MPLAB® Data Visualizer User's Guide

Notice to Development Tools Customers

Important:
All documentation becomes dated, and Development Tools manuals are no exception. Our tools and documentation are constantly evolving to meet customer needs, so some actual dialogs and/or tool descriptions may differ from those in this document. Please refer to our website (www.microchip.com/) to obtain the latest version of the PDF document.

Documents are identified with a DS number located on the bottom of each page. The DS format is DS<DocumentNumber><Version>, where <DocumentNumber> is an 8-digit number and <Version> is an uppercase letter.

For the most up-to-date information, find help for your tool at onlinedocs.microchip.com/.

Table of Contents

1. Preface

MPLAB Data Visualizer documentation and support information is discussed in this section.

1.1. Conventions Used in This Guide

The following conventions may appear in this documentation:

Table 1-1. Documentation Conventions

Description	Represents	Examples
Arial font:		
Italic characters	Referenced books	*MPLAB® Data Visualizer User's Guide*
	Emphasized text	...is the *only* compiler...
Initial caps	A window	the Output window
	A dialog	the Settings dialog
	A menu selection	Select File and then Save.
Quotes	A field name in a window or dialog	"Save project before build"
Underlined, italic text with right angle bracket	A menu path	*File>Save*
Bold characters	A dialog button	Click **OK**
	A tab	Click the **Power** tab
N'Rnnnn	A number in verilog format, where N is the total number of digits, R is the radix and n is a digit.	4'b0010, 2'hF1
Text in angle brackets < >	A key on the keyboard	Press <Enter>, <F1>
Courier New font:		
Plain Courier New	Sample source code	`#define START`
	Filenames	`autoexec.bat`
	File paths	`C:\Users\User1\Projects`
	Keywords	`static`, `auto`, `extern`
	Command-line options	`-Opa+, -Opa-`
	Bit values	`0, 1`
	Constants	`0xFF, 'A'`
Italic Courier New	A variable argument	*`file.o`*, where *`file`* can be any valid filename
Square brackets [ ]	Optional arguments	`xc8 [options] files`
Curly brackets and pipe character: { | }	Choice of mutually exclusive arguments; an OR selection	`errorlevel {0|1}`
Ellipses...	Replaces repeated text	*`var_name`* `[, var_name...]`
	Represents code supplied by user	`void main (void) { ... }`

1.2. Recommended Reading

This document describes how to use the MPLAB Data Visualizer. Other useful documents are listed below. The following Microchip documents are available and recommended as supplemental reference resources.

Atmel Data Visualizer User's Guide

A previous data visualizer, this program processes and visualizes data. The Atmel Data Visualizer is capable of receiving data from various sources such as the Embedded Debugger Data Gateway Interface (DGI) and COM ports.

Data Gateway Interface User's Guide

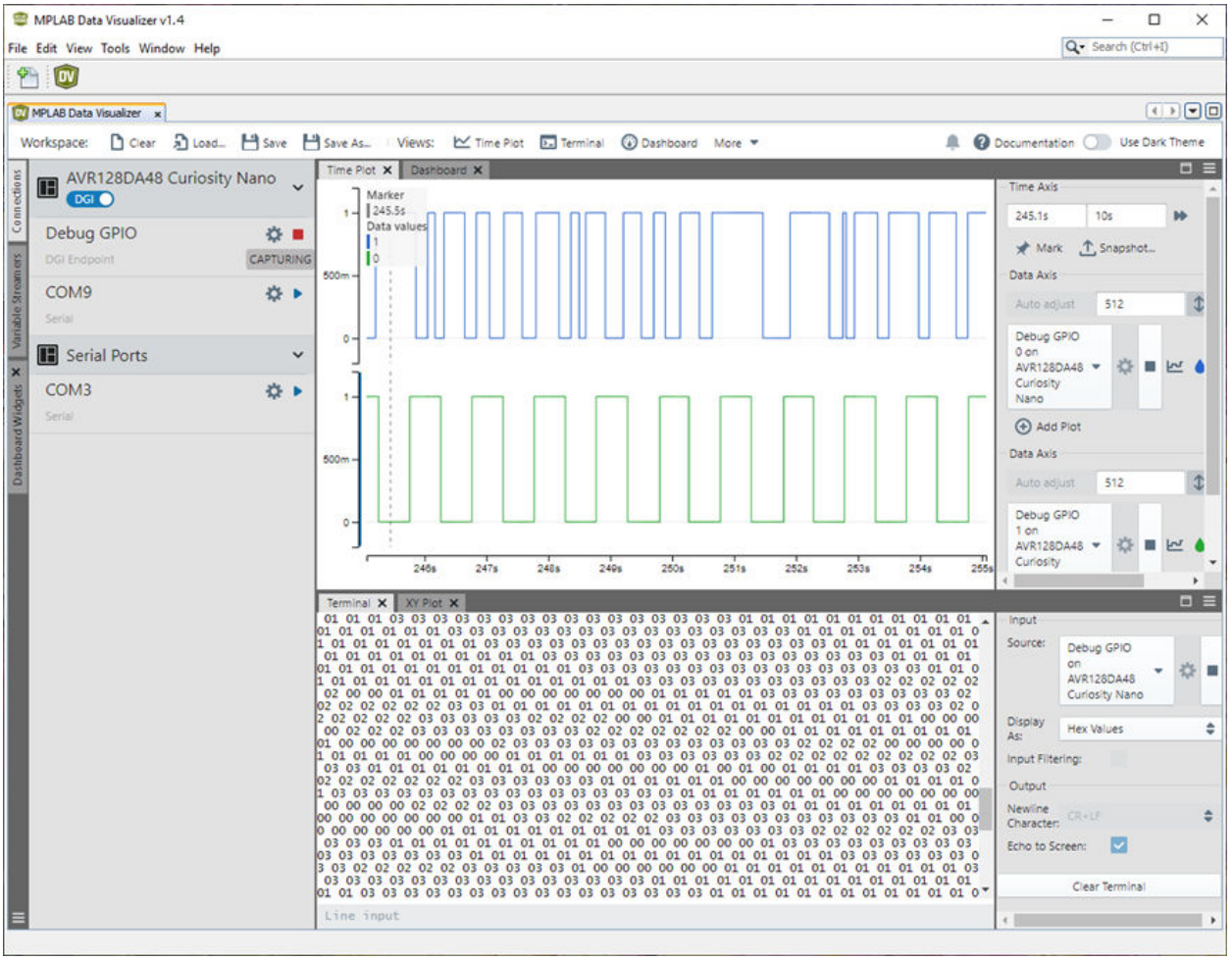
The Data Gateway Interface (DGI) is a USB interface for handling the low-level transport of data to and from a target MCU. The DGI is available on a selection of tools and on-board debuggers, such as the Power Debugger and the EDBG, as found on Xplained Pro boards.

2. Overview

The MPLAB Data Visualizer is a program used to process and visualize data from a running embedded target. The program may be accessed from within MPLAB X IDE or as a standalone program.

Note: The MPLAB Data Visualizer is always being updated to add or improve features. Not all images in this document reflect interface updates, especially if those updates are not critical to understanding the feature being shown in the image.

Figure 2-1. Output of Streaming Data



2.1. Features

It can be difficult to troubleshoot data on an embedded target while your application is running. In the same way a debugger helps you debug your application code, MPLAB Data Visualizer helps you debug your data. With MPLAB Data Visualizer, you can see how key data points in your application change during runtime, e.g., visualize values captured by a sensor on your development board.

MPLAB Data Visualizer has the following features:

- Capture data streamed from a running embedded target via serial port (CDC) or the Data Gateway Interface (DGI).
- Monitor power data produced by DGI compatible kits and tools, as well as the Curiosity Nano Explorer.

- Decode data fields at runtime using the Data Stream Protocol format.
- Stream data, as well as read and write global variables, using the Data Visualizer Run Time (DVRT) protocol.
- Visualize the raw or decoded data in a Graph as a time series or XY plot, or display the data in a terminal.
- Configure the Graph as a rolling time plot or as a simple oscilloscope with edge triggers.
- Actively monitor data or input data to an application using widgets on the Dashboard.
- Concurrently stream data and debug target code using the MPLAB® X IDE.
- Automatically load Data Stream and Dashboard configuration for the data stream.
- Analyze plotted data using cursors to measure bandwidth, pulse width and more.
- Log data to file as it is being captured, using CSV or compressed Feather (Apache Arrow) format.
- Save a plot snapshot to a CSV or JSON file.
- Add Plugins for more functionality.

2.2. Basic Operation

The Data Visualizer operates as an MPLAB X IDE plugin or as a standalone program. As a **plugin**, you can debug your code while using the Data Visualizer functions at the same time. As a **standalone program**, you can't debug your code. However, you CAN debug in MPLAB X IDE while streaming in standalone from the same kit.

Find out how to connect the embedded target to your PC in External Connections.

Learn about different ways to visualize your data, including examples, under Visualization.

For an understanding of how variable values are plotted, see Variable Streamers.

2.2.1. Interface

The MPLAB Data Visualizer interface is comprised of three areas:

1. In the **center** are tabbed windows for data display. The Time Plot is the window used most. A tab may be dragged to move the window. Double clicking on a tab will fill the center area with the window; double clicking again will restore the window to its previous size.
2. On the **left** are panes used to identify and set up displayed data. The Connections tab identifies connected data sources and provides options for display. The Variable Steamers tab displays controls to set up variable values for display. The Dashboard Widgets tab can be used with the center Dashboard tab to control and display parameters from application firmware.
   To adjust the size of the pane, drag the border of the pane to the desired size. Clicking on a tab will collapse/expand the pane.
3. On the **right** are panes to format the data for display. Options available depend on the data being displayed.

   Click on the window icon to fill the center area with the associated window; click again to restore the window to its previous size. Click on the “Toggle Properties” icon to show/hide the pane.

Controls on the interface are discussed in the following sections.

Tip: Information on most controls is located in pop-ups. Just hover the cursor over the control.

2.2.1.1. Toolbar Controls

Visualizer toolbars, on the top of the interface, have the controls listed in the table below.

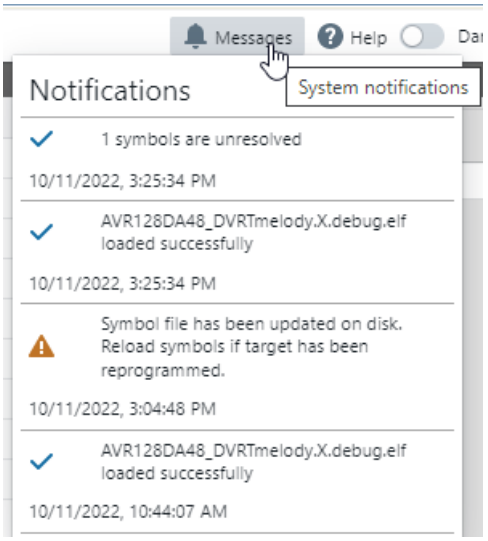
Table 2-1. Toolbar Controls

Operations	More Operations	Description
Workspace: Clear		Clear data and settings in the workspace. All streaming of data will be stopped. The workspace is the state of the data visualizer. So saving and then loading the workspace will restore the data and settings.
Workspace: Load		Load data and settings from a previous session into the visualizer.
Workspace: Save		Save data and settings from your current session into a file.
More Operations		More selections for Workspace.
	Workspace: Save As	Save a Workspace to a file with a different path and/or name.
Views: Time Plot		Select/make active the Time Plot window.
Views: Terminal		Select/make active the Terminal window.
More Operations		More selections for Views.
	Views: Variable Streamers	Select/make active the Variable Streamers tab.
	Views: DVRT Protocol	Select/make active the DVRT Protocol tab.
	Views: DVRT Streaming Table	Select/make active the DVRT Streaming Table window.
	Views: DVRT Read/Write Data Table	Select/make active the DVRT Read/Write Data Table window.
	Views: Dashboard	Select/make active the Dashboard window.
	Views: Dashboard Widgets	Select/make active the Dashboard Widgets tab. Drag to add/remove widgets to/from the Dashboard window.
	Views: XY Plot	Select/make active the XY Plot window.
Protocols: DVRT Session		Select/make active DVRT Protocol tab, Streaming Data Table window and Read/Write Data Table window.
Messages		View notifications generated by visualizer operation.
Help		Show web help for the data visualizer.
Dark Theme		*Enabled*: Workspace background is black with complementary color scheme. *Disabled*: Workspace background is white with complementary color scheme.

2.2.1.2. Notifications and Messages

MPLAB Data Visualizer displays notifications when certain operations complete, either with or without error. These notifications are displayed briefly, so can be missed, If you miss a notification, you can always find a list for the current session under Messages.

Figure 2-2. Notifications listed under Messages



2.2.1.3. Standalone Menus

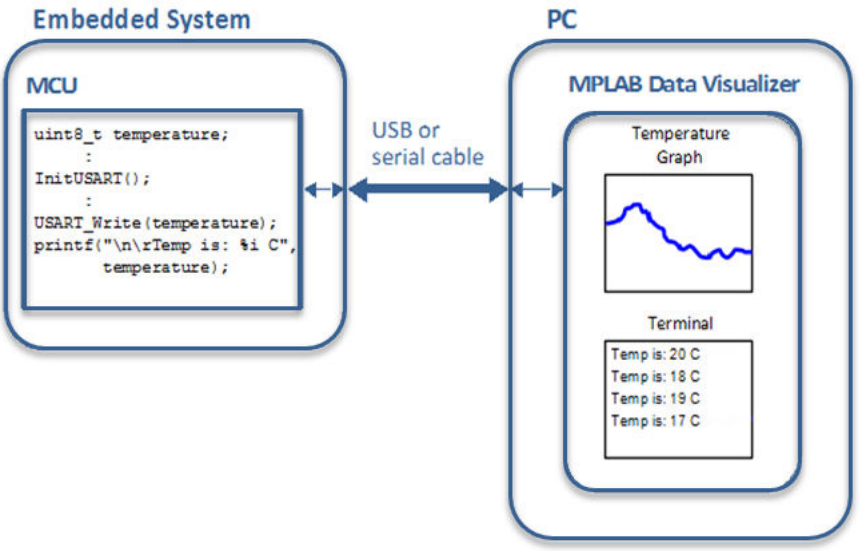
If MPLAB Data Visualizer is used as a standalone application, a menu bar will be available with menus and items of similar function to MPLAB X IDE menu items. Basic text editor functions are included to allow editing of, for example, protocol definition (`.ds`) files.

Menu>Item	Items and Descriptions
File>*Items*	Basic File menu items. Choices are: New File, Open (Recent) File, Exit File.
Edit>*Items*	Basic Edit menu items. Choices are: Undo/Redo, Cut/Copy/Paste, Delete, Find/Replace.
View>*Items*	Basic View menu items. Choices are: Editors, Split, IDE Log, Toolbars, Show Only Editor, Full Screen.
Tools>Plugins	Open the Plugins dialog to add, delete or manage plugins.
Tools>Options	Select data visualizer options: • General - web and proxy options • Keymap - keymapping options • Appearance - interface appearance options • Miscellaneous - File and output font/color options
Window> *Items*	Basic Window menu items. Choices are: Favorites, Output, Editor, IDE Tools (Notifications or Processes), Configure (size, float, dock, split, etc.), Reset, Close, Close all Documents, Close Other Documents, Document Groups, Documents.
Help>*Items*	Basic Help menu items. Choices are: Help Contents, Online Docs and Support, Keyboard Shortcuts Card, Check for Updates, About.

2.2.2. Connections

The MPLAB Data Visualizer captures and displays data coming from a running embedded target. As an example, a temperature value is sent to the visualizer and plotted in a graph or displayed in the terminal.

Figure 2-3. Connection Example

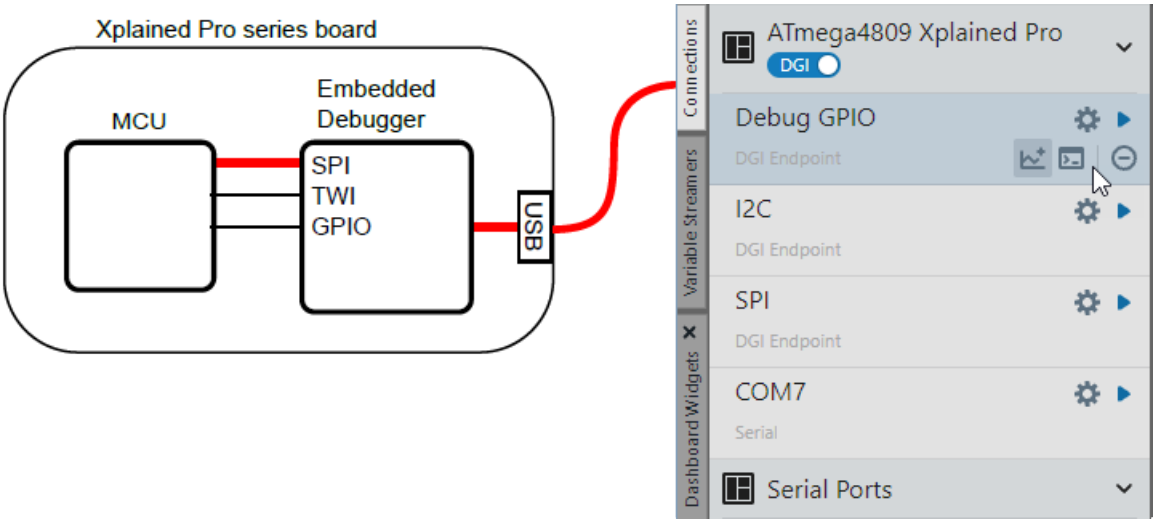


2.2.3. Data Gateway Interface (DGI)

The MPLAB Data Visualizer can communicate with a target board's embedded debugger via the Data Gateway Interface (DGI). When the board is connected to the computer with the USB cable, the visualizer will show that DGI is available for the target and a list of available interfaces (SPI, UART, etc.) will appear below.

In the figure below, the SPI interface is enabled. The MCU can now communicate with the visualizer on its SPI port.

Figure 2-4. Date Gateway Interface



2.3. Installation

The visualizer operates in two ways: as an MPLAB X IDE plugin or as a standalone program.

Support for the data visualizer began with MPLAB X IDE v5.30.

Note: If you are using macOS Catalina and above, please see the Release Notes concerning blocking issues. To locate the release notes, see Finding Release Notes.

2.3.1. MPLAB X IDE Plugin

The MPLAB Data Visualizer plugin is now part of MPLAB X IDE as of v5.50. Please see the release notes about MPLAB version dependencies for specific plugin versions.

There are several ways to access the Data Visualizer in MPLAB X IDE, as shown in the table below.

Table 2-2. Data Visualizer Access in MPLAB X IDE

Method	Description
Data Visualizer icon: 	Click on the icon or select **MPLAB Data Visualizer** from the drop-down menu to open the data visualizer in MPLAB X IDE. Select **DVRT for Project <Project Name>** or **X2CScope for Project <Project Name>** from the drop-down menu to open the data visualizer where the protocol session is synchronized with the project's ELF file. See Automatically Loading the ELF File in the Plugin Data Visualizer.
MPLAB X IDE Menu: **Window** > **Debugging** > **Data Visualizer**	Open the Data Visualizer.
MPLAB X IDE Menu: **Window** > **Debugging** > **Power Monitoring**	Open the Data Visualizer for power monitoring

To see if an updated plugin version is available, go to *Tools>Plugins>Updates* and click **Check for Updates**. For more information on plugins, see the MPLAB X IDE documentation, "**Add Plugin Tools**."

2.3.2. Standalone Application

To find, download and install the visualizer as a standalone program, go to the MPLAB® Data Visualizer web page. Follow the install wizard screens. Once the visualizer is installed, launch it from the install location.

For the standalone application, a menu bar will be available with menu items of similar function to MPLAB X IDE menu items. See Standalone Menus.

3. External Connections

External connections refer to the hardware connections used between the target hardware and the PC. These connections define the type of communication between the target and the MPLAB Data Visualizer. The type of connection depends on the target.

To connect the target to the PC: Follow the instructions for the device or demonstration board.

Once a supported target has been connected to the PC, it should be listed in the Connections tab on the left side. The supported types of communication are:

- DGI Tools - tools or kits that support the Data Gateway Interface. It is capable of input streaming communication over SPI, I2C, USART and Debug GPIO. The feature set varies by tool.
- Serial/CDC Connections - selection specifies communication with any serial port on the system that can be set up using baud rate, parity, data bits and stop bits.
- Curiosity Nano Explorer - supports streaming power data from the embedded Power Monitor.

3.1. Connection Options

When connections are displayed on the Connection tab, the following options are available for each source.

Table 3-1. Common Options for Data Sources

Icon	Option	Description
	Source options	Select options/settings for the chosen source.
	Start Capturing Stop Capturing.	Start/Stop streaming the raw data from the source connection. For Debug DGIO, all pin values will be streamed. To plot selected pin values, use the right pane.
	Add to new plot Add to selected plot.	Create a new plot for raw data. Add raw data to selected plot. See View Data in the Time Plot.
	Send to Terminal.	Send streaming data to terminal window. See View Data as Text in the Terminal.
	Remove one or more selected visualizations.	Remove the current plot, terminal display and/or variable streamer.
	New variable streamer.	Configure and plot variable values. See Variable Streamers.
	Configure logging to file.	See Log and Save Data To a File.

Source options can only be edited when the source is not capturing. Editable numeric fields can be assigned to any number including scientific notation with the letter e. Other symbols cannot be entered. The value can be incremented/decremented with the arrow up/down key. A numeric field might enforce a min-max range or for the value to be integer.

3.2. Data Gateway Interface (DGI)

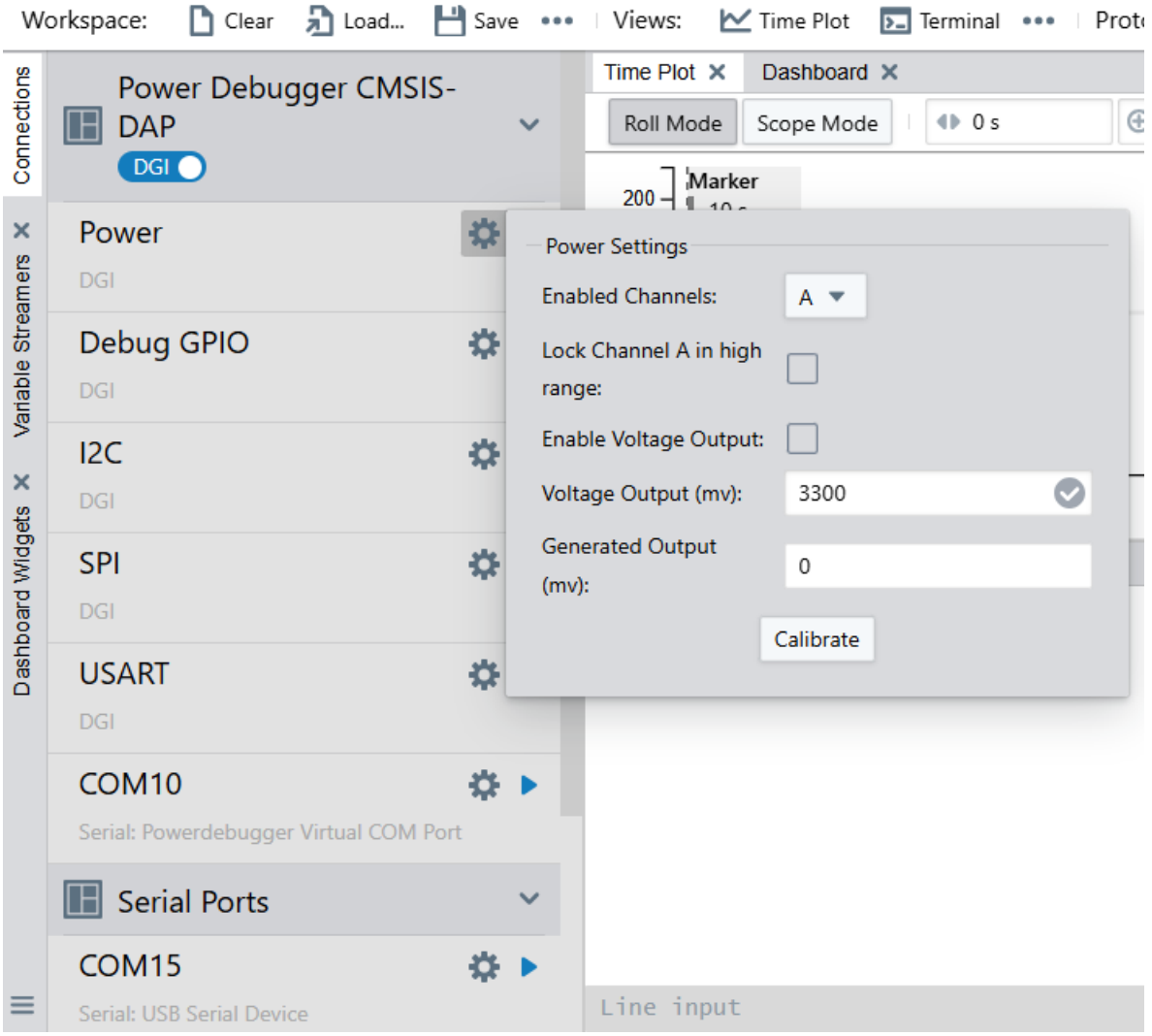
The Data Gateway Interface is available on most kits with an embedded debugger. The visualizer DGI controls can communicate with a DGI device. Set up DGI controls in the Data Sources (left) pane.

All detected DGI devices are listed on the left pane under **Connections**. The visualizer accepts streaming input from a DGI-capable board.

To set up interface options, click on the gear icon on the interface pane (such as Power). Enter settings and click the icon again to save settings and close.

To enable an interface, see Control Data Capture and Visualizations.

Figure 3-1. Data Gateway Interface



3.2.1. Power Interface

The Power interface measures the power consumption of the connected circuitry. For more information on the hardware part of the power interface, see the user guide of the debugging tool to be used for the power measurements.

Select the Power interface beneath the debug tool DGI. Set up the interface using the controls under “Power Settings.” Available controls will vary depending on the capabilities of the connected debugging tool.

Table 3-2. Power Settings

Control	Value	Usage
Enabled Channels	A, A+B	Either enable channel A only or both channels A and B. Channel A is always enabled.

Table 3-2. Power Settings (continued)

Control	Value	Usage
Lock Channel A in high range	Unchecked, checked	On the Power Debugger, channel A can be locked to the high range to avoid automatic switching to the low range. This allows detection of short spikes in current consumption without critical samples being lost when switching between the ranges.
Enable Voltage Output	Unchecked, checked	The Power Debugger features an adjustable target supply that can be used to power the target application. Disabling this setting sets the output voltage of this supply to 0. Enabling it sets the supply to the voltage level of the "Voltage Output" setting.
Voltage Output (mV)	Between 1600 mV and 5500 mV	When the "Enable Voltage Output" setting is enabled, the value of this setting is the target supply voltage of the Power Debugger. Changes made to this setting while Voltage Output is enabled are committed when clicking the yellow checkmark button.
Generated Output (mV)	This setting is read-only	This is a read-only setting that shows the voltage level of the target supply voltage of the Power Debugger. The value is controlled by the "Enable Voltage Output" and "Voltage Output (mV)" settings.
Calibrate	Click to activate	Triggers the calibration procedure of the current measurement circuitry. To achieve full measurement accuracy on the current measurement hardware, it should be calibrated before running any measurements.

Tip: The channel A range lock will not force the debugger to return to the high current range if already running in the low range. Either wait for a current high enough to force it to change, or simply Stop and Start the debugger.

Tip: Each power signal time plot uses system resources. Reduce the number of concurrent plots for better performance.

3.2.2. GPIO Interface

The GPIO interface contains the bit values of the enabled Debug GPIO pins. A packet of unsigned 8-bit data is transmitted every time a pin toggles. For further details on the physical part of the GPIO interface, see the user guide of the debugging tool to be used to sample the GPIO data.

On the Data Sources (left) pane, when the GPIO interface is selected, the GPIO settings are displayed on the lower section.

Table 3-3. GPIO Settings

Field Name	Values	Usage
GPIO 0 Change Triggers Bus Read	ON, OFF	Monitor change on GPIO pin 0 to trigger a bus read
GPIO 1 Change Triggers Bus Read	ON, OFF	Monitor change on GPIO pin 1 to trigger a bus read
GPIO 2 Change Triggers Bus Read	ON, OFF	Monitor change on GPIO pin 2 to trigger a bus read
GPIO 3 Change Triggers Bus Read	ON, OFF	Monitor change on GPIO pin 3 to trigger a bus read

Important: When plotting the Debug GPIO data source, all GPIOs are sampled but only those GPIOs that have change triggers enabled will trigger a sample on change. For example, if GPIO n (n = 0,1,2) all have "GPIO n Change Triggers Bus Read" disabled, but GPIO 3 has this function enabled, then GPIO values will only be sampled when GPIO 3 changes; that is, all four GPIO values will be read only when GPIO 3 changes.

3.2.3. USART Interface

The USART source streams the raw values received on the USART interface. For further details on the physical part of the USART interface, see the user guide of the debug tool to be used to sample the USART data.

On the Data Sources (left) pane, when the USART source is selected, the USART settings are displayed on the lower section.

Note: Asynchronous serial protocols (e.g., UART protocols used by DGI USART and CDC Virtual COM port interfaces) use the baud rates listed in Serial Port.

Table 3-4. USART Settings

Field Name	Values	Usage
Baud Rate	0-2000000	Baud rate for UART interface in Asynchronous mode.
Char Length	5, 6, 7, or 8 bits	Number of bits in each transfer.
Parity	None, Even, Odd, Mark, or Space	Parity type used for communication.
Stop bits	1, 1.5, or 2 bits	Number of Stop bits.

3.2.4. I2C Interface

The I2C source streams the raw values received on the I2C interface. For further details on the physical part of the I2C interface, see the user guide of the debug tool to be used to sample the I2C data.

The I2C Configuration options are displayed under the **DGI** section of the Data Sources (left) pane. When an I2C connection is selected, the I2C settings are displayed in the lower section of this pane.

Table 3-5. I2C Settings

Field Name	Values	Usage
Speed	0	The expected operation speed of the interface in Hertz helps the client device adjust the timings. Up to 400 kHz is supported.
Address	1	Address of the client device.
Kit-side Timestamping	Check to enable	Target timestamping.

3.2.5. SPI Interface

The SPI source streams the raw values received on the SPI interface. For further details on the physical part of the SPI interface, see the user guide of the debug tool to be used to sample the SPI data.

Important: The SPI hardware module uses an active-low Chip Select (CS) signal. Any data sent when the CS pin is high will be ignored.

