Gecho Viewer

User Guide

ZEBRA

Y11608-301-0101

Copyrights, Trademarks, Patents, Limitations of Liability, and Disclaimers

This section lists copyrights, acknowledgments, patent notices, limitations of liability, and disclaimers.

Copyright

3 Overlook Point, Lincolnshire, Illinois 60069, USA.

GechoViewer.exe has been incorporated and modified by Zebra Technologies Corporation. Zebra's modifications are licensed under Apache 2.0 License.

Proprietary Statement

Trademarks

Patents

This product may be protected by one or more patents.
Patents: http://www.zebra.com/patents.

Zebra Software

The object code or source code (collectively, the “Zebra Software”) is the exclusive property of Zebra or its licensors. Zebra Software is protected by copyright and other intellectual property rights. Zebra Software is licensed, not sold. Zebra Software should not be distributed except pursuant to a valid license agreement approved by Zebra containing appropriate restrictions. Use of Zebra Software is governed by the terms of the license agreement that accompanies or is included with the Zebra Software. UNAUTHORIZED COPYING, DISTRIBUTION, MODIFICATION, OR OTHER USE OF ZEBRA SOFTWARE IS STRICTLY PROHIBITED.

Limitations of Liability

In no event will Zebra or its suppliers be liable for any indirect, special, incidental, economic, cover or consequential damages arising out of the use of or inability to use the product, user documentation or related technical support, including without limitation, damages or costs relating to the loss of profits, business, goodwill, even if advised of the possibility of such damages. In no event will Zebra and its suppliers’ liability exceed the amount paid by you, for the product.

Because some jurisdictions do not allow the exclusion or limitation of liability for consequential or incidental damages, the above limitation may not apply to you.

Product Improvements

Continuous improvements of products is a policy of Zebra Technologies. All specifications and designs are subject to change without notice.

Disclaimer

Zebra reserves the right to make changes in specifications at any time and without notice. The information provided by this document is believed to be accurate and reliable. However, neither Zebra nor its suppliers assume any responsibility for its use; or for any infringements of patents or other rights of third parties resulting from its use. No license is granted under any patents or patent right of Zebra.

Publication Date

November 28, 2024

Contents

Chapter

1

Introduction

This chapter describes the basic features of the Gecho Viewer, and resources for more help.

Gecho Viewer overview

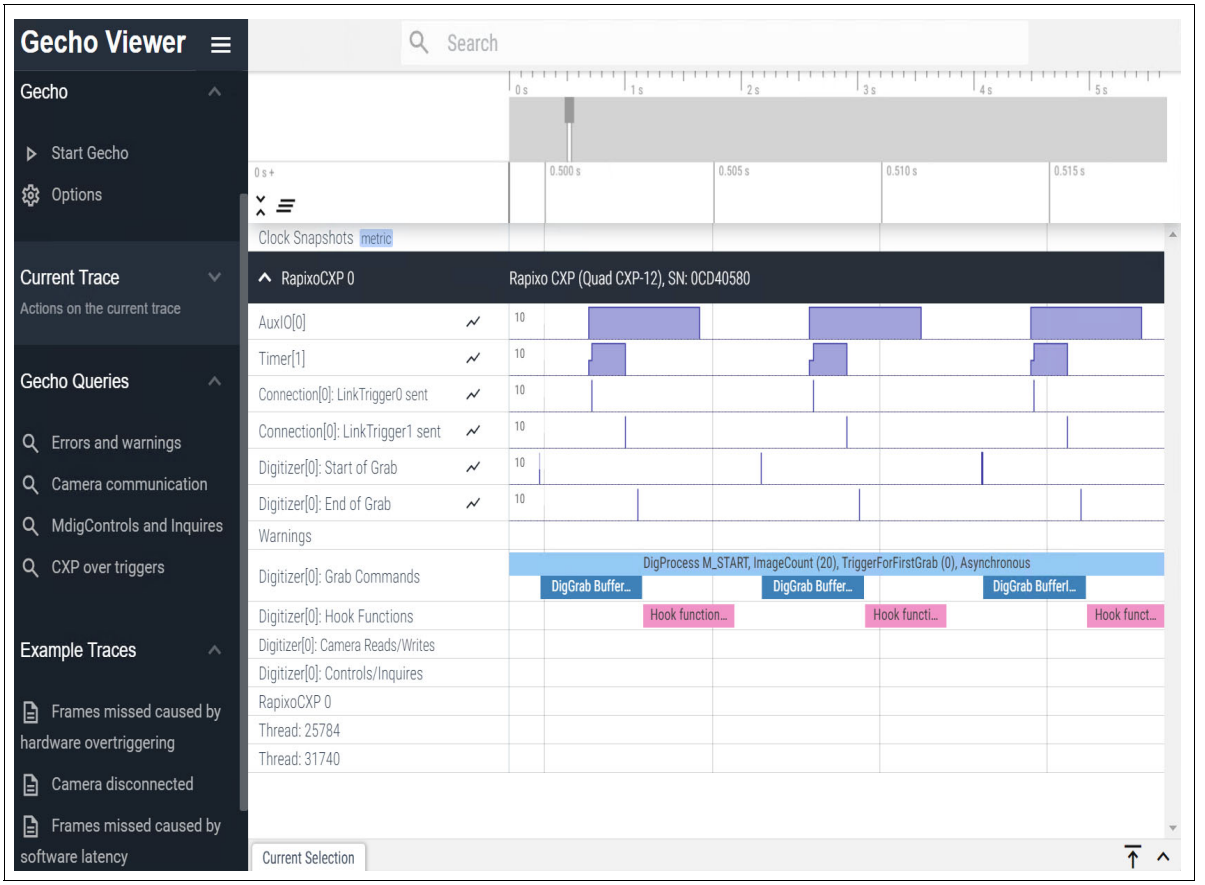
Gecho Viewer is an event logging application for Aurora Imaging Library applications that use a Zebra frame grabber (for example, Zebra Rapixo CXP). It allows you to log, visualize, and analyze hardware and software events that are seen by your frame grabber during the execution of your Aurora Imaging Library application (for example, camera communication, auxiliary I/Os, CXP triggers, on-board frame grabber events, and Aurora Imaging Library application calls related to the frame grabber). Debugging machine vision systems that use frame grabbers in high-end inspection can be tedious, time consuming, and error prone. Gecho Viewer simplifies and streamlines this complex process, allowing you to see events from the perspective of your frame grabber(s) so that even the fastest and most complicated machine vision systems can be easily analyzed and debugged. Gecho Viewer runs Aurora Imaging Gecho in the background to log the real-time events with minimal overhead.

Gecho Viewer allows you to easily:

- View and save a fully reconstructed precise timeline of events seen by the frame grabber(s), along with their associated details (for example, duration or start time), to assist with performance and trace analysis.
- Debug your setup during your bring-up phase.
- Confirm that your trigger sequence does what you expect.
- Diagnose problems.

Aurora Imaging Gecho and Gecho Viewer are distributed with Aurora Imaging Library and Aurora Imaging Library Lite.

Gecho Viewer with a loaded trace is shown below:



Gecho Viewer features

Gecho Viewer offers several features and functionality to assist in your trace analysis. The following features are provided with Gecho Viewer:

- **Logging events.** Logs hardware and software events when they are seen by your frame grabber during the execution of your Aurora Imaging Library application. Traces contain enough detail to fully reconstruct the timeline of events. Gecho Viewer runs Aurora Imaging Gecho in the background to log the events with minimal overhead. The following hardware and software events are logged by Gecho Viewer.

  Hardware events received and generated by the frame grabber:

  - Auxiliary I/O changes.
  - Timer events.
  - CXP triggers.
  - CXP protocol and connection errors.
  - Grabbing events.

