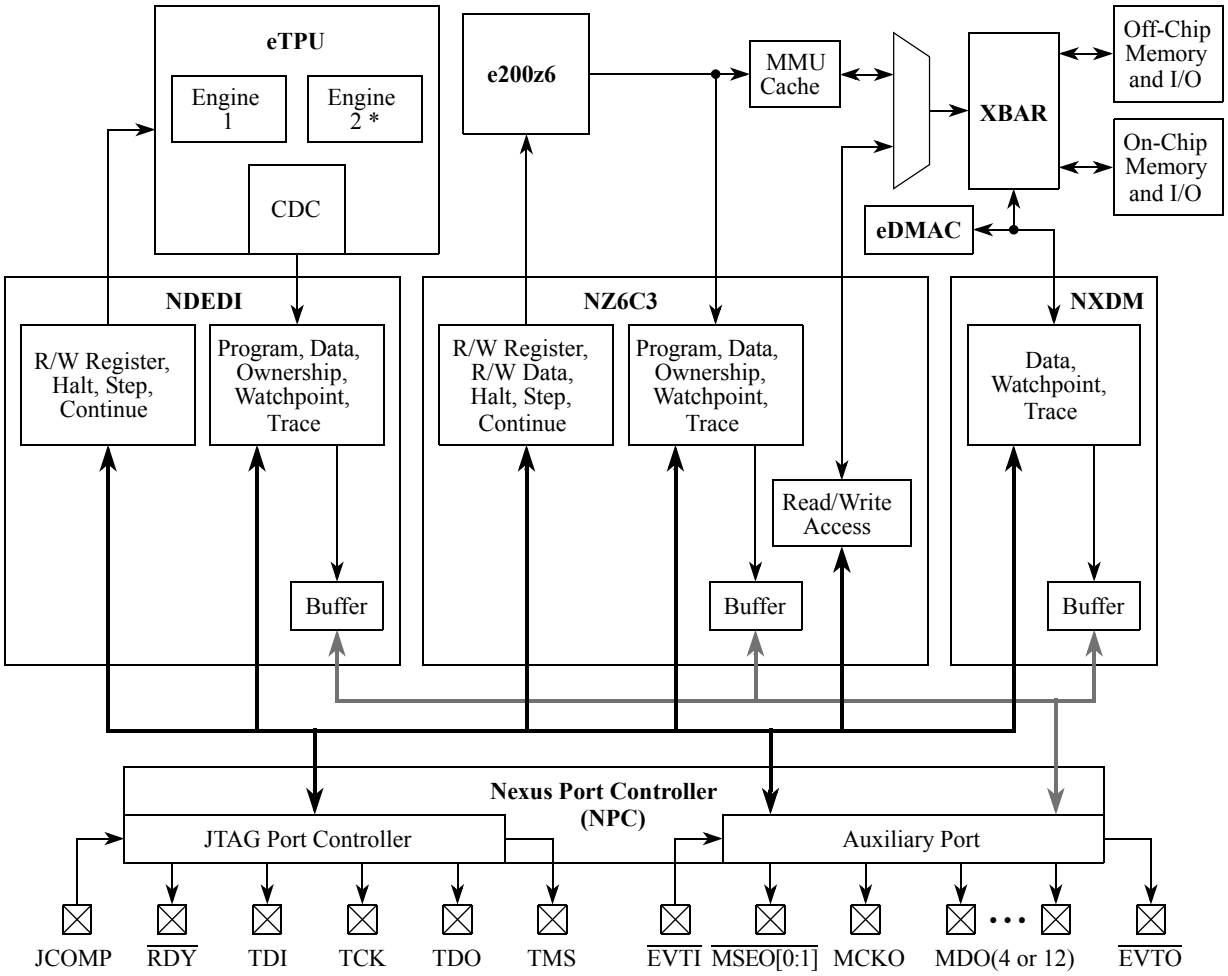
The remainder of the chapter contains sections that discuss the remaining three modules of the Nexus development interface:

- Nexus dual-eTPU development interface (NDEDI). The MPC5554 has two eTPU engines, whereas the MPC5553 has one eTPU engine. Refer to Section 25.9, "Nexus Dual eTPU Development Interface (NDEDI)" and the *eTPU Reference Manual* for information about the NDEDI.
- Nexus e200z6 core interface (NZ6C3). In this chapter, the NZ6C3 interface is discussed in Section 25.10, "e200z6 Class 3 Nexus Module (NZ6C3) through Section 25.11, "NZ6C3 Memory Map/Register Definition."
- Nexus crossbar eDMA interface (NXDM). Refer to Section 25.12, "Nexus Crossbar eDMA Interface (NXDM)"

Communication to the NDI is handled via the auxiliary port and the JTAG port.

- The auxiliary port is comprised of 9 or 17 output pins and 1 input pin. The output pins include 1 message clock out (MCKO) pin, 4 or 12 message data out (MDO) pins, 2 message start/end out (MSEO) pins, 1 ready (RDY) pin, and 1 event out (EVTO) pin. Event in (EVTI) is the only input pin for the auxiliary port.
- The JTAG port consists of four inputs and one output. These pins include JTAG compliance select (JCOMP), test data input (TDI), test data output (TDO), test mode select (TMS), and test clock input (TCK). TDI, TDO, TMS, and TCK are compliant with the IEEE® 1149.1-2001 standard and are shared with the NDI through the test access port (TAP) interface. JCOMP along with power-on reset and the TAP state machine are used to control reset for the NDI module. Ownership of the TAP is achieved by loading the appropriate enable instruction for the desired Nexus client in the JTAG controller (JTAGC) when JCOMP is asserted. See Table 25-4 for the JTAGC opcodes to access the different Nexus clients.

25.1.1 Block Diagram



* The MPC5553 has only one eTPU engine; the MPC5554 has two eTPU engines.

Figure 25-1. NDI Block Diagram

25.1.2 Features

The NDI module is compliant with the IEEE-ISTO 5001-2003 standard. The following features are implemented:

- 15 or 23 bit full duplex pin interface for medium and high visibility throughput.
- One of two modes selected by register configuration: full port mode (FPM) and reduced port mode (RPM). FPM comprises 12 MDO pins, and RPM comprises 4 MDO pins.
- Auxiliary output port.
  - 1 MCKO (message clock out) pin
  - 4 or 12 MDO (message data out) pins
  - 2 $\overline{\text{MSEO}}$ (message start/end out) pins
  - 1 $\overline{\text{RDY}}$ (ready) pin
  - 1 $\overline{\text{EVTO}}$ (event out) pin

- Auxiliary input port.
  - 1 $\overline{\text{EVTI}}$ (event in) pin
  - 5 pin JTAG port (JCOMP, TDI, TDO, TMS, and TCK)
- Host processor (e200z6) development support features (NZ6C3).
  - IEEE-ISTO 5001-2003 standard class 3 compliant.
  - Data trace via data write messaging (DWM) and data read messaging (DRM). This allows the development tool to trace reads and/or writes to selected internal memory resources.
  - Ownership trace via ownership trace messaging (OTM). OTM facilitates ownership trace by providing visibility of which process ID or operating system task is activated. An ownership trace message is transmitted when a new process/task is activated, allowing development tools to trace ownership flow.
  - Program trace via branch trace messaging (BTM). Branch trace messaging displays program flow discontinuities (direct branches, indirect branches, exceptions, etc.), allowing the development tool to interpolate what transpires between the discontinuities. Thus, static code can be traced.
  - Watchpoint messaging (WPM) via the auxiliary port.
  - Watchpoint trigger enable of program and/or data trace messaging.
  - Data tracing of instruction fetches via private opcodes.
  - Subset of PowerPC Book E software debug facilities with OnCE block (Nexus class 1 features).
- eDMA development support features (NXDM).
  - Data trace via data write messaging (DWM) and data read messaging (DRM). This allows the development tool to trace DMA generated reads and/or writes to selected address ranges in the MPC5554's memory map.
  - Watchpoint messaging (WPM) via the auxiliary port.
  - Watchpoint trigger enable/disable of data trace messaging.
- eTPU development support features (NDEDI).
  - IEEE-ISTO 5001-2002 standard Class 3 compliant for the eTPU engines.
  - Data trace via data write messaging and data read messaging. This allows the development tool to trace reads and writes to selected shared parameter RAM (SPRAM) address ranges. Four data trace windows are shared between the two eTPU engines.
  - Ownership trace via ownership trace messaging (OTM). OTM facilitates ownership trace by providing visibility of which channel is being serviced. An ownership trace message is transmitted to indicate when a new channel service request is scheduled, allowing the development tools to trace task flow. A special OTM is sent when the engine enters in idle state, meaning that all requests were serviced and no new requests are yet scheduled.
  - Program trace via branch trace messaging. BTM displays program flow discontinuities (start, jumps, return, etc.), allowing the development tool to interpolate what transpires between the discontinuities. Thus static code can be traced. The branch trace messaging method uses the branch/predicate method to reduce the number of generated messages.
  - Watchpoint messaging via the auxiliary port. WPM provides visibility of the occurrence of the eTPUs' watchpoints and breakpoints.
  - Nexus based breakpoint/watchpoint configuration and single step support.
- Run-time access to the on-chip memory map via the Nexus read/write access protocol. This feature supports accesses for run-time internal visibility, calibration variable acquisition, calibration constant tuning, and external rapid prototyping for powertrain automotive development systems.
- All features are independently configurable and controllable via the IEEE® 1149.1 I/O port.

- The NDI block reset is controlled with JCOMP, power-on reset, and the TAP state machine. These sources are independent of system reset.
- System clock locked status indication via MDO0 following power-on reset.

25.1.3 Modes of Operation

The NDI block is in reset when the TAP controller state machine is in the TEST-LOGIC-RESET state. The TEST-LOGIC-RESET state is entered on the assertion of the power-on reset signal, negation of JCOMP, or through state machine transitions controlled by TMS. Assertion of JCOMP allows the NDI to move out of the reset state, and is a prerequisite to grant Nexus clients control of the TAP. Ownership of the TAP is achieved by loading the appropriate enable instruction for the desired Nexus client in the JTAGC controller (JTAGC) block when JCOMP is asserted.

Following negation of power-on reset, the NPC remains in reset until the system clock achieves lock. In PLL bypass mode, the NDI can transition out of the reset state immediately following negation of power-on reset. Refer to Section 25.4.5, "System Clock Locked Indication" for more details.

25.1.3.1 Nexus Reset Mode

In Nexus reset mode, the following actions occur:

- Register values default back to their reset values.
- The message queues are marked as empty.
- The auxiliary output port pins are negated if the NDI controls the pads.
- The TDO output buffer is disabled if the NDI has control of the TAP.
- The TDI, TMS, and TCK inputs are ignored.
- The NDI block indicates to the MCU that it is not using the auxiliary output port. This indication can be used to three-state the output pins or use them for another function.

25.1.3.2 Full-Port Mode

In full-port mode, all the available MDO pins are used to transmit messages. All trace features are enabled or can be enabled by writing the configuration registers via the JTAG port. The number of MDO pins available is 12.

25.1.3.3 Reduced-Port Mode

In reduced-port mode, a subset of the available MDO pins are used to transmit messages. All trace features are enabled or can be enabled by writing the configuration registers via the JTAG port. The number of MDO pins available is 4. Unused MDO (MDO[11:4]) pins can be used as GPIO. Details on GPIO functionality configuration can be found in Chapter 6, "System Integration Unit (SIU)."

25.1.3.4 Disabled-Port Mode

In disabled-port mode, message transmission is disabled. Any debug feature that generates messages can not be used. The primary features available are class 1 features and read/write access.

25.1.3.5 Censored Mode

When the device is in censored mode, reading the contents of internal flash externally is not allowed. To prevent Nexus modules from violating censorship, the NPC is held in reset when in censored mode,

asynchronously holding all other Nexus modules in reset as well. This prevents Nexus read/write to memory mapped resources and the transmission of Nexus trace messages. Refer to Table 13-17 for information on Nexus port enabling and disabling regarding censorship.

25.2 External Signal Description

The auxiliary and JTAG pin interfaces provide for the transmission of messages from Nexus modules to the external development tools and for access to Nexus client registers. The auxiliary/JTAG pin definitions are outlined in Table 25-1.

Table 25-1. Signal Properties

Name	Port	Function	Reset State
$\overline{\text{EVTO}}$	Auxiliary	Event Out pin	Negated
$\overline{\text{EVTI}}$	Auxiliary	Event In pin	Pulled Up
MCKO	Auxiliary	Message Clock Out pin (from NPC)	Enabled
MDO[3:0] or MDO[11:0]	Auxiliary	Message Data Out pins	Driven Low[1]
$\overline{\text{MSEO}}$[1:0]	Auxiliary	Message Start/End Out pins	Negated
$\overline{\text{RDY}}$	Auxiliary	Ready Out pin	Negated
JCOMP	JTAG	JTAG Compliancy and TAP Sharing Control	Pulled Down
TCK	JTAG	Test Clock Input	Pulled Up
TDI	JTAG	Test Data Input	Pulled Up
TDO	JTAG	Test Data Output	Pulled Up
TMS	JTAG	Test Mode Select Input	Pulled Up

[1] Following a power-on reset, MDO0 remains asserted until power-on reset is exited and the system clock achieves lock.

25.2.1 Detailed Signal Descriptions

This section describes each of the signals listed in Table 25-1 in more detail.

25.2.1.1 Event Out ($\overline{\text{EVTO}}$)

$\overline{\text{EVTO}}$ is an output pin that is asserted upon breakpoint occurrence to provide breakpoint status indication or to signify that an event has occurred. The $\overline{\text{EVTO}}$ output of the NPC is generated based on the values of the individual $\overline{\text{EVTO}}$ signals from all Nexus modules that implement the signal.

25.2.1.2 Event In ($\overline{\text{EVTI}}$)

$\overline{\text{EVTI}}$ is used to initiate program and data trace synchronization messages or to generate a breakpoint. $\overline{\text{EVTI}}$ is edge-sensitive for synchronization and breakpoint generation.

25.2.1.3 Message Data Out (MDO[3:0/11:0])

Message data out (MDO) are output pins used for uploading OTM, BTM, DTM, and other messages to the development tool. The development tool should sample MDO on the rising edge of MCKO. The width of the MDO bus used is determined by the Nexus PCR[FPM] configuration.

Following a power-on reset, MDO0 remains asserted until power-on reset is exited and the system clock achieves lock.

25.2.1.4 Message Start/End Out (MSEO[1:0])

$\overline{\text{MSEO}}$[1:0] are output pins that indicates when a message on the MDO pins has started, when a variable length packet has ended, or when the message has ended. The development tool should sample the $\overline{\text{MSEO}}$ pins on the rising edge of MCKO.

25.2.1.5 Ready ($\overline{\text{RDY}}$)

$\overline{\text{RDY}}$ is an output pin that indicates when a device is ready for the next access.

25.2.1.6 JTAG Compliancy (JCOMP)

The JCOMP signal enables or disables the TAP controller. The TAP controller is enabled when JCOMP asserted, otherwise the TAP controller remains in reset.

25.2.1.7 Test Data Output (TDO)

The TDO pin transmits serial output for instructions and data. TDO is tri-stateable and is actively driven in the SHIFT-IR and SHIFT-DR controller states. TDO is updated on the falling edge of TCK and sampled by the development tool on the rising edge of TCK.

25.2.1.8 Test Clock Input (TCK)

The TCK pin is used to synchronize the test logic and control register access through the JTAG port.

25.2.1.9 Test Data Input (TDI)

The TDI pin receives serial test instruction and data. TDI is sampled on the rising edge of TCK.

25.2.1.10 Test Mode Select (TMS)

The TMS pin is used to sequence the IEEE® 1149.1-2001 TAP controller state machine. TMS is sampled on the rising edge of TCK.

25.3 Memory Map

The NDI block contains no memory mapped registers. Nexus registers are accessed by the development tool via the JTAG port using a register index and a client select value. The client select is controlled by loading the correct access instruction into the JTAG controller; refer to Section 25.4.1. OnCE registers are accessed by loading the appropriate value in the RS[0:6] field of the OnCE command register (OCMD) via the JTAG port.

Table 25-2 shows the NDI registers by client select and index values. Table 25-3 shows the OnCE register addressing.

Table 25-2. NDI Registers

Client Select	Index	Register
e200z6 Control/Status Registers		
0bxxxx	0	Device ID (DID)
0b0000	2	e200z6 Development Control1 (PPC_DC1)
0b0000	3	e200z6 Development Control2 (PPC_DC2)
0b0000	4	e200z6 Development Status (PPC_DS)
0b0000	6	e200z6 User Base Address (PPC_UBA)
0b0000	7	Read/Write Access Control/Status (RWCS)
0b0000	9	Read/Write Access Address (RWA)
0b0000	10	Read/Write Access Data (RWD)
0b0000	11	e200z6 Watchpoint Trigger (PPC_WT)
0b0000	13	e200z6 Data Trace Control (PPC_DTC)
0b0000	14	e200z6 Data Trace Start Address 0 (PPC_DTSA1)
0b0000	15	e200z6 Data Trace Start Address 1 (PPC_DTSA2)
0b0000	18	e200z6 Data Trace End Address 0 (PPC_DTEA1)
0b0000	19	e200z6 Data Trace End Address 1 (PPC_DTEA2)
0bxxxx	127	Port Configuration Register (PCR)
eDMA Control/Status Registers		
0b0001	2	eDMA Development Control (AHB_DC)
0b0001	11	eDMA Watchpoint Trigger (AHB_WT)
0b0001	13	eDMA Data Trace Control (AHB_DTC)
0b0001	14	eDMA Data Trace Start Address 0 (AHB_DTSA1)
0b0001	15	eDMA Data Trace Start Address 1 (AHB_DTSA2)
0b0001	18	eDMA Data Trace End Address 0 (AHB_DTEA1)
0b0001	19	eDMA Data Trace End Address 1 (AHB_DTEA2)
0b0001	22	eDMA Breakpoint/Watchpoint Control 1 (AHB_BWC1)
0b0001	23	eDMA Breakpoint/Watchpoint Control 2 (AHB_BWC2)
0b0001	30	eDMA Breakpoint/Watchpoint Address 1 (AHB_BWA1)
0b0001	31	eDMA Breakpoint/Watchpoint Address 2 (AHB_BWA2)
0bxxxx	127	Port Configuration Register (PCR)

Table 25-2. NDI Registers (continued)

Client Select	Index	Register
eTPU1 Control/Status Registers		
0bxxxx	0	Device ID (DID)
0b0010	2	eTPU1 Development Control (NDI_eTPU1_DC)
0b0010	4	eTPU1 Development Status (NDEDI_eTPU1_DS)
0b0000	7	Read/Write Access Control/Status (RWCS)
0b0000	9	Read/Write Access Address (RWA)
0b0000	10	Read/Write Access Data (RWD)
0b0010	11	eTPU1 Watchpoint Trigger (NDI_eTPU1_WT)
0b0010	13	eTPU1 Data Trace Control (NDI_eTPU1_DTC)
0b0010	22	eTPU1 Breakpoint/Watchpoint Control 1 (NDEDI_eTPU1_BWC1)
0b0010	23	eTPU1 Breakpoint/Watchpoint Control 2 (NDEDI_eTPU1_BWC2)
0b0010	24	eTPU1 Breakpoint/Watchpoint Control 3 (NDEDI_eTPU1_BWC3)
0b0010	30	eTPU1 Breakpoint/Watchpoint Address 1 (NDEDI_eTPU1_BWA1)
0b0010	31	eTPU1 Breakpoint/Watchpoint Address 2 (NDEDI_eTPU1_BWA2)
0b0010	38	eTPU1 Breakpoint/Watchpoint Data 1 (NDEDI_eTPU1_BWD1)
0b0010	39	eTPU1 Breakpoint/Watchpoint Data 1 (NDEDI_eTPU1_BWD2)
0b0010	64	eTPU1 Program Trace Channel Enable (NDI_eTPU1_PTCE)
0b0010	69	eTPU1 Microinstruction Debug Register (NDEDI_eTPU1_INST)
0b0010	70	eTPU1 Microprogram Counter Debug Register (NDEDI_eTPU1_MPC)
0b0010	71	eTPU1 Channel Flag Status Register (NDEDI_eTPU1_CFSR)
0bxxxx	127	Port Configuration Register (PCR)
eTPU2 Control/Status Registers (MPC5554 Only)		
0bxxxx	0	Device ID (DID)
0b0011	2	eTPU2 Development Control (NDI_eTPU2_DC)
0b0011	4	eTPU2 Development Status (NDEDI_eTPU2_DS)
0b0000	7	Read/Write Access Control/Status (RWCS)
0b0000	9	Read/Write Access Address (RWA)
0b0000	10	Read/Write Access Data (RWD)
0b0011	11	eTPU2 Watchpoint Trigger (NDI_eTPU2_WT)
0b0011	13	eTPU2 Data Trace Control (NDI_eTPU2_DTC)
0b0011	22	eTPU2 Breakpoint/Watchpoint Control 1 (NDEDI_eTPU2_BWC1)
0b0011	23	eTPU2 Breakpoint/Watchpoint Control 2 (NDEDI_eTPU2_BWC2)

Table 25-2. NDI Registers (continued)

Client Select	Index	Register
0b0011	24	eTPU2 Breakpoint/Watchpoint Control 3 (NDEDI_eTPU2_BWC3)
0b0011	30	eTPU2 Breakpoint/Watchpoint Address 1 (NDEDI_eTPU2_BWA1)
0b0011	31	eTPU2 Breakpoint/Watchpoint Address 2 (NDEDI_eTPU2_BWA2)
0b0011	38	eTPU2 Breakpoint/Watchpoint Data 1 (NDEDI_eTPU2_BWD1)
0b0011	39	eTPU2 Breakpoint/Watchpoint Data 1 (NDEDI_eTPU2_BWD2)
0b0011	64	eTPU2 Program Trace Channel Enable (NDI_eTPU2_PTCE)
0b0011	69	eTPU2 Microinstruction Debug Register (NDEDI_eTPU2_INST)
0b0011	70	eTPU2 Microprogram Counter Debug Register (NDEDI_eTPU2_MPC)
0b0011	71	eTPU2 Channel Flag Status Register (NDEDI_eTPU2_CFSR)
0bxxxx	127	Port Configuration Register (PCR)
eTPU CDC Control/Status Registers		
0b0100	13	eTPU CDC Data Trace Control (NDEDI_CDC_DTC)
eTPU1/eTPU2/CDC Shared Control/Status Registers		
0b0010 or 0b0011 or 0b0100	65	eTPU Data Trace Address Range 0 (eTPU_DTAR0)
0b0010 or 0b0011 or 0b0100	66	eTPU Data Trace Address Range 1 (eTPU_DTAR1)
0b0010 or 0b0011 0r 0b0100	67	eTPU Data Trace Address Range 2 (eTPU_DTAR2)
0b0010 or 0b0011 0r 0b0100	68	eTPU Data Trace Address Range 3 (eTPU_DTAR3)

Table 25-3. OnCE Register Addressing

OCMD, RS[0:6]	Register Selected
000 0000	Reserved
000 0001	Reserved
000 0010	JTAG ID (read-only)
000 0011 - 000 1111	Reserved
001 0000	CPU Scan Register (CPUSCR)
001 0001	No Register Selected (Bypass)
001 0010	OnCE Control Register (OCR)
001 0011	Reserved
001 0100 - 001 1111	Reserved

Table 25-3. OnCE Register Addressing (continued)

OCMD, RS[0:6]	Register Selected
010 0000	Instruction Address Compare 1 (IAC1)
010 0001	Instruction Address Compare 2 (IAC2)
010 0010	Instruction Address Compare 3 (IAC3)
010 0011	Instruction Address Compare 4 (IAC4)
010 0100	Data Address Compare 1 (DAC1)
010 0101	Data Address Compare 2 (DAC2)
010 0110	Reserved
010 0111	Reserved
010 1000 - 010 1011	Reserved
010 1100	Debug Counter Register (DBCNT)
010 1101	Debug PCFIFO (PCFIFO) (read-only)
010 1110 - 010 1111	Reserved
011 0000	Debug Status Register (DBSR)
011 0001	Debug Control Register 0 (DBCR0)
011 0010	Debug Control Register 1 (DBCR1)
011 0011	Debug Control Register 2 (DBCR2)
011 0100	Debug Control Register 3 (DBCR3)
011 0101 - 101 1111	Reserved (do not access)
111 0000 - 111 1011	General Purpose Register Selects [0:11]
111 1100	Nexus3-Access
111 1101	LSRL Select
111 1110	Enable_OnCE (and Bypass)
111 1111	Bypass

25.4 NDI Functional Description

25.4.1 Enabling Nexus Clients for TAP Access

Once the NDI is out of the reset state, the loading of a specific instruction in the JTAG controller (JTAGC) block is required to grant the NDI ownership of the TAP. Each Nexus client has its own JTAGC instruction opcode for ownership of the TAP, granting that client the means to read/write its registers. The JTAGC instruction opcode for each Nexus client is shown in Table 25-4. Once the JTAGC opcode for a client has been loaded, the client is enabled by loading its NEXUS-ENABLE instruction. The NEXUS-ENABLE

instruction opcode for each Nexus client is listed in Table 25-5. Opcodes for all other instructions supported by Nexus clients can be found in the relevant sections of this chapter.

Table 25-4. JTAG Client Select Instructions

JTAGC Instruction	Opcode	Description
ACCESS_AUX_TAP_NPC	10000	Enables access to the NPC TAP controller
ACCESS_AUX_TAP_ONCE	10001	Enables access to the e200z6 OnCE TAP controller
ACCESS_AUX_TAP_eTPU	10010	Enables access to the eTPU Nexus TAP controller
ACCESS_AUX_TAP_DMAN3	10011	Enables access to the eDMA Nexus TAP controller

Table 25-5. Nexus Client JTAG Instructions

Instruction	Description	Opcode
NPC JTAG Instruction Opcodes		
NEXUS_ENABLE	Opcode for NPC Nexus Enable instruction (4-bits)	0x0
BYPASS	Opcode for the NPC BYPASS instruction (4-bits)	0xF
e200z6 OnCE JTAG Instruction Opcodes[1]		
NEXUS3_ACCESS	Opcode for e200z6 OnCE Nexus Enable instruction (10-bits)	0x7C
BYPASS	Opcode for the e200z6 OnCE BYPASS instruction (10-bits)	0x7F
NDEDI JTAG Instruction Opcodes		
NEXUS_ENABLE	Opcode for NDEDI Nexus Enable instruction (4-bits)	0x0
BYPASS	Opcode for the NDEDI BYPASS instruction (4-bits)	0xF
eDMA Nexus JTAG Instruction Opcodes		
NEXUS_ACCESS	Opcode for eDMA Nexus Enable instruction (4-bits)	0x0
BYPASS	Opcode for the eDMA Nexus BYPASS instruction (4-bits)	0xF

[1] Refer to the e200Z6 Reference Manual for a complete list of available OnCE instructions.

25.4.2 Configuring the NDI for Nexus Messaging

The NDI is placed in disabled mode upon exit of power-on reset. If message transmission via the auxiliary port is desired, a write to the port configuration register (PCR) located in the NPC is then required to enable the NDI and select the mode of operation. Asserting MCKO_EN in the PCR places the NDI in enabled mode and enables MCKO. The frequency of MCKO is selected by writing the MCKO_DIV field. Asserting or negating the FPM bit selects full-port or reduced-port mode, respectively. When writing to the PCR, the PCR lsb must be written to a logic 0. Setting the lsb of the PCR enables factory debug mode and prevents the transmission of Nexus messages.

Figure 25-6 describes the NDI configuration options.

Table 25-6. NDI Configuration Options

JCOMP Asserted	MCKO_EN bit of the Port Configuration Register	FPM bit of the Port Configuration Register	Configuration
No	X	X	Reset
Yes	0	X	Disabled
Yes	1	1	Full-Port Mode
Yes	1	0	Reduced-Port Mode

25.4.3 Programmable MCKO Frequency

MCKO is an output clock to the development tools used for the timing of $\overline{\text{MSEO}}$ and MDO pin functions. MCKO is derived from the system clock, and its frequency is determined by the value of the MCKO_DIV field in the port configuration register (PCR) located in the NPC. Possible operating frequencies include one-half, one-quarter, and one-eighth system clock speed.

Figure 25-7 shows the MCKO_DIV encodings. In this table, SYS_CLK represents the system clock frequency. The default value selected if a reserved encoding is programmed is SYS_CLK/2.

Table 25-7. MCKO_DIV Values

MCKO_DIV[2:0]	MCKO Frequency
0b000	Reserved
0b001	SYS_CLK/2
0b010	Reserved
0b011	SYS_CLK/4
0b100	Reserved
0b101	Reserved
0b110	Reserved
0b111	SYS_CLK/8

25.4.4 Nexus Messaging

Most of the messages transmitted by the NDI include a SRC field. This field is used to identify which source generated the message. Figure 25-8 shows the values used for the SRC field by the different clients on the MPC5553/MPC5554. These 4-bit values are specific to the MPC5553/MPC5554. The same values are used for the client select values written to the client select control register.

Table 25-8. SRC Packet Encodings

SRC[3:0]	Client
0b0000	e200z6
0b0001	eDMA
0b0010	eTPU1 (ENGINE1_SRC)

Table 25-8. SRC Packet Encodings

SRC[3:0]	Client
0b0011	eTPU2 (ENGINE2_SRC) [1]
0b0100	eTPU CDC[2] (CDC_SRC)
0b0101-0b1111	Reserved

[1] MPC5554 only, not in the MPC5553.

[2] CDC is the eTPU Coherent Dual-Parameter Controller. Refer to the *eTPU Reference Manual* for more information.

25.4.5 System Clock Locked Indication

Following a power-on reset, the lsb of the auxiliary output port pins (MDO0) can be monitored to provide the lock status of the system clock. MDO0 is driven to a logic 1 until the system clock achieves lock after exiting power-on reset. Once the system clock is locked, MDO0 is negated and tools may begin Nexus configuration. Loss of lock conditions that occur subsequent to the exit of power-on reset and the initial lock of the system clock do not cause a Nexus reset, and therefore do not result in MDO0 driven high.

25.5 Nexus Port Controller (NPC)

The Nexus port controller (NPC) is that part of the NDI that controls access and arbitration of the MPC5553/MPC5554's internal Nexus modules. The NPC contains the port configuration register (PCR) and the device identification register (DID). The contents of the DID are the same as the JTAGC device identification register.

25.5.1 Overview

The MPC5553/MPC5554 incorporates multiple modules that require development support. Each of these modules implements a development interface based on the IEEE-ISTO 5001-2001 standard and must share the input and output ports that interface with the development tool. The NPC controls the usage of these ports in a manner that allows the individual modules to share the ports, while appearing to the development tool as a single module.

25.5.2 Features

The NPC performs the following functions:

- Controls arbitration for ownership of the Nexus auxiliary output port
- Nexus device identification register and messaging
- Generates MCKO enable and frequency division control signals
- Controls sharing of $\overline{\text{EVTO}}$
- Control of the device-wide debug mode
- Generates asynchronous reset signal for Nexus modules based on JCOMP input, censorship status, and power-on reset status
- System clock locked status indication via MDO0 during Nexus reset
- Provides Nexus support for censorship mode

25.6 Memory Map/Register Definition

This section provides a detailed description of the NPC registers accessible to the end user. Individual bit-level descriptions and reset states of the registers are included.

25.6.1 Memory Map

Table 25-9 shows the NPC registers by index values. The registers are not memory-mapped and can only be accessed via the TAP. The NPC does not implement the client select control register because the value does not matter when accessing the registers. Note that the bypass register (refer to Section 25.6.2.1) and instruction register (refer to Section 25.6.2.2) have no index values. These registers are not accessed in the same manner as Nexus client registers.

Table 25-9. NPC Memory Map

Index	Register Name	Register Description	Size (bits)
0	DID	Device ID register	32
127	PCR	Port configuration register	32

25.6.2 Register Descriptions

This section consists of NPC register descriptions. Additional information regarding references to the TAP controller state may be found in Section 24.4.3, “TAP Controller State Machine.”

25.6.2.1 Bypass Register

The bypass register is a single-bit shift register path selected for serial data transfer between TDI and TDO when the BYPASS instruction or any unimplemented instructions are active. After entry into the Capture-DR state, the single-bit shift register is set to a logic 0. Therefore, the first bit shifted out after selecting the bypass register is always a logic 0.

25.6.2.2 Instruction Register

The NPC uses a 4-bit instruction register as shown in Figure 25-2. The instruction register is accessed via the SELECT_IR_SCAN path of the tap controller state machine, and allows instructions to be loaded into the module to enable the NPC for register access (NEXUS_ENABLE) or select the bypass register as the shift path from TDI to TDO (BYPASS or unimplemented instructions).

Instructions are shifted in through TDI while the TAP controller is in the Shift-IR state, and latched on the falling edge of TCK in the Update-IR state. The latched instruction value can only be changed in the Update-IR and test-logic-reset TAP controller states. Synchronous entry into the test-logic-reset state results in synchronous loading of the BYPASS instruction. Asynchronous entry into the test-logic-reset state results in asynchronous loading of the BYPASS instruction. During the Capture-IR TAP controller state, the instruction register is loaded with the value of the previously executed instruction, making this value the register’s read value when the TAP controller is sequenced into the Shift-IR state.

	3	2	1	0
R	Previous Instruction Opcode			
W	Instruction Opcode			
Reset:	BYPASS Instruction Opcode (0xF)			

Figure 25-2. 4-Bit Instruction Register

25.6.2.3 Nexus Device ID Register (DID)

The NPC device identification register, shown in Figure 25-3, allows the part revision number, design center, part identification number, and manufacturer identity code of the part to be determined through the auxiliary output port.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PRN				DC						PIN					
W																
Reset for MPC5553	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	1
Reset for MPC5554	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
Reg Index	0															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PIN				MIC											1
W																
Reset for MPC5553	0	0	1	1	0	0	0	0	0	0	0	1	1	1	0	1
Reset for MPC5554	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1
Reg Index	0															

Figure 25-3. Nexus Device ID Register (DID)

Table 25-10. DID Register Field Descriptions

Bits	Name	Description
31–28	PRN	Part revision number. Contains the revision number of the part. This field changes with each revision of the device or module.
27–22	DC	Design center. Indicates the Freescale design center. For both the MPC5554 and MPC5553, this value is 0x20.
21–12	PIN	Part identification number. Contains the part number of the device. The PIN for the MPC5553 is 0x53, for the MPC5554 it is 0x0.

Table 25-10. DID Register Field Descriptions (continued)

Bits	Name	Description
11–1	MIC	Manufacturer identity code. Contains the reduced Joint Electron Device Engineering Council (JEDEC) ID for Freescale, 0xE.
0	—	Fixed per JTAG 1149.1 1 Always set

25.6.2.4 Port Configuration Register (PCR)

The PCR, shown in Figure 25-4, is used to select the NPC mode of operation, enable MCKO and select the MCKO frequency, and enable or disable MCKO gating. This register should be configured as soon as the NPC is enabled.

NOTE

The mode (MCKO_GT) or clock division (MCKO_DIV) bits must not be modified after MCKO has been enabled. Changing the mode or clock division while MCKO is enabled can produce unpredictable results.

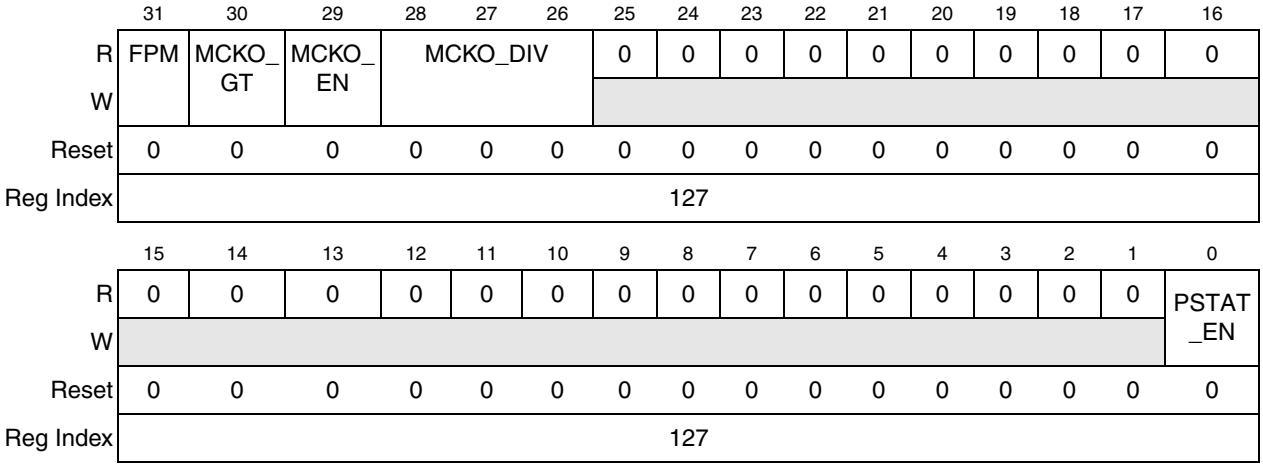


Figure 25-4. Port Configuration Register (PCR)

Table 25-11. PCR Field Descriptions

Bits	Name	Description
31	FPM	Full port mode. Determines if the auxiliary output port uses the full MDO port or a reduced MDO port to transmit messages. 0 The subset of MDO[3:0] pins are used to transmit messages. 1 All MDO[11:0] pins are used to transmit messages.
30	MCKO_GT	MCKO clock gating control. Enables or disables MCKO clock gating. If clock gating is enabled, the MCKO clock is gated when the NPC is in enabled mode but not actively transmitting messages on the auxiliary output port. When clock gating is disabled, MCKO is allowed to run even if no auxiliary output port messages are being transmitted. 0 MCKO gating is disabled. 1 MCKO gating is enabled.

Table 25-11. PCR Field Descriptions (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>29</td><td>MCKO_EN</td><td>MCKO enable. Enables the MCKO clock. When enabled, the frequency of MCKO is determined by the MCKO_DIV field.
0 MCKO clock is driven to zero.
1 MCKO clock is enabled.</td></tr>
<tr><td>28–26</td><td>MCKO_DIV [2:0]</td><td>MCKO division factor. Determines the frequency of MCKO relative to the system clock frequency when MCKO_EN is asserted. The table below shows the meaning of MCKO_DIV values. In this table, SYS_CLK represents the system clock frequency.
<table>
<tr><th>MCKO_DIV[2:0]</th><th>MCKO Frequency</th></tr>
<tr><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>SYS_CLK/2</td></tr>
<tr><td>2</td><td>Reserved</td></tr>
<tr><td>3</td><td>SYS_CLK/4</td></tr>
<tr><td>4</td><td>Reserved</td></tr>
<tr><td>5</td><td>Reserved</td></tr>
<tr><td>6</td><td>Reserved</td></tr>
<tr><td>7</td><td>SYS_CLK/8</td></tr>
</table></td></tr>
<tr><td>25–1</td><td>—</td><td>Reserved.</td></tr>
<tr><td>0</td><td>PSTAT_EN</td><td>Processor status mode enable. Enables processor status (PSTAT) mode. In PSTAT mode, all auxiliary output port MDO pins are used to transmit processor status information, and Nexus messaging is unavailable.
0 PSTAT mode disabled
1 PSTAT mode enabled
Note: PSTAT mode is intended for factory processor debug only. The PSTAT_EN bit should be written to disable PSTAT mode by the customer. No Nexus messages are transmitted under any circumstances when PSTAT mode is enabled</td></tr>
</table>

25.7 NPC Functional Description

25.7.1 NPC Reset Configuration

The NPC is placed in disabled mode upon exit of reset. If message transmission via the auxiliary port is desired, a write to the PCR is then required to enable the NPC and select the mode of operation. Asserting MCKO_EN places the NPC in enabled mode and enables MCKO. The frequency of MCKO is selected by writing the MCKO_DIV field. Asserting or negating the FPM bit selects full-port or reduced-port mode, respectively.

Table 25-12 describes the NPC reset configuration options.

Table 25-12. NPC Reset Configuration Options

JCOMP Asserted?	PCR[MCKO_EN]	PCR[FPM]	Configuration
No	X	X	Reset
Yes	0	X	Disabled

Table 25-12. NPC Reset Configuration Options (continued)

JCOMP Asserted?	PCR[MCKO_EN]	PCR[FPM]	Configuration
Yes	1	1	Full-Port Mode
Yes	1	0	Reduced-Port Mode

25.7.2 Auxiliary Output Port

The auxiliary output port is shared by each of the Nexus modules on the device. The NPC communicates with each of the individual modules and arbitrates for access to the port. Additional information about the auxiliary port is found in Section 25.2, "External Signal Description."

25.7.2.1 Output Message Protocol

The protocol for transmitting messages via the auxiliary port is accomplished with the $\overline{\text{MSEO}}$ functions. The $\overline{\text{MSEO}}$ pins are used to signal the end of variable-length packets and the end of messages. They are not required to indicate the end of fixed-length packets. MDO and $\overline{\text{MSEO}}$ are sampled on the rising edge of MCKO.

Figure 25-5 illustrates the state diagram for $\overline{\text{MSEO}}$ transfers. All transitions not included in the figure are reserved, and must not be used.

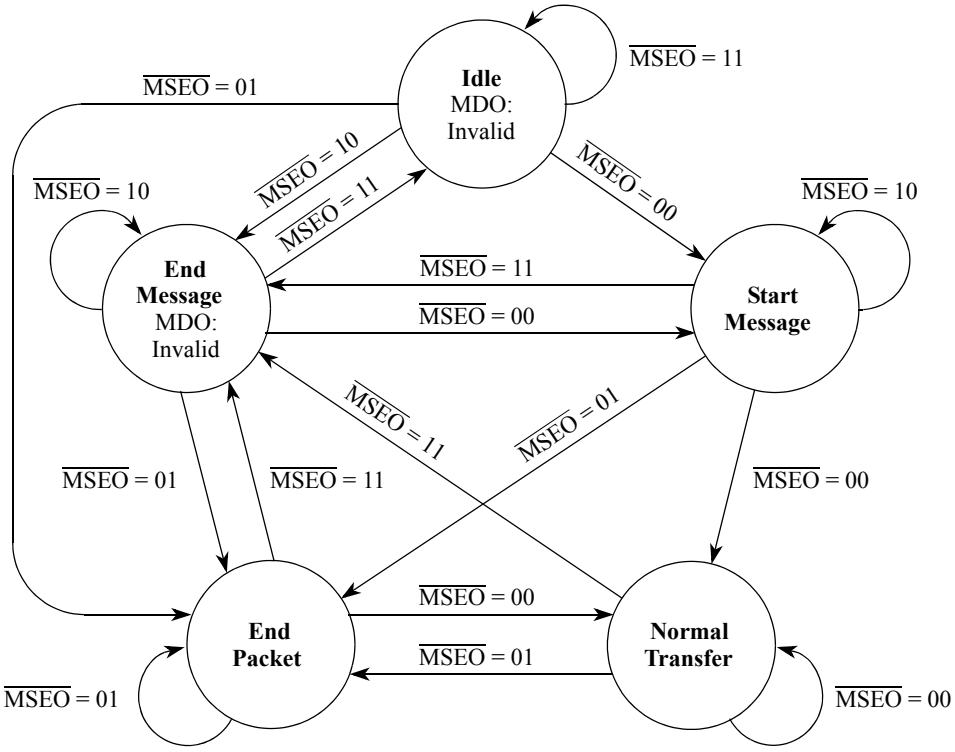


Figure 25-5. $\overline{\text{MSEO}}$ Transfers

25.7.2.2 Output Messages

In addition to sending out messages generated in other Nexus modules, the NPC can also output the device ID message contained in the device ID register on the MDO pins. The device ID message can also be sent out serially through TDO.

Table 25-13 describes the device ID message that the NPC can transmit on the auxiliary port. The TCODE is the first packet transmitted.

Table 25-13. NPC Output Messages

Message Name	Min. Packet Size (bits)	Max Packet Size (bits)	Packet Type	Packet Name	Packet Description
Device ID Message	6	6	Fixed	TCODE	Value = 1
	32	32	Fixed	ID	DID register contents

Figure 25-6 shows the various message formats that the pin interface formatter has to encounter.

Figure 25-6. Message Field Sizes

Message	TCODE	Field #1	Field #2	Field #3	Field #4	Field #5	Min. Size[1] (bits)	Max Size[2] (bits)
Device ID Message	1	Fixed = 32	NA	NA	NA	NA	38	38

[1] Minimum information size. The actual number of bits transmitted depends on the number of MDO pins

[2] Maximum information size. The actual number of bits transmitted depends on the number of MDO pins

The double edges in Figure 25-6 indicate the starts and ends of messages. Fields without shaded areas between them are grouped into super-fields and can be transmitted together without end-of-packet indications between them.

25.7.2.2.1 Rules of Messages

The rules of messages include the following:

- A variable-sized field within a message must end on a port boundary. (Port boundaries depend on the number of MDO pins active with the current reset configuration.)
- A variable-sized field may start within a port boundary only when following a fixed-length field.
- Super-fields must end on a port boundary.
- When a variable-length field is sized such that it does not end on a port boundary, it is necessary to extend and zero fill the remaining bits after the highest order bit so that it can end on a port boundary.
- Multiple fixed-length packets may start and/or end on a single clock.
- When any packet follows a variable-length packet, it must start on a port boundary.
- The field containing the TCODE number is always transferred out first, followed by subsequent fields of information.
- Within a field, the lowest significant bits are shifted out first. Figure 25-7 shows the transmission sequence of a message that is made up of a TCODE followed by three fields.

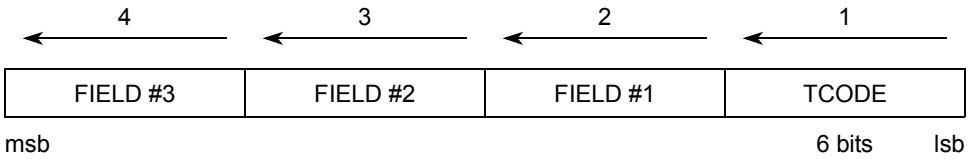


Figure 25-7. Transmission Sequence of Messages

25.7.2.3 IEEE® 1149.1-2001 (JTAG) TAP

The NPC uses the IEEE® 1149.1-2001 TAP for accessing registers. Each of the individual Nexus modules on the device implements a TAP controller for accessing its registers as well. TAP signals include TCK, TDI, TMS, and TDO. Detailed information about the TAP controller state machine may be found in Section 24.4.3, "TAP Controller State Machine."

The IEEE® 1149.1-2001 specification may be ordered for further detail on electrical and pin protocol compliance requirements.

The NPC implements a Nexus controller state machine that transitions based on the state of the IEEE® 1149.1-2001 state machine shown in Figure 25-5. The Nexus controller state machine is defined by the IEEE-ISTO 5001-2003 standard. It is shown in Figure 25-9.

The instructions implemented by the NPC TAP controller are listed in Table 25-14. The value of the NEXUS-ENABLE instruction is 0b0000. Each unimplemented instruction acts like the BYPASS instruction. The size of the NPC instruction register is 4-bits.

Table 25-14. Implemented Instructions

Instruction Name	Private/Public	Opcode	Description
NEXUS-ENABLE	Public	0x0	Activate Nexus controller state machine to read and write NPC registers.
BYPASS	Private	0xF	NPC BYPASS instruction. Also the value loaded into the NPC IR upon exit of reset.

Data is shifted between TDI and TDO starting with the least significant bit as illustrated in Figure 25-8. This applies for the instruction register and all Nexus tool-mapped registers.

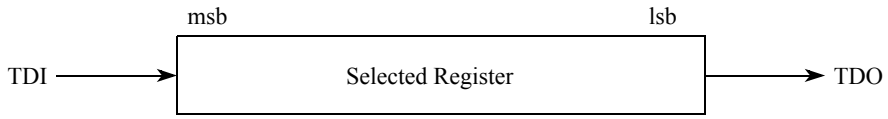


Figure 25-8. Shifting Data Into a Register

25.7.2.3.1 Enabling the NPC TAP Controller

Assertion of the power-on reset signal, entry into censored mode, or negating JCOMP resets the NPC TAP controller. When not in power-on reset or censored mode, the NPC TAP controller is enabled by asserting JCOMP and loading the ACCESS_AUX_TAP_NPC instruction in the JTAGC. Loading the NEXUS-ENABLE instruction then grants access to NPC registers.

25.7.2.3.2 Retrieving Device IDCODE

The Nexus TAP controller does not implement the IDCODE instruction. However, the device identification message can be output by the NPC through the auxiliary output port or shifted out serially by accessing the NPC device ID register through the TAP. If the NPC is enabled, transmission of the device identification message on the auxiliary output port MDO pins occurs immediately after a write to the PCR. Transmission of the device identification message serially through TDO is achieved by performing a read of the register contents as described in Section 25.7.2.3.4.

