Freescale Semiconductor
Application Note

Document Number: AN3772
Rev. 0, 05/2009

Using the Knock Window eTPU Function

by: David Paterson

Contents

1 Introduction

This application note provides simple C interface functions for the automotive knock window function used as part of the complete eTPU Set 2 Automotive Function Set. The functions can be used on any Freescale product that has an eTPU module. Example code is available for MPC55xx and MPC563xM devices. Read this application note in conjunction with application notes *AN2864 -- General C Functions for the eTPU* and *AN3768 -- eTPU Automotive Set (Set 2*).

2 Function Overview

The knock window function outputs a number of knock windows with specific properties within a complete engine cycle and without CPU intervention. The number of windows is programmable, interrupts can be generated, and pulse polarity is selectable. An update function allows for the window properties to be changed during runtime.

3 Functional Description

This function works in conjunction with CRANK and CAM eTPU functions to produce pulses referenced to angles on the crankshaft. This can be used in knock detection caused by the explosive detonation of the air and or fuel mixture (as opposed to a controlled burn), after the spark event occurs (pre-ignition happens before the spark event, when spontaneous combustion occurs). The pulse output by the eTPU can be used as a source for an analog-to-digital converter (ADC) trigger. The number of knock windows is selectable from one to eight. Interrupts can be generated on the open or close of a knock window and the pulse can be active low or active high. An update function allows the window properties to be changed at runtime. A typical usage

diagram is shown in the following figure. The function supports external knock application-specific integrated circuits (ASICs) as well as single-chip knock detection solutions using the internal ADC. This function can can be used for other applications that require angle to angle based events. It does not support receiving links from other channels. If a link is received, all latches are cleared and the function is reinitialized.

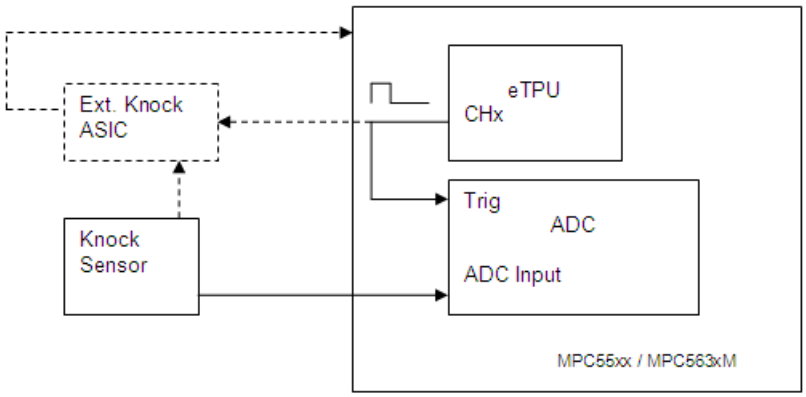

Figure 1. Knock Window Usage Diagram

3.1 Performance and Use of the eTPU Knock Window Function

Performance — Like all eTPU functions, the knock window function performance in an application that 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 more eTPU channels that are active, the larger the impact on performance. 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 knock window software release, available from Freescale.

Limitations — The maximum number of knock windows that can be generated in one 720° engine cycle is 8. To prevent the current window from overlapping (for example, closing after the next window opens), the user is responsibe for checking the parameters to ensure that angle widths do not cause an overlap with the next window open angle.

4 C Level Application Program Interface (API) for eTPU Knock Window Functions

The following functions provide easy access to the knock window function. Use of these functions eliminates the need to directly control eTPU registers. The API consists of two functions. These functions can be found in the etpu_knock_window.c and etpu_knock_window.h files. The functions are described below and are available from Freescale as part of this application note download. In addition, the eTPU C-compiler generates a file called etpu_knock_window_auto.h. This file contains information relating to the eTPU knock window function, including details of how the eTPU data memory is organized and definitions for various API parameters.

```
int32_t fs_etpu_knock_window_init (uint8_t channel,
                                   uint8_t priority,
                                   uint8_t number_of_windows,
                                   uint8_t edge_polarity,
                                   uint8_t edge_interrupt,
                                   uint8_t cam_chan,
                                   uint32_t *knock_window_angle_table)

int32_t fs_etpu_knock_window_update (uint8_t channel,
                                     uint8_t cam_channel,
                                     uint8_t update_window_number,
                                     uint32_t angle_open_new,
                                     uint32_t angle_width_new)
```

The initialization and update functions dynamically allocate eTPU data memory. Dynamic allocation of eTPU data memory occurs if the channel has a zero in its channelpParameter base address field. The channel parameter base address field is updated by the API with a non-zero value to point to the parameter RAM allocated to the channel. The fs_etpu_knock_window_init and fs_etpu_knock_window_update APIs do not allocate new parameter RAM if the channel has a non-zero value in its channel parameter base address field; this means the channel has already been assigned.

