Logic Functions for Coincidence Logic Unit

User Manual

Version 1.5

UQDevices

Universal Quantum Devices

295 Hagey Boulevard, 1st Floor, West Entrance

Waterloo, ON, Canada N2L 3G1

Phone: +1 519 404 6844 Email: info@uqdevices.com Url: www.uqdevices.com

Empowered by

dotfast-consulting.at

Thomas Lehner, Dotfast Consulting

Table of Contents

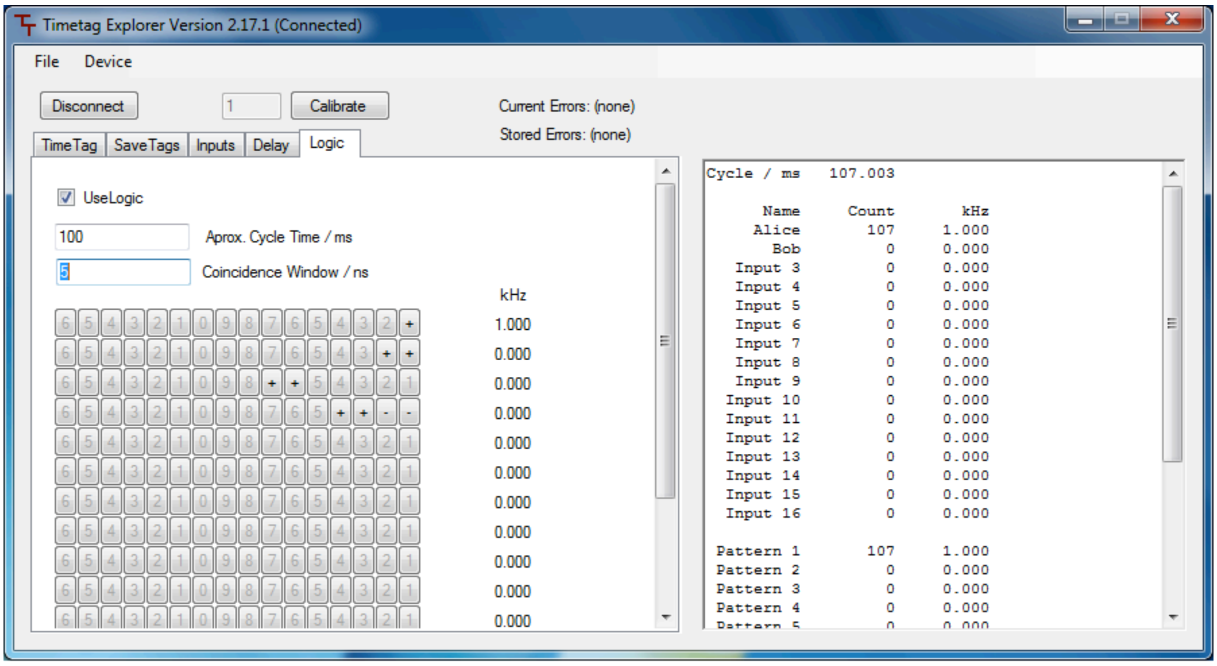
Revision History

Date	Document Version	FPGA Version	DLL Version	Remark
14.08.10	1.0			Initial revision
19.05.11	1.1		2.14	Put logic functions into separate dll Definition of SetOutputPattern() changed
26.1.12	1.2			Defined output delay
22.3.12	1.3			Add information about Timetag Explorer
09.10.13	1.4	2.17	2.17	Update information about new drivers
19.02.14	1.5	2.17.3	2.17.1	Add description of standard implementation and LabView and Visual Basic code.

Timetag Explorer Interface

Logic Tab

This tab is only available with the UQD devices that include the special “Logic” feature. It allows testing and studying of some of the coincidence features of the logic, like measuring single and coincident detections from arbitrary channels, and adjusting delays between channels. The output on the right side gives the single count rates per cycle time and in thousands per second (`kHz`) for each `Input` channel and, below, the count rates for the arbitrary coincidence `Patterns`.



