Freescale Semiconductor
Application Note

AN2852
Rev. 0, 06/2005

Using the Output Compare (OC) eTPU Function

by: Geoff Emerson
Microcontroller Division

Table Of Contents

This output compare (OC) application note is intended to describe simple C interface routines to the OC eTPU function. The function can be used on any product which has an eTPU module. Example code is available for the MPC5554 device. This application note should be read in conjunction with application note AN2864, “General C Functions for the eTPU.”

1 Function Overview

The OC function generates a single edge or a single pulse. Two events are scheduled at offset counts relative to a reference value (these are called event times). Each event occurs when its event time is greater than or equal to the value of a chosen timebase. The timebase is selected from either of two free running timer counter registers (TCR1/2). When each event occurs, a programmed pin action for that event takes place. The actual times at which each scheduled event occurs can be captured from either of the timer counter registers (TCR1/2). The initial

pin state is programmable and the reference value is selectable. There are three reference modes: immediate reference mode, address reference mode, and value mode.

The eTPU OC function is based on the OC TPU function. The OC eTPU function offers the following enhancements over the OC TPU function:

- 22-bit offset values are supported (versus 15-bit offset value on the TPU3).
- The dual action hardware of the eTPU is used to allow two actions to be programmed. This means that a pulse or a single transition can be accommodated.
- When generating a pulse, the dual action functionality of the eTPU is used to allow pulse edges to be one TCR count apart.

TPU3 continuous pulse mode is not supported. Similar functionality can be achieved with the use of the eTPU QOM function. See application note AN2857: "Using the Queued Output Match (QOM) eTPU Function."

2 Functional Description

Offset1/2 are relative match offsets, not absolute match values. The first match time is calculated by adding Offset1 to a reference value. The second match time is calculated by adding Offset2 to the first match time.

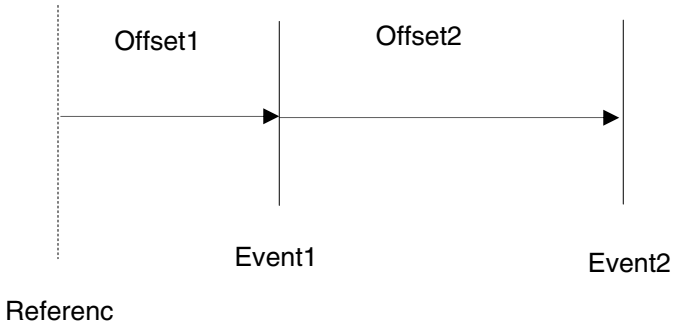


Figure 1. References, Events and Offsets

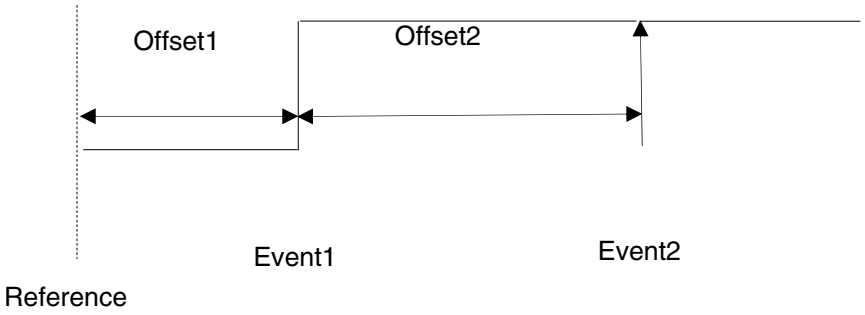
The reference from which the first match in a sequence is scheduled can be the immediate value of the selected timebase (timer count register: TCR), a reference value contained in eTPU data memory, or an absolute TCR value. Using a reference from eTPU data memory allows a transition or pulse to be referenced to a value derived by another eTPU channel.

The pin state when a match occurs is programmable. The pin can be driven high, low, or no change.

The initial pin state, after initialization but before Event1, can be programmed to be high, low, or no change.