  Software events:

  - Aurora Imaging Library application calls related to the frame grabber.
  - Camera and digitizer communication.
  - Grab and hook events.
  - Errors and warnings (for example, disconnection or missed frames).

- **Sentry mode.** Allows you to specify a keyword to trigger when to log an event. For example, set AuxIO[2] with Sentry mode to generate an event log whenever that signal occurs. This is helpful when diagnosing issues that are infrequent.

- **Analyze while logging.** Allows you to analyze an active log without stopping your trace, you can analyze what has already been logged.

- **Buffer queues.** Allows you to see how buffer queues are used, when acquired images are processed, and the duration of grab-related hooked function calls.

- **Trace file generation.** Automatically saves traces to a file.

- **Rich trace-based metrics and analysis.** Provides details and contextual information for events seen by your frame grabber (for example, the duration of a signal or start time of a signal) and presents it on a precise timeline. You can easily measure time elapsed, latencies, and detect anomalies during acquisition, among other metrics.

- **Visualize events.** Generates a high precision visualization of what is happening on or seen by your frame grabber(s). This allows you to easily understand what is happening in your system and simplifies troubleshooting.

- **SQL queries.** Allows you to query whether certain situations occur (for example, missed triggers) and jump to those time slices in the timeline. Most commonly needed queries are provided on the left sidebar. To query for other situations, create your own query either starting from the queries provided or from scratch; Gecho Viewer is backed by the SQLite query engine.

- **Example traces.** Includes example traces (logging samples) of common problems are included to help you recognize them in a trace.

  The examples provided include:

  - Camera disconnected.

  - Frames missed caused by hardware overtriggering.

  - Frames missed caused by software latency.

Service Information

If you have a problem with your equipment, contact Zebra Global Customer Support for your region. Contact information is available at: zebra.com/support. When contacting support, please have the following information available:

- Serial number of the unit
- Model number or product name
- Software/firmware type and version number

Zebra responds to calls by email, telephone, or fax within the time limits set forth in support agreements.

If your problem cannot be solved by Zebra Customer Support, you may need to return your equipment for servicing and will be given specific directions. Zebra is not responsible for any damages incurred during shipment if the approved shipping container is not used. Shipping the units improperly can possibly void the warranty.

If you purchased your Zebra business product from a Zebra business partner, contact that business partner for support.

Chapter

2

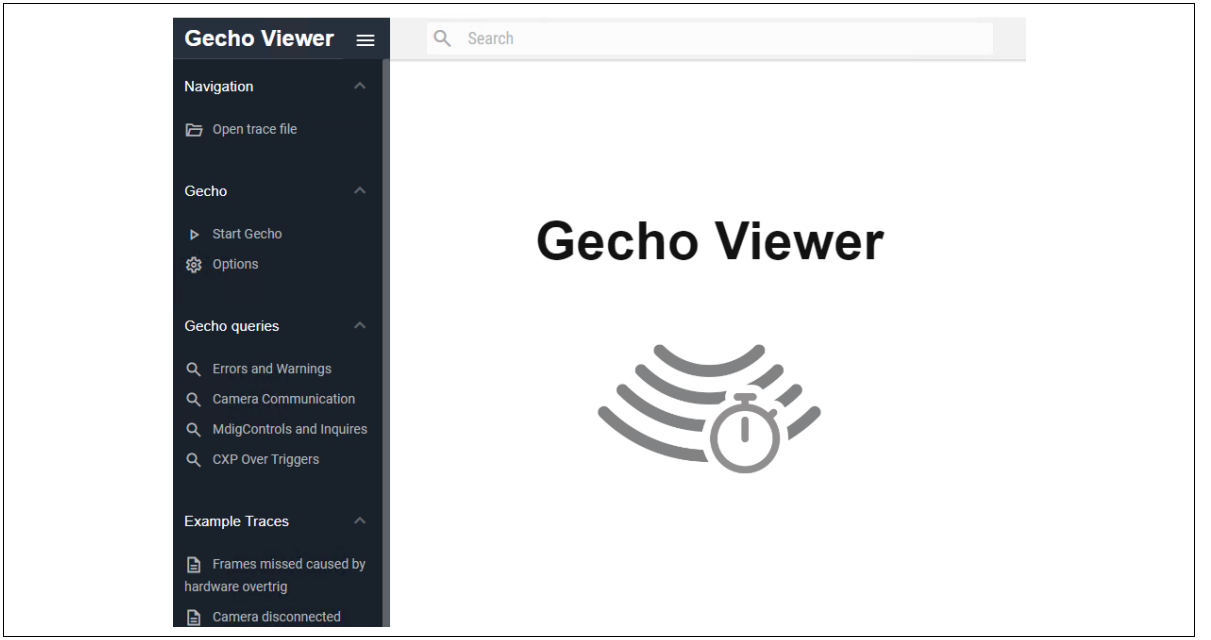
Using Gecho Viewer

This chapter describes the basic flow of events, as well as the Gecho Viewer interface and how it is used.

Basic steps for using Gecho Viewer

The following is the basic methodology for using Gecho Viewer:

1. Launch Gecho Viewer by opening the Aurora Imaging Configurator utility, expanding the **Boards** item, selecting Rapixo CXP, and clicking the Gecho Viewer button.



2. With Gecho Viewer open, click **Start Gecho**, to begin a trace. Any Aurora Imaging Library acquisition hardware or software events that are detected by the installed Zebra frame grabber are logged in a trace. If multiple boards of the same type are running, events are logged for all boards. If different board families are present, Gecho Viewer will arbitrarily chose one; select the required board type under **Options**.

3. Start your Aurora Imaging Library application. If it uses the frame grabber, you will see the counter next to **Load Trace** increase, indicating the amount of event data being logged. Note that some Aurora Imaging Library tools (for example, Aurora Imaging Capture Works, Aurora Imaging Rapixo Bench, and the Aurora Imaging Configurator utility) run Aurora Imaging Library in the background, so

their events are logged and displayed in Gecho Viewer as well. This means that you could open Aurora Imaging Capture Works and begin an acquisition to analyze its associated events. Note that you can start Aurora Imaging Gecho while your application is running or before starting it.

4. Click on **Load Trace**. A graphical representation of the current trace up to this point in time will be presented. If you don't stop the trace before loading the trace, Gecho Viewer will continue logging events in the background.

5. Analyze the trace. For typical use cases, see the *Typical cases and example traces* later in this chapter.

6. Stop the trace. The trace is automatically loaded after being saved to the directory specified below **Current Trace** in the left sidebar. You can click on the listed directory to copy the path.

❖ Note that the saved trace file is overwritten each time you start a trace. If necessary for later inspection, you can click Download in the left sidebar.

