Freescale Semiconductor
Application Note

Document Number: AN2841
Rev. 1, 04/2005

Using the Hall Decoder (HD) eTPU Function

Covers the MCF523*x*, MPC5500, and all eTPU-equipped Devices

by: Milan Brejl, Michal Princ
System Application Engineers, Roznov Czech System Center
Valeriy Phillipov
System Application Engineer, Kiev Embedded Software Lab

1 Introduction

The Hall decoder (HD) enhanced time processor unit (eTPU) function is one of the functions included in the DC motor control eTPU function set (set3). This application note is intended to provide simple C interface routines to the HD eTPU function. The routines are targeted at the MCF523*x* and MPC5500 families of devices, but they could be easily used with any device that has an eTPU.

Table of Contents

2 Function Overview

The HD eTPU function is intended to process signals generated by Hall sensors in motion control systems. It uses one, two, three, or four channels to decode Hall sensor signals, and to produce position and direction information for the CPU. HD also provides a capability to commutate motor phases driven by the PWMC function.

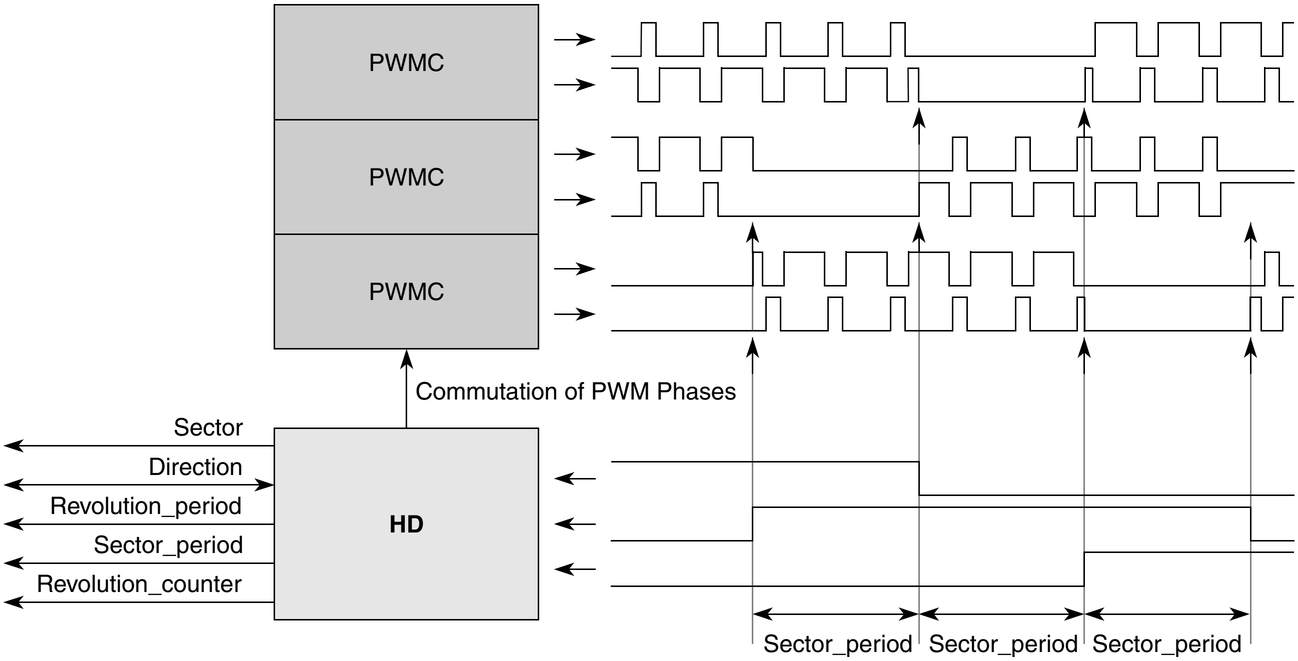

Figure 1. Functionality of HD

3 Function Description

The HD function uses one to four eTPU channels configured as inputs. The primary purpose of this function is to decode the signals derived from Hall effect sensors used with a brushless motor, and perform commutations of PWM phases. HD function calculates the following parameters for the CPU:

- **Sector** - This parameter determines the position of the motion system in one of the sectors. Sector parameter value is encoded based on the input signal states; see Figure 2.

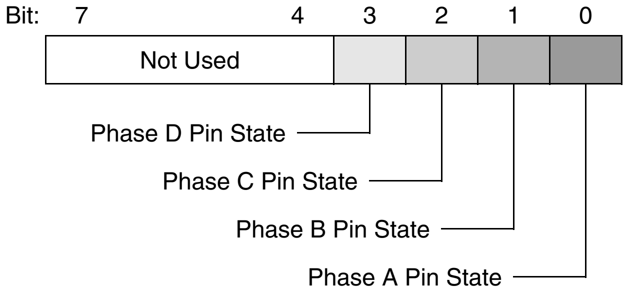

Figure 2. Encoding Sector Parameter

- **Direction** - This parameter determines the direction of the motion system either positive (incremental) or negative (decremental).
  The direction is calculated only in configurations with 3 or 4 input channels (phases).
- **Revolution Counter** - This parameter determines the number of electrical revolutions. The revolution counter is incremented or decremented on each revolution, based on the current direction. The revolution counter is calculated only in configurations with 3 or 4 input channels (phases).

- **Revolution Period** - This parameter determines the TCR time of the last revolution. The parameter value is updated each time the sector is changed. The revolution period is measured from the last edge of similar type (low-high / high-low), on the same channel, to the current edge.
- **Sector Period** - This parameter determines the TCR time between the last two changes of the sector. The parameter value is updated each time the sector is changed. The sector period is measured from the last edge to the current edge; see Figure 3.

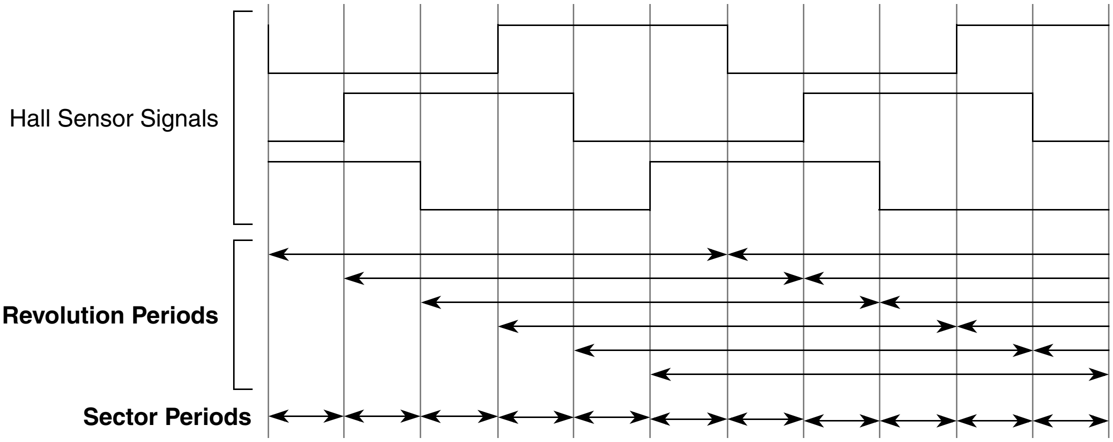

Figure 3. Measuring Revolution Period and Sector Period

- **Last Edge Time** - This parameter stores the TCR time of the last incoming edge.

The HD function is capable of performing commutations to PWMC phases. A detailed description of commutation is in Section 3.2, “Commutation.”

