Freescale Semiconductor
Application Note

Document Number: AN2854
Rev. 1, 10/2008

Using the Synchronized Pulse-Width Modulation eTPU Function

by: Geoff Emerson
Microcontroller Solutions Group

This application note is intended to describe simple C interface routines to the synchronized pulse-width modulation (SPWM) eTPU function. The function 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 Function Overview

The synchronized pulse-width modulation function generates a pulse-width modulated (PWM) waveform in which the CPU can change the period or high time at any time. When synchronized to a function on a second channel, SPWM low-to-high transitions have a configurable relationship to transitions on the second channel.

The eTPU SPWM function is loosely based on the SPWM TPU function. The SPWM eTPU function offers

© Freescale Semiconductor, Inc., 2008. All rights reserved.

freescale™ semiconductor

the following enhancements over the SPWM TPU function:

- 22-bit values are supported (versus 15-bit offset value on the TPU3).
- The dual action hardware of the eTPU is used to accommodate reduced latencies, and the complete range of duty cycles between 0 and 100 can be accommodated.
- Slave channels can have delays of up to one period.

2 Functional Description

The following definitions are used in the SPWM descriptions:

- Synchronization means that a relationship exists between waveforms occurring on different channels.
- A link means that a signal (link service request) has been sent from a master channel (linking channel) to a slave channel (linked channel).

SPWM channels may be configured as either master or slave. A master mode channel can be used to provide links to up to eight slave channels. Master channels should not be sent links. When a slave channel receives a link, it will generate slave edges relative to a reference time (stored in Data RAM). It need not be an SPWM channel, which provides the link for the slave channel. Other functions, such as eTPU IC function, may be used to generate the link. This means that a periodic signal may be “captured” using the IC function, and slave SPWM channels can then be synchronized to the periodic signal.

The value of the free-running timer counter register (TCR) is stored in a Data RAM location by the eTPU SPWM function when a rising edge occurs. This Data RAM location is referred to as RisingEdge. If the SPWM function is being used as master, then the slave channels must be synchronized to the master channel. This is achieved by populating a pointer (called *MasterRisingEdgePtr) on the slave channel with the address of the master channel’s RisingEdge variable. See the example program described in Section 5, “Examples of Function Use,” for an example of how this is achieved.

When a rising edge occurs on the master channel, the SPWM function generates an interrupt and DMA request. Optionally the falling edge of the master channel can generate an interrupt and DMA request. The DMA request signal for a given channel may not be connected; the connection depends on the specific integration of the eTPU. The application programmer can use either the DMA request or the interrupt request, depending on the specific integration.

Coherent updates of Period and Duty cycle are not supported by the SPWM function.

Figure 1 shows how master and slave channels operate together.

