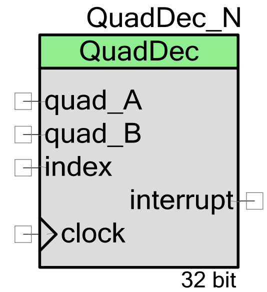
# Quadrature Decoder

**1.10**




## Features

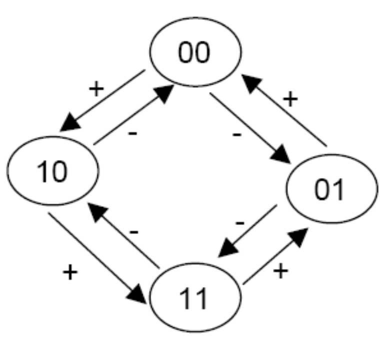
- Adjustable counter size: 8, 16 or 32-bits
- Counter resolution of 1x, 2x, or 4x the frequency of the A and B inputs, for more accurate determination of position or speed
- Optional Index input to determine absolute position
- Optional Glitch Filtering to reduce the impact or system generated noise on the input (s).

## General Description

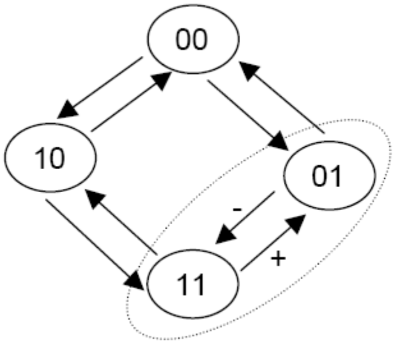
The Quadrature Decoder (QuadDec) component provides the ability to count transitions on a pair of digital signals. The signals are typically provided by a speed / position feedback system mounted on a motor or trackball.

The signals – typically called A and B - are positioned 90° degrees out-of-phase, which results in a “Gray” code output. A Gray code is a sequence where only one bit changes on each count; this is essential to avoid glitches. This allows direction to be detected as well as relative position. A third optional signal - index - is used as a reference to establish an absolute position once per rotation.

Quadrature phase signals are typically decoded with a state machine and an up/down counter. A conventional decoder has four states, corresponding to all possible values of the A and B inputs. The state transition diagram is shown below (same-state transitions are not depicted). State transitions marked with a “+” and “-“ indicate increment and decrement operations on the quadrature phase counter.




For each full cycle of the quadrature phase signal, the quadrature phase counter changes by four ticks. Lower resolution counters can also be used by implementing up/down operations on only a subset of the state transitions. A quarter-resolution decoder is shown below.




All inputs are sampled using a clock signal derived internally within the PSoC3/5 part.

### When to use a Quadrature Decoder

A Quadrature Decoder is used to decode input that senses the current position, velocity and direction of an object (mouse, trackball, robotic axles). Also it can be used for precision measurement of speed, acceleration and position of a motor's rotor and with rotary knobs, to determine user input.

## Input/Output Connections

This section describes the various input and output connections for the Quadrature Decoder component.

### quad_A – Input (Required)

One of the outputs of the quadrature encoder.

### quad_B – Input (Required)

One of the outputs of the quadrature encoder.

### Index – Input (Optional)

Detects a reference position for the quadrature encoder. If an index input is used then when A, B, and index are all zero the counter shall be reset to zero. Additional logic is typically added by the user to gate the index pulse. Index gating allows the counter to only be reset during one of may possible rotations. An example is a linear actuator that only resets the counter when at the far limit of travel as signaled by a mechanical limit switch whose output is ANDed with the Index pulse.

### clock – Input (Required)

Clock for sampling and glitch filtering the inputs. If glitch filtering is used then the filtered outputs shall not change until three successive samples of the input are the same value. For effective glitch filtering, the sample clock period should be greater than the maximum time during which glitching is expected to take place. A counter can be incremented / decremented at a resolution of 1x, 2x, or 4x the frequency of the A and B inputs.

