
Counters

This lesson focuses on the counter functionality of a DAQ device. It begins with an overview of counters including counter signals, the parts of a counter, the pins you connect a counter signal to, basic counter terminology, and different types of counter chips. The lesson also describes the DAQmx VIs used for counter operations.

Topics

- A. Counter Signals
- B. Counter Chips
- C. Counter I/O
- D. Edge Counting
- E. Advanced Edge Counting
- F. Pulse Generation
- G. Pulse Measurement
- H. Frequency Measurements
- I. Position Measurement

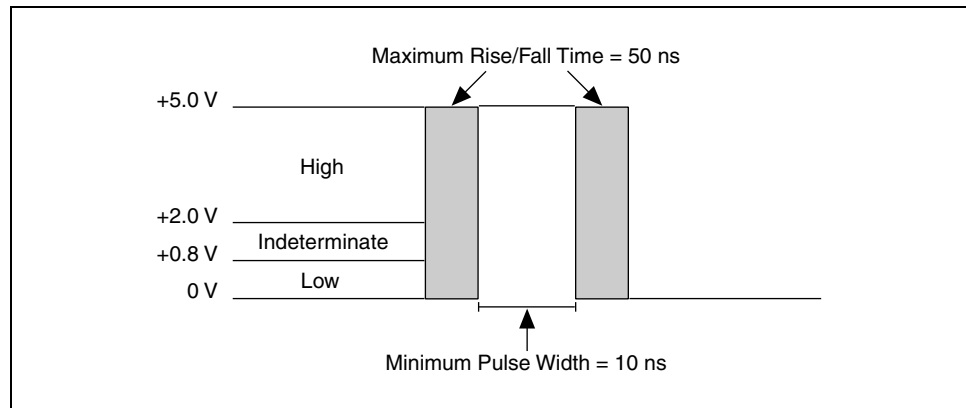
A. Counter Signals

Counters operate with TTL-compatible signals. A TTL-compatible signal has the following specifications:

$0\text{ V} - 0.8\text{ V} = \text{logic low}$

$2\text{ V} - 5\text{ V} = \text{logic high}$

Maximum Rise/Fall Time = 50 ns



Digital I/O devices can set or monitor the state of a digital line. Counters, however, are not only concerned with the state of the signal but also with the transition from one state to another. A counter can detect rising edges (transition from logic low to logic high) and falling edges (transition from logic high to logic low). Two important parameters related to sensing rising and falling edges are the rise/fall time and the minimum pulse width. The rise/fall time is a measure of how quickly the signal transitions from low to high or high to low. For a counter to detect the edge, the transition must occur within 50 ns or less, as defined by the specifications for a TTL-compatible signal.

In addition to this time restraint, there must be a minimum delay from the time a counter detects a rising or falling edge until it can detect another rising or falling edge. This delay is known as the minimum pulse width. The minimum pulse width depends on the counter chip that is used. E Series devices have a DAQ-STC chip, which has minimum pulse width of 10 ns for both the source and the gate. Consult the hardware documentation for the specific DAQ device to determine the minimum pulse width required for the counters.

M Series devices have a custom designed NI-STC 2 chip. The NI-STC 2 is an application-specific integrated circuit that controls interboard and intraboard synchronization and timing for multifunction DAQ operations. The NI-STC 2, used on all M Series devices, was also designed to be

compatible with the latest ADC technology, including the 18-bit ADCs used on the high-accuracy devices.

There are five different types of counter measurements—edge counting, pulse generation, pulse measurement, frequency measurement, and position measurement.

Parts of a Counter

A counter has the following main components:

- **Count Register**—Stores the current count of the counter. You can query the count register with software.
- **Source**—An input signal that can change the current count stored in the count register. The counter looks for rising or falling edges on the source signal. Whether a rising or falling edge changes the count is software-selectable. The type of edge selected is called the active edge of the signal. When an active edge is received on the source signal, the count changes. Whether an active edge increments or decrements the current count is also software-selectable. The source signal must be TTL-compatible.
- **Gate**—An input signal that determines if an active edge on the source changes the count. Counting can occur when the gate is high, low, or between various combinations of rising and falling edges. Gate settings are made in software. The gate is similar to a line mask in digital I/O because it allows you to acknowledge or ignore active edges on the source.
- **Out**—An output signal that generates pulses or a series of pulses, otherwise known as a pulse train. The output signal is TTL-compatible.

Counter Pins

Analog input, analog output, and digital I/O all have dedicated pins for input or output operations. Counters use a combination of Programmable Function Input (PFI) pins and dedicated pins for their operations. The output pins for counters are used solely for generating pulses on the out of a counter. The source and gate pins for counters are PFI pins and can be used for applications other than the source or gate of a counter. For example, pin 3 on the 68-pin connector can be used as PFI9, the gate of counter 0, or both. The ability to use one pin for multiple applications offers a great deal of flexibility. For example, you could wire an external TTL signal into pin 3 and use it to trigger an analog input operation and be a gate for a counter operation.

Counter Terminology

The following terms are important to understand when using counters.

- **Terminal Count**—The last count before a counter reaches 0. For example, when a counter that increments the count reaches its maximum count, it has reached the terminal count. The next increment of the count forces the counter to roll over and start counting at 0.
- **Resolution**—How high the counter can count before reaching the terminal count, specified in bits. The following formula calculates the maximum count based on the resolution:

$$\text{max count} = 2^{(\text{resolution})} - 1$$

Common counter resolutions are 16-, 24-, or 32-bits.

- **Timebase**—A signal of known frequency that is provided by the DAQ device. Typical frequencies for timebases range from 100 Hz to 80 MHz. The timebase can be routed internally to the source of a counter to provide a signal of known frequency.

B. Counter Chips

Depending on the DAQ device, you could be using the DAQ-STC, NI-STC 2 or NI-TIO counter chip.

DAQ-STC

The DAQ-STC is a 24-bit counter developed by National Instruments with a wide range of functionality that is used on E Series devices. The DAQ-STC increments or decrements the count, changes the count direction instantly by using a hardware signal, and offers 100 kHz and 20 MHz timebases.

The DAQ-STC is more widely available on current NI devices than any of the other chips. Throughout the remainder of this lesson, the term counter refers to the DAQ-STC chip.

NI-STC 2

The NI-STC 2 chip used on the M Series devices is a custom designed application-specific integrated circuit (ASIC) that controls interboard and intraboard synchronization and timing for multifunction DAQ operations. The NI-STC 2 manages all digital signals on the device, including inputs from the ADC, outputs to the DAC, digital I/O lines, and counter/timers. In addition to providing more digital I/O lines, the NI-STC 2 also has six DMA channels dedicated to I/O operations. With six DMA channels, all six operations on the device (analog input, analog output, digital input, digital output, counter/timer 0, and counter/timer 1) can execute simultaneously on their own dedicated DMA channel. This feature dramatically improves

execution speed and data throughput when multiple operations are performed simultaneously. The NI-STC 2 incorporates two 80 MHz, 32-bit counter/timers with built-in encoder support.

NI-TIO

The NI-TIO is the premier counter chip offered on NI devices. It is a 32-bit counter that is software-compatible with the DAQ-STC. The NI-TIO increments and decrements the count, supports encoders and a hardware trigger signal, has digital filters to remove glitches, changes the frequency of a pulse train instantly, and offers 100 kHz, 20 MHz, and 80 MHz timebases. The NI-TIO is used on the 660x family of devices. The M Series DAQ cards also support the NI-TIO counter features, such as two-edge separation and quadrature encoding and can be used for counter synchronization.



Note The remainder of this lesson focuses on the use of the DAQ-STC.

C. Counter I/O

Like analog input, analog output, and digital I/O; counter operations use the DAQmx Read VI. For counter operations, select a counter instance of the DAQmx Read VI. The DAQmx Write VI is not used with counters, as you will see in later exercises in this lesson. The DAQmx Create Virtual Channel VI, DAQmx Timing VI, and DAQmx Triggering VI are also used to configure counter measurements or generations.

DAQmx Create Virtual Channel VI

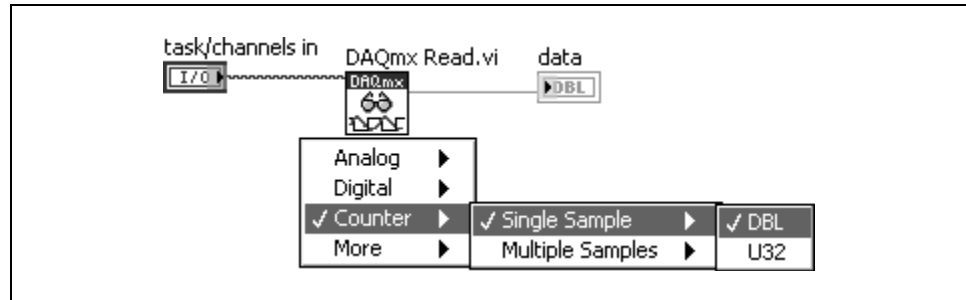
To programmatically create a counter input or output channel, select the Counter Input or Counter Output instance of the DAQmx Create Virtual Channel VI.

A Counter Input channel allows you to measure either frequency, period, count edges, pulse width, or semi period.

The configuration options for a Counter Output channel allow you to generate a pulse in terms of frequency, time, or ticks.

DAQmx Read VI

To read a sample or samples from a counter task, select a counter instance from the DAQmx Read VI pull-down menu. For counters, you can read only a single channel at a time, so the single or multiple channel selection window is no longer available.



Select to read either a single sample or multiple samples at once. When reading single samples, select to return the data as either a double-precision, floating-point numeric or unsigned 32-bit integer (U32). Multiple samples return as a 1D array of double-precision, floating-point numerics or a 1D array of unsigned 32-bit integers.

DAQmx Timing VI

For counter operations, select the Sample Clock or Implicit instances of the DAQmx Timing VI. The Sample Clock instance allows you to configure the actual timing rates. The Implicit instance sets only the number of samples to acquire or generate without specifying timing. Use the Implicit instance of the DAQmx Timing VI later in this lesson when you generate pulse trains.

