PLEORA TECHNOLOGIES INC.

iPORT™ Advanced Features
User Guide

Pleora Technologies

Document Number

EX001-000-0003, Version 2.0, 7/8/14

Table of Contents

Chapter 1

About this Guide

This chapter describes the purpose and scope of this guide and provides a list of complimentary guides.

The following topics are covered in this chapter:

About this Guide

This user guide provides you with the information you need to configure Pleora's powerful, advanced video interface features, which allow you to control and synchronize the external devices in your vision system solution.

Using Pleora's configuration tool, eBUS Player, you can trigger, route, time, and add data to the general purpose inputs and outputs (GPIO) signals that interface to camera heads and industrial sensors.

Applicable Products

The advanced features are available in the following Pleora iPORT products:

- iPORT CL-Ten External Frame Grabber
- iPORT NTx-Ten Embedded Video Interface
- iPORT NTx-U3 Embedded Video Interface
- iPORT NTx-GigE Embedded Video Interface
- iPORT SB-U3 External Frame Grabber
- iPORT SB-GigE External Frame Grabber

This guide describes all of Pleora's video interface advanced features. Depending on the video interface you are using and the firmware version on the video interface, you may not have access to the full set of advanced features. Please refer to the corresponding user guide for your video interface to view the list of advanced features supported.

Related Documents

The iPORT Advanced Features User Guide is complemented by the following guides:

Table 1: Related Documents

Guide	Details	Consult this Guide When...
eBUS Player Quick Start Guide	Provides introductory information to familiarize you with the eBUS Player software application and provides initial setup steps.	You are using eBUS Player for the first time and want information about performing common tasks.
eBUS Player User Guide	Provides in-depth details about setting up and using the eBUS Player software application to control your GigE Vision compliant video transmitters (cameras) and receivers.	You are familiar with eBUS Player and want to know how to fully configure your IP engine using all of eBUS Player's functionality.
eBUS SDK Programmer's Guide	Provides information and instructions to developers who are integrating the Pleora Technologies' eBUS SDK with their own application in order to communicate with Pleora's GigE Vision compliant products. In this guide, you can review information about the underlying technology being used, along with a high-level view of how the eBUS SDK fits into the system and your application.	You are planning to integrate your software with the Pleora eBUS SDK and want to view sample code and including Application Programming Interface (API) class descriptions and design guidelines.

Chapter 2

About the Advanced Features

This chapter provides an overview of Pleora's video interface advanced features and typical applications.

The following topics are covered in this chapter:

Introducing Pleora's Video Interface Advanced Features

Pleora's video interface advanced features are available in many of Pleora Technologies' embedded video interfaces and external frame grabber products, allowing more camera and GPIO control through simplified configuration.

The advanced features provide the following functionality:

- Simple boolean-based programming for routing of GPIO, camera control, video sync and internal signals
- PLC enhanced functionality:
  - Timers
  - Counters
  - Delayer
  - Rescaler
- Input re-synchronization
- Input debouncing
- Event queues
- Action commands
- Chunk generators

As mentioned in the previous chapter, the available advanced features differ for each video interface. Depending on the Pleora video interface you are using and the firmware version on the video interface, you may not have access to the full set of advanced features. Please refer to the corresponding user guide for your video interface to view the list of advanced features supported.

Typical Advanced Feature Applications

Pleora's video interface advanced features are used in various vision system environments; the following list captures the typical applications:

- Image acquisition control
- Frame acquisition
- Shutter control
- Trigger control
- Trigger source selection (external, software, internal)
- Trigger delay
- Synchronized action commands using IEEE 1588
- Strobe control
- Exposure control
- Readout control
- Iris control

Configuring the Advanced Features

The configuration instructions in this guide are written for Pleora's eBUS Player application, which is available as part of the eBUS SDK, available on the installation CD, or at http://www.pleora.com/support-center. Once you are familiar with the advanced features, you have the option to create your own configuration tool using the Pleora eBUS SDK.

For detailed configuration steps, see "Advanced Features Usage Examples" on page 53.

Chapter 3

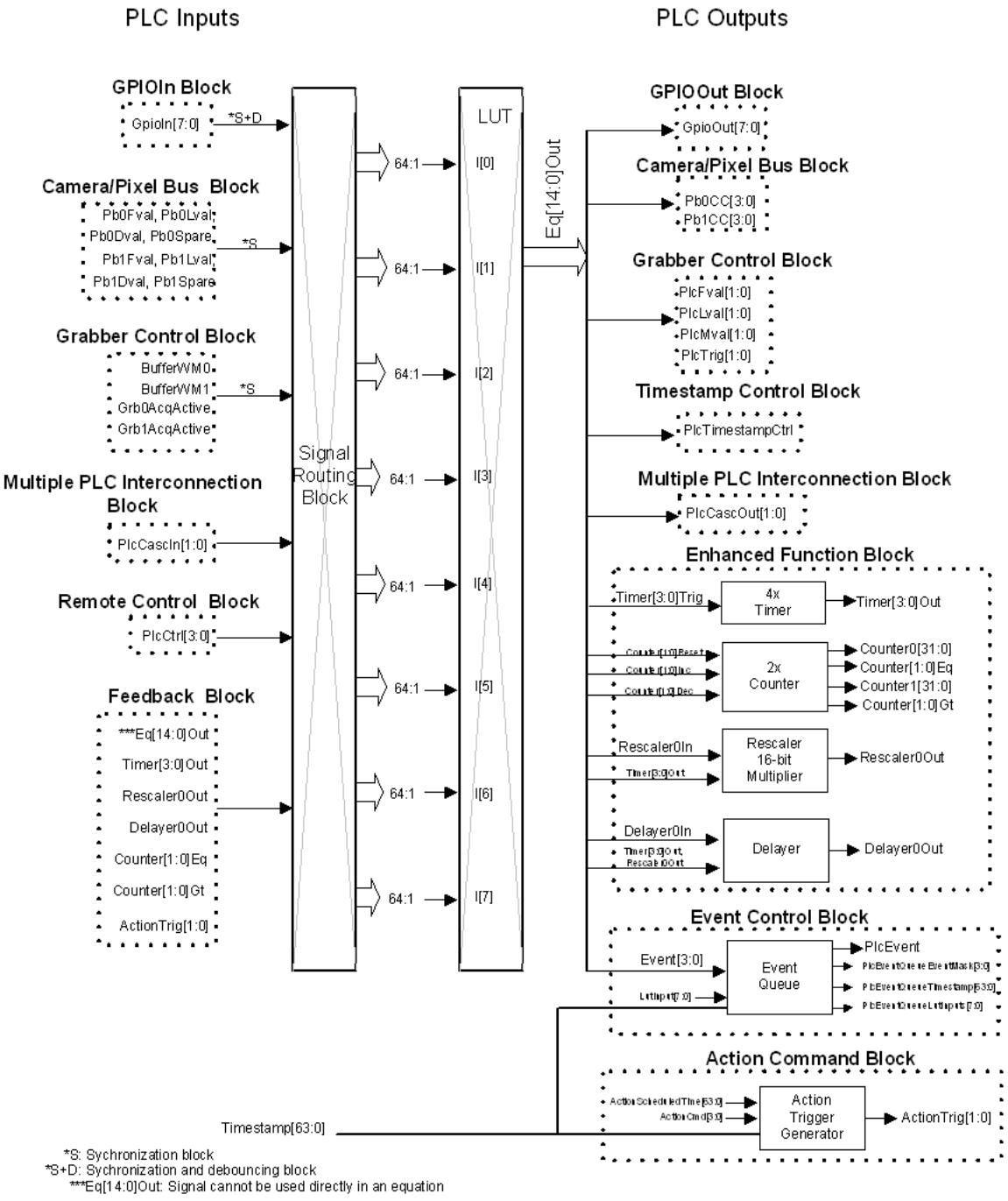
Programmable Logic Controller (PLC)

This chapter describes the PLC block components and signals that you can route to synchronize the elements of your vision system.

The following topics are covered in this chapter:

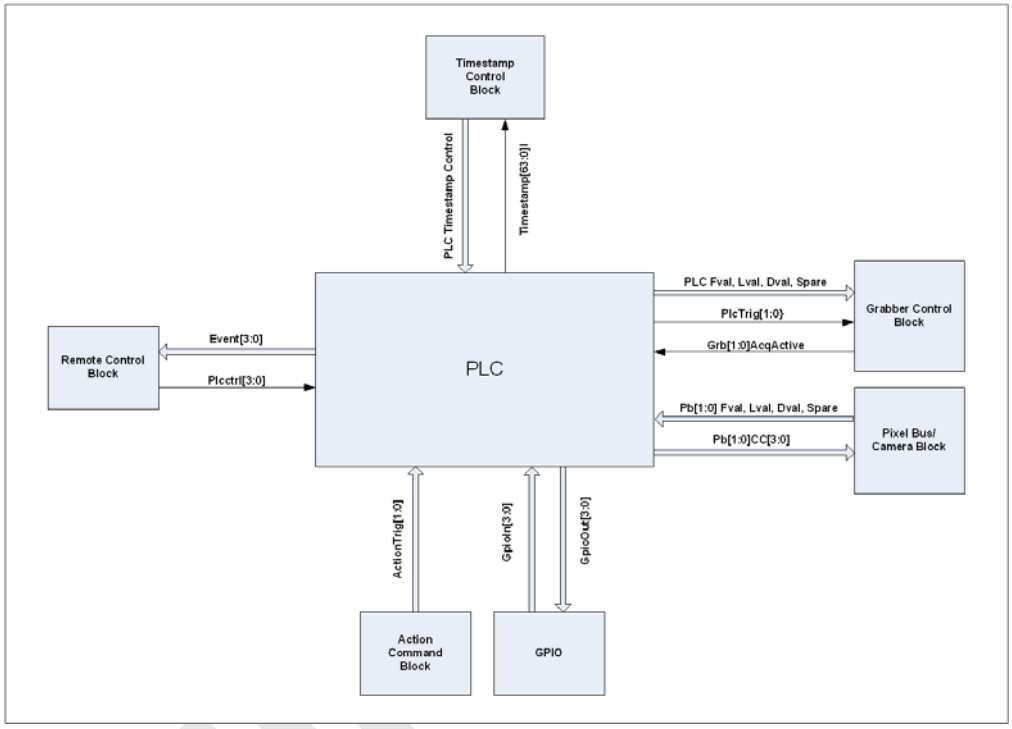
PLC Signal Routing

The following diagram provides you with a detailed view of how the video interface accepts signals from various function blocks, routes them through the signal routing block, look-up table (LUT), enhanced function block, and finally to the external outputs.



Key PLC Functional Blocks and Signals

The following diagram provides a high-level view of the key PLC functional blocks and signals.



Configuring the PLC using eBUS Player

You can use Pleora's eBUS Player, provided with the eBUS SDK, to enter simple Boolean equations to configure the enhanced features of your video interface.

This guide describes the eBUS Player functionality used to configure the PLC and other advanced features. For more detailed information about eBUS Player, see the eBUS Player User Guide.

Signal Routing Block and the LUT

The signal routing block consists of 8 identical 64-to-1 multiplexers, which correspond to each of the 8 LUT inputs. The signal routing block supports up to 64 inputs including signals from a second video source, and a second PLC.

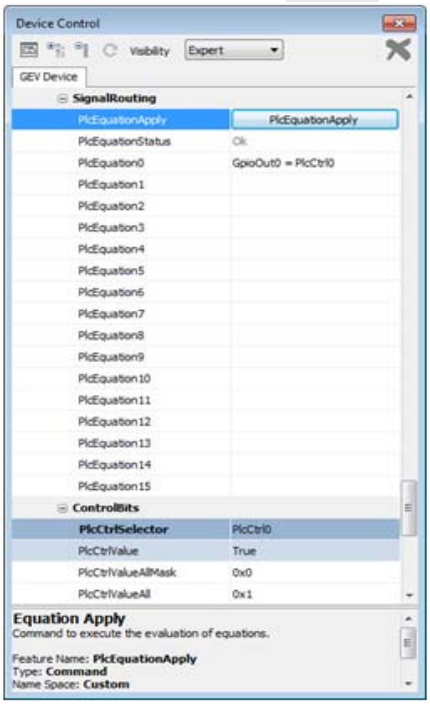
The LUT allows you to program up to 15 logical PLC functions, using a maximum of 8 shared inputs for all of the equations. Each logical function is defined through a Boolean equation. The equations control how the signals are routed through the signal routing block to the LUT, and through the various function blocks.

Typically, in vision systems, the signal routing block routes signals to the enhanced function blocks and then feeds the signals back to the inputs (feedback inputs) for re-routing to the outputs.

To configure the signal routing block equations

1. Start eBUS Player and connect to the video interface.
2. Under **Parameters and Controls,** click **Device control**.
3. Click **Guru** in the **Visibility** list.

1. In the **SignalRouting** section, select one of the available equations and enter the Boolean equation required to route the appropriate signals.
2. Click **PlcEquationApply** to apply the equation.



Equation Status

The **PlcEquationsStatus** feature displays the first of any errors that occur as the video interface processes the Boolean equations. The status appears if errors occur after you click the **PlcEquationApply** button.

