TMS320C5515/14/05/04 DSP
Direct Memory Access (DMA) Controller

User's Guide

Literature Number: SPRUFT2
October 2010

List of Figures

List of Tables

Read This First

About This Manual

This document describes direct memory access (DMA) controller on the TMS320C5515/14/05/04 Digital Signal Processor (DSP).

Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.
- Registers in this document are shown in figures and described in tables.
  - Each register figure shows a rectangle divided into fields that represent the fields of the register. Each field is labeled with its bit name, its beginning and ending bit numbers above, and its read/write properties below. A legend explains the notation used for the properties.
  - Reserved bits in a register figure designate a bit that is used for future device expansion.

Related Documentation From Texas Instruments

The following documents describe the TMS320C5515/14/05/04 Digital Signal Processor (DSP) Digital Signal Processor (DSP). Copies of these documents are available on the internet at http://www.ti.com.

SWPU073 — TMS320C55x 3.0 CPU Reference Guide. This manual describes the architecture, registers, and operation of the fixed-point TMS320C55x digital signal processor (DSP) CPU.

SPRU652 — TMS320C55x DSP CPU Programmer's Reference Supplement. This document describes functional exceptions to the CPU behavior.

SPRUFO1A — TMS320C5515/14/05/04/VC05/VC04 Digital Signal Processor (DSP) Inter-Integrated Circuit (I2C) Peripheral User's Guide. This document describes the inter-integrated circuit (I2C) peripheral in the TMS320C5515/14/05/04/VC05/VC04 Digital Signal Processor (DSP) devices. The I2C peripheral provides an interface between the device and other devices compliant with Phillips Semiconductors Inter-IC bus (I2C-bus) specification version 2.1 and connected by way of an I2C-bus. This document assumes the reader is familiar with the I2C-bus specification.

SPRUFO2 — TMS320C5515/14/05/04/VC05/VC04 Digital Signal Processor (DSP) Timer/Watchdog Timer User's Guide. This document provides an overview of the three 32-bit timers in the TMS320C5515/14/05/04/VC05/VC04 Digital Signal Processor (DSP) devices. The 32-bit timers of the device are software programmable timers that can be configured as general-purpose (GP) timers. Timer 2 can be configured as a GP, a Watchdog (WD), or both simultaneously.

SPRUFO3 — TMS320C5515/14/05/04/VC05/VC04 Digital Signal Processor (DSP) Serial Peripheral Interface (SPI) User's Guide. This document describes the serial peripheral interface (SPI) in the TMS320C5515/14/05/04/VC05/VC04 Digital Signal Processor (DSP) devices. The SPI is a high-speed synchronous serial input/output port that allows a serial bit stream of programmed length (1 to 32 bits) to be shifted into and out of the device at a programmed bit-transfer rate. The SPI supports multi-chip operation of up to four SPI slave devices. The SPI can operate as a master device only.

SPRUFO4 — TMS320C5515/14/05/04/VC05/VC04 Digital Signal Processor (DSP) General-Purpose Input/Output (GPIO) User's Guide. This document describes the general-purpose input/output (GPIO) on the TMS320C5515/14/05/04/VC05/VC04 digital signal processor (DSP) devices. The GPIO peripheral provides dedicated general-purpose pins that can be configured as either inputs or outputs. When configured as an input, you can detect the state of an internal register. When configured as an output you can write to an internal register to control the state driven on the output pin.

SPRUFO5 — TMS320C5515/14/05/04/VC05/VC04 Digital Signal Processor (DSP) Universal Asynchronous Receiver/Transmitter (UART) User's Guide. This document describes the universal asynchronous receiver/transmitter (UART) peripheral in the TMS320C5515/14/05/04/VC05/VC04 Digital Signal Processor (DSP) devices. The UART performs serial-to-parallel conversions on data received from a peripheral device and parallel-to-serial conversion on data received from the CPU.

SPRUFP1 — TMS320C5515/05/VC05 Digital Signal Processor (DSP) Successive Approximation (SAR) Analog to Digital Converter (ADC) User's Guide. This document provides an overview of the Successive Approximation (SAR) Analog to Digital Converter (ADC) on the TMS320C5515/14/05/04/VC05/VC04 Digital Signal Processor (DSP) devices. The SAR is a 10-bit ADC using a switched capacitor architecture which converts an analog input signal to a digital value.

SPRUFP3 — TMS320C5515/05/VC05 Digital Signal Processor (DSP) Liquid Crystal Display Controller (LCDC) User's Guide. This document describes the liquid crystal display controller (LCDC) in the TMS320C5515/14/05/04/VC05/VC04 Digital Signal Processor (DSP) devices. The LCD controller includes a LCD Interface Display Driver (LIDD) controller.

SPRUFT2— TMS320C5515/14/05/04 DSP Direct Memory Access (DMA) Controller User's Guide This document describes the features and operation of the DMA controller that is available on the TMS320C5515/14/05/04 Digital Signal Processor (DSP) devices. The DMA controller is used to move data among internal memory, external memory, and peripherals without intervention from the CPU and in the background of CPU operation.

SPRUGU6— TMS320C5515/14/05/04 DSP External Memory Interface (EMIF) User's Guide. This document describes the operation of the external memory interface (EMIF) in the TMS320C5515/14/05/04 Digital Signal Processor (DSP) devices. The purpose of the EMIF is to provide a means to connect to a variety of external devices.

SPRUFO6— TMS320C5515/14/05/04/VC05/VC04 DSP Multimedia Card (MMC)/Secure Digital (SD) Card Controller This document describes the Multimedia Card (MMC)/Secure Digital (SD) Card Controller on the TMS320C5515/14/05/04 Digital Signal Processor (DSP) devices. The multimedia card (MMC)/secure digital (SD) card is used in a number of applications to provide removable data storage. The MMC/SD card controller provides an interface to external MMC and SD cards.

SPRUFX2— TMS320C5515/14/05/04 Digital Signal Processor (DSP) Real-Time Clock (RTC) User's Guide. This document describes the operation of the Real-Time Clock (RTC) module in the TMS320C5515/14/05/04 Digital Signal Processor (DSP) devices. The RTC also has the capability to wake-up the power management and apply power to the rest of the device through an alarm, periodic interrupt, or external WAKEUP signal.

SPRUFX4— TMS320C5515/14/05/04 Digital Signal Processor (DSP) Inter-IC Sound (I2S) Bus User's Guide. This document describes the features and operation of Inter-IC Sound (I2S) Bus in the TMS320C5515/14/05/04 Digital Signal Processor (DSP) devices. This peripheral allows serial transfer of full duplex streaming data, usually streaming audio, between DSP and an external I2S peripheral device such as an audio codec.

SPRUFX5— TMS320C5515 DSP System User's Guide. This document describes various aspects of the TMS320C5515 digital signal processor (DSP) including: system memory, device clocking options and operation of the DSP clock generator, power management features, interrupts, and system control.

SPRUGH5— TMS320C5505 DSP System User's Guide. This document describes various aspects of the TMS320C5505 digital signal processor (DSP) including: system memory, device clocking options and operation of the DSP clock generator, power management features, interrupts, and system control.

SPRUFX6— TMS320C5514 DSP System User's Guide. This document describes various aspects of the TMS320C5514 digital signal processor (DSP) including: system memory, device clocking options and operation of the DSP clock generator, power management features, interrupts, and system control.

SPRUGH6— TMS320C5504 DSP System User's Guide. This document describes various aspects of the TMS320C5504 digital signal processor (DSP) including: system memory, device clocking options and operation of the DSP clock generator, power management features, interrupts, and system control.

SPRUGH9— TMS320C5515 DSP Universal Serial Bus 2.0 (USB) Controller User's Guide This document describes the universal serial bus 2.0 (USB) in the TMS320C5515 Digital Signal Processor (DSP) devices. The USB controller supports data throughput rates up to 480 Mbps. It provides a mechanism for data transfer between USB devices.

SPRABB6— FFT Implementation on the TMS320VC5505, TMS320C5505, and TMS320C5515 DSPs This document describes FFT computation on the TMS320VC5505 and TMS320C5505/15 DSPs devices.

Direct Memory Access (DMA) Controller

1 Introduction

This document describes the features and operation of the direct memory access (DMA) controller in the TMS320C5515/14/05/04 Digital Signal Processor (DSP). The DMA controller allows movement of data between internal/external memory and other peripherals without CPU intervention.

1.1 Purpose of the DMA Controller

The DMA controller is used to move data among internal memory, external memory, and peripherals without intervention from the CPU and in the background of CPU operation.

The DSP includes four DMA controllers with four DMA channels each for a total of 16 DMA channels. Aside from the DSP resources they can access, all four DMA controllers are identical. Throughout this document the general operation of each DMA controller will be described. Differences between each DMA controller will be noted when necessary.

1.2 Key Features of the DMA Controller

The DMA controller has the following features:

- Operation that is independent of the CPU.
- Four channels per DMA controller, which allow the DMA controller to keep track of the context of four independent block transfers.
- Event synchronization. DMA transfers in each channel can be made dependent on the occurrence of selected events. For details, see Section 2.7.
- An interrupt for each channel. Each channel can send an interrupt to the CPU on completion of the programmed transfer. See Interrupt Support in Section 2.14.
- A dedicated clock idle domain. The user can put the four device DMA controllers into a low-power state by turning off their input clock. See Power Management in Section 2.15.
- Ping-Pong mode for DMA transfer. This mode provides double buffering capability fully implemented in hardware. For details, see Section 2.9.

To read about the registers used to program the DMA controller, see Section 4.

