Freescale Semiconductor
Application Note

AN2970
Rev. 2, 02/2012

Using the Analog Sensing for AC Motors (ASAC) eTPU Function

Covers the MCF523*x*, MPC5500, MPC5600 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 analog sensing for AC motors (ASAC) enhanced time processor unit (eTPU) function is one of the functions included in the AC motor control eTPU function set (set4). This application note provides simple C interface routines to the ASAC eTPU function. The routines are targeted at the MCF523*x*, MPC5500 and MPC5600 families of devices, but can easily be used with any device that has an eTPU.

2 Function Overview

The ASAC function is useful for preprocessing analog values that are measured by an AD converter and transferred to the eTPU data memory by DMA transfer. The ASAC function is also useful for triggering the AD converter and synchronizing other eTPU functions.

Table of Contents

3 Function Description

ASAC function performs these operations:

- **Gets values from an AD converter result queue.**

  Up to four values from the queue can be processed. The converted values are read from the queue as 16-bit words at specified queue address offsets.
- **Performs bit alignment.**

  Bit alignment is performed by shifting the value from the result queue left by 8, 10, 12 or 16 bits; this creates a 24-bit fractional value.
- **Removes DC offset.**

  DC offsets are removed from the measured samples. The DC offsets can be set manually or measured.
- **Filters the measured values.**

  The measured values can be filtered using an exponentially-weighted moving average (EWMA) filter. The EWMA filter is defined by the following equation:

  $$y(n) = forget\_factor \cdot y(n - 1) + (1 - forget\_factor) \cdot x(n)$$

  where

  $x(n)$ is the n-th step filter input,

  $y(n)$ is the n-th step filter output, and

  $forget\_factor$ is the only filter parameter, forgetting factor. It is a value between 0 and 1, usually close to 1.
- **Processes phase currents of a 3-phase motor.**
  - If two LEMs are used to measure two phase currents, the ASAC function can calculate the third phase current. The calculation presumes that the sum of the three phase current values is 0.
  - If shunt resistors are used to measure the phase currents, based on the actual rotor position in one of the six sectors, the ASAC function can use only two of the three measured currents and calculate the third current. The sector value can be provided by the PWMMAC eTPU function.
  - Phase currents can be negated after DC offset removal. Positive values become negative and vice versa.

- **optionally, drives dead-time compensation.**

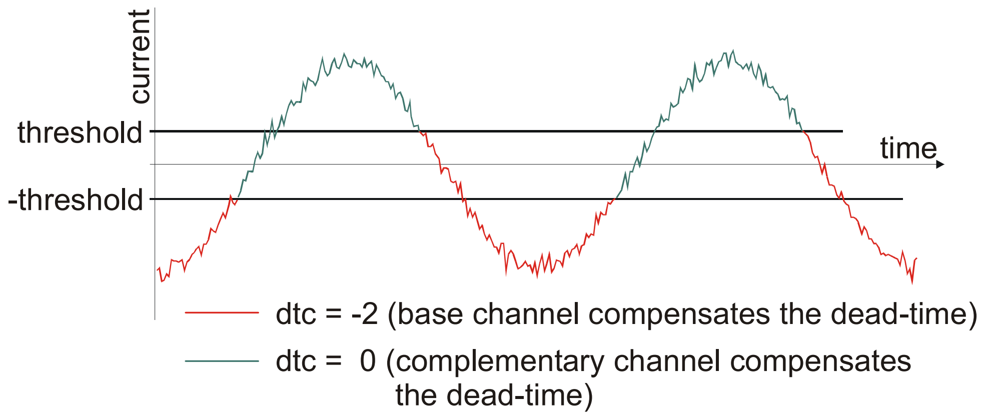

Figure 1. Phase Current and Dead-Time Compensation Parameter (DTC) Value

The three phase currents can be compared with threshold values, resulting in dead-time compensation parameters (DTC) that are supplied to the corresponding PWM phases. Figure 1 illustrates the dead-time compensation technique for one phase.

- **Triggers the AD converter by the generated signal.**

  The generated signal edges can be used to trigger the AD converter. The first edge is at the beginning of the ASAC period and the second edge is in the middle of the period.

  The generated signal polarity is selectable. The position of the triggering edge relative to the PWM period edge times is adjustable.

  On MPC5500 and MPC5600, the ASAC function can be assigned to one of five eTPU channels (channels 26 to 31) to activate one of five enhanced queued analog-to-digital converter (eQADC) triggers internally.

- **Generates eQADC conversion commands queue.**

  To ensure the correct order of phase current sampling on the MPC5500, it is necessary to supply the eQADC module by adequate conversion commands. The ASAC generates the applied queue of the eQADC conversion commands based on actual sector value, provided by the PWMMAC eTPU function and the defined eQADC conversion commands table.

- **Generates DMA request.**

  A DMA request is generated either on the first second edge of the generated signal. DMA transfers are useful for AD converter triggering on MCF523x devices (first edge), or for transferring the eQADC conversion commands from eTPU DATA RAM to eQADC on MPC5500 devices (second edge).

- **Synchronizes processing of other eTPU functions.**

  The ASAC function can execute processing of two other eTPU functions via an eTPU link. The link to one of the functions (inner-loop controller) is executed just after the ASAC process of the measured values (on the second edge). The link to the second of the functions (outer-loop controller) is executed on the first edge of the generated pulse, and only once per a defined number of periods.

