SMART ARM-based Microcontrollers

AT11483: Quadrature Decoder (QDEC) for SAM3/4 Devices

APPLICATION NOTE

Introduction

This application note describes the basic functionality of the QDECs with code example using Atmel®| SMART SAM3/4 devices. A combination of peripheral modules are used to decode the Quadrature Encoder signals: I/O pins are used as input to the Quadrature Decoder (QDEC), which connects to a Timer/Counter.

Quadrature encoders are used to determine the position and speed of rotary devices such as servo-motors, volume control wheels, and PC mice. The decoded quadrature signals are used as a sensory input for the system to determine the absolute or relative position of the rotary device. This relative position can be used in a control loop for applications such as servo-motor.

The software example mentioned in this document is available in the latest version of the Atmel Software Framework (ASF).

Table of Contents

1. Glossary

ASF	Atmel Software Framework
Atmel Studio	Integrated Development Environment (IDE) for Atmel Microcontrollers
CPR	Cycles Per Revolution
GPIO	General Purpose Input/Output
IDE	Integrated Development Environment
PPR	Pulses Per Revolution
QDEC	Quadrature Decoder
TC	Timer Counter
UART	Universal Asynchronous Receiver Transmitter

2. Pre-requisites

The solution discussed in this application note require the following:

1. Atmel Studio 7 or later versions.
2. Atmel Software Framework version 3.31 or later versions.
3. SAM4E-EK Evaluation Kit (SAM4E-EK) with USB cable.
4. A motor that generates quadrature encoded signals.

Note: This application note provides an overview of QDEC functionality available in TC Peripheral. Refer the specific device datasheet for better understanding of the peripheral.

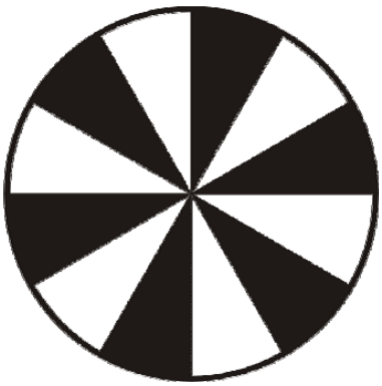
3. QDEC Theory

3.1. Quadrature Encoders

A quadrature encoder uses two signals to encode rotation and direction. The two quadrature encoder signals (QDPH0 and QDPH90) are characterized by two square waves phase shifted 90 degrees relative to each other. This can be implemented using a quadrature encoder disk or with a rate encoder disk with the sensors that are 90 degrees out of phase. The quadrature encoder disk is shown in Figure 3-1 Quadrature Encoder Disk and the rate encoder disk is shown in Figure 3-2 30 Degree Rate Encoder Disk.

Figure 3-1. Quadrature Encoder Disk

Figure 3-2. 30 Degree Rate Encoder Disk



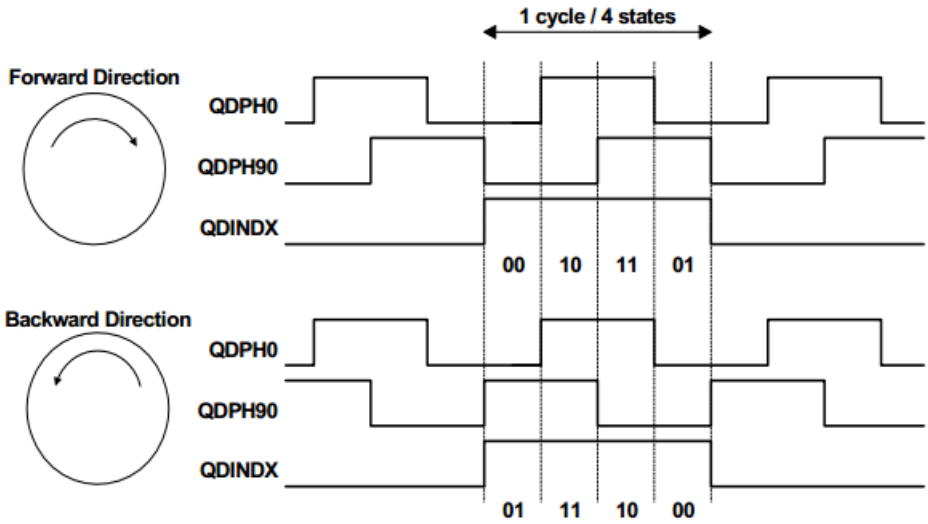
The rotational movement can be measured by counting the edges of the two waveforms. The phase relationship between the two square waves determines the direction of rotation.

Figure 3-3 Quadrature Signals from a Rotary Encoder shows typical quadrature signals from a rotary encoder. The signals *QDPH0* and *QDPH90* are the two quadrature signals. Figure 3-3 Quadrature Signals from a Rotary Encoder shows how the phase relationship determines the direction of rotation. When *QDPH0* leads *QDPH90*, the rotation is defined as positive or forward. When *QDPH90* leads *QDPH0*, the rotation is defined as negative or reverse. The concatenation of the two phase signals is called the quadrature state or the phase state.

The index signal shown in the figure as QDINDX is used for absolute positioning. This index signal can be high for a maximum of four states. If the index signal is high for four states as shown in Figure 3-3 Quadrature Signals from a Rotary Encoder, any state can be chosen as the index recognition state.

Quadrature encoders are commonly used as position sensors in motor applications. They are also found in other rotary sensors such as the ball tracker in computer mice.

Figure 3-3. Quadrature Signals from a Rotary Encoder



3.2. Quadrature Encoder Output Signals

Quadrature encoders have two or three output lines: Two-output encoders can provide information about the relative position for a rotary device. These two outputs have four (quad) states – based on which it has been named. Unless the initial rotary displacement is known, a two-output encoder can only be used to calculate relative movement, speed, and position. The absolute rotary displacement cannot be determined. However, having a third reference signal as an index signal to generate a pulse for every revolution can resolve this.