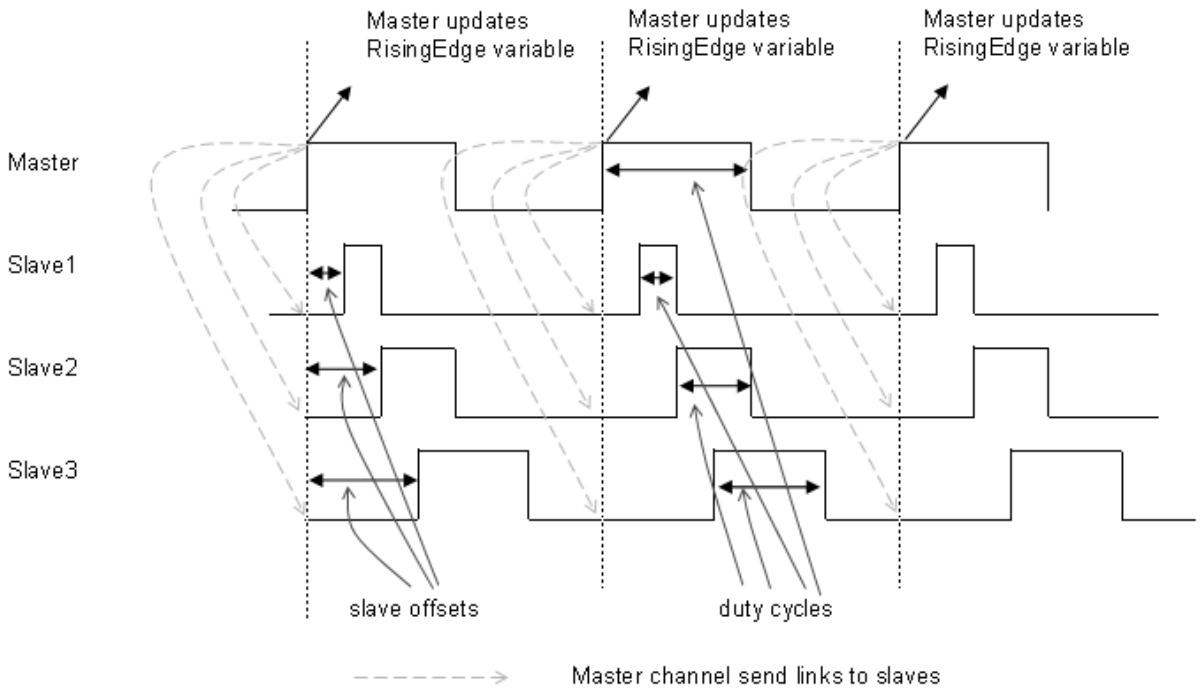

Figure 1. Basic Operation of Master and Slave SPWM Channels

Each channel has its own duty cycle. Each slave channel has its own offset to the rising edge of the master.

2.1 Notes on Performance and Use of eTPU SPWM Function

2.1.1 Performance

Like all eTPU functions, the SPWM 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. Increased eTPU loading will potentially result in increased latencies. 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 SPWM software release, available from Freescale.

2.1.2 Changing Operation Modes

To reconfigure the SPWM 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.

3 C Level API for eTPU SPWM Function

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

The API consists of 6 functions:

1. Master initialization routine: fs_etpu_spwm_init_master
2. Slave initialization routine: fs_etpu_spwm_init_slave
3. Master run routine: fs_etpu_spwm_run_master
4. Change duty cycle of master or slave channel: fs_etpu_spwm_duty
5. Change frequency and duty of a master channel: fs_etpu_spwm_update_master
6. Return master channel frequency: fs_etpu_spwm_get_freq_master

3.1 Master Initialization Routine: fs_etpu_spwm_init_master

```
int32_t fs_etpu_spwm_init_master ( uint8_t channel,
                                   uint32_t freq,
                                   uint16_t duty,
                                   uint8_t timebase,
                                   uint32_t timebase_freq,
                                   uint8_t reference_mode,
                                   uint32_t *reference_ptr,
                                   uint8_t INT_DMA_on_falling_edge,
                                   uint32_t link1,
                                   uint32_t link2
```

This routine is used to initialize a channel to use the SPWM function as a master.

In order for the SPWM 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 example 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 the initialization routine is called 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 the 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 channel's data memory.

This function has the following parameters:

- channel (uint8_t): The SPWM master 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.
- freq (uint32_t): This is the frequency of the SPWM. The range of this parameter is determined by the complete system but normally would be between 1 Hz–100 kHz.
- duty (uint16_t): This is the initial duty cycle of the SPWM. Duty has a range of 0–10000 to represent 0–100 % with 0.01 % resolution. So Duty = 7550 would result in a duty cycle of 75.5 %.
- timebase (uint8_t): This is the timebase which the SPWM channel will use. This parameter should be assigned one of the following values (definitions are found in the utilities file etpu_util.h):
  - FS_ETPU_TCR1
  - FS_ETPU_TCR2
- timebase_freq (uint32_t): This is the pre-scaled frequency of the timebase (either TCR1/TCR2), supplied by the eTPU to the function.
- reference_mode (uint8_t): This determines whether the function will run in immediate mode or relative to a timing reference stored in Data RAM. This parameter should be assigned a value of:
  - FS_ETPU_SPWM_REF_IN_PRAM
  - FS_ETPU_SPWM_IMMEDIATE