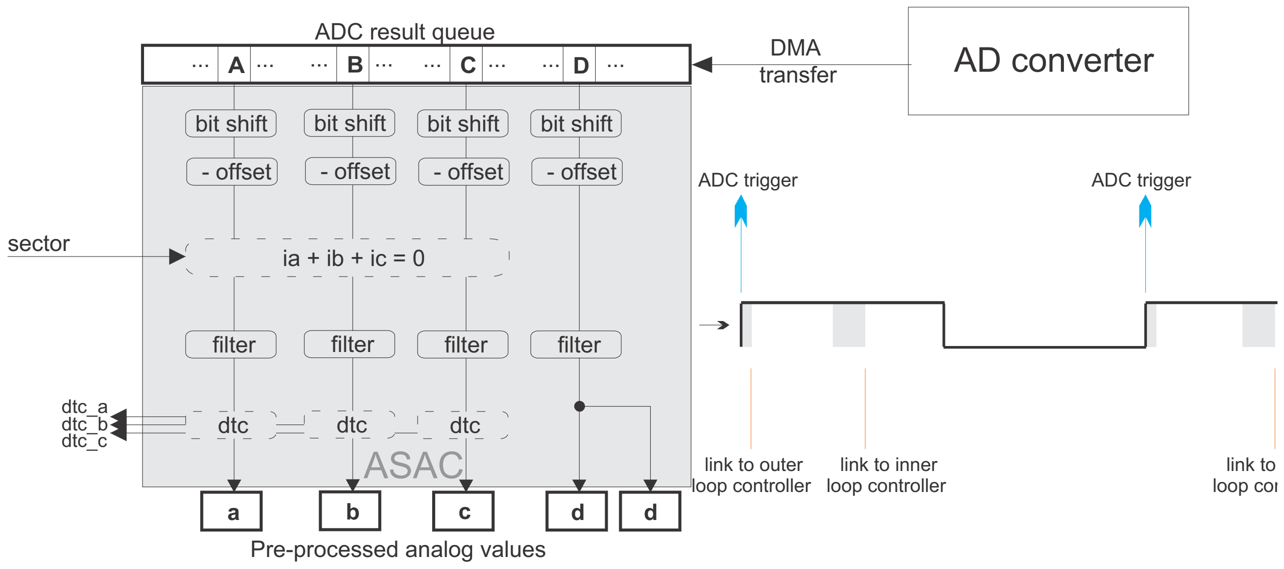

Figure 2. ASAC Processing Overview

3.1 Modes of Operation

The ASAC function can operate in one of these modes:

- **Periodic mode**

  In this mode, the ASAC function generates the pulses that trigger the AD converter periodically in a defined period.

- **Synchronized mode**

  This mode is useful when the AD triggering and other ASAC function processing must be synchronized with the PWM signals generated by motor-control PWM eTPU functions (PWMMAC, PWMF). Even when the PWM periods are changed during the run, the ASAC function generates the pulses that trigger the AD converter synchronously with the PWM period. The first edge of the pulse is generated, in an adjustable time, before or after the PWM edge-time.

3.2 Interrupts

The ASAC function periodically generates an interrupt service request to the CPU either on the first or second edge of the generated signal.

4 C Level API for Function

The following routines provide easy access for an application developer to the ASAC function. Use of these functions eliminates the need to directly control the eTPU registers. There are 20 functions added to the application programming interface (API). The routines can be found in the `etpu_asac.h` and `etpu_asac.c` files, which should be included in the link file along with the top level development file(s).

Figure 3 shows the ASAC API state flow and lists API functions that can be used in each of its states.

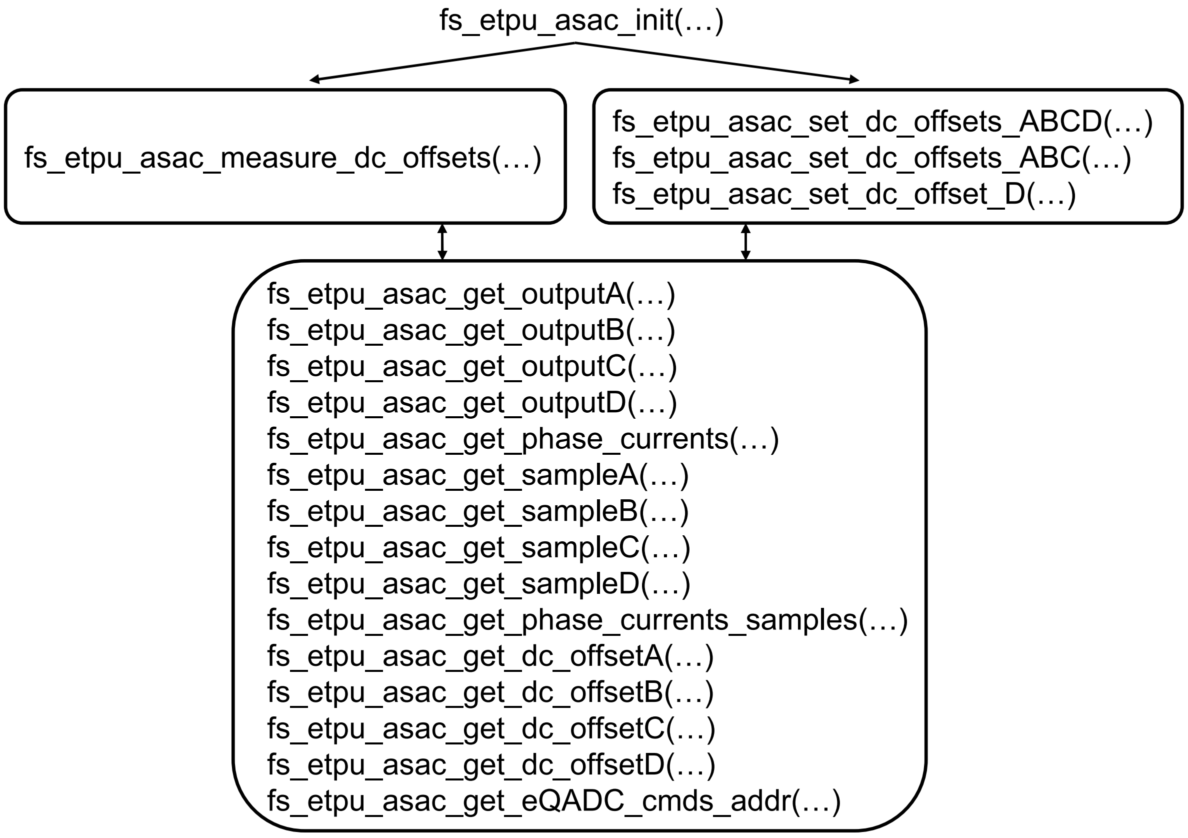
fs_etpu_asac_init(…)
fs_etpu_asac_measure_dc_offsets(…)
fs_etpu_asac_set_dc_offsets_ABCD(…)
fs_etpu_asac_set_dc_offsets_ABC(…)
fs_etpu_asac_set_dc_offset_D(…)
fs_etpu_asac_get_outputA(…)
fs_etpu_asac_get_outputB(…)
fs_etpu_asac_get_outputC(…)
fs_etpu_asac_get_outputD(…)
fs_etpu_asac_get_phase_currents(…)
fs_etpu_asac_get_sampleA(…)
fs_etpu_asac_get_sampleB(…)
fs_etpu_asac_get_sampleC(…)
fs_etpu_asac_get_sampleD(…)
fs_etpu_asac_get_phase_currents_samples(…)
fs_etpu_asac_get_dc_offsetA(…)
fs_etpu_asac_get_dc_offsetB(…)
fs_etpu_asac_get_dc_offsetC(…)
fs_etpu_asac_get_dc_offsetD(…)
fs_etpu_asac_get_eQADC_cmds_addr(…)

