Pleora Technologies

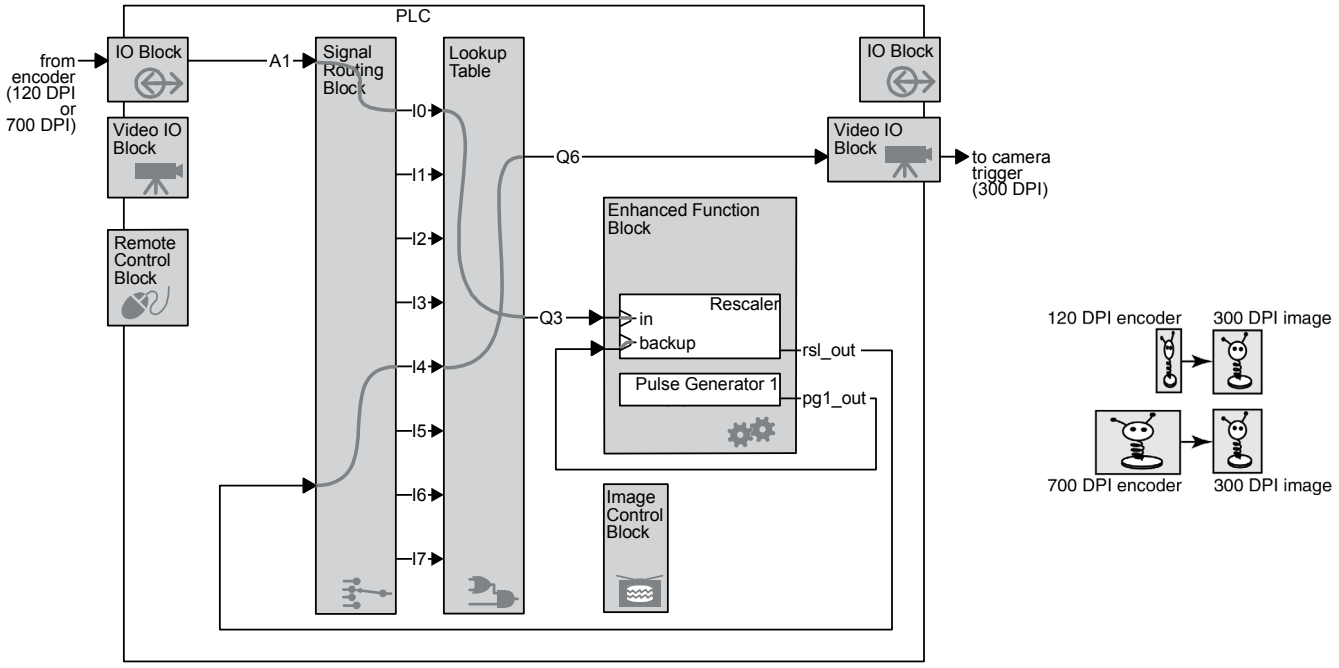
iPORT

Programmable Logic Controller

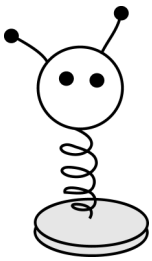
Reference Guide

SIMPLY | CONNECT™

Pleora Technologies



Rescaling a conveyor belt's encoder signal to capture exactly 300 DPI with a line-scan camera (page 70)



...the industry's richest feature set

Version 2.4

2/25/08

Contents

About 7
Understanding the PLC 9
Understanding the PLC workflow 9
Understanding the PLC wiring 10
Understanding the diagrams 11
Understanding the PLC icons 12
What now? 12
Demonstrating the PLC in 10 minutes 13
Monitoring your PLC signals with LED indicator 13
Monitoring your PLC signals with a feedback loop 15
Demonstrating the Remote Control Block 16
Demonstrating the Lookup Table 16
Demonstrating the Pulse Generator (Periodic) 17
Demonstrating the Pulse Generator (Triggered) 18
Demonstrating the Delayer 19
Demonstrating the Rescaler 20
Demonstrating the Rescaler (with backup) 21
Demonstrating the General Purpose Counter 23
Demonstrating the Interrupt FIFO 24
Configuring the PLC with the iPORT Vision Suite 27
Configuring the PLC with the SDK 27
Configuring the PLC with Coyote 27
Configuring the PLC with the iPORT PureGEV Suite 31
IO Block 31
Video IO Block 31
Remote Control Block 32
Signal Routing Block 32
Lookup Table 32
Enhanced Function Block 33
Special PLC settings 37
IO Block 39
Monitoring the state of the IO Block 40
Synchronization Block 40
Input Debouncing Block 41
TTL Input Block 42
TTL Output Block 43
LVDS Input Block 43
Optically Isolated Input Block 44
Optically Isolated Output Block 45
Video IO Block 47
How the Video IO Block works 47
Video IO Block (for Camera Link imaging data) 48
Video IO Block (for analog video) 48
Video IO Block (for LVDS imaging data) 49
Remote Control Block 51

Signal Routing Block **53**
How the Signal Routing Block works 53
Lookup Table **55**
How the Lookup Table works 56
Enhanced Function Block **57**
Overview of the Enhanced Function Block 58
Pulse Generator 59
Rescaler 60
Delayer 61
General Purpose Counter 62
Interrupt FIFO 64
Counter Trigger Generator 64
Timestamp Counter 65
Image Control Block **67**
Real world samples **69**
Triggering an area-scan camera with a presence detector 69
Acquiring a 300 DPI image with a line-scan camera 70
Glossary of signal names **73**
The PLC at a glance **75**
Programmable Logic Controller 75
Enhanced Function Block 76
PLC and Enhanced Function Block (on one page) 77

About

The iPORT™ Programmable Logic Controller (PLC) is a powerful tool that lets you control external machines, react to inputs, and create a machine vision system. By controlling your system using the PLC, you can make functional changes, adjust timing, or add features without having to add any new hardware.

Don’t read this book straight through!

To learn the PLC effectively (and hands on), we recommend starting with “Understanding the PLC” on page 9, followed by “Demonstrating the PLC in 10 minutes” on page 13. Each demonstration highlights a particular aspect of the PLC, and includes links to relevant sections in this guide. Once you finish the demonstrations, you’ll have a solid foundation for creating your own powerful, customized system.

Using the PLC with Coyote, the iPORT Vision Suite SDK, and the iPORT PureGEV Suite

The *iPORT Programmable Logic Controller Reference Guide* is written as generally as possible and the high-level concepts are always identical for all IP Engines. Specific names, directions, and screenshots are taken from iPORT Coyote.

The PLC features map one-to-one with the iPORT Vision Suite SDK; for the list of parameters, see “CyDeviceExtensionConstants.h” in the *iPORT C++ SDK Reference Guide*.

For GEV-enabled IP Engines, the PLC can vary dramatically, depending how it’s been configured (AutoGEV lets can hide or remap features). However, by default, the features are very similar. To convert the Coyote-centric names for your GEV-enabled IP Engine, use the tables in “Configuring the PLC with the iPORT PureGEV Suite” on page 31.

8 About

Understanding the PLC

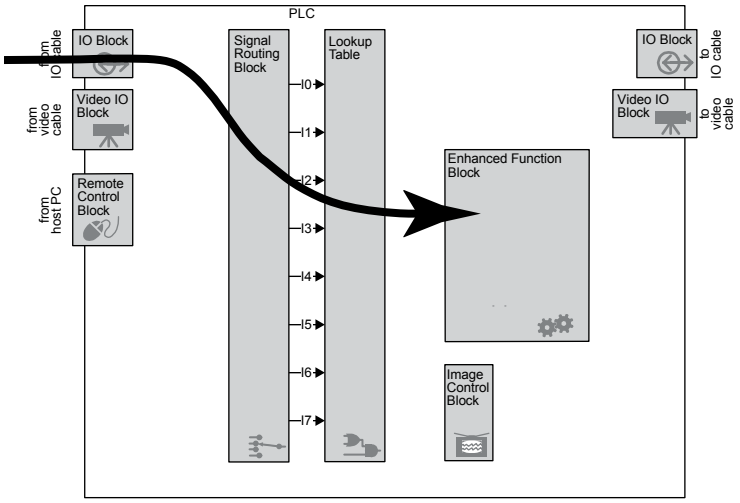
Once you understand the overall concept of how the PLC works, programming the PLC will quickly become natural to you.

In this section:

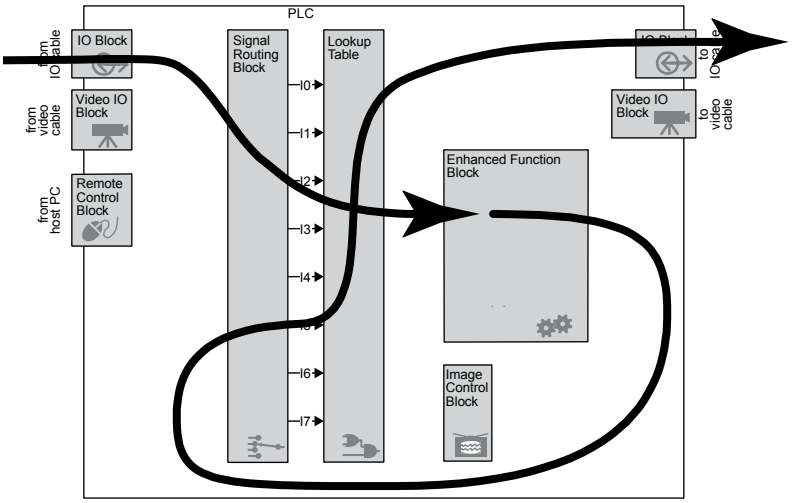
Understanding the diagrams 11
Understanding the PLC wiring 10
Understanding the PLC icons 12
What now? 12

Understanding the PLC workflow

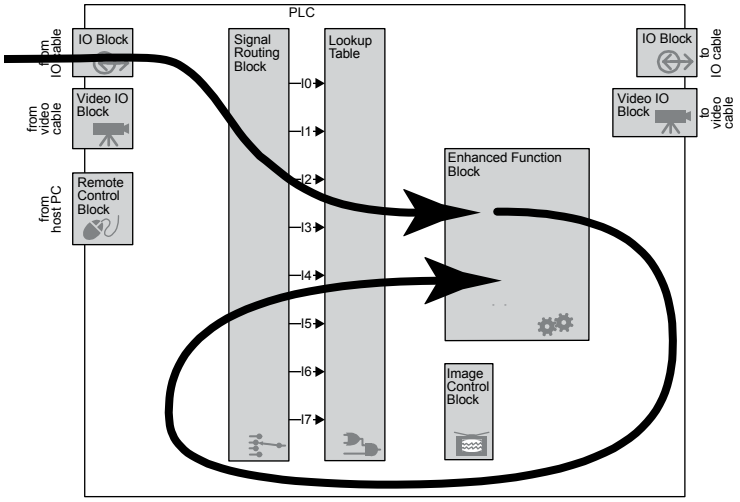
In general, you route signals from the inputs on the left to the Enhanced Function Block. The Enhanced Function Block has many features that let you manipulate your signals.



Once you manipulate your signal, you can route it to the outputs on the right. However, you must route the signal back through the Signal Routing Block and the Lookup Table.

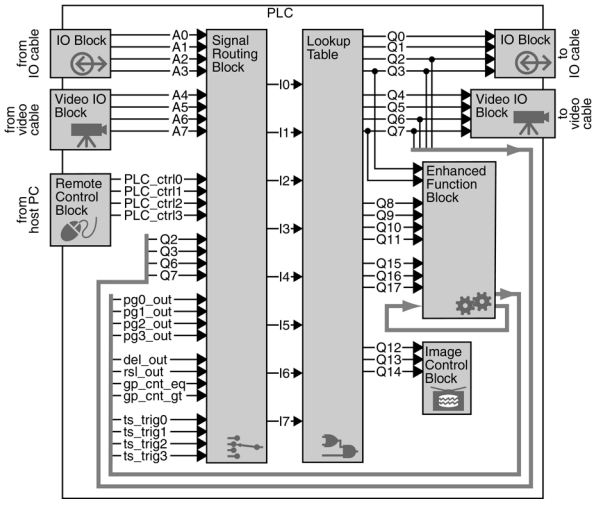


If you're not finished processing the signal, you can route it back to the Enhanced Function Block again.

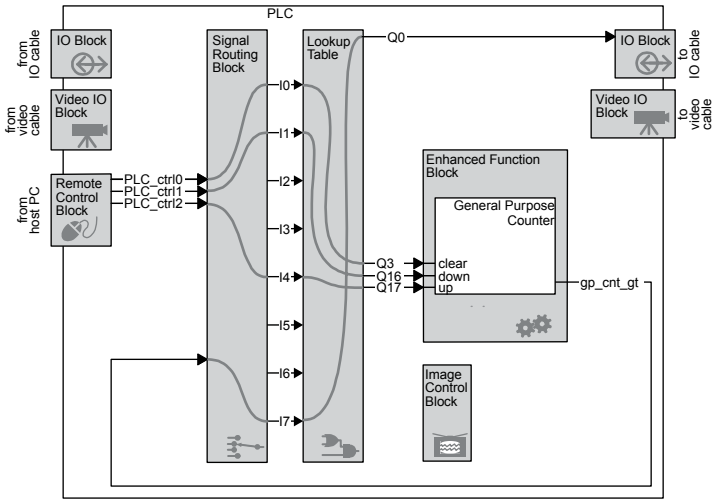


Understanding the PLC wiring

The wiring within the PLC is fixed; you control what happens in the gray blocks.

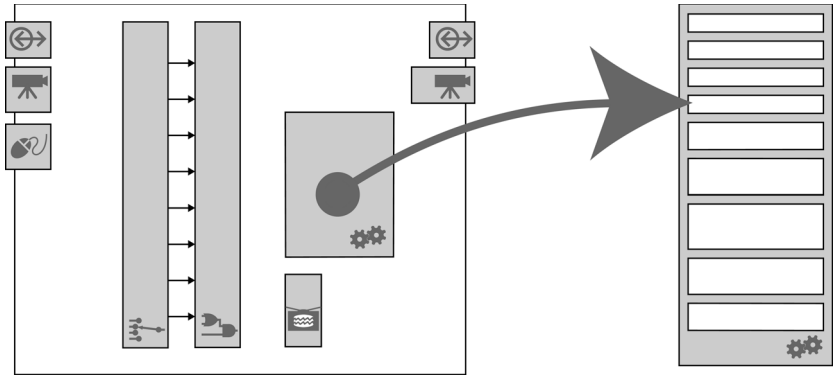


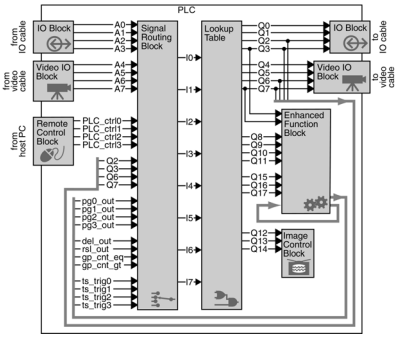
By reconfiguring the gray blocks, you can manipulate and reroute signals. The sample below is a counter taken from “Demonstrating the General Purpose Counter” on page 23. For clarity, the unused wiring isn’t shown.

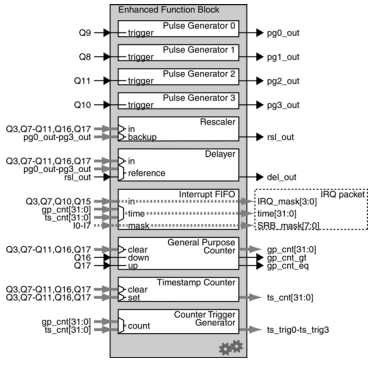


Understanding the diagrams

The PLC diagram includes a simplified version of the Enhanced Function Block.







For easier reference, you can find large versions of both diagrams in the back of this guide. See “The PLC at a glance” on page 75.

Understanding the PLC icons

To help you distinguish blocks from each other, each one features an icon that provides a distinct visual cue and suggests its function. You can find the icon at the bottom of every block.

IO Block

The IO Block lets you communicate between the PLC and the "outside world" through the IP Engine's 12-pin IO connector. See "IO Block" on page 39.

Video IO Block

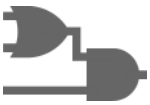
The Video IO Block lets you communicate between the PLC and the camera through the IP Engine's video connector. The block receives a video signal and provides the PLC with discrete frame valid (FVAL), line valid (LVAL), and similar signals. Some cameras let you send control signals back to the camera. See "Video IO Block" on page 47.

Remote Control Block

The Remote Control Block lets you send inputs from your host PC to the PLC. The PLC control bits let you simulate inputs and test your configuration without having to connect external components to your IP Engine. You can use the Remote Control Block to simulate inputs from your encoders, cameras, presence detectors, and other components. Once you're confident that your system works (at your desk), you can cable everything together, change a few configurations, and test your finished design (on site). You can also control this block from the iPORT SDK. See "Remote Control Block" on page 51.

Signal Routing Block

In its simplest terms, the Signal Routing Block is a group of switches that let you route signals to the Lookup Table (described next). See "Signal Routing Block" on page 53.



Lookup Table

The Lookup Table lets you route signals within the PLC, as well as your signals using simple or complex Boolean expressions. Using the Signal Routing Block and the Lookup Table, you can route signals *from* almost anywhere *to* almost anywhere. See "Lookup Table" on page 55.

Enhanced Function Block

The Enhanced Function Block lets you perform complex functions on your signals. You can delay a signal, generate a pulsed signal, count pulses, generate interrupts to the PC, and more. See "Enhanced Function Block" on page 57.

Image Control Block

The Image Control Block lets you configure the image LVAL, FVAL, and TRIG signals used by the IP Engine's image grabber. You can use the signals exactly as they come from the camera or you can manipulate them to create your own signal. This block is especially important if you're acquiring images using a line-scan camera. See "Image Control Block" on page 67.

What now?

Now that you've seen an overview of the PLC, we recommend that you follow the fun demonstrations; see "Demonstrating the PLC in 10 minutes" on page 13. From there, you can read the rest of this guide as you require (to get precise details about how various blocks work).

Demonstrating the PLC in 10 minutes

The following demonstrations let you learn how the PLC works using brief tutorials. Each demonstration shows a particular feature in the PLC and builds on the previous demonstration. Running through all of them will give you a good understanding of how to use the PLC.

Once you've set up your system, the demonstrations really will take under 10 minutes each but for better understanding, consider reading the background for each demonstration.

The demonstrations use an LED indicator that lets you see the results of your configurations. Though the PLC is lightning fast, the demonstrations use low frequency signals that can be seen with the human eye (and shown with a regular LED).