  In immediate mode, the first rising edge on the master channel is scheduled to happen one period ahead of the current value of the selected TCR. The value of the selected TCR is captured by the fs_etpu_spwm_run_master API routine and stored to the eTPU Data RAM. When the host service request is sent by the API the eTPU SPWM function accesses this stored TCR value in order to calculate and schedule the first rising edge.

  In reference_in_pram mode, the eTPU SPWM function accesses a Data RAM location as specified by the *reference_ptr API parameter in order to calculate and schedule the first rising edge. The master channel's first rising edge will be one period beyond the value stored in the Data RAM location. Care must be taken to ensure that the reference value is not in the past; otherwise incorrect operation may occur.
- reference_ptr (uint32_t *): This is the address of the reference when reference_mode is FS_ETPU_SPWM_REF_IN_PRAM.
- INT_DMA_on_falling_edge (uint8_t): Determines if an interrupt/DMA request is generated on falling edge or not. This parameter should be assigned a value of:
  - FS_ETPU_SPWM_NO_FALLING_EDGE_INT_DMA,
  - FS_ETPU_SPWM_FALLING_EDGE_INT_DMA
- link1 (uint32_t): This is a packed 32-bit parameter consisting of 4 × 8-bit channel numbers.
- link2 (uint32_t): This is a packed 32-bit parameter consisting of 4 × 8-bit channel numbers. Refer to Section 4.6, "Linking Less Than 8 Channels."

3.2 Slave Initialization Routine: fs_etpu_spwm_init_slave

```
int32_t fs_etpu_spwm_init_slave (uint8_t channel,
                                 uint8_t priority,
                                 uint32_t freq,
                                 uint16_t duty,
                                 uint32_t delay,
                                 uint8_t timebase,
                                 uint32_t timebase_freq,
                                 uint32_t *MasterRisingEdgePtr
          )
```

This routine is used to initialize a channel to use the SPWM function as a slave.

In order for the SPWM 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. With automatic allocation, as each channel is initialized 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 used in all of the example 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 the initialization routine is called 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 channel's data memory.

This routine has the following parameters:

- channel (uint8_t): The SPWM master 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 SPWM 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
- freq (uint32_t): This is the frequency of the SPWM. The range of this parameter is determined by the complete system but normally would be between 1 Hz – 100 kHz. If being used in conjunction with fs_etpu_spwm_init_master the freq parameters must have the same value.
- duty (uint16_t): This is the initial duty cycle of the SPWM. Duty has a range of 0–10000 to represent 0–100 % with 0.01 % resolution. So Duty = 7550 would result in a duty cycle of 75.5 %.

- delay (uint32_t): This is the delay in micro-seconds of the rising edge of the slave relative to the rising edge of the master or reference.
- timebase (uint8_t): This is the timebase which the SPWM channel will use. This parameter should be assigned one of the following values (definitions are found in the utilities file etpu_util.h):
  - FS_ETPU_TCR1
  - FS_ETPU_TCR2

  If being used in conjunction with fs_etpu_spwm_init_master the timebase parameters must have the same value.
- timebase_freq (uint32_t): This is the pre-scaled frequency of the timebase (either TCR1/TCR2), supplied by the eTPU to the function. If being used in conjunction with fs_etpu_spwm_init_master the timebase_freq parameters must have the same value.
- *MasterRisingEdgePtr (uint32_t *): The address of the variable on the eTPU which stores the time of the next rising edge of the master channel.