3.1 Example Configurations

The HD function has been designed to provide as much flexibility as possible in processing the Hall sensor signals and driving the commutations. This flexibility may allow the HD function to meet the needs of unusual drive schemes, for example, when used in driving an SR motor. However, since the primary purpose of the HD function is to drive BLDC motors in a conventional manner, there are several pre-defined configurations. The Hall sensor positions on stator and generated Hall sensor signals of the pre-defined configurations are illustrated in Figure 4 through 9. Each of the pre-defined configuration covers the number of phases, internal masks for setting the direction, and other settings.

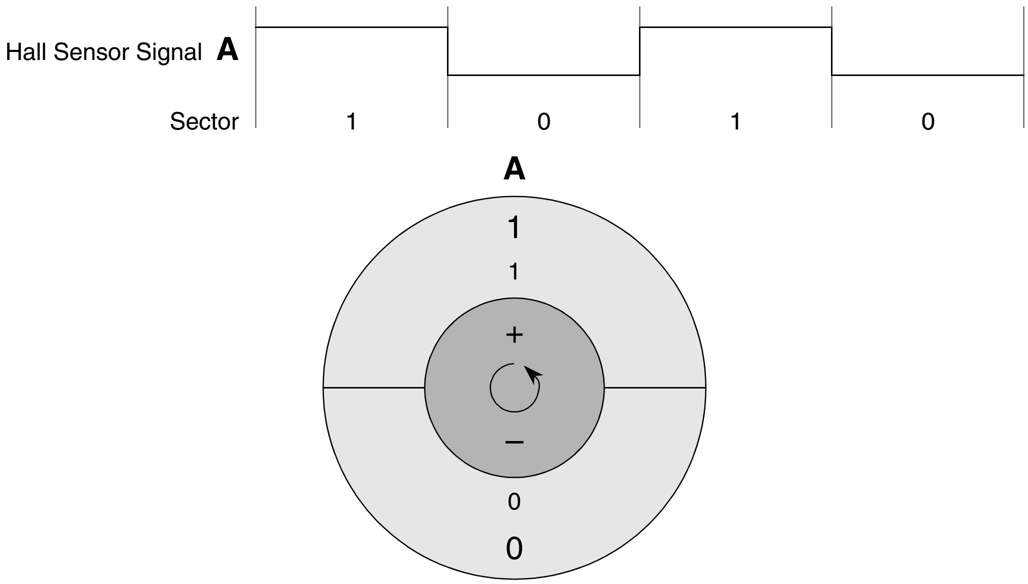

Figure 4. Pre-defined Configurations: One Phase

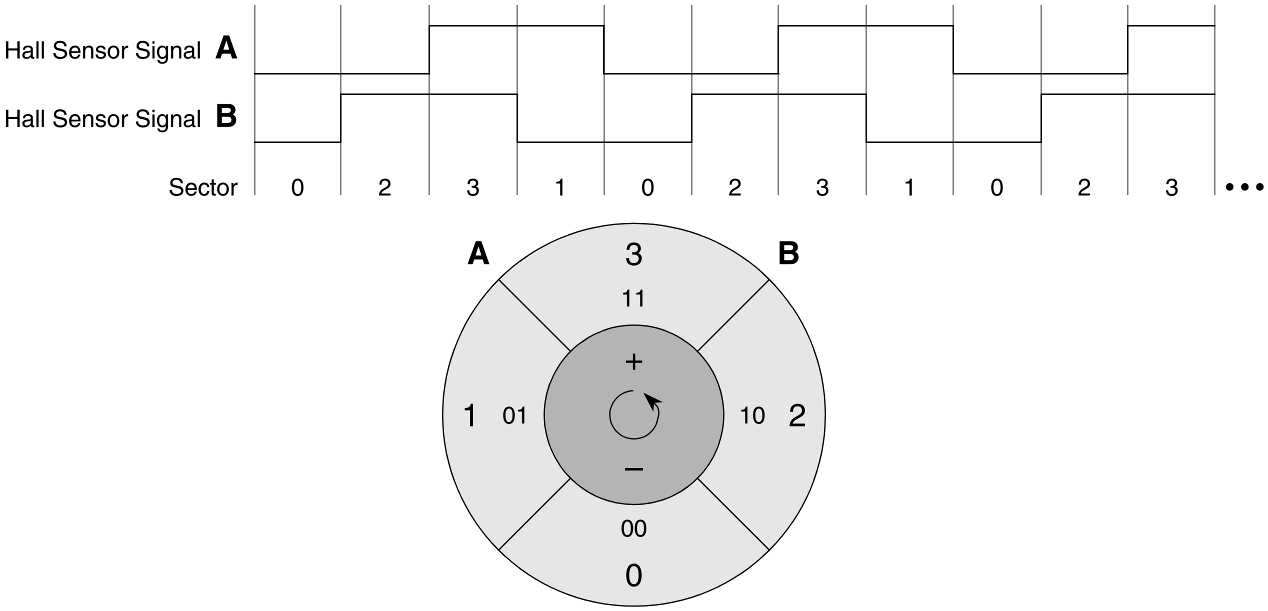

Figure 5. Pre-defined Configurations: Two Phases

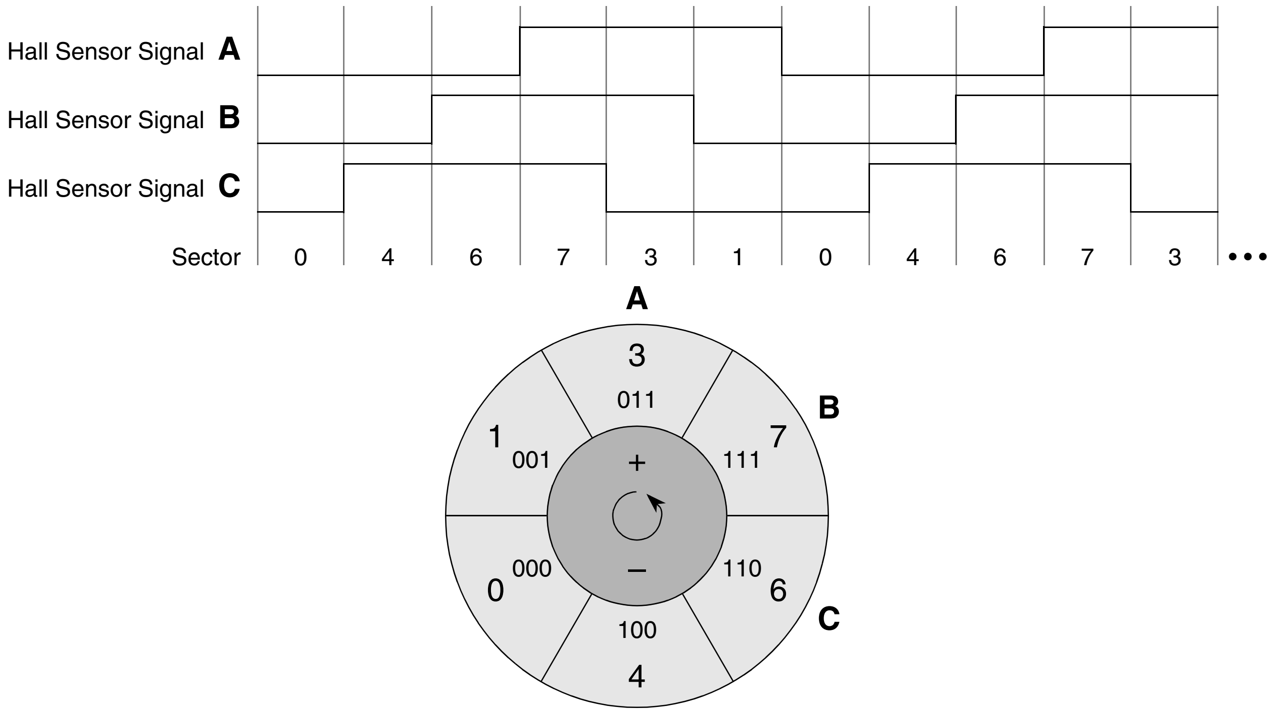

Figure 6. Pre-defined Configurations: Three Phases - by 60 Deg

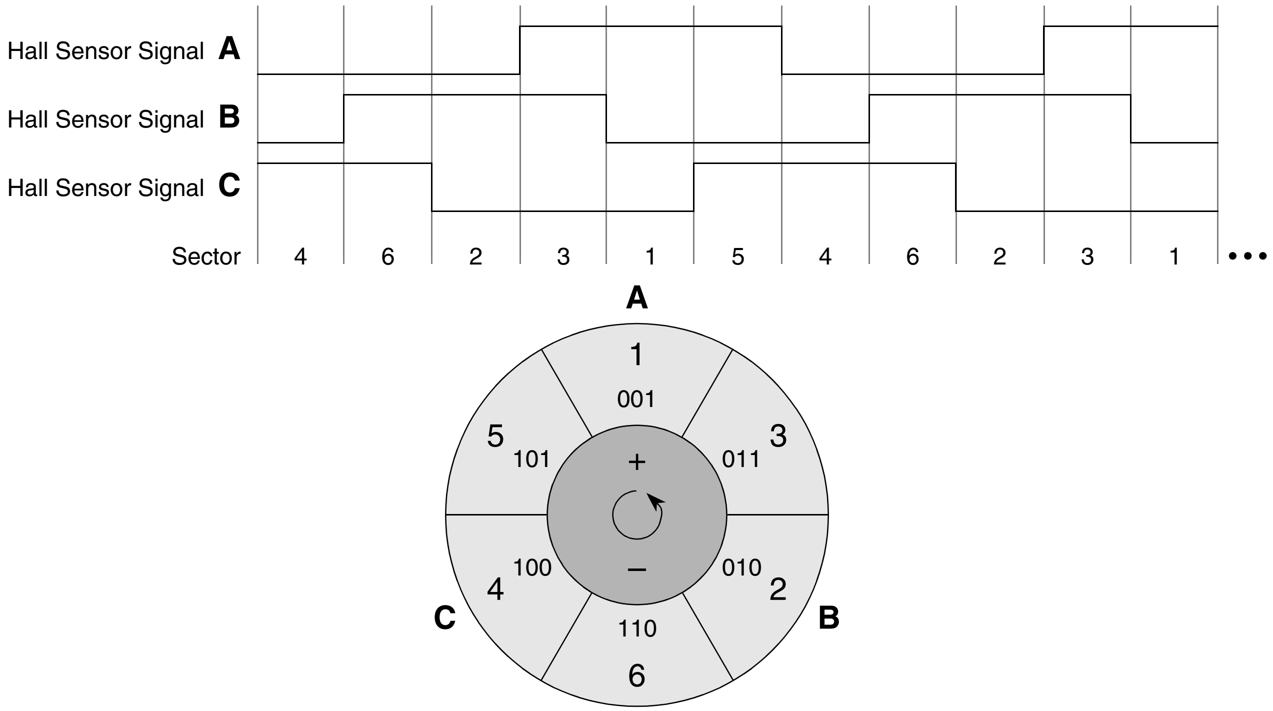