Current Errors: (none)
Stored Errors: (none)

[✓] UseLogic
100 Aprox. Cycle Time / ms
5 Coincidence Window / ns

```
                                                      kHz
6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 +                       1.000
6 5 4 3 2 1 0 9 8 7 6 5 4 3 + +                       0.000
6 5 4 3 2 1 0 9 8 + + 5 4 3 2 1                       0.000
6 5 4 3 2 1 0 9 8 7 6 5 + + - -                       0.000
6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1                       0.000
6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1                       0.000
6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1                       0.000
6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1                       0.000
6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1                       0.000
6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1                       0.000
6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1                       0.000
```

```
Cycle / ms    107.003

     Name     Count     kHz
    Alice       107     1.000
      Bob         0     0.000
  Input 3         0     0.000
  Input 4         0     0.000
  Input 5         0     0.000
  Input 6         0     0.000
  Input 7         0     0.000
  Input 8         0     0.000
  Input 9         0     0.000
 Input 10         0     0.000
 Input 11         0     0.000
 Input 12         0     0.000
 Input 13         0     0.000
 Input 14         0     0.000
 Input 15         0     0.000
 Input 16         0     0.000

Pattern 1       107     1.000
Pattern 2         0     0.000
Pattern 3         0     0.000
Pattern 4         0     0.000
Pattern 5         0     0.000
```

Aprox. Cycle time

The software tries to read the data with this measurement time. Because of computer processing time, the actual cycle time is different. This time `Cycle / ms` is displayed on the right and changes slightly. Only the actual cycle time is used to calculate detection rates.

Coincidence Window

Here the width of the coincidence window is defined. This is the maximum time from the first detection to the last detection in the coincidence event. When all logically selected detections occur within that time window, then the event is counted. Otherwise the event is not counted.

Coincidence Patterns

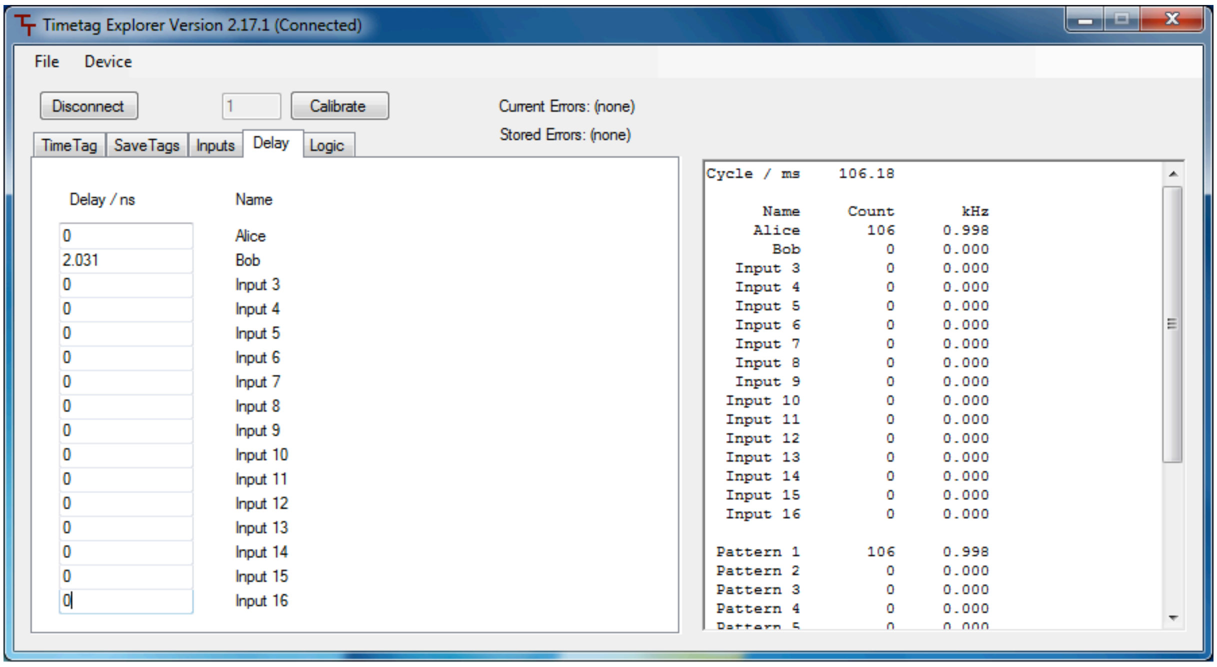
Each line of buttons defines one coincidence event. Clicking on the button changes the role of the corresponding input.