The SPI interface is under the **DGI** section of the Data Sources (left) pane. When an SPI connection is selected, the SPI settings are displayed in the lower section of this pane.

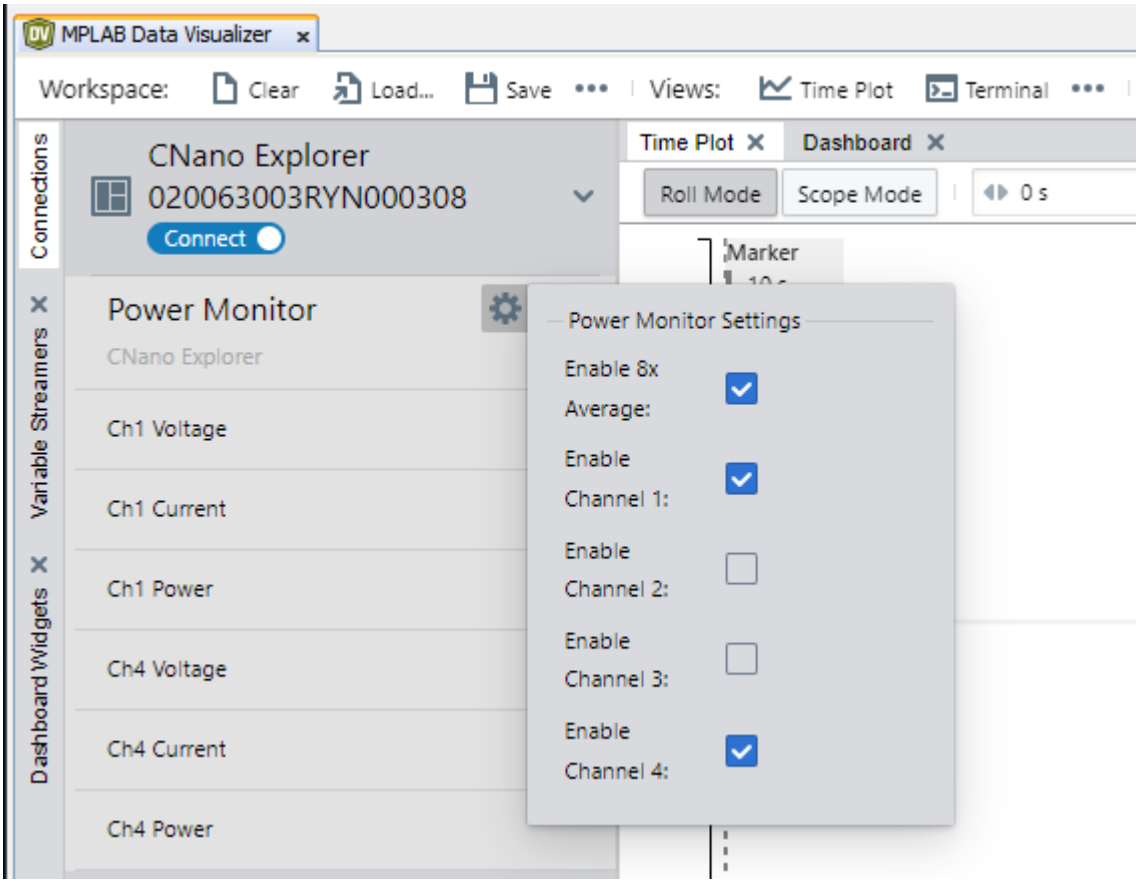
Table 3-6. SPI Settings

Field Name	Values	Usage
Char Length	5, 6, 7, or 8 bits	Number of bits in each transfer.
Mode	• Clock idle normally low, Sample data on rising edge. • Clock idle normally low, Sample data on falling edge. • Clock idle normally high, Sample data on falling edge. • Clock idle normally high, Sample data on rising edge.	SPI mode, controlling clock phase and sampling.
Force CS Sync	Check to enable.	The SPI interface is only enabled after the Chip Select line has toggled twice.
Kit-side Timestamping	Check to enable.	Target timestamping.

3.3. Curiosity Nano Explorer

The Curiosity Nano Explorer allows users to explore and experiment with the microcontroller peripherals of their Curiosity Nano development board. The board supports basic Power Monitoring by an embedded PAC1944 device. The board has four power domains, all of which can be visualized in the Data Visualizer. When connected to the PC, the board is listed in the left pane under Connections. The Power Monitor data source section allows selecting active channels, and lists the data points sampled for each channel.

Figure 3-2. Curiosity Nano Explorer



Power Monitor

The PAC1944 power monitor on the Curiosity Nano Explorer enables power measurement on three of the board's power domains.

- Channels 1 and 4 monitor the CNANO P3V3 domain, providing:
  - Channel 1: High-accuracy, low-range measurements.
  - Channel 4: Lower-accuracy, higher-range measurements.
- Channel 2 monitors the peripherals P3V3 domain.
- Channel 3 monitors the peripherals P5V0 domain.

For detailed specifications including current sense range and resolution, refer to the Power Monitor section in the Curiosity Nano Explorer User Guide.

Important: If a kit mounted on the Explorer board is configured to be a host (master) on the I²C bus, ensure the I²C slide switch S500 is set to **NC** to isolate the USB bridge and power monitor from the main I²C bus.

Table 3-7. Power Monitor Settings

Field Name	Values	Usage
Enable 8x Average	Unchecked, checked	Use a rolling 8x average on the data from the Power Monitor. Applies uniformly to current, voltage and power, for all 4 channels. The average is performed on the Power Monitor hardware, which has a sample rate independent of the rate of the data polled and displayed in the Data Visualizer.
Enable Channel N	Unchecked, checked	Enable channel number N on the Power Monitor. Disabled channels will not be read from the hardware and will reduce consumed resources. At least one channel must be enabled.

3.4. Serial Port

The Data Visualizer can be connected to a target board via a standard PC serial port. Set up serial controls in the Data Sources (left) pane.

Baud rate, Stop bits and parity must be set to match the required settings for the communication partner. Serial port data is treated as unsigned 8-bit data in the Graph and Terminal.

Table 3-8. Configuration

Field name	Values	Usage
Baud rate	9600-2000000	Baud rate of serial interface.
Char Length	5, 6, 7, or 8 bits	Number of bits in each transfer.
Parity	None, Even, Odd, Mark, or Space	Parity type used for communication.
Stop bits	1, 1.5, or 2 bits	Number of Stop bits.
DTR	Checkbox	Assert DTR signal while connected.
RTS	Checkbox	Assert RTS signal while connected.

Notes: Asynchronous serial protocols (e.g., UART protocols used by DGI USART and CDC Virtual COM port interfaces) use the following **baud rates**:

- 9600
- 19200
- 38400
- 57600
- 115200
- 230400
- 500000
- 1000000
- 2000000

Using any other baud rates will not work for protocols over asynchronous interfaces (DGI UART and Serial port/CDC Virtual COM port).

Difference between tty and cu ports

See the list article by Godfrey van der Linden posted on the Apple® Listserv at lists.apple.com/archives/list/darwin-dev@lists.apple.com/message/PN3ZLI6MB7FM7WM7I67OCS4D4BFPOK3S/.

4. Visualization

Incoming data can be visualized using the modules contained under this section.

- The **Terminal** displays data as text, either as raw values or ASCII encoded characters. It is also capable of sending text-based data.
- The **Time Plot** visualizes incoming data over time as plotted waveforms. The Cursor helps analyze the data, and can provide output values for setting thresholds.
- The **Power Analysis** function of the time plot is made specifically for analyzing power consumption over time. It can also be used with code profiling to visualize Program Counter samples to get an overview of the program execution versus power consumption.
- The **XY Plot** visualizes incoming x vs y data as a plot.
- The **Dashboard** displays or can input data using widgets.

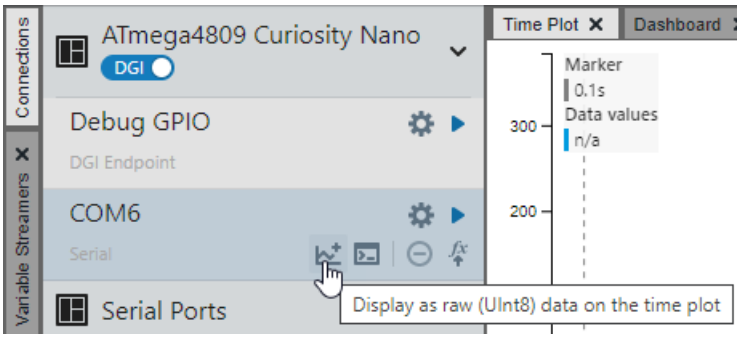
4.1. Data Capture and Visualization

When a target is connected to your PC, the data visualizer will display available data sources. To begin streaming data from these sources, choose from the following methods.

4.1.1. Select a Visualization Method First

Hover over a source to see available visualization options, such as plot raw data , display text in the terminal , or create a variable streamer . Once selected, the data will begin streaming using the option selected. You can select one or more visualizations.

Figure 4-1. Select a Visualization Method

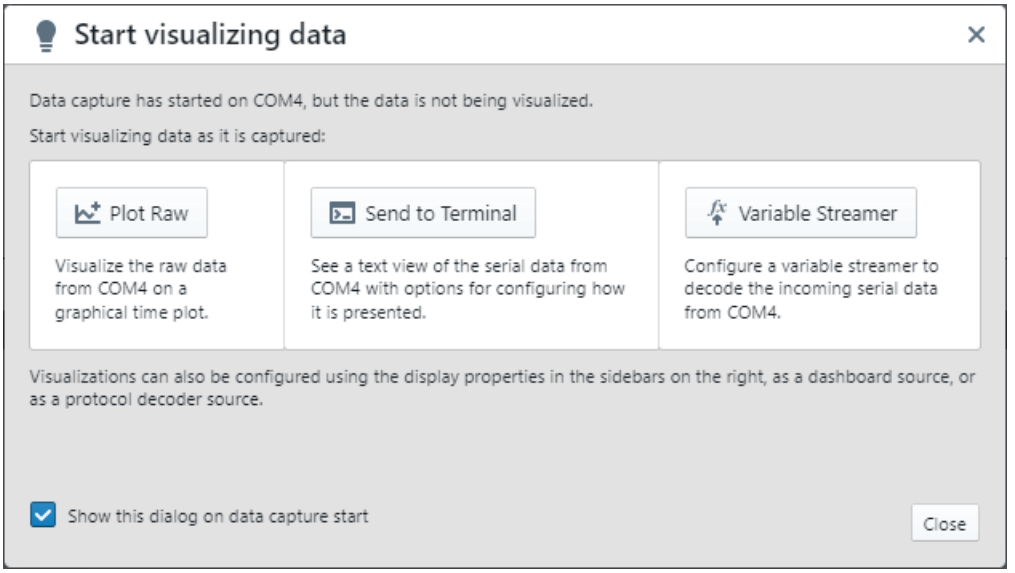


The source can also be dragged and dropped onto the Time Plot as described in Connections and Protocols, on the Terminal as described in Connections Tab or on the Dashboard as described in Placing Widgets on the Dashboard and Sending Values to and from Widgets.

4.1.2. Begin Capturing Data First

If a data capture session is started without selecting a visualization (by clicking the Start Capturing button) the quick action dialog will pop up to direct you to available visualizations (see figure below). Click on one or more of the quick action buttons, e.g., “Plot Raw” to begin visualization on the Time Plot.

Figure 4-2. Available Visualizations

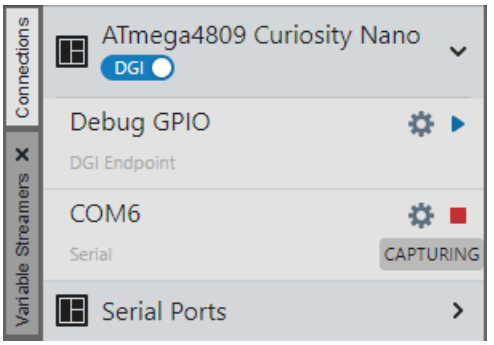


Note: If you no longer want this dialog to appear, you can unselect "Show this dialog on data capture start." If you later wish to see it again, go to *Tools>Options>Embedded>MPLAB Data Visualizer* to enable it.

4.1.3. Control Data Capture and Visualizations

When you stop hovering over the source, the visualization method controls are hidden and the message CAPTURING is displayed (see figure below).

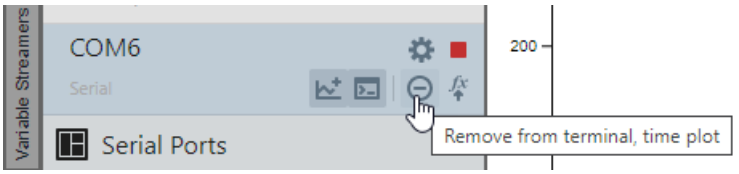
Figure 4-3. Data Capture In Progress



Click Stop Capturing to halt data flow but keep visualization settings. Click Start Capturing to resume data flow.

To remove all data capture and visualization methods, hover over the source and click the Remove button (see figure below).

Figure 4-4. Remove Visualizations and Data Capture



Alternately, you can remove each data capture and visualizations individually by clicking on a previously-selected button.

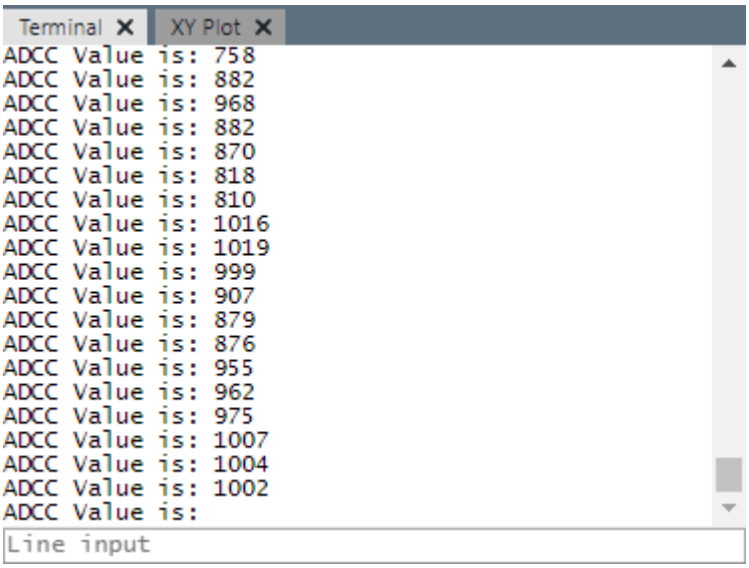
4.2. View Data as Text in the Terminal

The Terminal is a raw terminal for displaying and sending simple text or numeric values.

4.2.1. Terminal Window

The Terminal window of the MPLAB Data Visualizer shows streaming data from the target in different formats.

Figure 4-5. Terminal Window - Application Streaming Data

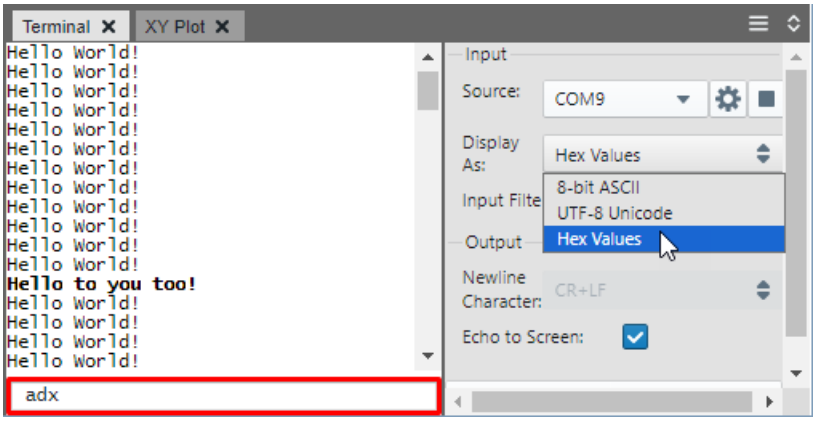


The Terminal can also stream characters and lines of text to a target when connected through a COM port by typing either in the terminal area or the text box (Line input), as shown in the figure below. Terminal echo (typed content) is bolded. If an incorrect character is typed in Line input, the box will be outlined in red.

The history of Line input can be viewed by using the up or down arrow keyboard keys. Press the up arrow one time to replace any current content with the most recently-sent content. Press the up (or down) arrow repeatedly to cycle through the history of content backwards (or forwards). Selected content can be resent by pressing the Enter key.

History is only kept for the duration of the current terminal session. Up to 20 entries will be saved; after this, the oldest entry is replaced by a new one. If the same content is sent two times in a row, a new entry will not be added to the history.

Figure 4-6. Terminal Window - Terminal Input to Target



Terminal Setup

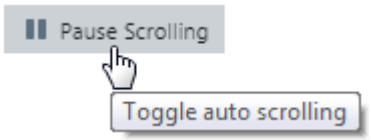
Use the Connections tab on the left pane to choose the data source. Verify that the baud rate is correct.

Use the Terminal Visualization Controls (right) pane to select one or more formats from the drop-down list.

When enabled, the terminal is active continuously, either scrolling data or accepting input.

Terminal Display

You can toggle the data streaming view by hovering over the top of the Terminal window and clicking “Pause Scrolling/Scroll to End.” Although the window view is paused, data continues to stream in the background.

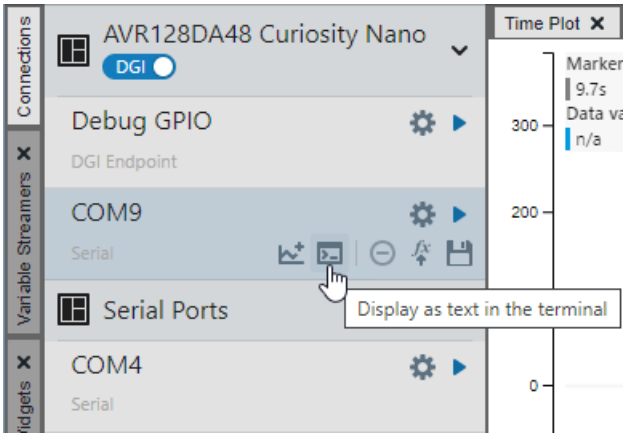


When data is paused, you may use <Ctrl>+<C> and <Ctrl>+<V> (<Cmd>+<C> and <Cmd>+<V> for macOS) for copying/pasting text in the terminal.

4.2.2. Connections Tab

Click the **Connections** tab to the left of the window to see available connections. After you connect a target to the PC, that connection will be displayed and the on-board data sources will be listed.

Figure 4-7. Available Connections



Find a data source that allows sending data to the terminal, i.e. has a **Display as text in the terminal** button . Select the **Source options** button to view or change source settings, then select , or drag the source and drop it onto the Terminal window. If you are connected and wish to change the settings, you can disconnect by clicking now named **Remove from the terminal**. Make your changes and then click the button again to reconnect.

For connection details, see External Connections.

4.2.3. Terminal Visualization Controls (Right) Pane

In the Terminal area, the Terminal Visualization Controls (right) pane is for selecting the source and controlling the format of streaming data.

Table 4-1. Input Data

Control	Description
Source	All data sources, apart from Debug GPIO, are supported.
Display As	Select how the data stream is translated to terminal characters. Current selections are: • ASCII • UTF-8 • Hex values • Numbers No new line characters are output in Hex values mode. Try Numbers mode for more control.
Input Filtering	Click the checkbox to filter out ANSI/VT100 terminal control characters 1B, 90, 98, 9B, 9D, 9E and 9F from the input stream, as these have special meaning to the embedded terminal component.
Items Per Line	For the Numbers display option, specifies how many items to output before a line break is inserted. If the field is empty, no line breaks are inserted.
Formatting	For the Numbers display option, specifies the item formatting, e.g. "%10.2f". The format specification is the same as for the Dashboard's Label widget, see Label.

The controls in the output section are only enabled for a COM port connection.

Table 4-2. Output Data

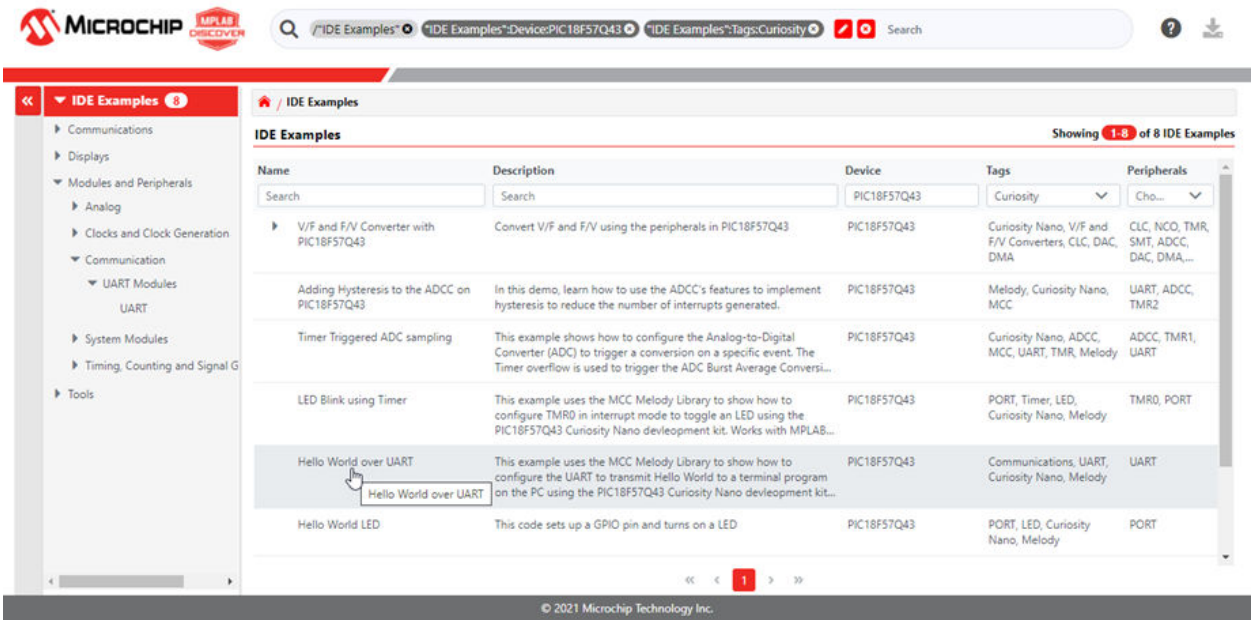
Control	Description
Newline Character	Select which character(s) will represent a newline in the output stream. • None • CR+LF: Carriage return + line feed • LF: Line feed
Echo to Screen	Check to echo typed characters to the screen.

To delete the content of the terminal window, click **Clear Terminal**.

4.2.4. Terminal Example

Find example code using MPLAB Discover. On the main page, filter for Device = PIC18F57Q43 and Tags = Curiosity. Find the example “Hello World over UART.”

Figure 4-8. MPLAB Discover - Search



Click on the example to open a page with information on what hardware and software was used, how MCC Melody was used to develop the code, and how to view output in the data visualizer. Example code can also be downloaded from this page.

Figure 4-9. MPLAB Discover - Example Page

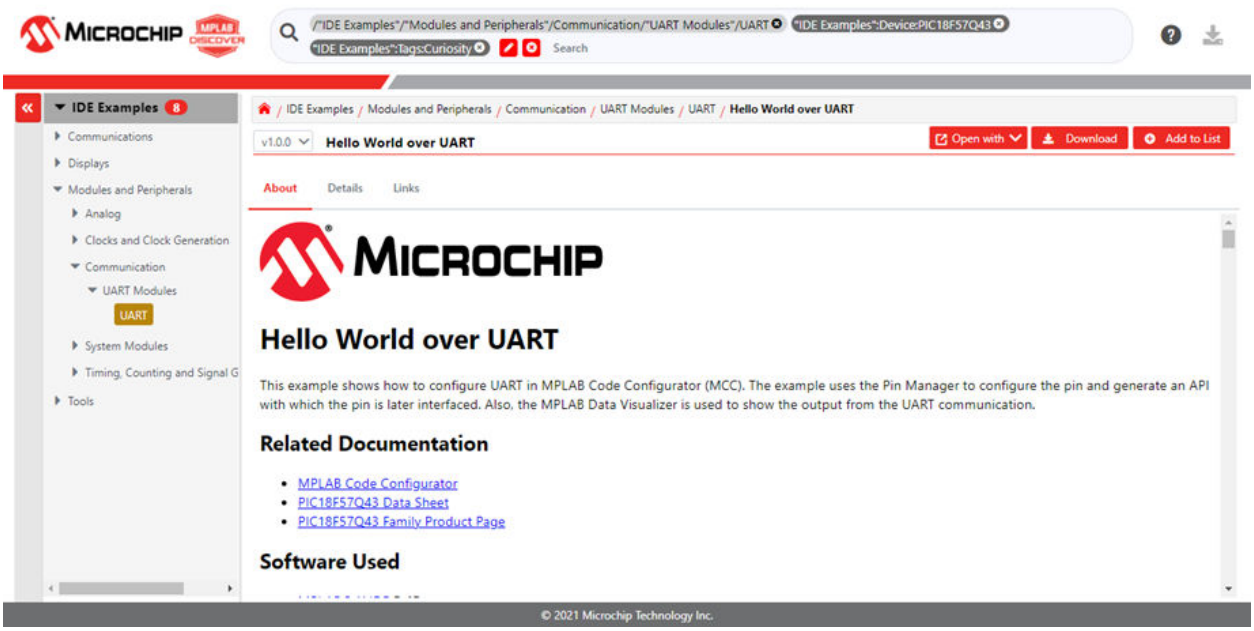
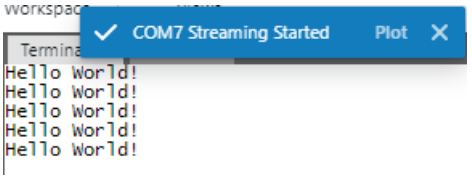


Figure 4-10. Terminal Output



4.3. View Data in the Time Plot

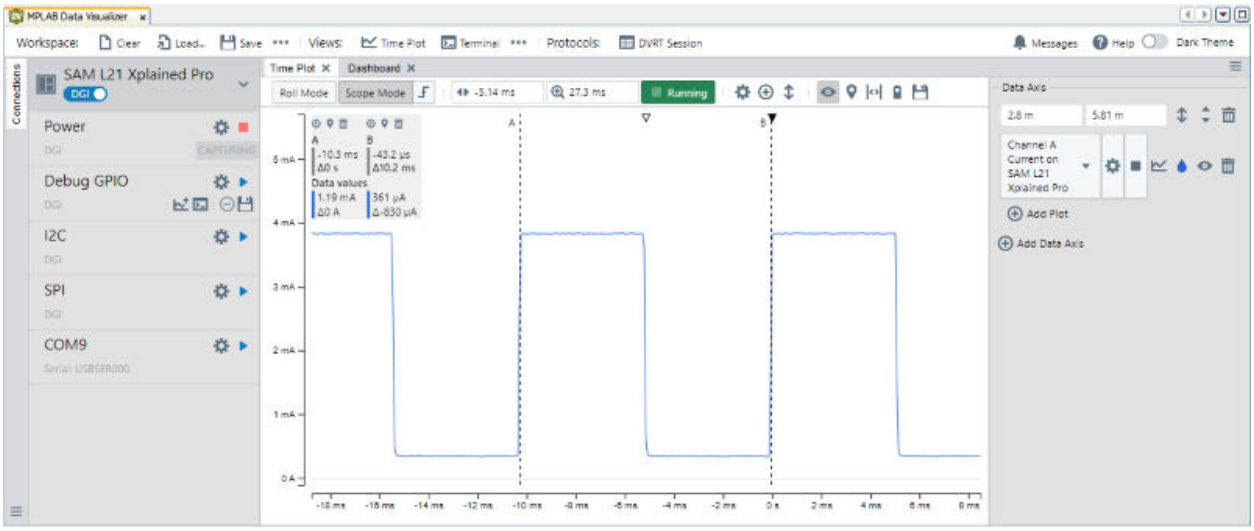
The Time Plot is a versatile graph plotting tool. The large plot area has one time axis, and one or more value axes (Y axes). The value axes may be stacked on top of each other or split into multiple plots. Before anything can be plotted, a data source must be configured. There are multiple ways to configure the data sources:

- From within the plot itself
- From the **Connections** tab
- From the **Variable Streamers** tab

4.3.1. Time Plot Window

The Time Plot window shows the data plot(s) vs. time and provides tools for data analysis. Most functions of the time plot are controlled using the toolbar at the top of the time plot panel and the axis and plot configuration controls at the right side of the time plot.

Figure 4-11. Time Plot



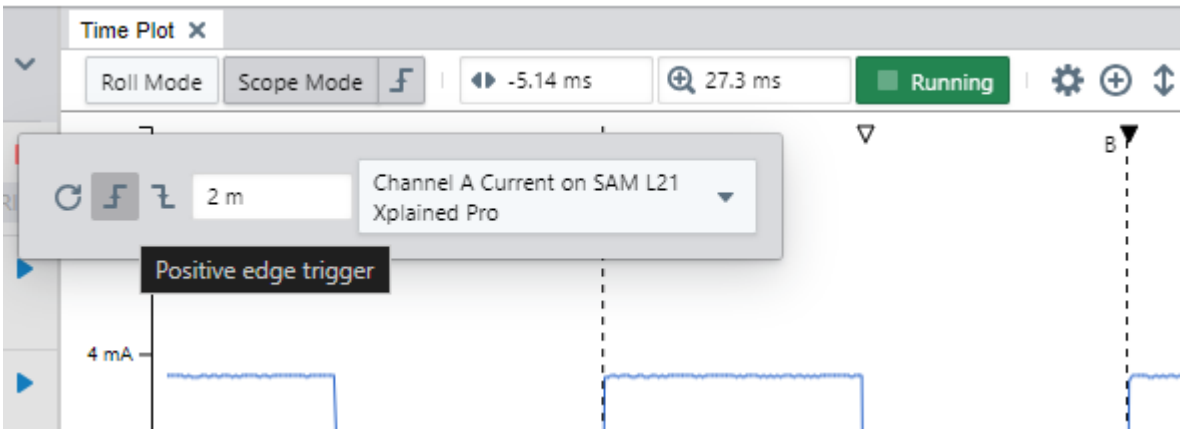
4.3.1.1. Time Plot Mode

The time plot has two major modes of operation:

- Roll Mode: In this mode, the plotted waveform scrolls from right to left on the screen.
- Scope Mode: In this mode, the waveform is redrawn periodically or when a trigger condition is encountered.