DAQmx Trigger VI

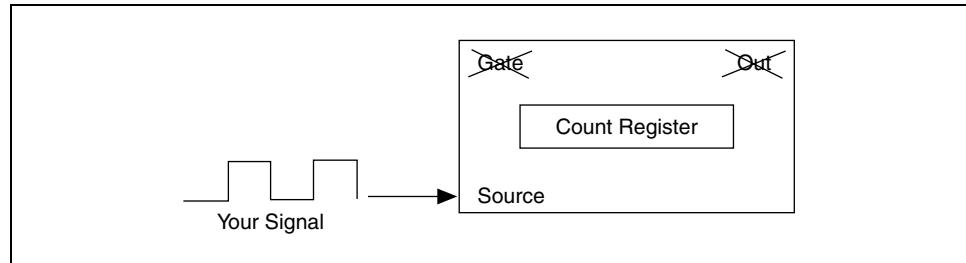
Use the DAQmx Trigger VI to configure triggering for the task. The instances of this polymorphic VI correspond to the trigger and trigger type to configure. Configure counter triggers settings the same way you configure analog input and analog output triggers. In addition, use the DAQmx Trigger Property Node to configure settings for a pause trigger. Refer to Lesson 4, *Analog Input*, of this manual for more information about configuring triggers.

D. Edge Counting

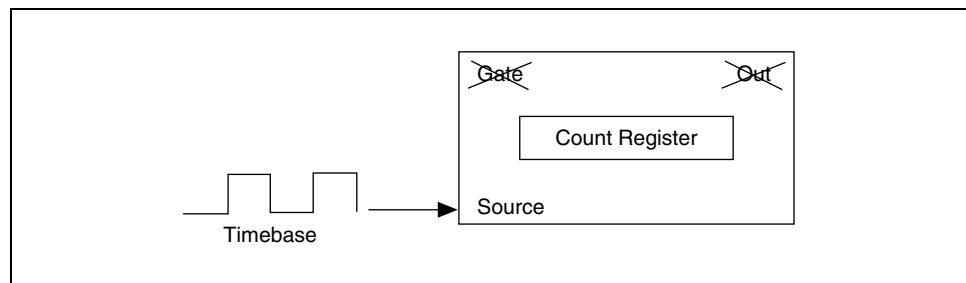
Edge counting is the most basic counter operation. In edge counting, the focus is on measuring the source signal. This section describes simple edge counting and time measurement.

Simple Edge Counting

Simple edge counting fits into the basic definition of a counter. The active edges of the source signal increment the value of the count register. An active edge can be software-selected to be a falling or rising edge. The gate and out are not used for simple edge counting.



Time measurement is a variation on simple edge counting. When you perform simple edge counting, the source is the unknown. You use the counter to help you measure the source. When you perform time measurement, the source is a timebase of known frequency. You can use your knowledge of the timebase frequency to help you measure elapsed time. Time measurement uses a timebase for the source instead of using the signal you are trying to measure.



The following formula calculates the elapsed time:

$$\text{elapsed time} = (\text{count register value}) \times (\text{timebase period})$$

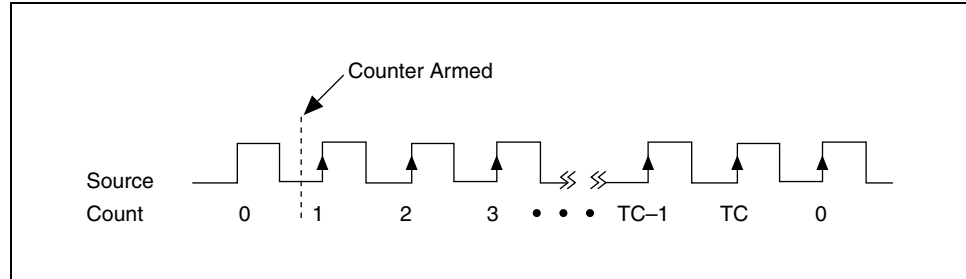
where timebase period = 1/timebase frequency.

The only difference between time measurement and simple edge counting is the signal that is used for the source.

Time Measurement

When a counter is configured for simple edge counting or time measurement, the count increments when an active edge is received on the source. You can use LabVIEW to specify if the active edge is rising or falling.

In the following example, the rising edge was selected as the active edge. The count increments by one every time a rising edge is reached.



Notice that the count cannot increment until the counter has been armed (started). A counter has a fixed number it can count to as determined by the resolution of the counter. For example, a 24-bit counter can count to:

$$2^{(\text{Counter resolution})} - 1 = 2^{(24)} - 1 = 16,777,215$$

When a 24-bit counter reaches the value of 16,777,215, it has reached the terminal count. The next active edge forces the counter to roll over and start at 0.

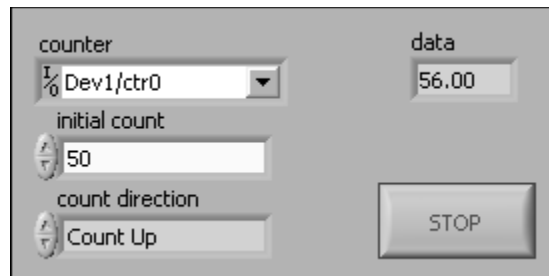
Exercise 9-1 Simple Edge Counting

Objective: To build a VI that performs simple edge counting.

In this exercise, you will build a VI that records the number of times an edge occurs.

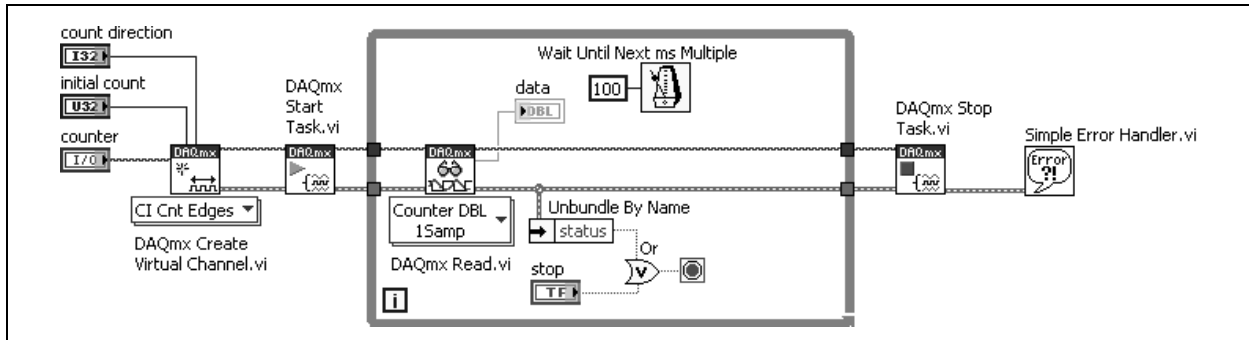
Front Panel

1. Open a blank VI. The following front panel results from building the block diagram.



Block Diagram

2. Build the following block diagram.



- a. Place the DAQmx Create Virtual Channel VI on the block diagram. This VI creates a new virtual channel. Select the **Counter Input»Count Edges** instance from the pull-down menu.

Right-click the **count direction**, **initial count**, and **counter** inputs and select **Create»Control** from the shortcut menu.



- b. Place the DAQmx Start Task VI on the block diagram. This VI starts a task.



- c. Place the DAQmx Read VI on the block diagram. Select the **Counter»Single Sample»DBL** instance from the pull-down menu to read a single double-precision, floating-point value from the counter.



Right-click the **data** output and select **Create»Indicator** from the shortcut menu.

- d. Place the DAQmx Stop Task VI on the block diagram. This VI stops the task.
 - e. Place the Wait Until Next ms Multiple function on the block diagram.
 - f. Place the Unbundle by Name function on the block diagram.
 - g. Place the Or function on the block diagram.
 - h. Place the Simple Error Handler VI on the block diagram.
3. Save the VI as `Simple Edge Counting.vi` in the `C:\Exercises\LabVIEW DAQ` directory.
 4. Wire channel A from the quadrature encoder on the DAQ Signal Accessory to Counter 0 Source for E-Series and M-Series devices.
 5. On the front panel, set the controls with the following values:
 - **Counter:** `DevX/ctr0`, where *X* corresponds to the device number of your DAQ device
 - **Initial Count:** 0
 - **Count Direction:** `Count Up`
 6. Run the VI. Rotate the quadrature encoder on the DAQ Signal Accessory. The **Count** indicator should increase in value. The quadrature encoder generates a pulse that feeds into the source of counter 0.
 7. Stop and close the VI.

End of Exercise 9-1

E. Advanced Edge Counting

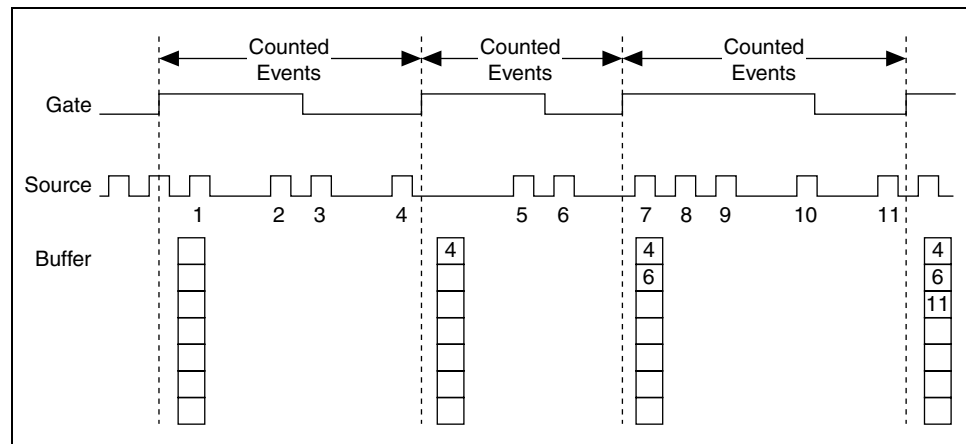
In addition to performing simple edge counting, NI-DAQmx can be easily configured to perform more advanced edge counting methods. These methods include pause trigger (gated) counting, and continuous and finite buffered counting.