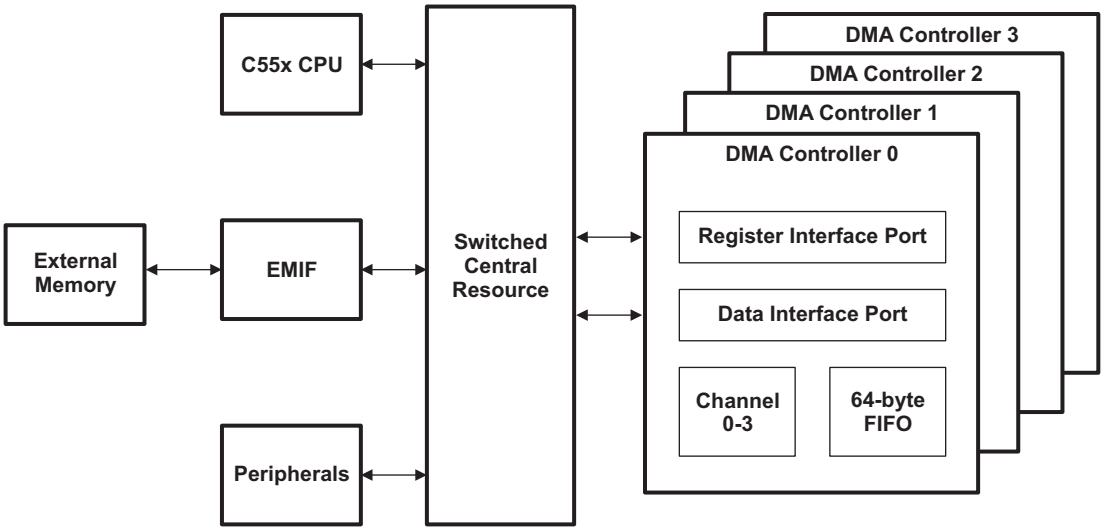
1.3 Block Diagram of the DMA Controller

Figure 1 is a conceptual diagram of connections between the DMA controller and other parts of the DSP. The DMA controller is made up of the following blocks:

- Register interface port. The CPU uses this port to access the DMA controller registers.
- Data interface port. The DMA controller accesses internal dual-access RAM (DARAM), internal single-access RAM (SARAM), external memory, and on-chip peripherals through its data interface port.
- Data transfers are carried out by the four DMA channels. (The DMA channels are described in Section 2.3)
- 64-byte FIFO. As data is read from the source address, it is placed in the DMA controller FIFO. The four DMA channels must share the DMA controller FIFO; the FIFO can only be accessed by a single channel at a time.

It is possible for multiple channels to request access to the DMA controller FIFO at the same time. In this case the DMA controller arbitrates amongst the DMA channels using a round-robin arbitration scheme.

Figure 1. Conceptual Block Diagram of the DMA Controller



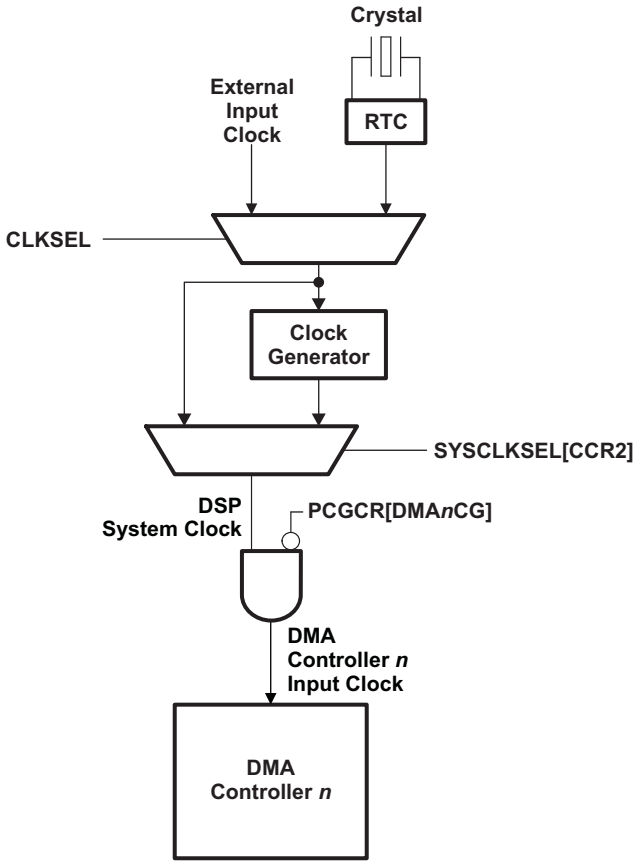
2 DMA Controller Architecture

2.1 Clock Control

As shown in Figure 2, the clock generator receives either the real-time clock (RTC) or a signal from an external clock source and produces the DSP system clock. This clock is used by the DSP CPU and peripherals.

The DSP includes logic which can be used to gate the clock to its on-chip peripherals, including each of the four DMA controllers. The input clock to the DMA controllers can be enabled and disabled through the peripheral clock gating configuration registers (PCGCR1 and PCGCR2).

Figure 2. Clocking Diagram for the DMA Controller



2.2 Memory Map

On the DSP, although all DMA controllers can access internal dual-access RAM (DARAM) and single-access RAM (SARAM), each DMA controller can only access a subset of on-chip peripherals. Also, only DMA controller 3 has access to external memory. The addresses from the point of view of the CPU as compared to the DMA controller are different for DARAM, SARAM and external memory. The DMA controller reads on-chip and off-chip memory by adding the offsets introduced in Table 1. The memory map, as seen by the DMA controllers and the CPU, is shown in Table 1. Peripherals not shown in Table 1 are not accessible by the DMA controllers.

Table 1. DMA Controller Memory Map

DMA Start *Byte* Address	CPU Start *Word* Address (I/O Space)	CPU Start *Word* Address (Data Space)	DSP Memory Map	DMA Controller 0 Memory Map	DMA Controller 1 Memory Map	DMA Controller 2 Memory Map	DMA Controller 3 Memory Map
0000 2800h	00 2800h	-	I2S0	I2S0	Reserved	Reserved	Reserved
0000 3A00h	00 3A00h	-	MMC/SD0	MMC/SD0			
0000 3B00h	00 3B00h	-	MMC/SD1	MMCSD1			
0001 0000h*	-	00 0000h (1)	DARAM	DARAM	DARAM	DARAM	DARAM
0009 0000h	-	00 8000h	SARAM	SARAM	SARAM	SARAM	SARAM
0000 1B00h	00 1B00h	-	UART	Reserved	UART	Reserved	Reserved
0000 2A00h	00 2A00h	-	I2S2		I2S2		
0000 1A00h	00 1A00h	-	I2C		Reserved	I2C	
0000 2B00h	00 2B00h	-	I2S3			I2S3	
0000 2900h	00 2900h	-	I2S1			Reserved	I2S1
0000 7000h	00 7000h	-	10-bit SAR			10-bit SAR	Reserved
0100 0000h	-	02 8000h	SDRAM			Reserved	SDRAM
0200 0000h	-	40 0000h	EMIF CS2				EMIF CS2
0300 0000h	-	60 0000h	EMIF CS3				EMIF CS3
0400 0000h	-	70 0000h	EMIF CS4				EMIF CS4
0500 0000h	-	78 0000h	EMIF CS5				EMIF CS5

(1) Word addresses 00 0000h-00 005Fh (which correspond to byte addresses 00 0000h-00 00BFh) are reserved for the memory-mapped registers (MMRs) of the DSP CPU.

2.3 DMA Channels

Each DMA controller has four channels to transfer data among the DSP resources (DARAM, SARAM, external memory, and peripherals). Each channel reads data from the source address and writes data to the destination address.

The DMA first in, first out (FIFO) buffer is used by the channels to store transfer data; this allows the data transfer to occur in two stages (see Figure 2).

Data read access Transfer of data from the source address to the DMA FIFO buffer.

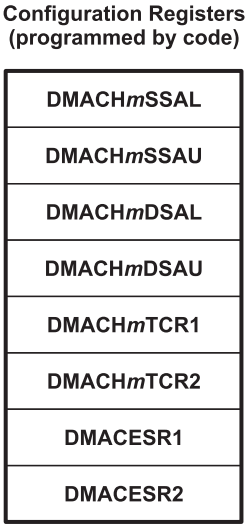
Data write access Transfer of data from the DMA FIFO buffer to the destination address.

Figure 3. Two-Part DMA Transfer

The set of conditions under which transfers occur in a channel is called the channel context. Each of the four channels contains a register structure for programming and updating the channel context (see Figure 4). The user code modifies the configuration registers. The DMA channel becomes active when the channel is enabled (EN = 1 in DMACH*m*TCR2).

The channel configuration registers cannot be programmed while the channel is active (EN = 1 in DMACH*m*TCR2). Modifying channel registers while the DMA channel is running may cause unpredictable operation of the channel. To change a DMA channel configuration, the channel must first be disabled (EN = 0 in DMACH*m*TCR2). The DMA controller will always complete any on-going burst transfer before stopping channel activity. Note that a block transfer may consist of a number of burst transfers. The channel is considered to be active until it completes the burst transfer during which the channel is disabled. After a channel has been disabled, the channel context must be fully reloaded.

Figure 4. Registers for Controlling the Context of a Channel



2.4 Channel Source and Destination Start Addresses

During a data transfer in a DMA channel, the first address at which data is read is called the source start address. The first address to which the data is written is called the destination start address. These are byte addresses. Each channel contains the following registers for specifying the start addresses.

In Ping-Pong mode, the source or destination start address is the start address of the Ping buffer. The length of the **Ping** and the **Pong** buffers should be half of the DMA transfer size as programmed in TCR1. The first half is assumed to be the **Ping** buffer and the second half is assumed to be the **Pong** buffer. These two buffers are required to be contiguous in the memory space and are of equal size. The programmer is responsible for allocating these buffers contiguously. It is recommended to consider the **Ping** and the **Pong** buffers to be a single data buffer in the memory space so that the compiler always allocates them next to each other.