To run a demonstration:

1. Attach an output indicator to your IP Engine. We recommend the LED indicator described in "Monitoring your PLC signals with LED indicator" on page 13. However, you can also use the feedback loop described in "Monitoring your PLC signals with a feedback loop" on page 15.
2. Cable your IP Engine to your PC and connect to it using Coyote. See the *iPORT Quick Start Guide*.
3. Follow the procedure for a demonstration. We recommend doing them in order, starting from "Demonstrating the Remote Control Block" on page 16. To configure the PLC or use the PLC control bits, see "Configuring the PLC with the iPORT Vision Suite" on page 27.
4. Continue trying demonstrations.
   Have fun and feel free to experiment on your own!

In this section:

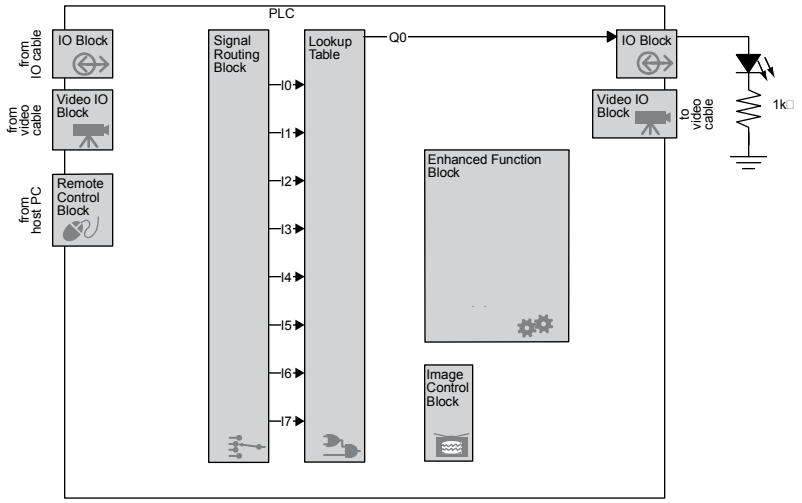
Monitoring your PLC signals with LED indicator 13
Monitoring your PLC signals with a feedback loop 15
Demonstrating the Remote Control Block 16
Demonstrating the Lookup Table 16
Demonstrating the Pulse Generator (Periodic) 17
Demonstrating the Pulse Generator (Triggered) 18
Demonstrating the Delayer 19
Demonstrating the Rescaler 20
Demonstrating the Rescaler (with backup) 21
Demonstrating the Interrupt FIFO 24

Monitoring your PLC signals with LED indicator

Background information: see "IO Block" on page 39.

When you're configuring the PLC, having an output indicator lets you confirm that the outputs you generate in the PLC are correct (and actually happening). Output indicators also give you the tools you

need to test your setup, find problems, and experiment with the powerful features in the Enhanced Function Block.



The LED indicator turns on when the TTL_OUT0 is high. In general, the LED will typically let you see frequencies up about 30 Hz, after which the LED will appear to be continuously set at a particular intensity. (This is a function of the human eye's ability to register changes.)

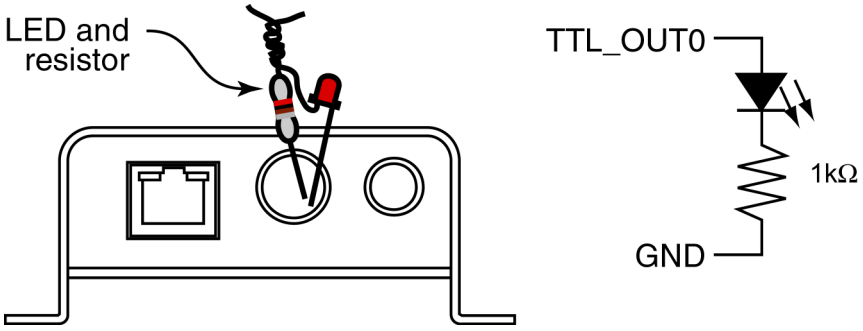
NOTE! Improper insertion of the wires may cause mechanical damage. For a better connection with a lower risk of electrical or mechanical damage to the IP Engine, solder your circuit to a Hirose mating connector.

NOTE! Incorrect wiring or accidentally shorting a connection to chassis ground can damage the IP Engine. Always disconnect the IP Engine's power before making connections and confirm the circuit before using it.

NOTE! The pinouts vary by IP Engine model. Consult your hardware guide to confirm the pinouts for your particular model.

To create an LED indicator:

1. Inspect your iPORT IP Engine for the model number (e.g. PT1000-CL4, etc.)
2. Consult the corresponding hardware guide.
3. In the hardware guide, note the IO pinouts for TTL_OUT0 and TTL_IN0.
4. Turn off the IP Engine.
5. Connect the LED and resistor to the TTL_OUT0 and the GND pins. The resistance value is a suggestion; ensure the value you use is appropriate for your LED and doesn't exceed the rated output current for the IP Engine's TTL_OUT0 signal.



6. Confirm the circuit is correct.

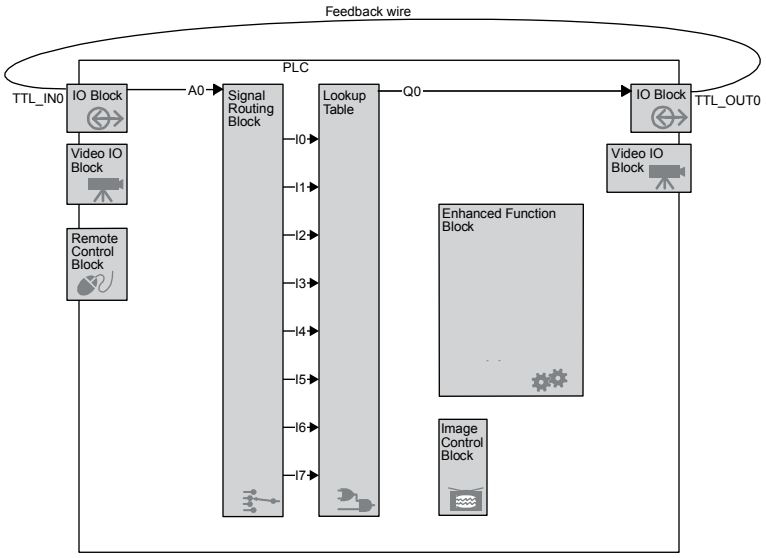
To view activity on TTL_OUT0:

- Watch for the LED to turn on, turn off, or flicker. Fast signals may produce a very faint light or seem to be continuously on, so you may have to interpret the result.

Monitoring your PLC signals with a feedback loop

Background information: see “IO Block” on page 39.

A feedback loop lets you reroute an output signal back into the PLC. You can monitor the signals using Coyote.



The feedback loop sends the TTL_OUT0 signal back into the PLC where you can see it using the **PLC Control Bits** dialog. Though the PLC control bits are accurate within the IP Engine, Coyote only samples the values roughly every 200 ms. Thus, you may be able to see frequencies up to 2-2.5 Hz, and should consistently see state changes on a signals 1 Hz or slower.

Consider using the LED indicator instead. Though the feedback loop is slightly easier to implement, the LED indicator shows state changes more accurately (and is more fun). See “Monitoring your PLC signals with LED indicator” on page 13.

For precise monitoring, connect your IP Engine’s TTL_OUT0 signal to an oscilloscope.

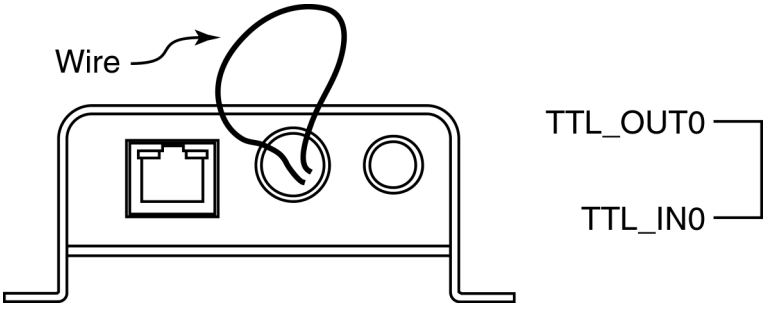
(Also see the notes in “Monitoring your PLC signals with LED indicator” on page 13.)

To create a feedback loop:

1. Inspect your iPORT IP Engine for the model number (e.g. PT1000-CL4, etc.)
2. Consult the corresponding hardware guide.
3. In the hardware guide, note the IO pinouts for TTL_OUT0 and TTL_IN0.
4. Cut a ~75 mm [3”] piece of 24 AWG solid core wire. Strip 5 mm [1/4”] of insulation from each end.

NOTE! For a more reliable feedback loop, solder a IO connector that cables (shorts) the TTL_OUT0 and TTL_IN0 pins together.

5. Turn off the IP Engine.
6. Using the wire, connect TTL_OUT0 and TTL_IN0.



7. Confirm that the circuit is correct.

To view activity on TTL_OUT0:

- See “Monitoring the IO Block” on page 27.

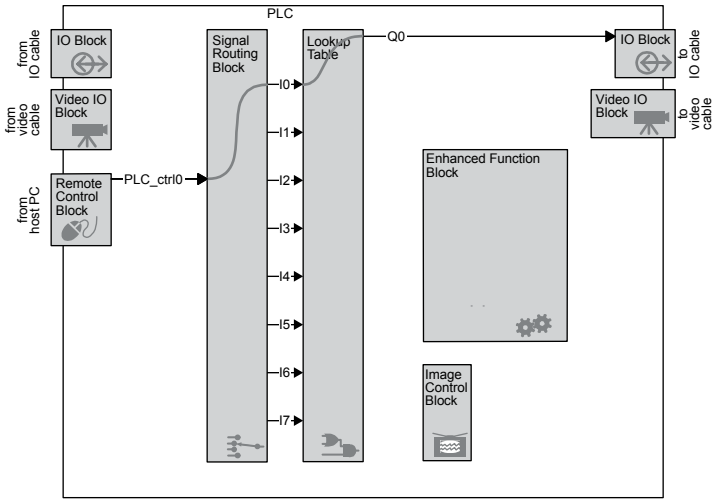
Demonstrating the Remote Control Block

Background information: see “Remote Control Block” on page 51.

This demonstration lets you directly control the state of the output by making changes to PLC_ctrl0. You can watch the output change with each mouse click.

Procedure

1. Create an output indicator. See “Monitoring your PLC signals with LED indicator” on page 13.
2. In the **IP Engine** tab, make the following configurations and click **Apply**.



Signal Routing Block
I0: PLC Control Bit 0

Lookup Table
Q0=I0

3. Control the circuit using PLC_ctrl0.

Results

When you set PLC_ctrl0 to true, the output is true. When you set it to false, the output is false.

Demonstrating the Lookup Table

Background information: see “Lookup Table” on page 55.

This demonstration uses the Lookup Table to perform Boolean operations on your inputs.

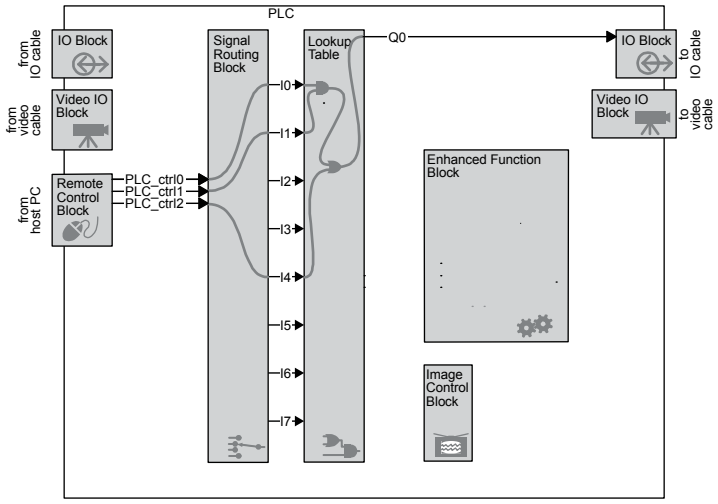
Your inputs PLC_ctrl0, PLC_ctrl1, and PLC_ctrl1 are output as a single signal after the following operation:

$$output = (PLCCTRL0 \cap PLCCTRL1) \cup PLCCTRL2$$

Procedure

1. Create an output indicator. See “Monitoring your PLC signals with LED indicator” on page 13.

2. In the **IP Engine** tab, make the following configurations and click **Apply.**



Lookup Table:

Q0=(I0&I1)|I4

Signal Routing Block:

I0: PLC Control Bit 0

I1: PLC Control Bit 1

I4: PLC Control Bit 2

3. Control the circuit using PLC_ctrl0, PLC_ctrl1, and PLC_ctrl2.

Results

When you set PLC_ctrl2 to true, the output is true. The output is also true if you set both PLC_ctrl0 and PLC_ctrl1 to true.

Demonstrating the Pulse Generator (Periodic)

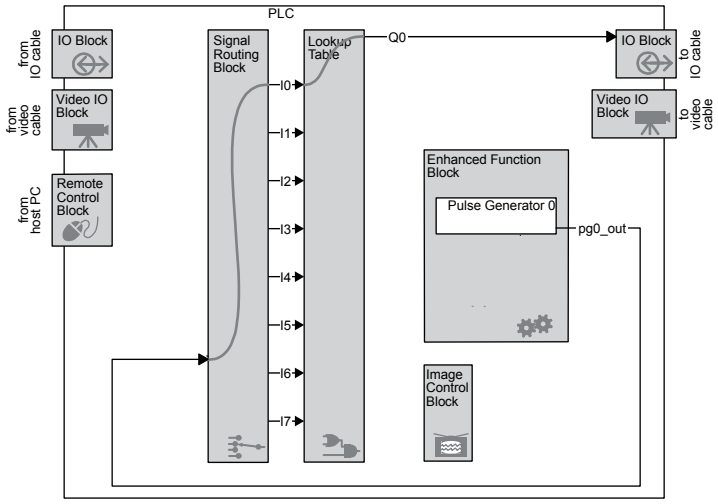
Background information: See “Overview of the Enhanced Function Block” on page 58 and “Pulse Generator” on page 59.

This demonstration uses a Pulse Generator to output a continuous 1 Hz pulse that is high for 0.25 s and low for 0.75 s.

Procedure

1. Create an output indicator. See “Monitoring your PLC signals with LED indicator” on page 13.

2. In the **IP Engine** tab, make the following configurations and click **Apply**.



Pulse Generator 0
- Width (high): 1000
- Delay (low): 3000
- Granularity factor: 8320
- Emit periodic pulse: True
- Trigger mode: NA
- Pulse frequency: ~1 Hz (read only)

Signal Routing Block
- I0: Pulse Generator 0 Output

Lookup Table:
- Q0=I0

Results

The PLC outputs a steady 1 Hz signal with a 25% duty cycle.

Demonstrating the Pulse Generator (Triggered)

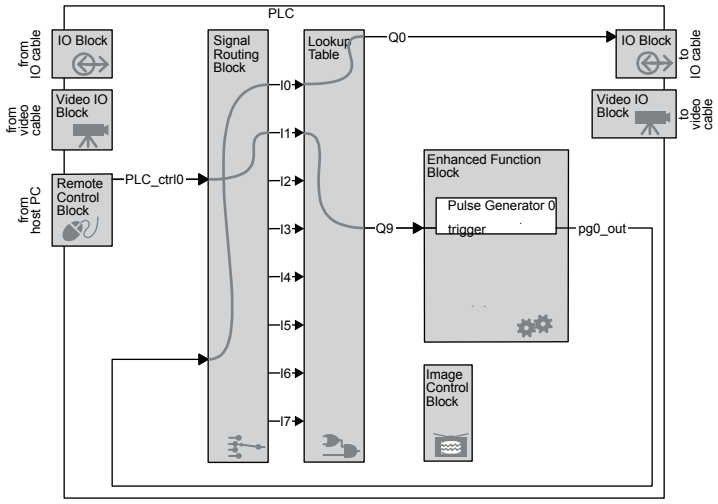
Background information: see “Pulse Generator” on page 59.

This demonstration uses a Pulse Generator to emit a single pulse at your command. The pulse is low for 1 s and high for 1 s. The pulse starts with the low, so you can incorporate a built-in delay, if you like.

Procedure

1. Create an output indicator. See “Monitoring your PLC signals with LED indicator” on page 13.

2. In the **IP Engine** tab, make the following configurations and click **Apply**.



Pulse Generator 0
- Width (high): 4000
- Delay (low): 4000
- Granularity factor: 8320
- Emit periodic pulse: False
- Trigger mode: Triggered on rising edge
- Pulse frequency: ~0.50 Hz (read only)

Signal Routing Block
- I0: Pulse Generator 0 Output
- I1: PLC Control Bit 0

Lookup Table
- Q0=I0
- Q9=I1

3. Control the circuit using PLC_ctrl0.

Results

When you set PLC_ctrl0 to true, Pulse Generator 0 emits a single pulse. Notice that the low is emitted first. Subsequent changes to PLC_ctrl0 are ignored until the single pulse is complete.

If you change the **Trigger mode** to **Triggered on high level**, the pulse is requested with every IP Engine system clock cycle. The requests are ignored until the pulse is complete. Thus, you see a continuous stream of complete pulses whenever PLC_ctrl0 is true.

Demonstrating the Delayer

Background information: see “Delayer” on page 61.

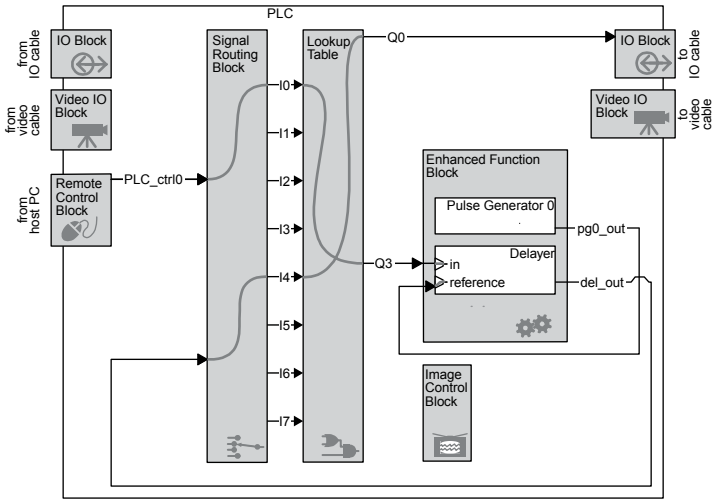
This demonstration uses the Delayer to delay a signal by about three seconds. Use PLC_ctrl0 to input the signal; your button clicks are output after a three second delay.