Pause Trigger (Gated) Counting

In pause triggering, also known as gated triggering, an additional TTL signal enables/disables the count register. The counter value increases when the gate level is either high or low, depending on the configuration settings you choose with the DAQmx Trigger property node.

Continuous Buffered Edge Counting

In continuous buffered edge counting, an additional TTL signal latches the current count register value into a buffer. Thus, the value in the buffer is only updated on the gate's active edge. The following illustration demonstrates this transfer of the count register into the buffer.



Buffered edge counting is useful to measure the elapsed time between sequential edges occurring on the counter's gate. Active edges on the gate, latch the current counter register values into PC memory. Using either interrupts or DMA (software configurable with DAQmx Channel Property node), count register values are transferred individually to a software buffer across the PCI bus.

Finite Buffered Edge Counting

Finite buffered edge counting follows the same method for data transfer as continuous buffered edge counting, except that only a finite number of counts are acquired. As you will see in the following exercise, use the DAQmx Timing VI to set the number of samples to acquire.

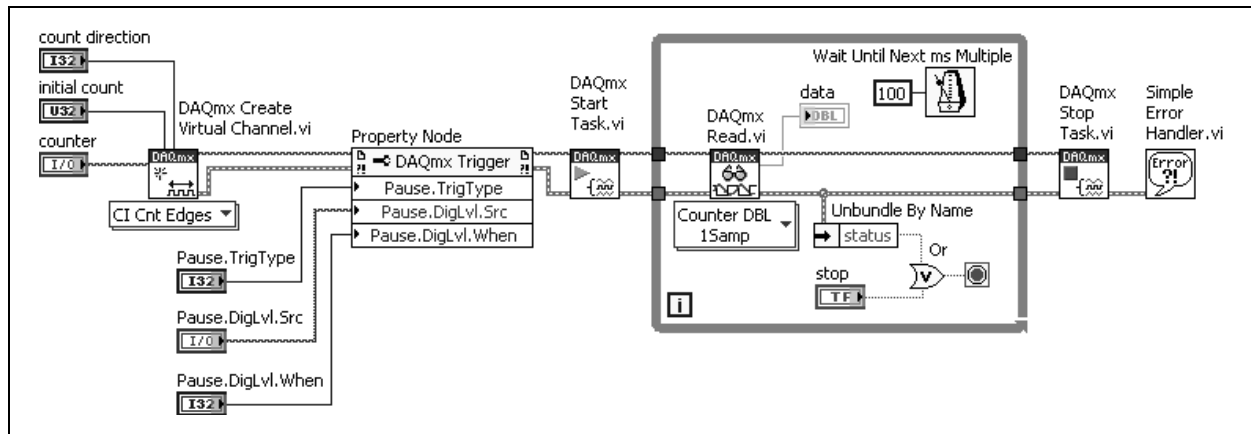
Exercise 9-2 Advanced Edge Counting

Objective: To use pause trigger, and finite buffered methods to perform edge counting.

Pause Trigger (Gated) Counting

In pause trigger, or gated, counting, you use an additional line to control the counter's gate. The gate pauses the increment or decrement of the counter when the gate is either in a high or low state (depending on the settings you choose).

1. Open the Simple Edge Counting VI located in the C:\Exercises\LabVIEW DAQ directory.
2. Select **File»Substitute Copy for Original** and save the VI as Simple Edge Counting - Gated.vi in the C:\Exercises\LabVIEW DAQ directory.
3. Modify the block diagram as shown in the following figure.



- a. Place the DAQmx Trigger Property Node on the block diagram. Use this Property Node to perform additional configuration of the task trigger. Resize the Property Node to contain three properties. Right-click each terminal and select the following properties from the shortcut menu:
 - **Properties»More»Pause»Trigger Type**
 - **Properties»More»Pause»Digital Level»Source**
 - **Properties»More»Pause»Digital Level»Pause When**
- b. For each property, right-click the terminal and select **Create»Control** from the shortcut menu.

- Switch to the front panel

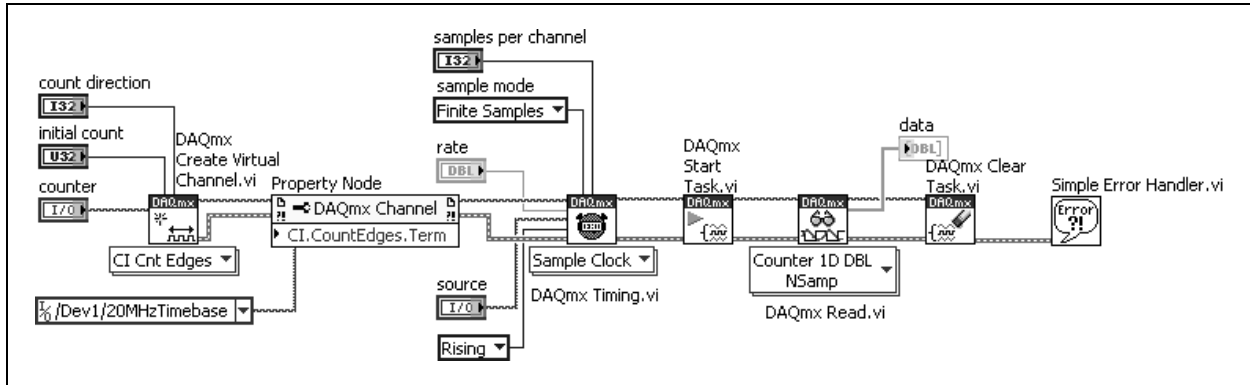


- Select the following values for the new controls.
 - Trigger Type:** Digital Level
 - Source:** /DevX/PFI0, where X corresponds to the device number of your DAQ device. PFI0 corresponds to the Digital Trigger button on the DAQ Signal Accessory.
 - Pause When:** Low
- Run the VI. Rotate the quadrature encoder knob. To pause the counting, hold down the Digital Trigger button. While you press the Digital Trigger button, rotate the quadrature encoder knob and notice that the count value remains unchanged.
- Stop the VI and change the **Pause When** value to High.
- Run the VI and observe the behavior with this trigger setting.
- Save and close the VI.

Finite Buffered Counting

In finite buffered counting, you specify the total number of edges to count. Counting stops when this number is reached.

- Open the Simple Edge Counting VI located in the C:\Exercises\LabVIEW DAQ directory.
- Select **File»Substitute Copy for Original** and save the VI as Simple Edge Counting - Finite Buffered.vi in the C:\Exercises\LabVIEW DAQ directory.
- Modify the block diagram as shown in the following figure.



a. Right-click the While Loop and select **Remove While Loop** from the shortcut menu.



b. Place the DAQmx Channel Property Node on the block diagram. Use this Property Node to configure additional settings about a channel. Right-click the terminal and select the **Properties»Counter Input»Count Edges»Input Terminal** property from the shortcut menu. Right-click the terminal to select **Create»Constant** from the shortcut menu, and then click the pull-down menu to select the 20 MHz Timebase for your DAQ device. Instead of counting the clicks on the quadrature encoder, count the edges of the internal timebase.



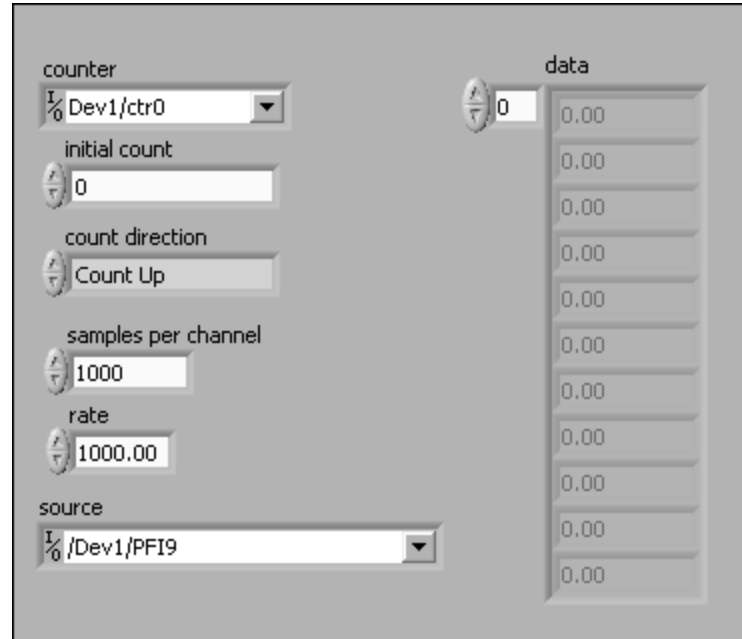
c. Place the DAQmx Timing VI on the block diagram and select the Sample Clock instance from the pull-down menu. Set the **sample mode** to **Finite Samples**. Right-click the **samples per channel** input and select **Create»Control** from the shortcut menu. Right-click the **rate** input and select **Create»Control** from the shortcut menu. Right-click the **source** input and select **Create»Control** from the shortcut menu. Right-click the **active edge** input and select **Create»Constant** from the shortcut menu.

d. Change the DAQmx Read VI to read one or more samples from the counter. Select the **Counter»Multiple Samples»1D DBL** instance from the pull-down menu to read one or more single double-precision, floating-point values from the counter. Right-click the **data** output and select **Create»Indicator** from the shortcut menu to display a 1D array of samples.



e. Replace the DAQmx Stop Task VI with the DAQmx Clear Task VI on the block diagram. This VI clears all resources assigned to the task.