Figure 7. Pre-defined Configurations: Three Phases - by 120 Deg

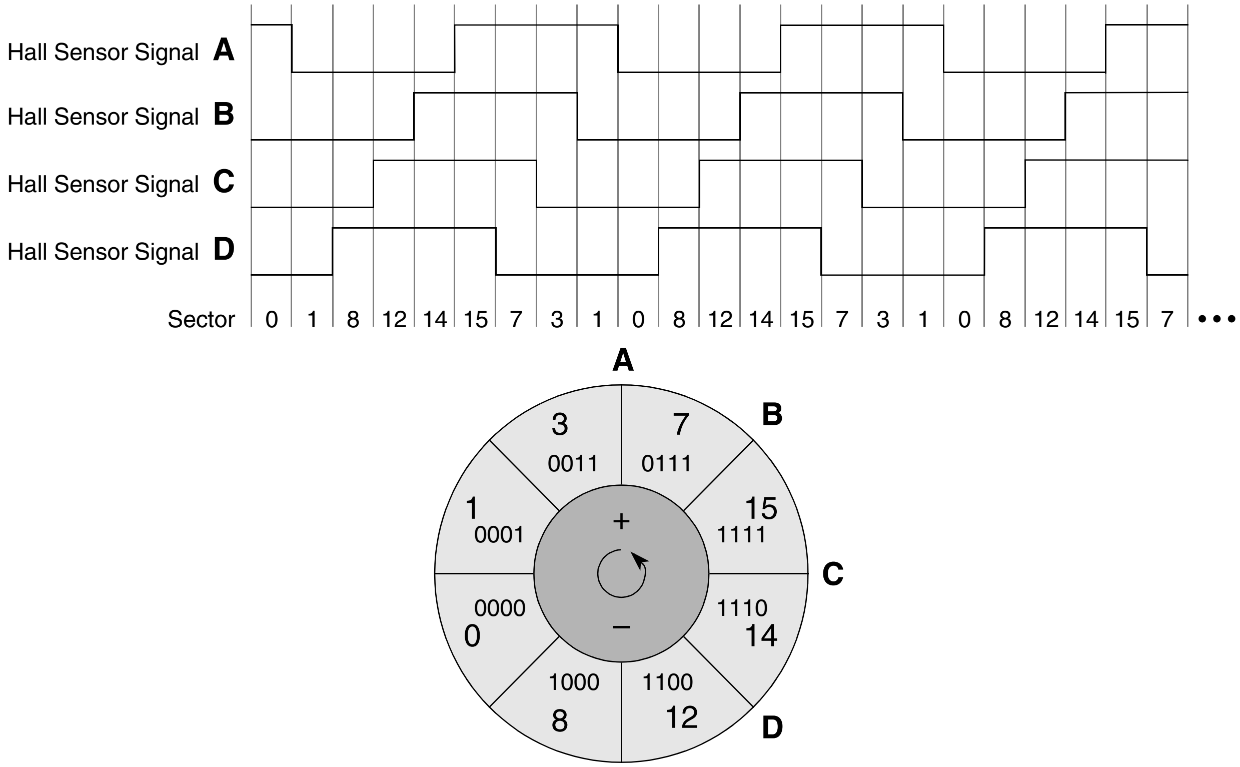

Figure 8. Pre-defined Configurations: Four Phases - by 45 Deg

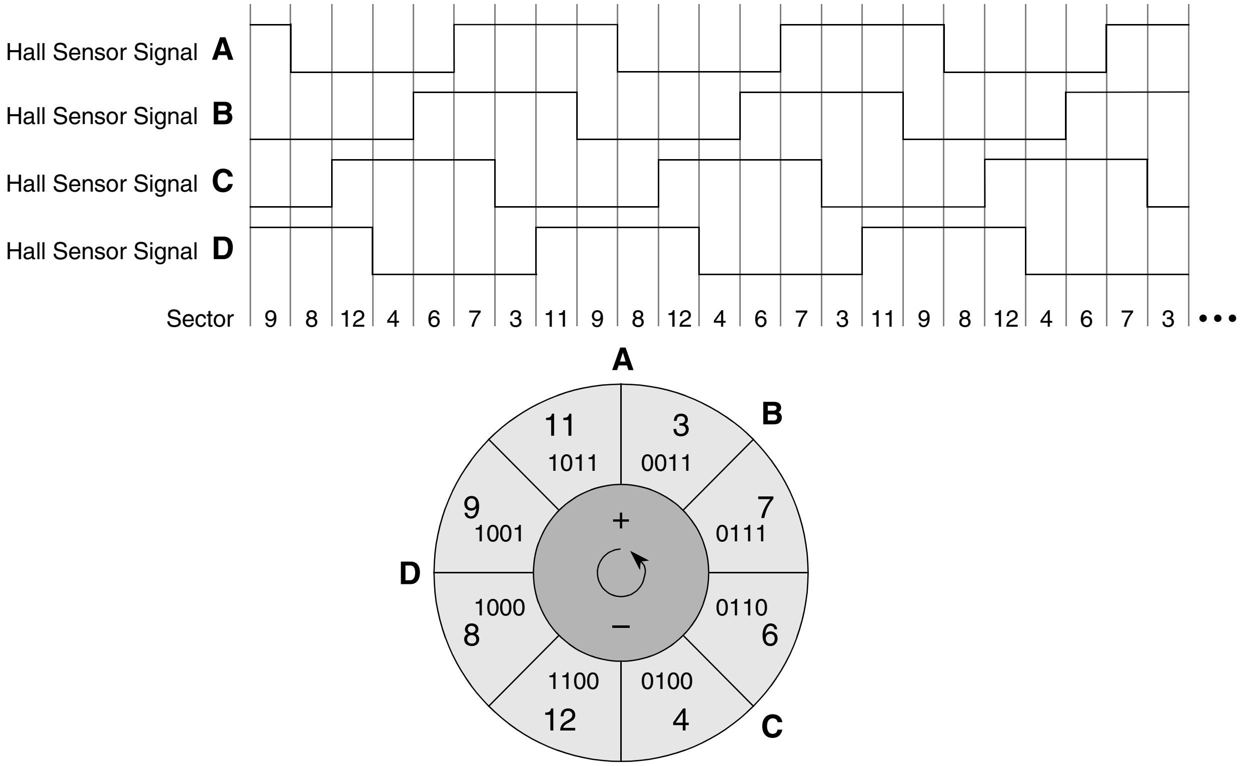

Figure 9. Pre-defined Configurations: Four Phases - by 90 and 45 Deg

3.2 Commutation

The HD function has optional support for commutation capability using the PWMC function. The user should create 1, 2, 3, or 4 commutation tables, according to the number of HD phases and pass their pointers to the HD initialization function. When an edge of any input Hall sensor signal is detected, a commutation command to one or more PWMC phases is sent, and then the sector value is changed. The commutation tables must be defined as structures of uint32_t. Each phase’s commutation table consists of eight commutation commands. Each table entry is one commutation command.

The order of table entries is as follows:

1) Commutation command to execute on LH transition, on incremental direction, as first (lh_i_0).

