

# Using the Quadrature Decoder (QD) eTPU Function

Covers the MCF523x, MPC5500, and all eTPU-equipped Devices

by: Milan Brejl, Michal Princ  
System Application Engineers, Roznov Czech System Center  
Andrey Butok  
System Application Engineer, Kiev Embedded Software Lab

## 1 Introduction

The quadrature decoder (QD) enhanced time processor unit (eTPU) function is one of the functions included in the DC motor control eTPU function set (set3) and AC motor control eTPU function set (set4). This application note is intended to provide simple C interface routines to the QD eTPU function. The routines are targeted at the MCF523x and the MPC5500 families of devices, but they could be easily used with any device that has an eTPU.

## 2 Function Overview

The QD eTPU function set is intended to process signals generated by a shaft encoder in motion control systems. It uses two channels to decode a pair of out-of-phase encoder signals and to produce a 24-bit, bi-directional position counter, together with direction information, for the CPU.

### Table of Contents

1	Introduction.....	1
2	Function Overview.....	1
3	Function Description.....	2
4	C Level API for Function.....	8
5	Example Use of Function .....	19
6	Summary and Conclusions .....	21
7	References .....	21

## Function Description

Additional input channels can also be processed. The index channel receives a pulse on each revolution. Based on the actual direction, a revolution counter is incremented or decremented on the index pulse. A further additional input channel can indicate a home position. When the position is reached, appropriate actions are taken (reset of the position counter and the revolution counter).

# 3 Function Description

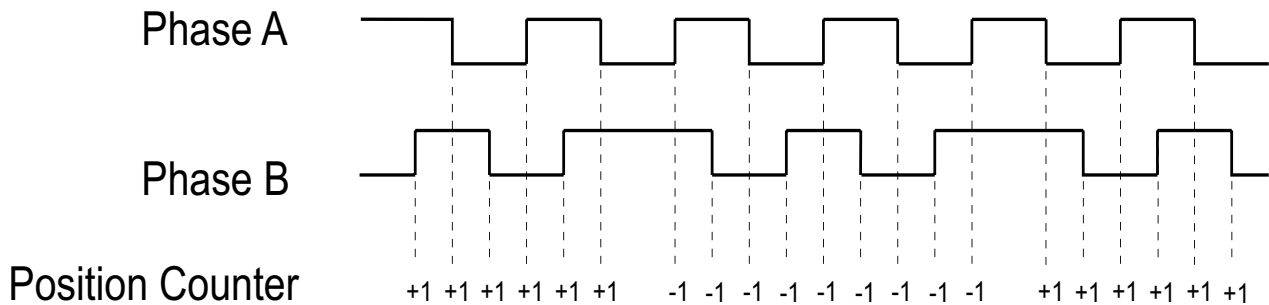
The QD function uses a pair of eTPU channels to decode quadrature signals in order to increment or decrement a 24-bit position counter. The two out of phase encoder signals are called phase A (primary channel) and phase B (secondary channel). The counter is updated when a valid transition is detected on either one of the two inputs. The counter is incremented or decremented depending on the lead/lag relationship of the two signals at the time of servicing the transition.

## 3.1 Modes of Operation

The QD function operates in several modes: slow, normal and fast. The QD function itself ensures switching between these modes depending on the current encoder speed.

### 3.1.1 Slow Mode

This mode applies at slow speeds, when the shaft starts to rotate or when the shaft is stopping. The current motion system speed (determined by the period between last two transitions) must be less than a defined slow-to-normal threshold. In slow mode, both quadrature signals are decoded by the eTPU, and the counter is updated by one for each valid transition on either channel (see [Figure 1](#)).



**Figure 1. Slow and Normal Mode of Operation**

The counter is incremented or decremented depending on the lead/lag relationship of the two signals at the time of transition service. The direction is calculated according to the lead/lag relationship of the two QD signals each slow mode thread. See [Table 1](#) for a definition of the lead/lag test.

Table 1. Lead/Leg Test Results

Serviced Transition	Test Description
Primary Rising	If last secondary transition was falling, then primary channel is leading the secondary channel, and the PC is incremented. If last secondary transition was rising, then primary channel is lagging the secondary channel, and the PC is decremented.
Primary Falling	If last secondary transition was rising, then primary channel is leading the secondary channel, and the PC is incremented. If last secondary transition was falling, then primary channel is lagging the secondary channel, and the PC is decremented.
Secondary Rising	If last primary transition was rising, then primary channel is leading the secondary channel, and the PC is incremented. If last primary transition was falling, then primary channel is lagging the secondary channel, and the PC is decremented.
Secondary Falling	If last primary transition was falling, then primary channel is leading the secondary channel, and the PC is incremented. If last primary transition was rising, then primary channel is lagging the secondary channel, and the PC is decremented.

### 3.1.2 Normal Mode

This mode applies when the current motion system speed is greater than a defined slow-to-normal threshold, but less than a specified normal-to-fast threshold. The counter is updated by one for each valid transition on either channel as in slow mode (see [Figure 1](#)) but the direction is not calculated in normal mode. The last direction calculated in the slow mode is kept.

### 3.1.3 Fast Mode

This mode applies when the motion system speed is greater than a defined normal-to-fast threshold. In the fast mode, only one channel is serviced. The position counter is updated by 4 each fourth transition (see [Figure 2](#)). The other three other transitions are ignored. In fast mode, the eTPU can reliably decode at more than quadruple the maximum count rate of normal mode. No direction decoding is done in fast mode; the counter is updated in the same direction as when the last transition was serviced in slow mode.

The function should run in fast mode until the motion system speed falls below a fast-to-normal threshold. Then the mode is switched back to the normal. The speed thresholds must be determined by the overall eTPU system activity, and evaluated for each application. When all thresholds are set to zero, the QD function runs in slow mode only.