3.3. Quadrature Decoding

The event system has extensions that enables to decode a quadrature signal and use this as a source for a Timer/Counter.

The rotary displacement using a two-output encoder is shown in Figure 3-5 Timer/Counter Value without Index/Reset Signal. The absolute position can be determined using a three-output encoder as shown in Figure 3-4 Timer/Counter Value with Index/Reset Signal.

When the QDINDX signal occurs the Timer/Counter value will be reset if not equal to BOTTOM, and an error bit will be set (ERRIF in INTFLAGS-interrupt flag register to the Timer/Counter). This enables the system to detect or skip an error in the system or reset the counter for first pass.

The speed and acceleration can be calculated by calculating the rate of change in the Timer/Counter register. This is shown in Figure 3-6 Timer/Counter Value with Decreasing Speed (Backwards Rotation).

Figure 3-4. Timer/Counter Value with Index/Reset Signal

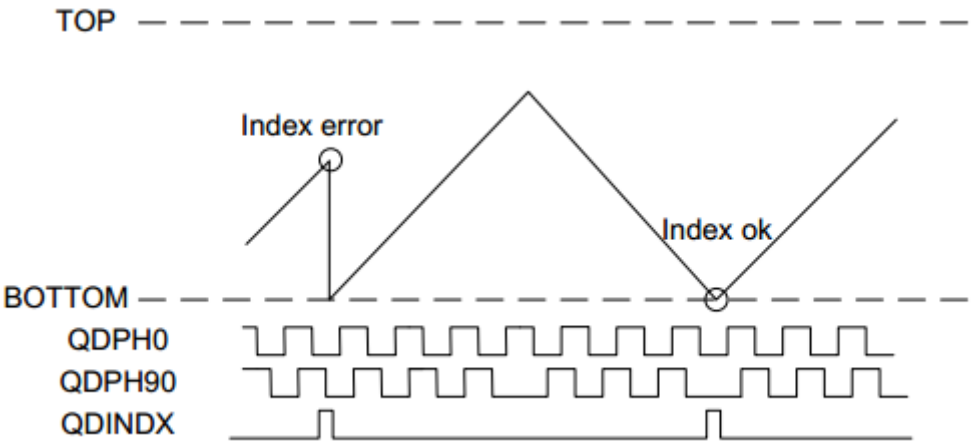


Figure 3-5. Timer/Counter Value without Index/Reset Signal

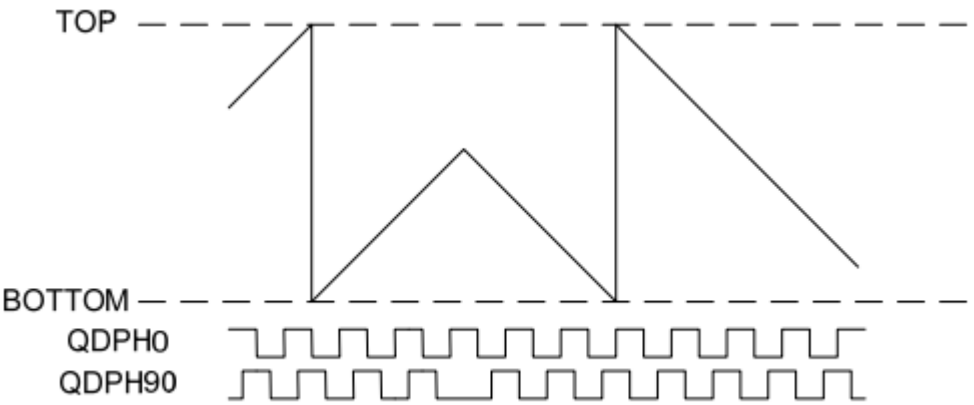
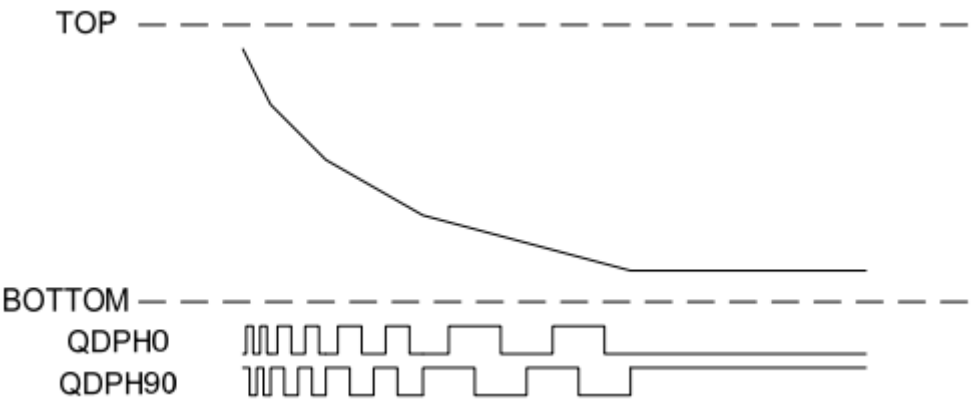


Figure 3-6. Timer/Counter Value with Decreasing Speed (Backwards Rotation)



4. QDEC Implementation in SAM3/4 MCUs

On Atmel SAM3/4 devices, the quadrature decoder (QDEC) is present in the Timer Counter. It is driven by TIOA0, TIOB0, and TIOB1 input pins of the Timer and drives the timer/counter of channel 0 and 1. Channel 2 can be used as a time base in case of speed measurement requirements.

