Freescale Semiconductor
Application Note

Document Number: AN2851
Rev. 0, 08/2007

Using the eTPU Input Capture Function

by: David Paterson
MCD Applications EKB

This eTPU Input Capture (IC) application note provides simple C routines to interface to the IC eTPU function. These routines can be used on any product that 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.”

Contents

1 Overview

The IC function captures, counts, and records the times of edges, with no CPU intervention required. The number of counts can be a single specified number or a continuous count. Edges counted can be rising, falling, or both. Links can be made to other channels on completion of a certain number of counts for a specified eTPU channel.

The eTPU IC function is based on the TPU IC function. The following enhancements have been made over the TPU IC function.

- Linking is not limited to eight channels
- The maximum count (number of edges to count before generating an exception request) is 24 bits
- 64 eTPU channels are available for link operation

2 Functional Description

The IC function can count positive, negative or both types of edges on the specified channel. It can count to a specified number of edges or count continuously. Timestamp data is also recorded from the selected timebase when the edge is captured.

The function can operate in three modes: single, continuous, or link. In single mode, the function captures only one input edge. In continuous mode, the function captures input edges continuously. In link mode, the function can link to another channel after a specified number of input edge counts.

2.1 Notes on the Performance and Use of eTPU IC Function

2.1.1 Performance

Like all eTPU functions, IC function performance in an application is dependent, to some extent, upon the service time (latency) of other active eTPU channels. This is due to the operational nature of the scheduler. The more eTPU channels that are active, the more performance decreases. 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 IC software release available from Freescale.

2.1.2 Linked Operation

The IC function can perform links only to other eTPU functions with input link capability. This function does not support receiving a link from another channel.

2.1.3 Changing Operating Modes

To change operation modes on the channel while it is still running, the channel must first be disabled. This can be done using the fs_etpu_disable function, found in file etpu_utils.h.

2.1.4 Noise Immunity

Features in the hardware of the eTPU and the microcode of the IC function protect, to a large extent, the counter from erroneous updates due to noise. All eTPU input channels incorporate a digital filter, which rejects pulses of less than a programmable duration.

3 C Level API for eTPU IC Function

The following routines provide easy access for the application developer to the IC function. Using these routines eliminates the need to directly control the eTPU registers. The API consists of five functions.

These functions can be found in the etpu_ic.h and etpu_ic.c files. The routines are described below and are available from Freescale. In addition, the eTPU C-compiler generates a file called etpu_ic_auto.h. This file contains information relating to the eTPU IC function, including details of how the eTPU Data Memory is organized and definitions for various API parameters.

```
int32_t fs_etpu_ic_init                 (uint8_t channel,
                                         uint8_t priority,
                                         uint8_t mode,
                                         uint8_t timebase,
                                         uint8_t edge,
                                         uint32_t max_count)

int32_t fs_etpu_ic_init_link            (uint8_t channel,
                                         uint8_t priority,
                                         uint8_t mode,
                                         uint8_t timebase,
                                         uint8_t edge,
                                         uint32_t max_count,
                                         uint32_t link1,
                                         uint32_t link2)

int32_t fs_etpu_ic_read_trans_count     (uint8_t channel)

int32_t fs_etpu_ic_read_final_time      (uint8_t channel)

int32_t fs_etpu_ic_read_last_time       (uint8_t channel)
```

3.1 Initialization Routine

The initialization routines dynamically allocate eTPU Data Memory. If dynamic allocation is not required, then the clock channel's Channel Parameter Base Address field should be written with a non-zero value before calling the fs_etpu_spi_init function.

Dynamic allocation of eTPU Data Memory occurs if the channel has a zero in its Channel Parameter Base Address field. The API updates the Channel Parameter Base Address field with a non-zero value to point to the parameter RAM allocated to the channel. The fs_etpu_ic_init and fs_etpu_ic_init_link APIs will not allocate new parameter RAM if the channel has a non-zero value in its Channel Parameter Base Address field; this means that channel has already been assigned.

These initialization routines are used to initialize an eTPU channel for the eTPU IC functions. After the channel has been initialized, the channel waits for specified events as decided in the functions, and performs captures, counts, timestamps (with current timebase value) and links.

The functions have the following parameters:.