### Interrupt – Output (Required)

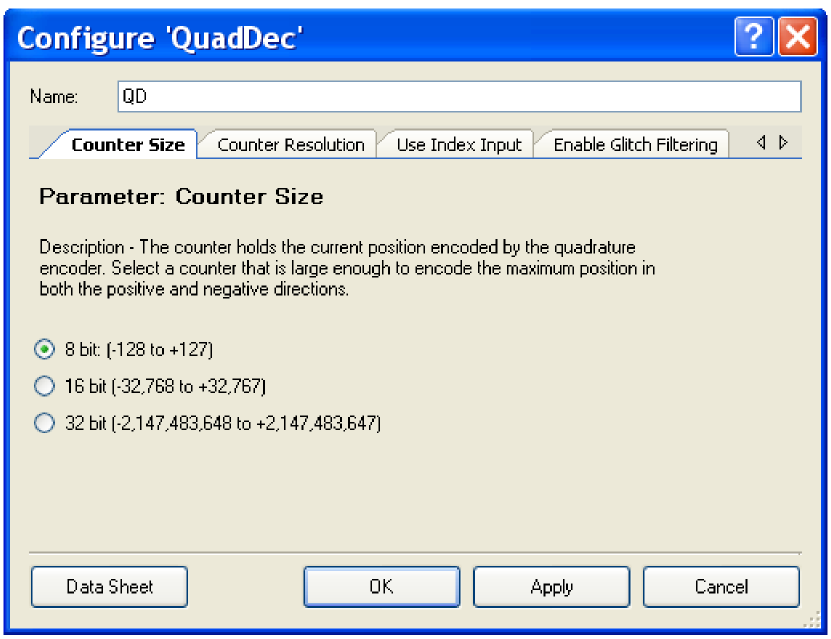
Interrupt on one or more of the following events:

- counter overflow and underflow.
- counter reset due to index input (if index is used).
- invalid state transition on the A and B inputs.

## Parameters and Setup

Drag a Quadrature Decoder component onto your design and double-click it to open the Configure dialog.

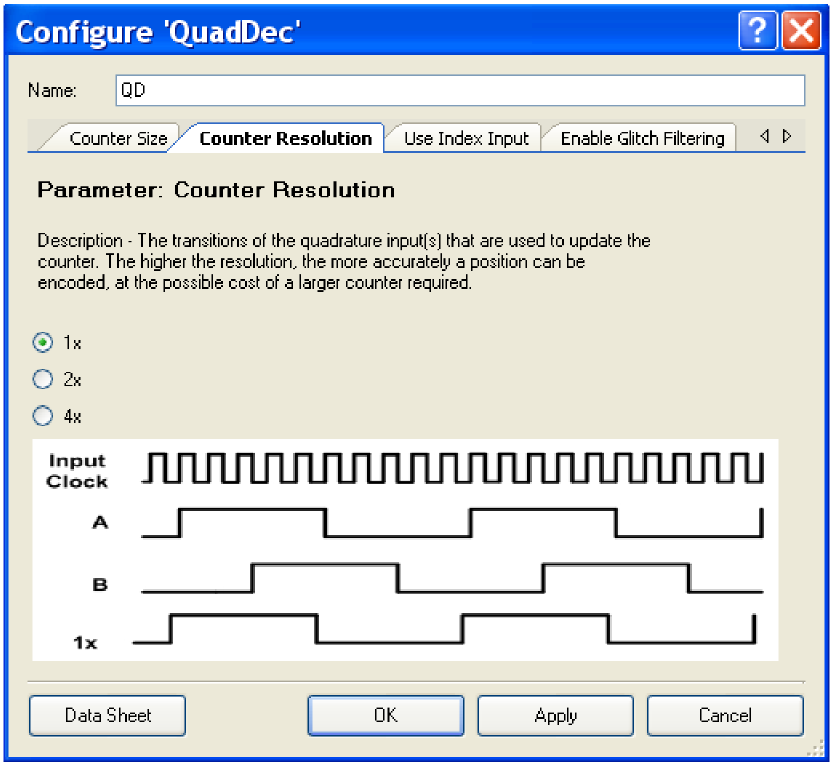
### Counter Size Tab



### Counter Size

Counter size, in bits. The counter holds the current position encoded by the quadrature encoder. Select a counter that is large enough to encode the maximum position in both the positive and negative directions. The 32-bit counter implements the lower 16-bits in the hardware counter and the upper 16-bits in software to reduce hardware resource usage. Settings: 8, 16, or 32 bits.