These modes are selected using the two leftmost buttons on the timeplot toolbar:

Figure 4-12. Time Plot Toolbar with Trigger Options Open



Note: When Scope Mode is active, an additional button appears for setting trigger options. In Roll Mode, the time plot behaves like the classic time plot in MPLAB Data Visualizer prior to version 1.4.

Configuring Triggers

The trigger function of the Scope Mode is configured by clicking the **Trigger Settings** button to the right of the **Scope Mode** button. The icon on this button shows the active trigger mode at any time.

The trigger settings are:

- Trigger mode:
  - No trigger (periodical updates)
  - Positive edge trigger

  - Negative edge trigger
- Trigger threshold: Data value threshold.
- Trigger source: The data source to monitor.

When **No Trigger** is selected, the time plot is not updating on a trigger condition. Instead it is repeatedly updating at a rate that corresponds to the current Time Scale of the time plot. For example, if the time scale is 0.1 second, the update rate will be around 0.1 seconds. This is a relatively easy setting to start with.

When **Positive** or **Negative Edge Trigger** is selected, the Data Visualizer will start an update cycle with capturing enough data to fill the area to the left of the trigger point. It will then start looking for the trigger condition. When the trigger condition is detected, the Data Visualizer will continue to capture enough data to fill the area to the right of the trigger point, and after this the plot is updated.

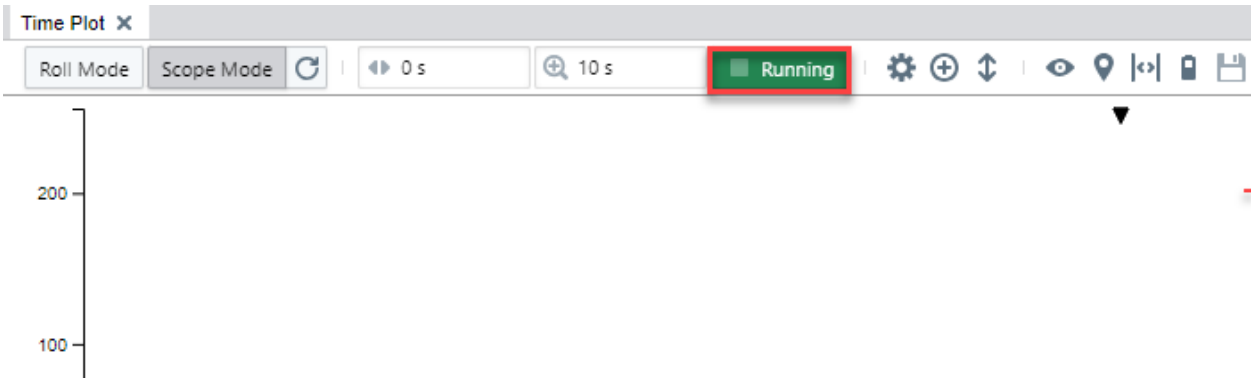
If the Time Scale is larger than 1 second, the part of the plot that is after the trigger point will be plotted live as data is being captured. Combined with setting the Time Reference to the left side, this is similar to an EKG plot.

If the time plot does not update as expected when a trigger condition has been configured, check the configured trigger settings first. Switch to the simpler Roll Mode temporarily to check that data is being captured and plotted in this mode. Also, check that the plot is in Run Mode, that the plots have been configured properly, and that the plotted data sources and the trigger source are capturing data.

4.3.1.2. Running and Stopped Mode

The run mode button controls whether the time plot is updating or not.

Figure 4-13. Run Mode



Stopped Mode: The time plot content does not update. In this state, it is possible to pan and zoom in the time plot to investigate the plotted data in detail.

Running Mode: In this mode, the plot is updated as data is captured. The behavior depends on the selected Time Mode. In Roll Mode, the plot is scrolling from the left towards the right. In Scope Mode, the plot is redrawn periodically or on a trigger condition.
Note: In stopped mode, captured data might not be visible even though streaming is enabled. To ensure that the most recent data is being displayed live, set the run mode to **Running** and the time mode to **Roll**.

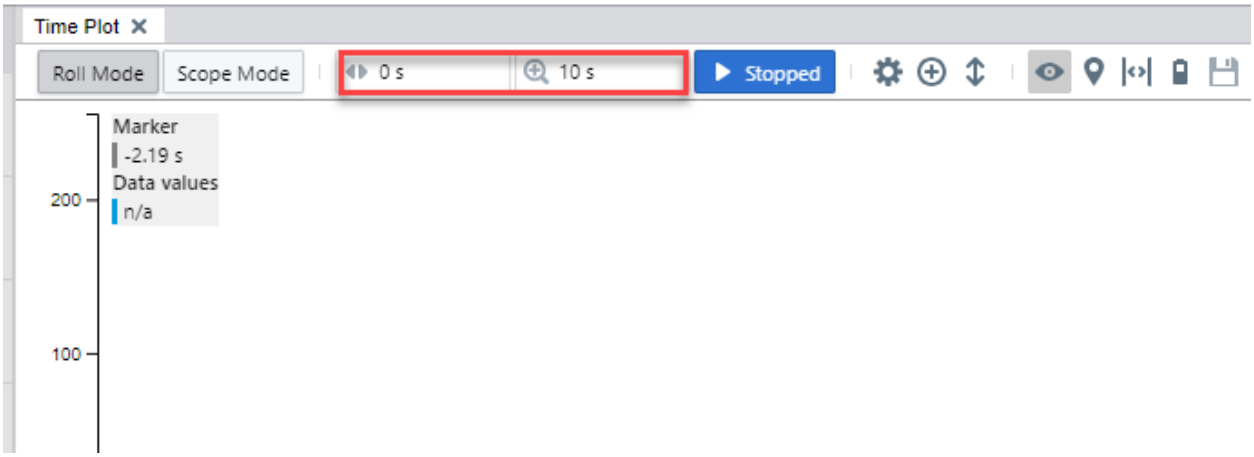
4.3.1.3. Panning and Zooming

In the time plot, Time Offset refers to the amount that the plot has been panned to the left or right. Time Scale refers to the zoom level, or the time spanned by the visible time axis.

Two numeric inputs in the time plot toolbar control the time offset and time scale. The inputs accept numbers with a metric suffix such as “5 ms”. When edited, the new value will apply after pressing enter or clicking somewhere else with the mouse.

The default time offset is 0, and the default time scale is 10 seconds. The buttons inside the input controls allow resetting the time offset to 0, as well as increasing and decreasing the time scale.
Note: Before MPLAB Data Visualizer 1.4, the time offset and scale settings were found in the time axis controls on the panel to the right side of the time plot.

Figure 4-14. Time Offset and Scale

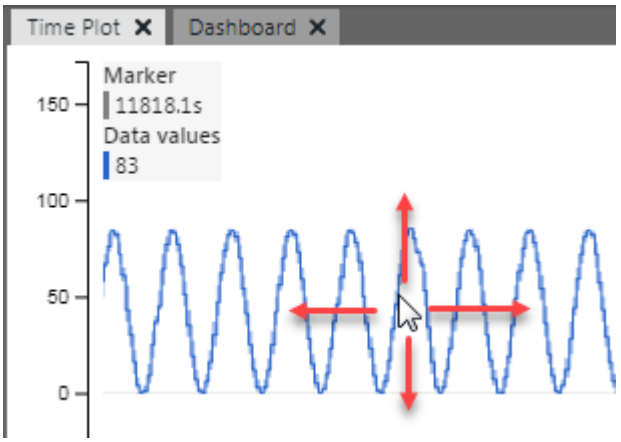


To manually pan the plot, click on it and drag:

- Click on the data axis on the left side of the plot to pan the plot vertically.
- Click on the plot area or on the time axis to pan the time axis.

Note: It is also possible to pan in both directions from the plot area by holding the Shift key when dragging.

Figure 4-15. Plot Pan and Zoom

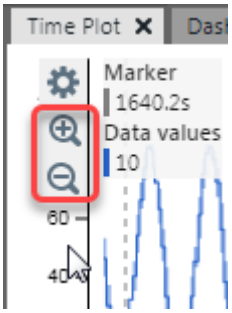


To manually zoom in the plot, use these controls:

- Use the mouse wheel to zoom in and out on the axis. The plot will resize accordingly.
- Use the mouse wheel to zoom in and out in the graph area.
- Click on an axis to see options to zoom in and out.

Note: It is also possible to zoom vertically from the plot area by holding the **Shift** key while using the mouse wheel.

Figure 4-16. Axis Zoom



Above the plot area are two triangular markers labeled as **trigger point** and **reference point**. When the time offset is 0, the two markers are rendered on top of each other. When the time plot is panned manually to either side, the trigger point will move with the waveform while the reference point stays put. The position of the reference point can be configured in the **Time Plot Options** dialog. In run mode, zoom operations are around the reference point.

4.3.1.4. Add Data Axes

To add another data axis,click on the **Add Data Axis** icon  on the time plot toolbar.

Data axis controls are displayed in the right side panel. Alternatively, select the axis and click on **Axis Options**. On the axis controls is another selection to add a data axis. Both methods provide a control to add a plot to the axis.

Figure 4-17. Data Axis Options in the Right Side Panel

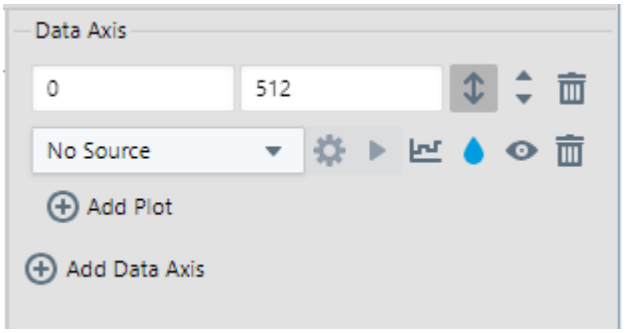
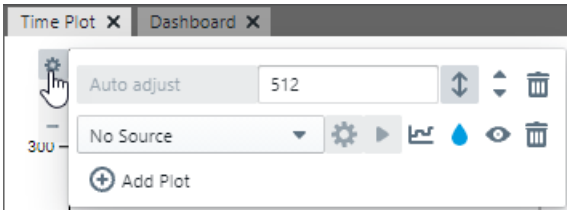


Figure 4-18. Axis Options on the Data Axis



In addition, once the axis is set up, data can be plotted from the Data Sources pane by clicking the **plot** button on the selected data source.

Figure 4-19. Data Sources - Plot

If several data axes are present, their vertical ordering can be adjusted with the arrow buttons on the Axis Options. The data axes can also be moved by hovering on an axis and using the ellipsis handle to drag the axis and drop it on the desired plot area. An axis dropped on the upper or lower part of a plot area will be placed on a new plot above or below the current one.

4.3.1.5. Configure Data Axes and Plots

The axis and plot settings in the right pane control which data sources are plotted on which axes, as well as the appearance of the plots.

Figure 4-20. Graph Visualization Controls

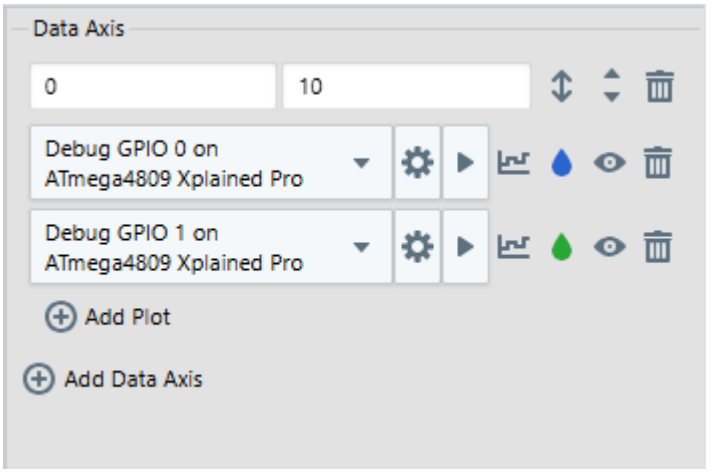


Table 4-3. Data Axis

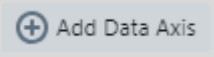
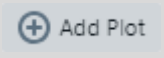
Control	Description
Offset	Displays the vertical offset of the plot relative to the center of the data axis.
Scale	Displays the value range of the data axis.
	Auto adjust enable/disable. When enabled, automatically adjust range of axis. When disabled, manually adjust range of axis.
	If more than one data axes, move this axis up or down relative to others.
	Click to delete this axis from the graph.
	Add another data axis to the graph under current axes.

Table 4-4. Data Axis - Plot Source and Format

Control	Description
Data Source	Select the data source to plot from the drop down list. See the Data Sources pane for selection and setup.

Table 4-4. Data Axis - Plot Source and Format (continued)

Control	Description
	Click to select how data points are shown on the graph. • Connect points with stepped lines (default). • Connect points with straight lines. • Only draw points, not lines.
	Click to select a graph color.
	Show/hide graph plot.
	Click to delete this plot.
	Add another data source to plot on the current axis.

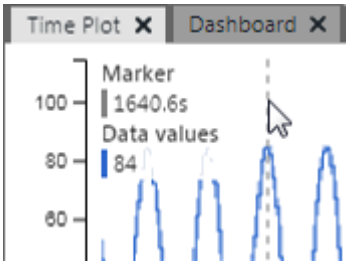
4.3.1.6. View Plot Values

MPLAB Data Visualizer has several tools for viewing and analyzing plot data.

4.3.1.6.1. Plot Marker

The time plot marker is a dashed vertical line that follows mouse movements along the time axis. Time offset and data values for the current location of the marker are displayed in the top left corner of the graph. To toggle this display, click on the top of the graph and click on the **Toggle Inspect Values** icon  .

Figure 4-21. Marker and Inspect Values Display



4.3.1.6.2. Plot Cursors

A vertical cursor can be used in a similar manner to the plot marker, except that the cursor does not follow mouse movements; it must be dragged to a position and it will not move until dragged again. To add one or more cursors to the display, click on the top of the graph and click on the **Add vertical cursor** icon  . “Inspect Values” will then display cursor values in place of marker values (see figure).

Figure 4-22. Vertical Cursor A

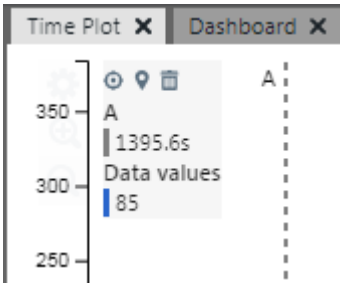


Table 4-5. Inspect Values Display

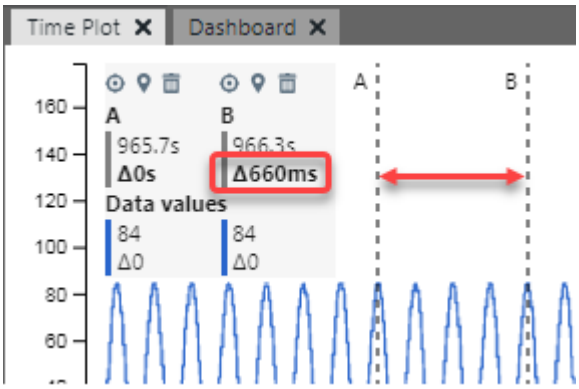
	Jump to cursor. The cursor will be centered on the graph and scrolling is paused. If the cursor has scrolled away with the plot, jump to its location.
	Reposition the cursor to the center of the graph. Hold the shift key when repositioning to move all cursors the same amount.
	Delete the cursor.
A, B, C, etc.	The letter number of the cursor.
Time Values	For each cursor, the value where the cursor intersects the time axis is displayed (timestamp). For two or more cursors, a time difference (delta) is displayed, with the leftmost cursor as reference.
Frequency Value	For two or more cursors, hover over the time values to see a frequency value, with the leftmost cursor as reference.
Data Values	For each cursor, the value where the cursor intersects the data plot is displayed at the current zoom level. If there is more than one plot, a color bar corresponding to the plot color will signify the associated data value. For two or more cursors, a data difference (delta) is displayed, with the leftmost cursor as reference. Cursors behave differently in Run Mode vs Stopped Mode. When in Run mode, the cursor values will update every 100 ms. This is to give an idea of data fluctuations and isn't intended for use where accurate data analysis is needed. See Stopped Mode or data logging for applications where better accuracy is desired. Off screen cursors will show n/a during Run Mode, but will update when Stop is clicked. For Stopped Mode, off screen cursors will show data values as long as the corresponding data hasn't been discarded.

Use Two Cursors for Bandwidth

Two vertical cursors can be used to determine bandwidth. Using the time delta, for example in the figure below, the time difference between the position of A and of B is 660 ms. Therefore the bandwidth is 660 ms / 4 cycles ≅ 165 ms/cycle.

Note: As there may be variation between cycles, it is usually best to measure time over several cycles to provide an average value.

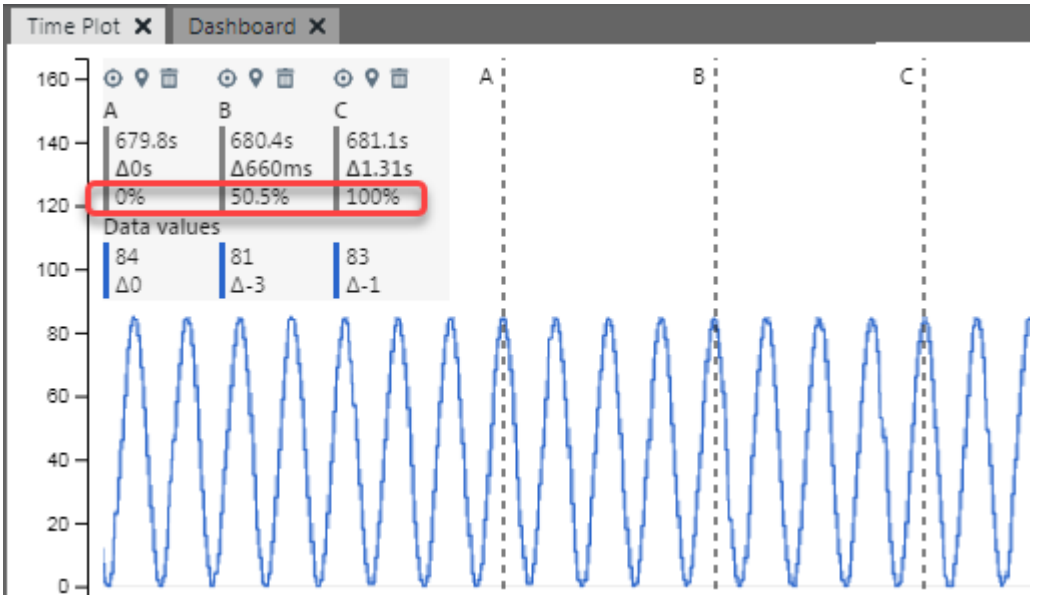
Figure 4-23. Add Two Vertical Cursors - Bandwidth



Use Three Cursors for Duty Cycle

Adding a third cursor allows you to calculate the duty cycle. If A-C is the period, then A-B is shown as a percentage of that (50.5%).

Figure 4-24. Add Three Vertical Cursors - Duty Cycle



Additional cursors may be added to the graph.

4.3.1.7. Automatically Adjust Data Axes

To toggle an automatic adjustment of all data axes, click on the **Toggle Auto Adjust** button in the toolbar at the top of the time plot . Also, double clicking on an axis will toggle auto adjust for that axis.

Figure 4-25. Before Auto Adjust

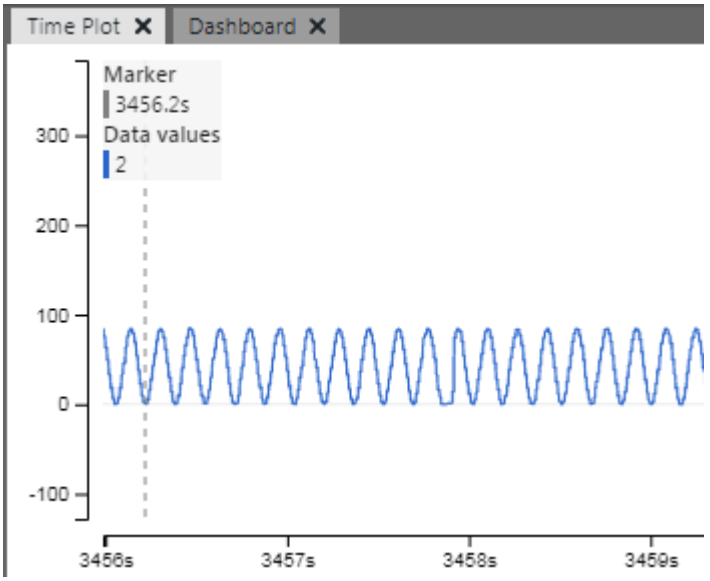
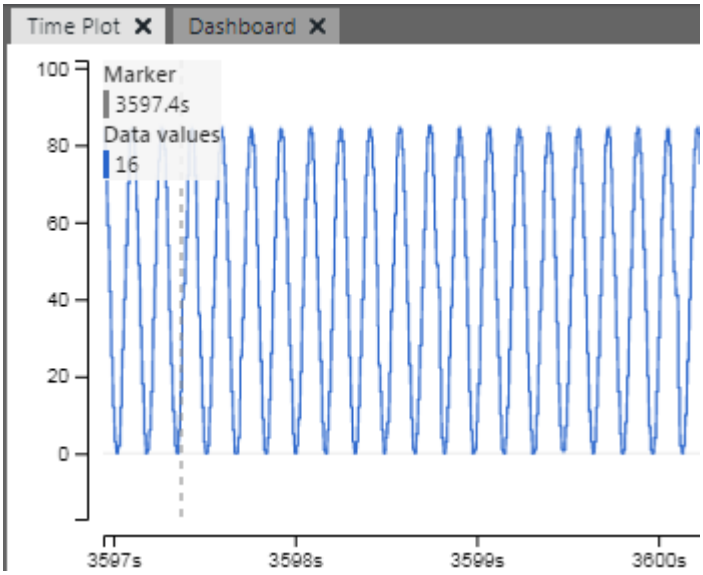


Figure 4-26. After Auto Adjust



4.3.1.8. Marking Data and Saving a Data Snapshot

The time plot supports marking a range of data using two cursors and saving data in this range to disk.

Note: Prior to MPLAB Data Visualizer 1.4, the controls for marking a data range sand saving a snapshot to disk were located in the right side panel.

Related Links

Saving Already Captured Data

4.3.1.9. Inspecting Power Data and PC Profiling

The time plot has a function for inspecting power data, as well as for displaying polled PC samples along with plotted data.

Related Links

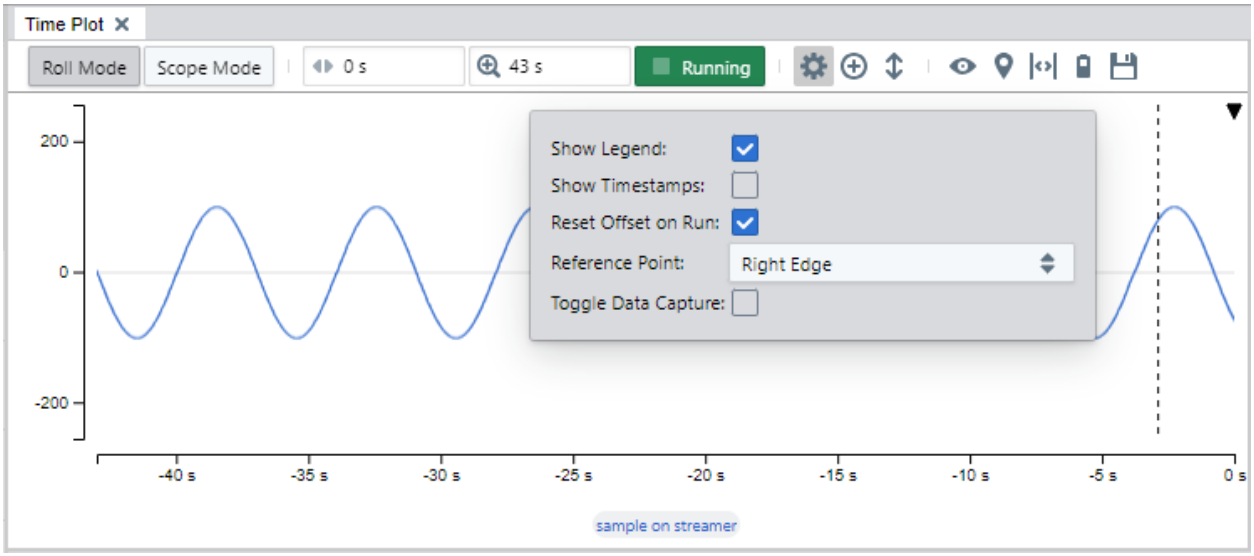
View Power Data

PC Sampling

4.3.1.10. Other Time Plot Settings

Click the cog button on the time plot toolbar to display the time plot settings dialog.

Figure 4-27. Time Plot Settings



The settings dialog include the following settings:

- Show legend: When enabled, labels with the plotted signal names are displayed under the plot.
- Show timestamps: When enabled, the time axis and inspect values boxes will display sample timestamps instead of offset from the trigger point.
- Reset offset on run: When enabled, the time offset will be set back to 0 when entering Run Mode (this is the default).
- Reference point: Selects the position of the reference point within the plot area. Time Offset is relative to this point.
- **Notes:** The reference point is always reset to the default location when changing the Time Mode:
  - In Roll Mode, the default setting is the right edge of the plot area (where data 'arrives')
  - In Scope Mode, the default setting is center
- Toggle data capture: This option lets the time plot control the data source capture state. The time plot's Running/Stopped button will not control whether the data source is capturing data or not by default. If the plot is stopped for a long time while the source is capturing data, there is a chance that the data that is being displayed is discarded in the background to save memory. By enabling this option, there is less chance that data will be discarded while the plot is stopped.

4.3.2. Connections and Protocols

Click a tab to the left of the window to see available connections and protocols. After you connect a target to the PC, that connection will be displayed in the Connections tab and the on-board data sources will be listed. Some sources have options that can be set using the **Source options** button . Select to plot the raw source data. If you are connected and wish to change the settings, you can disconnect by clicking . Make your changes and then click the button again to reconnect.

Sources that are not a part of a group can also be plotted on the Time Plot by dragging and dropping. Drop on an axis to add the source to it, or on a plot to add it to the rightmost axis on that plot. Holding shift while dropping on axis or plot adds the source on a new axis on the same plot. You can also place the source on a new axis on a new plot above or below by dropping on the upper or lower area of the plot.

4.3.3. Variable Streamers Tab

Click the **Variable Streamers** tab to create, edit or delete variable streamers. **Import Variable Streamer** is a way to import `.ds` or `.txt` files that were used by the Atmel Studio Data Visualizer. More details on file format can be found in this document: ww1.microchip.com/downloads/aemDocuments/documents/OTH/ProductDocuments/UserGuides/40001903B.pdf

To view decoding statistics, streaming data information in real-time, click on the eye icon.

When there are variables in a data stream, the stream must be decoded before the variables can be visualized. For details, see Variable Streamers.

Figure 4-28. Variable Streamers Tab

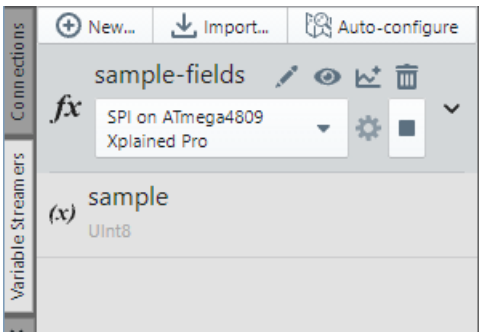
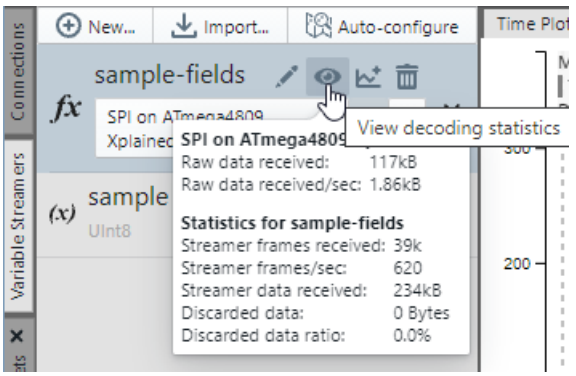


Figure 4-29. View Decoding Statistics



4.4. Time Plot Examples

Below are examples of use for the Time Plot.

4.4.1. Example of Plotting Data

An ATmega4809 Xplained Pro Board is used to generate data that is output via the DGI/SPI to the visualizer and plotted using variable streamers.

4.4.1.1. Example Setup

Complete the following instructions to set up the example software and hardware.

MPLAB X IDE

Download and install MPLAB X IDE 6.05 (or later) for free from the link below.

www.microchip.com/mplab/mplab-x-ide

Select the following packs: ATmega_DFP v3.0.158 (or later), EDBG_TP v1.5.495 (or later).

MPLAB XC8 C Compiler

Download and install the MPLAB XC8 C compiler v2.35 (or later) for free from the link below. A PRO version of the compiler with additional optimizations and features is also available for purchase.

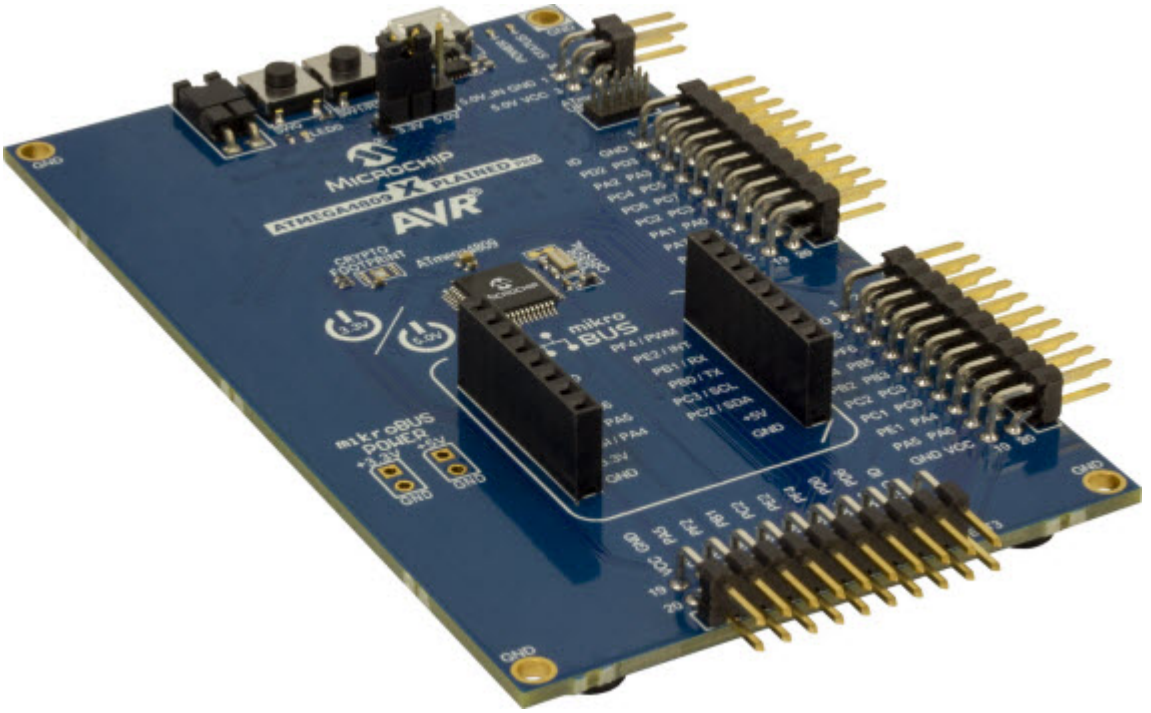
www.microchip.com/mplab/compilers

ATmega4809 Xplained Pro Board - ATMEGA4809-XPRO

Acquire this evaluation board (see image below) from microchipDirect or a distributor. Then connect the board to your computer via the enclosed USB cable to install the drivers.