Figure 3. ASAC API State Flow

All ASAC API routines are described in order and listed below:

- Initialization Function:

```
int32_t fs_etpu_asac_init( uint8_t    channel,
                           uint8_t    priority,
                           uint8_t    polarity,
                           uint8_t    mode,
                           uint8_t    measure_samples_mask,
                           uint8_t    DTC_option,
                           uint8_t    phase_current_option,
                           uint8_t    negate_phase_currents,
                           uint8_t    cfifo_update,
                           uint8_t    DMA_interrupt_option,
                           uint24_t   period,
                           uint24_t   start_offset,
                           int24_t     egde_offset,
                           uint8_t     PWMMAC_chan,
                           uint24_t    measure_time,
                           uint8_t     periods_per_outerloop,
                           uint8_t     SC_BC_outerloop_chan,
                           uint8_t     PMSMVC_ACIMVC_innerloop_chan,
                           uint32_t *  result_queue,
                           uint8_t     queue_offset_a,
                           uint8_t     queue_offset_b,
                           uint8_t     queue_offset_c,
                           uint8_t     queue_offset_d,
                           fract24_t   forget_factor_a,
                           fract24_t   forget_factor_b,
                           fract24_t   forget_factor_c,
                           fract24_t   forget_factor_d,
                           uint8_t     bit_shift,
                           fract24_t   dtc_threshold,
                           uint8_t     outputA_chan,
                           uint16_t    outputA_offset,
                           uint8_t     outputB_chan,
                           uint16_t    outputB_offset,
                           uint8_t     outputC_chan,
                           uint16_t    outputC_offset,
                           uint8_t     outputD1_chan,
                           uint16_t    outputD1_offset,
                           uint8_t     outputD2_chan,
                           uint16_t    outputD2_offset,
                           uint8_t     eQADC_cmds_number,
             etpu_asac_eQADC_cmds_t * eQADC_cmds_current_measurements,
                           uint32_t *  eQADC_cmds_other_measurements)
```

- Change Operation Functions:

```
int32_t fs_etpu_asac_measure_dc_offsets(uint8_t channel,
                                        int8_t  measure_dc_offsets_mask)
int32_t fs_etpu_asac_set_dc_offsets_ABCD(uint8_t    channel,
                                         ufract24_t dc_offset_a,
                                         ufract24_t dc_offset_b,
                                         ufract24_t dc_offset_c,
                                         ufract24_t dc_offset_d)
int32_t fs_etpu_asac_set_dc_offsets_ABC(uint8_t    channel,
                                        ufract24_t dc_offset_a,
                                        ufract24_t dc_offset_b,
                                        ufract24_t dc_offset_c)
int32_t fs_etpu_asac_set_dc_offset_D(uint8_t    channel,
                                     ufract24_t dc_offset_d)
```

- Value Return Functions

```
fract24_t fs_etpu_asac_get_outputA(uint8_t channel)
fract24_t fs_etpu_asac_get_outputB(uint8_t channel)
fract24_t fs_etpu_asac_get_outputC(uint8_t channel)
fract24_t fs_etpu_asac_get_outputD(uint8_t channel)
int32_t fs_etpu_asac_get_phase_currents(uint8_t channel,
                                        asac_abc_t * p_i_abc)
fract24_t fs_etpu_asac_get_sampleA(uint8_t channel)
fract24_t fs_etpu_asac_get_sampleB(uint8_t channel)
fract24_t fs_etpu_asac_get_sampleC(uint8_t channel)
fract24_t fs_etpu_asac_get_sampleD(uint8_t channel)
int32_t fs_etpu_asac_get_phase_currents_samples(uint8_t channel,
                                                asac_abc_t * p_i_abc)
ufract24_t fs_etpu_asac_get_dc_offsetA(uint8_t channel)
ufract24_t fs_etpu_asac_get_dc_offsetB(uint8_t channel)
ufract24_t fs_etpu_asac_get_dc_offsetC(uint8_t channel)
ufract24_t fs_etpu_asac_get_dc_offsetD(uint8_t channel)
uint32_t fs_etpu_asac_get_eQADC_cmds_addr( uint8_t channel)
```

4.1 Initialization Function

4.1.1 int32_t fs_etpu_asac_init(...)

