AN2434

Interfacing Quadrature Encoder using CCL with TCA and TCB

Introduction

Author: Kristian Saxrud Bekken, Microchip Technology Inc.

Incremental quadrature encoders are used in a great number of applications across many disciplines as they provide a low-cost way of measuring motion in systems with moving parts. Some typical examples include measuring the position of a physical control wheel or measuring the rotor angle and velocity in an electrical motor.

Some AVR® microcontrollers, like the AVR XMEGA® E5, include dedicated quadrature decoding functionality, but the decoding can also be accomplished by utilizing some of the core independent peripherals available on smaller, less feature rich controllers. This application note describes how to decode and keep track of quadrature encoded signals from an incremental position sensor using an AVR by combining Core Independent Peripherals (CIPs) such as Configurable Custom Logic (CCL), Event System (EVSYS), 16-bit Timer/Counter Type A (TCA), and 16-bit Timer/Counter Type B (TCB).

The described setup uses eight I/O pins. If implemented on an ATtiny1617 using the code provided with this application note, the device can decode quadrature pulses with a frequency of up to 2.5 MHz.

Features

- Setup for decoding quadrature encoded incremental position data with use of CIPs:
  - Configurable Custom Logic (CCL)
  - Event System (EVSYS)
  - 16-bit Timer/Counter Type A (TCA)
  - 16-bit Timer/Counter Type B (TCB)
- Up to 16-bit resolution
- Device independent
- Minimal RAM usage
- Minimal core usage

Table of Contents

1. Relevant Devices

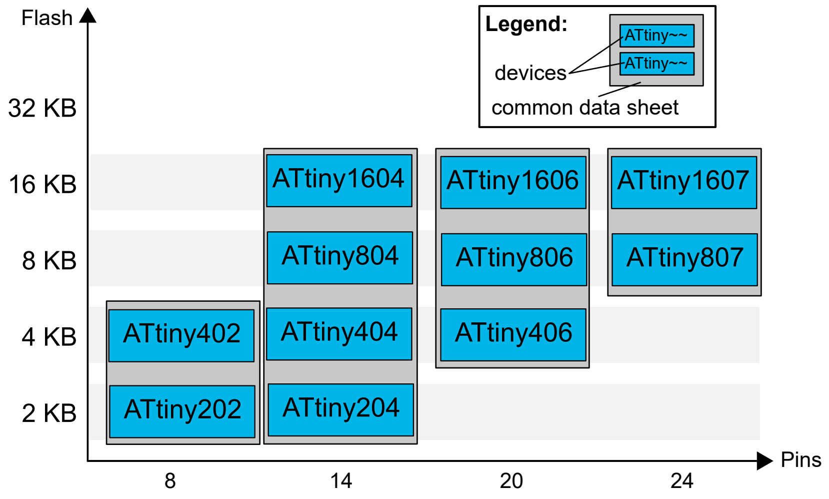
This chapter lists the relevant devices for this document.

1.1 tinyAVR® 0-series

The figure below shows the tinyAVR® 0-series, laying out pin count variants and memory sizes:

- Vertical migration is possible without code modification, as these devices are fully pin- and feature compatible.
- Horizontal migration to the left reduces the pin count and therefore, the available features.

Figure 1-1. tinyAVR® 0-series Overview



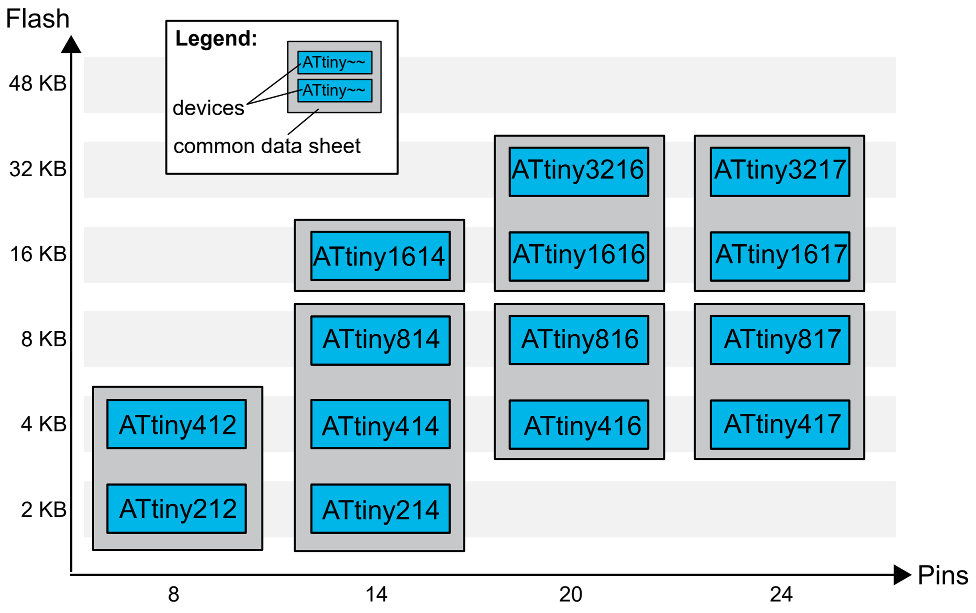
Devices with different Flash memory size typically also have different SRAM and EEPROM.

1.2 tinyAVR® 1-series

The following figure shows the tinyAVR 1-series devices, laying out pin count variants and memory sizes:

- Vertical migration upwards is possible without code modification, as these devices are pin compatible and provide the same or more features. Downward migration may require code modification due to fewer available instances of some peripherals.
- Horizontal migration to the left reduces the pin count and, therefore, the available features.

Figure 1-2. tinyAVR® 1-series Overview



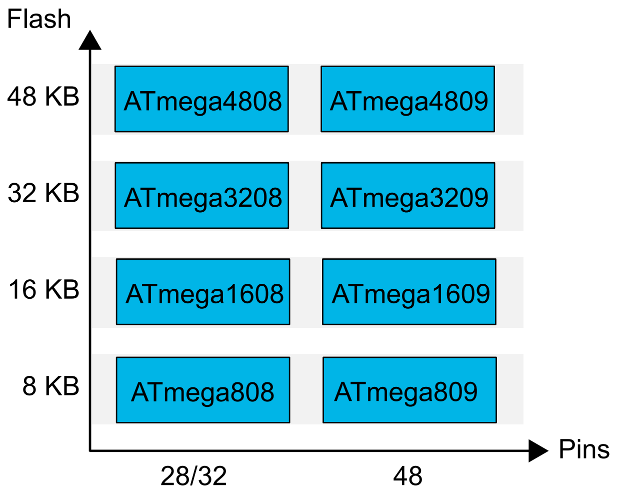
Devices with different Flash memory size typically also have different SRAM and EEPROM.

1.3 megaAVR® 0-series

The figure below shows the megaAVR 0-series devices, laying out pin count variants and memory sizes:

- Vertical migration is possible without code modification, as these devices are fully pin and feature compatible.
- Horizontal migration to the left reduces the pin count and, therefore, the available features.

Figure 1-3. megaAVR® 0-series Overview



Devices with different Flash memory size typically also have different SRAM and EEPROM.