For more information about this board, go to:

www.microchip.com/developmenttools/ProductDetails/PartNo/ATMEGA4809-XPRO



4.4.1.2. Create Example Project

MPLAB X IDE requires a project for development of application code.

Preliminaries

Before creating the project ensure:

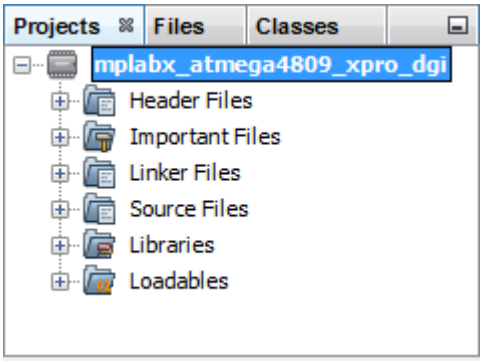
- You have installed the compiler and MPLAB X IDE can detect it. If not, go to *Tools>Options>Embedded>Build Tools* to view the Toolchain list. If the compiler is not there, click **Add** to browse and add it.
- You have plugged the Xplained Pro board into your computer with the USB cable.

Create Project

Select *File>New Project* or the **New Project** icon  to open the Project wizard. Follow the steps below to create your project. Click **Next** to move to the next step.

1. **Choose Project**: Click on the “Microchip Embedded” category and then the “Standalone Project” project.
2. **Select Device (and Tool)**: Enter the device “ATmega4809.” Then enter the tool “ATmega4809 Xplained Pro-SN: ATML#” where the tool serial number (SN) contains the prefix “ATML” followed by a multi-digit number.
3. **Select Compiler**: Under *Compiler Toolchains>XC8*, Select the most-current compiler version.
4. **Select Project Name and Folder**: Name your project. For example, “mplabx_atmega4809_xpro_dgi.” For Windows OS, the default project folder is `C:\Users\<UserName>\MPLABXProjects`.

After clicking **Finish**, the project tree should appear in the Projects window.



4.4.1.3. Add Files to Project

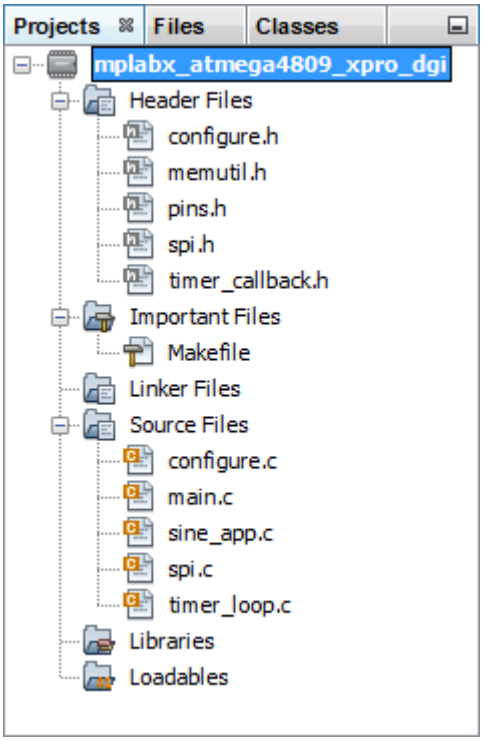
To add C header files to the project, right click on the “Header Files” folder and select *New>C Header File*.

To add C source code to the project, right click on the “Source Files” folder and select *New>C Main File* (once, for `main.c`) or *New>C Source File* for all other files.

Example code for this project is found in Example of Plotting Data - Code Listing.

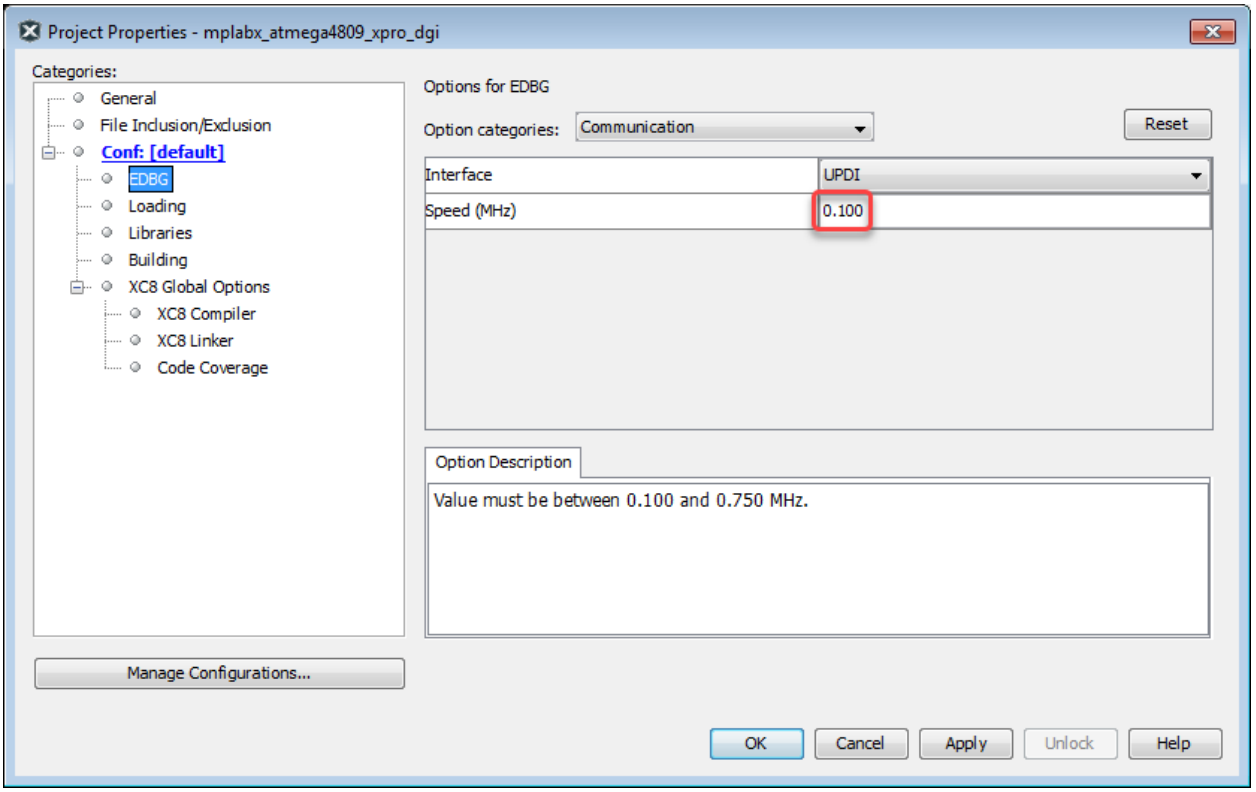
The completed project should look like the figure below.

Figure 4-30. Project with Files



4.4.1.4. Set Project Communications Options

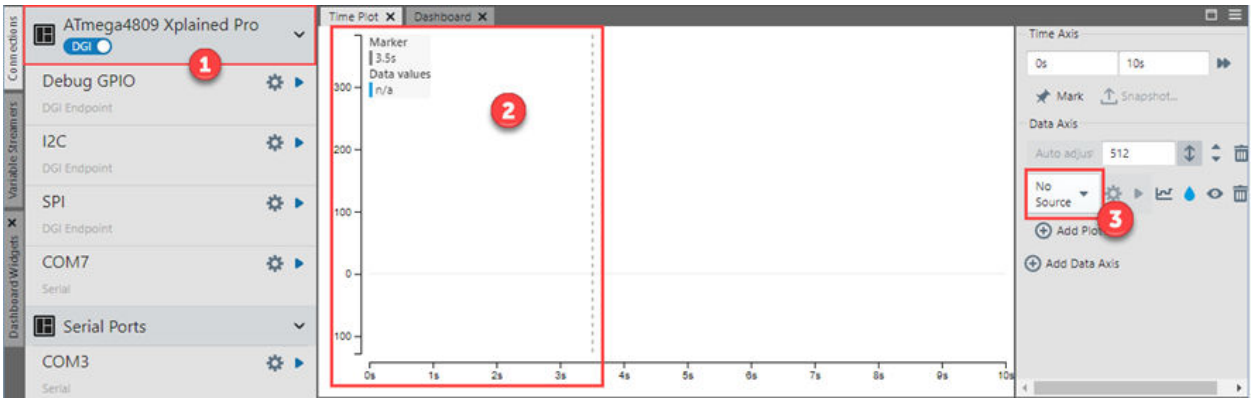
Right click on the project name in the Projects window to open the Project Properties. Under EDBG Communication options, make the Speed “0.100” to match the settings in code.



4.4.1.5. Open MPLAB Data Visualizer

Open the MPLAB Data Visualizer plug-in by clicking the toolbar icon or selecting *Windows>Debugger>Data Visualizer*.

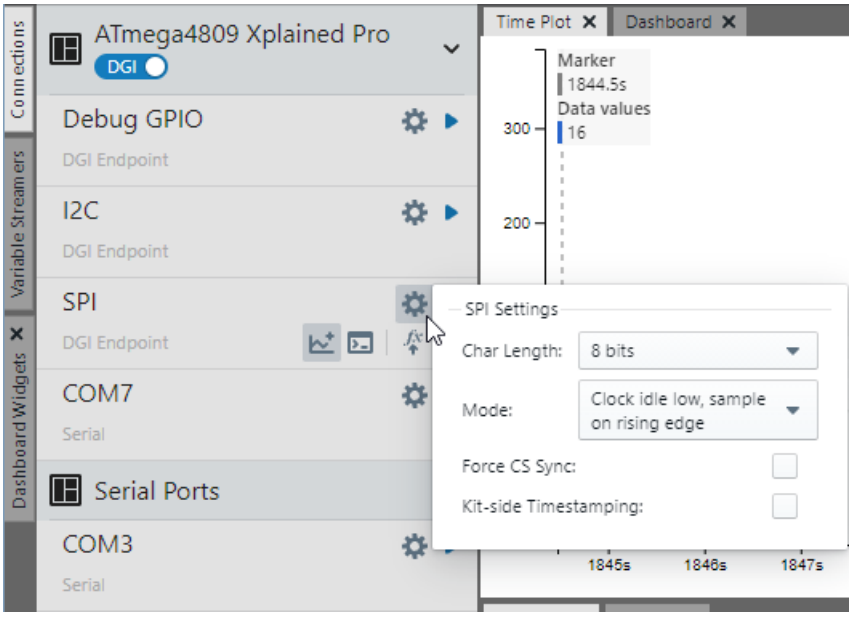
1. In the Data Source pane, on the **Connections** tab, you should see "ATmega4809 XPlained Pro" with "DGI" enabled.
2. In the Graph tab, the sliding marker (dashed gray line) will be at zero and there will be no data values (if available would be blue to match source color).
3. In the Visualization pane, the "Source" box for the Time Axis should say "No Source."



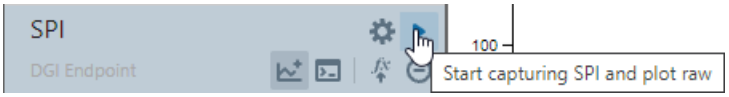
4.4.1.6. Debug Project and Visualize Output

To begin debugging the project, click on the "Debug Project" icon .

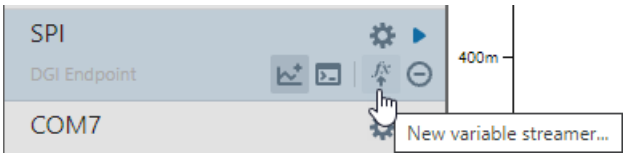
In MPLAB Data Visualizer, under “ATmega4809 Xplained Pro” (DGI enabled), are the available connections. On the SPI connection, click on the gear icon to view SPI settings. For details on what these settings mean, see SPI Interface.



In general to begin streaming data, click on the arrow to plot raw data. However, for this example variable data will be plotted.



To plot streaming variable data, set up a variable streamer by clicking the fx control (New Variable Streamer).



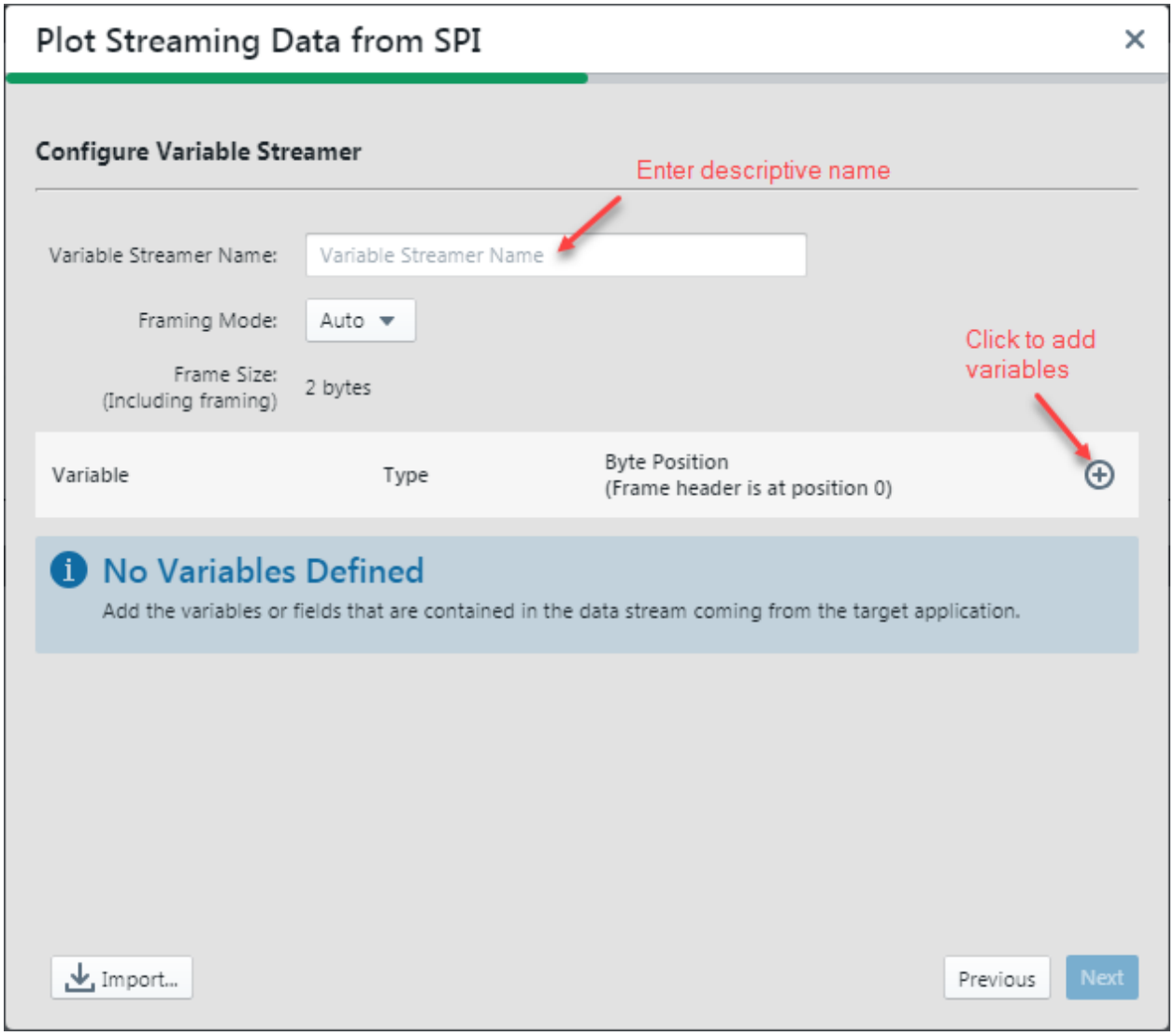
4.4.1.6.1. Plot Streaming Data for SPI

Configure Variable Streamer - Initial View

For this example, the values of a variable will be streamed and plotted. The first dialog of the plotting wizard will look as below. Name the variable streamer to identify the setup later. Then add the variable by clicking on the “+” as shown to display text boxes for entry.

Note: If a previous setup had been saved, you could load it by clicking the **Import** button on the bottom of the dialog.

Also, using the “?” key opens a keyboard shortcuts dialog, and using the “Esc” key closes the dialog.

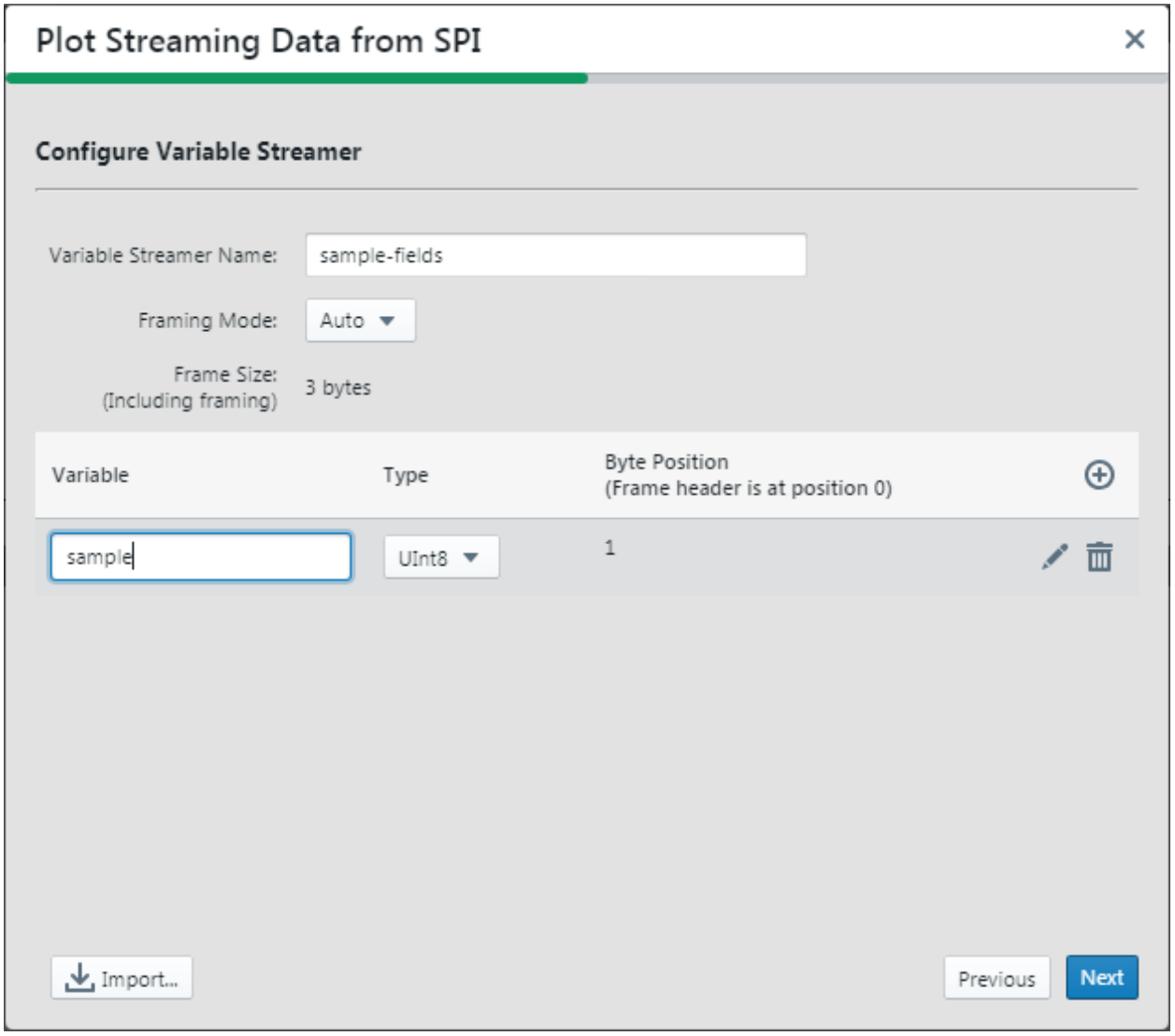


Configure Variable Streamer - Enter Data

The dialog below shows the previous dialog with data entered. Specified information on the variable has been provided.

In order to decode a data stream, the variables (or fields in the data stream) must be defined. The data streamed will be of the format shown in Stream Format.

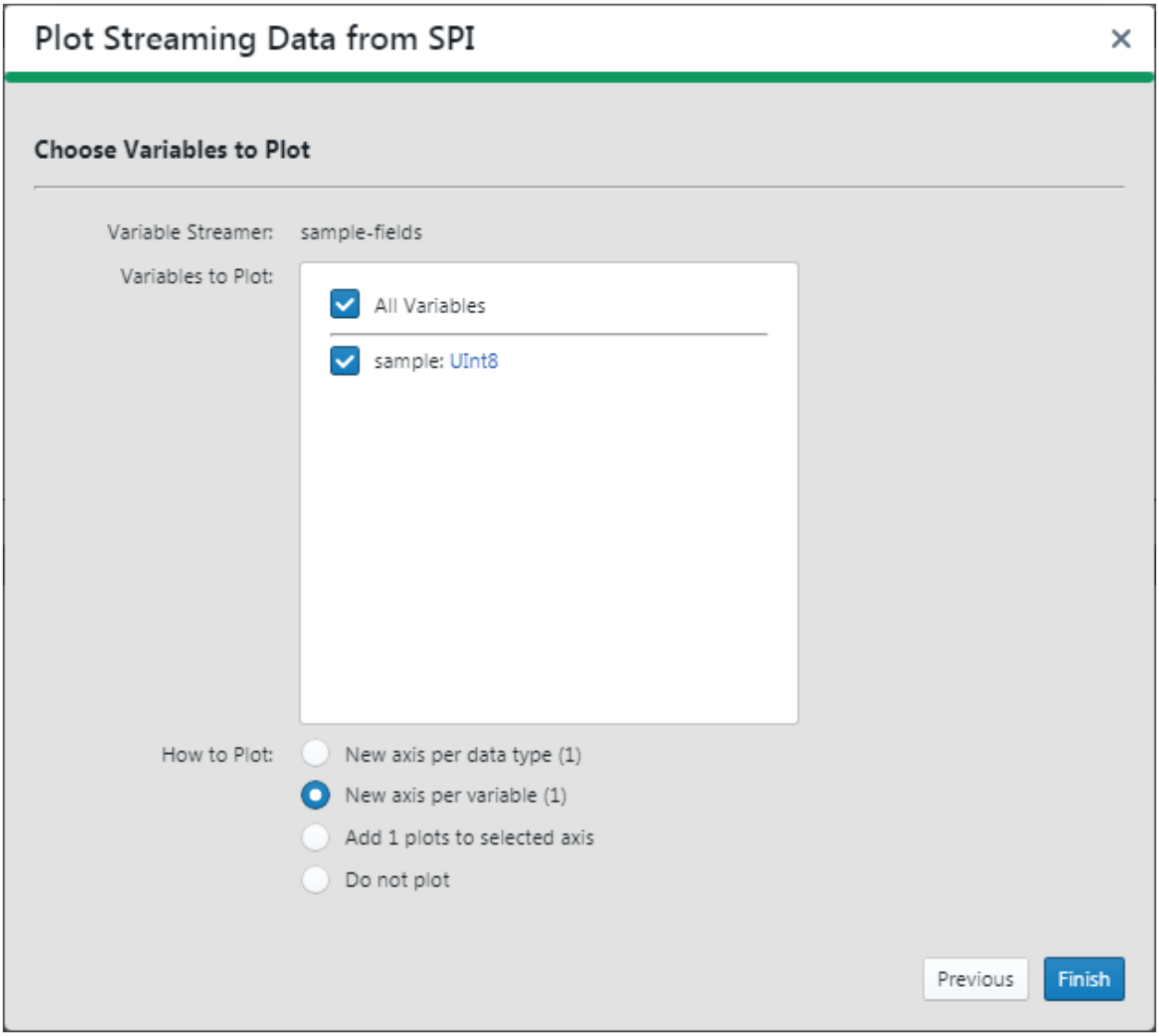
Click **Next** to proceed with setup.



Configure Variable Streamer - How to Plot

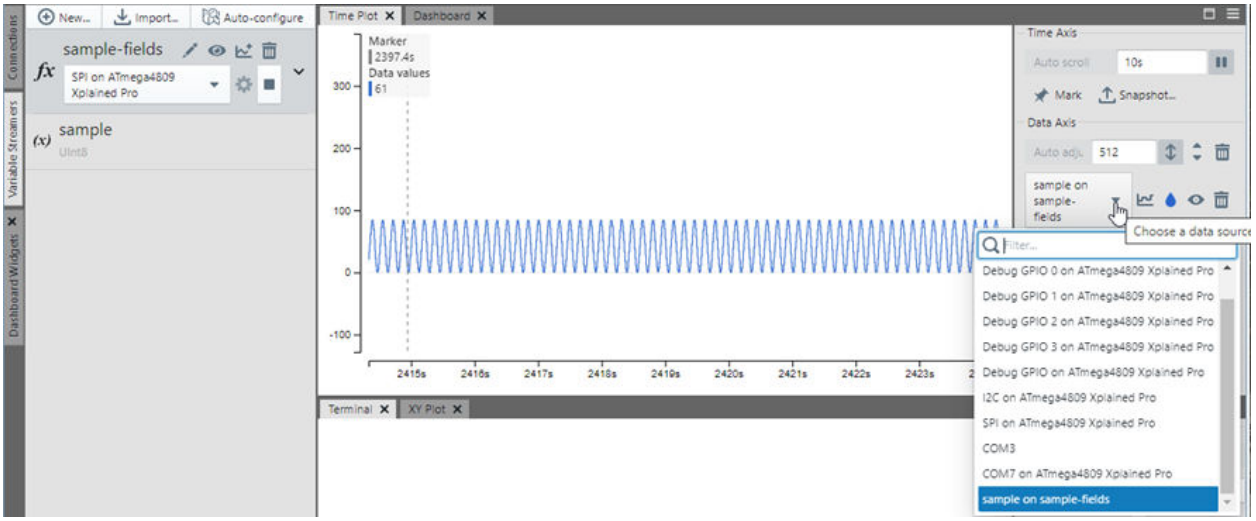
This dialog shows a summary of the previous one and a selection list of how to plot the data. For this example, “New axis per variable (1)” has been selected.

Click **Finish** to proceed to plot.

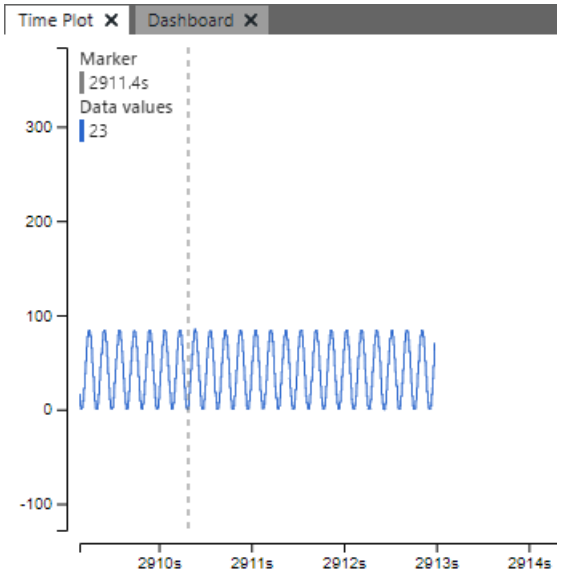


4.4.1.6.2. View Data based on Plot Setup

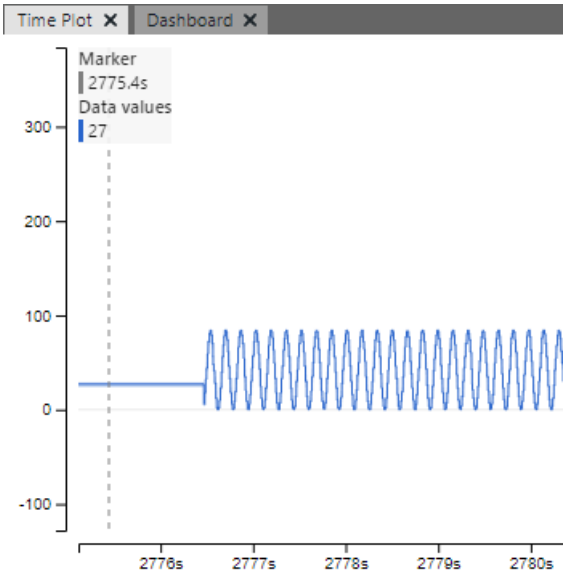
The Time Axis Source should now be set to “sample on sample-fields” and data should be plotted on the graph. In order to toggle the data scrolling in the graph, press "Pause Scrolling/Show Live Data" on the Graph banner, or use the Space key.



When debug is paused  , data output is stopped.



When debug is continued,  , data is again shown on the graph.