3.3 Run Master Routine: fs_etpu_spwm_run_master

```
void fs_etpu_spwm_run_master ( uint8_t channel,
                               uint8_t priority )
```

This routine sets the master SPWM channel running, which in turn will send links to the slave channels. The fs_etpu_spwm_init_master and the fs_etpu_spwm_init_slave routines must have been run before this routine is called.

This routine has the following parameters:

- channel (uint8_t): The SPWM master 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 SPWM master 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

3.4 Set Duty Cycle: fs_etpu_spwm_duty

```
void fs_etpu_spwm_duty( uint8_t channel,
                        uint16_t duty)
```

This routine updates a channel's duty cycle. It is used for both master and slave channels. This function has the following parameters:

- Channel (uint8_t): The SPWM 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.

- duty (uint16_t): This is the initial duty cycle of the SPWM. Duty has a range of 0–10000 to represent 0–100 % with 0.01 % resolution. So Duty = 7550 would result in a duty cycle of 75.5 %.

3.5 Update Master Frequency and Duty Cycle: fs_etpu_spwm_update_master

```
int32_t fs_etpu_spwm_update_master( uint8_t channel,
                                    uint32_t freq,
                                    uint16_t duty,
                                    uint32_t timebase_freq
                                  )
```

This routine updates a master channel's duty cycle and frequency. This function has the following parameters:

- Channel (uint8_t): The SPWM master 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.
- freq (uint32_t): This is the frequency of the SPWM. The range of this parameter is determined by the complete system but normally would be between 1 Hz–100 kHz.
- duty (uint16_t): This is the initial duty cycle of the SPWM. Duty has a range of 0–10000 to represent 0–100% with 0.01 % resolution. So Duty = 7550 would result in a duty cycle of 75.5%.
- timebase_freq (uint32_t): This is the pre-scaled frequency of the timebase (either TCR1/TCR2), supplied by the eTPU to the function.

3.6 Interrogate Frequency of a Master Channel: fs_etpu_spwm_get_freq_master

```
uint32_t fs_etpu_spwm_get_freq_master( uint8_t channel,
                                       uint32_t timebase_freq
                                     )
```

This routine returns the frequency in Hertz of the master channel's frequency. This function has the following parameters:

- Channel (uint8_t): The SPWM master 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.
- timebase_freq: This is the pre-scaled frequency of the timebase (either TCR1/TCR2), supplied by the eTPU to the function.

4 Notes and Limitations on Use of SPWM Function

4.1 API Routines: Order of Use

The API routines need to be called in a particular order. This order is:

1. fs_etpu_spwm_init_master
2. fs_etpu_spwm_init_slave

3. fs_etpu_spwm_run_master

This is because the slave initialization API sets up a pointer to the master channel's RisingEdge variable. The location of this variable in Parameter RAM is not known until after the master channel's initialization.

4.2 Stopping and Restarting a Master Channel

If the master channel is to be stopped and restarted, the slave channels must be disabled prior to stopping the master channel. The slave channels must be re-enabled prior to re-enabling the master channel.

It must be confirmed that each slave channel is not currently being serviced after it has been disabled, before proceeding. Disabling a channel prevents future service and will not stop a currently running thread.

For example, if channel SPWM0 were a master and channels SPWM1, SPWM4, and SPWM5 were slaves, then the master and slaves must be disabled and re-enabled in the following manner:

```
fs_etpu_disable(SPWM1);
/* wait if channel is being serviced */
while ((ETPU.CSSR_A.R >> SPWM1 ) &0x1 == 0x1 );

fs_etpu_disable(SPWM4);
/* wait if channel is being serviced */
while ((ETPU.CSSR_A.R >> SPWM4 ) &0x1 == 0x1 );

fs_etpu_disable(SPWM5);
/* wait if channel is being serviced */
while ((ETPU.CSSR_A.R >> SPWM5 ) & 0x1 == 0x1 );

/* now that all slave channels have been successfully stopped
   it is safe to stop the master channel*/
fs_etpu_disable(SPWM0);
fs_etpu_enable(SPWM1,FS_ETPU_PRIORITY_MIDDLE);
fs_etpu_enable(SPWM4,FS_ETPU_PRIORITY_MIDDLE);
fs_etpu_enable(SPWM5,FS_ETPU_PRIORITY_MIDDLE);

fs_etpu_spwm_run_master ( SPWM0, FS_ETPU_PRIORITY_MIDDLE);
```