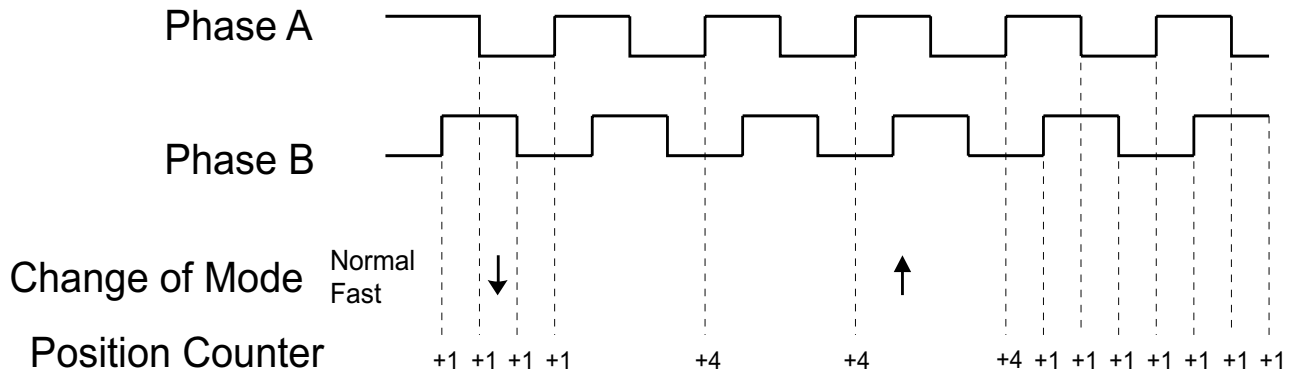


Figure 2. Fast Mode of Operation

### 3.2 Noise Immunity

The QD function uses an acceptance windows, defined in time, within which the next transition is expected. The window open and close times are calculated based on the last leading period (see 3.4) and given coefficients (window\_ratio1, window\_ratio2), which should be determined based on the maximum motion system acceleration and deceleration.

If a transition comes prior to the acceptance window open time, the transition is not accepted. If a transition comes within the acceptance window, the transition is accepted. If no transition comes up to the window close time, a transition is inserted.

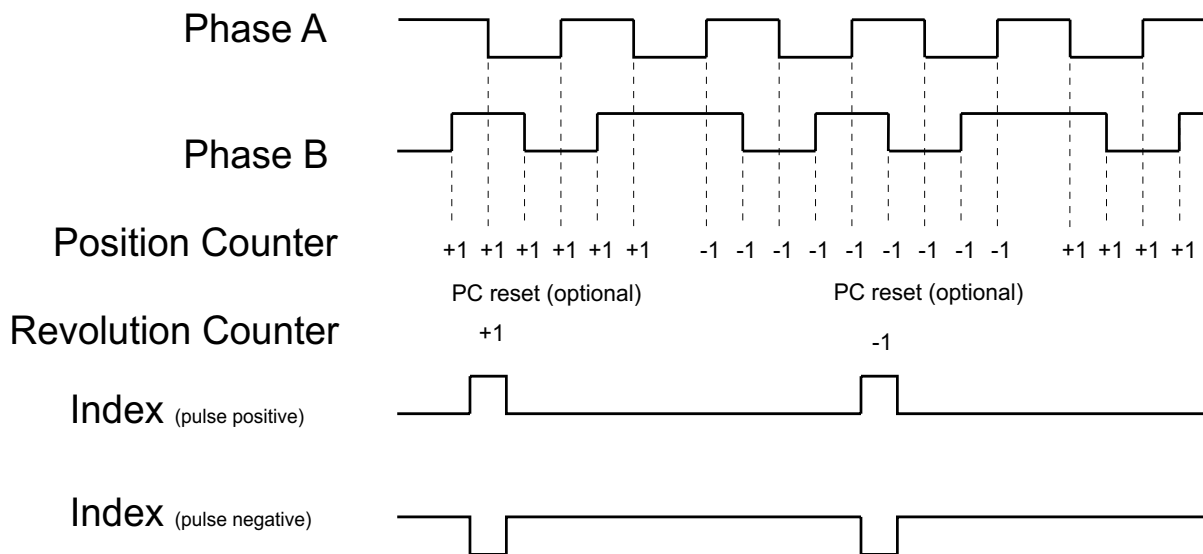
The acceptance windows are only in normal and fast modes used. In the slow mode, all transitions are accepted, and no transition is inserted.

### 3.3 Automatic Reset of Position Counter

When the QD is used in configuration without an index channel, the position counter can be automatically reset each revolution. The user can optionally set a maximum value for the position counter (PC): when the absolute value of PC reaches this maximum, it is automatically reset to zero. The maximum value for the position counter (PC) must be divisible by 4.

### 3.4 Index Channel

The QD index function can monitor an input signal that indicates each revolution of the motion system with a pulse. The function can be configured for the index pulse of positive or negative polarity. When the pulse is detected, the index function increments or decrements the revolution Counter by 1, depending on the current direction. Optionally, it can also reset the position counter.



**Figure 3. Index Signal**

Based on the phase A and phase B pin states at the time when the index pulse is active, there are two types of QD configuration:

- CONFIGURATION 0 - both phase A and phase B pins are 0 when the index pulse is active
- CONFIGURATION 1 - both phase A and phase B pins are 1 when the index pulse is active

This parameter together with the actual motion system direction determines which of four primary and secondary channel edges is the leading edge. The leading edge is the only edge detected in the fast mode. The time between two leading edges is called leading period. [Figure 4](#) and [Figure 5](#) describes leading edge occurrence for both motion system directions.