25.7.2.3.3 Loading NEXUS-ENABLE Instruction

Access to the NPC registers is enabled by loading the NPC NEXUS-ENABLE instruction when NPC has ownership of the TAP. This instruction is shifted in via the SELECT-IR-SCAN path and loaded in the UPDATE-IR state. At this point, the Nexus controller state machine, shown in Figure 25-9, transitions to the REG_SELECT state. The Nexus controller has three states: idle, register select, and data access. Table 25-15 illustrates the IEEE® 1149.1 sequence to load the NEXUS-ENABLE instruction.

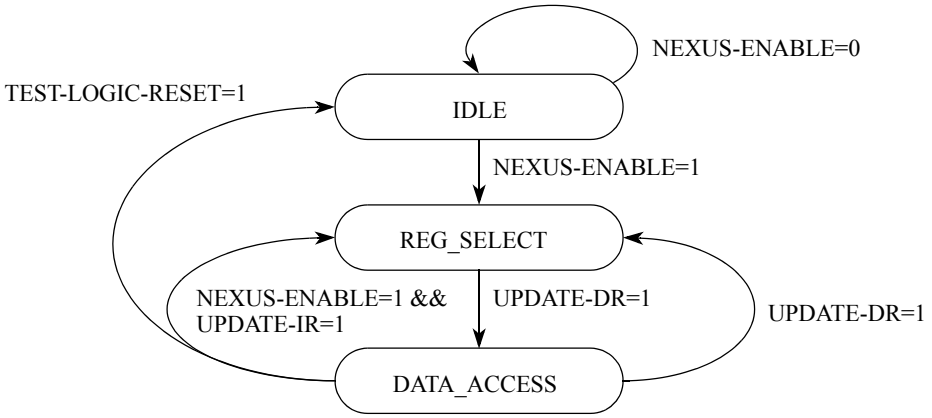


Figure 25-9. NEXUS Controller State Machine

Table 25-15. Loading NEXUS-ENABLE Instruction

Clock	TDI	TMS	IEEE® 1149.1 State	Nexus State	Description
0	—	0	RUN-TEST/IDLE	IDLE	IEEE 1149.1-2001 TAP controller in idle state
1	—	1	SELECT-DR-SCAN	IDLE	Transitional state
2	—	1	SELECT-IR-SCAN	IDLE	Transitional state
3	—	0	CAPTURE-IR	IDLE	Internal shifter loaded with current instruction
4	—	0	SHIFT-IR	IDLE	TDO becomes active, and the IEEE® 1149.1-2001 shifter is ready. Shift in all but the last bit of the NEXUS_ENABLE instruction.
5-7	0	0	3 TCKS in SHIFT-IR	IDLE	
8	0	1	EXIT1-IR	IDLE	Last bit of instruction shifted in
9	—	1	UPDATE-IR	IDLE	NEXUS-ENABLE loaded into instruction register
10	—	0	RUN-TEST/IDLE	REG_SELECT	Ready to be read/write Nexus registers

25.7.2.3.4 Selecting a Nexus Client Register

When the NEXUS-ENABLE instruction is decoded by the TAP controller, the input port allows development tool access to all Nexus registers. Each register has a 7-bit address index.

All register access is performed via the SELECT-DR-SCAN path of the IEEE® 1149.1–2001 TAP controller state machine. The Nexus controller defaults to the REG_SELECT state when enabled. Accessing a register requires two passes through the SELECT-DR-SCAN path: one pass to select the register and the second pass to read/write the register.

The first pass through the SELECT-DR-SCAN path is used to enter an 8-bit Nexus command consisting of a read/write control bit in the lsb followed by a 7-bit register address index, as illustrated in Figure 25-10. The read/write control bit is set to 1 for writes and 0 for reads.

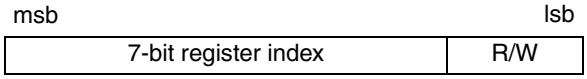


Figure 25-10. IEEE® 1149.1 Controller Command Input

The second pass through the SELECT-DR-SCAN path is used to read or write the register data by shifting in the data (lsb first) during the SHIFT-DR state. When reading a register, the register value is loaded into the IEEE® 1149.1-2001 shifter during the CAPTURE-DR state. When writing a register, the value is loaded from the IEEE® 1149.1-2001 shifter to the register during the UPDATE-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated once the required number of bits have been acquired.

Table 25-16 illustrates a sequence that writes a 32-bit value to a register.

Table 25-16. Write to a 32-Bit Nexus Client Register

Clock	TMS	IEEE 1149.1 State	Nexus State	Description
0	0	RUN-TEST/IDLE	REG_SELECT	IEEE 1149.1-2001 TAP controller in idle state
1	1	SELECT-DR-SCAN	REG_SELECT	First pass through SELECT-DR-SCAN path
2	0	CAPTURE-DR	REG_SELECT	Internal shifter loaded with current value of controller command input.
3	0	SHIFT-DR	REG_SELECT	TDO becomes active, and write bit and 6 bits of register index shifted in.
7 TCKs				
11	1	EXIT1-DR	REG_SELECT	Last bit of register index shifted into TDI
12	1	UPDATE-DR	REG_SELECT	Controller decodes and selects register
13	1	SELECT-DR-SCAN	DATA_ACCESS	Second pass through SELECT-DR-SCAN path
14	0	CAPTURE-DR	DATA_ACCESS	Internal shifter loaded with current value of register
15	0	SHIFT-DR	DATA_ACCESS	TDO becomes active, and outputs current value of register while new value is shifted in through TDI
31 TCKs				
47	1	EXIT1-DR	DATA_ACCESS	Last bit of current value shifted out TDO. Last bit of new value shifted in TDI.
48	1	UPDATE-DR	DATA_ACCESS	Value written to register
49	0	RUN-TEST/IDLE	REG_SELECT	Controller returned to idle state. It could also return to SELECT-DR-SCAN to write another register.

25.7.2.4 Nexus Auxiliary Port Sharing

Each of the Nexus modules on the MCU implements a request/grant scheme to arbitrate for control of the Nexus auxiliary port when Nexus data is ready to be transmitted.

All modules arbitrating for the port are given fixed priority levels relative to each other. If multiple modules have the same request level, this priority level is used as a tie-breaker. To avoid monopolization of the port, the module given the highest priority level alternates following each grant. Immediately out of reset the order of priority, from highest to lowest, is: NPC, NZ6C3, NDEDI, and NXDM. This arbitration mechanism is controlled internally and is not programmable by tools or the user.

25.7.2.5 Nexus JTAG Port Sharing

Each of the individual Nexus modules on the device implements a TAP controller for accessing its registers. When JCOMP is asserted, only the module whose ACCESS_AUX_TAP instruction is loaded has control of the TAP (See Section 24.4.4, "JTAGC Instructions"). This allows the interface to all of these individual TAP controllers to appear to be a single port from outside the device. Once a Nexus module has ownership of the TAP, that module acts like a single-bit shift register, or bypass register, if no register is selected as the shift path.

25.7.2.6 MCKO

MCKO is an output clock to the development tools used for the timing of $\overline{\text{MSEO}}$ and MDO pin functions. MCKO is derived from the system clock and its frequency is determined by the value of the MCKO_DIV[2:0] field in the PCR. Possible operating frequencies include one-half, one-quarter, and one-eighth system clock speed. MCKO is enabled by setting the MCKO_EN bit in the PCR.

The NPC also controls dynamic MCKO clock gating when in full- or reduced-port modes. The setting of the MCKO_GT bit inside the PCR determines whether or not MCKO gating control is enabled. The MCKO_GT bit resets to a logic 0. In this state gating of MCKO is disabled. To enable gating of MCKO, the MCKO_GT bit in the PCR is written to a logic 1. When MCKO gating is enabled, MCKO is driven to a logic 0 if the auxiliary port is enabled but not transmitting messages and there are no pending messages from Nexus clients.

25.7.2.7 $\overline{\text{EVTO}}$ Sharing

The NPC controls sharing of the $\overline{\text{EVTO}}$ output between all Nexus clients that produce an $\overline{\text{EVTO}}$ signal. $\overline{\text{EVTO}}$ is driven for one MCKO period whenever any module drives its $\overline{\text{EVTO}}$. When there is no active MCKO, such as in disabled mode, the NPC assumes an MCKO frequency of one-half system clock speed when driving $\overline{\text{EVTO}}$. $\overline{\text{EVTO}}$ sharing is active as long as the NPC is not in reset.

25.7.2.8 Nexus Reset Control

The JCOMP input that is used as the primary reset signal for the NPC is also used by the NPC to generate a single-bit reset signal for other Nexus modules. If JCOMP is negated, an internal reset signal is asserted, indicating that all Nexus modules should be held in reset. This internal reset signal is also asserted during a power-on reset, or if nex_disable is asserted, indicating the device is in censored mode. This single bit reset signal functions much like the IEEE® 1149.1-2001 defined $\overline{\text{TRST}}$ signal and allows JCOMP reset information to be provided to the Nexus modules without each module having to sense the JCOMP signal directly or monitor the status of censored mode.

25.8 NPC Initialization/Application Information

25.8.1 Accessing NPC Tool-Mapped Registers

To initialize the TAP for NPC register accesses, the following sequence is required:

1. Enable the NPC TAP controller. This is achieved by asserting JCOMP and loading the ACCESS_AUX_TAP_NPC instruction in the JTAGC.
2. Load the TAP controller with the NEXUS-ENABLE instruction.

To write control data to NPC tool-mapped registers, the following sequence is required:

1. Write the 7-bit register index and set the write bit to select the register with a pass through the SELECT-DR-SCAN path in the TAP controller state machine.
2. Write the register value with a second pass through the SELECT-DR-SCAN path. Note that the prior value of this register is shifted out during the write.

To read status and control data from NPC tool-mapped registers, the following sequence is required:

1. Write the 7-bit register index and clear the write bit to select register with a pass through SELECT-DR-SCAN path in the TAP controller state machine.
2. Read the register value with a second pass through the SELECT-DR-SCAN path. Data shifted in is ignored.

See the IEEE®-ISTO 5001-2003 standard for more detail.

25.9 Nexus Dual eTPU Development Interface (NDEDI)

The enhanced timing processor unit (eTPU) has its own Nexus class 3 interface, the Nexus dual eTPU development interface (NDEDI). The two (MPC5554) eTPU engines and a coherent dual parameter controller (CDC) appear as three separate Nexus clients. Refer to the *Enhanced Time Processor Unit Reference Manual* for more information about the NDEDI module.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PRN				DC						PIN					
W																
Reset for MPC5553	0	0	0	1	1	0	0	0	0	0	0	1	0	0	1	0
Reset for MPC5554	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
Reg Index	0															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PIN				MIC											1
W																
Reset for MPC5553	0	1	0	0	0	0	0	0	0	0	0	1	1	1	0	1
Reset for MPC5554	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1
Reg Index	0															

Figure 25-11. NDEDI Device ID Register (DID)

Table 25-17. NDEDI DID Register Field Descriptions

Bits	Name	Description
31–28	PRN	Part revision number. Contains the revision number of the part. This field changes with each revision of the device or module.
27–22	DC	Design center. Indicates the Freescale design center. For both the MPC5554 and MPC5553, this value is 0x20.
21–12	PIN	Part identification number. Contains the part number of the device.
11–1	MIC	Manufacturer identity code. Contains the reduced Joint Electron Device Engineering Council (JEDEC) ID for Freescale, 0xE.
0	—	Fixed per JTAG 1149.1 1 Always set

25.10 e200z6 Class 3 Nexus Module (NZ6C3)

The NZ6C3 module provides real-time development capabilities for the MPC5553/MPC5554 core in compliance with the IEEE®-ISTO Nexus 5001-2003 standard. This module provides development support capabilities without requiring the use of address and data pins for internal visibility.

25.10.1 Introduction

This section defines the auxiliary pin functions, transfer protocols and standard development features of the NZ6C3 module. The development features supported are Program trace, data trace, watchpoint messaging, ownership trace, and read/write access via the JTAG interface.

NOTE

Throughout this section references are made to the auxiliary port and its specific signals, such as MCKO, MSEO[0:1], MDO[11:0] and others. In actual use the MPC5553/MPC5554 NPC module arbitrates the access of the single auxiliary port. To simplify the description of the function of the NZ6C3 module, the interaction of the NPC is omitted and the behavior described as if the module has its own dedicated auxiliary port. The auxiliary port is fully described in Section 25.2, "External Signal Description," on page 25-5.

25.10.2 Block Diagram

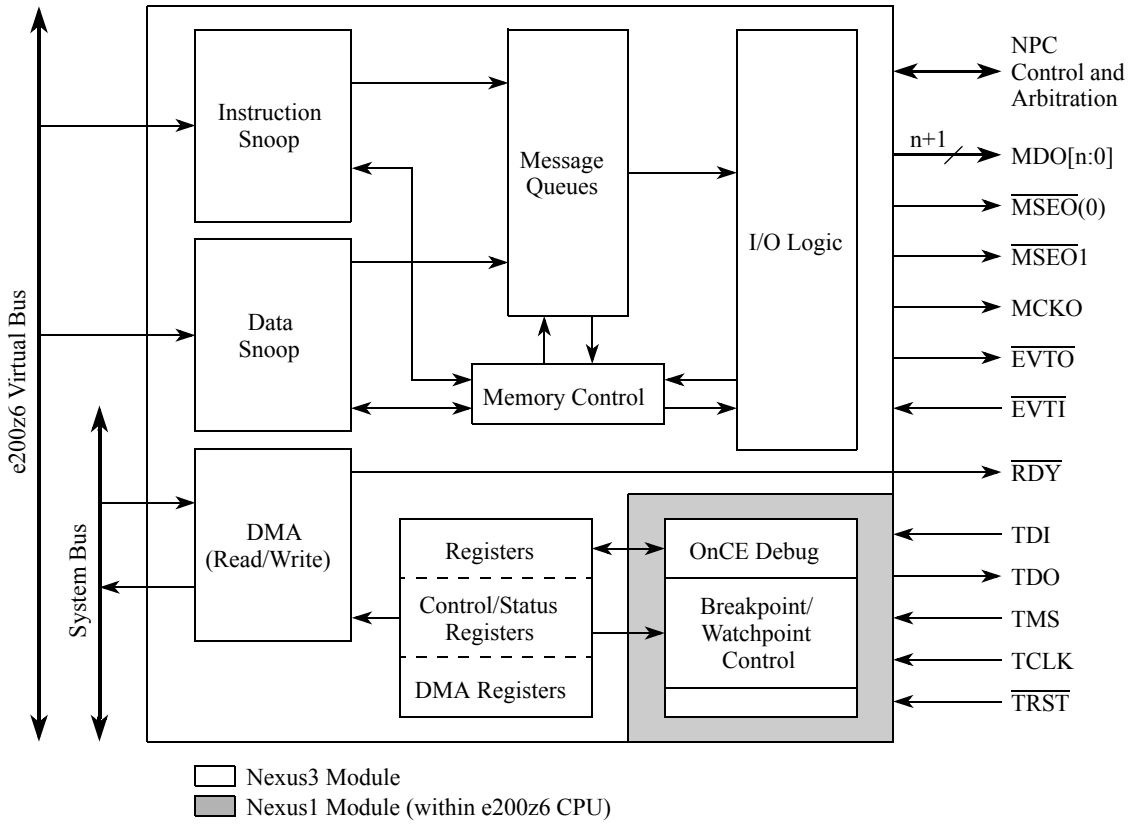


Figure 25-12. e200z6 Nexus3 Functional Block Diagram

25.10.3 Overview

Table 25-18 contains a set of terms and definitions associated with the NZ6C3 module.

Table 25-18. Terms and Definitions

Term	Description
IEEE®-ISTO 5001	Consortium and standard for real-time embedded system design. World wide Web documentation at http://www.ieee-isto.org/Nexus5001
Auxiliary Port	Refers to Nexus auxiliary port. Used as auxiliary port to the IEEE® 1149.1 JTAG interface.
Branch Trace Messaging (BTM)	Visibility of addresses for taken branches and exceptions, and the number of sequential instructions executed between each taken branch.
Client	A functional block on an embedded processor which requires development visibility and controllability. Examples are a central processing unit (CPU) or an intelligent peripheral.
Data Read Message (DRM)	External visibility of data reads to memory-mapped resources.
Data Write Message (DWM)	External visibility of data writes to memory-mapped resources.
Data Trace Messaging (DTM)	External visibility of how data flows through the embedded system. This may include DRM and/or DWM.
JTAG Compliant	Device complying to IEEE® 1149.1 JTAG standard
JTAG IR & DR Sequence	JTAG instruction register (IR) scan to load an opcode value for selecting a development register. The JTAG IR corresponds to the OnCE command register (OCMD). The selected development register is then accessed via a JTAG data register (DR) scan.
Nexus1	The e200z6 (OnCE) debug module. This module integrated with each e200z6 processor provides all static (core halted) debug functionality. This module is compliant with Class1 of the IEEE®-ISTO 5001 standard.
Ownership Trace Message (OTM)	Visibility of process/function that is currently executing.
Public Messages	Messages on the auxiliary pins for accomplishing common visibility and controllability requirements
Standard	The phrase ‘according to the standard’ is used to indicate according to the IEEE®-ISTO 5001 standard.
Transfer Code (TCODE)	Message header that identifies the number and/or size of packets to be transferred, and how to interpret each of the packets.
Watchpoint	A data or instruction breakpoint which does not cause the processor to halt. Instead, a pin is used to signal that the condition occurred. A watchpoint message is also generated.

25.10.4 Features

The NZ6C3 module is compliant with Class 3 of the IEEE®-ISTO 5001-2003 standard. The following features are implemented:

- Program trace via branch trace messaging (BTM). Branch trace messaging displays program flow discontinuities (direct and indirect branches, exceptions, etc.), allowing the development tool to interpolate what transpires between the discontinuities. Thus static code may be traced.

- Data trace via data write messaging (DWM) and data read messaging (DRM). This provides the capability for the development tool to trace reads and/or writes to selected internal memory resources.
- Ownership trace via ownership trace messaging (OTM). OTM facilitates ownership trace by providing visibility of which process ID or operating system task is activated. An ownership trace message is transmitted when a new process/task is activated, allowing the development tool to trace ownership flow.
- Run-time access to embedded processor registers and memory map via the JTAG port. This allows for enhanced download/upload capabilities.
- Watchpoint messaging via the auxiliary pins.
- Watchpoint trigger enable of program and/or data trace messaging.
- Higher speed data input/output via the auxiliary port.
- Registers for program trace, data trace, ownership trace and watchpoint trigger.
- All features controllable and configurable via the JTAG port.

25.10.5 Enabling Nexus3 Operation

The Nexus module is enabled by loading a single instruction (ACCESS_AUX_TAP_ONCE, as shown in Table 25-4) into the JTAGC instruction register (IR), and then loading the corresponding OnCE OCMD register with the NEXUS3_ACCESS instruction (refer to Table 25-5). For the e200z6 Class 3 Nexus module, the OCMD value is 0b00_0111_1100. Once enabled, the module will be ready to accept control input via the JTAG pins. See Section 25.4.1, “Enabling Nexus Clients for TAP Access” for more information.

The Nexus module is disabled when the JTAG state machine reaches the test-logic-reset state. This state can be reached by the assertion of the JCOMP pin or by cycling through the state machine using the TMS pin. The Nexus module will also be disabled if a power-on-reset (POR) event occurs. If the Nexus3 module is disabled, no trace output will be provided, and the module will disable (drive inactive) auxiliary port output pins MDO[n:0], MSEO[1:0], MCKO. Nexus registers will not be available for reads or writes.

25.10.6 TCODEs Supported by NZ63C

The Nexus3 pins allow for flexible transfer operations via public messages. A TCODE defines the transfer format, the number and/or size of the packets to be transferred, and the purpose of each packet. The IEEE®-ISTO 5001-2003 standard defines a set of public messages. The NZ6C3 module supports the public TCODEs seen in Table 25-19. Each message contains multiple packets transmitted in the order shown in the table.

Table 25-19. Public TCODEs Supported by NZ63C

Message Name	Packet Size (bits)		Packet Name	Packet Type	Packet Description
	Min	Max			
Debug Status	6	6	TCODE	Fixed	TCODE number = 0 (0x00)
	4	4	SRC	Fixed	source processor identifier
	8	8	STATUS	Fixed	Debug status register (DS[31:24])

Table 25-19. Public TCODEs Supported by NZ63C (continued)

Message Name	Packet Size (bits)		Packet Name	Packet Type	Packet Description
	Min	Max			
Ownership Trace Message	6	6	TCODE	Fixed	TCODE number = 2 (0x02)
	4	4	SRC	Fixed	source processor identifier
	32	32	PROCESS	Fixed	Task/Process ID tag
Program Trace - Direct Branch Message[1]	6	6	TCODE	Fixed	TCODE number = 3 (0x03)
	4	4	SRC	Fixed	source processor identifier
	1	8	I-CNT	Variable	# sequential instructions executed since last taken branch
Program Trace - Indirect Branch Message[1]	6	6	TCODE	Fixed	TCODE number = 4 (0x04)
	4	4	SRC	Fixed	source processor identifier
	1	8	I-CNT	Variable	# sequential instructions executed since last taken branch
	1	32	U-ADDR	Variable	unique part of target address for taken branches/exceptions
Data Trace - Data Write Message	6	6	TCODE	Fixed	TCODE number = 5 (0x05)
	4	4	SRC	Fixed	source processor identifier
	3	3	DSIZ	Fixed	data size (Refer to Table 25-23)
	1	32	U-ADDR	Variable	unique portion of the data write address
	1	64	DATA	Variable	data write values (see Section 25.11.13, “Data Trace,” for details)
Data Trace - Data Read Message	6	6	TCODE	Fixed	TCODE number = 6 (0x06)
	4	4	SRC	Fixed	source processor identifier
	3	3	DSIZ	Fixed	data size (Refer to Table 25-23)
	1	32	U-ADDR	Variable	unique portion of the data read address
	1	64	DATA	Variable	data read values (see Section 25.11.13, “Data Trace,” for details)
Error Message	6	6	TCODE	Fixed	TCODE number = 8 (0x08)
	4	4	SRC	Fixed	source processor identifier
	5	5	ECODE	Fixed	error code
Program Trace - Direct Branch Message w/ Sync[1]	6	6	TCODE	Fixed	TCODE number = 11 (0x0B)
	4	4	SRC	Fixed	source processor identifier
	1	8	I-CNT	Variable	# sequential instructions executed since last taken branch
	1	32	F-ADDR	Variable	full target address (leading zeros truncated)

Table 25-19. Public TCODEs Supported by NZ63C (continued)

Message Name	Packet Size (bits)		Packet Name	Packet Type	Packet Description
	Min	Max			
Program Trace - Indirect Branch Message w/ Sync[1]	6	6	TCODE	Fixed	TCODE number = 12 (0x0C)
	4	4	SRC	Fixed	source processor identifier
	1	8	I-CNT	Variable	# sequential instructions executed since last taken branch
	1	32	F-ADDR	Variable	full target address (leading zeros truncated)
Data Trace - Data Write Message w/ Sync	6	6	TCODE	Fixed	TCODE number = 13 (0x0D)
	4	4	SRC	Fixed	source processor identifier
	3	3	DSZ	Fixed	data size (Refer to Table 25-23)
	1	32	F-ADDR	Variable	full access address (leading zeros truncated)
	1	64	DATA	Variable	data write values (see Section 25.11.13, “Data Trace,” for details)
Data Trace - Data Read Message w/ Sync	6	6	TCODE	Fixed	TCODE number = 14 (0x0E)
	4	4	SRC	Fixed	source processor identifier
	3	3	DSZ	Fixed	data size (Refer to Table 25-23)
	1	32	F-ADDR	Variable	full access address (leading zeros truncated)
	1	64	DATA	Variable	data read values (see Section 25.11.13, “Data Trace,” for details)
Watchpoint Message	6	6	TCODE	Fixed	TCODE number = 15 (0x0F)
	4	4	SRC	Fixed	source processor identifier
	4	4	WPHIT	Fixed	# indicating watchpoint sources
Resource Full Message	6	6	TCODE	Fixed	TCODE number = 27 (0x1B)
	4	4	SRC	Fixed	source processor identifier
	4	4	RCODE	Fixed	resource code (Refer to RCODE values in Table 25-21) - indicates which resource is the cause of this message
	1	32	HIST	Variable	branch / predicate instruction history (see Section 25.11.12.1, “Branch Trace Messaging (BTM)”)
Program Trace - Indirect Branch History Message	6	6	TCODE	Fixed	TCODE number = 28 (0x1C) (see footnote 1 below)
	4	4	SRC	Fixed	source processor identifier
	1	8	I-CNT	Variable	# sequential instructions executed since last taken branch
	1	32	U-ADDR	Variable	unique part of target address for taken branches/exceptions
	1	32	HIST	Variable	branch / predicate instruction history (see Section 25.11.12.1, “Branch Trace Messaging (BTM)”)

Table 25-19. Public TCODEs Supported by NZ63C (continued)

Message Name	Packet Size (bits)		Packet Name	Packet Type	Packet Description
	Min	Max			
Program Trace - Indirect Branch History Message w/ Sync	6	6	TCODE	Fixed	TCODE number = 29 (0x1D) (see footnote 1 below)
	4	4	SRC	Fixed	source processor identifier
	1	8	I-CNT	Variable	# sequential instructions executed since last taken branch
	1	32	F-ADDR	Variable	full target address (leading zero (0) truncated)
	1	32	HIST	Variable	branch / predicate instruction history (see Section 25.11.12.1, “Branch Trace Messaging (BTM)”)
Program Trace - Program Correlation Message	6	6	TCODE	Fixed	TCODE number = 33 (0x21)
	4	4	SRC	Fixed	source processor identifier
	4	4	EVCODE	Fixed	event correlated w/ program flow (Refer to Table 25-22)
	1	8	I-CNT	Variable	# sequential instructions executed since last taken branch
	1	32	HIST	Variable	branch / predicate instruction history (see Section 25.11.12.1, “Branch Trace Messaging (BTM)”)

1 The user can select between the two types of program trace. The advantages for each are discussed in Section 25.11.12.1, “Branch Trace Messaging (BTM). If the branch history method is selected, the shaded TCODES above will not be messaged out.

Table 25-20 shows the error code encodings used when reporting an error via the Nexus3 Error Message.

Table 25-20. Error Code Encoding (TCODE = 8)

Error Code (ECODE)	Description
00000	Ownership trace overrun
00001	Program trace overrun
00010	Data trace overrun
00011	Read/write access error
00101	Invalid access opcode (Nexus register unimplemented)
00110	Watchpoint overrun
00111	(Program trace or data trace) and ownership trace overrun
01000	(Program trace or data trace or ownership trace) and watchpoint overrun
01001–0111	Reserved
11000	BTM lost due to collision w/ higher priority message
11001–11111	Reserved

Table 25-21 shows the encodings used for resource codes for certain messages.

Table 25-21. RCODE values (TCODE = 27)

Resource Code (RCODE)	Description
0001	Program trace, branch / predicate instruction history. This type of packet is terminated by a stop bit set to 1 after the last history bit.

Table 25-22 shows the event code encodings used for certain messages.

Table 25-22. Event Code Encoding (TCODE = 33)

Event Code (EVCODE)	Description
0000	Entry into debug mode
0001	Entry into low power mode (CPU only)
0010–1111	Reserved for future functionality

Table 25-23 shows the data trace size encodings used for certain messages.

Table 25-23. Data Trace Size Encodings (TCODE = 5, 6, 13, 14)

DTM Size Encoding	Transfer Size
000	Byte
001	Half-word (2 bytes)
010	Word (4 bytes)
011	Double-word (8 bytes)
100	String (3 bytes)
101–111	Reserved

25.11 NZ6C3 Memory Map/Register Definition

This section describes the NZ6C3 programmer's model. NZ6C3 registers are accessed using the JTAG/OnCE port in compliance with IEEE® 1149.1. See Section 25.11.10, " NZ6C3 Register Access via JTAG / OnCE" for details on NZ6C3 register access.

NOTE

NZ6C3 registers and output signals are numbered using bit 0 as the least significant bit. This bit ordering is consistent with the ordering defined by the IEEE®-ISTO 5001 standard.

Table 25-24 details the register map for the NZ6C3 module.

Table 25-24. NZ6C3 Memory Map

Access Opcode	Register Name	Register Description	Read Address	Write Address
0x1	CSC	Client select control[1]	0x02	—
See NPC	PCR	Port configuration register[1]	—	—

Table 25-24. NZ6C3 Memory Map (continued)

Access Opcode	Register Name	Register Description	Read Address	Write Address
0x2	DC1	Development control 1	0x04	0x05
0x3	DC2	Development control 2	0x06	0x07
0x4	DS	Development status	0x08	—
0x7	RWCS	Read/write access control/status	0x0E	0x0F
0x9	RWA	Read/write access address	0x12	0x13
0xA	RWD	Read/write access data	0x14	0x15
0xB	WT	Watchpoint trigger	0x16	0x17
0xD	DTC	Data trace control	0x1A	0x1B
0xE	DTSA1	Data trace start address 1	0x1C	0x1D
0xF	DTSA2	Data trace start address 2	0x1E	0x1F
0x12	DTEA1	Data trace end address 1	0x24	0x25
0x13	DTEA2	Data trace end address 2	0x26	0x27
0x14 -> 0x3F	—	Reserved	0x28->0x7E	0x29->0x7F

[1] The CSC and PCR registers are shown in this table as part of the Nexus programmer's model. They are only present at the top level Nexus3 controller (NPC), not in the NZ6C3 module. The device's CSC register is readable through Nexus3, but the PCR is shown for reference only.

25.11.1 Development Control Register 1, 2 (DC1, DC2)

The development control registers are used to control the basic development features of the NZ6C3 module. Development control register 1 is shown in Figure 25-13 and its fields are described in Table 25-25.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	OPC	MCK_DIV		EOC		0	PTM	WEN	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Nexus Reg	0x2															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	OVC			EIC		TM		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Nexus Reg	0x2															

Figure 25-13. Development Control Register 1 (DC1)

Table 25-25. DC1 Field Descriptions

Bits	Name	Description
31	OPC[1]	Output port mode control. 0 Reduced-port mode configuration (4 MDO pins) 1 Full-port mode configuration (12 MDO pins)
30–29	MCK_DIV [1:0][1]	MCKO clock divide ratio. 00 MCKO is 1x processor clock freq. 01 MCKO is 1/2x processor clock freq. 10 MCKO is 1/4x processor clock freq. 11 MCKO is 1/8x processor clock freq.
28–27	EOC [1:0]	EVTO control. 00 EVTO upon occurrence of watchpoints (configured in DC2) 01 EVTO upon entry into debug mode 10 EVTO upon timestamping event 11 Reserved
26	—	Reserved.
25	PTM	Program trace method. 0 Program trace uses traditional branch messages 1 Program trace uses branch history messages
24	WEN	Watchpoint trace enable. 0 Watchpoint Messaging disabled 1 Watchpoint Messaging enabled
23–8	—	Reserved.
7–5	OVC [2:0]	Overrun control. 000 Generate overrun messages 001–010 mReserved 011 Delay processor for BTM / DTM / OTM overruns 1XX Reserved
4–3	EIC [1:0]	EVTI control. 00 EVTI is used for synchronization (program trace/ data trace) 01 EVTI is used for debug request 1X Reserved
2–0	TM [2:0]	Trace mode. Any or all of the TM bits may set, enabling one or more traces. 000 No trace 1XX Program trace enabled X1X Data trace enabled XX1 Ownership trace enabled

[1] The output port mode control bit (OPC) and MCKO divide bits (MCK_DIV) are shown for clarity. These functions are controlled globally by the NPC port control register (PCR).

Development control register 2 is shown in Figure 25-14 and its fields are described in Table 25-26.

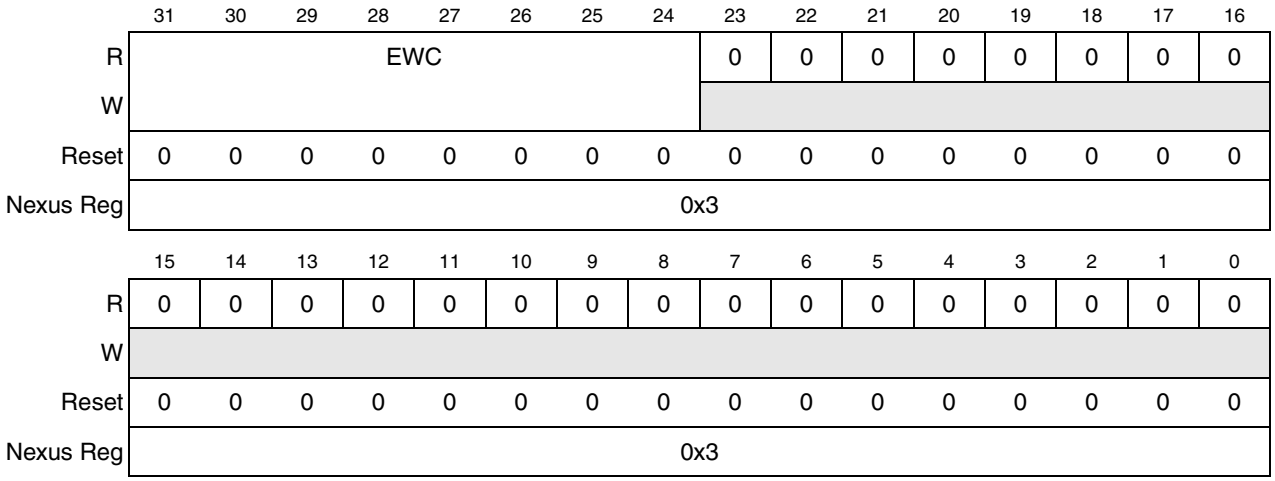


Figure 25-14. Development Control Register 2 (DC2)

Table 25-26. DC2 Field Descriptions

Bits	Name	Description
31–24	EWC [7:0]	$\overline{\text{EVTO}}$ watchpoint configuration. Any or all of the bits in EWC may be set to configure the $\overline{\text{EVTO}}$ watchpoint. 00000000No Watchpoints trigger $\overline{\text{EVTO}}$ 1XXXXXXXWatchpoint #0 (IAC1 from Nexus1) triggers $\overline{\text{EVTO}}$ X1XXXXXXWatchpoint #1 (IAC2 from Nexus1) triggers $\overline{\text{EVTO}}$ XX1XXXXXWatchpoint #2 (IAC3 from Nexus1) triggers $\overline{\text{EVTO}}$ XXX1XXXXWatchpoint #3 (IAC4 from Nexus1) triggers $\overline{\text{EVTO}}$ XXXX1XXXWatchpoint #4 (DAC1 from Nexus1) triggers $\overline{\text{EVTO}}$ XXXXX1XXWatchpoint #5 (DAC2 from Nexus1) triggers $\overline{\text{EVTO}}$ XXXXXX1XWatchpoint #6 (DCNT1 from Nexus1) triggers $\overline{\text{EVTO}}$ XXXXXXX1Watchpoint #7 (DCNT2 from Nexus1) triggers $\overline{\text{EVTO}}$
23–0	—	Reserved.

NOTE

The EOC bits in DC1 must be programmed to trigger $\overline{\text{EVTO}}$ on watchpoint occurrence for the EWC bits to have any effect.

25.11.2 Development Status Register (DS)

The development status register is used to report system debug status. When debug mode is entered or exited, or an e200z6-defined low power mode is entered, a debug status message is transmitted with DS[31:24]. The external tool can read this register at any time.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DBG	0	0	0	LPC		CHK	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Nexus Reg	0x4															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Nexus Reg	0x4															

Figure 25-15. Development Status Register (DS)

Table 25-27. DS Field Descriptions

Bits	Name	Description
31–28	DBG	e200z6 CPU debug mode status. 0 CPU not in debug mode 1 CPU in debug mode
27–26	LPC [1:0]	e200z6 CPU low power mode status. 00 Normal (run) mode 01 CPU in halted state 10 CPU in stopped state 11 Reserved
25	CHK	e200z6 CPU checkstop status. 0 CPU not in checkstop state 1 CPU in checkstop state
24–0	—	Reserved.

25.11.3 Read/Write Access Control/Status (RWCS)

The read write access control/status register provides control for read/write access. Read/write access provides DMA-like access to memory-mapped resources on the system bus either while the processor is halted, or during runtime. The RWCS register also provides read/write access status information as shown in Table 25-29.

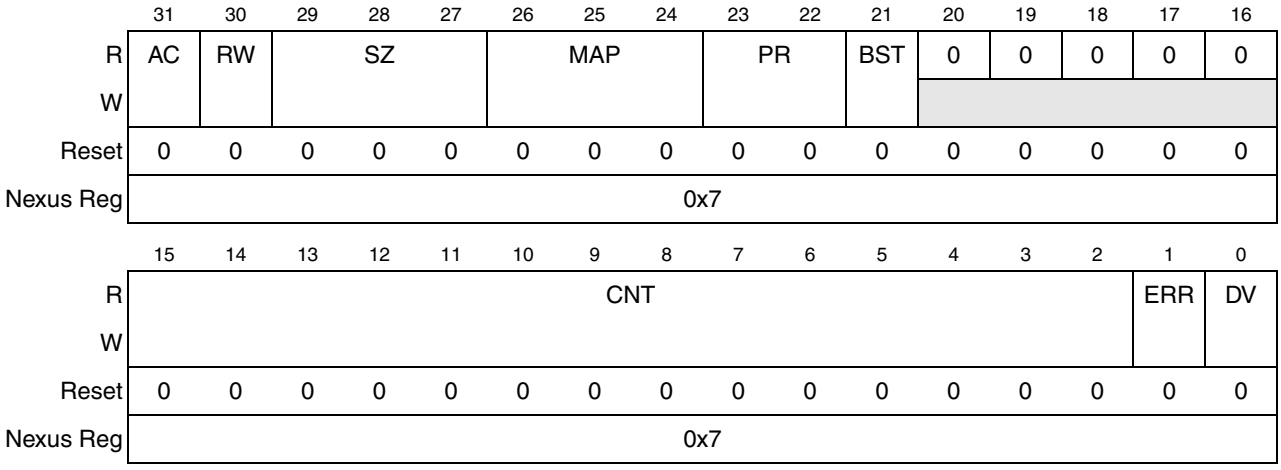


Figure 25-16. Read/Write Access Control/Status Register (RWCS)

Table 25-28. RWCS Field Description

Bits	Name	Description
31	AC	Access control. 0 End access 1 Start access
30	RW	Read/write select. 0 Read access 1 Write access
29–27	SZ [2:0]	Word size. 000 8-bit (byte) 001 6-bit (half-word) 010 32-bit (word) 011 64-bit (double-word - only in burst mode) 100–111 Reserved (default to word)
26–24	MAP [2:0]	MAP select. 000 Primary memory map 001-111 Reserved
23–22	PR [1:0]	Read/write access priority. 00 Lowest access priority 01 Reserved (default to lowest priority) 10 Reserved (default to lowest priority) 11 Highest access priority
21	BST	Burst control. 0 Module accesses are single bus cycle at a time. 1 Module accesses are performed as burst operation.
20–16	—	Reserved.
15–2	CNT [13:0]	Access control count. Number of accesses of word size SZ
1	ERR	Read/write access error. See Table 25-29.
0	DV	Read/write access data valid. See Table 25-29.

Table 25-29 details the status bit encodings.

Table 25-29. Read/Write Access Status Bit Encoding

Read Action	Write Action	ERR	DV
Read access has not completed	Write access completed without error	0	0
Read access error has occurred	Write access error has occurred	1	0
Read access completed without error	Write access has not completed	0	1
Not allowed	Not allowed	1	1

25.11.4 Read/Write Access Data (RWD)

The read/write access data register provides the data to/from system bus memory-mapped locations when initiating a read or a write access.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Read/Write Data															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Nexus Reg	0x9															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Read/Write Data															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Nexus Reg	0x9															

Figure 25-17. Read/Write Access Data Register (RWD)

25.11.5 Read/Write Access Address (RWA)

The read/write access address register provides the system bus address to be accessed when initiating a read or a write access.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Read/Write Address															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Nexus Reg	0xA															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Read/Write Address															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Nexus Reg	0xA															

Figure 25-18. Read/Write Access Address Register (RWA)

25.11.6 Watchpoint Trigger Register (WT)

The watchpoint trigger register allows the watchpoints defined within the e200z6 Nexus1 logic to trigger actions. These watchpoints can control program and/or data trace enable and disable. The WT bits can be used to produce an address related ‘window’ for triggering trace messages.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PTS			PTE			DTS			DTE			0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Nexus Reg	0xB															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Nexus Reg	0xB															

Figure 25-19. Watchpoint Trigger Register (WT)

Table 25-30 details the watchpoint trigger register fields.

Table 25-30. WT Field Descriptions

Bits	Name	Description
31–29	PTS [2:0]	Program trace start control. 000 Trigger disabled 001 Use watchpoint #0 (IAC1 from Nexus1) 010 Use watchpoint #1 (IAC2 from Nexus1) 011 Use watchpoint #2 (IAC3 from Nexus1) 100 Use watchpoint #3 (IAC4 from Nexus1) 101 Use watchpoint #4 (DAC1 from Nexus1) 110 Use watchpoint #5 (DAC2 from Nexus1) 111 Use watchpoint #6 or #7 (DCNT1 or DCNT2 from Nexus1)
28–26	PTE [2:0]	Program trace end control. 000 Trigger disabled 001 Use watchpoint #0 (IAC1 from Nexus1) 010 Use watchpoint #1 (IAC2 from Nexus1) 011 Use watchpoint #2 (IAC3 from Nexus1) 100 Use watchpoint #3 (IAC4 from Nexus1) 101 Use watchpoint #4 (DAC1 from Nexus1) 110 Use watchpoint #5 (DAC2 from Nexus1) 111 Use watchpoint #6 or #7 (DCNT1 or DCNT2 from Nexus1)

Table 25-30. WT Field Descriptions (continued)

Bits	Name	Description
25–23	DTS [2:0]	Data trace start control. 000 Trigger disabled 001 Use watchpoint #0 (IAC1 from Nexus1) 010 Use watchpoint #1 (IAC2 from Nexus1) 011 Use watchpoint #2 (IAC3 from Nexus1) 100 Use watchpoint #3 (IAC4 from Nexus1) 101 Use watchpoint #4 (DAC1 from Nexus1) 110 Use watchpoint #5 (DAC2 from Nexus1) 111 Use watchpoint #6 or #7 (DCNT1 or DCNT2 from Nexus1)
22–20	DTE [2:0]	Data trace end control. 000 Trigger disabled 001 Use watchpoint #0 (IAC1 from Nexus1) 010 Use watchpoint #1 (IAC2 from Nexus1) 011 Use watchpoint #2 (IAC3 from Nexus1) 100 Use watchpoint #3 (IAC4 from Nexus1) 101 Use watchpoint #4 (DAC1 from Nexus1) 110 Use watchpoint #5 (DAC2 from Nexus1) 111 Use watchpoint #6 or #7 (DCNT1 or DCNT2 from Nexus1)
19–0	—	Reserved.

NOTE

The WT bits will only control program/data trace if the TM bits in the development control register 1 (DC1) have not already been set to enable program and data trace, respectively.

25.11.7 Data Trace Control Register (DTC)

The data trace control register controls whether DTM messages are restricted to reads, writes, or both for a user programmable address range. There are two data trace channels controlled by the DTC for the Nexus3 module. Each channel can also be programmed to trace data accesses or instruction accesses.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RWT1		RWT2		0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Nexus Reg	0xD															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	RC1	RC2	0	0	DI1	DI2	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Nexus Reg	0xD															

Figure 25-20. Data Trace Control Register (DTC)

Table 25-31 details the data trace control register fields.

Table 25-31. DTC Field Description

Bits	Name	Description
31–30	RWT1 [1:0]	Read/write trace 1. 00 No trace enabled X1 Enable data read trace 1X Enable data write trace
29–28	RWT2 [1:0]	Read/write trace 2. 00 No trace enabled X1 Enable data read trace 1X Enable data write trace
27–8	—	Reserved.
7	RC1	Range control 1. 0 Condition trace on address within range 1 Condition trace on address outside of range
6	RC2	Range control 2 0 Condition trace on address within range 1 Condition trace on address outside of range
5–4	—	Reserved.
3	DI1	Data access/instruction access trace 1. 0 Condition trace on data accesses 1 Condition trace on instruction accesses
2	DI2	Data access/instruction access trace 2 0 Condition trace on data accesses 1 Condition trace on instruction accesses
1–0	—	Reserved.

25.11.8 Data Trace Start Address Registers 1 and 2 (DTSA*n*)

The data trace start address registers define the start addresses for each trace channel.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Data Trace Start Address															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Nexus Reg	0xE															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Data Trace Start Address															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Nexus Reg	0xE															

Figure 25-21. Data Trace Start Address Register 1 (DTSA1)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Data Trace Start Address															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Nexus Reg	0xF															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Data Trace Start Address															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Nexus Reg	0xF															

Figure 25-22. Data Trace Start Address Register 2 (DTSA2)

25.11.9 Data Trace End Address Registers 1 and 2 (DTEA*n*)

The data trace end address registers define the end addresses for each trace channel.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Data Trace End Address															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Nexus Reg	0x12															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Data Trace End Address															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Nexus Reg	0x12															

Figure 25-23. Data Trace End Address Register 1 (DTEA1)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Data Trace End Address															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Nexus Reg	0x13															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Data Trace End Address															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Nexus Reg	0x13															

Figure 25-24. Data Trace End Address Register 2 (DTEA2)

Table 25-32 illustrates the range that will be selected for data trace for various cases of DTSA being less than, greater than, or equal to DTEA.

Table 25-32. Data Trace—Address Range Options

Programmed Values	Range Control Bit Value	Range Selected
DTSA < DTEA	0	DTSA -> <- DTEA
DTSA < DTEA	1	<- DTSA DTEA ->
DTSA > DTEA	N/A	Invalid range—no trace
DTSA = DTEA	N/A	Invalid range—no trace

NOTE

DTSA must be less than DTEA in order to guarantee correct data write/read traces. Data trace ranges are exclusive of the DTSA and DTEA addresses.

25.11.10 NZ6C3 Register Access via JTAG / OnCE

Access to Nexus3 register resources is enabled by loading a single instruction (ACCESS_AUX_TAP_ONCE) into the JTAGC instruction register (IR), and then loading the corresponding OnCE OCMD register with the NEXUS3_ACCESS instruction (refer to Table 25-5). For the NZ6C3 module, the OCMD value is 0b00_0111_1100.

Once the ACCESS_AUX_TAP_ONCE instruction has been loaded, the JTAG/OnCE port allows tool/target communications with all Nexus3 registers according to the register map in Table 25-24.

Reading/writing of a NZ6C3 register then requires two (2) passes through the data-scan (DR) path of the JTAG state machine (see Section 25.11.17).

1. The first pass through the DR selects the NZ6C3 register to be accessed by providing an index (see Table 25-24), and the direction (read/write). This is achieved by loading an 8-bit value into the JTAG data register (DR). This register has the following format:

(7-bits)	(1-bit)
Nexus Register Index	R/W

RESET Value: 0x00

Nexus Register Index:	Selected from values in Table 25-24
Read/Write (R/W):	0 Read 1 Write

2. The second pass through the DR then shifts the data in or out of the JTAG port, lsb first.
   a) During a read access, data is latched from the selected Nexus register when the JTAG state machine passes through the capture-DR state.
   b) During a write access, data is latched into the selected Nexus register when the JTAG state machine passes through the update-DR state.

25.11.11Ownership Trace

This section details the ownership trace features of the NZ6C3 module.

25.11.11.1Overview

Ownership trace provides a macroscopic view, such as task flow reconstruction, when debugging software written in a high level (or object-oriented) language. It offers the highest level of abstraction for tracking operating system software execution. This is especially useful when the developer is not interested in debugging at lower levels.

25.11.11.2Ownership Trace Messaging (OTM)

Ownership trace information is messaged via the auxiliary port using an ownership trace message (OTM). The e200z6 processor contains a PowerPC Book E defined process ID register within the CPU.

The process ID register is updated by the operating system software to provide task/process ID information. The contents of this register are replicated on the pins of the processor and connected to Nexus. The process ID register value can be accessed using the **mfspr**/**mtspr** instructions. Please refer to the *e200z6 PowerPC*^{TM} *Core Reference Manual*for more details on the process ID register.