4.1 Initialization Function

The initialization function (fs_etpu_knock_window_init) is used to initialize an eTPU channel for the eTPU knock window function. After the channel has been initialized, it waits for the desired angles to occur and output pulse(s) as specified. A table of windows containing open and width angles must be defined which the initialization function points to. The initialization function points this out. The table of window structures must be implemented as shown in Figure 2. Also see the example code.

Figure 2. Knock Window Angle Table Structure

Each entry is represented as a fixed point number with two decimal places (0.01 accuracy). For example, 3001 represents 30.01° and 15050 represents 150.50°.

The function has the following parameters:

NOTE

To prevent the current window from overlapping (for example, closing after the next window opens), the user is responsibe for checking the parameters to ensure that angle widths do not cause an overlap with the next window open angle.

- channel (uint8_t) — The knock window channel number. For products with a single eTPU, this parameter must be assigned a value of 0 – 31. For devices with two eTPUs, assigned a value of 0 – 31 for eTPU_A and 64 – 95 for eTPU_B.
- priority (uint8_t) — The priority to assign to the eTPU 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

- number_of_windows (uint8_t) — This is the total number of knock windows per engine cycle. The range is from 1 to 8.
- edge_polarity (uint8_t) — This selects the polarity of the output signal. This parameter must be assigned a value of:
  - FS_ETPU_KNOCK_FM0_RISING_EDGE
  - FS_ETPU_KNOCK_FM0_FALLING_EDGE

- edge_interrupt (uint8_t) — This selects an interrupt on open or close of the knock window. This parameter must be assigned a value of:
  - FS_ETPU_KNOCK_FM1_INT_OPEN
  - FS_ETPU_KNOCK_FM1_INT_CLOSE

- cam_channel (uint8_t) — CAM channel number

- *knock_window_angle_table (uint32_t) — This is a pointer to the table of knock windows. The entries in the table must be specified from 0 to 71999, where 71999 represents 719.99 °. See Figure 2.

Return Notes — Returns error code if the channel could not 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

Warning — This function does not configure the pin; it only configures the eTPU. In a system, a pin may need to be configured to select the eTPU functionality. See example code.

4.2 Update Function

The update function is used to update a particular knock window's parameters by specifying the channel, update window number, new knock window properties, and open and angle widths. The coherent dual-parameter controller (CDC) eTPU hardware is used to coherently copy the two update parameters (angle_open_new and angle_width_new) to eTPU data memory. This ensures that start and width angles are coherent. After the update function has been executed, the specified knock window is altered on the next scheduled knock window output.

NOTE

To prevent the current window from overlapping (for example, closing after the next window opens), the user is responsibe for checking the parameters to ensure that angle widths do not cause an overlap with the next window open angle.

The function has the following parameters:

- channel (uint8_t) — The knock window channel number for products with a single eTPU, this parameter should be assigned a value of 0 – 31. For devices with two eTPUs, this parameter should be assigned a value of 0 – 31 for eTPU_A, and 64-95 for eTPU_B.
- cam_channel (uint8_t) — CAM channel number.
- update_window_number (uint8_t) — The index number of the knock window to update.The range is from one to eight.
- angle_open_new (uint32_t) — Value of new open angle for the knock window to be updated. Range is 0 to 71999. 71999 represents 719.99°.
- angle_width_new (uint32_t) — Value of new angle width for the knock window to be updated. Range is 0 to 71999. 71999 represents 719.99°.

Return Notes — Returns error code if the channel update could not be performed. 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

5 Simple Example of Function Usage

Description — This simple example outputs two knock window pulses with a width of 50° starting at 100° and 200° on a 720° engine cycle. The polarity is active high and an interrupt is generated when the knock window opens.