+ The input must have an active edge in the coincidence window for the event to be counted.

- The input must not have an active edge in the coincidence window for the event to be counted.

Delay Tab

This tab allows the user to define individual delays for each input channel.



Current Errors: (none)
Stored Errors: (none)

Delay / ns	Name
0	Alice
2.031	Bob
0	Input 3
0	Input 4
0	Input 5
0	Input 6
0	Input 7
0	Input 8
0	Input 9
0	Input 10
0	Input 11
0	Input 12
0	Input 13
0	Input 14
0	Input 15
0	Input 16

```
Cycle / ms    106.18

      Name     Count       kHz
     Alice       106     0.998
       Bob         0     0.000
   Input 3         0     0.000
   Input 4         0     0.000
   Input 5         0     0.000
   Input 6         0     0.000
   Input 7         0     0.000
   Input 8         0     0.000
   Input 9         0     0.000
  Input 10         0     0.000
  Input 11         0     0.000
  Input 12         0     0.000
  Input 13         0     0.000
  Input 14         0     0.000
  Input 15         0     0.000
  Input 16         0     0.000

 Pattern 1       106     0.998
 Pattern 2         0     0.000
 Pattern 3         0     0.000
 Pattern 4         0     0.000
 Pattern 5         0     0.000
```

This feature is very useful in the logic mode, where the delays of individual channels can be fine-tuned to maximize concidence signals.

Software Interface

The interface functions are located in the dll ttInterface.dll. There is also support for C++, both for Windows and Linux environments.

A normal implementation of the Logic mode proceeds as follows:

Create a TTInterface object, open the TTInterface using method `Open()`, then create a `Logic` object and call method `SwitchLogicMode()`. Next use the `TTInterface` object to get the time resolution of the Coincidence Logic Unit with `GetResolution()` and set the count thresholds for each channel with `SetInputThreshold()`. Back in the `Logic` object, set the delays for each channel with `SetDelay(int input, int delay)` and set the coincidence window with `SetWindowWidth(int window)`, both in units of the time resolution. The logical pattern of counts to be recorded should be set up also (see description under `CalcCount`). Now, in a repeating loop, call `ReadLogic()` to capture data and `CalcCount(int pos, int neg)` or `CalcCountPos(int pos)` to return the number of counts for each logical pattern. The loop can be delayed to count for a certain time interval, and use `GetTimeCounter()` to find the actual counting time for each loop, which will be slightly longer than the set delay due to computer processing time. After the counting loop is complete, close the `TTInterface` object with `Close()` to end the session.

Initialization

```
public Logic(ttInterface t)
```

The constructor takes a `ttInterface` object which must be already created, and creates a `Logic` object. The `ttInterface` object can already be opened or not.

```
public void SwitchLogicMode()
```

This method must be called before the other logic functions can be used.

```
public void SetWindowWidth(int window)
```

This method sets width of the coincidence window. The parameter `window` is given in internal units (156.25 ps). The window width can be set as high as 2^{24}-1.

```
public void SetDelay(int input, int delay)
```

This method sets the virtual delay line for each channel. The parameter `delay` is given in internal units (156.25 ps).

The parameter `input` identifies the channel and ranges from 1 to 16.

Counter Read Out

```
public int[] ReadLogic()
```

This method reads the counter out of the device. The return parameter is not needed (used for debug purposes only).

The data is automatically stored in the `Logic` object for later processing.

This method can be called any time after calling `SwitchLogicMode()`. The time between calls to `ReadLogic()` defines the measurement time interval of the captured data.

The device uses a double buffered memory system. The new measurement starts immediately and is running in the background during the call to `ReadLogic()`. For this reason not a single pulse is omitted during readout.

```
public int CalcCount(int pos, int neg)
```

This method operates on the data read by the last call to ReadLogic(). It calculates how often a certain event pattern has occurred in the last measurement interval.

The parameters `pos` and `neg` are bit coded. The rightmost bit corresponds to input 1, the next to input 2, and so on.

The parameter `pos` is given by the integer value of the binary number defined by the bit coded Coincidence Event. It defines the inputs that must have an active edge in the coincidence window for the event to be counted, with range [0..65535].