Pulse Generator 0 emits a ~23 Hz pulse that tells the Delayer how often to sample the input signal. The input signal comes from your manual changes to PLC_ctrl0. (You could increase the sampling accuracy by increasing the output frequency from Pulse Generator 0, but for manual inputs, 23 samples per second is more than adequate.)

Procedure

1. Create an output indicator. See “Monitoring your PLC signals with LED indicator” on page 13.

2. In the **IP Engine** tab, make the following configurations and click **Apply**.



Pulse Generator 0
- Width (high): 65535
- Delay (low): 65535
- Granularity factor: 10
- Emit periodic pulse: True
- Trigger mode: Not applicable (any setting is okay)
- Pulse frequency: ~23 Hz (read only)

Delayer 0
- Delay count: 69
- Reference timing signal: Pulse Generator 0 output
- Input signal: Q3

Signal Routing Block
- I0: PLC Control Bit 0
- I4: Delayer 0 output

Lookup Table
- Q3=I0
- Q0=I4

3. Control the circuit using PLC control bit 0.

Results

When you set PLC_ctrl0 to true, the output becomes true after a three second delay. When you set it false, the output becomes false after three seconds. The output will always accurately track your inputs to PLC_ctrl0.

Demonstrating the Rescaler

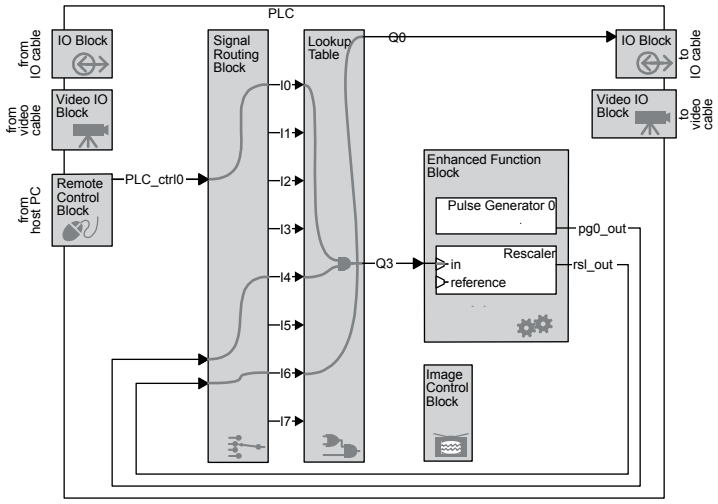
Background information: see “Rescaler” on page 60.

This demonstration uses the Rescaler to turn a 36 Hz pulse into a 3.6 Hz pulse. You can use PLC_ctrl0 to stop and start the 36 Hz input to the Rescaler.

Procedure

1. Create an output indicator. See “Monitoring your PLC signals with LED indicator” on page 13.

2. In the **IP Engine** tab, make the following configurations and click **Apply**.



Pulse Generator 0
- Width (high): 65535
- Delay (low): 65535
- Granularity factor: 6
- Emit periodic pulse: True
- Trigger mode: Not applicable (any setting is okay)
- Pulse frequency: ~36 Hz (read only)

Rescaler
- Granularity: 256 system clock cycles
- Multiplier: Frequency X 16
- Divider: 160
- Input signal: Q3
- Backup enabled: False
- Backup input signal: Not applicable (any setting is okay)
- Target frequency: Not applicable (any setting is okay)

Signal Routing Block
- I0: PLC Control Bit 0
- I4: Pulse Generator 0 output
- I6: Rescaler 0 output

Lookup Table
- Q0=I6
- Q3=I0&I4

3. Control the circuit using PLC_ctrl0.

Results

When PLC_ctrl0 is true, the 36 Hz signal from Pulse Generator 0 passes into the Rescaler and emerges as a 3.6 Hz signal. When PLC_ctrl0 is false, the 36 Hz signal is interrupted; the Rescaler's output may stops, and may be either high or low.

Demonstrating the Rescaler (with backup)

Background information: see "Rescaler" on page 60.

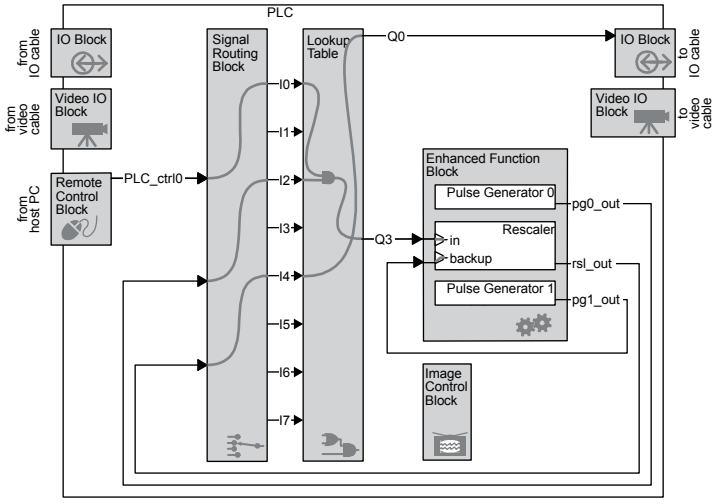
This demonstration uses the Rescaler to turn a 1001 Hz pulse into a 3.9 Hz pulse. However, if the 1001 Hz is interrupted, the Rescaler automatically outputs its backup pulse.

Pulse Generator 0 generates the 1001 Hz signal which the Rescaler uses as its primary input. The Rescaler divides the signal frequency by about 256 (times 16, divided by 4095) and outputs the result as a rescaled 3.9 Hz pulse.

Pulse Generator 1 generates a 1.1 Hz signal that the Rescaler uses as a backup. If you interrupt the primary input, the Rescaler outputs the backup signal without rescaling it. You can interrupt the primary signal by setting PLC_ctrl0 to false.

Procedure

1. Create an output indicator. See “Monitoring your PLC signals with LED indicator” on page 13.
2. In the **IP Engine** tab, make the following configurations and click **Apply.**



Pulse Generator 0
- Width (high): 10
- Delay (low): 5
- Granularity factor: 2080
- Emit periodic pulse: True
- Trigger mode: Not applicable (any setting is okay)
- Pulse frequency: ~1001 Hz (read only)

Pulse Generator 1
- Width (high): 100
- Delay (low): 500
- Granularity factor: 50000
- Emit periodic pulse: True
- Trigger mode: Not applicable (any setting is okay)
- Pulse frequency: ~1.1 Hz (read only)

Rescaler
- Granularity: 16 system clock cycles
- Multiplier: Frequency X 16
- Divider: 4095
- Input signal: Q3
- Backup enabled: True
- Backup switchover delay: 4095
- Backup input signal: Pulse Generator #1
- Target frequency: Not applicable (any setting is okay)

Signal Routing Block
- I0: PLC control bit 0
- I2: Pulse Generator 0 output
- I4: Rescaler 0 output

Lookup Table
Q3=I0&!I2
Q0=I4

3. Control the circuit using PLC_ctrl0.

Results

When PLC_ctrl0 is true, the PLC outputs a pulse that 3.9 Hz pulse. When PLC_ctrl0 is false, the output is a 1.1 Hz pulse.

The **Backup switchover delay** configures how long the Rescaler waits before switching to the backup signal. Configure this setting carefully. If you set this value too low (a value of 10), the Rescaler switches immediately to the backup signal because the wait is shorter than the input frequency. A value of 100 causes the Rescaler to switch back and forth.

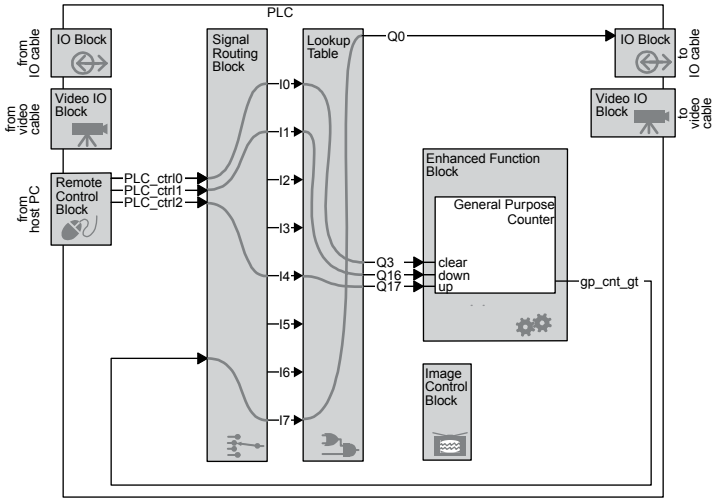
Demonstrating the General Purpose Counter

Background information: see “General Purpose Counter” on page 62.

This demonstration uses the General Purpose Counter to keep a continuous count. You can use the PLC control bits to clear, decrement, or increment the count. When the count is greater than 4, the output is true.

Procedure

1. Create an output indicator. See “Monitoring your PLC signals with LED indicator” on page 13.
2. In the **IP Engine** tab, make the following configurations and click **Apply**.



Counter
Increment trigger mode: Rising edge
Decrement trigger mode: Rising edge
Clear trigger mode: Rising edge
Clear signal: Q3
Compare value: 4

Signal Routing Block
I0: PLC control bit 0
I1: PLC control bit 1
I4: PLC control bit 2
I7: Counter 0 Greater

Lookup Table
Q3=I0
Q16=I1

Q17=I4
Q0=I7

3. Control the circuit using PLC_ctrl0, PLC_ctrl1, and PLC_ctrl2.

Results

Use PLC_ctrl0 to clear the count value, PLC_ctrl1 to decrement it, and PLC_ctrl2 to increment it. If you decrement the count below 0, the value wraps around to 4294967295 (which is greater than 4). The control use rising edges, so two mouse clicks are required to make a change.

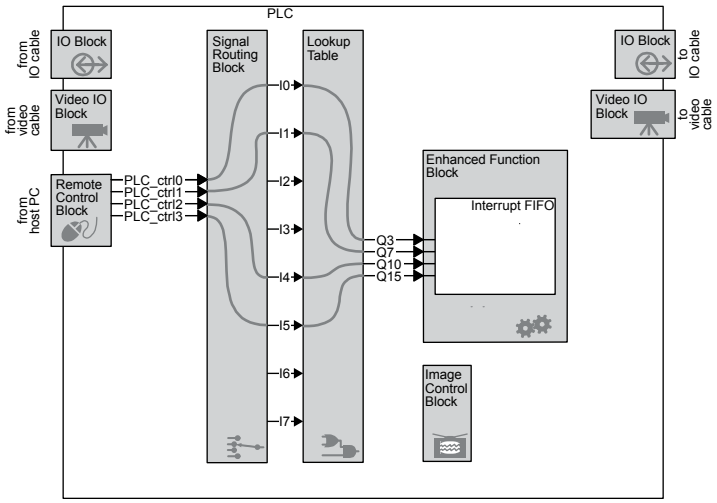
Demonstrating the Interrupt FIFO

Background information: see “Interrupt FIFO” on page 64.

This demonstration sends interrupt requests to your PC. The interrupts aren’t configured to do anything, but you can use Coyote to track them. This demonstration doesn’t use an output indicator, so the procedure differs slightly.

Procedure

1. Select the **Acquisition** tab.
2. Click **Configure**.
3. Move the **Configuration** dialog so that the **Acquisition** tab is visible.
4. In the **IP Engine** tab, make the following configurations and click **Apply**.



Signal Routing Block
I0: PLC Control Bit 0
I1: PLC Control Bit 1
I4: PLC Control Bit 2
I5: PLC Control Bit 3

Lookup Table
Q3=I0
Q7=I1
Q10=I4
Q15=I5

GPIO Interrupts
Q15 Enabled: Yes
Q3 Enabled: Yes
Q7 Enabled: Yes
Q10 Enabled: Yes

5. Control the circuit using PLC_ctrl0, PLC_ctrl1, PLC_ctrl2, and PLC_ctrl3.

Results

When you create a rising edge with any of the control bits, the IP Engine sends an interrupt request to the PC. Information about each interrupt appears in the **PLC Interrupts** entry on the **Acquisition** tab. (See the *iPORT Coyote Software Guide*.)

Configuring the PLC with the iPORT Vision Suite

In this section:

Configuring the PLC with the SDK 27
Configuring the PLC with Coyote 27

Configuring the PLC with the SDK

To configure the PLC with the iPORT Vision Suite SDK:

- See CyDevice in the *iPORT C++ SDK Reference Guide*.

Configuring the PLC with Coyote

Coyote lets you configure the entire PLC. Once configured, you can save the configuration file and reload it as you like. For a complete description of Coyote's interface, see the *iPORT Coyote Software Guide*.

In this section:

Saving your PLC configuration 27
Monitoring the IO Block 27
Manually controlling your circuit with the Remote Control Block 28
Configuring the IO Block 28
Configuring the Video IO Block 29
Configuring the Signal Routing Block 29
Configuring the Lookup Table 29
Configuring the Input Routing Block 29
Configuring the Enhanced Function Block 29
Configuring the Image Control Block 29
Configuring special PLC settings 30

Saving your PLC configuration

To save your PLC configuration:

- From the main page of Coyote, select **File** > **Save**.
  See the *iPORT Coyote Software Guide*.

Monitoring the IO Block

To monitor the state of the input side of the IO Block:

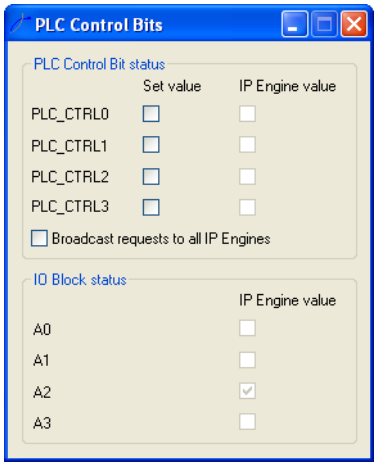
- From the main page of Coyote, select **IP Engine** > **PLC Control Bits**.
  The **PLC Control Bits** dialog appears. The polled state of the IO Block appears in the **IO Block status** pane. If you're monitoring feedback ("Monitoring your PLC signals with a feedback loop"

on page 15), TTL_IN0 is typically **A0**, but consult your hardware guide to be sure. For a complete description of the dialog, see the *iPORT Coyote Software Guide*.

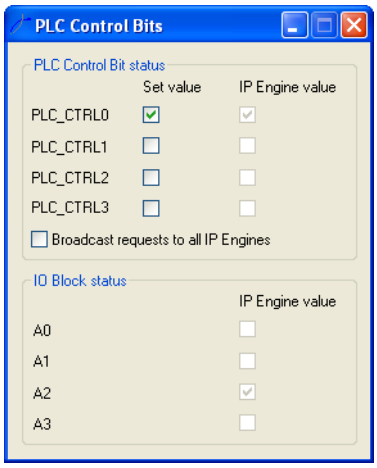
Manually controlling your circuit with the Remote Control Block

To set the Remote Control Block:

1. From the main page of Coyote, select **IP Engine** > **PLC Control Bits**.
   The **PLC Control Bits** dialog appears.



2. In the **PLC Control Bit status** pane, check and uncheck the **Set value** of **PLC_ctrl0** through **PLC_ctrl3**.
   The **IP Engine value** shows the states of the bits in the actual IP Engine. Thus, the **IP Engine value** will follow the **Set value** pane. To learn more about the **PLC Control Bits** dialog, see the *iPORT Coyote Software Guide*.



NOTE! The propagation delay for registry entries (including the PLC control bits) depends on your CPU usage, Windows scheduling, network adapter speed, network traffic, and switch propagation times. Actual times will vary, but are in the order of tens of milliseconds. For hard realtime applications, create your IP Engine output signal using the Counter Trigger Generator. See “Counter Trigger Generator” on page 64.

Configuring the IO Block

Except for the Debouncing Block, the IO Block doesn’t require any configuring.

To access the Debouncing Block configurations:

- From the **Configuration** dialog, select **IP Engine** > **Programmable Logic Controller** > **Input Debouncing Block**.
  For a description of the settings, see “Input Debouncing Block” on page 41.

Configuring the Video IO Block

The Video IO Block doesn't require any configuring.

Configuring the Signal Routing Block

To access the Signal Routing Block configurations from Coyote:

- From the **Configuration** dialog, select the **IP Engine** > **Programmable Logic Controller** > **Signal Routing Block and Lookup Table**.

Configuring the Lookup Table

To access the Lookup Table configurations:

1. From the **Configuration** dialog, select **IP Engine** > **Programmable Logic Controller** > **Signal Routing Block and Lookup Table**.
2. Replace **LOCKED** with your Lookup Table configurations and click **Apply**.

NOTE! The Lookup Table configurations (Q1=I2, etc.) are automatically converted into a lookup table format. If you wish to make a change to the Lookup Table after closing and reopening Coyote, retype all the values. The values are maintained if you save the XML.

Configuring the Input Routing Block

To access the Input Routing Block configurations from Coyote:

- From the **Configuration** dialog, select **IP Engine** > **Programmable Logic Controller** > **Signal Routing Block and Lookup Table**.
  See "Signal Routing Block" on page 53.

Configuring the Enhanced Function Block

To access the Enhanced Functionality Block configurations from Coyote:

1. From the **Configuration** dialog, select **IP Engine** > **Programmable Logic Controller** > **Enhanced Function Block**.
2. Select the function that you require (e.g. **Pulse Generator 0**, **Delayer**, etc.).
3. Make changes as necessary and click **Apply**.

See "Enhanced Function Block" on page 57.

To enable the Interrupt FIFO interrupts:

1. From the **Configuration** dialog, select **IP Engine** > **Programmable Logic Controller** > **PLC Interrupts**.
2. Enable the interrupts that you require and click **Apply**.

See "Interrupt FIFO" on page 64.

NOTE! To minimize unnecessary CPU usage, don't enable interrupts that you don't require.

Configuring the Image Control Block

To enable image triggering on Q14:

1. From the **Configuration** dialog, select the **Grabber** tab.
2. Enable **PLC Triggerable**.
   You can now trigger the image grabber using signals sent to Q14.

To override the FVAL and LVAL settings:

1. From the **Configuration** dialog, select the **Grabber Extensions** tab.

2. Expand **Camera Link**. Change the **Frame Valid function selection** or **Line Valid function selection** as desired and click **Apply**.
   The grabber uses the Boolean combination that you select. See “Image Control Block” on page 67.

To configure the DVAL, LVAL, or FVAL polarity and edge sensitivity settings:

1. From the **Configuration** dialog, select the **Grabber Extensions** tab.
2. Expand **Camera Link**. Make changes as necessary and click **Apply**.