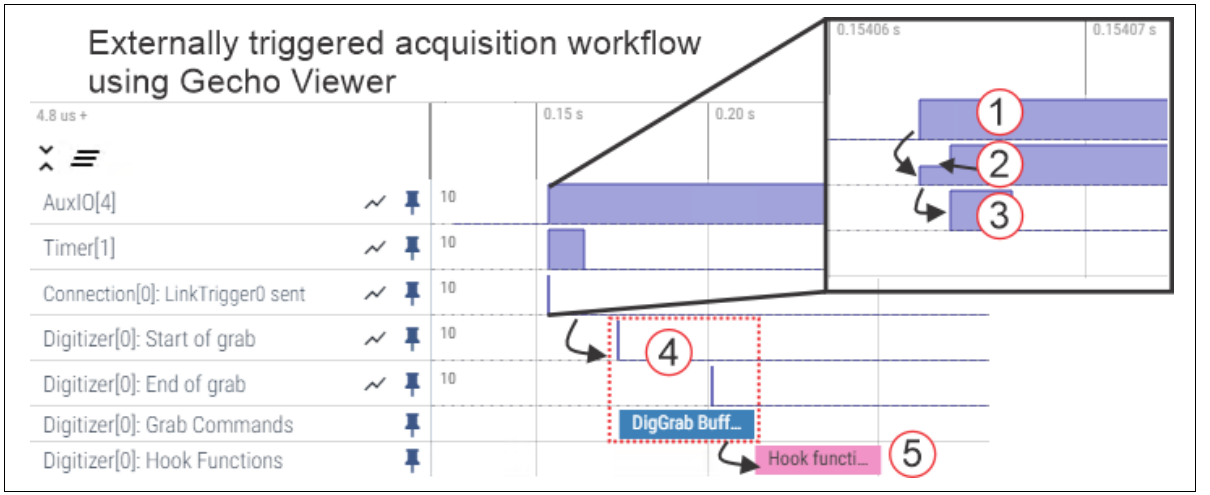
Options and loading a trace from file

Note that you can change the selected board type, as well as filter the software and hardware events that are registered by the trace by going to **Options**. You must make changes to these options before starting a trace for you to see them in your active trace.

If you have already run a trace and want to review it, click **Open trace file**.

Typical sequence of events during a triggered grab

Gecho Viewer logs events that are useful for understanding and troubleshooting setups which run Aurora Imaging Library applications that use Zebra frame grabbers (for example, camera communication, auxiliary I/Os, CXP triggers, on-board frame grabber events, and Aurora Imaging Library application calls related to the frame grabber). Each process is unique and generates a different set of events, but most triggered grab applications have a similar sequence of events. The following image illustrates a trace of the sequence of events displayed in Gecho Viewer:



1. A software or hardware event occurs. In the image above, the Aux I/O 4 signal is activated.

2. The event typically triggers a timer (to add a delay); the timer initiates its delay period followed by its activation period. In the image above, the event triggers Timer 1. The delay period is illustrated at half height on the timeline.

3. Upon the rising edge of the timer's activation period, a CXP trigger is sent to the camera to begin exposure. In the image above, LinkTrigger0 is activated on the rising edge of Timer 1 and the camera starts its designated exposure period.

4. When exposure completes, the camera starts sending data to the frame grabber and a Start of Grab event occurs. Upon the event, the frame grabber starts grabbing the incoming frame. When the camera ends its read out, an End of Grab event occurs.

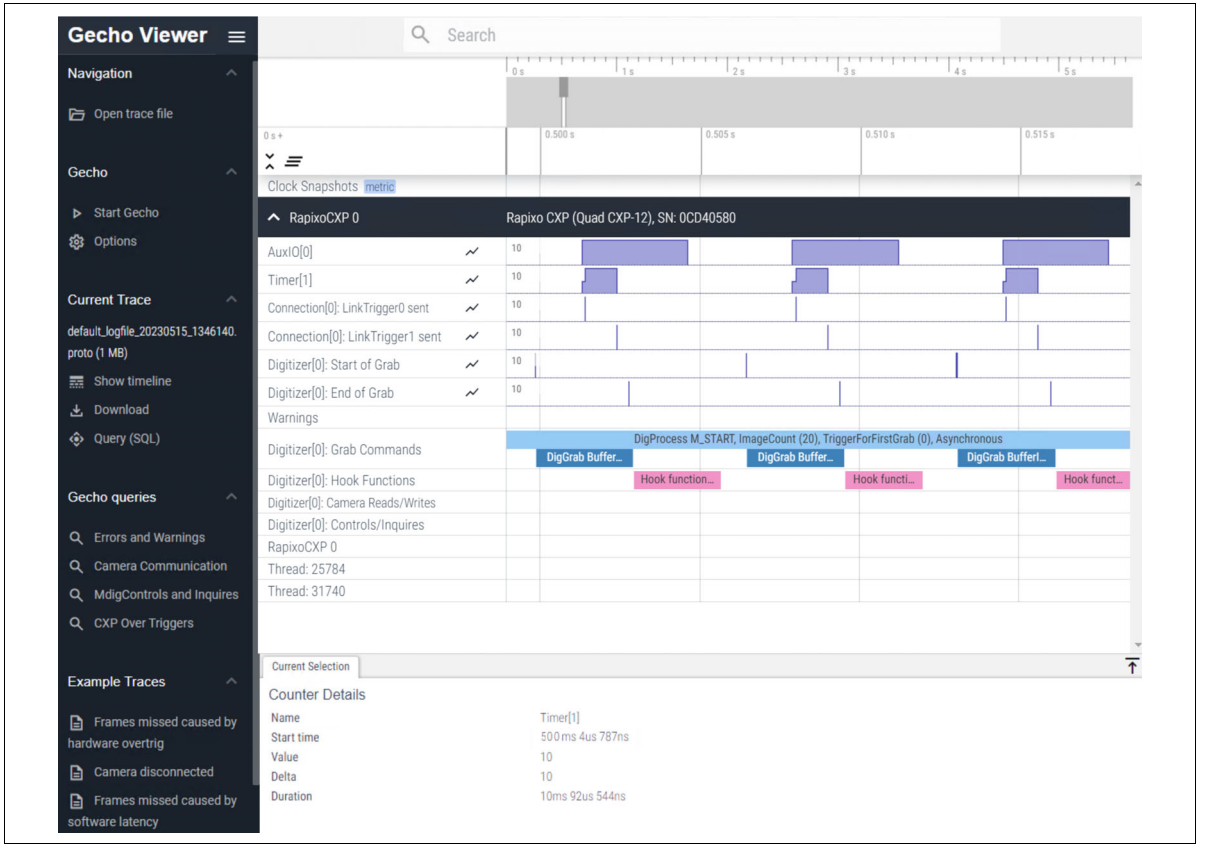
5. Upon the End of Grab event, the hook function is called (if MdigProcess() was called), which is run on the same Windows thread for all the buffers specified for that MdigProcess() call. There is a latency period between the DMA transfer and the hook function call, but the End of Grab event does not include the transfer to Host. In the image above, a hook function is called upon the End of Grab event.

   By analyzing the timing and duration of events seen by the frame grabber, you can confirm that your trigger sequence is working as expected. Note that the events that are logged depend on your Aurora Imaging Library application; your trace might include additional events or fewer events. For a list of the events that might be visible while using Gecho Viewer, see *Chapter 3: Gecho Viewer events*.

Navigating Gecho Viewer

Launch Gecho Viewer using Aurora Imaging Configurator. Once open, start a new trace or load the ongoing trace or a saved trace. You can also query the loaded trace for specific events.

The timeline of logged events is shown in the main window. You can select individual instances of events to see additional information listed in the **Current Selection** tab. The following image is the default look of the Gecho Viewer with a loaded trace and the **Current Selection** tab open.



❖ Note that when dealing with multiple frame grabbers of the same type, events are grouped per frame grabber; you can expand or collapse their timelines based on your needs.

Starting, saving, and loading traces

Gecho Viewer automatically detects hardware that is connected to and seen by your frame grabber if the frame grabber's Aurora Imaging Library driver is installed. To begin a new trace using the default options, click **Start Gecho**, and then run your Aurora Imaging Library application.

In Gecho Viewer when an Aurora Imaging Library application that uses the selected frame grabber is running, the counter next to **Load Trace** increments to indicate the amount of event data being logged. Note that some Aurora Imaging utilities also allocate/use the frame grabber (for example, Aurora Imaging Capture Works and Aurora Imaging Configurator), so these events are also traced. While the trace is running, you can refresh the timeline with live data by clicking **Load Trace.** Once you have enough trace data, you can begin your trace inspection. Click **Stop Gecho** when you want to complete your trace; the timeline automatically updates.