2. Quadrature Encoding of Signals from Incremental Position Sensors

Incremental position sensors, commonly known as incremental encoders, measure motion by detecting discrete steps of linear or angular displacement. Information about the actual position or angle is not detected, only the fact that a motion corresponding to a known discrete step has taken place. The steps are equally spaced within the physical motion envelope of the sensor. A microcontroller or other external circuitry is required to keep track of the detected increments in order to obtain information about actual position and speed. This is what mainly differentiates incremental encoders from absolute encoders, which can provide correct position data directly after start-up without the need for motion. Incremental encoders, on the other hand, need to be put to a known position when power is cycled in order to provide correct position data.

Due to its low cost, the incremental rotary encoder is widely used in many industries and applications, and it is available with resolutions ranging from a few counts to thousands of counts per revolution. Linear incremental encoders are also available.

The simplest form of incremental encoders have a single output line on which a signal toggles to indicate each position increment. This signal does not provide information about the direction of movement. When direction sensing is required, a dual-channel quadrature encoded output is commonly used.
The two output channels of an incremental quadrature encoder typically carry two square wave signals, A and B, that are 90 degrees out of phase. The signal levels are toggled once for each incremental step in an alternating way. When combining the signals to a two-bit value, the values of the four consecutive steps within one encoder cycle provides a gray coded signal containing both the position increments and the direction of motion, as shown in the table and the two figures below. One encoder cycle consists of four encoder counts. Since the signal is gray coded, only a single bit is toggled for each increment. This can facilitate signal error detection and more robust decoding.

Table 2-1. Quadrature Signal Coding

Positive Direction		Negative Direction	
A	B	A	B
0	0	0	0
1	0	0	1
1	1	1	1
0	1	1	0

Figure 2-1. Positive Direction Waveforms

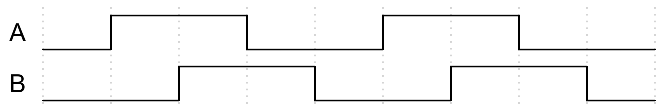
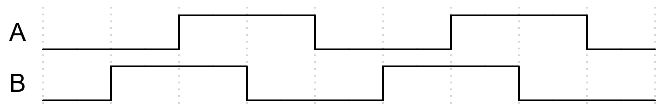


Figure 2-2. Negative Direction Waveforms



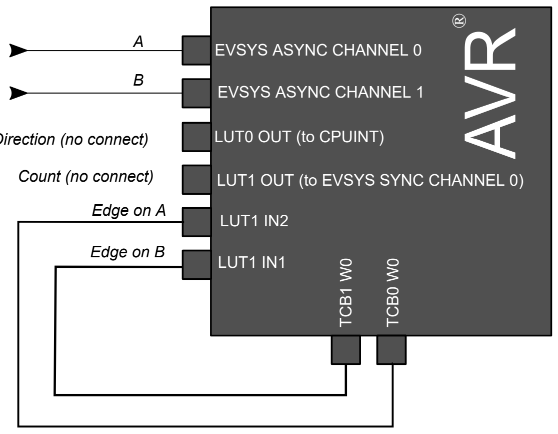
Many incremental encoders also provide an additional third signal called an index pulse or Z-pulse. This signal only goes high on one specific position in the motion envelope of the sensor, which means once per mechanical revolution for rotary encoders. This can be used as an accurate reference point for position calibration.

3. Required Device Resources

The resources required to implement the described setup are:

- One 16-bit Timer/Counter Type A (TCA)
- Two 16-bit Timer/Counter Type B (TCB)
- One Configurable Custom Logic (CCL) with two available LUTs
- Four event channels (three asynchronous and one synchronous)
- Eight I/O pins that can be set up as outlined in the figure below

Figure 3-1. Required I/O Pins for Implementation



4. Implementation Overview

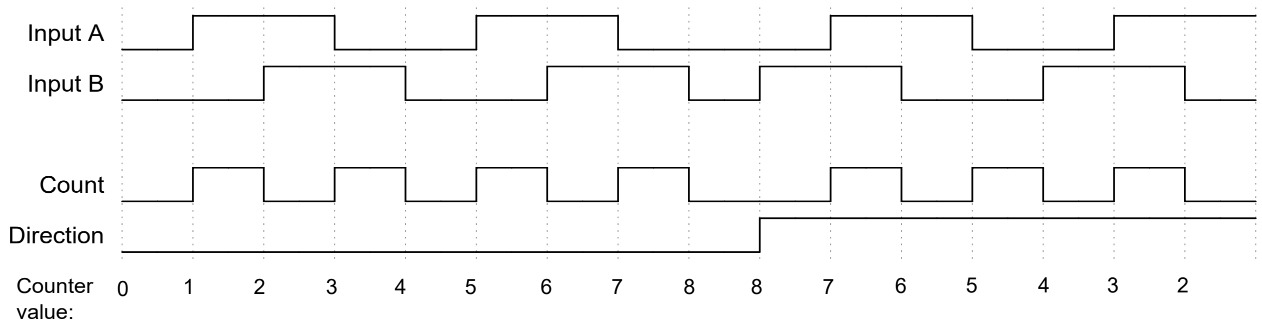
This application note focuses on the generation of two binary signals from the quadrature pulses created by an incremental encoder. One signal for counting incremental pulses and one for keeping track of counting direction. Many of the operations can be offloaded from the CPU using Core Independent Peripherals like the Configurable Custom Logic (CCL), Event System (EVSYS), and Timer/Counters. Since both the count signal and the direction signal can be generated from logic expressions that take the quadrature signals as inputs, the idea is to resolve these expressions using the Configurable Custom Logic (CCL) module. The two signals can then be passed via the event system to the 16-bit Timer/Counter Type A (TCA), which have the ability to count events. The counter value will be increased or decreased according to the count signal and the configured direction. The registers of the Timer/Counter hold the counter value and counting direction to be mapped to a physical position.

The count signal must indicate which of the two quadrature inputs contains the previous edge. It shall be low if the previous edge was on A, and high if it was on B. This gives a single increment or decrement of the counter value on each edge of the count signal. Any consecutive edge on the same channel indicates a direction change and must not produce an edge on the count signal. The edge detection is performed without CPU intervention by passing each input via the event system to a 16-bit Timer/Counter Type B (TCB) with dual event edge detection enabled.

The direction signal represents the physical direction of motion as indicated by the last edge on either of the two quadrature input signals.