4. Switch to the front panel



5. Select the following values for the new controls:

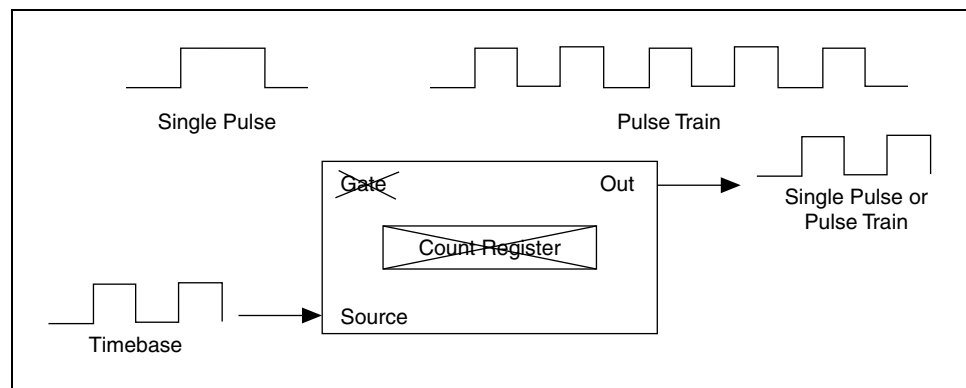
- **Samples per Channel:** 1000
- **Rate:** 1000
- **Source:** /DevX/PFI9, where *X* corresponds to the device number of your DAQ device. PFI9 is the default pin for the Counter 0 Gate.

6. Wire the square wave of the function generator to Counter 0 Gate.
7. Run the VI to see the latched value of the register at every rising edge of the gate signal in the Data field.
8. Save and close the VI.

End of Exercise 9-2

F. Pulse Generation

A counter not only measures TTL signals, but it also generates TTL signals. Using a counter to generate a TTL signal is known as pulse generation. The output signal shown in the following illustration is generated on the out of the counter. The generated signal can be a single pulse or a continuous set of pulses known as a pulse train. The counter uses a timebase as the source to help generate the pulse. For now, you only need to understand that a timebase helps generate the pulse. Knowing how the timebase generates a pulse is not necessary to program the counter for pulse generation and is beyond the scope of this course.

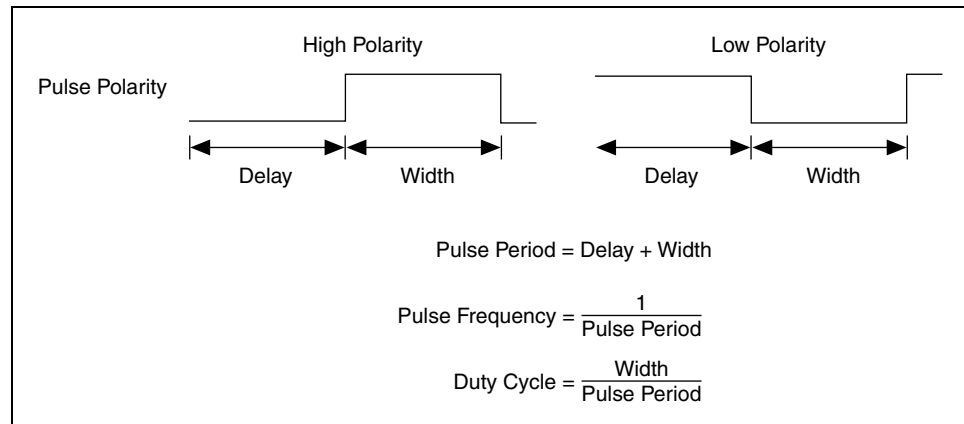


Pulse Characteristics

To generate a pulse, you must understand certain characteristics of a pulse. A pulse has two parts: the delay and the width. The delay is the first phase of the pulse, and the width is the second phase of the pulse. The delay and width are always at opposite logic levels.

For example, if the delay is logic low, the width must be logic high. A pulse can be characterized as high or low polarity. A high polarity pulse has a delay that is logic low and a width that is logic high. A low polarity pulse has a delay that is logic high and a width that is logic low. The naming convention for the pulse polarity corresponds to the logic level of its width. The period of a pulse is the time taken by the pulse to complete one cycle, so by adding the delay time to the width time, you can obtain the pulse period. After determining the period of the pulse, take the inverse to obtain the frequency of the pulse.

The delay and the width of a pulse are not always equal, so you need a property of a pulse that helps you determine if the delay is larger than the width or vice versa. The parameter you use is called the duty cycle. The following illustration shows the formula. The duty cycle gives you a number between 0 and 1. This number is often converted into a percentage. A pulse where the delay is equal to the width has a duty cycle of 0.5, or 50%. A duty cycle greater than 50% means the width is larger than the delay, and a duty cycle less than 50% means the delay is larger than the width.



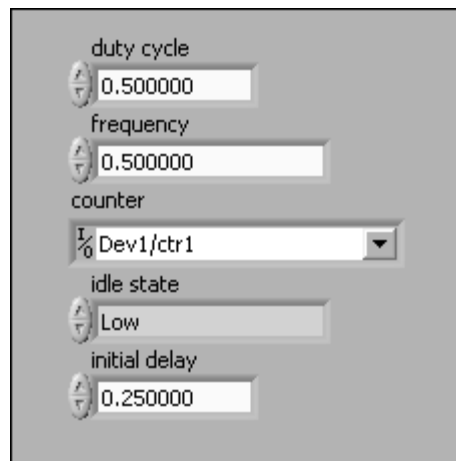
Exercise 9-3 Pulse Generation

Objective: To build a VI that generates a single pulse using a counter.

This VI demonstrates how to output a value to a counter. Although you output a frequency in this exercise, you also can output counter ticks and time with the same concept presented in this exercise.

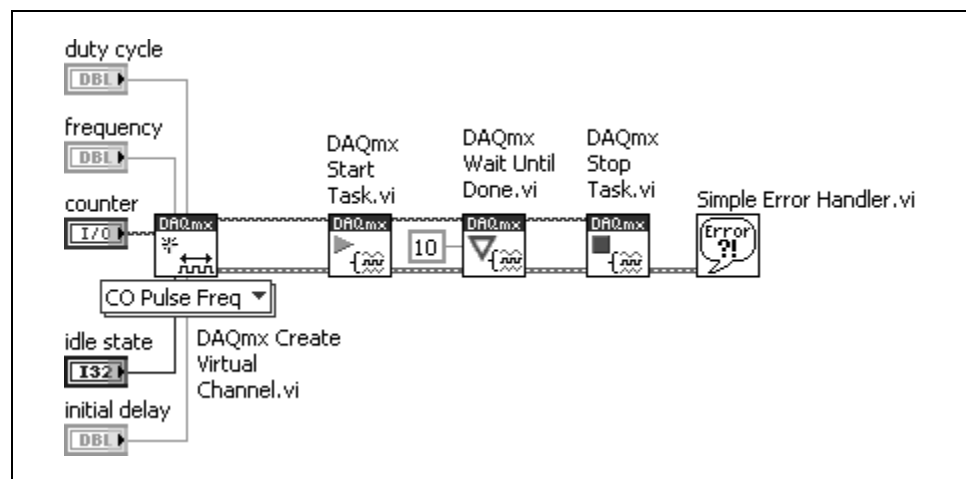
Front Panel

1. Open a blank VI.
2. The following front panel will result from building the block diagram.



Block Diagram

3. Build the following block diagram.



- a. Place the DAQmx Create Virtual Channel VI on the block diagram. This VI creates a new virtual channel. Select the **Counter Output»Pulse Generation»Frequency** instance from the pull-down menu.

Right-click the **duty cycle**, **frequency**, **counter**, **idle state**, and **initial delay** inputs and select **Create»Control** from the shortcut menu.



- b. Place the DAQmx Start Task VI on the block diagram. This VI starts a task.



- c. Place the DAQmx Wait Until Done VI on the block diagram. This VI waits for the generation to complete. Use this VI to ensure that the counter output completes before the task stops.

Right-click the **timeout** input and select **Create»Constant** from the shortcut menu. The default value is 10 seconds.



- d. Place the DAQmx Stop Task VI on the block diagram. This VI stops the task.

4. Save the VI as `Single Pulse Generation.vi` in the `C:\Exercises\LabVIEW DAQ` directory.
5. On the front panel of the VI, modify the following controls:
 - **Counter:** `/DevX/ctr1`, where *X* corresponds to the device number of your DAQ device
 - **Duty Cycle:** 0.5
 - **Frequency:** 0.5
 - **Idle State:** Low
 - **Initial Delay:** 0.25
6. Wire counter 1 out to counter 0 source and counter 1 out to analog in 1 on the DAQ Signal Accessory.
7. Open the Continuous Acquire with MIO VI located in the `C:\Exercises\LabVIEW DAQ` directory.
8. Set the controls on the front panel of the Continuous Acquire with MIO VI with the following values:
 - **Physical Channels:** `DevX/ai1`, where *X* corresponds to the device number of your DAQ device
 - **Samples per Channel:** 1000
 - **Rate:** 10000
9. Run the Continuous Acquire with MIO VI.
10. Run the Single Pulse Generation VI. You should see the pulse appear on the Continuous Acquire with MIO VI. Notice that the signal starts low, goes high, and returns to low.



Note It is easier to see the pulse if you disable Autoscale Y by right-clicking the graph while the VI is running and selecting **AutoScale Y** from the shortcut menu to remove the checkmark from the item. A good y-scale to see the pulse is -2 to 6.

11. Stop the Continuous Acquire with MIO VI.
12. Open the Simple Edge Counting VI located in the `C:\Exercises\LabVIEW DAQ` directory.
13. On the front panel, set the controls with the following values:
 - **Counter:** `DevX/ctr0`, where *X* corresponds to the device number of your DAQ device
 - **Initial Count:** 0
 - **Count Direction:** `Count Up`
14. Run the Simple Edge Counting VI.
15. Run the Single Pulse Generation VI and notice that the value in the Data field increases on the Simple Edge Counting VI.