There are two conditions which will cause an ownership trace message.

1. When new information is updated in the OTR register or process ID register by the e200z6 processor, the data is latched within Nexus, and is messaged out via the auxiliary port, allowing development tools to trace ownership flow.
2. When the periodic (255) OTM message counter expires (after 255 queued messages without an OTM), an OTM will be sent. The data will be sent from either the latched OTR data or the latched process ID data. This allows processors using virtual memory to be regularly updated with the latest process ID.

Ownership trace information is messaged out in the following format:

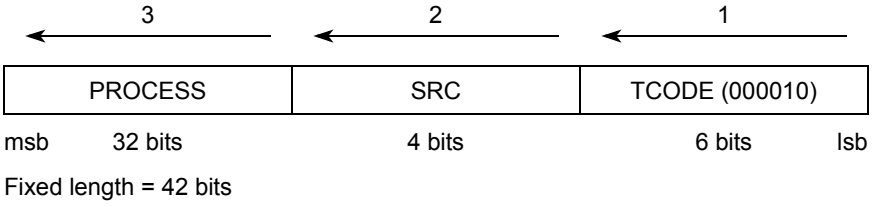


Figure 25-25. Ownership Trace Message Format

25.11.11.3OTM Error Messages

An error message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard incoming messages until it has completely emptied the queue. Once emptied, an error message will be queued. The error encoding will indicate which types of messages attempted to be queued while the FIFO was being emptied.

If only an OTM message attempts to enter the queue while it is being emptied, the error message will incorporate the OTM only error encoding (00000). If both OTM and either BTM or DTM messages attempt to enter the queue, the error message will incorporate the OTM and (program or data) trace error encoding (00111). If a watchpoint also attempts to be queued while the FIFO is being emptied, then the error message will incorporate error encoding (01000).

NOTE

The OVC bits within the DC1 register can be set to delay the CPU in order to alleviate (but not eliminate) potential overrun situations.

Error information is messaged out in the following format (see Table 25-20)

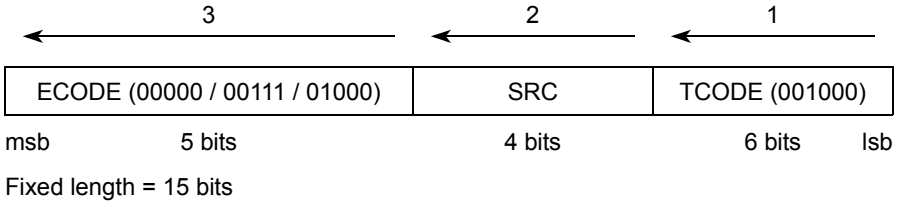


Figure 25-26. Error Message Format

25.11.11.4 OTM Flow

Ownership trace messages are generated when the operating system writes to the e200z6 process ID register or the memory mapped ownership trace register.

The following flow describes the OTM process:

1. The process ID register is a system control register. It is internal to the e200z6 processor and can be accessed by using PPC instructions **mtspr** and **mfspr**. The contents of this register are replicated on the pins of the processor and connected to Nexus.
2. OTR/process ID register reads do not cause ownership trace messages to be transmitted by the NZ6C3 module.
3. If the periodic OTM message counter expires (after 255 queued messages without an OTM), an OTM is sent using the latched data from the previous OTM or process ID register write.

25.11.12 Program Trace

This section details the program trace mechanism supported by NZ6C3 for the e200z6 processor. Program trace is implemented via branch trace messaging (BTM) as per the Class 3 IEEE®-ISTO 5001-2003 standard definition. Branch trace messaging for e200z6 processors is accomplished by snooping the e200z6 virtual address bus (between the CPU and MMU), attribute signals, and CPU status.

25.11.12.1Branch Trace Messaging (BTM)

Traditional branch trace messaging facilitates program trace by providing the following types of information:

- Messaging for taken direct branches includes how many sequential instructions were executed since the last taken branch or exception. Direct (or indirect) branches not taken are counted as sequential instructions.
- Messaging for taken indirect branches and exceptions includes how many sequential instructions were executed since the last taken branch or exception and the unique portion of the branch target address or exception vector address.

Branch history messaging facilitates program trace by providing the following information:

- Messaging for taken indirect branches and exceptions includes how many sequential instructions were executed since the last predicate instruction, taken indirect branch, or exception, the unique

portion of the branch target address or exception vector address, as well as a branch/predicate instruction history field. Each bit in the history field represents a direct branch or predicated instruction where a value of one (1) indicates taken, and a value of zero (0) indicates not taken. Certain instructions (**evsel**) generate a pair of predicate bits which are both reported as consecutive bits in the history field.

25.11.12.1.1 e200z6 Indirect Branch Message Instructions (PowerPC Book E)

Table 25-33 shows the types of instructions and events which cause indirect branch messages or branch history messages to be encoded.

Table 25-33. Indirect Branch Message Sources

Source of Indirect Branch Message	Instructions
Taken branch relative to a register value	**bcctr**, **bcctrl**, **bclr**, **bclrl**
System Call / Trap exceptions taken	**sc**, **tw**, **twi**
Return from interrupts / exceptions	**rfi**, **rfci**, **rfdi**

25.11.12.1.2 e200z6 Direct Branch Message Instructions (PowerPC Book E)

Table 25-34 shows the types of instructions which will cause direct branch messages or will toggle a bit in the instruction history buffer to be messaged out in a resource full message or branch history message.

Table 25-34. Direct Branch Message Sources

Source of Direct Branch Message	Instructions
Taken direct branch instructions	**b**, **ba**, **bl**, **bla**, **bc**, **bca**, **bcl**, **bcla**
Instruction Synchronize	**isync**

25.11.12.1.3 BTM Using Branch History Messages

Traditional BTM messaging can accurately track the number of sequential instructions between branches, but cannot accurately indicate which instructions were conditionally executed, and which were not.

Branch history messaging solves this problem by providing a predicated instruction history field in each indirect branch message. Each bit in the history represents a predicated instruction or direct branch. A value of one (1) indicates the conditional instruction was executed or the direct branch was taken. A value of zero (0) indicates the conditional instruction was not executed or the direct branch was not taken. Certain instructions (**evsel**) generate a pair of predicate bits which are both reported as consecutive bits in the history field.

Branch history messages solve predicated instruction tracking and save bandwidth since only indirect branches cause messages to be queued.

25.11.12.1.4 BTM Using Traditional Program Trace Messages

Based on the PTM bit in the DC register (DC[PTM]), program tracing can utilize either branch history messages (DC[PTM] = 1) or traditional direct/indirect branch messages (DC[PTM] = 0).

Branch history will save bandwidth and keep consistency between methods of program trace, yet may lose temporal order between BTM messages and other types of messages. Since direct branches are not messaged, but are instead included in the history field of the indirect branch history message, other types

of messages may enter the FIFO between branch history messages. The development tool cannot determine the ordering of "events" that occurred with respect to direct branches simply by the order in which messages are sent out.

Traditional BTM messages maintain their temporal ordering because each event that can cause a message to be queued will enter the FIFO in the order it occurred and will be messaged out maintaining that order.

25.11.12.2BTM Message Formats

The e200z6 Nexus3 module supports three types of traditional BTM messages—direct, indirect, and synchronization messages. It supports two types of branch history BTM messages—indirect branch history, and indirect branch history with synchronization messages. Debug status messages and error messages are also supported.

25.11.12.2.1 Indirect Branch Messages (History)

Indirect branches include all taken branches whose destination is determined at run time, interrupts and exceptions. If DC[PTM] is set, indirect branch information is messaged out in the following format:

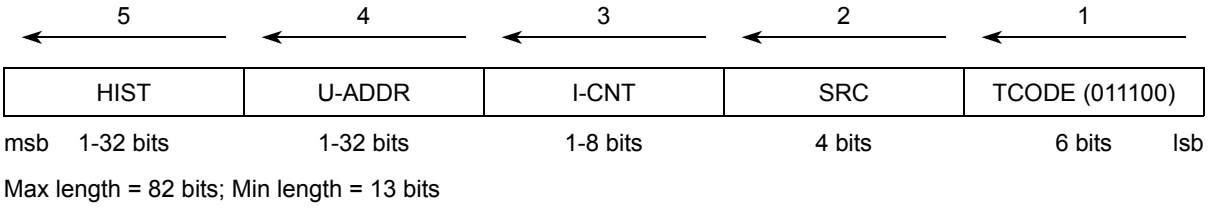


Figure 25-27. Indirect Branch Message (History) Format

25.11.12.2.2 Indirect Branch Messages (Traditional)

If DC[PTM] is cleared, indirect branch information is messaged out in the following format:

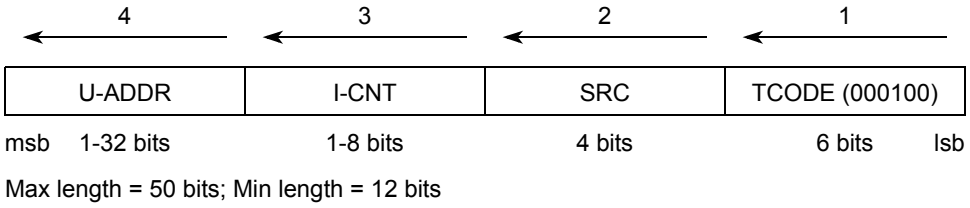


Figure 25-28. Indirect Branch Message Format

25.11.12.2.3 Direct Branch Messages (Traditional)

Direct branches (conditional or unconditional) are all taken branches whose destination is fixed in the instruction opcode. Direct branch information is messaged out in the following format:

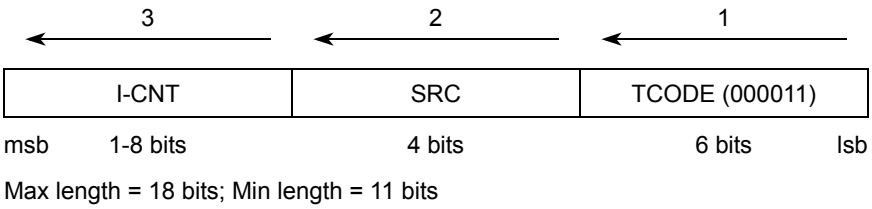


Figure 25-29. Direct Branch Message Format

NOTE

When DC[PTM] is set, direct branch messages will not be transmitted. Instead, each direct branch or predicated instruction will toggle a bit in the history buffer.

25.11.12.2.4 Resource Full Messages

The resource full message is used in conjunction with the branch history messages. The resource full message is generated when the internal branch/predicate history buffer is full. If synchronization is needed at the time this message is generated, the synchronization is delayed until the next branch trace message that is not a resource full message.

The current value of the history buffer is transmitted as part of the resource full message. This information can be concatenated by the tool with the branch/predicate history information from subsequent messages to obtain the complete branch history for a message. The internal history value is reset by this message, and the I-CNT value is reset as a result of a bit being added to the history buffer.

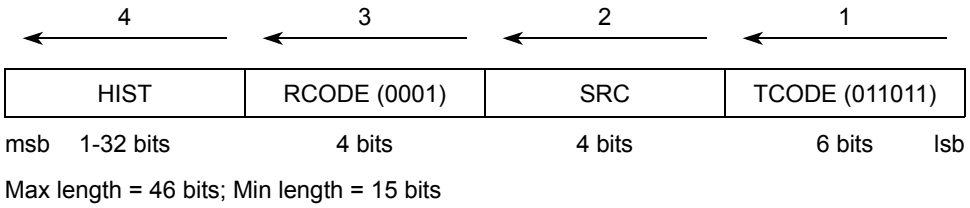


Figure 25-30. Resource Full Message Format

25.11.12.2.5 Debug Status Messages

Debug status messages report low power mode and debug status. Entering/exiting debug mode as well as entering a low power mode will trigger a debug status message. Debug status information is sent out in the following format:

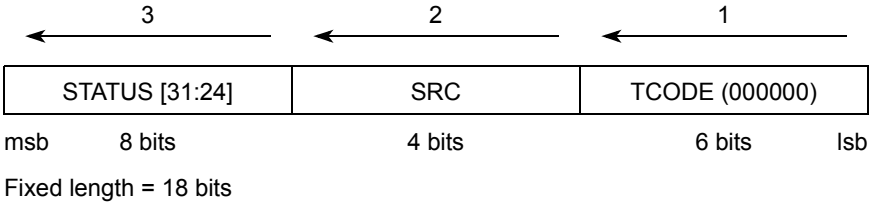


Figure 25-31. Debug Status Message Format

25.11.12.2.6 Program Correlation Messages

Program correlation messages are used to correlate events to the program flow that may not be associated with the instruction stream. In order to maintain accurate instruction tracing information when entering debug mode or a CPU low power mode (where tracing may be disabled), this message is sent upon entry into one of these two modes and includes the instruction count and branch history. Program correlation is messaged out in the following format:

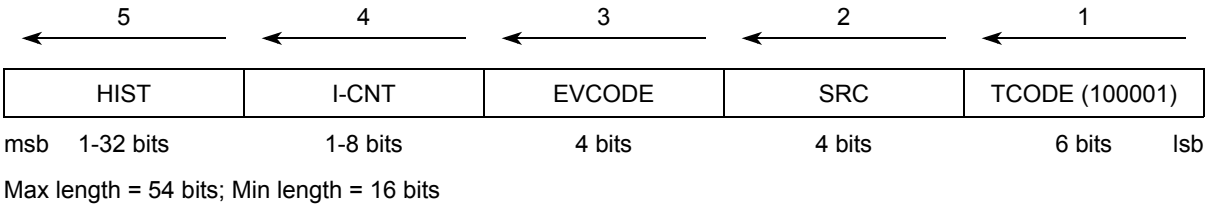


Figure 25-32. Program Correlation Message Format

25.11.12.2.7 BTM Overflow Error Messages

An error message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard incoming messages until it has completely emptied the queue. Once emptied, an error message will be queued. The error encoding will indicate which types of messages attempted to be queued while the FIFO was being emptied.

If only a program trace message attempts to enter the queue while it is being emptied, the error message will incorporate the program trace only error encoding (00001). If both OTM and program trace messages attempt to enter the queue, the error message will incorporate the OTM and program trace error encoding (00111). If a watchpoint also attempts to be queued while the FIFO is being emptied, then the error message will incorporate error encoding (01000).

NOTE

The OVC bits within the DC1 register can be set to delay the CPU in order to alleviate (but not eliminate) potential overrun situations.

Error information is messaged out in the following format

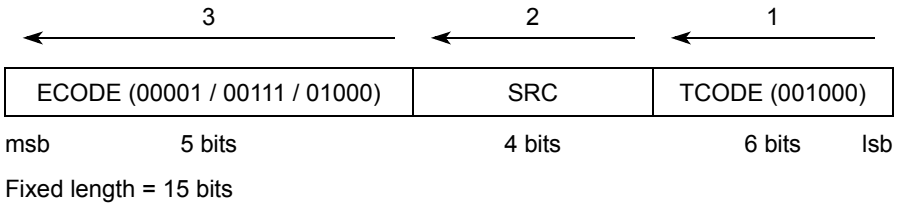


Figure 25-33. Error Message Format

25.11.12.2.8 Program Trace Synchronization Messages

A program trace direct/indirect branch with sync message is messaged via the auxiliary port (provided program trace is enabled) for the following conditions (see Table 25-35):

- Initial program trace message upon the first direct/indirect branch after exit from system reset or whenever program trace is enabled
- Upon direct/indirect branch after returning from a CPU low power state
- Upon direct/indirect branch after returning from debug mode
- Upon direct/indirect branch after occurrence of queue overrun (can be caused by any trace message), provided program trace is enabled
- Upon direct/indirect branch after the periodic program trace counter has expired indicating 255 *without-sync* program trace messages have occurred since the last *with-sync* message occurred
- Upon direct/indirect branch after assertion of the event in (EVTI) pin if the EIC bits within the DC1 register have enabled this feature

- Upon direct/indirect branch after the sequential instruction counter has expired indicating 255 instructions have occurred between branches
- Upon direct/indirect branch after a BTM message was lost due to an attempted access to a secure memory location.
- Upon direct/indirect branch after a BTM message was lost due to a collision entering the FIFO between the BTM message and either a watchpoint message or an ownership trace message

If the NZ6C3 module is enabled at reset, a EVTI assertion initiates a program trace direct/indirect branch with sync message (if program trace is enabled) upon the first direct/indirect branch. The format for program trace direct/indirect branch with sync messages is as follows:

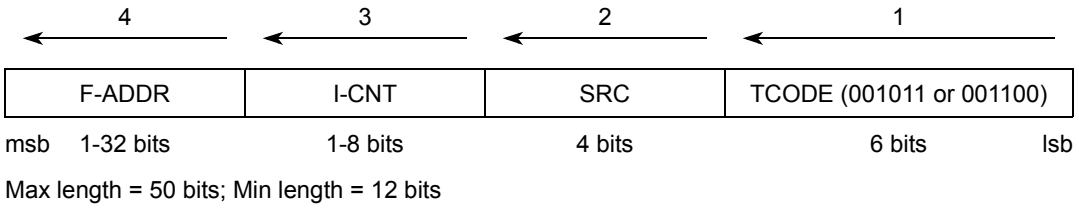


Figure 25-34. Direct/Indirect Branch with Sync Message Format

The formats for program trace direct/indirect branch with sync. messages and indirect branch history with sync. messages are as follows

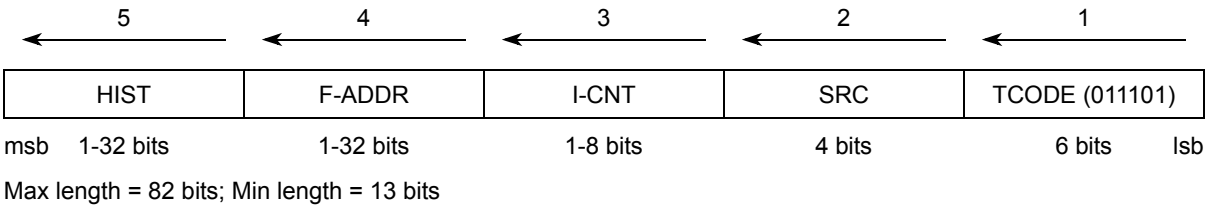


Figure 25-35. Indirect Branch History with Sync. Message Format

Exception conditions that result in program trace synchronization are summarized in Table 25-35.

Table 25-35. Program Trace Exception Summary

Exception Condition	Exception Handling
System Reset Negation	At the negation of JTAG reset (JCOMP), queue pointers, counters, state machines, and registers within the NZ6C3 module are reset. Upon the first branch out of system reset (if program trace is enabled), the first program trace message is a direct/indirect branch with sync. message.
Program Trace Enabled	The first program trace message (after program trace has been enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a low power mode or debug mode the next direct/indirect branch will be converted to a direct/indirect branch with sync. message.
Queue Overrun	An error message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard messages until it has completely emptied the queue. Once emptied, an error message will be queued. The error encoding will indicate which types of messages attempted to be queued while the FIFO was being emptied. The next BTM message in the queue will be a direct/indirect branch with sync. message.

Table 25-35. Program Trace Exception Summary (continued)

Exception Condition	Exception Handling
Periodic Program Trace Sync.	A forced synchronization occurs periodically after 255 program trace messages have been queued. A direct/indirect branch with sync. message is queued. The periodic program trace message counter then resets.
Event In	If the Nexus module is enabled, an EVTI assertion initiates a direct/indirect branch with sync. message upon the next direct/indirect branch (if program trace is enabled and the EIC bits of the DC1 register have enabled this feature).
Sequential Instruction Count Overflow	When the sequential instruction counter reaches its maximum count (up to 255 sequential instructions may be executed), a forced synchronization occurs. The sequential counter then resets. A program trace direct/indirect branch with sync.message is queued upon execution of the next branch.
Attempted Access to Secure Memory	For devices which implement security, any attempted branch to secure memory locations will temporarily disable program trace & cause the corresponding BTM to be lost. The following direct/indirect branch will queue a direct/indirect branch with sync. message. The count value within this message will be inaccurate since the re-enable of program trace is not necessarily aligned on an instruction boundary.
Collision Priority	All messages have the following priority: WPM -> OTM -> BTM -> DTM. A BTM message which attempts to enter the queue at the same time as a watchpoint message or ownership trace message will be lost. An error message will be sent indicating the BTM was lost. The following direct/indirect branch will queue a direct/indirect branch with sync. message. The count value within this message will reflect the number of sequential instructions executed after the last successful BTM Message was generated. This count will include the branch which did not generate a message due to the collision.

25.11.12.3BTM Operation

25.11.12.3.1 Enabling Program Trace

Both types of branch trace messaging can be enabled in one of two ways:

- Setting the TM field of the DC1 register to enable program trace (DC1[TM])
- Using the PTS field of the WT register to enable program trace on watchpoint hits (e200z6 watchpoints are configured within the CPU)

25.11.12.3.2 Relative Addressing

The relative address feature is compliant with the IEEE®-ISTO 5001-2003 standard recommendations, and is designed to reduce the number of bits transmitted for addresses of indirect branch messages.

The address transmitted is relative to the target address of the instruction which triggered the previous indirect branch (or sync) message. It is generated by XOR'ing the new address with the previous address, and then using only the results up to the most significant 1 in the result. To recreate this address, an XOR of the (most-significant 0-padded) message address with the previously decoded address gives the current address.

Previous address (A1) =0x0003FC01, New address (A2) = 0x0003F365

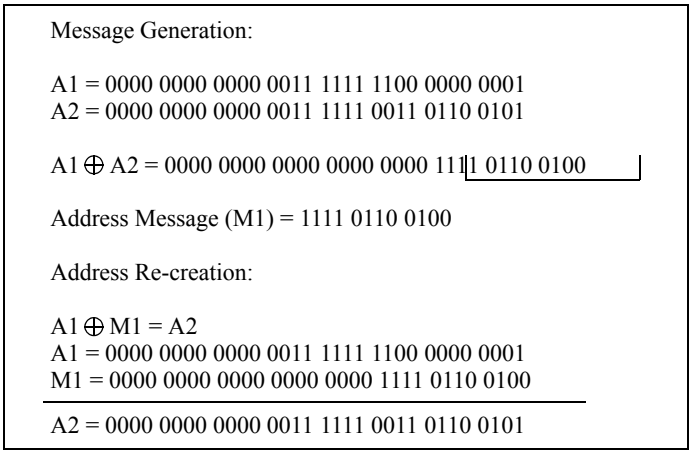


Figure 25-36. Relative Address Generation and Re-creation

25.11.12.3.3 Branch/Predicate Instruction History (HIST)

If DC[PTM] is set, BTM messaging will use the branch history format. The branch history (HIST) packet in these messages provides a history of direct branch execution used for reconstructing the program flow. This packet is implemented as a left-shifting shift register. The register is always pre-loaded with a value of one (1). This bit acts as a stop bit so that the development tools can determine which bit is the end of the history information. The pre-loaded bit itself is not part of the history, but is transmitted with the packet.

A value of one (1) is shifted into the history buffer on a taken branch (condition or unconditional) and on any instruction whose predicate condition executed as true. A value of zero (0) is shifted into the history buffer on any instruction whose predicate condition executed as false as well as on branches not taken. This will include indirect as well as direct branches not taken. For the **evsel** instruction, two bits are shifted in, corresponding to the low element (shifted in first) and the high element (shifted in second) conditions.

25.11.12.3.4 Sequential Instruction Count (I-CNT)

The I-CNT packet, is present in all BTM messages. For traditional branch messages, I-CNT represents the number of sequential instructions, or non-taken branches in between direct/indirect branch messages.

For branch history messages, I-CNT represents the number of instructions executed since the last taken/non-taken direct branch, last taken indirect branch or exception. Not taken indirect branches are considered sequential instructions and cause the instruction count to increment. I-CNT also represents the number of instructions executed since the last predicate instruction.

The sequential instruction counter overflows when its value reaches 255. The next BTM message will be converted to a synchronization type message.

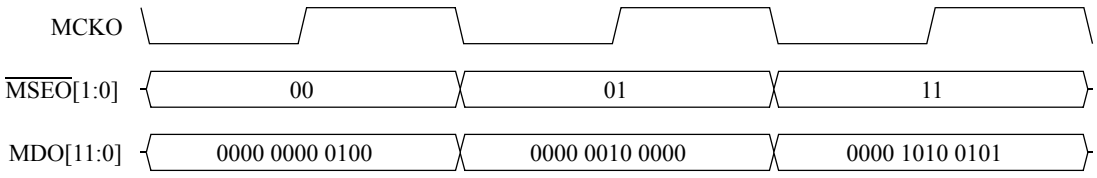
25.11.12.3.5 Program Trace Queueing

NZ6C3 implements a message queue. Messages that enter the queue are transmitted via the auxiliary pins in the order in which they are queued.

NOTE

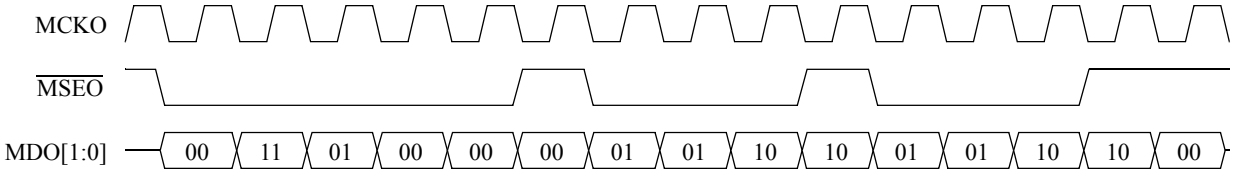
If multiple trace messages need to be queued at the same time, Watchpoint Messages will have the highest priority (WPM –> OTM –> BTM –> DTM).

25.11.12.4Program Trace Timing Diagrams



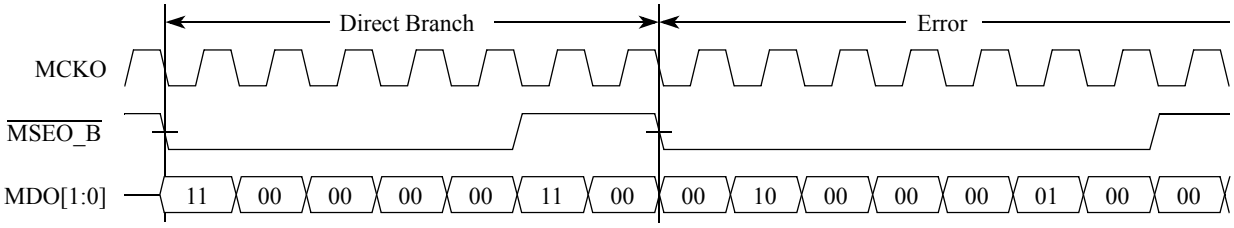
TCODE = 4
Source Processor = 0b0000
Number of Sequence Instructions = 128
Relative Address = 0xA5

Figure 25-37. Program Trace (MDO = 12)—Indirect Branch Message (Traditional)



TCODE = 28
Source Processor = 0b0000
Number of Sequential Instructions = 0
Relative Address = 0xA5
Branch History = 0b1010_0101 (with Stop)

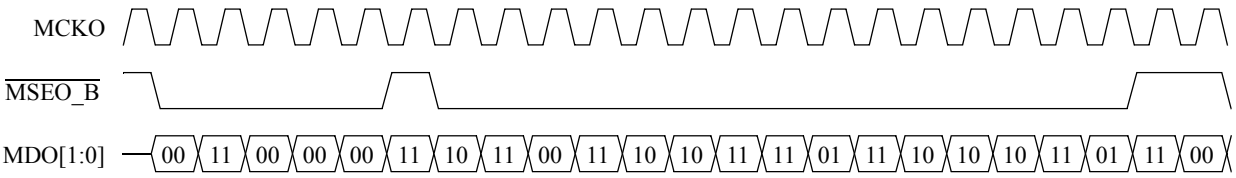
Figure 25-38. Program Trace (MDO = 2)—Indirect Branch Message (History)



DBM:
TCODE = 3
Source Processor = 0b0000
Number of Sequential Instructions = 3

Error:
TCODE = 8
Source Processor = 0b0000
Error Code = 1 (Queue Overrun – BTM Only)

Figure 25-39. Program Trace—Direct Branch (Traditional) and Error Messages



TCODE = 12
Source Processor = 0b0000
Number of Sequential Instructions = 3
Full Target Address = 0xDEAD_FACE

Figure 25-40. Program Trace—Indirect Branch with Sync. Message

25.11.13 Data Trace

This section deals with the data trace mechanism supported by the NZ6C3 module. Data trace is implemented via data write messaging (DWM) and data read messaging (DRM), as per the IEEE®-ISTO 5001-2003 standard.

25.11.13.1Data Trace Messaging (DTM)

Data trace messaging for e200z6 is accomplished by snooping the e200z6 virtual data bus (between the CPU and MMU), and storing the information for qualifying accesses (based on enabled features and matching target addresses). The NZ6C3 module traces all data access that meet the selected range and attributes.

NOTE

Data trace is only performed on the e200z6 virtual data bus. This allows for data visibility for the incorporated data cache. Only e200z6 CPU initiated accesses will be traced. No DMA accesses to the AHB system bus will be traced.

Data trace messaging can be enabled in one of two ways:

- Setting the TM field of the DC1 register to enable data trace (DC1[TM]).
- Using WT[DTS] to enable data trace on watchpoint hits (e200z6 watchpoints are configured within the Nexus1 module)

25.11.13.2 DTM Message Formats

The Nexus3 module supports five types of DTM messages: data write, data read, data write synchronization, data read synchronization and error messages.

25.11.13.2.1 Data Write Messages

The data write message contains the data write value and the address of the write access, relative to the previous data trace message. Data write message information is messaged out in the following format:

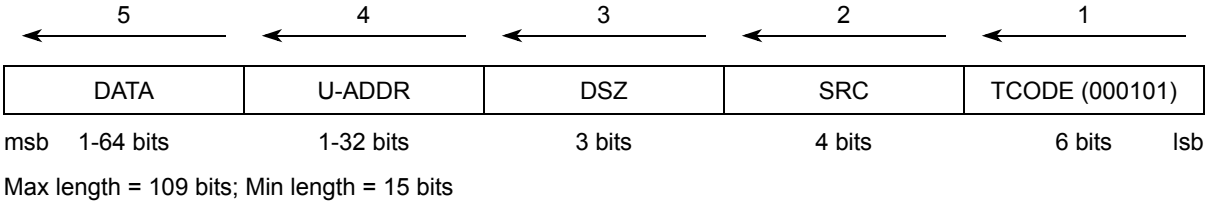


Figure 25-41. Data Write Message Format

25.11.13.2.2 Data Read Messages

The data read message contains the data read value and the address of the read access, relative to the previous data trace message. Data read message information is messaged out in the following format:

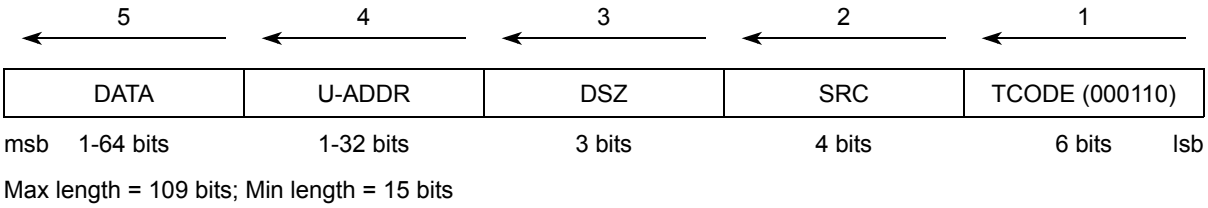


Figure 25-42. Data Read Message Format

NOTE

For the e200z6 based CPU, the double-word encoding (data size = 0b000) will indicate a double-word access and will be sent out as a single data trace message with a single 64-bit data value.

25.11.13.2.3 DTM Overflow Error Messages

An error message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard incoming messages until it has completely emptied the queue. Once emptied, an error message will be queued. The error encoding will indicate which types of messages attempted to be queued while the FIFO was being emptied.

If only a data trace message attempts to enter the queue while it is being emptied, the error message will incorporate the data trace only error encoding (00010). If both OTM and data trace messages attempt to enter the queue, the error message will incorporate the OTM and data trace error encoding (00111). If a watchpoint also attempts to be queued while the FIFO is being emptied, then the error message will incorporate error encoding (01000).

NOTE

The OVC bits within the DC1 register can be set to delay the CPU in order to alleviate (but not eliminate) potential overrun situations.

Error information is messaged out in the following format:

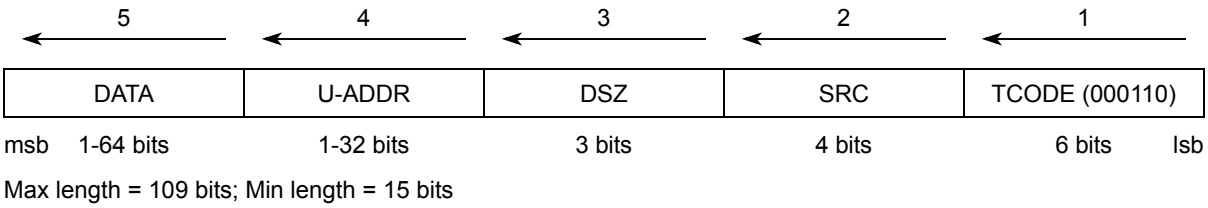


Figure 25-43. Error Message Format

25.11.13.2.4 Data Trace Synchronization Messages

A data trace write/read with sync. message is messaged via the auxiliary port (provided data trace is enabled) for the following conditions (see Table 25-36):

- Initial data trace message after exit from system reset or whenever data trace is enabled
- Upon exiting debug mode
- After occurrence of queue overrun (can be caused by any trace message), provided data trace is enabled
- After the periodic data trace counter has expired indicating 255 *without-sync* data trace messages have occurred since the last *with-sync* message occurred

- Upon assertion of the event in (EVTI) pin, the first data trace message will be a synchronization message if the EIC bits of the DC1 register have enabled this feature
- Upon data trace write/read after the previous DTM message was lost due to an attempted access to a secure memory location
- Upon data trace write/read after the previous DTM message was lost due to a collision entering the FIFO between the DTM message and any of the following: watchpoint message, ownership trace message, or branch trace message

Data trace synchronization messages provide the full address (without leading zeros) and insure that development tools fully synchronize with data trace regularly. Synchronization messages provide a reference address for subsequent data messages, in which only the unique portion of the data trace address is transmitted. The format for data trace write/read with sync. messages is as follows:

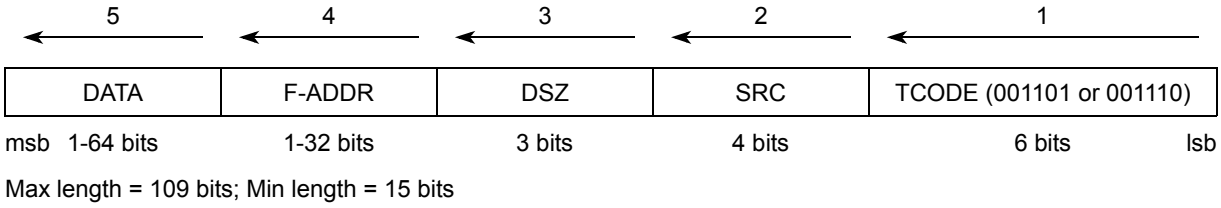


Figure 25-44. Data Write/Read with Sync. Message Format

Exception conditions that result in data trace synchronization are summarized in Table 25-36.

Table 25-36. Data Trace Exception Summary

Exception Condition	Exception Handling
System Reset Negation	At the negation of JTAG reset (JCOMP), queue pointers, counters, state machines, and registers within the NZ6C3 module are reset. If data trace is enabled, the first data trace message is a data write/read with sync. message.
Data Trace Enabled	The first data trace message (after data trace has been enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a low power mode or debug mode the next data trace message will be converted to a data write/read with sync. message.
Queue Overrun	An error message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard messages until it has completely emptied the queue. Once emptied, an error message will be queued. The error encoding will indicate which types of messages attempted to be queued while the FIFO was being emptied. The next DTM message in the queue will be a data write/read with sync. message.
Periodic Data Trace Sync.	A forced synchronization occurs periodically after 255 data trace messages have been queued. A data write/read with sync. message is queued. The periodic data trace message counter then resets.
Event In	If the Nexus module is enabled, a EVTI assertion initiates a data trace write/read with sync. message upon the next data write/read (if data trace is enabled and the EIC bits of the DC1 register have enabled this feature).
Attempted Access to Secure Memory	For devices which implement security, any attempted read or write to secure memory locations will temporarily disable data trace & cause the corresponding DTM to be lost. A subsequent read/write will queue a data trace read/write with sync. message.

Table 25-36. Data Trace Exception Summary (continued)

Exception Condition	Exception Handling
Collision Priority	All messages have the following priority: WPM –> OTM –> BTM –> DTM. A DTM message which attempts to enter the queue at the same time as a watchpoint message or ownership trace message or branch trace message will be lost. A subsequent read/write will queue a data trace read/write with sync. message.

25.11.13.3DTM Operation

25.11.13.3.1 DTM Queueing

NZ6C3 implements a message queue for DTM messages. Messages that enter the queue are transmitted via the auxiliary pins in the order in which they are queued.

NOTE

If multiple trace messages need to be queued at the same time, watchpoint messages will have the highest priority (WPM –> OTM –> BTM –> DTM).

25.11.13.3.2 Relative Addressing

The relative address feature is compliant with the IEEE®-ISTO 5001-2003 standard recommendations, and is designed to reduce the number of bits transmitted for addresses of data trace messages. Refer to Section 25.11.12.3.2, “ Relative Addressing for details.

25.11.13.3.3 Data Trace Windowing

Data write/read messages are enabled via the RWT1(2) field in the data trace control register (DTC) for each DTM channel. Data trace windowing is achieved via the address range defined by the DTEA and DTSA registers and by the RC1(2) field in the DTC. All e200z6 initiated read/write accesses which fall inside or outside these address ranges, as programmed, are candidates to be traced.

25.11.13.3.4 Data Access/Instruction Access Data Tracing

The Nexus3 module is capable of tracing both instruction access data or data access data. Each trace window can be configured for either type of data trace by setting the DI1(2) field within the data trace control register for each DTM channel.

25.11.13.3.5 e200z6 Bus Cycle Special Cases

Table 25-37. e200z6 Bus Cycle Cases

Special Case	Action
e200z6 bus cycle aborted	Cycle ignored
e200z6 bus cycle with data error ($\overline{\text{TEA}}$)	Data Trace Message discarded
e200z6 bus cycle completed without error	Cycle captured & transmitted
e200z6 bus cycle initiated by NZ6C3	Cycle ignored
e200z6 bus cycle is an instruction fetch	Cycle ignored

Table 25-37. e200z6 Bus Cycle Cases (continued)

Special Case	Action
e200z6 bus cycle accesses misaligned data (across 64-bit boundary)—both 1st & 2nd transactions within data trace range	1st & 2nd cycle captured & 2 DTM's transmitted (see Note)
e200z6 bus cycle accesses misaligned data (across 64-bit boundary)—1st transaction within data trace range; 2nd transaction out of data trace range	1st cycle captured and transmitted; 2nd cycle ignored
e200z6 bus cycle accesses misaligned data (across 64-bit boundary)—1st transaction out of data trace range; 2nd transaction within data trace range	1st cycle ignored; 2nd cycle capture and transmitted

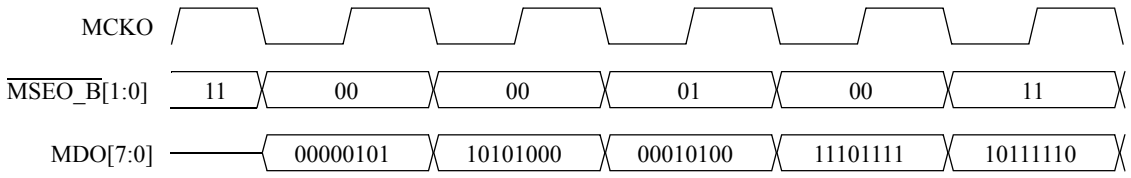
NOTE

For misaligned accesses (crossing 64-bit boundary), the access is broken into two accesses. If both accesses are within the data trace range, two DTMs will be sent: one with a size encoding indicating the size of the original access (that is, word), and one with a size encoding for the portion which crossed the boundary (that is, 3-byte).

NOTE

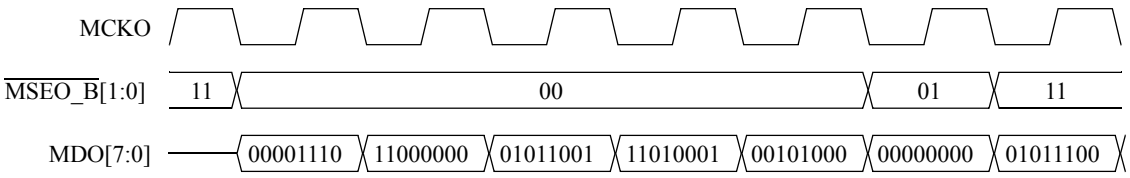
An STM to the cache's store buffer within the data trace range will initiate a DTM message. If the corresponding memory access causes an error, a checkstop condition will occur. The debug/development tool should use this indication to invalidate the previous DTM.

25.11.13.4 Data Trace Timing Diagrams (8 MDO Configuration)



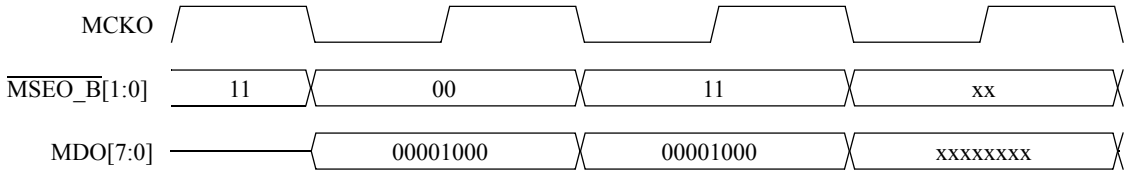
TCODE = 5
Source Processor = 0b0000
Data Size = 010 (Half-Word)
Relative Address = 0xA5
Write Data = 0xBEEF

Figure 25-45. Data Trace—Data Write Message



TCODE = 14
Source Processor = 0b0000
Data Size = 000 (Byte)
Full Access Address = 0x0146_8ACE
Write Data = 0x5C

Figure 25-46. Data Trace—Data Read with Sync Message



TCODE = 8
Source Processor = 0b0000
Error Code = 2 (Queue Overrun – DTM Only)

Figure 25-47. Error Message (Data Trace only encoded)

25.11.14 Watchpoint Support

This section details the watchpoint features of the NZ6C3 module.

25.11.14.1Overview

The NZ6C3 module provides watchpoint messaging via the auxiliary pins, as defined by the IEEE®-ISTO 5001-2003 standard.

NZ6C3 is not compliant with Class4 breakpoint/watchpoint requirements defined in the standard. The breakpoint/watchpoint control register is not implemented.

25.11.14.2Watchpoint Messaging

Enabling watchpoint messaging is done by setting the watchpoint enable bit in the DC1 register. Setting the individual watchpoint sources is supported through the e200z6 Nexus1 module. The e200z6 Nexus1 module is capable of setting multiple address and/or data watchpoints. Please refer to the e200z6 Core Reference Manual for more information on watchpoint initialization.

When these watchpoints occur, a watchpoint event signal from the Nexus1 module causes a message to be sent to the queue to be messaged out. This message includes the watchpoint number indicating which watchpoint caused the message.

The occurrence of any of the e200z6 defined watchpoints can be programmed to assert the event out $\overline{\text{EVTO}}$ pin for one (1) period of the output clock (MCKO).

Watchpoint information is messaged out in the following format

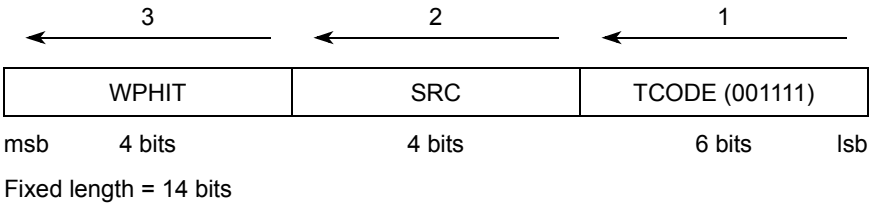


Figure 25-48. Watchpoint Message Format.

Table 25-38. Watchpoint Source Encoding

Watchpoint Source (8 bits)	Watchpoint Description
00000001	e200z6 Watchpoint #0 (IAC1 from Nexus1)
00000010	e200z6 Watchpoint #1 (IAC2 from Nexus1)
00000100	e200z6 Watchpoint #2 (IAC3 from Nexus1)
00001000	e200z6 Watchpoint #3 (IAC4 from Nexus1)
00010000	e200z6 Watchpoint #4 (DAC1 from Nexus1)
00100000	e200z6 Watchpoint #5 (DAC2 from Nexus1)
01000000	e200z6 Watchpoint #6 (DCNT1 from Nexus1)
10000000	e200z6 Watchpoint #7 (DCNT2 from Nexus1)

25.11.14.3Watchpoint Error Message

An error message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard messages until it has completely emptied the queue. Once emptied, an error message will be queued. The error encoding will indicate which types of messages attempted to be queued while the FIFO was being emptied.

If only a watchpoint message attempts to enter the queue while it is being emptied, the error message will incorporate the watchpoint only error encoding (00110). If an OTM and/or program trace and/or data trace message also attempts to enter the queue while it is being emptied, the error message will incorporate error encoding (01000).

NOTE

The OVC bits within the DC1 register can be set to delay the CPU in order to alleviate (but not eliminate) potential overrun situations.

Error information is messaged out in the following format (see Table 25-20)

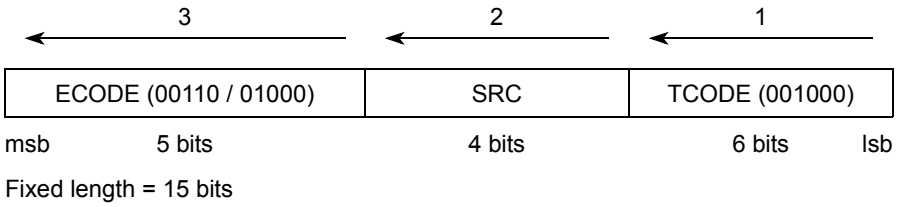


Figure 25-49. Error Message Format

25.11.14.4Watchpoint Timing Diagram (2 MDO/1 MSEO Configuration)

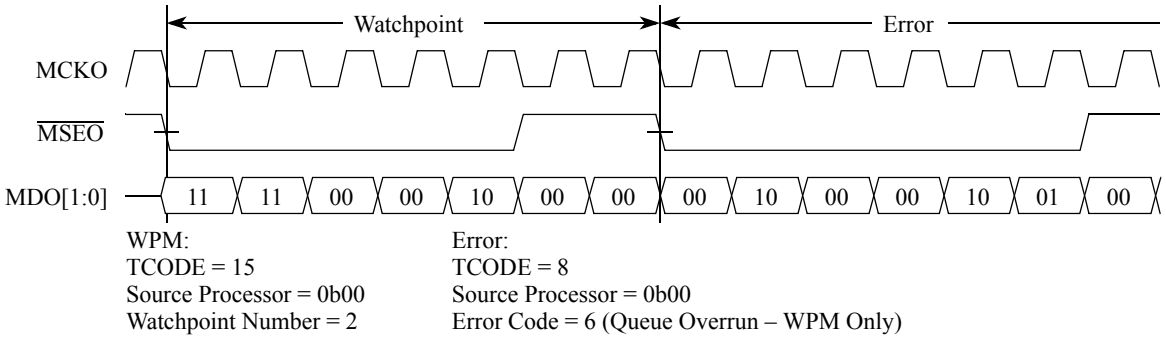


Figure 25-50. Watchpoint Message & Watchpoint Error Message

25.11.15 NZ6C3 Read/Write Access to Memory-Mapped Resources

The read/write access feature allows access to memory-mapped resources via the JTAG/OnCE port. The read/write mechanism supports single as well as block reads and writes to e200z6 system bus resources.

The NZ6C3 module is capable of accessing resources on the e200z6 system bus, with multiple configurable priority levels. Memory-mapped registers and other non-cached memory can be accessed via the standard memory map settings.

All accesses are setup and initiated by the read/write access control/status register (RWCS), as well as the read/write access address (RWA) and read/write access data registers (RWD).

Using the read/write access registers (RWCS/RWA/RWD), memory-mapped e200z6 system bus resources can be accessed through NZ6C3. The following subsections describe the steps which are required to access memory-mapped resources.

