Freescale Semiconductor
Application Note

Document Number: AN2864
Rev. 2, 09/2015

General C Functions for the eTPU

Covers the MCF523x, MPC5500, MPC5600, MPC5700, MPX40 and all eTPU-equipped Devices

by: Milan Brejl and Jeff Loeliger

Contents

1 Introduction

The Enhanced Time Processing Unit (eTPU) is a programmable I/O controller with its own core and memory system, allowing it to perform complex timing and I/O management independently of the CPU. The eTPU module is a peripheral for 32-bit devices on the automotive and industrial markets (currently Power Architecture® MPC5700, MPC5600 and MPC5500 families, PXR40 and the ColdFire MCF523x).

This application note provides simple C interface routines to the eTPU module.

2 Functional Overview

The eTPU Utilities etpu_util.c/.h includes low-level functions for using the eTPU or eTPU2 module. Using these functions prevents the upper software level from direct access to eTPU registers.

The included routines can be divided into several groups by application usage:

- eTPU Module Initialization
  - –fs_etpu_init
  - –fs_etpu2_init (eTPU2-only)
- eTPU Channel Initialization

  - −fs_etpu_chan_init
  - −fs_etpu_malloc
  - −fs_etpu_malloc2

- Run-Time eTPU Module Control
  - −fs_timer_start
  - −fs_etpu_get_global_exceptions, fs_etpu_clear_global_exceptions
  - −fs_etpu_get_global_error

- Run-Time eTPU Channel Control
  - −fs_etpu_get_hsr, fs_etpu_set_hsr
  - −fs_etpu_enable, fs_etpu_disable
  - −fs_etpu_interrupt_enable, fs_etpu_interrupt_disable
  - −fs_etpu_get_chan_interrupt_flag, fs_etpu_clear_chan_interrupt_flag
  - − fs_etpu_get_chan_interrupt_oveflow_flag, fs_etpu_clear_chan_interrupt_oveflow_flag
  - −fs_etpu_dma_enable, fs_etpu_dma_disable
  - −fs_etpu_get_chan_dma_flag, fs_etpu_clear_chan_dma_flag
  - −fs_etpu_get_chan_dma_oveflow_flag, fs_etpu_clear_chan_dma_oveflow_flag

- eTPU DATA RAM Access
  - −fs_etpu_data_ram
  - −fs_etpu_get_chan_local_32, fs_etpu_get_chan_local_24, fs_etpu_get_chan_local_24s, fs_etpu_get_chan_local_16, fs_etpu_get_chan_local_8
  - −fs_etpu_set_chan_local_32, fs_etpu_set_chan_local_24, fs_etpu_set_chan_local_16, fs_etpu_set_chan_local_8
  - −fs_etpu_get_global_32, fs_etpu_get_global_24, fs_etpu_get_global_24s, fs_etpu_get_global_16, fs_etpu_get_global_8
  - −fs_etpu_set_global_32, fs_etpu_set_global_24, fs_etpu_set_global_16, fs_etpu_set_global_8
  - −fs_etpu_coherent_read_32, fs_etpu_coherent_read_24
  - −fs_etpu_coherent_write_32, fs_etpu_coherent_write_24

- eTPU Load Evaluation
  - −fs_etpu_get_idle_cnt_a, fs_etpu_clear_idle_cnt_a (eTPU2-only)
  - fs_etpu_get_idle_cnt_b, fs_etpu_clear_idle_cnt_b (eTPU2-only)

- Others
  - −fs_memcpy32, fs_memset32

When the eTPU is used in an application, the first eTPU task is the eTPU module initialization. The fs_etpu_init() function initializes the eTPU global settings, including uploading the eTPU code into CODE RAM, initialization of time-bases, and more. On eTPU2, the fs_etpu_init2() function can be consequently used to initialize eTPU2-only settings.

Individual eTPU channels can be initialized after the eTPU module initialization. The eTPU channel initialization includes configuration of channel registers (fs_etpu_chan_init()) and allocation of DATA RAM (fs_etpu_malloc(), fs_etpu_malloc2()). This task requires detailed information about the eTPU function design. That is why each Freescale eTPU function comes with API routines for the CPU, which handles the channel initialization and run-time interfaces. The channel initialization API routine must use the fs_etpu_malloc() or fs_etpu_malloc2() for proper allocation of the eTPU DATA RAM.

After the eTPU module and eTPU channels are initialized, all eTPU channels can be started synchronously by fs_timer_start(), which starts global time-bases common for eTPU and eMIOS. The run-time eTPU module control functions include also routines to handle the global module exceptions (fs_etpu_get_global_exceptions(), fs_etpu_clear_global_exceptions()) and global error (fs_etpu_get_global_error()).