## Counter Resolution Tab

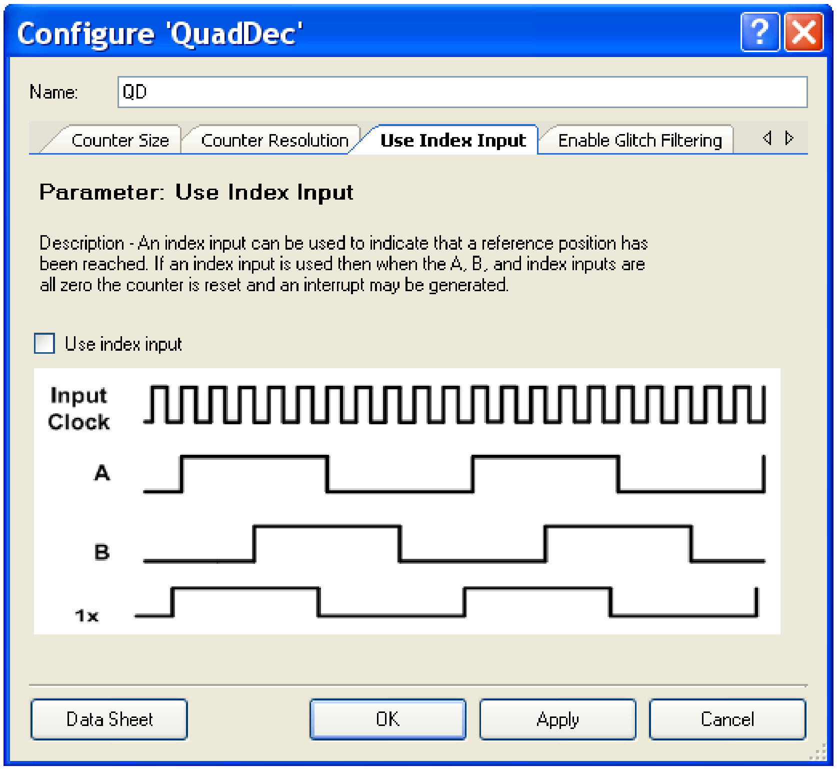



### Counter Resolution

Number of counts recorded in one period of the A and B inputs.

The transitions of the quadrature input(s) that are used to update the counter. The higher the resolution, the more accurately the position can be resolved, at the possible cost of a larger counter. Settings: 1x, 2x, or 4x.

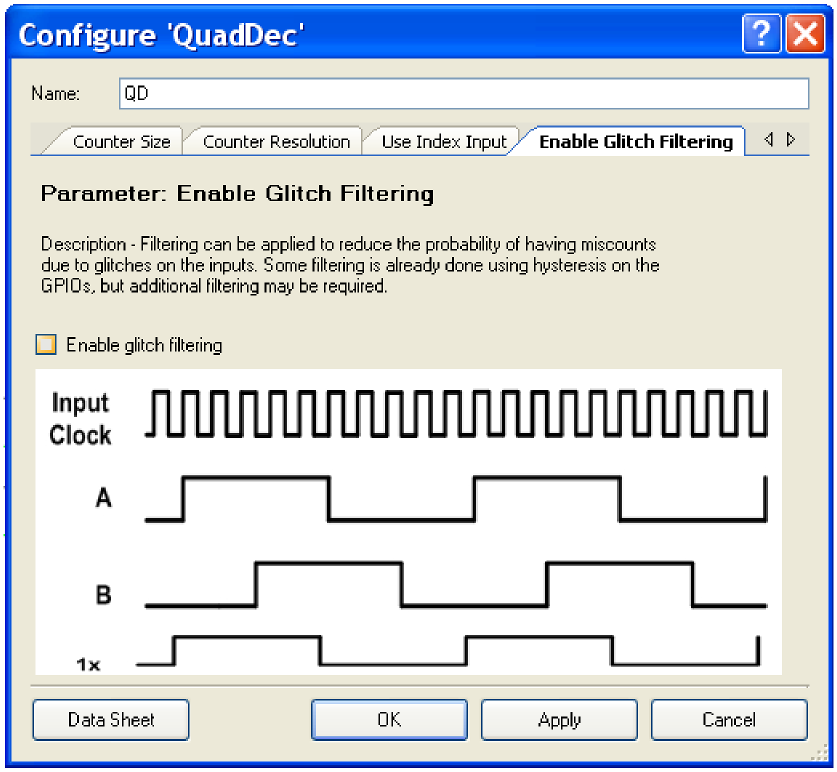
## Use Index Input Tab




### Use Index Input

Boolean whether or not a third input – index – exists and will be used. An index input can be used to indicate that a reference position has been reached. If an index input is used then when the A, B, and index inputs are all zero the counter is reset and an interrupt may be generated. Settings: True or false.

### Enable Glitch Filtering Tab




#### Enable Glitch Filtering

Boolean whether or not to apply digital glitch filtering to all inputs. Filtering can be applied to reduce the probability of having miscounts due to glitches on the inputs. Some filtering is already done using hysteresis on the GPIOs, but additional filtering may be required.