NOTE

Read/write access can only access memory mapped resources when system reset is de-asserted.

Misaligned accesses are NOT supported in the e200z6 Nexus3 module.

25.11.15.1Single Write Access

1. Initialize the read/write access address register (RWA) through the access method outlined in Section 25.11.10, " NZ6C3 Register Access via JTAG / OnCE" using the Nexus register index of 0x9 (see Table 25-24). Configure as follows:
   - Write Address -> 0xnnnnnnnn (write address)
2. Initialize the read/write access control/status register (RWCS) through the access method outlined in Section 25.11.10, " NZ6C3 Register Access via JTAG / OnCE," using the Nexus Register Index of 0x7(see Table 25-24). Configure the bits as follows:
   - Access Control RWCS[AC] –> 0b1 (to indicate start access)
   - Map Select RWCS[MAP] –> 0b000 (primary memory map)
   - Access Priority RWCS[PR] –> 0b00 (lowest priority)
   - Read/Write RWCS[RW] –> 0b1 (write access)
   - Word Size RWCS[SZ] –> 0b0xx (32-bit, 16-bit, 8-bit)
   - Access Count RWCS[CNT] –> 0x0000 or 0x0001 (single access)

NOTE

Access count RWCS[CNT] of 0x0000 or 0x0001 will perform a single access.

3. Initialize the read/write access data register (RWD) through the access method outlined in Section 25.11.10, " NZ6C3 Register Access via JTAG / OnCE," using the Nexus register index of 0xA (see Table 25-24). Configure as follows:
   - Write Data -> 0xnnnnnnnn (write data)
4. The NZ6C3 module will then arbitrate for the system bus and transfer the data value from the data buffer RWD register to the memory mapped address in the read/write access address register (RWA). When the access has completed without error (ERR=1'b0), NZ6C3 asserts the RDY pin and clears the DV bit in the RWCS register. This indicates that the device is ready for the next access.

NOTE

Only the RDY pin as well as the DV and ERR bits within the RWCS provide read/write access status to the external development tool.

25.11.15.2Block Write Access (Non-Burst Mode)

1. For a non-burst block write access, follow Steps 1, 2, and 3 outlined in Section 25.11.15.1, “Single Write Access to initialize the registers,” but using a value greater than one (0x1) for the RWCS[CNT] field.
2. The NZ6C3 module will then arbitrate for the system bus and transfer the first data value from the RWD register to the memory mapped address in the read/write access address register (RWA). When the transfer has completed without error (ERR = 0), the address from the RWA register is incremented to the next word size (specified in the SZ field) and the number from the CNT field is decremented. Nexus will then assert the RDY pin. This indicates that the device is ready for the next access.
3. Repeat step 3 in Section 25.11.15.1, “Single Write Access” until the internal CNT value is zero (0). When this occurs, the DV bit within the RWCS will be cleared to indicate the end of the block write access.

25.11.15.3Block Write Access (Burst Mode)

1. For a burst block write access, follow Steps 1 and 2 outlined in Section 25.11.15.1, “Single Write Access” to initialize the registers, using a value of four (double-words) for the CNT field and a RWCS[SZ] field indicating 64-bit access.
2. Initialize the burst data buffer (read/write access data register) through the access method outlined in Section 25.11.10, “ NZ6C3 Register Access via JTAG / OnCE,” using the Nexus register Index of 0xA (see Table 25-24).
3. Repeat step 2 until all double-word values are written to the buffer.

NOTE

The data values must be shifted in 32-bits at a time lsb first (that is, double-word write = two word writes to the RWD).

4. The Nexus module will then arbitrate for the system bus and transfer the burst data values from the data buffer to the system bus beginning from the memory mapped address in the read/write access address register (RWA). For each access within the burst, the address from the RWA register is incremented to the next double-word size (specified in the SZ field) modulo the length of the burst, and the number from the CNT field is decremented.
5. When the entire burst transfer has completed without error (ERR = 0), NZ6C3 will then assert the RDY pin, and the DV bit within the RWCS will be cleared to indicate the end of the block write access.

NOTE

The actual RWA value as well as the CNT field within the RWCS are not changed when executing a block write access (burst or non-burst). The original values can be read by the external development tool at any time.

25.11.15.4Single Read Access

1. Initialize the read/write access address register (RWA) through the access method outlined in Section 25.11.10, " NZ6C3 Register Access via JTAG / OnCE," using the Nexus register index of 0x9 (see Table 25-24). Configure as follows:
   - Read Address –> 0xnnnnnnnn (read address)
2. Initialize the read/write access control/status register (RWCS) through the access method outlined in Section 25.11.10, " NZ6C3 Register Access via JTAG / OnCE," using the Nexus register index of 0x7 (see Table 25-24). Configure the bits as follows:
   - Access Control RWCS[AC] –> 0b1 (to indicate start access)
   - Map Select RWCS[MAP] –> 0b000 (primary memory map)
   - Access Priority RWCS[PR] –> 0b00 (lowest priority)
   - Read/Write RWCS[RW] –> 0b0 (read access)
   - Word Size RWCS[SZ] –> 0b0xx (32-bit, 16-bit, 8-bit)
   - Access Count RWCS[CNT] –> 0x0000 or 0x0001 (single access)

NOTE

Access Count (CNT) of 0x0000 or 0x0001 will perform a single access.

3. The NZ6C3 module will then arbitrate for the system bus and the read data will be transferred from the system bus to the RWD register. When the transfer is completed without error (ERR = 0), Nexus asserts the RDY pin and sets the DV bit in the RWCS register. This indicates that the device is ready for the next access.
4. The data can then be read from the read/write access data register (RWD) through the access method outlined in Section 25.11.10, " NZ6C3 Register Access via JTAG / OnCE," using the Nexus register index of 0xA (see Table 25-24).

NOTE

Only the RDY pin as well as the DV and ERR bits within the RWCS provide Read/Write Access status to the external development tool.

25.11.15.5Block Read Access (Non-Burst Mode)

1. For a non-burst block read access, follow Steps 1 and 2 outlined in Section 25.11.15.4, "Single Read Access" to initialize the registers, but using a value greater than one (0x1) for the CNT field in the RWCS register.
2. The NZ6C3 module will then arbitrate for the system bus and the read data will be transferred from the system bus to the RWD register. When the transfer has completed without error (ERR=0b0), the address from the RWA register is incremented to the next word size (specified in the SZ field) and the number from the CNT field is decremented. Nexus will then assert the RDY pin. This indicates that the device is ready for the next access.
3. The data can then be read from the read/write access data register (RWD) through the access method outlined in Section 25.11.10, " NZ6C3 Register Access via JTAG / OnCE," using the Nexus register index of 0xA (see Table 25-24).

4. Repeat steps 3 and 4 in Section 25.11.15.4, "Single Read Access" until the CNT value is zero (0). When this occurs, the DV bit within the RWCS is set to indicate the end of the block read access.

25.11.15.6Block Read Access (Burst Mode)

5. For a burst block read access, follow Steps 1 and 2 outlined in Section 25.11.15.4, "Single Read Access" to initialize the registers, using a value of four (double-words) for the CNT field and an RWCS[SZ] field indicating 64-bit access.
6. The NZ6C3 module will then arbitrate for the system bus and the burst read data will be transferred from the system bus to the data buffer (RWD register). For each access within the burst, the address from the RWA register is incremented to the next double-word (specified in the SZ field) and the number from the CNT field is decremented.
7. When the entire burst transfer has completed without error (ERR = 0), Nexus will then assert the RDY pin and the DV bit within the RWCS will be set to indicate the end of the block read access.
8. The data can then be read from the burst data buffer (read/write access data register) through the access method outlined in Section 25.11.10, " NZ6C3 Register Access via JTAG / OnCE," using the Nexus register index of 0xA (see Table 25-24).
9. Repeat step 3 until all double-word values are read from the buffer.

NOTE

The data values must be shifted out 32-bits at a time lsb first (that is, double-word read = two word reads from the RWD).

NOTE

The actual RWA value as well as the CNT field within the RWCS are not changed when executing a block read access (burst or non-burst). The original values can be read by the external development tool at any time.

25.11.15.7Error Handling

The NZ6C3 module handles various error conditions as follows:

25.11.15.7.1 System Bus Read/Write Error

All address and data errors that occur on read/write accesses to the e200z6 system bus will return a transfer error. If this occurs:

1. The access is terminated without re-trying (AC bit is cleared).
2. The ERR bit in the RWCS register is set.
3. The error message is sent (TCODE = 8) indicating read/write error.

25.11.15.7.2 Access Termination

The following cases are defined for sequences of the read/write protocol that differ from those described in the above sections:

1. If the AC bit in the RWCS register is set to start read/write accesses and invalid values are loaded into the RWD and/or RWA, then a system bus access error may occur. This is handled as described above.

2. If a block access is in progress (all cycles not completed), and the RWCS register is written, then the original block access is terminated at the boundary of the nearest completed access.
   a) If the RWCS is written with the AC bit set, the next read/write access will begin and the RWD can be written to/ read from.
   b) If the RWCS is written with the AC bit cleared, the read/write access is terminated at the nearest completed access. This method can be used to break (early terminate) block accesses.

25.11.15.8 Read/Write Access Error Message

The read/write access error message is sent out when an system bus access error (read or write) has occurred.

Error information is messaged out in the following format:

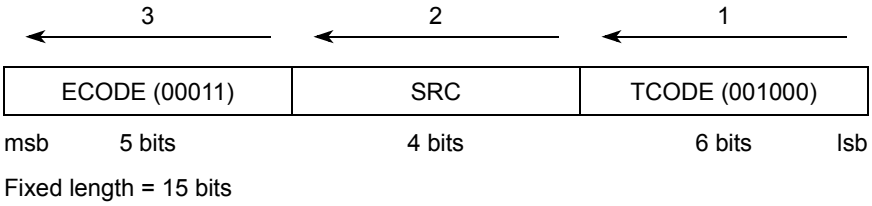


Figure 25-51. Error Message Format

25.11.16 Examples

The following are examples of program trace and data trace messages.

Table 25-39 illustrates an example indirect branch message with an 8 MDO / 2 MSEO configuration.

Note that T0 and S0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- Ix = Number of instructions (variable)
- Ax = Unique portion of the address (variable)

Table 25-39. Indirect Branch Message Example (12 MDO / 2 MSEO)

Clock	MDO[11:0]												MSEO[1:0]		State
	11	10	9	8	7	6	5	4	3	2	1	0			
0	X	X	X	X	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	I1	I0	S3	S2	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	0	0	0	0	0	0	0	0	I5	I4	I3	I2	0	1	End Packet
3	0	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	1	1	End Packet/End Message
4	X	X	S3	S2	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start of Next Message

Table 25-40 illustrates an example of direct branch message with 12 MDO / 2 MSEO.

Note that T0 and I0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- Ix = Number of instructions (variable)

Table 25-40. Direct Branch Message Example (12 MDO / 2 MSEO)

Clock	MDO[11:0]												MSEO[1:0]		State
	11	10	9	8	7	6	5	4	3	2	1	0			
0	X	X	X	X	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	I1	I0	S3	S2	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	0	0	0	0	0	0	0	0	0	0	I3	I2	1	1	End Packet/End Message
3	X	X	X	X	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start of Next Message

Table 25-41 an example data write message with 12 MDO / 2 MSEO configuration

Note that T0, A0, D0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- Zx = Data size (fixed)
- Ax = Unique portion of the address (variable)
- Dx = Write data (variable - 8, 16 or 32-bit)

Table 25-41. Direct Write Message Example (12 MDO / 2 MSEO)

Clock	MDO[11:0]												MSEO[1:0]		State
	11	10	9	8	7	6	5	4	3	2	1	0			
0	X	X	X	X	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	Z1	Z0	S3	S2	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	0	0	0	0	0	0	0	A3	A2	A1	A0	Z2	0	1	End Packet
3	X	X	X	X	D7	D6	D5	D4	D3	D2	D1	D0	1	1	End Packet/End Message

25.11.17 IEEE® 1149.1 (JTAG) RD/WR Sequences

This section contains example JTAG/OnCE sequences used to access resources.

25.11.17.1JTAG Sequence for Accessing Internal Nexus Registers

Table 25-42. Accessing Internal Nexus3 Registers via JTAG/OnCE

Step #	TMS Pin	Description
1	1	IDLE –> SELECT-DR_SCAN
2	0	SELECT-DR_SCAN –> CAPTURE-DR (Nexus command register value loaded in shifter)
3	0	CAPTURE-DR –> SHIFT-DR
4	0	(7) TCK clocks issued to shift in direction (rd/wr) bit and first 6 bits of Nexus reg. addr.
5	1	SHIFT-DR –> EXIT1-DR (7th bit of Nexus reg. shifted in)
6	1	EXIT1-DR –> UPDATE-DR (Nexus shifter is transferred to Nexus command register)
7	1	UPDATE-DR –> SELECT-DR_SCAN
8	0	SELECT-DR_SCAN –> CAPTURE-DR (Register value is transferred to Nexus shifter)
9	0	CAPTURE-DR –> SHIFT-DR
10	0	(31) TCK clocks issued to transfer register value to TDO pin while shifting in TDI value
11	1	SHIFT-DR –> EXIT1-DR (msb of value is shifted in/out of shifter)
12	1	EXIT1-DR –> UPDATE -DR (if access is write, shifter is transferred to register)
13	0	UPDATE-DR–> RUN-TEST/IDLE (transfer complete - Nexus controller to reg. select state)

25.11.17.2 JTAG Sequence for Read Access of Memory-Mapped Resources

Table 25-43. Accessing Memory-Mapped Resources (Reads)

Step #	TCLK clocks	Description
1	13	Nexus Command = write to read/write access address register (RWA)
2	37	Write RWA (initialize starting read address—data input on TDI)
3	13	Nexus Command = write to read/write control/status register (RWCS)
4	37	Write RWCS (initialize read access mode and CNT value—data input on TDI)
5	—	Wait for falling edge of RDY pin
6	13	Nexus Command = read read/write access data register (RWD)
7	37	Read RWD (data output on TDO)
8	—	If CNT > 0, go back to Step #5

25.11.17.3JTAG Sequence for Write Access of Memory-Mapped Resources

Table 25-44. Accessing Memory-Mapped Resources (Writes)

Step #	TCLK clocks	Description
1	13	Nexus Command = write to read/write access control/status register (RWCS)
2	37	Write RWCS (initialize write access mode and CNT value—data input on TDI)
3	13	Nexus Command = write to read/write address register (RWA)
4	37	Write RWA (initialize starting write address—data input on TDI)
5	13	Nexus Command = read read/write access data register (RWD)
6	37	Write RWD (data output on TDO)
7	—	Wait for falling edge of RDY pin
8	—	If CNT > 0, go back to Step #5

25.12 Nexus Crossbar eDMA Interface (NXDM)

The third module of the MPC5553/MPC5554 NDI interface is the e200z6 eDMA Nexus module (NXDM) which is compliant with the Class 3 defined data trace feature of the IEEE®-ISTO 5001-2003 standard. The NXDM can be programmed to trace data accesses for the eDMA module on the system bus. This eDMA module as well as the Nexus module are components of the e200z6 platform. All output messages and register accesses are compliant with the protocol defined in the IEEE®-ISTO 5001 standard.

NOTE

Throughout this section references are made to the auxiliary port and its specific signals, such as MCKO, MSEO[1:0], MDO[12:0] and others. In actual use the MPC5553/MPC5554 NPC module arbitrates the access of the single auxiliary port. To simplify the description of the function of the NXDM module, the interaction of the NPC is omitted and the behavior described as if the module has its own dedicated auxiliary port. The auxiliary port function is fully described in Section 25.2, "External Signal Description."

25.12.1 Block Diagram

Figure 25-52 shows a block diagram of the NXDM.

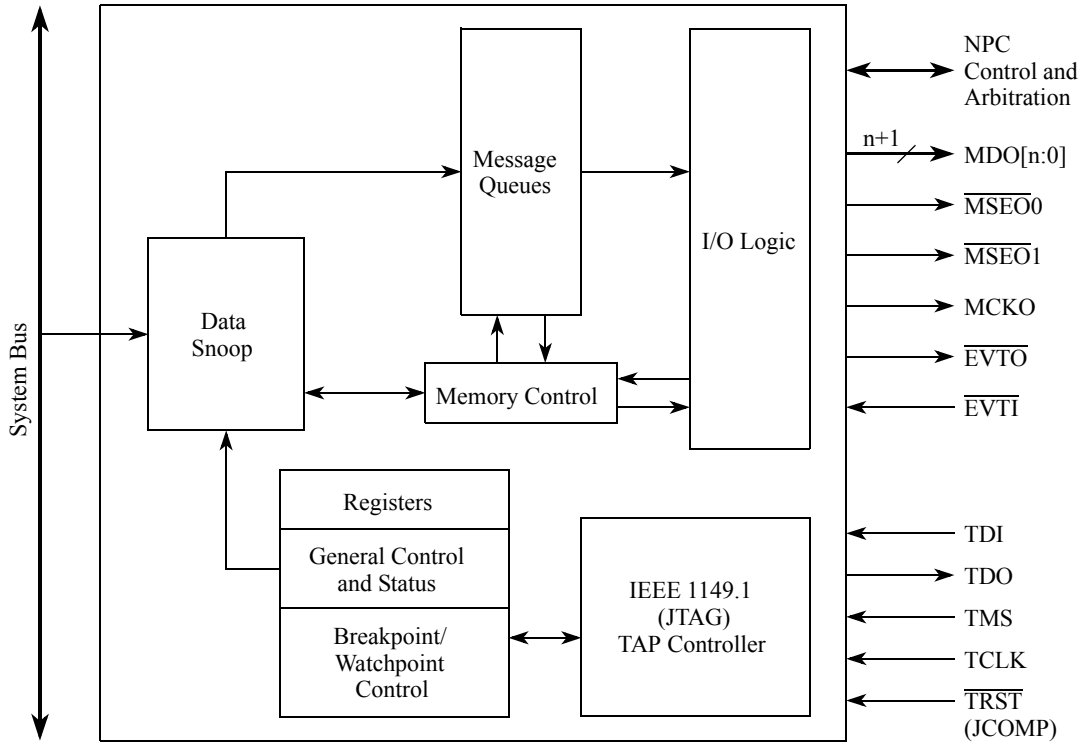


Figure 25-52. NXDM Block Diagram

25.12.2 Features

Feautures include the following:

- Data trace via data write messaging (DWM) and data read messaging (DRM). This provides the capability for the development tool to trace reads and/or writes through the eDMA module to (selected) internal memory resources.
- Watchpoint messaging via the auxiliary pins.
- Watchpoint trigger enable of data trace messaging (DTM).
- Registers for data trace, watchpoint generation, and watchpoint trigger.
- All features controllable and configurable via the JTAG port.
- Power management.
  - Low power design
  - Dynamic power management of FIFOs and control logic

25.13 External Signal Description

The NXDM module uses the same pins and pin protocol as defined in Section 25.2.

25.13.1 Rules for Output Messages

The NXDM module observes the same rules for output messages as the NPC. See Section 25.7.2.2.1, "Rules of Messages."

25.13.2 Auxiliary Port Arbitration

The NXDM module arbitrates for the shared Nexus port. This arbitration is handled by the NPC (See Section 25.5) based on prioritized requests from the NXDM and the other Nexus clients sharing the port.

25.14 NXDM Programmers Model

This section describes the NXDM programmers model. Nexus registers are accessed using the JTAG port in compliance with IEEE® 1149.1. See Chapter 24, "IEEE 1149.1 Test Access Port Controller (JTAGC)" and Section 25.7.2.3 for details on Nexus register access.

25.14.1 NXDM Nexus Register Map

Table 25-45. NXDM Register Map

Nexus Register	Nexus Access Opcode	Read/Write	Read Address	Write Address
Client Select Control (CSC) [1]	0x1	R	0x02	–
Port Configuration Register (PCR) [1]	See NPC	R/W	–	–
Development Control 1 (DC1)	0x2	R/W	0x04	0x05
Development Control 2 (DC2)	0x3	R/W	0x05	0x06
Watchpoint Trigger (WT)	0xB	R/W	0x16	0x17
Data Trace Control (DTC)	0xD	R/W	0x1A	0x1B
Data Trace Start Address 1 (DTSA1)	0xE	R/W	0x1C	0x1D
Data Trace Start Address 2 (DTSA2)	0xF	R/W	0x1E	0x1F

Table 25-45. NXDM Register Map (continued)

Nexus Register	Nexus Access Opcode	Read/Write	Read Address	Write Address
Data Trace End Address 1 (DTEA1)	0x12	R/W	0x24	0x25
Data Trace End Address 2 (DTEA2)	0x13	R/W	0x26	0x27
Breakpoint/Watchpoint Control Register 1 (BWC1)	0x16	R/W	0x2C	0x2D
Breakpoint/Watchpoint Control Register 2 (BWC2)	0x17	R/W	0x2E	0x2F
Breakpoint/Watchpoint Address Register 1 (BWA1)	0x1E	R/W	0x3C	0x3D
Breakpoint/Watchpoint Address Register 2 (BWA2)	0x1F	R/W	0x3E	0x3F
Reserved	0x20–0x3F	–	0x40–0x7E	0x41–0x7F

[1] The CSC and PCR registers are shown in this table as part of the Nexus programmer's model. They are only present at the top level Nexus3 controller (NPC), not in the NXDM module. The device's CSC register is readable through Nexus3; the PCR is shown for reference only.

25.14.2 NXDM Registers

Detailed register definitions for the NXDM implementation are as follows:

25.14.2.1 Development Control Registers (DC1 and DC2)

The development control registers control the basic development features of the NXDM module.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	OPC	MCK_DIV		EOC		0	0	WEN	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	EIC		TM		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-53. Development Control Register 1 (DC1)

Table 25-46. DC1 Field Description

Bit	Name	Description
31	OPC[1]	Output port mode control 0 Reduced port mode configuration 1 Full port mode configuration
30–29	MCK_DIV[1]	MCK_DIV - nexus message clock divide ratio 00 MCKO is 1x system bus clock freq. 01 MCKO is 1/2x system bus clock freq. 10 MCKO is 1/4x system bus clock freq. 11 MCKO is 1/8x system bus clock freq.
28–27	EOC	$\overline{\text{EVTO}}$ control 00 $\overline{\text{EVTO}}$ upon occurrence of watchpoint (internal or external) 01 $\overline{\text{EVTO}}$ upon entry into system-level debug mode (ipg_debug) 1X Reserved
26–25	–	Reserved, read as 0.
24	WEN	Watchpoint trace enable 0 Watchpoint messaging disabled 1 Watchpoint messaging enabled.
23–5	–	Reserved, read as 0.
4–3	EIC	$\overline{\text{EVTI}}$ control 00 $\overline{\text{EVTI}}$ for synchronization (Data Trace) 01 Reserved 10 $\overline{\text{EVTI}}$ disabled for this module 11 Reserved
2–0	TM	Trace mode 000 No Trace 1XX Reserved X1X Data trace enabled XX1 Reserved

[1] The output port mode control bit (OPC) and MCKO divide bits (MCK_DIV) are shown for clarity. These functions are controlled globally by the NPC port control register (PCR).

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	EWC								0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-54. Development Control Register 2 (DC2)

Table 25-47. DC2 Field Description

Bit	Name	Description
31–24	EWC[1]	$\overline{\text{EVTO}}$ Watchpoint Configuration 00000000 = No watchpoints trigger $\overline{\text{EVTO}}$ 1XXXXXXX = Reserved X1XXXXXX = Reserved XX1XXXXX = Reserved XXX1XXXX = Reserved XXXX1XXX = Internal watchpoint #1 triggers $\overline{\text{EVTO}}$ XXXXX1XX = Internal watchpoint #2 triggers $\overline{\text{EVTO}}$ XXXXXX1X = Reserved XXXXXXX1 = Reserved
23–0	–	Reserved, read as 0.

[1] The EOC bits in DC1 must be programmed to trigger $\overline{\text{EVTO}}$ on watchpoint occurence for the EWC bits to have any effect.

25.14.2.2 Watchpoint Trigger Register (WT)

The watchpoint trigger register allows the watchpoints defined internally to the NXDM module to trigger actions. These watchpoints can control data trace enable and disable. The WT bits can be used to produce an address related window for triggering trace messages.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	DTS			DTE			0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-55. Watchpoint Trigger Register (WT)

Table 25-48. WT Field Description

Bit	Name	Description
31–26	–	Reserved, read as 0.
25–23	DTS	DTS - Data trace start control 000 Trigger disabled 001-100 Reserved 101 Use internal watchpoint #1 (BWA1 register) 110 Use internal watchpoint #2 (BWA2 register) 111 Reserved

Table 25-48. WT Field Description (continued)

Bit	Name	Description
22–20	DTE	DTE - Data trace end control 000 Trigger disabled 001-100 Reserved 101 Use internal watchpoint #1 (BWA1 register) 110 Use internal watchpoint #2 (BWA2 register) 111 Reserved
19–0	–	Reserved, read as 0.

NOTE

The WT bits will ONLY enable data trace if the tm bits within the development control register (DC) have not already been set to enable data trace.

25.14.2.3 Data Trace Control Register (DTC)

The data trace control register controls whether DTM Messages are restricted to reads, writes or both for a user programmable address range. There are two data trace channels controlled by the DTC for the NXDM module.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RWT1		RWT2						0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	RC1	RC2	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-56. Data Trace Control Register (DTC)

Table 25-49. DTC Field Description

Bit	Name	Description
31–30	RWT1	Read/write trace 1 00 No trace messages generated X1 Enable data read trace 1X Enable data write trace
29–28	RWT2	Read/write trace 2 00 No trace messages generated X1 Enable data read trace 1X Enable data write trace
27–8	–	Reserved, read as 0.

Table 25-49. DTC Field Description (continued)

Bit	Name	Description
7	RC1	Range control 1 0 Condition trace on address within range (endpoints inclusive) 1 Condition trace on address outside of range (endpoints exclusive)
6	RC2	Range control 2 0 Condition trace on address within range (endpoints inclusive) 1 Condition trace on address outside of range (endpoints exclusive)
5–0	–	Reserved, read as 0.

25.14.2.4 Data Trace Start Address Registers 1 and 2 (DTSA1 and DTSA2)

The data trace start address registers define the start addresses for each trace channel.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DATA TRACE START ADDRESS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DATA TRACE START ADDRESS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-57. Data Trace Start Address Registers (DTSA1, DTSA2)

25.14.2.5 Data Trace End Address Registers 1 and 2 (DTEA1 and DTEA2)

The data trace end address registers define the end addresses for each trace channel.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DATA TRACE END ADDRESS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DATA TRACE END ADDRESS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-58. Data Trace Start Address Registers (DTEA1, DTEA2)

Table 25-50 below illustrates the range that will be selected for data trace for various cases of DTSA being less than, greater than, or equal to DTEA.

Table 25-50. Data Trace - Address Range Options

Programmed Values	Range Control Bit Value	Range Selected
DTSA < or = DTEA	0	DTSA-> <-DTEA
DTSA < or = DTEA	1	<- DTSA DTEA ->
DTSA > DTEA	N/A	Invalid range, no trace

NOTE

DTSA must be less than (or equal to) DTEA in order to guarantee correct data write/read traces. When the range control bit is 0 (internal range), accesses to DTSA and DTEA addresses will be traced. When the range control bit is 1 (external range), accesses to DTSA and DTEA will not be traced.

25.14.2.6 Breakpoint / Watchpoint Control Register 1 (BWC1)

Breakpoint/watchpoint control register 1 controls attributes for generation of NXDM Watchpoint#1.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BWE1		BRW1		0	0	0	0	0	0	0	0	0	0	BWR1	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BWT1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-59. Break / Watchpoint Control Register 1 (BWC1)

Table 25-51. BWC1 Field Description

Bit	Name	Description
31–30	BWE1	Breakpoint/watchpoint #1 enable 00 Internal Nexus watchpoint #1 disabled 01-10 Reserved 11 Internal Nexus watchpoint #1 enabled
29–28	BRW1	Breakpoint/watchpoint #1 read/write select 00 Watchpoint #1 hit on read accesses 01 Watchpoint #1 hit on write accesses 10 Watchpoint #1 on read or write accesses 11 Reserved
27–18	–	Reserved, read as 0.

Table 25-51. BWC1 Field Description (continued)

Bit	Name	Description
17–16	BWR1	Breakpoint/watchpoint #1 register compare 00 No register compare (same as BWC1[31:30] = 2'b00) 01 Reserved 10 Compare with BWA1 value 11 Reserved
15	BWT1	Breakpoint/watchpoint #1 type 0 Reserved 1 Watchpoint #1 on data accesses
14–0	–	Reserved, read as 0.

25.14.2.7 Breakpoint / Watchpoint Control Register 2 (BWC2)

Breakpoint/watchpoint control register2 controls attributes for generation of nxdm watchpoint #2.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BWE2		BRW2		0	0	0	0	0	0	0	0	0	0	BWR2	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BWT2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-60. Break / Watchpoint Control Register 2 (BWC2)

Table 25-52. BWC2 Field Description

Bit	Name	Description
31–30	BWE2	Breakpoint/watchpoint #2 enable 00Internal Nexus watchpoint #2 disabled 01-10 Reserved 11 Internal Nexus watchpoint #2 enabled
29–28	BRW2	Breakpoint/watchpoint #2 read/write select 00 Watchpoint #2 hit on read accesses 01 Watchpoint #2 hit on write accesses 10 Watchpoint #2 on read or write accesses 11 Reserved
27–18	–	Reserved, read as 0.
17–16	BWR2	Breakpoint/watchpoint #2 register compare 00 No register compare (same as BWC1[31:30] = 2'b00) 01 Reserved 10 Compare with BWA2 value 11 Reserved

Table 25-52. BWC2 Field Description (continued)

Bit	Name	Description
15	BWT2	Breakpoint/watchpoint #2 Type 0 Reserved 1 Watchpoint #2 on data accesses
14–0	–	Reserved, read as 0.

25.14.2.8 Breakpoint/Watchpoint Address Registers 1 and 2 (BWA1 and BWA2)

The breakpoint/watchpoint address registers are compared with bus addresses in order to generate internal watchpoints.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BREAKPOINT / WATCHPOINT ADDRESS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BREAKPOINT / WATCHPOINT ADDRESS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-61. Breakpoint / Watchpoint Address Registers (BWA1, BWA2)

25.14.2.9 Unimplemented Registers

Unimplemented registers are those with client select and index value combinations other than those listed in Table 25-45. For unimplemented registers, the NXDM module will drive TDO to zero during the “SHIFT-DR” state. It will also transmit an error message with the invalid access opcode encoding.

25.14.2.10 Programming Considerations (RESET)

If Nexus3 register configuration is to occur during system reset (as opposed to debug mode), all NXDM configuration should be completed between the negation of JCOMP and system reset de-assertion, after the JTAG ID Register has been read by the tool.

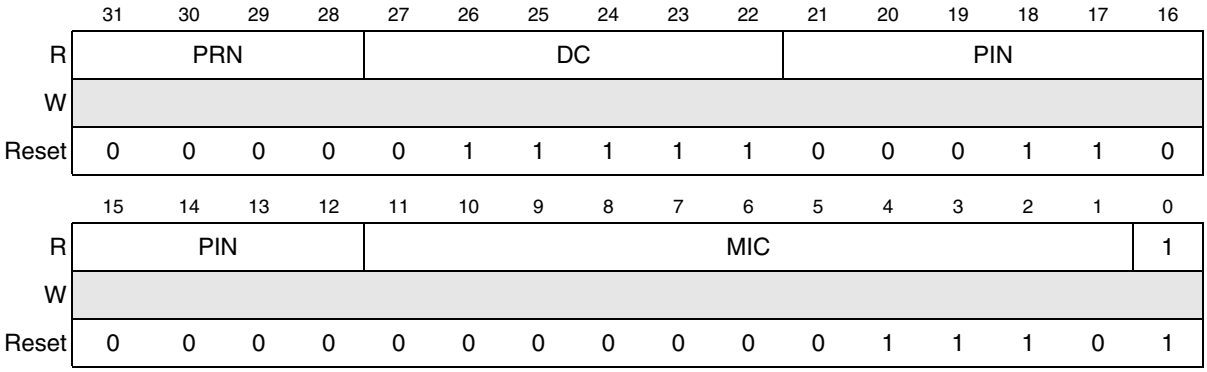
25.14.2.11 IEEE® 1149.1 (JTAG) Test Access Port

The NXDM module uses the IEEE® 1149.1 TAP controller for accessing Nexus resources. The JTAG signals themselves are shared by all TAP controllers on the device. Refer to Chapter 24, “IEEE 1149.1 Test Access Port Controller (JTAGC) for more information on the JTAG interface.

The NXDM module implements a 4-bit instruction register (IR). The valid instructions and method for register access are outlined in Section 25.7.2.3**.**

25.14.2.11.1 NXDM JTAG ID Register

This JTAG ID register that is included in the NXDM module provides key development attributes to the development tool concerning the NXDM block. The register is accessed through the standard JTAG IR/DR paths. See Chapter 23, "Voltage Regulator Controller (VRC) and POR Module."



	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PRN				DC						PIN					
W																
Reset	0	0	0	0	0	1	1	1	1	1	0	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PIN				MIC											1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1

Figure 25-62. NXDM JTAG ID Register

Table 25-53. NXDM JTAG ID Field Descriptions

Bit	Name	Description
31–28	PRN[1]	Embedded part revision number (0x0)
27–22	DC	Freescale design center ID number (0x1F)
21–12	PIN	NXDM module part identification number, defines the features set. (0x60)
11–1	MIC	Manufacturer identity code 0x00E Freescale
0	—	Fixed per JTAG 1149.1 1 Always set

[1] The revision number is initially 0 and could change in the future.

25.14.2.11.2 Enabling the NXDM TAP Controller

Assertion of a power-on-reset signal or assertion of the JCOMP pin resets all TAP controllers on the MPC5553/MPC5554 device. Upon exit from the test-logic-reset state, the IR value is loaded with the JTAG ID. When the NXDM TAP is accessed, this information will help the development tool obtain information about the Nexus module it is accessing, such as version, sequence, feature set, etc.

25.14.2.11.3 NXDM Register Access via JTAG

Access to Nexus register resources is enabled by loading a single instruction (NEXUS_ACCESS) into the JTAG Instruction Register (IR). This IR is part of the IEEE® 1149.1 TAP controller within the NXDM module. See Section 24.4.4, "JTAGC Instructions."

Once the JTAG NEXUS_ACCESS instruction has been loaded, the JTAG port allows tool/target communications with all Nexus registers according to the map in Table 25-45.

Reading/writing of a Nexus register then requires two (2) passes through the data-scan (DR) path of the JTAG state machine (see Chapter 24, "IEEE 1149.1 Test Access Port Controller (JTAGC)").

1. The first pass through the DR selects the Nexus register to be accessed by providing an index (see Table 25-45), and the direction (read/write). This is achieved by loading an 8-bit value into the JTAG data register (DR). This register has the following format:

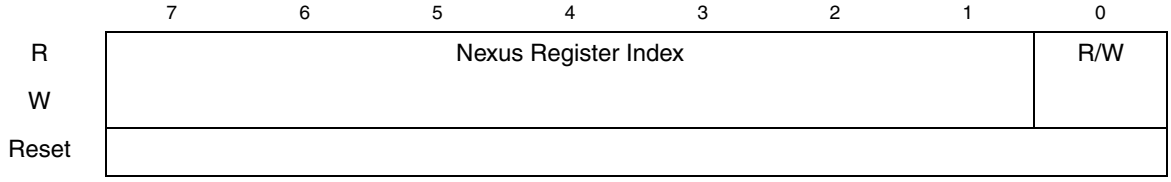


Figure 25-63. JTAG DR for NEXUS Register Access

Table 25-54. DR Read/Write Encoding

Nexus Register Index	Selected from Values in Table 3-1
Read/Write (R/W)	0 Read 1 Write

2. The second pass through the DR then shifts the data in or out of the JTAG port, lsb first.
   a) During a read access, data is latched from the selected Nexus register when the JTAG state machine passes through the capture-DR state.
   b) During a write access, data is latched into the selected Nexus register when the JTAG state machine passes through the update-DR state.

25.14.3 Functional Description

25.14.4 Enabling NXDM Operation

The NXDM module is enabled by loading a single instruction (ACCESS_AUX_TAP_DMAN3, as shown in Table 25-4) into the JTAG instruction register (IR), and then loading the corresponding OnCE OCMD register with the NEXUS_ACCESS instruction (refer to Table 25-5). Once enabled, the module will be ready to accept control input via the JTAG pins.

The Nexus module is disabled when the JTAG state machine reaches the test-logic-reset state. This state can be reached by the assertion of the JCOMP pin or by cycling through the state machine using the TMS pin. The Nexus module will also be disabled if a power-on reset (POR) event occurs.

If the NXDM module is disabled, no trace output will be provided, and the module will disable (drive inactive) auxiliary port output pins (MDO[11:0], MSEO[1:0], MCKO). Nexus registers will not be available for reads or writes.

25.14.5 TCODEs Supported by NXDM

The NXDM pins allow for flexible transfer operations via public messages. A TCODE defines the transfer format, the number and/or size of the packets to be transferred, and the purpose of each packet. The IEEE®-ISTO 5001-2003 standard defines a set of public messages. The NXDM block currently supports the public TCODEs seen in Table 25-55.

Table 25-55. Public TCODEs Supported

Message Name	Packet Size (bits)		Packet Name	Packet Type	Packet Description
	Min	Max			
Data Trace - Date Write Message	6	6	TCODE	Fixed	TCODE number = 5
	4	4	SRC	Fixed	source processor identifier
	3	3	DSZ	Fixed	data size (refer to Table 25-57)
	1	32	U-ADDR	Variable	unique portion of the data write value
	1	64	DATA	Variable	data write value
Data Trace - Data Read Message	6	6	TCODE	Fixed	TCODE number = 6
	4	4	SRC	Fixed	source processor identifier
	3	3	DSZ	Fixed	data size (refer to Table 25-57)
	1	32	U-ADDR	Variable	unique portion of the data read value
	1	64	DATA	Variable	data read value
Error Message	6	6	TCODE	Fixed	TCODE number = 8
	4	4	SRC	Fixed	source processor identifier (mulitple Nexus configuration)
	5	5	ECODE	Fixed	error code (refer to Table 25-56)
Data Trace - Data Write Message w/ Sync	6	6	TCODE	Fixed	TCODE number = 13 (0xD)
	4	4	SRC	Fixed	source processor identifier (mulitple Nexus configuration)
	3	3	DSZ	Fixed	data size (refer to Table 25-57)
	1	32	F-ADDR	Variable	full access address (leading zero (0) truncated)
	1	64	DATA	Variable	data write value
Data Trace - Data Read Message w/ Sync	6	6	TCODE	Fixed	TCODE number = 14 (0xE)
	4	4	SRC	Fixed	source processor identifier (mulitple Nexus configuration)
	3	3	DSZ	Fixed	data size (refer to Table 25-57)
	1	32	F-ADDR	Variable	full access address (leading zero (0) truncated)
	1	64	DATA	Variable	data read valued
Watchpoint Message	6	6	TCODE	Fixed	TCODE number = 15 (0xF)
	4	4	SRC	Fixed	source processor identifier (mulitple Nexus configuration)
	4	4	WPHIT	Fixed	# indicating watchpoint sources

Table 25-56. Error Code (ECODE) Encoding (TCODE = 8)

Error Code (ECODE)	Description
00000	Reserved
00001	Reserved
00010	Data Trace overrun
00011	Reserved
00100	Reserved
00101	Invalid access opcode (Nexus Register unimplemented)
00110	Watchpoint overrun
00111	Reserved
01000	Data Trace and Watchpoint overrun
01001–11111	Reserved

Table 25-57. Data Trace Size (DSZ) Encodings (TCODE = 5,6,13,14)

DTM Size Encoding	Transfer Size
000	Byte
001	Halfword (2 bytes)
010	Word (4 bytes)
011	Doubleword (8 bytes)
100–111	Reserved

25.14.5.1 Data Trace

This section deals with the data trace mechanism supported by the NXDM module. Data trace is implemented via data write messaging (DWM) and data read messaging (DRM).

25.14.5.2 Data Trace Messaging (DTM)

NXDM data trace messaging is accomplished by snooping the NXDM data bus, and storing the information for qualifying accesses (based on enabled features and matching target addresses). The NXDM module traces all data access that meet the selected range and attributes.

NOTE

Data trace is ONLY performed on DMA accesses to the system bus.

25.14.5.3 DTM Message Formats

The NXDM block supports five types of DTM Messages — data write, data read, data write synchronization, data read synchronization and error messages.

25.14.5.3.1 Data Write and Data Read Messages

The data write and data read messages contain the data write/read value and the address of the write/read access, relative to the previous data trace message. Data write message and data read message information is messaged out in the following format:

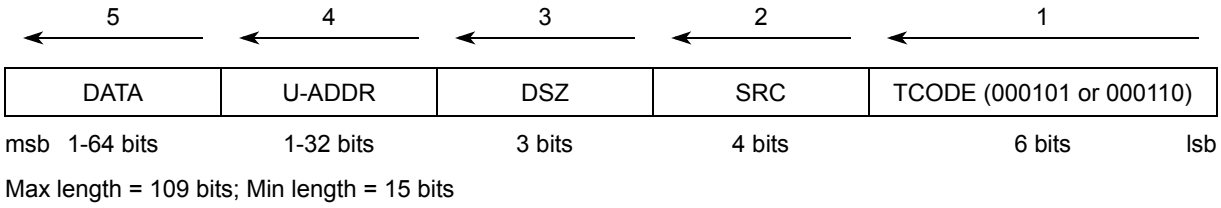


Figure 25-64. Data Write/Read Message Format

25.14.5.3.2 DTM Overflow Error Messages

An error message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard incoming messages until it has completely emptied the queue. Once emptied, an error message will be queued. The error encoding will indicate which types of messages attempted to be queued while the FIFO was being emptied.

If only a data trace message attempts to enter the queue while it is being emptied, the error message will incorporate the data trace only error encoding (00010). If a watchpoint also attempts to be queued while the fifo is being emptied, then the error message will incorporate error encoding (01000).

Error information is messaged out in the following format:

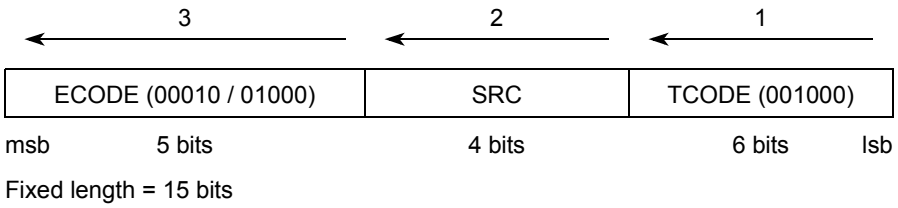


Figure 25-65. Error Message Format

25.14.5.3.3 Data Trace Synchronization Messages

A data trace write/read w/ sync. Message is messaged via the auxiliary port (provided data trace is enabled) for the following conditions (see Table 25-58):

- Initial data trace message upon exit from system reset or whenever data trace is enabled will be a synchronization message.
- Upon returning from a low power state, the first data trace message will be a synchronization message.
- Upon returning from debug mode, the first data trace message will be a synchronization message.
- After occurrence of queue overrun (can be caused by any trace message), the first data trace message will be a synchronization message.
- After the periodic data trace counter has expired indicating 255 *without-sync* data trace messages have occurred since the last *with-sync* message occurred.
- Upon assertion of the Event In ($\overline{\text{EVTI}}$) pin, the first data trace message will be a synchronization message if the eic bits of the dc register have enabled this feature.

- Upon data trace write/read after the previous dtm message was lost due to an attempted access to a secure memory location.
- Upon data trace write/read after the previous dtm message was lost due to a collision entering the fifo between the dtm message and any of the following: error message, or watchpoint message.

Data Trace synchronization messages provide the full address (without leading zeros) and insure that development tools fully synchronize with data trace regularly. Synchronization messages provide a reference address for subsequent DTMs, in which only the unique portion of the data trace address is transmitted. The format for data trace write/read w/ sync. Messages is as follows:

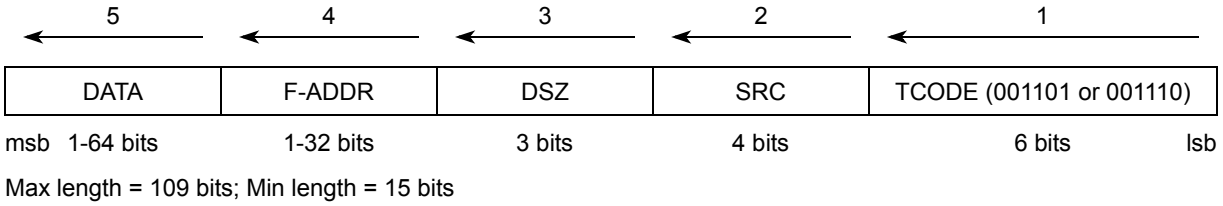


Figure 25-66. Data Write/Read w/ Sync Message Format

Exception conditions that result in data trace synchronization are summarized in Table 25-58., "Data Trace Exception Summary."

Table 25-58. Data Trace Exception Summary

Exception Condition	Exception Handling
System Reset Negation	At the negation of JTAG reset (JCOMP), queue pointers, counters, state machines, and registers within the NXDM module are reset. If data trace is enabled, the first data trace message is a data write/read w/ sync. message.
Data Trace Enabled	The first data trace message (after data trace has been enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a low power mode or debug mode the next data trace message will be converted to a data write/read w/ sync. message.
Queue Overrun	An error message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard messages until it has completely emptied the queue. Once emptied, an error message will be queued. The error encoding will indicate which types of messages attempted to be queued while the FIFO was being emptied. The next DTM message in the queue will be a data write/read w/ sync. message.
Periodic Data Trace Synchronization	A forced synchronization occurs periodically after 255 data trace messages have been queued. A data write/read w/ sync. message is queued. The periodic data trace message counter then resets.
Event In	if the nexus module is enabled, an $\overline{\text{evti}}$ assertion initiates a data trace write/read w/ sync. message upon the next data write/read (if data trace is enabled and the eic bits of the dc register have enabled this feature).
Attempted Access to Secure Memory	Any attempted read or write to secure memory locations will temporarily disable data trace & cause the corresponding DTM to be lost. A subsequent read/write will queue a data trace read/write w/ sync. message.
Collision Priority	All messages have the following priority: Error -> WPM -> DTM. A DTM message which attempts to enter the queue at the same time as an error message, or watchpoint message will be lost. A subsequent read/write will queue a data trace read/write w/ sync. message.

25.14.5.4 DTM Operation

25.14.5.4.1 Enabling Data Trace Messaging

Data trace messaging can be enabled in one of two ways.

- Setting the DC1[TM] field to enable data trace
- Using the WT[DTS] field to enable data trace on watchpoint hits

25.14.5.4.2 DTM Queueing

NXDM implements a programmable depth queue for queuing all messages. Messages that enter the queue are transmitted via the auxiliary pins in the order in which they are queued.

NOTE

If multiple trace messages need to be queued at the same time, watchpoint messages will have the highest priority (WPM -> DTM).

25.14.5.4.3 Relative Addressing

The relative address feature is compliant with IEEE®-ISTO Nexus 5001-2003 and is designed to reduce the number of bits transmitted for addresses of data trace messages. Relative addressing is the same as described for the NZ6C3 in Section 25.11.12.3.2, “ Relative Addressing.”

25.14.5.4.4 Data Trace Windowing

Data write/read messages are enabled via the RWT1(2) field in the data trace control register (DTC) for each DTM channel. Data trace windowing is achieved via the address range defined by the DTEA and DTSA registers and by the RC1(2) field in the DTC. All eDMA initiated read/write accesses which fall inside or outside these address ranges, as programmed, are candidates to be traced.

25.14.5.4.5 System Bus Cycle Special Cases

Table 25-59. System Bus Cycle Special Cases

Special Case	Action
System bus cycle aborted (DABORT asserted)	Cycle ignored
System bus cycle with data error	Data Trace Message discarded
System bus cycle completed without error	Cycle captured and transmitted
System bus cycle is an instruction fetch	Cycle ignored

25.14.5.5 Data Trace Timing Diagrams (8 MDO configuration)

Data trace timing for the NXDM is the same as for the NZ6C3. See Section Section 25.11.13.4, “ Data Trace Timing Diagrams (8 MDO Configuration).”

25.14.6 Watchpoint Support

The NXDM module provides watchpoint messaging via the auxiliary pins, as defined by IEEE®-ISTO 5001-2003.