- **channel (uint8_t)**: The IC 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 IC 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
- **mode (uint8_t):** The operation mode in which the eTPU IC function will run. The following eTPU IC operation mode definitions are found in the etpu_ic_auto.h file.
  - ETPU_IC_MODE_SINGLE (only one edge count)
  - ETPU_IC_MODE_CONT (continuously counts edges)
  - ETPU_IC_MODE_LINK (links to another channel and continuously counts edges)
- **timebase (uint8_t):** The timebase the eTPU IC channel will use. The following eTPU timebase definitions are found in utilities file etpu_utils.h.
  - FS_ETPU_TCR1
  - FS_ETPU_TCR2
- **edge (uint8_t):** This is the type of edge to detect and are found in the etpu_ic_auto.h file. It should be assigned a value of:
  - ETPU_IC_FALLING_EDGE
  - ETPU_IC_RISING_EDGE
  - ETPU_IC_ANY_EDGE
- **max_count (uint32_t):** This is the number of edges to count before generating an exception request. For example, enter value ‘1’ for 1 count, value ‘2’ for 2 counts.
- **link1 (uint32_t):** This is a packed 32-bit parameter with 4 x 8-bit channel fields in it. See Figure 1 for details. (For example, 0b11001000 would link to Channel 8 on the other eTPU engine; 0b00001000 would link to Channel 8 on the current eTPU engine.)
- **link2 (uint32_t):** This is a packed 32 bit parameter with 4 x 8-bit channel fields in it. See Figure 1 for details. (For example, 0b11001000 would link to Channel 8 on the other eTPU engine; 0b00001000 would link to Channel 8 on the current eTPU engine.)

7	6	5	4	3	2	1	0
Engine Selection		reserved[1]	Channel Number				

1 Reserved bits must be written 0.

Engine Selection	Description
00	This engine
01	Engine 1
10[1]	Engine 2
11	The other engine

1 This can occur only if the link service request came from the other engine or from serviced channel itself.

Figure 1. Microengine Link Register (eTPU Block Guide Reference)

NOTE: Return

This IC function returns an error code if the channel cannot be initialized. The error codes that can be returned are found in utilities file etpu_utils.h:

- FS_ETPU_ERROR_MALLOC
- FS_ETPU_ERROR_FREQ
- FS_ETPU_ERROR_VALUE
- FS_ETPU_ERROR_CODESIZE

NOTE

This IC function configures only the eTPU, not the pin; you may have to configure a pin to select the eTPU in your application (see example code).

4 Examples of Function Usage

4.1 Simple Example

4.1.1 Description

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

4.1.2 Example Code

The example consists of two files:

- ic_example1.c
- ic_example1.h

File ic_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 ic_example1.h).

Two pins are used in this example. EMIOS0 is configured as GPIO and ETPUA1 is configured for eTPU functionality; then the IC function is initialized on channel IC_1 (ETPUA1).

The channel is configured to capture one rising edge with middle priority in single mode with timer TCR1 as follows:.

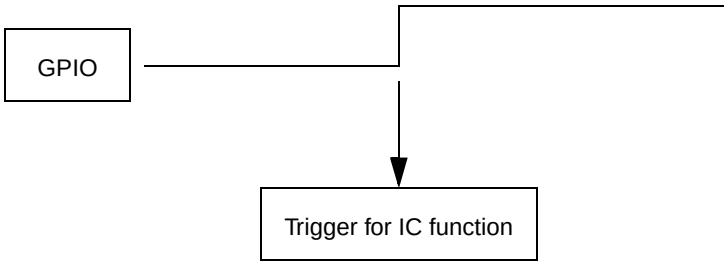
```
error_code = fs_etpu_ic_init(IC_1, FS_ETPU_PRIORITY_MIDDLE, ETPU_IC_MODE_SINGLE,
             FS_ETPU_TCR1, ETPU_IC_RISING_EDGE, 1);
```

The GPIO pin is then toggled to trigger the eTPU IC function. Note that an external connection is required (from ETPUA1 to EMIOS0).

