Atmel AT03078: Using the AFEC in SAM4E Microcontroller

Atmel 32-bit Microcontroller

Features

- AFEC in SAM4E
- Single ended conversion
- Temperature measurement
- Dual conversion multi-triggered by different event
- Automatic comparison function
- Enhanced resolution mode
- Differential Input conversion
- Application measuring low input signal with high precision

Introduction

The purpose of this application note is to explain how to use the Analog Front End Controller (AFEC) implemented in the SAM4E microcontroller family devices. It gives brief description of AFEC module characteristics. It also provides implementation of some functions on SAM4E-EK, such as input gain, temperature measurements, conversion triggered by Timer Counter, automatic comparison and enhanced resolution. These features can be used in user’s practical application. Finally, it describes an application measuring low input signal with high precision.
Table of Contents

1. AFEC Module Introduction ................................................................. 3
   1.1 Selecting Voltage Reference.......................................................... 3
   1.2 Automatic Calibration ................................................................. 4
   1.3 Reducing Power Consumption ..................................................... 4
   1.4 User Sequence Conversion ......................................................... 4
   1.5 Internal AFEC Offset ................................................................. 5
   1.6 Main Structure Used In AFEC API .............................................. 5

2. Single Ended Conversion ................................................................. 6
   2.1 Introduction .................................................................................. 6
   2.2 Implementation ............................................................................ 6

3. Temperature Measurement .............................................................. 8
   3.1 Introduction .................................................................................. 8
   3.2 Implementation ............................................................................ 8

4. Dual AFEC Conversion Triggered by Different Event ..................... 9
   4.1 Introduction .................................................................................. 9
   4.2 Implementation ............................................................................ 9

5. Automatic Comparison Function .................................................... 12
   5.1 Introduction .................................................................................. 12
   5.2 Implementation ............................................................................ 12

6. Enhanced Resolution Mode .............................................................. 13
   6.1 Introduction .................................................................................. 13
   6.2 Implementation ............................................................................ 13

7. Differential Input Conversion ........................................................... 15
   7.1 Introduction .................................................................................. 15
   7.2 Implementation ............................................................................ 15

8. Sampling Small Signal Using AFEC ................................................ 17
   8.1 Introduction .................................................................................. 17
   8.2 Implementation ............................................................................ 17

9. Revision History ............................................................................... 19
1. **AFEC Module Introduction**

The AFEC module manages an Analog Front End based on a high-performance Analog-to-Digital converter capable of conversion rates up to one million samples per second (Msps) with a resolution of 12 bits. It integrates a 16-to-1 analog multiplexer, making possible the analog-to-digital conversions of 16 analog lines. The conversion is performed on a full range between 0V and the reference voltage ADVREF. The input is selectable, both single-ended and fully differential measurements can be done. The conversion results are reported in a common register AFEC_LCDR for all channels, as well as in a channel-dedicated register AFEC_CDR. Relative voltage can be calculated by Formula 1.1, TOP is the top value given by the configured resolution. For 12 bit mode TOP is 4095 and 16 bit mode TOP is 16 * 4095.

\[
V_{channel} = \frac{RESULT \times VREF}{TOP}
\]  
(Formula 1.1)

The AFEC has a built in Programmable Gain Amplifier (PGA) and Programmable Offset per channel (through a DAC) to allow full range conversion.

The temperature sensor is internally connected to channel 15. The measure of the temperature can be made in different ways through the AFEC.

Conversions of the active analog channels are started with a software or hardware trigger. The software trigger is provided by writing the Control Register (AFEC_CR). The hardware trigger can be one of the TIOA outputs of the Timer Counter channels, PWM Event line, or the external trigger input of the AFEC (ADTRG).

The AFEC features automatic comparison functions. It compares converted values to a low threshold, a high threshold or both.

The AFEC supports enhanced resolution mode which can be extended up to 16-bit resolution by digital averaging. In this mode the AFEC trades conversion performance for accuracy by averaging multiple samples, thus providing a digital low-pass filter function.

The Atmel Software Framework (ASF) provides API to implement AFEC module functions.