The run-time eTPU channel control functions include enabling, disabling, checking and clearing channel interrupts and channel DMA requests, enabling and disabling a channel operation and passing a host service request to a channel.

The eTPU function API routines uses the eTPU DATA RAM access functions to read or write channel-local or global eTPU variables. The eTPU features hardware for coherent read or write of 2 variables in DATA RAM. Its usage is supported by fs_etpu_coherent_read/write_32/24().

The eTPU2 features a support for eTPU load evaluation. There are Idle Counter registers on each engine which can be accessed using fs_etpu_get/clear_idle_cnt_a/b().

3 Functional Description

A detailed description of each eTPU utility function is included in the source code. For easier reading, documentation in a format of Microsoft Compiled HTML Help (CHM) is generated by DoxyGen and included with the source code in AN2864SW package as doxygen.chm. Additionally, the same documentation in a form of a set of HTML files packed into a single ZIP package is included.

4 eTPU Initialization Template

A template of eTPU initialization code is included in AN2864SW package, in folder init_template. The file etpu_gct.c_ creates two functions to be used by the application in order to initialize the eTPU. The function my_system_etpu_init() should be called within initialization of device modules. The function my_system_etpu_start() should be called after all device modules, including the interrupt and DMA controller, are configured. This function effectively starts the eTPU.

The file etpu_gct.c_ also defines an eTPU configuration structure (my_etpu_config) which includes all global eTPU settings. The file etpu_gct.h_ includes an assignment of eTPU functions to channels and definition of masks enabling channel interrupts, DMA requests and “output disable” feature for individual channels. Initialization of individual channel functions is assumed to be handled by function API calls, which is indicated in my_system_etpu_init().

5 eTPU Project Structure

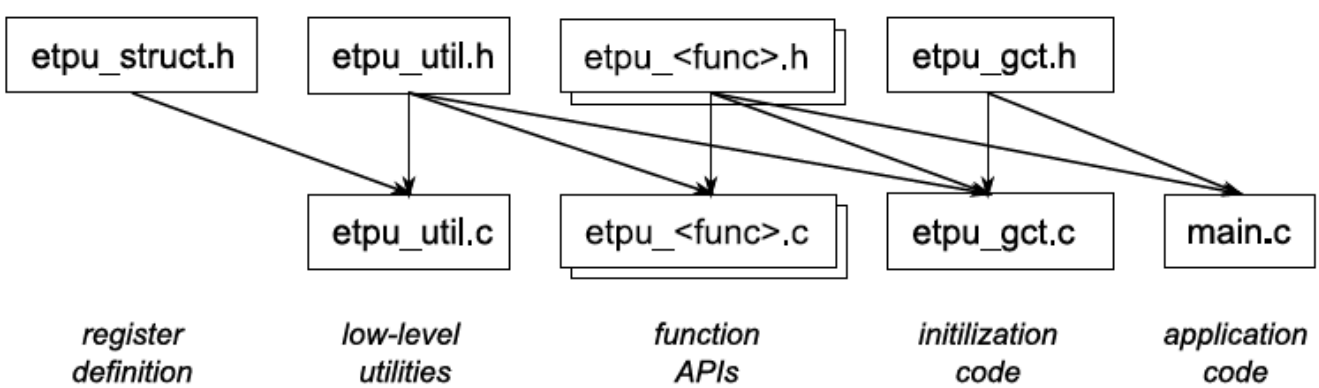

Figure 1. eTPU project Structure

An application using the eTPU module should include the following files:

- etpu_struct.h, which defines the eTPU registers
- etpu_util.c/.h, which includes low-level eTPU utilities
- etpu_<func>.c/.h, which are the eTPU function APIs, handling the initialization and run-time interface of the individual eTPU functions, using the eTPU low-level utilities

- etpu_gct.c/.h, which includes eTPU initialization code specific for the user application, implemented using eTPU utilities and API calls
- main.c, which stands for the actual application code, implemented using etpu_gct function calls for initialization and API calls for run-time eTPU interface.

6 Summary

This application note provides overview of general C functions for the eTPU (eTPU utilities) and their usage. The source code of these functions is available for download from the Freescale web – package AN2864SW. A detailed description of individual utility functions is included with the source code (doxydoc.chm). The eTPU utilities can be used on any device featuring the eTPU module. Both module versions, eTPU and eTPU2, are supported.

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number AN2864
Revision 2, 09/2015