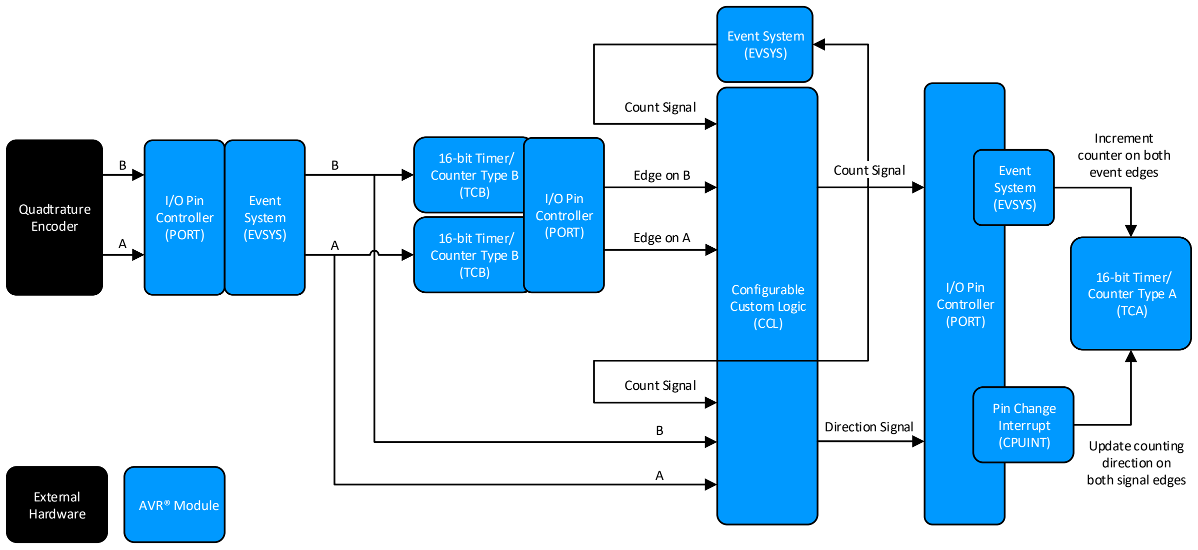
The first figure below outlines the desired count signal, direction signal, and counter value, along with the input signals.

Figure 4-1. Desired Waveforms and Counter Value for the Decoding Scheme



By moving as many operations as possible to the peripherals instead of executing them on the CPU, the available FLASH, SRAM, and CPU cycles are increased - leaving more headroom for implementing additional functionality. The figure below outlines how the peripherals and the flow of signals between them are set up.

Figure 4-2. Modules and Signals Application Overview


Event System (EVSYS)
Count Signal
Quadtrature Encoder
B
A
I/O Pin Controller (PORT)
Event System (EVSYS)
B
A
16-bit Timer/ Counter Type B (TCB)
16-bit Timer/ Counter Type B (TCB)
I/O Pin Controller (PORT)
Edge on B
Edge on A
Configurable Custom Logic (CCL)
Count Signal
I/O Pin Controller (PORT)
Event System (EVSYS)
Increment counter on both event edges
16-bit Timer/ Counter Type A (TCA)
Count Signal
Direction Signal
Pin Change Interrupt (CPUINT)
Update counting direction on both signal edges
B
A
External Hardware
AVR® Module

Since the single event input for the 16-bit Timer/Counter Type A (TCA) is used for counting incremental pulses, a pin change interrupt is used for updating the timer count direction.

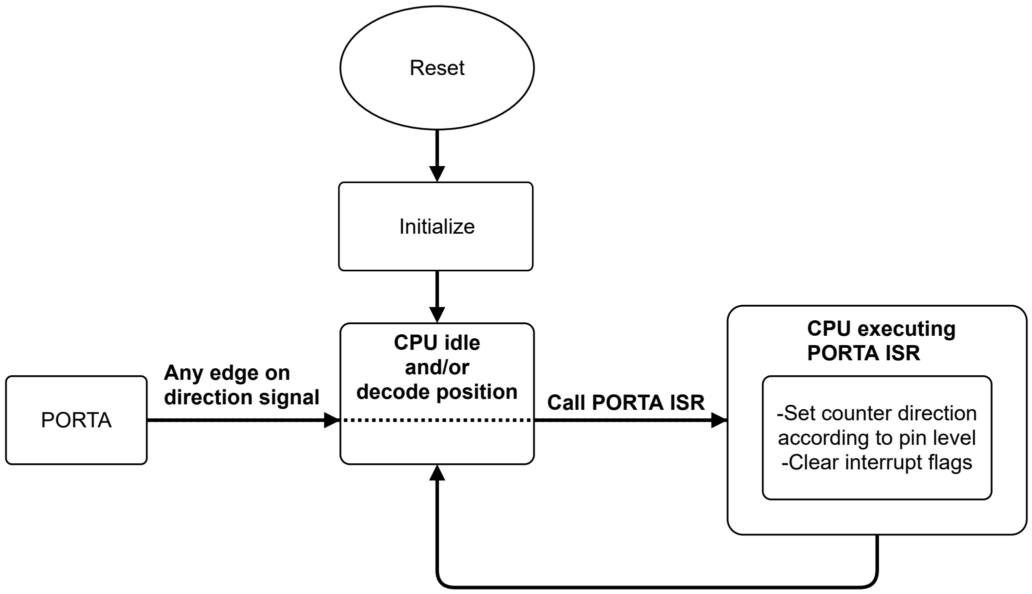
Descriptions of how the different modules are configured are provided in the following sections.

5. CPU Flow and Position Decoding

In the code accompanying this application note, the CPU is utilized only for mapping the TCA data to an actual position value, in addition to updating the direction setting of the TCA upon direction changes. The position mapping is done in a basic polled fashion from the main loop, while the direction updates are executed by a pin change interrupt. The reason for not including a more elegant way of mapping the position value is to keep the example focused on the actual quadrature decoding setup. Additionally, the rate and fashion of this mapping is considered to be highly specific for the final application, and thus this part of the code most likely needs to be modified.

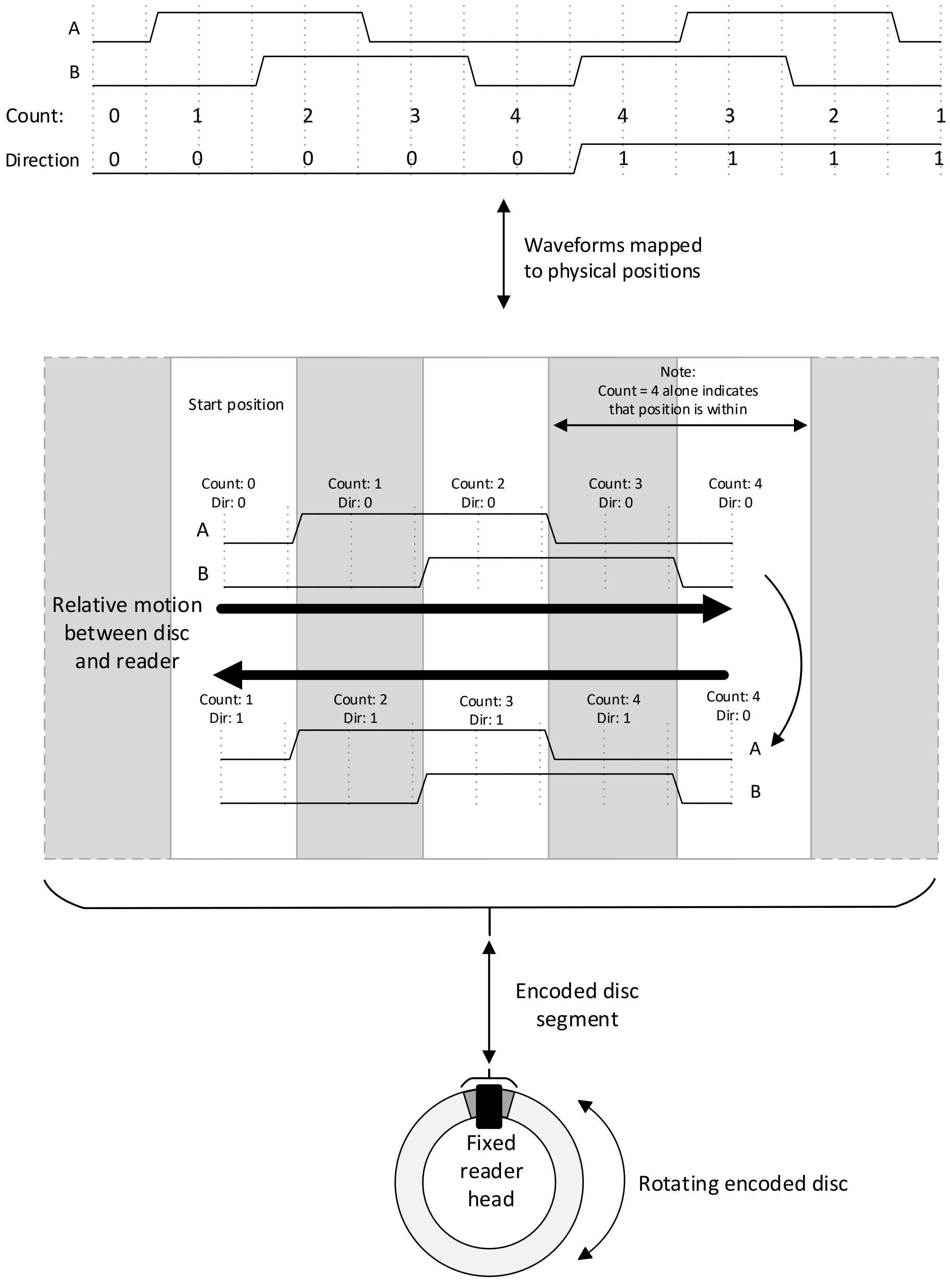
The figure below illustrates how the CPU is utilized.