1.1 **Selecting Voltage Reference**

There are two voltage reference sources on SAM4E-EK, one is 3.3V and the other is 5V. User can select it by jumper JP3. Figure 1.1 shows it.

![Figure 1-1. Voltage Reference Source Selections](image-url)
1.2 Automatic Calibration

The automatic calibration sequence can be started at any time, in ASF it is done by calling:

```c
afec_start_calibration()
```

EOCAL bit in AFE_ISR register is set after the calibration is finished, it is checked with:

```c
afec_get_interrupt_status()
```

For each calibrated channels, the corresponding OFFSET bit in AFEC_CDOR register must be also programmed to 1 prior to launch the autocalibration sequence.

If free run mode (never wait for any trigger) is used then automatic calibration must be run before enabling the free run mode. In any case automatic calibration should not be started while free run mode is active.

If the gain settings for a given channel are changed, the automatic calibration must then be started again. Changing the AFE reference voltage (ADVREF) requires a new calibration as well.

1.3 Reducing Power Consumption

The AFEC provides sleep mode and fast wake-up mode to further reduce power consumption.

In sleep mode the AFEC automatically deactivates when it is not being used for conversions. The AFEC core and reference voltage circuitry are OFF between conversions.

In fast wake-up mode the AFEC is not fully deactivated while no conversion is requested, thereby providing less power saving but faster wakeup. The Voltage reference is ON between conversions and AFEC core is OFF.

In ASF it is done by calling:

```c
afec_set_power_mode()
```

Table 1.1 shows the state of core and reference voltage in these power modes.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Voltage Reference</th>
<th>AFEC Core</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal mode</td>
<td>ON</td>
<td>ON</td>
</tr>
<tr>
<td>Fast wake-up mode</td>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>Sleep mode</td>
<td>OFF</td>
<td>OFF</td>
</tr>
</tbody>
</table>

1.4 User Sequence Conversion

Normally the enabled channels start conversion in numeric order. For example, if the channel0, channel4 and channel8 are enabled, then the conversion sequence is channel0 → channel4 → channel8.

Specially, user also can configure sequence by calling:

```c
afec_configure_sequence(ch_list)
```

For example, if user wants to configure the conversion sequence as channel8 -> channel4 -> channel0, ch_list is set as below:

```c
enum afec_channel_num ch_list[] =
{
    AFEC_CHANNEL_8,
    AFEC_CHANNEL_4,
    AFEC_CHANNEL_0
}
```
After configuring the user sequence with above API, the user could get customized conversion sequence.

1.5 Internal AFEC Offset

In AFEC, there is an offset by default (not the user offset at user interface level, AFEC). This intrinsic AFEC offset is used by the AFEC in order to be able to work in negative and positive ranges. So, when users are going to start conversion in default state, all the results will be in range between VDDREF/2 and VDDREF.

If user wants to use the full range (0 ~ VDDREF) in conversion, an offset equal to VDDREF/2 = 0x800 has to be set, it is done by calling:

```
afec_channel_set_analog_offset(0x800)
```

1.6 Main Structure Used In AFEC API

The main structure is as below:

```c
struct afec_config {
    /** Resolution */
    enum afec_resolution resolution;
    /** Master Clock */
    uint32_t mck;
    /** AFEC Clock */
    uint32_t afec_clock;
    /** Start Up Time */
    enum afec_startup_time startup_time;
    /** Analog Settling Time = (settling_time + 1) / AFEC clock */
    enum afec_settling_time settling_time;
    /** Tracking Time = tracktim / AFEC clock */
    uint8_t tracktim;
    /** Transfer Period = (transfer * 2 + 3) / AFEC clock */
    uint8_t transfer;
    /** Analog Change */
    bool anach;
    /** Use Sequence Enable */
    bool useq;
    /** TAG of AFE_LDCR register */
    bool tag;
    /** Single Trigger Mode */
    bool stm;
    /** AFE Bias Current Control */
    uint8_t ibctl;
};
```

```c
struct afec_ch_config {
    /** Differential Mode */
    bool diff;
    /** Gain Value */
    enum afec_gainvalue gain;
};
```

The `afec_config` is the structure for AFEC module configuration, which will be used in AFEC initialization API. Its default value can get by calling:

```
afec_get_config_defaults(afec_config)
```

The `afec_ch_config` is the structure for AFEC channel configuration, which will be used in AFEC channel configuration API. Its default value can get by calling:

```
afec_ch_get_config_defaults(afec_ch_config)
```
2. Single Ended Conversion

2.1 Introduction

Single ended conversion is the most basic type of conversion. A single voltage, either from an external input or from an internal reference, is used as input to the AFEC. Using gain features in AFEC can allow small signal input.