This routine is used to initialize the eTPU channel for the ASAC function. This function has these parameters:

- **channel (uint8_t)**—The ASAC channel number; should be assigned a value of 0-31 for ETPU_A, and 64-95 for ETPU_B.

- **priority (uint8_t)**—The priority to assign to the ASAC function; should be assigned one of these values:
  - — FS_ETPU_PRIORITY_HIGH
  - — FS_ETPU_PRIORITY_MIDDLE
  - — FS_ETPU_PRIORITY_LOW
  - — FS_ETPU_PRIORITY_DISABLED
- **polarity (uint8_t)**—The generated pulse polarity; should be assigned one of these values:
  - — FS_ETPU_ASAC_PULSE_HIGH
  - — FS_ETPU_ASAC_PULSE_LOW
- **mode (uint8_t)**—Mode configuration parameter; should be assigned one of these values:
  - — FS_ETPU_ASAC_MODE_PERIODIC
  - — FS_ETPU_ASAC_MODE_SYNC
- **measure_samples_mask (uint8_t)**—Defines which measured samples are processed by ASAC function; should be assigned one of these:
  - — FS_ETPU_ASAC_SAMPLE_A_B_C (phase currents)
  - — FS_ETPU_ASAC_SAMPLE_D (DC_BUS voltage)
  - — FS_ETPU_ASAC_SAMPLE_A_B_C_D
- **DTC_option (uint8_t)**—Turns on/off the dead-time compensation; should be assigned one of these values:
  - — FS_ETPU_ASAC_DTC_OFF
  - — FS_ETPU_ASAC_DTC_ON
- **phase_current_option (uint8_t)**—Turns on/off the calculation of the third phase current from two measured phase currents; should be assigned one of these values:
  - — FS_ETPU_ASAC_PHASE_CURR_USE_ALL
  - — FS_ETPU_ASAC_PHASE_CURR_USE_AB_CALC_C
  - — FS_ETPU_ASAC_PHASE_CURR_USE_BC_CALC_A
  - — FS_ETPU_ASAC_PHASE_CURR_USE_CA_CALC_B
  - — FS_ETPU_ASAC_PHASE_CURR_USE_SECTOR
- **negate_phase_currents (uint8_t)**—Turns on/off the negation of measured phase currents; should be assigned one of these values:
  - — FS_ETPU_ASAC_PHASE_CURRENTS_POS
  - — FS_ETPU_ASAC_PHASE_CURRENTS_NEG
- **cfifo_update (uint8_t)**—Enables the ASAC eTPU function to modify order of ADC commands according to actual PWMAC sector values; should be assigned one of these values:
  - — FS_ETPU_ASAC_CFIFO_UPDATE_ON
  - — FS_ETPU_ASAC_CFIFO_UPDATE_OFF

- **DMA_interrupt_option (uint8_t)**—Defines whether the DMA request and the interrupt to CPU are generated on the first ASAC edge (DMA_interrupt_option=FS_ETPU_ASAC_DMA_INTR_ON_FIRST_EDGE) or on the second ASAC edge (DMA_interrupt_option=FS_ETPU_ASAC_DMA_INTR_ON_SECOND_EDGE).
- **period (uint24_t)**—The ASAC period, as a number of TCR1 clocks; applies in the periodic mode only (mode=FS_ETPU_ASAC_MODE_PERIODIC).
- **start_offset (uint24_t)**—Used to synchronize various eTPU functions that generate a signal. For ASAC, the first pulse starts the *start_offset* TCR1 clocks after initialization. This parameter applies in the periodic mode only (mode=FS_ETPU_ASAC_MODE_PERIODIC).
- **edge_offset (int24_t)**—Offset, either positive or negative, between PWM period edge_times and the ASAC pulse first edge as number of TCR1 clocks. This parameter applies in the synchronized mode only (mode = FS_ETPU_ASAC_MODE_SYNC).
- **PWMMAC_chan (uint8_t)**—PWMMAC channel number from which the PWM edge_times are taken; applies in the synchronized mode only (mode=FS_ETPU_ASAC_MODE_SYNC).
- **measure_time (uint24_t)**—Time from the first (triggering) edge to the second edge, at which the result queue is supposed to be ready in the DATA_RAM (in TCR1 clocks).
- **periods_per_outerloop (uint8_t)**—A link service request is generated to the *outerloop_chan* each *periods_per_outerloop* period.
- **SC_BC_outerloop_chan (uint8_t)**—Number of a channel on which an outer-loop controller (in slave mode) runs. To avoid activating any inner-loop controller, set the number to a channel with priority disabled. This parameter should be assigned a value of 0-31 for ETPU_A, and of 64-95 for ETPU_B.
- **PMSMVC_ACIMVC_innerloop_chan (uint8_t)**—Number of a channel on which an inner-loop controller (in slave mode) runs. To avoid activating any inner-loop controller, set a number of a channel with priority disabled. This parameter should be assigned a value of 0-31 for ETPU_A, and of 64-95 for ETPU_B.
- **result_queue (uint32_t *)**—Pointer to the result queue. Result queue is an array of 16-bit words that contains the measured values.
- **queue_offset_a (uint8_t)**—Sample A position in the result queue, offset in bytes.
- **queue_offset_b (uint8_t)**—Sample B position in the result queue, offset in bytes.
- **queue_offset_c (uint8_t)**—Sample C position in the result queue, offset in bytes.
- **queue_offset_d (uint8_t)**—Sample D position in the result queue, offset in bytes.
- **forget_factor_a (fract24_t)**—EWMA filter forgettingfFactor applied to sample A. This parameter must be a value between 0 (0x000000) and 1 (0x7FFFFF), usually close to 1, assigned in fract24.
- **forget_factor_b (fract24_t)**—EWMA filter forgetting factor applied to sample B. This parameter must be a value between 0 (0x000000) and 1 (0x7FFFFF), usually close to 1, assigned in fract24.
- **forget_factor_c (fract24_t)**—EWMA filter forgetting factor applied to sample C. This parameter must be a value between 0 (0x000000) and 1 (0x7FFFFF), usually close to 1, assigned in fract24.