Table 2. Registers Used to Define the Start Addresses for a DMA Transfer

Register	Load with...
DMACH*m*SSAL	Source start address (least-significant part)
DMACH*m*SSAU	Source start address (most-significant part)
DMACH*m*DSAL	Destination start address (least-significant part)
DMACH*m*DSAU	Destination start address (most-significant part)

Table 1 in Section 2.2 shows a high-level memory map of the DSP as seen by the DMA controllers and the CPU. The table shows both the word addresses (23-bit addresses) used by the CPU and byte addresses (32-bit addresses) used by the DMA controller.

The following sections explain how to determine the start address for memory accesses and I/O accesses.

CAUTION

All data buffers in on-chip or off-chip memory should be 32-bit aligned. For more information on managing memory, see the *TMS320VC55x Assembly Language Tools User Guide* (SPRU280).

Additionally, the amount of data (in bytes) to be transferred as programmed in the LENGTH field in DMACH*m*TCR1 should be a multiple of 4 bytes x $2^{BURSTMODE}$ field in DMACH*m*TCR2, i.e. LENGTH = (4 x $2^{BURSTMODE}$) bytes.

2.4.1 Start Address for On-Chip Memory

The CPU uses word addresses and the DMA uses byte addresses. Furthermore, an offset must be added to CPU addresses to generate DMA addresses.

Follow these steps to program the DMA controller with a byte address corresponding to a CPU word address:

1. Identify the correct start address. If you have a word address, shift it left by 1 bit to form a byte address of 32 bits. For example, the CPU word address for SARAM block 0 (00 8000h) should be converted to byte address 0001 0000h.
2. Add correct offset to the CPU byte address. For DARAM, add a value of 01 0000h to the desired byte address. For SARAM, add a value of 08 0000h. For example, since byte address 0001 0000h corresponds to SARAM block 0, a value of 0008 0000h should be added to the byte address to generate 0009 0000h.
3. Load the 16 least significant bits (LSBs) of the byte address into DMACH*m*SSAL (for source) or DMACH*m*DSAL (for destination).
4. Load the 16 most significant bits (MSBs) of the byte address into DMACH*m*SSAU (for source) or DMACH*m*DSAU (for destination).

2.4.2 Start Address for External Memory Space

The CPU uses word addresses and the DMA uses byte addresses. Furthermore, the CPU and DMA controller use different starting addresses for the external memory chip select spaces.

1. Calculate the address offset. Determine the starting word address for the chip select space being used, and then subtract it from the CPU word address (see Table 1 for starting addresses of all chip select spaces). For example, if you are using word address 40 8000h (NAND - CS0 , external memory), then subtract 40 0000h (starting word address for external NAND space in CS0) to generate 01 0000h (00 8000h (40 8000h - 40 0000h).
2. Convert the offset to a byte address offset. Shift the offset left by 1 bit to form a byte address offset of 32 bits. For example, offset 00 8000h should be converted to byte address offset 0001 0000h.
3. Calculate the correct DMA byte address. Use Table 1 to identify the starting DMA byte address of chip select space being used, and then add it from the byte address offset to generate the DMA byte address. For example, byte address offset 0001 0000h becomes 0201 0000h (0200 0000h + 0001 0000h).
4. Load the 16 least significant bits (LSBs) of the byte address into DMACH*m*SSAL (for source) or DMACH*m*DSAL (for destination).
5. Load the 16 most significant bits (MSBs) of the byte address into DMACH*m*SSAU (for source) or DMACH*m*DSAU (for destination).

2.4.3 Start Address for I/O Space

The CPU uses word addresses and the DMA uses byte addresses. The following steps describe how to program the DMA controller with a byte address for a peripheral memory-mapped register:

1. Identify the correct DMA byte address for the peripheral memory-mapped register. The starting DMA byte addresses for the memory-mapped register space of the DSP peripherals are given in Table 1.
2. Load the 16 least significant bits (LSBs) of the byte address into DMACH*m*SSAL (for source) or DMACH*m*DSAL (for destination).
3. Load the 16 most significant bits (MSB) of the byte address into DMACH*m*SSAU (for source) or DMACH*m*DSAU (for destination).

2.5 Updating Addresses in a Channel

During data transfers in a DMA channel, the DMA controller begins its read and write accesses at the start addresses you specify (as described in Section 2.4). Each time the DMA controller services a channel, it transfers the number of double words specified in the BURSTMODE of the channel's transfer control register (DMACH*m*TCR2). If constant addressing mode is selected (DST/SRCAMODE = 10b), the channel does not update the addressing registers (DMACSSA and DMACDSA). Otherwise, if the channel is set to automatic-post increment addressing mode (DST/SRCAMODE = 00b), the channel increments the value in the addressing registers by the total number of bytes transferred.

To change the source or destination address of a channel, the channel must first be disabled by setting EN = 0. The CPU can then update the Source/Destination Address registers and the Transfer Control registers before restarting the channel.

2.6 Data Burst Capability

During a read-write transaction (from source address to destination address) through the Switched Central Resource (see Figure 1), the DMA controller moves data one double word at a time by default. Since every transaction request involves cycle overheads (see Section 2.11), the DMA throughput can be increased by programming the DMA channel to move multiple double words during a transaction, provided both the source and destination targets associated with the transfer support burst capability. The DMA controller can burst to and from SARAM, DARAM, external memory (EMIF) and UART. For a list of DSP resources that support data bursting, see Table 3.

The BURSTMODE bits of the Transfer Control Register 2 (DMACH*m*TCR2) specify the number of double words the DMA controller moves each time it services a channel, i.e., the DMA controller executes a burst of n double words (n = 2, 4, 8, or 16) each time a channel is serviced instead of moving only 1 double word.

Note that the DMA controller services one channel at a time. Each time the DMA services a channel it must transfer the number of double words specified by the burst mode bits. Therefore, care must be taken when programming a channel to use a high burst count since this may impact the minimum amount of time it takes the DMA controller to service other channels. DMA channels are serviced in a round-robin fashion.

Table 3. Destinations/Sources That Support DMA Bursting

Destination/Source Address	Burst Mode Supported
DARAM	1, 2, 4, 8, 16 double words
SARAM	1, 2, 4, 8, 16 double words
UART (1)	1, 2, 4, 8 double words
EMIF	1, 2, 4, 8, 16 double words

(1) The UART treats each double word transfer as a single byte. Therefore, an 8 double word transfer from the DMA to the UART will yield 8 new bytes in the UART FIFO. For more information, see the *Universal Asynchronous Receiver/Transmitter (UART) Reference Guide* (SPRUFO5).

2.7 Synchronizing Channel Activity to DSP Peripheral Events

Activity in a channel can be synchronized to an event in a DSP peripheral. Synchronization is enabled by setting SYNCMODE = 1 in DMACH*m*TCR2. Using the CH*n*EVT bits of DMACESR1 and DMACESR2, the user can specify which synchronization event triggers channel activity. Note that synchronization to an event signaled by the driving of an external interrupt pin is not supported.

If event synchronization is enabled, the channel will wait for the event from the peripheral as programmed in the DMACESR1 and/or DMACESR2 registers before reading from the source address into the DMA FIFO. The synchronization event will trigger the channel to transfer the number of bytes specified by the BURSTMODE bits into the FIFO. Once the FIFO has been filled, the DMA channel will begin writing to the destination address to empty the FIFO.

In non-synchronized transfers (SYNCMODE = 0), the channel sends an access request to the source address as soon as the channel is enabled (EN = 1 in DMACH*m*TCR2). The channel will transfer data from the source address to the FIFO, and then to the destination address until the entire block transfer has been completed or until the user program disables the channel.

The synchronization events are not buffered. Hence, if a synchronization event occurs when the DMA channel is still servicing the previous event, the overlapping (second) event is dropped. In this scenario, the DMA controller does not disable the affected channel nor does it signal an error to the CPU. Therefore, care must be taken when programming the DSP to ensure that the DMA has enough clock cycles to transfer the required number of data bytes on each synchronization event.

Note that some peripherals can generate interrupts whenever a data underrun or overrun condition occurs. The user program can use these interrupts to detect dropped synchronization events.

CAUTION

When using synchronization events, you must set EN = 1 and SYNCMODE = 1 during the same write cycle to DMACH*m*TCR2. The DMA channel will transfer the first data value when it received the synchronization event specified by CH*n*EVT in DMACESR1 and DMACESR2. Also, when disabling the channel, you must set EN = 0 and SYNCMODE = 0 during the same write cycle to DMACH*m*TCR2.

2.8 Channel Auto-Initialization Capability

After a block transfer is completed (all of the bytes specified by LENGTH in DMACH*m*TCR1 have been moved), the DMA controller automatically disables the channel (EN = 0). If it is necessary for the channel to be used again, the CPU can reprogram the new channel context and re-enable the DMA channel, or the DMA controller can automatically initialize the new context and re-enable the channel.

When auto-initialization is used, after each block transfer is completed, the DMA controller automatically reloads the transfer control register and the source and destination start address registers and re-enables the channel allowing the channel to run again. Auto-initialization is enabled by setting the AUTORLD = 1 in the transfer control register (DMACH*m*TCR2).