The parameter `neg` defines the inputs that must have no active edge in the coincidence window for the pattern to be counted. The parameter `neg` is optional, with range [0..65535].

pos Coincidence Pattern	pos	neg	Coincidence Event
0000000000000001	1	0	Singles of input 1
0000000000000010	2	0	Singles of input 2
0000000000000100	4	0	Singles of input 3
0000000000001000	8	0	Singles of input 4
0000000000000011	3	0	Coincidence of input 1 and 2
0000000000000111	7	0	Coincidence of inputs 1, 2, and 3
0000000000000011	3	4	Coincidence of inputs 1 and 2 without interaction of input 3

```
public int CalcCountPos(int pos)
```

This method equals `CalcCount` except that the `neg` parameter is always 0. It has better run time.

```
public int GetTimeCounter()
```

The time counter measures the time between the last two calls to ReadLogic. The result is given in multiples of 5 ns.

The time when ReadLogic() is called will vary over time because of the limited realtime performance of personal computers. The correct count rates can be obtained when the count values from `CalcCount` are divided by the result from `GetTimeCounter()`.

Outputs

```
public void SetOutputWidth(int width)
```

This function defines the length of the generated output pulses. The parameter is given in 5 ns increments. The maximum value is 255.

```
public void SetOutputPattern(int output, int pos, int neg)
```

This function sets the pattern of the output pulses based on the input coincidence events.

The parameter `output` identifies which output to change, with range [1..3].

The parameter `pos` is the bit coded value of the Coincidence Event of the inputs that must be present, with range [0..65535] as for `CalcCount`.

The parameter `neg` is the bit coded value of the Coincidence Event of the inputs that must *not* be present, with range [0..65535].

Method call	Result
`SetOutputPattern(1, 1, 0)`	Output 1 fires when input 1 is present
`SetOutputPattern(2, 6, 0)`	Output 2 fires when inputs 2 and 3 are present within the coincidence window
`SetOutputPattern(3, 6, 8)`	Output 3 fires when inputs 2 and 3 are present and input 4 is not present at the same time

```
public void SetOutputEventCount(int events)
```

This function is an advanced feature that can be used to fine tune the delay of the outputs. Roughly speaking, `events` is the number of events you expect to occur in one 5 ns time slice under worst-case conditions.

Increasing the value by one will increase the delay of the outputs by 10 ns. With the standard setting of 5 the output delay will be 350 ns.

When the input rate is too high for the given event count, the `OutTooLate` Error Flag will be raised. No output pulse will be generated in this condition. There is never an output pulse generated in the wrong timing. For this reason `OutTooLate` is more a kind of warning, not a hard error condition. It just means that not all pulses are generated.

Hint: When two events occur at the exact same time, the event coming from the input with the smaller input number will be processed first. For this reason input 1 has a slightly better real time performance than input 16.

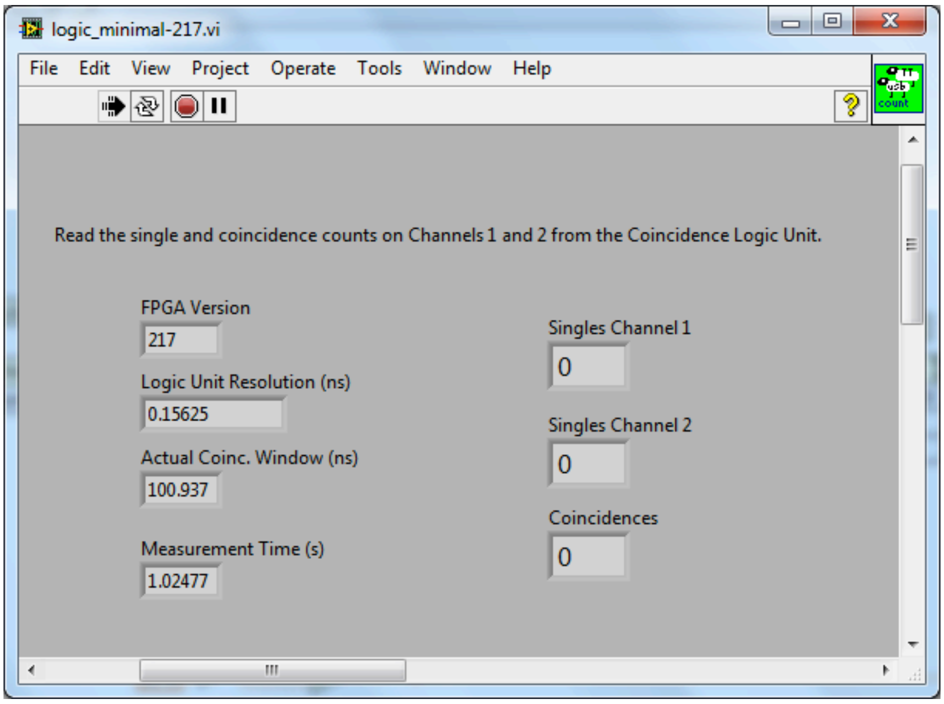
Minimal implementations of logic functions