- **forget_factor_d (fract24_t)**—EWMA filter forgetting factor applied to sample D. This parameter must be a value between 0 (0x000000) and 1 (0x7FFFFF), usually close to 1, assigned in fract24.
- **bit_shift (uint8_t)** —Defines how to align data from the result queue into fract24 (or int24); should be assigned one of these values:
  - FS_ETPU_ASAC_SHIFT_LEFT_BY_8
  - FS_ETPU_ASAC_SHIFT_LEFT_BY_10
  - FS_ETPU_ASAC_SHIFT_LEFT_BY_12
  - FS_ETPU_ASAC_SHIFT_LEFT_BY_16
- **dtc_threshold (fract24_t)**—Defines the threshold of dead-time compensation states; must be a value between 0 and 1. This parameter applies only if DTC_option is set to FS_ETPU_ASAC_DTC_ON.
- **outputA_chan (uint8_t)**—ASAC writes processed sample_A to a recipient function input parameter. This is the output_A recipient function channel number. This parameter should be assigned a value of 0-31 for ETPU_A, and of 64-95 for ETPU_B.
- **outputA_offset (uint16_t)**—ASAC writes processed sample_A to a recipient function input parameter. This is the output_A recipient function parameter offset. Function parameter offsets are defined in etpu_<func>_auto.h file.
- **outputB_chan (uint8_t)**—ASAC writes processed sample_B to a recipient function input parameter. This is the output_B recipient function channel number. This parameter should be assigned a value of 0-31 for ETPU_A and of 64-95 for ETPU_B.
- **outputB_offset (uint16_t)**—ASAC writes processed sample_B to a recipient function input parameter. This is the output_B recipient function parameter offset. Function parameter offsets are defined in etpu_<func>_auto.h file.
- **outputC_chan (uint8_t)**—ASAC writes processed sample_C to a recipient function input parameter. This is the output_C recipient function channel number. This parameter should be assigned a value of 0-31 for ETPU_A and of 64-95 for ETPU_B.
- **outputC_offset (uint16_t)**—ASAC writes processed sample_C to a recipient function input parameter. This is the output_C recipient function parameter offset. Function parameter offsets are defined in etpu_<func>_auto.h file.
- **outputD1_chan (uint8_t)**—The number of the first channel that requires processed sample_D. (ASAC output_d receiver1).
- **outputD1_offset (uint16_t)**—Defines the processed sample_D offset in the scope of outputD1_chan function parameters.
- **outputD2_chan (uint8_t)**—The number of the first channel that requires processed sample_D. (ASAC output_d receiver2)
- **outputD2_offset (uint16_t)**—Defines the processed sample_D offset in the scope of outputD2_chan function parameters.
- **eQADC_cmds_number (uint8_t)**—The total number of eQADC conversion commands, which should be sent through eDMA to eQADC when the first triggering ASAC edge occurs.

- **eQADC_cmds_current_measurements (etpu_asac_eQADC_cmds_t *)**—Pointer to the table of eQADC conversion commands for current measurement.
- **eQADC_cmds_other_measurements (uint32_t *)**—Pointer to the table of eQADC conversion commands for measurement of other quantities (DC_BUS voltage, BackEMF, temperature).

4.2 Change Operation Functions

4.2.1 int32_t fs_etpu_asac_measure_dc_offsets(uint8_t channel, int8_t measure_dc_offsets_mask)

This function enables setting the *dc_offsets* by measurement. Ensure the power-stage is in a stand-by mode and call this function. The values measured represent the DC offsets, and are stored as *dc_offsets* parameters. Later the *dc_offsets* are used to remove the DC-offsets from measured values. This function has these parameters:

- **channel (uint8_t)**—The ASAC channel number; must be assigned the same value as the channel parameter of the initialization function was assigned.
- **measure_dc_offsets_mask (int8_t)**—Determines which samples can be used as *dc_offsets*; should be assigned one of these values:
  — FS_ETPU_ASAC_DC_OFFSET_SAMPLE_A_B_C
  — FS_ETPU_ASAC_DC_OFFSET_SAMPLE_D
  — FS_ETPU_ASAC_DC_OFFSET_SAMPLE_A_B_C_D

This function returns 0 if the DC offsets were successfully measured. If the ASAC channel has any pending HSRs, the DC offsets are not set and this function should be called again later. In this case, a sum of pending HSR numbers is returned.