Once you have finished acquiring a trace, you can also save it for later inspection. To save/download a generated trace file, click **Download** in the left sidebar. If you already have a trace that you want to load, click **Open trace file** in the left sidebar.

Gecho Viewer hotkeys and mouse actions

After you load a trace, use hotkeys to navigate through the information found on the trace timeline. The table below describes hotkeys for Gecho Viewer:

Interface element	Description
WASD keys	Use the W and S keys to zoom in and out, and use the A and D keys to scroll left and right. Move your mouse pointer to the area on which to zoom in (don't click); you might need to do this multiple times as you zoom in. Scrolling left and right is useful when already zoomed into the timeline.
Scroll wheel	Use the scroll wheel to scroll up and down the list of logged events.
Click and drag on timeline	Click and drag to create a selection box. This allows you to measure the duration between two events or an entire group of events. If you click and drag over a group of events, all the events will be listed with associated timing information in the Current Selection tab; you can double-click on a specific one to go to that event.
Click on events	Click on an individual event to ascertain additional specialized information relating to the selected event (for example, start time, duration, and slice ID). The information is displayed in the Current Selection tab; in some cases, arrows showing associated events are also displayed (for example, clicking on a hook function event displays an arrow indicating the grab command whose grab buffer it is processing).

Duration and time of events

By default, all times of logged events are provided in UTC. You can change how duration is displayed by right-clicking on duration, selecting **Duration precision**, and choosing between **Full** or **Human readable**.

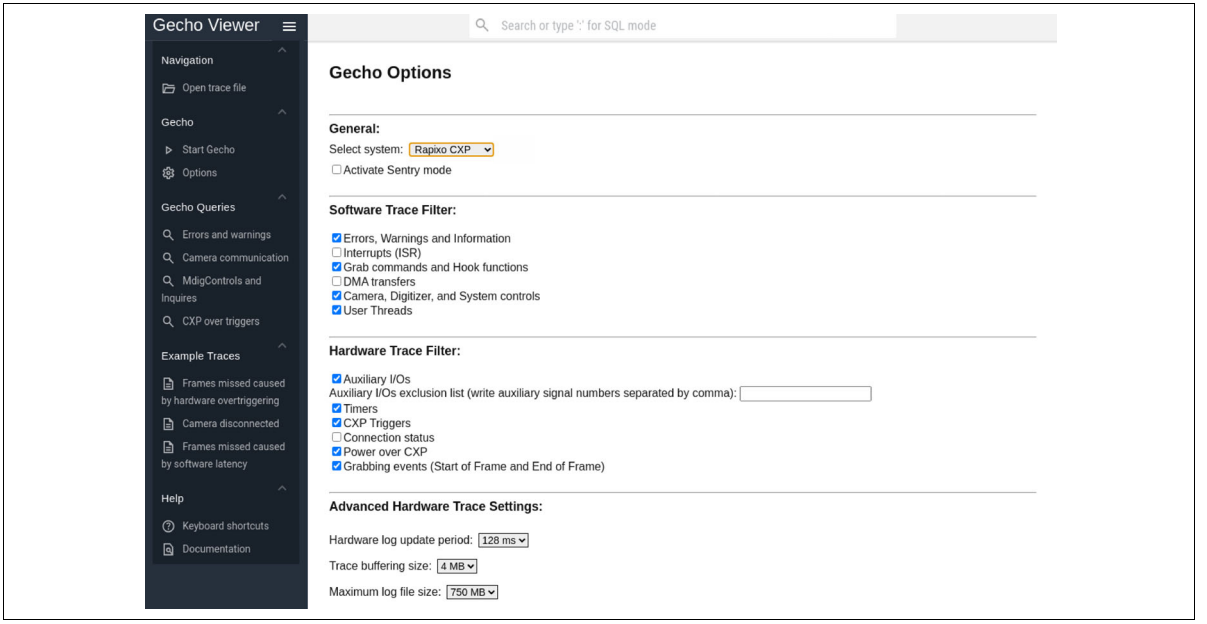
Selecting multiple events

Select multiple events by clicking and dragging a selection box over the required events. If you select a group of timers or counters, you can also see the counter or timer count across all the selected events, along with the average rate across the selected events.

Multiple grab events can be selected and sorted based on timestamp or duration. This is helpful when searching for processing events that take the most time, which can assist in troubleshooting.

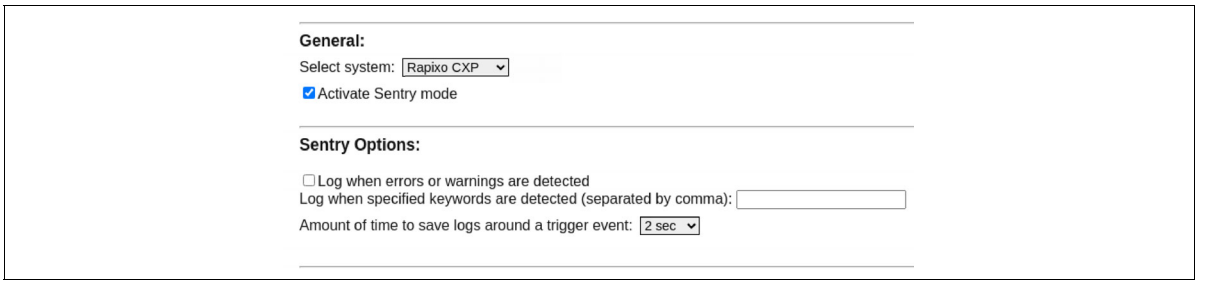
Gecho Viewer options

By default, all hardware events, and all software events except for DMA transfers and interrupts, are traced. This is useful to get a general idea of what your frame grabber sees, but is not necessary if you are looking for a specific set of events. To enable or disable the logging of specific events, click **Options**, and check the events to trace; uncheck those you do not want. You need to stop and restart your trace once you have selected a different set of events to trace.



Sentry mode

When enabled, Sentry mode logs events that occur around errors, warnings, or keywords. This is used to assist you with troubleshooting by giving you information surrounding the conditions that occur during errors, warnings, and keywords. To set Sentry mode, select **Activate Sentry mode** and enable **Log when errors or warnings are detected.**You can also set the number of seconds to log events before and after the trigger to assist with troubleshooting.



You can also set specific keywords that trigger the logs of all events that occur around the specified trigger. To do this, after you have activated Sentry mode, enter the keywords in the **Log when specified keywords are detected** text box, separated by commas. Events are only logged when the keyword is registered during acquisition. If both **Log when errors or warnings are detected** and he **Log when specified keywords are detected** are enabled, events are logged when an error or warning occurs or when the keywords are detected.

Software trace filter

To filter out unnecessary software events that are not in use or helpful, select only the software signals that are necessary for your trace.

Hardware trace filter

To filter out unnecessary signals that are not in use or helpful, select only the hardware signals that are necessary for your trace. You can also specify a list of auxiliary I/O signals that should be excluded by listing the signals' numbers in the text box next to Auxiliary I/O, separated by commas. For example, encoders can generate many unnecessary logs and excluding them can help you focus on more useful signals.