These implementations consist of a small but useful script that opens the Coincidence Logic Unit, sets some parameters, and reads once the single and coincidence counts on channels one and two.

Visual Basic

```
Imports TimeTag

Public Class Form1

    Dim myTagger As TTInterface
    Dim mylogic As Logic
    Dim Resolution As Double
    Dim Errors As Integer


    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
        Dim Singles1, Singles2, Coincidences As Integer
        Dim FPGA_v, TimerCounter As Double

        'open connection to device
        myTagger = New TTInterface
        myTagger.Open()
        'get some basic info and display
        Resolution = myTagger.GetResolution()
        FPGA_v = myTagger.GetFpgaVersion()
        MsgBox("FPGA Version: " & FPGA_v.ToString & vbCrLf & _
               "Timing Resolution: " & Resolution.ToString & " s" & vbCrLf
               & vbCrLf & vbCrLf & _
               vbTab & vbTab & "Start Measurment Now?")

        'set input thresholds for channel 1 and 2
        myTagger.SetInputThreshold(1, 0.3)
        myTagger.SetInputThreshold(2, 0.3)

        'initiate logic mode
        mylogic = New Logic(myTagger)
        mylogic.SwitchLogicMode()

        'conicidence window (in units of 156ps)
        mylogic.SetWindowWidth(20)

        'count events for  1 sec:
        'take a first snapshot of counters
        mylogic.ReadLogic()

        'delay thread for about 1sec
        Threading.Thread.Sleep(1000)

        'take second snapshot of counters
        mylogic.ReadLogic()

        'how long was the true measurement time between the two ReadLogic
               comands? Given in multiples of 5ns, therfore we multiply by
               this unit to obtain sec
        TimerCounter = mylogic.GetTimeCounter() * 0.000000005

        'retrive observed counts for singles 1,2 and  coincidences 12
        singles1 = mylogic.CalcCountPos(1)
        singles2 = mylogic.CalcCountPos(2)
        coincidences = mylogic.CalcCountPos(3)

        'show  data
        MsgBox("Measurment Time: " & vbTab & vbTab &
TimerCounter.ToString("F4") & " s" & vbCrLf & _
               "Singles Channel 1:" & vbTab & vbTab & Singles1.ToString &
                vbCrLf & _
               "Singles Channel 2:" & vbTab & vbTab & Singles2.ToString &
               vbCrLf & _
               "Coincidences Chs 1&2:  " & vbTab & Coincidences.ToString)

        Close()

    End Sub


    Private Sub Form1_FormClosing(sender As Object, e As
FormClosingEventArgs) Handles Me.FormClosing
        MsgBox("Thanks for testing the counter. Bye")

        ' close connection to device
        myTagger.Close()

    End Sub

      End Class
```

LabView



Read the single and coincidence counts on Channels 1 and 2 from the Coincidence Logic Unit.