4.3 Timing of Updates

When routines fs_etpu_spwm_duty or fs_etpu_spwm_update_master are used to update either the duty cycle or frequency and duty cycle of a channel, the new values become effective in two SPWM cycles' time. This is because the SPWM function calculates and schedules the new edges one period ahead. It also takes one cycle for the new values to be 'latched' by the SPWM function. Figure 2 shows this behavior.

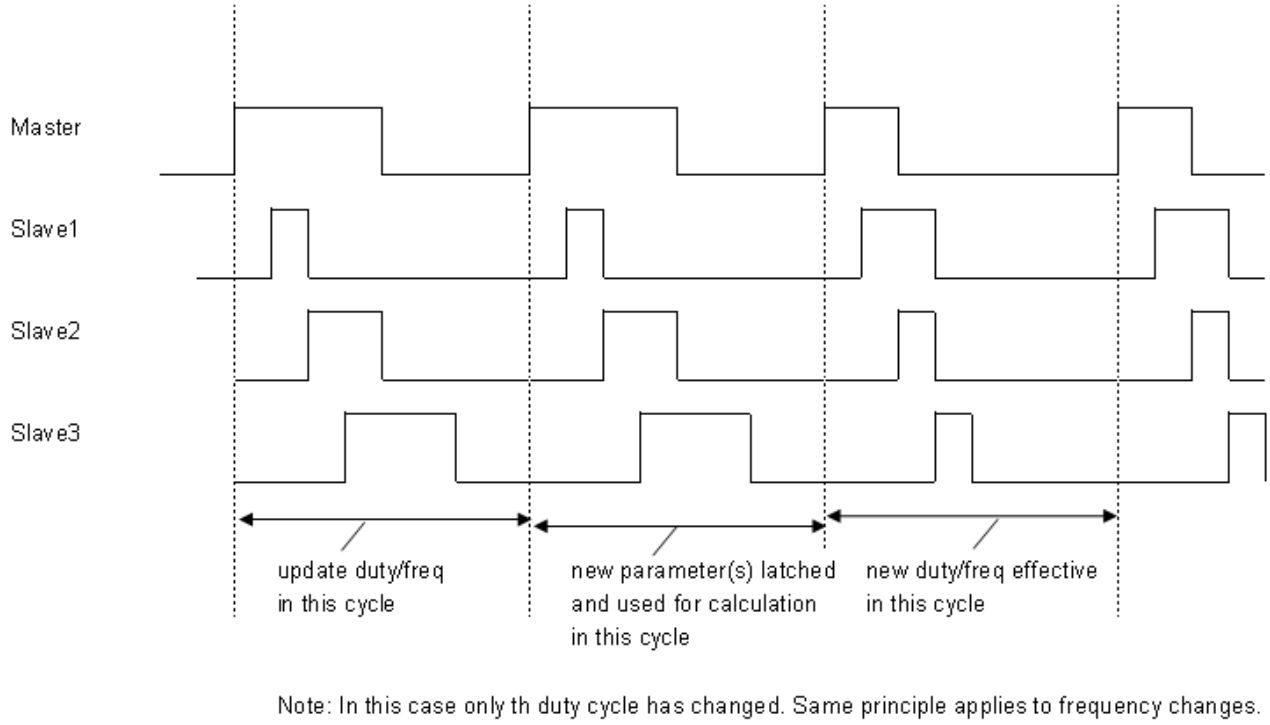

Figure 2. Timing of Updates

4.4 Duty of Slave Channel Not Automatically Updated when Master Channel Frequency Changed

When the fs_etpu_spwm_update_master routine is used to change a master channel's frequency, a new value for the eTPU SPWM function variable ActiveTime is calculated and written by the API. This means that, assuming duty cycle is not changed, the high time of the master channel will be the same proportion of the old and new periods. However, the slave channel's ActiveTime parameter is not recalculated as this is a CPU (host side) calculation. Hence it is the application programmer's responsibility to update the duty cycle of the slave channels by using the fs_etpu_spwm_duty routine.

4.5 Links Must be Received on a Periodic Basis

When slave channels have been configured to be synchronized to a reference source that is something other than an SPWM master channel, it is important that the links be generated on a once-per-period basis. If two slave frame edges occur and no link is received between them by the slave channel, then the reference for the slave will be one period behind and the SPWM function will not behave correctly. An error state will be entered and the channel will stop.