Advanced hardware trace settings

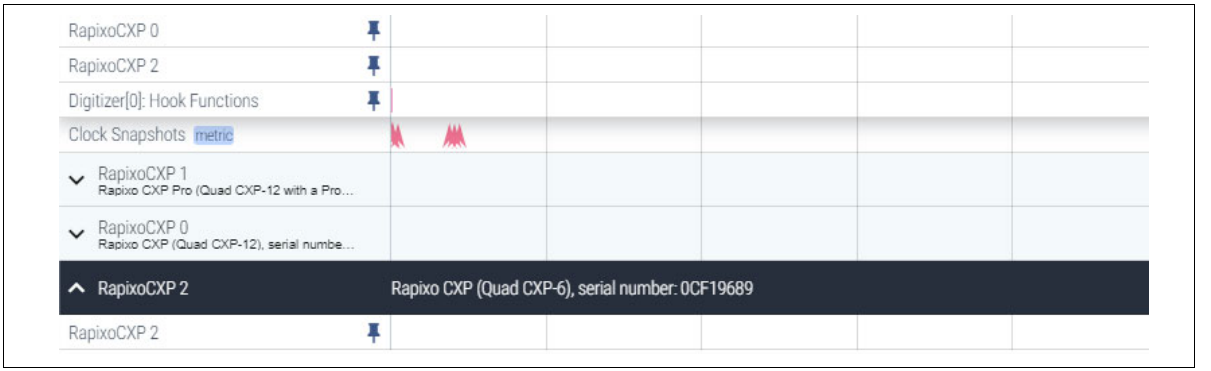
To help streamline and organize your log files, set various trace file settings, such as setting the update period for a log, the trace buffering size, and the maximum log file size.

Organizing and querying traces

Gecho Viewer provides search and query functionalities for events logged in a trace. Generally hardware logs are grouped together and software logs are grouped together. You can also pin certain events so that they appear at the top of the list, instead of having to search through the generated events to find them.

Pinning events

Pinning events allows you to display events that you want to compare closer together, when you have many events or events from multiple frame grabbers. To add a specific event to your favorites, hover your mouse over the required event and click the pin ( ). When the pin becomes solid ( ), it is saved and the event now appears above the list of frame grabber(s), regardless of whether that frame grabber's timeline is expanded or not. The order in which you pin events is the order in which they appear in the pinned list.



Using the search bar

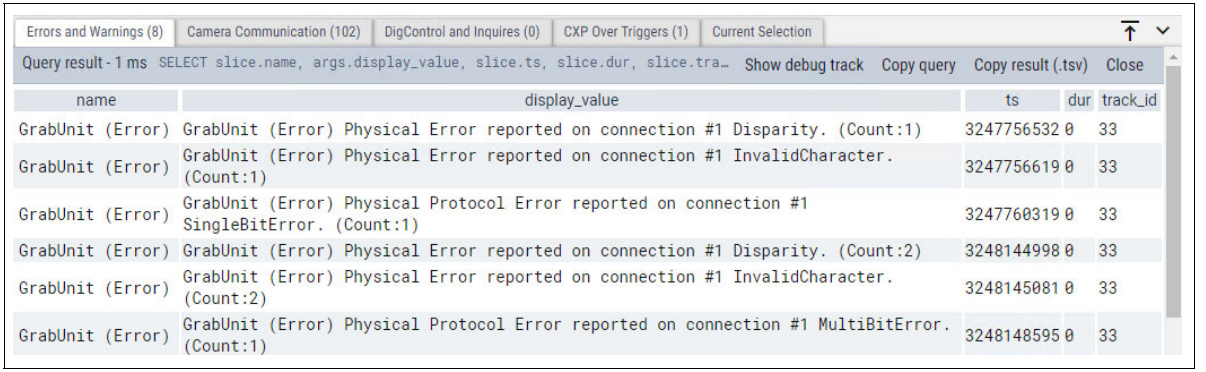
The search bar available at the top of the Gecho Viewer window allows you to search for specific functions, buffer IDs, or events that are of interest to you, without the need for specific SQLite syntax. When you search for something, you also see how many times it occurs in the current trace, and at which instance you are currently looking.

Using predefined queries or creating your own

To find if and when certain situations occur during the trace, Gecho Viewer allows you to query the traced information. Predefined common queries are quickly accessible in the sidebar, but you can also customize or create your own queries; queries are written in SQLite.

Predefined queries

Predefined common queries are found on the left sidebar under the **Gecho Queries** heading. When you click on a query, its associated tab opens below the timeline, in the same place as the **Current Selection** tab.The image below shows what the query tabs and their associated options look like in Gecho Viewer:



name	display_value	ts	dur	track_id
GrabUnit (Error)	GrabUnit (Error) Physical Error reported on connection #1 Disparity. (Count:1)	3247756532	0	33
GrabUnit (Error)	GrabUnit (Error) Physical Error reported on connection #1 InvalidCharacter. (Count:1)	3247756619	0	33
GrabUnit (Error)	GrabUnit (Error) Physical Protocol Error reported on connection #1 SingleBitError. (Count:1)	3247760319	0	33
GrabUnit (Error)	GrabUnit (Error) Physical Error reported on connection #1 Disparity. (Count:2)	3248144998	0	33
GrabUnit (Error)	GrabUnit (Error) Physical Error reported on connection #1 InvalidCharacter. (Count:2)	3248145081	0	33
GrabUnit (Error)	GrabUnit (Error) Physical Protocol Error reported on connection #1 MultiBitError. (Count:1)	3248148595	0	33

To find where the queried events occur, double-click on a result in its associated tab, and the timeline will align to the timestamp where it occurred. The duration, timestamp, and track_id also appear next to each queried event.

The four predefined queries provided by Gecho Viewer are:

- **Errors and Warnings.** Any detected errors or warnings that occur during the trace are displayed in a corresponding tab (for example, Connection#N is now in an unlock state, or Physical Error is reported on Connection #N).

- **Camera Communication**. Any camera communication event that occurs between the frame grabber and the camera are displayed in a corresponding tab. For each camera communication event, the register address on the frame grabber is displayed and the data address on the camera is displayed. When possible, Gecho Viewer will map the data address of the camera to a meaningful camera features (for example, AcquisitionFrameRate or ExposureTime).

- **MdigControls and Inquires**. Any MdigControl or MdigInquire function calls from the application to the frame grabber appears in a corresponding tab.

- **CXP overtriggers**. Any missed triggers that occur during the trace are displayed in a corresponding tab.

Custom queries

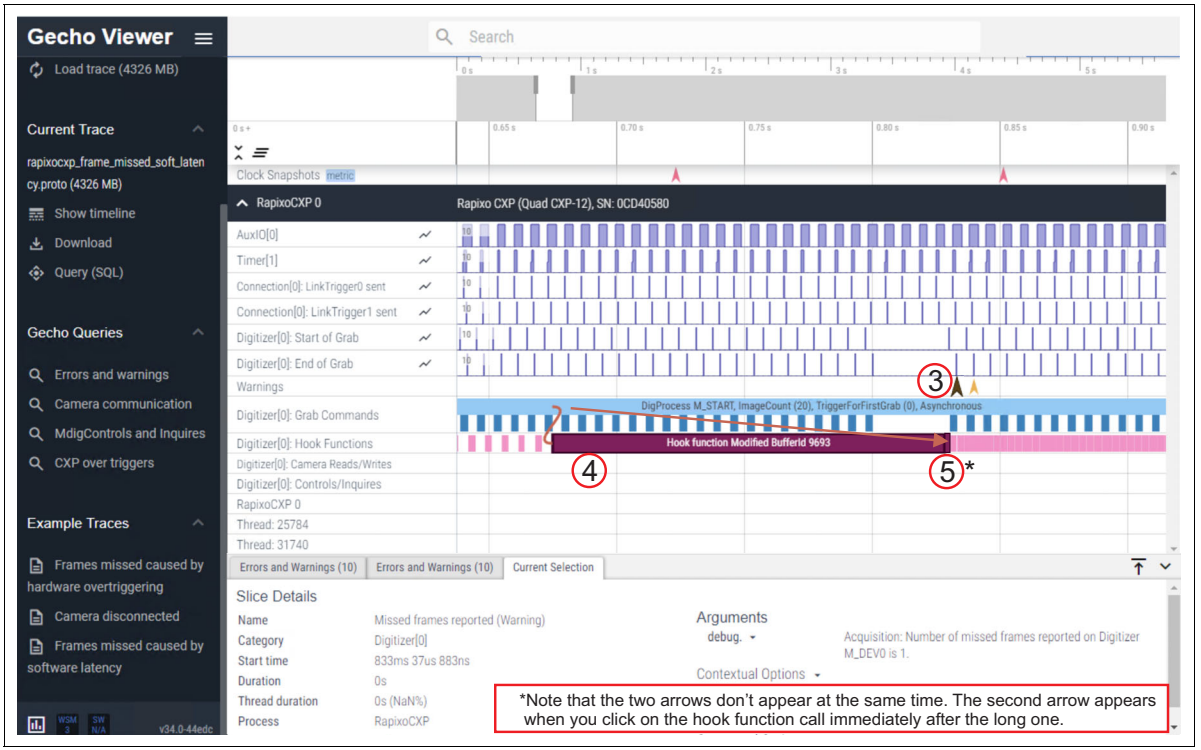
At the top of the selected query tab, you will have options to **Show debug track**, **Copy query**, **Copy result** (**.tsv**), or **Close**. Select **Copy query** in the top right of the query's tab to copy the whole query and do a more narrow search with that specific query (for example, searching for a specific time slice). Once the query is copied, select **Query** (**SQL**) under the **Current Trace** heading, paste the query, modify it accordingly, and press CTRL+ENTER to search; the list of time slices that meet the specified query will be listed in a corresponding tab. Double-click on a result in the tab, and the timeline will align to the timestamp where it occurred. Note that the query should follow the SQLite syntax.