FPGA Version: 217

Logic Unit Resolution (ns): 0.15625

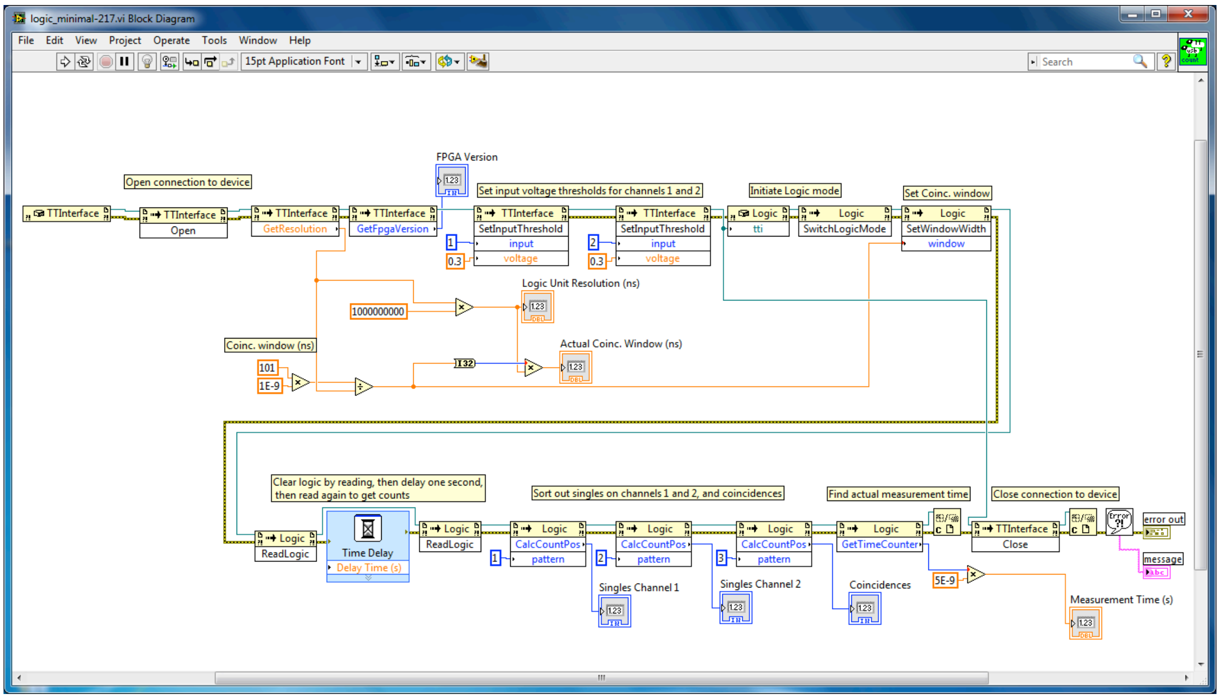
Actual Coinc. Window (ns): 100.937

Measurement Time (s): 1.02477

Singles Channel 1: 0

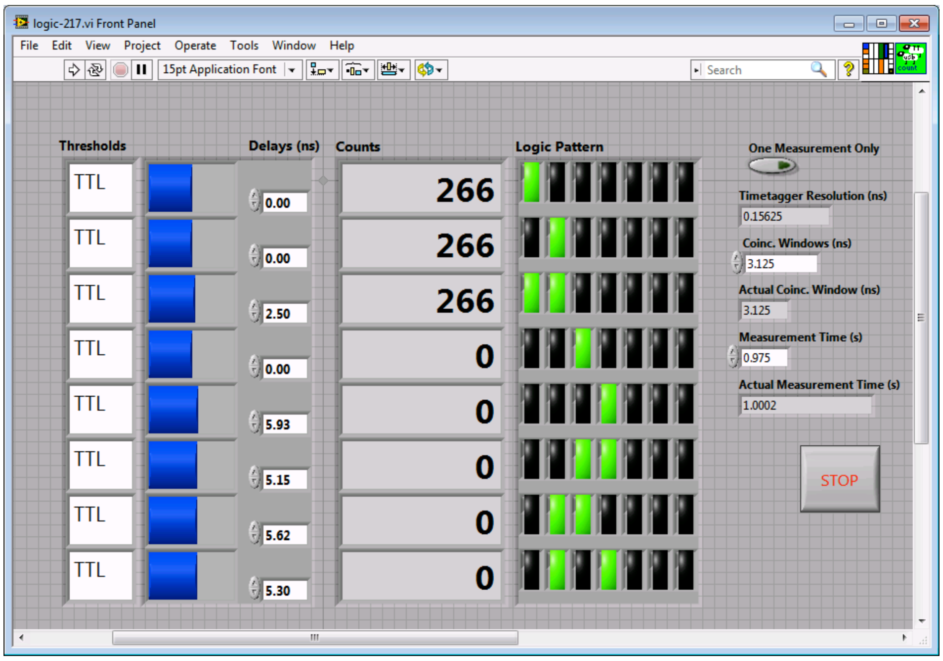
Singles Channel 2: 0

Coincidences: 0


logic_minimal-217.vi Block Diagram
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
Open connection to device
TTInterface
TTInterface Open
TTInterface GetResolution
TTInterface GetFpgaVersion
FPGA Version
Set input voltage thresholds for channels 1 and 2
TTInterface SetInputThreshold input voltage
1
0.3
2
0.3
Initiate Logic mode
Logic tti
Logic SwitchLogicMode
Set Coinc. window
Logic SetWindowWidth window
Logic Unit Resolution (ns)
1000000000
Actual Coinc. Window (ns)
I32
Coinc. window (ns)
101
1E-9
Clear logic by reading, then delay one second, then read again to get counts
Logic ReadLogic
Time Delay
Delay Time (s)
Sort out singles on channels 1 and 2, and coincidences
Logic CalcCountPos pattern
1
2
3
Singles Channel 1
Singles Channel 2
Find actual measurement time
Logic GetTimeCounter
Coincidences
5E-9
Close connection to device
TTInterface Close
Measurement Time (s)
error out
message

Multi-Channel LabView Implementation of logic functions

This description is based on the LabView VI **logic-*ver*.vi** (current *ver* is 217) and sub-VI **generate logic pattern2.vi**. Below is the standard front panel.



Thresholds	Delays (ns)	Counts	Logic Pattern
TTL	0.00	266	🟩⬛⬛⬛⬛⬛⬛
TTL	0.00	266	⬛🟩⬛⬛⬛⬛⬛
TTL	2.50	266	🟩🟩⬛⬛⬛⬛⬛
TTL	0.00	0	⬛⬛🟩⬛⬛⬛⬛
TTL	5.93	0	⬛⬛⬛🟩⬛⬛⬛
TTL	5.15	0	⬛⬛🟩🟩⬛⬛⬛
TTL	5.62	0	⬛🟩🟩⬛⬛⬛⬛
TTL	5.30	0	⬛🟩⬛🟩⬛⬛⬛

One Measurement Only

Timetagger Resolution (ns): 0.15625

Coinc. Windows (ns): 3.125

Actual Coinc. Window (ns): 3.125

Measurement Time (s): 0.975

Actual Measurement Time (s): 1.0002

Thresholds

Thresholds allows the voltage threshold to be set for each channel. Two standard voltages are NIM and TTL.