If no errors occur, the status displays **OK**.

The following table provides descriptions of the possible equation errors displayed in the **PlcEquationsStatus box.**

Table 2: Equation Error Status Descriptions

Equation Error Status	Description
LUT now out of sync with PlcEquations: PlcEquationApply required	The equations have been updated and are no longer synchronized with the LUT contents. You must click PlcEquationApply to fix this error.
PlcEquation <equation number>: identifier too large at column <column number	The identifier used for the equation, identified in the error message by number, is too large
PlcEquation <equation number>: Undefined identifier <string> at column <column number	The equation does not include a valid signal name
PlcEquation <equation number>: Output signal is already assigned	The equation at that location is attempting to use an output that has already been assigned a value
PlcEquation<equation number>: = expected at column <column number	The equation at that location is missing an “=”
PlcEquation <equation number>: Number of expression elements exceeds maximum number of <element limit>	The equation at that location contains too many expression elements. You need to simplify the expression.
Too many input variables in use at PlcEquation <equation number	Number of LUT input variables exceeds maximum possible LUT inputs
PlcEquation <equation number>: expression too complicated for PLC	The equation at that location is too complicated to be evaluated. You need to simplify the expression.
PlcEquation <equation number>: illegal expression	The equation at that location does not have valid syntax
PlcEquation <equation number>: No matching ("	The equation at that location is missing a “(“
PlcEquation <equation number>: Unmatched (at column <column number>	The equation is missing a “)” to match the “(“ at that location

General Syntax Rules

The following table provides rules you must follow for entering syntax. Input signals are shown as I0-I7, and output signals are shown as Q0-Q17. The length of each expression is limited to 1024 bytes. Comments can be added to the end of expressions by using // before the comment.

You can comment out full equations by adding // to the beginning of the equation.

Table 3: Syntax Rules

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	Q1 = I7 & I8
	Q2 = Q1 | I5	Q2 = (I7 & I8) | I5
Equations must be separated by a carriage return or an EOL symbol.	Q3 = I7,Q15=I8	Q3 = I7
		Q15 = I8

Example Equations

Your video interface can process up to 15 advanced feature equations at one time. eBUS Player provides configuration fields for equations 0-14.

The following table provides some example equations you can enter.

Table 4: Example Equations

Feature	Equation
PlcEquation0	GpioOut0 = GpioIn0 // Comments
PlcEquation1	Timer0Trig = GpioIn1 & PlcCtrl0
PlcEquation2	Pb0CC0 = (Timer0Out & PlcCtrl1) | (Timer0Trig & !PlcCtrl1)

Equation Definitions

The sections provide descriptions for the elements of equations:

Equations

Equations are written using the following format:

Variable out = expression

You can use an equation empty by using NULL in place of the equation.

Expressions

An expression can be made up of the following elements:

- Primary operator
- Unary operator
- Binary operator

Primary Operators

A primary operator can be made up of the following elements:

- Literal Operator
- Variable_In
- Expression

Literal Operators

The following table provides descriptions for Literal operators.

Table 5: Literal Operator Descriptions

Operators (Increasing Precedence)	Description
1	True / On
0	False / Off

Unary Operators

The following table provides descriptions for unary operators

Table 6: Unary Operator Descriptions

Operator	Description
!	Logical Negotiation

Binary Operators

The following table provides descriptions for binary operators.

Table 7: Binary Operator Descriptions

Operators (Increasing Precedence)	Description
&	Logical AND
|	Logical OR
^	Logical XOR

Variable Out

Defined names for all PLC outputs or enhanced functions inputs.

Variable In

Defined names for all PLC LUT inputs. A variable_in can also be equal to variable_out.

Null Operator

A Null operator is an empty string.

Signal Routing Block Boolean Expression Syntax

The following table provides examples of the Boolean expressions used to configure the Signal Routing block.

Table 8: Signal Routing Block Boolean Expression Syntax

Syntax	Valid Construction	Example Line
Line	*Output* = *Expression* EOL (end of line)	
Boolean operators	& (and)	Q14 = I4 & I6
	| (or)	Q15 = I3 | I5
	^ (xor)	Q9 = I1 ^ I8
Not	! (not)	Q0 = !I0
		Q10 = !(I8 & I5)
Delimiter	( )	Q0 = !(I0)
		Q3 = !(I1 | (I7 ^ I5))
		Q6 = (I3 | I5) ^ (I1 & I2)
Boolean constants	1, true, TRUE	Q0 = 1

Table 8: Signal Routing Block Boolean Expression Syntax

Syntax	Valid Construction	Example Line
	0, false, FALSE	Q3 = TRUE
		Q6 = I3 ^ true
EOL	\r	
	\n	
	\r\n	
	\n\r	
Output	Q0, Q1, Q2, ..., Q16, Q17	
Input	I0, I1, I2, ..., I6, I7	
Expression		Q1 = I5//input
		Q1 = !I5//not input
		Q1 = FALSE//Boolean constant
Combined *Expression*	*Expression* Boolean operator *Expression*	Q1 = I5 & I3
		Q16 = I8 | I6

Functional Blocks

This section provides descriptions of the PLC functional blocks.

GPIO Blocks

The GPIO blocks allow the video interface to communicate with exterior objects in your vision system through the 12-pin circular connector of the video interface.

Camera/Pixel Bus Block

The Camera/Pixel Bus block allows the communication between the camera and the video interface. The block receives a video signal and provides the video with discrete frame valid (FVAL), line valid (LVAL), and similar signals. Some cameras allow you to send control signals back to the camera through the Pb0CC[3:0] or Pb1CC[3:0] for the second video source.

Grabber Control Block

The grabber control block allows you to configure the image LVAL, FVAL, and TRIG signals used by the video interface'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. Unlike other blocks, the grabber control block doesn't have any outputs; the updated (manipulated) LVAL, FVAL, and TRIG signals are used only by the video interface's image grabber.

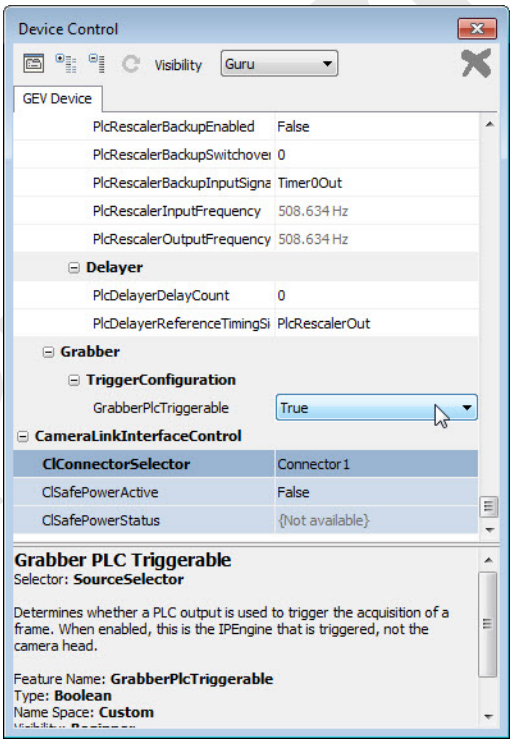
Configuring the Grabber Control Block

Using the eBUS Player Device Control dialog box, you can put the grabber control block into "trigger mode", which enables it to grab one image block when the rising edge of the **PlcTrig0** or the **PlcTrig1** signal occurs.

To put the grabber control block into trigger mode

1. Start eBUS Player and connect to the video interface.

   For more information, see the *eBUS Player User Guide*.
2. Under **Parameters and Controls**, click **Device control**.
3. Click **Guru** in the **Visibility** list.
4. In the **Grabber** section, select **True** from the **GrabberPlcTriggerable** list.



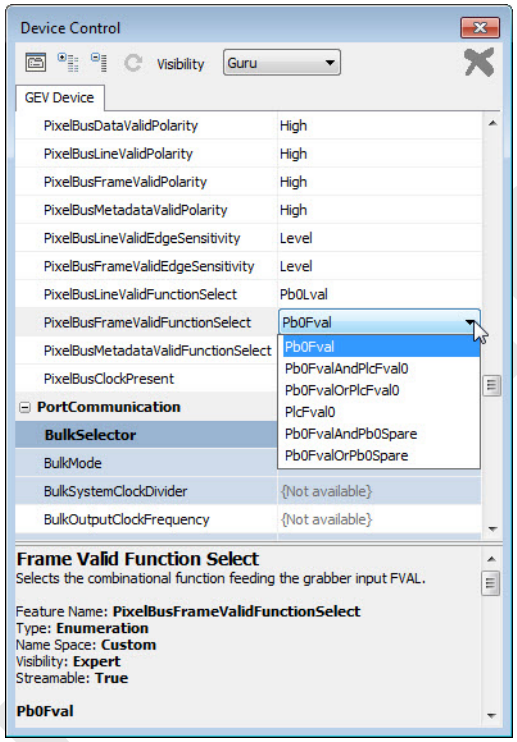
Modifying Grabber Control Block Signal Selection

Using the eBUS Player Device Control dialog box, you can configure the grabber control block to grab specified combinations or arrangements of signals from the PLC.

You can configure the selection of the following three signals:

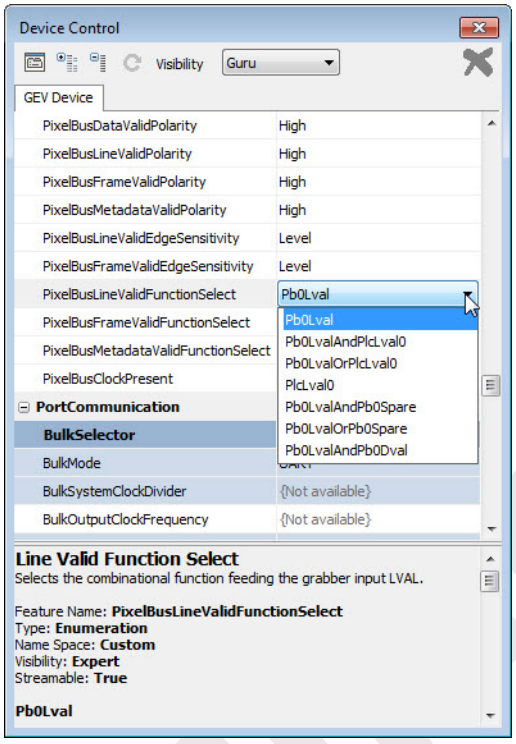
Grabber Input FVAL

- In the **PixelBusInterfaceControl** section, select **PixelBusFrameValidFunctionSelect**, and select from the options shown in the following image.



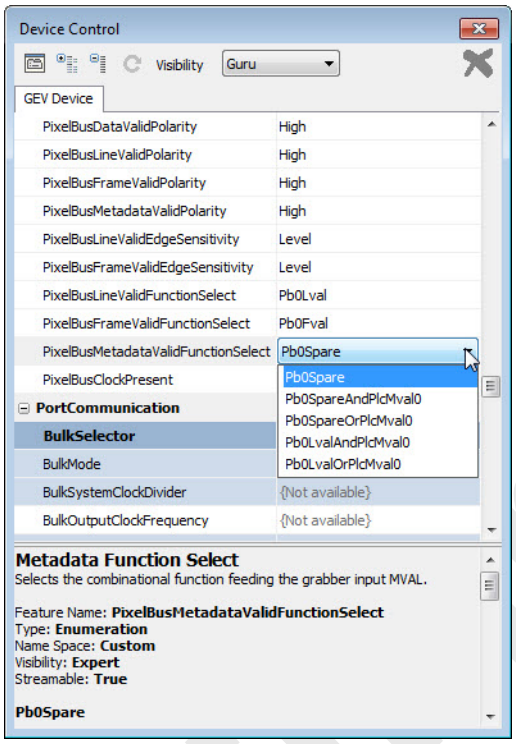
Grabber Input LVAL

- In the **PixelBusLineValidFunctionSelect** list, select from the options shown in the following image.



Grabber Input MVAL

- In the **PixelBusMetadataFunctionSelect** list, select from the options shown in the following image.



When you first open the **Device Control** dialog box, the default values for the signals as they are transmitted from the video source are shown.

Using the BufferWM0/BufferWM1 Signals

The acquisition block tracks the on-board memory used by the image block data that has not yet been transmitted on the Ethernet link. For example, when the data rate from the camera is higher than the Ethernet link's available bandwidth, memory usage increases. Memory thresholds are set using eBUS Player's Device Control dialog box. When the memory threshold is reached, the state of the BufferWM0/BufferWM1 signals become high. When memory usage is lower than the threshold, the state of the BufferWM0/BufferWM1 signals become low. These signals can be used to inform the video source to stop outputting data.

Using the Grb0AcqActive/Grb1AcqActive Signals

The Grb0AcqActive/Grb1AcqActive signals are a delayed version of the FVAL signal when the video interface is grabbing an image block. The PLC chunk data is captured at the first video interface system clock after the Grb0AcqActive/Grb1AcqActive signals are asserted. There is an approximate 8 to 12 pixel clock delay between the Grb0AcqActive/Grb1AcqActive signals and the Pb0Fval/Pb1Fval signals.