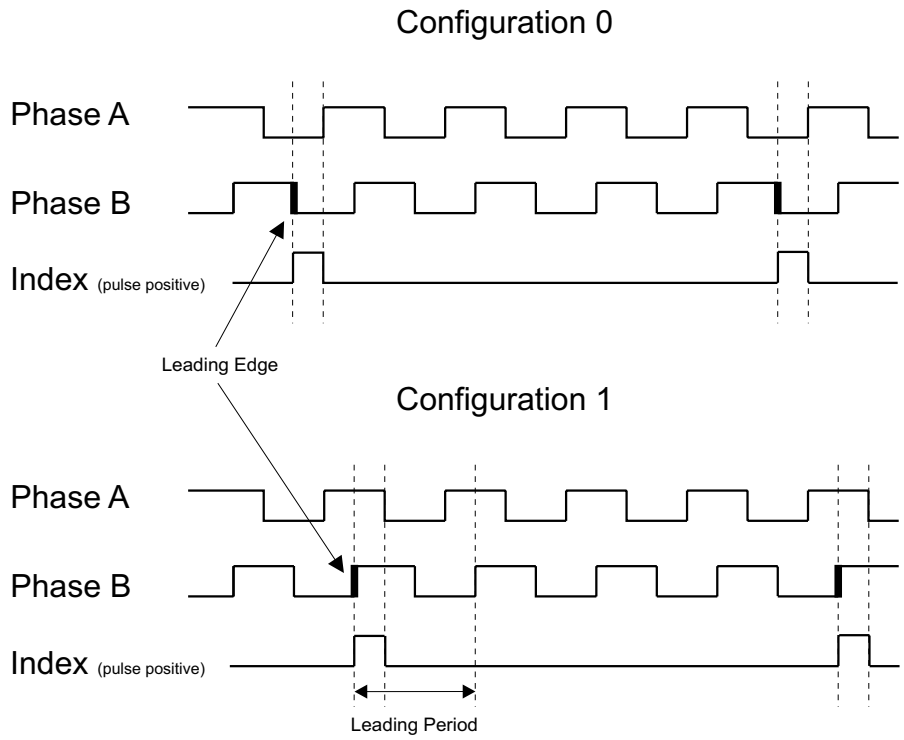


Figure 4. Leading Edge Occurrence for the Incremental Direction

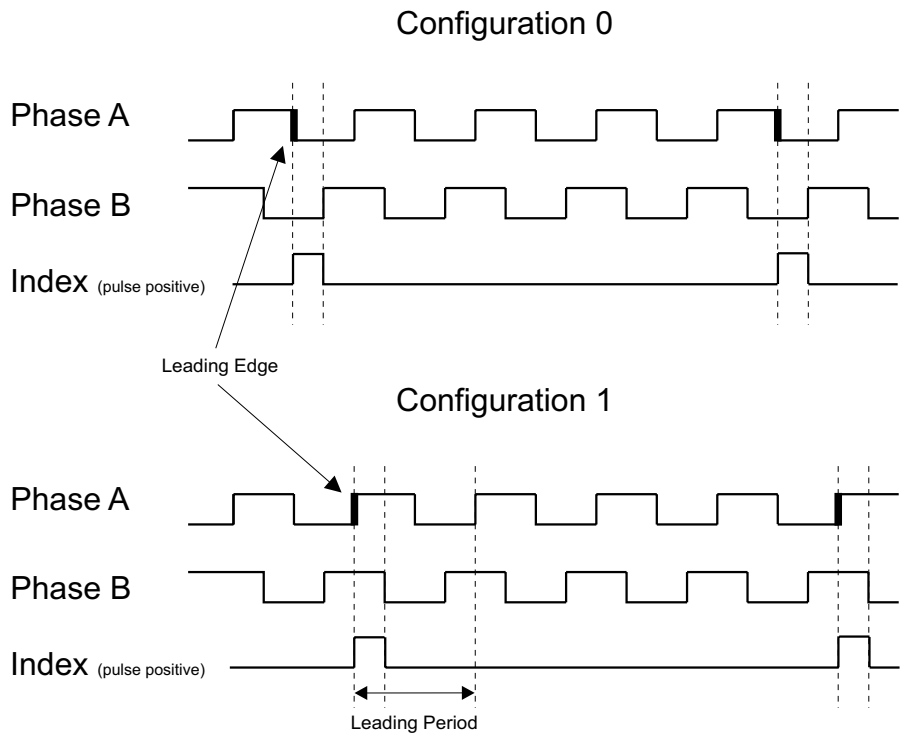


Figure 5. Leading Edge Occurrence For The Decremental Direction

## 3.5 Home Channel

The QD home function can monitor an input signal, which indicates the home position of the motion system by a pulse. The function can be configured to react on either a low-high transition, a high-low transition, or any transition. In this way, the user can select whether the home signal is of positive or negative polarity, and the action to be taken when the home position is either arrived at, left, or both.

When the specified transition is detected, the QD home function resets the position counter and the revolution counter.