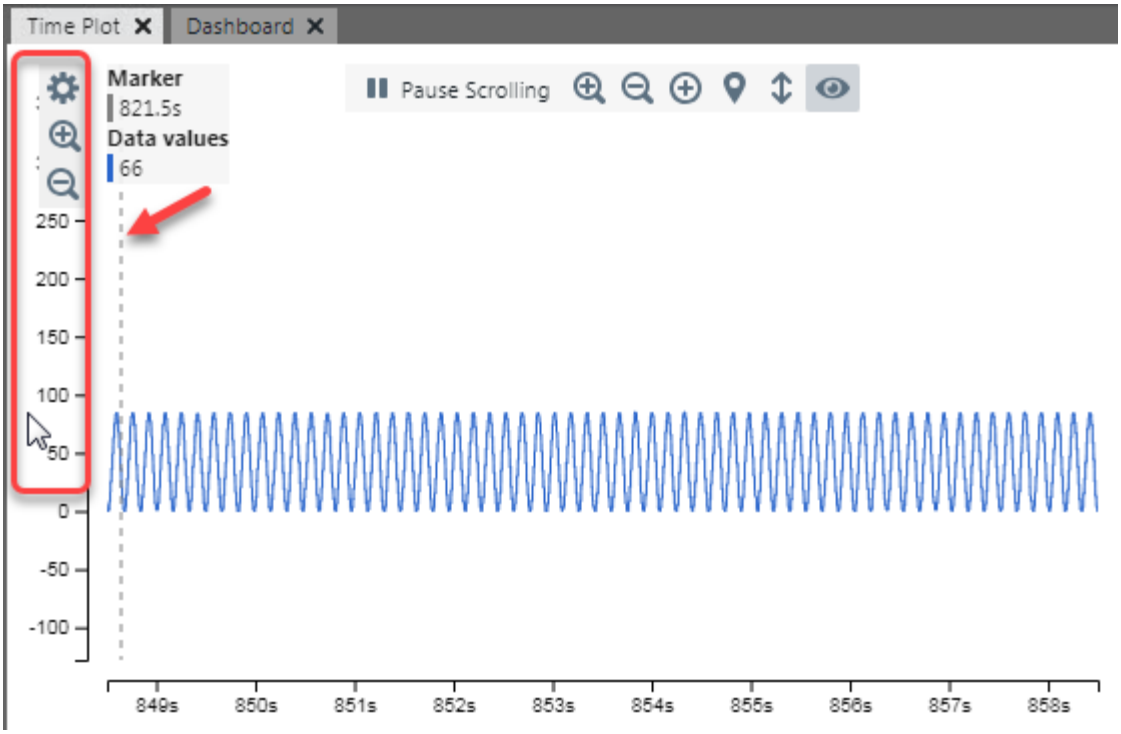


4.4.1.6.3. Analyzing Plot Data with Tools

Graph tools may be used to change the view of and analyze plot data.

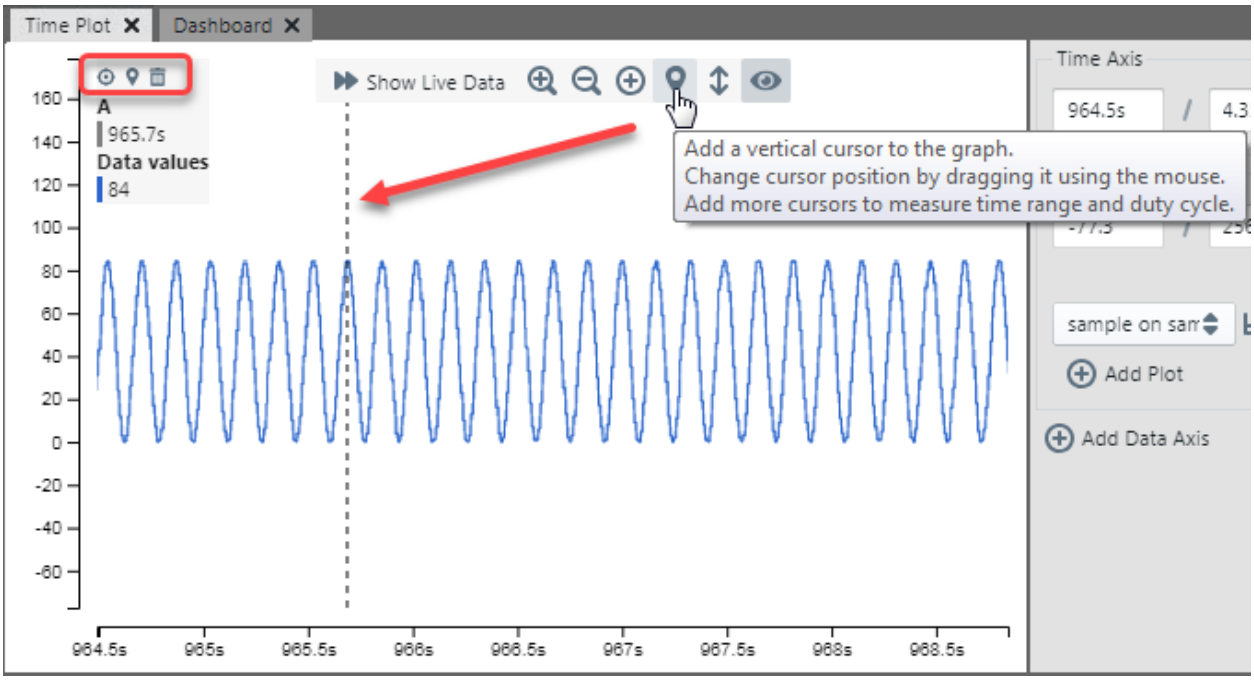
Adjusting an axis range and plot location can make viewing data easier. In the image below, the Data Value (vertical) axis is circled for example. Click on or near the axis you want to adjust. Then use the mouse wheel to zoom in or out on the axis range. You can also click and hold to drag the axis one way or the other, thus moving the plot accordingly. There are also controls at one end of the axis that can be used to zoom in or out and set plot characteristics as on the Visualization pane.

On the graph there is a rolling vertical marker that will follow mouse movements and show the corresponding Time (horizontal axis) and Data Value (vertical axis).

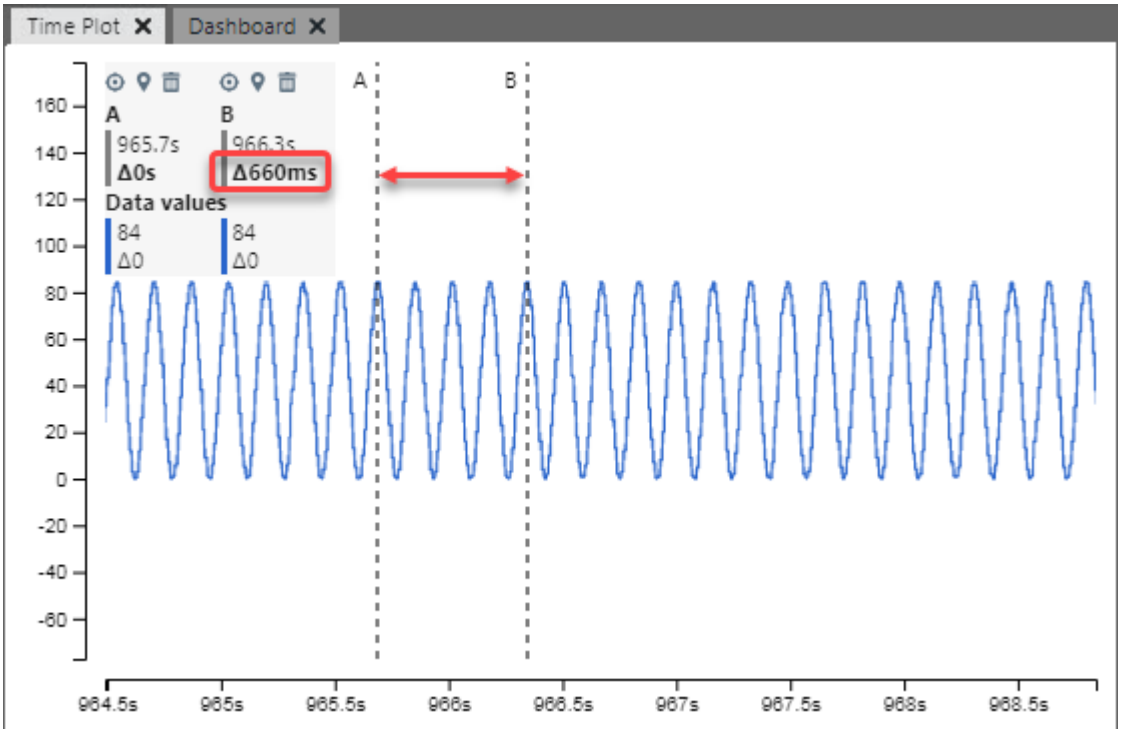


To enable a vertical cursor that may be dragged and dropped at a plot location, click as shown below to enable cursor A. This will disable the rolling marker.

Controls for the cursor will appear where the rolling marker information had been located.



To determine time between plot points, enable another vertical cursor, set its location, and then view the time delta.



4.4.2. Example of Multiple Data Plots

An AVR128DA48 Curiosity Nano board is used to demonstrate how to use GPIO pins to generate multiple data plots, either on the same axis or different axes.

4.4.2.1. Example Setup

Example Setup Follow the instructions in the following sections to set up example software and hardware.

MPLAB X IDE

Download and install MPLAB X IDE 6.05 or later for free from the link below.

www.microchip.com/mplab/mplab-x-ide

Ensure the following pack is selected: AVR-Dx_DFP v2.2.157 (or later).

MPLAB XC8 C Compiler

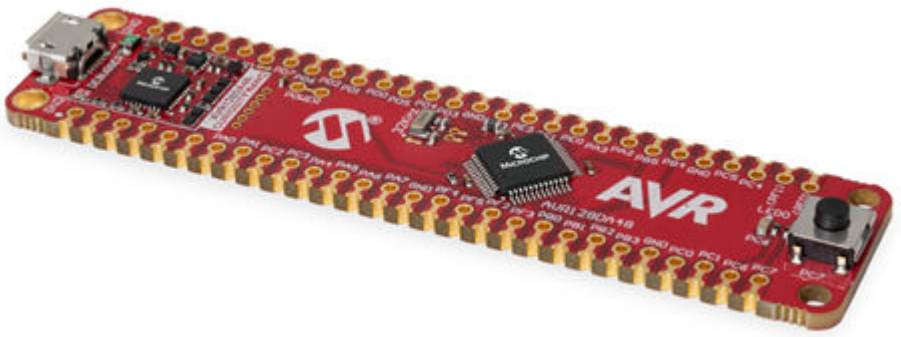
Download and install the MPLAB XC8 C compiler v2.35 or later for free from the link below. A PRO version of the compiler with additional optimizations and features is also available for purchase.

www.microchip.com/mplab/compilers

AVR128DA48 Curiosity Nano Evaluation Kit - DM164151

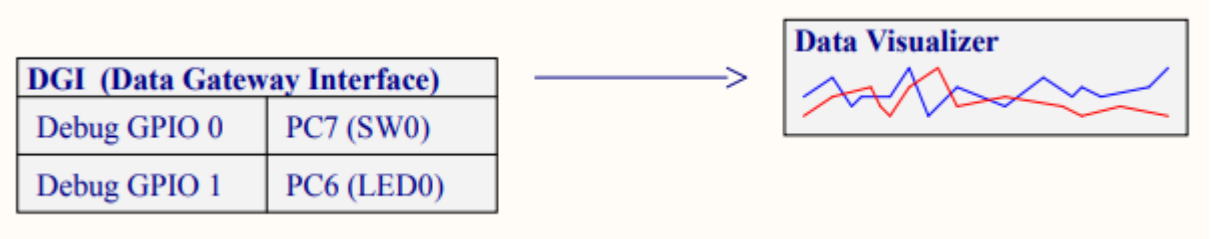
Acquire this evaluation board (see image below) from microchipDirect or a distributor. Then connect the board to your computer via the enclosed USB cable to install the drivers. For more information about this board, go to:

www.microchip.com/DevelopmentTools/ProductDetails/PartNO/DM164151

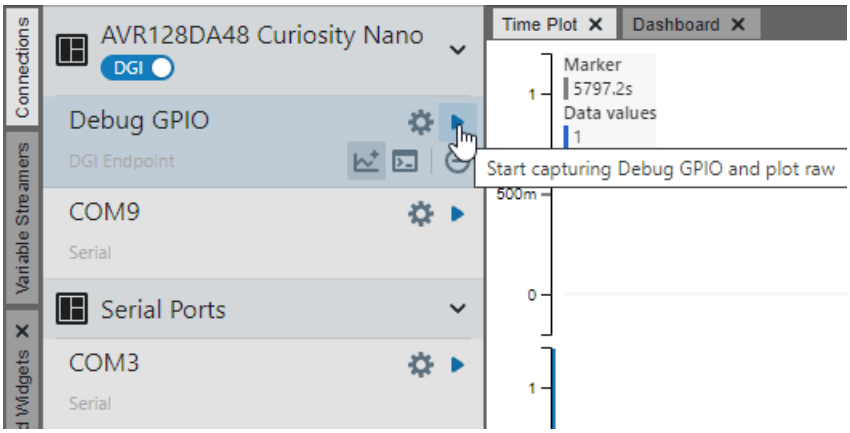


4.4.2.2. Plug and Play with the Curiosity Nano

The AVR128DA48 Curiosity Nano Evaluation board is designed to be plug-and-play. Therefore plug the board into the PC using the USB cable and then launch MPLAB X IDE. When the IDE opens, you should see a **Kit Window** tab with information about the Curiosity Nano. Click on the board schematics link and on the first page of the schematics find GPIO pin references that can be used with the MPLAB Data Visualizer. PC7 is attached to the board pushbutton switch and PC6 is attached to LED0.

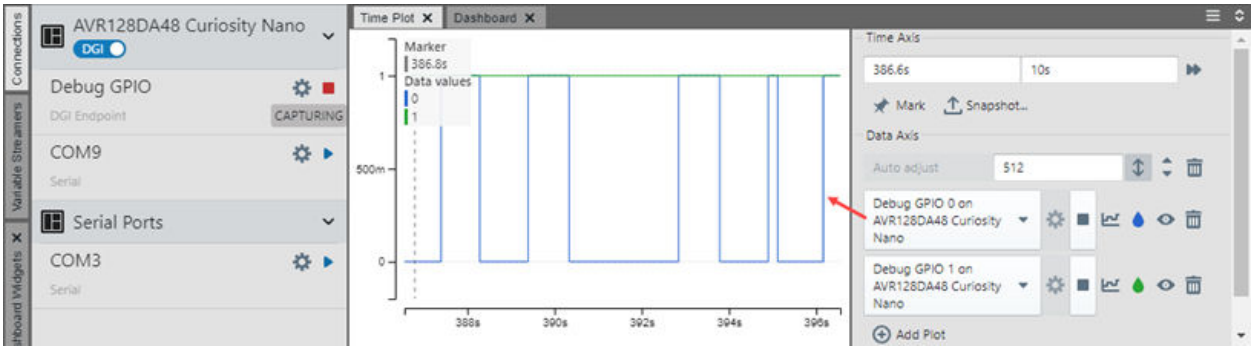


Open the MPLAB Data Visualizer plugin. The visualizer will display available data sources, including Debug GPIO for the GPIO pins. Click on the arrow to plot all pins.



On the Graph, both GPIO pin outputs are plotted on a single axis. On the right side of the graph you will see information about each plot, including its color-coding.

Now it's time to play with the Curiosity Nano. Press the board switch to see a pulse on GPIO 0. Note that there is lag between when the button is pressed and when the pulse appears. Refer again to the schematic to see that there is no pull-up on the button. However, pin pull-ups can be enabled using software. Also, GPIO 1 is only showing a single-line plot, but a pulse produced from a timer could provide a more interesting plot. Therefore it is time to create a project and add some code.

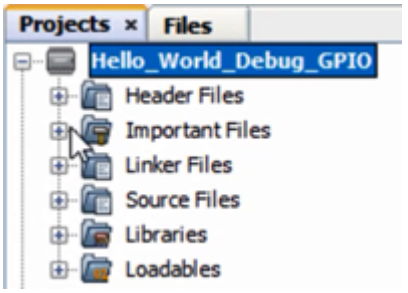


4.4.2.3. Create Example Project

Select *File>New Project* or the **New Project** icon  to open the Project wizard. Follow the steps below to create your project. Click **Next** to move to the next step.

1. **Choose Project**: Click on the "Microchip Embedded" category and then the "Standalone Project" project.
2. **Select Device (and Tool)**: Enter the device "AVR128DA48." Then enter the tool "AVR128DA48 Curiosity Nano-SN: MCHPL#" where the tool serial number (SN) contains the prefix "MCHP" followed by a multi-digit number.
3. **Select Compiler**: Under *Compiler Toolchains>XC8*, select the version used in this example or the most-current version.
4. **Select Project Name and Folder**: Name your project. For example, "Hello World Debug GPIO." For Windows OS, the default project folder is `C:\Users\<UserName>\MPLABXProjects`.

After clicking **Finish**, the project tree should appear in the Projects window.



4.4.2.4. Create an Application

Create an application by adding source code to the project.

4.4.2.4.1. Create a New Main Source File

To add a new source file to the project:

1. Right click on the project source folder and select *New>avr-main.c*.
2. Under "Name and Location," change the file name to `main.c`. Then click **Finish**.

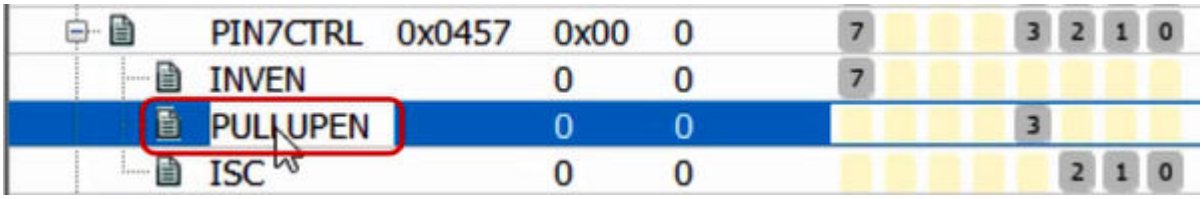
The new file will open in an Editor window. By default the code will look like this.

```
/*
 * File:   main.c
 * Author: Microchip Technology Inc.
 *
 * Created on May 19, 2020, 1:37 PM
 */


#include <avr/io.h>

int main(void) {
    /* Replace with your application code */
    while (1) {
    }
}
```

4.4.2.4.2. Add Code for a Pullup on PC7

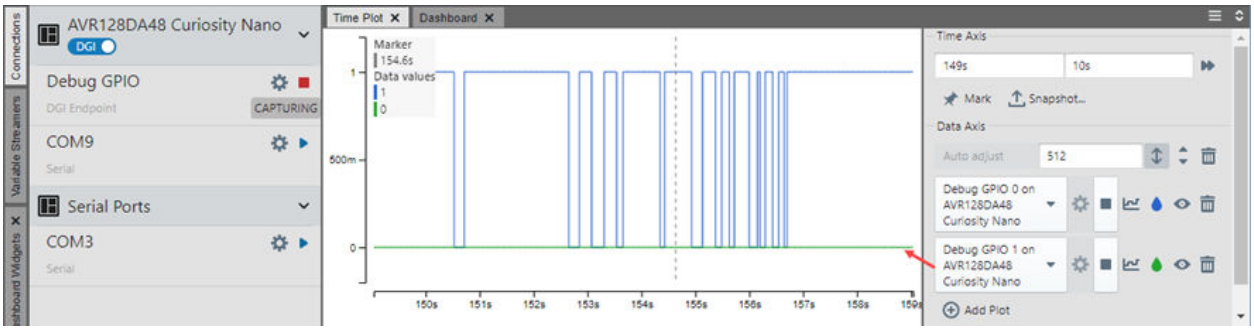
Code for enabling a pull-up on Port C, Pin 7, will be added to the source code. The View IO window is helpful in locating the correct register selection.

Figure 4-31. View IO Window




```
#include <avr/io.h>

int main(void) {

    PORTC.PIN7CTRL = PORT_PULLUPEN_bm; /* Enable PC7 Pullup */
    PORTC.DIR = PIN6_bm; /* Turn on LED */

    while (1) {
    }
}
```

Program the part  to see the better response of the PC7 switch (GPIO 0) and the turning on of the PC6 LED (GPIO 1).



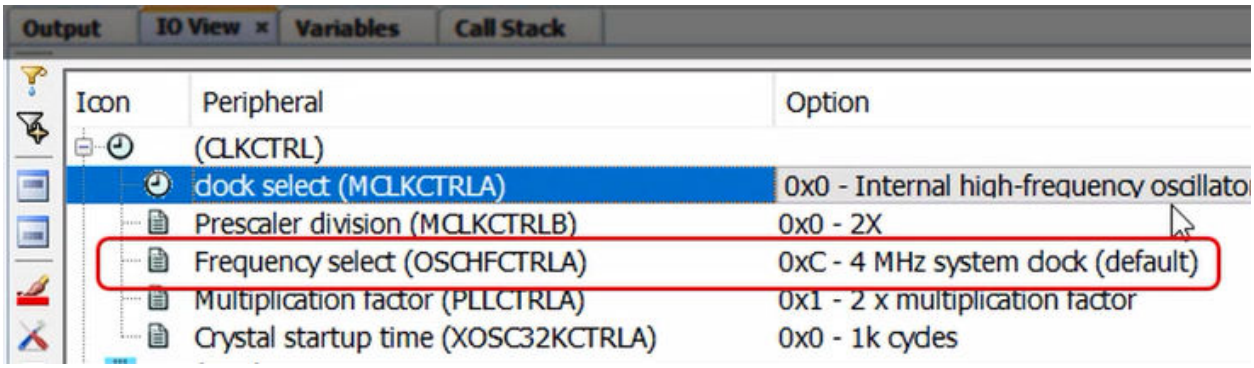
4.4.2.4.3. Add Code to Toggle LED with a Delay

Previously code was added to the main source file to enable a pull-up on PC7 for better switch response and the user LED was turned on. Now code is added to toggle the LED on and off. Due to the speed of the device this could be imperceptible unless a delay is added to slow down the toggle. Information on the delay used can be found at:

www.nongnu.org/avr-libc/user-manual/group__util__delay.html

A `#define` for the CPU frequency needs to be added as well as a `#include` for delay support.

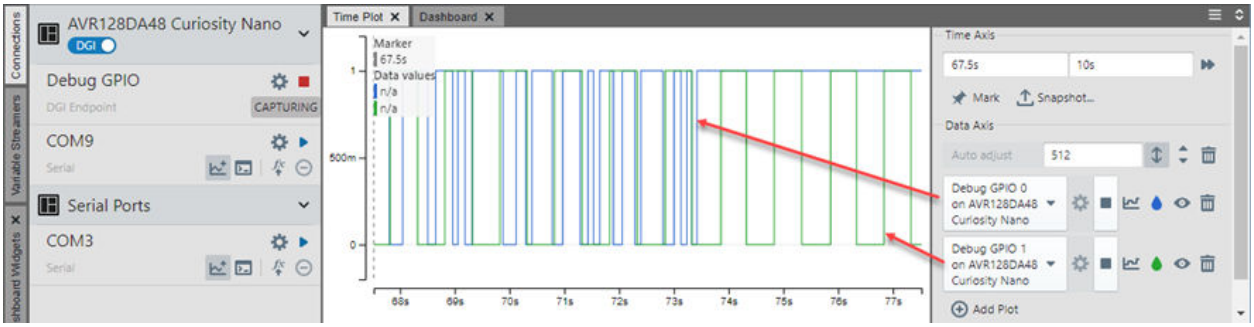
To determine the `F_CPU` for the Curiosity Nano, open the View IO window again and perform a debug run of the code to get live values in the window. Once the code is in the `while(1)` loop, Pause and look at CLKCTRL to find the value.



The updated code will be as shown below.

```
#include <avr/io.h>
#define    F_CPU    4000000UL
#include <util/delay.h>

int main(void) {

    PORTC.PIN7CTRL = PORT_PULLUPEN_bm; /* Enable PC7 Pullup */
    PORTC.DIR = PIN6_bm; /* Turn on LED */

    while (1) {
        PORTC.OUTTGL = PIN6_bm; /* Toggle LED on/off */
        _delay_ms(500); /* wait between toggles */
    }
}
```

Plotting the GPIO pins again, you can see PC7 (GPIO 0) in blue with different sized pulses from button presses and PC6 (GPIO 1) in green with pulses from toggling.



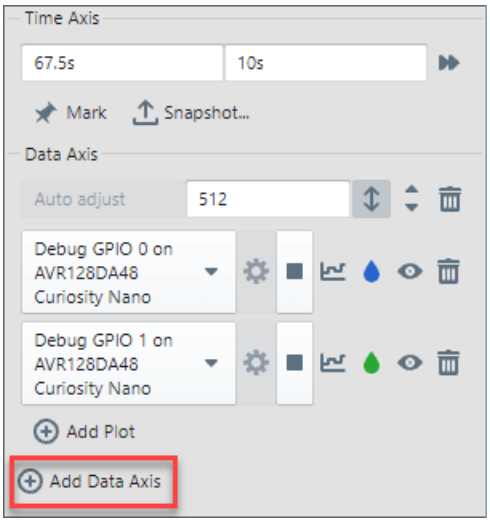
4.4.2.5. Plot Configurations

The project code produces outputs on both GPIO pins, which creates a crowded display on a single axis, even with color-coded plots. It would be better to have each plot on its own axes. Once the plots are each on an axis, it is easy to view the effect GPIO options have on the plots.

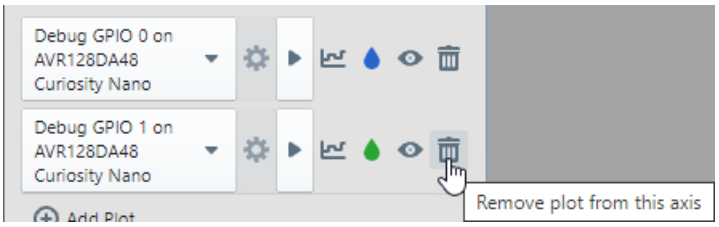
4.4.2.5.1. Move Each Plot to an Axis

For this example, there are only two plots on one axis, but the same procedure would work for more.

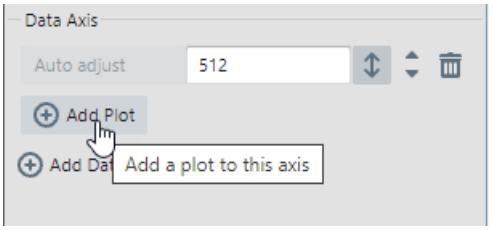
1. Add another axis to the Graph. An empty axis will appear beneath the original.

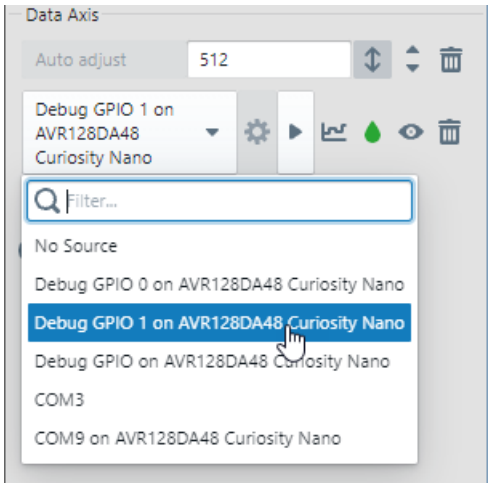


2. Delete one plot from the original axis.

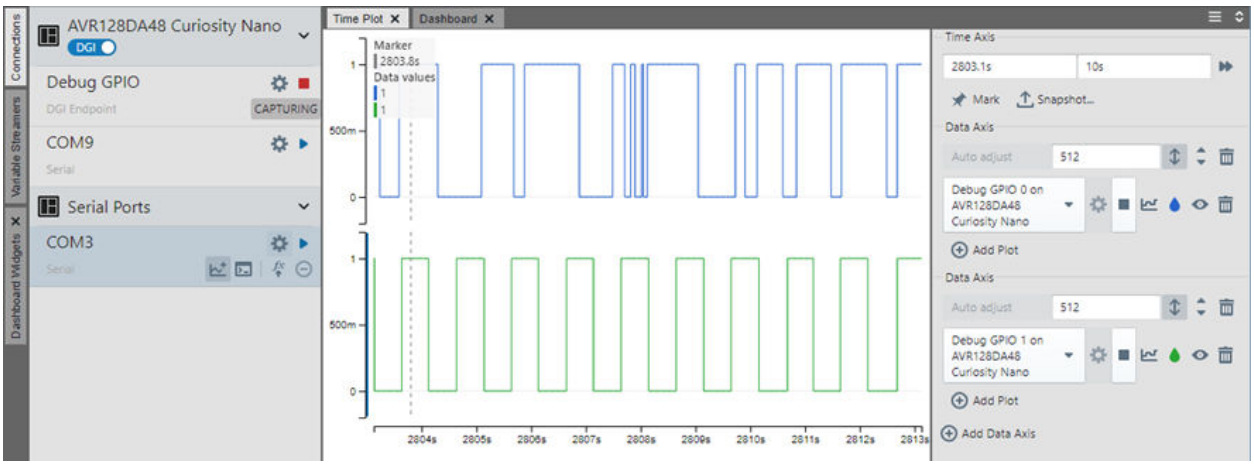


3. Add the deleted plot to the new axis. Click “Add Plot” and then select the deleted plot from the drop-down list.





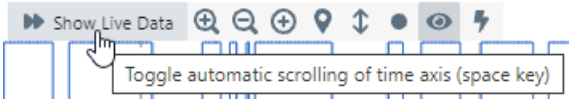
4. Each plot will be on its own axis and will be much more visible.



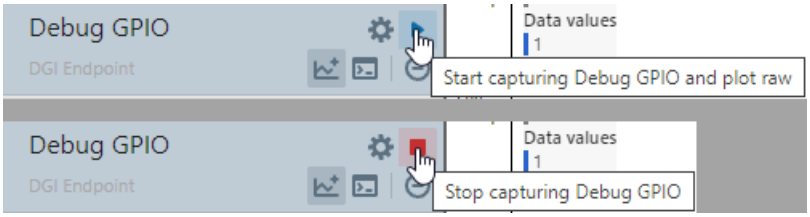
4.4.2.5.2. Start and Stop Plots

There are several ways to start and stop streaming plots.

The first way is to start and stop the scrolling of plots but not the capture of data by clicking on the Show Live Data/Pause Data button on the toolbar or by alternately pressing the space key.



The second way is to start and stop the capture of data. Use the start and stop capturing controls on the Connection used for data streaming.