2.2 Implementation

To implement the single ended conversion, connecting the conversion signal with AFEC channel is needed. After enabling AFEC clock, it could configure resolution, clock frequency, trigger mode. The input mode and gain of selected channel could be configured too. Enabling selected channel is needed before AFEC start conversion. When the conversion is end, the data ready flag would be set in AFEC_ISR register. The interrupt will be generated if enabling data ready interrupt. It can get conversion result in interrupt handler.

For example, AFEC0 channel4 is used to convert an external signal on SAM4E-EK. The step of implementation is as below.

1) Connect this signal with BNC CN2, which is connected with AFEC0 channel4. Figure 2.1 show the connection.
2) Configure AFEC0 resolution and clock, such as 12-bit resolution, 6 MHz clock, etc.
3) Configure input mode, gain for channel4, such as single ended input mode, gain = 1.
4) Set trigger mode to software trigger.
5) Enable channel4.
6) Enable data ready interrupt and set callback function.
7) The interrupt will be generated when conversion data is ready. In the callback function, it can read latest conversion data. The voltage value can get according to Formula 1.1.

Figure 2-1. Connection To External Signal

To implement this, the below programming steps could be used with ASF.

Step 2_1: Enable AFEC0 clock with Power Management Controller (PMC) module.

afec_enable(AFEC0)
Step 2_2: Get AFEC0 default configuration.

afec_get_config_defaults(afec_config)

Note: The section 1.6 describes the afec_config structure and its default value is as below:

afec_config.resolution = AFEC_12_BITS;
afec_config.mck = sysclk_get_cpu_hz();
afec_config.clock = 6000000UL;
afec_config.startup_time = 4;
afec_config.settling_time = 0;
afec_config.tracktim = 2;
afec_config.transfer = 1;
afec_config.anach = true;
afec_config.usetq = false;
afec_config.tag = true;
afec_config.stm = true;
afec_config.ibctl = 1;

Step 2_3: Initialize AFEC0 with above default configuration.

afec_init(AFEC0, afec_config)

Step 2_4: Get AFEC0 channel default configuration.

afec_ch_get_config_defaults(afec_ch_config)

Note: The section 1.6 describes the afec_ch_config structure and its default value is as below.

afec_ch_config.diff = false;
afec_ch_config.gain = 1;

Step 2_5: Configure channel4 with above default channel configuration.

afec_ch_set_config(AFEC0, AFEC_CHANNEL_4, afec_ch_config)

Step 2_6: Set the conversion trigger mode to software trigger.

afec_set_trigger(AFEC0, AFEC_TRIG_SW)

Step 2_7: Enable channel4.

afec_channel_enable(AFEC0, AFEC_CHANNEL_4)

Step 2_8: Enable conversion data ready interrupt and set callback function.

afec_set_callback(AFEC0, AFEC_INTERRUPT_DATA_READY, callback, 1)

Step 2_9: Start trigger AFEC0 conversion.

afec_start_software_conversion(AFEC0)

Step 2_10: Read the last converted data.

afec_get_latest_value(AFEC0)
3. Temperature Measurement

3.1 Introduction

There is a temperature sensor internally connected to AFEC0 channel15, so it is convenient for user to apply in temperature measurement without extra component. It can use below formula to calculate the temperature value.

\[
\text{TEMP} = \frac{(V_{\text{Temp}} - 1440)}{4.7} + 27
\]

(Formula 3.1)

3.2 Implementation

The implementation of temperature measurement is to use AFEC channel to convert signal from temperature sensor. To use the internal temperature sensor in AFEC, AFEC0 channel15 should be enabled. After get the conversion data, it can calculate the temperature according to Formula 1.1 and Formula 3.1.