CAUTION

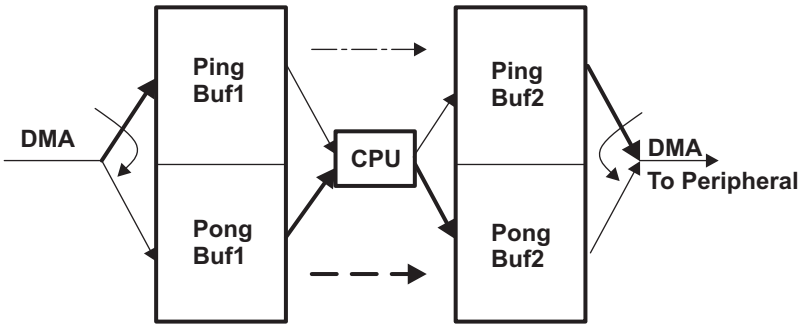
The auto-initialization feature can only be used when event synchronization is used (SYNCMODE = 1 in DMACH*m*TCR2).

Using auto-initialization feature without event synchronization can lead to unintended behavior of the DMA controller.

2.9 Ping-Pong DMA Mode

The Ping-Pong mode for DMA transfer can be used to do continuous processing of incoming data without losing any samples. Every channel in the DMA controller has the capability of being configured in the Ping-Pong mode. This mode can be initiated by setting PING_PONG_EN bit of the Transfer Control Register 2 (TCR2) to 1. For more information, see Section 4.3. You should also define the transfer buffer length in bytes in the Transfer Control Register 1 (TCR1). The length of the **Ping** and the **Pong** buffers should be half of the DMA transfer size as programmed in TCR1. The first half is assumed to be the **Ping** buffer and the second half is assumed to be the **Pong** buffer. These two buffers are required to be contiguous in the memory space and of equal size. The programmer is responsible for allocating these buffers contiguously. It is recommended to consider the **Ping** and the **Pong** buffers to be a single data buffer in the memory space so that the compiler always allocates them next to each other.

Figure 5. Ping-Pong Mode for DMA Data Transfer



As shown in Figure 5, in Ping-Pong mode, DMA starts filling up the **Ping** buffer first. Once the **Ping** buffer is full, a DMA interrupt is generated to the CPU and the LAST_XFER bit (bit 1 in TCR2) is set to 0, which indicates that the data in the **Ping** buffer can be processed. The CPU can start processing the samples in the **Ping** buffer while the DMA is transferring the data to the **Pong** buffer. When the **Pong** buffer is full, another DMA interrupt is sent to the CPU to indicate the availability of data in the **Pong** buffer and the LAST_XFER bit in TCR2 is set to 1. If the AUTORLD bit = 1 in TCR2 (Section 4.3), then the DMA automatically reinitiates the DMA transfer until either EN or AUTORLD bit is set to 0. In the case that the AUTORLD bit is set to 0, DMA stops data transfer after the **Pong** buffer is full and the interrupt is generated. It also resets the EN bit to 0 in TCR2.

At any time during the DMA transfer, LAST_XFER bit of TRC2 (Section 4.3) can be polled to find whether the last completed transfer was the **Ping** or **Pong** buffer.

2.10 Monitoring Channel Activity

The DMA controller can send an interrupt to the CPU whenever a channel has completed a block transfer. Each channel has an interrupt enable (DMA*n*CH*m*IE) bit in the interrupt enable register (DMAIER) and corresponding status bits in the interrupt flag register (DMAIFR). When a channel completes a block transfer, the DMA controller checks the corresponding DMA*n*CH*m*IE bit and acts accordingly:

- If the DMA*n*CH*m*IE bit is 1 (the interrupt is enabled), the DMA controller sets the corresponding status bit and sends the associated interrupt request to the CPU. Your program must manually clear bits in DMAIFR by writing a 1 to them.
- If the DMA*n*CH*m*IE bit is 0, no interrupt is sent and the status bit is not affected.

Each channel also includes a STATUS bit in DMACH*m*TCR2 to indicate the state of the channel transfer. The DMA controller sets the channel STATUS bit to 1 if:

- A nonzero value is written on LENGTH in DMACH*m*TCR1.
- A write access is performed to DMACH*m*TCR2 and LENGTH has a nonzero value.

The DMA controller clears the STATUS bit to 0 if:

- The all the bytes specified by LENGTH in DMACH*m*TCR1 have been transferred.
- A value of 0 is written to LENGTH in DMACH*m*TCR1.

The LAST_XFER bit which is bit 1 in TCR2 (Section 4.3) indicates whether the last completed transfer was the **Ping** or the **Pong** buffer. This bit is valid only when Ping-Pong DMA mode is enabled (PING_PONG_EN bit in TCR2 is 1).

LAST_XFER bit in TCR2 is set to 0 if:

- The last completed transfer was the **Ping** buffer.

LAST_XFER bit in TCR2 is set to 1 if:

- The last completed transfer was the **Pong** buffer.

2.11 Latency in DMA Transfers

Each element transfer in a channel is composed of a read access (a transfer from the source location to the DMA controller FIFO) and a write access (a transfer from the DMA controller FIFO to the destination location). The time to complete this activity depends on factors such as:

- The selected frequency of the CPU clock signal. This signal, as propagated to the DMA controller, determines the timing for all DMA transfers.
- Wait states or other extra cycles added by or resulting from an interface.
- Activity on other channels. Since channels are serviced in a sequential order, the number of pending DMA service requests in the other channels affects how often a given channel can be serviced.
- Competition from the CPU or other DMA controllers. If a DMA controller and the CPU request access to the same internal memory block or peripheral in the same cycle and the memory block or peripheral cannot service both requests at the same time, the CPU request has higher priority. The DMA request is serviced as soon as there are no pending CPU requests.
- The timing of synchronization events (if the channel is synchronized). The DMA controller cannot service a synchronized channel until the synchronization event has occurred. For more details on synchronization, see Section 2.7.

The minimum (best-case) latency for a burst DMA transfer can be summarized as follows:

- For transfers initiating from internal memory: the first access for word read and write takes 8 cycles,, while consecutive accesses take 2 more cycles. Thus the DMA takes 2N + 6 system clock cycles to complete a burst transfer, where N corresponds to the burst size in words.
- For transfers initiating from a peripheral source: the first access for word read and write takes 6 cycles, , while consecutive accesses take 2 more cycles. Thus the DMA takes 2N + 4 system clock cycles to complete a burst transfer, where N corresponds to the burst size in words.

The burst size of the DMA is specified through the BURSTMODE bits. Note that a block transfer may consist of a number of burst transfers. For accesses to external memory, the number of cycles the DMA controller takes to read or write data is determined by the EMIF settings, including the memory type used, programmed timings, and any delays caused by the memory itself (such as control of the wait pin).

2.12 Reset Considerations

The DMA controller has one reset source: a hardware reset. This reset is always initiated during a full chip reset. Alternatively, software can force a hardware reset on all DMA controllers through the DMA_RST bit of the peripheral reset control register (PRCR). See the device data manual for more details on PRCR. Please note that the DMA controller input clock must be enabled when using DMA_RST (see Section 2.2).

When a hardware reset occurs, all the registers of the DMA controllers are set to their default values. The DMA controllers remain inactive until programmed by software.

2.13 Initialization

To initialize the DMA controller follow these steps:

1. Ensure the DMA controller is out of reset by setting the DMA_RST bit to 0 in the peripheral reset control register (PRCR). PRCR is a chip configuration register, it is not part of the DMA controller, see the device data manual for more details.
2. Enable the DMA controller input clock by setting the corresponding DMA*n*CG bit to 0 in the peripheral clock gating configuration registers (PCGCR1 and PCGCR2). PCGCR1 and PCGCR2 are chip configuration registers, they are not part of the DMA controller, see the device data manual for more details.
3. Ensure that all DMA channel interrupt flags are cleared by writing a 1 to the bits of the DMA interrupt flag register (DMAIFR). Also, ensure all DMA interrupt flags in the CPU interrupt flag registers (IFR0 and IFR1) are cleared.
4. If using interrupts, enable the desired channel interrupt by setting the DMA*n*CH*m*IE bits of the interrupt enable register (DMAIER). The CPU interrupt enable bit (INTEN) in the transfer control register 2 (DMACH*m*TCR2) must also be set.
5. If using synchronization events, select the event to be used through the CH*m*EVT bits of the channel event source registers (DMA*n*CESR1 and DMA*n*CESR2). The synchronization mode bit (SYNCMODE) of DMACH*m*TCR2 must also be set, although this should be done only when the channel is ready to be enabled.
6. Load the source address to the source start address registers (DMACH*m*SSAL and DMACH*m*SSAU). See Section 2.4, Start Address in a Channel, for more information on calculating the correct source start address.
7. Load the destination address to the destination start address registers (DMACH*m*DSAL and DMACH*m*DSAU). See Section 2.4, Start Address in a Channel, for more information on calculating the correct destination start address.
8. Load the DMA transfer control register 1 (DMACH*m*TCR1) with the number of double words to transfer. Note that the number of double words must be specified in bytes. For example, for a 256 double word transfer, program this field with 1024 (256 x 4 = 1024). When Ping-Pong DMA mode is enabled, this is the size of the Ping and the Pong buffer combined. For more details, see Section 2.4.
9. Configure DMACH*m*TCR2 accordingly. Through this register you can specify the source and destination addressing modes and burst mode. You can also enable automatic reload, event synchronization, CPU interrupts and Ping-Pong mode. Note that you must keep EN = 0 and SYNCMODE = 0 during this step.
10. If the DMA channel is servicing a peripheral, ensure that the corresponding peripheral is not active and hence not generating synchronization events.
11. Enable the DMA channel by setting EN = 1 (and SYNCMODE = 1 if using synchronization events).
12. If necessary, enable peripheral being serviced the DMA channel.

If using synchronization events, the DMA channel will start a data transfer when an event is received. Otherwise, the DMA channel will start the transfer immediately. At the end of the block transfer, if interrupts are enabled, the DMA controller will generate a CPU interrupt. If interrupts are not enabled, your program can poll DMACH*m*TCR1 until either EN or STATUS are cleared to 0 by the DMA controller to determine when the DMA has finished a block transfer. If AUTORLD is set the DMA controller will restart the specified transfer (for more details, see Section 2.8,).