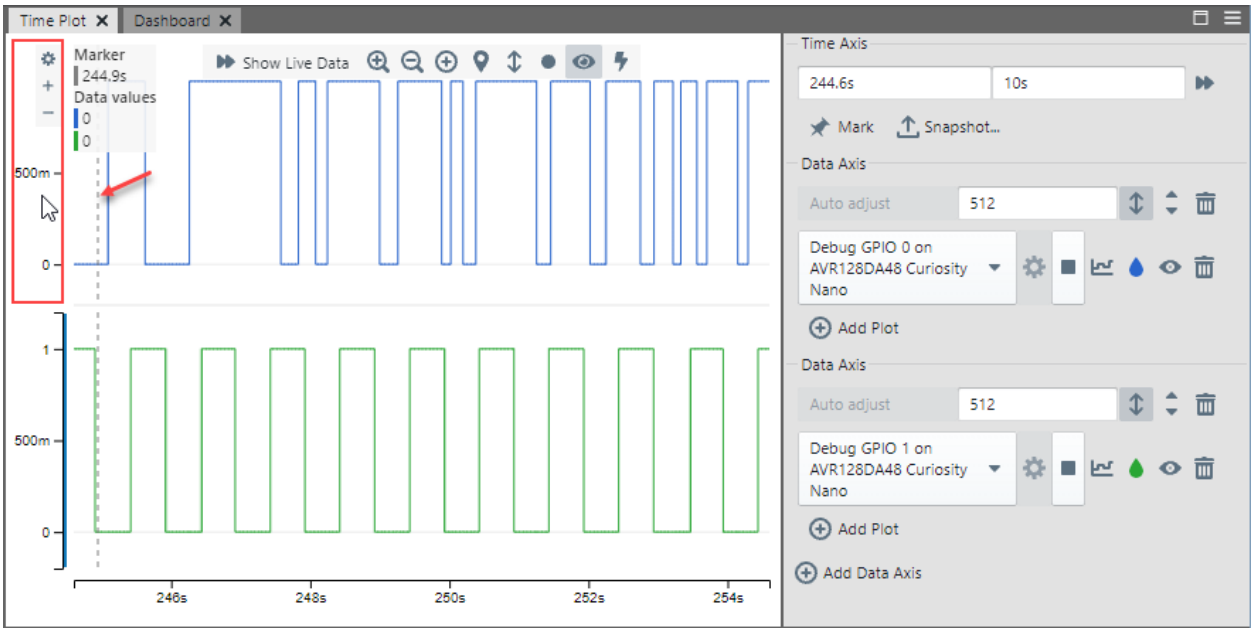
The third way it to halt and then continue the execution of the application in MPLAB X IDE by using the Pause and Continue buttons. This only applies for the MPLAB Data Visualizer plugin.

4.4.2.5.3. Analyzing Plot Data with Tools

Graph tools may be used to change the view of and analyze plot data.

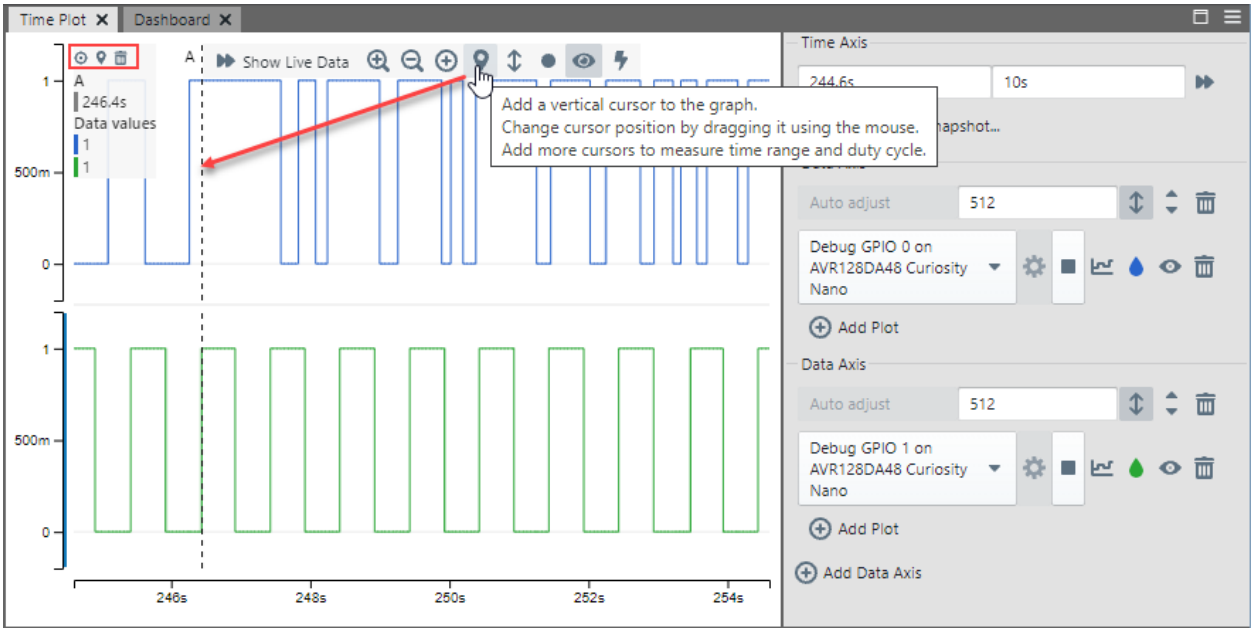
Adjusting an axis range and plot location can make viewing data easier. In the image below, the Data Value (vertical) axis is circled for example. Click on or near the axis you want to adjust. Then use the mouse wheel to zoom in or out on the axis range. You can also click and hold to drag the axis one way or the other, thus moving the plot accordingly. Also. controls at one end of the axis can be used to zoom in or out, and set plot characteristics, as on the Visualization pane.

On the graph there is a rolling vertical marker that will follow mouse movements and show the corresponding Time (horizontal axis) and Data Value (vertical axis).

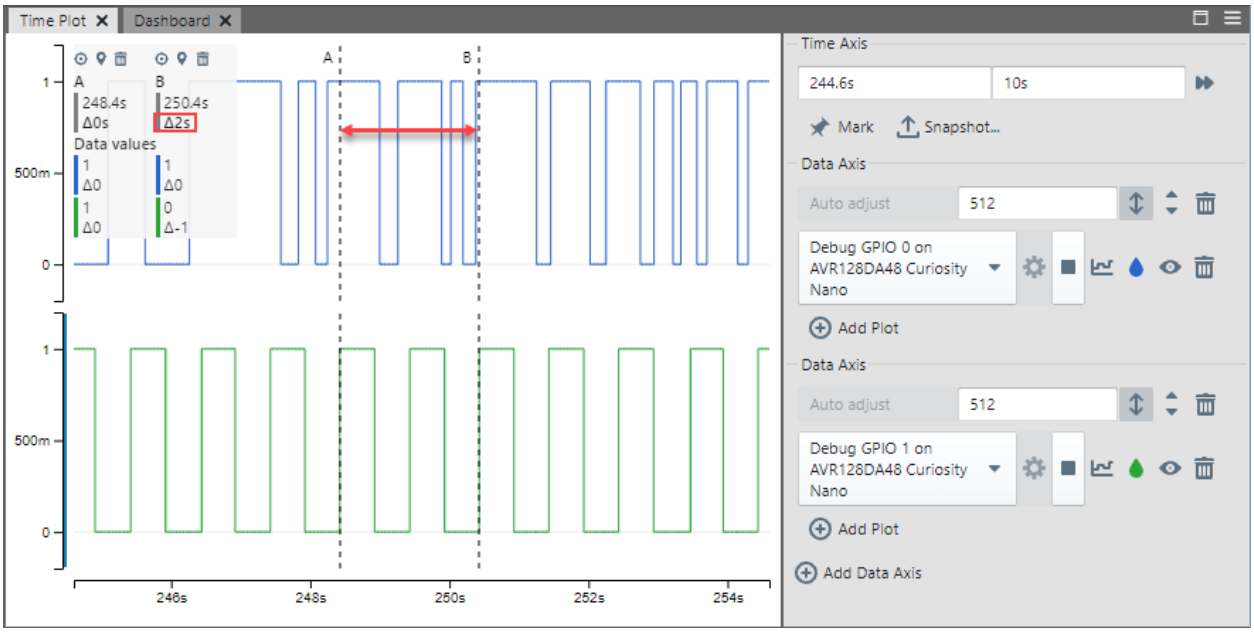


To enable a vertical cursor that may be dragged and dropped at a plot location, click as shown below to enable cursor A. This will disable the rolling marker.

Controls for the cursor will appear where the rolling marker information had been located.



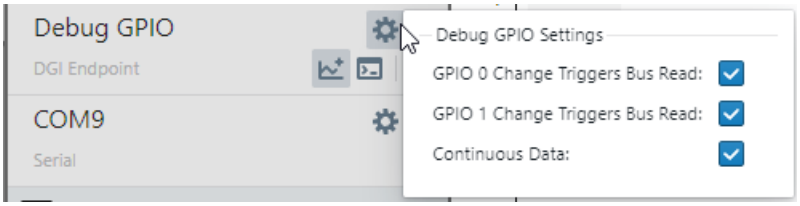
To determine time between plot points, enable another vertical cursor, set its location, and then view the time delta.



4.4.2.5.4. Debug GPIO Options

When the Debug GPIO data source is selected, options are visible by clicking the gear icon. All options are enabled by default.

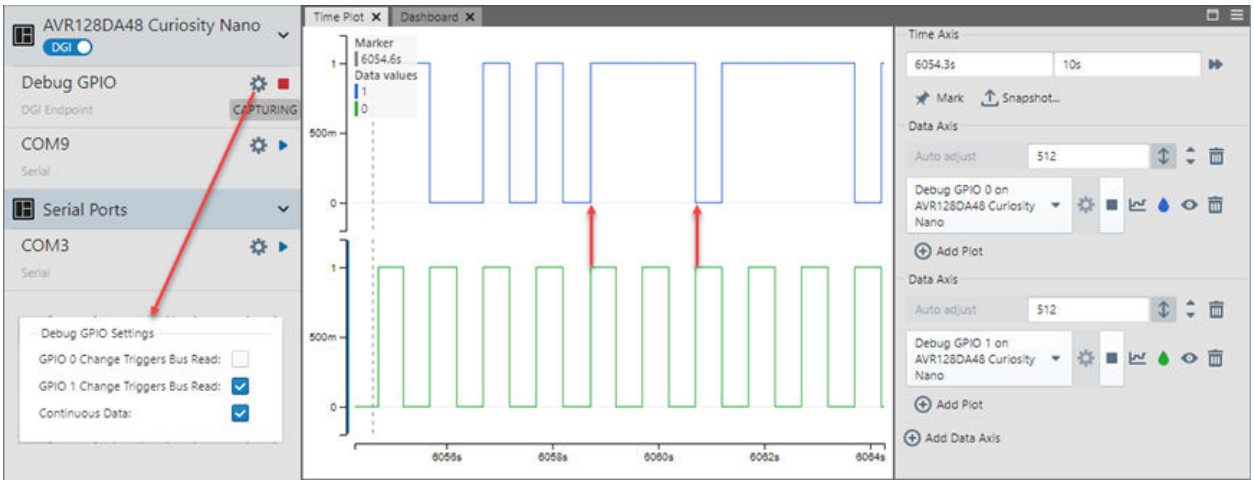
Figure 4-32. Debug GPIO Options



"GPIO x Change Triggers Bus Read" means that whenever there is a change on GPIO x, the GPIO bus is read and data displayed on the plots.

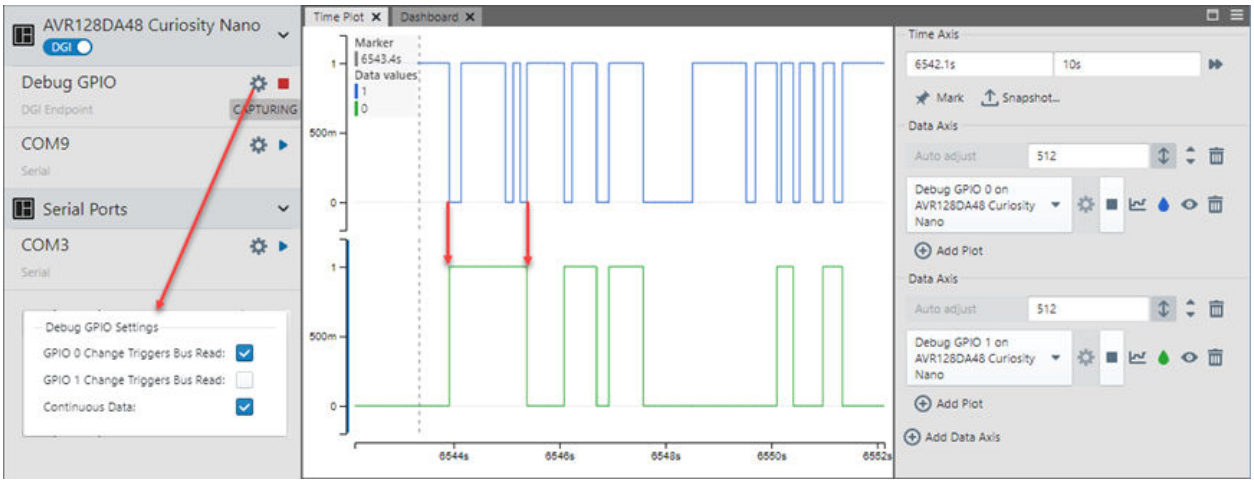
If "GPIO 0 Change Triggers Bus Read" is disabled and "GPIO 1 Change Triggers Bus Read" is enabled, the bus will only be read when GPIO 1 toggles, meaning that even for a quick button press, only changes that occur between GPIO 1 toggles will be displayed.

Figure 4-33. GPIO 0 Option Disabled, GPIO 1 Option Enabled



If "GPIO 0 Change Triggers Bus Read" is enabled and "GPIO 1 Change Triggers Bus Read" is disabled, the bus will only be read when GPIO 0 changes with a button press, meaning that even though GPIO 1 toggles at a consistent rate, only changes that occur when GPIO 0 changes will be displayed.

Figure 4-34. GPIO 0 Option Enabled, GPIO 1 Option Disabled



4.5. View Power Data

Use the Time Plot to view power data, such as current and voltage measurements, and determine power consumption of the target application.

4.5.1. Basic Current Measurement

In order to make power measurements, you must use a tool that supports this features. For demonstration purposes, the Power Debugger will be used. For more information on this tool, see:

www.microchip.com/DevelopmentTools/ProductDetails/ATPOWERDEBUGGER

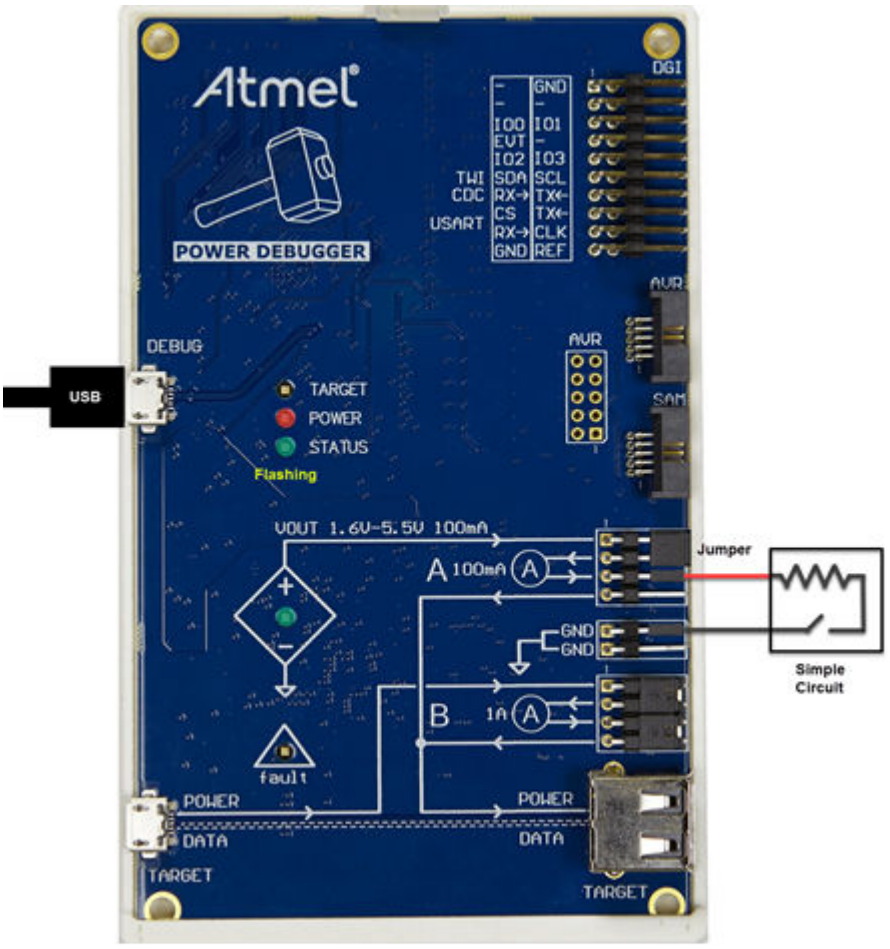
4.5.1.1. Hardware Setup

Connect the Power Debugger to the PC using the USB cable. The POWER LED should turn red.

Then connect to a target. A simple connection (shown in the figure below) is used to measure circuit current by using channel A. The circuit will be powered by the debugger's VOUT, which has been jumpered to A+. A- has been connected to the circuit. Another connection is from the circuit to board ground.

When MPLAB Data Visualizer senses the debugger and establishes communication, the STATUS LED will flash green. The level of VOUT can be set in the data visualizer and the VOUT LED will glow green.

Figure 4-35. Simple Connections



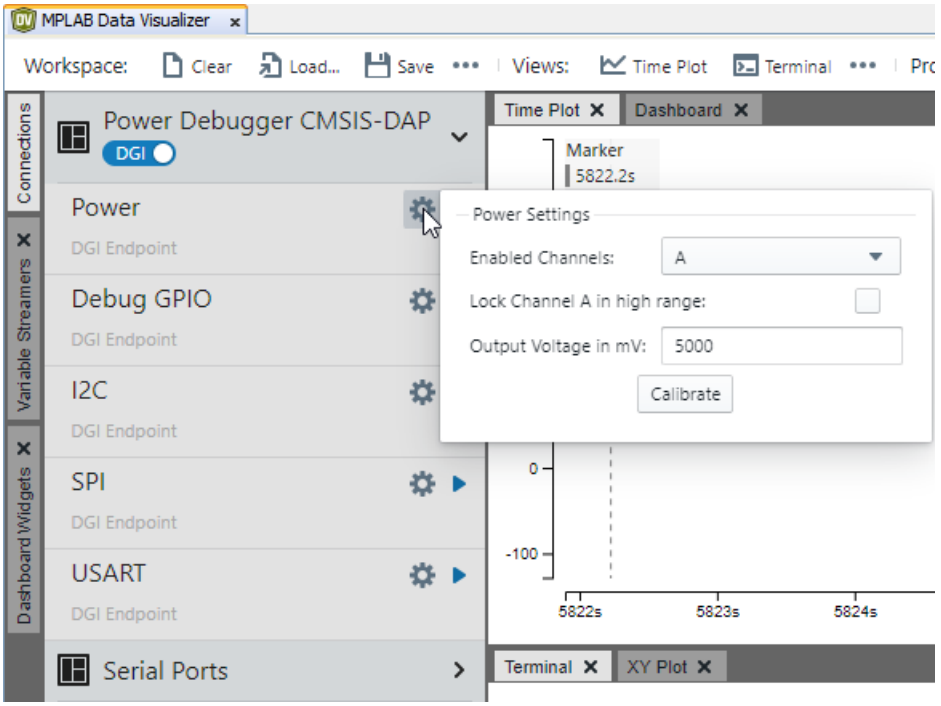
4.5.1.2. MPLAB Data Visualizer Setup

Launch the MPLAB Data Visualizer. When the tool is detected, a power selection will be available under the tool DGI connection (see figure below).

Click on the **Power Source** options icon and then enter Power Settings. Click on the icon again to save settings and close.

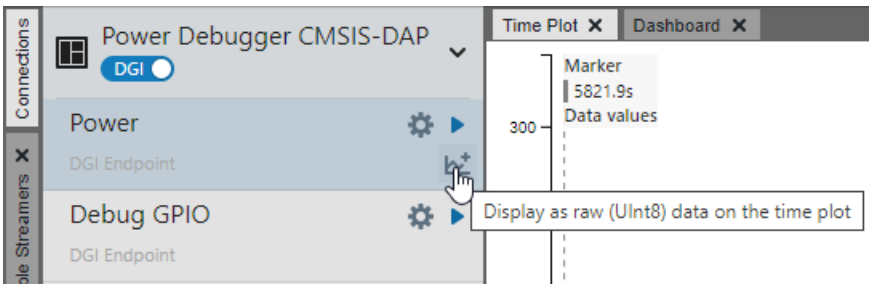
In the image below, the selections are Channel A for current sense and VOUT as 5000 mV (5V).

Figure 4-36. Power Selection Setup



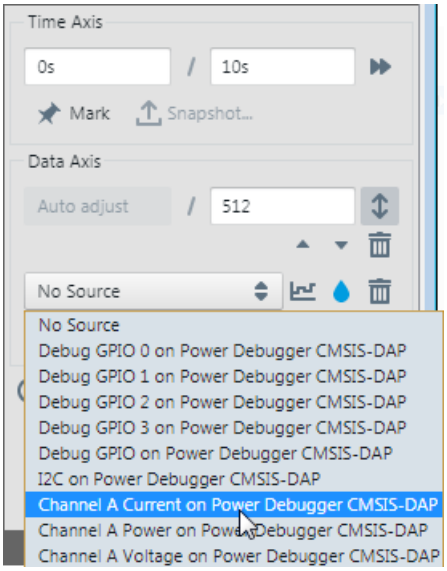
To see the power data plots, hover over the Power interface until the icon for displaying all sources on the time plot is visible and click on it.

Figure 4-37. Plot Power Sources



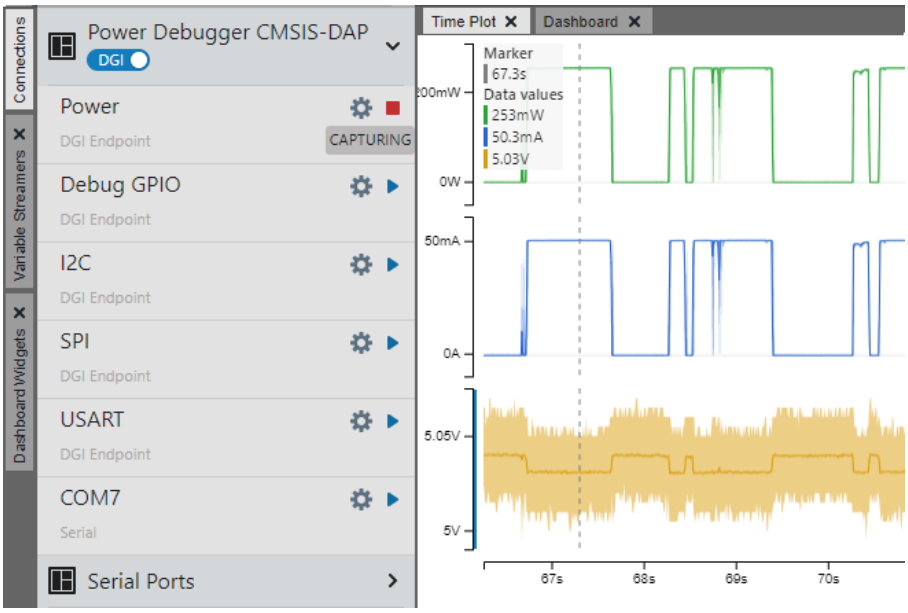
To select a single plot, select it from the plot options.

Figure 4-38. Plot Single Source



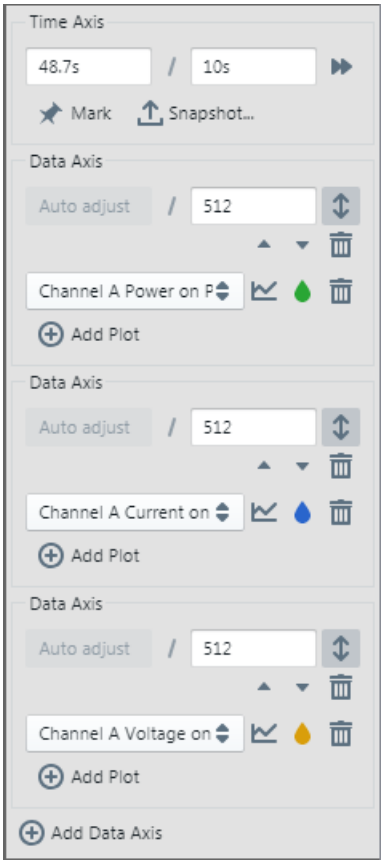
If all sources are plotted, by default the power plot will be first, the current plot second and the voltage plot third. For more on the noise in the third plot, see Noise in Power Plots.

Figure 4-39. Power Plot



To change plot formatting use the controls on the right pane.

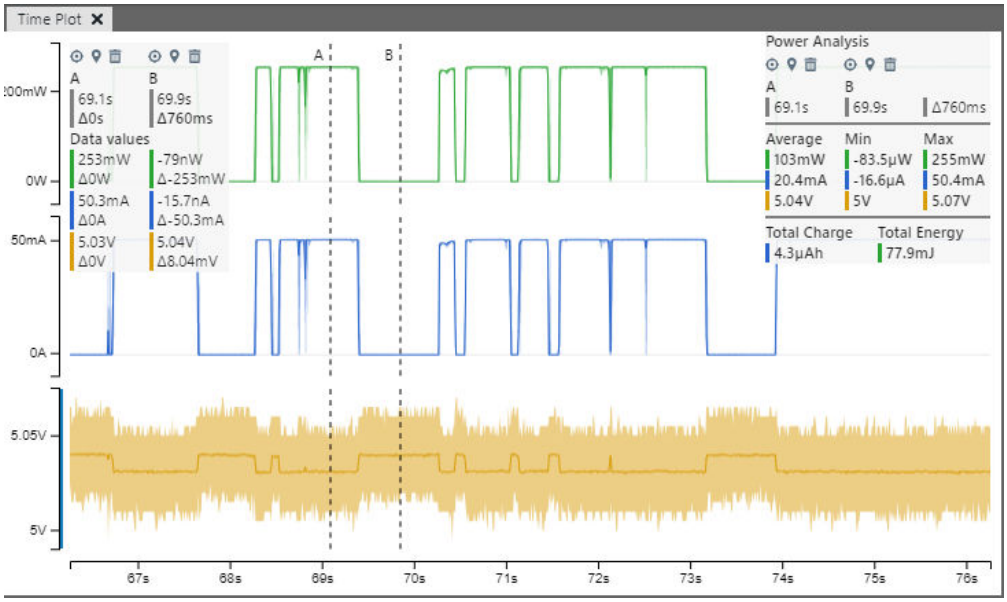
Figure 4-40. Plot Controls



4.5.1.3. Power Analysis using Cursors

In order to analyze the current and other power measurements more closely, select Power Analysis from the window toolbar . Two cursors (A and B) will appear, with the value inspector on the top left and Power Analysis data on the top right. Drag the cursors to see different values.

Figure 4-41. Values using Power Analysis



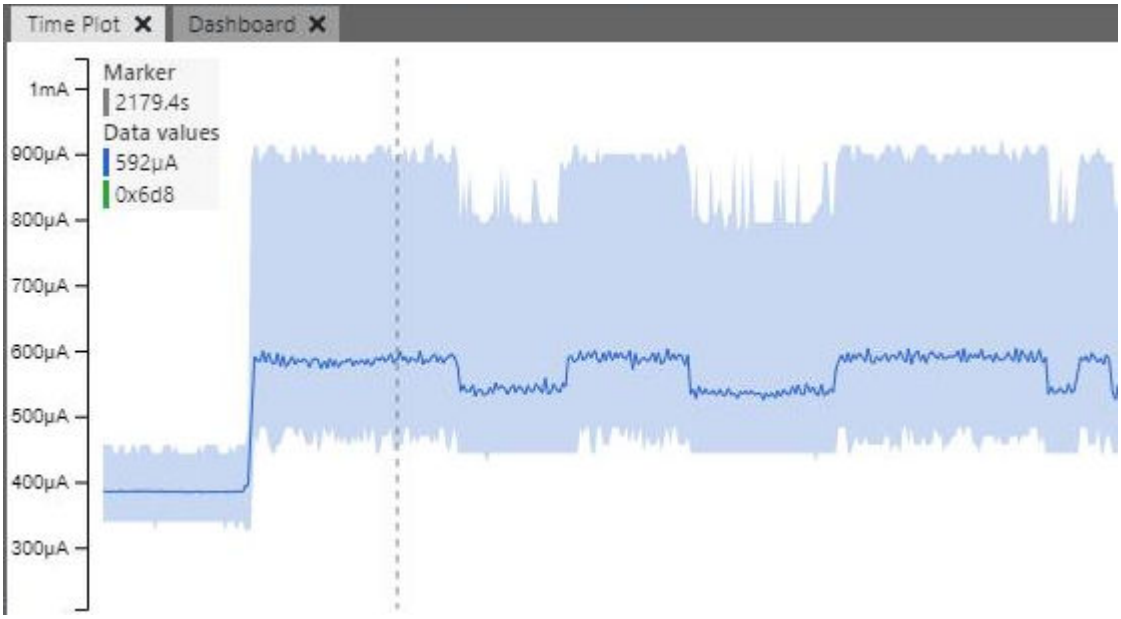
Related Links

Plot Cursors

4.5.1.4. Noise in Power Plots

When the target device itself consumes very little power, noise can be visible on power plots, as it becomes a relatively larger part of the measured power. This noise could be an effect of the target device interacting with a debugger in a debug session, or noise in the power supply or measurement circuits.

For example, in the current plot for low power consumption, noise is visible in the plot.



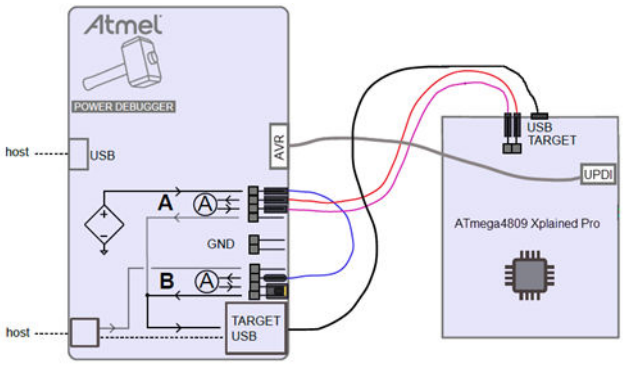
However in the current plot for higher power consumption, the noise is much less.