4.2.2 int32_t fs_etpu_asac_set_dc_offsets_ABCD(uint8_t channel, ufract24_t dc_offset_a, ufract24_t dc_offset_b, ufract24_t dc_offset_c, ufract24_t dc_offset_d)

This function enables to set the *dc_offsets* of samples A, B, C, and D by values. It has these parameters:

- **channel (uint8_t)**—The ASAC channel number; must be assigned the same value as the channel parameter of the initialization function was assigned.
- **dc_offset_a (ufract24_t)**—The DC offset to remove from sample A. The DC offset must be in the same format as the sample: after the bit alignment.
- **dc_offset_b (ufract24_t)**—The DC offset to remove from sample B. The DC offset must be in the same format as the sample: after the bit alignment.
- **dc_offset_c (ufract24_t)**—The DC offset to remove from sample C. The DC offset must be in the same format as the sample: after the bit alignment.
- **dc_offset_d (ufract24_t)**—The DC offset to remove from sample D. The DC offset must be in the same format as the sample: after the bit alignment.

4.2.3 int32_t fs_etpu_asac_set_dc_offsets_ABC(uint8_t channel, ufract24_t dc_offset_a, ufract24_t dc_offset_b, ufract24_t dc_offset_c)

This function enables to set the *dc_offsets* of samples A, B, and C by values. It has these parameters:

- **channel (uint8_t)**—The ASAC channel number; must be assigned the same value as the channel parameter of the initialization function was assigned.
- **dc_offset_a (ufract24_t)**—The DC offset to remove from sample A. The DC offset must be in the same format as the sample: after the bit alignment.
- **dc_offset_b (ufract24_t)**—The DC offset to remove from sample B. The DC offset must be in the same format as the sample: after the bit alignment.
- **dc_offset_c (ufract24_t)** —The DC offset to remove from sample C. The DC offset must be in the same format as the sample: after the bit alignment.

4.2.4 int32_t fs_etpu_asac_set_dc_offset_D(uint8_t channel, ufract24_t dc_offset_d)

This function enables to set the *dc_offset* of sample D by values. It has these parameters:

- **channel (uint8_t)**—The ASAC channel number; must be assigned the same value as the channel parameter of the initialization function was assigned.
- **dc_offset_d (ufract24_t)**—The DC offset to remove from sample D. The DC offset must be in the same format as the sample: after the bit alignment.

4.3 Value Return Functions

4.3.1 fract24_t fs_etpu_asac_get_outputA(uint8_t channel)

This function returns outputA value after filtration. It has this parameter:

- **channel (uint8_t)**—The ASAC channel number; must be assigned the same value as the channel parameter of the initialization function was assigned.

4.3.2 fract24_t fs_etpu_asac_get_outputB(uint8_t channel)

This function returns outputB value after filtration. It has this parameter:

- **channel (uint8_t)**—The ASAC channel number; must be assigned the same value as the channel parameter of the initialization function was assigned.

4.3.3 fract24_t fs_etpu_asac_get_outputC(uint8_t channel)

This function returns outputC value after filtration. It has this parameter:

- **channel (uint8_t)**—The ASAC channel number; must be assigned the same value as the channel parameter of the initialization function was assigned.

4.3.4 fract24_t fs_etpu_asac_get_outputD(uint8_t channel)

This function returns outputD value after filtration. It has this parameter:

- **channel (uint8_t)**—The ASAC channel number; must be assigned the same value as the channel parameter of the initialization function was assigned.

4.3.5 fract24_t fs_etpu_asac_get_phase_currents(uint8_t channel, asac_abc_t * p_i_abc)

This function returns phase currents (outputs A, B, and C after filtration). It has these parameters:

- **channel (uint8_t)**—The ASAC channel number; must be assigned the same value as the channel parameter of the initialization function was assigned.
- **p_i_abc (asac_abc_t *)**—The pointer to return structure of phase currents.

4.3.6 fract24_t fs_etpu_asac_get_sampleA(uint8_t channel)

This function returns outputA value prior to filtration. It has this parameter:

- **channel (uint8_t)**—The ASAC channel number; must be assigned the same value as the channel parameter of the initialization function was assigned.

4.3.7 fract24_t fs_etpu_asac_get_sampleB(uint8_t channel)

This function returns outputB value prior to filtration. It has this parameter:

- **channel (uint8_t)**—The ASAC channel number; must be assigned the same value as the channel parameter of the initialization function was assigned.

4.3.8 fract24_t fs_etpu_asac_get_sampleC(uint8_t channel)

This function returns outputC value - prior to filtration. This function has the following parameter:

- **channel (uint8_t)** - This is the ASAC channel number. This parameter must be assigned the same value as the channel parameter of the initialization function was assigned.

4.3.9 fract24_t fs_etpu_asac_get_sampleD(uint8_t channel)

This function returns outputD value prior to filtration. It has this parameter:

- **channel (uint8_t)**—The ASAC channel number; must be assigned the same value as the channel parameter of the initialization function was assigned.

4.3.10 fs_etpu_asac_get_phase_currents_samples(uint8_t channel, asac_abc_t * p_i_abc)

This function returns phase currents prior to filtration. It has these parameters:

- **channel (uint8_t)**—The ASAC channel number; must be assigned the same value as the channel parameter of the initialization function was assigned.
- **p_i_abc (asac_abc_t *)**—The pointer to return structure of phase currents.

4.3.11 ufract24_t fs_etpu_asac_get_dc_offsetA(uint8_t channel)

This function returns sampleA DC offset. It has this parameter:

- **channel (uint8_t)** —The ASAC channel number; must be assigned the same value as the channel parameter of the initialization function was assigned.