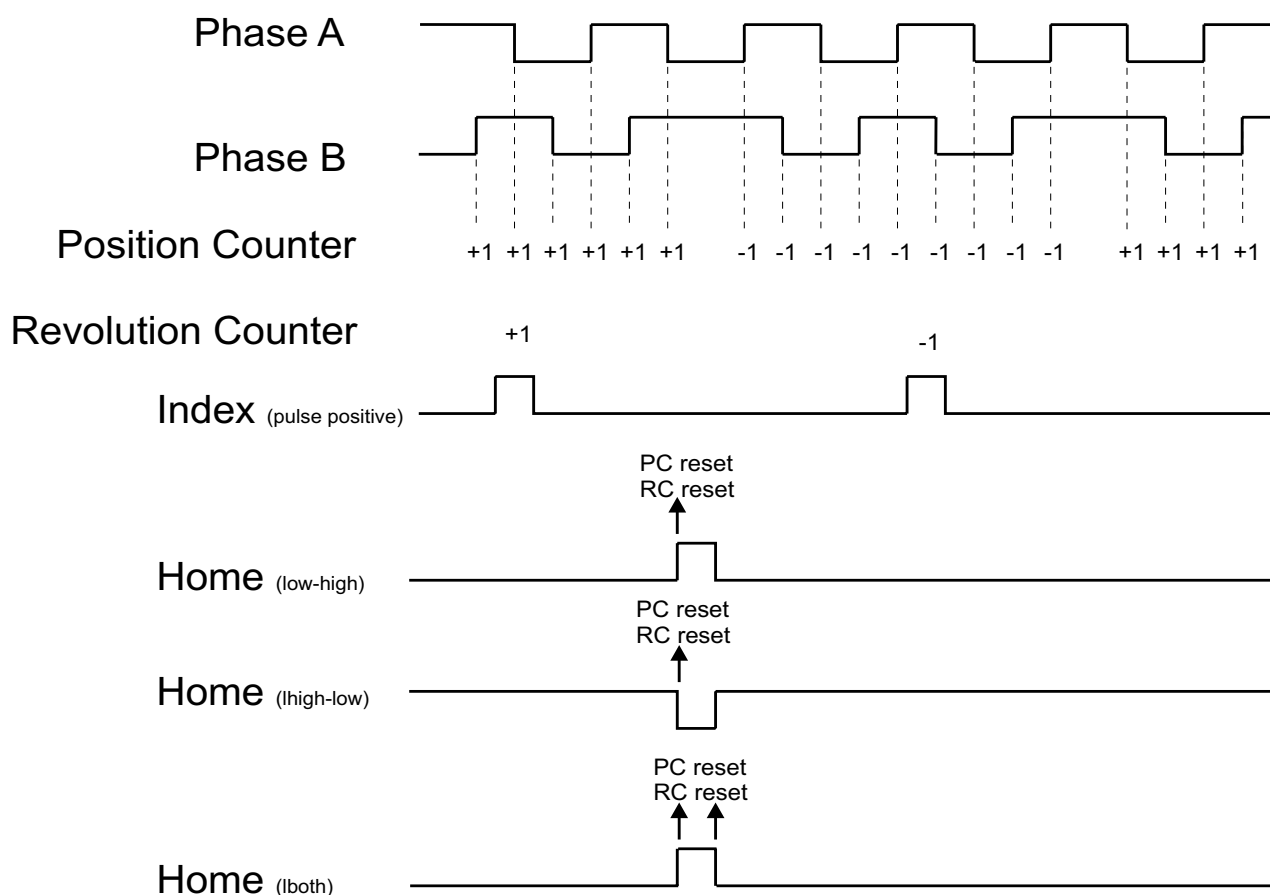


Figure 6. Home Signal

## 3.6 Angle Interrupt

The QD function can be configured to generate an interrupt when the position counter reaches one of two defined values.

## 3.7 Performance

Like all eTPU functions, the QD function performance in an application is to some extent dependent upon the service time (latency) of other active eTPU channels. This is due to the operational nature of the scheduler.

The influence of the QD function on the overall eTPU performance can be expressed by the following parameter:

- maximum eTPU busy-time per one encoder pulse**  
 This value determines the eTPU time necessary for processing all transitions which could come within one encoder pulse.

Table 2 lists the maximum eTPU busy-times per one encoder pulse in eTPU cycles, depending on the mode of operation.

**Table 2. Maximum eTPU Busy-time per One Encoder Pulse**

Mode	Maximum Etpu Busy-time Per One Qd Pulse [Etpu Cycles]
Slow Mode - phase A + phase B	624
Slow Mode - phase A + phase B + Index	720
Normal Mode - phase A + phase B	672
Normal Mode - phase A + phase B + Index	732
Fast Mode - phase A + phase B	246
Fast Mode - phase A + phase B + Index	310

The eTPU module clock is equal to the CPU clock on MPC5500 devices, as well as the peripheral clock (half of the CPU clock) on MCF523x devices. For example, the eTPU module clock is 132 MHz on a 132-MHz MPC5554, and one eTPU cycle takes 7.58ns. The eTPU module clock is only 75 MHz on a 150-MHz MCF5235, and one eTPU cycle takes 13.33ns.

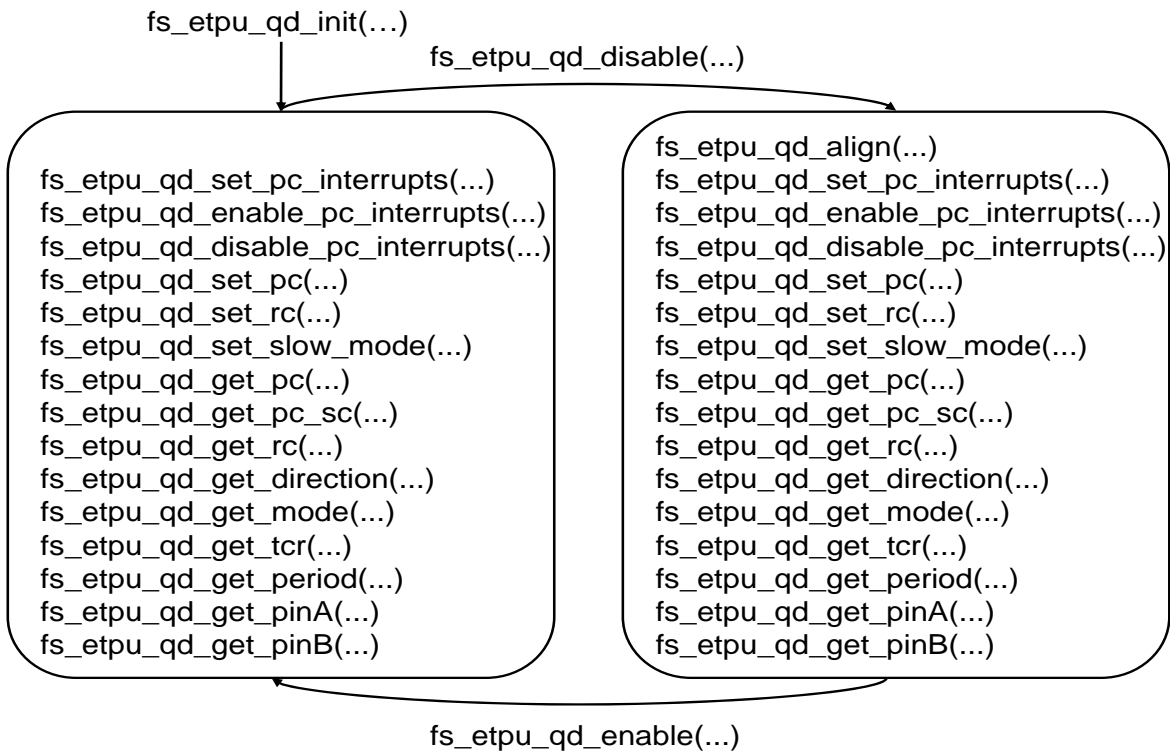
The performance is influenced by compiler efficiency. The above numbers, that were measured on the code compiled by eTPU compiler version 1.0.0.5, are given for guidance only and are subject to change. For up to date information refer to the information provided in the particular eTPU function set release available from Freescale.