Delays

Delays allows the delay for each channel in nanoseconds to be set, in order to align coincident counts between channels. The delays can only be set to multiples of the Coincidence Logic Unit’s resolution.

Counts

Counts displays the number of events in the last Measurement Time corresponding to a given logic pattern.

Logic Pattern

Logic Pattern is the pattern of coincidences to report in counts. Channel 1 is the leftmost button and channel 6 is the rightmost (expand the control to see more channels). Green means an event on the channel must be present, red means an event on the channel must not be present, and black means to ignore the channel. For example, the first three patterns in the above screenshot are for single counts on channel 1, single counts on channel 2, and coincident counts between channels 1 and 2, respectively.

STOP

Always use the **STOP** button, rather than LabView’s Abort Execution so that the connection to the Coincidence Logic Unit is closed correctly.

Other settings

One Measurement Only, when enabled, returns the counts for only one Measurement Time, and then stops execution (otherwise measurements run continuously). This is useful, for example, when changing settings between each measurement.

Timetagger Resolution reports the resolution of the Coincidence Logic Unit in nanoseconds.

Coinc. Window controls the time window in nanoseconds within which counts on different channels are counted as coincident.

Actual Coinc. Window reports the coincidence window actually used in nanoseconds, which must be a multiple of the Coincidence Logic Unit's resolution.

Measurement Time chooses the time in seconds between reading the Coincidence Logic Unit's buffer, meaning Counts are displayed as summed over the Measurement Time.

Actual Measurement Time reports the real time in seconds between reading the Counts, due to finite processing time of the PC.