4.1.3 Program Output

The final timestamp (captured TCR value) is read using the function defined in etpu_ic.c:

```
final_value = fs_etpu_ic_read_final_time (IC_1);
```



4.2 Complex Example

4.2.1 Description

This section describes a more complex use of the IC function. The eMIOS module is configured to run with the same clock frequency as the eTPU, and an eMIOS channel is set up to drive the eTPU channel using Single Action Output Compare (SAOC) mode. The IC function counts five rising edges in continuous mode, and compares the fourth and fifth timestamps, to ensure the timing is correct. Note that an external connection is required (from EMIOS3 to ETPUA0).

4.2.2 Example Code

The example consists of two files:

- ic_example2.c
- ic_example2.h

File ic_example2.c contains the main() routine. This routine initializes the MPC5554 device for 128 MHz CPU operation, sets up the eMIOS module, and initializes the eTPU according to the information in the my_etpu_config structure (stored in file ic_example2.h).

Two pins are used in this example. EMIOS3 is configured as SAOC mode and ETPUA0 is configured for eTPU functionality; then the IC function is initialized on channel IC_1 (ETUA0).

The channel is configured to capture five rising edges with middle priority in continuous mode with timer TCR1 and then links to IC_2 on completion of the five counts. IC_2 could be, for example, another eTPU function that has link input functionality (QOM, SPWM, or other custom functions). After counting five rising edges, the channel continues to count.

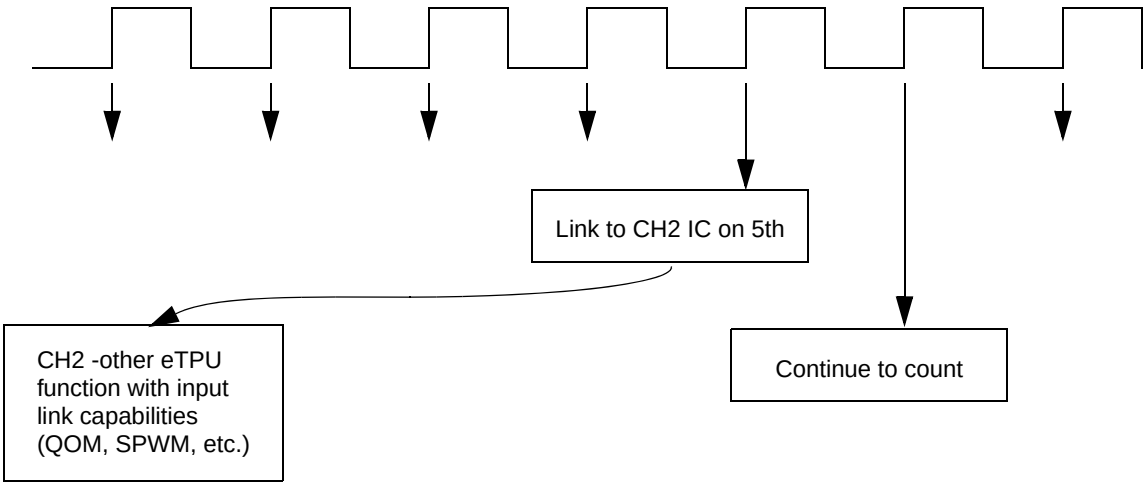
```
error_code = fs_etpu_ic_init_link(IC_1, FS_ETPU_PRIORITY_MIDDLE, ETPU_IC_MODE_CONT,
             FS_ETPU_TCR1, ETPU_IC_RISING_EDGE, 5, IC_2, 0);
```

The eMIOS channel is then configured to produce a 50% duty cycle waveform.

4.2.3 Program Output

The program waits for all five edges to be captured, then compares the fourth and fifth timestamp (captured TCR value) to ensure the timing was correct:

```
last_time = fs_etpu_ic_read_last_time (IC_1);        //Read value when 2nd last IC occurred
final_time = fs_etpu_ic_read_final_time (IC_1);      //Read value when last IC occurred
if (final_time - last_time != 0xA02){}               //Wait here if difference is not 0xA02
                                                     //eMIOS counter = 0 to 0x500 = 0x501
                                                     //counts
```



5 Summary and Conclusions

This eTPU IC application note describes and provides examples of how to use the Input Capture eTPU function. The simple C interface routines for the IC eTPU function allow easy implementation of the IC function in applications. The functions are targeted at the MPC5500 family of devices, but they can be used with any device that has an eTPU.

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

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)
www.freescale.com/support

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
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: AN2851
Rev. 0
08/2007

freescale™
semiconductor