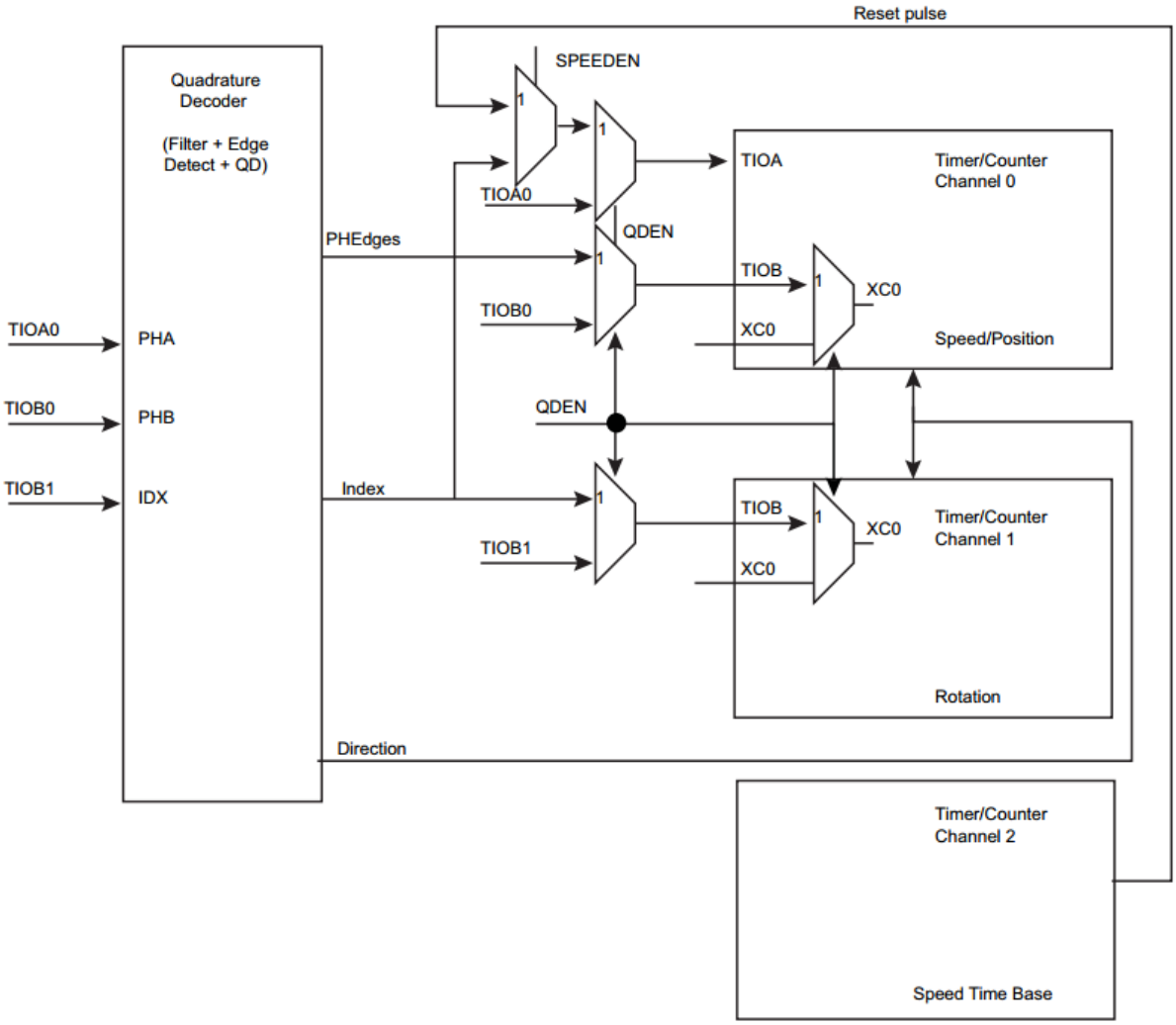
Either speed or position/revolution can be measured. Position channel 0 accumulates the edges of PHA and PHB input signals providing a high accuracy on motor position. Whereas channel 1 accumulates the index pulses of the sensor and therefore provides the number of rotations. The concatenation of both values provide a high level of precision in determining the position of the motion system.

In the speed mode, position cannot be measured but the revolution can be measured.

Inputs from the rotary sensor can be filtered prior to down-stream processing. Accommodation of input polarity, phase definition, and other factors are configurable. The interrupts can be generated during different events.

The TIOA0 and TIOB0 input pins must be driven by the two dedicated quadrature signals from a rotary sensor mounted on the shaft of the off-chip motor. A third reference signal from the rotary sensor can be processed through pin TIOB1. It is typically dedicated to be driven by an index signal if it is provided by the sensor. This signal is not required to decode the quadrature signals PHA and PHB. The field TCCLKS of TC_CMRx must be configured to select XC0 input such as 0x101. The field TC0XC0S has no effect as soon as the QDEC is enabled.