Multiple PLC Interconnection Blocks

The multiple PLC interconnection block can receive up to two outputs from a second PLC.

Remote Control Inputs

The remote control inputs let you send inputs from your host computer to the video interface. You can control this block using eBUS Player. You can configure each of the four control bits individually, or you can apply individual settings for multiple control bits at one time using the **PlcCtrlValueAll** and **PlcCtrlValueMask** feature.

To configure a remote control input

1. Start eBUS Player and connect to the video interface.

   For more information, see the *eBUS Player User Guide*.
2. Under **Parameters and Controls,** click **Device control**.
3. Click **Guru** in the **Visibility** list.
4. In the **ControlBits** section of the **Plc** category, select one of the inputs from the **PlcCtrlSelector** list.
5. In the **PlcCtrlValue** list, click **True** to set the signal state to high, or **False** to set the signal state to low.

To configure more than one remote control input at one time

1. In the **ControlBits** section of the **Plc** category, enter a value into the **PlcCtrlValueAll** box to indicate the signal state to apply to each of the four control bit inputs.

   For example, to set the **PlcCtrl0** and **PlcCtrl1** control bits to high while setting the **PlcCtrl2** and **PlcCtrl3** control bits to low, enter **0x3**.

   To have the expected bit set, the corresponding bit in **PlcCtrlValueMask** should be set to **1**; otherwise the bit state will not be changed.
2. In the **ControlBits** section of the **Plc** category, enter a formula into the **PlcCtrlValueMask** box.

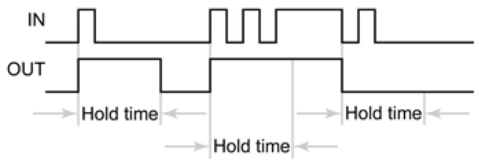
   For example, enter **0xF** to apply the **PlcCtrlValueAll** formula to all of the control bits.

Feedback Inputs

The feedback inputs accept signals from the advanced features, such as the timer, rescaler, delayer, counters, and the action trigger and routes them through the signal routing block to the appropriate outputs.

Input Debouncing Block

The input debouncing block filters spurious transitions from input signals. Each input signal can be independently configured to hold signal transitions between 480 ns and ~31 ms. While holding the first transition, the input debouncing block ignores further transitions for the configured duration.

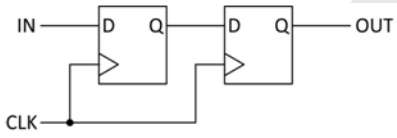


The hold times can be configured in increments of 480 ns (16 cycles of the video interface system clock). Setting the hold value to 0 disables the input debouncing block for that input signal.

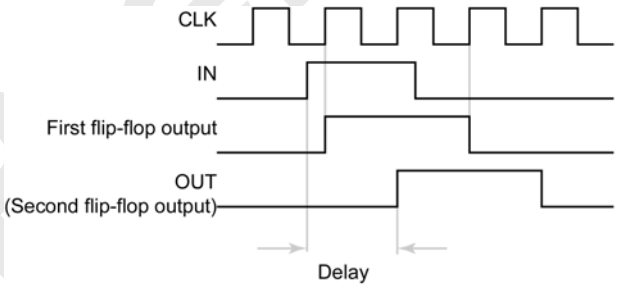
Hold time = Hold value * 480 ns

Input Synchronization Block

The synchronization block samples input signals in time with the video interface system clock. 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.

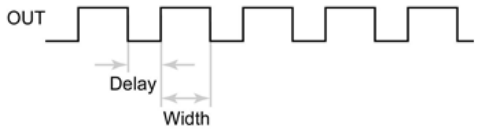


Enhanced Functional Blocks

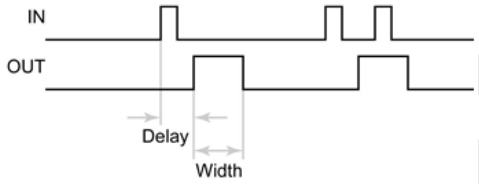
The following sections provide descriptions of the enhanced functional blocks.

Timer

The timer generates a pulsed digital signal with a configurated frequency and duty cycle.



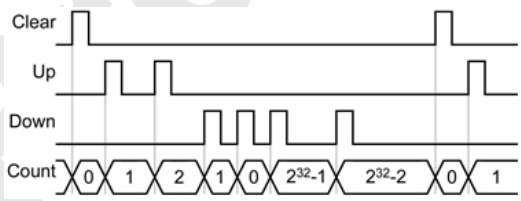
The timer can be configured to emit a continuous (periodic) pulse. The timer can also be configured to emit a single pulse after receiving an input trigger signal. It outputs the low section of the pulse first with a programmable built-in delay. Subsequent inputs are ignored until the timer completes its pulse.



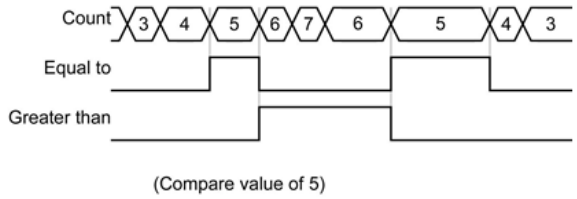
Depending on the video interface model, there may be up to four timers available, numbered 0 through 3.

Counter

The counter can maintain a count between 0 and 2^{32}-1 (long integer). Different inputs can be used to increment, decrement, or clear the counter value.

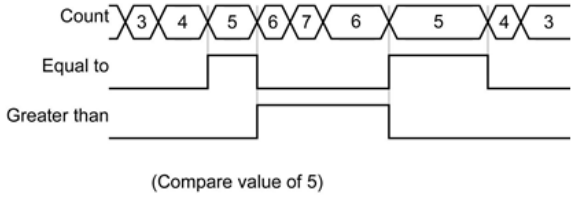


The counter outputs two separate signals that indicate when the count is equal to and greater than the compare value that can be set.

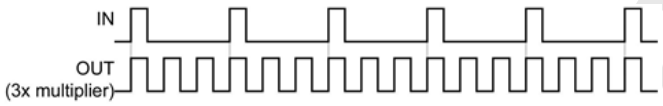


Rescaler

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

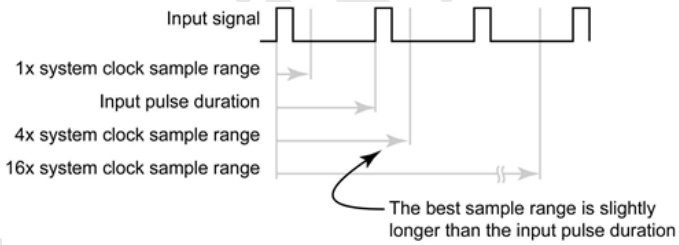


The rescaler samples the frequency of the input signal, calculates the new output frequency, and emits a clock with a 50% duty cycle.



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

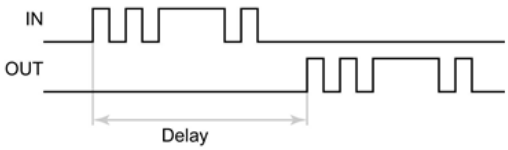
During its sample range, the rescaler samples the input signal up to 65536 times (for 16-bit rescalers). Based on its samples, the rescaler determines the period of the input signal. To accurately sample your input signal, the sample duration should be as close as possible to the input period while still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

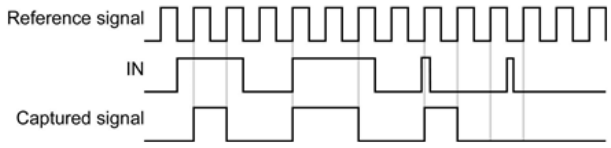
Delayer

You can delay an input trigger signal using the delayer.

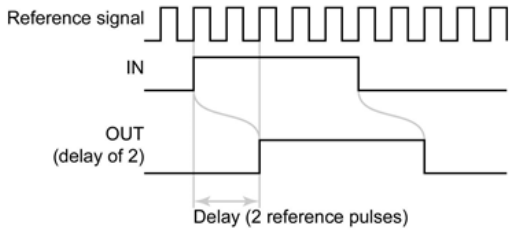


The delayer uses the rising edges of a periodic reference signal for timing. On each rising edge of the reference signal, the delayer samples the input signal. After n pulses (up to 65535) of the reference signal, the delay outputs the sampled value. The delayer can store up to 128 transitions (64 pulses).

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.



PLC Signal Descriptions

The following table shows the primary input and output signals you use in the Boolean equations when configuring the advanced features.

Table 9: PLC Signal Descriptions

Primary Signal Name	PLC Equation Usage	Description
Pb0Fval	Input	Channel #1 Pixel Bus Frame Valid Input signals.
Pb0Lval	Input	Channel #1 Pixel Bus Line Valid Input signals.
Pb0Dval	Input	Channel #1 Pixel Bus Data Valid Input signals.
Pb0Spare	Input	Channel #1 Pixel Bus Spare Input signals.
Pb1Fval	Input	Channel #2 Pixel Bus Frame Valid Input signals. For one-channel products, this input is tied to zero.
Pb1Lval	Input	Channel #2 Pixel Bus Frame Valid Input signals. For one-channel products, this input is tied to zero.
Pb1Dval	Input	Channel #2 Pixel Bus Data Valid Input signals. For one channel products, this input is tied to zero.

Table 9: PLC Signal Descriptions

Primary Signal Name	PLC Equation Usage	Description
Pb1Spare	Input	Channel #2 Pixel Bus Spare Input signals. For one-channel products, this input is tied to zero.
GpioIn[7:0]	Input	GPIO Input lines. Unused most significant bits are tied to zero.
BufferWM0	Input	Channel #1 Grabber Water Mark Almost Full Status
BufferWM1	Input	Channel #2 Grabber Water Mark Almost Full Status For one-channel products, this input is tied to zero.
Grb0AcqActive	Input	Channel #1 Grabber Acquisition Active Status
Grb1AcqActive	Input	Channel #2 Grabber Acquisition Active Status For one-channel products, this input is tied to zero.
PlcCascIn[1:0]	Input	PLC Cascade Input signals. Allow to connect up to two outputs of another instance of a PLC in the IP Engine. When there is no other PLC, these inputs are tied to zero.
PlcCtrl[3:0]	Input	PLC Control signals. These signals are coming from registers, directly controllable through software (host).
Pb0CC[3:0]	Input, Output	Channel #1 Pixel Bus Camera Control Output signals CC[4:1] for a Camera Link-type bus.
Pb1CC[3:0]	Input, Output	Channel #2 Pixel Bus Camera Control Output signals. TCC for a Camera Link-type bus For one channel products, these outputs are left unconnected.
GpioOut[7:0]	Input, Output	GPIO Output lines. Unused most significant bits are left unconnected.
PlcFval[1:0]	Input, Output	PLC Frame Valid signal. Signal to be routed to grabber FVAL external input. PLC_FVAL0 is for channel #1. PLC_FVAL1 is for channel #2 if two channels product, otherwise left unconnected.
PlcLval[1:0]	Input, Output	PLC Line Valid signal. Signal to be routed to grabber LVAL external input. PLC_LVAL0 is for channel #1. PLC_LVAL1 is for channel #2 if two channels product, otherwise left unconnected.
PlcMval[1:0]	Input, Output	PLC Metadata Valid signal. Signal to be routed to grabber MVAL external input. PLC_MVAL0 is for channel #1. PLC_MVAL1 is for channel #2 if two channels product, otherwise left unconnected.
PlcTrig[1:0]	Input, Output	PLC Trigger signal. Signal to be routed to grabber TRIG external input. PLC_TRIG0 is for channel #1. PLC_TRIG1 is for channel #2 if two channels product, otherwise left unconnected.
PlcTimestampCtrl	Input, Output	PLC Timestamp Control signal. Signal to be routed to PT_TIMESTAMP module. It can be used to reset the timestamp or to set it to a user-configured value, if supported.

Table 9: PLC Signal Descriptions

Primary Signal Name	PLC Equation Usage	Description
PlcCascOut[1:0]	Input, Output	PLC Cascade Output signals. Allow to connect up to two outputs to another instance of a PLC in the video interface. When there is no other PLC, then these outputs are left unconnected.
Timer0Trig	Input, Output	Timer #0 Trigger Input signal.
Timer0Out	Input	Timer #0 Output signal.
Timer1Trig	Input, Output	Timer #0 Trigger Input signal.
Timer1Out	Input	Timer #0 Output signal.
Timer2Trig	Input, Output	Timer #0 Trigger Input signal.
Timer2Out	Input	Timer #0 Output signal.
Timer3Trig	Input, Output	Timer #0 Trigger Input signal.
Timer3Out	Input	Timer #0 Output signal.
Counter0Reset	Input, Output	Counter #0 Reset signal.
Counter0Inc	Input, Output	Counter #0 Increment signal.
Counter0Dec	Input, Output	Counter #0 Decrement signal.
Counter0Eq	Input	Counter #0 Equal signal.
Counter0Gt	Input	Counter #0 Greater signal.
Counter1Reset	Input, Output	Counter #1 Reset signal.
Counter1Inc	Input, Output	Counter #1 Increment signal.
Counter1Dec	Input, Output	Counter #1 Decrement signal.
Counter1Eq	Input	Counter #1 Equal signal.
Counter1Gt	Input	Counter #1 Greater signal.
Rescaler0In	Input, Output	Rescaler #0 Input signal.
Rescaler0Out	Input	Rescaler #0 Output signal.
Delayer0In	Input, Output	Delayer #0 Input signal.
Delayer0Out	Input	Delayer #0 Output signal.
Event0	Input, Output	Event Queue Input 0 signal.
Event1	Input, Output	Event Queue Input 1 signal.
Event2	Input, Output	Event Queue Input 2 signal.
Event3	Input, Output	Event Queue Input 3 signal.
ActionTrig0	Input	Action Trigger Generator Output 0 signal.
ActionTrig1	Input	Action Trigger Generator Output 1 signal.