4.5.2. Multiple Current Measurements

More than one plot can be shown on each axis. To do this, both the A and B channels of the Power Debugger will be used to gather device and target power data from an ATmega4809 Xplained Pro target board.

4.5.2.1. Hardware Setup

Connect the Power Debugger to the target board according to the block diagram. Connections are described below.



To power the target: the Power Debugger variable voltage supply will be used. This enables testing a USB device over the full voltage range of the USB specification.

To do:

- Connect the variable voltage output to the input of the B channel using a strap cable. This enables measurement of the total current drawn by the target.
- Connect the output of the B channel to the POWER input of the USB type-A jack. This is simply done by using a jumper.
- Connect a USB cable from the type-A jack to the TARGET USB micro-jack.

To measure the current drawn by the target MCU: the Xplained PRO board has a power 2-pin header next to the TARGET USB connector. Usually a jumper is in place here connecting VCC_TARGET to VCC_MCU. By removing this jumper and routing this circuit through a current measurement channel we can measure the current drawn only by the MCU, and not the whole board.

To do:

- Connect VCC_TARGET on the target board to the input of the A channel.
- Connect the output of the A channel to the VCC_MCU pin of the target board.

To program and debug the target device: Although the Xplained PRO has debug capability, we will use the Power Debugger for this purpose.

To do:

• Connect the 10-pin debug cable from the AVR header of the Power Debugger to the UPDI DEBUG header of the Xplained PRO.

To connect all this to the host computer:

To do:

- Connect a USB cable from the DEBUG connector of the Power Debugger to the host computer.
- Connect a USB cable from the TARGET connector of the Power Debugger to the host computer.

Tip: GND connections are not necessary when the programming header is connected.

4.5.2.2. Debug and Visualize

The application used for the target is based on Example 2 from these code examples:

MPLAB XC8 User's Guide For Embedded Engineers AVR Code Examples

To use the code for the ATmega4809 Xplained Pro instead of the Atmega4809 Curiosity board, change “PORTD” to “PORTB” in the code.

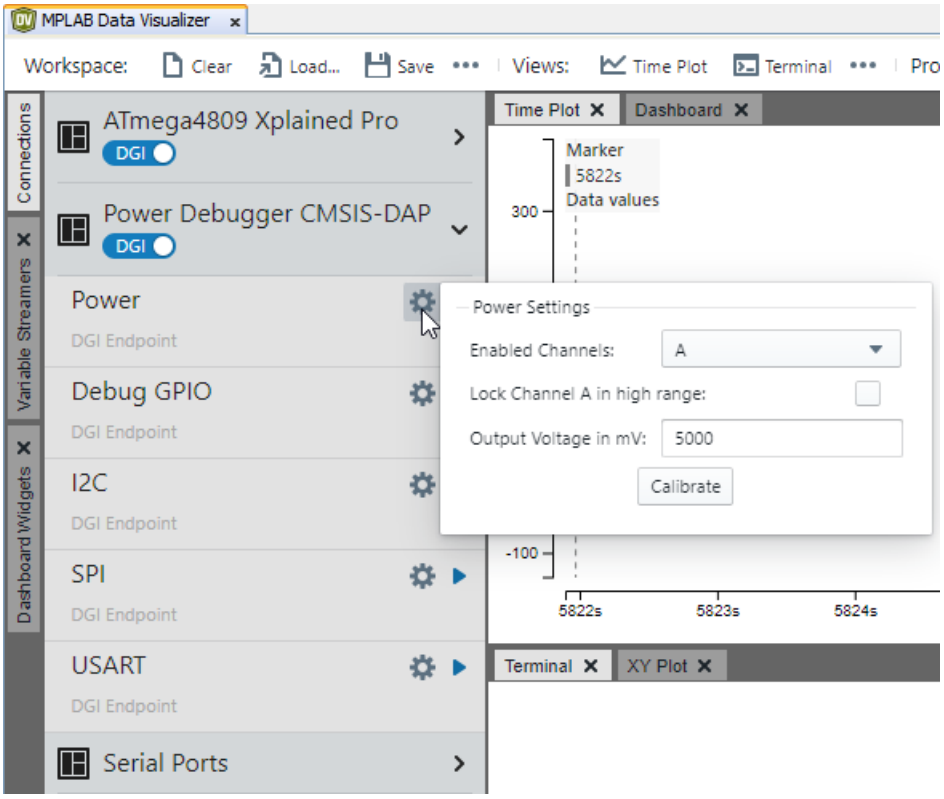
To do:

1. Open Example 2 in MPLAB X IDE v5.50 or later.
2. Make the code changes specified.

Open the MPLAB Data Visualizer in MPLAB X IDE.

To do:

1. Select *Window>Debugging>Data Visualizer*. This will open the MPLAB Data Visualizer.
2. The Data Visualizer should find the Power Debugger DGI interfaces. Click on the Power Source options icon and then enter Power Settings. Click on the icon again to save settings and close.

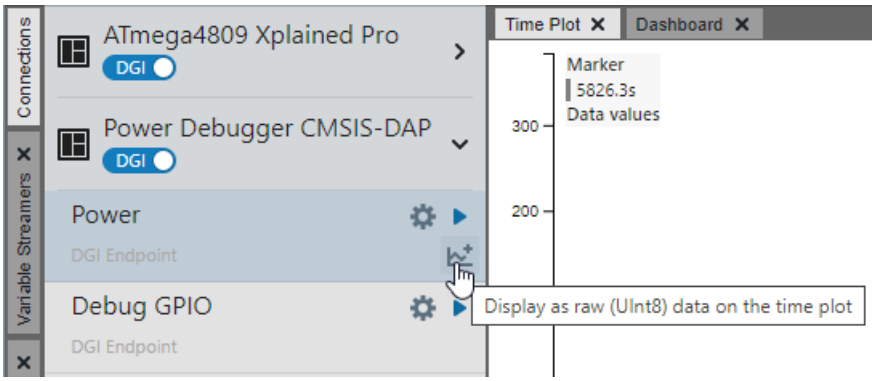


3. The Data Visualizer should find the ATmega4809 Xplained Pro as it is now powered. However, only Power Debugger connections will be used.

Now the example can be debugged and the output plotted in the visualizer.

To do:

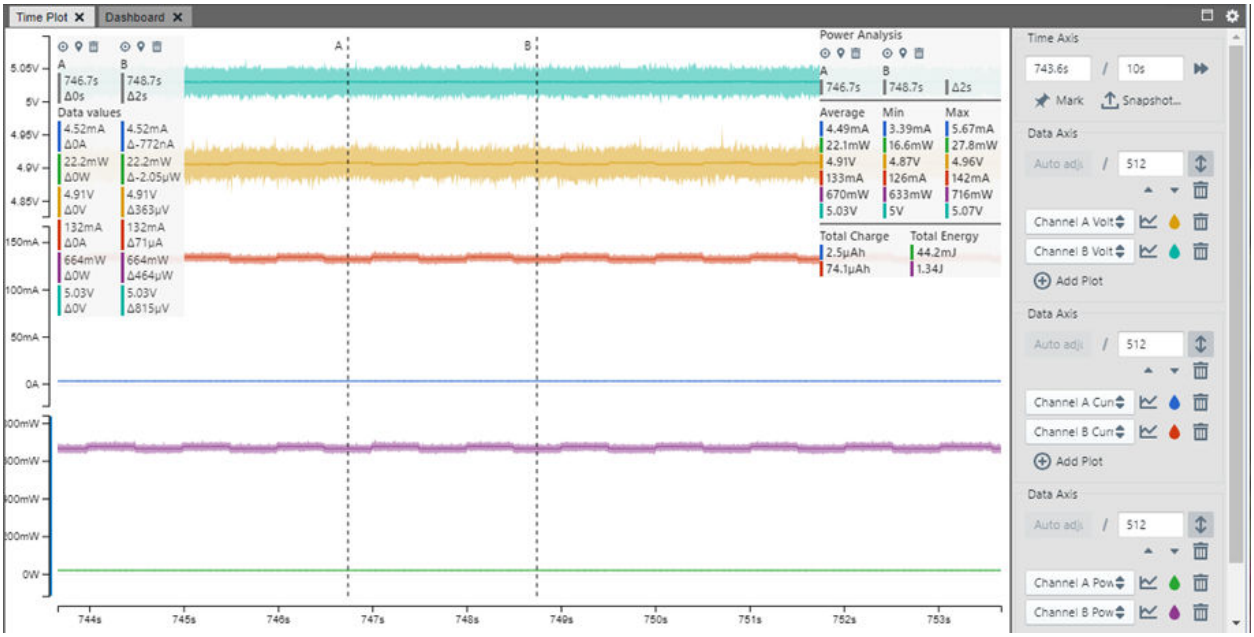
1. Debug the example. The User LED will flash on the Xplained Pro board when running.
2. On the Power interface, select to display all sources on the time plot.



4.5.2.3. Zoom In

The MPLAB Data Visualizer will show two power measurement plots for each graph. One is for the ATmega4809 device and the other is for the target board. Although you can use the Power Analysis option to help understand the data, it is difficult to see the plots clearly.

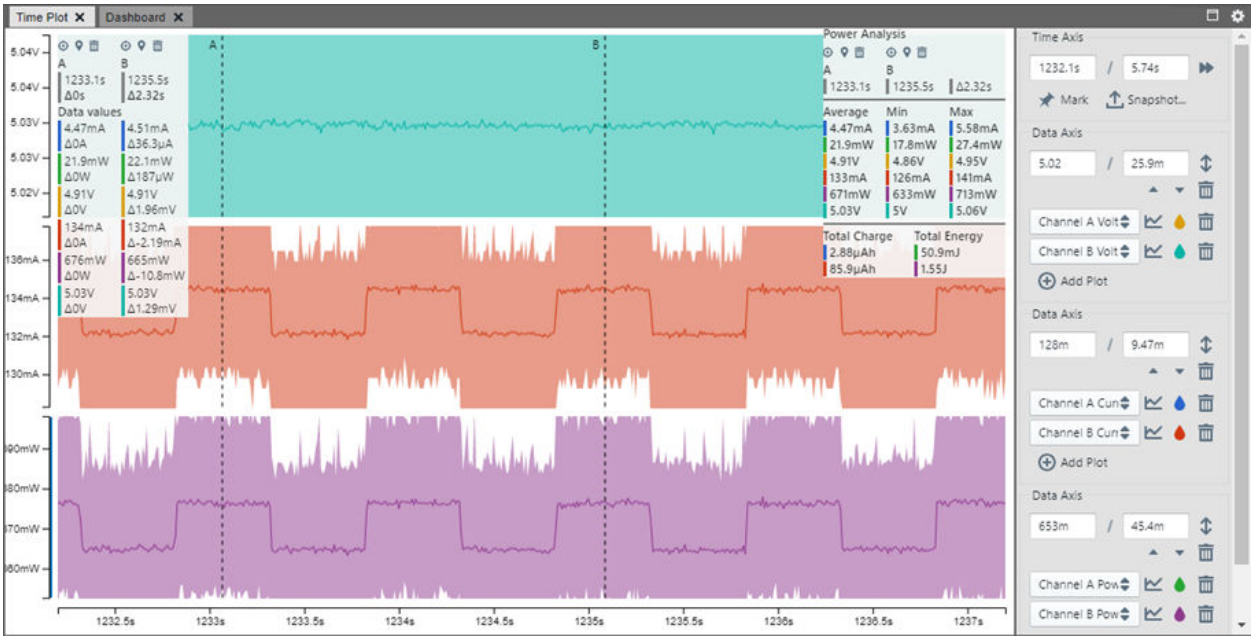
Figure 4-42. Plots of Channel A and B Power Data



To better see the plots, it may be useful to zoom in on the plots for each channel. Move the mouse cursor to the plot value axis and use the wheel to zoom in and out.

For more about noise in the plots, see Noise in Power Plots.

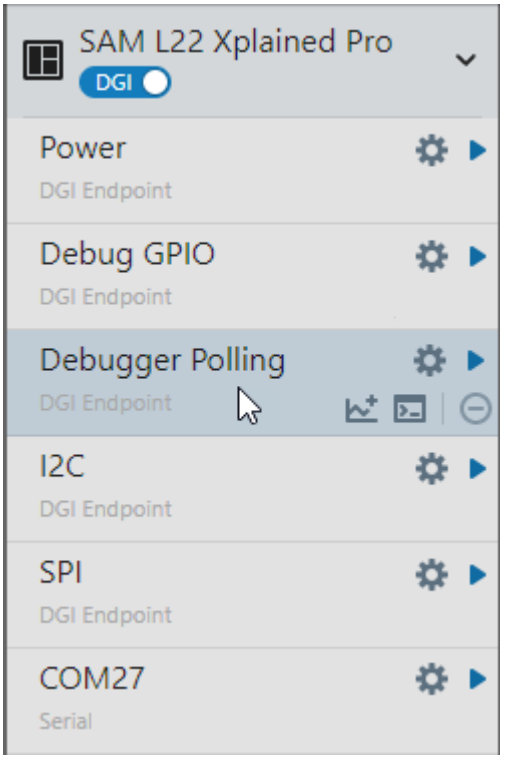
Figure 4-43. Zoom for Channel B Plots



4.6. PC Sampling

For selected debuggers and kits, the DGI section of the connections panel will contain a “Debugger Polling” interface section. Through this interface, the Data Visualizer can display sampled PC values in visualizations, e.g., as points in the time plot.

Figure 4-44. Debugger Polling Interface



Note: PC sampling is only available in the plugin version of MPLAB Data Visualizer, as it depends on resources in the debugger implementation in order to sample the PC (Program Counter).

Note: Debuggers, like MPLAB ICE 4 or the Power Debugger, always report the Debugger Polling interface as available, irrespective of which target device is being debugged. However, not all device architectures support PC sampling. In general, the PC sampling feature will work for all SAM devices as well as newer AVR devices.

The Data Visualizer can show PC values and, e.g., current measurements in the same graph. This allows you to see a correlation between consumed power and the executed code.

The sampled PC values will only show part of the code execution, as in most cases it is impossible to read out the PC values as fast as the target is executing instructions. The sampled values are still useful to indicate which code segment is being executed at any point in time. Each time the PC is read out from the target, we get the exact address of the code location currently being executed.

4.6.1. PC Sampling Example

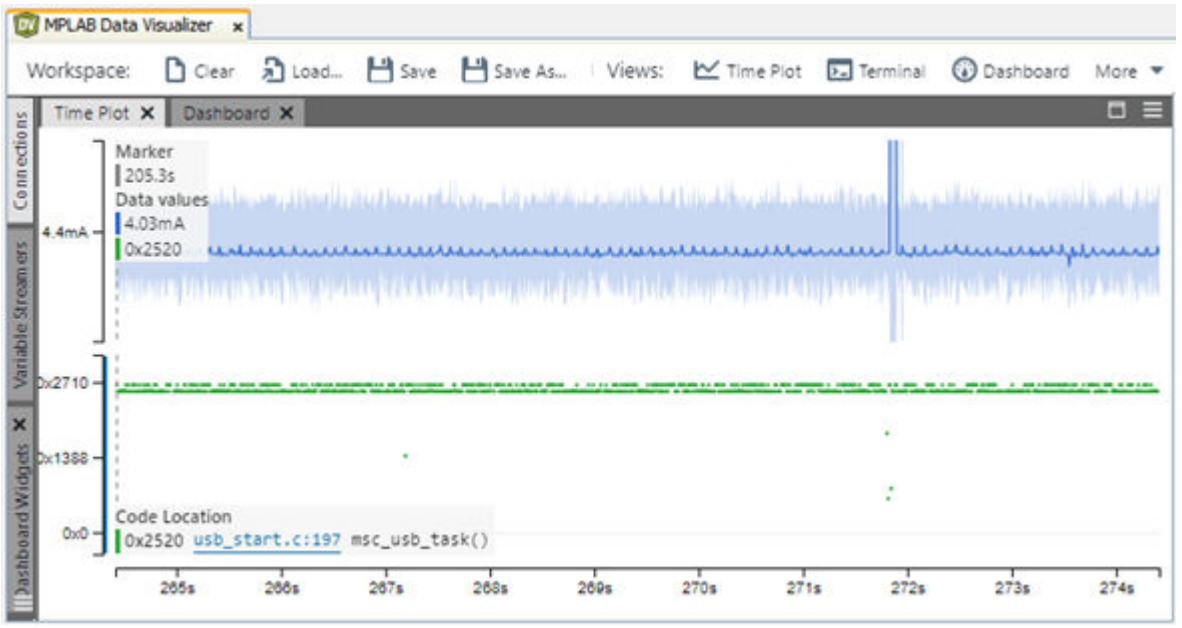
This section shows PC sampling along with Power Profiling in the same timeplot.

This example is based on an USB MSC example project from Start. We used the SAM L22 Xplained Pro kit, but the procedure is the same for other debuggers and kits that support PC sampling.

Make sure that debugging is started and open the Data Visualizer window.

Find the Debugger Polling interface in the list of DGI interfaces for the connected kit. Hover over the interface and plot all sources by clicking  . This will add the PC Sampling signal to the current timeplot (as shown in the figure below), in green. The figure also shows a plot of the current consumption, that has already been added, in blue.

Figure 4-45. How to add PC Sampling



In this particular example the mass storage device is listed as a drive in Windows Explorer. A"format" operation on the drive is performed while it is being profiled in the Data Visualizer. The image below shows a spike in power consumption during this operation. Also, some extra activity is shown by the sampled PC values at the same time.

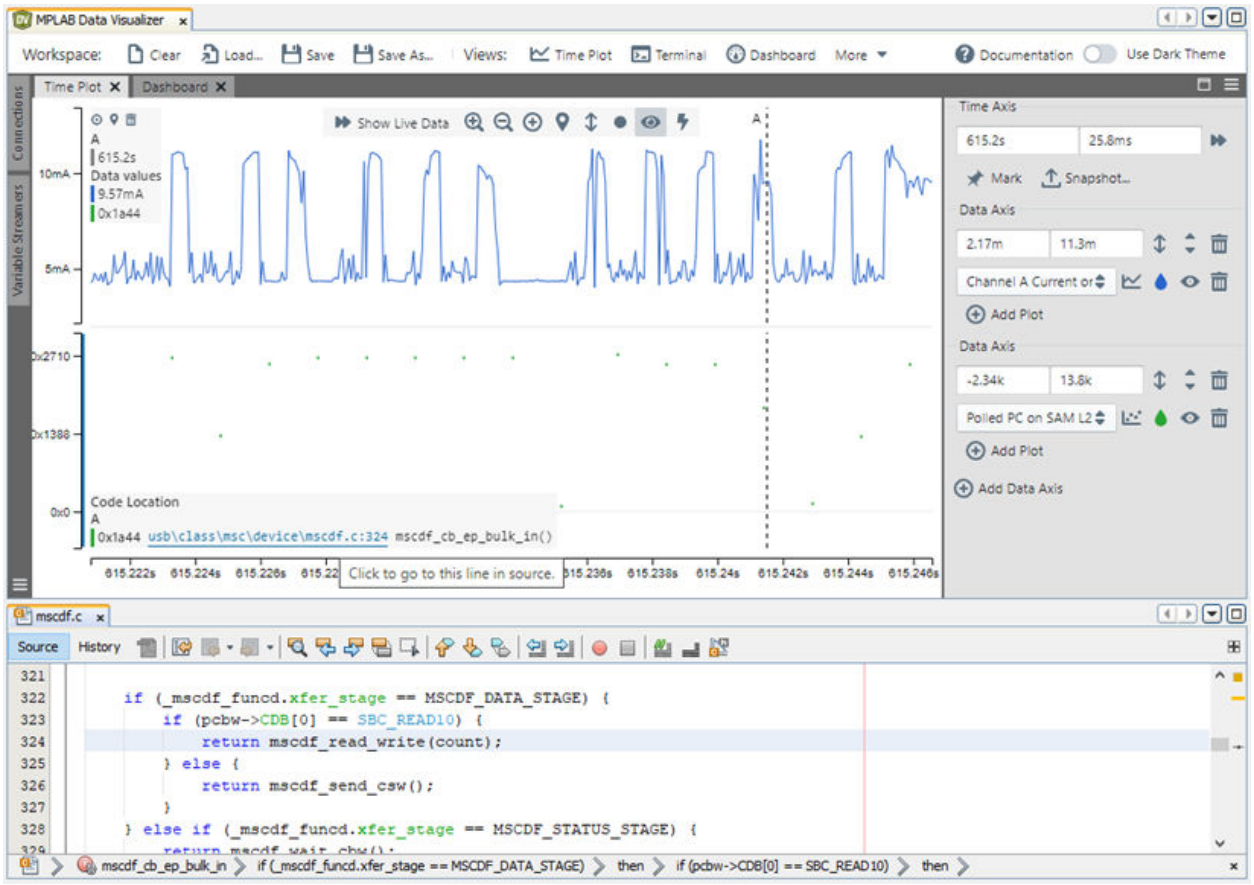
Figure 4-46. PC Sampling and Power Profiling

Zoom in on the area of the timeplot that shows the increased power consumption.

Make sure the "Inspect"  function is enabled. When a PC Sampling signal is being plotted, the PC value that is closest to the cursor will be shown in the inspect values panel at the left of the plot area. The source code location will be decoded (if possible) and shown in the lower part of the plot area. In this example a location with PC value equal to 0x1A44 is being decoded to a function related to the USB MSC handling.

By clicking on the source code link, the file will be opened with the cursor located on the corresponding source line.

Figure 4-47. PC Value and Source Code Location



4.7. View Two Sets of Data in the XY Plot

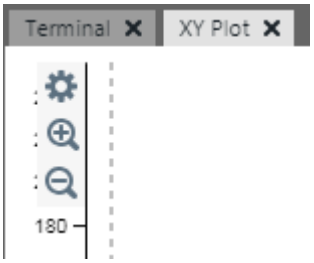
Instead of plotting data vs. time, plot one data set vs. another using the XY Plot. This is useful for inputting data for machine learning, which is discussed in an example.

4.7.1. XY Plot Window

The XY Plot window shows the x vs. y data plot provided by connected sources. The data points in this window are synchronized with the time axis on the Time Plot window. Viewing live data on the Time Plot window or dragging the time axis left or right will change the data shown here.

Use the XY Plot Visualization Controls (right) pane to select Axis options. Alternately, you can edit axis information from the graph ( ). Axis zoom in/out and a plot marker are also available for this graph.

Figure 4-48. XY Plot Controls



Additional functionality may be available from supported plug-in tools.

Related Links

Plot Marker

4.7.2. XY Plot Visualization Controls Pane

The XY Plot Visualization Controls (right) pane is for controlling visualization (graphing) of xy data.

Figure 4-49. Graph Visualization Controls

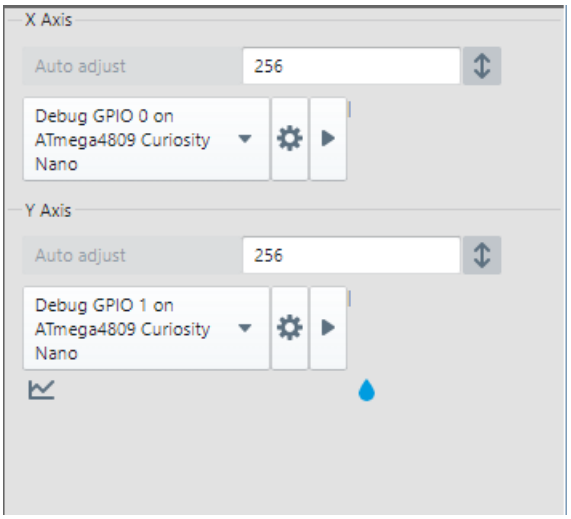


Table 4-6. Data Axis

Control	Description
Offset	When auto adjust enabled, “Auto adjust” displayed. When auto adjust disabled, current offset of plot shown.
Scale	Specify the resolution of the data axis in seconds.
	Auto adjust enable/disable. When enabled, automatically adjust range of axis. When disabled, manually adjust range of axis.

Table 4-7. Data Axis - Plot Source and Format

Control	Description
Data Source	Select the data source to plot from the drop down list. See the Data Sources pane for selection and setup.
	Select source settings.
/	Begin/stop streaming data. Clicking either x or y will begin/stop streaming for both.
	Click to select how data points are shown on the graph. • Connect points with stepped lines (default) • Connect points with straight lines • Only draw points, not lines
	Click to select a graph color.

4.7.3. XY Plot Example

Below is an example of use for the XY Plot and Time Plot.

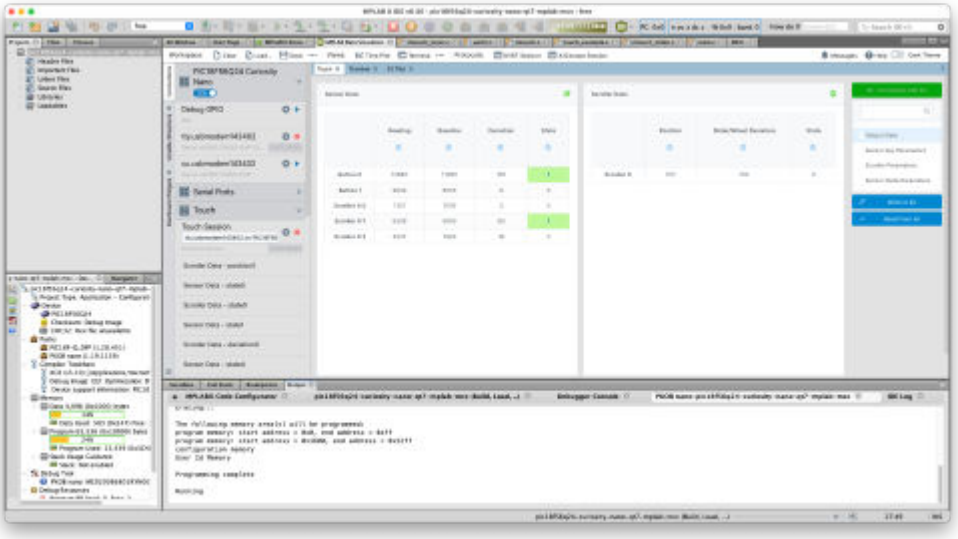
4.7.3.1. MPLAB Touch Plugin Example

The MPLAB Touch Plugin allows you to visualize touch tuning data within MPLAB Data Visualizer. The plugin supports two-way communication through which you can modify the touch configuration without having to rebuild the project - thus reduces development and tuning time.

To find supported plugins for MPLAB Data Visualizer, select *Tools>Plugins* in MPLAB X IDE. Locate the plugin, click "Install" and follow any instructions. After installing the plugin, a restart will be required. A Touch connection type will appear with any applicable connections below.

Go to the Introduction to Touch Plugin for more information.

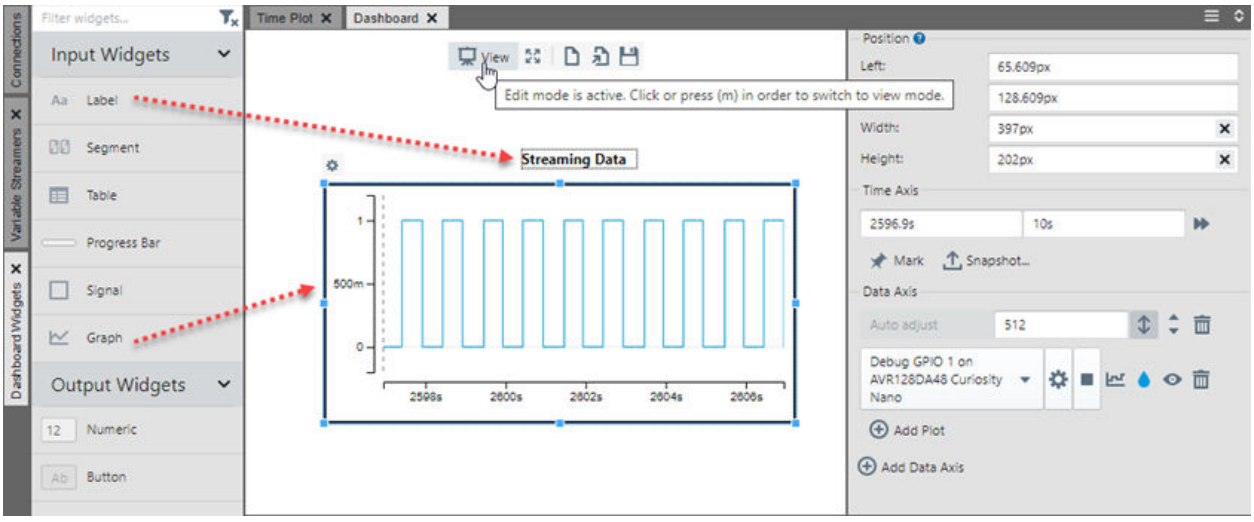
Figure 4-50. MPLAB Touch Plugin



4.8. View Data using Dashboard Widgets

The Dashboard is a customizable Graphical User Interface (GUI) that can be used to control and display parameters from application firmware. Widgets (button, label, slider, etc.) are dragged from the Dashboard Widgets tab (on the Data Sources pane) to the Dashboard window to form the GUI. Each widget can have a data field associated with it to send or receive values.

Figure 4-51. Dashboard



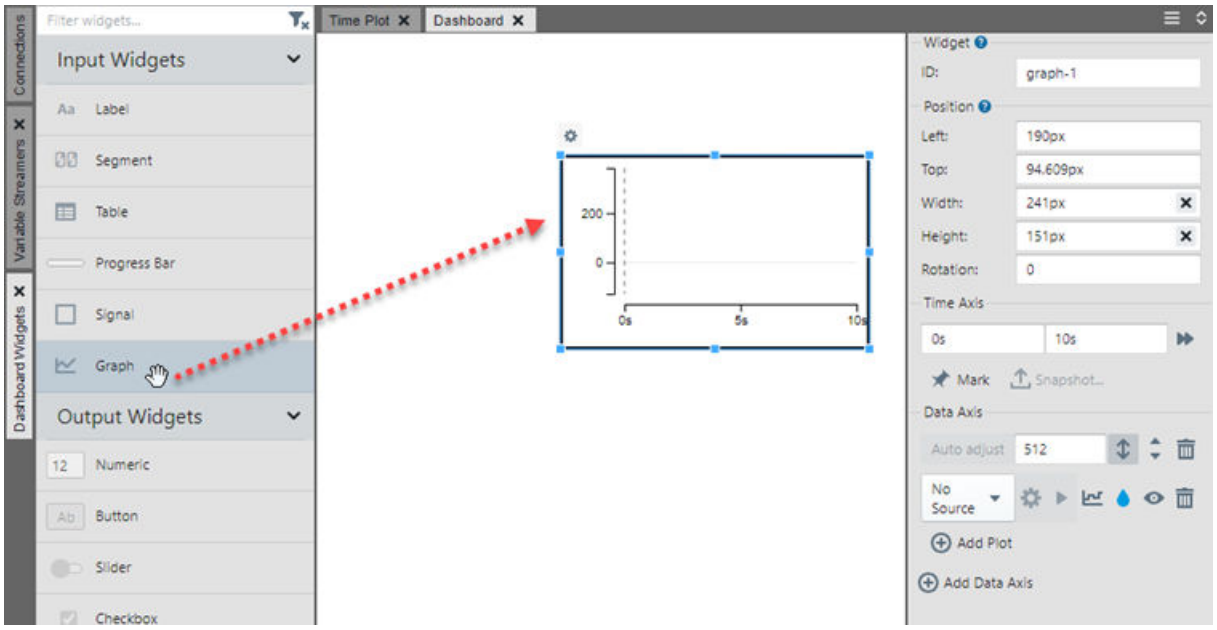
For an example on how to configure each widget, see the following sections.