Example Code — The example code is partly generated by Freescale's eTPU graphical configuration tool (GCT). See output files eng_pos_etpu_gct.c and eng_pos_etpu_gct.h.

The knock window function:

```
#define KNOCK_WINDOW0_CHANNELETPU_ENGINE_A_CHANNEL(10)
fs_etpu_knock_window_init (KNOCK_WINDOW0_CHANNEL, /* engine: A; channel: 10 */
                           FS_ETPU_PRIORITY_MIDDLE, /* priority: Middle */
                           knock_number_of_windows, /* number_of_windows: */
                                                    /* knock_number_of_windows */
                     FS_ETPU_KNOCK_FM0_RISING_EDGE, /* edge_polarity: */
```

```
                                        /* FS_ETPU_KNOCK_FM0_RISING_EDGE*/
            FS_ETPU_KNOCK_FM1_INT_OPEN,/* edge_interrupt: */
                                        /* FS_ETPU_KNOCK_FM1_INT_OPEN */
                        APP_ENG_POS0_CAM, /* cam_chan: APP_ENG_POS0_CAM */
    &my_knock_window_angle_table); /* knock_window_angle_table: */
                                        /* &my_knock_window_angle_table */
```

A tooth generator function is used to simulate CRANK and CAM signals. The engine position functions are used to generate the angle clock.

The highlevel main.c file includes various knock setup codes, assigns the eTPU data memory, and obtains the functions parameter and channel base addresses.

```
#define KNOCK_NUMBER_OF_WINDOWS 2
uint32_t knock_number_of_windows = KNOCK_NUMBER_OF_WINDOWS;
uint32_t my_knock_window_angle_table[KNOCK_NUMBER_OF_WINDOWS*2];
```

The knock window angle table is defined as:

```
my_knock_window_angle_table[0] = 10000;   /* Knock Window 1 (open)  */
my_knock_window_angle_table[1] = 5000;    /* Knock Window 1 (width) */
my_knock_window_angle_table[2] = 20000;   /* Knock Window 2 (open)  */
my_knock_window_angle_table[3] = 5000;    /* Knock Window 2 (width) */
```

Program Output — Toothgen outputs on channels 30 and 5 must be connected to engine position channels 4 and 0 respectively. The knock window is output on channel 10.

The interrupt bit is set by the eTPU hardware, but interrupts are not enabled and no interrupt service routine (ISR) exists (in this example). The application developer would need to enable interrupts and associated ISR if that is what the application needs. The channel interrupt status register indicates channel 10 has a pending interrupt to the host CPU.

The CRANK has 34 physical teeth and two missing teeth, each tooth relates to a 10° angle. In Figure 3 you can see that the knock windows are output correctly as per the example description.

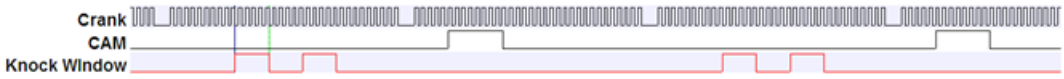

Figure 3. Simple Example

6 Complex Example of Function Usage

Description — This complex example outputs eight knock window pulses evenly spaced across the engine cycle with a width of 30° starting at 30° on a 720° engine cycle. The polarity is active high and an interrupt is generated, in this case, the knock window. The update function is used to update all knock windows to a width of 20°.

Example code — The example code is partly generated by Freescale's eTPU graphical configuration tool (GCT). See output files `eng_pos_etpu_gct.c` and `eng_pos_etpu_gct.h`.

The knock window function:

```
#define KNOCK_WINDOW0_CHANNELETPU_ENGINE_A_CHANNEL(10)
fs_etpu_knock_window_init (KNOCK_WINDOW0_CHANNEL, /* engine: A; channel: 10 */
                           FS_ETPU_PRIORITY_MIDDLE, /* priority: Middle */
                           knock_number_of_windows, /* number_of_windows:*/
                                                    /* knock_number_of_windows*/
                     FS_ETPU_KNOCK_FM0_RISING_EDGE, /* edge_polarity:*/
                                                    /* FS_ETPU_KNOCK_FM0_RISING_EDGE*/
                         FS_ETPU_KNOCK_FM1_INT_OPEN,/* edge_interrupt:*/
                                                    /* FS_ETPU_KNOCK_FM1_INT_OPEN*/
                                  APP_ENG_POS0_CAM, /* cam_chan: APP_ENG_POS0_CAM */
                &amp;my_knock_window_angle_table); /* knock_window_angle_table:*/
                                                  /*&amp;my_knock_window_angle_table*/
```

A tooth generator function is used to simulate CRANK and CAM signals and the engine position functions are used to generate the angle clock.