Chapter 4

Using the Event Queue

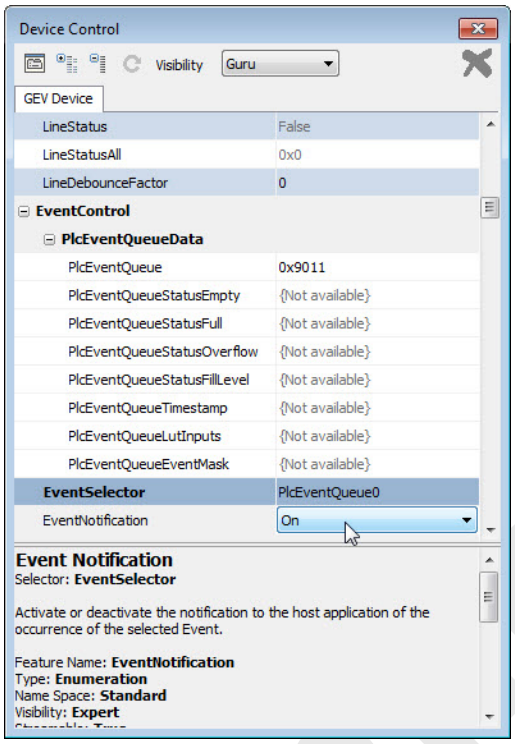
You can generate an interrupt on the host computer using the event queue feature. When the rising edge of the event signal occurs, the video interface sends a GenICam event packet containing the following:

- PlcEventQueueEventMask: Bitfield indicating the source(s) of the PlcEventQueue event.
- PlcEventQueueTimestamp: Timestamp value associated to the occurrence of the PlcEventQueue event.
- PlcEventQueueLutInputs: State of the PLC LUT inputs causing the PlcEventQueue event.

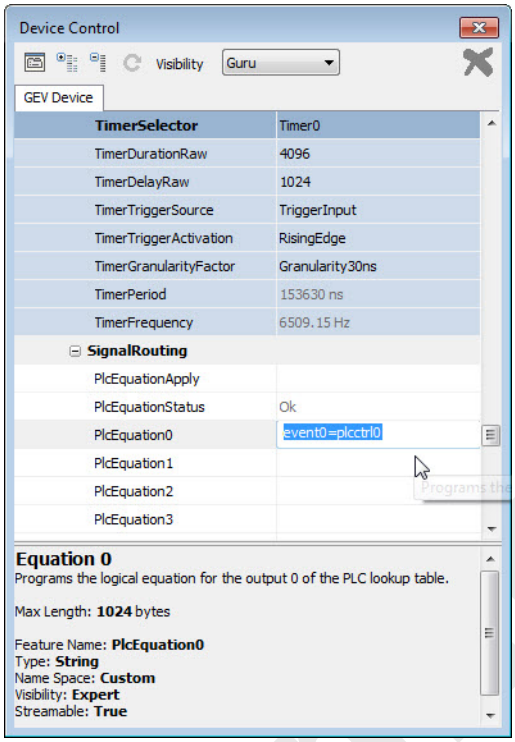
To configure the video interface to generate an event

1. Start eBUS Player and connect to the video interface.
   For more information, see the *eBUS Player User Guide*.
2. Under **Parameters and Controls,** click **Device control**.
1. Click **Guru** in the **Visibility** list.

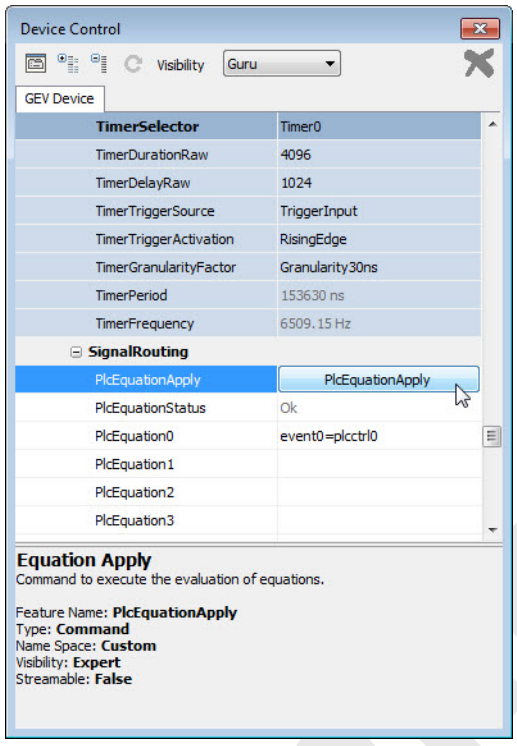
2. In the **EventSelector** section of the **EventControl** category, select **On** in the **EventNotification** list.



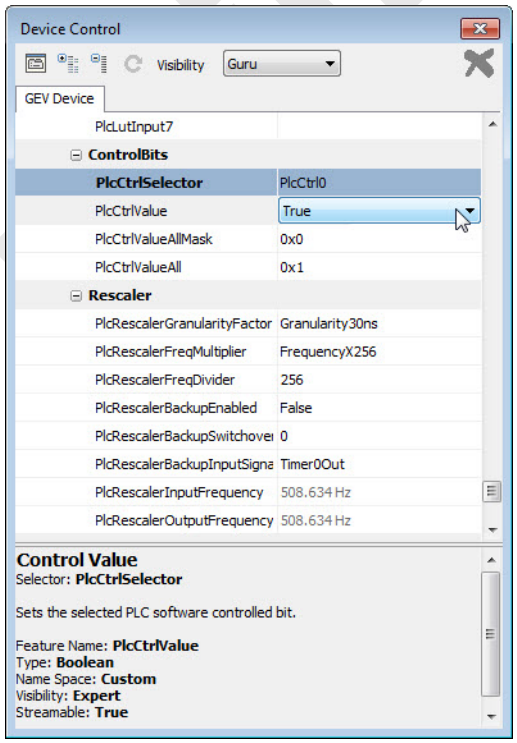
3. In the **SignalRouting** section of the **Plc** category, select an available equation, and enter an equation, such **event0 = plcctrl0**, as the one shown in the image below.



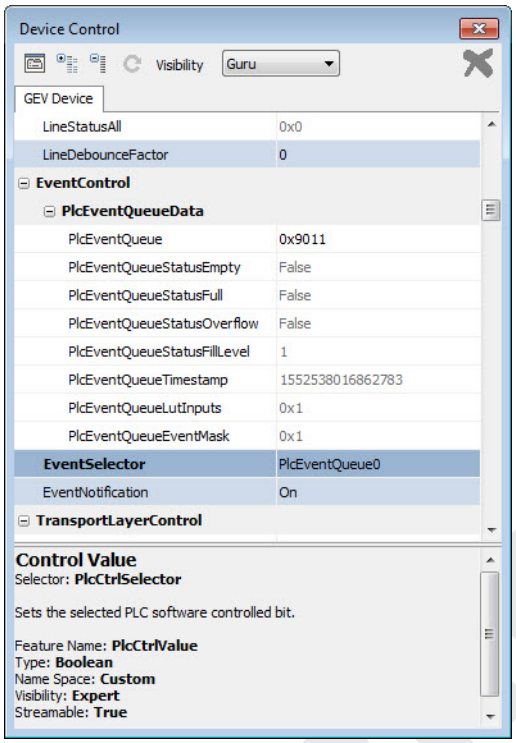
4. Click the **PlcEquationApply** button.



5. In the **ControlBits** section, select **True** in the **PlcCtrlValue** list.



You can view the event status in the **PlcEventQueueData** section of the **EventControl** section.



Chapter 5

Configuring Action Commands Using IEEE 1588 Precision Time Protocol (PTP)

This chapter describes how to configure and send actions commands using the IEEE 1588 precision time protocol (PTP).

The following topics are covered in this chapter:

Using Action Commands

Action commands allow you to trigger an action on one or several devices at the same time using the value of the video interface timestamp counter. When the 64-bit timestamp is equal or greater than the configured time value, the action trigger generator emits a signal on up to two outputs. These signals are routed into the signal routing block and can be used to trigger other events or actions. For example, you can use an action command to trigger multiple cameras to capture images at the same time, trigger a GPIO output (such as a strobe), and generate an interrupt to tell the computer to do something when the action command is executed. Action commands are available in eBUS SDK version 3.1 (and later).

Action commands can be immediate or scheduled. Immediate action commands are broadcast to the network and executed immediately on devices that are configured to receive the commands. The moment at which the command is received and executed by the devices varies slightly, depending on your network configuration. Scheduled action commands are also broadcast to the network and provide a way for you to trigger an action at a specific moment in the future.

Actions are queued and triggered on a First In, First Out (FIFO) basis. Once an action is triggered, it is automatically removed from the FIFO PlcActionTrigQueue.

Actions registered to be triggered at a time that has already passed are immediately triggered (as specified in the GigE Vision specification), bypassing any pending registered action in the queue. If the scheduled time for an action is set to zero, the action is immediately triggered.

The action trigger generator does not support actions that are scheduled out of order; if a second registered action is scheduled to be triggered earlier than a first action (that is already registered), the second action will be executed immediately after the first action. Immediately following the action, the GEV_STATUS_ACTION_LATE status is sent to the internal CPU (when the action trigger generator is used with a GigE Vision Action Command). The action trigger generator can also send a GEV_EVENT_ACTION_LATE event if the message channel is opened and this event generation has been enabled. The timestamp value comparison is completed and the status is sent at the registration time, not at the execution time.

The action trigger generator also provides a status to the internal CPU indicating whether or not the FIFO PlcActionTrigQueue is empty or full, and always provides a status on the FIFO PlcActionTrigQueue before any new action is queued. The action trigger generator can clear the content of the FIFO PlcActionTrigQueue. If the FIFO PlcActionTrigQueue is full, the action trigger generator returns the GEV_STATUS_OVERFLOW status and the next action is not queued.

Using Action Commands with IEEE 1588 PTP

To use scheduled action commands, your GigE Vision device must be configured to use a Precision Time Protocol, such as the IEEE 1588 Precision Time Protocol (PTP) to synchronize the clocks between the devices on the network.

You can implement the IEEE 1588 PTP on Pleora's video interfaces using any of the following methods:

- PTP Daemons running on the Windows or Linux operating system
- Network switch that supports IEEE 1588 PTP
- All video interfaces connected to one NIC (with multiple ports) that supports IEEE 1588 PTP
- Multiple video interfaces on a network, with one resolved as the master clock to which the other devices are synchronized

About IEEE Precision Time Protocol (PTP)

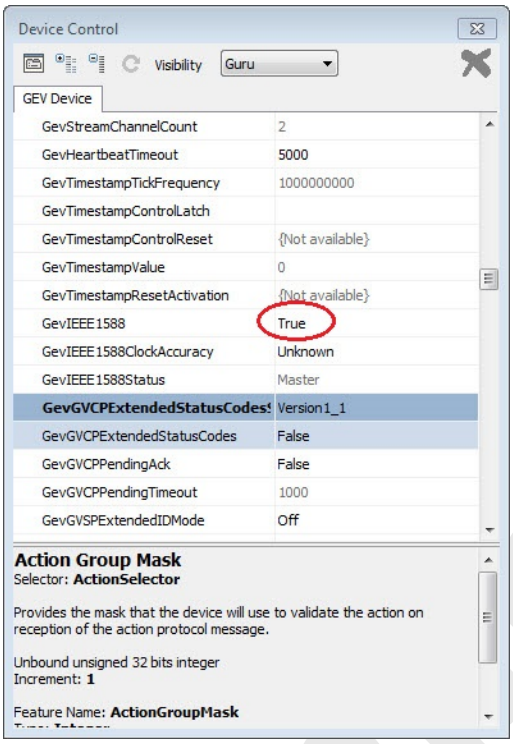
The IEEE Precision Time Protocol (PTP), introduced in the GigE Vision 2.0 standard as an optional feature, allows you to precisely synchronize distributed clocks with an accuracy of less than one microsecond on an Ethernet network. In most network configurations, one master clock is used to synchronize slave clocks.

Using scheduled action commands with the IEEE 1588 PTP ensures that a single GigE Vision 2.0 compliant device (or multiple devices) is configured to trigger an action or actions with sub-millisecond precision.