## 4 C Level API for Function

The following routines provide easy access, for the application developer, to the QD function. Use of these functions eliminates the need to directly control the eTPU registers. There are 19 functions added to the application programming interface (API). The routines can be found in the `etpu_qd.h` and `etpu_qd.c` files, which should be included in the link file along with the top level development file(s).



Figure 7 shows the QD API state flow and lists API functions which can be used in each of its states.



**Figure 7. QD API State Flow**

All QD API routines will be described in order and are listed below:

- Initialization Functions:

```

int32_t fs_etpu_qd_init( uint8_t  channel_primary,
                        uint8_t  channel_home,
                        uint8_t  channel_index,
                        uint8_t  signals,
                        uint8_t  priority,
                        uint8_t  configuration,
                        uint8_t  timer,
                        uint24_t  pc_max,
                        uint24_t  slow_normal_threshold,
                        uint24_t  normal_slow_threshold,
                        uint24_t  normal_fast_threshold,
                        uint24_t  fast_normal_threshold,
                        fract24_t  window_ratio1,
  
```

```
fract24_t window_ratio2,  
uint8_t  home_transition,  
uint8_t  index_pulse,  
uint8_t  index_pc_reset,  
uint32_t etpu_tcr_freq,  
uint24_t pc_per_rev)
```

- **Change Operation Functions:**

```
int32_t fs_etpu_qd_disable( uint8_t channel_primary,  
                             uint8_t channel_home,  
                             uint8_t channel_index,  
                             uint8_t signals)  
  
int32_t fs_etpu_qd_enable( uint8_t channel_primary,  
                            uint8_t channel_home,  
                            uint8_t channel_index,  
                            uint8_t signals,  
                            uint8_t priority)  
  
int32_t fs_etpu_qd_align(uint8_t channel_primary,  
                          int24_t pc)  
  
int32_t fs_etpu_qd_set_pc_interrupts(uint8_t channel_primary,  
                                     int24_t pc_interrupt1,  
                                     int24_t pc_interrupt2)  
  
int32_t fs_etpu_qd_enable_pc_interrupts(uint8_t channel_primary)  
int32_t fs_etpu_qd_disable_pc_interrupts(uint8_t channel_primary)  
int32_t fs_etpu_qd_set_pc(uint8_t channel_primary,  
                           int24_t pc)  
  
int32_t fs_etpu_qd_set_rc(uint8_t channel_primary,  
                           int24_t rc)  
  
int32_t fs_etpu_qd_set_slow_mode(uint8_t channel_primary)
```

- **Value Return Functions:**

```
int24_t fs_etpu_qd_get_pc(uint8_t channel_primary)  
int24_t fs_etpu_qd_get_pc_sc(uint8_t channel_primary)  
int24_t fs_etpu_qd_get_rc(uint8_t channel_primary)  
int8_t  fs_etpu_qd_get_direction(uint8_t channel_primary)  
uint8_t fs_etpu_qd_get_mode(uint8_t channel_primary)  
uint24_t fs_etpu_qd_get_tcr(uint8_t channel_primary)
```

```
uint24_t fs_etpu_qd_get_period(uint8_t channel_primary)
uint8_t fs_etpu_qd_get_pinA(uint8_t channel_primary)
uint8_t fs_etpu_qd_get_pinB(uint8_t channel_primary)
```

## 4.1 Initialization Function

### 4.1.1 int32\_t fs\_etpu\_qd\_init(...)

This routine is used to initialize the eTPU channels for the QD function. This function has the following parameters:

- **channel\_primary (uint8\_t)** - This is the primary channel number (phase A). The secondary channel (phase B) is always a channel one higher. This parameter should be assigned a value of 0-31 for ETPU\_A, and 64-95 for ETPU\_B.
- **channel\_home (uint8\_t)** - If a home signal is processed, this is the home channel number. This parameter should be assigned a value of 0-31 for ETPU\_A, and 64-95 for ETPU\_B. This parameter is ignored if the parameter signal is set to FS\_ETPU\_QD\_PRIM\_SEC, or FS\_ETPU\_QD\_PRIM\_SEC\_INDEX.
- **channel\_index (uint8\_t)** - If an index signal is processed, this is the index channel number. This parameter should be assigned a value of 0-31 for ETPU\_A, and 64-95 for ETPU\_B. This parameter is ignored if grouping is set to FS\_ETPU\_QD\_PRIM\_SEC, or FS\_ETPU\_QD\_PRIM\_SEC\_HOME.
- **signals (uint8\_t)** - This parameter determines which QD signals are used. This parameter should be assigned a value of:
  - FS\_ETPU\_QD\_PRIM\_SEC,
  - FS\_ETPU\_QD\_PRIM\_SEC\_INDEX,
  - FS\_ETPU\_QD\_PRIM\_SEC\_HOME or
  - FS\_ETPU\_QD\_PRIM\_SEC\_INDEX\_HOME.
- **priority (uint8\_t)** - This is the priority to assign to the QD function. This parameter should be assigned a value of:
  - FS\_ETPU\_PRIORITY\_HIGH,
  - FS\_ETPU\_PRIORITY\_MIDDLE,
  - FS\_ETPU\_PRIORITY\_LOW or
  - FS\_ETPU\_PRIORITY\_DISABLED.
- **configuration (uint8\_t)** - This is the configuration of channels. This parameter should be assigned a value of:
  - FS\_ETPU\_QD\_CONFIGURATION\_0 (both QD pins are 0 when the INDEX pulse is active)
  - FS\_ETPU\_QD\_CONFIGURATION\_1 (both QD pins are 1 when the INDEX pulse is active)