Figure 5-1. CPU Execution Flow Outline


Reset
Initialize
CPU idle and/or decode position
PORTA
Any edge on direction signal
Call PORTA ISR
CPU executing PORTA ISR
-Set counter direction according to pin level
-Clear interrupt flags

This decoding scheme is based on only incrementing or decrementing the counter value on quadrature pulses not corresponding to a direction change. For this reason the counter value alone indicates only that the physical position is in the interval corresponding to [counter -1, counter +1]. This gives an uncertainty, or precision, of two times the distance between two adjacent encoder pulses. By inspecting also the direction value, the uncertainty is reduced by half as it will identify in which of the two intervals, [counter - 1, counter] or [counter, counter + 1], the actual position resides. The figure below outlines the decoding principle with a mapping between the quadrature waveforms and the relative motion between the reader and the encoder disc of a rotary encoder.

Figure 5-2. Physical Position Decoding Example, Rotary Encoder Segment



In the code example, updating of the counter value and the direction bit are kept separated in the sense that only the direction bit is updated for each physical direction change. This minimizes the CPU cycles needed when the direction change interrupt is triggered. The value of the direction bit is, therefore, taken into account by subtracting it from the counter value before the counter value is mapped to its corresponding physical position.

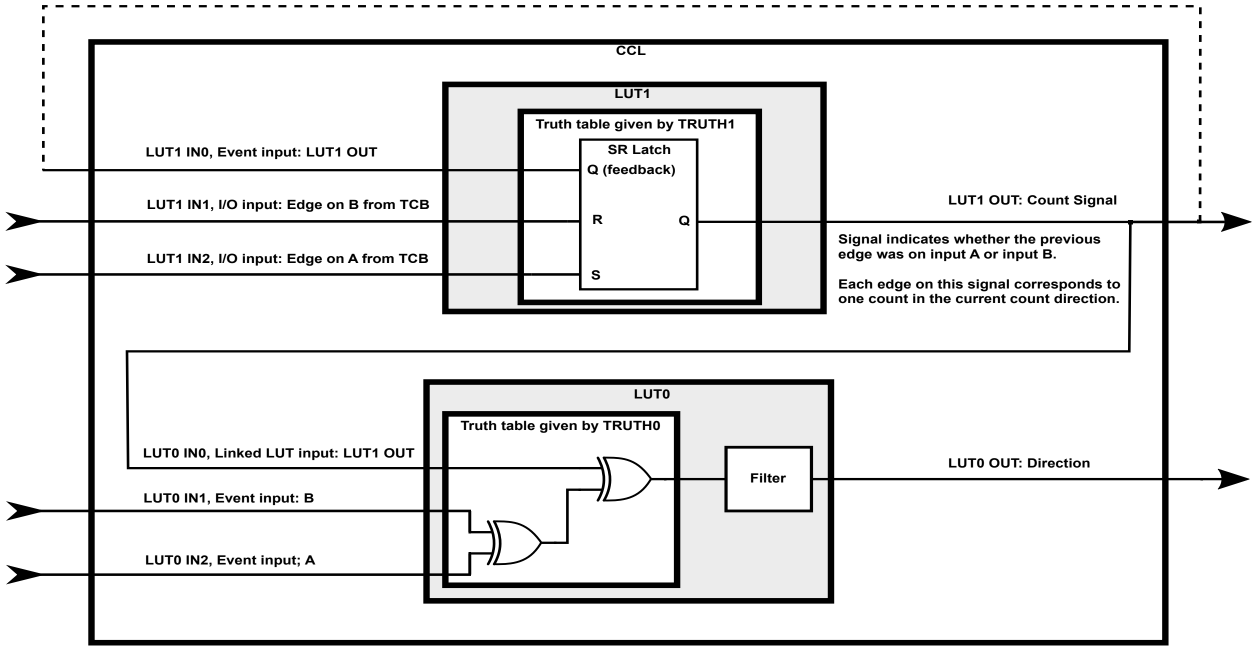
6. Configurable Custom Logic (CCL) Setup

The Configurable Custom Logic (CCL) module creates the count signal and the direction signal by acting as glue logic. The module contains a number of look-up tables (LUTs), which each can perform a logic operation based on its three inputs. The inputs are selectable from a large number of both internal and external signals.

For this application note two LUTs are utilized; one for each of the generated signals. The basic behavior of each LUT is given by its TRUTH register. The TRUTH register defines a truth table for the LUT that maps each logical combination of the three inputs to a true or false (high or low) output value. Each bit in the TRUTH register represents one of the eight possible input value combinations, and the value of the bit represents the LUT output value for that combination.

The figure below shows an overview of the Configurable Custom Logic (CCL) configuration for this application.

Figure 6-1. Configurable Custom Logic (CCL) Setup


CCL
LUT1
Truth table given by TRUTH1
SR Latch
Q (feedback)
R
Q
S
LUT1 IN0, Event input: LUT1 OUT
LUT1 IN1, I/O input: Edge on B from TCB
LUT1 IN2, I/O input: Edge on A from TCB
LUT1 OUT: Count Signal
Signal indicates whether the previous edge was on input A or input B.
Each edge on this signal corresponds to one count in the current count direction.
LUT0
Truth table given by TRUTH0
Filter
LUT0 IN0, Linked LUT input: LUT1 OUT
LUT0 IN1, Event input: B
LUT0 IN2, Event input; A
LUT0 OUT: Direction

LUT1 Setup

LUT1 is used to generate a signal, which indicates which of the quadrature input channels contain the last detected edge. Each edge of this signal corresponds to a quadrature count. This functionality is equivalent to an SR latch that takes the output of an edge detector connected to each quadrature signal as an input. The truth table for a generic active-high SR latch is included below.