Figure 4-1. Predefined Connection of the Quadrature Decoder with Timer Counters



4.1. Input Pre-processing

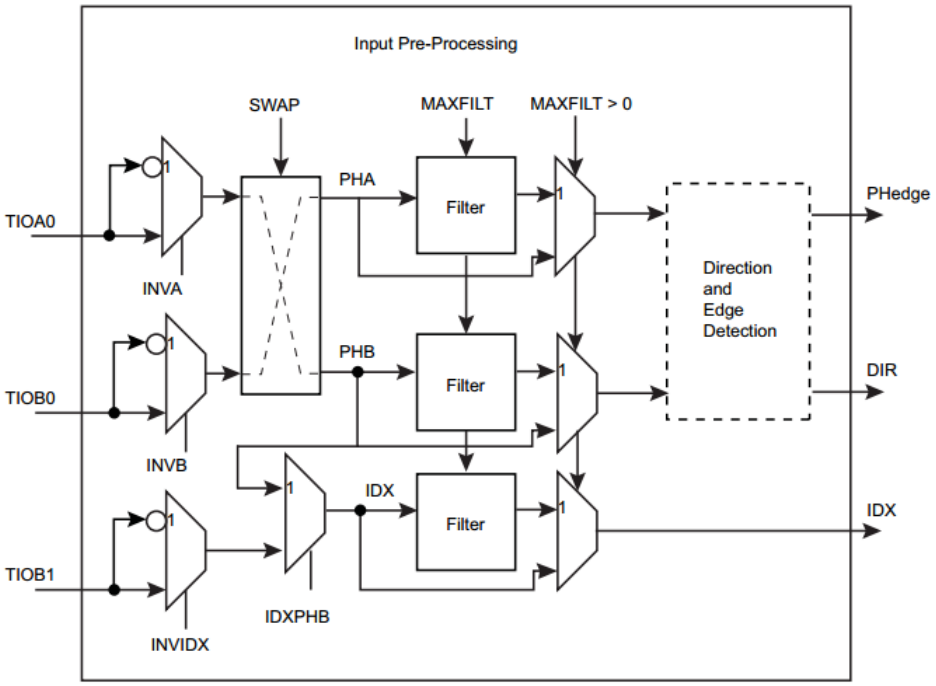
Input pre-processing take cares of rotary sensor factors such as polarities and phase definition followed by configurable digital filtering.

Each input can be negated. Swapping PHA and PHB can also be configured.

The MAXFILT field in the TC_BMR is used to configure a minimum duration for which the pulse is stated as valid.

When the filter is active, pulses with a duration lower than MAXFILT +1 × t peripheral clock ns are not passed to downstream logic.

Figure 4-2. Input Stage



The input filtering can efficiently remove spurious pulses generated by the presence of particulate contamination on the optical or magnetic disk of the rotary sensor.

Such spurious pulses can also occur in the environments with high levels of electro-magnetic interference. If vibration occurs, or even when rotation is fully stopped and the shaft of the motor is in such a position that the beginning of one of the reflective or magnetic bars on the rotary sensor disk is aligned with the light or magnetic (Hall) receiver cell of the rotary sensor. Any vibration can toggle between the PHA and PHB signals for a short duration.

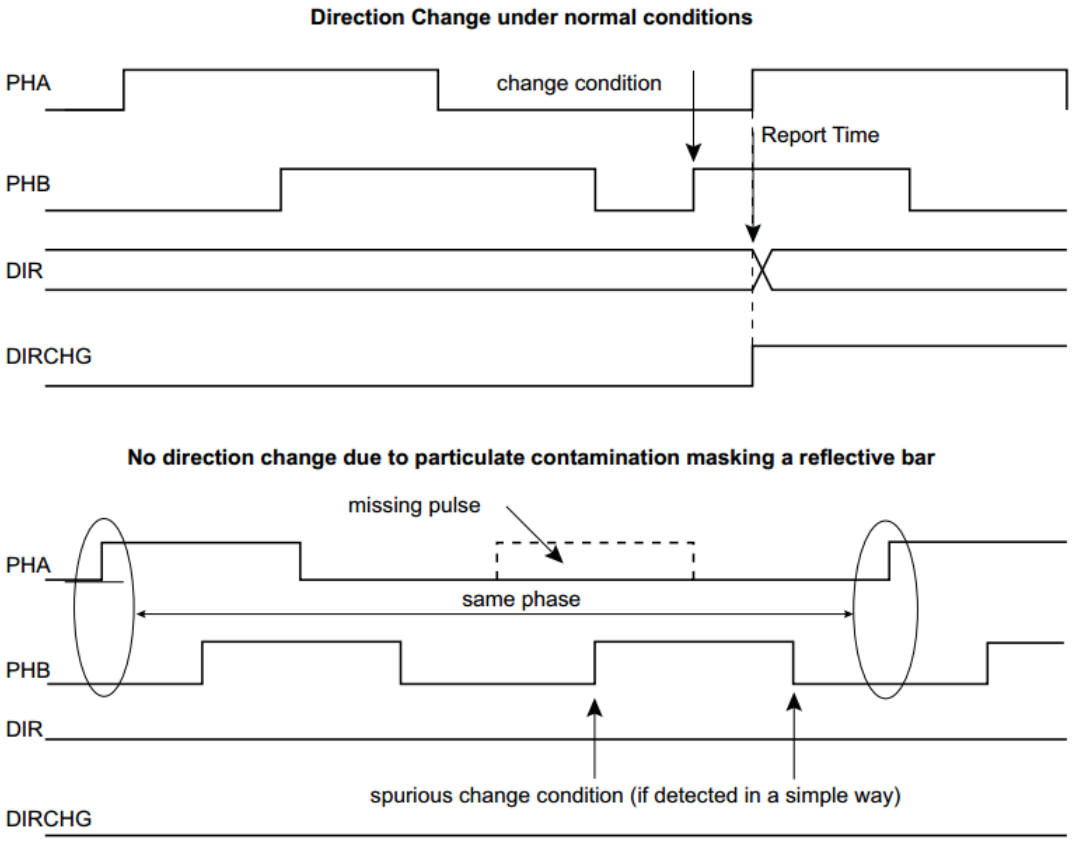
4.2. Direction Status and Change Detection

After filtering, the quadrature signals are analyzed to extract the rotation direction and edges of the two quadrature signals detected in order to be counted by timer/counter logic downstream.

The direction status can be directly read at any time in the TC_QISR. The polarity of the direction flag status depends on the configuration written in TC_BMR. The INVA, INVB, INVIDX, and SWAP bits modifies the polarity of DIR flag.

Any change in rotation direction is reported in the TC_QISR and can generate an interrupt. The direction change condition is reported as soon as two consecutive edges on a phase signal have sampled the same value on the other phase signal and there is an edge on the other signal. The two consecutive edges of one phase signal sampling the same value on other phase signal is not sufficient to declare a direction change. The particulate contamination may mask one or more reflective bars on the optical or magnetic disk of the sensor.