See “Image Control Block” on page 67.

Configuring special PLC settings

To configure the special PLC settings:

1. From the **Configuration** dialog, select **IP Engine** > **Programmable Logic Controller** > **PLC Configuration**.
2. Make changes as necessary and click **Apply**.

NOTE! Dropdowns with a value of **0**, **1**, **2**, or **3** are null values that make no changes. IP Engines have very few special settings so most of the configurations are unused.

Configuring the PLC with the iPORT PureGEV Suite

GEV-enabled IP Engines generally match the behavior described in this guide, with a few exceptions. In general, the GEV-enabled IP Engines add the prefix PLC_ to the signal names.

To configure the PLC with AutoGEV:

1. Configure the PLC as you require. See the *iPORT AutoGEV Software Guide*.
2. Optionally, set the visibility attribute of unneeded or unchangeable features to invisible.
3. Optionally, create virtual features to control the PLC, and set the underlying features' attributes to invisible.
4. Upload your project to your IP Engine.

To control the PLC with GEVPlayer:

- Change the features in this section as required. To change features in GEVPlayer, see the *iPORT PureGEV Quick Start Guide*.

In this section:

IO Block 31
Video IO Block 31
Video IO Block 31
Signal Routing Block 32
Lookup Table 32
Enhanced Function Block 33
Special PLC settings 37

IO Block

See "IO Block" on page 39.

IO Block features

Display name	Name (identifier)
A*n* (for *n* values 0 to 3)	Line*n*

Video IO Block

The Video IO Block doesn't require any configuring.

See “Video IO Block” on page 47.

Video IO Block features

Display name	Name (identifier)
A*n* (for *n* values 4 to 7)	PLC_A*n*

Remote Control Block

Unlike Coyote, which distinguishes the desired value (the value you set) from the actual value (the value in the IP Engine), the iPORT PureGEV Suite makes the distinction intrinsically.

See “Remote Control Block” on page 51.

Remote Control Block features

Display name	Name (identifier)
PLC_CTRL0 PLC_CTRL*n* (for *n* values of 0 through 3)	PLC_ctrl0 PLC_ctrl*n*

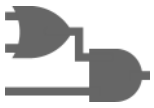
Signal Routing Block

See “Signal Routing Block” on page 53.

Signal Routing Block features

Display name	Name (identifier)
I0 I*n* (for n values of 0 through 7)	PLC_I0 PLC_I*n*

Lookup Table



Rather than accepting an equation in the form of a string, GEV-enabled IP Engines accept a series of separate variables and operators that you specify using dropdown selections. For example, to set...

Q0 = I5 & (I3 | I4)

... make the following settings:

Lookup Table
PLC_Q0_Variable0 = PLC_I5
PLC_Q0_Operator0 = AndParenthesis
PLC_Q0_Variable1 = PLC_I3
PLC_Q0_Operator1 = Or
PLC_Q0_Variable2 = PLC_I4
PLC_Q0_Operator2 = Or
PLC_Q0_Variable3 = Zero

In this case, the equation supports 4 variables, though only 3 were used. To keep from unintentionally changing the value of your equation, extend short equations with Or Zero. Parentheses are closed automatically at the end of the equation. Thus, the finished iPORT PureGEV Suite equation would be:

Q0=I5 &(I3 | I4 | 0

... which is the iPORT PureGEV Suite equivalent of...

Q0 = I5 &(I3 | I4)

See “Lookup Table” on page 55.

Lookup Table features

Display name	Name (identifier)
Q0	PLC_Q0_*
Q*n*	PLC_Q*n*_*

Enhanced Function Block

See “Enhanced Function Block” on page 57.

In this section:

Pulse Generator 33
Rescaler 34
Delayer 35
General Purpose Counter 35
Interrupt FIFO 35
Counter Trigger Generator 36
Timestamp Counter 36

Pulse Generator

The Pulse Generators have PLC_* feature names, as well as virtual features that replicate the functionality with Timer* feature names. The latter were included to better match the *GenICam Standard Feature Naming Convention* (see www.machinevisiononline.org).

See “Pulse Generator” on page 59.

Pulse Generator features

Display name	Name (identifier)
Pulse Generator 0 Pulse Generator *n* (for n values of 0 through 3)	PLC_pg0_* PLC_pg*n*_* *or* TimerSelector
Width (high)	PLC_pg0_Width PLC_pg*n*_Width *or* TimerDurationRaw
Delay (low)	PLC_pg0_Delay PLC_pg*n*_Delay *or* TimerDelayRaw
Granularity factor	PLC_pg0_GranularityFactor PLC_pg*n*_GranularityFactor *or* TimerGranularityFactor

Pulse Generator features

Display name	Name (identifier)
Emit periodic pulse	PLC_pg0_TriggerSource PLC_pg*n*_TriggerSource *or* TimerTriggerSource
Trigger mode	PLC_pg0_TriggerActivation PLC_pg*n*_TriggerActivation *or* TimerTriggerSource
Pulse period (ns)	PLC_pg0_PulsePeriod PLC_pg*n*_PulsePeriod *or* TimerPeriod
Pulse frequency (Hz)	PLC_pg0_PulseFrequency PLC_pg*n*_PulseFrequency *or* TimerFrequency

Rescaler

See “Rescaler” on page 60.

Rescaler features

Display name	Name (identifier)
Granularity	PLC_rsl0_Granularity
Multiplier	PLC_rsl0_Multiplier
Divider	PLC_rsl0_Divider
Input signal	PLC_rsl0_InputSignal
Backup enabled	PLC_rsl0_BackupEnabled
Backup switchover delay	PLC_rsl0_BackupSwitchoverDelay
Backup input signal	PLC_rsl0_BackupInputSignal
Input frequency	PLC_rsl0_InputFrequency
Output frequency	PLC_rsl0_OutputFrequency
Target frequency Recommended granularity Recommended multiplier Recommended divider	Not available
Rescaler sample size	PLC_rsl0_SampleSize
PLC_rsl_out	PLC_rsl0_out

Delayer

See “Delayer” on page 61.

Delayer features

Display name	Name (identifier)
Delay count	PLC_del0_DelayCount
Reference timing signal	PLC_del0_ReferenceTimingSignal
Input signal	PLC_del0_InputSignal
del_out	PLC_del0_out

General Purpose Counter

See “General Purpose Counter” on page 62.

General Purpose Counter features

Display name	Name (identifier)
Increment trigger mode	PLC_gp_cnt0_IncrementActivation
Decrement trigger mode	PLC_gp_cnt0_DecrementActivation
Clear trigger mode	PLC_gp_cnt0_ResetActivation
Clear signal	PLC_gp_cnt0_ResetSource
Compare value	PLC_gp_cnt0_CompareValue
Current counter value	PLC_gp_cnt0_Value
gp_cnt_eq	Counter1Eq
gp_cnt_gt	Counter1Gt

Interrupt FIFO

See “Interrupt FIFO” on page 64.

Interrupt FIFO features

Display name	Name (identifier)
Q15 Enabled	PLC_Interrupt_FIFO0_Q15_Enabled
Q3 Enabled	PLC_Interrupt_FIFO0_Q3_Enabled
Q7 Enabled	PLC_Interrupt_FIFO0_Q7_Enabled
Q10 Enabled	PLC_Interrupt_FIFO0_Q10_Enabled
IRQ_mask[3:0]	PLC_Interrupt_FIFO0_IRQ_mask
time[31:0]	PLC_Interrupt_FIFO0_time
SRB_mask[7:0]	PLC_Interrupt_FIFO0_SRB_mask

Counter Trigger Generator

See “Counter Trigger Generator” on page 64.

Counter Trigger Generator features

Display name	Name (identifier)
FIFO full	PLC_ts_trig_FIFOFull
FIFO empty	PLC_ts_trig_FIFOEmpty
Trigger mask values	PLC_ts_trig_0_Enable PLC_ts_trig_1_Enable PLC_ts_trig_2_Enable PLC_ts_trig_3_Enable
Counter selector	PLC_ts_trig_CounterSelect
Trigger’s time	PLC_ts_trig_Time
Arm command	PLC_ts_trig_Arm
ts_trig*n*	PLC_ts_trig*n*

Timestamp Counter

The Timestamp Counter varies slightly from that described See “Timestamp Counter” on page 65.

Timestamp Counter features

Display name	Name (identifier)
Counter select	CounterSelector
Granularity	Not available
Set trigger mode	CounterTriggerSource
Clear trigger mode	CounterResetActivation
Set input signal	Counter
Clear input signal	CounterResetSource
Broadcast	Not available
Set counter value Current counter value	CounterValue

Special PLC settings

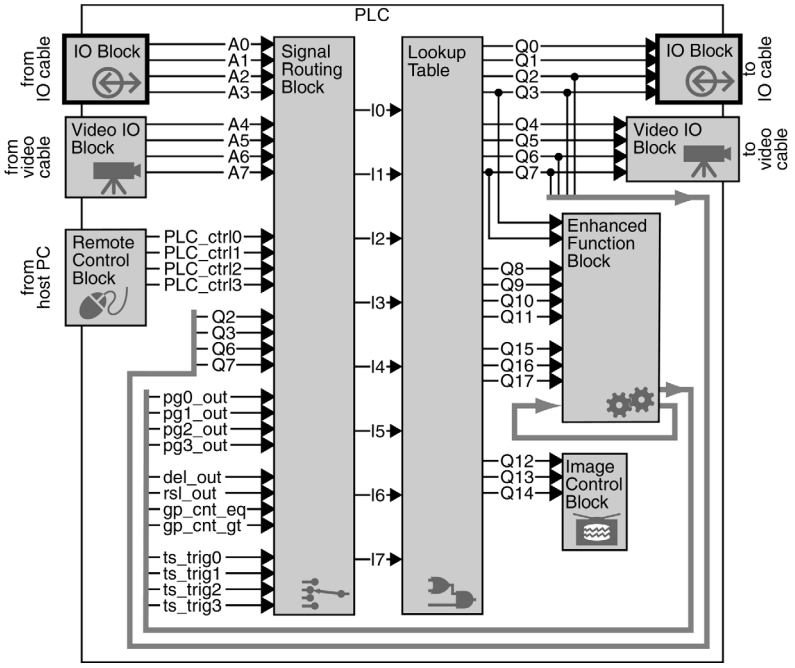
See “Configuring special PLC settings” on page 30.

Special PLC settings

Display name	Name (identifier)
PLC Configuration	Line0Configuration Line*n*Configuration PLC_Q0_Configuration PLC_Q*n*_Configuration

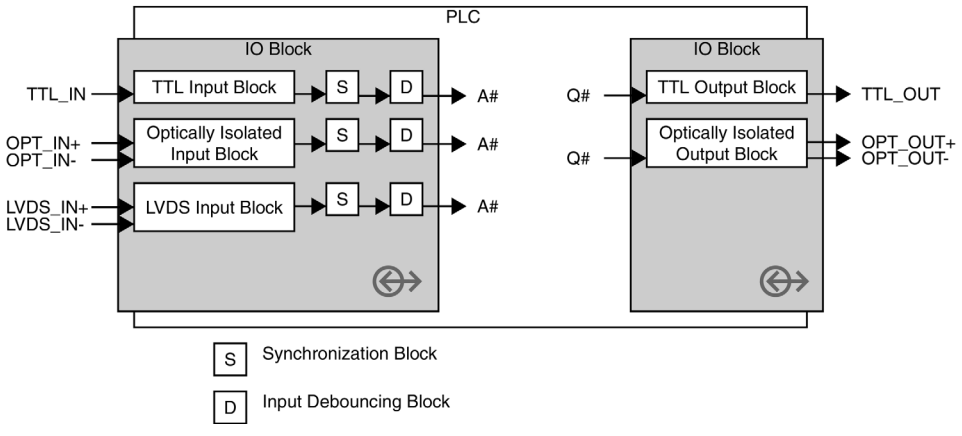
IO Block

The IO Block lets you communicate with the PLC through the IP Engine's 12-pin IO connector. The IO Block consists of an input (on the left) and an output (on the right).



The main function of the IO Block is to convert external signals to low voltage TTL signals usable by the PLC (and vice versa). This section describes the electrical circuits used to convert between PLC signals and external TTL, LVDS, and optically-isolated signals. The IO Block filters inputs and outputs to reduce the risk of damage due to electrostatic discharge (ESD) and accidental shorts.

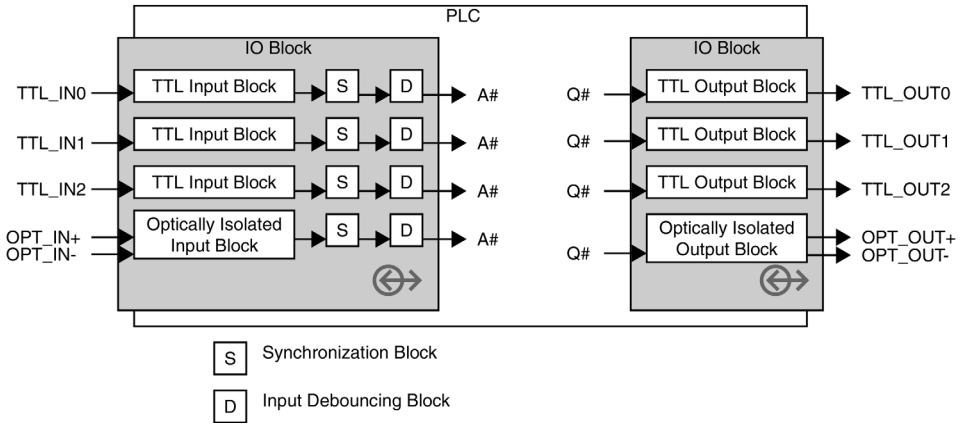
(All possible input and output blocks)



Except for the Input Debouncing Block, the IO Block doesn't need to be configured.

The pinouts on the IP Engine's IO connector vary depending on your model; for the specific pinouts for yours, see your hardware guide. For example, the IO Block for your IP Engine might have several TTL inputs and outputs, an optically-isolated input and output, but no LVDS input.

(Typical IO Block)



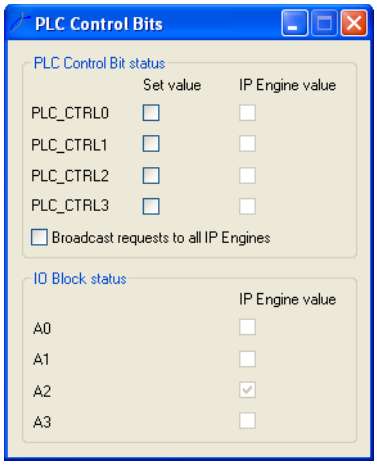
To configure the IO Block, see "Configuring the IO Block" on page 28.

In this section:

Monitoring the state of the IO Block 40
Synchronization Block 40
Input Debouncing Block 41
TTL Input Block 42
TTL Output Block 43
LVDS Input Block 43
Optically Isolated Input Block 44
Optically Isolated Output Block 45

Monitoring the state of the IO Block

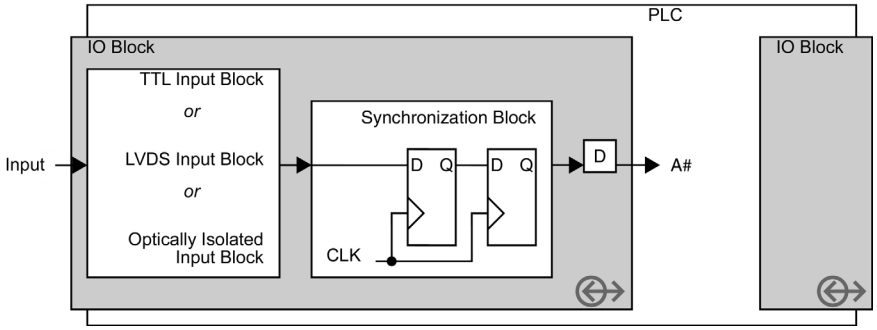
Pleora's Coyote camera interface application lets you monitor the state of the input side (left side) of the IO Block. Your PC polls the state of the IP Engine's IO Block at a rate of about 4 Hz and displays the results in the IO Block status pane. See "Monitoring the IO Block" on page 27.



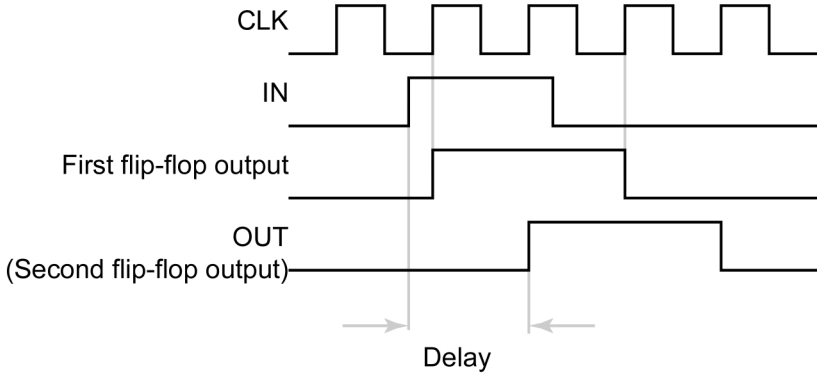
Synchronization Block

The Synchronization Block samples input signals in time with the IP Engine's system clock. (Both the IO Block and the Video IO Block have Synchronization Blocks within.) The system clock has a 30 ns

period (33 MHz clock cycle). To maximize stability of the input signals and minimize the risk of meta-stability problems, the Synchronization Block uses two consecutive flip-flops.



The Synchronization Block latches the input on every rising edge of the system clock.

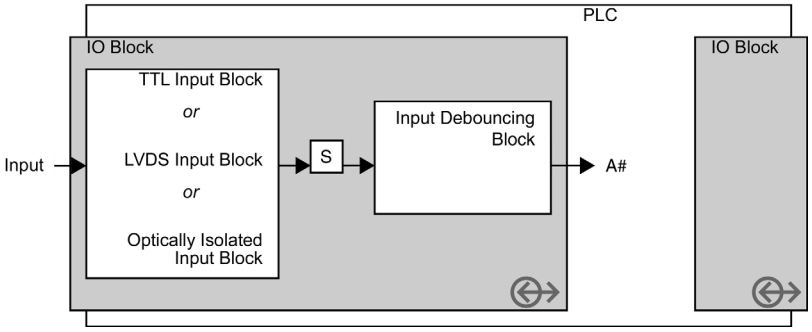


Synchronization Block specifications

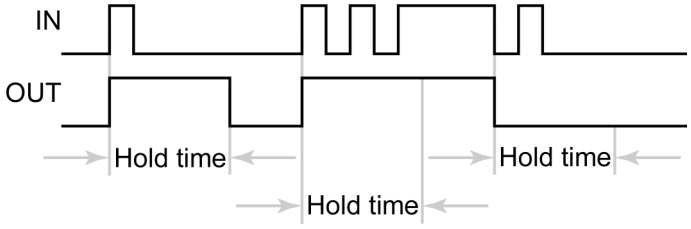
Specification	Value
Propagation delay	minimum: 30 ns maximum: 60 ns

Input Debouncing Block

The Input Debouncing Block lets you ignore spurious transitions from input signals A0-A3.