Matches are scheduled using both of the eTPU's action units. Each match offset can have a maximum value of 0x40_0000. This allows the second future match to be up to 0x80_0000 TCR counts in the future.

If both pin actions are programmed to be the same, then an edge rather than a pulse will be generated. In the example below, the pin actions for event1 and event2 are set to drive the pin high.



2.1 Initialization Reference Modes

There are three reference modes in which the function can be initialized:

1) Immediate reference mode
2) Address reference mode
3) Value mode

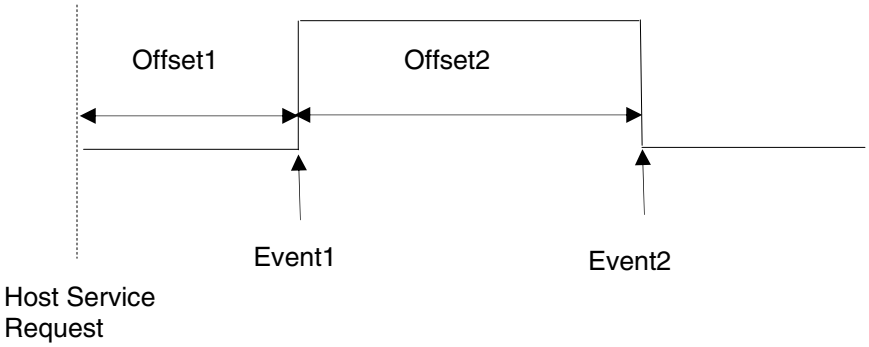
In order to avoid improper operation, the following condition must be met:

Reference + first offset amount + second offset amount < 0x80_0000 + current TCR value

If this condition is not met, then the first, and possibly second, events may be scheduled in the past causing immediate matches to occur.

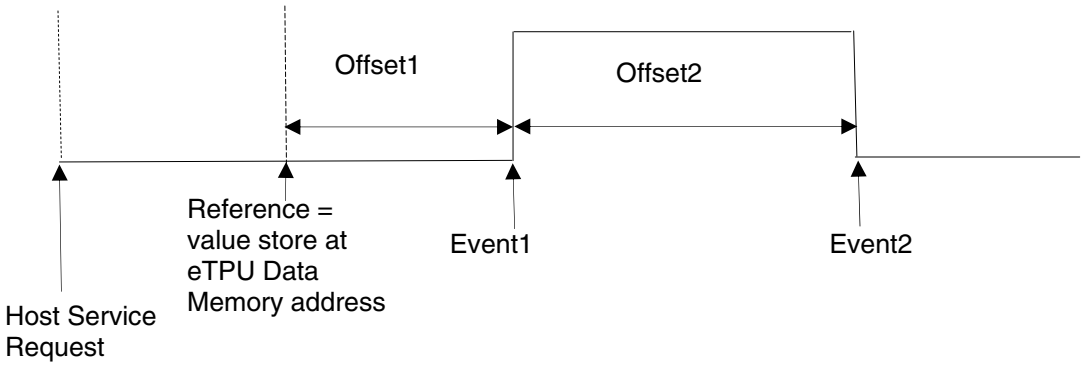
2.1.1 Immediate Reference Mode

In this mode, the function adds offset1 to the current value of the selected timebase to generate the match value for the first event. The match value of the second event is generated by adding offset2 to the first event's match time.