To implement this, the below programming steps could be used with ASF.

Step 3_1: Enable AFEC0 clock with PMC module.

\[\text{afec\_enable(AFEC0)}\]

Step 3_2: Get AFEC0 default configuration.

\[\text{afec\_get\_config\_defaults(afec\_config)}\]

Note: About default configuration, please refer to Step 2_2 description.

Step 3_3: Initialize AFEC0 with above default configuration.

\[\text{afec\_init(afec\_config)}\]

Step 3_4: Set the conversion trigger mode to software trigger.

\[\text{afec\_set\_trigger(AFEC0, AFEC\_TRIG\_SW)}\]

Step 3_5: Enable channel15 to turn on the temperature sensor.

\[\text{afec\_channel\_enable(AFEC0, AFEC\_CHANNEL\_15)}\]

Step 3_6: Enable conversion data ready interrupt and set callback function.

\[\text{afec\_set\_callback(AFEC0, AFEC\_INTERRUPT\_DATA\_READY, callback, 1)}\]

Step 3_7: Start trigger AFEC conversion.

\[\text{afec\_start\_software\_conversion(AFEC0)}\]

Step 3_8: Read the last converted data.

\[\text{afec\_get\_latest\_value(AFEC0)}\]
4. Dual AFEC Conversion Triggered by Different Event

4.1 Introduction

The AFEC has two instances AFEC0 and AFEC1, so user can set different configuration to each instance, such as resolution, trigger mode, etc.

4.2 Implementation

The dual AFEC conversion is for AFEC0 and AFEC1 to be configured to do conversions at the same time. The implementation is like two single ended conversions, but one for each AFEC instance.

For example, a potentiometer is connected to AFEC0 channel5 and an external input is connected to AFEC1 channel0 on SAM4E-EK. Figure 4.1 and Figure 4.2 show the connection.

The AFEC0 conversion is triggered by Timer Counter (TC) with a fixed rate of one second. The AFEC1 conversion is triggered by software every three seconds.

After getting the conversion results, it can calculate the voltage values of AFEC0 channel5 and AFEC1 channel0 according to Formula 1.1. Figure 4.3 shows the results. From the results it could be found AFEC1 channel0 converts one time after AFEC0 channel5 converts three times. The reason is due to different configurations in trigger mode and conversion rates.

Figure 4-1. Connection To Potentiometer
To implement this, the below programming steps could be used with ASF.

**Step 4_1:** Enable AFEC0 and AFEC1 clock with PMC module.
```
afec_enable(AFEC0)
afec_enable(AFEC1)
```

**Step 4_2:** Get AFEC0 and AFEC1 default configuration.
```
afec_get_config_defaults(afec_config)
```

**Note:** About default configuration, please refer to Step 2_2 description.

**Step 4_3:** Initialize AFEC0 and AFEC1 with above default configuration.
```
afec_init(AFEC0, afec_config)
afec_init(AFEC1, afec_config)
```

**Step 4_4:** Set the AFEC0 conversion trigger mode to hardware trigger and trigger event is TIOA output of Timer Counter (TC) channel.
```
afec_set_trigger(AFEC0, AFEC_TRIG_TIO_CH_0)
```

**Step 4_5:** Configure TC0 channel0 in waveform mode and start TC0.
```
configure_tc_trigger()
```

**Note:** This function is implemented in application file afec_example4.c.

**Step 4_6:** Set the AFEC1 conversion trigger mode to software trigger.
```
afec_set_trigger(AFEC1, AFEC_TRIG_SW)
```

**Step 4_7:** Enable AFEC0 channel5 and AFEC1 channel0.
```
afec_channel_enable(AFEC0, AFEC_CHANNEL_5)
afec_channel_enable(AFEC1, AFEC_CHANNEL_0)
```

**Step 4_8:** Enable conversion data ready interrupt and register callback function.
```
afec_set_callback(AFEC0, AFEC_INTERRUPT_DATA_READY, callback0, 1)
```
afec_set_callback(AFEC1, AFEC_INTERRUPT_DATA_READY, callback1, 1)

**Step 4_9:** Start trigger AFEC1 conversion every three seconds in while loop.

afec_start_software_conversion(AFEC1)

**Step 4_10:** Read the last converted data.

afec_get_latest_value(AFEC0)
afec_get_latest_value(AFEC1)

**Figure 4-3. Dual AFEC Conversion Result**

![Dual AFEC Conversion Result](image.png)
5. **Automatic Comparison Function**

5.1 **Introduction**

This feature is useful if applications wish to check the input signal is within a specific voltage. Normally, the application needs to check the result in software for every conversion. But with this feature, the AFEC will inform the application via an interrupt or event when the configured threshold is reached, which will reduce the CPU load.