4.6 Linking Less Than 8 Channels

If fewer than eight slave channels are being synchronized by the master, then parameters link1 and link2 of the API routine fs_etpu_spwm_init_master must be padded with either a disabled channel (which cannot respond to the link it receives) or to the first channel being linked.

For example, if channels 8, 15, and 31 are to be synchronized to the master SPWM channel, then link1 = 0x080F1F08 and link2 = 0x08080808. The channel used for padding must reside on the same eTPU engine as the master channel.

4.7 Channel Priority and Channel Numbers

The master channel and all related slave channels must have equal priority. In addition, the master channel number must be numerically less than any of the slave channel numbers. These conditions ensure that if a master and slave channel are both requesting service, then the master channel will be serviced first. This can happen if a slave channel has a small delay parameter.

4.8 Initial Behavior when Using Reference Other Than SPWM Master Channel

The initial behavior of slave channels is different depending on what the reference (*MasterRisingEdgePtr) source is. If the reference source is an SPWM master, then the first master pulse will have a corresponding slave pulse on each slave channel. If the reference is something other than an SPWM master channel, then the first slave pulse will be delayed by one period. If the reference is something other than an SPWM master channel, then the delay parameter that is specified to the fs_eptu_slave_init() routine must be more than one period.

Figure 3 compares these behaviors.