End of Exercise 9-3

Exercise 9-4 Pulse Train Generation

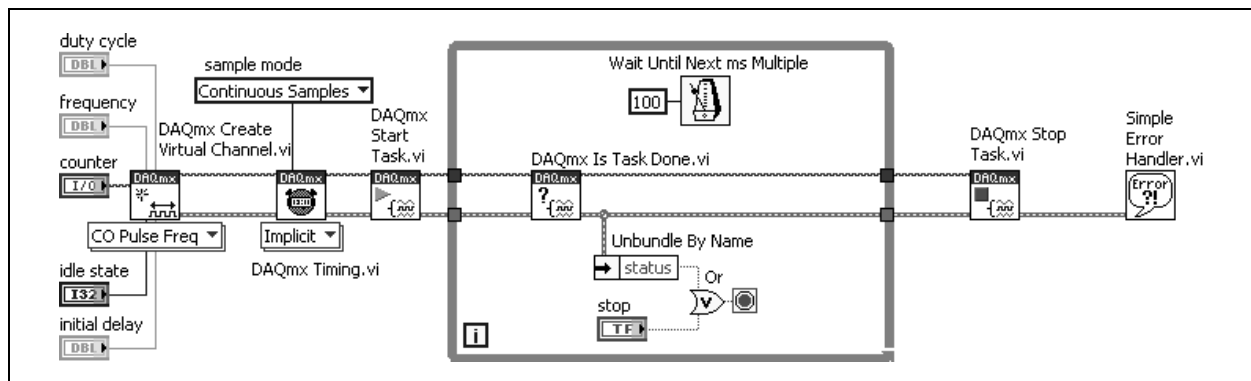
Objective: To build a VI that generates a pulse train.

In this exercise, you will modify the Single Pulse Generation VI to generate a pulse train or a series of pulses.

1. Open the VI Single Pulse Generation VI located in the `C:\Exercises\LabVIEW DAQ` directory.

Block Diagram

2. Modify the block diagram as shown in the following figure.



- a. Place the DAQmx Timing VI on the block diagram. Select the Implicit instance from the pull-down menu to configure timing so the task generates samples without specifying timing. A pulse train generation is ideal for the Implicit timing, because the pulse train itself contains all the timing parameters.



- b. Place the DAQmx Is Task Done VI on the block diagram. This VI waits for the generation to complete. Use this VI to ensure that the counter output completes before the task stops.



- c. Place the Wait Until Next ms Multiple function on the block diagram.



- d. Place the Unbundle by Name function on the block diagram.

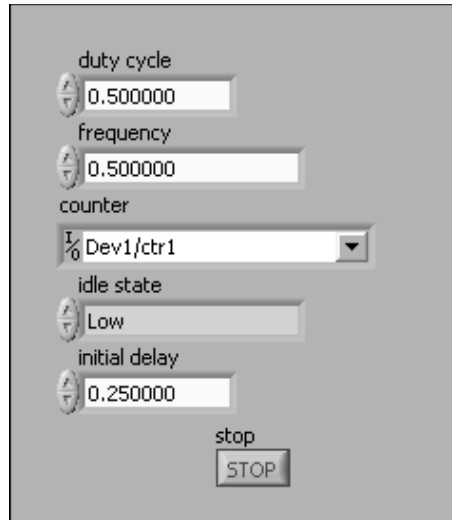


- e. Place the Or function on the block diagram.



- f. Place the Simple Error Handler VI on the block diagram.

3. Switch to the front pane.



4. Set the front panel controls with the following values.
 - **Counter:** /DevX/ctr1, where *X* corresponds to the device number of your DAQ device
 - **Duty Cycle:** 0.5
 - **Frequency:** 0.5
 - **Idle State:** Low
 - **Initial Delay:** 0.25
5. Wire counter 1 out on the DAQ Signal Accessory to analog in 1.
6. Select **File»Substitute Copy for Original** and save the VI as `Pulse Train Generator.vi` in the `C:\Exercises\LabVIEW DAQ` directory.
7. Open the Continuous Acquire with MIO VI located in the `C:\Exercises\LabVIEW DAQ` directory.
8. Set the controls on the front panel of the Continuous Acquire with MIO VI with the following values:
 - **Physical Channels:** DevX/ai1, where *X* corresponds to the device number of your DAQ device
 - **Samples per Channel:** 1000
 - **Rate:** 10000
9. Run the Continuous Acquire with MIO VI.
10. Run the Pulse Train Generator VI. You should see the pulse train appear on the graph of the Continuous Acquire with MIO VI.
11. Stop and close the Pulse Train Generator VI.

End of Exercise 9-4

Exercise 9-5 Retriggerable Pulse Train Generation

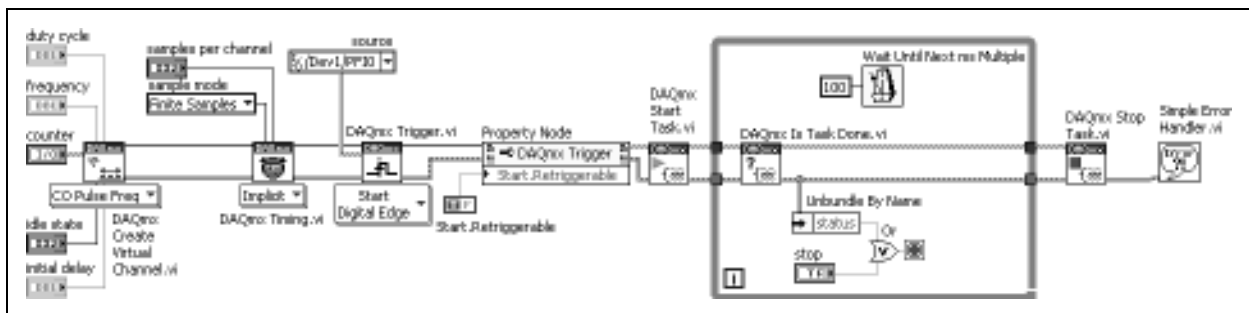
Objective: To build a VI that generates a retriggerable pulse train.

You can configure single pulse generation and finite pulse train generation to be retriggerable. The counter remains armed after generating the first pulse and can respond to triggers on the gate by generating a pulse on the counter's output line. In this exercise, you modify the Pulse Train Generator VI to generate a retriggerable pulse train.

1. Open the Pulse Train Generator VI located in the C:\Exercises\LabVIEW DAQ directory.
2. Select **File»Substitute Copy for Original** and save the VI as Retriggerable Pulse Train.vi in the C:\Exercises\LabVIEW DAQ directory.

Block Diagram

3. Modify the block diagram as shown in the following figure.



- a. Modify the DAQmx Timing VI located on the block diagram. Leave the task timing set to **Implicit**. Set the **sample mode** to **Finite Samples**. Right-click the **samples per channel** input and select **Create»Control** from the shortcut menu.

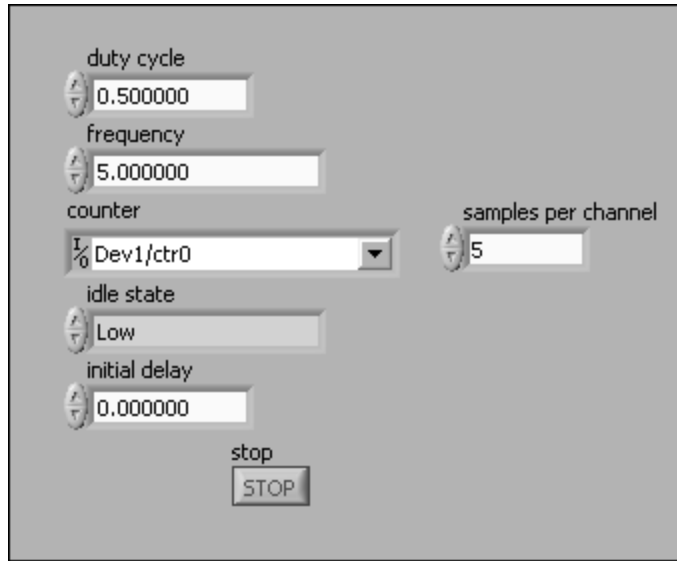


- b. Place the DAQmx Trigger VI on the block diagram. From the pull-down menu, select **Start»Digital Edge**. Now, right-click the **source** input and select **Create»Constant** from the shortcut menu. From the pull-down menu, select **DevX/PFI0**, where *X* corresponds to the device number of your DAQ device.



- c. Place the DAQmx Trigger Property Node on the block diagram. Use this Property Node to configure additional settings for the task triggering. Right-click the node and select **Properties»Start»More»Retriggerable** from the shortcut menu. Now, again right-click the Property Node and select **Create»Constant** from the shortcut menu. Set the Boolean constant to **True** to allow the pulse train to be retriggerable.

- Switch to the front panel.

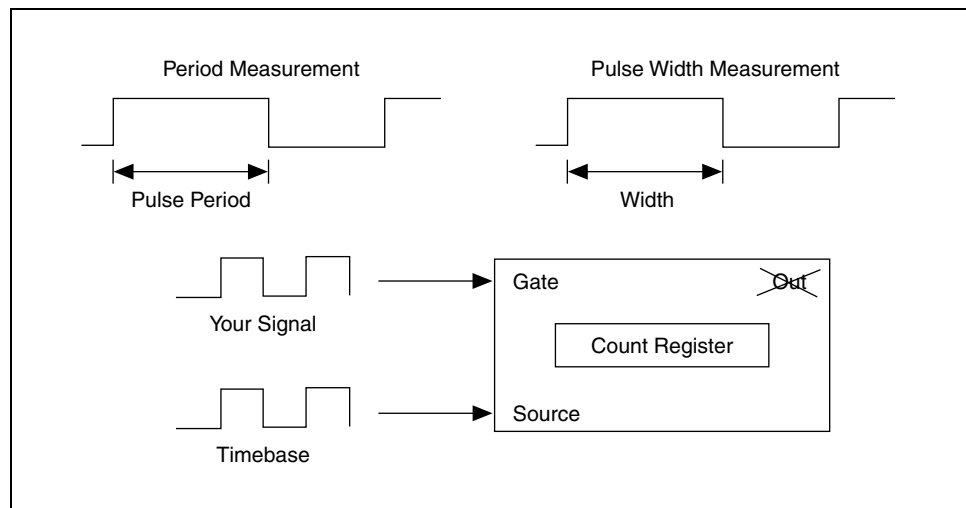


- Set the controls with the following values:
 - Counter:** DevX/ctr0, where X corresponds to the device number of your DAQ device
 - Duty Cycle:** 0.5
 - Frequency:** 5
 - Idle State:** Low
 - Initial Delay:** 0
 - Samples per channel:** 5
- Wire counter 0 out to analog in 1 on the DAQ Signal Accessory.
- Launch MAX and open a test panel for your NI-DAQmx device. Click the **Analog Input** tab and select the channel DevX/ai1. Set the **Mode** to Continuous, the **Rate (Hz)** to 50.00, and the **Samples To Read** to 50. Click the **Start** button.
- Run the VI. Press the Digital Trigger button on the DAQ Signal Accessory to trigger the pulse generation. Watch the test panel and count the number of pulses. The number of pulses should be equivalent to the samples per channel control value.
- Press the Digital Trigger button again. The pulse train is produced each time the trigger is enabled, making it a retriggerable pulse train.
- Stop the VI and the test panel. Minimize the MAX window.
- Save and close the VI.