Table 6-1. SR Latch Truth Table

S	R	OUT
0	0	Hold state (no change)
0	1	0 (Clear)
1	0	1 (Set)
1	1	Forbidden state

To accomplish this the setup is based on having the three following inputs for this LUT:

- The current output of the LUT
- Signal strobe indicating an edge on B
- Signal strobe indicating an edge on A

The LUT output is fed back to IN0 via the event system.

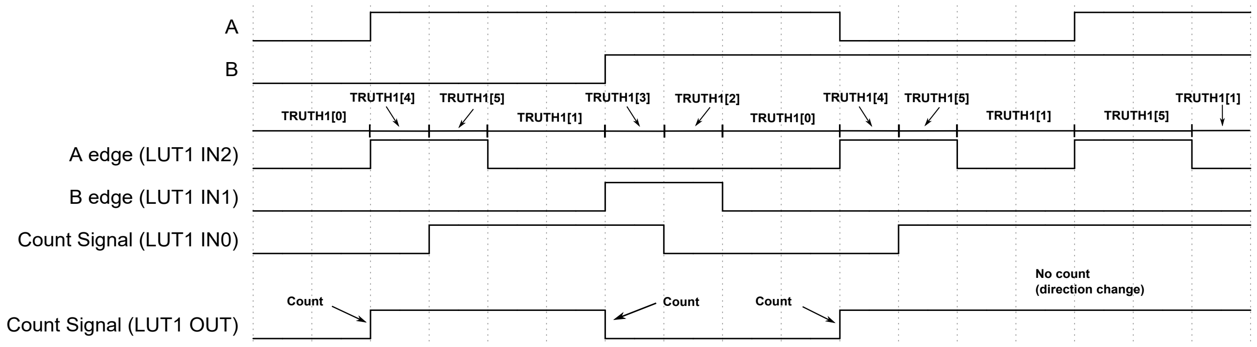
Implementing the SR latch functionality described above is done by setting up the TRUTH1 register as indicated in the table below, where "edge on A" acts as Set and "edge on B" acts as Clear. The bit combination '`0b00110010`' given by THRUTH1[0-7] in the OUT column, equivalent to the hexadecimal value 0x32, is the resulting 8-bit value to write to the TRUTH1 register.

Table 6-2. Truth Table for LUT1

IN[2]	IN[1]	IN[0]	OUT (TRUTH1)	Corresponding SR Latch Output
Edge on A	Edge on B	Count Signal Feedback	Count Signal Output	
0	0	0	0 (TRUTH1[0])	Hold state (no change)
0	0	1	1 (TRUTH1[1])	Hold state (no change)
0	1	0	0 (TRUTH1[2])	0 (Clear)
0	1	1	0 (TRUTH1[3])	0 (Clear)
1	0	0	1 (TRUTH1[4])	1 (Set)
1	0	1	1 (TRUTH1[5])	1 (Set)
1	1	0	0 (TRUTH1[6])	Forbidden state
1	1	1	0 (TRUTH1[7])	Forbidden state

The waveform below shows the output signal for an example sequence of input signals, as well as the corresponding row in the LUT1 truth table for each input value combination. Additionally, the actual A and B input signals are included. The main purpose of the figure is to show the mapping between the LUT inputs and outputs. Pulse lengths on LUT1 IN0 and IN1, as well as the delay from LUT1 OUT to LUT1 IN0, have therefore been exaggerated for clarity.

Figure 6-2. LUT1 Example Waveform



LUT0 Setup

LUT0 is used for generating a signal that indicates the direction of the quadrature pulses at any given time, indicating whether the total pulse count should be increasing or decreasing for each count. To accomplish this the setup is based on having the three following inputs for LUT0:

- Output from LUT1 indicating which channel contained the previous edge
- B
- A

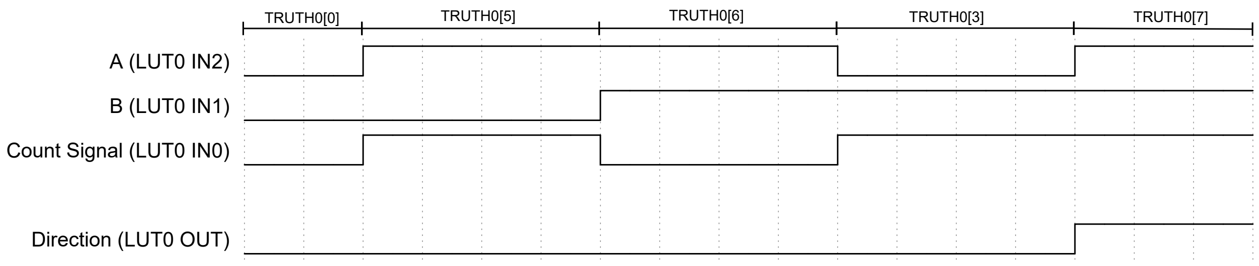
As illustrated for LUT0 in the figure included initially, the direction signal is created by performing a series of two XOR operations. The first takes the A and B signals as inputs, while the second takes the count signal and the output from the first XOR as inputs. In order to configure the LUT as described, its TRUTH register needs to be set up according to the table below. The bit combination 0b10010110 given by THRUTH0[0-7] in the OUT column, equivalent to the hexadecimal value 0x96, is the resulting 8-bit value to write to the TRUTH0 register.

Table 6-3. Truth Table for LUT0

IN[2]	IN[1]	IN[0]	OUT (TRUTH0)
A	**B**	**Count Signal from LUT1**	**Direction Signal**
0	0	0	0 (TRUTH0[0])
0	0	1	1 (TRUTH0[1])
0	1	0	1 (TRUTH0[2])
0	1	1	0 (TRUTH0[3])
1	0	0	1 (TRUTH0[4])
1	0	1	0 (TRUTH0[5])
1	1	0	0 (TRUTH0[6])
1	1	1	1 (TRUTH0[7])

The waveform below shows the output signal for an example sequence of input signals that includes a direction change, as well as the corresponding row in the LUT0 truth table for each input value combination.

Figure 6-3. LUT0 Example Waveform



There is a propagation delay from the A and B input signals to the count signal. Consequently, when a quadrature pulse resulting in a changed count signal is received, the change will arrive at LUT0 IN0 after the corresponding change on LUT0 IN1 or IN2. If the LUT output is not filtered, it will cause a glitch on LUT0 OUT, which will invoke the direction update interrupt unnecessarily. The filter in LUT0 is enabled to mitigate the issue.

7. Setup of Timer/Counters

This chapter describes the setup of the two different timer modules used in this application note, the 16-bit Timer/Counter Type B (TCB) and the 16-bit Timer/Counter Type A (TCA).