Typical cases and example traces

Example traces are available to illustrate typical cases where Gecho Viewer can be used to assist with troubleshooting and analyzing your machine vision system. The following subsections explain the three example traces that are available on the left sidebar of Gecho Viewer. In the following examples, numbers shown in the image reflect the step that is described in the text under the screenshot.

Case 1: Missed frames caused by software latency

The **Frames missed caused by software latency** example trace shows what the timeline looks like when software latency causes a missed frame. The following screenshot, along with numbered steps, explain how to analyze the example trace:



To analyze the example trace:

1. Load the **Frames missed caused by software latency** example trace by clicking on it in the sidebar. You see that the grabbing was initiated using MdigProcess with 20 buffers.

2. Zoom out on the timeline by pressing the S key. You see that warnings have occurred and that some hook function calls ran much longer than others.

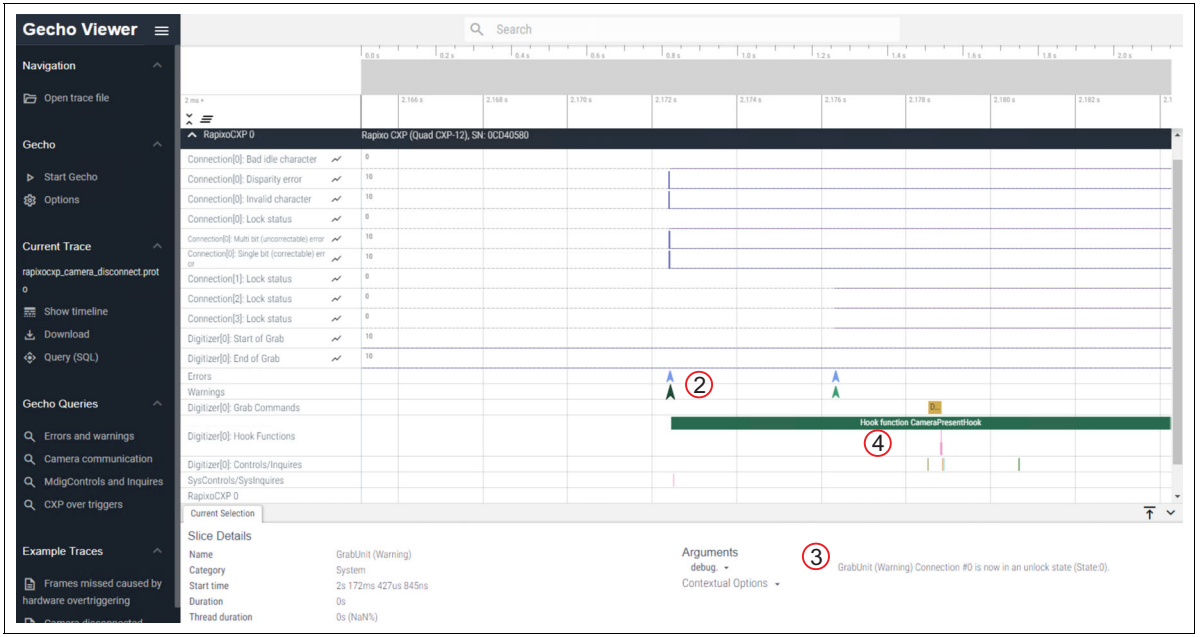
3. Click on a warning and zoom in on it by pressing on the W key and keeping your mouse pointer over the area in the timeline while zooming. Besides the hook function taking too long, you see that some grabs are missing. The warning is also described in Current Selection tab.

4. Click on the instance of the hook function that took too long. An arrow shows which buffer it was processing, and the **Current Selection** tab reports how long it took. Note that although many grabs are buffered during the processing of the long hook function call, at some point there are no more buffers to buffer the grab. After the long hook function instance finishes, processing starts on the buffer following the one that caused the delay.

5. Click on the hook function call just after the long one. An arrow appears showing which buffer it is processing. You can see that the delay before being processed was quite substantial. With each subsequent buffer, the delay between being grabbed and being processed is reduced. Until finally processing catches up.

Case 2: Missed frames caused by a disconnected camera

The **Camera disconnected** example trace shows what the timeline looks like when there is a physical issue with the cable or the camera itself. Below is a screenshot of the example, along with numbered steps that explain how to analyze the example trace.



To analyze the example trace:

1. Load the **Camera disconnected** example trace by clicking on it in the sidebar.

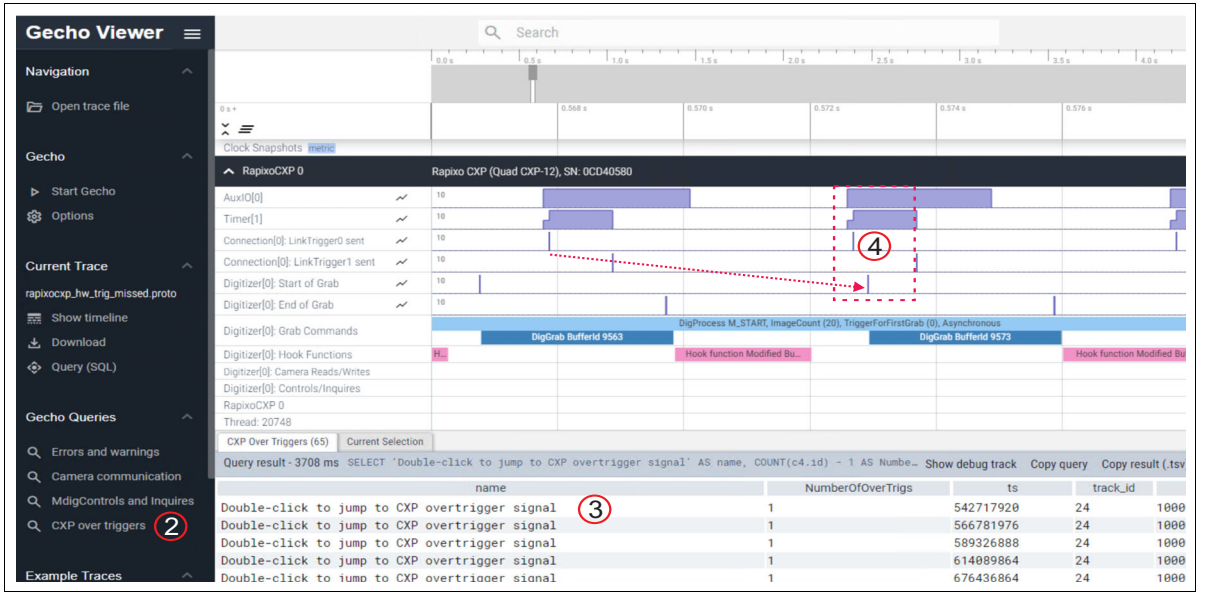
2. Use the W key to zoom in on it a bit and use the D key to scroll to the far right of the timeline. At the end of the timeline, you will notice a warning; click on it.