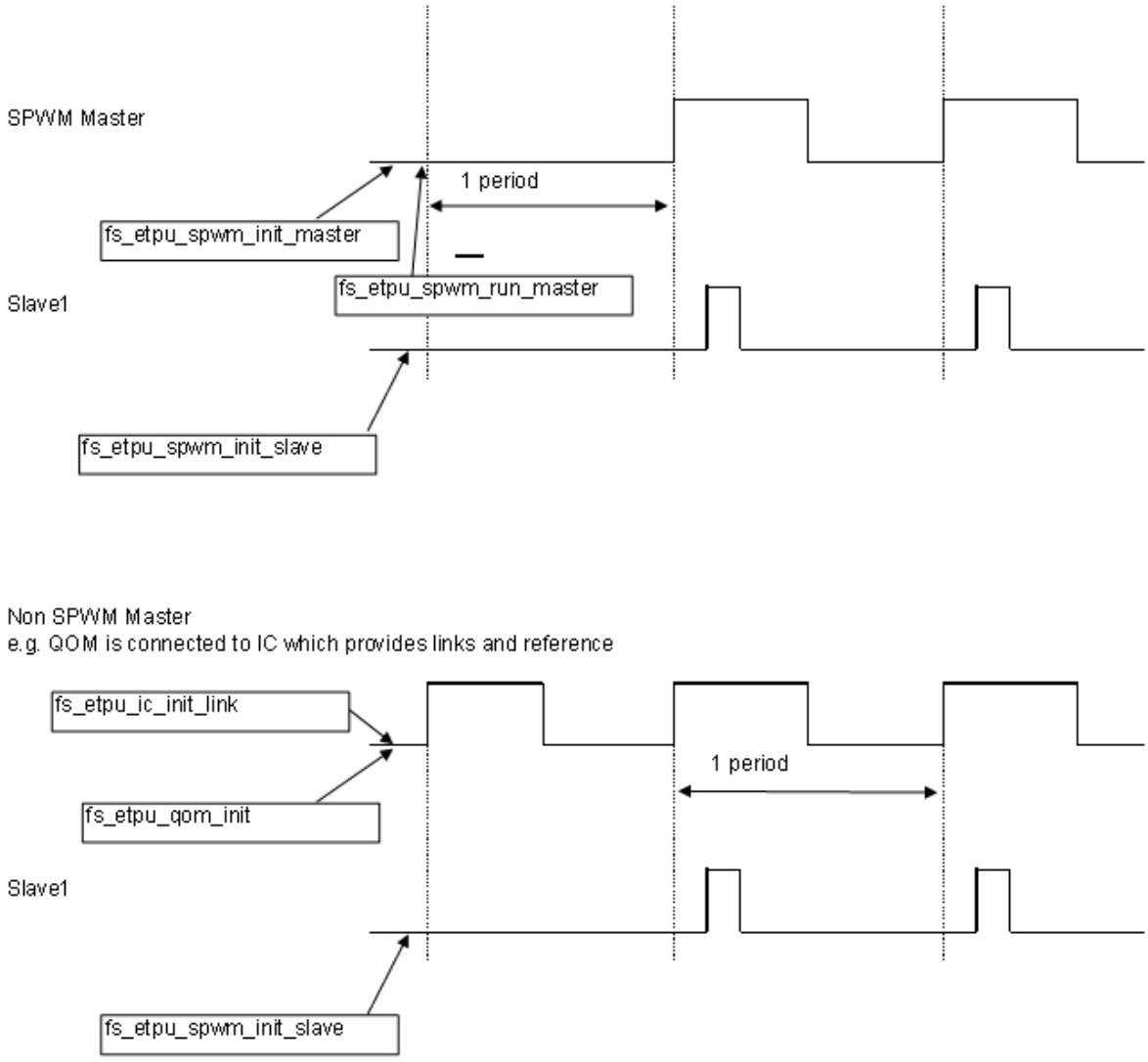

Figure 3. Behavior with SPWM Master and Non-SPWM Reference

5 Examples of Function Use

This section describes a simple use of the SPWM function, as well as how to initialize the eTPU module and assign the eTPU SPWM function to eTPU master and slave channels.

The example consists of two files:

- SPWM_example1.h
- SPWM_example1.c

These files are contained in the SPWM API software file AN2854SW, available at the Freescale.com website.

File SPWM_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 SPWM_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.

5.1 Functionality Description

Channel SPWM0 is initialized to run the SPWM function as master. Its frequency is 10 KHz with a 50% duty cycle. TCR1 is selected as the timebase and the timebase frequency is set at 8 MHz. The initial reference is immediate (i.e. relative to current value of TCR1). An interrupt will be generated on each falling edge of the master SPWM signal. Links will be sent to channel 1, 4 and 5 on each SPWM rising edge.

The memory address where the TCR count of the master's rising edge is stored is derived (Master_rising_edge_ptr). This will be used by the slave channels.

Slave channels SPWM1, SPWM4, and SPWM5 are initialized to run the SPWM function in slave mode. They will have duty cycles of 25 %, 12 %, and 6 % respectively. The delays for these channels are programmed to be 10 µs, 20 µs, and 30 µs respectively. The frequency is the same as for the master channel (10 KHz), the timebase is the same as for the master channel (TCR1), and the timebase frequency is the same as for the master channel (8 MHz). The MasterRisingEdgePtr parameter is set to the previously derived Master_rising_edge_ptr. This ensures that the slave channels are synchronized to the rising edge of the SPWM master channel (SPWM0).

