Freescale Semiconductor
Application Note

Document Number: AN3770
Rev. 0, 04/2009

Using the FUEL eTPU Function

by: Michal Princ
System Application Engineer
Roznov Czech System Center

1 Introduction

This application note provides a simple C interface for the FUEL eTPU 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 enhanced time processor unit (eTPU) module. Example code is available for MPC55xx and MPC563xM devices. This application note must be read in conjunction with application note *AN2864 – General C Functions for the eTPU* and application note *AN3768 – eTPU Automotive Set (Set 2)*.

2 Function Overview

The FUEL eTPU function generates one or more fuel injection pulses and thus controls the amount of fuel in the intake manifold. Fuel injection pulses start at a calculated engine cycle angle (0–720 degrees) based on a specified end angle and the amount of fuel required. Multiple additional injection pulses are allowed.

Contents

freescale™ semiconductor

Injection time for the last engine cycle (per a cylinder) as well as the total injection time over all cylinders is stored. The eTPU synchronization functions, CAM and CRANK, provide angular information to the FUEL function.

3 Functional Description

3.1 Injection Pulse Description

The fuel driver can drive a timed pulse controlled by a combination of time and engine angle information. The user must specify:

1. The injection pulse width, consists of two values:
   - *Injection Time* — Defines the amount of fuel required.
   - *Compensation Time* — Compensates for the opening and closing time of the valve. This value is influenced by numerous parameters including the injector parameters, temperature, battery voltage, and so on.
2. Angle-based parameters that determine the injection pulse start angle:
   - *Cylinder Offset Angle* — The angle offset for each engine cylinder.
   - *Injection Normal End Angle* — This is the angle where the injection must be finished, before the absolute latest point. This parameter is global for all cylinders. The sum of the *Injection Normal End Angle + Cylinder Offset Angle* gives the angle at which the injection must be finished.
   - *Drop Dead Angle* — This angle corresponds to the closing angle of the intake valve plus an offset. This is the point where no additional fuel can be put into the cylinder. This parameter is global for all cylinders. The sum of the *Drop Dead Angle + Cylinder Offset Angle* gives the absolute latest angle where the injection needs to be finished.

Based on the above defined parameters the beginning of the injection pulse is calculated as follows:

Injection Start Angle = Injection Normal End Angle + Cylinder Offset Angle – time_to_angle(Injection Time + Compensation Time)

Calculation of the *Injection Start Angle* is performed typically twice per engine cycle, at the *Drop Dead Angle* and at the *Recalc Angle*, see Figure 1. The CPU can update the *Injection Time* at any time. If the update comes between the last *Drop Dead Angle* and the scheduled *Injection Start Angle* it triggers a recalculation of the *Injection Start Angle*.