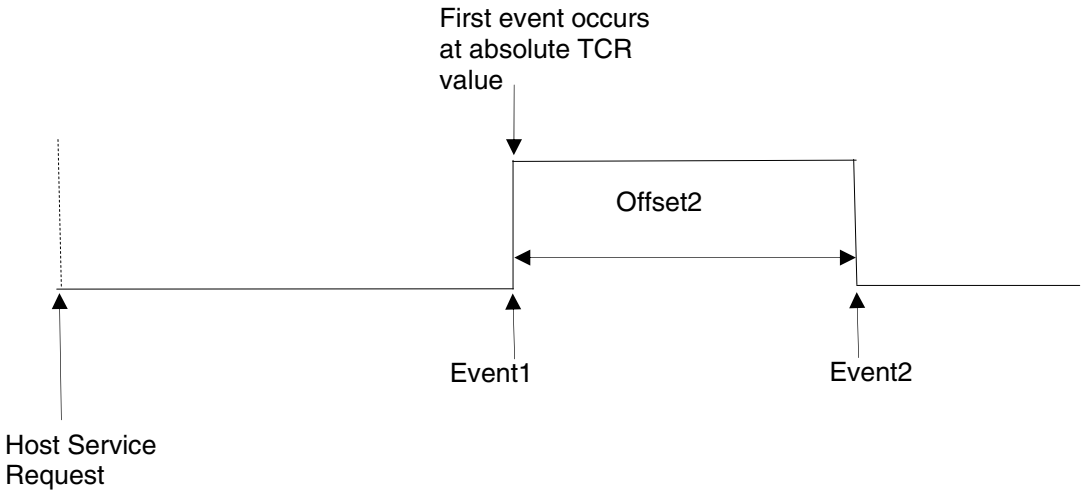
2.1.2 Address Reference Mode

In this mode, the function adds offset1 to a value stored at a memory address to generate the match value for the first event. The match value of the second event is generated by adding offset2 to the first event's match time. Note that the reference value may be either a future value or a value that occurred before the host service request was issued.



2.1.2.1 Value Reference Mode

In this mode, the first event occurs at an absolute TCR value. Offset1 is not used. The match value of the second event is generated by adding offset2 to the first event's match time, i.e. the absolute TCR value.



2.2 Notes on the Performance and Use of the eTPU OC Function

2.2.1 Performance

Like all eTPU functions, the OC 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 eTPU scheduler. The more channels that are active, the more performance decreases. However, worst-case latency in any eTPU application can be closely estimated. To analyze the performance of an application that appears to approach the limits of the eTPU, use the guidelines given in the *eTPU Reference Manual* and the information provided in the eTPU OC software release available from Freescale. In the case of the OC function, the effects of latency will only be apparent in the initialization thread. The initialization thread needs to be completed before the required output pulses are scheduled.

2.2.2 Changing Operation Modes

In order to re-configure the OC function on the channel while it is still running, the channel must first be disabled. This can be done using the fs_etpu_disable function, which can be found in file etpu_utils.h.

2.2.3 Match and Capture Timebases

The match timebase can be either TCR1 or TCR2. The selected match timebase is used for the matching of both events. The capture timebase need not necessarily be the same as the capture timebase. It is possible to capture a different TCR for the first and second events.

3 C Level API for eTPU OC Function

The following routines provide easy access for the user to interface to the OC function. Use of these routines eliminates the need to directly control the eTPU registers. This function can be found in the etpu_oc.h and etpu_oc.c files. The routines are described below and are available from Freescale. In addition, the eTPU compiler generates a file called etpu_oc_auto.h. This file contains information relating to the eTPU OC function, including details on how the eTPU data memory is organized and definitions for various API parameters.

The API consists of 4 functions:

1) Initialization routine - immediate reference: fs_etpu_oc_init_immed
2) Initialization routine - value reference: fs_etpu_oc_init_value
3) Initialization routine - address reference: fs_etpu_oc_init_ref
4) Return match and capture times - fs_etpu_oc_data

3.1 Initialization Routine - Immediate Reference: fs_etpu_oc_init_immed

```
uint8_t fs_etpu_oc_init_immed (uint8_t channel,
                               uint8_t priority,
                               uint8_t match_timebase,
                               uint32_t offset1,
                               uint8_t pin_action_capture_timebase1,
                               uint32_t offset2,
                               uint8_t pin_action_capture_timebase2,
                               uint8_t init_pin)
```

This routine is used to initialize a channel to use the OC function with an immediate TCR reference.

In order for the OC function to run, it needs to use some of the eTPU data memory. There is not any fixed amount of data memory associated with each channel in the eTPU. The memory needs to be allocated in a way that makes sure each channel has its own memory that will not be used by any other channels. There are two ways to allocate this memory: automatically or manually. Using automatic allocation to initialize each channel, it reserves some of the eTPU data memory for its own use. With manual configuration, the eTPU data memory is defined when the system is designed.