For more specific examples of programming the DMA controller, see Section 3, DMA Transfer Examples.

NOTE: If a DMA controller is programmed to access on-chip memory, user should make sure that the MPORT is not idled in the Idle Configuration Register (ICR). Note that the the value programmed in the ICR takes effect only on running the 'idle' instruction on the CPU. For more information on these registers, see the *DSP System* Chapter.

2.14 Interrupt Support

2.14.1 Interrupt Events and Requests

Each of the four channels of a DMA controller has its own interrupt which the user can enable or disable a channel interrupt though the DMA*n*CH*m* bits of the DMA interrupt mask register (DMAIMR). The interrupts from the four DMA controllers are combined into a single CPU interrupt. You can determine which DMA channel generated the interrupt by reading the bits of the DMA interrupt flag register (DMAIFR). Your program must manually clear bits in DMAIFR by writing a 1 to them.

2.14.2 Interrupt Multiplexing

As described in the previous section, on the DSP, the interrupts from each of the four DMA controllers are combined into a single CPU interrupt. However, the resulting DMA interrupt is not multiplexed with any other interrupt source.

2.15 Power Management

Each DMA controller can be idled independently to conserve power if it is not being actively used. This is achieved by turning off the peripheral clock of each DMA controller in the peripheral clock gating configuration register (PCGCR). For more information on the PCGCR register, see the *TMS320C5505 System Guide* (SPRUGH6).

2.16 Emulation Considerations

The DMA controller is not interrupted by emulation events such as an emulation breakpoint. However, an emulation suspend may halt activity in a peripheral being serviced by the DMA controller. In this case, the DMA controller activity will be indirectly suspended.

3 DMA Transfer Examples

The DMA controller can be used to perform two basic types of transfers: block transfers and peripheral servicing transfers. The following sections provide examples for these two typical use case scenarios.

3.1 Block Move Example

The most basic transfer performed by the DMA is a block move. During device operation it is often necessary to transfer a block of data from one location to another, usually between on-chip and off-chip memory.

In this example, a section of data is to be copied from external asynchronous memory (SRAM) to internal single-access memory (SARAM). A data block of 256 double words (1024 bytes) residing at CPU word address 78 0000h (external memory) needs to be transferred to internal CPU word address 02 7000h (SARAM, block 31), as shown in Figure 6.

The source address for the transfer is set to the equivalent DMA byte address of the data block in external memory, and the destination address is set to the equivalent DMA byte address of the data block in SARAM. More specifically the equivalent DMA byte addresses for source and destination buffers described in this example are 0500 0000h and 000C E000h, respectively. For more information on DMA byte addresses, see Section 2.2.

Figure 7 shows the DMA channel register contents during the transfer after the channel is enabled.

Figure 6. Block Move Example

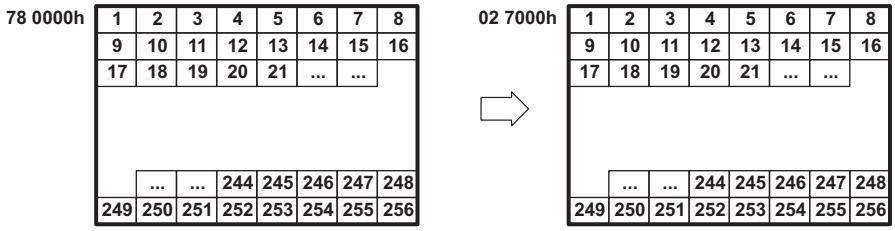


Figure 7. Block Move Example DMA Configuration

(a) DMA Register Contents

Register Contents	
0500h	0000h
000Ch	E000h
E020h	0400h

→

Parameter	
DMACH*m*SSAU	DMACH*m*SSAL
DMACH*m*DSAU	DMACH*m*DSAL
DMACH*m*TCR2	DMACH*m*TCR1

(b) Channel Transfer Control Options

15	14	13	12	11 10	9 8
1	1	1	1	00	00
EN	STATUS	INTEN	AUTORLD	RSV	DSTAMODE

7 6	5 3	2	1	0
10	000	1	0	0
SRCAMODE	BURSTMODE	SYNCMODE	LAST_XFER	PING_PONG_EN

3.2 Peripheral Servicing Example

The DMA controllers can service peripherals in the background of CPU operation, without requiring any CPU intervention. Through proper initialization of the DMA channels, they can be configured to continuously service on-chip and off-chip peripherals throughout device operation. Each DMA controller has a set of synchronization events which can trigger activity in DMA channels specified by the user. When programming a DMA channel to service a peripheral, it is necessary to know how data is to be presented to the DSP. Data is always provided with some kind of synchronization event as either one data sample per event (non-bursting) or multiple data samples per event (bursting).

3.2.1 Non-Bursting Peripherals

Non-bursting peripherals include the on-chip inter-integrated circuit (I2C) module and many external devices, such as codecs. Regardless of the peripheral, the DMA channel configuration is the same.

The I2C transmit and receive data streams are treated independently by the DMA. The transmit and receive data streams can have completely different counts, data sizes, and formats. Figure 8 shows DMA servicing incoming I2C data.

To transfer the incoming data stream to its proper location in internal memory, the DMA channel must be set up for a non-burst transfer with synchronization enabled. Since a receive event (ICREVT) is generated for every data sample as it arrives, it is necessary to have the DMA transfer each data sample individually. Figure 8 shows the DMA channel register contents for this transfer after the channel is enabled.

The source address of the DMA channel is set to the data receive register (ICDRR) address for the I2C, and the destination address is set to the start of the data block in internal memory. Since the address of ICDRR is fixed, the source address mode is set to 10b (constant address) and the destination address mode is set to 00b (automatic post-increment). Note that in this example the destination address is set to the DMA byte address 000C E000h, which corresponds to SARAM block 31. For more information on DMA byte addresses, see Section 2.2.

Note that the DMA will transfer a full double word from ICDRR to the destination address every time a receive synchronization event is generated by the I2C. When allocating memory for the receive buffer, two 16-bit words must be allocated for every I2C data sample.

Figure 8. Servicing Incoming I2C Data Example

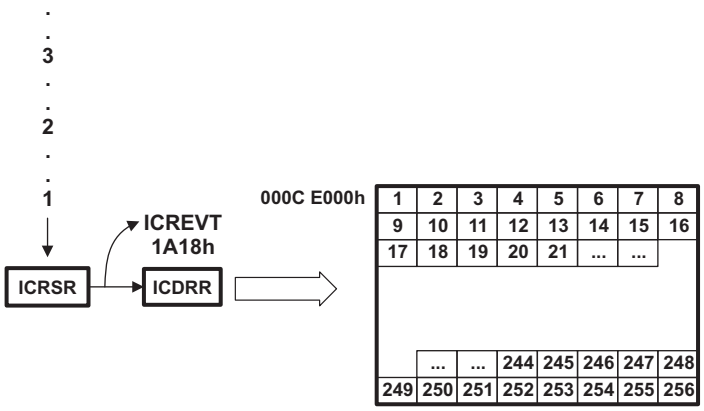


Figure 9. Servicing Incoming I2C Data Example DMA Configuration

(a) DMA Register Contents

Register Contents	
0000h	1A18h
000Ch	E000h
F084h	0400h

→

Parameter	
DMACH*m*SSAU	DMACH*m*SSAL
DMACH*m*DSAU	DMACH*m*DSAL
DMACH*m*TCR2	DMACH*m*TCR1

(b) Channel Transfer Control Options

15	14	13	12	11 10	9 8
1	1	1	1	00	00
EN	STATUS	INTEN	AUTORLD	RSV	DSTAMODE

7 6	5 3	2	1	0
10	000	1	0	0
SRCAMODE	BURSTMODE	SYNCMODE	LAST_XFER	PING_PONG_EN

3.2.2 Bursting Peripherals

Bursting peripherals include the universal asynchronous receiver/transmitter (UART) and the external memory controller (EMIF). For these peripherals, the DMA can be configured to transfer multiple data samples every time the channel is serviced.

The UART transmit and receive data streams are treated independently by the DMA. The transmit and receive data streams can have completely different counts, data sizes, and formats. Furthermore, the DMA burst size feature can be used to empty or fill the UART FIFO every time the UART generates a synchronization event. Figure 10 shows DMA servicing incoming UART data.

To transfer the incoming data in the UART FIFO to its proper location in internal memory, the DMA channel must be set up for a burst transfer with synchronization enabled. Since a receive event (URXEVT) is generated every time the FIFO trigger level is reached, it is necessary to have the DMA channel burst transfer size match the UART FIFO trigger level. For example, if the UART FIFO trigger level is set to 8 bytes, the DMA channel burst size must be set to 8 double words. Note that although the DMA always transfers double words, the UART treats each double word request as a single byte request. Also, when allocating memory for the receive buffer, four bytes must be allocated for every UART data sample.

Figure 11 shows the DMA channel register contents for this transfer after the channel is enabled. The source address of the DMA channel is set to the receive buffer register (RBR) address for the UART, and the destination address is set to the start of the data block in internal memory. Since the address of RBR is fixed, the source address mode is set to 10b (constant address) and the destination address mode is set to 00b (automatic post-increment).

Note that in this example the destination address is set to the DMA byte address 000C E000h, which corresponds to SARAM block 31.

For more information on DMA byte addresses, see Section 2.2.