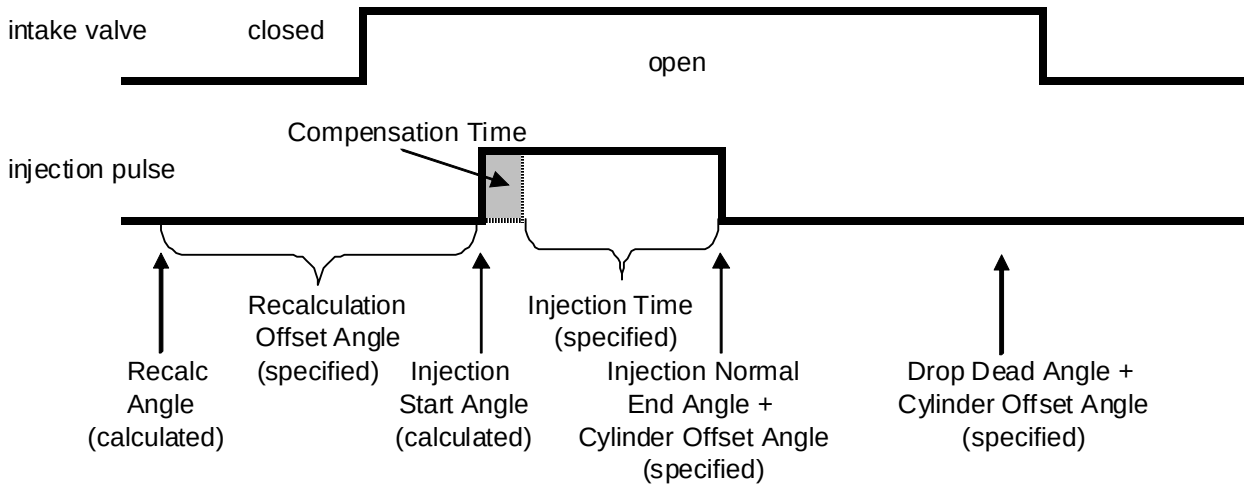

Figure 1. Injection Pulse Definition

3.2 Start Angle Re-Calculation

The application requires the end of the pulse to occur at a given engine angle, the injection pulse start angle must be predicted based on the actual engine speed. The pulse completion angle is subject to changes in engine speed while the pulse is running. The closer the prediction is made to the start time of the pulse, the better the prediction is. The first calculation of the next engine cycle *Injection Start Angle* is performed at the *Drop Dead Angle*, but it deals with a rough prediction. The *Injection Start Angle* needs to be re-calculated again, just before the estimated *Injection Start Angle*. You must specify the *Recalculation Offset Angle* parameter that determines the *Recalc Angle*, see Figure 1.

Recalc Angle = Injection Start Angle – Recalculation Offset Angle

3.3 Modifying the Amount of Fuel, Additional Injection Pulse(s)

The value of the injection time can be changed by the CPU at any time, even if the injection pulse is already finished (before the *Drop Dead Angle + Cylinder Offset Angle* point). If it is changed afterwards, the changes are valid for the next engine cycle.

3.3.1 Main Injection Pulse

After the injection pulse has started, it is a pure time pulse with no further angle references.

If the pulse is active, modifying the injection time via the CPU results in a longer or shorter pulse (worst case, is an immediate end of pulse). If the pulse gets longer, it stops at the *Drop Dead Angle + Cylinder Offset Angle* point. No additional fuel can be put into the cylinder after this point. If the pulse gets longer, the *Compensation Time* is used only once.

The function tracks *Real Injection Time* to know how much fuel has been injected in that cylinder. This is calculated by:

Real Injection Time = Real End Edge Time – Real Start Edge Time – Compensation Time

3.3.2 Additional Injection Pulse(s)

The CPU can request to change the *Injection Time* even if the main injection pulse is already finished. If this request comes after the point *Drop Dead Angle + Cylinder Offset Angle* the change applies for the next engine cycle. However, if the request comes before the point *Drop Dead Angle + Cylinder Offset Angle*, and if the value of the *Injection Time* is increased by the CPU, another injection pulse may happen (almost) immediately.

This pulse happens only if it is longer than a predefined parameter, *Minimum Injection Time*, preventing very short injection pulses being scheduled. Every pulse adds the *Compensation Time* because the valve needs to open and close once per pulse.

Pulses are not allowed to be back-to-back. After the pulse switches off, it must stay off for some time. Using the parameter *Minimum Off Time*, the FUEL function makes sure the signal is inactive between two consecutive pulses for this *Minimum Off Time*.

At the end of each additional injection pulse the *Real Injection Time* is updated:

Real Injection Time = Real Injection Time + Real End Edge Time – Real Start Edge Time – Compensation Time

3.4 Example of a Complex Injection Pulse

This section describes how complex injection pulse generation works, see Figure 2. In this example, the pulse has been scheduled with an *Injection Time* of 100 units. This defines the start point of the pulse (*Compensation Time* is always included).

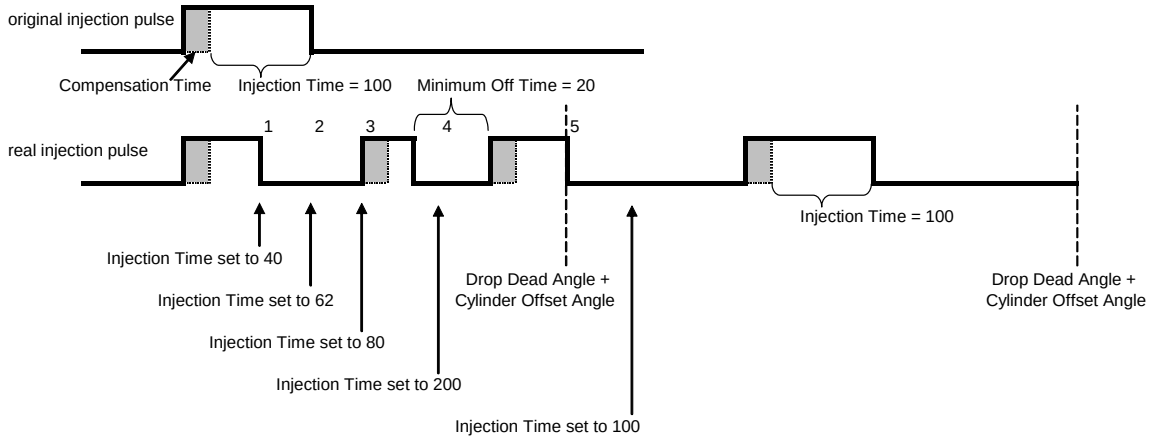

Figure 2. Injection Pulse

At point 1, the CPU decides to reduce the *Injection Time* to 40. As the current pulse is already longer than this value, an immediate end of the pulse is scheduled. The *Real Injection Time* is calculated (assume 60 in this example).