Automatic allocation is simpler and is used in all of the examples programs. The routine uses automatic allocation if the channel parameter base address field for a channel is zero. This is the reset condition of the field so normally you don’t need to do anything except call the initialization API routine.

If you call the initialization routine more than once, it will only allocate data memory the first time it is called. The initialization routine will write a value to the channel parameter base address field, so on subsequent calls, it will not allocate more memory.

If the eTPU data memory is allocated manually, then a value must be written to channel parameter base address before the initialization routine is called. This is normally only used if the user wants to pre-define the location of each channels data memory.

After the channel has been initialized, the OC function will be executed as specified. This function has the following parameters:

- Channel (uint8_t): The OC channel number. For devices with two eTPUs, this parameter should be assigned a value of 0-31 for eTPU_A and 64-95 for eTPU_B. For products with a single eTPU, this parameter should be assigned a value of 0-31.
- Priority (uint8_t): The priority to assign to the eTPU OC channel. The following eTPU priority definitions are found in utilities file etpu_utils.h.
  - FS_ETPU_PRIORITY_HIGH
  - FS_ETPU_PRIORITY_MIDDLE
  - FS_ETPU_PRIORITY_LOW

  - FS_ETPU_PRIORITY_DISABLED
- Match_timebase (uint8_t): The timebase which the eTPU OC channel will use for matches. The same timebase will be used for both matches. The following eTPU OC match_timebase definitions are found in the etpu_oc_auto.h file:
  - FS_OC_MATCH_TCR1
  - FS_OC_MATCH_TCR2
- Offset1 (uint32_t): The number of selected TCR counts from the host service request to the first event.
- Offset2 (uint32_t): The number of selected TCR counts from the first match to the second event.
- Pin_action_capture_timebase1/2(unit8_t): The pin action at the first/second match and the capture timebase for those events. The following eTPU OC pin_action_capture_timebase1/2 definitions are found in the etpu_oc_auto.h file:
  - FS_ETPU_OC_PIN_LOW_CAPTURE_TCR1
  - FS_ETPU_OC_PIN_LOW_CAPTURE_TCR2
  - FS_ETPU_OC_PIN_HIGH_CAPTURE_TCR1
  - FS_ETPU_OC_PIN_HIGH_CAPTURE_TCR2
  - FS_ETPU_OC_PIN_TOGGLE_CAPTURE_TCR1
  - FS_ETPU_OC_PIN_TOGGLE_CAPTURE_TCR2
  - FS_ETPU_OC_PIN_NO_CHANGE_CAPTURE_TCR1
  - FS_ETPU_OC_PIN_NO_CHANGE_CAPTURE_TCR2
- Init_pin (uint8_t): The state of the pin from initialization until the first event. The following eTPU OC init_pin definitions are found in the etpu_oc_auto.h file:
  - FS_ETPU_OC_INIT_PIN_LOW
  - FS_ETPU_OC_INIT_PIN_HIGH
  - FS_ETPU_OC_INIT_PIN_NO_CHANGE

3.2 Initialization Routine - Value Reference: fs_etpu_oc_init_value

```
uint8_t fs_etpu_oc_init_value (uint8_t channel,
                               uint8_t priority,
                               uint8_t match_timebase,
                               uint32_t value,
                               uint8_t pin_action_capture_timebase1,
                               uint32_t offset2,
                               uint8_t pin_action_capture_timebase2,
```

uint8_t init_pin)

This routine is used to initialize channel to use the OC function in value reference mode.

In order for the OC function to run, it needs to use some of the eTPU data memory. There is not any fixed amount of data memory associated with each channel in the eTPU. The memory needs to be allocated in a way that makes sure each channel has its own memory that will not be used by any other channels. There are two ways to allocate this memory: automatically or manually. Using automatic allocation to initialize each channel, it reserves some of the eTPU data memory for its own use. With manual configuration, the eTPU data memory is defined when the system is designed.