4.3.12 ufract24_t fs_etpu_asac_get_dc_offsetB(uint8_t channel)

This function returns sampleB DC offset. It has this parameter:

- **channel (uint8_t)**—The ASAC channel number; must be assigned the same value as the channel parameter of the initialization function was assigned.

4.3.13 ufract24_t fs_etpu_asac_get_dc_offsetC(uint8_t channel)

This function returns sampleC DC offset. It has this parameter:

- **channel (uint8_t)**—The ASAC channel number; must be assigned the same value as the channel parameter of the initialization function was assigned.

4.3.14 ufract24_t fs_etpu_asac_get_dc_offsetD(uint8_t channel)

This function returns sampleD DC offset. It has this parameter:

- **channel (uint8_t)**—The ASAC channel number; must be assigned the same value as the channel parameter of the initialization function was assigned.

4.3.15 uint32_t fs_etpu_asac_get_eQADC_cmds_addr(uint8_t channel)

This function returns address of the eQADC command queue beginning. It has this parameter:

- **channel (uint8_t)**—The ASAC channel number; must be assigned the same value as the channel parameter of the initialization function was assigned.

5 Example Use of Function

5.1 Demo Applications

Using the ASAC eTPU function is demonstrated in these applications:

- “Permanent Magnet Synchronous Motor Vector Control, Driven by eTPU on MCF523x,” AN3002.
- “Permanent Magnet Synchronous Motor Vector Control, Driven by eTPU on MPC5500,” AN3206.
- “AC Induction Motor Vector Control, Driven by eTPU on MPC5500,” AN3001.
- “Permanent Magnet Synchronous Motor with Resolver, Vector Control, Driven by eTPU on MPC5500,” AN4480.