At point 2, the CPU decides to increase the *Injection Time* to 62. As the previous injection pulse had a length of 60 (stored in *Real Injection Time*), the delta pulse is calculated by 62-60=2. As the result is smaller than the *Minimum Injection Time* (assume 5 for this example), no pulse happens.

At point 3, the CPU increases the *Injection Time* to 80. The delta pulse is calculated by *Additional Pulse Length = Injection Time – Real Injection Time* = 80 - 60 = 20. This value is longer than the *Minimum Injection Time*, therefore an immediate pulse is scheduled with a length of (20 + *Compensation Time*). After the pulse the *Real Injection Time* is updated to 80.

At point 4, the CPU increases the *Injection Time* to 200. This pulse is not scheduled immediately, because the *Minimum Off Time* is not met. The eTPU waits for the *Minimum Off Time* to expire and then calculates a new pulse length:

Additional Pulse Length = Injection Time – Real Injection Time = 200 - 80 = 120.

This value is longer than the *Minimum Injection Time*, therefore a pulse is scheduled with a length of (120 + *Compensation Time*).

During this additional injection pulse the absolute angle at which the injection needs to be finished happens (*Drop Dead Angle + Cylinder Offset Angle*). This results in an immediate end of the pulse. The *Real Injection Time* is updated by the latest pulse and, in this example, set to 130. This value is added to the *Sum Of Injection Time*, which is used for the fuel consumption display on the dashboard.

Sum Of Injection Time = Sum Of Injection Time + Real Injection Time

The *Real Injection Time* is also copied to the CPU updating the CPU with the amount of real fuel injected in the last engine cycle. At this point the *Real Injection Time* is reset to 0.

The next change of the *Injection Time* to 100, that occurs after the *Drop Dead Angle + Cylinder Offset Angle* point, applies to the next engine cycle.

3.5 Performance and Use of the eTPU FUEL Function

3.5.1 FUEL Channel Assignment

Ensure all FUEL functions are using the same TCR1 clock. This can be executed by assigning all FUEL channels to the same eTPU engine, or by ensuring TCR1 clock synchronization on both eTPU engines.

3.5.2 FUEL Response to the Stall Condition on the CRANK Channel

If a STALL condition is detected on the CRANK channel, see application note *AN3769 — Using the Engine Position (CAM and CRANK) eTPU Functions*, the CRANK function sends a link to all output functions including all initialized FUEL channels. On the link, the FUEL channels immediately stop any injection pulses and re-sync (re-initialize) themselves, therefore no host intervention is required. All eTPU set2 output functions behave the same way in this situation.

3.5.3 FUEL Minimum Off Time, Limitations

As described previously, the *Minimum Off Time* parameter is introduced to ensure the fuel signal is inactive between two consecutive pulses for the defined minimum time period. This applies only for the time between the main injection pulse and the additional injection pulse. Minimum off time is not applied between two consecutive primary injection pulses (two engine cycles). The CPU must check the requested injection pulse width against the actual engine cycle period.

3.5.4 Performance

Like all eTPU functions, the FUEL 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 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 FUEL software release available from Freescale.

4 C Level Application Program Interface (API) for eTPU FUEL Function

The following functions provide easy access for the application developer to the Fuel function. Use of these functions eliminates the need to directly control the eTPU registers. The API consists of 14 functions. These functions can be found in the *etpu_fuel.c* and *etpu_fuel.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_fuel_auto.h*. This file contains information relating to the eTPU FUEL function including details of how the eTPU Data Memory is organized and definitions for various API parameters.