Automatic allocation is simpler and is used in all of the examples programs. The routine uses automatic allocation if the channel parameter base address field for a channel is zero. This is the reset condition of the field so normally you don't need to do anything except call the initialization API routine.

If you call the initialization routine more than once, it will only allocate data memory the first time it is called. The initialization routine will write a value to the channel parameter base address field, so on subsequent calls, it will not allocate more memory.

If the eTPU data memory is allocated manually, then a value must be written to channel parameter base address before the initialization routine is called. This is normally only used if the user wants to pre-define the location of each channels data memory.

After the channel has been initialized, the OC function will be executed as specified. This function has the following parameters:

- Channel (uint8_t): The OC channel number. For devices with two eTPUs, this parameter should be assigned a value of 0-31 for eTPU_A and 64-95 for eTPU_B. For products with a single eTPU, this parameter should be assigned a value of 0-31.
- Priority (uint8_t): The priority to assign to the eTPU OC channel. The following eTPU priority definitions are found in utilities file etpu_utils.h.
  - FS_ETPU_PRIORITY_HIGH
  - FS_ETPU_PRIORITY_MIDDLE
  - FS_ETPU_PRIORITY_LOW
  - FS_ETPU_PRIORITY_DISABLED
- Match_timebase (uint8_t): The timebase that the eTPU OC channel will use for matches. The same timebase will be used for both matches. The following eTPU OC match_timebase definitions are found in the etpu_oc_auto.h file:
  - FS_OC_MATCH_TCR1
  - FS_OC_MATCH_TCR2
- Value (uint32_t): The absolute selected TCR count at which the first event will be scheduled to occur.
- Offset2 (uint32_t): The number of selected TCR counts from the first match to the second event.

- Pin_action_capture_timebase1/2(unit8_t): The pin action at the first/second match and the capture timebase for those events. The following eTPU OC pin_action_capture_timebase1/2 definitions are found in the etpu_oc_auto.h file:
  - FS_ETPU_OC_PIN_LOW_CAPTURE_TCR1
  - FS_ETPU_OC_PIN_LOW_CAPTURE_TCR2
  - FS_ETPU_OC_PIN_HIGH_CAPTURE_TCR1
  - FS_ETPU_OC_PIN_HIGH_CAPTURE_TCR2
  - FS_ETPU_OC_PIN_TOGGLE_CAPTURE_TCR1
  - FS_ETPU_OC_PIN_TOGGLE_CAPTURE_TCR2
  - FS_ETPU_OC_PIN_NO_CHANGE_CAPTURE_TCR1
  - FS_ETPU_OC_PIN_NO_CHANGE_CAPTURE_TCR2
- Init_pin (uint8_t): The state of the pin from initialization until the first event. The following eTPU OC init_pin definitions are found in the etpu_oc_auto.h file:
  - FS_ETPU_OC_INIT_PIN_LOW
  - FS_ETPU_OC_INIT_PIN_HIGH
  - FS_ETPU_OC_INIT_PIN_NO_CHANGE

3.3 Initialization Routine - Address Reference: fs_etpu_oc_init_ref

```
uint8_t fs_etpu_oc_init_ref (uint8_t channel,
                             uint8_t priority,
                             uint8_t match_timebase,
                             uint32_t offset1,
                             uint8_t pin_action_capture_timebase1,
                             uint32_t offset2,
                             uint8_t pin_action_capture_timebase2,
                             uint8_t init_pin,
                             uint32_t * ref)
```

This routine is used to initialize channel to use the OC function with a value stored in a eTPU Data Memory address location as reference.

In order for the OC function to run, it needs to use some of the eTPU data memory. There is not any fixed amount of data memory associated with each channel in the eTPU. The memory needs to be allocated in a way that makes sure each channel has its own memory that will not be used by any other channels. There are two ways to allocate this memory: automatically or manually. Using automatic allocation to initialize each channel, it reserves some of the eTPU data memory for its own use. With manual configuration, the eTPU data memory is defined when the system is designed.

Automatic allocation is simpler and is used in all of the examples programs. The routine uses automatic allocation if the channel parameter base address field for a channel is zero. This is the reset condition of the field so normally you don't need to do anything except call the initialization API routine.