To configure the video interface to use IEEE 1588

1. Start eBUS Player and connect to the video interface.
   For more information, see the *eBUS Player User Guide*.
2. Under **Parameters and Controls**, click **Device control**.
3. Click **Expert** in the **Visibility** list.

4. In the **TransportLayerControl** category, select **True**.



Configuring Action Commands

You can configure a video interface to send action commands to one or several devices using the following procedure.

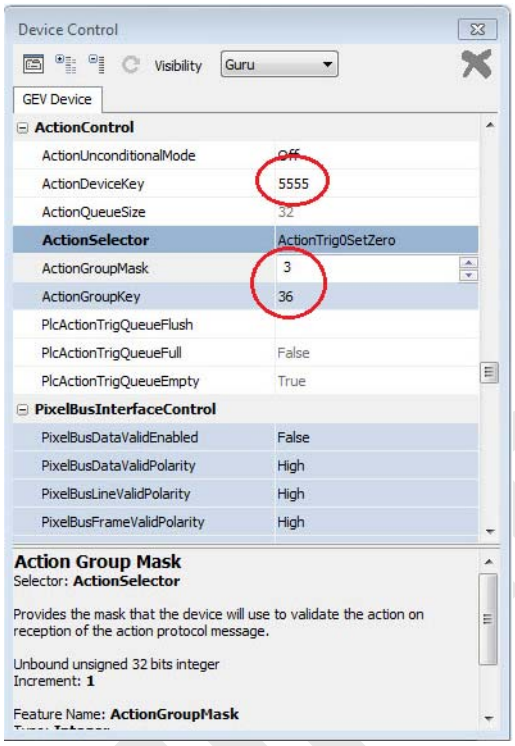
To configure the video interface to perform an action command

1. Start eBUS Player and connect to the video interface.
2. Under **Parameters and Controls**, click **Device control**.
3. Click **Guru** in the **Visibility** list.
4. Under **Action Control**, select the action that you want the device to perform when the command is executed in the **ActionSelector** list.
5. Under **Action Control**, specify the following information.

   Device Key. Authorizes the action on the device. Devices will take action only if their configured device key and group key matches the value entered into the eBUS Player **ActionDeviceKey** box.

   Group Key. Defines the group of devices on which the action command is executed. Each device will take action only if their configured group key matches the value entered into the eBUS Player **ActionGoupKey** box.

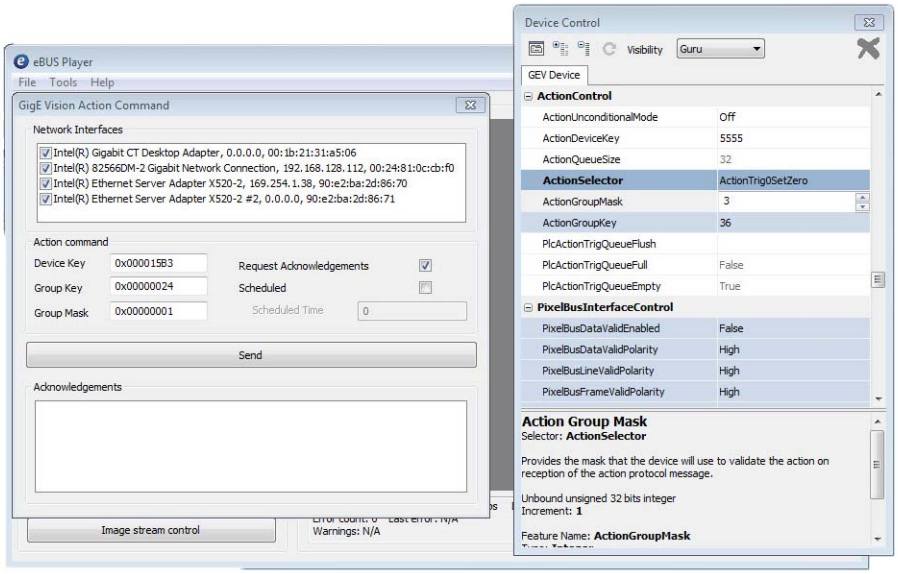
Group Mask. Defines a subset of the group of devices on which the action command is executed. The masks for the action command and the device settings cannot have the AND operation evaluate to zero.If the AND operation evaluates to zero, the action will not be performed.



To send an action command

1. Start eBUS Player and connect to the video interface.
1. Select **Tools** > **GigE Vision Action Command**.
2. Under **Network Interfaces**, select the NIC or NICs from which you want to send the action command.
3. Under **Action command**, specify the **Device Key**, **Group Key**, and **Group Mask**.

The **Device Key** and **Group Mask** must match the value you specified in step 5 of "To configure the video interface to perform an action command" on page 38.



4. Optionally, you can select the **Scheduled** check box and specify the time at which you want the action command to be executed in the **Scheduled Time** box.

GevIEEE1588 must be **True** to send a scheduled action commend, or you will receive a **No Ref Time** acknowledgement.

5. Click **Send** to broadcast the action command to all devices that meet the criteria specified in step 3 and are connected to the NIC selected in step 2.

   Acknowledgements appear in the Acknowledges section.

Action Command Acknowledgements

As outlined in version 2.0 of the GigE Vision standard, devices are not required to send an acknowledgment in response to an action command. For devices that require an acknowledgement, eBUS Player includes the option to force the device to send an acknowledgement.

The following table provides descriptions of the possible action command acknowledgements.

Table 10: Action Command Acknowledgement Descriptions

Acknowledgement	Description
No Ref Time	IEEE 1588 is not enabled; you must enable IEEE 1588 to send a schedule action
Late	The action command was executed immediately because it was sent when the scheduled time value was configured for a time earlier than the current GevTimestampValue

Table 10: Action Command Acknowledgement Descriptions

Acknowledgement	Description
OK	The action command was sent and received and will be executed at the specified scheduled time.
Overflow	The PlcActionTrigQueue buffer is full (32 commands are stored in the queue)

To configure a video interface to send an action command acknowledgement

1. Start eBUS Player and connect to the video interface.
2. Select **Tools** > **GigE Vision Action Command**.
3. In the **Action Command** panel, select the **Request Acknowledgements** check box.

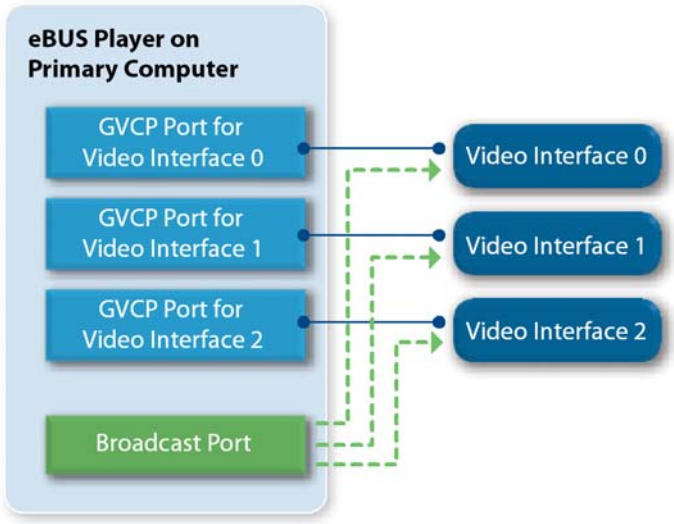
Broadcasting Action Commands

You can broadcast action commands using eBUS Player running on either a primary or secondary computer.

Broadcasting Action Commands from a Primary Computer

Using eBUS Player running on a primary computer, you can broadcast action commands to all devices on the subnet from a port, as shown in the following diagram.

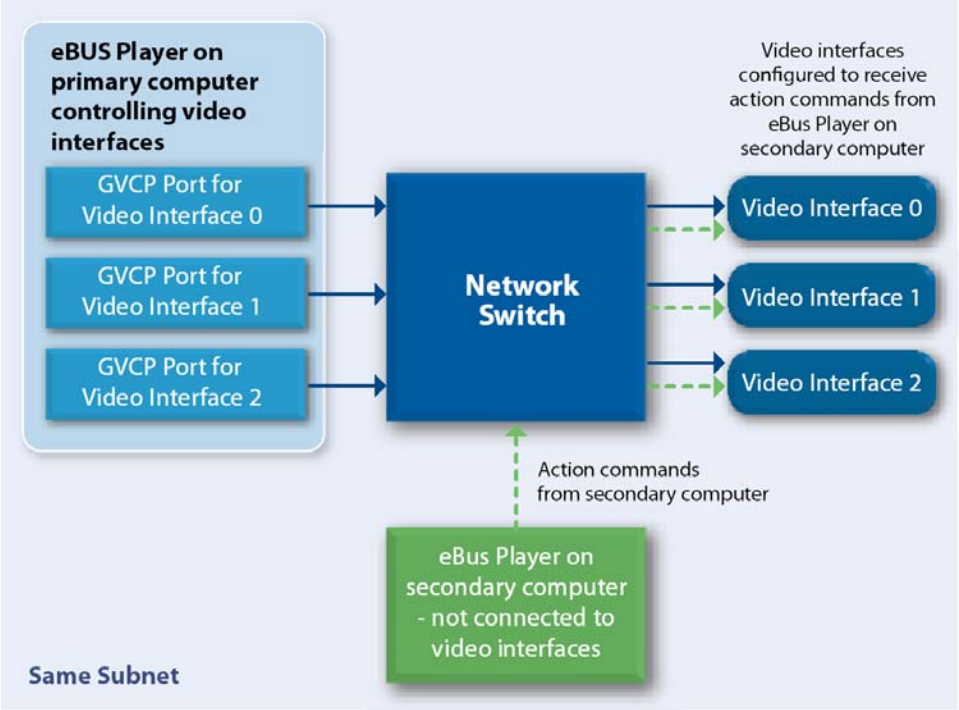
Figure 1: Broadcasting Actions from a Primary Computer



Broadcasting from a Secondary Computer

Action commands can also be sent by a second eBUS Player application running on a secondary computer or another device on the same subnet, as shown in the following diagram.

Figure 2: Broadcasting Action Commands from a Secondary Computer



ActionUnconditionalMode

You can use eBUS Player, connected on a network, to send action commands to devices when eBUS Player is not directly connected to the devices if you first take the following steps:

- Ensure all NICs and devices are on the same subnet (either configured as LLA or with persistent/ static IPs)
- Configure the **Device Key**, **Group Key** and **Group Mask** of each device
- Set **ActionUnconditionalMode** to **True**
- Disconnect each device from eBUS Player.

  You can now use a secondary application to send action commands to these devices.

Chapter 6

Extended Chunk Mode Support

This chapter describes the extended chunk mode feature of your video interface.

The following topics are covered in this chapter:

About the Extended Chunk Mode Feature

The extended chunk mode feature allows you to append metadata to some of the payload types, such as an image payload, as chunk data. For example, when the extended chunk feature is enabled, you can attach the position of the conveyor belt (metadata) to the image of the product on the conveyor belt. Metadata is attached to the data block as “chunk data”, which is also referred to simply as “chunks”.

A chunk includes the chunk data and chunk tags (CID and length fields).

The table below describes the general structure of any chunk.

Table 11: Chunk Structure Descriptions

Position	Format	Size	Description
0	data	K Bytes	The data transported by the chunk. This section must be a multiple of 4 bytes. If it is not, the data must be padded with zeros to a multiple of four bytes such that K is a multiple of 4 bytes. This ensures *CID* and *length* fields are 32-bit aligned within the chunk. Endianness of the chunk data itself is defined in the XML device description file.
K	CID	4 bytes	The chunk identifier. The CID tag MUST use network byte order (big-endian).
K+4	length	4 bytes	The length of the data (in bytes, must be a multiple of 4). The length tag MUST use network byte order (big-endian).

There are two sources of metadata supported by Pleora’s video interfaces: metadata generated by the camera and metadata generated by the video interface.

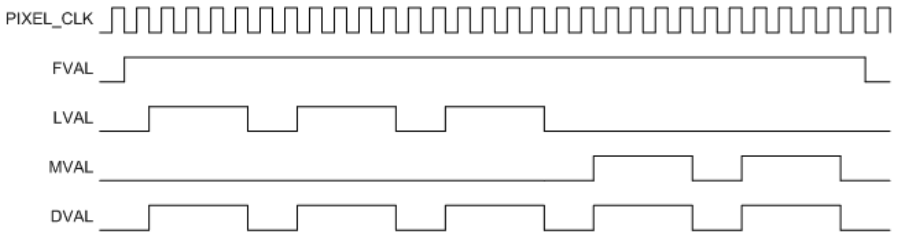
MetaData Generated by the Camera

Cameras can send metadata containing camera-specific information to the video interface using the MVAL (Metadata Valid) signal, which is supplied by the camera. The MVAL signal should be connected to the SPARE signal pin on the pixel bus of the video interface.

The MVAL signal functions in a similar manner as the LVAL signal and is only valid when the FVAL signal is valid.

Metadata must be transmitted on the pixel bus at the end of the image data. All image lines (LVAL) will be dropped after a first MVAL assertion.

The following timing diagram provides an example of image data combined with metadata in a single frame. One image payload type block contains three chunks: image chunk data and two metadata chunks.