FUEL channels initialization functions:

```
int32_t fs_etpu_fuel_init_3cylinders(uint8_t  channel_1,
                                     uint8_t  channel_2,
                                     uint8_t  channel_3,
                                     uint8_t  cam_chan,
                                     int8_t  priority,
                                     uint8_t  polarity,
                                     uint24_t cylinder_offset_angle_1,
                                     uint24_t cylinder_offset_angle_2,
                                     uint24_t cylinder_offset_angle_3,
                                     uint24_t drop_dead_angle,
                                     uint24_t injection_normal_end_angle,
```

```
                                    uint24_t recalculation_offset_angle,
                                    uint24_t injection_time_us_1,
                                    uint24_t injection_time_us_2,
                                    uint24_t injection_time_us_3,
                                    uint24_t compensation_time_us,
                                    uint24_t minimum_injection_time_us,
                                    uint24_t minimum_off_time_us)
int32_t fs_etpu_fuel_init_4cylinders(uint8_t  channel_1,
                                    uint8_t  channel_2,
                                    uint8_t  channel_3,
                                    uint8_t  channel_4,
                                    uint8_t  cam_chan,
                                    uint8_t  priority,
                                    uint8_t  polarity,
                                    uint24_t cylinder_offset_angle_1,
                                    uint24_t cylinder_offset_angle_2,
                                    uint24_t cylinder_offset_angle_3,
                                    uint24_t cylinder_offset_angle_4,
                                    uint24_t drop_dead_angle,
                                    uint24_t injection_normal_end_angle,
                                    uint24_t recalculation_offset_angle,
                                    uint24_t injection_time_us_1,
                                    uint24_t injection_time_us_2,
                                    uint24_t injection_time_us_3,
                                    uint24_t injection_time_us_4,
                                    uint24_t compensation_time_us,
                                    uint24_t minimum_injection_time_us,
                                    uint24_t minimum_off_time_us)
int32_t fs_etpu_fuel_init_6cylinders(uint8_t  channel_1,
                                    uint8_t  channel_2,
```

```
                                    uint8_t  channel_3,
                                    uint8_t  channel_4,
                                    uint8_t  channel_5,
                                    uint8_t  channel_6,
                                    uint8_t  cam_chan,
                                    uint8_t  priority,
                                    uint8_t  polarity,
                                    uint24_t cylinder_offset_angle_1,
                                    uint24_t cylinder_offset_angle_2,
                                    uint24_t cylinder_offset_angle_3,
                                    uint24_t cylinder_offset_angle_4,
                                    uint24_t cylinder_offset_angle_5,
                                    uint24_t cylinder_offset_angle_6,
                                    uint24_t drop_dead_angle,
                                    uint24_t injection_normal_end_angle,
                                    uint24_t recalculation_offset_angle,
                                    uint24_t injection_time_us_1,
                                    uint24_t injection_time_us_2,
                                    uint24_t injection_time_us_3,
                                    uint24_t injection_time_us_4,
                                    uint24_t injection_time_us_5,
                                    uint24_t injection_time_us_6,
                                    uint24_t compensation_time_us,
                                    uint24_t minimum_injection_time_us,
                                    uint24_t minimum_off_time_us)
```

Change operation functions:

```
int32_t fs_etpu_fuel_set_injection_time(uint8_t  channel,
                                        uint24_t injection_time_us)
int32_t fs_etpu_fuel_set_drop_dead_angle(uint8_t  channel,
                                         uint24_t drop_dead_angle)
int32_t fs_etpu_fuel_set_normal_end_angle(uint8_t  channel,
```

```
                                                uint24_t normal_end_angle)
    int32_t fs_etpu_fuel_set_recalc_offset_angle(uint8_t  channel,
                                                uint24_t recalc_offset_angle)
    int32_t fs_etpu_fuel_set_compensation_time(uint8_t  channel,
                                                uint24_t compensation_time_us)
    int32_t fs_etpu_fuel_set_minimum_injection_time(uint8_t  channel,
                                                    uint24_t minimum_injection_time_us)
    int32_t fs_etpu_fuel_set_minimum_off_time(uint8_t  channel,
                                                uint24_t minimum_off_time_us)
    int32_t fs_etpu_fuel_switch_off(uint8_t channel)
    int32_t fs_etpu_fuel_switch_on(uint8_t channel)
```

Value return functions:

```
    uint24_t fs_etpu_fuel_get_sum_of_injection_time(void)
    uint24_t fs_etpu_fuel_get_CPU_real_injection_time(uint8_t channel)
```

The FUEL initialization functions dynamically allocate eTPU Data Memory. Dynamic allocation of eTPU Data Memory occurs if the channel has a zero in its channel parameter base address field. The channel parameter base address (CPBA) field is updated by the API with a non-zero value to point to the parameter RAM allocated to the channel. The *fs_etpu_fuel_init_3cylinders*, *fs_etpu_fuel_init_4cylinders* and *fs_etpu_fuel_init_6cylinders* APIs does not allocate new parameter RAM if the channels have a non-zero value in their channel parameter base address fields, this means that channels have already been assigned.

4.1 Initialization Functions

4.1.1 int32_t fs_etpu_fuel_init_3cylinders(...), int32_t fs_etpu_fuel_init_4cylinders(...), int32_t fs_etpu_fuel_init_6cylinders(...)