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

5.1.1 Function Calls

An example of ASAC initialization and using ASAC API functions is stated in the following lines. It deals with pieces of code used in the demo application “PMSM Vector Control” running on PowerPC MPC5554. The ASAC function is initialized in PWM synchronized mode with the pulse low polarity (high-low edge triggers the A/D converter). Three phase currents and DC-bus voltage are processed. Based on the actual motor position, only two phase currents are measured while the third one is calculated. The ASAC function synchronizes processing of the speed controller and PMSM vector control eTPU functions.

```
/*******************************************************************************
* Parameters
*******************************************************************************/
ASAC_channel = 14;
measure_time_us = 8;
p_result_queue = (uint32_t *)0xc3fc8110;
ia_queue_offset = 0;
ib_queue_offset = 2;
ic_queue_offset = 4;
u_dcbus_queue_offset = 6;
filter_time_constant_i_us = 200;
filter_time_constant_u_us = 1000;

/*******************************************************************************
 *
 * Initialize Analog Sensing for AC Motors (ASAC)
 *
 ******************************************************************************/
/*******************************************************************************
 * 1) Calculate forgetting factor
 *******************************************************************************
 *  The ASAC function passes the samples through an EWMA filter. The EWMA filter
 *  forgetting factor can be calculated based on the required filter time
 *  constant according to the following equations:
 *
 *    filter_time_constant = (effective_filter_length - 1)/sample_frequency
 *
 *    forgetting_factor = (effective_filter_length - 1)/effective_filter_length
 *
 *  =>
 *                          filter_time_constant*sample_frequency
 *    forgetting_factor = -----------------------------------------
 *                        filter_time_constant*sample_frequency + 1
 *
 *    where the sample_frequency is equal to PWM_freq.
 *
 *
 *  Let's denote: A = filter_time_constant*sample_frequency
 *                B = filter_time_constant*sample_frequency + 1
 *
 *  Then, using integer arithmetics:
 *    (fract24)forgetting_factor = (A<<23)/B + (((((A<<23)%B)<<8)/B)>>8)
 *
 *******************************************************************************/
   A = ASAC_filter_time_constant_i_us*PWM_freq_hz;
   B = (ASAC_filter_time_constant_i_us*PWM_freq_hz + 1000000)>>8;
   ASAC_forget_factor_i = ((A/B)<<15) + ((((A%B)<<8)/B)<<7);
   A = ASAC_filter_time_constant_u_us*PWM_freq_hz;
   B = (ASAC_filter_time_constant_u_us*PWM_freq_hz + 1000000)>>8;
   ASAC_forget_factor_u = ((A/B)<<15) + ((((A%B)<<8)/B)<<7);

/*******************************************************************************
 * 2) Calculate edge_offset
 *******************************************************************************
 * trigger the A/D converter queue 16ns before the PWM period start due to ETRIG
 * filter in eQADC module
 *******************************************************************************/
   edge_offset = -(16*(etpu_tcr1_freq/1000)/1000000);

/*******************************************************************************
 * 3) eQADC conversion commands table definition
 *******************************************************************************
 *          trig  ADC0:     ADC1:
 *  CFIFO2: ----> iA/iB/iC  iB/iC/iA - dynamically set according to sector value
 *                iC/iA/iB
 *                UdcBus
 *  RFIFO0: iA
 *  RFIFO1: iB
 *  RFIFO2: iC   UdcBus
 *******************************************************************************/
```

```
    eQADC_cmds_current_measurements.ADC0_iA_cmd = 0x01000200; /* iA ADC0 RFIFO0 */
    eQADC_cmds_current_measurements.ADC1_iA_cmd = 0x03000200; /* iA ADC1 RFIFO0 */
    eQADC_cmds_current_measurements.ADC0_iB_cmd = 0x01100300; /* iB ADC0 RFIFO1 */
    eQADC_cmds_current_measurements.ADC1_iB_cmd = 0x03100300; /* iB ADC1 RFIFO1 */
    eQADC_cmds_current_measurements.ADC0_iC_cmd = 0x01200400; /* iC ADC0 RFIFO2 */
    eQADC_cmds_current_measurements.ADC1_iC_cmd = 0x03200400; /* iC ADC1 RFIFO2 */
    eQADC_cmds_other_measurements = 0x81200000; /* UdcBus ADC0 RFIFO2 */
/*******************************************************************************
 * 4) Initialize ASAC function
 ******************************************************************************/
err_code = fs_etpu_asac_init(
      ASAC_channel, /* channel */
      FS_ETPU_PRIORITY_MIDDLE, /* priority */
      ASAC_polarity, /* polarity */
      FS_ETPU_ASAC_MODE_SYNC, /* mode */
      FS_ETPU_ASAC_SAMPLE_A_B_C_D, /* measure_samples_mask */
      FS_ETPU_ASAC_DTC_ON, /* DTC_option */
      FS_ETPU_ASAC_PHASE_CURR_USE_SECTOR, /*phase_current_option */
      FS_ETPU_ASAC_PHASE_CURRENTS_NEG, /* negate_phase_currents */
      FS_ETPU_ASAC_CFIFO_UPDATE_ON, /* cfifo_update */
      FS_ETPU_ASAC_DMA_INTR_ON_SECOND_EDGE, /* DMA_interrupt_option */
      0, /* period */
      0, /* start_offset */
      ASAC_edge_offset, /* egde_offset */
      PWM_master_channel, /* PWMMAC_chan */
      (ASAC_measure_time_us * etpu_tcr1_freq)/1000000, /* measure_time */
      PWM_freq_hz/SC_freq_hz, /* periods_per_outerloop */
      SC_channel, /* SC_BC_outerloop_chan */
      PMSMVC_channel, /* PMSMVC_ACIMVC_innerloop_chan */
      ASAC_result_queue, /* pointer to result queue */
      ASAC_ia_queue_offset, /* queue_offset_a */
      ASAC_ib_queue_offset, /* queue_offset_b */
      ASAC_ic_queue_offset, /* queue_offset_c */
      ASAC_u_dcbus_queue_offset, /* queue_offset_d */
      ASAC_forget_factor_i, /* forget_factor_a */
      ASAC_forget_factor_i, /* forget_factor_b */
      ASAC_forget_factor_i, /* forget_factor_c */
      ASAC_forget_factor_u, /* forget_factor_d */
      ASAC_bit_shift, /* bit_shift */
      0x028F5C, /* dtc_threshold: 0.02 */
      PMSMVC_channel, /* outputA_chan */
      FS_ETPU_PMSMVC_IABC_OFFSET, /* outputA_offset */
      PMSMVC_channel, /* outputB_chan */
      FS_ETPU_PMSMVC_IABC_OFFSET + 4, /* outputB_offset */
      PMSMVC_channel, /* outputC_chan */
      FS_ETPU_PMSMVC_IABC_OFFSET + 8, /* outputC_offset */
      PMSMVC_channel, /* outputD1_chan */
      FS_ETPU_PMSMVC_UDCBUSACTUAL_OFFSET, /* outputD1_offset */
      BC_channel, /* outputD2_chan */
```

```
        FS_ETPU_BC_UDCBUSMEASURED_OFFSET, /* outputD2_offset */
        4, /* eQADC_cmds_number */
        &eQADC_cmds_current_measurements, /* *eQADC_cmds_current_measurements */
        &eQADC_cmds_other_measurements /* *eQADC_cmds_other_measurements */
    );
    if (err_code != 0)
        return (err_code);

/* measure phase currents DC offsets */
fs_etpu_asac_measure_dc_offsets( ASAC_CHANNEL, FS_ETPU_ASAC_DC_OFFSET_SAMPLE_A_B_C);


/* read sampleA DC offset */
dc_offsetA = fs_etpu_asac_get_dc_offsetA( ASAC_CHANNEL);


/* read sample A prior to filtration */
sampleA = fs_etpu_asac_get_sampleA( ASAC_CHANNEL);


/* read processed sample A */
outputA = fs_etpu_asac_get_outputA( ASAC_CHANNEL);
```

6 Summary and Conclusions

This eTPU application note provides a description of analog sensing for AC Motors eTPU function usage and examples. The simple C interface routines to the ASAC eTPU function enable easy implementation of this function in applications. The demo application is targeted at the MCF523*x* family of devices, but it can easily be reused with any device that has an eTPU.

6.1 References

1. “The Essential of Enhanced Time Processing Unit,” AN2353
2. “General C Functions for the eTPU,” AN2864
3. “Using the AC Motor Control eTPU Function Set (set4),” AN2968
4. *Enhanced Time Processing Unit Reference Manual*, ETPURM
5. eTPU Graphical Configuration Tool, http://www.freescale.com/etpu, ETPUGCT
6. “Using the AC Motor Control PWM eTPU Functions,” AN2969.
7. “Permanent Magnet Synchronous Motor Vector Control, Driven by eTPU on MCF523x,” AN3002

8. “Permanent Magnet Synchronous Motor Vector Control, Driven by eTPU on MPC5500,” AN3206
9. “AC Induction Motor Vector Control, Driven by eTPU on MPC5500,” AN3001
10. “Permanent Magnet Synchronous Motor with Resolver, Vector Control, Driven by eTPU on MPC5500,” AN4480

7 Revision history

Table 1. Revision history

Revision number	Revision date	Description of changes
2	01 May 2012	Updated for support of motor drives with resolver position sensor.

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: AN2970
Rev. 2
02/2012