The highlevel `main.c` file includes various knock setup codes, assigns the eTPU data memory, and obtains the function parameter and channel base addresses.

```
#define KNOCK_NUMBER_OF_WINDOWS 8
uint32_t knock_number_of_windows = KNOCK_NUMBER_OF_WINDOWS;
uint32_t my_knock_window_angle_table[KNOCK_NUMBER_OF_WINDOWS*2];
```

The knock window angle table is defined as:

```
 my_knock_window_angle_table[0] = 3000;    /* Knock Window 1 (open) */
 my_knock_window_angle_table[1] = 3000;    /* Knock Window 1 (width */
 my_knock_window_angle_table[2] = 9000;    /* Knock Window 2 (open) */
 my_knock_window_angle_table[3] = 3000;    /* Knock Window 2 (width)*/
 my_knock_window_angle_table[4] = 15000;   /* Knock Window 3 (open) */
 my_knock_window_angle_table[5] = 3000;    /* Knock Window 3 (width)*/
 my_knock_window_angle_table[6] = 21000;   /* Knock Window 4 (open) */
 my_knock_window_angle_table[7] = 3000;    /* Knock Window 4 (width)*/
 my_knock_window_angle_table[8] = 27000;   /* Knock Window 5 (open) */
 my_knock_window_angle_table[9] = 3000;    /* Knock Window 5 (width)*/
 my_knock_window_angle_table[10] = 33000;  /* Knock Window 6 (open) */
 my_knock_window_angle_table[11] = 3000;   /* Knock Window 6 (width)*/
 my_knock_window_angle_table[12] = 39000;  /* Knock Window 7 (open) */
 my_knock_window_angle_table[13] = 3000;   /* Knock Window 7 (width)*/
 my_knock_window_angle_table[14] = 44000;  /* Knock Window 8 (open) */
 my_knock_window_angle_table[15] = 3000;   /* Knock Window 8 (width)*/
```

The following segment of code in main.c allows the knock window parameters to be updated. The example uses Freescale's FreeMaster application to update the knock window angles at runtime.

```
for (j=0; j&lt;(knock_number_of_windows); j++)
{
error = fs_etpu_knock_window_update ( KNOCK_WINDOW0_CHANNEL,
                                      APP_ENG_POS0_CAM,
                                      j,
                                      my_knock_window_angle_table[j*2],
                                      my_knock_window_angle_table[j*2+1] );
}
```

Program Output — Toothgen outputs on channels 30 and 5 must be connected to engine position channels 4 and 0. The knock window is output on channel 10. In this example the channel interrupt bit is set on open of window but this is not serviced. The channel interrupt status register indicates that channel 10 has a pending interrupt to the host CPU. The CRANK has 34 physical teeth and two missing teeth, each tooth relates to a 10° angle. In Figure 4 you can see that the knock windows are output correctly as per the example description.

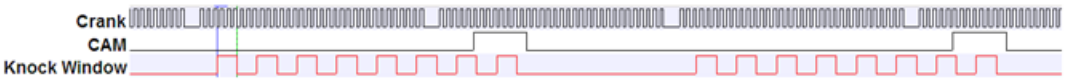

Figure 4. Complex Example

7 Conclusion

This eTPU Knock Window application note provides the user with a description of the knock window function usage and examples. The simple C interface functions to the knock window eTPU functions enable easy implementation of the knock window function in applications. The functions are targeted for the MPC55xx and MPC563xM family of devices, but can be used with any device that has 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

Europe, Middle East, and Africa:

Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+44 8 52200080 (English)
+44 80 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 153-0065
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:

Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
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: 303-675-2140
LDCForFreescaleSemiconductor@hibbertgroup.com

RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics as their non-RoHS-complaint and/or non-Pb-free counterparts. For further information, see
http://www.freescale.com or contact your Freescale sales representative.

For information on Freescale's Environmental Products program, go to
http://www.freescale.com/epp .

Document Number: AN3772
Rev. 0, 05/2009