16-bit Timer/Counter Type A (TCA) Setup

The 16-bit Timer/Counter Type A (TCA) is used for keeping track of the accumulated incremental movement as indicated by the quadrature pulses on A and B.

The 16-bit Timer/Counter Type A (TCA) has one event input and is configured to count both edges of the event signal. It also has a configurable count direction. By setting the generated count signal as the event input and updating the counter direction according to the generated direction signal, the timer/counter can keep track of the incremental encoder pulses requiring CPU only to update the count direction.

Furthermore, the counter period is set up corresponding to the total number of encoder increments to be counted before the counter value is reset to zero. For applications like tracking the position of a control dial or shaft that always resides in a given angular interval, the size of the measurement window given in encoder counts could be less than or equal to 16 bits. In these cases the counter period should be set up so that zero and the period value represent the endpoints of the measurement window, including sufficient buffers to avoid counter underflow or overflow.

In rotary applications where the measurement window is unconstrained or larger than 16 bits, the counter period should be set to the encoder resolution minus one. Subtracting by one must be done to account for the zero position, so that the total number of unique positions equals the encoder resolution. For these applications the position is only tracked within each encoder revolution, thus if position tracking across multiple revolutions is required the accumulated number of revolutions must be counted as well.

The 16-bit Timer/Counter Type A (TCA) is used in Normal mode without any compare channels or waveform outputs.

16-bit Timer/Counter Type B (TCB) Setup

Two instances of 16-bit Timer/Counter Type B (TCB) are used to detect edges on the quadrature input signals A and B. Both are configured identically.

The 16-bit Timer/Counter Type B (TCB) has one event input and the ability to trigger actions on both event edges. The actions are given by the configured TCB mode. In Single-Shot mode, a single count sequence can be initiated upon detecting an edge on the input event. If the TCB waveform output is enabled, it will be held high until the counter reaches the compare value.

The TCB is configured as an edge detector that outputs a signal strobe of configurable length upon detection of both rising and falling edges on the event input. This is accomplished by enabling the following features:

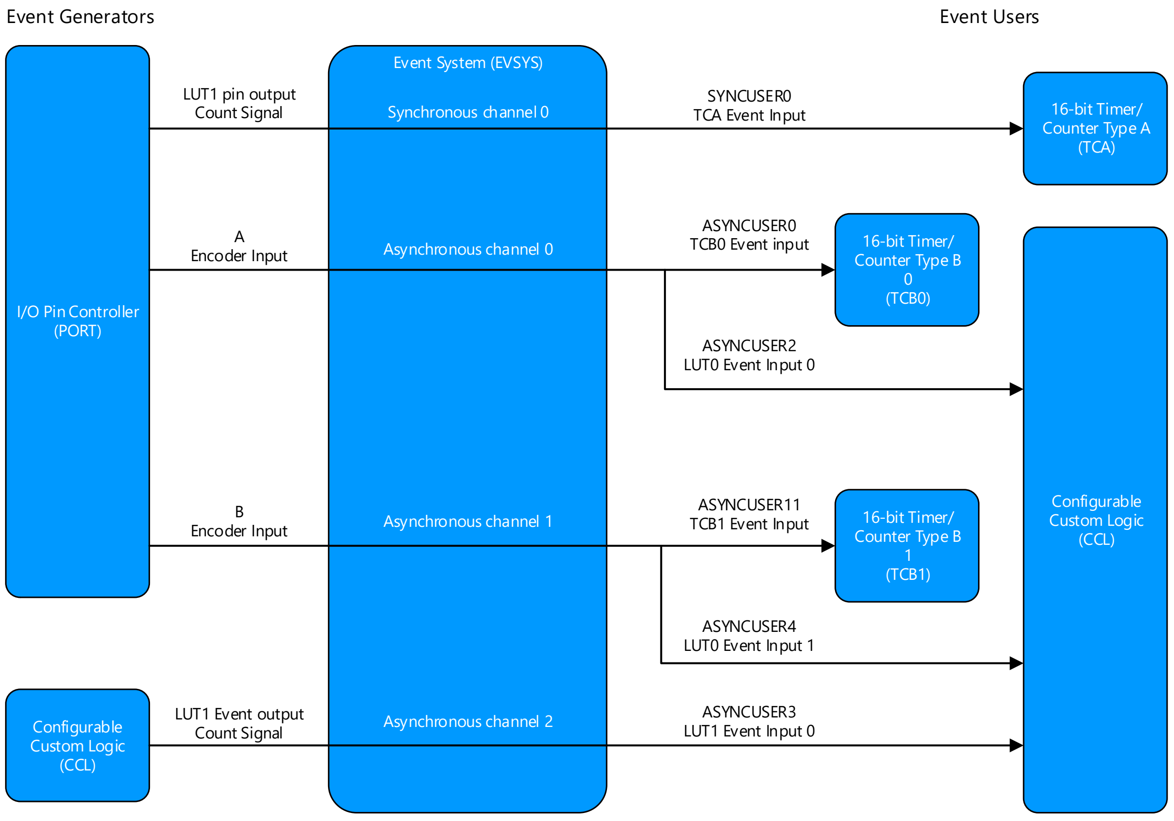
- Single-shot mode
- Event input with dual edge detection
- Asynchronous waveform output

The configuration is utilized by routing the quadrature inputs via the I/O Pin Controller (PORT) module and the Event System (EVSYS) to the event inputs of the two 16-bit Timer/Counter Type B (TCB) instances. The asynchronous waveform outputs are then routed to the Configurable Custom Logic (CCL) via their output pins.

8. Event System Setup

The Event System is used for routing signals internally between modules on the device. An overview of the Event System configuration used for this application note is shown in the figure below.

Figure 8-1. Event System Overview


Event Generators
Event Users
I/O Pin Controller (PORT)
Configurable Custom Logic (CCL)
Event System (EVSYS)
Synchronous channel 0
Asynchronous channel 0
Asynchronous channel 1
Asynchronous channel 2
LUT1 pin output Count Signal
A Encoder Input
B Encoder Input
LUT1 Event output Count Signal
SYNCUSER0 TCA Event Input
ASYNCUSER0 TCB0 Event input
ASYNCUSER2 LUT0 Event Input 0
ASYNCUSER11 TCB1 Event Input
ASYNCUSER4 LUT0 Event Input 1
ASYNCUSER3 LUT1 Event Input 0
16-bit Timer/ Counter Type A (TCA)
16-bit Timer/ Counter Type B 0 (TCB0)
16-bit Timer/ Counter Type B 1 (TCB1)
Configurable Custom Logic (CCL)