Figure 10. Servicing Incoming UART Data Example

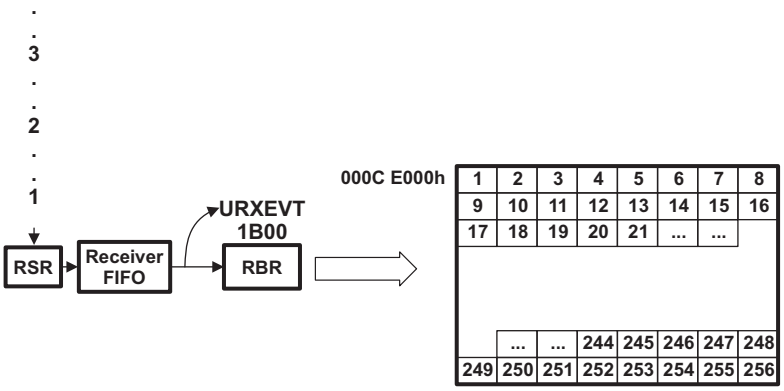


Figure 11. Servicing Incoming UART Data Example DMA Configuration

(a) DMA Register Contents

Register Contents	
0000h	1B00h
000Ch	E000h
F09Ch	0400h

→

Parameter	
DMACH*m*SSAU	DMACH*m*SSAL
DMACH*m*DSAU	DMACH*m*DSAL
DMACH*m*TCR2	DMACH*m*TCR1

(b) Channel Transfer Control Options

15	14	13	12	11 10	9 8
1	1	1	1	00	00
EN	STATUS	INTEN	AUTORLD	RSV	DSTAMODE

7 6	5 3	2	1	0
10	000	1	0	0
SRCAMODE	BURSTMODE	SYNCMODE	LAST_XFER	PING_PONG_EN

3.3 Ping-Pong DMA Example

The example here describes Ping-Pong transfer from an external codec (non-bursting peripheral) to the internal memory. Figure 12 shows DMA transfer of incoming data from the codec through the I2S0 in a Ping-Pong fashion. To transfer the incoming data stream to its destination in the internal memory, the DMA channel must be set up for a non-burst transfer with synchronization enabled. A receive synchronization event is generated for every data sample as it arrives; hence it is necessary to have the DMA transfer each data sample individually.

Figure 12 shows the DMA channel register contents for this transfer after the channel is enabled. The source address of the DMA channel is set to the left data0 receive register (I2S0RXLT0) address for I2S0, and the destination address is set to the start of the data block in internal memory. Since the address of I2S0RXLT0 is fixed, the source address mode is set to 10b (constant address) and the destination address mode is set to 00b (automatic post-increment). Note that in this example the destination address is set to the DMA byte address 000C E000h, which corresponds to SARAM block 31 at 0004 E000h and DMA transfer length is 1K bytes. For more information on DMA byte addresses, see Section 2.2. When allocating memory for the receive buffer, two 16-bit words must be allocated for every I2S data sample. It is also assumed that 1K bytes data buffer has been allocated at 000C E000h.

On every receive event generated by the I2S, the DMA will transfer a full double word from I2S0RXLT0 to the destination address. After the DMA has transferred the 128th sample, it sends an interrupt to the CPU, sets the LAST_XFER bit to 0 and continues the transfer. Once the 256th sample is transferred, the DMA controller again generates an interrupt to the CPU and sets the LAST_XFER bit to 1. Since the AUTORLD bit has been set to 1, the destination address is reloaded and the transfer resumes.

Figure 12. Servicing Incoming I2S Data Example in Ping-Pong DMA Mode

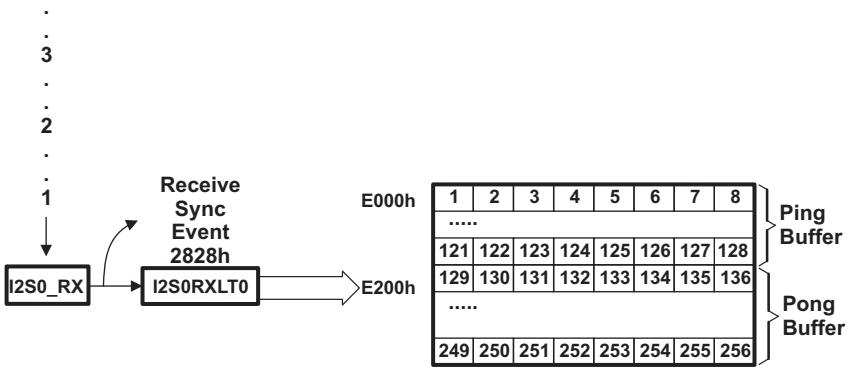


Figure 13. Servicing Incoming I2S Data Example DMA Configuration

(a) DMA Register Contents

Register Contents	
0000h	2828h
000Ch	E000h
F085h	0400h

→

Parameter	
DMACH*m*SSAU	DMACH*m*SSAL
DMACH*m*DSAU	DMACH*m*DSAL
DMACH*m*TCR2	DMACH*m*TCR1

(b) Channel Transfer Control Options

15	14	13	12	11 10	9 8
1	1	1	1	00	00
EN	STATUS	INTEN	AUTORLD	RSV	DSTAMODE

7 6	5 3	2	1	0
10	000	1	0	1
SRCAMODE	BURSTMODE	SYNCMODE	LAST_XFER	PING_PONG_EN

4 Registers

Table 4 through Table 8 list the memory-mapped registers associated with the four direct memory access (DMA) controllers. The DMA controller registers can be accessed by the CPU at the word addresses specified in each table. Note that the CPU accesses all peripheral registers through its I/O space. All other register addresses not listed in the tables below should be considered as reserved locations and the register contents should not be modified.

There are several other registers that affect the operation of the DMA controllers. The DMA interrupt flag and enable registers (DMAIFR and DMAIER) are used to control the interrupt generation of the four DMA controllers. In addition, there are two registers per DMA controller which control event synchronization in each channel—the DMA*n* channel event source registers (DMA*n*CESR1 and DMA*n*CESR2). These registers are not part of the DMA controllers; they are part of the DSP system. For more information on these registers, see the *TMS320VC5505 DSP System Guide* (SPRUFP0).

Table 4. System Registers Related to the DMA Controllers[(1)]

CPU Word Address	Acronym	Register Description
1C30h	DMAIFR	DMA Interrupt Flag Register
1C31h	DMAIER	DMA Interrupt Enable Register
1C1Ah	DMA0CESR1	DMA0 Channel Event Source Register 1
1C1Bh	DMA0CESR2	DMA0 Channel Event Source Register 2
1C1Ch	DMA1CESR1	DMA1 Channel Event Source Register 1
1C1Dh	DMA1CESR2	DMA1 Channel Event Source Register 2
1C36h	DMA2CESR1	DMA2 Channel Event Source Register 1
1C37h	DMA2CESR2	DMA2 Channel Event Source Register 2
1C38h	DMA3CESR1	DMA3 Channel Event Source Register 1
1C39h	DMA3CESR2	DMA3 Channel Event Source Register 2

(1) Refer to the *TMS320VC5505 DSP System Guide* (SPRUFP0) for more information on these registers.

Table 5. DMA Controller 0 (DMA0) Registers

CPU Word Address	Acronym	Register Description	Section
0C00h	DMACH0SSAL	Channel 0 Source Start Address (Lower Part) Register	Section 4.1
0C01h	DMACH0SSAU	Channel 0 Source Start Address (Upper Part) Register	Section 4.1
0C02h	DMACH0DSAL	Channel 0 Destination Start Address (Lower Part) Register	Section 4.2
0C03h	DMACH0DSAU	Channel 0 Destination Start Address (Upper Part) Register	Section 4.2
0C04h	DMACH0TCR1	Channel 0 Transfer Control Register 1	Section 4.3
0C05h	DMACH0TCR2	Channel 0 Transfer Control Register 2	Section 4.3
0C20h	DMACH1SSAL	Channel 1 Source Start Address (Lower Part) Register	Section 4.1
0C21h	DMACH1SSAU	Channel 1 Source Start Address (Upper Part) Register	Section 4.1
0C22h	DMACH1DSAL	Channel 1 Source Start Address (Lower Part) Register	Section 4.2
0C23h	DMACH1DSAU	Channel 1 Destination Start Address (Upper Part) Register	Section 4.2
0C24h	DMACH1TCR1	Channel 1 Transfer Control Register 1	Section 4.3
0C25h	DMACH1TCR2	Channel 1 Transfer Control Register 2	Section 4.3
0C40h	DMACH2SSAL	Channel 2 Source Start Address (Lower Part) Register	Section 4.1
0C41h	DMACH2SSAU	Channel 2 Source Start Address (Upper Part) Register	Section 4.1
0C42h	DMACH2DSAL	Channel 2 Destination Start Address (Lower Part) Register	Section 4.2
0C43h	DMACH2DSAU	Channel 2 Destination Start Address (Upper Part) Register	Section 4.2
0C44h	DMACH2TCR1	Channel 2 Transfer Control Register 1	Section 4.3
0C45h	DMACH2TCR2	Channel 2 Transfer Control Register 2	Section 4.3
0C60h	DMACH3SSAL	Channel 3 Source Start Address (Lower Part) Register	Section 4.1
0C61h	DMACH3SSAU	Channel 3 Source Start Address (Upper Part) Register	Section 4.1
0C62h	DMACH3DSAL	Channel 3 Destination Start Address (Lower Part) Register	Section 4.2

Table 5. DMA Controller 0 (DMA0) Registers (continued)

CPU Word Address	Acronym	Register Description	Section
0C63h	DMACH3DSAU	Channel 3 Destination Start Address (Upper Part) Register	Section 4.2
0C64h	DMACH3TCR1	Channel 3 Transfer Control Register 1	Section 4.3
0C65h	DMACH3TCR2	Channel 3 Transfer Control Register 2	Section 4.3