- **timer (uint8\_t)** - This is the timer to use as a reference for the QD signals. The eTPU timer definitions are defined in *etpu\_util.h*:
  - FS\_ETPU\_TCR1
  - FS\_ETPU\_TCR2
- **pc\_max (uint24\_t)** - Maximum value of the position counter. This parameter is optional and can be set to zero; then the automatic position counter reset is not performed.
- **slow\_normal\_threshold (uint24\_t)** - This is the threshold for automatic switching from slow mode to normal mode, in [rpm].
- **normal\_slow\_threshold (uint24\_t)** - This is the threshold for automatic switching from normal mode to slow mode, in [rpm].
- **normal\_fast\_threshold (uint24\_t)** - This is the threshold for automatic switching from normal mode to fast mode, in [rpm].
- **fast\_normal\_threshold (uint24\_t)** - This is the threshold for automatic switching from fast mode to normal mode, in [rpm]. (When all thresholds are set to zero QD function runs in slow mode only.)
- **window\_ratio1 (fract24\_t)** - This is the ratio which applies when scheduling the window beginning (in normal and fast mode). This parameter determines how much time prior the expected next QD edge the window opens. This is a fractional value in format (9.15) (0x00800000 corresponds to 1.0) and should be assigned a value between 0.5 (0x00400000) and 0.9 (0x00733333).
- **window\_ratio2 (fract24\_t)** - This is the ratio which applies when scheduling the window ending (in normal and fast mode). This parameter determines how much time after the expected next QD edge the window opens. This is a fractional value in format (9.15) (0x00800000 corresponds to 1.0) and should be assigned a value between 1.1 (0x008CCCCC) and 1.5 (0x00C00000).
- **home\_transition (uint8\_t)** - This parameter selects the type of home signal transition to detect. This parameter should be assigned a value of:
  - FS\_ETPU\_QD\_HOME\_TRANS\_LOW\_HIGH
  - FS\_ETPU\_QD\_HOME\_TRANS\_HIGH\_LOW
  - FS\_ETPU\_QD\_HOME\_TRANS\_ANY
 This parameter is ignored if Home channel is not used.
- **index\_pulse (uint8\_t)** - This parameter selects the polarity of the of index signal pulse. This parameter should be assigned a value of:
  - FS\_ETPU\_QD\_INDEX\_PULSE\_POSITIVE
  - FS\_ETPU\_QD\_INDEX\_PULSE\_NEGATIVE
 This parameter is ignored if Index channel is not used.

- **index\_pc\_reset (uint8\_t)** - This parameter selects an action to perform on index signal transition detection. This parameter should be assigned to a value of:
  - FS\_ETPU\_QD\_INDEX\_PC\_NO\_RESET
  - FS\_ETPU\_QD\_INDEX\_PC\_RESET
 This parameter is ignored if index channel is not used.
- **etpu\_tcr\_freq (uint32\_t)** - This is the frequency of TCR module used by QD channels, in [Hz].
- **qd\_pc\_per\_rev (uint24\_t)** - This is the number of QD position counter increments per revolution.

## 4.2 Change Operation Functions

### 4.2.1 int32\_t fs\_etpu\_qd\_disable( uint8\_t channel\_primary, uint8\_t channel\_home, uint8\_t channel\_index, uint8\_t signals)

This function disables the eTPU quadrature decoder channels. This function has the following parameters:

- **channel\_primary (uint8\_t)** - This is the primary channel number (phase A). This parameter must be assigned the same value as was assigned to the *channel\_primary* parameter in the initialization routine.
- **channel\_home (uint8\_t)** - If a home signal is processed, this is the home channel number. This parameter must be assigned the same value as was assigned to the *channel\_home* parameter in the initialization routine.
- **channel\_index (uint8\_t)** - If an index signal is processed, this is the index channel number. This parameter must be assigned the same value as was assigned to the *channel\_index* parameter in the initialization routine.
- **signals (uint8\_t)** - This parameter determines which QD signals are used. This parameter should be assigned a value of:
  - FS\_ETPU\_QD\_PRIM\_SEC,
  - FS\_ETPU\_QD\_PRIM\_SEC\_INDEX,
  - FS\_ETPU\_QD\_PRIM\_SEC\_HOME or
  - FS\_ETPU\_QD\_PRIM\_SEC\_INDEX\_HOME.

### 4.2.2 `int32_t fs_etpu_qd_enable( uint8_t channel_primary, uint8_t channel_home, uint8_t channel_index, uint8_t signals, uint8_t priority)`

This function enables the eTPU quadrature decoder channels. This function has the following parameters:

- **channel\_primary (uint8\_t)** - This is the primary channel number (phase A). This parameter must be assigned the same value as was assigned to the *channel\_primary* parameter in the initialization routine.
- **channel\_home (uint8\_t)** - If a home signal is processed, this is the home channel number. This parameter must be assigned the same value as was assigned to the *channel\_home* parameter in the initialization routine.
- **channel\_index (uint8\_t)** - If an index signal is processed, this is the index channel number. This parameter must be assigned the same value as was assigned to the *channel\_index* parameter in the initialization routine.
- **signals (uint8\_t)** - This parameter determines which QD signals are used. This parameter should be assigned a value of:
  - FS\_ETPU\_QD\_PRIM\_SEC,
  - FS\_ETPU\_QD\_PRIM\_SEC\_INDEX,
  - FS\_ETPU\_QD\_PRIM\_SEC\_HOME or
  - FS\_ETPU\_QD\_PRIM\_SEC\_INDEX\_HOME.
- **priority (uint8\_t)** - This is the priority to assign to the QD channels. This parameter should be assigned a value of:
  - FS\_ETPU\_PRIORITY\_HIGH,
  - FS\_ETPU\_PRIORITY\_MIDDLE or
  - FS\_ETPU\_PRIORITY\_LOW.

### 4.2.3 `int32_t fs_etpu_qd_align(uint8_t channel_primary, int24_t pc)`

This function changes the position counter value to ensure its divisibility by 4 when leading edge occurs. Leading edge is the one of four primary and secondary channel edges which is detected in fast mode, incrementing position counter by 4. This depends on QD Configuration, and of the QD primary and secondary pin states. Position counter dedicated for speed controller (SC) is reset to 0. This function has the following parameters:

- **channel\_primary (uint8\_t)** - This is the primary channel number (phase A). This parameter must be assigned the same value as was assigned to the *channel\_primary* parameter in the initialization routine.
- **pc (int24\_t)** - This is the position counter value to be set. The actual value set is in range from  $pc-1$  to  $pc+2$ , to ensure position counter divisibility by 4 on the leading edge.