The DVAL signal can be used to throttle metadata, if required.

Chunk data on pixel bus can only be 8 bits wide using Data [7:0], the lowest 8 bits of the pixel data bit.

The following diagram shows an example of a chunk and its content transmitted using 8-bits on the pixel bus. FVAL, LVAL and DVAL are not illustrated to simplify the diagram. Below the diagram, the table provides descriptions of the data transmitted in the GVSP block.

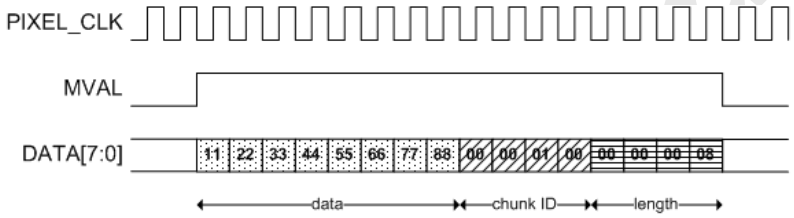


Table 12: GVSP Block Data Descriptions

Byte 0	Byte 1	Byte 2	Byte 3	Description
0x11	0x22	0x33	0x44	Chunk data = 0x8877665544332211 (if chunk data is defined in the XML as little-endian)
0x55	0x66	0x77	0x88	
0x00	0x00	0x01	0x00	Chunk ID = 0x00000100 = 256
0x00	0x00	0x00	0x08	Chunk length = 0x00000008 = 8

The chunk (metadata) data provided on the pixel bus by the video source must be a multiple of 4 bytes. No padding is done on the chunk data by the video interface.

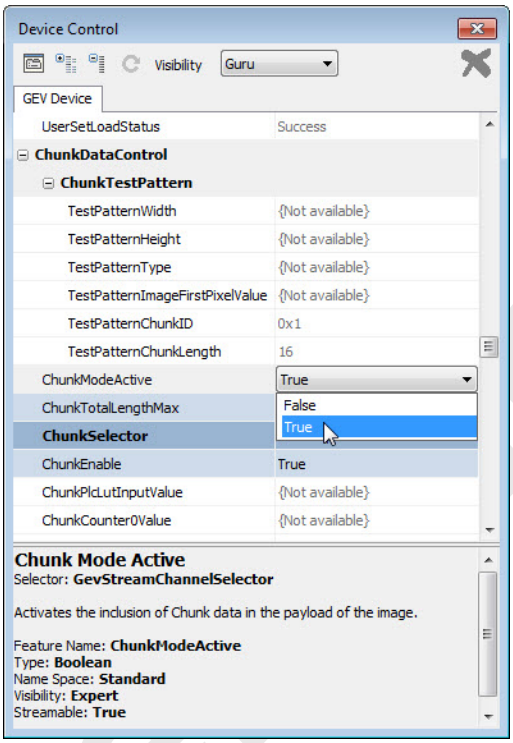
The chunk ID for metadata must be provided on the pixel bus by the video source right after the chunk data. The values of chunk ID provided by the video source must be 0x80000000 or above. Values below 0x80000000 are reserved by Pleora for its own purposes. The chunk ID for the image inserted by the video interface is fixed to 0. The chunk layout ID in the data trailer is fixed to 0.

The number of clock cycles on the pixel bus between the last LVAL signal and the first MVAL signal must be at least one. The number of clock cycles on the pixel bus between two MVAL signals must be at least one. The number of clock cycles on the pixel bus between the assertion of FVAL and the first metadata valid under MVAL must be at least 8.

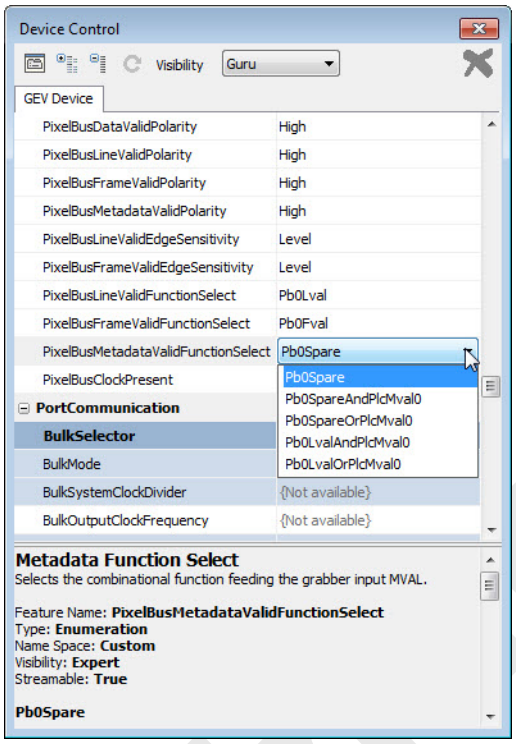
To configure metadata generated by the camera

1. Start eBUS Player and connect to the video interface.

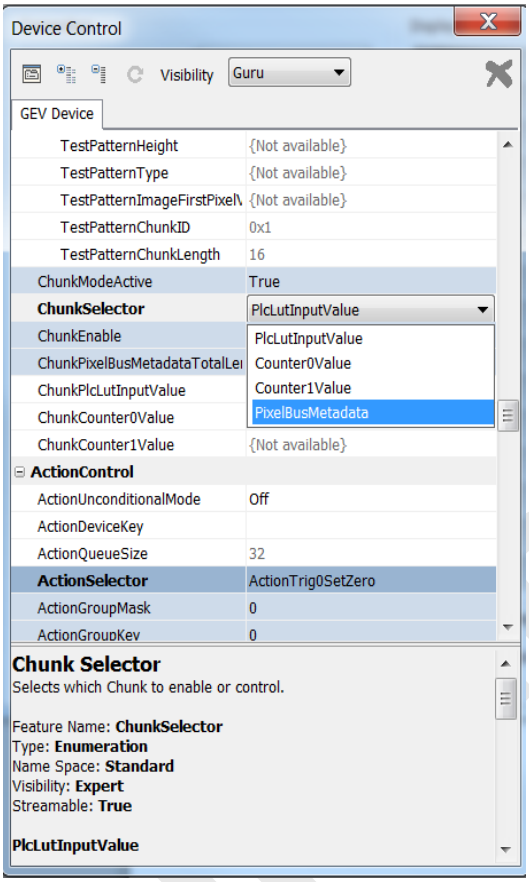
   For more information, see the *eBUS Player User Guide*.
2. Under **Parameters and Controls,** click **Device control**.
3. Click **Expert** in the **Visibility** list.
4. In the In the **ChunkDataControl** section, click **True** in the **ChunkModeActive** list.



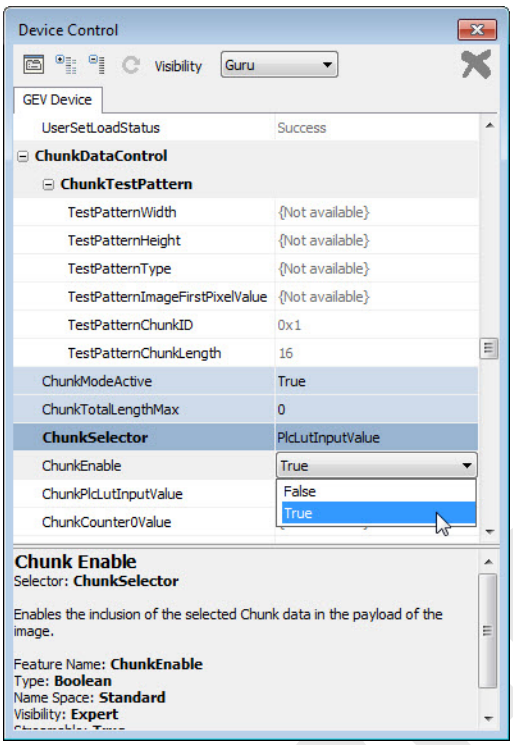
5. In the **PixelBusInterfaceControl** section, select an option from the **PixelBusMetadaDataValidFunctionSelect** list.



6. In the **ChunkSelector** list, select **PixelBusMetadata**.



7. In the **ChunkEnable** list, select **True**.



PLC Metadata Generated by the Video Interface

When the extended chunk feature is enabled, you can append the following PLC metadata, generated by the video interface, to the payload (image payload):

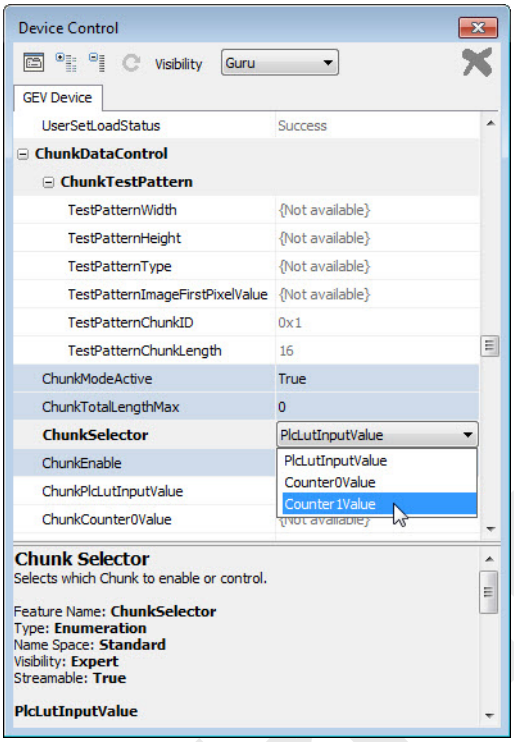
- PLCLutInputValue (returns the value of the 8-bit LUT inputs)
- ChunkCounter0Value (returns the value of PLC counter 0)
- Counter1Value (returns the value of PLC counter 1)

The extended chunk feature is only available for areascan cameras.

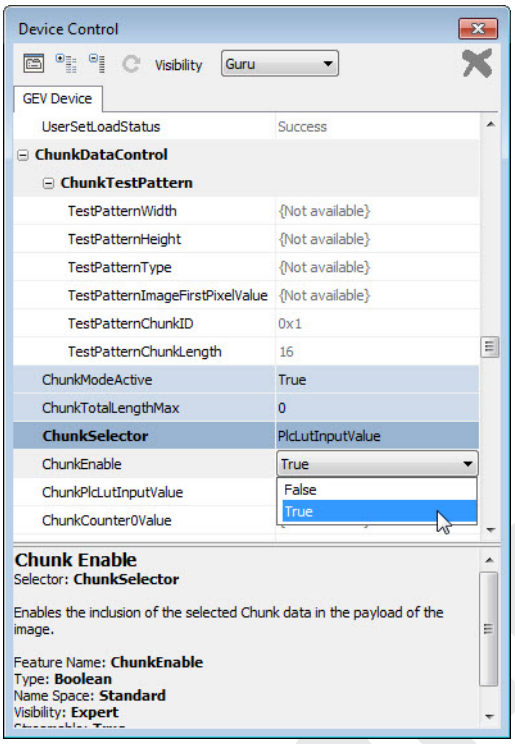
To configure metadata generated by the video interface

1. Complete steps 1-4 in the procedure, "To configure metadata generated by the camera" on page 46.

2. In the **ChunkSelector** list, click the chunk data that you want to include and enable each of the types.



3. In the **ChunkEnable** list, click **True**.



Chapter 7

Advanced Features Usage Examples

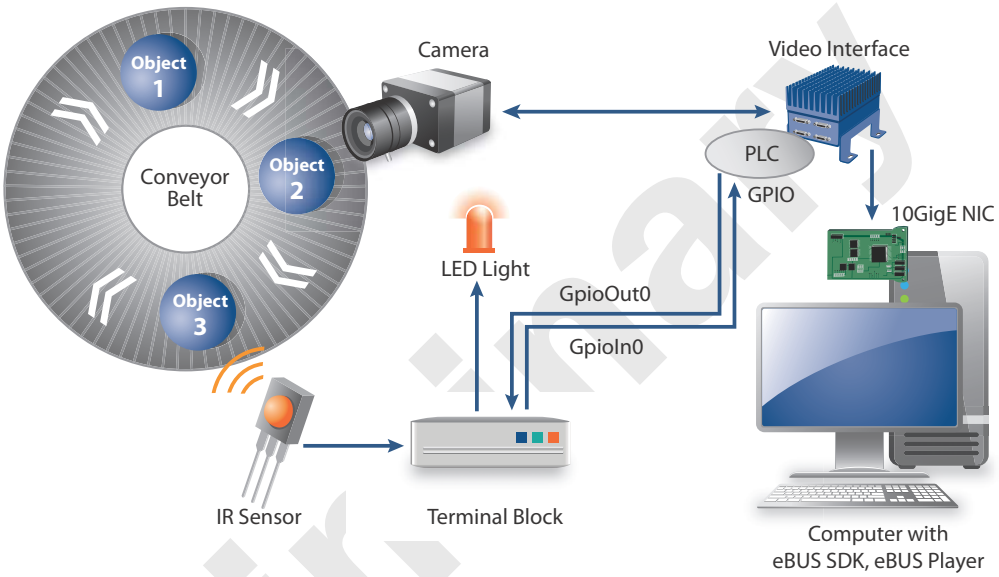
This chapter provides examples of how to configure the advanced features described in the previous chapters.