3. Notice that a grab happened and then there were no more grabs. Instead, warnings happened on the connections (Link 0 and Link1), and their lock status went from undefined to 0. When you wiggle the cable, there are always some invalid characters and errors that happen just before the unlock.

4. On the software side, notice that the CameraPresent hook function was called. Click on it. The **Current Selection** tab reports that the hook function was called (because the camera was not present), and stopped MdigProcess.

Case 3: Frames missed caused by hardware overtriggering

The **Frames missed caused by hardware overtriggering** example trace shows what the timeline looks like when a missed frame occurs due to hardware overtriggering. In this example, the frequency of triggers is increasing, and at certain points, the frequency becomes too high so some triggers are missed. In most cases, this type of issue does not report an error because if a CXP trigger is sent to the camera while it is exposing a frame, it doesn't register it and just starts again upon the next trigger. To detect this type of error, you need to analyze the trace appropriately when you know that the frequency is quite high or can occasionally be quite high; in this case Gecho Viewer does not report an error. Using the **CXP overtriggers** query is a good way to quickly check if any missed triggers occurred. Note that if a trigger is sent to a timer while the timer is active, it misses it and just starts again on the next trigger, but an error is generated. Below is a screenshot of the example, along with numbered steps that explain how to analyze the example trace:



To analyze the example trace:

1. Load the **Frames missed caused by hardware overtriggering** example trace by clicking on it in the sidebar. Notice that there are no reported errors to zoom in on, except that the triggers are occurring very frequently so there is a potential of overtriggering occurring.

2. Click on the **CXP overtriggers** query in the sidebar. It finds that in fact some triggers have been missed and lists them in a **CXP overtriggers** tab.

3. Double-click on one of the entries in the list. Then, zoom out a little to see the grab that happens before and after the missed trigger.

4. Generally, after LinkTrigger0, there is a short delay before the Start of Grab. If a CXP trigger is sent to the camera during that time, an overtrigger event occurs. Notice that the Start of Grab event happens after the CXP trigger of the next grab with the AuxIO event that occurs shortly after 0.572 seconds, this causes the CXP trigger to be missed by the camera.

Chapter

3

Gecho Viewer events

This chapter describes possible Gecho Viewer events.

Gecho Viewer events

This chapter describes events that are logged by Gecho Viewer. Using this information, you can interpret and analyze the timeline of events generated or seen by your frame grabber; they are always from the perspective of the frame grabber. Depending on your workflow and equipment, different combinations of events are logged. Gecho Viewer only logs and displays those events that are visible to the frame grabber and have been selected for tracing; you can filter out events for tracing under **Options**. This chapter lists the events in the order that they appear in Gecho Viewer.

❖ Note that hardware events are reported in the clock domain of the frame grabber; software events are reported in the clock domain of the Host system. Drifts are always automatically resolved. There are systematic clock snapshots to resync the two clocks in the application, so that you can visualize the events in the same clock domain.

Hardware events

Hardware events are indicated by a rising edge symbol ( ). These hardware events are always from the perspective of the frame grabber. To see additional information about any event (for example, start time, value, or duration), click on its associated timing diagram and inspect the details under the **Current Selection** tab. The following table describes various hardware events and their associated outputs.

Event name	Description
AuxIO[n]	Indicates the state of the specified auxiliary I/O. When the event timing diagram is high, the specified auxiliary I/O is active; when it is low, the auxiliary I/O is not active. When there is no solid line, this specifies that the state is unknown. Note that the Aurora Imaging Library application doesn't need to use the auxiliary I/O for activity to be recorded.
Timer[n]	Indicates the state of the specified timer. When the event timing diagram is high, the timer is in the active phase; when it is low, the timer is not active. When it is halfway, it specifies the delay phase is active on the timer. Note that if the output signal of the timer has been inverted (MdigControl with M_TIMER_OUTPUT_INVERTER), the event timing diagram is inverted.
Digitizer[n]: Start of Grab	Indicates when the camera has started its readout of a frame.
Digitizer[n]: End of Grab	Indicates when the camera has ended its read out of a frame.
Connection status	Indicates the trigger, lock, and reset status of the specified connection coming to and from the camera.
Connection[n]: LinkTriggerN sent	Indicates when a trigger packet has been sent on the specified connection from the frame grabber to the camera.
Connection[n]: Lock status	Indicates the lock status of the downconnection receiver on the specified connection. The direction of data flow on the downconnection receiver is from the camera to the frame grabber, using a high speed connection. When the timing diagram is high, the downconnection receiver is locked to the camera's downconnection signal.
Upconnection[n]: Lock status of data forwarding	Indicates the lock status of the upconnection receiver on the specified connection. The direction of data flow on the upconnection receiver is from the slave frame grabber to the current frame grabber, using a low speed connection. This only applies to the transmitter ports of data forwarding boards (for example, Zebra Rapixo CXP Quad Data Forwarding). When the timing diagram is high, the upconnection receiver is locked to the slave frame grabber's upconnection signal.
Connection[n]: Reset state	Indicates the reset status of the downconnection receiver on the specified connection. The direction of data flow on the downconnection receiver is from the camera to the frame grabber, using a high speed connection. When the timing diagram is high, the downconnection receiver has been reset.
Upconnection[n]: Reset state	Indicates the reset status of the upconnection receiver on the specified connection. The direction of data flow on the upconnection receiver is from the slave frame grabber to the current frame grabber, using a low speed connection. This only applies to the transmitter ports of data forwarding boards (for example, Zebra Rapixo CXP Quad Data Forwarding). This is set to one when the upconnection receiver has been reset.