Figure 4-3. Rotation Change Detection



4.3. Position and Rotation Measurement

When the POSEN bit is set in the TC_BMR, the motor axis position is processed on channel 0 by means of the PHA and PHB edge detections. The number of motor revolutions are recorded on channel 1 if the IDX signal is provided on the TIOB1 input. The position measurement can be read in the TC_CV0 register and the rotation measurement can be read in the TC_CV1 register.

The channel 0 and 1 must be configured in Capture mode (TC_CMR0.WAVE = 0). 'Rising edge' must be selected as the External Trigger Edge (TC_CMR.ETRGEDG = 0x01) and 'TIOA' must be selected as the External Trigger (TC_CMR.ABETRG = 0x1).

In parallel, the number of edges are accumulated on timer/counter channel 0 and can be read on the TC_CV0 register. Therefore, the accurate position can be read on both TC_CV registers and concatenated to form a 32-bit word. The timer/counter channel 0 is cleared for each increment of IDX count value. Depending on the quadrature signals, the direction is decoded and allows to count up or down in timer/counter channels 0 and 1. The direction status is reported on TC_QISR.

4.4. Speed Measurement

When SPEEDEN is set in the TC_BMR, the speed measure is enabled on channel 0. A time base must be defined on channel 2 by writing the TC_RC2 period register. The Channel 2 must be configured in Waveform mode (WAVE bit set) in TC_CMR2. The WAVSEL field must be defined with 0x10 to clear the counter by comparison and matching with TC_RC value. The Field ACPC must be defined at 0x11 to toggle TIOA output.

When QDEN and SPEEDEN are set, this time base is automatically feedback to TIOA of channel 0.

The Channel 0 must be configured in Capture mode (WAVE = 0 in TC_CMR0). The ABETRG bit of TC_CMR0 must be configured at 1 to select TIOA as a trigger for this channel.

To clear the counter on a rising edge of the TIOA signal, EDGTRG must be set to 0x01 and field LDRA must be set accordingly to 0x01. To load TC_RA0 at the same time as the counter is cleared (LDRB must be set to 0x01) LDRA must be set accordingly to 0x01. At the end of each time base period the differentiation required for the speed calculation is performed.

The process must be started by configuring the bits CLKEN and SWTRG in the TC_CCR. The speed can be read from the field RA in TC_RA0. The Channel 1 can still be used to count the number of revolutions of the motor.

5. Quadrature Decoder Example Application

The example covered in this section covers the features such as position measurement, rotation measurement, and speed measurement.

5.1. Basic Configuration

Basic configuration example covers pin initialization, clock initialization, and USART initialization functions. The following functions are used for setting the basic configurations.

- `sysclk_init()`
- `board_init()`
- `configure_console()`

The detailed information about these functions are explained in the upcoming sections.

5.1.1. Sysclk and Board Initialization

The `syclk_init()` is an ASF function used to configure the generic clocks and clock sources as per the settings in the `conf_clocks.h` file. The `board_init()` initializes the board hardware of SAM4E-EK Evaluation Kit.

5.1.2. UART Initialization

In this application UART0 is used to communicate with the PC terminal application to display the QDEC results. The `configure_console()` function initializes the UART0 module connected to the SAM4E-EK board serial connector. It configures the corresponding UART pins and the studio serial initializations for the standard library APIs such as scanf and printf.

5.2. QDEC Configurations

The `configure_tc_qdec()` function initializes the basic QDEC configurations such as QDEC mode, QDEC I/O pins initialization. The mode configuration includes configuring TC Channel 0 as speed/ Position mode, Channel 1 as Rotation mode, and Channel 2 as base timer if speed mode is selected. The files `conf_example.h` has the user configurations macros, which defines the each QDEC instance enable/disable, mode, TC channel for QDEC instance, and I/O pins assignment. The following block diagram explains the usage of TC for the QDEC functionality for the different modes.

Figure 5-1. TC1 (QDEC1) Position and Rotation Measurement Mode

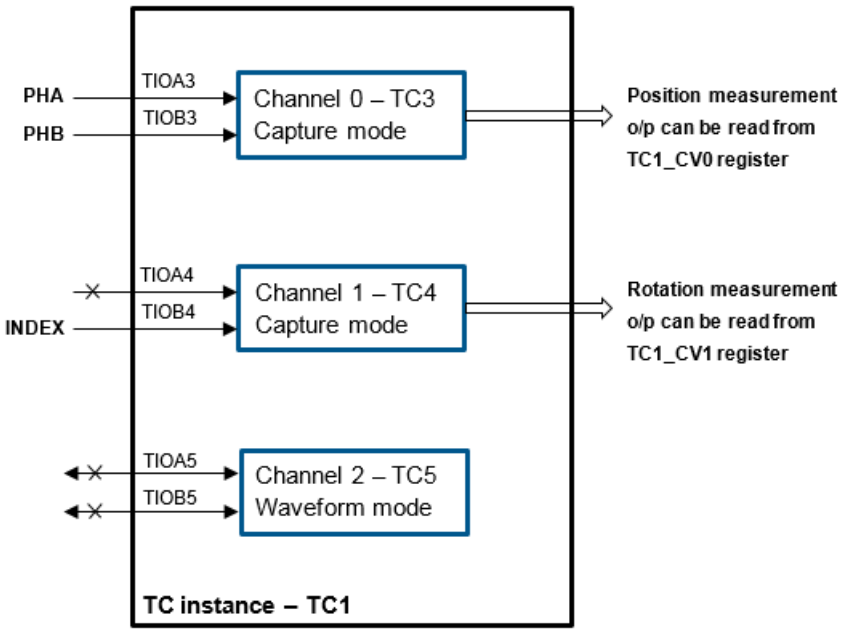
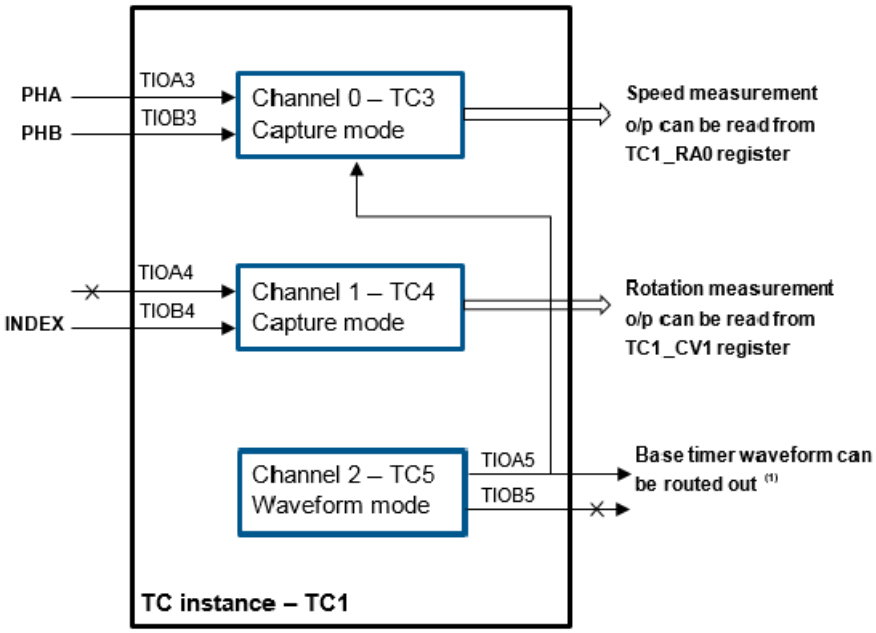