The initialization functions (*fs_etpu_fuel_init_Xcylinders*) are used to initialize the eTPU channels for the eTPU FUEL functions. Based on the number of the engine cylinders it is possible to initialize 3, 4 or 6 eTPU channels for the FUEL function. The functions have the following parameters:

- channel_X (uint8_t) — The number of the channel that generates the fuel signal for the cylinder #X, X = 1–6. For products with a single eTPU, this parameter must be assigned a value of 0–31. For devices with two eTPUs, this parameter must be assigned a value of 0–31 for eTPU_A and 64–95 for eTPU_B. The user must ensure all FUEL functions use the same TCR1 clock. This can be executed by assigning all FUEL channels to the same eTPU engine, or by ensuring TCR1 clock synchronization on both eTPU engines.
- cam_chan (uint8_t) — This is the channel number the CAM function is assigned to. The FUEL function reads the TCR2_Counts_Per_Engine_Cycle value from CAM.

- priority (uint8_t) — This is the priority to assign to all FUEL channels. This parameter must be assigned a value of:
  - FS_ETPU_ PRIORITY_HIGH
  - FS_ETPU_ PRIORITY_MIDDLE
  - FS_ETPU_ PRIORITY_LOW
  - FS_ETPU_ PRIORITY_DISABLED
- polarity (uint8_t) — This is the polarity of the output signal. It must be assigned one of these values:
  - FS_ETPU_FUEL_PULSE_HIGH
  - FS_ETPU_FUEL_PULSE_LOW
- cylinder_offset_angle_X (uint24_t) — Offset angle for the engine cylinder #X, X = 1-6. Range 0 to 71999. 71999 represents 719.99 degrees.
- drop_dead_angle (uint24_t) — This is the closing angle of the intake valve. No additional fuel can be put into the cylinder after this angle. Range 0 to 71999. 71999 represents 719.99 degrees. This parameter is global and sets the drop dead angle for all the fuel signals.
- injection_normal_end_angle (uint24_t) — This is the angle where the injection should be finished. Range 0 to 71999. 71999 represents 719.99 degrees. This parameter is global and sets the normal end angle for all the fuel signals.
- recalculation_offset_angle (uint24_t) — (Injection_Start_Angle-Recalculation_Offset_Angle) defines the angle where the re-calc thread is scheduled. Range 0 to 71999. 71999 represents 719.99 degrees. This parameter is global and sets the recalculation angle for all the fuel signals.
- injection_time_us_X (uint24_t) — This is the length of the injection pulse, in microseconds, for the engine cylinder #X, X = 1–6. If it is not necessary to generate injection pulses immediately after the FUEL INIT, set these parameters to zero and later use the fs_etpu_fuel_set_injection_time function to change the injection time for all individual FUEL channels.
- compensation_time_us (uint24_t) — Each generated injection pulse is prolonged by this time to compensate for the opening and closing time of the valve. The time is measured in microseconds. This value is influenced by valve parameters, temperature, battery voltage, and so on. This parameter is global and sets the compensation time for all the fuel signals.
- minimum_injection_time_us (uint24_t) — This is the required minimum injection pulse width, in microseconds. This parameter is global and sets the minimum injection time for all the fuel signals.
- minimum_off_time_us (uint24_t) — This is the minimum time between two injection pulses, in microseconds. This parameter is global and sets the normal off time for all the fuel signals.

Return Notes — The initialization function returns an error code if the channel can not be initialized. The error codes that can be returned are found in the utilities file etpu_utils.h:

- FS_ETPU_ERROR_MALLOC
- FS_ETPU_ERROR_VALUE

WARNING

These functions do not configure pins, they only configure the eTPU. In a device, pins may need to be configured to select the eTPU functionality. See example code.

4.2 Change Operation Functions

These functions are used to set or change function parameters of the eTPU FUEL functions. The functions and their parameters are described in this section:

4.2.1 int32_t fs_etpu_fuel_set_injection_time(uint8_t channel, uint24_t injection_time_us)

This function is used to change the injection time on a specific FUEL channel. It is not possible to change the injection time for all initialized FUEL channels by calling this function, instead you have to call this function several times based on the number of initialized FUEL channels. This function has the following parameters:

- channel (uint8_t) — The number of the channel that generates the fuel signal.
- injection_time_us (uint24_t) — This is the length of the injection pulse in microseconds.

Return notes — This function returns 0 if the injection time update was successful. This function must only be used if the FUEL channel has no pending HSRs. If it is called and there are pending HSRs, a sum of pending HSR numbers is returned and this function must be called again later.