Watchpoint messages can be generated using the NXDM defined internal watchpoints.

25.14.6.1 Watchpoint Messaging

Enabling watchpoint messaging is accomplished by setting the watchpoint messaging enable bit, DC1[WEN]. Using the BWC1 and BWC2 registers, two independently controlled internal watchpoints can be initialized. When a DMA access address matches on BWA1 or BWA2, a watchpoint message will be transmitted.

The Nexus module provides watchpoint messaging using the TCODE. When either of the two possible watchpoint sources asserts, a message will be sent to the queue to be messaged out. This message indicates the watchpoint number.

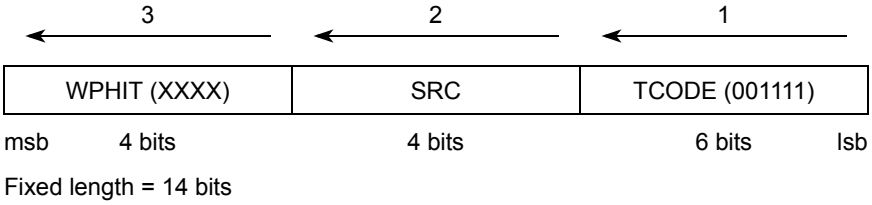


Figure 25-67. Watchpoint Message Format

Table 25-60. Watchpoint Source Description

Watchpoint Source (4 bits)	Watchpoint Description
XXX1	Reserved
XX1X	Reserved
X1XX	Internal Watchpoint #1 (BWA1 match)
1XXX	Internal Watchpoint #2 (BWA2 match)

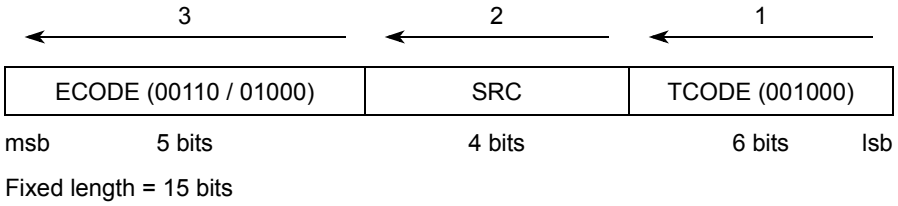
25.14.6.2 Watchpoint Error Message

An error message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard messages until it has completely emptied the queue. Once emptied, an error message will be queued. The error encoding will indicate which types of messages attempted to be queued while the FIFO was being emptied.

If only a watchpoint message attempts to enter the queue while it is being emptied, the error message will incorporate the watchpoint only error encoding (00110). If a data trace message also attempts to enter the queue while it is being emptied, the error message will incorporate error encoding (01000).

Error information is messaged out in the following format (see Figure 25-68).

Figure 25-68. Error Message Format



25.15 Revision History

Substantive Changes since Rev 3.0
Fixed title of Table 25-52 to be BWC2 rather than BWC1.
Table 25-18: fixed cross references to other sections, removed notes about Multiple Nesux Configurations since all parts do have multiple Nexus modules.

Appendix A
MPC5553/MPC5554 Register Map

Table A-1. Module Base Addresses

Module	Base Address	Page
Peripheral Bridge A (PBRIDGEA)	0xC3F0_0000	Page A-2
Frequency Modulated Phase-Locked Loop (FMPLL)	0xC3F8_0000	Page A-2
External Bus Interface (EBI)	0xC3F8_4000	Page A-2
Flash Module and Flash Bus Interface Unit (FLASH)	0xC3F8_8000	Page A-3
System Integration Unit (SIU)	0xC3F9_0000	Page A-3
Enhanced Modular Input/Output Subsystem (eMIOS)	0xC3FA_0000	Page A-25
Enhanced Time Processing Unit (eTPU)	0xC3FC_0000	Page A-25
Peripheral Bridge B (PBRIDGEB)	0xFFF0_0000	Page A-35
System Bus Crossbar Switch (XBAR)	0xFFF0_4000	Page A-36
Error Correction Status Module (ECSM)	0xFFF4_0000	Page A-37
Enhanced Direct Memory Access (eDMA)	0xFFF4_4000	Page A-38
Interrupt Controller (INTC)	0xFFF4_8000	Page A-42
Fast Ethernet Controller (FEC) — in MPC5553 only	0xFFF4_C000	Page A-53
Enhanced Queued Analog-to-Digital Converter (eQADC)	0xFFF8_0000	Page A-54
Deserial / Serial Peripheral Interface (DSPIx)	0xFFF9_0000 (DSPI A)[1] 0xFFF9_4000 (DSPI B) 0xFFF9_8000 (DSPI C) 0xFFF9_C000 (DSPI D)	Page A-58
Enhanced Serial Communication Interface (eSCIx)	0xFFFB_0000 (A) 0xFFFB_4000 (B)	Page A-59
FlexCAN2 Controller Area Network (CANx)	0xFFFC_0000 (FlexCAN A) 0xFFFC_4000 (FlexCAN B)[1] 0xFFFC_8000 (FlexCAN C)	Page A-59
Boot Assist Module (BAM)	0xFFFF_C000	Page A-60