If filtering is selected, it is applied to all inputs. The filtered outputs do not change until three successive samples of the input are the same value.

For effective filtering, the period of the sample clock should be greater than the maximum time during which glitching is expected to take place.  Settings: True or false.

## Clock Selection

There is one internal clock in this component - bus clock. It clockings the status register and generates interrupts. A clock source for clocking Quadrature Decoder component must be connected.

## Placement

The Quadrature decoder component is placed in the UDB array and all placement information is provided to the API through the cyfitter.h file.

# Resources

| Resolution | Digital Blocks | | | | | API Memory (Bytes) | | Pins (per External I/O) |
|---|---|---|---|---|---|---|---|---|
| | Datapaths | Macro cells | Status Registers | Control Registers | Counter7 | Flash | RAM | |
| 8-Bits | 1 | | 1 | 0 | 0 | | | |
| 16-Bits | 2 | | 1 | 0 | 0 | | | |
| 32-Bits | 2 | | 1 | 0 | 0 | | | |

# Application Programming Interface

Application Programming Interface (API) routines allow you to configure the component using software. The following table lists and describes the interface to each function. The subsequent sections cover each function in more detail.

By default, PSoC Creator assigns the instance name "QuadDec_1" to the first instance of a component in a given design. You can rename it to any unique value that follows the syntactic rules for identifiers. The instance name becomes the prefix of every global function name, variable, and constant symbol.

| Function | Description |
|---|---|
| QuadDec_1_Start | Initializes UDBs and other relevant hardware. Resets counter to 0, enables or disables all relevant interrupts. Starts monitoring the inputs and counting. |
| QuadDec_1_Stop | Turns off UDBs and other relevant hardware. |
| QuadDec_1_GetCounter | Reports the current value of the counter |
| QuadDec_1_SetCounter | Sets the current value of the counter |
| QuadDec_1_GetEvents | Reports the current status of events |
| QuadDec_1_SetInterruptMask | Enables / disables interrupts due to the events. For the 32-bit counter, the overflow and underflow interrupts cannot be disabled; these bits are ignored. |
| QuadDec_1_GetInterruptMask | Reports the current interrupt mask settings |

## QuadDec_1_Start

**Description:** Initializes UDBs and other relevant hardware. Resets counter to 0, enables or disables all relevant interrupts. Starts monitoring the inputs and counting.

**Parameters:** None

**Return Value:** None

**Side Effects:** None

## QuadDec_1_Stop

**Description:** Turns off UDBs and other relevant hardware.

**Parameters:** None

**Return Value:** None

**Side Effects:** None

## QuadDec_1_GetCounter

**Description:** Reports the current value of the counter.

**Parameters:** None

**Return Value:** The counter value. Return type is signed and per the counter size setting. A positive value indicates clockwise movement (B before A).

**Side Effects:** None

## QuadDec_1_SetCounter

**Description:** Sets the current value of the counter.

**Parameters:** The new value. Parameter type is signed and per the counter size setting.

**Return Value:** None

**Side Effects:** None

## QuadDec_1_GetEvents

**Description:** Reports the current status of events.

**Parameters:** None

**Return Value:**

| Bit | Description |
|---|---|
| 0 | Counter overflow. |
| 1 | Counter underflow. |
| 2 | Counter reset due to index, if index input is used. |
| 3 | Invalid A, B inputs state transition. |

**Side Effects:** None

## QuadDec_1_SetInterruptMask

**Description:** Enables / disables interrupts due to the events. For the 32-bit counter, the overflow and underflow interrupts cannot be disabled; these bits are ignored.

**Parameters:** Enable / disable bits in an 8-bit value,where 1 enables the interrupt.

**Return Value:** None

**Side Effects:** None

## QuadDec_1_GetInterruptMask

**Description:** Reports the current interrupt mask settings.

**Parameters:** None

**Return Value:** Enable / disable bits in an 8-bit value, where 1 enables the interrupt.For the 32-bit counter, the overflow and underflow enable bits are always set.

**Side Effects:** None

## Sample Firmware Source Code

The following is a C language example demonstrating the basic functionality of the QuadDec component. This example assumes the component has been placed in a design with the default name "QuadDec_1."