4.2.2 int32_t fs_etpu_fuel_set_drop_dead_angle(uint8_t channel, uint24_t drop_dead_angle)

This function sets the drop dead angle. The drop dead angle parameter is global for all initialized FUEL channels. This function changes the drop dead angle for all generated fuel signals.

This function has the following parameters:

- channel (uint8_t) — The number of any channel that generates the fuel signal, typically the FUEL channel number for the first cylinder.
- drop_dead_angle (uint24_t) — This is the closing angle of the intake valve. No additional fuel can be put into the cylinder after this angle. Range 0 to 71999. 71999 represents 719.99 degrees.

Return notes — The error code that can be returned is:

- FS_ETPU_ERROR_VALUE

4.2.3 int32_t fs_etpu_fuel_set_normal_end_angle(uint8_t channel, uint24_t normal_end_angle)

This function changes the normal end angle. The normal end angle parameter is global for all initialized FUEL channels. This function changes the normal end angle for all generated fuel signals.

This function has the following parameters:

- channel (uint8_t) — The number of any channel that generates the fuel signal, typically the FUEL channel number for the first cylinder.
- normal_end_angle (uint24_t) — This is an angle when the injection should be finished. Range 0 to 71999. 71999 represents 719.99 degrees.

Return notes — The error code that can be returned is:

- FS_ETPU_ERROR_VALUE

4.2.4 int32_t fs_etpu_fuel_set_recalc_offset_angle(uint8_t channel, uint24_t recalc_offset_angle)

This function sets the recalculation offset angle. The recalculation offset angle parameter is global for all initialized FUEL channels. This function changes the recalculation offset angle for all generated fuel signals.

This function has the following parameters:

- channel (uint8_t) — The number of any channel which generates the fuel signal, typically the FUEL channel number for the first cylinder.
- recalc_offset_angle (uint24_t) — (Injection_Start_Angle-Recalculation_Offset_Angle) defines the angle when the re-calc thread is scheduled. Range 0 to 71999. 71999 represents 719.99 degrees.

Return notes — The error code that can be returned is:

- FS_ETPU_ERROR_VALUE

4.2.5 int32_t fs_etpu_fuel_set_compensation_time(uint8_t channel, uint24_t compensation_time_us)

This function changes the compensation time. The compensation time parameter is global for all initialized FUEL channels. This function changes the compensation time for all generated fuel signals.

This function has the following parameters:

- channel (uint8_t) — The number of any channel that generates the fuel signal, typically the FUEL channel number for the first cylinder.
- compensation_time_us (uint24_t) — Each generated injection pulse is prolonged by this time to compensate for the opening and closing time of the valve. The time is measured in microseconds. This value is influenced by valve parameters, temperature, battery voltage, and so on.

4.2.6 int32_t fs_etpu_fuel_set_minimum_injection_time(uint8_t channel, uint24_t minimum_injection_time_us)

This function changes the minimum injection time. The minimum injection time parameter is global for all initialized FUEL channels. This function changes the minimum injection time for all generated fuel signals.

This function has the following parameters:

- channel (uint8_t) — The number of any channel which generates the fuel signal, typically the FUEL channel number for the first cylinder.
- minimum_injection_time_us (uint24_t) — This is the required minimum injection pulse width, in microseconds.

4.2.7 int32_t fs_etpu_fuel_set_minimum_off_time(uint8_t channel, uint24_t minimum_off_time_us)

This function sets the minimum off time. The minimum off time parameter is global for all initialized FUEL channels. This function changes the minimum off time for all generated fuel signals.

This function has the following parameters:

- channel (uint8_t) — The number of any channel which generates the fuel signal, typically the FUEL channel number for the first cylinder.
- minimum_off_time_us (uint24_t) —This is the minimum time between two injection pulses, in microseconds.

4.2.8 int32_t fs_etpu_fuel_switch_off(uint8_t channel)

This function switches the injection pulse generation off. Switching off must not be executed by setting the *Injection Time* = 0, as this shortens an injection pulse that is currently in progress, resulting in an incorrect pulse. Instead this function must be used. In this case the CPU can switch off at any time, but a pulse that has already been started will correctly finished.

It is not possible to switch off all initialized FUEL channels by calling this function, instead the user has to call this function several times based on the number of initialized FUEL channels.

This function has the following parameter:

- channel (uint8_t) — The number of the FUEL channel that needs to be stopped.

4.2.9 int32_t fs_etpu_fuel_switch_on(uint8_t channel)

This function switches the injection pulse generation on. This function is used after the injections were suspended and the CPU requests to turn them back on again.