[1] MPC5554 Only

Table A-2. MPC5554 / MPC5553 Detailed Register Map

Register Description	Register Name	Used Size	Address	Reference
Peripheral Bridge A (PBRIDGEA)			0xC3F0_0000	Chapter 5, "Peripheral Bridge (PBRIDGE_A, PBRIDGE_B)"
Peripheral bridge A master privilege control register	PBRIDGEA_MPCR	32-bit	Base + 0x0000	
Reserved	—	—	Base + (0x0004-0x001F)	
Peripheral bridge A peripheral access control register 0	PBRIDGEA_PACR0	32-bit	Base + 0x0020	
Reserved	—	—	Base + (0x0024-0x003F)	
Peripheral bridge A off-platform peripheral access control register 0	PBRIDGEA_OPACR0	32-bit	Base + 0x0040	
Peripheral bridge A off-platform peripheral access control register 1	PBRIDGEA_OPACR1	32-bit	Base + 0x0044	
Peripheral bridge A off-platform peripheral access control register 2	PBRIDGEA_OPACR2	32-bit	Base + 0x0048	
Reserved	—	—	Base + (0x004C-0xC3F7_FFFF)	
Frequency Modulated Phase-Locked Loop (FMPLL)			0xC3F8_0000	Chapter 11, "Frequency Modulated Phase Locked Loop (FMPLL) and System Clocks"
Synthesizer control register	FMPLL_SYNCR	32-bit	Base + 0x0000	
Synthesizer status register	FMPLL_SYNSR	32-bit	Base + 0x0004	
Reserved	—	—	(Base + 0x0008)-0xC3F8_3FFF	
External Bus Interface (EBI)			0xC3F8_4000	Chapter 12, "External Bus Interface (EBI)"
Module configuration register	EBI_MCR	32-bit	Base + 0x0000	
Reserved	—	—	Base + (0x0004-0x0007)	
Transfer error status register	EBI_TESR	32-bit	Base + 0x0008	
Bus monitor control register	EBI_BMCR	32-bit	Base + 0x000C	
Base register bank 0	EBI_BR0	32-bit	Base + 0x0010	
Option register bank 0	EBI_OR0	32-bit	Base + 0x0014	
Base register bank 1	EBI_BR1	32-bit	Base + 0x0018	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
Option register bank 1	EBI_OR1	32-bit	Base + 0x001C	
Base register bank 2	EBI_BR2	32-bit	Base + 0x0020	
Option register bank 2	EBI_OR2	32-bit	Base + 0x0024	
Base register bank 3	EBI_BR3	32-bit	Base + 0x0028	
Option register bank 3	EBI_OR3	32-bit	Base + 0x002C	
EBI Calibration Base Register Bank 0	EBI_CAL_BR0	32-bit	Base + 0x0040	
EBI Calibration Option Register Bank 0	EBI_CAL_OR0	32-bit	Base + 0x0044	
EBI Calibration Base Register Bank 1	EBI_CAL_BR1	32-bit	Base + 0x0048	
EBI Calibration Option Register Bank 1	EBI_CAL_OR1	32-bit	Base + 0x004C	
EBI Calibration Base Register Bank 2	EBI_CAL_BR2	32-bit	Base + 0x0050	
EBI Calibration Option Register Bank 2	EBI_CAL_OR2	32-bit	Base + 0x0054	
EBI Calibration Base Register Bank 3	EBI_CAL_BR3	32-bit	Base + 0x0058	
EBI Calibration Option Register Bank 3	EBI_CAL_OR3	32-bit	Base + 0x005C	
Flash Module and Flash Bus Interface Unit (FLASH)			0xC3F8_8000	Chapter 13, “Flash Memory”
Module configuration register	FLASH_MCR	32-bit	Base + 0x0000	
Low/mid address space block locking register	FLASH_LMLR	32-bit	Base + 0x0004	
High address space block locking register	FLASH_HLR	32-bit	Base + 0x0008	
Secondary low/mid address space block locking register	FLASH_SLMLR	32-bit	Base + 0x000C	
Low/mid address block select register	FLASH_LMSR	32-bit	Base + 0x0010	
High address space block select register	FLASH_HSR	32-bit	Base + 0x0014	
Address register	FLASH_AR	32-bit	Base + 0x0018	
Bus interface unit control register	FLASH_BIUCR	32-bit	Base + 0x001C	
Bus interface unit access protection register	FLASH_BIUAPR	32-bit	Base + 0x0020	
Reserved	—	—	(Base + 0x0024)-0xC3F8_FFFF)	
System Integration Unit (SIU)			0xC3F9_0000	Chapter 6, “System Integration Unit (SIU)”
MCU ID Register	SIU_MIDR		Base + 0x0004	
Reserved	—	—	Base + (0x0008-0x000B)	
Reset status register	SIU_RSR		Base + 0x000C	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
System reset control register	SIU_SRCR		Base + 0x0010	
External interrupt status register	SIU_EISR		Base + 0x0014	
DMA/Interrupt request enable register	SIU_DIRER		Base + 0x0018	
DMA/Interrupt request status register	SIU_DIRSR		Base + 0x001C	
Overrun status register	SIU_OSR		Base + 0x0020	
Overrun request enable register	SIU_ORER		Base + 0x0024	
IRQ rising-edge event enable register	SIU_IREER		Base + 0x0028	
IRQ falling-edge event enable register	SIU_IFEER		Base + 0x002C	
IRQ digital filter register	SIU_IDFR		Base + 0x0030	
Reserved	—	—	Base + (0x0034-0x003F)	
Pad configuration register 0 ($\overline{CS0}$)	SIU_PCR0	16-bits	Base + 0x0040	
Pad configuration register 1 ($\overline{CS1}$)	SIU_PCR1	16-bits	Base + 0x0042	
Pad configuration register 2 ($\overline{CS2}$)	SIU_PCR2	16-bits	Base + 0x0044	
Pad configuration register 3 ($\overline{CS3}$)	SIU_PCR3	16-bits	Base + 0x0046	
Pad configuration register 4 (ADDR8)	SIU_PCR4	16-bits	Base + 0x0048	
Pad configuration register 5 (ADDR9)	SIU_PCR5	16-bits	Base + 0x004A	
Pad configuration register 6 (ADDR10)	SIU_PCR6	16-bits	Base + 0x004C	
Pad configuration register 7 (ADDR11)	SIU_PCR7	16-bits	Base + 0x004E	
Pad configuration register 8 (ADDR12)	SIU_PCR8	16-bits	Base + 0x0050	
Pad configuration register 9 (ADDR13)	SIU_PCR9	16-bits	Base + 0x0052	
Pad configuration register 10 (ADDR14)	SIU_PCR10	16-bits	Base + 0x0054	
Pad configuration register 11 (ADDR15)	SIU_PCR11	16-bits	Base + 0x0056	
Pad configuration register 12 (ADDR16)	SIU_PCR12	16-bits	Base + 0x0058	
Pad configuration register 13 (ADDR17)	SIU_PCR13	16-bits	Base + 0x005A	
Pad configuration register 14 (ADDR18)	SIU_PCR14	16-bits	Base + 0x005C	
Pad configuration register 15 (ADDR19)	SIU_PCR15	16-bits	Base + 0x005E	
Pad configuration register 16 (ADDR20)	SIU_PCR16	16-bits	Base + 0x0060	
Pad configuration register 17 (ADDR21)	SIU_PCR17	16-bits	Base + 0x0062	
Pad configuration register 18 (ADDR22)	SIU_PCR18	16-bits	Base + 0x0064	
Pad configuration register 19 (ADDR23)	SIU_PCR19	16-bits	Base + 0x0066	
Pad configuration register 20 (ADDR24)	SIU_PCR20	16-bits	Base + 0x0068	
Pad configuration register 21 (ADDR25)	SIU_PCR21	16-bits	Base + 0x006A	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
Pad configuration register 22 (ADDR26)	SIU_PCR22	16-bits	Base + 0x006C	
Pad configuration register 23 (ADDR27)	SIU_PCR23	16-bits	Base + 0x006E	
Pad configuration register 24 (ADDR28)	SIU_PCR24	16-bits	Base + 0x0070	
Pad configuration register 25 (ADDR29)	SIU_PCR25	16-bits	Base + 0x0072	
Pad configuration register 26 (ADDR30)	SIU_PCR26	16-bits	Base + 0x0074	
Pad configuration register 27 (ADDR31)	SIU_PCR27	16-bits	Base + 0x0076	
Pad configuration register 28 (DATA0)	SIU_PCR28	16-bits	Base + 0x0078	
Pad configuration register 29 (DATA1)	SIU_PCR29	16-bits	Base + 0x007A	
Pad configuration register 30 (DATA2)	SIU_PCR30	16-bits	Base + 0x007C	
Pad configuration register 31 (DATA3)	SIU_PCR31	16-bits	Base + 0x007E	
Pad configuration register 32 (DATA4)	SIU_PCR32	16-bits	Base + 0x0080	
Pad configuration register 33 (DATA5)	SIU_PCR33	16-bits	Base + 0x0082	
Pad configuration register 34 (DATA6)	SIU_PCR34	16-bits	Base + 0x0084	
Pad configuration register 35 (DATA7)	SIU_PCR35	16-bits	Base + 0x0086	
Pad configuration register 36 (DATA8)	SIU_PCR36	16-bits	Base + 0x0088	
Pad configuration register 37 (DATA9)	SIU_PCR37	16-bits	Base + 0x008A	
Pad configuration register 38 (DATA10)	SIU_PCR38	16-bits	Base + 0x008C	
Pad configuration register 39 (DATA11)	SIU_PCR39	16-bits	Base + 0x008E	
Pad configuration register 40 (DATA12)	SIU_PCR40	16-bits	Base + 0x0090	
Pad configuration register 41 (DATA13)	SIU_PCR41	16-bits	Base + 0x0092	
Pad configuration register 42 (DATA14)	SIU_PCR42	16-bits	Base + 0x0094	
Pad configuration register 43 (DATA15)	SIU_PCR43	16-bits	Base + 0x0096	
Pad configuration register 44 (DATA16)	SIU_PCR44	16-bits	Base + 0x0098	
Pad configuration register 45 (DATA17)	SIU_PCR45	16-bits	Base + 0x009A	
Pad configuration register 46 (DATA18)	SIU_PCR46	16-bits	Base + 0x009C	
Pad configuration register 47 (DATA19)	SIU_PCR47	16-bits	Base + 0x009E	
Pad configuration register 48 (DATA20)	SIU_PCR48	16-bits	Base + 0x00A0	
Pad configuration register 49 (DATA21)	SIU_PCR49	16-bits	Base + 0x00A2	
Pad configuration register 50 (DATA22)	SIU_PCR50	16-bits	Base + 0x00A4	
Pad configuration register 51 (DATA23)	SIU_PCR51	16-bits	Base + 0x00A6	
Pad configuration register 52 (DATA24)	SIU_PCR52	16-bits	Base + 0x00A8	
Pad configuration register 53 (DATA25)	SIU_PCR53	16-bits	Base + 0x00AA	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
Pad configuration register 54 (DATA26)	SIU_PCR54	16-bits	Base + 0x00AC	
Pad configuration register 55 (DATA27)	SIU_PCR55	16-bits	Base + 0x00AE	
Pad configuration register 56 (DATA28)	SIU_PCR56	16-bits	Base + 0x00B0	
Pad configuration register 57 (DATA29)	SIU_PCR57	16-bits	Base + 0x00B2	
Pad configuration register 58 (DATA30)	SIU_PCR58	16-bits	Base + 0x00B4	
Pad configuration register 59 (DATA31)	SIU_PCR59	16-bits	Base + 0x00B6	
Pad configuration register 60 (TSIZ0)	SIU_PCR60	16-bits	Base + 0x00B8	
Pad configuration register 61 (TSIZ1)	SIU_PCR61	16-bits	Base + 0x00BA	
Pad configuration register 62 (RD_$\overline{WR}$)	SIU_PCR62	16-bits	Base + 0x00BC	
Pad configuration register 63 ($\overline{BDIP}$)	SIU_PCR63	16-bits	Base + 0x00BE	
Pad configuration register 64 ($\overline{WE0}$)	SIU_PCR64	16-bits	Base + 0x00C0	
Pad configuration register 65 ($\overline{WE1}$)	SIU_PCR65	16-bits	Base + 0x00C2	
Pad configuration register 66 ($\overline{WE2}$)	SIU_PCR66	16-bits	Base + 0x00C4	
Pad configuration register 67 ($\overline{WE3}$)	SIU_PCR67	16-bits	Base + 0x00C6	
Pad configuration register 68 ($\overline{OE}$)	SIU_PCR68	16-bits	Base + 0x00C8	
Pad configuration register 69 ($\overline{TS}$)	SIU_PCR69	16-bits	Base + 0x00CA	
Pad configuration register 70 ($\overline{TA}$)	SIU_PCR70	16-bits	Base + 0x00CC	
Pad configuration register 71 ($\overline{TEA}$)	SIU_PCR71	16-bits	Base + 0x00CE	
Pad configuration register 72 ($\overline{BR}$)	SIU_PCR72	16-bits	Base + 0x00D0	
Pad configuration register 73 ($\overline{BG}$)	SIU_PCR73	16-bits	Base + 0x00D2	
Pad configuration register 74 ($\overline{BB}$)	SIU_PCR74	16-bits	Base + 0x00D4	
Pad configuration register 75 (MDO4)	SIU_PCR75	16-bits	Base + 0x00D6	
Pad configuration register 76 (MDO5)	SIU_PCR76	16-bits	Base + 0x00D8	
Pad configuration register 77 (MDO6)	SIU_PCR77	16-bits	Base + 0x00DA	
Pad configuration register 78 (MDO7)	SIU_PCR78	16-bits	Base + 0x00DC	
Pad configuration register 79 (MDO8)	SIU_PCR79	16-bits	Base + 0x00DE	
Pad configuration register 80 (MDO9)	SIU_PCR80	16-bits	Base + 0x00E0	
Pad configuration register 81 (MDO10)	SIU_PCR81	16-bits	Base + 0x00E2	
Pad configuration register 82 (MDO11)	SIU_PCR82	16-bits	Base + 0x00E4	
Pad configuration register 83 (CNTXA)	SIU_PCR83	16-bits	Base + 0x00E6	
Pad configuration register 84 (CNRXA)	SIU_PCR84	16-bits	Base + 0x00E8	
Pad configuration register 85 (CNTXB)	SIU_PCR85	16-bits	Base + 0x00EA	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
Pad configuration register 86 (CNRXB)	SIU_PCR86	16-bits	Base + 0x00EC	
Pad configuration register 87 (CNTXC)	SIU_PCR87	16-bits	Base + 0x00EE	
Pad configuration register 88 (CNRXC)	SIU_PCR88	16-bits	Base + 0x00F0	
Pad configuration register 89 (TXDA)	SIU_PCR89	16-bits	Base + 0x00F2	
Pad configuration register 90 (RXDA)	SIU_PCR90	16-bits	Base + 0x00F4	
Pad configuration register 91 (TXDB)	SIU_PCR91	16-bits	Base + 0x00F6	
Pad configuration register 92 (RXDB)	SIU_PCR92	16-bits	Base + 0x00F8	
Pad configuration register 93 (SCKA)	SIU_PCR93	16-bits	Base + 0x00FA	
Pad configuration register 94 (SINA)	SIU_PCR94	16-bits	Base + 0x00FC	
Pad configuration register 95 (SOUTA)	SIU_PCR95	16-bits	Base + 0x00FE	
Pad configuration register 96 (PCSA0)	SIU_PCR96	16-bits	Base + 0x0100	
Pad configuration register 97 (PCSA1)	SIU_PCR97	16-bits	Base + 0x0102	
Pad configuration register 98 (PCSA2)	SIU_PCR98	16-bits	Base + 0x0104	
Pad configuration register 99 (PCSA3)	SIU_PCR99	16-bits	Base + 0x0106	
Pad configuration register 100 (PCSA4)	SIU_PCR100	16-bits	Base + 0x0108	
Pad configuration register 101 (PCSA6)	SIU_PCR101	16-bits	Base + 0x010A	
Pad configuration register 102 (SCKB)	SIU_PCR102	16-bits	Base + 0x010C	
Pad configuration register 103 (SINB)	SIU_PCR103	16-bits	Base + 0x010E	
Pad configuration register 104 (SOUTB)	SIU_PCR104	16-bits	Base + 0x0110	
Pad configuration register 105 (PCSB0)	SIU_PCR105	16-bits	Base + 0x0112	
Pad configuration register 106 (PCSB1)	SIU_PCR106	16-bits	Base + 0x0114	
Pad configuration register 107 (PCSB2)	SIU_PCR107	16-bits	Base + 0x0116	
Pad configuration register 108 (PCSB3)	SIU_PCR108	16-bits	Base + 0x0118	
Pad configuration register 109 (PCSB4)	SIU_PCR109	16-bits	Base + 0x011A	
Pad configuration register 110 (PCSB5)	SIU_PCR110	16-bits	Base + 0x011C	
Pad configuration register 111 (ETRIG0)	SIU_PCR9111	16-bits	Base + 0x011E	
Pad configuration register 112 (ETRIG1)	SIU_PCR112	16-bits	Base + 0x0120	
Pad configuration register 113 (TCRCLKA)	SIU_PCR113	16-bits	Base + 0x0122	
Pad configuration register 114 (ETPUA0)	SIU_PCR114	16-bits	Base + 0x0124	
Pad configuration register 115 (ETPUA1)	SIU_PCR115	16-bits	Base + 0x0126	
Pad configuration register 116 (ETPUA2)	SIU_PCR116	16-bits	Base + 0x0128	
Pad configuration register 117 (ETPUA3)	SIU_PCR117	16-bits	Base + 0x012A	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
Pad configuration register 118 (ETPUA4)	SIU_PCR118	16-bits	Base + 0x012C	
Pad configuration register 119 (ETPUA5)	SIU_PCR119	16-bits	Base + 0x012E	
Pad configuration register 120 (ETPUA6)	SIU_PCR120	16-bits	Base + 0x0130	
Pad configuration register 121 (ETPUA7)	SIU_PCR121	16-bits	Base + 0x0132	
Pad configuration register 122 (ETPUA8)	SIU_PCR122	16-bits	Base + 0x0134	
Pad configuration register 123 (ETPUA9)	SIU_PCR123	16-bits	Base + 0x0136	
Pad configuration register 124 (ETPUA10)	SIU_PCR124	16-bits	Base + 0x0138	
Pad configuration register 125 (ETPUA11)	SIU_PCR125	16-bits	Base + 0x013A	
Pad configuration register 126 (ETPUA12)	SIU_PCR126	16-bits	Base + 0x013C	
Pad configuration register 127 (ETPUA13)	SIU_PCR127	16-bits	Base + 0x013E	
Pad configuration register 128 (ETPUA14)	SIU_PCR128	16-bits	Base + 0x0140	
Pad configuration register 129 (ETPUA15)	SIU_PCR129	16-bits	Base + 0x0142	
Pad configuration register 130 (ETPUA16)	SIU_PCR130	16-bits	Base + 0x0144	
Pad configuration register 131 (ETPUA17)	SIU_PCR131	16-bits	Base + 0x0146	
Pad configuration register 132 (ETPUA18)	SIU_PCR132	16-bits	Base + 0x0148	
Pad configuration register 133 (ETPUA19)	SIU_PCR133	16-bits	Base + 0x014A	
Pad configuration register 134 (ETPUA20)	SIU_PCR134	16-bits	Base + 0x014C	
Pad configuration register 135 (ETPUA21)	SIU_PCR135	16-bits	Base + 0x014E	
Pad configuration register 136 (ETPUA22)	SIU_PCR136	16-bits	Base + 0x0150	
Pad configuration register 137 (ETPUA23)	SIU_PCR137	16-bits	Base + 0x0152	
Pad configuration register 138 (ETPUA24)	SIU_PCR138	16-bits	Base + 0x0154	
Pad configuration register 139 (ETPUA25)	SIU_PCR139	16-bits	Base + 0x0156	
Pad configuration register 140 (ETPUA26)	SIU_PCR140	16-bits	Base + 0x0158	
Pad configuration register 141 (ETPUA27)	SIU_PCR141	16-bits	Base + 0x015A	
Pad configuration register 142 (ETPUA28)	SIU_PCR142	16-bits	Base + 0x015C	
Pad configuration register 143 (ETPUA29)	SIU_PCR143	16-bits	Base + 0x015E	
Pad configuration register 144 (ETPUA30)	SIU_PCR144	16-bits	Base + 0x0160	
Pad configuration register 145 (ETPUA31)	SIU_PCR145	16-bits	Base + 0x0162	
Pad configuration register 146 (TCRCLKB)	SIU_PCR146	16-bits	Base + 0x0164	
Pad configuration register 147 (ETPUB0)	SIU_PCR147	16-bits	Base + 0x0166	
Pad configuration register 148 (ETPUB1)	SIU_PCR148	16-bits	Base + 0x0168	
Pad configuration register 149 (ETPUB2)	SIU_PCR149	16-bits	Base + 0x016A	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
Pad configuration register 150 (ETPUB3)	SIU_PCR150	16-bits	Base + 0x016C	
Pad configuration register 151 (ETPUB4)	SIU_PCR151	16-bits	Base + 0x016E	
Pad configuration register 152 (ETPUB5)	SIU_PCR152	16-bits	Base + 0x0170	
Pad configuration register 153 (ETPUB6)	SIU_PCR153	16-bits	Base + 0x0172	
Pad configuration register 154 (ETPUB7)	SIU_PCR154	16-bits	Base + 0x0174	
Pad configuration register 155 (ETPUB8)	SIU_PCR155	16-bits	Base + 0x0176	
Pad configuration register 156 (ETPUB9)	SIU_PCR156	16-bits	Base + 0x0178	
Pad configuration register 157 (ETPUB10)	SIU_PCR157	16-bits	Base + 0x017A	
Pad configuration register 158 (ETPUB11)	SIU_PCR158	16-bits	Base + 0x017C	
Pad configuration register 159 (ETPUB12)	SIU_PCR159	16-bits	Base + 0x017E	
Pad configuration register 160 (ETPUB13)	SIU_PCR160	16-bits	Base + 0x0180	
Pad configuration register 161 (ETPUB14)	SIU_PCR161	16-bits	Base + 0x0182	
Pad configuration register 162 (ETPUB15)	SIU_PCR162	16-bits	Base + 0x0184	
Pad configuration register 163 (ETPUB16)	SIU_PCR163	16-bits	Base + 0x0186	
Pad configuration register 164 (ETPUB17)	SIU_PCR164	16-bits	Base + 0x0188	
Pad configuration register 165 (ETPUB18)	SIU_PCR165	16-bits	Base + 0x018A	
Pad configuration register 166 (ETPUB19)	SIU_PCR166	16-bits	Base + 0x018C	
Pad configuration register 167 (ETPUB20)	SIU_PCR167	16-bits	Base + 0x018E	
Pad configuration register 168 (ETPUB21)	SIU_PCR168	16-bits	Base + 0x0190	
Pad configuration register 169 (ETPUB22)	SIU_PCR169	16-bits	Base + 0x0192	
Pad configuration register 170 (ETPUB23)	SIU_PCR170	16-bits	Base + 0x0194	
Pad configuration register 171 (ETPUB24)	SIU_PCR171	16-bits	Base + 0x0196	
Pad configuration register 172 (ETPUB25)	SIU_PCR172	16-bits	Base + 0x0198	
Pad configuration register 173 (ETPUB26)	SIU_PCR173	16-bits	Base + 0x019A	
Pad configuration register 174 (ETPUB27)	SIU_PCR174	16-bits	Base + 0x019C	
Pad configuration register 175 (ETPUB28)	SIU_PCR175	16-bits	Base + 0x019E	
Pad configuration register 176 (ETPUB29)	SIU_PCR176	16-bits	Base + 0x01A0	
Pad configuration register 177 (ETPUB30)	SIU_PCR177	16-bits	Base + 0x01A2	
Pad configuration register 178 (ETPUB31)	SIU_PCR178	16-bits	Base + 0x01A4	
Pad configuration register 179 (EMIOS0)	SIU_PCR179	16-bits	Base + 0x01A6	
Pad configuration register 180 (EMIOS1)	SIU_PCR180	16-bits	Base + 0x01A8	
Pad configuration register 181 (EMIOS2)	SIU_PCR181	16-bits	Base + 0x01AA	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
Pad configuration register 182 (EMIOS3)	SIU_PCR182	16-bits	Base + 0x01AC	
Pad configuration register 183 (EMIOS4)	SIU_PCR183	16-bits	Base + 0x01AE	
Pad configuration register 184 (EMIOS5)	SIU_PCR184	16-bits	Base + 0x01B0	
Pad configuration register 185 (EMIOS6)	SIU_PCR185	16-bits	Base + 0x01B2	
Pad configuration register 186 (EMIOS7)	SIU_PCR186	16-bits	Base + 0x01B4	
Pad configuration register 187 (EMIOS8)	SIU_PCR187	16-bits	Base + 0x01B6	
Pad configuration register 188 (EMIOS9)	SIU_PCR188	16-bits	Base + 0x01B8	
Pad configuration register 189 (EMIOS10)	SIU_PCR189	16-bits	Base + 0x01BA	
Pad configuration register 190 (EMIOS11)	SIU_PCR190	16-bits	Base + 0x01BC	
Pad configuration register 191 (EMIOS12)	SIU_PCR191	16-bits	Base + 0x01BE	
Pad configuration register 192 (EMIOS13)	SIU_PCR192	16-bits	Base + 0x01C0	
Pad configuration register 193 (EMIOS14)	SIU_PCR193	16-bits	Base + 0x01C2	
Pad configuration register 194 (EMIOS15)	SIU_PCR194	16-bits	Base + 0x01C4	
Pad configuration register 195 (EMIOS16)	SIU_PCR195	16-bits	Base + 0x01C6	
Pad configuration register 196 (EMIOS17)	SIU_PCR196	16-bits	Base + 0x01C8	
Pad configuration register 197 (EMIOS18)	SIU_PCR197	16-bits	Base + 0x01CA	
Pad configuration register 198 (EMIOS19)	SIU_PCR198	16-bits	Base + 0x01CC	
Pad configuration register 199 (EMIOS20)	SIU_PCR199	16-bits	Base + 0x01CE	
Pad configuration register 200 (EMIOS21)	SIU_PCR200	16-bits	Base + 0x01D0	
Pad configuration register 201 (EMIOS22)	SIU_PCR201	16-bits	Base + 0x01D2	
Pad configuration register 202 (EMIOS23)	SIU_PCR202	16-bits	Base + 0x01D4	
Pad configuration register 203 (GPIO203)	SIU_PCR203	16-bits	Base + 0x01D6	
Pad configuration register 204 (GPIO204)	SIU_PCR204	16-bits	Base + 0x01D8	
Pad configuration register 205 (GPIO205)	SIU_PCR205	16-bits	Base + 0x01DA	
Pad configuration register 206 (GPIO206)	SIU_PCR206	16-bits	Base + 0x01DC	
Pad configuration register 207 (GPIO207)	SIU_PCR207	16-bits	Base + 0x01DE	
Pad configuration register 208 (PLLCFG0)	SIU_PCR208	16-bits	Base + 0x01E0	
Pad configuration register 209 (PLLCFG1)	SIU_PCR209	16-bits	Base + 0x01E2	
Pad configuration register 210 (RSTCFG)	SIU_PCR210	16-bits	Base + 0x01E4	
Pad configuration register 211 (BOOTCFG0)	SIU_PCR211	16-bits	Base + 0x01E6	
Pad configuration register 212 (BOOTCFG1)	SIU_PCR212	16-bits	Base + 0x01E8	
Pad configuration register 213 (WKPCFG)	SIU_PCR213	16-bits	Base + 0x01EA	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
Pad configuration register 214 (ENGCLK)	SIU_PCR214	16-bits	Base + 0x01EC	
Pad configuration register 215 (AN12)	SIU_PCR215	16-bits	Base + 0x01EE	
Pad configuration register 216 (AN13)	SIU_PCR216	16-bits	Base + 0x01F0	
Pad configuration register 217 (AN14)	SIU_PCR217	16-bits	Base + 0x01F2	
Pad configuration register 218 (AN15)	SIU_PCR218	16-bits	Base + 0x01F4	
Pad configuration register 219 (MCKO)	SIU_PCR219	16-bits	Base + 0x01F6	
Pad configuration register 220 (MDO0)	SIU_PCR220	16-bits	Base + 0x01F8	
Pad configuration register 221 (MDO1)	SIU_PCR221	16-bits	Base + 0x01FA	
Pad configuration register 222 (MDO2)	SIU_PCR222	16-bits	Base + 0x01FC	
Pad configuration register 223 (MDO3)	SIU_PCR223	16-bits	Base + 0x01FE	
Pad configuration register 224 ($\overline{\text{MSEO0}}$)	SIU_PCR224	16-bits	Base + 0x0200	
Pad configuration register 225 ($\overline{\text{MSEO1}}$)	SIU_PCR225	16-bits	Base + 0x0202	
Pad configuration register 226 ($\overline{\text{RDY}}$)	SIU_PCR226	16-bits	Base + 0x0204	
Pad configuration register 227 ($\overline{\text{EVTO}}$)	SIU_PCR227	16-bits	Base + 0x0206	
Pad configuration register 228 (TDO)	SIU_PCR228	16-bits	Base + 0x0208	
Pad configuration register 229 (CLKOUT)	SIU_PCR229	16-bits	Base + 0x020A	
Pad configuration register 230 ($\overline{\text{RSTOUT}}$)	SIU_PCR230	16-bits	Base + 0x020C	
Reserved	—	—	Base + (0x020E-0x05FF)	
GPIO pin data output register 0	SIU_GPDO0	8-bits	Base + 0x0600	
GPIO pin data output register 1	SIU_GPDO1	8-bits	Base + 0x0601	
GPIO pin data output register 2	SIU_GPDO2	8-bits	Base + 0x0602	
GPIO pin data output register 3	SIU_GPDO3	8-bits	Base + 0x0603	
GPIO pin data output register 4	SIU_GPDO4	8-bits	Base + 0x0604	
GPIO pin data output register 5	SIU_GPDO5	8-bits	Base + 0x0605	
GPIO pin data output register 6	SIU_GPDO6	8-bits	Base + 0x0606	
GPIO pin data output register 7	SIU_GPDO7	8-bits	Base + 0x0607	
GPIO pin data output register 8	SIU_GPDO8	8-bits	Base + 0x0608	
GPIO pin data output register 9	SIU_GPDO9	8-bits	Base + 0x0609	
GPIO pin data output register 10	SIU_GPDO10	8-bits	Base + 0x060A	
GPIO pin data output register 11	SIU_GPDO11	8-bits	Base + 0x060B	
GPIO pin data output register 12	SIU_GPDO12	8-bits	Base + 0x060C	
GPIO pin data output register 13	SIU_GPDO13	8-bits	Base + 0x060D	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
GPIO pin data output register 14	SIU_GPDO14	8-bits	Base + 0x060E	
GPIO pin data output register 15	SIU_GPDO15	8-bits	Base + 0x060F	
GPIO pin data output register 16	SIU_GPDO16	8-bits	Base + 0x0610	
GPIO pin data output register 17	SIU_GPDO17	8-bits	Base + 0x0611	
GPIO pin data output register 18	SIU_GPDO18	8-bits	Base + 0x0612	
GPIO pin data output register 19	SIU_GPDO19	8-bits	Base + 0x0613	
GPIO pin data output register 20	SIU_GPDO20	8-bits	Base + 0x0614	
GPIO pin data output register 21	SIU_GPDO21	8-bits	Base + 0x0615	
GPIO pin data output register 22	SIU_GPDO22	8-bits	Base + 0x0616	
GPIO pin data output register 23	SIU_GPDO23	8-bits	Base + 0x0617	
GPIO pin data output register 24	SIU_GPDO24	8-bits	Base + 0x0618	
GPIO pin data output register 25	SIU_GPDO25	8-bits	Base + 0x0619	
GPIO pin data output register 26	SIU_GPDO26	8-bits	Base + 0x061A	
GPIO pin data output register 27	SIU_GPDO27	8-bits	Base + 0x061B	
GPIO pin data output register 28	SIU_GPDO28	8-bits	Base + 0x061C	
GPIO pin data output register 29	SIU_GPDO29	8-bits	Base + 0x061D	
GPIO pin data output register 30	SIU_GPDO30	8-bits	Base + 0x061E	
GPIO pin data output register 31	SIU_GPDO31	8-bits	Base + 0x061F	
GPIO pin data output register 32	SIU_GPDO32	8-bits	Base + 0x0620	
GPIO pin data output register 33	SIU_GPDO33	8-bits	Base + 0x0621	
GPIO pin data output register 34	SIU_GPDO34	8-bits	Base + 0x0622	
GPIO pin data output register 35	SIU_GPDO35	8-bits	Base + 0x0623	
GPIO pin data output register 36	SIU_GPDO36	8-bits	Base + 0x0624	
GPIO pin data output register 37	SIU_GPDO37	8-bits	Base + 0x0625	
GPIO pin data output register 38	SIU_GPDO38	8-bits	Base + 0x0626	
GPIO pin data output register 39	SIU_GPDO39	8-bits	Base + 0x0627	
GPIO pin data output register 40	SIU_GPDO40	8-bits	Base + 0x0628	
GPIO pin data output register 41	SIU_GPDO41	8-bits	Base + 0x0629	
GPIO pin data output register 42	SIU_GPDO42	8-bits	Base + 0x062A	
GPIO pin data output register 43	SIU_GPDO43	8-bits	Base + 0x062B	
GPIO pin data output register 44	SIU_GPDO44	8-bits	Base + 0x062C	
GPIO pin data output register 45	SIU_GPDO45	8-bits	Base + 0x062D	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
GPIO pin data output register 46	SIU_GPDO46	8-bits	Base + 0x062E	
GPIO pin data output register 47	SIU_GPDO47	8-bits	Base + 0x062F	
GPIO pin data output register 48	SIU_GPDO48	8-bits	Base + 0x0630	
GPIO pin data output register 49	SIU_GPDO49	8-bits	Base + 0x0631	
GPIO pin data output register 50	SIU_GPDO50	8-bits	Base + 0x0632	
GPIO pin data output register 51	SIU_GPDO51	8-bits	Base + 0x0633	
GPIO pin data output register 52	SIU_GPDO52	8-bits	Base + 0x0634	
GPIO pin data output register 53	SIU_GPDO53	8-bits	Base + 0x0635	
GPIO pin data output register 54	SIU_GPDO54	8-bits	Base + 0x0636	
GPIO pin data output register 55	SIU_GPDO55	8-bits	Base + 0x0637	
GPIO pin data output register 56	SIU_GPDO56	8-bits	Base + 0x0638	
GPIO pin data output register 57	SIU_GPDO57	8-bits	Base + 0x0639	
GPIO pin data output register 58	SIU_GPDO58	8-bits	Base + 0x063A	
GPIO pin data output register 59	SIU_GPDO59	8-bits	Base + 0x063B	
GPIO pin data output register 60	SIU_GPDO60	8-bits	Base + 0x063C	
GPIO pin data output register 61	SIU_GPDO61	8-bits	Base + 0x063D	
GPIO pin data output register 62	SIU_GPDO62	8-bits	Base + 0x063E	
GPIO pin data output register 63	SIU_GPDO63	8-bits	Base + 0x063F	
GPIO pin data output register 64	SIU_GPDO64	8-bits	Base + 0x0640	
GPIO pin data output register 65	SIU_GPDO65	8-bits	Base + 0x0641	
GPIO pin data output register 66	SIU_GPDO66	8-bits	Base + 0x0642	
GPIO pin data output register 67	SIU_GPDO67	8-bits	Base + 0x0643	
GPIO pin data output register 68	SIU_GPDO68	8-bits	Base + 0x0644	
GPIO pin data output register 69	SIU_GPDO69	8-bits	Base + 0x0645	
GPIO pin data output register 70	SIU_GPDO70	8-bits	Base + 0x0646	
GPIO pin data output register 71	SIU_GPDO71	8-bits	Base + 0x0647	
GPIO pin data output register 72	SIU_GPDO72	8-bits	Base + 0x0648	
GPIO pin data output register 73	SIU_GPDO73	8-bits	Base + 0x0649	
GPIO pin data output register 74	SIU_GPDO74	8-bits	Base + 0x064A	
GPIO pin data output register 75	SIU_GPDO75	8-bits	Base + 0x064B	
GPIO pin data output register 76	SIU_GPDO76	8-bits	Base + 0x064C	
GPIO pin data output register 77	SIU_GPDO77	8-bits	Base + 0x064D	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
GPIO pin data output register 78	SIU_GPDO78	8-bits	Base + 0x064E	
GPIO pin data output register 79	SIU_GPDO79	8-bits	Base + 0x064F	
GPIO pin data output register 80	SIU_GPDO80	8-bits	Base + 0x0650	
GPIO pin data output register 81	SIU_GPDO81	8-bits	Base + 0x0651	
GPIO pin data output register 82	SIU_GPDO82	8-bits	Base + 0x0652	
GPIO pin data output register 83	SIU_GPDO83	8-bits	Base + 0x0653	
GPIO pin data output register 84	SIU_GPDO84	8-bits	Base + 0x0654	
GPIO pin data output register 85	SIU_GPDO85	8-bits	Base + 0x0655	
GPIO pin data output register 86	SIU_GPDO86	8-bits	Base + 0x0656	
GPIO pin data output register 87	SIU_GPDO87	8-bits	Base + 0x0657	
GPIO pin data output register 88	SIU_GPDO88	8-bits	Base + 0x0658	
GPIO pin data output register 89	SIU_GPDO89	8-bits	Base + 0x0659	
GPIO pin data output register 90	SIU_GPDO90	8-bits	Base + 0x065A	
GPIO pin data output register 91	SIU_GPDO91	8-bits	Base + 0x065B	
GPIO pin data output register 92	SIU_GPDO92	8-bits	Base + 0x065C	
GPIO pin data output register 93	SIU_GPDO93	8-bits	Base + 0x065D	
GPIO pin data output register 94	SIU_GPDO94	8-bits	Base + 0x065E	
GPIO pin data output register 95	SIU_GPDO95	8-bits	Base + 0x065F	
GPIO pin data output register 96	SIU_GPDO96	8-bits	Base + 0x0660	
GPIO pin data output register 97	SIU_GPDO97	8-bits	Base + 0x0661	
GPIO pin data output register 98	SIU_GPDO98	8-bits	Base + 0x0662	
GPIO pin data output register 99	SIU_GPDO99	8-bits	Base + 0x0663	
GPIO pin data output register 100	SIU_GPDO100	8-bits	Base + 0x0664	
GPIO pin data output register 101	SIU_GPDO101	8-bits	Base + 0x0665	
GPIO pin data output register 102	SIU_GPDO102	8-bits	Base + 0x0666	
GPIO pin data output register 103	SIU_GPDO103	8-bits	Base + 0x0667	
GPIO pin data output register 104	SIU_GPDO104	8-bits	Base + 0x0668	
GPIO pin data output register 105	SIU_GPDO105	8-bits	Base + 0x0669	
GPIO pin data output register 106	SIU_GPDO106	8-bits	Base + 0x066A	
GPIO pin data output register 107	SIU_GPDO107	8-bits	Base + 0x066B	
GPIO pin data output register 108	SIU_GPDO108	8-bits	Base + 0x066C	
GPIO pin data output register 109	SIU_GPDO109	8-bits	Base + 0x066D	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
GPIO pin data output register 110	SIU_GPDO110	8-bits	Base + 0x066E	
GPIO pin data output register 111	SIU_GPDO111	8-bits	Base + 0x066F	
GPIO pin data output register 112	SIU_GPDO112	8-bits	Base + 0x0670	
GPIO pin data output register 113	SIU_GPDO113	8-bits	Base + 0x0671	
GPIO pin data output register 114	SIU_GPDO114	8-bits	Base + 0x0672	
GPIO pin data output register 115	SIU_GPDO115	8-bits	Base + 0x0673	
GPIO pin data output register 116	SIU_GPDO116	8-bits	Base + 0x0674	
GPIO pin data output register 117	SIU_GPDO117	8-bits	Base + 0x0675	
GPIO pin data output register 118	SIU_GPDO118	8-bits	Base + 0x0676	
GPIO pin data output register 119	SIU_GPDO119	8-bits	Base + 0x0677	
GPIO pin data output register 120	SIU_GPDO120	8-bits	Base + 0x0678	
GPIO pin data output register 121	SIU_GPDO121	8-bits	Base + 0x0679	
GPIO pin data output register 122	SIU_GPDO122	8-bits	Base + 0x067A	
GPIO pin data output register 123	SIU_GPDO123	8-bits	Base + 0x067B	
GPIO pin data output register 124	SIU_GPDO124	8-bits	Base + 0x067C	
GPIO pin data output register 125	SIU_GPDO125	8-bits	Base + 0x067D	
GPIO pin data output register 126	SIU_GPDO126	8-bits	Base + 0x067E	
GPIO pin data output register 127	SIU_GPDO127	8-bits	Base + 0x067F	
GPIO pin data output register 128	SIU_GPDO128	8-bits	Base + 0x0680	
GPIO pin data output register 129	SIU_GPDO129	8-bits	Base + 0x0681	
GPIO pin data output register 130	SIU_GPDO130	8-bits	Base + 0x0682	
GPIO pin data output register 131	SIU_GPDO131	8-bits	Base + 0x0683	
GPIO pin data output register 132	SIU_GPDO132	8-bits	Base + 0x0684	
GPIO pin data output register 133	SIU_GPDO133	8-bits	Base + 0x0685	
GPIO pin data output register 134	SIU_GPDO134	8-bits	Base + 0x0686	
GPIO pin data output register 135	SIU_GPDO135	8-bits	Base + 0x0687	
GPIO pin data output register 136	SIU_GPDO136	8-bits	Base + 0x0688	
GPIO pin data output register 137	SIU_GPDO137	8-bits	Base + 0x0689	
GPIO pin data output register 138	SIU_GPDO138	8-bits	Base + 0x068A	
GPIO pin data output register 139	SIU_GPDO139	8-bits	Base + 0x068B	
GPIO pin data output register 140	SIU_GPDO140	8-bits	Base + 0x068C	
GPIO pin data output register 141	SIU_GPDO141	8-bits	Base + 0x068D	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
GPIO pin data output register 142	SIU_GPDO142	8-bits	Base + 0x068E	
GPIO pin data output register 143	SIU_GPDO143	8-bits	Base + 0x068F	
GPIO pin data output register 144	SIU_GPDO144	8-bits	Base + 0x0690	
GPIO pin data output register 145	SIU_GPDO145	8-bits	Base + 0x0691	
GPIO pin data output register 146	SIU_GPDO146	8-bits	Base + 0x0692	
GPIO pin data output register 147	SIU_GPDO147	8-bits	Base + 0x0693	
GPIO pin data output register 148	SIU_GPDO148	8-bits	Base + 0x0694	
GPIO pin data output register 149	SIU_GPDO149	8-bits	Base + 0x0695	
GPIO pin data output register 150	SIU_GPDO150	8-bits	Base + 0x0696	
GPIO pin data output register 151	SIU_GPDO151	8-bits	Base + 0x0697	
GPIO pin data output register 152	SIU_GPDO152	8-bits	Base + 0x0698	
GPIO pin data output register 153	SIU_GPDO153	8-bits	Base + 0x0699	
GPIO pin data output register 154	SIU_GPDO154	8-bits	Base + 0x069A	
GPIO pin data output register 155	SIU_GPDO155	8-bits	Base + 0x069B	
GPIO pin data output register 156	SIU_GPDO156	8-bits	Base + 0x069C	
GPIO pin data output register 157	SIU_GPDO157	8-bits	Base + 0x069D	
GPIO pin data output register 158	SIU_GPDO158	8-bits	Base + 0x069E	
GPIO pin data output register 159	SIU_GPDO159	8-bits	Base + 0x069F	
GPIO pin data output register 160	SIU_GPDO160	8-bits	Base + 0x06A0	
GPIO pin data output register 161	SIU_GPDO161	8-bits	Base + 0x06A1	
GPIO pin data output register 162	SIU_GPDO162	8-bits	Base + 0x06A2	
GPIO pin data output register 163	SIU_GPDO163	8-bits	Base + 0x06A3	
GPIO pin data output register 164	SIU_GPDO164	8-bits	Base + 0x06A4	
GPIO pin data output register 165	SIU_GPDO165	8-bits	Base + 0x06A5	
GPIO pin data output register 166	SIU_GPDO166	8-bits	Base + 0x06A6	
GPIO pin data output register 167	SIU_GPDO167	8-bits	Base + 0x06A7	
GPIO pin data output register 168	SIU_GPDO168	8-bits	Base + 0x06A8	
GPIO pin data output register 169	SIU_GPDO169	8-bits	Base + 0x06A9	
GPIO pin data output register 170	SIU_GPDO170	8-bits	Base + 0x06AA	
GPIO pin data output register 171	SIU_GPDO171	8-bits	Base + 0x06AB	
GPIO pin data output register 172	SIU_GPDO172	8-bits	Base + 0x06AC	
GPIO pin data output register 173	SIU_GPDO173	8-bits	Base + 0x06AD	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
GPIO pin data output register 174	SIU_GPDO174	8-bits	Base + 0x06AE	
GPIO pin data output register 175	SIU_GPDO175	8-bits	Base + 0x06AF	
GPIO pin data output register 176	SIU_GPDO176	8-bits	Base + 0x06B0	
GPIO pin data output register 177	SIU_GPDO177	8-bits	Base + 0x06B1	
GPIO pin data output register 178	SIU_GPDO178	8-bits	Base + 0x06B2	
GPIO pin data output register 179	SIU_GPDO179	8-bits	Base + 0x06B3	
GPIO pin data output register 180	SIU_GPDO180	8-bits	Base + 0x06B4	
GPIO pin data output register 181	SIU_GPDO181	8-bits	Base + 0x06B5	
GPIO pin data output register 182	SIU_GPDO182	8-bits	Base + 0x06B6	
GPIO pin data output register 183	SIU_GPDO183	8-bits	Base + 0x06B7	
GPIO pin data output register 184	SIU_GPDO184	8-bits	Base + 0x06B8	
GPIO pin data output register 185	SIU_GPDO185	8-bits	Base + 0x06B9	
GPIO pin data output register 186	SIU_GPDO186	8-bits	Base + 0x06BA	
GPIO pin data output register 187	SIU_GPDO187	8-bits	Base + 0x06BB	
GPIO pin data output register 188	SIU_GPDO188	8-bits	Base + 0x06BC	
GPIO pin data output register 189	SIU_GPDO189	8-bits	Base + 0x06BD	
GPIO pin data output register 190	SIU_GPDO190	8-bits	Base + 0x06BE	
GPIO pin data output register 191	SIU_GPDO191	8-bits	Base + 0x06BF	
GPIO pin data output register 192	SIU_GPDO192	8-bits	Base + 0x06C0	
GPIO pin data output register 193	SIU_GPDO193	8-bits	Base + 0x06C1	
GPIO pin data output register 194	SIU_GPDO194	8-bits	Base + 0x06C2	
GPIO pin data output register 195	SIU_GPDO195	8-bits	Base + 0x06C3	
GPIO pin data output register 196	SIU_GPDO196	8-bits	Base + 0x06C4	
GPIO pin data output register 197	SIU_GPDO197	8-bits	Base + 0x06C5	
GPIO pin data output register 198	SIU_GPDO198	8-bits	Base + 0x06C6	
GPIO pin data output register 199	SIU_GPDO199	8-bits	Base + 0x06C7	
GPIO pin data output register 200	SIU_GPDO200	8-bits	Base + 0x06C8	
GPIO pin data output register 201	SIU_GPDO201	8-bits	Base + 0x06C9	
GPIO pin data output register 202	SIU_GPDO202	8-bits	Base + 0x06CA	
GPIO pin data output register 203	SIU_GPDO203	8-bits	Base + 0x06CB	
GPIO pin data output register 204	SIU_GPDO204	8-bits	Base + 0x06CC	
GPIO pin data output register 205	SIU_GPDO205	8-bits	Base + 0x06CD	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
GPIO pin data output register 206	SIU_GPDO206	8-bits	Base + 0x06CE	
GPIO pin data output register 207	SIU_GPDO207	8-bits	Base + 0x06CF	
GPIO pin data output register 208	SIU_GPDO208	8-bits	Base + 0x06D0	
GPIO pin data output register 209	SIU_GPDO209	8-bits	Base + 0x06D1	
GPIO pin data output register 210	SIU_GPDO210	8-bits	Base + 0x06D2	
GPIO pin data output register 211	SIU_GPDO211	8-bits	Base + 0x06D3	
GPIO pin data output register 212	SIU_GPDO212	8-bits	Base + 0x06D4	
GPIO pin data output register 213	SIU_GPDO213	8-bits	Base + 0x06D5	
Reserved	—	—	Base + (0x06D6-0x07FF)	
GPIO pin data input register 0	SIU_GPDI0	8-bits	Base + 0x0800	
GPIO pin data input register 1	SIU_GPDI1	8-bits	Base + 0x0801	
GPIO pin data input register 2	SIU_GPDI2	8-bits	Base + 0x0802	
GPIO pin data input register 3	SIU_GPDI3	8-bits	Base + 0x0803	
GPIO pin data input register 4	SIU_GPDI4	8-bits	Base + 0x0804	
GPIO pin data input register 5	SIU_GPDI5	8-bits	Base + 0x0805	
GPIO pin data input register 6	SIU_GPDI6	8-bits	Base + 0x0806	
GPIO pin data input register 7	SIU_GPDI7	8-bits	Base + 0x0807	
GPIO pin data input register 8	SIU_GPDI8	8-bits	Base + 0x0808	
GPIO pin data input register 9	SIU_GPDI9	8-bits	Base + 0x0809	
GPIO pin data input register 10	SIU_GPDI10	8-bits	Base + 0x080A	
GPIO pin data input register 11	SIU_GPDI11	8-bits	Base + 0x080B	
GPIO pin data input register 12	SIU_GPDI12	8-bits	Base + 0x080C	
GPIO pin data input register 13	SIU_GPDI13	8-bits	Base + 0x080D	
GPIO pin data input register 14	SIU_GPDI14	8-bits	Base + 0x080E	
GPIO pin data input register 15	SIU_GPDI15	8-bits	Base + 0x080F	
GPIO pin data input register 16	SIU_GPDI16	8-bits	Base + 0x0810	
GPIO pin data input register 17	SIU_GPDI17	8-bits	Base + 0x0811	
GPIO pin data input register 18	SIU_GPDI18	8-bits	Base + 0x0812	
GPIO pin data input register 19	SIU_GPDI19	8-bits	Base + 0x0813	
GPIO pin data input register 20	SIU_GPDI20	8-bits	Base + 0x0814	
GPIO pin data input register 21	SIU_GPDI21	8-bits	Base + 0x0815	
GPIO pin data input register 22	SIU_GPDI22	8-bits	Base + 0x0816	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
GPIO pin data input register 23	SIU_GPDI23	8-bits	Base + 0x0817	
GPIO pin data input register 24	SIU_GPDI24	8-bits	Base + 0x0818	
GPIO pin data input register 25	SIU_GPDI25	8-bits	Base + 0x0819	
GPIO pin data input register 26	SIU_GPDI26	8-bits	Base + 0x081A	
GPIO pin data input register 27	SIU_GPDI27	8-bits	Base + 0x081B	
GPIO pin data input register 28	SIU_GPDI28	8-bits	Base + 0x081C	
GPIO pin data input register 29	SIU_GPDI29	8-bits	Base + 0x081D	
GPIO pin data input register 30	SIU_GPDI30	8-bits	Base + 0x081E	
GPIO pin data input register 31	SIU_GPDI31	8-bits	Base + 0x081F	
GPIO pin data input register 32	SIU_GPDI32	8-bits	Base + 0x0820	
GPIO pin data input register 33	SIU_GPDI33	8-bits	Base + 0x0821	
GPIO pin data input register 34	SIU_GPDI34	8-bits	Base + 0x0822	
GPIO pin data input register 35	SIU_GPDI35	8-bits	Base + 0x0823	
GPIO pin data input register 36	SIU_GPDI36	8-bits	Base + 0x0824	
GPIO pin data input register 37	SIU_GPDI37	8-bits	Base + 0x0825	
GPIO pin data input register 38	SIU_GPDI38	8-bits	Base + 0x0826	
GPIO pin data input register 39	SIU_GPDI39	8-bits	Base + 0x0827	
GPIO pin data input register 40	SIU_GPDI40	8-bits	Base + 0x0828	
GPIO pin data input register 41	SIU_GPDI41	8-bits	Base + 0x0829	
GPIO pin data input register 42	SIU_GPDI42	8-bits	Base + 0x082A	
GPIO pin data input register 43	SIU_GPDI43	8-bits	Base + 0x082B	
GPIO pin data input register 44	SIU_GPDI44	8-bits	Base + 0x082C	
GPIO pin data input register 45	SIU_GPDI45	8-bits	Base + 0x082D	
GPIO pin data input register 46	SIU_GPDI46	8-bits	Base + 0x082E	
GPIO pin data input register 47	SIU_GPDI47	8-bits	Base + 0x082F	
GPIO pin data input register 48	SIU_GPDI48	8-bits	Base + 0x0830	
GPIO pin data input register 49	SIU_GPDI49	8-bits	Base + 0x0831	
GPIO pin data input register 50	SIU_GPDI50	8-bits	Base + 0x0832	
GPIO pin data input register 51	SIU_GPDI51	8-bits	Base + 0x0833	
GPIO pin data input register 52	SIU_GPDI52	8-bits	Base + 0x0834	
GPIO pin data input register 53	SIU_GPDI53	8-bits	Base + 0x0835	
GPIO pin data input register 54	SIU_GPDI54	8-bits	Base + 0x0836	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
GPIO pin data input register 55	SIU_GPDI55	8-bits	Base + 0x0837	
GPIO pin data input register 56	SIU_GPDI56	8-bits	Base + 0x0838	
GPIO pin data input register 57	SIU_GPDI57	8-bits	Base + 0x0839	
GPIO pin data input register 58	SIU_GPDI58	8-bits	Base + 0x083A	
GPIO pin data input register 59	SIU_GPDI59	8-bits	Base + 0x083B	
GPIO pin data input register 60	SIU_GPDI60	8-bits	Base + 0x083C	
GPIO pin data input register 61	SIU_GPDI61	8-bits	Base + 0x083D	
GPIO pin data input register 62	SIU_GPDI62	8-bits	Base + 0x083E	
GPIO pin data input register 63	SIU_GPDI63	8-bits	Base + 0x083F	
GPIO pin data input register 64	SIU_GPDI64	8-bits	Base + 0x0840	
GPIO pin data input register 65	SIU_GPDI65	8-bits	Base + 0x0841	
GPIO pin data input register 66	SIU_GPDI66	8-bits	Base + 0x0842	
GPIO pin data input register 67	SIU_GPDI67	8-bits	Base + 0x0843	
GPIO pin data input register 68	SIU_GPDI68	8-bits	Base + 0x0844	
GPIO pin data input register 69	SIU_GPDI69	8-bits	Base + 0x0845	
GPIO pin data input register 70	SIU_GPDI70	8-bits	Base + 0x0846	
GPIO pin data input register 71	SIU_GPDI71	8-bits	Base + 0x0847	
GPIO pin data input register 72	SIU_GPDI72	8-bits	Base + 0x0848	
GPIO pin data input register 73	SIU_GPDI73	8-bits	Base + 0x0849	
GPIO pin data input register 74	SIU_GPDI74	8-bits	Base + 0x084A	
GPIO pin data input register 75	SIU_GPDI75	8-bits	Base + 0x084B	
GPIO pin data input register 76	SIU_GPDI76	8-bits	Base + 0x084C	
GPIO pin data input register 77	SIU_GPDI77	8-bits	Base + 0x084D	
GPIO pin data input register 78	SIU_GPDI78	8-bits	Base + 0x084E	
GPIO pin data input register 79	SIU_GPDI79	8-bits	Base + 0x084F	
GPIO pin data input register 80	SIU_GPDI80	8-bits	Base + 0x0850	
GPIO pin data input register 81	SIU_GPDI81	8-bits	Base + 0x0851	
GPIO pin data input register 82	SIU_GPDI82	8-bits	Base + 0x0852	
GPIO pin data input register 83	SIU_GPDI83	8-bits	Base + 0x0853	
GPIO pin data input register 84	SIU_GPDI84	8-bits	Base + 0x0854	
GPIO pin data input register 85	SIU_GPDI85	8-bits	Base + 0x0855	
GPIO pin data input register 86	SIU_GPDI86	8-bits	Base + 0x0856	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
GPIO pin data input register 87	SIU_GPDI87	8-bits	Base + 0x0857	
GPIO pin data input register 88	SIU_GPDI88	8-bits	Base + 0x0858	
GPIO pin data input register 89	SIU_GPDI89	8-bits	Base + 0x0859	
GPIO pin data input register 90	SIU_GPDI90	8-bits	Base + 0x085A	
GPIO pin data input register 91	SIU_GPDI91	8-bits	Base + 0x085B	
GPIO pin data input register 92	SIU_GPDI92	8-bits	Base + 0x085C	
GPIO pin data input register 93	SIU_GPDI93	8-bits	Base + 0x085D	
GPIO pin data input register 94	SIU_GPDI94	8-bits	Base + 0x085E	
GPIO pin data input register 95	SIU_GPDI95	8-bits	Base + 0x085F	
GPIO pin data input register 96	SIU_GPDI96	8-bits	Base + 0x0860	
GPIO pin data input register 97	SIU_GPDI97	8-bits	Base + 0x0861	
GPIO pin data input register 98	SIU_GPDI98	8-bits	Base + 0x0862	
GPIO pin data input register 99	SIU_GPDI99	8-bits	Base + 0x0863	
GPIO pin data input register 100	SIU_GPDI100	8-bits	Base + 0x0864	
GPIO pin data input register 101	SIU_GPDI101	8-bits	Base + 0x0865	
GPIO pin data input register 102	SIU_GPDI102	8-bits	Base + 0x0866	
GPIO pin data input register 103	SIU_GPDI103	8-bits	Base + 0x0867	
GPIO pin data input register 104	SIU_GPDI104	8-bits	Base + 0x0868	
GPIO pin data input register 105	SIU_GPDI105	8-bits	Base + 0x0869	
GPIO pin data input register 106	SIU_GPDI106	8-bits	Base + 0x086A	
GPIO pin data input register 107	SIU_GPDI107	8-bits	Base + 0x086B	
GPIO pin data input register 108	SIU_GPDI108	8-bits	Base + 0x086C	
GPIO pin data input register 109	SIU_GPDI109	8-bits	Base + 0x086D	
GPIO pin data input register 110	SIU_GPDI110	8-bits	Base + 0x086E	
GPIO pin data input register 111	SIU_GPDI111	8-bits	Base + 0x086F	
GPIO pin data input register 112	SIU_GPDI112	8-bits	Base + 0x0870	
GPIO pin data input register 113	SIU_GPDI113	8-bits	Base + 0x0871	
GPIO pin data input register 114	SIU_GPDI114	8-bits	Base + 0x0872	
GPIO pin data input register 115	SIU_GPDI115	8-bits	Base + 0x0873	
GPIO pin data input register 116	SIU_GPDI116	8-bits	Base + 0x0874	
GPIO pin data input register 117	SIU_GPDI117	8-bits	Base + 0x0875	
GPIO pin data input register 118	SIU_GPDI118	8-bits	Base + 0x0876	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
GPIO pin data input register 119	SIU_GPDI119	8-bits	Base + 0x0877	
GPIO pin data input register 120	SIU_GPDI120	8-bits	Base + 0x0878	
GPIO pin data input register 121	SIU_GPDI121	8-bits	Base + 0x0879	
GPIO pin data input register 122	SIU_GPDI122	8-bits	Base + 0x087A	
GPIO pin data input register 123	SIU_GPDI123	8-bits	Base + 0x087B	
GPIO pin data input register 124	SIU_GPDI124	8-bits	Base + 0x087C	
GPIO pin data input register 125	SIU_GPDI125	8-bits	Base + 0x087D	
GPIO pin data input register 126	SIU_GPDI126	8-bits	Base + 0x087E	
GPIO pin data input register 127	SIU_GPDI127	8-bits	Base + 0x087F	
GPIO pin data input register 128	SIU_GPDI128	8-bits	Base + 0x0880	
GPIO pin data input register 129	SIU_GPDI129	8-bits	Base + 0x0881	
GPIO pin data input register 130	SIU_GPDI130	8-bits	Base + 0x0882	
GPIO pin data input register 131	SIU_GPDI131	8-bits	Base + 0x0883	
GPIO pin data input register 132	SIU_GPDI132	8-bits	Base + 0x0884	
GPIO pin data input register 133	SIU_GPDI133	8-bits	Base + 0x0885	
GPIO pin data input register 134	SIU_GPDI134	8-bits	Base + 0x0886	
GPIO pin data input register 135	SIU_GPDI135	8-bits	Base + 0x0887	
GPIO pin data input register 136	SIU_GPDI136	8-bits	Base + 0x0888	
GPIO pin data input register 137	SIU_GPDI137	8-bits	Base + 0x0889	
GPIO pin data input register 138	SIU_GPDI138	8-bits	Base + 0x088A	
GPIO pin data input register 139	SIU_GPDI139	8-bits	Base + 0x088B	
GPIO pin data input register 140	SIU_GPDI140	8-bits	Base + 0x088C	
GPIO pin data input register 141	SIU_GPDI141	8-bits	Base + 0x088D	
GPIO pin data input register 142	SIU_GPDI142	8-bits	Base + 0x088E	
GPIO pin data input register 143	SIU_GPDI143	8-bits	Base + 0x088F	
GPIO pin data input register 144	SIU_GPDI144	8-bits	Base + 0x0890	
GPIO pin data input register 145	SIU_GPDI145	8-bits	Base + 0x0891	
GPIO pin data input register 146	SIU_GPDI146	8-bits	Base + 0x0892	
GPIO pin data input register 147	SIU_GPDI147	8-bits	Base + 0x0893	
GPIO pin data input register 148	SIU_GPDI148	8-bits	Base + 0x0894	
GPIO pin data input register 149	SIU_GPDI149	8-bits	Base + 0x0895	
GPIO pin data input register 150	SIU_GPDI150	8-bits	Base + 0x0896	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
GPIO pin data input register 151	SIU_GPDI151	8-bits	Base + 0x0897	
GPIO pin data input register 152	SIU_GPDI152	8-bits	Base + 0x0898	
GPIO pin data input register 153	SIU_GPDI153	8-bits	Base + 0x0899	
GPIO pin data input register 154	SIU_GPDI154	8-bits	Base + 0x089A	
GPIO pin data input register 155	SIU_GPDI155	8-bits	Base + 0x089B	
GPIO pin data input register 156	SIU_GPDI156	8-bits	Base + 0x089C	
GPIO pin data input register 157	SIU_GPDI157	8-bits	Base + 0x089D	
GPIO pin data input register 158	SIU_GPDI158	8-bits	Base + 0x089E	
GPIO pin data input register 159	SIU_GPDI159	8-bits	Base + 0x089F	
GPIO pin data input register 160	SIU_GPDI160	8-bits	Base + 0x08A0	
GPIO pin data input register 161	SIU_GPDI161	8-bits	Base + 0x08A1	
GPIO pin data input register 162	SIU_GPDI162	8-bits	Base + 0x08A2	
GPIO pin data input register 163	SIU_GPDI163	8-bits	Base + 0x08A3	
GPIO pin data input register 164	SIU_GPDI164	8-bits	Base + 0x08A4	
GPIO pin data input register 165	SIU_GPDI165	8-bits	Base + 0x08A5	
GPIO pin data input register 166	SIU_GPDI166	8-bits	Base + 0x08A6	
GPIO pin data input register 167	SIU_GPDI167	8-bits	Base + 0x08A7	
GPIO pin data input register 168	SIU_GPDI168	8-bits	Base + 0x08A8	
GPIO pin data input register 169	SIU_GPDI169	8-bits	Base + 0x08A9	
GPIO pin data input register 170	SIU_GPDI170	8-bits	Base + 0x08AA	
GPIO pin data input register 171	SIU_GPDI171	8-bits	Base + 0x08AB	
GPIO pin data input register 172	SIU_GPDI172	8-bits	Base + 0x08AC	
GPIO pin data input register 173	SIU_GPDI173	8-bits	Base + 0x08AD	
GPIO pin data input register 174	SIU_GPDI174	8-bits	Base + 0x08AE	
GPIO pin data input register 175	SIU_GPDI175	8-bits	Base + 0x08AF	
GPIO pin data input register 176	SIU_GPDI176	8-bits	Base + 0x08B0	
GPIO pin data input register 177	SIU_GPDI177	8-bits	Base + 0x08B1	
GPIO pin data input register 178	SIU_GPDI178	8-bits	Base + 0x08B2	
GPIO pin data input register 179	SIU_GPDI179	8-bits	Base + 0x08B3	
GPIO pin data input register 180	SIU_GPDI180	8-bits	Base + 0x08B4	
GPIO pin data input register 181	SIU_GPDI181	8-bits	Base + 0x08B5	
GPIO pin data input register 182	SIU_GPDI182	8-bits	Base + 0x08B6	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
GPIO pin data input register 183	SIU_GPDI183	8-bits	Base + 0x08B7	
GPIO pin data input register 184	SIU_GPDI184	8-bits	Base + 0x08B8	
GPIO pin data input register 185	SIU_GPDI185	8-bits	Base + 0x08B9	
GPIO pin data input register 186	SIU_GPDI186	8-bits	Base + 0x08BA	
GPIO pin data input register 187	SIU_GPDI187	8-bits	Base + 0x08BB	
GPIO pin data input register 188	SIU_GPDI188	8-bits	Base + 0x08BC	
GPIO pin data input register 189	SIU_GPDI189	8-bits	Base + 0x08BD	
GPIO pin data input register 190	SIU_GPDI190	8-bits	Base + 0x08BE	
GPIO pin data input register 191	SIU_GPDI191	8-bits	Base + 0x08BF	
GPIO pin data input register 192	SIU_GPDI192	8-bits	Base + 0x08C0	
GPIO pin data input register 193	SIU_GPDI193	8-bits	Base + 0x08C1	
GPIO pin data input register 194	SIU_GPDI194	8-bits	Base + 0x08C2	
GPIO pin data input register 195	SIU_GPDI195	8-bits	Base + 0x08C3	
GPIO pin data input register 196	SIU_GPDI196	8-bits	Base + 0x08C4	
GPIO pin data input register 197	SIU_GPDI197	8-bits	Base + 0x08C5	
GPIO pin data input register 198	SIU_GPDI198	8-bits	Base + 0x08C6	
GPIO pin data input register 199	SIU_GPDI199	8-bits	Base + 0x08C7	
GPIO pin data input register 200	SIU_GPDI200	8-bits	Base + 0x08C8	
GPIO pin data input register 201	SIU_GPDI201	8-bits	Base + 0x08C9	
GPIO pin data input register 202	SIU_GPDI202	8-bits	Base + 0x08CA	
GPIO pin data input register 203	SIU_GPDI203	8-bits	Base + 0x08CB	
GPIO pin data input register 204	SIU_GPDI204	8-bits	Base + 0x08CC	
GPIO pin data input register 205	SIU_GPDI205	8-bits	Base + 0x08CD	
GPIO pin data input register 206	SIU_GPDI206	8-bits	Base + 0x08CE	
GPIO pin data input register 207	SIU_GPDI207	8-bits	Base + 0x08CF	
GPIO pin data input register 208	SIU_GPDI208	8-bits	Base + 0x08D0	
GPIO pin data input register 209	SIU_GPDI209	8-bits	Base + 0x08D1	
GPIO pin data input register 210	SIU_GPDI210	8-bits	Base + 0x08D2	
GPIO pin data input register 211	SIU_GPDI211	8-bits	Base + 0x08D3	
GPIO pin data input register 212	SIU_GPDI212	8-bits	Base + 0x08D4	
GPIO pin data input register 213	SIU_GPDI213	8-bits	Base + 0x08D5	
Reserved	—	—	Base + (0x08D6-0x08FF)	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
eQADC trigger input select register	SIU_ETISR	32-bits	Base + 0x0900	
External IRQ input select register	SIU_EIISR	32-bits	Base + 0x0904	
DSPI input select register	SIU_DISR	32-bits	Base + 0x0908	
Reserved	—	—	Base + (0x090C-0x097F)	
Chip configuration register	SIU_CCR	32-bits	Base + 0x0980	
External clock control register	SIU_ECCR	32-bits	Base + 0x0984	
Compare A high register	SIU_CARH	32-bits	Base + 0x0988	
Compare A low register	SIU_CARL	32-bits	Base + 0x098C	
Compare B high register	SIU_CBRH	32-bits	Base + 0x0990	
Compare B low register	SIU_CBRL	32-bits	Base + 0x0994	
Reserved	—	—	(Base + 0x0998)-0xC3F9_FFFF)	
Enhanced Modular Input/Output Subsystem (eMIOS)			0xC3FA_0000	Chapter 17, “Enhanced Modular Input/Output Subsystem (eMIOS)”
Module configuration register	EMIOS_MCR	32-bit	Base + 0x0000	
Global flag register	EMIOS_GFR	32-bit	Base+ 0x0004	
Output update disable register	EMIOS_OUDR	32-bit	Base + 0x0008	
Reserved	—	—	Base + (0x000C-0x001F)	
Unified channel n, where n = 0-23	UC base addresses (UCn)		Base + (0x0020 * (n+1))	
Channel A data register n	EMIOS_CADRn	32-bit	UCnBase + 0x00	
Channel B data register n	EMIOS_CBDRn	32-bit	UCnBase + 0x04	
Channel counter register n	EMIOS_CCNTRn	32-bit	UCnBase + 0x08	
Channel control register n	EMIOS_CCRn	32-bit	UCnBase + 0x0C	
Channel status register n	EMIOS_CSRn	32-bit	UCnBase + 0x10	
Reserved	—	—	(UCnBase + 0x14)-0xC3FB_FFFF	
Enhanced Time Processing Unit (eTPU)			0xC3FC_0000	Chapter 18, “Enhanced Time Processing Unit (eTPU)”
eTPU module configuration register	ETPU_MCR	32-bit	Base + 0x0000	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
eTPU coherent dual-parameter controller register	ETPU_CDCR	32-bit	Base + 0x0004	
Reserved	—	—	Base + (0x0008-0x000B)	
eTPU miscellaneous compare register	ETPU_MISCCMPR	32-bit	Base + 0x000C	
eTPU SCM off-range data register	ETPU_SCMOFFDATAR	32-bit	Base + 0x0010	
eTPU A engine configuration register	ETPU_ECR_A	32-bit	Base + 0x0014	
eTPU B engine Configuration register[2]	ETPU_ECR_B[2]	32-bit	Base + 0x0018	
Reserved	—	—	Base + (0x001C-0x001F)	
eTPU A time base configuration register	ETPU_TBCR_A	32-bit	Base + 0x0020	
eTPU A time base 1	ETPU_TB1R_A	32-bit	Base + 0x0024	
eTPU A time base 2	ETPU_TB2R_A	32-bit	Base + 0x0028	
eTPU A STAC bus interface configuration register	ETPU_REDCR_A	32-bit	Base + 0x002C	
Reserved	—	—	Base + (0x0030-0x003F)	
eTPU B time base configuration register[2]	ETPU_TBCR_B[2]	32-bit	Base + 0x0040	
eTPU B time base 1	ETPU_TB1R_B[2]	32-bit	Base + 0x0044	
eTPU B time base 2	ETPU_TB2R_B[2]	32-bit	Base + 0x0048	
eTPU B STAC bus interface configuration register[2]	ETPU_REDCR_B[2]	32-bit	Base + 0x004C	
Reserved	—	—	Base + (0x0050-0x01FF)	
eTPU A channel interrupt status register	ETPU_CISR_A	32-bit	Base + 0x0200	
eTPU B channel interrupt status register [2]	ETPU_CISR_B[2]	32-bit	Base + 0x0204	
Reserved	—	—	Base + (0x0208-0x020F)	
eTPU A channel data transfer request status register	ETPU_CDTRSR_A	32-bit	Base + 0x0210	
eTPU B channel data transfer request status register[2]	ETPU_CDTRSR_B[2]	32-bit	Base + 0x0214	
Reserved	—	—	Base + (0x0218-0x021F)	
eTPU A channel interrupt overflow status register	ETPU_CIOSR_A	32-bit	Base + 0x0220	
eTPU B channel interrupt overflow status register[2]	ETPU_CIOSR_B[2]	32-bit	Base + 0x0224	
Reserved	—	—	Base + (0x0228-0x022F)	
eTPU A channel data transfer request overflow status register	ETPU_CDTROSR_A	32-bit	Base + 0x0230	
eTPU B channel data transfer request overflow status register[2]	ETPU_CDTROSR_B[2]	32-bit	Base + 0x0234	
Reserved	—	—	Base + (0x0238-0x023F)	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
eTPU A channel interrupt enable register	ETPU_CIER_A	32-bit	Base + 0x0240	
eTPU B channel interrupt enable register[2]	ETPU_CIER_B[2]	32-bit	Base + 0x0244	
Reserved	—	—	Base + (0x0248-0x024F)	
eTPU A channel data transfer request enable register	ETPU_CDTRER_A	32-bit	Base + 0x0250	
eTPU B channel data transfer request enable register[2]	ETPU_CDTRER_B[2]	32-bit	Base + 0x0254	
Reserved	—	—	Base + (0x0258-0x027F)	
eTPU A channel pending service status register	ETPU_CPSSR_A	32-bit	Base + 0x0280	
eTPU B channel pending service status register[2]	ETPU_CPSSR_B[2]	32-bit	Base + 0x0284	
Reserved	—	—	Base + (0x0288-0x028F)	
eTPU A channel service status register	ETPU_CSSR_A	32-bit	Base + 0x0290	
eTPU B channel service status register[2]	ETPU_CSSR_B[2]	32-bit	Base + 0x0294	
Reserved	—	—	Base + (0x0298-0x03FF)	
eTPU A channel 0 configuration register	ETPU_C0CR_A	32-bit	Base + 0x0400	
eTPU A channel 0 status and control register	ETPU_C0SCR_A	32-bit	Base + 0x0404	
eTPU A channel 0 host service request register	ETPU_C0HSRR_A	32-bit	Base + 0x0408	
Reserved	—	—	Base + (0x040C-0x040F)	
eTPU A channel 1 configuration register	ETPU_C1CR_A	32-bit	Base + 0x0410	
eTPU A channel 1 status and control register	ETPU_C1SCR_A	32-bit	Base + 0x0414	
eTPU A channel 1 host service request register	ETPU_C1HSRR_A	32-bit	Base + 0x0418	
Reserved	—	—	Base + (0x041C-0x041F)	
eTPU A channel 2 configuration register	ETPU_C2CR_A	32-bit	Base + 0x0420	
eTPU A channel 2 status and control register	ETPU_C2SCR_A	32-bit	Base + 0x0424	
eTPU A channel 2 host service request register	ETPU_C2HSRR_A	32-bit	Base + 0x0428	
Reserved	—	—	Base + (0x042C-0x042F)	
eTPU A channel 3 configuration register	ETPU_C3CR_A	32-bit	Base + 0x0430	
eTPU A channel 3 status and control register	ETPU_C3SCR_A	32-bit	Base + 0x0434	
eTPU A channel 3 host service request register	ETPU_C3HSRR_A	32-bit	Base + 0x0438	
Reserved	—	—	Base + (0x043C-0x043F)	
eTPU A channel 4 configuration register	ETPU_C4CR_A	32-bit	Base + 0x0440	
eTPU A channel 4 status and control register	ETPU_C4SCR_A	32-bit	Base + 0x0444	
eTPU A channel 4 host service request register	ETPU_C4HSRR_A	32-bit	Base + 0x0448	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
Reserved	—	—	Base + (0x044C-0x044F)	
eTPU A channel 5 configuration register	ETPU_C5CR_A	32-bit	Base + 0x0450	
eTPU A channel 5 status and control register	ETPU_C5SCR_A	32-bit	Base + 0x0454	
eTPU A channel 5 host service request register	ETPU_C5HSRR_A	32-bit	Base + 0x0458	
Reserved	—	—	Base + (0x045C-0x045F)	
eTPU A channel 6 configuration register	ETPU_C6CR_A	32-bit	Base + 0x0460	
eTPU A channel 6 status and control register	ETPU_C6SCR_A	32-bit	Base + 0x0464	
eTPU A channel 6 host service request register	ETPU_C6HSRR_A	32-bit	Base + 0x0468	
Reserved	—	—	Base + (0x046C-0x046F)	
eTPU A channel 7 configuration register	ETPU_C7CR_A	32-bit	Base + 0x0470	
eTPU A channel 7 status and control register	ETPU_C7SCR_A	32-bit	Base + 0x0474	
eTPU A channel 7 host service request register	ETPU_C7HSRR_A	32-bit	Base + 0x0478	
Reserved	—	—	Base + (0x047C-0x047F)	
eTPU A channel 8 configuration register	ETPU_C8CR_A	32-bit	Base + 0x0480	
eTPU A channel 8 status and control register	ETPU_C8SCR_A	32-bit	Base + 0x0484	
eTPU A channel 8 host service request register	ETPU_C8HSRR_A	32-bit	Base + 0x0488	
Reserved	—	—	Base + (0x048C-0x048F)	
eTPU A channel 9 configuration register	ETPU_C9CR_A	32-bit	Base + 0x0490	
eTPU A channel 9 status and control register	ETPU_C9SCR_A	32-bit	Base + 0x0494	
eTPU A channel 9 host service request register	ETPU_C9HSRR_A	32-bit	Base + 0x0498	
Reserved	—	—	Base + (0x049C-0x049F)	
eTPU A channel 10 configuration register	ETPU_C10CR_A	32-bit	Base + 0x04A0	
eTPU A channel 10 status and control register	ETPU_C10SCR_A	32-bit	Base + 0x04A4	
eTPU A channel 10 host service request register	ETPU_C10HSRR_A	32-bit	Base + 0x04A8	
Reserved	—	—	Base + (0x04AC-0x04AF)	
eTPU A channel 11 configuration register	ETPU_C11CR_A	32-bit	Base + 0x04B0	
eTPU A channel 11 status and control register	ETPU_C11SCR_A	32-bit	Base + 0x04B4	
eTPU A channel 11 host service request register	ETPU_C11HSRR_A	32-bit	Base + 0x04B8	
Reserved	—	—	Base + (0x04BC-0x04BF)	
eTPU A channel 12 configuration register	ETPU_C12CR_A	32-bit	Base + 0x04C0	
eTPU A channel 12 status and control register	ETPU_C12SCR_A	32-bit	Base + 0x04C4	
eTPU A channel 12 host service request register	ETPU_C12HSRR_A	32-bit	Base + 0x04C8	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
Reserved	—	—	Base + (0x04CC-0x04CF)	
eTPU A channel 13 configuration register	ETPU_C13CR_A	32-bit	Base + 0x04D0	
eTPU A channel 13 status and control register	ETPU_C13SCR_A	32-bit	Base + 0x04D4	
eTPU A channel 13 host service request register	ETPU_C13HSRR_A	32-bit	Base + 0x04D8	
Reserved	—	—	Base + (0x04DC-0x04DF)	
eTPU A channel 14 configuration register	ETPU_C14CR_A	32-bit	Base + 0x04E0	
eTPU A channel 14 status and control register	ETPU_C14SCR_A	32-bit	Base + 0x04E4	
eTPU A channel 14 host service request register	ETPU_C14HSRR_A	32-bit	Base + 0x04E8	
Reserved	—	—	Base + (0x04EC-0x04EF)	
eTPU A channel 15 configuration register	ETPU_C15CR_A	32-bit	Base + 0x04F0	
eTPU A channel 15 status and control register	ETPU_C15SCR_A	32-bit	Base + 0x04F4	
eTPU A channel 15 host service request register	ETPU_C15HSRR_A	32-bit	Base + 0x04F8	
Reserved	—	—	Base + (0x04FC-0x04FF)	
eTPU A channel 16 configuration register	ETPU_C16CR_A	32-bit	Base + 0x0500	
eTPU A channel 16 status and control register	ETPU_C16SCR_A	32-bit	Base + 0x0504	
eTPU A channel 16 host service request register	ETPU_C16HSRR_A	32-bit	Base + 0x0508	
Reserved	—	—	Base + (0x050C-0x050F)	
eTPU A channel 17 configuration register	ETPU_C17CR_A	32-bit	Base + 0x0510	
eTPU A channel 17 status and control register	ETPU_C17SCR_A	32-bit	Base + 0x0514	
eTPU A channel 17 host service request register	ETPU_C17HSRR_A	32-bit	Base + 0x0518	
Reserved	—	—	Base + (0x051C-0x051F)	
eTPU A channel 18 configuration register	ETPU_C18CR_A	32-bit	Base + 0x0520	
eTPU A channel 18 status and control register	ETPU_C18SCR_A	32-bit	Base + 0x0524	
eTPU A channel 18 host service request register	ETPU_C18HSRR_A	32-bit	Base + 0x0528	
Reserved	—	—	Base + (0x052C-0x052F)	
eTPU A channel 19 configuration register	ETPU_C19CR_A	32-bit	Base + 0x0530	
eTPU A channel 19 status and control register	ETPU_C19SCR_A	32-bit	Base + 0x0534	
eTPU A channel 19 host service request register	ETPU_C19HSRR_A	32-bit	Base + 0x0538	
Reserved	—	—	Base + (0x053C-0x053F)	
eTPU A channel 20 configuration register	ETPU_C20CR_A	32-bit	Base + 0x0540	
eTPU A channel 20 status and control register	ETPU_C20SCR_A	32-bit	Base + 0x0544	
eTPU A channel 20 host service request register	ETPU_C20HSRR_A	32-bit	Base + 0x0548	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
Reserved	—	—	Base + (0x054C-0x054F)	
eTPU A channel 21 configuration register	ETPU_C21CR_A	32-bit	Base + 0x0550	
eTPU A channel 21 status and control register	ETPU_C21SCR_A	32-bit	Base + 0x0554	
eTPU A channel 21 host service request register	ETPU_C21HSRR_A	32-bit	Base + 0x0558	
Reserved	—	—	Base + (0x055C-0x055F)	
eTPU A channel 22 configuration register	ETPU_C22CR_A	32-bit	Base + 0x0560	
eTPU A channel 22 status and control register	ETPU_C22SCR_A	32-bit	Base + 0x0564	
eTPU A channel 22 host service request register	ETPU_C22HSRR_A	32-bit	Base + 0x0568	
Reserved	—	—	Base + (0x056C-0x056F)	
eTPU A channel 23 configuration register	ETPU_C23CR_A	32-bit	Base + 0x0570	