**Note** If you rename your component you must also edit the example code as appropriate to match the component name you specify.

```
#include <device.h>

void main()
{
    QuadDec_1_Start();
}
```

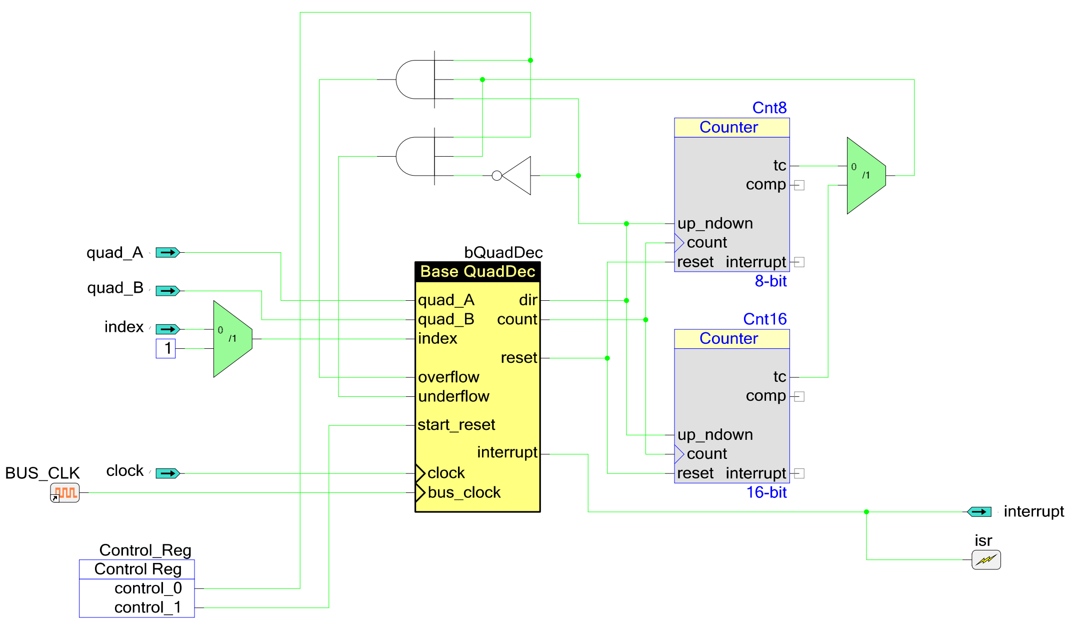
# Functional Description

## Default Configuration

The default configuration for the Quadrature Decoder is 8 – bit Up&Down Counter with 1x resolution.

# Block Diagram and Configuration

The Quadrature Decoder is only available as a UDB configuration of blocks. The API is described above and the registers are described here to define the overall implementation of the Quadrature Decoder.




# Registers

## Status

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| **Value** | reserved | | | | invalid in | reset | underflow | overflow |

The status register is a read only register which contains the various status bits defined for the Quadrature Decoder. The value of this register is available with the QuadDec_1_GetEvents()function call. The interrupt output signal is generated from an OR'ing of the masked bit-fields within the status register.

You can set the mask using the QuadDec_1_SetInterruptMask() function call and upon receiving an interrupt you can retrieve the interrupt source by reading the Status register with the QuadDec_1_GetEvents() function call. The Status register is clear on read so the interrupt source is held until the QuadDec_1_GetEvents() function is called. All operations on the status register must use the following defines for the bit-fields as these bit-fields may be moved around within the status register at build time.

There are several bit-fields masks defined for the status registers. Any of these bit-fields may be included as an interrupt source. All bit-fields configured as sticky bits in the status register. The defines are available in the generated header file (.h) as follows:

#### QuadDec_1_COUNTER_OVERFLOW

Defined as the bit-mask of the Status register bit “Counter overflow”.

#### QuadDec_1_COUNTER_UNDERFLOW

Defined as the bit-mask of the Status register bit “Counter underflow”.

#### QuadDec_1_RESET

Defined as the bit-mask of the Status register bit “Reset due index”.

#### QuadDec_1_INVALID_IN

Defined as the bit-mask of the Status register bit “invalid state transition on the A and B inputs”.

# DC and AC Electrical Characteristics

The following values are indicative of expected performance and based on initial characterization data.

## 5.0V/3.3V DC and AC Electrical Characteristics

| Parameter | Typical | Min | Max | Units | Conditions and Notes |
|---|---|---|---|---|---|
| Input | | | | | |
| Input Voltage Range | --- | | Vss to Vdd | V | |
| Input Capacitance | --- | | --- | pF | |
| Input Impedance | --- | | --- | Ω | |
| Maximum Clock Rate | --- | | 67 | MHz | |