5.2 **Implementation**

To implement automatic comparison function, a signal connecting with AFEC input channel is needed. The window threshold and comparison mode also need be configured before AFEC conversion start. Comparison event interrupt is enabled and callback function is set. The interrupt will be generated when the input voltage satisfies threshold condition according to comparison mode.

For example, a potentiometer as Figure 4.1 is used to implement this function. The window threshold is configured to 0 ~ 1650mv and comparison mode is in the window. The interrupt will be generated when the potentiometer voltage is adjusted in range 0 ~ 1650mv. In callback function, the terminal will output current potentiometer voltage and the result is as Figure 5.1 showing.

To implement this, the below programming steps could be used with ASF.

**Step 5_1:** Enable AFEC0 clock with PMC module.

```c
afec_enable(AFEC0)
```

**Step 5_2:** Get AFEC0 default configuration.

```c
afec_get_config_defaults(afec_config)
```

*Note:* About default configuration, please refer to Step 2_2 description.

**Step 5_3:** Initialize AFEC0 with above default configuration.

```c
afec_init(AFEC0, afec_config)
```

**Step 5_4:** Set the conversion trigger mode to software trigger.

```c
afec_set_trigger(AFEC0, AFEC_TRIG_SW)
```

**Step 5_5:** Set the comparison mode and threshold.

```c
afec_set_comparison_mode(AFEC0, AFEC_CMP_MODE_2, AFEC_CHANNEL_5, 0)
afec_set_comparison_window(AFEC0, 0, 0x800)
```

*Note:* The **AFEC_CMP_MODE_2** is in the window mode, user also can set other mode, such as lower than low threshold of window, higher than high threshold of window and out of the window.

**Step 5_6:** Enable AFEC0 channel5.

```c
afec_channel_enable(AFEC0, AFEC_CHANNEL_5)
```

**Step 5_7:** Enable comparison event interrupt and set callback function.

```c
afec_set_callback(AFEC0, AFEC_INTERRUPT_COMP_ERROR, callback, 1)
```
6. Enhanced Resolution Mode

6.1 Introduction

12-bit resolution is sufficient for most use cases, but if higher accuracy is desired, the AFEC can increase conversion accuracy by using enhanced resolution mode.

6.2 Implementation

The implementation of enhanced resolution mode is based on single ended conversion but modifying the resolution higher than 12-bit in AFEC configuration.

For example, the external input voltage is 100mv and 100.1mv, Figure 6.1 and Figure 6.2 show output results when the resolution is 12-bit. Figure 6.3 and Figure 6.4 show the output results when the resolution is 16-bit.

Comparing the Figure 6.1/6.2 to Figure 6.3/6.4, it could be found that with 12-bit resolution the changes in AFEC output is bigger than 0.1mv or has no change when input voltage changes 0.1mv. But with 16-bit resolution the changes in AFEC output is about 0.1mv and more accurate compared with 12-bit resolution.

The reason is the voltage 'step' representing 1 LSB is $\frac{3300}{4095} = 0.806$ mv with 12-bit resolution, i.e. the accuracy is about 0.8 mv, AFEC cannot distinguish 0.1 mv change from input accurately. But with 16-bit resolution the voltage 'step' representing 1 LSB is $\frac{3300}{(4095*16)} = 0.05$ mv, so AFEC could distinguish 0.1 mv change from input accurately.

To implement this, the below programming steps could be used with ASF.

Step 6_1: Enable AFEC0 clock with PMC module.

\[
\text{afec_enable(AFEC0)}
\]

Step 6_2: Get AFEC0 default configuration.

\[
\text{afec_get_config_defaults(afec_config)}
\]

Note: About default configuration, please refer to Step 2_2 description.