2) Commutation command to execute on LH transition, on incremental direction, as second (lh_i_1).

3) Commutation command to execute on LH transition, on decremental direction, as first (lh_d_0).

4) Commutation command to execute on LH transition, on decremental direction, as second (lh_d_1).

5) Commutation command to execute on HL transition, on incremental direction, as first (hl_i_0).

6) Commutation command to execute on HL transition, on incremental direction, as second (hl_i_1).

7) Commutation command to execute on HL transition, on decremental direction, as first (hl_d_0).

8) Commutation command to execute on HL transition, on decremental direction, as second (hl_d_1).

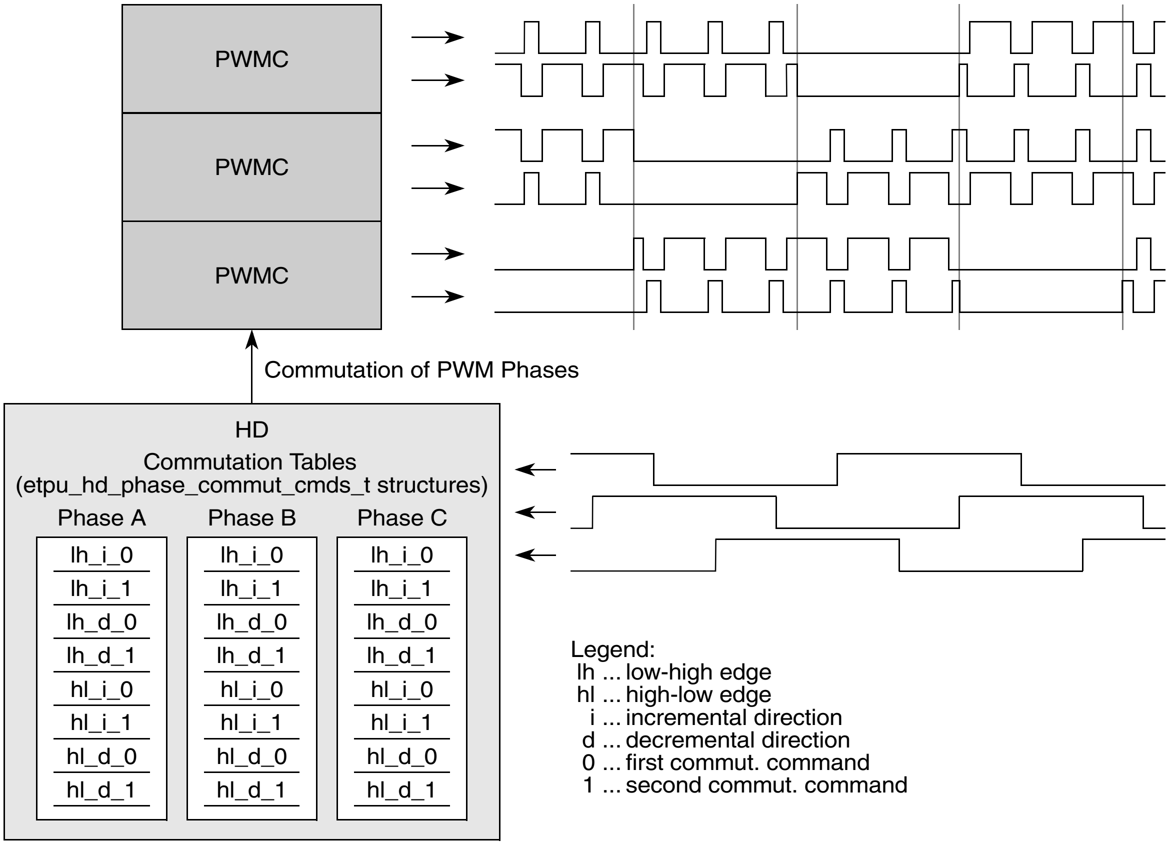
PWMC
PWMC
PWMC
Commutation of PWM Phases
HD
Commutation Tables
(etpu_hd_phase_commut_cmds_t structures)
Phase A
Phase B
Phase C
lh_i_0
lh_i_1
lh_d_0
lh_d_1
hl_i_0
hl_i_1
hl_d_0
hl_d_1
Legend:
lh ... low-high edge
hl ... high-low edge
i ... incremental direction
d ... decremental direction
0 ... first commut. command
1 ... second commut. command