Event name	Description
CXP stream errors	Indicates errors related to acquisition on CXP streams.
Digitizer[n]: Incorrect packet size	Indicates that a CXP packet received by the frame grabber does not contain the amount of data specified by the DSIZEP field in the packet header. When the timing diagram is high, a CXP packet with the wrong amount of data is detected.
Digitizer[n]: Incorrect amount of data received for a line	Indicates that a CXP line received by the frame grabber does not contain the amount of data specified by the DSIZEL field in the frame header. In a line scan, the data received is compared to the value of the GHTOTAL register, since information from DSIZEL is not used for a line scan. This is set to one when a CXP line with the wrong amount of data is detected.
Digitizer[n]: Incorrect number of lines	Indicates that a CXP frame received by the frame grabber does not contain the number of lines per frame specified by the YSIZE field in the frame header. This is set to one when a CXP frame with the wrong amount of lines is detected.
Digitizer[n]: Incorrect packet tag	Indicates that the packet tag number does not match what was expected. This could indicate that there are dropped packets. This is set to one when a CXP packet with a wrong tag is detected.
Connection errors	Indicates errors related to the CXP connections.
Connection[n]: CDC FIFO overrun	Indicates that an overrun has occurred in the CXP front-end FIFO. This is set to one when an overrun occurs. Possible conditions that cause this error can include: a wrong tag, an error in the data packet (for example, missing End of Packet), a packet greater than the maximum supported size (8 K), or some physical error. This kind of error is typically accompanied by other errors.
Connection[n]: Bad idle character	Indicates that a bit error was found in the IDLE WORD. This is set to one when a bad character is detected. This error is self-corrected since the specific IDLE character will be ignored.
Connection[n]: Multi bit (uncorrectable) error	Indicates that the specified connection has detected header information that cannot be decoded and that there is an error in more than one bit. This is set to one when a uncorrectable error occurs. For example, the header information 0x02020303 is not correctable because you cannot deduce whether the correct character to repeat is 02 or 03.
Connection[n]: Single bit (correctable) error	Indicates that the specified connection has detected a single bit error in the header information, which can be corrected if 3 of the 4 bytes of the repeated character are the same. This is set to one when a correctable error occurs. For example, the header information 0x02020302 is correctable because you can deduce that 02 is the repeatable character.
Connection[n]: CRC error	Indicates that the specified connection has detected a cyclical redundancy check (CRC) error in a stream data packet, which results from a corruption that occurs during the transmission of a packet. This is set to one when a CRC error occurs. This error indicates either the CRC code or a pixel value is corrupted. Typically, this results from the physical layer and might be accompanied by 8b/10b or bad character errors. This is an uncorrectable error.
Connection[n]: Comma realign	Indicates that a realignment has occurred after the initial alignment has been made. When the timing diagram is high, the DWORD alignment on the serial data steam has been lost. Typically, this occurs due to physical errors and issues with signal integrity on the connection.

Event name	Description
Connection[n]: Disparity error	Indicates that a wrong disparity was registered by the decoder after the disparity check was performed. This is set to one when a 8b/10b disparity error occurs. Typically, this occurs due to physical errors and issues with signal integrity on the connection.
Connection[n]: Invalid character	Indicates that a character is not part of the valid symbols that can be sent on the 8b/10b protocol, monitored by the GTX transceiver, was detected. This is set to one when a 8b/10b bad character error is detected. Typically, this occurs due to physical errors and issues with signal integrity on the connection.
PoCXP (Error): Pixel buffer overrun	Indicates that the pixel buffer of the CoaXPress module is full but is still trying to handle incoming traffic, meaning an overflow occurred. This is set to one when a pixel buffer overrun has occurred. The pixel buffer is where data coming from all streams of all connections is stored before being formatted.
PoCXP (Error): Output buffer overrun	Indicates that the output buffer of the CoaXPress module is full but is still trying to handle incoming traffic, meaning an overflow occurred. This is set to one when a output buffer overrun has occurred. The output buffer contains already formed packets coming from all CXP connections and streams and is the last boundary before the memory bridge.
PoCXP (Error): PoCXP over-current detected on connection #N	Indicates that an over-current condition has been detected, which is defined as any current above 1 A over a pre-set interval of time. This is set to one when an over-current condition has been detected. Power is automatically cut on the Power-over-CXP line when this is detected.

Software events

Software events do not have any indication symbol next to them but are found below the hardware signals in the events list. These software events are always from the perspective of the frame grabber and not the connected camera. To see additional information about any of the software events (for example, start time, value, or duration), click on their associated timing diagram and inspect the details under the **Current Selection** tab. The following table describes various software events and their associated outputs.

Event name	Description
Digitizer[n]: Interrupts (ISR)	Indicates software events that occur when the driver is notified of a corresponding hardware event. These are interrupts generated by the frame grabber and are dispatched by the operating system (for example, Windows or Linux) to the driver.
Start of Grab Frame Interrupt	Indicates that the Start of Grab Frame signal has occurred and initiated an interrupt.
End of Grab Frame Interrupt	Indicates that the End of Grab Frame signal has occurred and initiated an interrupt.
Digitizer[n]: Grab Commands	Indicates the start and duration of the acquisition of a frame. This software event is initiated by the **Start of Grab Frame** signal and completes when the frame is completely present in the grab buffer.
Digitizer[n]: Camera Read/Write	Indicates when camera features are being read or written. The camera address where the data is read from or written to is also displayed. The length in which the timing diagram is high reflects the time that it takes to perform the operation. The actual duration is reported in the **Current** tab.
Digitizer[n]: DMA Transfers	Indicates when a DMA transfer from on-board memory to Host memory takes place, along with the destination BufferID (Aurora Imaging Library buffer identifier) associated with the transfer.
Digitizer[n]: Hook Functions	Indicates when a hook-handler function is called and the function's duration. If the hook-handler function was set up using MdigProcess and you click on a hook-handler function call in the timeline, an arrow indicates the grab command whose grab buffer it is processing.
Digitizer[n]: DigInquire/DigControl	Indicates when a MdigControl or MdigInquire call was made and its duration. It also indicates the control/inquire type affected. The length in which the timing diagram is high reflects the time that it takes to perform the operation. The actual duration is reported in the **Current** tab.
Thread: [thread ID]	Indicates the timeline of a thread. Clicking on a specific event in the timeline, displays the event and its duration.
SysControls/SysInquires	Indicates when a MsysControl or MsysInquire call was made and its duration. It also indicates the control/inquiry type affected (for example, M_TEMPERATURE_FPGA, which inquires the temperature of the board). The length in which the timing diagram is high reflects the time that it takes to perform the operation. The actual duration is reported in the **Current** tab.
Operation Error	Indicates when an error has been detected by internal calls that are used to monitor the operation of the system. Note that Aurora Imaging Configurator does some internal MsysControl and MsysInquire calls, and if you have Aurora Imaging Configurator open while a board without a fan is installed (for example, Zebra Rapixo CXP Quad), you will notice the following error: “Operation error: This system does not have a fan.” This is normal; it is not a real error because there is no fan on this board.

Appendix A: Glossary

This appendix defines some of the specialized terms used in the Gecho Viewer documentation.

Glossary

- **Auxiliary I/O signals.**

  Auxiliary input/output signals. Non-video digital signals that can support one or more functionalities depending on the auxiliary signal (for example, trigger input or timer output). These signals are also known as general purpose I/O signals or GPIO signals.

- **Bandwidth.**

  A term describing the capacity to transfer data. Greater bandwidth is needed to sustain a higher transfer rate. Greater bandwidth can be achieved, for example, by using a wider bus or by increasing the clock frequency at which an interface or a processing core operates (for example, increasing the DDR4 SDRAM clock frequency).

- **CoaXPress (CXP).**

  An asymmetric high-speed communication standard used primarily for video and image data transfer. CoaXPress supports the transmission of video data, control signals, triggers, and power on the same coaxial line. For each connection, CoaXPress supports downlink data rates of up to 12.5 Gbits/sec (1.25, 2.5, 3.125, 5.0, 6.25, and 12.5 Gbits/sec) and an uplink data rate from 0.02 Gbits/sec (21.33 Mbits/sec) to 0.4 Gbits/sec (42.67 Mbits/sec).

- **Event.**

  A moment that indicates when any external I/Os, on-board events, or functions have been called.

- **Exposure time.**

  Refers to the period during which the image sensor of a video source is exposed to light. As the length of this period increases, so does the image brightness.

- **Frame.**

  A single image grabbed from a video source.

- **Latency.**

  The time from when a command is sent to when its operation is started.

- **Trace.**

  Tracing involves collecting highly detailed data about the execution of a setup. A single continuous session of recording is called a trace file or trace for short. Traces contain enough detail to fully reconstruct the timeline of events, and often include low-level kernel events like scheduler context switches, thread wakeups, or syscalls.