5.2 Sample Program Output

Figure 4 shows the output of the example program as captured by an oscilloscope.

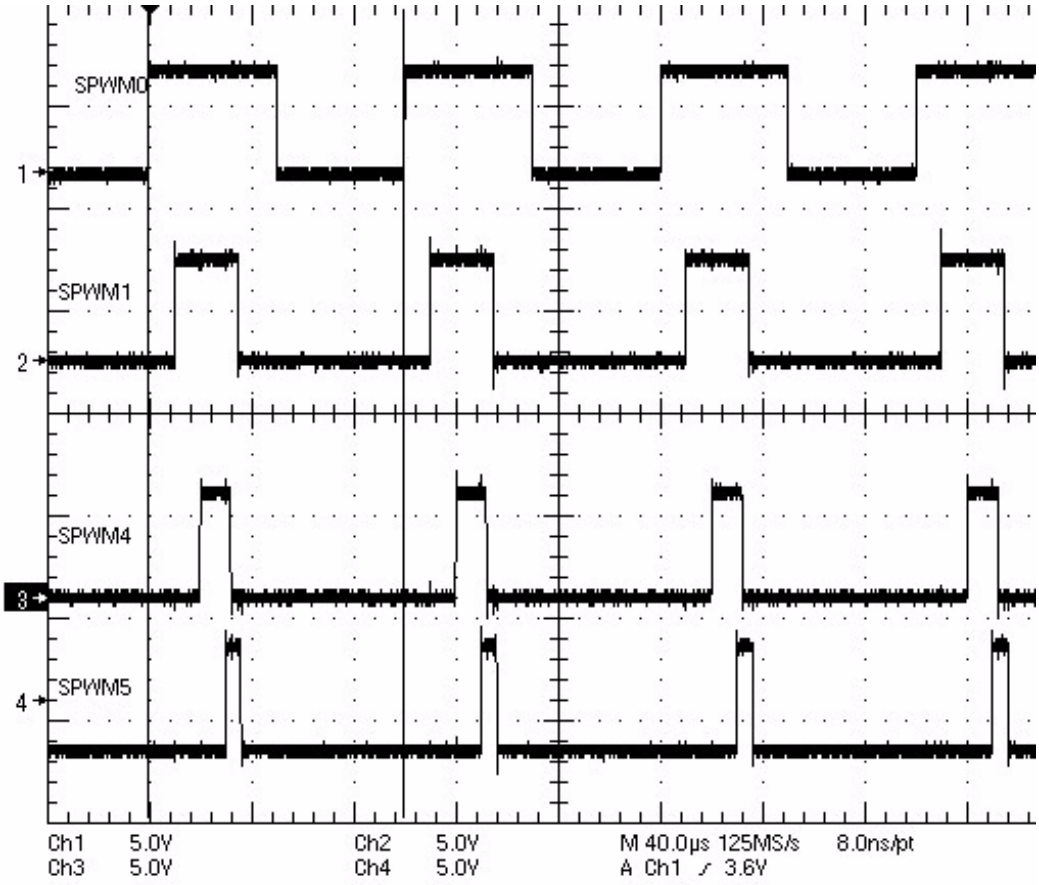

Figure 4. Sample Program Output

6 Summary and Conclusion

This eTPU SPWM application note provides a description of the output compare eTPU function usage and examples of its use. The simple C interface routines to the SPWM eTPU function enable easy implementation of the SPWM function in applications. The functions are targeted for the MPC5500 and the MCF53x families of devices, but they can be used with any device that contains an eTPU.

THIS PAGE IS INTENTIONALLY BLANK

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: AN2854
Rev. 1
10/2008

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typicals”, must be validated for each customer application by customer’s technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics as their non-RoHS-compliant 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.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.
© Freescale Semiconductor, Inc. 2008. All rights reserved.

freescale™
semiconductor