Figure 10. Commutation

Example of Commutation Tables definition:

```
#include "HD_etpu_gct.h"        // defines channel numbers like PWMMDC0_PHASEA_BASE_CHANNEL,...
#include "etpu_hd.h"            // defines phase commutation commands structure type etpu_hd_phase_commut_cmds_t
#include "etpu_pwmmdc.h"        // defines commutation commands and phase options

etpu_hd_phase_commut_cmds_t phaseA_commut_cmds = {
(FS_ETPU_PWMMDC_DUTY_POS<<24)+(FS_ETPU_PWMMDC_OFF_LOW      <<16)+(FS_ETPU_PWMMDC_OFF_LOW       <<8)+PWMMDC0_PHASEB_BASE_CHANNEL,
(FS_ETPU_PWMMDC_DUTY_NEG<<24)+(FS_ETPU_PWMMDC_ON_ACTIVE_LOW<<16)+(FS_ETPU_PWMMDC_ON_ACTIVE_HIGH<<8)+PWMMDC0_PHASEA_BASE_CHANNEL,
(FS_ETPU_PWMMDC_DUTY_NEG<<24)+(FS_ETPU_PWMMDC_OFF_LOW      <<16)+(FS_ETPU_PWMMDC_OFF_LOW       <<8)+PWMMDC0_PHASEA_BASE_CHANNEL,
(FS_ETPU_PWMMDC_DUTY_POS<<24)+(FS_ETPU_PWMMDC_ON_ACTIVE_LOW<<16)+(FS_ETPU_PWMMDC_ON_ACTIVE_HIGH<<8)+PWMMDC0_PHASEB_BASE_CHANNEL,
(FS_ETPU_PWMMDC_DUTY_NEG<<24)+(FS_ETPU_PWMMDC_OFF_LOW      <<16)+(FS_ETPU_PWMMDC_OFF_LOW       <<8)+PWMMDC0_PHASEB_BASE_CHANNEL,
(FS_ETPU_PWMMDC_DUTY_POS<<24)+(FS_ETPU_PWMMDC_ON_ACTIVE_LOW<<16)+(FS_ETPU_PWMMDC_ON_ACTIVE_HIGH<<8)+PWMMDC0_PHASEA_BASE_CHANNEL,
(FS_ETPU_PWMMDC_DUTY_POS<<24)+(FS_ETPU_PWMMDC_OFF_LOW      <<16)+(FS_ETPU_PWMMDC_OFF_LOW       <<8)+PWMMDC0_PHASEA_BASE_CHANNEL,
(FS_ETPU_PWMMDC_DUTY_NEG<<24)+(FS_ETPU_PWMMDC_ON_ACTIVE_LOW<<16)+(FS_ETPU_PWMMDC_ON_ACTIVE_HIGH<<8)+PWMMDC0_PHASEB_BASE_CHANNEL
};
etpu_hd_phase_commut_cmds_t phaseB_commut_cmds = {
(FS_ETPU_PWMMDC_DUTY_POS<<24)+(FS_ETPU_PWMMDC_OFF_LOW      <<16)+(FS_ETPU_PWMMDC_OFF_LOW       <<8)+PWMMDC0_PHASEC_BASE_CHANNEL,
(FS_ETPU_PWMMDC_DUTY_NEG<<24)+(FS_ETPU_PWMMDC_ON_ACTIVE_LOW<<16)+(FS_ETPU_PWMMDC_ON_ACTIVE_HIGH<<8)+PWMMDC0_PHASEB_BASE_CHANNEL,
(FS_ETPU_PWMMDC_DUTY_NEG<<24)+(FS_ETPU_PWMMDC_OFF_LOW      <<16)+(FS_ETPU_PWMMDC_OFF_LOW       <<8)+PWMMDC0_PHASEB_BASE_CHANNEL,
(FS_ETPU_PWMMDC_DUTY_POS<<24)+(FS_ETPU_PWMMDC_ON_ACTIVE_LOW<<16)+(FS_ETPU_PWMMDC_ON_ACTIVE_HIGH<<8)+PWMMDC0_PHASEC_BASE_CHANNEL,
(FS_ETPU_PWMMDC_DUTY_NEG<<24)+(FS_ETPU_PWMMDC_OFF_LOW      <<16)+(FS_ETPU_PWMMDC_OFF_LOW       <<8)+PWMMDC0_PHASEC_BASE_CHANNEL,
(FS_ETPU_PWMMDC_DUTY_POS<<24)+(FS_ETPU_PWMMDC_ON_ACTIVE_LOW<<16)+(FS_ETPU_PWMMDC_ON_ACTIVE_HIGH<<8)+PWMMDC0_PHASEB_BASE_CHANNEL,
(FS_ETPU_PWMMDC_DUTY_POS<<24)+(FS_ETPU_PWMMDC_OFF_LOW      <<16)+(FS_ETPU_PWMMDC_OFF_LOW       <<8)+PWMMDC0_PHASEB_BASE_CHANNEL,
(FS_ETPU_PWMMDC_DUTY_NEG<<24)+(FS_ETPU_PWMMDC_ON_ACTIVE_LOW<<16)+(FS_ETPU_PWMMDC_ON_ACTIVE_HIGH<<8)+PWMMDC0_PHASEC_BASE_CHANNEL
};
etpu_hd_phase_commut_cmds_t phaseC_commut_cmds = {
(FS_ETPU_PWMMDC_DUTY_POS<<24)+(FS_ETPU_PWMMDC_OFF_LOW      <<16)+(FS_ETPU_PWMMDC_OFF_LOW       <<8)+PWMMDC0_PHASEA_BASE_CHANNEL,
(FS_ETPU_PWMMDC_DUTY_NEG<<24)+(FS_ETPU_PWMMDC_ON_ACTIVE_LOW<<16)+(FS_ETPU_PWMMDC_ON_ACTIVE_HIGH<<8)+PWMMDC0_PHASEC_BASE_CHANNEL,
(FS_ETPU_PWMMDC_DUTY_NEG<<24)+(FS_ETPU_PWMMDC_OFF_LOW      <<16)+(FS_ETPU_PWMMDC_OFF_LOW       <<8)+PWMMDC0_PHASEC_BASE_CHANNEL,
(FS_ETPU_PWMMDC_DUTY_POS<<24)+(FS_ETPU_PWMMDC_ON_ACTIVE_LOW<<16)+(FS_ETPU_PWMMDC_ON_ACTIVE_HIGH<<8)+PWMMDC0_PHASEA_BASE_CHANNEL,
(FS_ETPU_PWMMDC_DUTY_NEG<<24)+(FS_ETPU_PWMMDC_OFF_LOW      <<16)+(FS_ETPU_PWMMDC_OFF_LOW       <<8)+PWMMDC0_PHASEA_BASE_CHANNEL,
(FS_ETPU_PWMMDC_DUTY_POS<<24)+(FS_ETPU_PWMMDC_ON_ACTIVE_LOW<<16)+(FS_ETPU_PWMMDC_ON_ACTIVE_HIGH<<8)+PWMMDC0_PHASEC_BASE_CHANNEL,
(FS_ETPU_PWMMDC_DUTY_POS<<24)+(FS_ETPU_PWMMDC_OFF_LOW      <<16)+(FS_ETPU_PWMMDC_OFF_LOW       <<8)+PWMMDC0_PHASEC_BASE_CHANNEL,
(FS_ETPU_PWMMDC_DUTY_NEG<<24)+(FS_ETPU_PWMMDC_ON_ACTIVE_LOW<<16)+(FS_ETPU_PWMMDC_ON_ACTIVE_HIGH<<8)+PWMMDC0_PHASEA_BASE_CHANNEL
};
```

3.3 Interrupts

The HD function generates an interrupt service request to the CPU on each detected edge.

3.4 Performance

Like all eTPU functions, the HD 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 influence of the HD function on the overall eTPU performance can be expressed by the following parameter:

- **maximum eTPU busy-time per sector**

  This value determines the eTPU time necessary for processing one transition of a Hall sensor signal.

Table 1 lists the maximum eTPU busy-times per sector in eTPU cycles that depend on the mode of commutation processing. Where commutation processing is enabled, the value also covers the processing of two commutation commands performed by the PWMC function.

Table 1. Maximum eTPU Busy-Time per Sector

Commutation Processing Mode	Maximum eTPU Busy-time per Sector [eTPU cycles]
Commutation processing disabled	56
Commutation processing enabled	308

The eTPU module clock is equal to the CPU clock on MPC5500 devices, as well as the peripheral clock (half of the CPU clock) on MCF523x devices. For example, the eTPU module clock is 132 MHz on a 132-MHz MPC5554, and one eTPU cycle takes 7.58ns; it is only 75 MHz on a 150-MHz MCF5235, and one eTPU cycle takes 13.33ns.