Table 6. DMA Controller 1 (DMA1) Registers

CPU Word Address	Acronym	Register Description	Section
0D00h	DMACH0SSAL	Channel 0 Source Start Address (Lower Part) Register	Section 4.1
0D01h	DMACH0SSAU	Channel 0 Source Start Address (Upper Part) Register	Section 4.1
0D02h	DMACH0DSAL	Channel 0 Destination Start Address (Lower Part) Register	Section 4.2
0D03h	DMACH0DSAU	Channel 0 Destination Start Address (Upper Part) Register	Section 4.2
0D04h	DMACH0TCR1	Channel 0 Transfer Control Register 1	Section 4.3
0D05h	DMACH0TCR2	Channel 0 Transfer Control Register 2	Section 4.3
0D20h	DMACH1SSAL	Channel 1 Source Start Address (Lower Part) Register	Section 4.1
0D21h	DMACH1SSAU	Channel 1 Source Start Address (Upper Part) Register	Section 4.1
0D22h	DMACH1DSAL	Channel 1 Destination Start Address (Lower Part) Register	Section 4.2
0D23h	DMACH1DSAU	Channel 1 Destination Start Address (Upper Part) Register	Section 4.2
0D24h	DMACH1TCR1	Channel 1 Transfer Control Register 1	Section 4.3
0D25h	DMACH1TCR2	Channel 1 Transfer Control Register 2	Section 4.3
0D40h	DMACH2SSAL	Channel 2 Source Start Address (Lower Part) Register	Section 4.1
0D41h	DMACH2SSAU	Channel 2 Source Start Address (Upper Part) Register	Section 4.1
0D42h	DMACH2DSAL	Channel 2 Destination Start Address (Lower Part) Register	Section 4.2
0D43h	DMACH2DSAU	Channel 2 Destination Start Address (Upper Part) Register	Section 4.2
0D44h	DMACH2TCR1	Channel 2 Transfer Control Register 1	Section 4.3
0D45h	DMACH2TCR2	Channel 2 Transfer Control Register 2	Section 4.3
0D60h	DMACH3SSAL	Channel 3 Source Start Address (Lower Part) Register	Section 4.1
0D61h	DMACH3SSAU	Channel 3 Source Start Address (Upper Part) Register	Section 4.1
0D62h	DMACH3DSAL	Channel 3 Destination Start Address (Lower Part) Register	Section 4.2
0D63h	DMACH3DSAU	Channel 3 Destination Start Address (Upper Part) Register	Section 4.2
0D64h	DMACH3TCR1	Channel 3 Transfer Control Register 1	Section 4.3
0D65h	DMACH3TCR2	Channel 3 Transfer Control Register 2	Section 4.3

Table 7. DMA Controller 2 (DMA2) Registers

CPU Word Address	Acronym	Register Description	Section
0E00h	DMACH0SSAL	Channel 0 Source Start Address (Lower Part) Register	Section 4.1
0E01h	DMACH0SSAU	Channel 0 Source Start Address (Upper Part) Register	Section 4.1
0E02h	DMACH0DSAL	Channel 0 Destination Start Address (Lower Part) Register	Section 4.2
0E03h	DMACH0DSAU	Channel 0 Destination Start Address (Upper Part) Register	Section 4.2
0E04h	DMACH0TCR1	Channel 0 Transfer Control Register 1	Section 4.3
0E05h	DMACH0TCR2	Channel 0 Transfer Control Register 2	Section 4.3
0E20h	DMACH1SSAL	Channel 1 Source Start Address (Lower Part) Register	Section 4.1
0E21h	DMACH1SSAU	Channel 1 Source Start Address (Upper Part) Register	Section 4.1
0E22h	DMACH1DSAL	Channel 1 Destination Start Address (Lower Part) Register	Section 4.2
0E23h	DMACH1DSAU	Channel 1 Destination Start Address (Upper Part) Register	Section 4.2
0E24h	DMACH1TCR1	Channel 1 Transfer Control Register 1	Section 4.3
0E25h	DMACH1TCR2	Channel 1 Transfer Control Register 2	Section 4.3

Table 7. DMA Controller 2 (DMA2) Registers (continued)

CPU Word Address	Acronym	Register Description	Section
0E40h	DMACH2SSAL	Channel 2 Source Start Address (Lower Part) Register	Section 4.1
0E41h	DMACH2SSAU	Channel 2 Source Start Address (Upper Part) Register	Section 4.1
0E42h	DMACH2DSAL	Channel 2 Destination Start Address (Lower Part) Register	Section 4.2
0E43h	DMACH2DSAU	Channel 2 Destination Start Address (Upper Part) Register	Section 4.2
0E44h	DMACH2TCR1	Channel 2 Transfer Control Register 1	Section 4.3
0E45h	DMACH2TCR2	Channel 2 Transfer Control Register 2	Section 4.3
0E60h	DMACH3SSAL	Channel 3 Source Start Address (Lower Part) Register	Section 4.1
0E61h	DMACH3SSAU	Channel 3 Source Start Address (Upper Part) Register	Section 4.1
0E62h	DMACH3DSAL	Channel 3 Destination Start Address (Lower Part) Register	Section 4.2
0E63h	DMACH3DSAU	Channel 3 Destination Start Address (Upper Part) Register	Section 4.2
0E64h	DMACH3TCR1	Channel 3 Transfer Control Register 1	Section 4.3
0E65h	DMACH3TCR2	Channel 3 Transfer Control Register 2	Section 4.3

Table 8. DMA Controller 3 (DMA3) Registers

CPU Word Address	Acronym	Register Description	Section
0F00h	DMACH0SSAL	Channel 0 Source Start Address (Lower Part) Register	Section 4.1
0F01h	DMACH0SSAU	Channel 0 Source Start Address (Upper Part) Register	Section 4.1
0F02h	DMACH0DSAL	Channel 0 Destination Start Address (Lower Part) Register	Section 4.2
0F03h	DMACH0DSAU	Channel 0 Destination Start Address (Upper Part) Register	Section 4.1
0F04h	DMACH0TCR1	Channel 0 Transfer Control Register 1	Section 4.3
0F05h	DMACH0TCR2	Channel 0 Transfer Control Register 2	Section 4.3
0F20h	DMACH1SSAL	Channel 1 Source Start Address (Lower Part) Register	Section 4.1
0F21h	DMACH1SSAU	Channel 1 Source Start Address (Upper Part) Register	Section 4.1
0F22h	DMACH1DSAL	Channel 1 Destination Start Address (Lower Part) Register	Section 4.2
0F23h	DMACH1DSAU	Channel 1 Destination Start Address (Upper Part) Register	Section 4.2
0F24h	DMACH1TCR1	Channel 1 Transfer Control Register 1	Section 4.3
0F25h	DMACH1TCR2	Channel 1 Transfer Control Register 2	Section 4.3
0F40h	DMACH2SSAL	Channel 2 Source Start Address (Lower Part) Register	Section 4.1
0F41h	DMACH2SSAU	Channel 2 Source Start Address (Upper Part) Register	Section 4.1
0F42h	DMACH2DSAL	Channel 2 Destination Start Address (Lower Part) Register	Section 4.2
0F43h	DMACH2DSAU	Channel 2 Destination Start Address (Upper Part) Register	Section 4.2
0F44h	DMACH2TCR1	Channel 2 Transfer Control Register 1	Section 4.3
0F45h	DMACH2TCR2	Channel 2 Transfer Control Register 2	Section 4.3
0F60h	DMACH3SSAL	Channel 3 Source Start Address (Lower Part) Register	Section 4.1
0F61h	DMACH3SSAU	Channel 3 Source Start Address (Upper Part) Register	Section 4.1
0F62h	DMACH3DSAL	Channel 3 Destination Start Address (Lower Part) Register	Section 4.2
0F63h	DMACH3DSAU	Channel 3 Destination Start Address (Upper Part) Register	Section 4.2
0F64h	DMACH3TCR1	Channel 3 Transfer Control Register 1	Section 4.3
0F65h	DMACH3TCR2	Channel 3 Transfer Control Register 2	Section 4.3

4.1 Source Start Address Registers (DMACH*m*SSAL and DMACH*m*SSAU)

Each channel has two source start address registers, which are shown in Figure 14 and Figure 15 and described in Table 9 and Table 10. For the first access to the source port of the channel, the DMA controller generates a byte address by concatenating the contents of the two I/O-mapped registers. DMACH*m*SSAU supplies the upper bits, and DMACH*m*SSAL supplies the lower bits:

Source start address = DMACH*m*SSAU:DMACH*m*SSAL

The channel updates the source start address registers every time it is serviced the DMA controller. The amount of data transferred each time the channel is serviced is specified by the BURSTMODE bits of DMACH*m*TCR2.

The destination start address is supplied by DMACH*m*DSAL and DMACH*m*DSAU, which are described in Section 4.2.

NOTE:

1. You must load the source start address registers with a byte address. For more details, see Section 2.4.
2. There are four DMA controllers in the DSP, although all DMA controllers can access DARAM and SRAM, each DMA controller can only access a subset of on-chip peripherals. Also, only DMA controller 3 has access to external memory. For more details, see Section 2.2.
3. All data buffers in on-chip or off-chip memory should be aligned on an even boundary. For more information on managing memory, see the *TMS320VC55x Assembly Language Tools User Guide* (SPRU280).

Figure 14. Source Start Address Register - Lower Part (DMACH*m*SSAL)

15	0
SSAL	
RW-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figure 15. Source Start Address Register - Upper Part (DMACH*m*SSAU)

15	0
SSAU	
RW-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 9. Source Start Address Register - Lower Part (DMACH*m*SSAL) Field Description

Bit	Field	Value	Description
15-0	SSAL	0-FFFFh	Lower part of source start address (byte address).