eTPU A channel 23 status and control register	ETPU_C23CR_A	32-bit	Base + 0x0574	
eTPU A channel 23 host service request register	ETPU_C23HSRR_A	32-bit	Base + 0x0578	
Reserved	—	—	Base + (0x057C-0x057F)	
eTPU A channel 24 configuration register	ETPU_C24CR_A	32-bit	Base + 0x0580	
eTPU A channel 24 status and control register	ETPU_C24SCR_A	32-bit	Base + 0x0584	
eTPU A channel 24 host service request register	ETPU_C24HSRR_A	32-bit	Base + 0x0588	
Reserved	—	—	Base + (0x058C-0x058F)	
eTPU A channel 25 configuration register	ETPU_C25CR_A	32-bit	Base + 0x0590	
eTPU A channel 25 status and control register	ETPU_C25SCR_A	32-bit	Base + 0x0594	
eTPU A channel 25 host service request register	ETPU_C25HSRR_A	32-bit	Base + 0x0598	
Reserved	—	—	Base + (0x059C-0x059F)	
eTPU A channel 26 configuration register	ETPU_C26CR_A	32-bit	Base + 0x05A0	
eTPU A channel 26 status and control register	ETPU_C26SCR_A	32-bit	Base + 0x05A4	
eTPU A channel 26 host service request register	ETPU_C26HSRR_A	32-bit	Base + 0x05A8	
Reserved	—	—	Base + (0x05AC-0x05AF)	
eTPU A channel 27 configuration register	ETPU_C27CR_A	32-bit	Base + 0x05B0	
eTPU A channel 27 status and control register	ETPU_C27SCR_A	32-bit	Base + 0x05B4	
eTPU A channel 27 host service request register	ETPU_C27HSRR_A	32-bit	Base + 0x05B8	
Reserved	—	—	Base + (0x05BC-0x05BF)	
eTPU A channel 28 configuration register	ETPU_C28CR_A	32-bit	Base + 0x05C0	
eTPU A channel 28 status and control register	ETPU_C28SCR_A	32-bit	Base + 0x05C4	
eTPU A channel 28 host service request register	ETPU_C28HSRR_A	32-bit	Base + 0x05C8	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
Reserved	—	—	Base + (0x05CC-0x05CF)	
eTPU A channel 29 configuration register	ETPU_C29CR_A	32-bit	Base + 0x05D0	
eTPU A channel 29 status and control register	ETPU_C29SCR_A	32-bit	Base + 0x05D4	
eTPU A channel 29 host service request register	ETPU_C29HSRR_A	32-bit	Base + 0x05D8	
Reserved	—	—	Base + (0x05DC-0x05DF)	
eTPU A channel 30 configuration register	ETPU_C30CR_A	32-bit	Base + 0x05E0	
eTPU A channel 30 status and control register	ETPU_C30SCR_A	32-bit	Base + 0x05E4	
eTPU A channel 30 host service request register	ETPU_C30HSRR_A	32-bit	Base + 0x05E8	
Reserved	—	—	Base + (0x05EC-0x05EF)	
eTPU A channel 31 configuration register	ETPU_C31CR_A	32-bit	Base + 0x05F0	
eTPU A channel 31 status and control register	ETPU_C31SCR_A	32-bit	Base + 0x05F4	
eTPU A channel 31 host service request register	ETPU_C31HSRR_A	32-bit	Base + 0x05F8	
Reserved	—	—	Base + (0x05FC-0x07FF)	
eTPU B channel 0 configuration register [2]	ETPU_C0CR_B[2]	32-bit	Base + 0x0800	
eTPU B channel 0 status and control register [2]	ETPU_C0SCR_B[2]	32-bit	Base + 0x0804	
eTPU B channel 0 host service request register [2]	ETPU_C0HSRR_B[2]	32-bit	Base + 0x0808	
Reserved	—	—	Base + (0x080C-0x080F)	
eTPU B channel 1 configuration register [2]	ETPU_C1CR_B[2]	32-bit	Base + 0x0810	
eTPU B channel 1 status and control register [2]	ETPU_C1SCR_B[2]	32-bit	Base + 0x0814	
eTPU B channel 1 host service request register [2]	ETPU_C1HSRR_B[2]	32-bit	Base + 0x0818	
Reserved	—	—	Base + (0x081C-0x081F)	
eTPU B channel 2 configuration register [2]	ETPU_C2CR_B[2]	32-bit	Base + 0x0820	
eTPU B channel 2 status and control register [2]	ETPU_C2SCR_B[2]	32-bit	Base + 0x0824	
eTPU B channel 2 host service request register [2]	ETPU_C2HSRR_B[2]	32-bit	Base + 0x0828	
Reserved	—	—	Base + (0x082C-0x082F)	
eTPU B channel 3 configuration register [2]	ETPU_C3CR_B[2]	32-bit	Base + 0x0830	
eTPU B channel 3 status and control register [2]	ETPU_C3SCR_B[2]	32-bit	Base + 0x0834	
eTPU B channel 3 host service request register [2]	ETPU_C3HSRR_B[2]	32-bit	Base + 0x0838	
Reserved	—	—	Base + (0x083C-0x083F)	
eTPU B channel 4 configuration register [2]	ETPU_C4CR_B[2]	32-bit	Base + 0x0840	
eTPU B channel 4 status and control register [2]	ETPU_C4SCR_B[2]	32-bit	Base + 0x0844	
eTPU B channel 4 host service request register [2]	ETPU_C4HSRR_B[2]	32-bit	Base + 0x0848	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
Reserved	—	—	Base + (0x084C-0x084F)	
eTPU B channel 5 configuration register [2]	ETPU_C5CR_B[2]	32-bit	Base + 0x0850	
eTPU B channel 5 status and control register [2]	ETPU_C5SCR_B[2]	32-bit	Base + 0x0854	
eTPU B channel 5 host service request register [2]	ETPU_C5HSRR_B[2]	32-bit	Base + 0x0858	
Reserved	—	—	Base + (0x085C-0x085F)	
eTPU B channel 6 configuration register [2]	ETPU_C6CR_B[2]	32-bit	Base + 0x0860	
eTPU B channel 6 status and control register [2]	ETPU_C6SCR_B[2]	32-bit	Base + 0x0864	
eTPU B channel 6 host service request register [2]	ETPU_C6HSRR_B[2]	32-bit	Base + 0x0868	
Reserved	—	—	Base + (0x086C-0x086F)	
eTPU B channel 7 configuration register [2]	ETPU_C7CR_B[2]	32-bit	Base + 0x0870	
eTPU B channel 7 status and control register [2]	ETPU_C7SCR_B[2]	32-bit	Base + 0x0874	
eTPU B channel 7 host service request register [2]	ETPU_C7HSRR_B[2]	32-bit	Base + 0x0878	
Reserved	—	—	Base + (0x087C-0x087F)	
eTPU B channel 8 configuration register [2]	ETPU_C8CR_B[2]	32-bit	Base + 0x0880	
eTPU B channel 8 status and control register [2]	ETPU_C8SCR_B[2]	32-bit	Base + 0x0884	
eTPU B channel 8 host service request register [2]	ETPU_C8HSRR_B[2]	32-bit	Base + 0x0888	
Reserved	—	—	Base + (0x088C-0088F)	
eTPU B channel 9 configuration register [2]	ETPU_C9CR_B[2]	32-bit	Base + 0x0890	
eTPU B channel 9 status and control register [2]	ETPU_C9SCR_B[2]	32-bit	Base + 0x0894	
eTPU B channel 9 host service request register [2]	ETPU_C9HSRR_B[2]	32-bit	Base + 0x0898	
Reserved	—	—	Base + (0x081C-0x081F)	
eTPU B channel 10 configuration register [2]	ETPU_C10CR_B[2]	32-bit	Base + 0x08A0	
eTPU B channel 10 status and control register [2]	ETPU_C10SCR_B[2]	32-bit	Base + 0x08A4	
eTPU B channel 10 host service request register [2]	ETPU_C10HSRR_B[2]	32-bit	Base + 0x08A8	
Reserved	—	—	Base + (0x08AC-0x08AF)	
eTPU B channel 11 configuration register [2]	ETPU_C11CR_B[2]	32-bit	Base + 0x08B0	
eTPU B channel 11 status and control register [2]	ETPU_C11SCR_B[2]	32-bit	Base + 0x08B4	
eTPU B channel 11 host service request register [2]	ETPU_C11HSRR_B[2]	32-bit	Base + 0x08B8	
Reserved	—	—	Base + (0x08BC-0x08BF)	
eTPU B channel 12 configuration register [2]	ETPU_C12CR_B[2]	32-bit	Base + 0x08C0	
eTPU B channel 12 status and control register [2]	ETPU_C12SCR_B[2]	32-bit	Base + 0x08C4	
eTPU B channel 12 host service request register [2]	ETPU_C12HSRR_B[2]	32-bit	Base + 0x08C8	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
Reserved	—	—	Base + (0x08CC-0x08CF)	
eTPU B channel 13 configuration register [2]	ETPU_C13CR_B[2]	32-bit	Base + 0x08D0	
eTPU B channel 13 status and control register [2]	ETPU_C13SCR_B[2]	32-bit	Base + 0x08D4	
eTPU B channel 13 host service request register [2]	ETPU_C13HSRR_B[2]	32-bit	Base + 0x08D8	
Reserved	—	—	Base + (0x08DC-0x08DF)	
eTPU B channel 14 configuration register[2]	ETPU_C14CR_B[2]	32-bit	Base + 0x08E0	
eTPU B channel 14 status and control register [2]	ETPU_C14SCR_B[2]	32-bit	Base + 0x08E4	
eTPU B channel 14 host service request register [2]	ETPU_C14HSRR_B[2]	32-bit	Base + 0x08E8	
Reserved	—	—	Base + (0x08EC-0x08EF)	
eTPU B channel 15 configuration register [2]	ETPU_C15CR_B[2]	32-bit	Base + 0x08F0	
eTPU B channel 15 status and control register [2]	ETPU_C15SCR_B[2]	32-bit	Base + 0x08F4	
eTPU B channel 15 host service request register [2]	ETPU_C15HSRR_B[2]	32-bit	Base + 0x08F8	
Reserved	—	—	Base + (0x08FC-0x08FF)	
eTPU B channel 16 configuration register [2]	ETPU_C16CR_B[2]	32-bit	Base + 0x0900	
eTPU B channel 16 status and control register [2]	ETPU_C16SCR_B[2]	32-bit	Base + 0x0904	
eTPU B channel 16 host service request register [2]	ETPU_C16HSRR_B[2]	32-bit	Base + 0x0908	
Reserved	—	—	Base + (0x090C-0x090F)	
eTPU B channel 17 configuration register [2]	ETPU_C17CR_B[2]	32-bit	Base + 0x0910	
eTPU B channel 17 status and control register [2]	ETPU_C17SCR_B[2]	32-bit	Base + 0x0914	
eTPU B channel 17 host service request register [2]	ETPU_C17HSRR_B[2]	32-bit	Base + 0x0918	
Reserved	—	—	Base + (0x091C-0x091F)	
eTPU B channel 18 configuration register [2]	ETPU_C18CR_B[2]	32-bit	Base + 0x0920	
eTPU B channel 18 status and control register [2]	ETPU_C18SCR_B[2]	32-bit	Base + 0x0924	
eTPU B channel 18 host service request register [2]	ETPU_C18HSRR_B[2]	32-bit	Base + 0x0928	
Reserved	—	—	Base + (0x092C-0x092F)	
eTPU B channel 19 configuration register [2]	ETPU_C19CR_B[2]	32-bit	Base + 0x0930	
eTPU B channel 19 status and control register [2]	ETPU_C19SCR_B[2]	32-bit	Base + 0x0934	
eTPU B channel 19 host service request register [2]	ETPU_C19HSRR_B[2]	32-bit	Base + 0x0938	
Reserved	—	—	Base + (0x093C-0x093F)	
eTPU B channel 20 configuration register [2]	ETPU_C20CR_B[2]	32-bit	Base + 0x0940	
eTPU B channel 20 status and control register [2]	ETPU_C20SCR_B[2]	32-bit	Base + 0x0944	
eTPU B channel 20 host service request register [2]	ETPU_C20HSRR_B[2]	32-bit	Base + 0x0948	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
Reserved	—	—	Base + (0x094C-0x094F)	
eTPU B channel 21 configuration register [2]	ETPU_C21CR_B[2]	32-bit	Base + 0x0950	
eTPU B channel 21 status and control register [2]	ETPU_C21SCR_B[2]	32-bit	Base + 0x0954	
eTPU B channel 21 host service request register [2]	ETPU_C21HSRR_B[2]	32-bit	Base + 0x0958	
Reserved	—	—	Base + (0x095C-0x095F)	
eTPU B channel 22 configuration register [2]	ETPU_C22CR_B[2]	32-bit	Base + 0x0960	
eTPU B channel 22 status and control register [2]	ETPU_C22SCR_B[2]	32-bit	Base + 0x0964	
eTPU B channel 22 host service request register [2]	ETPU_C22HSRR_B[2]	32-bit	Base + 0x0968	
Reserved	—	—	Base + (0x096C-0x096F)	
eTPU B channel 23 configuration register [2]	ETPU_C23CR_B[2]	32-bit	Base + 0x0970	
eTPU B channel 23 status and control register [2]	ETPU_C23SCR_B[2]	32-bit	Base + 0x0974	
eTPU B channel 23 host service request register [2]	ETPU_C23HSRR_B[2]	32-bit	Base + 0x0978	
Reserved	—	—	Base + (0x097C-0x097F)	
eTPU B channel 24 configuration register [2]	ETPU_C24CR_B[2]	32-bit	Base + 0x0980	
eTPU B channel 24 status and control register [2]	ETPU_C24SCR_B[2]	32-bit	Base + 0x0984	
eTPU B channel 24 host service request register [2]	ETPU_C24HSRR_B[2]	32-bit	Base + 0x0988	
Reserved	—	—	Base + (0x098C-0x098F)	
eTPU B channel 25 configuration register [2]	ETPU_C25CR_B[2]	32-bit	Base + 0x0990	
eTPU B channel 25 status and control register [2]	ETPU_C25SCR_B[2]	32-bit	Base + 0x0994	
eTPU B channel 25 host service request register [2]	ETPU_C25HSRR_B[2]	32-bit	Base + 0x0998	
Reserved	—	—	Base + (0x099C-0x099F)	
eTPU B channel 26 configuration register [2]	ETPU_C26CR_B[2]	32-bit	Base + 0x09A0	
eTPU B channel 26 status and control register [2]	ETPU_C26SCR_B[2]	32-bit	Base + 0x09A4	
eTPU B channel 26 host service request register [2]	ETPU_C26HSRR_B[2]	32-bit	Base + 0x09A8	
Reserved	—	—	Base + (0x09AC-0x09AF)	
eTPU B channel 27 configuration register [2]	ETPU_C27CR_B[2]	32-bit	Base + 0x09B0	
eTPU B channel 27 status and control register [2]	ETPU_C27SCR_B[2]	32-bit	Base + 0x09B4	
eTPU B channel 27 host service request register [2]	ETPU_C27HSRR_B[2]	32-bit	Base + 0x09B8	
Reserved	—	—	Base + (0x09BC-0x09BF)	
eTPU B channel 28 configuration register [2]	ETPU_C28CR_B[2]	32-bit	Base + 0x09C0	
eTPU B channel 28 status and control register [2]	ETPU_C28SCR_B[2]	32-bit	Base + 0x09C4	
eTPU B channel 28 host service request register [2]	ETPU_C28HSRR_B[2]	32-bit	Base + 0x09C8	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
Reserved	—	—	Base + (0x09CC-0x09CF)	
eTPU B channel 29 configuration register [2]	ETPU_C29CR_B[2]	32-bit	Base + 0x09D0	
eTPU B channel 29 status and control register [2]	ETPU_C29SCR_B[2]	32-bit	Base + 0x09D4	
eTPU B channel 29 host service request register [2]	ETPU_C29HSRR_B[2]	32-bit	Base + 0x09D8	
Reserved	—	—	Base + (0x09DC-0x09DF)	
eTPU B channel 30 configuration register [2]	ETPU_C30CR_B[2]	32-bit	Base + 0x09E0	
eTPU B channel 30 status and control register [2]	ETPU_C30SCR_B[2]	32-bit	Base + 0x09E4	
eTPU B channel 30 host service request register [2]	ETPU_C30HSRR_B[2]	32-bit	Base + 0x09E8	
Reserved	—	—	Base + (0x09EC-0x09EF)	
eTPU B channel 31 configuration register [2]	ETPU_C31CR_B[2]	32-bit	Base + 0x09F0	
eTPU B channel 31 status and control register [2]	ETPU_C31SCR_B[2]	32-bit	Base + 0x09F4	
eTPU B channel 31 host service request register [2]	ETPU_C31HSRR_B[2]	32-bit	Base + 0x09F8	
Reserved	—	—	Base + (0x09FC-0x7FFF)	
Shared data memory (parameter RAM)	SDM	3Kbyte	Base + (0x8000-0x8BFF)	
Reserved	—	—	Base + (0x8C00-0xBFFF)	
SDM PSE mirror			Base + (0xC000-0xCBFF)	
Reserved	—	—	Base + (0xCC00-0xFFFF)	
Shared code memory	SCM	16 Kbyte (5554) 12Kbyte (5553)	Base + (0x1_0000-1_3FFF) (MPC5554) + (0x1_0000-1_2FFF) (MPC5553)	
Reserved	—	—	Base + (0x1_4000-FFEF_FFFF)	
Peripheral Bridge B (PBRIDGEB)			0xFFF0_0000	Chapter 5, "Peripheral Bridge (PBRIDGE_A, PBRIDGE_B)"
Peripheral bridge B master privilege control register	PBRIDGEB_MPCR	32-bit	Base + 0x0000	
Reserved	—	—	Base + (0x0004-0x001F)	
Peripheral bridge B peripheral access control register 0	PBRIDGEB_PACR0	32-bit	Base + 0x0020	
Reserved	—	—	Base + (0x0024-0x0027)	
Peripheral bridge B peripheral access control register 2	PBRIDGEB_PACR2	32-bit	Base + 0x0028	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
Reserved	—	—	Base + (0x002C-0x003F)	
Peripheral bridge B off-platform peripheral access control register 0	PBRIDGEB_OPACR0	32-bit	Base + 0x0040	
Peripheral bridge B off-platform peripheral access control register 1	PBRIDGEB_OPACR1	32-bit	Base + 0x0044	
Peripheral bridge B off-platform peripheral access control register 2	PBRIDGEB_OPACR2	32-bit	Base + 0x0048	
Peripheral bridge B off-platform peripheral access control register 3	PBRIDGEB_OPACR3	32-bit	Base + 0x004C	
Reserved	—	—	(Base + 0x0050)-0xFFF0_3FFF)	
System Bus Crossbar Switch (XBAR)			0xFFF0_4000	Chapter 7, "Crossbar Switch (XBAR)"
Master priority register 0	XBAR_MPR0	32-bit	Base + 0x0000	
Reserved	—	—	Base + (0x0004-0x000F)	
Slave general purpose control register 0	XBAR_SGPCR0	32-bit	Base + 0x0010	
Reserved	—	—	Base + (0x0014-0x00FF)	
Master priority register 1	XBAR_MPR1	32-bit	Base + 0x0100	
Reserved	—	—	Base + (0x0104-0x010F)	
Slave general purpose control register 1	XBAR_SGPCR1	32-bit	Base + 0x0110	
Reserved	—	—	Base + (0x0114-0x02FF)	
Master priority register 3	XBAR_MPR3	32-bit	Base + 0x0300	
Reserved	—	—	Base + (0x0304-0x030F)	
Slave general purpose control register 3	XBAR_SGPCR3	32-bit	Base + 0x0310	
Reserved	—	—	Base + (0x0314-0x05FF)	
Master priority register 6	XBAR_MPR6	32-bit	Base + 0x0600	
Reserved	—	—	Base + (0x0604-0x060F)	
Slave general purpose control register 6	XBAR_SGPCR6	32-bit	Base + 0x0610	
Reserved	—	—	Base + (0x0614-0x06FF)	
Master priority register 7	XBAR_MPR7	32-bit	Base + 0x0700	
Reserved	—	—	Base + (0x0704-0x070F)	
Slave general purpose control register 7	XBAR_SGPCR7	32-bit	Base + 0x0710	
Reserved	—	—	(Base + 0x0714)-0xFFF4_3FFF)	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
Error Correction Status Module (ECSM)			0xFFF4_0000	
Reserved	—	—	Base + (0x0000-0x0015)	
Software watchdog timer control register	ECSM_SWTCR[1]	16-bit	Base + 0x0016	
Reserved	—	—	Base + (0x0018-0x001A)	
Software watchdog timer service register	ECSM_SWTSR [1]	8-bit	Base + 0x001B	
Reserved	—	—	Base + (0x001C-0x001E)	
Software watchdog timer interrupt register	ECSM_SWTIR [1]	8-bit	Base + 0x001F	
Reserved	—		Base + (0x0020-0x0023)	
FEC Burst Optimization Master Control register	FBOMCR	32-bit	Base + 0x0024	
Reserved	—	—	Base + (0x0028-0x0042)	
ECC configuration register	ECSM_ECR	8-bit	Base + 0x0043	
Reserved	—	—	Base + (0x0044-0x0046)	
ECC status register	ECSM_ESR	8-bit	Base + 0x0047	
Reserved	—	—	Base + (0x0048-0x0049)	
ECC error generation register	ECSM_EEGR	16-bit	Base + 0x004A	
Reserved	—	—	Base + (0x004C-0x004F)	
Flash ECC address register	ECSM_FEAR	32-bit	Base + 0x0050	
Reserved	—	—	Base + (0x0054-0x0055)	
Flash ECC master number register	ECSM_FEMR	8-bit	Base + 0x0056	
Flash ECC attributes register	ECSM_FEAT	8-bit	Base + 0x0057	
Flash ECC data register high	ECSM_FEDRH	32-bit	Base + 0x0058	
Flash ECC data register low	ECSM_FEDRL	32-bit	Base + 0x005C	
RAM ECC address register	ECSM_REAR	32-bit	Base + 0x0060	
Reserved	—	—	Base + (0x0064-0x0065)	
RAM ECC master number register	ECSM_REMR	8-bit	Base + 0x0066	
RAM ECC attributes register	ECSM_REAT	8-bit	Base + 0x0067	
RAM ECC data register high	ECSM_REDRH	32-bit	Base + 0x0068	
RAM ECC data register low	ECSM_REDRL	32-bit	Base + 0x006C	
Reserved	—	—	(Base + 0x0070)-0xFFF4_3FFF	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
Enhanced Direct Memory Access (eDMA)			0xFFF4_4000	Chapter 9, "Enhanced Direct Memory Access (eDMA)"
Control register	EDMA_CR	32-bit	Base + 0x0000	
Error status register	EDMA_ESR	32-bit	Base + 0x0004	
Enable request register high (MPC5554 only)	EDMA_ERQRH	32-bit	Base + 0x0008	
Enable request register low	EDMA_ERQRL	32-bit	Base + 0x000C	
Enable error interrupt register high (MPC5554 only)	EDMA_EEIRH	32-bit	Base + 0x0010	
Enable error interrupt register low	EDMA_EEIRL	32-bit	Base + 0x0014	
Set enable request register	EDMA_SERQR	8-bit	Base + 0x0018	
Clear enable request register	EDMA_CERQR	8-bit	Base + 0x0019	
Set enable error interrupt register	EDMA_SEEIR	8-bit	Base + 0x001A	
Clear enable error interrupt request register	EDMA_CEEIR	8-bit	Base + 0x001B	
Clear interrupt request register	EDMA_CIRQR	8-bit	Base + 0x001C	
Clear error register	EDMA_CER	8-bit	Base + 0x001D	
Set START bit register	EDMA_SSBR	8-bit	Base + 0x001E	
Clear DONE status bit register	EDMA_CDSBR	8-bit	Base + 0x001F	
Interrupt request register high (MPC5554 only)	EDMA_IRQRH	32-bit	Base + 0x0020	
Interrupt request register low	EDMA_IRQRL	32-bit	Base + 0x0024	
Error register high (MPC5554 only)	EDMA_ERH	32-bit	Base + 0x0028	
Error register low	EDMA_ERL	32-bit	Base + 0x002C	
Reserved	—	—	Base + (0x0030-0x00FF)	
Channel priority register 0	EDMA_CPR0	8-bit	Base + 0x0100	
Channel priority register 1	EDMA_CPR1	8-bit	Base + 0x0101	
Channel priority register 2	EDMA_CPR2	8-bit	Base + 0x0102	
Channel priority register 3	EDMA_CPR3	8-bit	Base + 0x0103	
Channel priority register 4	EDMA_CPR4	8-bit	Base + 0x0104	
Channel priority register 5	EDMA_CPR5	8-bit	Base + 0x0105	
Channel priority register 6	EDMA_CPR6	8-bit	Base + 0x0106	
Channel priority register 7	EDMA_CPR7	8-bit	Base + 0x0107	
Channel priority register 8	EDMA_CPR8	8-bit	Base + 0x0108	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
Channel priority register 9	EDMA_CPR9	8-bit	Base + 0x0109	
Channel priority register 10	EDMA_CPR10	8-bit	Base + 0x010A	
Channel priority register 11	EDMA_CPR11	8-bit	Base + 0x010B	
Channel priority register 12	EDMA_CPR12	8-bit	Base + 0x010C	
Channel priority register 13	EDMA_CPR13	8-bit	Base + 0x010D	
Channel priority register 14	EDMA_CPR14	8-bit	Base + 0x010E	
Channel priority register 15	EDMA_CPR15	8-bit	Base + 0x010F	
Channel priority register 16	EDMA_CPR16	8-bit	Base + 0x0110	
Channel priority register 17	EDMA_CPR17	8-bit	Base + 0x0111	
Channel priority register 18	EDMA_CPR18	8-bit	Base + 0x0112	
Channel priority register 19	EDMA_CPR19	8-bit	Base + 0x0113	
Channel priority register 20	EDMA_CPR20	8-bit	Base + 0x0114	
Channel priority register 21	EDMA_CPR21	8-bit	Base + 0x0115	
Channel priority register 22	EDMA_CPR22	8-bit	Base + 0x0116	
Channel priority register 23	EDMA_CPR23	8-bit	Base + 0x0117	
Channel priority register 24	EDMA_CPR24	8-bit	Base + 0x0118	
Channel priority register 25	EDMA_CPR25	8-bit	Base + 0x0119	
Channel priority register 26	EDMA_CPR26	8-bit	Base + 0x011A	
Channel priority register 27	EDMA_CPR27	8-bit	Base + 0x011B	
Channel priority register 28	EDMA_CPR28	8-bit	Base + 0x011C	
Channel priority register 29	EDMA_CPR29	8-bit	Base + 0x011D	
Channel priority register 30	EDMA_CPR30	8-bit	Base + 0x011E	
Channel priority register 31	EDMA_CPR31	8-bit	Base + 0x011F	
Channel priority register 32	EDMA_CPR32	8-bit	Base + 0x0120	
Channel priority register 33	EDMA_CPR33	8-bit	Base + 0x0121	
Channel priority register 34	EDMA_CPR34	8-bit	Base + 0x0122	
Channel priority register 35	EDMA_CPR35	8-bit	Base + 0x0123	
Channel priority register 36	EDMA_CPR36	8-bit	Base + 0x0124	
Channel priority register 37	EDMA_CPR37	8-bit	Base + 0x0125	
Channel priority register 38	EDMA_CPR38	8-bit	Base + 0x0126	
Channel priority register 39	EDMA_CPR39	8-bit	Base + 0x0127	
Channel priority register 40	EDMA_CPR40	8-bit	Base + 0x0128	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
Channel priority register 41	EDMA_CPR41	8-bit	Base + 0x0129	
Channel priority register 42	EDMA_CPR42	8-bit	Base + 0x012A	
Channel priority register 43	EDMA_CPR43	8-bit	Base + 0x012B	
Channel priority register 44	EDMA_CPR44	8-bit	Base + 0x012C	
Channel priority register 45	EDMA_CPR45	8-bit	Base + 0x012D	
Channel priority register 46	EDMA_CPR46	8-bit	Base + 0x012E	
Channel priority register 47	EDMA_CPR47	8-bit	Base + 0x012F	
Channel priority register 48	EDMA_CPR48	8-bit	Base + 0x0130	
Channel priority register 49	EDMA_CPR49	8-bit	Base + 0x0131	
Channel priority register 50	EDMA_CPR50	8-bit	Base + 0x0132	
Channel priority register 51	EDMA_CPR51	8-bit	Base + 0x0133	
Channel priority register 52	EDMA_CPR52	8-bit	Base + 0x0134	
Channel priority register 53	EDMA_CPR53	8-bit	Base + 0x0135	
Channel priority register 54	EDMA_CPR54	8-bit	Base + 0x0136	
Channel priority register 55	EDMA_CPR55	8-bit	Base + 0x0137	
Channel priority register 56	EDMA_CPR56	8-bit	Base + 0x0138	
Channel priority register 57	EDMA_CPR57	8-bit	Base + 0x0139	
Channel priority register 58	EDMA_CPR58	8-bit	Base + 0x013A	
Channel priority register 59	EDMA_CPR59	8-bit	Base + 0x013B	
Channel priority register 60	EDMA_CPR60	8-bit	Base + 0x013C	
Channel priority register 61	EDMA_CPR61	8-bit	Base + 0x013D	
Channel priority register 62	EDMA_CPR62	8-bit	Base + 0x013E	
Channel priority register 63	EDMA_CPR63	8-bit	Base + 0x013F	
Reserved	—	—	Base + (0x0140-0x0FFF)	
Transfer control descriptor register 0	TCD0	256-bit	Base + 0x1000	
Transfer control descriptor register 1	TCD1	256-bit	Base + 0x1020	
Transfer control descriptor register 2	TCD2	256-bit	Base + 0x1040	
Transfer control descriptor register 3	TCD3	256-bit	Base + 0x1060	
Transfer control descriptor register 4	TCD4	256-bit	Base + 0x1080	
Transfer control descriptor register 5	TCD5	256-bit	Base + 0x10A0	
Transfer control descriptor register 6	TCD6	256-bit	Base + 0x10C0	
Transfer control descriptor register 7	TCD7	256-bit	Base + 0x10E0	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
Transfer control descriptor register 8	TCD8	256-bit	Base + 0x1100	
Transfer control descriptor register 9	TCD9	256-bit	Base + 0x1120	
Transfer control descriptor register 10	TCD10	256-bit	Base + 0x1140	
Transfer control descriptor register 11	TCD11	256-bit	Base + 0x1160	
Transfer control descriptor register 12	TCD12	256-bit	Base + 0x1180	
Transfer control descriptor register 13	TCD13	256-bit	Base + 0x11A0	
Transfer control descriptor register 14	TCD14	256-bit	Base + 0x11C0	
Transfer control descriptor register 15	TCD15	256-bit	Base + 0x11E0	
Transfer control descriptor register 16	TCD16	256-bit	Base + 0x1200	
Transfer control descriptor register 17	TCD17	256-bit	Base + 0x1220	
Transfer control descriptor register 18	TCD18	256-bit	Base + 0x1240	
Transfer control descriptor register 19	TCD19	256-bit	Base + 0x1260	
Transfer control descriptor register 20	TCD20	256-bit	Base + 0x1280	
Transfer control descriptor register 21	TCD21	256-bit	Base + 0x12A0	
Transfer control descriptor register 22	TCD22	256-bit	Base + 0x12C0	
Transfer control descriptor register 23	TCD23	256-bit	Base + 0x12E0	
Transfer control descriptor register 24	TCD24	256-bit	Base + 0x1300	
Transfer control descriptor register 25	TCD25	256-bit	Base + 0x1320	
Transfer control descriptor register 26	TCD26	256-bit	Base + 0x1340	
Transfer control descriptor register 27	TCD27	256-bit	Base + 0x1360	
Transfer control descriptor register 28	TCD28	256-bit	Base + 0x1380	
Transfer control descriptor register 29	TCD29	256-bit	Base + 0x13A0	
Transfer control descriptor register 30	TCD30	256-bit	Base + 0x13C0	
Transfer control descriptor register 31	TCD31	256-bit	Base + 0x13E0	
Transfer control descriptor register 32	TCD32	256-bit	Base + 0x1400	
Transfer control descriptor register 33	TCD33	256-bit	Base + 0x1420	
Transfer control descriptor register 34	TCD34	256-bit	Base + 0x1440	
Transfer control descriptor register 35	TCD35	256-bit	Base + 0x1460	
Transfer control descriptor register 36	TCD36	256-bit	Base + 0x1480	
Transfer control descriptor register 37	TCD37	256-bit	Base + 0x14A0	
Transfer control descriptor register 38	TCD38	256-bit	Base + 0x14C0	
Transfer control descriptor register 39	TCD39	256-bit	Base + 0x14E0	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
Transfer control descriptor register 40	TCD40	256-bit	Base + 0x1500	
Transfer control descriptor register 41	TCD41	256-bit	Base + 0x1520	
Transfer control descriptor register 42	TCD42	256-bit	Base + 0x1540	
Transfer control descriptor register 43	TCD43	256-bit	Base + 0x1560	
Transfer control descriptor register 44	TCD44	256-bit	Base + 0x1580	
Transfer control descriptor register 45	TCD45	256-bit	Base + 0x15A0	
Transfer control descriptor register 46	TCD46	256-bit	Base + 0x15C0	
Transfer control descriptor register 47	TCD47	256-bit	Base + 0x15E0	
Transfer control descriptor register 48	TCD48	256-bit	Base + 0x1600	
Transfer control descriptor register 49	TCD49	256-bit	Base + 0x1620	
Transfer control descriptor register 50	TCD50	256-bit	Base + 0x1640	
Transfer control descriptor register 51	TCD51	256-bit	Base + 0x1660	
Transfer control descriptor register 52	TCD52	256-bit	Base + 0x1680	
Transfer control descriptor register 53	TCD53	256-bit	Base + 0x16A0	
Transfer control descriptor register 54	TCD54	256-bit	Base + 0x16C0	
Transfer control descriptor register 55	TCD55	256-bit	Base + 0x16E0	
Transfer control descriptor register 56	TCD56	256-bit	Base + 0x1700	
Transfer control descriptor register 57	TCD57	256-bit	Base + 0x1720	
Transfer control descriptor register 58	TCD58	256-bit	Base + 0x1740	
Transfer control descriptor register 59	TCD59	256-bit	Base + 0x1760	
Transfer control descriptor register 60	TCD60	256-bit	Base + 0x1780	
Transfer control descriptor register 61	TCD61	256-bit	Base + 0x17A0	
Transfer control descriptor register 62	TCD62	256-bit	Base + 0x17C0	
Transfer control descriptor register 63	TCD63	256-bit	Base + 0x17E0	
Reserved	—	—	(Base + 0x1800)-0xFFF4_7FFF	
Interrupt Controller (INTC)			0xFFF4_8000	Chapter 10, “Interrupt Controller (INTC)’
Module configuration register	INTC_MCR	32-bit	Base + 0x0000	
Reserved	—	—	Base + (0x0004-0x0007)	
Current priority register	INTC_CPR	32-bit	Base + 0x0008	
Reserved	—	—	Base + (0x000C-0x000F)	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
interrupt acknowledge register	INTC_IACKR	32-bit	Base + 0x0010	
Reserved	—	—	Base + (0x0014-0x0017)	
End of interrupt register	INTC_EOIR	32-bit	Base + 0x0018	
Reserved	—	—	Base + (0x001C-0x001F)	
Software set/clear interrupt register 0	INTC_SSCIR0	8-bit	Base + 0x0020	
Software set/clear interrupt register 1	INTC_SSCIR1	8-bit	Base + 0x0021	
Software set/clear interrupt register 2	INTC_SSCIR2	8-bit	Base + 0x0022	
Software set/clear interrupt register 3	INTC_SSCIR3	8-bit	Base + 0x0023	
Software set/clear interrupt register 4	INTC_SSCIR4	8-bit	Base + 0x0024	
Software set/clear interrupt register 5	INTC_SSCIR5	8-bit	Base + 0x0025	
Software set/clear interrupt register 6	INTC_SSCIR6	8-bit	Base + 0x0026	
Software set/clear interrupt register 7	INTC_SSCIR7	8-bit	Base + 0x0027	
Reserved	—	—	Base + (0x0028-0x003F)	
Priority select register 0	INTC_PSR0	8-bit	Base + 0x0040	
Priority select register 1	INTC_PSR1	8-bit	Base + 0x0041	
Priority select register 2	INTC_PSR2	8-bit	Base + 0x0042	
Priority select register 3	INTC_PSR3	8-bit	Base + 0x0043	
Priority select register 4	INTC_PSR4	8-bit	Base + 0x0044	
Priority select register 5	INTC_PSR5	8-bit	Base + 0x0045	
Priority select register 6	INTC_PSR6	8-bit	Base + 0x0046	
Priority select register 7	INTC_PSR7	8-bit	Base + 0x0047	
Priority select register 8	INTC_PSR8	8-bit	Base + 0x0048	
Priority select register 9	INTC_PSR9	8-bit	Base + 0x0049	
Priority select register 10	INTC_PSR10	8-bit	Base + 0x004A	
Priority select register 11	INTC_PSR11	8-bit	Base + 0x004B	
Priority select register 12	INTC_PSR12	8-bit	Base + 0x004C	
Priority select register 13	INTC_PSR13	8-bit	Base + 0x004D	
Priority select register 14	INTC_PSR14	8-bit	Base + 0x004E	
Priority select register 15	INTC_PSR15	8-bit	Base + 0x004F	
Priority select register 16	INTC_PSR16	8-bit	Base + 0x0050	
Priority select register 17	INTC_PSR17	8-bit	Base + 0x0051	
Priority select register 18	INTC_PSR18	8-bit	Base + 0x0052	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
Priority select register 19	INTC_PSR19	8-bit	Base + 0x0053	
Priority select register 20	INTC_PSR20	8-bit	Base + 0x0054	
Priority select register 21	INTC_PSR21	8-bit	Base + 0x0055	
Priority select register 22	INTC_PSR22	8-bit	Base + 0x0056	
Priority select register 23	INTC_PSR23	8-bit	Base + 0x0057	
Priority select register 24	INTC_PSR24	8-bit	Base + 0x0058	
Priority select register 25	INTC_PSR25	8-bit	Base + 0x0059	
Priority select register 26	INTC_PSR26	8-bit	Base + 0x005A	
Priority select register 27	INTC_PSR27	8-bit	Base + 0x005B	
Priority select register 28	INTC_PSR28	8-bit	Base + 0x005C	
Priority select register 29	INTC_PSR29	8-bit	Base + 0x005D	
Priority select register 30	INTC_PSR30	8-bit	Base + 0x005E	
Priority select register 31	INTC_PSR31	8-bit	Base + 0x005F	
Priority select register 32	INTC_PSR32	8-bit	Base + 0x0060	
Priority select register 33	INTC_PSR33	8-bit	Base + 0x0061	
Priority select register 34	INTC_PSR34	8-bit	Base + 0x0062	
Priority select register 35	INTC_PSR35	8-bit	Base + 0x0063	
Priority select register 36	INTC_PSR36	8-bit	Base + 0x0064	
Priority select register 37	INTC_PSR37	8-bit	Base + 0x0065	
Priority select register 38	INTC_PSR38	8-bit	Base + 0x0066	
Priority select register 39	INTC_PSR39	8-bit	Base + 0x0067	
Priority select register 40	INTC_PSR40	8-bit	Base + 0x0068	
Priority select register 41	INTC_PSR41	8-bit	Base + 0x0069	
Priority select register 42	INTC_PSR42	8-bit	Base + 0x006A	
Priority select register 43	INTC_PSR43	8-bit	Base + 0x006B	
Priority select register 44	INTC_PSR44	8-bit	Base + 0x006C	
Priority select register 45	INTC_PSR45	8-bit	Base + 0x006D	
Priority select register 46	INTC_PSR46	8-bit	Base + 0x006E	
Priority select register 47	INTC_PSR47	8-bit	Base + 0x006F	
Priority select register 48	INTC_PSR48	8-bit	Base + 0x0070	
Priority select register 49	INTC_PSR49	8-bit	Base + 0x0071	
Priority select register 50	INTC_PSR50	8-bit	Base + 0x0072	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
Priority select register 51	INTC_PSR51	8-bit	Base + 0x0073	
Priority select register 52	INTC_PSR52	8-bit	Base + 0x0074	
Priority select register 53	INTC_PSR53	8-bit	Base + 0x0075	
Priority select register 54	INTC_PSR54	8-bit	Base + 0x0076	
Priority select register 55	INTC_PSR55	8-bit	Base + 0x0077	
Priority select register 56	INTC_PSR56	8-bit	Base + 0x0078	
Priority select register 57	INTC_PSR57	8-bit	Base + 0x0079	
Priority select register 58	INTC_PSR58	8-bit	Base + 0x007A	
Priority select register 59	INTC_PSR59	8-bit	Base + 0x007B	
Priority select register 60	INTC_PSR60	8-bit	Base + 0x007C	
Priority select register 61	INTC_PSR61	8-bit	Base + 0x007D	
Priority select register 62	INTC_PSR62	8-bit	Base + 0x007E	
Priority select register 63	INTC_PSR63	8-bit	Base + 0x007F	
Priority select register 64	INTC_PSR64	8-bit	Base + 0x0080	
Priority select register 65	INTC_PSR65	8-bit	Base + 0x0081	
Priority select register 66	INTC_PSR66	8-bit	Base + 0x0082	
Priority select register 67	INTC_PSR67	8-bit	Base + 0x0083	
Priority select register 68	INTC_PSR68	8-bit	Base + 0x0084	
Priority select register 69	INTC_PSR69	8-bit	Base + 0x0085	
Priority select register 70	INTC_PSR70	8-bit	Base + 0x0086	
Priority select register 71	INTC_PSR71	8-bit	Base + 0x0087	
Priority select register 72	INTC_PSR72	8-bit	Base + 0x0088	
Priority select register 73	INTC_PSR73	8-bit	Base + 0x0089	
Priority select register 74	INTC_PSR74	8-bit	Base + 0x008A	
Priority select register 75	INTC_PSR75	8-bit	Base + 0x008B	
Priority select register 76	INTC_PSR76	8-bit	Base + 0x008C	
Priority select register 77	INTC_PSR77	8-bit	Base + 0x008D	
Priority select register 78	INTC_PSR78	8-bit	Base + 0x008E	
Priority select register 79	INTC_PSR79	8-bit	Base + 0x008F	
Priority select register 80	INTC_PSR80	8-bit	Base + 0x0090	
Priority select register 81	INTC_PSR81	8-bit	Base + 0x0091	
Priority select register 82	INTC_PSR82	8-bit	Base + 0x0092	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
Priority select register 83	INTC_PSR83	8-bit	Base + 0x0093	
Priority select register 84	INTC_PSR84	8-bit	Base + 0x0094	
Priority select register 85	INTC_PSR85	8-bit	Base + 0x0095	
Priority select register 86	INTC_PSR86	8-bit	Base + 0x0096	
Priority select register 87	INTC_PSR87	8-bit	Base + 0x0097	
Priority select register 88	INTC_PSR88	8-bit	Base + 0x0098	
Priority select register 89	INTC_PSR89	8-bit	Base + 0x0099	
Priority select register 90	INTC_PSR90	8-bit	Base + 0x009A	
Priority select register 91	INTC_PSR91	8-bit	Base + 0x009B	
Priority select register 92	INTC_PSR92	8-bit	Base + 0x009C	
Priority select register 93	INTC_PSR93	8-bit	Base + 0x009D	
Priority select register 94	INTC_PSR94	8-bit	Base + 0x009E	
Priority select register 95	INTC_PSR95	8-bit	Base + 0x009F	
Priority select register 96	INTC_PSR96	8-bit	Base + 0x00A0	
Priority select register 97	INTC_PSR97	8-bit	Base + 0x00A1	
Priority select register 98	INTC_PSR98	8-bit	Base + 0x00A2	
Priority select register 99	INTC_PSR99	8-bit	Base + 0x00A3	
Priority select register 100	INTC_PSR100	8-bit	Base + 0x00A4	
Priority select register 101	INTC_PSR101	8-bit	Base + 0x00A5	
Priority select register 102	INTC_PSR102	8-bit	Base + 0x00A6	
Priority select register 103	INTC_PSR103	8-bit	Base + 0x00A7	
Priority select register 104	INTC_PSR104	8-bit	Base + 0x00A8	
Priority select register 105	INTC_PSR105	8-bit	Base + 0x00A9	
Priority select register 106	INTC_PSR106	8-bit	Base + 0x00AA	
Priority select register 107	INTC_PSR107	8-bit	Base + 0x00AB	
Priority select register 108	INTC_PSR108	8-bit	Base + 0x00AC	
Priority select register 109	INTC_PSR109	8-bit	Base + 0x00AD	
Priority select register 110	INTC_PSR110	8-bit	Base + 0x00AE	
Priority select register 111	INTC_PSR111	8-bit	Base + 0x00AF	
Priority select register 112	INTC_PSR112	8-bit	Base + 0x00B0	
Priority select register 113	INTC_PSR113	8-bit	Base + 0x00B1	
Priority select register 114	INTC_PSR114	8-bit	Base + 0x00B2	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
Priority select register 115	INTC_PSR115	8-bit	Base + 0x00B3	
Priority select register 116	INTC_PSR116	8-bit	Base + 0x00B4	
Priority select register 117	INTC_PSR117	8-bit	Base + 0x00B5	
Priority select register 118	INTC_PSR118	8-bit	Base + 0x00B6	
Priority select register 119	INTC_PSR119	8-bit	Base + 0x00B7	
Priority select register 120	INTC_PSR120	8-bit	Base + 0x00B8	
Priority select register 121	INTC_PSR121	8-bit	Base + 0x00B9	
Priority select register 122	INTC_PSR122	8-bit	Base + 0x00BA	
Priority select register 123	INTC_PSR123	8-bit	Base + 0x00BB	
Priority select register 124	INTC_PSR124	8-bit	Base + 0x00BC	
Priority select register 125	INTC_PSR125	8-bit	Base + 0x00BD	
Priority select register 126	INTC_PSR126	8-bit	Base + 0x00BE	
Priority select register 127	INTC_PSR127	8-bit	Base + 0x00BF	
Priority select register 128	INTC_PSR128	8-bit	Base + 0x00C0	
Priority select register 129	INTC_PSR129	8-bit	Base + 0x00C1	
Priority select register 130	INTC_PSR130	8-bit	Base + 0x00C2	
Priority select register 131	INTC_PSR131	8-bit	Base + 0x00C3	
Priority select register 132	INTC_PSR132	8-bit	Base + 0x00C4	
Priority select register 133	INTC_PSR133	8-bit	Base + 0x00C5	
Priority select register 134	INTC_PSR134	8-bit	Base + 0x00C6	
Priority select register 135	INTC_PSR135	8-bit	Base + 0x00C7	
Priority select register 136	INTC_PSR136	8-bit	Base + 0x00C8	
Priority select register 137	INTC_PSR137	8-bit	Base + 0x00C9	
Priority select register 138	INTC_PSR138	8-bit	Base + 0x00CA	
Priority select register 139	INTC_PSR139	8-bit	Base + 0x00CB	
Priority select register 140	INTC_PSR140	8-bit	Base + 0x00CC	
Priority select register 141	INTC_PSR141	8-bit	Base + 0x00CD	
Priority select register 142	INTC_PSR142	8-bit	Base + 0x00CE	
Priority select register 143	INTC_PSR143	8-bit	Base + 0x00CF	
Priority select register 144	INTC_PSR144	8-bit	Base + 0x00D0	
Priority select register 145	INTC_PSR145	8-bit	Base + 0x00D1	
Priority select register 146	INTC_PSR146	8-bit	Base + 0x00D2	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
Priority select register 147	INTC_PSR147	8-bit	Base + 0x00D3	
Priority select register 148	INTC_PSR148	8-bit	Base + 0x00D4	
Priority select register 149	INTC_PSR149	8-bit	Base + 0x00D5	
Priority select register 150	INTC_PSR150	8-bit	Base + 0x00D6	
Priority select register 151	INTC_PSR151	8-bit	Base + 0x00D7	
Priority select register 152	INTC_PSR152	8-bit	Base + 0x00D8	
Priority select register 153	INTC_PSR153	8-bit	Base + 0x00D9	
Priority select register 154	INTC_PSR154	8-bit	Base + 0x00DA	
Priority select register 155	INTC_PSR155	8-bit	Base + 0x00DB	
Priority select register 156	INTC_PSR156	8-bit	Base + 0x00DC	
Priority select register 157	INTC_PSR157	8-bit	Base + 0x00DD	
Priority select register 158	INTC_PSR158	8-bit	Base + 0x00DE	
Priority select register 159	INTC_PSR159	8-bit	Base + 0x00DF	
Priority select register 160	INTC_PSR160	8-bit	Base + 0x00E0	
Priority select register 161	INTC_PSR161	8-bit	Base + 0x00E1	
Priority select register 162	INTC_PSR162	8-bit	Base + 0x00E2	
Priority select register 163	INTC_PSR163	8-bit	Base + 0x00E3	
Priority select register 164	INTC_PSR164	8-bit	Base + 0x00E4	
Priority select register 165	INTC_PSR165	8-bit	Base + 0x00E5	
Priority select register 166	INTC_PSR166	8-bit	Base + 0x00E6	
Priority select register 167	INTC_PSR167	8-bit	Base + 0x00E7	
Priority select register 168	INTC_PSR168	8-bit	Base + 0x00E8	
Priority select register 169	INTC_PSR169	8-bit	Base + 0x00E9	
Priority select register 170	INTC_PSR170	8-bit	Base + 0x00EA	
Priority select register 171	INTC_PSR171	8-bit	Base + 0x00EB	
Priority select register 172	INTC_PSR172	8-bit	Base + 0x00EC	
Priority select register 173	INTC_PSR173	8-bit	Base + 0x00ED	
Priority select register 174	INTC_PSR174	8-bit	Base + 0x00EE	
Priority select register 175	INTC_PSR175	8-bit	Base + 0x00EF	
Priority select register 176	INTC_PSR176	8-bit	Base + 0x00F0	
Priority select register 177	INTC_PSR177	8-bit	Base + 0x00F1	
Priority select register 178	INTC_PSR178	8-bit	Base + 0x00F2	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
Priority select register 179	INTC_PSR179	8-bit	Base + 0x00F3	
Priority select register 180	INTC_PSR180	8-bit	Base + 0x00F4	
Priority select register 181	INTC_PSR181	8-bit	Base + 0x00F5	
Priority select register 182	INTC_PSR182	8-bit	Base + 0x00F6	
Priority select register 183	INTC_PSR183	8-bit	Base + 0x00F7	
Priority select register 184	INTC_PSR184	8-bit	Base + 0x00F8	
Priority select register 185	INTC_PSR185	8-bit	Base + 0x00F9	
Priority select register 186	INTC_PSR186	8-bit	Base + 0x00FA	
Priority select register 187	INTC_PSR187	8-bit	Base + 0x00FB	
Priority select register 188	INTC_PSR188	8-bit	Base + 0x00FC	
Priority select register 189	INTC_PSR189	8-bit	Base + 0x00FD	
Priority select register 190	INTC_PSR190	8-bit	Base + 0x00FE	
Priority select register 191	INTC_PSR191	8-bit	Base + 0x00FF	
Priority select register 192	INTC_PSR192	8-bit	Base + 0x0100	
Priority select register 193	INTC_PSR193	8-bit	Base + 0x0101	
Priority select register 194	INTC_PSR194	8-bit	Base + 0x0102	
Priority select register 195	INTC_PSR195	8-bit	Base + 0x0103	
Priority select register 196	INTC_PSR196	8-bit	Base + 0x0104	
Priority select register 197	INTC_PSR197	8-bit	Base + 0x0105	
Priority select register 198	INTC_PSR198	8-bit	Base + 0x0106	
Priority select register 199	INTC_PSR199	8-bit	Base + 0x0107	
Priority select register 200	INTC_PSR200	8-bit	Base + 0x0108	
Priority select register 201	INTC_PSR201	8-bit	Base + 0x0109	
Priority select register 202	INTC_PSR202	8-bit	Base + 0x010A	
Priority select register 203	INTC_PSR203	8-bit	Base + 0x010B	
Priority select register 204	INTC_PSR204	8-bit	Base + 0x010C	
Priority select register 205	INTC_PSR205	8-bit	Base + 0x010D	
Priority select register 206	INTC_PSR206	8-bit	Base + 0x010E	
Priority select register 207	INTC_PSR207	8-bit	Base + 0x010F	
Priority select register 208	INTC_PSR208	8-bit	Base + 0x0110	
Priority select register 209	INTC_PSR209	8-bit	Base + 0x0111	
Priority select register 210	INTC_PSR210	8-bit	Base + 0x0112	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
Priority select register 211	INTC_PSR211	8-bit	Base + 0x0113	
Priority select register 212	INTC_PSR212	8-bit	Base + 0x0114	
Priority select register 213	INTC_PSR213	8-bit	Base + 0x0115	
Priority select register 214	INTC_PSR214	8-bit	Base + 0x0116	
Priority select register 215	INTC_PSR215	8-bit	Base + 0x0117	
Priority select register 216	INTC_PSR216	8-bit	Base + 0x0118	
Priority select register 217	INTC_PSR217	8-bit	Base + 0x0119	
Priority select register 218	INTC_PSR218	8-bit	Base + 0x011A	
Priority select register 219	INTC_PSR219	8-bit	Base + 0x011B	
Priority select register 220	INTC_PSR220	8-bit	Base + 0x011C	
Priority select register 221	INTC_PSR221	8-bit	Base + 0x011D	
Priority select register 222	INTC_PSR222	8-bit	Base + 0x011E	
Priority select register 223	INTC_PSR223	8-bit	Base + 0x011F	
Priority select register 224	INTC_PSR224	8-bit	Base + 0x0120	
Priority select register 225	INTC_PSR225	8-bit	Base + 0x0121	
Priority select register 226	INTC_PSR226	8-bit	Base + 0x0122	
Priority select register 227	INTC_PSR227	8-bit	Base + 0x0123	
Priority select register 228	INTC_PSR228	8-bit	Base + 0x0124	
Priority select register 229	INTC_PSR229	8-bit	Base + 0x0125	
Priority select register 230	INTC_PSR230	8-bit	Base + 0x0126	
Priority select register 231	INTC_PSR231	8-bit	Base + 0x0127	
Priority select register 232	INTC_PSR232	8-bit	Base + 0x0128	
Priority select register 233	INTC_PSR233	8-bit	Base + 0x0129	
Priority select register 234	INTC_PSR234	8-bit	Base + 0x012A	
Priority select register 235	INTC_PSR234	8-bit	Base + 0x012B	
Priority select register 236	INTC_PSR236	8-bit	Base + 0x012C	
Priority select register 237	INTC_PSR237	8-bit	Base + 0x012D	
Priority select register 238	INTC_PSR238	8-bit	Base + 0x012E	
Priority select register 239	INTC_PSR239	8-bit	Base + 0x012F	
Priority select register 240	INTC_PSR240	8-bit	Base + 0x0130	
Priority select register 241	INTC_PSR241	8-bit	Base + 0x0131	
Priority select register 242	INTC_PSR242	8-bit	Base + 0x0132	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
Priority select register 243	INTC_PSR243	8-bit	Base + 0x0133	
Priority select register 244	INTC_PSR244	8-bit	Base + 0x0134	
Priority select register 245	INTC_PSR245	8-bit	Base + 0x0135	
Priority select register 246	INTC_PSR246	8-bit	Base + 0x0136	
Priority select register 247	INTC_PSR247	8-bit	Base + 0x0137	
Priority select register 248	INTC_PSR248	8-bit	Base + 0x0138	
Priority select register 249	INTC_PSR249	8-bit	Base + 0x0139	
Priority select register 250	INTC_PSR250	8-bit	Base + 0x013A	
Priority select register 251	INTC_PSR251	8-bit	Base + 0x013B	
Priority select register 252	INTC_PSR252	8-bit	Base + 0x013C	
Priority select register 253	INTC_PSR253	8-bit	Base + 0x013D	
Priority select register 254	INTC_PSR254	8-bit	Base + 0x013E	
Priority select register 255	INTC_PSR255	8-bit	Base + 0x013F	
Priority select register 256	INTC_PSR256	8-bit	Base + 0x0140	
Priority select register 257	INTC_PSR257	8-bit	Base + 0x0141	
Priority select register 258	INTC_PSR258	8-bit	Base + 0x0142	
Priority select register 259	INTC_PSR259	8-bit	Base + 0x0143	
Priority select register 260	INTC_PSR260	8-bit	Base + 0x0144	
Priority select register 261	INTC_PSR261	8-bit	Base + 0x0145	
Priority select register 262	INTC_PSR262	8-bit	Base + 0x0146	
Priority select register 263	INTC_PSR263	8-bit	Base + 0x0147	
Priority select register 264	INTC_PSR264	8-bit	Base + 0x0148	
Priority select register 265	INTC_PSR265	8-bit	Base + 0x0149	
Priority select register 266	INTC_PSR266	8-bit	Base + 0x014A	
Priority select register 267	INTC_PSR267	8-bit	Base + 0x014B	
Priority select register 268	INTC_PSR268	8-bit	Base + 0x014C	
Priority select register 269	INTC_PSR269	8-bit	Base + 0x014D	
Priority select register 270	INTC_PSR270	8-bit	Base + 0x014E	
Priority select register 271	INTC_PSR271	8-bit	Base + 0x014F	
Priority select register 272	INTC_PSR272	8-bit	Base + 0x0150	
Priority select register 273	INTC_PSR273	8-bit	Base + 0x0151	
Priority select register 274	INTC_PSR274	8-bit	Base + 0x0152	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
Priority select register 275	INTC_PSR275	8-bit	Base + 0x0153	
Priority select register 276	INTC_PSR276	8-bit	Base + 0x0154	
Priority select register 277	INTC_PSR277	8-bit	Base + 0x0155	
Priority select register 278	INTC_PSR278	8-bit	Base + 0x0156	
Priority select register 279	INTC_PSR279	8-bit	Base + 0x0157	
Priority select register 280	INTC_PSR280	8-bit	Base + 0x0158	
Priority select register 281	INTC_PSR281	8-bit	Base + 0x0159	
Priority select register 282	INTC_PSR282	8-bit	Base + 0x015A	
Priority select register 283	INTC_PSR283	8-bit	Base + 0x015B	
Priority select register 284	INTC_PSR284	8-bit	Base + 0x015C	
Priority select register 285	INTC_PSR285	8-bit	Base + 0x015D	
Priority select register 286	INTC_PSR286	8-bit	Base + 0x015E	
Priority select register 287	INTC_PSR287	8-bit	Base + 0x015F	
Priority select register 288	INTC_PSR288	8-bit	Base + 0x0160	
Priority select register 289	INTC_PSR289	8-bit	Base + 0x0161	
Priority select register 290	INTC_PSR290	8-bit	Base + 0x0162	
Priority select register 291	INTC_PSR291	8-bit	Base + 0x0163	
Priority select register 292	INTC_PSR292	8-bit	Base + 0x0164	
Priority select register 293	INTC_PSR293	8-bit	Base + 0x0165	
Priority select register 294	INTC_PSR294	8-bit	Base + 0x0166	
Priority select register 295	INTC_PSR295	8-bit	Base + 0x0167	
Priority select register 296	INTC_PSR296	8-bit	Base + 0x0168	
Priority select register 297	INTC_PSR297	8-bit	Base + 0x0169	
Priority select register 298	INTC_PSR298	8-bit	Base + 0x016A	
Priority select register 299	INTC_PSR299	8-bit	Base + 0x016B	
Priority select register 300	INTC_PSR300	8-bit	Base + 0x016C	
Priority select register 301	INTC_PSR301	8-bit	Base + 0x016D	
Priority select register 302	INTC_PSR302	8-bit	Base + 0x016E	
Priority select register 303	INTC_PSR303	8-bit	Base + 0x016F	
Priority select register 304	INTC_PSR304	8-bit	Base + 0x0170	
Priority select register 305	INTC_PSR305	8-bit	Base + 0x0171	
Priority select register 306	INTC_PSR306	8-bit	Base + 0x0172	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
Priority select register 307	INTC_PSR307	8-bit	Base + 0x0173	
Fast Ethernet Controller (FEC)			0xFFF4_C000	Chapter 14 “Fast Ethernet Controller (FEC)”
Interrupt Event Register	EIR	32-bit	Base + 0x0004	
Interrupt Mask Register	EIMR	32-bit	Base + 0x0008	
Receive Descriptor Active Register	RDAR	32-bit	Base + 0x0010	
Transmit Descriptor Active Register	TDAR	32-bit	Base + 0x0014	
Ethernet Control Register	ECR	32-bit	Base + 0x0024	
MII Management Frame Regsiter	MMFR	32-bit	Base + 0x0040	
MII Speed Control Register	MSCR	32-bit	Base + 0x0044	
MIB Control/Status Register	MIBC	32-bit	Base + 0x0064	
Receive Control Register	RCR	32-bit	Base + 0x0084	
Transmit Control Register	TCR	32-bit	Base + 0x00C4	
MAC Address Low Register	PALR	32-bit	Base + 0x00E4	
MAC Address Upper Register + Type Field	PAUR	32-bit	Base + 0x00E8	
Opcode + Pause Duration	OPD	32-bit	Base + 0x00EC	
Upper 32 bits of Individual Hash Table	IAUR	32-bit	Base + 0x0118	
Lower 32 Bits of Individual Hash Table	IALR	32-bit	Base + 0x011C	
Upper 32 bits of Group Hash Table	GAUR	32-bit	Base + 0x0120	
Lower 32 bits of Group Hash Table	GALR	32-bit	Base + 0x0124	
Transmit FIFO Watermark	TFWR	32-bit	Base + 0x0144	
FIFO Receive Bound Register	FRBR	32-bit	Base + 0x014C	
FIFO Receive FIFO Start Registers	FRSR	32-bit	Base + 0x0150	
Pointer to Receive Descriptor Ring	ERDSR	32-bit	Base + 0x0180	
Pointer to Transmit Descriptor Ring	ETDSR	32-bit	Base + 0x0184	
Maximum Receive Buffer Size	EMRBR	32-bit	Base + 0x0188	
MIB Block Counters	MIB		FFF4_C200	
Reserved	—	—	Base + (0xFFF0__8000)	
Reserved	—	—	Base + (0xFFF1_0000)	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
Enhanced Queued Analog-to-Digital Converter (eQADC)			0xFFF8_0000	Chapter 19, "Enhanced Queued Analog-to-Digital Converter (eQADC)"
Module configuration register	EQADC_MCR	32-bit	Base + 0x0000	
Reserved	—	—	Base + (0x0004-0x0007)	
Null message send format register	EQADC_NMSFR	32-bit	Base + 0x0008	
External trigger digital filter register	EQADC_ETDFR	32-bit	Base + 0x000C	
CFIFO push register 0	EQADC_CFPR0	32-bit	Base +0x0010	
CFIFO push register 1	EQADC_CFPR1	32-bit	Base +0x0014	
CFIFO push register 2	EQADC_CFPR2	32-bit	Base +0x0018	
CFIFO push register 3	EQADC_CFPR3	32-bit	Base +0x001C	
CFIFO push register 4	EQADC_CFPR4	32-bit	Base +0x0020	
CFIFO push register 5	EQADC_CFPR5	32-bit	Base +0x0024	
Reserved	—	—	Base + (0x0028-0x002F)	
Result FIFO pop register 0	EQADC_RFPR0	32-bit	Base + 0x0030	
Result FIFO pop register 1	EQADC_RFPR1	32-bit	Base + 0x0034	
Result FIFO pop register 2	EQADC_RFPR2	32-bit	Base + 0x0038	
Result FIFO pop register 3	EQADC_RFPR3	32-bit	Base + 0x003C	
Result FIFO pop register 4	EQADC_RFPR4	32-bit	Base + 0x0040	
Result FIFO pop register 5	EQADC_RFPR5	32-bit	Base + 0x0044	
Reserved	—	—	Base + (0x0048-0x004F)	
CFIFO control register 0	EQADC_CFCR0	16-bit	Base + 0x0050	
CFIFO control register 1	EQADC_CFCR1	16-bit	Base + 0x0052	
CFIFO control register 2	EQADC_CFCR2	16-bit	Base + 0x0054	
CFIFO control register 3	EQADC_CFCR3	16-bit	Base + 0x0056	
CFIFO control register 4	EQADC_CFCR4	16-bit	Base + 0x0058	
CFIFO control register 5	EQADC_CFCR5	16-bit	Base + 0x005A	
Reserved	—	—	Base + (0x005C-0x005F)	
Interrupt and DMA control register 0	EQADC_IDCR0	16-bit	Base + 0x0060	
Interrupt and DMA control register 1	EQADC_IDCR1	16-bit	Base + 0x0062	
Interrupt and DMA control register 2	EQADC_IDCR2	16-bit	Base + 0x0064	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
Interrupt and DMA control register 3	EQADC_IDCR3	16-bit	Base + 0x0066	
Interrupt and DMA control register 4	EQADC_IDCR4	16-bit	Base + 0x0068	
Interrupt and DMA control register 5	EQADC_IDCR5	16-bit	Base + 0x006A	
Reserved	—	—	Base + (0x006C-0x006F)	
FIFO and interrupt status register 0	EQADC_FISR0	32-bit	Base + 0x0070	
FIFO and interrupt status register 1	EQADC_FISR1	32-bit	Base + 0x0074	
FIFO and interrupt status register 2	EQADC_FISR2	32-bit	Base + 0x0078	
FIFO and interrupt status register 3	EQADC_FISR3	32-bit	Base + 0x007C	
FIFO and interrupt status register 4	EQADC_FISR4	32-bit	Base + 0x0080	
FIFO and interrupt status register 5	EQADC_FISR5	32-bit	Base + 0x0084	
Reserved	—	—	Base + (0x0088-0x008F)	
CFIFO transfer counter register 0	EQADC_CFTCR0	16-bit	Base + 0x0090	
CFIFO transfer counter register 1	EQADC_CFTCR1	16-bit	Base + 0x0092	
CFIFO transfer counter register 2	EQADC_CFTCR2	16-bit	Base + 0x0094	
CFIFO transfer counter register 3	EQADC_CFTCR3	16-bit	Base + 0x0096	
CFIFO transfer counter register 4	EQADC_CFTCR4	16-bit	Base + 0x0098	
CFIFO transfer counter register 5	EQADC_CFTCR5	16-bit	Base + 0x009A	
Reserved	—	—	Base + (0x009C-0x009F)	
CFIFO status snapshot register 0	EQADC_CFSSR0	32-bit	Base + 0x00A0	
CFIFO status snapshot register 1	EQADC_CFSSR1	32-bit	Base + 0x00A4	
CFIFO status snapshot register 2	EQADC_CFSSR2	32-bit	Base + 0x00A8	
CFIFO status register	EQADC_CFSR	32-bit	Base + 0x00AC	
Reserved	—	—	Base + (0x00B0-0x00B3	
SSI control register	EQADC_SSICR	32-bit	Base + 0x00B4	
SSI receive data register	EQADC_SSIRDR	32-bit	Base + 0x00B8	
Reserved	—	—	Base + (0x00BC-0x00FF)	
CFIFO 0 register 0	EQADC_CF0R0	32-bit	Base + 0x0100	
CFIFO 0 register 1	EQADC_CF0R1	32-bit	Base + 0x0104	
CFIFO 0 register 2	EQADC_CF0R2	32-bit	Base + 0x0108	
CFIFO 0 register 3	EQADC_CF0R3	32-bit	Base + 0x010C	
Reserved	—	—	Base + (0x0110-0x013F)	
CFIFO 1 register 0	EQADC_CF1R0	32-bit	Base + 0x0140	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
CFIFO 1 register 1	EQADC_CF1R1	32-bit	Base + 0x0144	
CFIFO 1 register 2	EQADC_CF1R2	32-bit	Base + 0x0148	
CFIFO 1 register 3	EQADC_CF1R3	32-bit	Base + 0x014C	
Reserved	—	—	Base + (0x0150-0x017F)	
CFIFO 2 register 0	EQADC_CF2R0	32-bit	Base + 0x0180	
CFIFO 2 register 1	EQADC_CF2R1	32-bit	Base + 0x0184	
CFIFO 2 register 2	EQADC_CF2R2	32-bit	Base + 0x0188	
CFIFO 2 register 3	EQADC_CF2R3	32-bit	Base + 0x018C	
Reserved	—	—	Base + (0x0190-0x01BF)	
CFIFO 3 register 0	EQADC_CF3R0	32-bit	Base + 0x01C0	
CFIFO 3 register 1	EQADC_CF3R1	32-bit	Base + 0x01C4	
CFIFO 3 register 2	EQADC_CF3R2	32-bit	Base + 0x01C8	
CFIFO 3 register 3	EQADC_CF3R3	32-bit	Base + 0x01CC	
Reserved	—	—	Base + (0x01D0-0x01FF)	
CFIFO 4 register 0	EQADC_CF4R0	32-bit	Base + 0x0200	
CFIFO 4 register 1	EQADC_CF4R1	32-bit	Base + 0x0204	
CFIFO 4 register 2	EQADC_CF4R2	32-bit	Base + 0x0208	
CFIFO 4 register 3	EQADC_CF4R3	32-bit	Base + 0x020C	
Reserved	—	—	Base + (0x0210-0x023F)	
CFIFO 5 register 0	EQADC_CF5R0	32-bit	Base + 0x0240	
CFIFO 5 register 1	EQADC_CF5R1	32-bit	Base + 0x0244	
CFIFO 5 register 2	EQADC_CF5R2	32-bit	Base + 0x0248	
CFIFO 5 register 3	EQADC_CF5R3	32-bit	Base + 0x024C	
Reserved	—	—	Base + (0x0250-0x02FF)	
RFIFO 0 register 0	EQADC_RF0R0	32-bit	Base + 0x0300	
RFIFO 0 register 1	EQADC_RF0R1	32-bit	Base + 0x0304	
RFIFO 0 register 2	EQADC_RF0R2	32-bit	Base + 0x0308	
RFIFO 0 register 3	EQADC_RF0R3	32-bit	Base + 0x030C	
Reserved	—	—	Base + (0x0310-0x033F)	
RFIFO 1 register 0	EQADC_RF1R0	32-bit	Base + 0x0340	
RFIFO 1 register 1	EQADC_RF1R1	32-bit	Base + 0x0344	
RFIFO 1 register 2	EQADC_RF1R2	32-bit	Base + 0x0348	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
RFIFO 1 register 3	EQADC_RF1R3	32-bit	Base + 0x034C	
Reserved	—	—	Base + (0x0350-0x037F)	
RFIFO 2 register 0	EQADC_RF2R0	32-bit	Base + 0x0380	
RFIFO 2 register 1	EQADC_RF2R1	32-bit	Base + 0x0384	
RFIFO 2 register 2	EQADC_RF2R2	32-bit	Base + 0x0388	
RFIFO 2 register 3	EQADC_RF2R3	32-bit	Base + 0x038C	
Reserved	—	—	Base + (0x0390-0x03BF)	
RFIFO 3 register 0	EQADC_RF3R0	32-bit	Base + 0x03C0	
RFIFO 3 register 1	EQADC_RF3R1	32-bit	Base + 0x03C4	
RFIFO 3 register 2	EQADC_RF3R2	32-bit	Base + 0x03C8	
RFIFO 3 register 3	EQADC_RF3R3	32-bit	Base + 0x03CC	
Reserved	—	—	Base + (0x03D0-0x03FF)	
RFIFO 4 register 0	EQADC_RF4R0	32-bit	Base + 0x0400	
RFIFO 4 register 1	EQADC_RF4R1	32-bit	Base + 0x0404	
RFIFO 4 register 2	EQADC_RF4R2	32-bit	Base + 0x0408	
RFIFO 4 register 3	EQADC_RF4R3	32-bit	Base + 0x040C	
Reserved	—	—	Base + (0x0410-0x043F)	
RFIFO 5 register 0	EQADC_RF5R0	32-bit	Base + 0x0440	
RFIFO 5 register 1	EQADC_RF5R1	32-bit	Base + 0x0444	
RFIFO 5 register 2	EQADC_RF5R2	32-bit	Base + 0x0448	
RFIFO 5 register 3	EQADC_RF5R3	32-bit	Base + 0x044C	
Reserved	—	—	Base + (0x0450-0x07FF)	
ADC0 control register	ADC0_CR		No memory mapped access	
ADC1 control register	ADC1_CR			
ADC time stamp control register	ADC_TSCR			
ADC time base counter register	ADC_TBCR			
ADC0 gain calibration constant register	ADC0_GCCR			
ADC1 gain calibration constant register	ADC1_GCCR			
ADC0 offset calibration constant register	ADC0_OCCR			
ADC1 offset calibration constant register	ADC1_OCCR			
Reserved	—	—	(Base + 0x0800)-0xFFF8_FFFF	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
Deserial / Serial Peripheral Interface (DSPIx)			0xFFF9_0000 (DSPI A) [2] 0xFFF9_4000 (DSPI B) 0xFFF9_8000 (DSPI C) 0xFFF9_C000 (DSPI D)	20.1, "Introduction"
Module configuration register	DSPIx_MCR	32-bit	Base + 0x0000	
Reserved	—	—	Base + (0x0004-0x0007)	
Transfer count register	DSPIx_TCR	32-bit	Base + 0x0008	
Clock and transfer attribute register 0	DSPIx_CTAR0	32-bit	Base + 0x000C	
Clock and transfer attribute register 1	DSPIx_CTAR1	32-bit	Base + 0x0010	
Clock and transfer attribute register 2	DSPIx_CTAR2	32-bit	Base + 0x0014	
Clock and transfer attribute register 3	DSPIx_CTAR3	32-bit	Base + 0x0018	
Clock and transfer attribute register 4	DSPIx_CTAR4	32-bit	Base + 0x001C	
Clock and transfer attribute register 5	DSPIx_CTAR5	32-bit	Base + 0x0020	
Clock and transfer attribute register 6	DSPIx_CTAR6	32-bit	Base + 0x0024	
Clock and transfer attribute register 7	DSPIx_CTAR7	32-bit	Base + 0x0028	
Status register	DSPIx_SR	32-bit	Base + 0x002C	
DMA/interrupt request select and enable register	DSPIx_RSER	32-bit	Base + 0x0030	
Push TX FIFO register	DSPIx_PUSHR	32-bit	Base + 0x0034	
Pop RX FIFO register	DSPIx_POPR	32-bit	Base + 0x0038	
Transmit FIFO registers 0	DSPIx_TXFR0	32-bit	Base + 0x003C	
Transmit FIFO registers 1	DSPIx_TXFR1	32-bit	Base + 0x0040	
Transmit FIFO registers 2	DSPIx_TXFR2	32-bit	Base + 0x0044	
Transmit FIFO registers 3	DSPIx_TXFR3	32-bit	Base + 0x0048	
Reserved	—	—	Base + (0x004C-0x007B)	
Receive FIFO registers 0	DSPIx_RXFR0	32-bit	Base + 0x007C	
Receive FIFO registers 1	DSPIx_RXFR1	32-bit	Base + 0x0080	
Receive FIFO registers 2	DSPIx_RXFR2	32-bit	Base + 0x0084	
Receive FIFO registers 3	DSPIx_RXFR3	32-bit	Base + 0x0088	
Reserved	—	—	Base + (0x008C-0x00BB)	
DSI configuration register	DSPIx_DSICR	32-bit	Base + 0x00BC	
DSI serialization data register	DSPIx_SDR	32-bit	Base + 0x00C0	
DSI alternate serialization data register	DSPIx_ASDR	32-bit	Base + 0x00C4	
DSI transmit comparison register	DSPIx_COMPR	32-bit	Base + 0x00C8	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
DSI deserialization data register	DSPIx_DDR	32-bit	Base + 0x00CC	
Reserved	—	—	(Base +0x00D0)-(0xFFF9_3FFF) (A) (0xFFF9_7FFF) (B) (0xFFF9_BFFF) (C) (0xFFFA_FFFF) (D)	
Enhanced Serial Communication Interface (eSCIx)			0xFFFB_0000 (A) 0xFFFB_4000 (B)	Chapter 21, "Enhanced Serial Communication Interface (eSCI)"
Control register 1	ESCIx_CR1	32-bit	Base + 0x0000	
Control register 2	ESCIx_CR2	16-bit	Base + 0x0004	
Data register	ESCIx_DR	16-bit	Base + 0x0006	
Status register	ESCIx_SR	32-bit	Base + 0x0008	
LIN control register	ESCIx_LCR	32-bit	Base + 0x000C	
LIN transmit register	ESCIx_LTR	32-bit	Base + 0x0010	
LIN receive register	ESCIx_LRR	32-bit	Base + 0x0014	
LIN CRC polynomial register	ESCIx_LPR	32-bit	Base + 0x0018	
Reserved	—	—	(Base +0x001C)-(0xFFFB_3FFF) (A) (0xFFFB_7FFF) (B)	
FlexCAN2 Controller Area Network (CANx)			0xFFFC_0000 (FlexCAN A) 0xFFFC_4000 (FlexCAN B)[2] 0xFFFC_8000 (FlexCAN C)	Chapter 22, "FlexCAN2 Controller Area Network"
Module configuration register	CANx_MCR	32-bit	Base + 0x0000	
Control register	CANx_CR	32-bit	Base + 0x0004	
Free running timer register	CANx_TIMER	32-bit	Base + 0x0008	
Reserved	—	—	Base + (0x000C-0x000F)	
Receive global mask register	CANx_RXGMASK	32-bit	Base + 0x0010	
Receive buffer 14 mask register	CANx_RX14MASK	32-bit	Base + 0x0014	
Receive buffer 15 mask register	CANx_RX15MASK	32-bit	Base + 0x0018	
Error counter register	CANx_ECR	32-bit	Base + 0x001C	
Error and status register	CANx_ESR	32-bit	Base + 0x0020	
Interrupt mask register high	CANx_IMRH	32-bit	Base + 0x0024	
Interrupt mask register low	CANx_IMRL	32-bit	Base + 0x0028	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
Interrupt flag register high	CANx_IFRH	32-bit	Base + 0x002C	
Interrupt flag register low	CANx_IFRL	32-bit	Base + 0x0030	
Reserved	—	—	Base + (0x0034-0x007F)	
Boot Assist Module (BAM)			0xFFFF_C000	Chapter 16, "Boot Assist Module (BAM)"
Message buffer 0	MB0	16-bit	Base + 0x0080	
Message buffer 1	MB1	16-bit	Base + 0x0090	
Message buffer 2	MB2	16-bit	Base + 0x00A0	
Message buffer 3	MB3	16-bit	Base + 0x00B0	
Message buffer 4	MB4	16-bit	Base + 0x00C0	
Message buffer 5	MB5	16-bit	Base + 0x00D0	
Message buffer 6	MB6	16-bit	Base + 0x00E0	
Message buffer 7	MB7	16-bit	Base + 0x00F0	
Message buffer 8	MB8	16-bit	Base + 0x0100	
Message buffer 9	MB9	16-bit	Base + 0x0110	
Message buffer 10	MB10	16-bit	Base + 0x0120	
Message buffer 11	MB11	16-bit	Base + 0x0130	
Message buffer 12	MB12	16-bit	Base + 0x0140	
Message buffer 13	MB13	16-bit	Base + 0x0150	
Message buffer 14	MB14	16-bit	Base + 0x0160	
Message buffer 15	MB15	16-bit	Base + 0x0170	
Message buffer 16	MB16	16-bit	Base + 0x0180	
Message buffer 17	MB17	16-bit	Base + 0x0190	
Message buffer 18	MB18	16-bit	Base + 0x01A0	
Message buffer 19	MB19	16-bit	Base + 0x01B0	
Message buffer 20	MB20	16-bit	Base + 0x01C0	
Message buffer 21	MB21	16-bit	Base + 0x01D0	
Message buffer 22	MB22	16-bit	Base + 0x01E0	
Message buffer 23	MB23	16-bit	Base + 0x01F0	
Message buffer 24	MB24	16-bit	Base + 0x0200	
Message buffer 25	MB25	16-bit	Base + 0x0210	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
Message buffer 26	MB26	16-bit	Base + 0x0220	
Message buffer 27	MB27	16-bit	Base + 0x0230	
Message buffer 28	MB28	16-bit	Base + 0x0240	
Message buffer 29	MB29	16-bit	Base + 0x0250	
Message buffer 30	MB30	16-bit	Base + 0x0260	
Message buffer 31	MB31	16-bit	Base + 0x0270	
Message buffer 32	MB32	16-bit	Base + 0x0280	
Message buffer 33	MB33	16-bit	Base + 0x0290	
Message buffer 34	MB34	16-bit	Base + 0x02A0	
Message buffer 35	MB35	16-bit	Base + 0x02B0	
Message buffer 36	MB36	16-bit	Base + 0x02C0	
Message buffer 37	MB37	16-bit	Base + 0x02D0	
Message buffer 38	MB38	16-bit	Base + 0x02E0	
Message buffer 39	MB39	16-bit	Base + 0x02F0	
Message buffer 40	MB40	16-bit	Base + 0x0300	
Message buffer 41	MB41	16-bit	Base + 0x0310	
Message buffer 42	MB42	16-bit	Base + 0x0320	
Message buffer 43	MB43	16-bit	Base + 0x0330	
Message buffer 44	MB44	16-bit	Base + 0x0340	
Message buffer 45	MB45	16-bit	Base + 0x0350	
Message buffer 46	MB46	16-bit	Base + 0x0360	
Message buffer 47	MB47	16-bit	Base + 0x0370	
Message buffer 48	MB48	16-bit	Base + 0x0380	
Message buffer 49	MB49	16-bit	Base + 0x0390	
Message buffer 50	MB50	16-bit	Base + 0x03A0	
Message buffer 51	MB51	16-bit	Base + 0x03B0	
Message buffer 52	MB52	16-bit	Base + 0x03C0	
Message buffer 53	MB53	16-bit	Base + 0x03D0	
Message buffer 54	MB54	16-bit	Base + 0x03E0	
Message buffer 55	MB55	16-bit	Base + 0x03F0	
Message buffer 56	MB56	16-bit	Base + 0x0400	
Message buffer 57	MB57	16-bit	Base + 0x0410	