The performance is influenced by compiler efficiency. The above numbers, measured on the code compiled by eTPU compiler version 1.0.0.5, are given for guidance only and are subject to change. For up to date information, refer to the information provided in the particular eTPU function set release available from Freescale.

4 C Level API for Function

The following routines provide the application developer easy access to the HD function. Use of these functions eliminates the need to directly control the eTPU registers. There are 17 functions added to the application programming interface (API). The routines can be found in the `etpu_hd.h` and `etpu_hd.c` files, which should be included in the link file along with the top level development file(s).

Figure 11 shows the HD API state flow and lists API functions which can be used in each of its states.

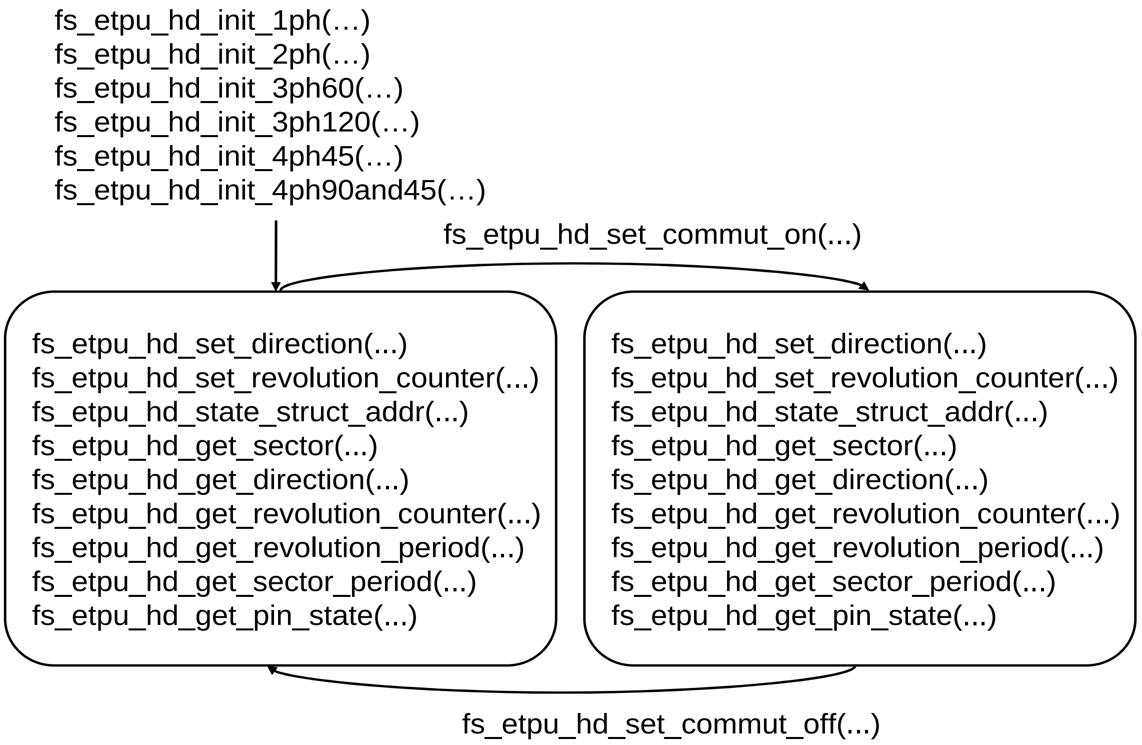
fs_etpu_hd_init_1ph(…)
fs_etpu_hd_init_2ph(…)
fs_etpu_hd_init_3ph60(…)
fs_etpu_hd_init_3ph120(…)
fs_etpu_hd_init_4ph45(…)
fs_etpu_hd_init_4ph90and45(…)
fs_etpu_hd_set_commut_on(...)
fs_etpu_hd_set_direction(...)
fs_etpu_hd_set_revolution_counter(...)
fs_etpu_hd_state_struct_addr(...)
fs_etpu_hd_get_sector(...)
fs_etpu_hd_get_direction(...)
fs_etpu_hd_get_revolution_counter(...)
fs_etpu_hd_get_revolution_period(...)
fs_etpu_hd_get_sector_period(...)
fs_etpu_hd_get_pin_state(...)
fs_etpu_hd_set_direction(...)
fs_etpu_hd_set_revolution_counter(...)
fs_etpu_hd_state_struct_addr(...)
fs_etpu_hd_get_sector(...)
fs_etpu_hd_get_direction(...)
fs_etpu_hd_get_revolution_counter(...)
fs_etpu_hd_get_revolution_period(...)
fs_etpu_hd_get_sector_period(...)
fs_etpu_hd_get_pin_state(...)
fs_etpu_hd_set_commut_off(...)

Figure 11. HD API State Flow

All HD API routines will be described in order and are listed below:

- Initialization Functions:

```
int32_t fs_etpu_hd_init_1ph( uint8_t channel_phaseA,
                             uint8_t priority,
                             uint8_t timer,
                             uint8_t direction,
                             uint8_t PWMMDC_chan,
               etpu_hd_phase_commut_cmds_t *phaseA_commut_cmds);

int32_t fs_etpu_hd_init_2ph( uint8_t channel_phaseA,
                             uint8_t channel_phaseB,
                             uint8_t priority,
                             uint8_t timer,
                             uint8_t direction,
                             uint8_t PWMMDC_chan,
               etpu_hd_phase_commut_cmds_t *phaseA_commut_cmds,
         etpu_hd_phase_commut_cmds_t *phaseB_commut_cmds);

int32_t fs_etpu_hd_init_3ph60( uint8_t channel_phaseA,
                               uint8_t channel_phaseB,
                               uint8_t channel_phaseC,
                               uint8_t priority,
                               uint8_t timer,
                               uint8_t revolution_on_off,
                               uint8_t direction_on_off,
                               uint8_t direction,
                               uint8_t PWMMDC_chan,
           etpu_hd_phase_commut_cmds_t *phaseA_commut_cmds,
           etpu_hd_phase_commut_cmds_t *phaseB_commut_cmds,
           etpu_hd_phase_commut_cmds_t *phaseC_commut_cmds);

int32_t fs_etpu_hd_init_3ph120( uint8_t channel_phaseA,
                                uint8_t channel_phaseB,
                                uint8_t channel_phaseC,
                                uint8_t priority,
                                uint8_t timer,
                                uint8_t revolution_on_off,
                                uint8_t direction_on_off,
                                uint8_t direction,
                                uint8_t PWMMDC_chan,
            etpu_hd_phase_commut_cmds_t *phaseA_commut_cmds,
            etpu_hd_phase_commut_cmds_t *phaseB_commut_cmds,
            etpu_hd_phase_commut_cmds_t *phaseC_commut_cmds);

int32_t fs_etpu_hd_init_4ph45( uint8_t channel_phaseA,
                               uint8_t channel_phaseB,
                               uint8_t channel_phaseC,
                               uint8_t channel_phaseD,
                               uint8_t priority,
                               uint8_t timer,
                               uint8_t revolution_on_off,
                               uint8_t direction_on_off,
                               uint8_t direction,
                               uint8_t PWMMDC_chan,
           etpu_hd_phase_commut_cmds_t *phaseA_commut_cmds,
           etpu_hd_phase_commut_cmds_t *phaseB_commut_cmds,
           etpu_hd_phase_commut_cmds_t *phaseC_commut_cmds,
           etpu_hd_phase_commut_cmds_t *phaseD_commut_cmds);
```

```
int32_t fs_etpu_hd_init_4ph90and45( uint8_t channel_phaseA,
                                    uint8_t channel_phaseB,
                                    uint8_t channel_phaseC,
                                    uint8_t channel_phaseD,
                                    uint8_t priority,
                                    uint8_t timer,
                                    uint8_t revolution_on_off,
                                    uint8_t direction_on_off,
                                    uint8_t direction,
                                    uint8_t PWMMDC_chan,
                etpu_hd_phase_commut_cmds_t *phaseA_commut_cmds,
                etpu_hd_phase_commut_cmds_t *phaseB_commut_cmds,
                etpu_hd_phase_commut_cmds_t *phaseC_commut_cmds,
                etpu_hd_phase_commut_cmds_t *phaseD_commut_cmds);
```