Table 10. Source Start Address Register - Upper Part (DMACH*m*SSAU) Field Description

Bit	Field	Value	Description
15-0	SSAU	0-FFFFh	Upper part of source start address (byte address).

4.2 Destination Start Address Registers (DMACHmDSAL and DMACHmDSAU)

Each channel has two destination start address registers, which are shown in Figure 16 as well as Figure 17 and described in Table 11 and Table 12. For the first access to the destination port of the channel, the DMA controller generates a byte address by concatenating the contents of the two I/O-mapped registers. DMACH*m*DSAU supplies the upper bits, and DMACH*m*DSAL supplies the lower bits:

Destination start address = DMACH*m*DSAU:DMACH*m*DSAL

The channel updates the source start address registers every time it is serviced the DMA controller. The amount of data transferred each time the channel is serviced is specified by the BURSTMODE bits of DMACH*m*TCR2.

The source start address is supplied by DMACH*m*SSAL and DMACH*m*SSAU, which are described in Section 4.1.

NOTE:

1. You must load the source start address registers with a byte address. For more details, see Section 2.4.
2. There are four DMA controllers in the DSP, although all DMA controllers can access DARAM and SRAM, each DMA controller can only access a subset of on-chip peripherals. Also, only DMA controller 3 has access to external memory. For more details, see Section 2.2.
3. All data buffers in on-chip or off-chip memory should be aligned on an even boundary. For more information on managing memory, see the *TMS320VC55x Assembly Language Tools User Guide* (SPRU280).

Figure 16. Destination Start Address Register - Lower Part (DMACH*m*DSAL)

15	0
DSAL	
RW-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figure 17. Destination Start Address Register - Upper Part (DMACH*m*DSAU)

15	0
DSAU	
RW-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11. DMA Destination Start Address Register - Lower Part (DMACH*m*DSAL) Field Description

Bit	Field	Value	Description
15-0	DSAL	0-FFFFh	Lower part of destination start address (byte address).

Table 12. DMA Destination Start Address Register - Upper Part (DMACH*m*DSAU) Field Description

Bit	Field	Value	Description
15-0	DSAU	0-FFFFh	Upper part of destination start address (byte address).

4.3 Transfer Control Registers (DMACHmTCR1 and DMACHmTCR2)

Each channel has two transfer control registers shown in Figure 18 and Figure 19. These I/O-mapped register enables you to specify the size of the transfer, enable event synchronization and auto-initialization, select the source and destination addressing mode, and specify the burst mode of the channel. You can also monitor the status of the DMA channel through these registers. Table 13 and Table 14 describes the fields of these registers.

CAUTION

When using synchronization events, you must set EN = 1 and SYNCMODE = 1 during the same write cycle to DMACH*m*TCR2. The DMA channel will transfer the first data value when it received the synchronization event specified by CH*n*EVT in DMACESR1 and DMACESR2. Also, when disabling the channel, you must set EN = 0 and SYNCMODE = 0 during the same write cycle to DMACH*m*TCR2.

The amount of data (in bytes) to be transferred as programmed in the LENGTH field in DMACH*m*TCR1 should be a multiple of 4 bytes x $2^{BURSTMODE}$ field in DMACH*m*TCR2, i.e., LENGTH = (4 x $2^{BURSTMODE}$) bytes.

Figure 18. Transfer Control Register 1 (DMACH*m*TCR1)

15	0
LENGTH	
RW-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figure 19. Transfer Control Register 2 (DMACH*m*TCR2)

15	14	13	12	11	10	9	8
EN	STATUS	INTEN	AUTORLD	RSV		DSTAMODE	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0		R/W-0	

7	6	5	3	2	1	0
SRCAMODE		BURSTMODE		SYNCMODE	LAST_XFER	PING_PONG_EN
R/W-0		R/W-0		R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 13. Transfer Control Register 1 (DMACH*m*TCR1) Field Description

Bit	Field	Value	Description
15-0	LENGTH		Size of transfer. When Ping-Pong mode is enabled, this is the size of the **Ping** and the **Pong** transfer combined.
		0000h - 0003h	Reserved, do not use.
		0004h - FFFCh	These bits specify the number of double words (specified in bytes) to be transferred in multiples of (4 x $2^{BURSTMODE}$) bytes.
			Note: These bits are not updated by the channel each time it is serviced by the DMA controller.

Table 14. Transfer Control Register 2 (DMACH*m*TCR2) Field Descriptions

Bit	Field	Value	Description
15	EN		Channel enable bit. Use EN to enable or disable transfers in the channel. The DMA controller clears EN once a block transfer in the channel is complete.
		0	The channel is disabled. the channel cannot be serviced by the DMA controller. IF DMA burst transfer is ongoing in the channel, the DMA controller allows the burst transfer to complete.
			Note: When you write a 0 to this bit, you must also write a 0 to the SYNCMODE bit.
		1	The channel is enabled. the channel can be serviced by the DMA in the next available time slot.
14	STATUS		Channel status bit. This bit indicates the status o the DMA channel transfer.
			The DMA controller sets the channel STATUS bit to 1 if:
			- A nonzero value is written on LENGTH in DMACH*m*TCR1
			- A write access is performed to DMACH*m*TCR2 and LENGTH has a nonzero value.
			The DMA controller clears the STATUS bit to 0 if:
			- All the bytes specified by LENGTH in DMACH*m*TCR1 have been transferred.
			- A value of 0 is written to LENGTH in DMACH*m*TCR1
		0	Corresponding DMA channel has transferred all the bytes specified by LENGTH in DMACH*m*TCR1.
		1	Corresponding DMA channel has not finished transferring all the bytes specified by LENGTH in DMACH*m*TCR1.
13	INTEN		CPU interrupt enable bit. The DMA channel is capable of generating a CPU interrupt when a block transfer is finished. In order for the CPU to receive the interrupt, the corresponding channel interrupt mask bit in the interrupt mask register (DMAIMR) must be set to 1.
		0	Disable channel interrupt.
		1	Enabled channel interrupt.
12	AUTORLD		Automatic reload bit. Once a transfer is finished, the DMA automatically reloads the transfer control register and the source and destination start address registers and restarts the transfer.
			Note: Automatic reload can only be used when SYNCMODE = 1.
		0	DMA transfer does not automatically reload.
		1	Upon completion of a full transfer, the registers are reloaded and the transfer is restarted.
11-10	Reserved	0	Reserved, always write zeroes to these bits.
9-8	DSTAMODE		Destination addressing mode bits. DSTAMODE determines the addressing mode used by the DMA controller when it writes to the destination address.
		0	Automatic post increment. The destination byte address is incremented by four each transfer.
		1h	Reserved, do not use.
		2h	Constant address.
		3h	Reserved, do not use.
7-6	SRAMODE		Source addressing mode bits. SRCAMODE determines the addressing mode used by the DMA controller when it reads from the source address.
		0	Automatic post increment. The source byte address is incremented by four after each transfer.
		1h	Reserved, do not use.
		2h	Constant address.
		3h	Reserved, do not use.
5-3	BURSTMODE		Burst mode bits. These bits specify the number of double word transfers that each channel performs at once before the DMA controller moves on to the active channel.
			Note: The burst mode selected must always be less than or equal to the number of bytes specified in DMACH*m*TCR1.
		0	1 double word (4 bytes).
		1h	2 double words (8 bytes).
		2h	4 double words (16 bytes).
		3h	8 double words (32 bytes).
		4h	16 double words (64 bytes).
		5h-7h	Reserved, do not use.

Table 14. Transfer Control Register 2 (DMACH*m*TCR2) Field Descriptions (continued)

Bit	Field	Value	Description
2	SYNCMODE		Synchronization mode bit. CH*n*EVT bits in DMACESR*n* determine which event in the DSP (for example, a timer countdown) initiates a DMA transfer in the channel. Multiple channels can have the same SYNCEVT value; in other words, one synchronization event can initiate activity in multiple channels. On each sync event, the DMA transfers the number of double words specified by the BURSTMODE bits. For example, BURSTMODE = 010b, the DMA will transfer a total of 4 double words per sync event. A DSP reset selects SYNCMODE = 0 (no synchronization). When SYNCMODE = 0, the DMA controller does not wait for a synchronization event before beginning a DMA transfer in the channel; channel activity begins as soon as the channel is enabled (EN = 1).
		0	When synchronization is disabled, the DMA controller does not wait for a synchronization event before beginning a DMA transfer. **Note:** When you write a 0 to the EN bit, you must also write a 0 to this bit.
		1	Activity in the DMA controller is synchronized to the event specified in the CH*n*EVT bits of DMACESR*n*. **Note:** When you set this bit to 1, you must also write a 1 to the EN bit.
1	LAST_XFER		Indicates whether the most recent completed transfer was the **Ping** buffer or the **Pong** buffer. This status bit is only valid when the PING_PONG_EN bit is set to 1.
		0	The last completed transfer was the **Ping** buffer
		1	The last completed transfer was the **Pong** buffer
0	PING_PONG_EN		Enable Ping-Pong DMA transfer mode.
		0	Ping-Pong mode is disabled
		1	Ping-Pong mode is enabled

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DLP® Products	www.dlp.com	Communications and Telecom	www.ti.com/communications
DSP	dsp.ti.com	Computers and Peripherals	www.ti.com/computers
Clocks and Timers	www.ti.com/clocks	Consumer Electronics	www.ti.com/consumer-apps
Interface	interface.ti.com	Energy	www.ti.com/energy
Logic	logic.ti.com	Industrial	www.ti.com/industrial
Power Mgmt	power.ti.com	Medical	www.ti.com/medical
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
RFID	www.ti-rfid.com	Space, Avionics & Defense	www.ti.com/space-avionics-defense
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Video and Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless-apps

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265