You can independently configure each input signal to hold signal transitions for between 480 ns and ~31 ms. While holding the first transition, the Input Debouncing Block ignores further transitions for the duration that you've configured.

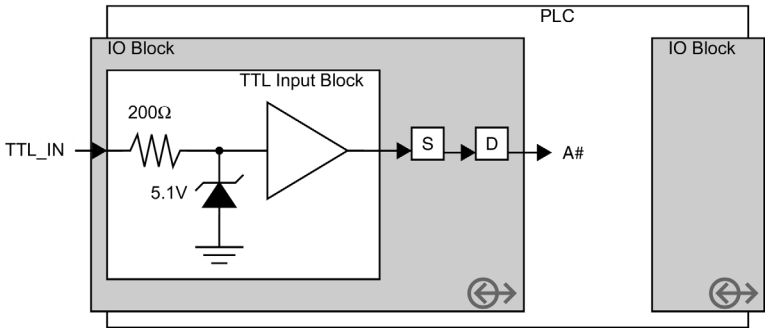


The hold times can be configured in increments of 480 ns (16 system clock cycles). Setting the hold value to 0 disables the Input Debouncing Block for that input signal.

Hold time = *Hold value* * 480 ns

TTL Input Block

The TTL Input Block accepts a TTL signal (0 V - 5.0 V) and converts it to a signal usable by the Signal Routing Block. The TTL Input Block uses the ground on the IO connector.

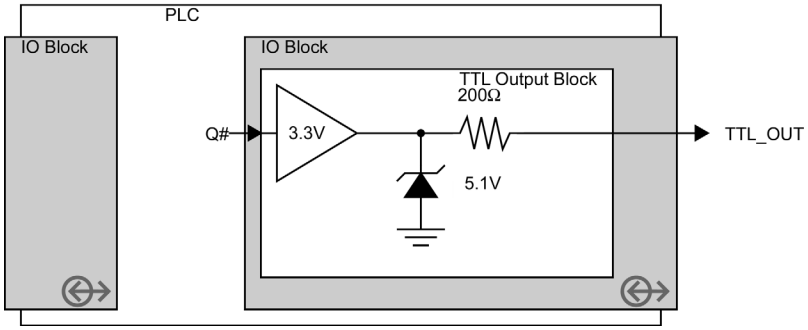


TTL Input Block specifications

Specification	Value
Maximum input frequency	16.5 MHz
Termination	200 ohms serial
Input current	minimum: 0 nA maximum: 20 uA
Input voltage maximum low	0.9 V
Input voltage minimum high	2.1 V

TTL Output Block

The TTL Output Block emits a TTL signal (0 V - 5.0 V). The TTL Output Block uses the ground on the IO connector.

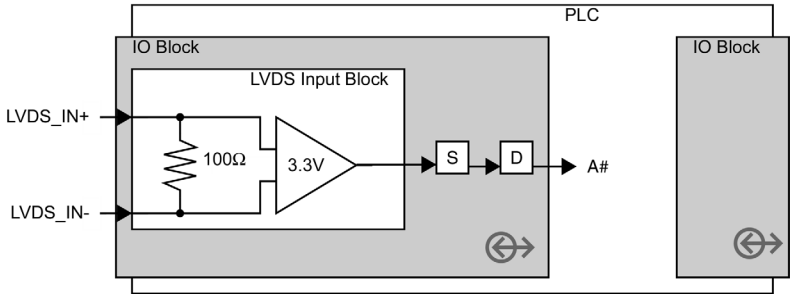


TTL Output Block specifications

Specification	Value
Termination	200 ohms serial
Output current maximum	sink: 8 mA source: 8 mA
Output voltage maximum low	0.44 V
Output voltage minimum high	2.48 V

LVDS Input Block

The LVDS Input Block accepts a low voltage differential signaling (LVDS) signal. The LVDS transmitter signals a high or low by running a small positive or negative current through a twisted wire pair. LVDS signaling is fast, requires little power, and more tolerant of line noise than TTL.



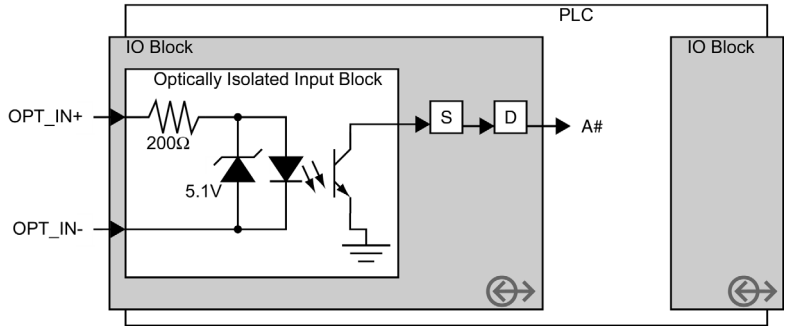
LVDS Input Block specifications

Specification	Value
Maximum input frequency	16.5 MHz
Termination	100 ohms differential
Input current	minimum: -10 uA maximum: 10 uA
Input voltage	minimum: 0.0 V maximum: 3.0 V

Optically Isolated Input Block

The Optically Isolated Input Block uses an LED and a light sensor to electrically decouple the IP Engine from the device that sends the signal. Optoisolators reduce the risk of an overvoltage problem causing collateral damage. They also let a machine with a noisy ground or a different ground voltage level communicate without affecting the IP Engine. Compared with TTL inputs, the response time of optoisolators is relatively slow, particularly for falling edges

The Optically Isolated Input Block accepts a 0 V - 5 V signal.

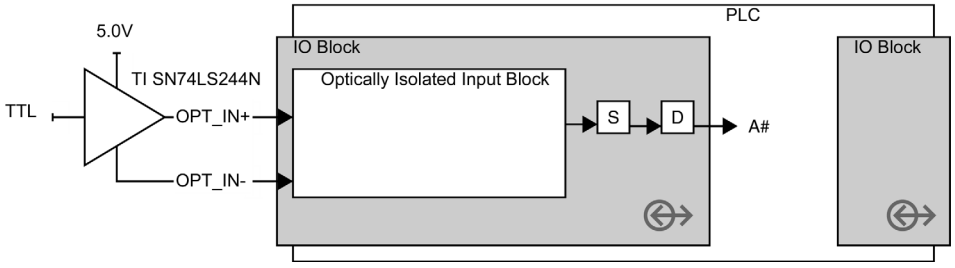


Optically Isolated Input Block specifications

Specification	Value
Termination	200 ohms serial
Input threshold voltage	1.44 V
Input current of low	minimum: 0.0 uA maximum: 20.0 uA
Input current of high	minimum: 4.5 mA maximum: 19.0 mA
Input voltage of low	minimum: 0.0 V maximum: 0.8 V
Input voltage of high	minimum: 2.0 V maximum: 5.0 V

Response time test circuit

The actual response time for an optoisolator depends on the current load and the voltage level that constitutes a high or low signal. Below is the test circuit used to measure response times.

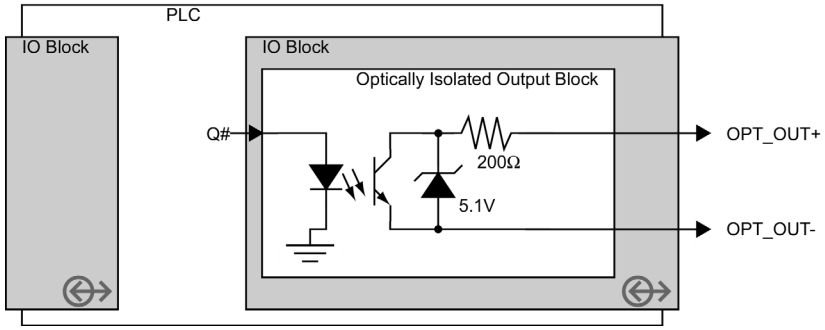


Optically Isolated Output Block response time test circuit specifications

Specification	Value
Edge response time	rising: 2.11 us falling: 30.56 us

Optically Isolated Output Block

The Optically Isolated Output Block is an open-collector circuit that manipulates an external pull-up voltage to produce a 0 V - 5 V signal. For an overview of optoisolators, see “Optically Isolated Input Block” on page 44.

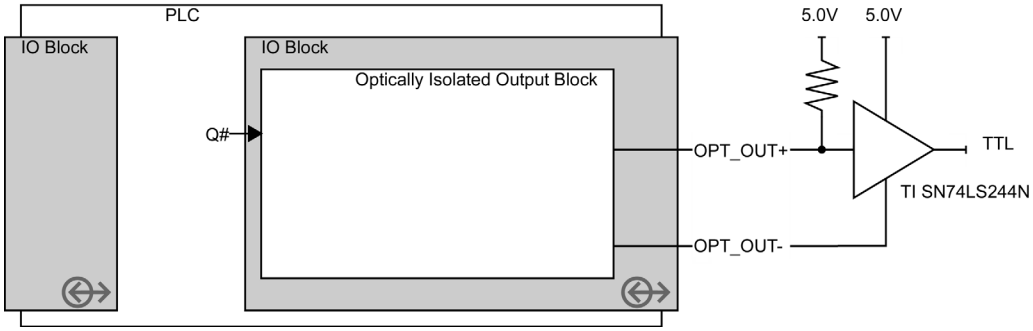


Optically Isolated Output Block specifications

Specification	Value
Termination	200 ohms serial
Output current	minimum: 0.0 mA maximum: 25.0 mA
Output voltage	minimum: 0.0 V maximum: 5.0 V

Response time test circuit

The actual response time for an optoisolator depends on the current load and the voltage level that constitutes a high or low signal. Below is the test circuit used to measure response times.

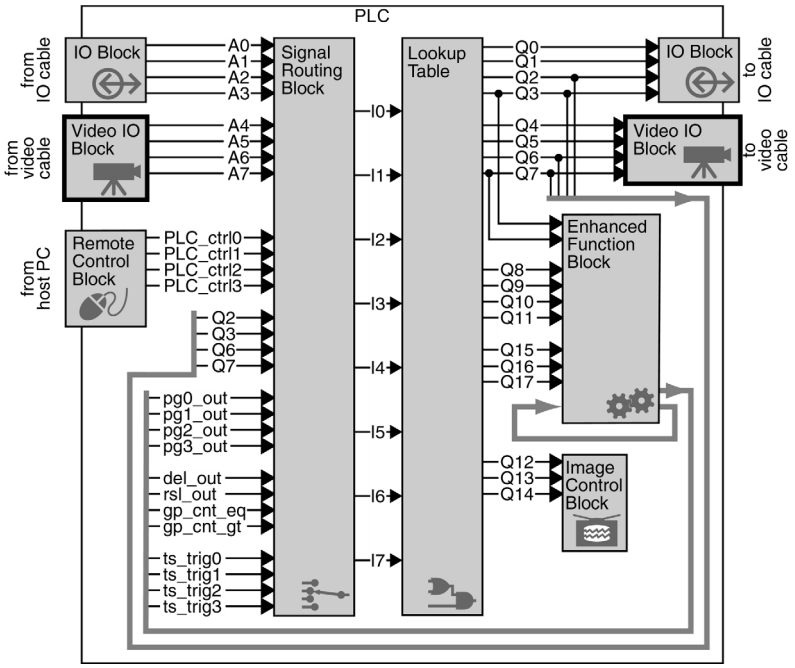


Optically Isolated Output Block response time test circuit specifications

Specification	Value
Output voltage	minimum: 0.5 V maximum: 5.0 V
Edge response time	rising: 32.89 us falling: 2.89 us

Video IO Block

The Video IO Block receives a video signal and provides the PLC with discrete frame valid (FVAL), line valid (LVAL), and similar signals. Some camera signal types permit camera control signals from the IP Engine. The Video IO Block consists of an input (on the left) and an output for camera control (on the right). The exact functionality of the Video IO Block depends on the video input format (e.g. Camera Link, analog video, etc.) and the features the format supports.



The Video IO Block doesn't need to be configured.

In this section:

How the Video IO Block works 47
Video IO Block (for Camera Link imaging data) 48
Video IO Block (for analog video) 48
Video IO Block (for LVDS imaging data) 49

How the Video IO Block works

The Video IO Block on your IP Engine can support Camera Link imaging data, analog video, or LVDS imaging data. All three camera signal formats have signals that let you (and the IP Engine's image grabber) distinguish frames and lines from each other. However, the formats differ from each other. Internally, the IP Engine converts the signal format to conform to the Camera Link standard.

The Camera Link standard defines each of the input signals:

FVAL
: Frame Valid. High for valid lines. (In practice, this lets you separate one *frame* from another.)

LVAL
: Line Valid. High for valid pixels. (In practice, it separates one *line* from another.)

DVAL
: Data Valid. High when data is valid (i.e. high for each pixel).

SPR
: Spare. A spare line for future use by the Camera Link standard.

The Camera Link Video IO Block and the LVDS Video IO Block use the actual signals sent through the video cable.

For analog video signals, the FVAL and LVAL signals are akin to the vertical-retrace and horizontal-retrace signals. However, the DVAL and SPR signals are of little significance for an analog camera signal, so the Analog Video IO Block replaces them with different signals:

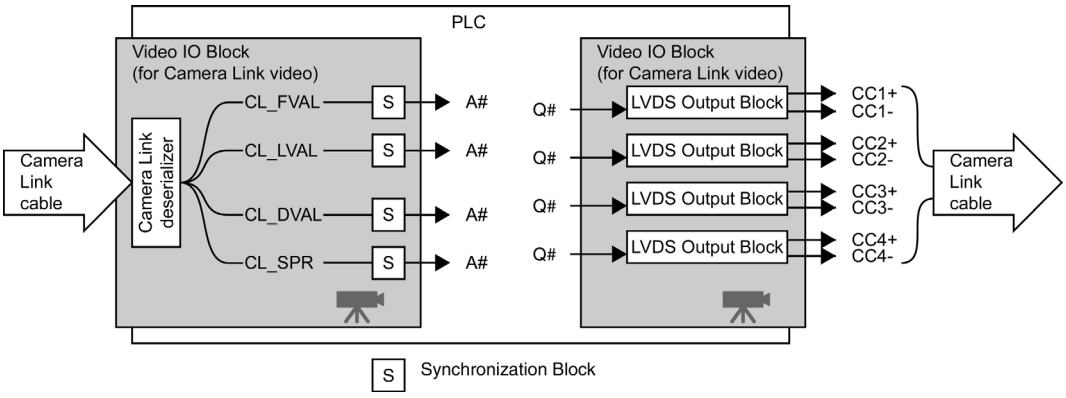
RTS1
: The Real-time status.
For Black and White Progressive Mode, RTS1 is “Horizontal sync” and provides a pulse at the beginning of a line.
Otherwise, the signal is equivalent to “Vertical and Horizontal lock.” The signal is high when the Analog Video Decoder detects a valid signal and is locked horizontally. Thus, for a good signal, it is always high.

FID
: The field identifier.
For interlaced signals, FID identifies all “even” lines of an interlaced video signal with a 0; all “odd” with a 1.
For progressive video signals, FID remains 0.

Although the Camera Link standard defines the FVAL and LVAL signals, some cameras may invert the signals or otherwise differ from the standard. To manipulate the FVAL, LVAL signals for controlling how the IP Engine’s image grabber acquires images, see “Image Control Block” on page 67.

Video IO Block (for Camera Link imaging data)

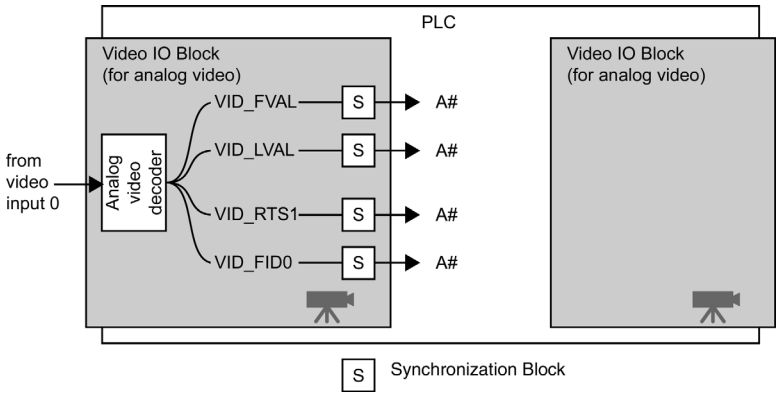
The Camera Link Video IO Block sends and receives its signal from the 26-pin Camera Link connector. The Camera Link standard transmits the FVAL, LVAL, DVAL, and spare signals over the Camera Link bus. The camera control signals use discrete LVDS outputs.



Video IO Block (for analog video)

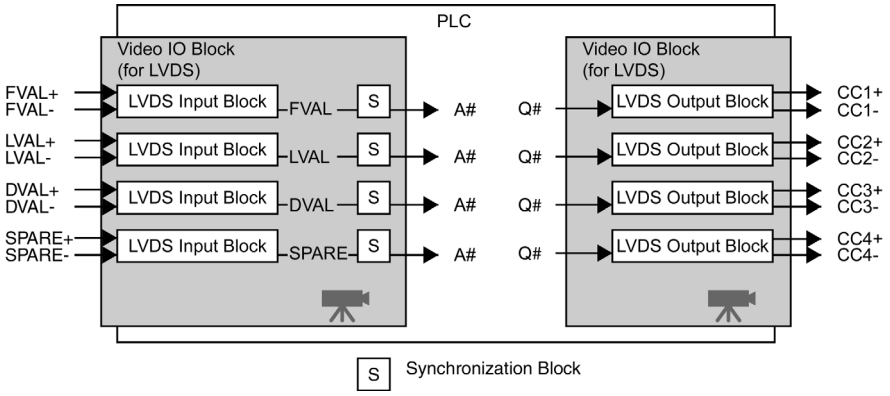
The Analog Video IO Block receives its signal from the coaxial cable attached to the BNC connector. The Analog Video IO Block interprets the signal and outputs the FVAL, DVAL, RTS1, and FID signals on discrete lines. Analog video transmission is one-way only; the format doesn’t let you send camera

control signals back to the camera using the video output cable. However, if your camera accepts external controls, you can use the IP Engine to send controls using the IO Block.