Figure 5-2. TC1 (QDEC1) Speed and Rotation Measurement Mode



Note: 1. An additional option is available in this example software to verify the base timer output. In this example, the test mode is selected using the following defines in `src\conf_example.h`.

```
#define TC_QDEC1_ACTIVATION             ENABLE
#define TC_QDEC1_MODE                   TC_QDEC_MODE_TEST_BASETIMER
```

5.2.1. QDEC Channel

Position measurement uses channel 0, Rotation measurement uses the channel 1, and Speed measurement uses channel 0 and channel 2. The following table summarizes the QDEC channels used in this example.

Table 5-1. TC Channel for QDEC Modes

Mode	Timer Channel	Timer Instance
Position	TC3	TC1(1)
Rotation	TC4	TC1(1)
Speed	TC3, TC5	TC1(1)

Note: 1. If QDEC1 is selected in the software, TC1 will be used.

For various devices, the detailed QDEC channel association with the TC peripheral are provided in FAQ: Multiplexed IO pins and Number of channels for Quadrature Decoder (QDEC) in SAM3/4 devices.

5.2.2. QDEC Input Signals

The quadrature decoder logic is driven by three inputs from the sensor (PHA, PHB, and Index). As mentioned below, those signals must be connected to the input pins such as TIOA3, TIOB3, and TIOB4 of the TC instance. If QDEC1 is selected in the software, TC1 will be used.

- PHA > TIOA3 (TCn channel 0)
- PHB > TIOB3 (TCn channel 0)
- Index > TIOB4 (TCn channel 1)

Note:

1. Index signal is only used for measuring Rotation. It is opitonal for the user to feed this signal for other measurements.
2. In this example application, Rotation measurement is configured by default in both Speed/Position mode. To measure the Rotation, connect the Index signal to the respective TC I/O pin.

For various devices, the detailed QDEC channel association with the TC peripheral are provided in FAQ: Multiplexed IO pins and Number of channels for Quadrature Decoder (QDEC) in SAM3/4 devices.

5.2.3. QDEC Direction Status and Change Detection

In this example, direction status directly reads from the TC_QISR.DIR bit. It will be printed with the QDEC results in terminal window using the function `tc_qdec_getresult()` and `tc_qdec_printresults()`. Where TC_QISR.DIR bit value is ‘0’ for clock wise direction and '1' for counter clock wise.

Note:

1. The polarity of the direction flag status depends on the configuration written in TC_BMR. The INVA, INVB, INVIDX, and SWAP bits modifies the polarity of DIR flag.
2. The direction change detection is disabled when QDTRANS is set in the TC_BMR. In this case, the DIR flag report must not be used.

5.2.4. QDEC Positon and Rotation Measurement

The following steps are used to configure the TC peripheral QDEC functionality for the Position and Rotation measurement mode. The TC1 (QDEC1) is used in this example, and functions used to configure this mode are explained.

Note: In this example, this mode is selected using the defines in `src\conf_example.h`.

```
#define TC_QDEC1_ACTIVATION           ENABLE
#define TC_QDEC1_MODE                 TC_QDEC_MODE_POSITION
```

1. Configure the I/O pins for the QDEC inputs using the function `config_tc_qdec_iopins()`.
2. Enable the TC channel 0, and channel 1 peripheral clocks using the function `sysclk_enable_peripheral_clock()`.
3. Initialize the TC channel 0 for the QDEC Position mode using the function `tc_init()`. Where,
   - Channel 0 is configured in Capture mode (TC_CMR1.WAVE = 0)
   - Field TCCLKS of TC_CMRx is configured to select XC0 input (i.e., 0x101)
   - 'Rising edge' selected as the External Trigger Edge (TC_CMR1.ETRGEDG = 0x01)
   - 'TIOA' selected as the External Trigger (TC_CMR1.ABETRG = 0x01)
4. Set the TC block mode configuration for QDEC position mode using the function `tc_set_block_mode()`. Where,
   - TC QDEC is enabled (TC_BMR.QDEN=1)
   - Position mode enabled (TC_BMR.POSEN=1)
   - QDEC edges is configured to detect on both edges of PHA and PHB 4x decoding (TC_BMR.EDGPHA=1)
   - Enabling input filler if needed (TC_BMR.MAXFILT=value)
5. Start the TC channel 0 (used for position mode), and channel 1 (used for Rotation mode) using the function `tc_start()`.
6. QDEC Position measurement results are read and printed in the terminal window using the function `tc_qdec_getresult()` and `tc_qdec_printresults()`. Where Position measurement result read in the TC_CV0 register and the Rotation measurement result read in the TC_CV1 register.