If you call the initialization routine more than once, it will only allocate data memory the first time it is called. The initialization routine will write a value to the channel parameter base address field, so on subsequent calls, it will not allocate more memory.

If the eTPU data memory is allocated manually, then a value must be written to channel parameter base address before the initialization routine is called. This is normally only used if the user wants to pre-define the location of each channels data memory.

After the channel has been initialized, the OC function will be executed as specified. This function has the following parameters:

- Channel (uint8_t) : The OC channel number. For devices with two eTPUs, this parameter should be assigned a value of 0-31 for eTPU_A and 64-95 for eTPU_B. For products with a single eTPU, this parameter should be assigned a value of 0-31.
- Priority (uint8_t): The priority to assign to the eTPU OC channel. The following eTPU priority definitions are found in utilities file etpu_utils.h.
  - FS_ETPU_PRIORITY_HIGH
  - FS_ETPU_PRIORITY_MIDDLE
  - FS_ETPU_PRIORITY_LOW
  - FS_ETPU_PRIORITY_DISABLED
- Match_timebase (uint8_t): The timebase which the eTPU OC channel will use for matches. The same timebase will be used for both matches. The following eTPU OC match_timebase definitions are found in the etpu_oc_auto.h file:
  - FS_OC_MATCH_TCR1
  - FS_OC_MATCH_TCR2
- Offset1 (uint32_t): The number of selected TCR counts from the reference value stored in eTPU data memory to the first event.
- Offset2 (uint32_t): The number of selected TCR counts from the first match to the second event.
- Pin_action_capture_timebase1/2(unit8_t) : The pin action at the first/second match and the capture timebase for those events. The following eTPU OC pin_action_capture_timebase1/2 definitions are found in the etpu_oc_auto.h file:
  - FS_ETPU_OC_PIN_LOW_CAPTURE_TCR1
  - FS_ETPU_OC_PIN_LOW_CAPTURE_TCR2
  - FS_ETPU_OC_PIN_HIGH_CAPTURE_TCR1
  - FS_ETPU_OC_PIN_HIGH_CAPTURE_TCR2
  - FS_ETPU_OC_PIN_TOGGLE_CAPTURE_TCR1
  - FS_ETPU_OC_PIN_TOGGLE_CAPTURE_TCR2
  - FS_ETPU_OC_PIN_NO_CHANGE_CAPTURE_TCR1

  - FS_ETPU_OC_PIN_NO_CHANGE_CAPTURE_TCR2
- Init_pin (uint8_t): The state of the pin from initialization until the first event. The following eTPU OC init_pin definitions are found in the etpu_oc_auto.h file:
  - FS_ETPU_OC_INIT_PIN_LOW
  - FS_ETPU_OC_INIT_PIN_HIGH
  - FS_ETPU_OC_INIT_PIN_NO_CHANGE
- Ref (uint32_t *): The address of the eTPU data memory location whose contents will be used as a reference.

3.4 Return Match and Capture Times - fs_etpu_oc_data

void fs_etpu_oc_data (uint8_t channel,

struct Match_and_Actual_times *these_times)

This routine populates a structure of type Match_and_Actual_times with the match and capture times for both. This routine would be called after the OC function has completed execution. This function has the following parameters:

- Channel (uint8_t) : The OC channel number. For devices with two eTPUs, this parameter should be assigned a value of 0-31 for eTPU_A and 64-95 for eTPU_B. For products with a single eTPU, this parameter should be assigned a value of 0-31.
- *these_times (struct Match_and_Actual_times**) :** A pointer to a structure where this routine will store the match and capture values for both events. This structure is defined in etpu_oc.h as follows:

```
struct Match_and_Actual_times
{
int32_t MatchTime1,
        MatchTime2,
        ActualTime1,
        ActualTime2;
};
```

The members of this structure are defined as follows:

MatchTime1 : The TCR count when event1 was scheduled to occur

MatchTime2 : The TCR count when event2 was scheduled to occur

ActualTime1 : The TCR count when event1 actually happened

ActualTime2 : The TCR count when event2 actually happened

4 Examples of Function Use

This section describes a simple use of the OC function and how to initialize the eTPU module and assign the eTPU OC function to an eTPU channel.