Step 6_3: Modify resolution to 16-bit.

\[
\text{afec_config.resolution = AFEC_16_BITS;}
\]

Step 6_4: Initialize AFEC0 with modified configuration.

\[
\text{afec_init(AFEC0, afec_config)}
\]

Step 6_5: Set the conversion trigger mode to software trigger.

\[
\text{afec_set_trigger(AFEC0, AFEC_TRIG_SW)}
\]

Step 6_6: Enable AFEC0 channel4.

\[
\text{afec_channel_enable(AFEC0, AFEC_CHANNEL_4)}
\]
Step 6_7: Enable conversion data ready interrupt and set callback function.

    afec_set_callback(AFEC0, AFEC_INTERRUPT_DATA_READY, callback, 1)

Step 6_8: Start trigger AFEC0 conversion.

    afec_start_software_conversion(AFEC0)

Step 6_9: Read the last converted data.

    afec_get_latest_value(AFEC0)

Figure 6-1. Output Result With 12-bit Resolution When Input Voltage = 100mv

Voltage: 100.571426 mv
Voltage: 100.813187 mv
Voltage: 100.732597 mv
Voltage: 100.571426 mv

Figure 6-2. Output Result With 12-bit Resolution When Input Voltage = 100.1mv

Voltage: 101.135529 mv
Voltage: 100.893776 mv
Voltage: 101.135529 mv
Voltage: 100.893776 mv

Figure 6-3. Output Result With 16-bit Resolution When Input Voltage = 100 mv

Voltage: 100.702377 mv
Voltage: 100.722527 mv
Voltage: 100.742676 mv
Voltage: 100.747711 mv

Figure 6-4. Output Result With 16-bit Resolution When Input Voltage = 100.1 mv

Voltage: 100.853477 mv
Voltage: 100.873627 mv
Voltage: 100.898811 mv
Voltage: 100.888733 mv
7. **Differential Input Conversion**

7.1 **Introduction**

The AFEC provides another input mode – differential input. By this mode, it can convert differential value between two inputs and range is (–VDDREF / 2, VDDREF / 2).

7.2 **Implementation**

The differential input conversion needs two signal input, one is as positive input and another as negative input. The two input pins must be a pair as Table 7.1 show. Configuring the input mode of channel to differential is needed. The implementation is similar to single ended conversion.

**Table 7-1. Input Pins and Channel Number in Differential Mode**

<table>
<thead>
<tr>
<th>Input Pins</th>
<th>Channel Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD0-AD1</td>
<td>CH0</td>
</tr>
<tr>
<td>AD2-AD3</td>
<td>CH2</td>
</tr>
<tr>
<td>AD4-AD5</td>
<td>CH4</td>
</tr>
<tr>
<td>AD6-AD7</td>
<td>CH6</td>
</tr>
<tr>
<td>AD8-AD9</td>
<td>CH8</td>
</tr>
<tr>
<td>AD10-AD11</td>
<td>CH10</td>
</tr>
<tr>
<td>AD12-AD13</td>
<td>CH12</td>
</tr>
<tr>
<td>AD14-AD15</td>
<td>CH14</td>
</tr>
</tbody>
</table>

For example, the positive input is connected to external input and the negative input is connected to potentiometer as Figure 2.1 and Figure 4.1 show. The channel4 and channel5 should be set working in differential mode before conversion starts. After conversion, it will get the result of differential input. According to Formula 1.1, it can calculate the differential voltage value VDIFF. If adjusting the potentiometer voltage to VPOT, the external input voltage should be VPOT + VDIFF.

To implement this, the below programming steps could be used with ASF.

**Step 7_1:** Enable AFEC0 clock with PMC module.

```c
afec_enable(AFEC0)
```

**Step 7_2:** Get AFEC0 default configuration.

```c
afec_get_config_defaults(afec_config)
```

Note: About default configuration, please refer to Step 2_2 description.