The following topics are covered in this chapter:

Creating a Demonstration Environment

You can use the example provided in the following diagram to create a demonstration environment that allows you to follow and test the procedures provided in this chapter.

Figure 3: PLC Demonstration Example



The usage examples included in this section make extensive use of the GpioOut0 signal, which maps to a pin on the 12-pin circular connector for your video interface. Please refer to the documentation for your video interface to locate and attach this pin to the voltage indicator. The ground pin on the 12-pin circular connector must be tied to the ground probe on the voltmeter or oscilloscope.

Setting up the Hardware

The advanced feature usage examples provided in this guide are based on the use of the following hardware:

- Pleora Technologies' video interface with power supply
- Computer, with 10-GigE NIC installed, running the latest version of the eBUS SDK with eBUS Player
- 10-GigE Direct Attach copper SFP+ cable (10-GigE connection between video interface and the computer)
- One of the following voltage indicators: LED with series resistor connected to a terminal block; or LED indicator assembly connected to a terminal block; or voltmeter; or oscilloscope
- 12-pin circular connector plug with cable harness (connection between the video interface 12-pin circular connector and the voltage indicator)
- Infrared (IR) sensor
- Camera Link or GigE camera connected to the Pleora video interface

Setting up the Software

The advanced feature usage examples provided in this guide are based on the use of Pleora's eBUS Player, which is provided on the eBUS SDK installation.

To configure the advanced features using eBUS Player, you must start eBUS Player and connect to the video interface. For more information about how to connect to the video interface, see the *eBUS Player User Guide*,

Controlling the Advanced Features from the Computer

The following procedure demonstrates how you can control the PLC and other advanced features from a computer, connected to the video interface, using the control bits that make up the remote control block.

To control the PLC remotely

1. Connect the hardware as shown in the "PLC Demonstration Example" on page 54.
2. Start eBUS Player and connect to the video interface.
3. Connect the voltage indicator or LED assembly to the pin on the circular connector that corresponds to **GpioOut0** signal on your video interface.
4. Under **Parameters and Controls,** click **Device control**.
5. Click **Guru** in the **Visibility** list.
6. In the **SignalRouting** section, select one of the available equations, for example, **PlcEquation0**, and enter the following Boolean equation: **GpioOut0** = **PlcCtrl0**.
7. Click **PlcEquationApply** to apply the equation.
8. In the **ControlBits** section of the **Plc** category, select one of the inputs from the **PlcCtrlSelector** list, for example, **PlcCtrl0**.
9. In the **PlcCtrlValue** list, click **True** to set the signal state to high, or **False** to set the signal state to low.

   When **PlcCtrl0** is set to **True**, the output on **GpioOut0** shows a high value on the voltmeter or oscilloscope, and when set to **False**, it shows a low value.

Using the Timer Feature to add a Pulse to a Signal

The following procedure demonstrates how you can configure the timer to output a continuous 1 Hz pulse that is high for 0.25 s and low for 0.75 s.

To configure the timer feature to add a pulse to a signal

1. Follow steps 1-5 in the procedure, "To control the PLC remotely" on page 56.
2. In the **Timer** section, select **Timer0** from the **TimerSelector** list.
3. Select **TimerDurationRaw** and enter **8138** to set the length of time that the state of the pulse is high.

   This value represents the number of internal clock ticks.
4. Select **TimerDelayRaw** and enter **24414** to set the length of time that the state of the pulse is low (the length of time before the signal becomes high).

   This value represents the number internal clock ticks.
5. Select **TimerTriggerSource** and select **Continuous** to set an internal triggering source for the timer and automatically trigger again after emitting the pulse.
6. Select **TimerGranularityFactor** and select **Granularity30p72us** to set the interval of the timer's internal clock tick.

7. In the **SignalRouting** section, select one of the equations, for example, PlcEquation0, and enter the following Boolean expression: **GpioOut0 = Timer0Out**.
8. Click **PlcEquationApply** to apply the equation.
9. In the **ControlBits** section of the **Plc** category, select one of the inputs from the **PlcCtrlSelector** list, for example, **PlcCtrl0**.
10. In the **PlcCtrlValue** list, click **True** to set the signal state to high, or **False** to set the signal state to low.

    The voltmeter indicates a steady 1 Hz signal with a 25% duty cycle on the signal, **GpioOut0**.

Using the Timer Feature to Trigger a Single Pulse

This following procedure demonstrates how you can use the timer feature to create a single pulse on a signal when triggered. The pulse is low for 1 s and high for 1 s.

To configure the timer to trigger a single pulse

1. Follow steps 1-5 in the procedure, "To control the PLC remotely" on page 56.
2. In the **Timer** section, select **Timer0** from the **TimerSelector** list.
3. Select **TimerDurationRaw** and enter **32552** to set the length of time that the state of the pulse is high.

    This value represents the number internal clock ticks.
4. Select **TimerDelayRaw** and enter **32552** to set the length of time that the state of the pulse is low (the length of time before the signal becomes high).

    This value represents the number internal clock ticks.
5. Select **TimerTriggerSource** and select **TriggerInput** to configure an external trigger input.
6. Select **TimerTriggerActivation** and select **RisingEdge** to configure the timer to trigger on the external input's rising edge.
7. Select **TimerGranularityFactor** and select **Granularity30p72us** to configure the interval of the timer's internal clock tick.
8. In the **SignalRouting** section, select one of the available equations, for example, **PlcEquation0**, and enter the following Boolean equation: **GpioOut0 = PlcCtrl0**.
9. In the **SignalRouting** section, select another of the available equations, for example, **PlcEquation1**, and enter the following Boolean equation: **Timer0Trig = PlcCtrl0**.
10. Click **PlcEquationApply** to apply the equations.
11. In the **ControlBits** section of the **Plc** category, select one of the inputs from the **PlcCtrlSelector** list, for example, **PlcCtrl0.**
12. In the **PlcCtrlValue** list, click **True** to set the signal state to high, or **False** to set the signal state to low.

    When **PlcCtrl0** is set to **True** from a **False** setting, **Timer0** emits a single pulse. The low signal state is emitted first. Subsequent changes to **PlcCtrl0** are ignored until the single pulse is complete.

Using the Delayer Feature to Delay a Signal

The following procedure demonstrates how you can configure the delayer feature to delay a signal for three seconds. The delayer feature operates by sampling the signal and then delaying it by a number of configured sampling intervals. In the following example, Timer0 is used to set the sampling frequency of approximately 16Hz, or 16 sampling intervals per second. To delay the signal by 3 seconds, 48 sampling intervals are required (3 x 16).

To configure the delayer to delay a signal

1. Follow steps 1-5 in the procedure, "To control the PLC remotely" on page 56.
2. In the **Timer** section, select **Timer0** from the **TimerSelector** list.
3. Select **TimerDurationRaw** and enter **65535** to configure the duration of the sampling interval (the length of time that the signal state is high).
4. Select **TimerDelayRaw** and enter **65535** to configure the length of time that the signal is delayed before the pulse can start (the length of time that the signal stat is low).
5. Select **TimerTriggerSource** and select **Continuous** to allow the timer to automatically trigger (no source required).
6. In the **TimerGranularityFactor** list, select **Granularity480ns** to configure the interval of the timer's internal clock tick.
7. In the **SignalRouting** section, select one of the available equations, for example, **PlcEquation0**, and enter the following Boolean equation: **GpioOut0 = Delayer0Out.**
8. In the **SignalRouting** section, select another of the available equations, for example, **PlcEquation1**, and enter the following Boolean equation: **Delayer0In = PlcCtrl0**.
9. Click **PlcEquationApply** to apply the equations.
10. In the **Delayer** section, select **PlcDelayerDelayCount** and enter **48**.
11. In the **PlcDelayerReferenceTimingSignal** list, select **Timer0Out**.
12. In the **ControlBits** section of the **Plc** category, select one of the inputs from the **PlcCtrlSelector** list, for example, **PlcCtrl0**.
13. In the **PlcCtrlValue** list, click **True** to set the signal state to high, or **False** to set the signal state to low.

    When **PlcCtrl0** is set to **True**, the output value becomes high after a three second delay. When it is set False, the output value becomes low after three seconds. The output will always track the inputs on **PlcCtrl0**.

Using the Rescaler Feature to Change a Signal Frequency

The following procedure demonstrates how you can configure the rescaler feature to change a 16 Hz pulse to a 1.6 Hz pulse. The **PlcCtrl0** control bit is used to stop and start the 16 Hz input to the rescaler. The rescaler requires a timing signal to perform the conversion. In this example, Timer0 is used to generate the 16 Hz pulse.

To configure the rescaler to change a signal frequency

1. Follow steps 1-5 in the procedure, "To control the PLC remotely" on page 56.
2. In the **Timer** section, select **Timer0** from the **TimerSelector** list.
3. Select **TimerDurationRaw** and enter **65535** to configure the duration of the 16 Hz pulse (the length of time that the 16 Hz pulse is high).
4. Select **TimerDelayRaw** and enter **65535** to set the duration of the delay before the pulse can start (the length of time that the 16 Hz pulse is low).
5. Select **TimerTriggerSource** and select **Continuous** to allow the timer to automatically trigger (no source required).
6. In the **TimerGranularityFactor** list, select **Granularity480ns** to configure the interval of the timer's internal clock tick.
7. In the **Rescaler** section, select **Granularity7p68us** from the **PlcRescalerGranularityFactor** list.
8. In the **PlcRescalerFreqMultiplierSelect** list, select **FrequencyX16**.
9. Select **PlcRescalerFreqDivider** list and enter **160**.
10. In the **PlcRescalerBackupEnabled** list, select **False**.
11. In the **SignalRouting** section, select one of the available equations, for example, **PlcEquation0**, and enter the following Boolean equation: **GpioOut0 = Rescaler0Out.**
12. In the **SignalRouting** section, select another of the available equations, for example, **PlcEquation1**, and enter the following Boolean equation: **Rescaler0In = Timer0Out & PlcCtrl0.**
13. Click **PlcEquationApply** to apply the equations.
14. In the **ControlBits** section of the **Plc** category, select one of the inputs from the **PlcCtrlSelector** list, for example, **PlcCtrl0**.
15. In the **PlcCtrlValue** list, click **True** to set the signal state to high, or **False** to set the signal state to low.

    When **PlcCtrl0** is **True**, the 16 Hz signal from **Timer0** passes into the rescaler and emerges as a 1.6 Hz signal. When **PlcCtrl0** is **False**, the 16 Hz signal is interrupted; the rescaler's output stops, and may be either a 'high' or 'low' value.

Using the Rescaler Feature with a Backup Signal

The following procedure demonstrates how you can configure the rescaler feature to change a 1017 Hz signal into an approximately 4 Hz signal and, if interrupted, to emit a backup signal.

In the following example, Timer0 generates the 1017 Hz signal that the rescaler uses as its primary input. The rescaler divides the signal frequency and outputs a rescaled 4 Hz pulse.

Timer1 generates a 1.1 Hz signal that the rescaler uses as a backup. If the 1017 Hz signal is interrupted, the rescaler outputs the 1.1 Hz backup signal. The 1017 Hz primary signal is manually interrupted by setting **PlcCtrl0** to **False**.

The rescaler does not rescale the backup signal.

To configure the rescaler to emit a backup signal

1. Follow steps 1-5 in the procedure, "To control the PLC remotely" on page 56.
2. In the **Timer** section, select **Timer0** from the **TimerSelector** list.
3. Select **TimerDurationRaw** and enter **20** to configure the duration of the 1017 Hz pulse (the length of time that the 1017 Hz pulse is high).
4. Select **TimerDelayRaw** and enter **11** to set the duration of the delay before the pulse can start (the length of time that the 1017 Hz pulse is low).
5. Select **TimerTriggerSource** and select **Continuous** to allow the timer to automatically trigger (no source required).
6. In the **TimerGranularityFactor** list, select **Granularity30p72us** to configure the interval of the timer's internal clock tick.
7. In the **Timer** section, select **Timer1** from the **TimerSelector** list
8. Select **TimerDurationRaw** and enter **78** to configure the duration of the 1017 Hz pulse (the length of time that the 1.1 Hz pulse is high).
9. Select **TimerDelayRaw** and enter **383** to set the duration of the delay before the pulse can start (the length of time that the 1.1 Hz pulse is low).
10. Select **TimerTriggerSource** and select **Continuous** to allow the timer to automatically trigger (no source required).
11. In the **TimerGranularityFactor** list, select **Granularity1p96608ms** to configure the interval of the timer's internal clock tick.
12. In the **SignalRouting** section, select one of the available equations, for example, **PlcEquation0**, and enter the following Boolean equation: **Rescaler0In = PlcCtrl0 & !Timer0Out.**
13. In the **SignalRouting** section, select another of the available equations, for example, **PlcEquation1**, and enter the following Boolean equation: **GpioOut0 = Rescaler0Out.**
14. Click **PlcEquationApply** to apply the equations.
15. In the **Rescaler** section, select **Granularity480ns** from the **PlcRescalerGranularityFactor** list.
16. In the **PlcRescalerFreqMultiplierSelect** list, select **FrequencyX16**.
17. Select **PlcRescalerFreqDivider** list and enter **4095**.
18. In the **PlcRescalerBackupEnabled** list, select **True**.