End of Exercise 9-5

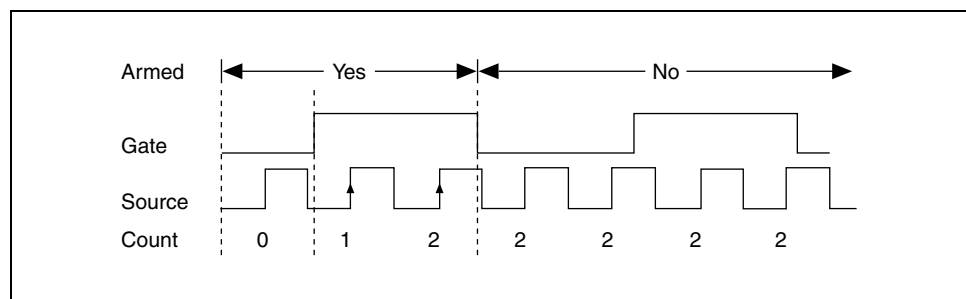
G. Pulse Measurement

When you measure a pulse, the signal you are measuring is used as the gate and the source is a timebase of known frequency, as shown in the following illustration. You can use the known frequency of the timebase and the value of the count register to determine characteristics of the gate pulse, such as pulse period and pulse width.



Period Measurement

Period measurement is one type of pulse measurement. With period measurement, you still count the active edges on the source signal. However, unlike simple edge counting, you only increment the count during the period of the gate signal. The following illustration shows period measurement that is started and stopped by a rising edge on the gate signal.



You also can start and stop counting between falling edges. The count reflects the number of rising edges on the source between the two rising edges on the gate. Therefore, to perform period measurement, you need a signal with two rising edges or two falling edges. A single pulse has only one rising edge and one falling edge, so you would not be able to measure its period.

In the previous example, one period of the gate signal has four counts. Remember that the source is a timebase of known frequency. Assume that you are using a 100 kHz timebase. The formula for calculating the period of the gate is as follows:

$$\text{pulse period} = \text{count} \times (1/\text{source frequency})$$

For the previous example, with the 100 kHz timebase, the formula yields:

$$\text{pulse period} = 4 \times (1/100,000) = 0.04 \text{ milliseconds}$$

Semi Period Measurement

Semi period measurement is very similar to period measurement, but you are only measuring the time between consecutive edges. The formula for calculating the semi period is as follows:

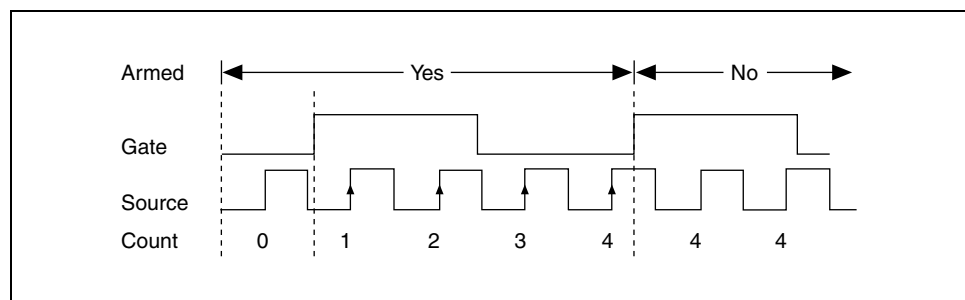
$$\text{pulse period} = \text{count} \times (1/(2 \times \text{source frequency}))$$

For the previous example, with the 100 kHz timebase, the formula yields:

$$\text{pulse period} = 4 \times (1/200,000) = 0.02 \text{ milliseconds}$$

Pulse Width Measurement

Pulse width measurement is very similar to period measurement. The difference is where you stop counting. With period measurement, you started and stopped counting with two rising edges on the gate signal. With pulse width measurement, you count only during the pulse width, so you start counting on one edge and end on the opposite edge. The count value increments only between two opposite edges, as shown in the following illustration.



The formula for calculating the pulse width is the same as the formula for the period:

$$\text{pulse width} = \text{count} \times (1/\text{source frequency})$$

For the previous example, a 100 kHz source yields:

$$\text{pulse width} = 2 \times (1/100,000) = 0.02 \text{ milliseconds}$$

0.02 milliseconds is half the value that you received for period measurement, so you must have a gate signal with a duty cycle of 50%.

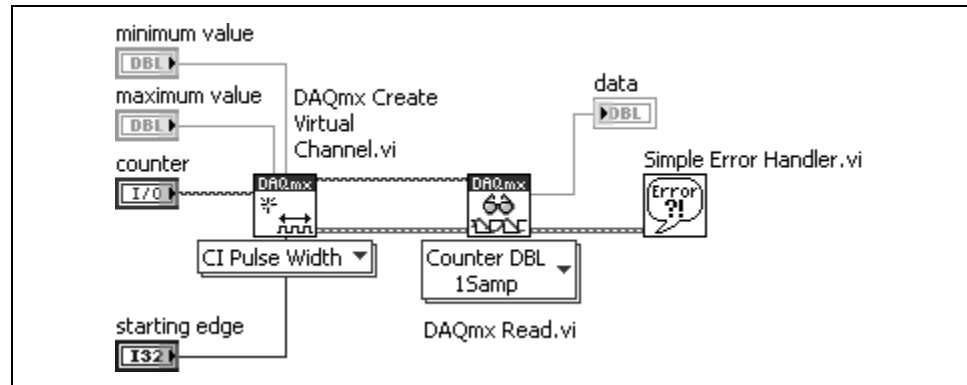
Exercise 9-6 Period, Semi Period, and Pulse Width Measurement

Objective: To build a VI to measure a pulse, period, and semi period.

1. Open a blank VI and switch to the block diagram.

Pulse Width Measurement

2. Create the following block diagram.



3. Save the VI as `Signal Measurements.vi` in the `C:\Exercises\LabVIEW DAQ` directory.
4. Wire Counter 0 Out to Counter 1 Gate on the DAQ Signal Accessory.
5. Open the Single Pulse Generation VI located in the `C:\Exercises\LabVIEW DAQ` directory.
6. Set the controls on the front panel of the Single Pulse Generation VI with the following values:
 - **Counter:** `DevX/ctr0`, where `X` corresponds to the device number of your DAQ device
 - **Duty Cycle:** 0.5
 - **Frequency:** 1
 - **Idle State:** Low
 - **Delay:** 0.5
7. Set the controls on the front panel of the Signal Measurements VI with the following values:
 - **Counter:** `DevX/ctr1`, where `X` corresponds to the device number of your DAQ device
 - **Starting edge:** Rising
 - **Maximum value:** 0.001000
 - **Minimum value:** 0.000001



Note The signal you are measuring must start low when you are measuring a high pulse. Likewise, the signal must start high when you are measuring a low pulse. If the signal starts high when you are measuring a high pulse, or vice versa, an error results.

8. Run the Signal Measurements VI.
9. Run the Single Pulse Generation VI.

You should see that the Signal Measurements VI measured the 0.5 second pulse the Single Pulse Generation VI generates.

Period Measurement

1. Close the Single Pulse Generation VI and open the Pulse Train Generator VI that you completed in Exercise 9-4.
2. Modify the Signal Measurements VI to read the Period instead of the Pulse Width. Create a control for the **Measurement Time** and set it to 0.05.
3. Set the controls on the front panel of the Pulse Train Generator VI with the following values:
 - **Counter:** /DevX/ctr0, where X corresponds to the device number of your DAQ device
 - **Duty Cycle:** 0.5
 - **Frequency:** 2
 - **Idle State:** Low
 - **Initial Delay:** 0.25
4. Run the Pulse Train Generator VI.
5. Run the Signal Measurements VI. Notice that you measure a period of 0.5 seconds.
6. Stop the Pulse Train Generator VI.

Semi Period Measurement

1. Modify the Signal Measurements VI to read the semi period instead of the Period.
2. Run the Pulse Train Generator VI, leaving the control values the same as for the period measurement.
3. Run the Signal Measurements VI. The semi period is 0.25 seconds.
4. Stop the Pulse Train Generator VI.
5. Save the Signal Measurements VI.
6. Close all VIs.

End of Exercise 9-6

H. Frequency Measurements

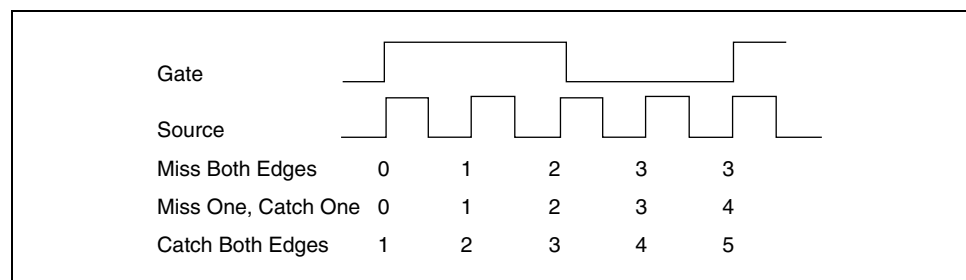
This section describes three ways to measure the frequency of a TTL pulse train using one or more counters. The frequency of a waveform is simply the inverse of its period at any given time. Thus, the easiest type of frequency measurement is merely the inverse of a period measurement. The two other types of frequency measurements become necessary because the first type of frequency measurement becomes increasingly inaccurate as the gate frequency approaches the timebase frequency of the counter.