Video IO Block (for LVDS imaging data)

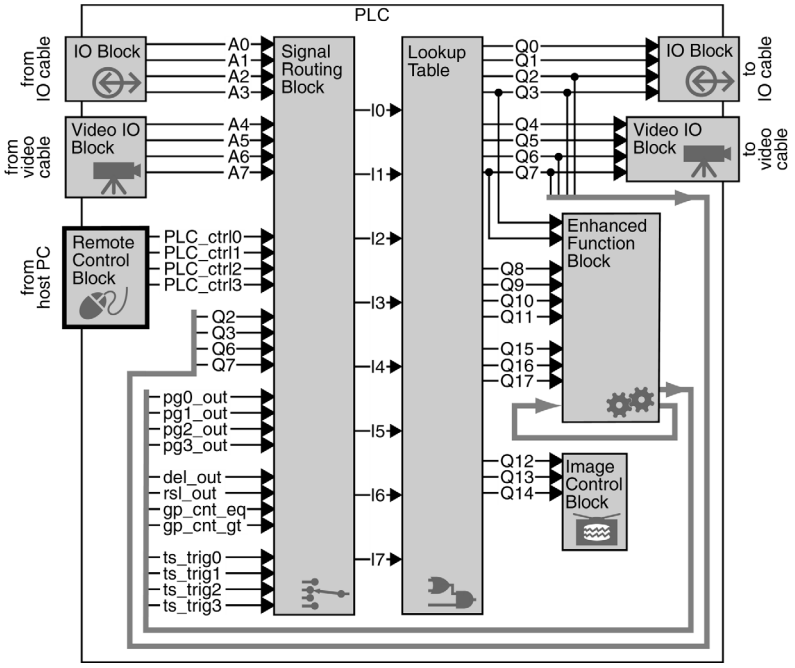
The LVDS Video IO Block receives its signal through the cable attached to the 68-pin LVDS connector. The LVDS Video IO Block receives and sends signals through discrete lines (i.e. FVAL, CC1, and other signals each have their own unshared line).



Remote Control Block

The Remote Control Block lets you send input signals to your IP Engine from Coyote. You can use these signals to simulate the inputs from switches, sensors, and other hardware without having to cable the actual equipment together. The input signals are called PLC control bits.

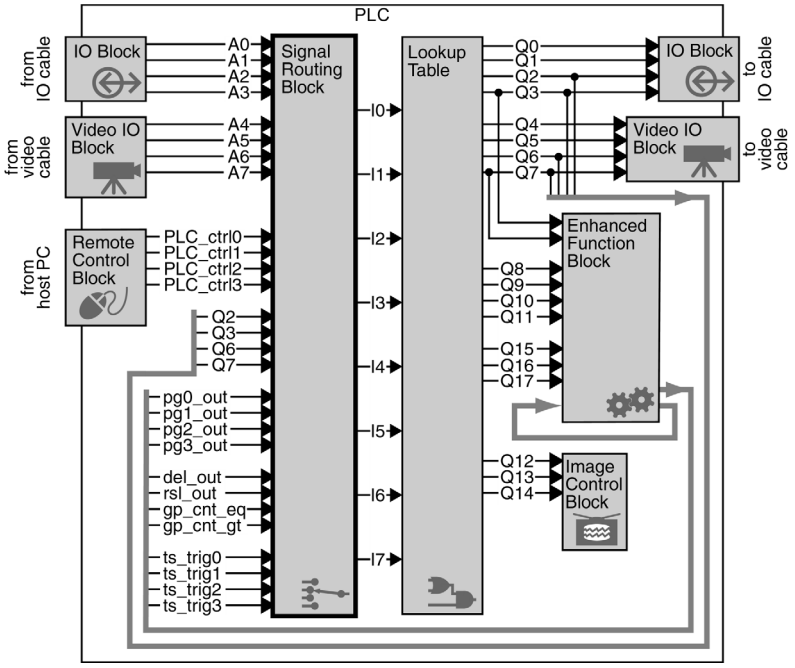
Within the IP Engine, the PLC control bits are binary registers that can be set and read using the iPORT SDK.



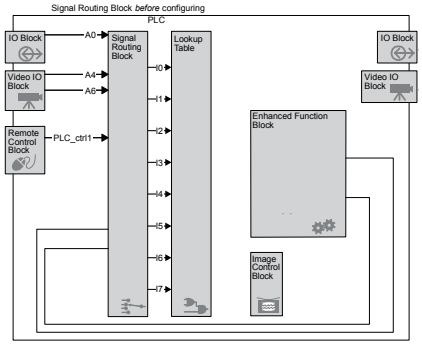
To use the Remote Control Block, see “Manually controlling your circuit with the Remote Control Block” on page 28.

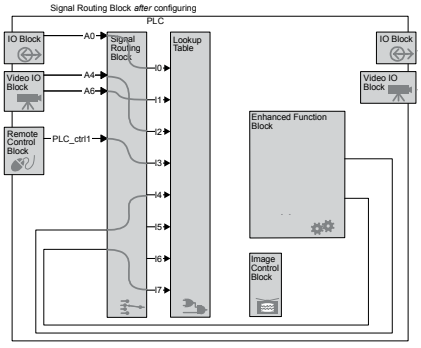
Signal Routing Block

In its simplest terms, the Signal Routing Block is a group of switches that let you route signals to the Lookup Table. You can direct PLC inputs and feedback inputs to signals I0 through I7.



The Signal Routing Block lets you redirect signals from the IO Block, the Video IO Block, Lookup Table, and the Enhanced Function Block *back* into the Lookup Table for further processing. Because most of the other blocks in the PLC use preconfigured inputs and outputs, the Signal Routing Block is the primary method of routing a signal from one block to another.





To configure the Signal Routing Block, see “Configuring the Signal Routing Block” on page 29.

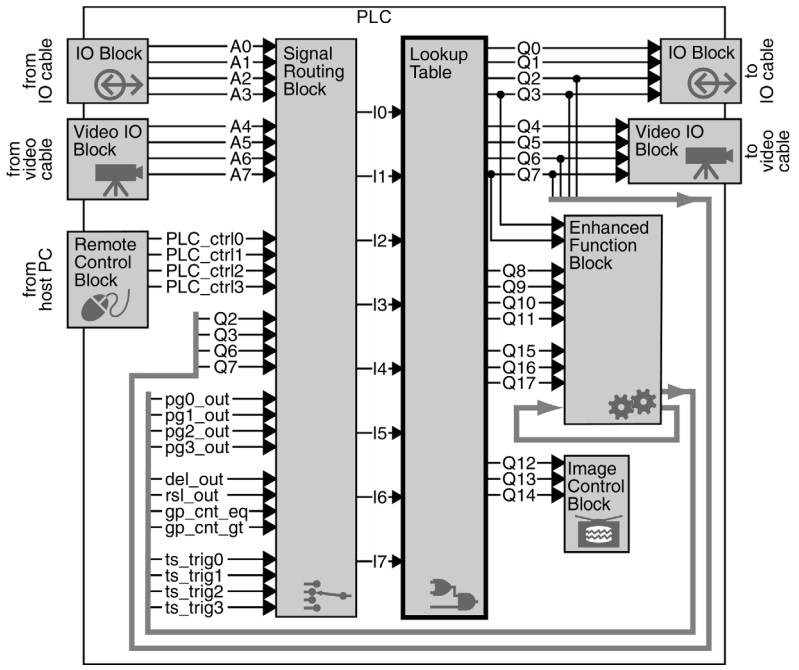
How the Signal Routing Block works

The Signal Routing Block has 8 outputs (I0 - I7). Each output uses a 16:1 multiplexer that connects to 16 inputs.

The Signal Routing Block has more than 16 input signals, so not every input can be connected to every one of signals I0 - I7. However, signals I0 - I7 are functionally identical, so connecting to a specific one isn’t important. If you can’t route the input with your first choice, simply choose another.

Lookup Table

The Lookup Table lets you connect any input signal I0-I7 to any Lookup Table output signal Q0-Q17.



You can manipulate your inputs using simple or complex Boolean expressions. The following expressions are both valid:

Q0 = I6

Q6 = !(I4 & I6) & ((I2 ^ I5) | I1)

Lookup Table syntax

Syntax	Valid construction	Sample line
Line	*Output* = *Expression* EOL (end of line)	
Output	Q0, Q1, Q2, ..., Q16, Q17	
Input	I0, I1, I2, ..., I6, I7	
Expression	*Input* Not *Input* Boolean constant	Q1=I5 Q1=!I5 Q1=FALSE
Combined *Expression*	*Expression* Boolean operator *Expression*	Q1=I5 & I3 Q16 = I8 | I6
Boolean operators	& (and) | (or) ^ (xor)	Q14 = I4 & I6 Q15 = I3 | I5 Q9 = I1 ^ I8

Lookup Table syntax

Syntax	Valid construction	Sample line
Not	!	Q0=!I0 Q10= !(I8 & I5)
Delimiter	( )	Q0 = !(I0) Q3 = !(I1 | (I7 ^ I5)) Q6 = (I3 | I5) ^ (I1 & I2)
Boolean constants	1, true, TRUE 0, false, FALSE	Q0 = 1 Q3 = TRUE Q6 = I3 ^ true
EOL	\r \n \r\n \n\r	(used only for SDK, not Coyote)

Incorrect Lookup Table usage

Rule	Incorrect syntax	Correct syntax
The output must be on the left hand side of the equation (the value is being assigned to Q4, not I5).	I5 = Q4	Q4 = I5
Outputs may not be on the right hand side of the equation.	Q1 = I7 & I8 Q2 = Q1 | I5	Q1 = I7 & I8 Q2 = (I7 & I8) | I5
Equations must be separated by a carriage return or an EOL symbol.	Q3 = I7,Q15=I8	Q3 = I7 Q15 = I8

To configure the Lookup Table, see “Configuring the Lookup Table” on page 29.

How the Lookup Table works

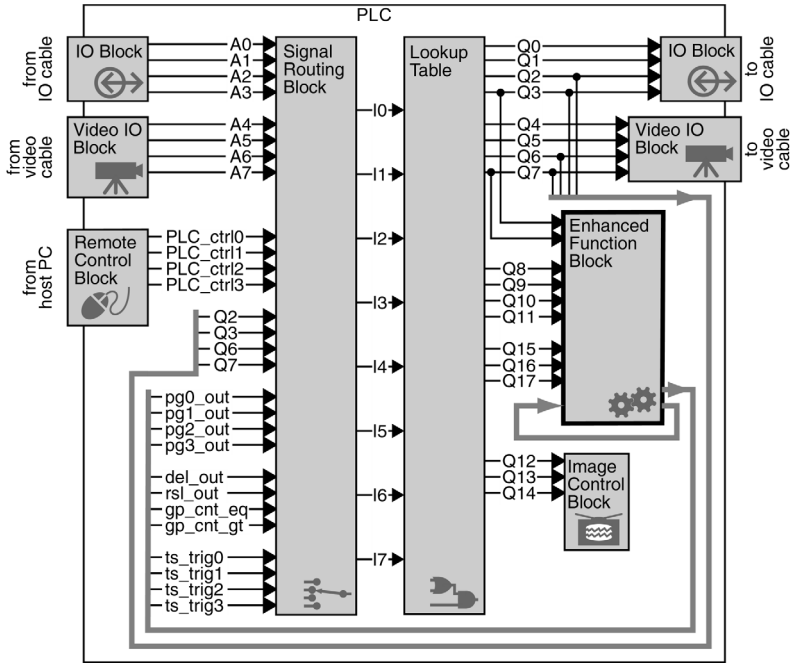
The Lookup Table has 8 inputs (I0 - I7) capable of two states each (true, false). Thus, the outputs have a total number of 256 input combinations. The result of each combination can be 1 or 0.

When you modify the equations in the Lookup Table, Coyote calculates the results of all 256 input combinations and stores the result of each output as a 256-bit lookup table (hence the name). There are 18 outputs (Q0 - Q17), so Coyote calculates 18 different lookup tables.

Coyote then passes the resulting 18 lookup tables to the IP Engine. Knowing the value of the 8 inputs, the PLC needs only look up the value of the resulting output (for each output), rather than calculate it. Thus, the Lookup Table can achieve a propagation delay of only one system clock cycle (30 ns), regardless of the complexity or number of Boolean expressions.

Enhanced Function Block

The Enhanced Functionality Block lets you perform complex functions on your signals. You can delay a signal, generate a pulsed signal, count pulses, generate interrupts to the PC, and more.



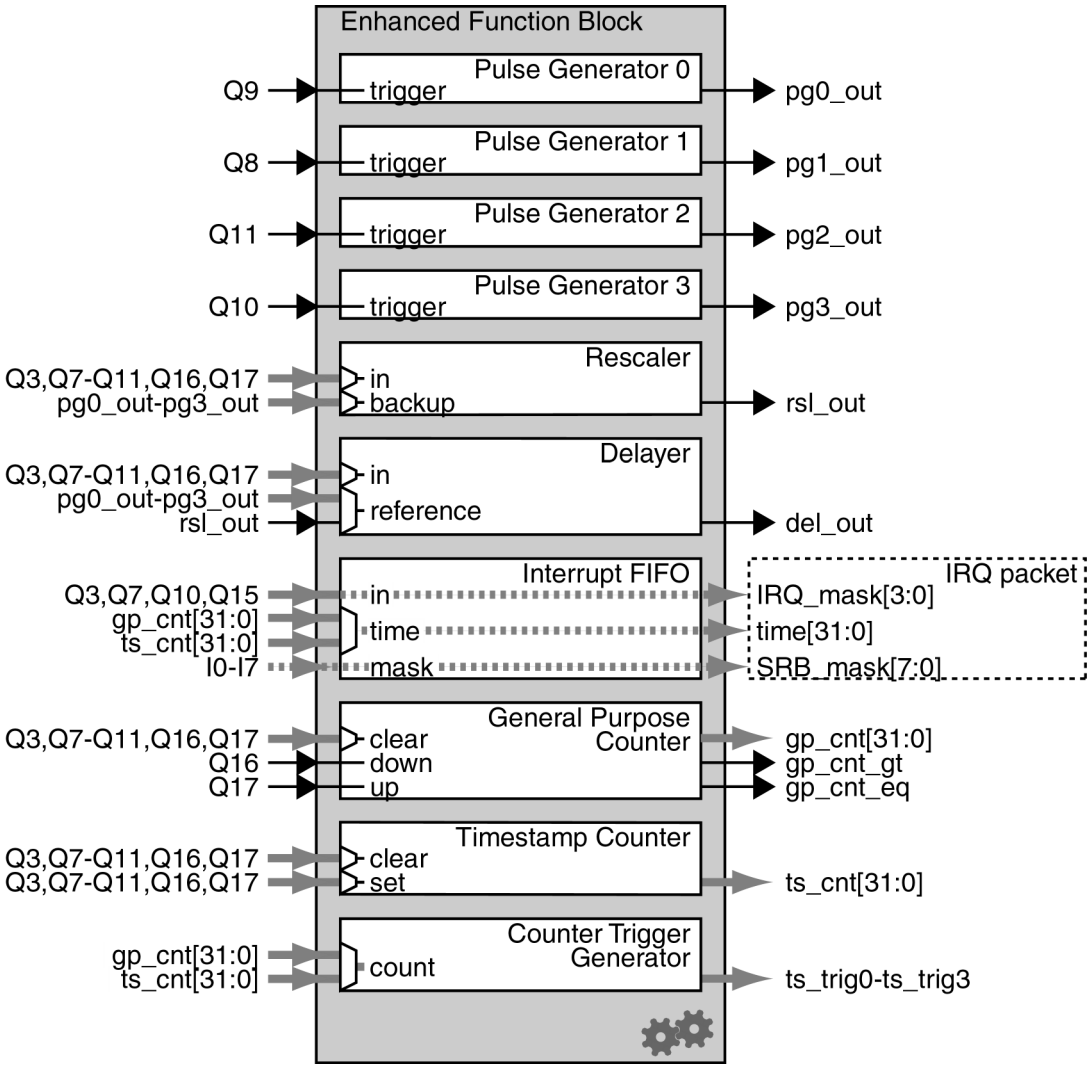
To configure the Enhanced Function Block, see “Configuring the Enhanced Function Block” on page 29.

In this section:

Overview of the Enhanced Function Block 58
Pulse Generator 59
Rescaler 60
Delayer 61
General Purpose Counter 62
Interrupt FIFO 64
Counter Trigger Generator 64
Timestamp Counter 65

Overview of the Enhanced Function Block

The Enhanced Function Block includes many discrete, independent functions.



Brief summaries of the parts of the Enhanced Function Block

Pulse Generator
: Create a pulse.

Rescaler
: Adjust the frequency of an input signal.

Delayer
: Delay a complex signal.

Interrupt FIFO
: Send an interrupt request to the PC.

General Purpose Counter
: Count pulses (such as from the encoder on a conveyor belt).

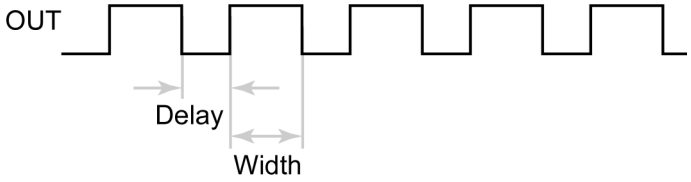
Timestamp Counter
: Use the IP Engine's onboard clock.

Counter Trigger Generator
: Set an "alarm clock" that outputs up to four signals.

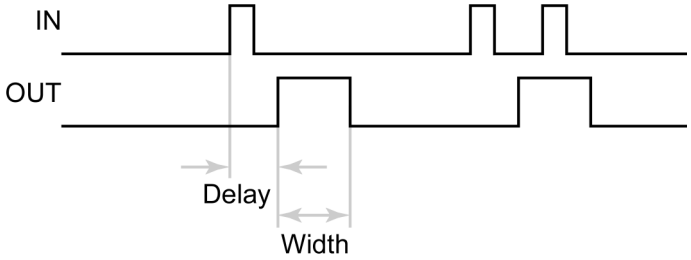
Put together, the parts of the Enhanced Function Block let you create very complex functionality!

Pulse Generator

The Pulse Generator lets you create a pulsed digital signal with a configurable frequency and duty cycle. You can configure the Pulse Generator to emit a continuous (periodic) pulse.



You can also configure the Pulse Generator to emit a single pulse after receiving an input trigger signal. The Pulse Generator outputs the low section of the pulse first, which lets you include a built-in delay. Subsequent inputs are ignored until the Pulse Generator completes its pulse.