Table A-2. MPC5554 / MPC5553 Detailed Register Map (continued)

Register Description	Register Name	Used Size	Address	Reference
Message buffer 58	MB58	16-bit	Base + 0x0420	
Message buffer 59	MB59	16-bit	Base + 0x0430	
Message buffer 60	MB60	16-bit	Base + 0x0440	
Message buffer 61	MB61	16-bit	Base + 0x0450	
Message buffer 62	MB62	16-bit	Base + 0x0460	
Message buffer 63	MB63	16-bit	Base + 0x0470	
Reserved	—	—	(Base + 0x0480)- 0xFFFC_3FFF (A) 0xFFFC_7FFF (B) 0xFFFF_FFFF (C)	—

1 The registers mapped in the ECSM module (0xFFF4_0014-0xFFF4_001F) provide control and configuration for a software watchdog timer, and are included as part of a standard Freescale ECSM block incorporated in the MPC5554. The eSys e200z6 core also provides this functionality and is the preferred method for watchdog implementation. In order to optimize code portability to other members of the eSys MPU family, use of the watchdog registers in the ECSM is not recommended.

2 MPC5554 Only

Table A-3. e200z6 Core SPR Numbers (Supervisor Mode)

Register	Description	SPR (decimal)
General Registers		
XER	Integer Exception Register	1
LR	Link Register	8
CTR	Count Register	9
GPR0–GPR31	General Purpose Registers	N/A
Special Purpose Registers		
SPRG0	Special Purpose Register 0	272
SPRG1	Special Purpose Register 1	273
SPRG2	Special Purpose Register 2	274
SPRG3	Special Purpose Register 3	275
SPRG4	Special Purpose Register 4	276
SPRG5	Special Purpose Register 5	277
SPRG6	Special Purpose Register 6	278
SPRG7	Special Purpose Register 7	279
USPRG0	User Special Purpose Register	256
BUCSR	Branch Unit Control and Status Register	1013
Exception Handling/Control Registers		

Table A-3. e200z6 Core SPR Numbers (Supervisor Mode) (continued)

Register	Description	SPR (decimal)
SRR0	Save and Restore Register 0	26
SRR1	Save and Restore Register 1	27
CSRR0	Critical Save and Restore Register 0	58
CSRR1	Critical Save and Restore Register 1	59
DSRR0	Debug Save and Restore Register 0	574
DSRR1	Debug Save and Restore Register 1	575
ESR	Exception Syndrome Register	62
MCSR	Machine Check Syndrome Register	572
DEAR	Data Exception Address Register	61
IVPR	Interrupt Vector Prefix Register	63
IVOR1	Interrupt Vector Offset Register 1	401
IVOR2	Interrupt Vector Offset Register 2	402
IVOR3	Interrupt Vector Offset Register 3	403
IVOR4	Interrupt Vector Offset Register 4	404
IVOR5	Interrupt Vector Offset Register 5	405
IVOR6	Interrupt Vector Offset Register 6	406
IVOR7	Interrupt Vector Offset Register 7	407
IVOR8	Interrupt Vector Offset Register 8	408
IVOR9	Not Supported	—
IVOR10	Interrupt Vector Offset Register 10	410
IVOR11	Interrupt Vector Offset Register 11	411
IVOR12	Interrupt Vector Offset Register 12	412
IVOR13	Interrupt Vector Offset Register 13	413
IVOR14	Interrupt Vector Offset Register 14	414
IVOR15	Interrupt Vector Offset Register 15	415
IVOR32	Interrupt Vector Offset Register 32	528
IVOR33	Interrupt Vector Offset Register 33	529
IVOR34	Interrupt Vector Offset Register 34	530
Processor Control Registers		
MSR	Machine State Register	N/A
PVR	Processor Version Register	287
PIR	Processor ID Register	286
SVR	System Version Register	1023

Table A-3. e200z6 Core SPR Numbers (Supervisor Mode) (continued)

Register	Description	SPR (decimal)
HID0	Hardware Implementation Dependent Register 0	1008
HID1	Hardware Implementation Dependent Register 1	1009
Timer Registers		
TBL	Time Base Lower Register	284
TBU	Time Base Upper Register	285
TCR	Timer Control Register	340
TSR	Timer Status Register	336
DEC	Decrementer Register	22
DECAR	Decrementer Auto-reload Register	54
Debug Registers		
DBCR0	Debug Control Register 0	308
DBCR1	Debug Control Register 1	309
DBCR2	Debug Control Register 2	310
DBCR3	Debug Control Register 3	561
DBSR	Debug Status Register	304
DBCNT	Debug Counter Register	562
IAC1	Instruction Address Compare Register 1	312
IAC2	Instruction Address Compare Register 2	313
IAC3	Instruction Address Compare Register 3	314
IAC4	Instruction Address Compare Register 4	315
DAC1	Data Address Compare Register 1	316
DAC2	Data Address Compare Register 2	317
Memory Management Registers		
MAS0	MMU Assist Register 0	624
MAS1	MMU Assist Register 1	625
MAS2	MMU Assist Register 2r	626
MAS3	MMU Assist Register 3	627
MAS4	MMU Assist Register 4	628
MAS6	MMU Assist Register 6	630
PID0	Process ID Register	48
MMUCSR0	MMU Control and Status Register 0	1012
MMUCFG	MMU Configuration Register	1015
TLB0CFG	TLB 0 Configuration Register	688

Table A-3. e200z6 Core SPR Numbers (Supervisor Mode) (continued)

Register	Description	SPR (decimal)
TLB1CFG	TLB 1 Configuration Register	689
Cache Registers		
L1CFG0	L1 Cache Configuration Register	515
L1CSR0	L1 Cache Control and Status Register 0	1010
L1FINV0	L1 Cache Flush and Invalidate Control Register 0	1016
APU Registers		
SPEFSCR	SPE APU Status and Control Register	512

Table A-4. e200z6 Core SPR Numbers (User Mode)

Register	Description	SPR (decimal)
General Registers		
CTR	Count Register	9
LR	Link Register	8
XER	Integer Exception Register	1
GPR0–GPR31	General Purpose Registers	N/A
Special Purpose Registers		
SPRG4	Special Purpose Register 4	260
SPRG5	Special Purpose Register 5	261
SPRG6	Special Purpose Register 6	262
SPRG7	Special Purpose Register 7	263
USPRG0	User Special Purpose Register	256
Timer Registers		
TBL	Time Base Lower Register	268
TBU	Time Base Upper Register	269
Cache Registers		
L1CFG0	L1 Cache Configuration Register	515
APU Registers		
SPEFSCR	SPE APU Status and Control Register	512

A.1 Revision History

Substantive Changes since Rev 3.0
For the FEC module, changed MDATA to MMFR as well as name of register to MII Management Frame Regsiter.

Appendix B
Calibration

B.1 Overview

The MPC5500 family of microcontrollers includes various specialized features to support automotive calibration. Many of these calibration features are not intended to be available for use by the final application software, and some MPC5500 devices support calibration signals which are not available in the standard 208, 324, and 416 BGA packages. Special calibration packaged devices with increased signal bond out are used to provide full access to all calibration resources for all MPC5500 variants.

Calibration hardware which makes use of these calibration packaged devices is detailed in Figure B-1. Freescale-produced "VertiCal bases" use the calibration-packaged MPC5500 device mounted on a small circuit board with a footprint which is compatible with that of the production BGA packaged MPC5500 device. A 156 way "VertiCal connector" on the top side of the VertiCal base allows VertiCal compliant "top board" hardware to be attached. Various types of top board hardware to support calibration and debug is available from Freescale and 3rd parties.

The VertiCal connector standard defines a set of signals which are used for communication between the microcontroller on the VertiCal Base Board and any attached calibration tools or "top boards". There are some differences in signal availability or sourcing for the VertiCal connector depending on the MPC5500 device variant being used.

The calibration system is illustrated in Figure B-1 and the VertiCal Base is illustrated in Figure B-2.

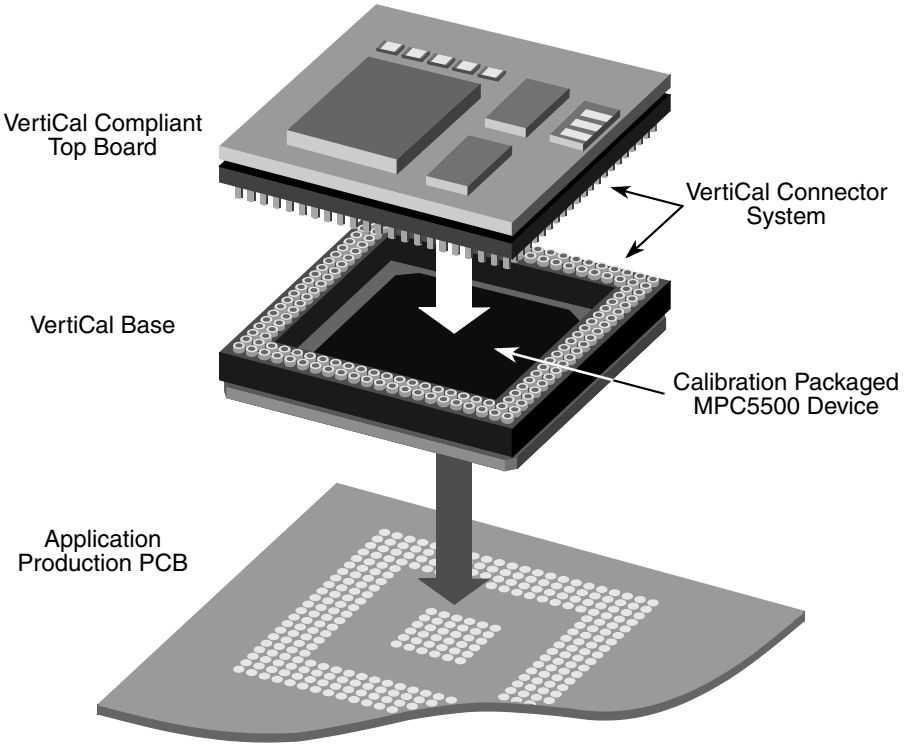


Figure B-1. Calibration System

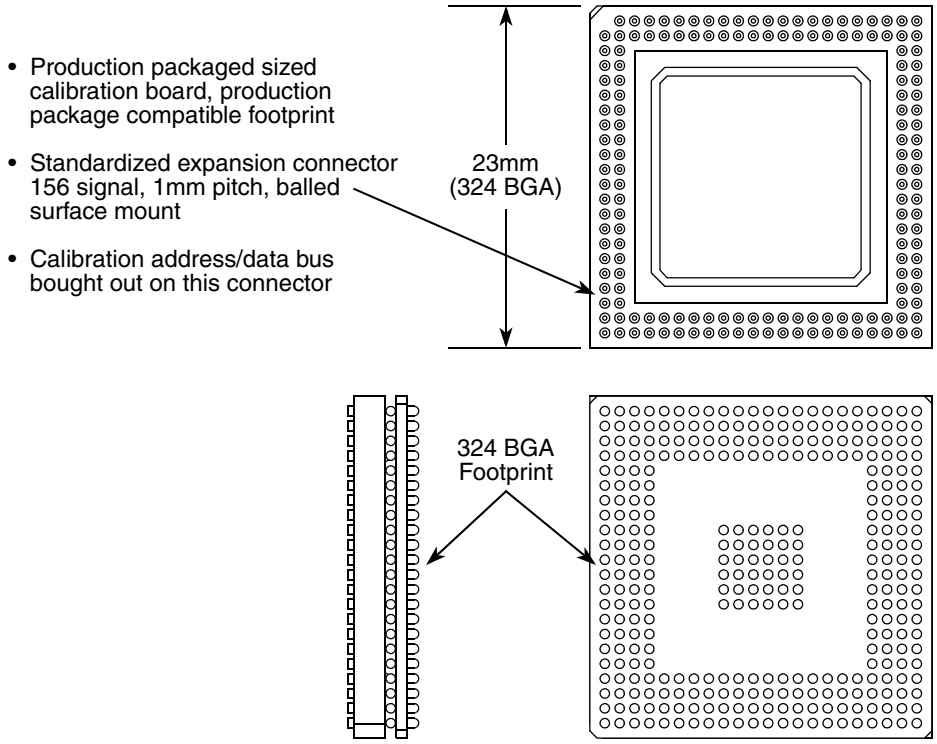


Figure B-2. VertiCal Base

B.2 Calibration Bus

The calibration bus is made up of address bus, data bus, bus control and clock signals, and is used by any tool which includes additional memory to hold calibration data or other code or data being developed. See Table B-1 for calibration bus signals. A 16-bit data bus and 19-bit address bus is included giving a basic addressing range of 1 MByte. Alternatively, the maximum memory addressable using just one chip select is 4 Mbytes. Refer to Table B-2.

The VertiCal connector supports up to 4 chip selects signals, although the actual number of chip selects available depends on which device of the MPC5500 family is used. The CAL_CS[0] chip select is available for all MPC5500 devices, and should be used as the default chip select for calibration use to ensure maximum portability of calibration tools across devices. These additional chip selects signals are configured and function like the non-calibration chip selects. The four chip selects, CAL_CS[0:3], have a higher priority in address decoding than the non-calibration chip selects, CS[0:3]. Refer to Section B.6, "Application Information," for application information on the number of calibration chip selects.

The additional CAL_CS[0:3] chip selects also have alternate functions as additional address bits, allowing a flexible choice between increased addressing range or increased chip select availability. Devices which support less than 4 calibration chip selects are designed to support this means of extending the contiguous calibration addressing range by omitting chip selects starting from CS1. For this reason CS1 is selected as the single unimplemented chip select on the MPC5553.

Table B-1. Calibration Bus Signals

VertiCal Signal Name	Function	Device Implementation Signal Name	
		MPC5554	MPC5553
Address/Data Bus (44)			
CAL_ADDR[12:26]	Address bus	ADDR[12:26]_ GPIO[8:22]	ADDR[12:26]_ GPIO[8:22]
CAL_ADDR[27:30]	Address bus	ADDR[27:30]_ GPIO[23:26]	ADDR[8:11]_ CAL_ADDR[27:30]_ GPIO[4:7]
$\overline{\text{CAL\_CS}}$[3]	Chip Selects	$\overline{\text{CS}}$[3]_ ADDR[11]_ GPIO[3]	CAL_ADDR11_ MDIO_$\overline{\text{CAL_CS}}$3_ GPIO73
$\overline{\text{CAL\_CS}}$[2]	Chip Select	$\overline{\text{CS}}$[2]_ ADDR[10]_ GPIO[2]	CAL_ADDR10_ MDC_$\overline{\text{CAL_CS}}$2_ GPIO72
$\overline{\text{CAL\_CS}}$[1]	Chip Select	$\overline{\text{CS}}$[1]_ ADDR[9]_ GPIO[1]	No Connect
$\overline{\text{CAL\_CS}}$[0]	Chip Select	$\overline{\text{CS}}$[0]_ ADDR[8]_ GPIO[0]	$\overline{\text{TEA}}$_$\overline{\text{CAL_CS}}$0_ GPIO71
CAL_DATA[0:15]	Data Bus	DATA[0:15]_ GPIO[28:43]	DATA[16:31]_*a*_ CAL_DATA[0:15]_ GPIO[44:59][1]
$\overline{\text{CAL\_OE}}$	Output Enable	$\overline{\text{OE}}$_GPIO68	$\overline{\text{OE}}$_GPIO68
CAL_RD_$\overline{\text{WR}}$	Read/Write	RD_$\overline{\text{WR}}$_ GPIO62	RD_$\overline{\text{WR}}$_ GPIO62
$\overline{\text{CAL\_TS}}$	Transfer Start	$\overline{\text{TS}}$_GPIO69	$\overline{\text{TS}}$_GPIO69
$\overline{\text{CAL\_WE}}$[0:1] $\overline{\text{CAL\_BE}}$[0:1]	Write Enable Byte Enable	$\overline{\text{WE}}$[0:1]_ $\overline{\text{BE}}$[64:65]_ GPIO[64:65]	$\overline{\text{WE}}$[2:3]_$\overline{\text{BE}}$[2:3]_ $\overline{\text{CAL\_WE}}$[0:1]_ GPIO[66:67]
Clock Synthesizer (1)			
CLKOUT	System Clock Output	CLKOUT	CLKOUT

[1] For these signals, "*a*" refers to the alternate function, see Table 2-1 for these alternate pin functions.

B.3 Device Specific Information

The various address bus, data bus and bus control signals are sourced from different device signals depending on the MPC5500 family being used as detailed in the following sections.

B.3.1 MPC5554 Calibration Bus Implementation

On the MPC5554 device there are no signals dedicated for calibration usage, and instead signals which are available for normal application usage must be shared for calibration. The calibration bus signals on the VertiCal connector (CAL_DATA, CAL_ADDR, CAL_CS etc.) are connected to the equivalent signals on the standard MPC5554 EBI. To allow calibration, all of the MPC5554 EBI signals included in the VertiCal connector and used by attached VertiCal top board must be available and configured for their primary EBI mode of operation. This requirement prohibits the use of required EBI signals as general purpose IO (GPIO) by the application. If the application itself uses the EBI to access external memory mapped devices, the application design must ensure that sufficient resources such as chip selects and addressing range are left available for calibration use. Since the calibration bus is shared with the standard device system bus, the bus loading of the pins may need to be adjusted for pins that are connected to both the standard bus and the calibration bus. The bus load for each signal is adjustable in the Pad Configuration Register for that pin.

B.3.2 MPC5553 Calibration Bus Implementation

The MPC5553 device is similar to the MPC5554 in that no signals are dedicated for calibration usage. Instead, signals which are available for normal application usage must be shared for calibration. The MPC5553 differs in that the calibration bus signals on the VertiCal connector are not all directly connected to the equivalent signals on the standard EBI. Instead some calibration pins are implemented as secondary functions on pins that are not normally needed. The purpose of this is to minimize the number of signals which must be reserved for calibration on applications which use the 324 BGA packaged device.

B.4 Signals and Pads

The following sections detail the signal descriptions for the calibration bus.

B.4.1 CAL_CS[0:3] — Calibration Chip Selects 0 - 3

CAL_CS[n] is asserted by the master to indicate that this transaction is targeted for a particular calibration memory bank.

The calibration chip selects are driven by the EBI. CAL_CS[n] is driven in the same clock as the assertion of TS and valid address, and is kept valid until the cycle is terminated. Bus timing is identical to standard EBI timing.

B.4.1.1 Number of Chip Selects and Maximum Memory Size

The trade-off between calibration chip selects and address lines is the same as the trade-off between non-calibration chip selects and address lines for the 324 pin package.

Table B-2. Number of Calibration Chip Selects Versus Memory Size

CAL_CS[0]	CAL_CS[2]/ CAL_ADDR[10]	CAL_CS[3]/ CAL_ADDR[11]	CAL_CS[0] maximum memory size (Mbytes)	CAL_CS[2] maximum memory size (Mbytes)	CAL_CS[3] maximum memory size (Mbytes)
CAL_CS[0]	CAL_ADDR[10]	CAL_ADDR[11]	4	-	-
CAL_CS[0]	CAL_CS[2]	CAL_ADDR[11]	2	2	-
CAL_CS[0]	CAL_CS[2]	CAL_CS[3]	1	1	1

B.4.2 Pad Ring

This section provides a list of the calibration pins and associated pad configuration registers (PCRs), including links to the detailed PCR information for each pin or pin group.

Refer to Table B-1 for device signal names.

For MPC5553, see:

- CAL_ADDR[27:30]: Section 6.3.1.12.2, "MPC5553: Pad Configuration Registers 4 - 7 (SIU_PCR4 - SIU_PCR7)," on page 6-21
- CAL_DATA[0:15]: Section 6.3.1.12.5, "MPC5553: Pad Configuration Register 44 (SIU_PCR44)," on page 6-23 through Section 6.3.1.12.20, "MPC5553: Pad Configuration Register 59 (SIU_PCR59)," on page 6-31
- CAL_WE[0:1]_CAL_BE[0:1]: Section 6.3.1.12.25, "MPC5553: Pad Configuration Registers 66 - 67 (SIU_PCR66 - SIU_PCR67)," on page 6-33
- CAL_CS[0]: Section 6.3.1.12.30, "MPC5553: Pad Configuration Register 71 (SIU_PCR71)," on page 6-36
- CAL_CS[2]: Section 6.3.1.12.32, "MPC5553: Pad Configuration Register 72 (SIU_PCR72)," on page 6-38
- CAL_CS[3]: Section 6.3.1.12.34, "MPC5553: Pad Configuration Register 73 (SIU_PCR73)," on page 6-39
- CLKOUT: Section 6.3.1.12.114, "Pad Configuration Register 229 (SIU_PCR229)," on page 6-78

For MPC5554, see:

- Address Bus pins: Section 6.3.1.12.3, "MPC5554: Pad Configuration Registers 4 - 27 (SIU_PCR4 - SIU_PCR27)," on page 6-21
- Data Bus pins: Section 6.3.1.12.4, "Pad Configuration Registers 28 - 59 (SIU_PCR28 - SIU_PCR59)," on page 6-22
- CAL_WE[0:1]_CAL_BE[0:1]: Section 6.3.1.12.26, "MPC5554: Pad Configuration Registers 64 - 67 (SIU_PCR64 - SIU_PCR67)," on page 6-34
- Chip Selects CS[0:3]: Section 6.3.1.12.1, "Pad Configuration Registers 0 - 3 (SIU_PCR0 - SIU_PCR3)," on page 6-20
- CLKOUT: Section 6.3.1.12.114, "Pad Configuration Register 229 (SIU_PCR229)," on page 6-78

The drive strength of the calibration pins may be adjusted in the PCRs.

B.4.3 CLKOUT

CLKOUT is supplied by the clock control block, not the EBI. Nevertheless, the same CLKOUT is used for both the non-calibration and calibration bus.

A drawback of having just one CLKOUT is that while the difference in board timing can be compensated by the adjustment in the drive strength, the CLKOUT timing, and hence the timing of the non-calibration bus, can have minor differences with a calibration tool from the production package.

B.5 Packaging

The addition of the calibration bus means that the device has more pads than can be connected to the balls on a 416 pin package. Therefore, the die is assembled in a 496 pin chip scale package (CSP) and this package is used in the VertiCal base assembly.

B.6 Application Information

B.6.1 Communication With Development Tool Using I/O

The development tool can require some I/Os for communication between the MCU and the development tool on the VertiCal connector. ETRIG[0:1] and GPIO[205] are available only in the 416 pin package. Since the application can not use these pins in the 208 and 324 pin packages, they are candidates for development tool use in a VertiCal connector. Using ETRIG[1] and GPIO[205] still leaves ETRIG[0] for the application in the 416 package.

B.6.2 Matching Access Delay to Internal Flash With Calibration Memory

One use of VertiCal in the Automotive environment is engine calibration. For this application, an SRAM Top Board is added onto the VertiCal connector. This allows the engine calibrator to modify settings in SRAM, possibly using the Nexus interface or even by using the SCI port or a CAN interface.

See Table 13-2 “Internal Flash External Emulation Mode.”

After the data is calibrated, it can be copied into the internal flash. The internal flash can be accessed faster than the calibration memory and this change in calibration data access time could change the overall system performance. To mitigate this change in system performance, the internal flash memory includes a feature that allows accesses to portions of the flash to be slowed down by adding extra wait states. This is done by multiply mapping the internal flash at different locations with different number of wait states. For example, the physical address of the flash array is 0x0000_0000 to 0x00FF_FFFF (depending on array size). That same flash data can be accessed at address 0x0100_0000 to 0x01FF_FFFF but accesses will be 1 clock cycle slower. That same flash data can be accessed at addresses 0x0200_0000 to 0x02FF_FFFF but accesses will be 2 clock cycles slower. This pattern is repeated through the memory map to addresses 0x1F00_0000 to 0x1FFF_FFFF where accesses will be 31 clock cycles slower.

The application would use this feature by mapping the calibration data to a region of the flash memory that has access timing to match the timing of the calibration RAM used when calibrating the data. This remapping of calibration data can be achieved by either using the translation feature of the MMU or rebuilding the code with a modified link file.

B.7 Revision History

Substantive Changes since Rev 3.0
Initial release of Appendix B, “Calibration.”