Period

The first frequency measurement is really a period measurement. When the period is acquired, take the inverse to compute frequency. The advantage of this method is that it uses only one counter and is easy to perform. However, this method relies on a relatively slow gate signal because the accuracy of a period measurement depends on the number of source edges that occur within one gate period.

Synchronization Error

As the gate frequency approaches the source frequency, period measurements suffer from synchronization error. For example, consider a period measurement that uses a 20 MHz internal timebase on the source. Now, suppose the gate signal is roughly 5 MHz, or 1/4 of the source frequency. The following illustration shows three possible scenarios in which the first and last source edges may or may not be included in the period measurement.



In the first scenario, the measurement misses the first and last source edges, counting a total of only three edges. The second scenario catches the first four edges and misses the last edge. The third scenario shows all five source edges being counted. The second scenario is obviously more accurate, but because the source edges are so closely synchronized with the gate edges, the counter is equally likely to pick any one of the three scenarios shown.

Pulse measurements always have an error of ± 1 source cycle, which is generally negligible when one source cycle accounts for only 1% (or less) of the pulse measurement. However, one source cycle accounts for between

33% to 20% of the measurement. This is known as synchronization error and can be avoided by choosing a different measurement scheme.

The following table shows how two frequencies, 50 kHz and 5 MHz, affect a period measurement.

Actual Frequency	Number of 50 ns Cycles	Measurement Error or +1 Cycle	Measurement Error of -1 Cycle	Frequency with Error of +1 Cycle	Frequency with Error of -1 Cycle
50 kHz	400	401	399	49.88 kHz	50.13 kHz
5 MHz	4	5	3	4 MHz	6.67 MHz

In NI-DAQmx, this method is called **Low Frequency with 1 Counter**.

Averaging

The second method of measuring frequency uses two counters—one to generate a pulse train with a known frequency and one to perform a period measurement. Counter 1 performs a period measurement, using the external signal as a source instead of the internal timebase. The gate signal of counter 1 comes from the output of counter 0, which is generating a pulse train. Because you know the frequency of the output of counter 0, you know exactly the length of the gate cycle on counter 1. Based on the number of source edges that arrive on the source of counter 1, you can deduce frequency, dividing the period measurement of counter 1 by the gate period. For example, if counter 0 outputs a pulse train of 10 Hz, the gate period is 0.1 s. If during that time, you count 100 source edges, you know the source frequency on counter 1 is $(100 \pm 1)/0.1$ or 1000 ± 10 Hz. In NI-DAQmx, this method is called **High Frequency with 2 Counters**.

Divide Down Method

The third method of measuring frequency also uses two counters except the counter that generates the pulse train (counter 0) uses the external signal as the source, and the counter that performs the period measurement (counter 1) uses the internal timebase as its source. Like the averaging method, the divide down method uses the pulse train from the output of counter 0 to gate the period measurement on counter 1.

The advantage of the divide down method is that it introduces less error than period measurements or averaging.

For example, suppose you program counter 0 for pulse train generation with pulse specs 5 and 5. This means that the delay and width are each made up of 5 periods of the source signal, and that the period of the resulting pulse train consists of 10 periods of the source signal (the source is divided down

by a factor of 10). In this example, counter 1 is configured for a period measurement, using the internal 20 MHz timebase as its source. If counter 1 registers 100 source edges during one gate period, you can deduce that the gate period lasted 5 μ s (50 ns \times 100 edges). You can therefore conclude that the external signal wired to the source of counter 0 had a period of 0.5 μ s or a frequency of 2 MHz.

Expressed as an equation,

$$F = (\text{pulse spec1} + \text{pulse spec2}) \times \text{timebase}/(\# \text{ of source edges} \pm 1)$$

In this example,

$$F = (5 + 5) \times 20,000,000/(100 \pm 1)$$

$$F = 200,000,000/101 \text{ to } 200,000,000/99$$

$$F = 1,980,198 \text{ to } 2,020,202 \text{ Hz}$$

In NI-DAQmx, this method is called **Large Range with 2 Counters**.

Exercise 9-7 Frequency Measurement

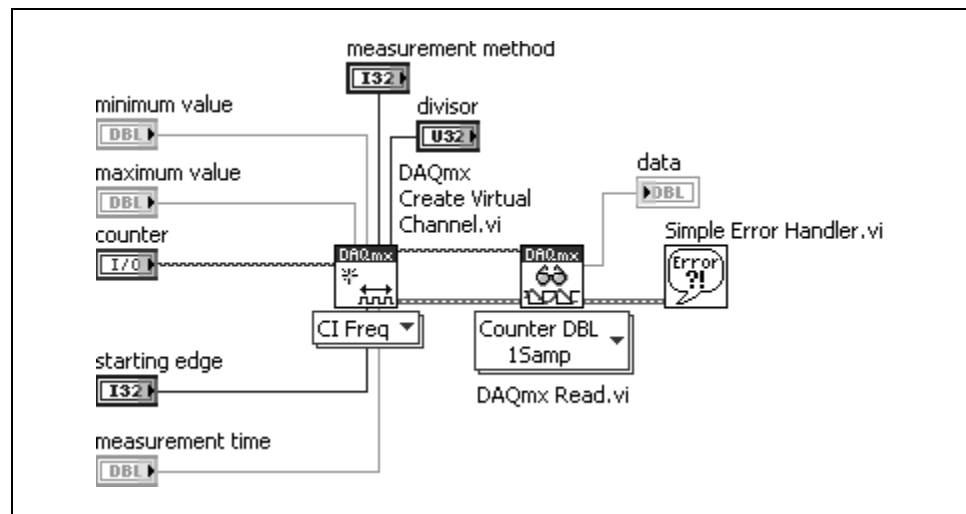
Objective: To build a VI that measures frequency using a counter.

This exercise explores three different methods of measuring frequency—inverse period, averaging, and divide down.

1. On the DAQ Signal Accessory, connect the square wave from the function generator to the gate of counter 1.
2. Open a blank VI and switch to the block diagram.

Inverse Period Method

3. Create the following block diagram.



- a. On the DAQmx Create Virtual Channel VI, right-click the **measurement method**, **minimum value**, **maximum value**, **counter**, **starting edge**, **measurement time**, and **divisor** inputs and select **Create»Control** from the shortcut menu.
 - b. On the DAQmx Read VI, select the **Counter»Single Sample»Double** instance from the pull-down menu. Right-click the **data** output and select **Create»Indicator** from the shortcut menu.
4. Save the VI as `Frequency Measurements.vi` in the `C:\Exercises\LabVIEW DAQ` directory.

- Switch to the front panel.



- Set the controls on the front panel with the following values:
 - Counter:** DevX/ctr1, where X corresponds to the device number of your DAQ board
 - Minimum value:** 2
 - Maximum value:** 10,000
 - Divisor:** 4
 - Starting edge:** Rising
 - Measurement time:** 1
- On the DAQ Signal Accessory, set the Frequency Range to 100 Hz – 10 kHz. Turn the Frequency Adjust Knob as low as possible. The function generator should now produce a 100 Hz signal.
- Run the VI. The data indicator should display approximately 100 Hz.
- Experiment with the VI by adjusting the Frequency Adjust Knob and running the VI again.

Averaging Method

- On the DAQ Signal Accessory, connect the square wave from the function generator to the source of counter 1.

11. Change the **measurement method** to High Frequency with 2 Counters. Decrease the **measurement time** to 0.001. Increase the **maximum value** to 1000000.
12. On the DAQ Signal Accessory, set the Frequency Range to 13 kHz – 1 MHz. Turn the Frequency Adjust Knob as high as possible.
13. Run the VI. The data indicator should display approximately 1 MHz.
14. Experiment with the VI by adjusting the Frequency Adjust Knob and running the VI again.

Divide Down Method

15. Change the **measurement method** to Large Range with 2 Counters. Set the **minimum value** to 5.
16. On the DAQ Signal Accessory, set the Frequency Range to 13 kHz – 1 MHz. Turn the Frequency Adjust Knob as high as possible.
17. Run the VI. The data indicator should display approximately 1 MHz.
18. Experiment with the VI by adjusting the Frequency Adjust Knob and running the VI again.

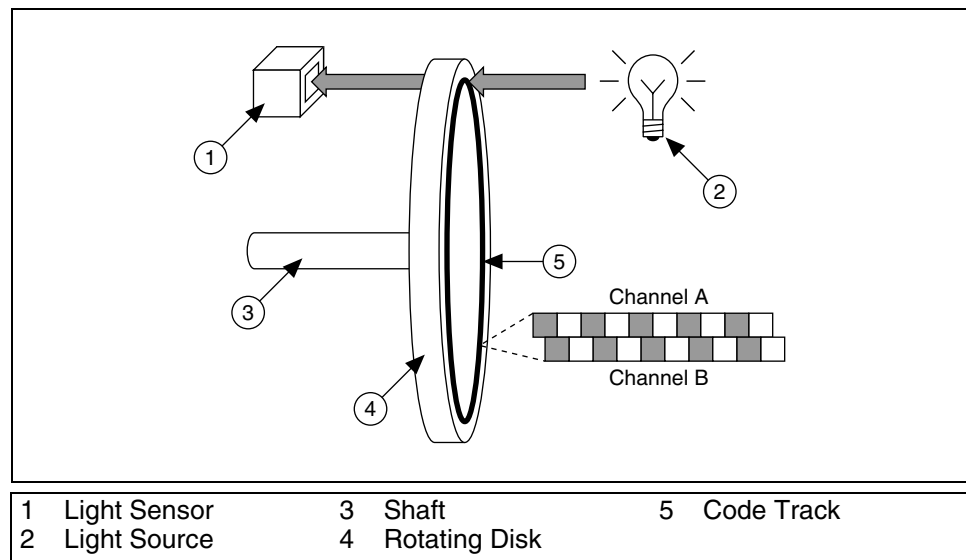
End of Exercise 9-7

I. Position Measurement

A quadrature encoder is a popular transducer used in counter applications. A quadrature encoder allows you to measure position, and it converts rotary motion into a measurable signal. The DAQ Signal Accessory has a quadrature encoder. Of the four counter chips, the NI-TIO is the only one that directly supports quadrature encoders. M-Series supports the NI-TIO counter features, such as quadrature encoding. Quadrature encoders *can* be measured with both the AM9513 and DAQ-STC, but neither chip was designed for encoder measurements. If you are measuring quadrature encoder signals, the best option is to use the NI-TIO chip.