Depending on the model, your iPORT IP Engine may have up to four Pulse Generators, numbered 0 through 3.

Pulse Generator formulas

$$durationOfHigh = (granularity + 1) \times width \times 30ns$$

$$durationOfLow = (granularity + 1) \times (delay + 1) \times 30ns$$

$$pulsePeriod = durationOfHigh + durationOfLow$$

$$pulseFrequency = \frac{1}{pulsePeriod}$$

Pulse Generator settings

Width (high)
The relative duration of the high section of the pulse. A value of 0 is treated as 1.

Delay (low)
The relative duration of the low section of the pulse. A value of 0 is treated as 1.

Granularity factor
A multiplier used to scale the **Width (high)** and **Delay (low)** in increments of 30 ns.

Emit periodic pulse
When checked, the Pulse Generator emits a continuous signal; when unchecked, the Pulse Generator emits pulses only when triggered by an input signal. The input signal for each Pulse Generator is fixed; see “The PLC at a glance” on page 75.

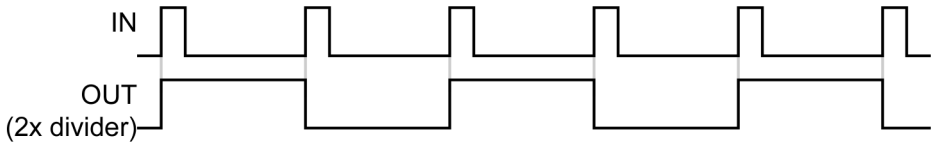
Trigger mode
The event that causes the Pulse Generator to emit pulses. If the mode is an edge, the Pulse Generator emits a single pulse; if the mode is a high or low state, the Pulse Generator emits pulses for as long as the condition is true. The Pulse Generator emits a complete pulse, even if the trigger condition becomes false before completion.

Pulse period (ns)
The duration of the full pulse. (read only)

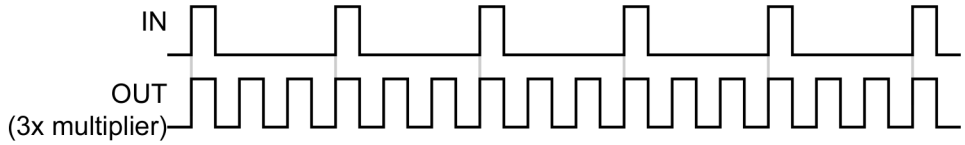
Pulse Frequency (Hz)
The number of pulse repetitions per second. (read only)

Rescaler

The Rescaler lets you change the frequency of a periodic input signal. You can use the Rescaler to multiply the period by up to 4096 or divide it by up to 4095.

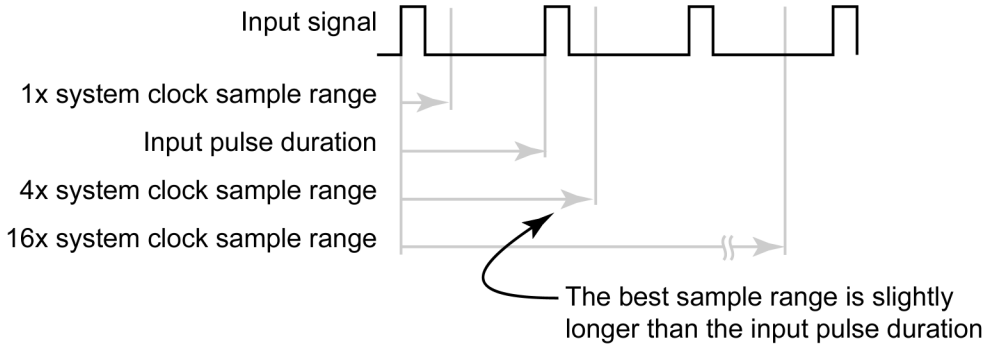


The Rescaler samples the frequency of the input signal, calculates the new output frequency, and emits a clock with a 50% duty cycle.



In input and output signals will not always align with each other. For example, an input period that is rescaled by 31.8 will rarely coincide with the output.

During its sample range, the Rescaler samples the input signal up to 65536 times (for 16-bit Rescalers). Based on its samples, the Rescaler determines the period of the input signal. To accurately sample your input signal, the sample duration should be as close as possible to the input period while still being longer.



The Rescaler samples at a rate determined by the **Granularity**; the maximum number of samples the Rescaler can take is determined by the **Rescaler Size**

Rescaler formulas

$$timeBetweenSamples = granularity \times 30ns$$

(The *granularity* is 1, 4, 16, or 256 system clock cycles)

$$maximumNumberOfSamples = 2^{rescalerSize}$$

(The *rescalerSize* is 12 or 16)

$$outputFrequency = \frac{inputFrequency \times multipler}{divider}$$

$$maximumSampledPeriod = granularity \times maximumNumberOfSamples \times 30ns$$

$$delayToSwitchOver = backupWindow \times 480ns$$

Rescaler settings

Granularity
The duration between samples, measured in system clock cycles (30 ns).

Multiplier
Divider
Variables that let you scale the frequency of the output signal.

Input signal
The source of the input signal to be rescaled.

Backup enabled
When enabled, the Rescaler switches to the **Backup input signal** should the **Input signal** stop.

Backup switchover delay
The duration of the delay to switchover, in increments of 480 ns. If the Rescaler detects no activity from the **Input signal** for *backupWindow* * 480 ns, it switches to the backup signal.

Backup input signal
The source of the replacement signal to be used if the **Input signal** stops. The **Backup input signal** isn't rescaled.

Input frequency
The pulse rate of the **Input signal**.

Output frequency
The pulse rate of the output from the Rescaler.

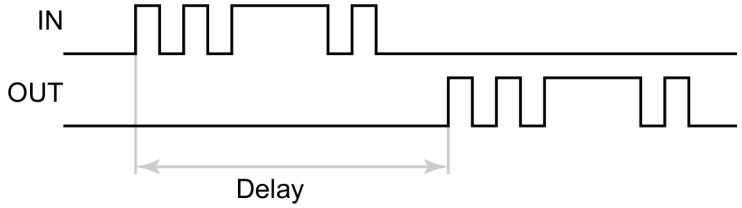
Target frequency
A calculator that reads the Input frequency and recommends settings for **Granularity**, **Multiplier**, and **Divider**. The value of the Target frequency doesn't affect the operation of the Rescaler.

Recommended granularity
Recommended multiplier
Recommended divider
Suggested settings based on the value set in the **Target frequency**.

Rescaler sample size
The maximum number of samples the Rescaler can take of the input signal before calculating the pulse width. The more samples the Rescaler makes, the more accurate the calculated pulse width. The 16-bit Rescaler takes up to 65356; the 12-bit, 4096.

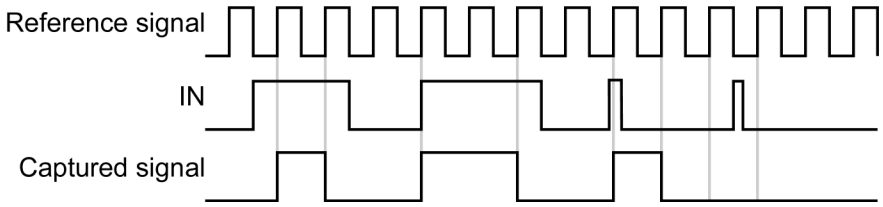
Delayer

The Delayer lets you delay an input trigger signal.

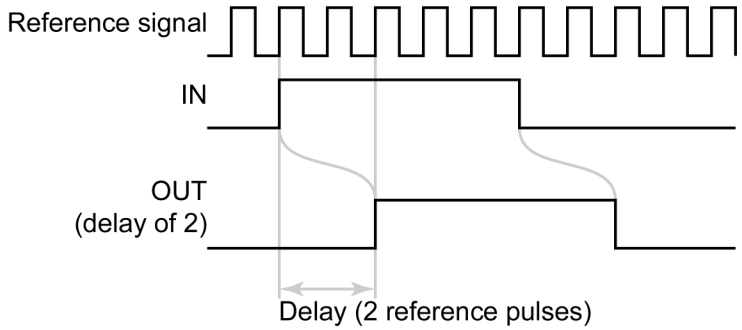


The Delayer uses the rising edges of a periodic reference signal for timing. On each rising edge of the reference signal, the Delayer samples the input signal. After *n* pulses (up to 65535) of the reference signal, the Delay outputs the sampled value. The Delayer can store up to 128 transitions (64 pulses).

Because the Delayer uses the reference timing signal for sampling, input pulses that are shorter than the period of the reference timing signal could be missed. Additionally, output granularity will be that of the reference timing signal.



To determine the length of the delay, select a reference signal and specify the number of rising edges that must occur before relaying the input trigger signal.



Delayer formulas

$$signalDelayTime = periodOfReferenceSignal \times delayCount$$

Delayer settings

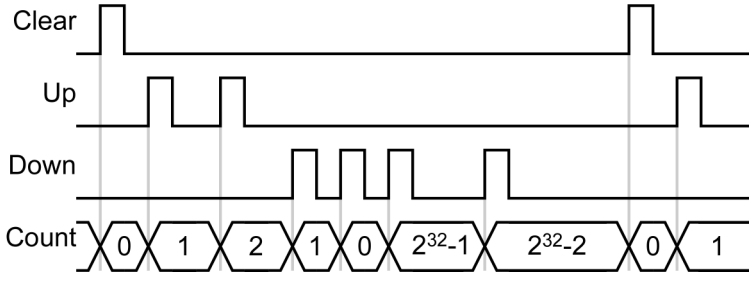
Delay count
The duration of the delay of the output signal, measured by the number of rising edges of the reference signal.

Reference timing signal
The source of the signal used for timing. The period of the reference signal determines both how often the Delayer samples the input and the duration of a delay count of 1.

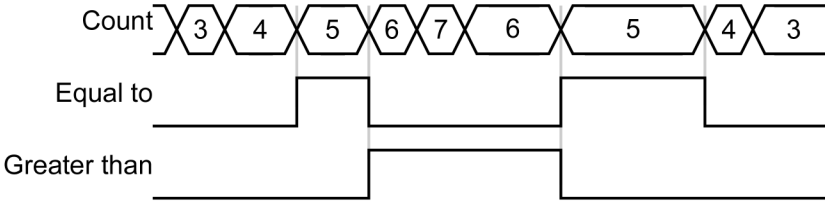
Input signal
The source of the signal to be delayed.

General Purpose Counter

The General Purpose Counter lets you maintain a count between 0 and 2^{32}-1 (long integer). You can use different inputs to increment, decrement, or clear the counter value.



The General Purpose Counter outputs two separate signals that indicate when the count is equal to and greater than the compare value that you set.



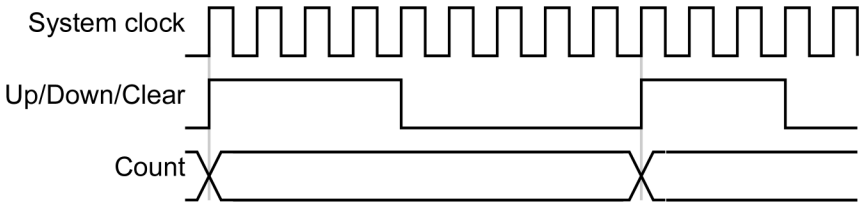
(Compare value of 5)

General Purpose Counter settings

Increment trigger mode
Decrement trigger mode
Clear trigger mode

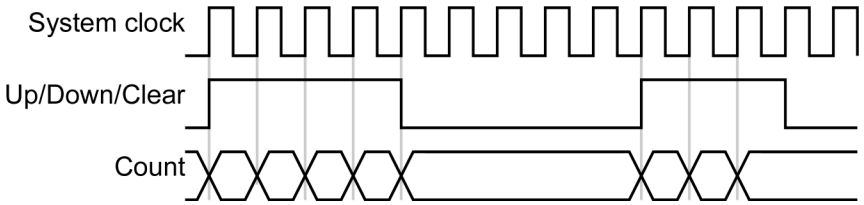
The states that cause the count to increment, decrement, and set to 0. You can change the count using edges (rising, falling, or both) or by level (high or low). If you use edge mode, the count changes when the edge occurs.

Change on rising edge



If you use level mode, the count changes with every system clock pulse.

Change on high level



Increments and decrements that occur simultaneously cancel each other. A clear trigger overrides other changes and sets the count to zero.

Clear signal

The input signal that triggers the clearing of the counter.

Compare value

The long integer to which the counter compares the **Current counter value**. The Counter outputs the result of the compare on the Greater Than and Equal To signals.

Current counter value

A read-only output of the current count. Coyote polls the IP Engine and updates the value every ~0.25 s.

Interrupt FIFO

The Interrupt FIFO lets you use the PLC to generate an interrupt on the host PC. The IP Engine sends an interrupt request (IRQ) packet containing the following:

- The values (state) of the four possible interrupt signals (IRQ_mask[3:0]);
- The time. The value can be the count of either General Purpose Counter or the Timestamp Counter (time[31:0]). To select a counter, use the "Counter select" configuration; see "Timestamp Counter" on page 65.
- The values (state) of I0-I7 in the Signal Routing Block (SRB_mask[7:0]).

To let the iPORT SDK act on the IRQ, you must register a callback function. That callback function will be invoked every time the PC receives an IRQ packet. See the CyDevice class in the *iPORT C++ SDK Reference Guide*.

Interrupt FIFO settings

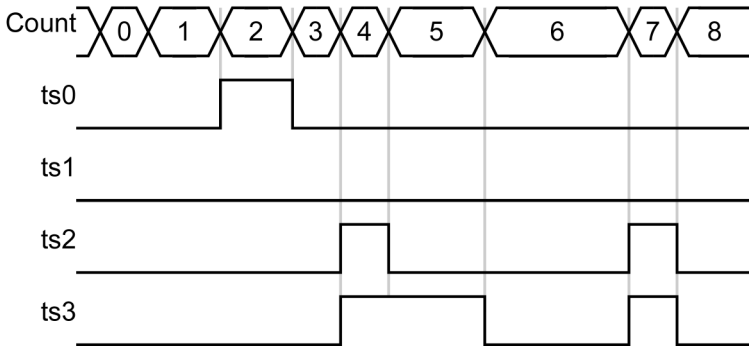
Q15 Enabled
Q3 Enabled
Q7 Enabled
Q10 Enabled

When enabled, a rising edge on the signal (Q15, etc.) generates an IRQ packet. The SRB mask correctly reflects the state of the four signals, even if some of them are disabled.

Counter Trigger Generator

The Counter Trigger Generator lets you configure the IP Engine to monitor a counter and emit a signal while the count equals a desired value. When the count equals the set value, the Counter Trigger Generator emits a signal on up to four outputs. These signals feed into the Signal Routing Block and can be used to trigger other events.

```
Trigger list
  If Count=2 then ts[3:0]=[0, 0, 0, 1]
  If Count=4 then ts[3:0]=[1, 1, 0, 0]
  If Count=5 then ts[3:0]=[1, 0, 0, 0]
  If Count=7 then ts[3:0]=[1, 1, 0, 0]
```



The Counter Trigger Generator lets you set up to 32 simultaneous count values for monitoring. Once an event occurs, it is automatically removed from the pool. You can monitor the count from either the General Purpose Counter or the Timestamp Counter. You can also switch back and forth between the two. However, to ensure that no events are accidentally lost, the trigger list must be empty before switching from one counter to another.

The Counter Trigger Generator uses a first-in, first-out algorithm, so a trigger list of 4, 82, 8, would require the clock to roll over before acting on the count value of 8.

The Counter Trigger Generator must be set from the iPORT SDK; see the *iPORT C++ SDK Reference Guide*.

Timestamp Counter

The Timestamp Counter is a configurable counter based on the IP Engine's system clock.

Timestamp Counter settings

Counter select
The signal used by the Interrupt FIFO. The time is also included in the footer of images sent from the IP Engine to the PC. (The counter selection for the Counter Trigger Generator is an independent setting made using the SDK).

Granularity
The amount of time that passes between counter increments. The minimum granularity is 480 ns, or 16 pulses of the 30 ns system clock. Because granularities of 1 us, 100 us, and 10 ms aren't evenly divisible by 30 ns, the IP Engine prevents drift by adjusting the number of system clock pulses between counter increments. For example, by cycling through increment times of 0.990 us, 0.990 us, and 1.020 us, the clock maintains an average granularity of 1.000 us. The variance between individual counter increments is +/- 20 ns; the maximum counter variance is +/- 10 ns.

Set trigger mode
Clear trigger mode
The events that cause the **Current counter value** to change to the **Set value** or to 0.

Set input signal
Clear input signal
The signal used to set the counter to **Set value** or 0.

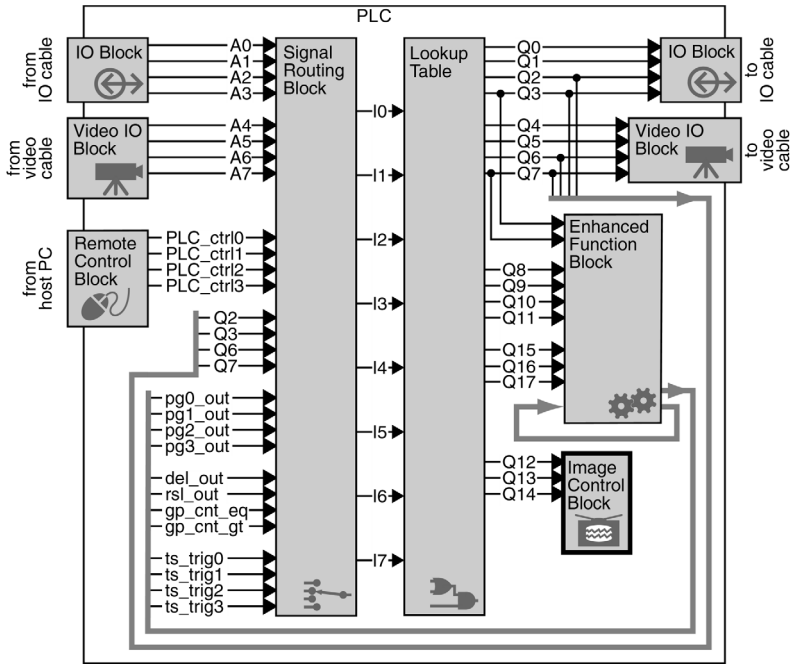
Broadcast
When enabled, you can set or clear the counters of all networked IP Engines simultaneously. To use it, configure **Set mode**/**Clear mode** to **On Apply**. When you click **Apply** in the **Configuration** dialog, the PC sends out a packet containing the new counter value. Latency and variability depend on your networking equipment.