#### 4.2.4 `int32_t fs_etpu_qd_set_pc_interrupts(uint8_t channel_primary, int24_t pc_interrupt1, int24_t pc_interrupt2)`

This function changes the `pc_interrupt` values. This function has the following parameters:

- **`channel_primary (uint8_t)`** - This is the primary channel number (phase A). This parameter must be assigned the same value as was assigned to the `channel_primary` parameter in the initialization routine.
- **`pc_interrupt1 (int24)`** - This is the `pc_interrupt1` value to be set. This value must be divisible by 4.
- **`pc_interrupt2 (int24)`** - This is the `pc_interrupt2` value to be set. This value must be divisible by 4.

#### 4.2.5 `int32_t fs_etpu_qd_enable_pc_interrupts(uint8_t channel_primary)`

This function enables the `pc_interrupt` processing. This function has the following parameter:

- **`channel_primary (uint8_t)`** - This is the primary channel number (phase A). This parameter must be assigned the same value as was assigned to the `channel_primary` parameter in the initialization routine.

#### 4.2.6 `int32_t fs_etpu_qd_disable_pc_interrupts(uint8_t channel_primary)`

This function disables the `pc_interrupt` processing. This function has the following parameter:

- **`channel_primary (uint8_t)`** - This is the primary channel number (phase A). This parameter must be assigned the same value as was assigned to the `channel_primary` parameter in the initialization routine.

#### 4.2.7 `int32_t fs_etpu_qd_set_pc(uint8_t channel_primary, int24_t pc)`

This function changes the position counter value. This function has the following parameters:

- **`channel_primary (uint8_t)`** - This is the primary channel number (phase A). This parameter must be assigned the same value as was assigned to the `channel_primary` parameter in the initialization routine.
- **`pc (int24)`** - This is the position counter value to be set.

### 4.2.8 `int32_t fs_etpu_qd_set_rc(uint8_t channel_primary, int24_t rc)`

This function changes the revolution counter value. This function has the following parameters:

- **channel\_primary (uint8\_t)** - This is the primary channel number (phase A). This parameter must be assigned the same value as was assigned to the *channel\_primary* parameter in the initialization routine.
- **rc (int24)** - This is the revolution counter value to be set.

### 4.2.9 `int32_t fs_etpu_qd_set_slow_mode(uint8_t channel_primary)`

This function changes the QD mode to slow. This function has the following parameter:

- **channel\_primary (uint8\_t)** - This is the primary channel number (phase A). This parameter must be assigned the same value as was assigned to the *channel\_primary* parameter in the initialization routine.

## 4.3 Value Return Functions

### 4.3.1 `int24_t fs_etpu_qd_get_pc(uint8_t channel_primary)`

This function returns the position counter value. This function has the following parameter:

- **channel\_primary (uint8\_t)** - This is the primary channel number (phase A). This parameter must be assigned the same value as was assigned to the *channel\_primary* parameter in the initialization routine.

The position counter value is returned as an `int24_t`.

### 4.3.2 `int24_t fs_etpu_qd_get_pc_sc(uint8_t channel_primary)`

This function returns the position counter for speed controller (SC) value. Position counter for SC is incremented and decremented the same way as the position counter, but neither the index pulse nor the automatic reset at `pc_max` reset position counter for SC. The SC function resets it on its own.

This function has the following parameter:

- **channel\_primary (uint8\_t)** - This is the primary channel number (phase A). This parameter must be assigned the same value as was assigned to the *channel\_primary* parameter in the initialization routine.

The position counter for SC value is returned as an `int24_t`.



### 4.3.3 `int24_t fs_etpu_qd_get_rc(uint8_t channel_primary)`

This function returns the revolution counter value. This function has the following parameter:

- **channel\_primary (uint8\_t)** - This is the primary channel number (phase A). This parameter must be assigned the same value as was assigned to the *channel\_primary* parameter in the initialization routine.

The revolution counter value is returned as an `int24_t`.

### 4.3.4 `int8_t fs_etpu_qd_get_direction(uint8_t channel_primary)`

This function returns the motion system direction. This function has the following parameter:

- **channel\_primary (uint8\_t)** - This is the primary channel number (phase A). This parameter must be assigned the same value as was assigned to the *channel\_primary* parameter in the initialization routine.

The returned value can be one of:

- `FS_ETPU_QD_DIRECTION_INC`
- `FS_ETPU_QD_DIRECTION_DEC`

### 4.3.5 `uint8_t fs_etpu_qd_get_mode(uint8_t channel_primary)`

This function returns the mode of QD operation. This function has the following parameter:

- **channel\_primary (uint8\_t)** - This is the primary channel number (phase A). This parameter must be assigned the same value as was assigned to the *channel\_primary* parameter in the initialization routine.

The mode of QD operation is returned as an `uint8_t`. The returned value can be one of:

- `FS_ETPU_QD_MODE_SLOW` (1)
- `FS_ETPU_QD_MODE_NORMAL` (2)
- `FS_ETPU_QD_MODE_FAST` (4)

### 4.3.6 `uint24_t fs_etpu_qd_get_tcr(uint8_t channel_primary)`

This function returns the TCR time of the last detected transition. This function has the following parameter:

- **channel\_primary (uint8\_t)** - This is the primary channel number (phase A). This parameter must be assigned the same value as was assigned to the *channel\_primary* parameter in the initialization routine.

The TCR time of the last detected transition is returned as an `uint24_t`.

### 4.3.7 uint24\_t fs\_etpu\_qd\_get\_period( uint8\_t channel\_primary)

This function returns the QD period. This function has the following parameter:

- **channel\_primary (uint8\_t)** - This is the primary channel number (phase A). This parameter must be assigned the same value as was assigned to the *channel\_primary* parameter in the initialization routine.

The period value is returned as an `uint24_t`.

### 4.3.8 uint8\_t fs\_etpu\_qd\_get\_pinA(uint8\_t channel\_primary)

This function returns the current state of primary (phase A) input signal. This function has the following parameter:

- **channel\_primary (uint8\_t)** - This is the primary channel number (phase A). This parameter must be assigned the same value as was assigned to the *channel\_primary* parameter in the initialization routine.