In this mode, the number of edges are accumulated on timer/counter channel 0 and read on the TC_CV0 register. Therefore, the accurate Position result read on both TC_CV registers and concatenated to form a 32-bit word.

The timer/counter channel 0 is cleared (TC_CV0) for each increment of IDX count value. Depending on the quadrature signals, the direction is decoded and allows to count up or down in timer/counter channels 0 (TC_CV0) and channel 1 (TC_CV1).

5.2.5. QDEC Speed Measurement Mode

The following steps are used to configure the TC peripheral QDEC functionality for the Speed measurement mode. The TC1 (QDEC1) is used in this example, and functions used to configure this mode are explained.

Note: In this example this mode is selected using the defines in `src\conf_example.h`.

```
#define TC_QDEC1_ACTIVATION           ENABLE
#define TC_QDEC1_MODE                 TC_QDEC_MODE_SPEED
```

1. Configure the I/O pins for the QDEC inputs using the function `config_tc_qdec_iopins()`.
2. Enable the TC channel 0, channel 1, and channel 2 peripheral clocks using the function `sysclk_enable_peripheral_clock()`.
3. Initialize the TC channel 0 for the QDEC Speed mode using the function `tc_init()`. Where,
   - Channel 0 is configured in Capture mode (TC_CMR1.WAVE = 0)
   - Field TCCLKS of TC_CMRx is configured to select XC0 input (i.e., 0x101)
   - Set EDGTRG 0x01, to clear the counter on a rising edge of the TIOA signal (TC_CMR1.ETRGEDG = 0x01)

- Set LDRA accordingly to 0x01, to load TC_RA0 at the same time as the counter is cleared (TC_CMR1.LDRA = 0x01)
- 'TIOA' selected as the External Trigger (TC_CMR1.ABETRG = 0x01)

4. Initialize the TC channel 2 as base timer for the QDEC Speed mode using the function `tc_init()`. Where,
   - Channel 2 is configured in Waveform mode (TC_CMR1.WAVE = 1)
   - Field TCCLKS of TC_CMRx is configured to select TIMER_CLOCK4 (i.e., 0x011)
   - Set WAVSEL 0x10, UP mode with automatic trigger on RC Compare (TC_CMR1.WAVSEL = 0x10)
   - Select the RC Compare Effect on TIOA as toggle (TC_CMR1.ACPC = 0x11)

   Note: When QDEN and SPEEDEN are set, this time base has automatically feedback to TIOA of channel 0.
5. Find the RC value for the base timer on TC channel 2 and load the value in RC using the function `tc_write_rc()`. In this example base timer is configured for 4ms. Since the waveform mode is configured as toggle on RC compare match, the base timer waveform toggles at 4ms interval. Every 8ms, we get a rising edge that is connected internally on TIOA. It clears channel 0 counter value at every 8ms base time. This will give the speed measurement value and the user must convert it to the real speed unit (exp.: RPM). The detailed speed conversion example is discussed in the upcoming sections.
6. Set the TC block mode configuration for QDEC position mode using the function `tc_set_block_mode()`. Where,
   - TC QDEC is enabled (TC_BMR.QDEN=1)
   - Speed mode enabled (TC_BMR.SPEEDEN=1)
   - QDEC edges is configured to detect on both edges of PHA and PHB 4x decoding (TC_BMR.EDGPHA=1)
   - Enabling input filler if needed (TC_BMR.MAXFILT=value)
7. Start the TC channel 0 (used for Speed mode), channel 1 (used for Rotation mode), and channel 2 (used for Speed mode) using the function `tc_start()`.
8. QDEC Speed measurement results are read and printed in the terminal window using the function `tc_qdec_getresult()` and `tc_qdec_printresults()`, respectively. Where Speed measurement result is read from the TC_RA0 register.

In this mode, the Speed result is read from TC_RA0 denotes the total number of edges counted within the time base period (8ms).

The number of edges are accumulated on timer/counter in channel 0. This counter clears at every time base period (8ms is used in this example). At the same time, Speed result is loaded to TC_RA0.

The Timer/counter channel 1 can still be used to count the number of revolutions of the motor.

6. Example Speed Calculation

The following configurations are used to measure the speed of the motor with TC1 (QDEC1) in the example application.

- PHA is TIOA3
- PHB is TIOB3
- IDX is TIOB4

Channel 0: Capture mode. Used to enter wave of motor.

Channel 1: Capture mode. Used to count number of revolutions.

Channel 2: Waveform mode. Time base to count.

To generate the 4ms time base waveform, the required hex value is loaded in RC2 register.

Input QDEC signal specification:

We have used a motor with 360 CPR (cycles per revolution) QDEC Encoder (CPR value is taken from QDEC encoder specification).

PPR = 4 * CPR = 1440 (PPR – pulses per revolution)

CPR = PPR / 4 = 360 (CPR – cycles per revolution)

The following PHA and PHB QDEC signals waveforms are observed when the motor speed is 3600 RPM.

Figure 6-1. PHA Signal Measurements

Figure 6-2. PHB Signal Measurements

Table 6-1. Observed Results

Input Signal (in RPM)	Manually Calculated Count (in PPR)	Measured Value (from RA0)	Deviation
520	100	97	-3
2500	480	468	-12
3500	576	566	-10
3600	691	672	-19

Note: In this exercise, the speed of the motor is not very accurate which in turn affects the speed results read from RA0.