Set counter value
The value to be set in the counter when the set event occurs.

Current counter value
The current value of the counter. (Read only)

Image Control Block

The Image Control Block lets you configure the image LVAL, FVAL, and TRIG signals used by the IP Engine's image grabber. You can use the default LVAL and FVAL signals exactly as they come from the camera or manipulate them to create your own signals.



The Image Control Block is particularly useful with line-scan cameras because you can control precisely how you grab your image.

Unlike other blocks within the PLC, the Image Control Block doesn't have any outputs; the massaged LVAL, FVAL, and TRIG signals are used only by the IP Engine's image grabber.

Modifying FVAL (Q12) and LVAL (Q13)

The PLC lets you modify the polarity, edge sensitivity and behavior of the FVAL and LVAL signals. See "Configuring the Image Control Block" on page 29.

Modifying TRIG (Q14)

The PLC lets you trigger the image grabber using a signal you send to Q14. You must enable the **PLC Triggerable** setting for the signals on Q14 to have an effect. See "Configuring the Image Control Block" on page 29.

Real world samples

The samples here give you an idea of how you can use the PLC. Before reading this section, consider reading “Demonstrating the PLC in 10 minutes” on page 13. To make the settings described here, see “Configuring the PLC with the iPORT Vision Suite” on page 27.

In this section:

Triggering an area-scan camera with a presence detector 69
Acquiring a 300 DPI image with a line-scan camera 70

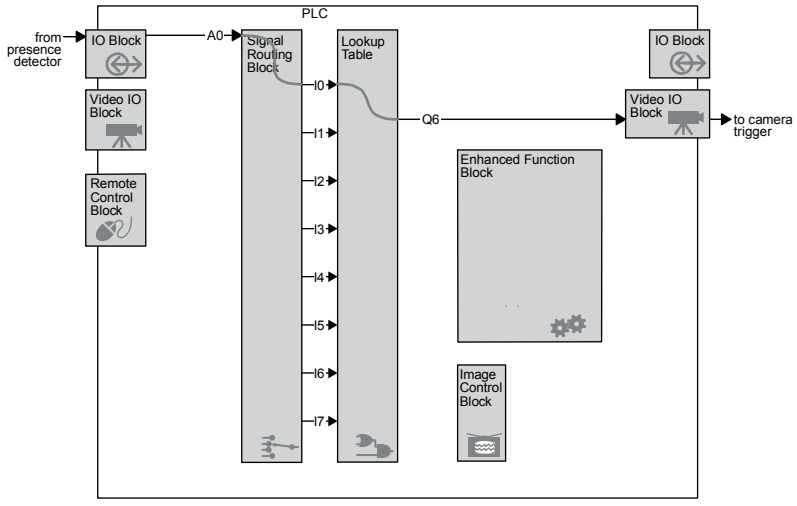
Triggering an area-scan camera with a presence detector

This sample is for using an area-scan camera to capture an image of an object moving on a conveyor belt. When the object passes in front of the presence detector, the PLC requests an image from the camera. The camera must be set to external trigger mode.

Instead of running the signal from your presence detector directly to the camera trigger, you can run it to your IP Engine and reroute signal using the PLC. By making the IP Engine the hub of your cabling, you’ll reduce complexity and reduce the risk of wiring errors.

Should you find the presence detector isn’t where you want it, you can easily incorporate a delayer that lets you precisely adjust how long to wait before acquiring an image. See “Demonstrating the Delayer” on page 19.

PLC settings

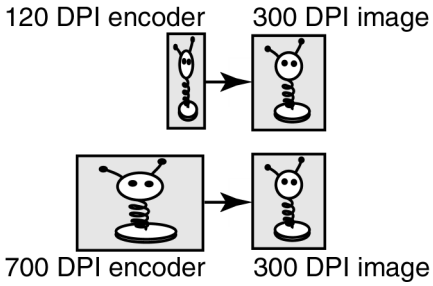


Signal Routing Block
I0: TTL Input 0

Lookup Table
Q3=I0

Acquiring a 300 DPI image with a line-scan camera

This sample is for using a line-scan camera to capture an image of an object on a conveyor belt. Unlike an area scan camera, the line-scan camera must be told when to capture each line. Typically, that signal could come from the conveyor belt's encoder. However, if your encoder outputs pulses at say, 700 DPI or 120 DPI, converting that signal to a 300 DPI signal could be frustratingly complex.

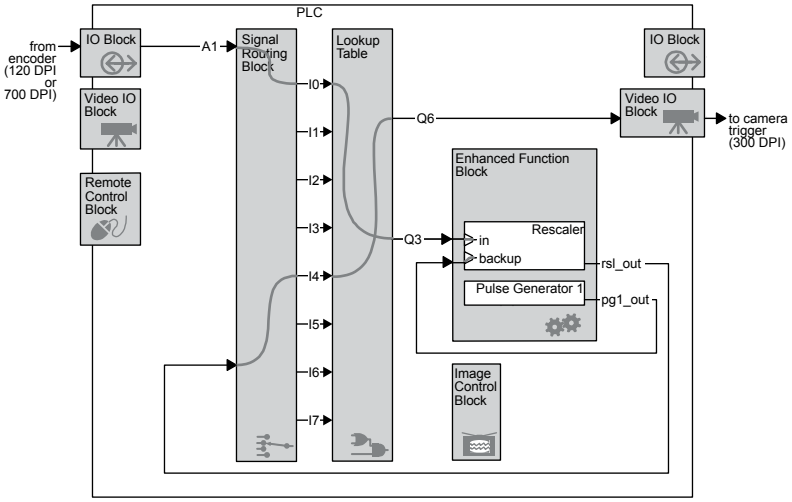


The Rescaler lets you easily divide or multiply signals, even if they're not simple ratios of each other. Since the final output is scaled from the encoder input, the result will always be 300 DPI, even when the conveyor speeds up or slows down.

The system also includes a backup signal that lets you continue to capture images, even if the encoder fails. If your conveyor belt usually moves at a rate of about 20in/s, you could set the backup pulse to 6000 Hz (300 dots/inch times 20 inches/s). The result wouldn't be *exactly* 300 DPI, but you could still capture usable images without requiring downtime to fix the failed encoder.

PLC settings

The settings below presume you have the PLC connected to an encoder and a line-scan camera set to external trigger mode.



Signal Routing Block
- I0: TTL Input 1
- I4: Rescaler 0 Output

Lookup Table
- Q3=I0
- Q6=I4

Pulse Generator 1
- Width (high): 925
- Delay (low): 925
- Granularity factor: 2
- Emit periodic pulse: True
- Trigger mode: NA (any setting is okay)

Pulse frequency (Hz): 6002 (read only)

Rescaler 0 (settings for 120 Hz input)
Granularity: 1 system clock cycle
Multiplier: Frequency X 4096
Divider: 1638
Input signal: Q3
Backup enabled: True
Backup switchover delay: 4095
Backup input signal: Pulse Generator #1
Input frequency: ~120 Hz (read only)
Output frequency: ~300 Hz (read only)

Rescaler 0 (settings for 700 Hz input)
Granularity: 1 system clock cycle
Multiplier: Frequency X 256
Divider: 597
Input signal: Q3
Backup enabled: True
Backup switchover delay: 4095
Backup input signal: Pulse Generator #1
Input frequency: ~700 Hz (read only)
Output frequency: ~300 Hz (read only)

Glossary of signal names

Below is an alphabetically arranged summary of the PLC signal names with brief descriptions and links that give you more information.

PLC signal names and descriptions

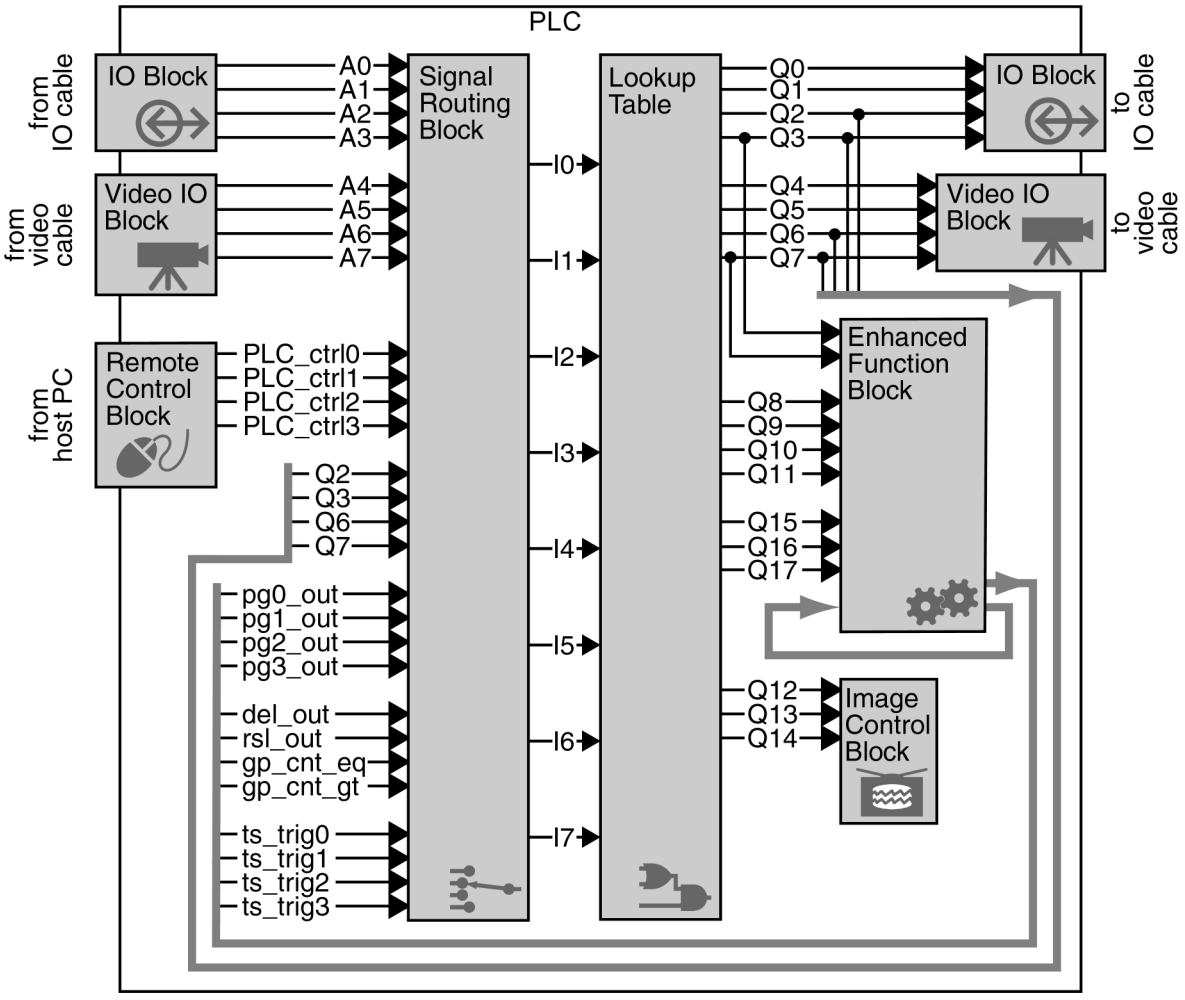
Signal	Description	See
A0 - A3	Signals from the IP Engine's IO port (after synchronizations and optional debouncing)	"IO Block" on page 39
A4 - A7	Signals from the camera (after separating them to produce LVAL/FVAL/etc. signals)	"Video IO Block" on page 47
CC1 - CC4	Camera control signals for LVDS cameras	"Video IO Block (for LVDS imaging data)" on page 49
CL_CC1 - CL_CC4	Camera control signals for Camera Link cameras	"Video IO Block (for Camera Link imaging data)" on page 48
CL_FVAL CL_LVAL CL_DVAL CL_SPR	Camera signals from Camera Link cameras	"Video IO Block (for Camera Link imaging data)" on page 48
del_out	Delayer output signal	"Delayer" on page 61
DVAL	Data valid signal (generic)	"How the Video IO Block works" on page 47
FVAL	Frame valid signal (generic)	"How the Video IO Block works" on page 47
gp_cnt_gt	General Purpose Counter 0 "greater than" output signal	"General Purpose Counter" on page 62
gp_cnt_eq	General Purpose Counter 0 "equal to" output signal	"General Purpose Counter" on page 62
gp_cnt[31:0]	General Purpose Counter 0 "Current counter value" signals (31 is MSB, 0 is LSB)	"General Purpose Counter" on page 62
I0 - I7	Signal Routing Block output signals	"Signal Routing Block" on page 53
IRQ_mask[3:0]	State of the interrupt request signals at the time of the interrupt request (From 0 to 3, the values are Q3, Q7, Q10, and Q15)	"Interrupt FIFO" on page 64
LVAL	Line valid signal (generic)	"How the Video IO Block works" on page 47
LVDS_IN*n*	Low voltage differential signal (LVDS) input to the IO Block	"LVDS Input Block" on page 43

PLC signal names and descriptions

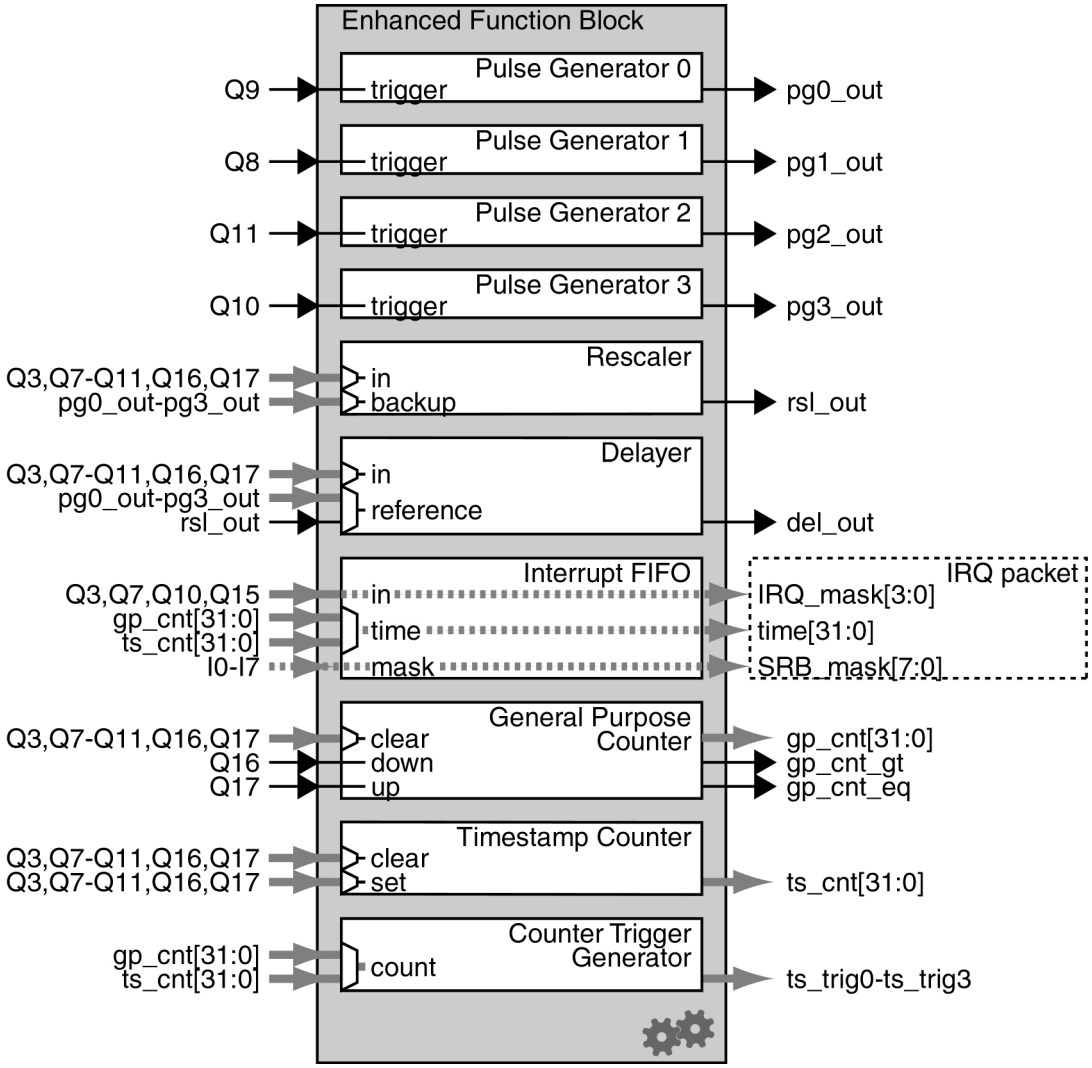
Signal	Description	See
OPT_IN*n*	Optically isolated input to the IO Block	“Optically Isolated Input Block” on page 44
OPT_OUT*n*	Optically isolated output from the IO Block	“Optically Isolated Output Block” on page 45
pg0_out - pg3_out	Pulse Generator 0 - 3 output signals.	“Pulse Generator” on page 59
Q0 - Q17	Lookup Table output signals	“Lookup Table” on page 55
PLC_ctrl0- PLC_ctrl3	PLC control bits	“Manually controlling your circuit with the Remote Control Block” on page 28
rsl_out	Rescaler output signal	“Rescaler” on page 60
SRB_mask[7:0]	State of the Signal Routing Block signals at the time of the interrupt request (From 0 to 7, the values are I0 to I7)	“Interrupt FIFO” on page 64
SPARE	Spare signal defined by the Camera Link standard	“How the Video IO Block works” on page 47
time[31:0]	State of the selected counter at the time of the interrupt request (The counter can be gp_cnt[31:0] or ts_cnt[31:0])	“Interrupt FIFO” on page 64
TRIG	Signal used to trigger the acquisition of an image by the IP Engine’s grabber (generic)	“Image Control Block” on page 67
ts_cnt[31:0]	Timestamp Counter output (31 is MSB, 0 is LSB)	“Timestamp Counter” on page 65
ts_trig0 - ts_trig3	Counter Trigger Generator outputs (set by the SDK when the trigger is configured)	“Counter Trigger Generator” on page 64
TTL_IN*n*	TTL input to the IO Block.	“TTL Input Block” on page 42
TTL_OUT*n*	TTL output from the IO Block.	“TTL Output Block” on page 43
VID_FVAL VID_LVAL VID_RTS1 VID_FID0	Camera signals from analog cameras	“Video IO Block (for analog video)” on page 48

The PLC at a glance

Programmable Logic Controller



Enhanced Function Block



PLC and Enhanced Function Block (on one page)