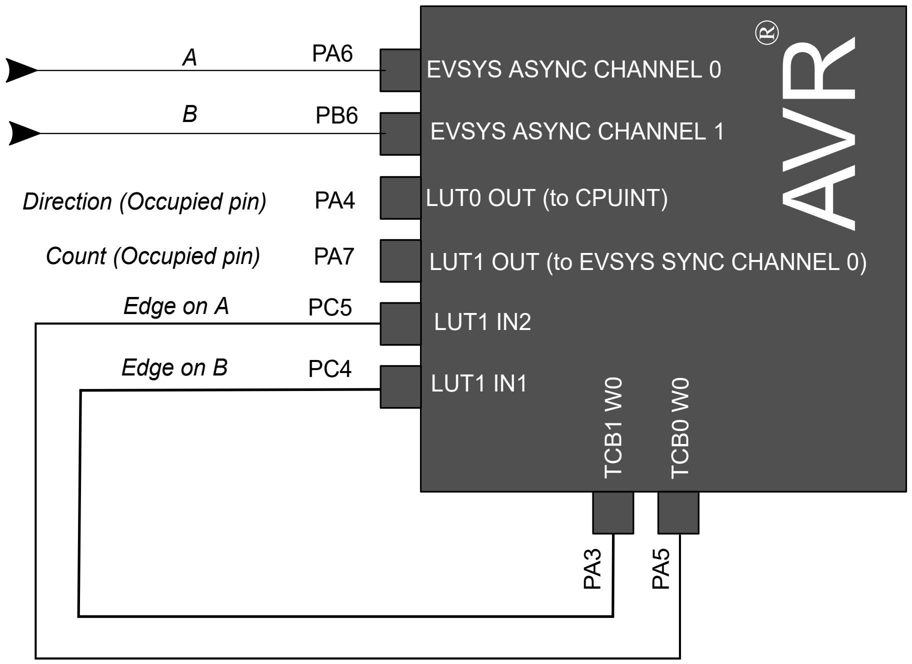
The two encoder inputs are each routed from the I/O Pin Controller (PORT) module to an asynchronous event channel. They are then passed on to the Configurable Custom Logic (CCL) and the two 16-bit Timer/Counter Type B (TCB) modules by configuring the event user registers to the correct event channels.

The output from LUT1 is used both as the event generator for asynchronous channel 2 directly, and as the generator for synchronous channel 0 indirectly via the pin. This is represented by the top and bottom arrows in the figure above. The reason is that the 16-bit Timer/Counter Type A (TCA) module accepts only synchronous events, while the Configurable Custom Logic (CCL) generates asynchronous events. Passing the asynchronous event via the I/O Pin Controller (PORT) module to generate synchronous events is, therefore, needed for this application.

9. Port Setup and Device Specific Details

A total of eight device pins are needed to realize the described setup. Four pins are configured as inputs, the other four as outputs. The figure below shows an example of specific pin allocation and their usage when implementing the application on an ATtiny1617 device. The accompanying code is written for the ATtiny1617, but with little effort, it can be ported to other devices that have the required peripherals.

Figure 9-1. Pin Connections Overview for an Implementation on ATtiny1617



The two encoder inputs are connected to the fully asynchronous pins PA6 and PB6 for minimal input latency. The other two configured pin inputs, PC4 and PC5, which are used by LUT1 on the device, are physically connected to the output pins of the TCB modules, PA3 and PA5. This is to minimize the latency from the TCB modules to LUT1 by taking advantage of the asynchronous pin output of the TCB.

The last two utilized pins are the output pins of the two LUTs, PA4 and PA7. In order to update the 16-bit Timer/Counter Type A (TCA) count direction, PA4 is also configured to sense both edges of the signal on the pin and initiate execution of the PORTA ISR, which handles the actual direction update. PA7 is used as the event generator for synchronous channel 0 in the event system due to the need of routing the count signal to the 16-bit Timer/Counter Type A (TCA).

10. Performance and Limitations

This chapter describes some experimental decoding performance figures as well as a set of limitations and considerations that have been identified for the described setup. This should be accounted for if the setup is to be utilized in a physical application.

Experimental Performance Figures

By stimulating a device with emulated quadrature pulses, the performance of the decoding scheme in terms of the maximum acceptable quadrature frequency have been explored experimentally using an ATtiny1617 running at a clock frequency of 20 MHz. These figures should be considered maximum levels for the tested device in a desktop setting with no considerable amount of noise.

For a unidirectional quadrature signal, the device could successfully count incoming pulses arriving at up to approximately 2.5 MHz. This corresponds to 75000 revolutions per minute for a rotary encoder providing 2000 counts per revolution.

In a situation where the encoder were to oscillate in such a way that one of the input signals toggles its level continuously, corresponding to a change of direction at every edge, the interrupt routine for updating the count direction would be initiated by each of these edges. The delay associated with starting and executing the interrupt reduces the obtainable decoding frequency. The maximum acceptable frequency of consecutive direction changes was found to be approximately 500 kHz.

For comparison, a purely interrupt-based decoding scheme was also tested. The routine decides whether to count up or down for each quadrature pulse, such that the decoding is unaffected by direction changes. The maximum decoding frequency for the interrupt-based solution was found to be approximately 220 kHz.

Two advantages of a CPU-based solution compared to the core independent approach is that software filters could be implemented in order to increase noise immunity, and it adds flexibility with regards to signal processing. This would then come at the cost of a lower maximum decoding frequency.

Considerations and Limitations

The main limitations that have been identified are based on the low degree of noise immunity and the usage of interrupts for updating the count direction of the timer/counter.

Noise suppression

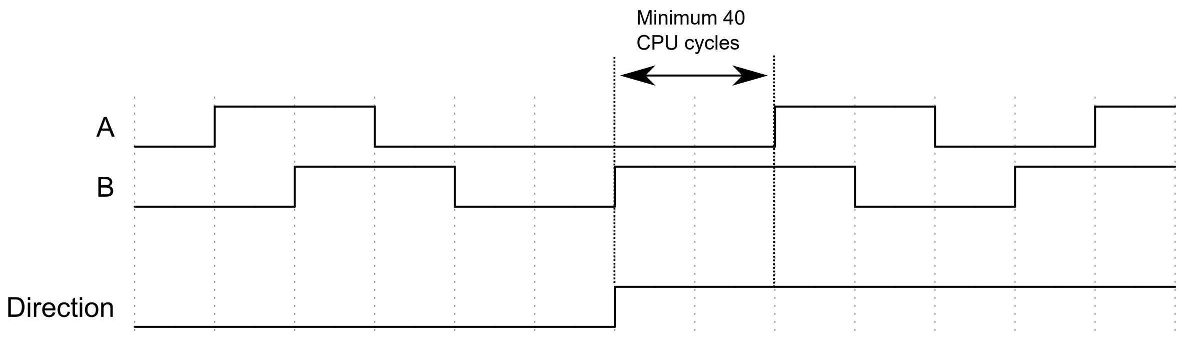
A low degree of internal signal filtering is used in this application in addition to edge detection on the input signals. Due to this, the decoding scheme is sensitive to noise and false spikes on the input lines. Specifically, false pulses lasting longer than one CPU clock cycle can be detected as quadrature counts or direction changes. The noise immunity will be increased by reducing the CPU clock frequency, which also reduces the decoding bandwidth accordingly. Another option is to add a level of external filtering or signal conditioning if necessary. Utilizing the index pulse signal from quadrature encoders that include this feature will reduce the functional impact if detection of false pulses can not be eliminated.

Interrupt-driven count direction update

Using an interrupt for updating the count direction of the 16-bit Timer/Counter Type A (TCA) has some functional implications. It causes the quadrature decoding to not be fully core independent as the CPU needs to execute an interrupt routine when a direction change is detected. The CPU cycles required for handling the interrupt makes the maximum frequency of direction changes lower than the maximum frequency of quadrature pulses from unidirectional motion.