How Encoders Work

An encoder is a transducer that allows you to measure position or distance. To understand how an encoder works, examine the quadrature encoder shown in the following illustration.

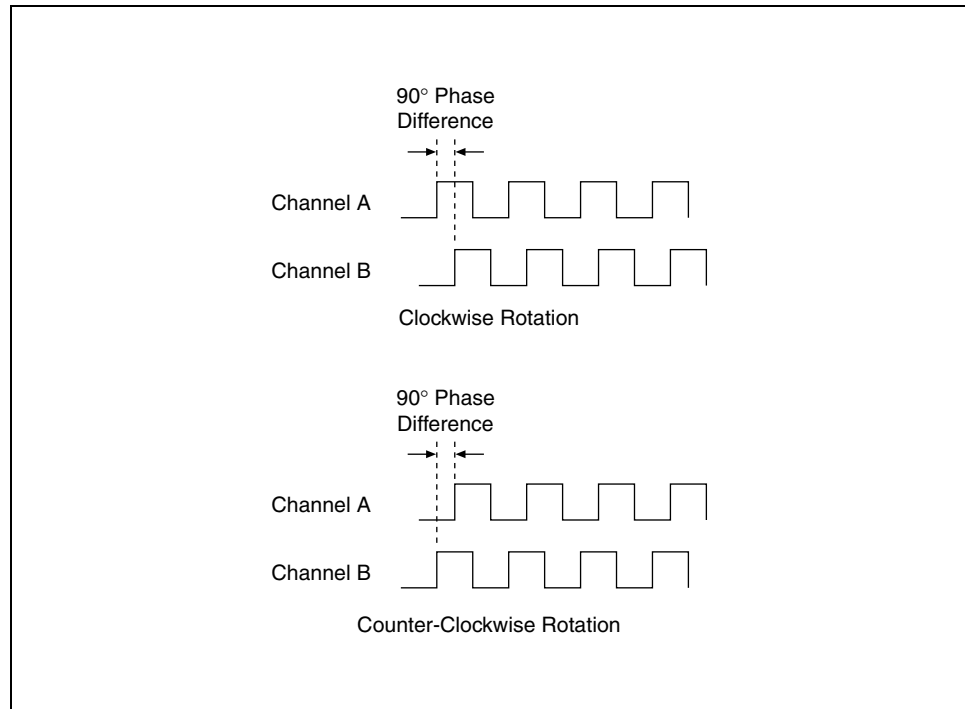


A quadrature encoder helps you convert rotary motion into a measurable signal. The rotary motion you are interested in is the spinning of the shaft. You could be interested in the direction the shaft is rotating, the speed the shaft is rotating, or both. A disk is attached to the shaft so that it rotates in the same direction and at the same speed as the shaft. The rotating disk is placed between a light source and a light sensor. The disk has a section of alternating opaque and transparent sections called the code track. An opaque section blocks the light from the source to the sensor, and a transparent section allows the light to pass from the source to the sensor. The code track consists of two rings of alternating opaque and transparent sections. Each ring produces a pulse train. The two rings are offset so that, depending on the direction the disk rotates, one pulse train leads the other.

The number of opaque and transparent sections determines how many pulses are produced per revolution. This is an important specification to know if you are keeping track of how many revolutions the shaft has completed. The quadrature encoder on the DAQ Signal Accessory produces 24 pulses per revolution.

Quadrature Encoder

Most encoders produce a TTL-compatible signal that can be used with a counter. As you saw earlier, a quadrature encoder produces two pulse trains. The two pulse trains are referred to as Channel A and Channel B. Channel A and Channel B are always out of phase by 90 degrees, as shown in the following illustration. The leading channel is determined by the direction of rotation. If the encoder is rotated in a clockwise direction, Channel A leads Channel B. If the encoder is rotated in a counter-clockwise direction, Channel B leads Channel A.



Note Some encoders might not produce a signal that is suitable for a counter. For example, some encoders produce a differential signal, and none of the four NI counter chips support differential inputs. If the signal produced by the encoder is not compatible with the counter chip you are using, you need to use signal conditioning before you send the signal to the counter.

Up/Down Line

To measure encoders with the DAQ-STC, you need to use a special input to the counter called the up/down line. The up/down line for counter 0 is the same pin as DIO6 on the pinout, and DIO7 for counter 1. The up/down line determines if an active edge on the source increments or decrements the count. If the signal sent to the up/down line is TTL high, an active edge on the source increments the count. If the signal sent to the up/down line is TTL low, an active edge on the source decrements the count.

DAQ-STC and Encoders

To measure an encoder with the DAQ-STC, both the source and the up/down lines are used. Connect Channel A to the source and Channel B to the up/down line. When you connect signals from the DAQ Signal Accessory, notice that Channel B is hardwired to DIO6, so the only connection that you need to make is from Channel A to the source of the counter you are using. Configure the counter for simple edge counting and set the active edge to falling. The count changes when a falling edge is received on Channel A, and Channel B determines if the count increments or decrements. When the encoder is turned clockwise, Channel A leads Channel B. When Channel A leads Channel B, the falling edge of Channel A occurs when Channel B is high, so the count increments. By the same logic, when the encoder is turned counter-clockwise, the count decrements. It is standard for clockwise motion to increment the count and counter-clockwise motion to decrement the count. If you would like to switch the two, you can set the active edge to rising instead of falling.

Refer to the *Using Quadrature Encoders with E Series DAQ Boards* Application Note for more information about using the DAQ-STC counter/timers to interface to quadrature encoders.

Exercise 9-8 Quadrature Encoder

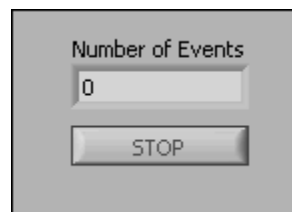
Objective: To count the pulses from the quadrature encoder on the DAQ Signal Accessory.

1. On the DAQ Signal Accessory, connect Channel A from the quadrature encoder to the SOURCE Input of Counter 0.

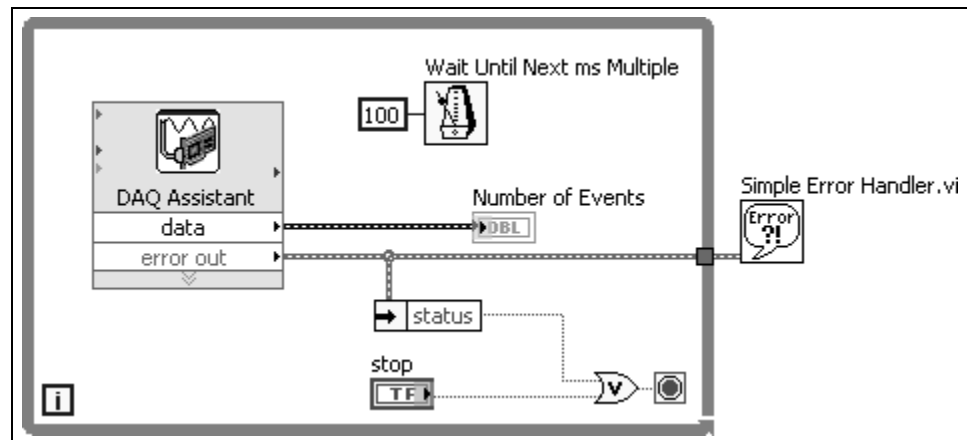


Note The up/down input for counter 0 is DIO6. Channel B on the DAQ Signal Accessory is hard-wired to DIO6. Therefore, no connection to the up/down input for counter 0 is necessary.

2. Open a blank VI and build the following front panel.



3. Save the VI as `Count Events.vi` in the `C:\Exercises\LabVIEW DAQ` directory.
4. Build the following block diagram.



5. Place the DAQ Assistant Express VI on the block diagram. Configure DAQ Assistant Express VI to use the counter to perform event counting.

- Select **Counter Input»Edge Count** for the measurement to make.
- Select **Dev1»ctr0** for the physical channel.
- Click the **Finish** button.
- Change the **Active Edge** pull-down menu to **Falling**.

- Change the **Count Direction** pull-down menu to **Externally Controlled**.

- Click the **OK** button to close the configuration dialog box.



6. Place the Wait until Next ms Multiple function on the block diagram. This function waits until the value of the millisecond timer becomes a multiple of the specified millisecond multiple. This function controls the While Loop execution rate. Set the millisecond multiple to 100.



7. Place the Unbundle by Name function on the block diagram.



8. Place the Or function on the block diagram.



9. Place the Simple Error Handler VI on the block diagram. In the event of an error, this VI displays a dialog box with information regarding the error and where it occurred.

10. Save the VI.

11. Run the VI. Rotate the quadrature encoder knob on the DAQ Signal Accessory. Notice that the Number of Events indicator increments as you rotate the knob. The quadrature encoder knob produces pulses as you rotate the knob. The counter counts these pulses.

Rotate the quadrature encoder knob in the other direction. Notice that the Number of Events indicator decrements when you rotate the knob clockwise, and increments when you rotate the knob counterclockwise.

If the VI does not work as you expect, you may need to reset the DAQ Device in Measurement and Automation Explorer.

12. Save and close the VI.

End of Exercise 9-8

Summary

- Counters accept and generate TTL signals.
- The main components of a counter are the source, gate, out, and count register.
- NI devices can have one of three different counter chips:
 - DAQ-STC (24-bit)— E Series devices
 - NI-STC 2 (32-bit)— M Series devices
 - NI-TIO (32-bit)—660x devices
- You can use the Easy Counter VIs to perform edge counting, pulse generation, pulse measurement, and frequency measurement.
- A quadrature encoder is a transducer that converts rotary motion into two pulse trains that are out of phase by 90 degrees.