It is not possible to switch on all initialized FUEL channels by calling this function; instead the user has to call this function several times based on the number of initialized FUEL channels.

This function has the following parameter:

- channel (uint8_t): The number of the FUEL channel which needs to be started.

4.3 Value Return Functions

4.3.1 uint24_t fs_etpu_fuel_get_sum_of_injection_time(void)

This function returns the sum of all injection times (total injection time applied through all initialized FUEL channels) in TCR1 TICKS. This function reads the global variable, it has no input parameter.

4.3.2 uint24_t fs_etpu_fuel_get_CPU_real_injection_time(uint8_t channel)

This function returns the real injection time of the defined cylinder (particular FUEL channel), in the last engine cycle, in TCR1 TICKS.

This function has the following parameter:

- channel (uint8_t) — The number of the defined FUEL channel.

5 Examples of Function Usage

5.1 Example 1

Description — This example demonstrates how to initialize four FUEL channels. The polarity of the fuel signal is active high and cylinder offsets are 0 deg, 180 deg, 360 deg and 540 deg. The injection time of the particular cylinders is set to 4000 microseconds (injection pulses are generated immediately after CAM and CRANK have established synchronization).

Example Code — The example application, which is a part of the FUEL API, shows how to initialize FUEL channels and how to create a simple application with the eTPU set2 FUEL function. The example code is targeted at MPC5500, see *fuel_ example.c* and *fuel_ example.h*.

The eTPU initialization routines *fuel_etpu_gct_example.c* and *fuel_etpu_gct_example.h* were generated by Freescale's eTPU graphical configuration tool (GCT). The GCT is a Windows application that offers a user-friendly graphical environment to configure the eTPU and generate initialization routines coded in C-language. It is available for free at Freescale website.

The following piece of code shows how the eTPU FUEL channels are initialized:

```
err_code = fs_etpu_fuel_init_4cylinders(FUEL0_FUEL_CH_1,/* engine: A; channel: 5 */
                                        FUEL0_FUEL_CH_2,/* engine: A; channel: 6 */
                                        FUEL0_FUEL_CH_3,/* engine: A; channel: 7 */
                                        FUEL0_FUEL_CH_4,/* engine: A; channel: 8 */
                                        APP_ENG_POS0_CAM,/* cam_chan: APP_ENG_POS0_CAM */
                                        FS_ETPU_PRIORITY_MIDDLE,/* priority: Middle */
                                        FS_ETPU_FUEL_PULSE_HIGH,/* polarity: Pulse active high */
                                        0,      /* cylinder_offset_angle_1: 0 */
                                        18000,  /* cylinder_offset_angle_2: 18000 */
                                        36000,  /* cylinder_offset_angle_3: 36000 */
                                        54000,  /* cylinder_offset_angle_4: 54000 */
                                        60000,  /* drop_dead_angle: 60000 */
                                        3000,   /* injection_normal_end_angle: 3000 */
                                        2000,   /* recalculation_offset_angle: 2000 */
                                        4000,   /* injection_time_us_1: 4000 */
                                        4000,   /* injection_time_us_2: 4000 */
                                        4000,   /* injection_time_us_3: 4000 */
                                        4000,   /* injection_time_us_4: 4000 */
                                        1,      /* compensation_time_us: 1 */
                                        2000,   /* minimum_injection_time_us: 2000 */
                                        1000);  /* minimum_off_time_us: 1000 */
```

5.1.1 Program Output

The fuel injection pulses are generated on the eTPU channels 5–8. Figure 3 shows the relationship between the active-high fuel signals and the CAM and CRANK signals.

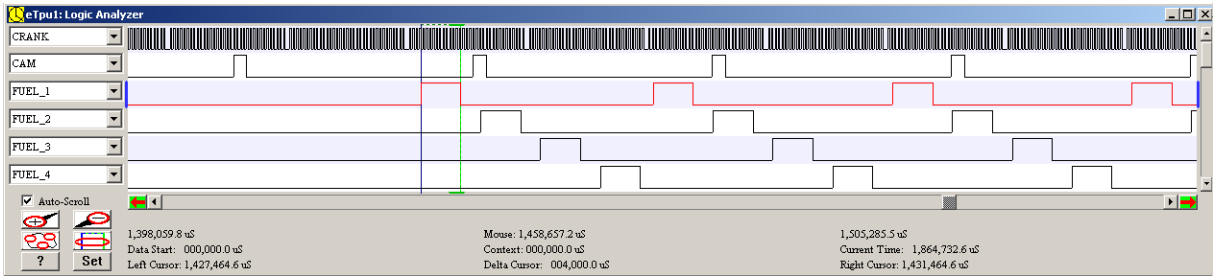

Figure 3. Simulator Outputs - Example 1

5.2 Example 2

Description — This example shows how to initialize three FUEL channels. The polarity of the fuel signal is active low and cylinder offsets are 0 deg, 240 deg, and 480 deg. The injection time of the particular cylinders is set to 0 after the initialization.