By inspecting the instructions needed to handle the direction change interrupt, it has been found that it will take approximately 40 CPU cycles from the corresponding interrupt flag is raised until the interrupt routine is completed. This corresponds well with the description in the above section regarding performance figures, where the maximum frequency of consecutive direction changes was found to be approximately 500 kHz at a CPU frequency of 20 MHz. The main consequence is that the time between a direction change and the subsequent quadrature pulse should be 40 CPU cycles or more for the direction to stay correct. This is illustrated in the figure below:

Figure 10-1. Time Constraint After Direction Change



11. Further Development

The code accompanying this application note sets up the device according to the quadrature decoding scheme. It also shows the conversion of the count and direction values to a physical encoder angle in a polled fashion. Described below are some suggestions for additional features that could be added if required.

Suggestions for further improvements:

- Move conversion of physical position or angle to an RTC-interrupt to be executed according to application needs
- Add sleepwalking for reduced power consumption when no CPU is needed
- Add estimation of encoder speed. This can be done by inspecting the difference in encoder counts at known time intervals, for example, based on periodic interrupts from the RTC.
- Add support for index-pulse detection to increase robustness by setting the counter to a known value when the pulse is detected. Accuracy can also be increased by using the index-pulse as an accurate reference point.

12. Get Source Code from Atmel | START

The example code is available through Atmel | START, which is a web-based tool that enables configuration of application code through a Graphical User Interface (GUI). The code can be downloaded for both Atmel Studio and IAR Embedded Workbench® via the direct example code-link(s) below or the *BROWSE EXAMPLES* button on the Atmel | START front page.

Atmel | START web page: start.atmel.com

Example Code

- Interfacing Quadrature Encoder using CCL with TCA and TCB:
  - http://start.atmel.com/#example/Atmel:quadrature_decoding_using_ccl_with_tca_and_tcb:1.0.0::Application:Quadrature_Decoding_using_CCL_with_TCA_and_TCB:

Press *User guide* in Atmel | START for details and information about example projects. The *User guide* button can be found in the example browser, and by clicking the project name in the dashboard view within the Atmel | START project configurator.

Atmel Studio

Download the code as an .atzip file for Atmel Studio from the example browser in Atmel | START, by clicking *DOWNLOAD SELECTED EXAMPLE*. To download the file from within Atmel | START, click *EXPORT PROJECT* followed by *DOWNLOAD PACK*.

Double-click the downloaded .atzip file and the project will be imported to Atmel Studio 7.0.

IAR Embedded Workbench

For information on how to import the project in IAR Embedded Workbench, open the Atmel | START user guide, select *Using Atmel Start Output in External Tools*, and *IAR Embedded Workbench*. A link to the Atmel | START user guide can be found by clicking *About* from the Atmel | START front page or *Help And Support* within the project configurator, both located in the upper right corner of the page.

13. Terms and Abbreviations

Table 13-1. Terms and Abbreviations

Phrase/Abbreviation	Explanation
CPU	Central Processing Unit. Commonly used for the microcontroller core.
ISR	Interrupt Service Routine
Signal strobe	Short pulse or signal
CCL LUT	Configurable Custom Logic Look-up Table. The AVR CCL module contains LUTs.
Measurement window	The set of positions that the encoder shall measure
Gray code	A way of encoding a digital signal where only one bit in the signal value changes between two values in the signal sequence
Event	Internal signal on the device that can be routed directly between peripherals by the Event System
RTC	Real-Time Counter

14. Revision History

Doc Rev.	Date	Comments
C	10/2018	The chapter on Relevant Devices has been updated to include 8/16 KB megaAVR 0-series devices.
B	02/2018	The chapter on Relevant Devices has been updated to include tinyAVR 0-series and megaAVR 0-series
A	08/2017	Initial document release.

The Microchip Web Site

Microchip provides online support via our web site at http://www.microchip.com/. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user’s guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Customer Change Notification Service

Microchip’s customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip web site at http://www.microchip.com/. Under “Support”, click on “Customer Change Notification” and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or Field Application Engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the web site at: http://www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip’s Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.

ISBN: 978-1-5224-3651-5

Quality Management System Certified by DNV

ISO/TS 16949
Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/
support
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Austin, TX
Tel: 512-257-3370

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453
Tel: 317-536-2380

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608
Tel: 951-273-7800

Raleigh, NC
Tel: 919-844-7510

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110
Tel: 408-436-4270

Canada - Toronto
Tel: 905-695-1980
Fax: 905-695-2078

ASIA/PACIFIC

Australia - Sydney
Tel: 61-2-9868-6733

China - Beijing
Tel: 86-10-8569-7000

China - Chengdu
Tel: 86-28-8665-5511

China - Chongqing
Tel: 86-23-8980-9588

China - Dongguan
Tel: 86-769-8702-9880

China - Guangzhou
Tel: 86-20-8755-8029

China - Hangzhou
Tel: 86-571-8792-8115

China - Hong Kong SAR
Tel: 852-2943-5100

China - Nanjing
Tel: 86-25-8473-2460

China - Qingdao
Tel: 86-532-8502-7355

China - Shanghai
Tel: 86-21-3326-8000

China - Shenyang
Tel: 86-24-2334-2829

China - Shenzhen
Tel: 86-755-8864-2200

China - Suzhou
Tel: 86-186-6233-1526

China - Wuhan
Tel: 86-27-5980-5300

China - Xian
Tel: 86-29-8833-7252

China - Xiamen
Tel: 86-592-2388138

China - Zhuhai
Tel: 86-756-3210040

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444

India - New Delhi
Tel: 91-11-4160-8631

India - Pune
Tel: 91-20-4121-0141

Japan - Osaka
Tel: 81-6-6152-7160

Japan - Tokyo
Tel: 81-3-6880- 3770

Korea - Daegu
Tel: 82-53-744-4301

Korea - Seoul
Tel: 82-2-554-7200

Malaysia - Kuala Lumpur
Tel: 60-3-7651-7906

Malaysia - Penang
Tel: 60-4-227-8870

Philippines - Manila
Tel: 63-2-634-9065

Singapore
Tel: 65-6334-8870

Taiwan - Hsin Chu
Tel: 886-3-577-8366

Taiwan - Kaohsiung
Tel: 886-7-213-7830

Taiwan - Taipei
Tel: 886-2-2508-8600

Thailand - Bangkok
Tel: 66-2-694-1351

Vietnam - Ho Chi Minh
Tel: 84-28-5448-2100

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

Finland - Espoo
Tel: 358-9-4520-820

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Garching
Tel: 49-8931-9700

Germany - Haan
Tel: 49-2129-3766400

Germany - Heilbronn
Tel: 49-7131-67-3636

Germany - Karlsruhe
Tel: 49-721-625370

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Germany - Rosenheim
Tel: 49-8031-354-560

Israel - Ra'anana
Tel: 972-9-744-7705

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Italy - Padova
Tel: 39-049-7625286

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Norway - Trondheim
Tel: 47-72884388

Poland - Warsaw
Tel: 48-22-3325737

Romania - Bucharest
Tel: 40-21-407-87-50

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

Sweden - Gothenberg
Tel: 46-31-704-60-40

Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820