The pin state is returned as an `uint8_t`.

### 4.3.9 uint8\_t fs\_etpu\_qd\_get\_pinB(uint8\_t channel\_primary)

This function returns the current state of secondary (phase B) input signal. This function has the following parameter:

- **channel\_primary (uint8\_t)** - This is the primary channel number (phase A). This parameter must be assigned the same value as was assigned to the *channel\_primary* parameter in the initialization routine.

The pin state is returned as an `uint8_t`.

## 4.4 TPU-Compatible C Level API for Function

The TPU3-compatible API provides backward-compatibility from eTPU to TPU3. The following functions allow control of the eTPU QD function using the TPU3 FQD API function calls.

- Initialization Function:

```
void tpu_fqd_init(struct TPU3_tag *tpu,
                 UINT8 channel,
                 UINT8 priority,
                 INT16 init_position)
```

- Value Return Functions:

```
UINT8 tpu_fqd_current_mode(struct TPU3_tag *tpu,
                          UINT8 channel)
INT16 tpu_fqd_position(struct TPU3_tag *tpu,
                      UINT8 channel)
```

```
void tpu_fqd_data(struct TPU3_tag *tpu,
                 UINT8 channel, INT16 *tcrl,
                 INT16 *edge,
                 INT16 *primary_pin,
                 INT16 *secondary_pin)
```

#### NOTE

tpu\_fqd\_init\_trans\_count(...) and tpu\_fqd\_mode(...) functions are not implemented!

## 5 Example Use of Function

### 5.1 Demo Application

The usage of the QD eTPU function is demonstrated in the application “BLDC Motor with Quadrature Encoder and Speed Closed Loop, driven by eTPU on MCF523x.” For a detailed description of the demo application, refer to the application note AN2957.

#### 5.1.1 Function Calls

```
/* initialize the QD function */
err_code = fs_etpu_qd_init( QD0_PHASEA, /* engine: A; channel: 1 */
                           ETPU_CHAN_NOT_USED, /* Home channel is not used */
                           QD0_INDEX, /* engine: A; channel: 3 */
                           FS_ETPU_QD_PRIM_SEC_INDEX, /* signals: Index, primary and secondary */
                           FS_ETPU_PRIORITY_HIGH, /* priority: High */
                           FS_ETPU_QD_CONFIGURATION_0, /* configuration: 0 */
                           FS_ETPU_TCR1, /* timer: TCR1 */
                           0, /* pc_max: 0 */
                           0.4 * SPEED_MAX_RPM, /* slow_normal_threshold */
                           0.35 * SPEED_MAX_RPM, /* normal_slow_threshold */
                           0.8 * SPEED_MAX_RPM, /* normal_fast_threshold */
                           0.75 * SPEED_MAX_RPM, /* fast_normal_threshold */
                           0x733333, /* window ratio 1: 0.9 * 2^23 */
                           0x8CCCCC, /* window ratio 2: 1.1 * 2^23 */
                           FS_ETPU_QD_HOME_TRANS_LOW_HIGH, /* Detect. of low-high transition on Home chan.*/
                           FS_ETPU_QD_INDEX_PULSE_POSITIVE, /* Index pulse of positive polarity */
                           FS_ETPU_QD_INDEX_PC_NO_RESET, /* PC is NOT reset to pc_init on Index transition */
```

## Example Use of Function

```
FS_ETPU_QD_ETPU_A_TCR1_FREQ, /* etpu_tcr_freq: frequency of eTPU engine A - TCR1 */
2000) /* qd_pc_per_rev: 4*500 */

/* disable QD channels */
fs_etpu_qd_disable( QD0_PHASEA,
                   QD0_PHASEB,
                   ETPU_CHAN_NOT_USED,
                   QD0_INDEX)

/* Align the QD position counter */
fs_etpu_qd_align(QD0_PHASEA, 0)

/* Enable QD channels */
fs_etpu_qd_enable( QD0_PHASEA,
                  QD0_PHASEB,
                  ETPU_CHAN_NOT_USED,
                  QD0_INDEX,
                  FS_ETPU_PRIORITY_MIDDLE);

/* set QD pc_interrupt values */
fs_etpu_qd_set_pc_interrupts(QD0_PHASEA, pc_interrupt1, pc_interrupt2);

/* enable QD pc_interrupt processing */
fs_etpu_qd_enable_pc_interrupts(QD0_PHASEA);

/* set QD mode to slow */
fs_etpu_qd_set_slow_mode(QD0_PHASEA);
```

## 6 Summary and Conclusions

This application note provides the user with a description of the quadrature decoder (QD) eTPU function usage and examples. The simple C interface routines to the QD eTPU function enable easy implementation of the QD in applications. The demo application is targeted at the MCF523x family of devices, but it could be easily reused with any device that has an eTPU.

## 7 References

1. “The Essential of Enhanced Time Processing Unit,” AN2353.
2. “General C Functions for the eTPU,” AN2864.
3. “Using the DC Motor Control eTPU Function Set (set3),” AN2958.
4. “Using the AC Motor Control eTPU Function Set (set4),” AN2968
5. Enhanced Time Processing Unit Reference Manual, ETPURM/D.
6. eTPU Graphical Configuration Tool, <http://www.freescale.com/etpu>, ETPUGCT.
7. “BLDC Motor with Quadrature Encoder and Speed Closed Loop, driven by eTPU on MCF523x,” AN2957.

**THIS PAGE INTENTIONALLY LEFT BLANK**

---

**THIS PAGE INTENTIONALLY LEFT BLANK**

## How to Reach Us:

### Home Page:

[www.freescale.com](http://www.freescale.com)

### E-mail:

[support@freescale.com](mailto: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](mailto: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](mailto: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](mailto: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 2666 8080  
[support.asia@freescale.com](mailto: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](mailto:LDCForFreescaleSemiconductor@hibbertgroup.com)

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals", must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.



Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. © Freescale Semiconductor, Inc. 2005. All rights reserved.

Document Number: AN2842  
Rev. 0  
04/2005