- Change Operation Functions:

```
int32_t fs_etpu_hd_set_commut_on ( uint8_t channel)
int32_t fs_etpu_hd_set_commut_off( uint8_t channel)
int32_t fs_etpu_hd_set_direction( uint8_t channel, uint8_t direction)
int32_t fs_etpu_hd_set_revolution_counter(uint8_t channel, int24_t value)
```

- Value Return Functions:

```
uint32_t fs_etpu_hd_state_struct_addr( uint8_t channel)
uint8_t fs_etpu_hd_get_sector( uint8_t channel)
uint8_t fs_etpu_hd_get_direction( uint8_t channel)
int24_t fs_etpu_hd_get_revolution_counter( uint8_t channel)
uint24_t fs_etpu_hd_get_revolution_period( uint8_t channel)
uint24_t fs_etpu_hd_get_sector_period( uint8_t channel)
uint8_t fs_etpu_hd_get_pin_state( uint8_t channel)
```

4.1 Initialization Function

4.1.1 int32_t fs_etpu_hd_init_1ph(...), int32_t fs_etpu_hd_init_2ph(...), int32_t fs_etpu_hd_init_3ph60(...), int32_t fs_etpu_hd_init_3ph120(...), int32_t fs_etpu_hd_init_4ph45(...), int32_t fs_etpu_hd_init_4ph90and45(...)

These routines are used to initialize the eTPU channel(s) for the HD function(s). They differ in the number of phases they initialize (1, 2, 3, or 4 phases) and the configuration of the Hall sensors (by 60deg, by 120deg, by 45deg, or by 90 and 45 deg).

Because the revolution counter and direction are calculated only in configurations with 3 or 4 input channels (phases), both the `revolution_on_off` and `direction_on_off` parameters are not used in `fs_etpu_hd_init_1ph(...)` and `fs_etpu_hd_init_2ph(...)` functions dedicated for one and two phases; see Section 3, “Function Description.”

The functions have the following parameters:

- **channel_phaseA (uint8_t)** - This is the phase A channel number. This parameter should be assigned a value of 0-31 for ETPU_A, and 64-95 for ETPU_B.
- **channel_phaseB (uint8_t)** - This is the phase B channel number. This parameter should be assigned a value of 0-31 for ETPU_A, and 64-95 for ETPU_B.
- **channel_phaseC (uint8_t)** - This is the phase C channel number. This parameter should be assigned a value of 0-31 for ETPU_A, and 64-95 for ETPU_B.
- **channel_phaseD (uint8_t)** - This is the phase D channel number. This parameter should be assigned a value of 0-31 for ETPU_A, and 64-95 for ETPU_B.
- **priority (uint8_t)** - This is the priority to assign to the all HD channels. This parameter should be assigned a value of:
  - FS_ETPU_PRIORITY_HIGH,
  - FS_ETPU_PRIORITY_MIDDLE OR
  - FS_ETPU_PRIORITY_LOW.
- **timer (uint8_t)** - This is the timer used for period measurements. This parameter should be assigned a value of:
  - FS_ETPU_HD_TCR1 or
  - FS_ETPU_HD_TCR2.
- **revolution_on_off (uint8_t)** - This parameter enables/disables counting of motor revolutions. This parameter should be assigned a value of:
  - FS_ETPU_HD_REV_COUNTING_ON or
  - FS_ETPU_HD_REV_COUNTING_OFF.
- **direction_on_off (uint8_t)** - This parameter enables/disables automatic determination of motor direction. This parameter should be assigned a value of:
  - FS_ETPU_HD_DIRECTION_AUTO_ON or
  - FS_ETPU_HD_DIRECTION_AUTO_OFF.
- **direction (uint8_t)** - This is the initial direction value. This parameter should be assigned a value of:
  - FS_ETPU_HD_DIRECTION_INC or
  - FS_ETPU_HD_DIRECTION_DEC.

- **PWMMDC_chan (uint8_t)** - This is the number of PWMMDC channel. After commutation the PWMMDC channel is notified to update the PWM phases. 0-31 for ETPU_A and 64-96 for ETPU_B
- ***phaseA_commut_cmds (etpu_hd_phase_commut_cmds_t)** - Pointer to the commutation table for phase A; `etpu_hd_phase_commut_cmds_t` structure type is defined in *etpu_hd.h* file. It is defined as a structure of eight `uint32_t` commutation commands.
- ***phaseB_commut_cmds (etpu_hd_phase_commut_cmds_t)** - Pointer to the commutation table for phase B; `etpu_hd_phase_commut_cmds_t` structure type is defined in *etpu_hd.h* file. It is defined as a structure of eight `uint32_t` commutation commands.
- ***phaseC_commut_cmds (etpu_hd_phase_commut_cmds_t)** - Pointer to the commutation table for phase C; `etpu_hd_phase_commut_cmds_t` structure type is defined in *etpu_hd.h* file. It is defined as a structure of eight `uint32_t` commutation commands.
- ***phaseD_commut_cmds (etpu_hd_phase_commut_cmds_t)** - Pointer to the commutation table for phase D; `etpu_hd_phase_commut_cmds_t` structure type is defined in *etpu_hd.h* file. It is defined as a structure of eight `uint32_t` commutation commands.

4.2 Change Operation Functions

4.2.1 int32_t fs_etpu_hd_set_commut_on (uint8_t channel)

This function enables commutation processing on one HD channel. It has the following parameter:

- **channel (uint8_t)** - This is the phase A channel number. This parameter must be assigned the same value as was assigned to the *channel_phaseA* parameter in the initialization routine. If there are more HDs running simultaneously on the eTPU(s), the channel parameter distinguishes which HD function is accessed.

4.2.2 int32_t fs_etpu_hd_set_commut_off (uint8_t channel)

This function disables commutation processing on one HD channel. It has the following parameter:

- **channel (uint8_t)** - This is the phase A channel number. This parameter must be assigned the same value as was assigned to the *channel_phaseA* parameter in the initialization routine. If there are more HDs running simultaneously on the eTPU(s), the channel parameter distinguishes which HD function is accessed.

4.2.3 int32_t fs_etpu_hd_set_direction(uint8_t channel, uint8_t direction)

This function sets the direction value. Use this function when the automatic direction detection is off (`direction_on_off=FS_ETPU_HD_DIRECTION_AUTO_OFF`), in order to select which commutation command is used. This function has the following parameters:

- **channel (uint8_t)** - This is the phase A channel number. This parameter must be assigned the same value as was assigned to the *channel_phaseA* parameter in the initialization routine. If there are more HDs running simultaneously on the eTPU(s), the channel parameter distinguishes which HD function is accessed.
- **direction (uint8_t)** - This parameter should be assigned a value of:
  - — FS_ETPU_HD_DIRECTION_INC or
  - — FS_ETPU_HD_DIRECTION_DEC

4.2.4 int32_t fs_etpu_hd_set_revolution_counter(uint8_t channel, int24_t value)

This function changes the revolution counter value. This function has the following parameters:

- **channel (uint8_t)** - This is the phase A channel number. This parameter must be assigned the same value as was assigned to the *channel_phaseA* parameter in the initialization routine. If there are more HDs running simultaneously on the eTPU(s), the channel parameter distinguishes which HD function is accessed.
- **value (int24_t)** - This is the revolution counter value to be set.