Example Code — This example is not a part of the FUEL API but can be created by modifying the example 1 codes. The FUEL channels initialization routine in *fuel_etpu_gct_example.c* must be changed as follows:

```
err_code = fs_etpu_fuel_init_3cylinders(FUEL0_FUEL_CH_1,/* engine: A; channel: 5 */
                                        FUEL0_FUEL_CH_2,/* engine: A; channel: 6 */
                                        FUEL0_FUEL_CH_3,/* engine: A; channel: 7 */
                                        APP_ENG_POS0_CAM,/* cam_chan: APP_ENG_POS0_CAM */
                                        FS_ETPU_PRIORITY_MIDDLE,/* priority: Middle */
                                        FS_ETPU_FUEL_PULSE_LOW,/* polarity: Pulse active low */
                                        0,      /* cylinder_offset_angle_1: 0 */
                                        24000,  /* cylinder_offset_angle_2: 24000*/
                                        48000,  /* cylinder_offset_angle_3: 48000*/
                                        60000,  /* drop_dead_angle: 60000 */
                                        3000,   /* injection_normal_end_angle: 3000 */
                                        2000,   /* recalculation_offset_angle: 2000 */
                                        0,      /* injection_time_us_1: 0 */
                                        0,      /* injection_time_us_2: 0 */
                                        0,      /* injection_time_us_3: 0 */
                                        1,      /* compensation_time_us: 1 */
                                        2000,   /* minimum_injection_time_us: 2000 */
                                        1000);  /* minimum_off_time_us: 1000 */
```

As the injection time of the particular cylinder is set to 0 at initialization, no fuel pulses are generated after the FUEL initialization and the host CPU has to set the injection time of the individual FUEL channels later. The *fs_etpu_fuel_set_injection_time* API function is used each time the CPU requests a change of the injection time on a particular FUEL channel. This function returns 0 if the injection time update was successful. If the FUEL channel has any pending HSRs, this function returns a sum of pending HSR numbers. This function must be called again to apply the requested new value of the injection time. The following technique can be used:

```
error_code = 1;
while (error_code != 0)
{
  error_code = fs_etpu_fuel_set_injection_time(FUEL0_FUEL_CH_1, 4000);
 }
error_code = 1;
while (error_code != 0)
{
  error_code = fs_etpu_fuel_set_injection_time(FUEL0_FUEL_CH_2, 4000);
 }
error_code = 1;
while (error_code != 0)
{
  error_code = fs_etpu_fuel_set_injection_time(FUEL0_FUEL_CH_3, 4000);
 }
```

5.2.1 Program Output

The fuel injection pulses are generated on the eTPU channels 5–8. Figure 4 shows the relationship between the active-low fuel signals and the CAM and CRANK signals.

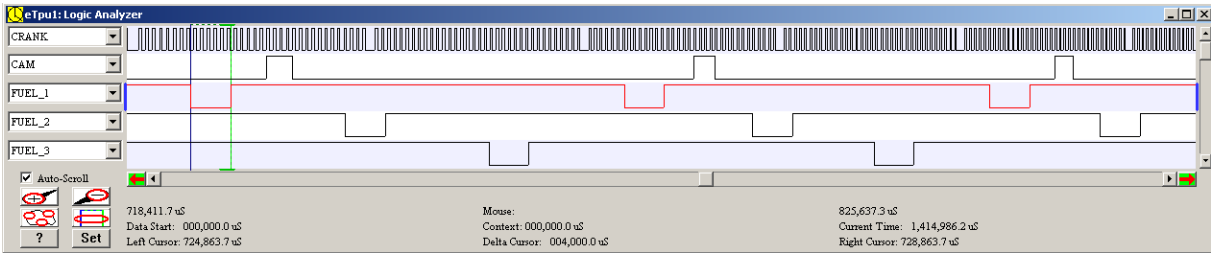

Figure 4. Simulator Outputs - Example 2

6 Summary and Conclusions

This application note provides the user with a description of the eTPU FUEL function (giving code and application examples). The simple C interface functions enable easy implementation of the FUEL function in port injection applications. The functions are targeted at the MPC55xx and MPC563xM family of devices, but can be used with any device that has an eTPU module.

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 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 +1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

freescale™
semiconductor

AN3770
Rev. 0
04/2009