**Step 7_3:** Initialize AFEC0 with above default configuration.

```c
afec_init(AFEC0, afec_config)
```

**Step 7_4:** Get AFEC0 channel default configuration.

```c
afec_ch_get_config_defaults(afec_ch_config)
```

Note: Please refer to Step 2_4 description

**Step 7_5:** Modify channel input mode to differential mode.

```c
afec_ch_config.diff = true
```
Step 7_6: Configure channel4 and channel5 with modified configuration.

afec_ch_set_config(AFEC0, AFEC_CHANNEL_4, afec_ch_config)
afec_ch_set_config(AFEC0, AFEC_CHANNEL_5, afec_ch_config)

Step 7_7: Set the conversion trigger mode to software trigger.

afec_set_trigger(AFEC0, AFEC_TRIG_SW)

Step 7_8: Enable AFEC0 channel4 and channel5.

afec_channel_enable(AFEC0, AFEC_CHANNEL_4)
afec_channel_enable(AFEC0, AFEC_CHANNEL_5)

Step 7_9: Enable conversion data ready interrupt and set callback function.

afec_set_callback(AFEC0, AFEC_INTERRUPT_DATA_READY, callback, 1)

Step 7_10: Start trigger AFEC conversion.

afec_start_software_conversion(AFEC0)

Step 7_11: Read the last converted data.

afec_get_latest_value(AFEC0)
8. Sampling Small Signal Using AFEC

8.1 Introduction
The voltage of signal from sensor is very low, and user can measure this signal with high precision by using several AFEC features mentioned above, such as gain and enhanced resolution mode.

8.2 Implementation
A Brew Master wants to measure the temperature of a process in his brewery. A slow varying signal represents the temperature measurement and the nominal voltage is 10 mv when temperature is 20°C, the voltage / temperature function is 0.1mv/0.01°C.

Because the voltage of signal is very small, to make AFEC work in good linearity area, it should set gain = 4. The input voltage range becomes to (0, VDDREF / 8). When the input voltage is 10mv, the output result should be (VDDREF / 2 + 4 * 10) mv. To distinguish 0.1mv accurately, it also should set resolution = 16-bit. The implementation is similar to single ended conversion.

According to Formula 1.1 it can calculate the voltage value VTEMP, the measurement temperature should be:

\[
(VTEMP - (VDDREF / 2 + 4 * 10)) / 4 * (0.01 / 0.1) + 20
\]

To implement this, the below programming steps could be used with ASF.

**Step 8_1**: Enable AFEC0 clock with PMC module.

```c
afec_enable(AFEC0)
```

**Step 8_2**: Get AFEC0 default configuration.

```c
afec_get_config_defaults(afec_config)
```

Note: About default configuration, please refer to Step 2_2 description.

**Step 8_3**: Modify resolution to 16-bit.

```c
afec_config.resolution = AFEC_16_BITS;
```

**Step 8_4**: Initialize AFEC0 with modified configuration.

```c
afec_init(AFEC0, afec_config)
```

**Step 8_5**: Get AFEC0 channel default configuration.

```c
afec_ch_get_config_defaults(afec_ch_config)
```

Note: About channel default configuration, please refer to Step 2_4 description.

**Step 8_6**: Modify channel4 input mode to differential mode.

```c
afec_ch_config.gain = 4;
```

**Step 8_7**: Configure channel4 with modified configuration.

```c
afec_ch_set_config(AFEC0, AFEC_CHANNEL_4, afec_ch_config)
```

**Step 8_8**: Set the conversion trigger mode to software trigger.

```c
afec_set_trigger(AFEC0, AFEC_TRIG_SW)
```

**Step 8_9**: Enable channel4.

```c
afec_channel_enable(AFEC0, AFEC_CHANNEL_4)
```
Step 8_10: Enable conversion data ready interrupt and set callback function.

    afec_set_callback(AFEC0, AFEC_INTERRUPT_DATA_READY, callback, 1)

Step 8_11: Start trigger AFEC0 conversion.

    afec_start_software_conversion(AFEC0)

Step 8_12: Read the last converted data.

    afec_get_latest_value(AFEC0)
9. Revision History

<table>
<thead>
<tr>
<th>Doc. Rev.</th>
<th>Date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>42143A</td>
<td>06/2013</td>
<td>Initial revision</td>
</tr>
</tbody>
</table>