The example consists of two files:

- OC_example1.h
- OC_example1.c

File OC_example1.c contains the main() routine. This routine initializes the MPC5554 device for 128-MHz CPU operation and initializes the eTPU according to the information in the my_etpu_config struct (stored in file OC_example1.h). The timebases are enabled by calling routine fs_timer_start(). Any interrupt or DMA requests are cleared. The pins used in this example are configured for eTPU operation.

4.1 Channel OC0 Functionality Description

The OC function is initialized in immediate reference mode on channel OC0 (ETUA2). The channel is set for medium priority with matches on TCR1. Event1 will be scheduled for 0xA00 TCR1 counts after the host service request, and event2 will be scheduled for 0xA00 + 0x500 TCR1 counts after the host service request. The pin state will be low between the host service request and Event1. At event1, the pin will drive high, and at event2, the pin will drive low. The actual values of TCR1 for both events will be captured and stored in two eTPU data memory locations by the function after the second match has occurred. The captured times can be accessed by using the API routine fs_etpu_oc_data. A channel interrupt and data transfer request will be generated by the function after the second match has occurred.

The host polls the channel interrupt request bit and once it becomes set the host clears it.

4.2 Channel OC1 Functionality Description

In this case, the first event is scheduled relative to the second event on channel OC0.

The OC function is initialized in address reference mode on channel OC1 (ETUA4). The channel is set for medium priority with matches on TCR1. Event1 will be scheduled for 0xA00 TCR1 counts after the value stored at address OC0_last_match_ptr and event2 will be scheduled for 0xA00 + 0x500 TCR1 counts after the value stored at address OC0_last_match_ptr. The pin state will be high between the host service request and Event1. At event1 and event2 the pin will toggle. The actual values of TCR1 at event1 and TCR2 at event2 will be captured and stored in two eTPU data memory locations. The captured times can be accessed by using the API routine fs_etpu_oc_data. A channel interrupt and data transfer request will be generated by the function after the second match has occurred.

4.3 Channel OC2 Functionality Description

In this case, the first event is scheduled relative to an absolute TCR1 value. The value is derived by reading the eTPU time base 1 (TCR1) visibility register (ETPUTB1R) for engine A and adding 0x1000 counts. This value is then stored in variable oc2_value.

The OC function is initialized in address reference mode on channel OC2 (ETUA10). The channel is set for medium priority with matches on TCR1. Event1 will be scheduled for when TCR1 equals oc2_value and, event2 will be scheduled for when TCR1 equal oc2_value + 0x500 TCR1 counts. The pin state will be unchanged (from what ever it happened to be before) between the host service request and Event1. At event1, the pin will drive high and at event2 the pin will drive low. The actual values of TCR1 for both

events will be captured and stored in two eTPU data memory locations by the function after the second match has occurred. The captured times can be accessed by using the API routine fs_etpu_oc_data. A channel interrupt and data transfer request will be generated by the function after the second match has occurred.

After the function finishes execution on OC2, the host clears the channel interrupt and data transfer request bits for channel OC1 and OC2.

4.4 Example Use of fs_etpu_oc_data

The match and capture time for channels OC0 and OC1 are placed into two structures, these_times_OC0 and these_times_OC1, by two calls to fs_etpu_oc_data. The returned data is tested to ensure that the values are as they should be.

Note that in a highly loaded system, these tests may not pass because the captured times are likely to be different to those generated in an unloaded (non-latent) system.

5 Summary and Conclusion

This eTPU OC application note provides the user with a description of the output compare eTPU function usage and examples. The simple C interface routines to the OC eTPU function enable easy implementation of the OC function in applications. The functions are targeted for the MPC5500 and the MCF53*x* families of devices, but they can be used with any device that contains an eTPU.

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

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

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

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

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
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
LDCForFreescaleSeminconductor@hibbertgroup.com

AN2852
Rev. 0
06/2005