19. Select **PlcRescalerBackupSwitchoverDelay** and enter **4095.**
20. In the **PlcRescalerBackupInputSignal** list, select **Timer1Out**.
21. In the **ControlBits** section of the **Plc** category, select one of the inputs from the **PlcCtrlSelector** list, for example, **PlcCtrl0**.
22. In the **PlcCtrlValue** list, click **True** to set the signal state to high, or **False** to set the signal state to low.

    When **PlcCtrl0** is **True**, the timer outputs a 4 Hz signal. When **PlcCtrl0** is **False**, the timer outputs a 1.1 Hz signal.

    PlcRescalerBackupSwitchoverDelay configures how long the rescaler waits before switching to the backup signal. Configure this setting carefully. If this value is set too low (**a value of 10 or lower**), 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.

Using the Counter Feature

The following procedure demonstrates how you can configure the counter feature to keep counts of various PLC activities. In this example, the remote control bits are used to clear, decrement, or increment the count. When the count is greater than 4, the signal output is **True** or (high).

To configure the counter

1. Follow steps 1-5 in the procedure, "To control the PLC remotely" on page 56.
2. In the **Counter** section, select **Counter0** from the **CounterSelector** list.
3. In the **CounterResetActivation** list, select **RisingEdge**.
4. Select **CounterDuration** and enter 4.
5. In the **CounterIncrementEventActivation** list, select **RisingEdge**.
6. In the **CounterDecrementEventActivation** list, elect **RisingEdge**.
7. In the **SignalRouting** section, select one of the available equations, for example, **PlcEquation0**, and enter the following Boolean equation: **Counter0Reset = PlcCtrl0.**
8. Select another of the available equations, for example, **PlcEquation1**, and enter the following Boolean equation: **Counter0Dec = PlcCtrl1.**
9. Select another of the available equations, for example, **PlcEquation2**, and enter the following Boolean equation: **Counter0Inc = PlcCtrl2.**
10. Select another of the available equations, for example, **PlcEquation3**, and enter the following Boolean equation: **GpioOut0 = Counter0Gt.**
11. Click **PlcEquationApply** to apply the equations.
12. In the **ControlBits** section of the **Plc** category, select three of the inputs from the **PlcCtrlSelector** list, for example, **PlcCtrl0**, **PlcCtrl1**, and **PlcCtrl2**.
13. In the **PlcCtrlValue** list, click **True** to set the signal state to high, or **False** to set the signal state to low.

    By changing **PlcCtrlValue** from **False** to **True**, **PlcCtrl0** clears the count value (sets it to zero); **Plc_Ctrl1** decrements the count value; and **PlcCtrl2** increments the count value. If the count is decremented below **0**, the value wraps around to **4294967295**.

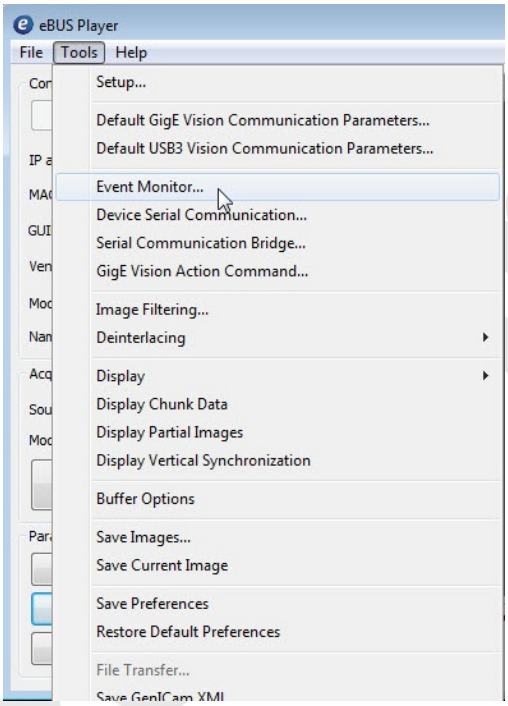
Using the Event Control Feature

The following procedure demonstrates how you can configure the event control feature to send interrupt requests to the host computer.

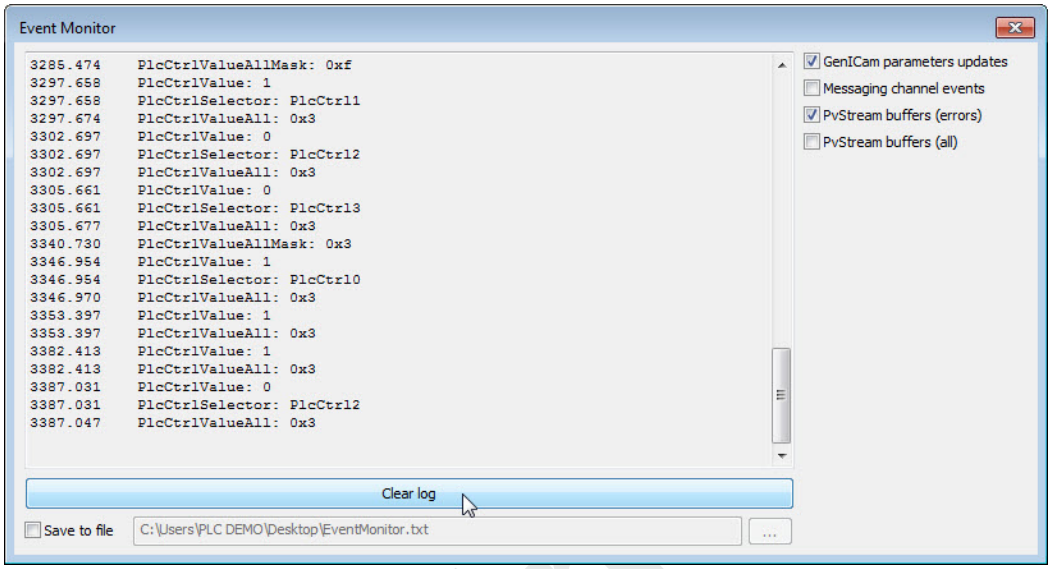
You can use the Event Monitor in eBUS Player to view the events. You should clear any existing events first.

To clear the event monitor

1. In eBUS Player, select **Event Monitor** from the **Tools** menu.



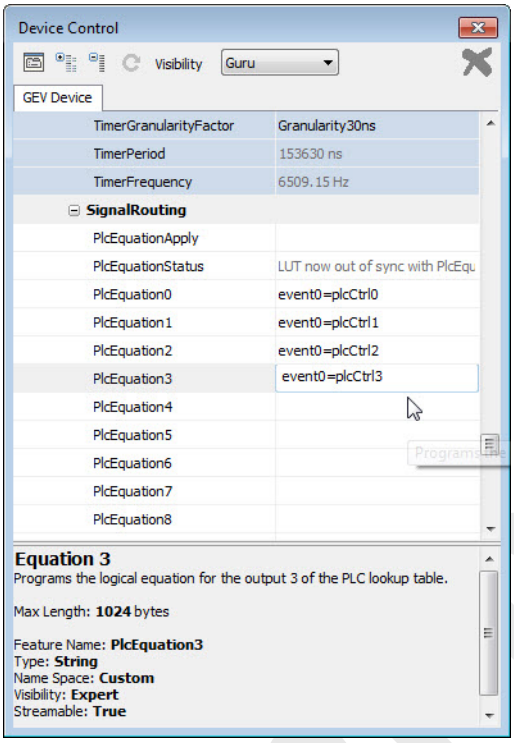
2. In the **Event Monitor** window, click **Clear log.**



To configure events

1. Follow steps 1-5 in the procedure, "To control the PLC remotely" on page 56.
2. In the **EventControl** section, select **On** from the **PlcEventQueue0** list.
3. Continue to select **On** from the **PlcEventQueue1**, **PlcEventQueue2**, and **PlcEventQueue3** lists.
4. In the **SignalRouting** section, select four of the available equations, for example, **PlcEquation0**, **PlcEquation1**, **PlcEquation2**, and **PlcEquation3 and enter one of the following Boolean equations in each equation field:**
   - Event0 = PlcCtrl0
   - Event1 = PlcCtrl1
   - Event2 = PlcCtrl2

- Event3 = PlcCtrl3



5. Click **PlcEquationApply** to apply the equations.
6. In the **ControlBits** section of the **Plc** category, select **PlcCtrl0** from the **PlcCtrlSelector** list.
7. In the **PlcCtrlValue** list, first select **True** and then **False**.
8. Repeat steps 6 and 7 for **PlcCtrl1**, **PlcCtrl2**, and **PlcCtrl3**.

   When **PlcCtrl0**, **PlcCtrl1**, **PlcCtrl2** and **PlcCtrl3**were set to **True**, the video interface sent interrupt requests to the host computer. Information about each interrupt appears in the **Event Monitor** dialog box.

Using the Scheduled Action Command Feature to Trigger a Pulse

The following procedure demonstrates how you can configure the action command feature to trigger a pulse on the **GpioOut0** signal at a scheduled time. You must enable the IEEE 1588 Precision Time Protocol (PTP) to synchronize the time between the devices on the network.

To configure a scheduled action command

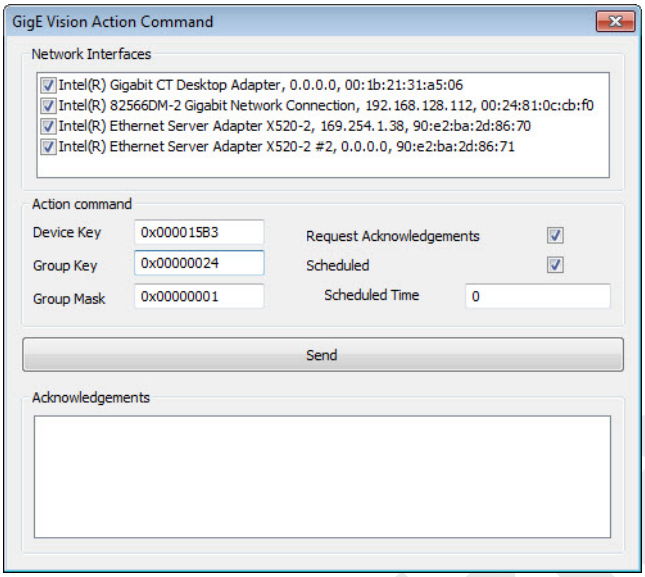
1. Follow steps 1-5 in the procedure, "To control the PLC remotely" on page 56.
2. In the **TransportLayerControl** section, select **True** in the **GevIEEE1588** list.

   The **GevIEEE1588Status** shows either **Master** or **Slave**.
3. In the **ActionControl** section, select **ActionTrig0Pulse** in the **ActionSelector** list.
4. Select **ActionDeviceKey** and enter **5555**.
5. Select **ActionGroupKey** and enter **36**.
6. Select **ActionGroupMask** and enter **3**.
7. In the **Timer** section, select **TriggerInput** from the **TimerTriggerSource** list.
8. In the **TimerGranularityFactor** list, select **Granularity1p96608ms**.

   The other settings in the **Timer** section can remain as the default settings.
9. In the **SignalRouting** section, select one of the equations, for example, **PlcEquation0**, and enter the following equation: **Timer0Trig = ActionTrig0**.
10. In the **SignalRouting** section, select another of the available equations, for example, **PlcEquation1**, and enter the following equation: **GpioOut0 = Timer0Out**.
11. Click **PlcEquationApply** to apply the equation.
12. In the **TransportLayerControl** section, click **GevTimestampControlLatch**.

    GevTimestampValue shows the current timestamp and is used as a reference in step 19 below.
13. In eBUS Player, select **GigE Vision Action Command** from the **Tools** menu.
14. In the **GigE Vision Action Command Action Command** window, enter **0x15B3** in the **Device Key** box.
15. Enter **0x24** in the **Group Key** box.
16. Enter **0x1** in the **Group Mask** box.
17. Select the **Scheduled** check box.
18. Select **Request Acknowledgements** if you want to view the action command status in the **Acknowledgments** panel.

The **GigE Vision Action Command** window is updated as shown in the following image.



19. Enter a time in the **Scheduled Time** box.

This value must be greater than the value shown in the **GevTimestampValue** box in the **Device Control** window, referenced in step 12 above.

20. Click **Send**.

21. You can view the status of the action command in the **Acknowledgements** panel if you selected **Request Acknowledgements**.

For information about the action command acknowledgements, see "Action Command Acknowledgements" on page 40

Chapter 8

Technical Support

On the Pleora Support Center, you can:

- Download the latest software.
- Log a support issue.
- View documentation for current and past releases.
- Browse for solutions to problems other customers have encountered.
- Get presentations and application notes.
- Get the latest news and information about our products.
- Decide which of Pleora's products work best for you.

To visit the Pleora Support Center

- Go to www.pleora.com and click **Support Center**.
  If you have not registered yet, you are prompted to register.
  Accounts are usually validated within one business day.