The manually calculated count is derived based on the following calculation.

```
Manually calculated count (in PPR)   = (PPR/time per revolution) * base time.

Time per revolution                  = 1000 / (RPM/60) ms
                                     = 1000 / (3600/60)
                                     = 16.66 ms

Manually calculated count (in PPR)   = (PPR/time per revolution) * base time.
                                     = (1440/16.66) * 8
                                     = 691

Time per cycle (from calculation)    = Time per revolution / (PPR/4)
                                     = 16.66ms / (1440/4)
                                     = 0.046277 ms(1)

Time per cycle (Measured using QDEC) = Base time / (RA0 value per revolution/4)
                                     = 8ms / (672/4)
                                     = 0.046190 ms(1)
```

Note: 1. Approximately 46.27µs which is PHA/PHB time period for one pulse is shown in Figure 6-1 PHA Signal Measurements and Figure 6-2 PHB Signal Measurements.

7. Limitation and Scalability

In speed mode, the results are accumulated within the time base (8ms). Depending on the user application and the requirement, the user must modify the time base with respect to their speed range.

For example, if the application requires low speed measurement, the time base can be increased to get the fine results. For higher speed measurement, the user must choose the lesser time base so that the number of pulses counted, does not overflow beyond the channel 0 counter (32-bit). If this is not taken care, the results may be inaccurate.

8. Execution of Application

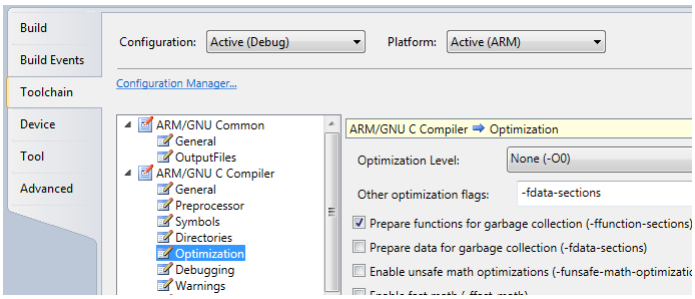
The firmware corresponding to this application note is available in the Atmel Software Framework and it can be imported from Atmel Studio as well.

Note: This chapter assumes that the application setup meets the requirements mentioned in Pre-requisites of this application note.

The following steps explains the execution of the application:

1. Import the example in Atmel Studio from **File** → **New** → **Example Project** → TC_QDEC_Example.
2. Select the appropriate macro definition in the `src\conf_example.h` file based on the execution mode needed.
3. Go to **Build** → **Build Solution** to compile the project.
4. After the project is compiled successfully, go to **Tools** → **Device Programming Window**.
5. Select appropriate tool, device and interface type, and click **Apply** to connect to the kit.
6. Check Device Signature and Target Voltage to ensure proper connection.
7. Go to **Memories** Tab. Browse the `*.hex/elf` file location and click **Program** to flash the device.
8. To debug the code, right click on the project in the solution explorer window → Go to **Project Properties**.
9. Go to **Tools** tab. Select appropriate tool as **Debugger/programmer** with appropriate debugging Interface.
10. Ensure that the optimization level is set as **None (-O0)** to utilize maximum debugging. Go to Atmel Studio → **Project** → **Project properties** → **Toolchain** → **ARM®/GNU C Compiler** → **Optimization** → O**ptimization Level** as shown in Figure 8-1 Set Optimization Level to change the Optimization Level.

Figure 8-1. Set Optimization Level



11. Go to **Debug** → **Start Debugging and Break** to debug the code and click **Start without debugging** to continue programming without debugging.
12. Open the terminal window with the baud rate settings specified in the example main file `src\tc_qdec_example.c`.
13. Based on the selected QDEC mode, the results will be printed in the terminal window.

9. References

1. SAM4E Series Datasheet: http://www.atmel.com/Images/Atmel-11157-32-bit-Cortex-M4-Microcontroller-SAM4E16-SAM4E8_Datasheet.pdf
2. Atmel Studio: The latest version of Atmel Studio can be downloaded from http://www.atmel.com/tools/atmelstudio.aspx
3. Hardware Tools User Guide: SAM4E-EK Evaluation Kit (SAM4E-EK) User Guide and Schematics from http://www.atmel.com/tools/SAM4E-EK.aspx?tab=documents
4. Online Tools User Guide: Online help for each tool is available at the link http://www.atmel.com/webdoc/
5. Atmel Software Framework (ASF):
   - Standalone ASF can be down-loadable from http://www.atmel.com/tools/avrsoftwareframework.aspx
   - ASF user guide can be downloaded from http://asf.atmel.com/docs/latest/
6. Atmel AT07898: SAM3/4S/4L/4E/4N/4CM/4C/G Timer Counter (TC) Driver: http://www.atmel.com/Images/Atmel-42301-SAM3-4S-4L-4E-4N-4CM-4C-G-Timer-Counter-TC-Driver_ApplicationNote_AT07898.pdf
7. FAQ: Multiplexed I/O pins and Number of channels for Quadrature Decoder (QDEC) in SAM3/4 devices: http://atmel.force.com/support/articles/en_US/FAQ/Multiplexed-IO-pins-and-Number-of-channels-for-Quadrature-Decoder-QDEC-in-SAM3-4-devices
8. FAQ: Can we use the TC QDEC channel 2 as position or Rotation mode when it is not used for speed mode: http://atmel.force.com/support/articles/en_US/FAQ/Can-we-use-the-TC-QDEC-channel-2-as-position-or-Rotation-mode-when-its-is-not-used-for-speed-mode

10. Revision History

Doc Rev.	Date	Comments
42706A	04/2016	Initial document release.

Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA **T:** (+1)(408) 441.0311 **F:** (+1)(408) 436.4200 | **www.atmel.com**