4.3 Value Return Function

4.3.1 uint32_t fs_etpu_hd_state_struct_addr(uint8_t channel)

This function returns address of the state structure beginning. This function has the following parameter:

- **channel (uint8_t)** - This is the phase A channel number. This parameter must be assigned the same value as was assigned to the *channel_phaseA* parameter in the initialization routine. If there are more HDs running simultaneously on the eTPU(s), the channel parameter distinguishes which HD function is accessed.

4.3.2 uint8_t fs_etpu_hd_get_sector(uint8_t channel)

This function returns the current sector value. This function has the following parameter:

- **channel (uint8_t)** - This is the phase A channel number. This parameter must be assigned the same value as was assigned to the *channel_phaseA* parameter in the initialization routine. If there are more HDs running simultaneously on the eTPU(s), the channel parameter distinguishes which HD function is accessed.

The sector value is returned as an `uint8_t`.

4.3.3 uint8_t fs_etpu_hd_get_direction(uint8_t channel)

This function returns the motion system direction. This function has the following parameter:

- **channel (uint8_t)** - This is the phase A channel number. This parameter must be assigned the same value as was assigned to the *channel_phaseA* parameter in the initialization routine. If there are more HDs running simultaneously on the eTPU(s), the channel parameter distinguishes which HD function is accessed.

The returned value can be one of:

— FS_ETPU_HD_DIRECTION_INC or

— FS_ETPU_HD_DIRECTION_DEC

4.3.4 int24_t fs_etpu_hd_get_revolution_counter(uint8_t channel)

This function returns the revolution counter value. This function has the following parameter:

- **channel (uint8_t)** - This is the phase A channel number. This parameter must be assigned the same value as was assigned to the *channel_phaseA* parameter in the initialization routine. If there are more HDs running simultaneously on the eTPU(s), the channel parameter distinguishes which HD function is accessed.

The revolution counter value is returned as an `int24_t`.

4.3.5 uint24_t fs_etpu_hd_get_revolution_period(uint8_t channel)

This function returns the revolution period value. This function has the following parameter:

- **channel (uint8_t)** - This is the phase A channel number. This parameter must be assigned the same value as was assigned to the *channel_phaseA* parameter in the initialization routine. If there are more HDs running simultaneously on the eTPU(s), the channel parameter distinguishes which HD function is accessed.

The revolution period value is returned as an `uint24_t`.

4.3.6 uint24_t fs_etpu_hd_get_sector_period(uint8_t channel)

This function returns the sector period value. This function has the following parameter:

- **channel (uint8_t)** - This is the phase A channel number. This parameter must be assigned the same value as was assigned to the *channel_phaseA* parameter in the initialization routine. If there are more HDs running simultaneously on the eTPU(s), the channel parameter distinguishes which HD function is accessed.

The sector period value is returned as an `uint24_t`.

4.3.7 uint8_t fs_etpu_hd_get_pin_state(uint8_t channel)

This function returns selected HD channel pin state. This function has the following parameter:

- **channel (uint8_t)** - This is the phase A channel number. This parameter must be assigned the same value as was assigned to the *channel_phaseA* parameter in the initialization routine. If there are more HDs running simultaneously on the eTPU(s), the channel parameter distinguishes which HD function is accessed.

The HD channel pin state value is returned as an `uint8_t`.

4.4 TPU-compatible C Level API for Function

The TPU-compatible API provides backward-compatibility from eTPU to TPU. The following functions allow control of the eTPU HD function using the TPU HALLD API function calls.

- Initialization Function:

```
void tpu_halld_init(struct TPU3_tag *tpu,
                    UINT8 channel,
                    INT16 direction,
                    INT16 state_no_address,
                    UINT8 mode)
```

- Change Operation Functions:

```
void tpu_halld_enable(struct TPU3_tag *tpu,
                      UINT8 channel,
                      UINT8 priority)
void tpu_halld_disable(struct TPU3_tag *tpu,
                       UINT8 channel)
void tpu_halld_set_direction(struct TPU3_tag *tpu,
                             UINT8 channel,
                             INT16 direction)
```

- Value Return Functions:

```
INT16 tpu_halld_get_state_no(struct TPU3_tag *tpu,
                             UINT8 channel)
INT16 tpu_halld_get_pin_state(struct TPU3_tag *tpu,
                             UINT8 channel)
```

NOTE

tpu_halld_set_state_no_address(...) function is not implemented!

5 Example Use of Function

5.1 Demo Applications

The usage of the HD eTPU function is demonstrated in the following applications:

- "3-Phase BLDC Motor with Speed Closed Loop, driven by eTPU on MCF523x," AN2892.
- "Three 3-Phase BLDC Motors with Speed Closed Loop, driven by eTPU on MCF523x," AN2948.
- "DC Motor with Speed and Current Closed Loop, driven by eTPU on MCF523x," AN2955.

For a detailed description of the demo applications, refer to the mentioned application notes.

The HD function is initialized in configuration with three phases and HD sensors by 120 degrees. Direction and revolution counting is enabled.

5.1.1 Function Calls

```
/* initialize the HD function */
err_code = fs_etpu_hd_init_3ph120 (
  HD0_PHASEA,/* engine: A; channel: 1 */
  HD0_PHASEB,/* engine: A; channel: 2 */
  HD0_PHASEC,/* engine: A; channel: 3 */
  FS_ETPU_PRIORITY_HIGH,/* priority: High */
  FS_ETPU_HD_TCR2,/* timer: TCR2 */
  FS_ETPU_HD_REV_COUNTING_ON,/* revolution_on_off: revolutions are counted */
  FS_ETPU_HD_DIRECTION_AUTO_ON,/* direction_on_off: direction is calculated */
  FS_ETPU_HD_DIRECTION_INC,/* direction: initial direction INC */
  PWMMDC0_MASTER,/* PWMMDC_chan: PWMMDC0_MASTER */
  &phaseA_commut_cmds,/* phaseA_commut_cmds: &phaseA_commut_cmds */
  &phaseB_commut_cmds,/* phaseB_commut_cmds: &phaseB_commut_cmds */
  &phaseC_commut_cmds);/* phaseC_commut_cmds: &phaseC_commut_cmds */

/* enable commutations by HD signals */
fs_etpu_hd_set_commut_on(HD0_PHASEA);
fs_etpu_hd_set_commut_on(HD0_PHASEB);
fs_etpu_hd_set_commut_on(HD0_PHASEC);
```

6 Summary and Conclusions

This application note provides the user with a description of the Hall decoder (HD) eTPU function usage and examples. The simple C interface routines to the HD eTPU function enable easy implementation of the HD in applications. The demo application is targeted at the MCF523*x* family of devices, but it could be easily reused with any device that has an eTPU.

7 References

1. “The Essential of Enhanced Time Processing Unit,” AN2353.
2. “General C Functions for the eTPU,” AN2864.
3. “Using the DC Motor Control eTPU Function Set (set3),” AN2958.
4. Enhanced Time Processing Unit Reference Manual, ETPURM/D.
5. eTPU Graphical Configuration Tool, http://www.freescale.com/etpu, ETPUGCT.
6. “3-Phase BLDC Motors with Speed Closed Loop, driven by eTPU on MCF523x,” AN2892.
7. "Three 3-Phase BLDC Motors with Speed Closed Loop, driven by eTPU on MCF523x," AN2948.
8. "DC Motor with Speed and Current Closed Loop, driven by eTPU on MCF523x," AN2955.

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
support@freescale.com

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)
support@freescale.com

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

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.

Document Number: AN2841
Rev. 1
04/2005