4.8.1. Dashboard Window

The Dashboard window is an area onto which widgets or data sources can be dragged and used to control or display application parameters. Available widgets are shown on the Dashboard Widgets Tab.

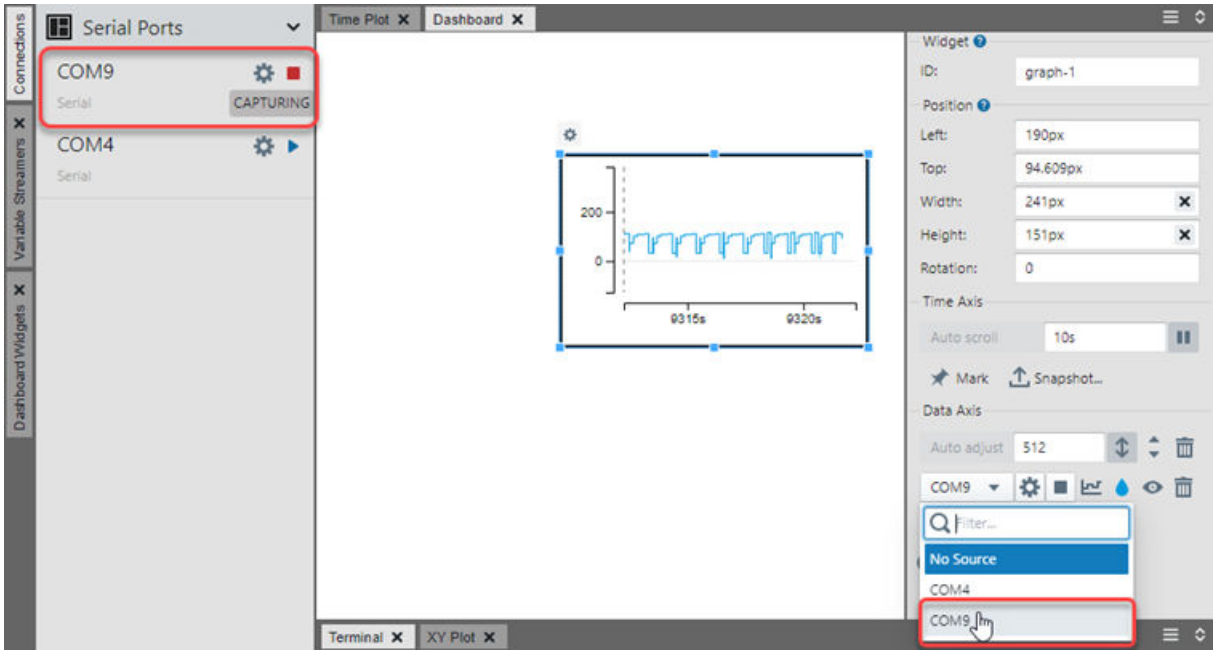
As an example, dragging the Graph (Input Widget) to the Dashboard results in widget graph-1 which may be set up by dragging the graph handles to resize and/or using the Widget Control Pane.

Figure 4-52. Drag Graph Widget to Dashboard Window



Click on the Connections tab to find the data source you want to connect to the widget. Then in the configuration panel, select that option from the drop down list.

Figure 4-53. Select Data Source for Graph



Note: Widgets can also be connected to each other and other data sources and sinks as described in Sending Values to and from Widgets.

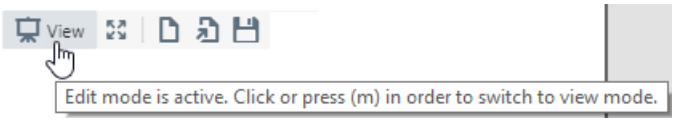
Related Links

Dashboard Widgets Tab

4.8.1.1. Dashboard Modes

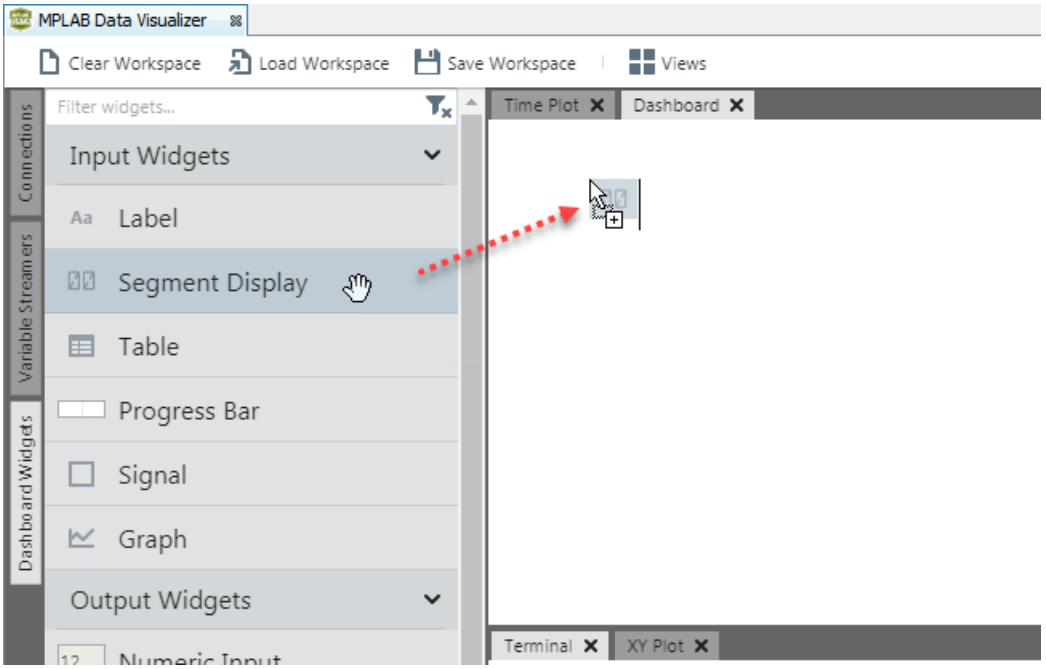
The Dashboard window has two modes: Edit and View. Click the mode on the menu bar to toggle to that mode.

You can add widgets in either mode, but you must be in Edit mode to edit widget properties.



4.8.1.2. Placing Widgets on the Dashboard

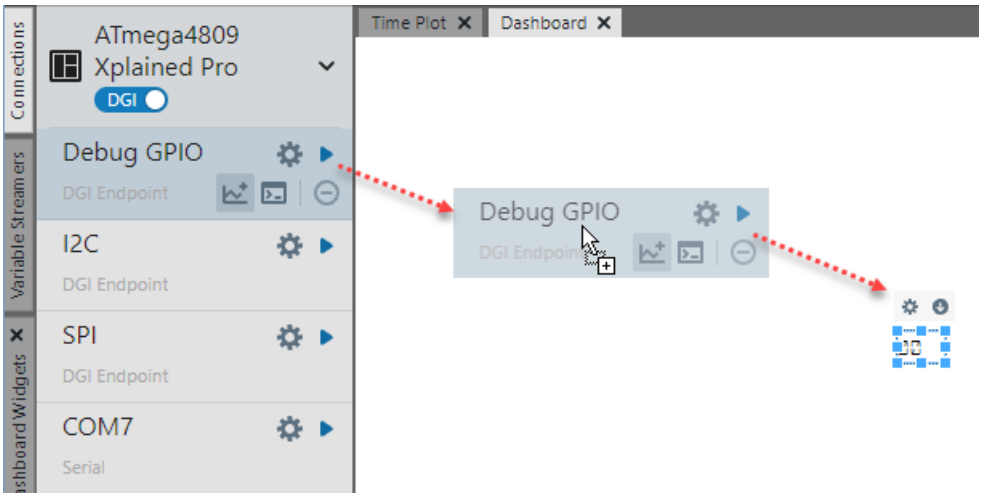
On the Data Sources pane, Dashboard Widgets tab, click, drag and drop a widget to add it to the Dashboard. This can be done only in Edit mode. For more on each widget, see Dashboard Widgets Tab.



Sources and variable streamer fields may also be dragged onto the window and will display as a Segment widget (see figure below). The segment will be automatically associated with the data source.

A streamer group will display as a Table widget. When Debug GPIO is dragged onto the Dashboard , the widget will show the GPIO bus value. Individual GPIO lines can be selected from the source drop down list. When Power is dragged onto the Dashboard, the widget will show Channel A Power. Other power data fields can be selected from the source drop down list.

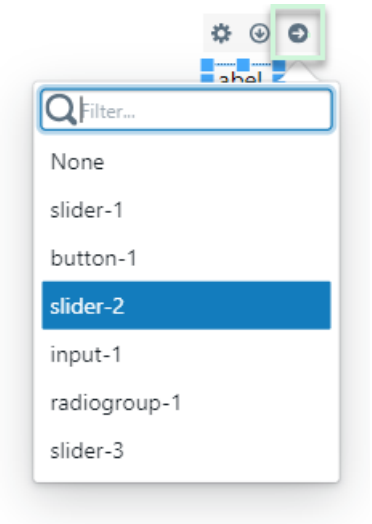
Figure 4-54. Drag Source to Dashboard Window



4.8.1.3. Sending Values to and from Widgets

To select where a widget receives or sends data, use the buttons that appear when clicking a widget on the Dashboard in Edit mode. If the widget can communicate with external connections, there is a vertical arrow button to enable the selection. In that case, it is also possible to drag a source onto the widget to configure it.

Figure 4-55. Widget ID Dropdown Menu



For input widgets, there is also a horizontal arrow button enabling receiving data from output widgets. Selectable widgets are displayed as their widget ID. This ID can be found by hovering over widget in Edit mode. It is also shown in the widget configuration options described in Configuring Widget Properties.

Figure 4-56. Locating the Widget ID by Hovering on the Widget

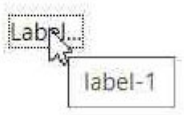
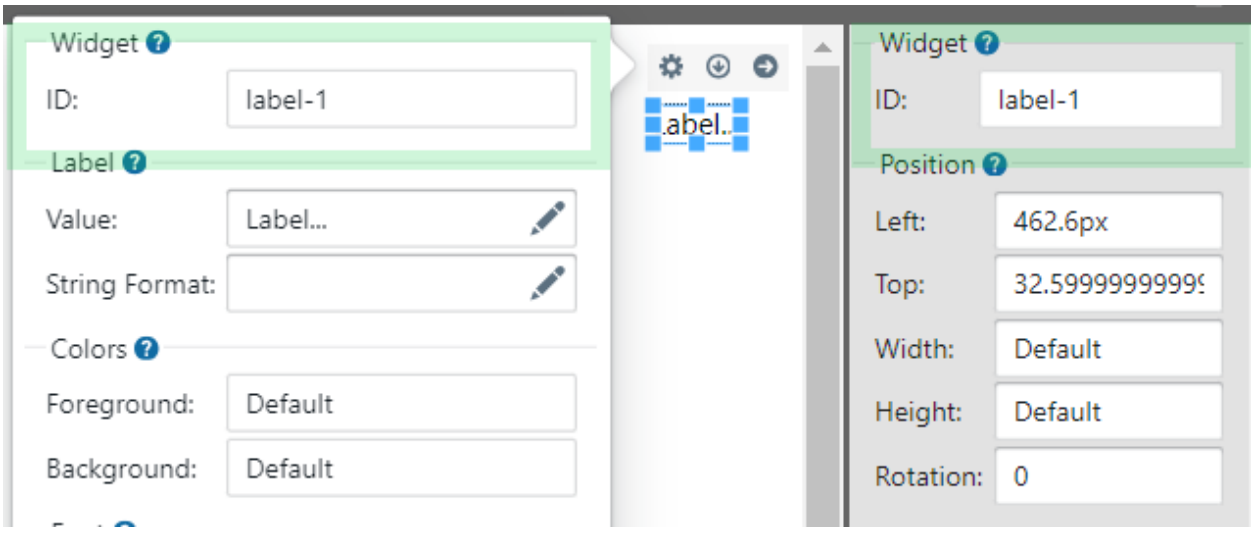


Figure 4-57. Locating the Widget ID from the Widget Configuration Options



Input widgets can only be connected to one external connection or output widget at a time. Output widgets can be connected to a single external connection, however several input widgets can receive

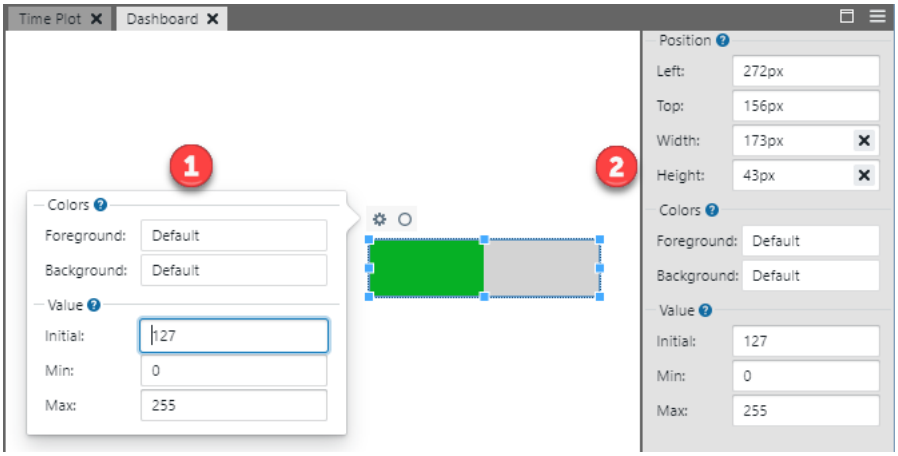
data from the same output widget. The selected connection or widget is highlighted. To disconnect, select the connection named **None**.

4.8.1.4. Configuring Widget Properties

All dashboard widgets can be configured when in Edit mode. The parameters will vary depending on widget type, but the methods for changing them are the same. See the figure for an example of a Progress Bar.

1. **Options Icon Method**: Click to select the widget. Then click the gear icon to pop up widget configuration options.
2. **Control Pane Method**: Click to select the widget. Then view Position and Configuration options on the Widget Control Pane (left of the Dashboard).

Figure 4-58. Widget Configuration



4.8.1.5. Moving and Resizing Widgets

All parameters related to position and size are available on the Widget Control Pane under “Position.” Widget position and size can be specified as either absolute or relative. To specify an absolute value, use the suffix "px." To specify relative to dashboard size, use the suffix “%.”

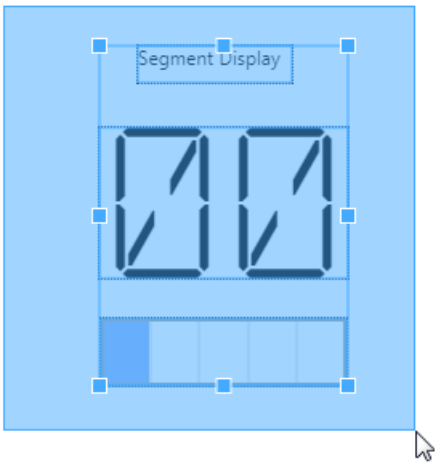
Figure 4-59. Absolute Position and Size

Figure 4-60. Relative Position and Size

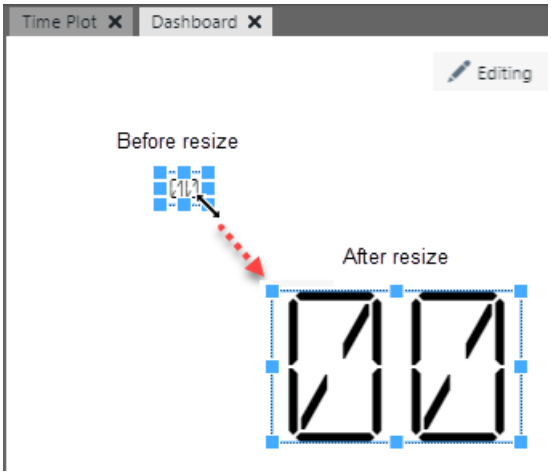
Widgets can also be dragged and dropped or resized on the Dashboard.

Note: Moving or resizing a widget will make Position values absolute (px).

- Move a widget or group of widgets to another location by clicking to select it, dragging it elsewhere on the Dashboard, and releasing it at that location. To select a group of widgets, click outside the group and drag around the widgets you want to group.



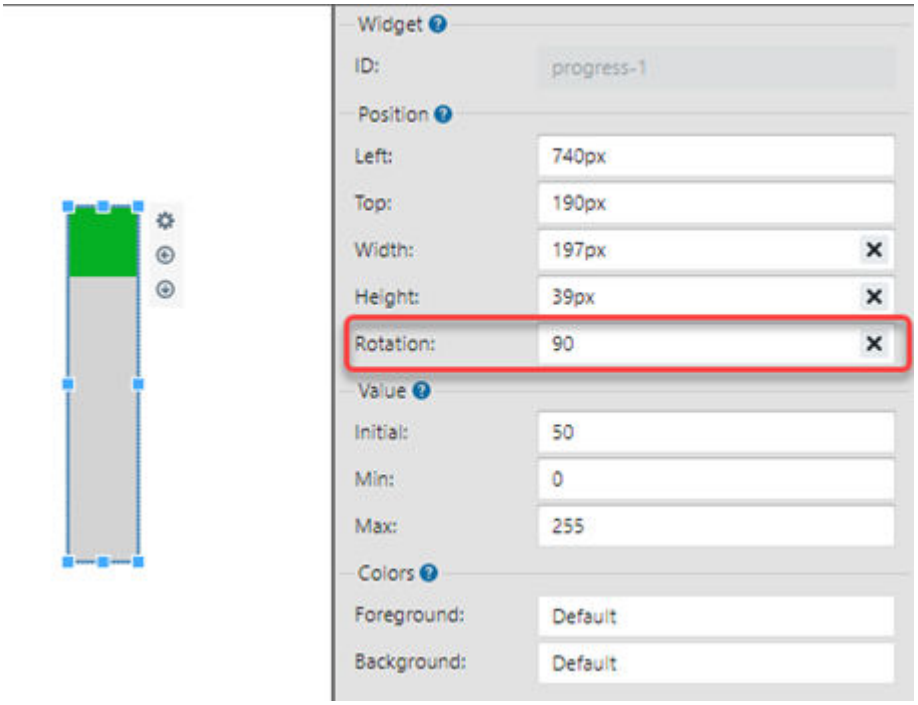
- Resize a widget or group of widgets by clicking to select it and then clicking and dragging a corner to resize.



4.8.1.6. Rotating Widgets

All widgets have a general rotation option. A setting option called **Rotation** is available in the widget settings pane, in the **Position** group. This sets the clockwise rotation of the widget in units of degree around the center of the widget box. The default is 0.

Figure 4-61. Rotation Option in the Widget Settings Pane



4.8.1.7. Deleting Widgets

In Edit mode, delete any Dashboard widget or group of widgets by clicking to select it and then hitting <Delete>. To select a group of widgets, click outside the group and drag around the widgets you want to group.

Note: This action is permanent and all configuration is lost after deletion.

4.8.1.8. Clearing, Importing or Exporting the Dashboard

To perform these Dashboard actions, you must be in Edit mode.

Table 4-8. Dashboard Options

Dashboard Toolbar Icon	Description
Clear Dashboard	Clear the entire contents of the Dashboard. This cannot be undone.
Import Dashboard	Import a previously saved Dashboard file (`.json` or `.db` file). After import, widget(s) source settings* will need to be selected.
Export Dashboard	Export the current Dashboard to a file (`.json` file.) Only the widget(s) are exported; source settings* are not.
* To preserve widget source settings, use Workspace Load or Save.	

4.8.1.9. Expanding the Dashboard

To expand the Dashboard, you must be in Edit mode.

To expand the Dashboard to full screen, click Toggle Fullscreen on the Dashboard menu or press “f” on the keyboard. To collapse back, repeat.

4.8.1.10. Tips and Tricks

The following information could be helpful when using the Dashboard:

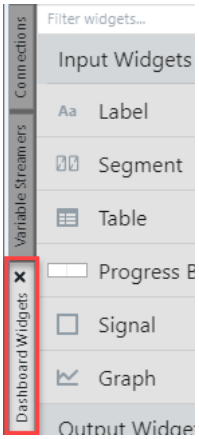
- When colors are default or cleared, most widgets will adapt to the light or dark theme selection . If colors are set, then the theme will not affect the colors. However, it is always possible to change or clear colors.

- When resizing a widget with text, the text may overflow the bounds of the widget. There is a "text wrapping" setting for the text widgets that changes how the text overflows. Nevertheless, the text will overflow if the widget is too small.
  To prevent text overflow, clear the width and height property of the widget (the 'x' on the right of those properties in the options.) The text widget will then size to adapt to the content.

4.8.2. Dashboard Widgets Tab

Click the **Dashboard Widgets Tab** to view, drag, and use available widgets in the Dashboard window. After a widget has been dropped on the dashboard, data fields (sink or source) can be selected to specify how the widget will be used, i.e., as inputs for streaming data or outputs to applications. For more specifing widget data source, see Dashboard Window and Sending Values to and from Widgets.

Figure 4-62. Dashboard Widgets Tab



All widgets have position parameters in common, as listed in the table below. The following sections will list only the configuration parameters specific to each widget.

Table 4-9. Common Widget Parameters - Position

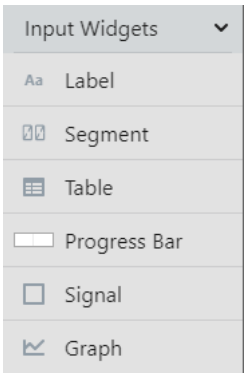
Parameter	Type	Usage
Left	Numeric	Horizontal placement in pixels or percentage.
Top	Numeric	Vertical placement in pixels or percentage.
Width	Numeric	Width of widget in pixels or percentage.
Height	Numeric	Height of widget in pixels or percentage.
Rotation	Degrees	Clockwise rotation of the widget around the center of the widget box.

Related Links

Dashboard Window

4.8.2.1. Input Widgets

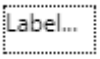
Figure 4-63. List of Input Widgets



Input Widgets accept input as a data field sink.

4.8.2.1.1. Label

The Label widget displays a text string.



Data Fields

The Label widget has a sink data field that accepts all types of sources. Any data sent to the label will be converted to a string and displayed as text.

Configuration

Hover over parameters in visualizer for more information.

Table 4-10. Label Group

Parameter	Type	Usage
Value	Text	Enter text for label.
String Format	Text	Enter values to change string format

The “Value” field is the initial value of the label. It is displayed if the widget is not connected to a data source, or before any data has been received.

The “String Format” field specifies the formatting that will be used when connected to a data source and data has been received. For example, “Voltage level is %0.1f V” will be formatted as “Voltage level is 3.1 V” once data has been received from the connected data source. For supported formatting parameters, refer to the table below.

Table 4-11. String Format Parameters

Parameter	Usage
%	Displays the % character
b	Displays integer as a binary number

Table 4-11. String Format Parameters (continued)

Parameter	Usage
c	Displays integer as ASCII value
d	Displays integer as signed decimal number
i	Displays integer as signed decimal number
e	Displays a float using scientific notation
u	Displays integer as unsigned decimal number
f	Displays float as is
g	Displays float as is
o	Displays integer as octal number
t	Displays value as true or false
x	Displays integer as a lower-case hexadecimal number
X	Displays integer as an upper-case hexadecimal number

Table 4-12. Colors Group

Parameter	Type	Usage
Foreground	Color selection dialog	Color of widget foreground.
Background	Color selection dialog	Color of widget background.

Table 4-13. Font Group

Parameter	Type	Usage
Font Size	Numeric	Enter font size
Font Weight	Dropdown box	Select the font weight (such as bold)
Font Style	Dropdown box	Select the font style (such as italics)
Font Family	Dropdown box	Select the font family (such as segment)

Table 4-14. Alignment Group

Parameter	Type	Usage
Horizontal	Dropdown box	Select the horizontal alignment of content
Vertical	Dropdown box	Select the vertical alignment of content
Line Wrap	Dropdown box	Select conditions to wrap line of content

4.8.2.1.2. Segment

The Segment widget element simulates a hex-digit LED display.

Data Fields

The Segment widget element has a sink data field that accepts all numeric data types. The value received is displayed.

Configuration

Hover over parameters in visualizer for more information.

Table 4-15. Segment Group

Parameter	Type	Usage
Min. Digits	Numeric	Enter the number of digits to display. Default = 2.
Num. Base	Numeric	Enter the number of the base. Default = 10 (decimal).

Table 4-16. Colors Group

Parameter	Type	Usage
Foreground	Color selection dialog	Color of widget foreground.
Background	Color selection dialog	Color of widget background.

Table 4-17. Font Group

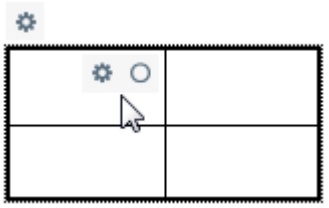
Parameter	Type	Usage
Font Size	Numeric	Enter font size
Font Weight	Dropdown box	Select the font weight (such as bold)
Font Style	Dropdown box	Select the font style (such as italics)
Font Family	Dropdown box	Select the font family (such as segment)

Table 4-18. Alignment Group

Parameter	Type	Usage
Horizontal	Dropdown box	Select the horizontal alignment of content
Vertical	Dropdown box	Select the vertical alignment of content
Line Wrap	Dropdown box	Select conditions to wrap line of content

4.8.2.1.3. Table

The Table widget displays one or more data sources in a table. Two modes are supported: Manual Labels and Auto Labels. In Manual Labels mode each cell can be manually configured to either be a Label cell or a Data cell. In Auto Labels mode, each cell is split into two fields, where the field to the left is a label with the name of the data stream and the field to the right is the actual data of the stream. The mode is selected by the check box named Auto Labels in the configuration.



Manual Labels

When using Manual Labels mode each cell either is a Label cell or a Data cell. By default all cells are Data cells. Label cells can be configured by setting the Label value (see figure below).

Only the Data cells have data fields.

Auto Labels

Each cell is associated with one data source and the name of the data source is shown to the left in the cell and the actual data to the right. The source name is automatically fetched from the source connected to the sink data field.

Delta1	0	**Delta6**	0
Delta2	0	**Delta7**	0
Delta3	0	**Delta8**	0
Delta4	0	**Delta9**	0
Delta5	0	**Delta10**	0

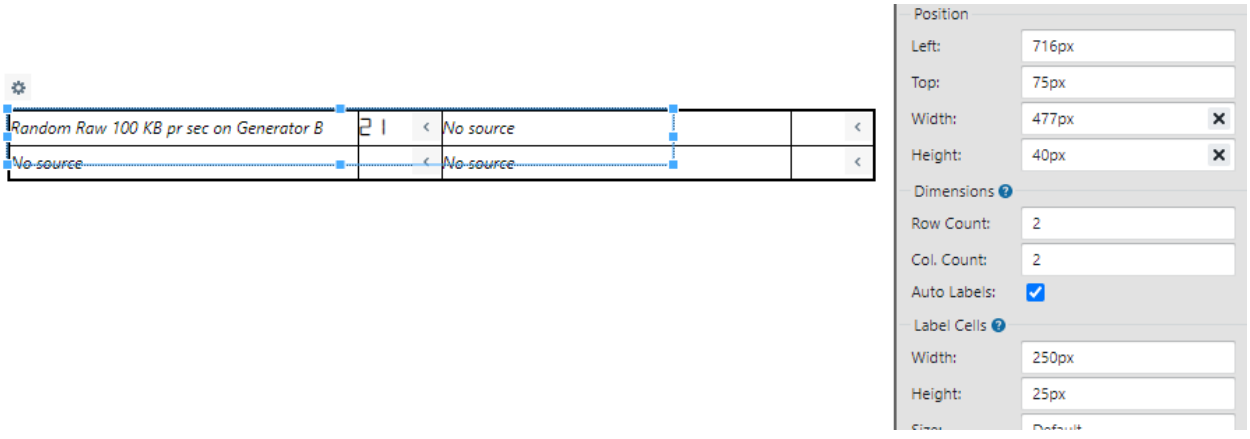
The Table element has one data field per table cell accepting any data source. The data will be converted to a string and displayed as text.

Resize Table

Cell sizes are treated as a minimum size. If the table is resized to be larger than the sum of cell sizes, the cells grow proportionally. If the table is shrunk to be smaller than the sum of cell sizes, the table will maintain a minimum size and **overflow the widget**, as shown below.

It is possible to have the table widget dynamically resized by clearing the width and height values (clicking on the 'x' button for those values).

Figure 4-64. Table after resize - overflow



Configuration

Hover over parameters in visualizer for more information.

Table 4-19. Dimensions Group

Parameter	Type	Usage
Row Count	Numeric	Number of rows in the table
Col. Count	Numeric	Number of columns in the table
Auto Labels	Check box	Check to use Auto Labels

Table 4-20. Colors Group

Parameter	Type	Usage
Foreground	Color selection dialog	Color of widget foreground.
Background	Color selection dialog	Color of widget background.

Table 4-21. Label Cells, Data Cells Group

Parameter	Type	Usage
Min. Width	Numeric	Width of the label or data part of each cell. Note that changing this width will change the total width of the table.
Min. Height	Numeric	Height of the label or data part of each cell. Note that changing this height will change the total height of the table.
Font Size	Numeric	Enter font size.
Font Weight	Drop down box	Select from Normal or Bold.
Font Style	Drop down box	Select from Normal or Italics.
Font Family	Drop down box	Select Segment.
Horizontal	Drop down box	Select the horizontal alignment of cell contents.

4.8.2.1.4. Progress Bar

The Progress bar widget is a linear bar that shows the position of a value between the Min and Max values.

Data Fields

The Progress bar widget has a sink data field that accepts all numeric data types. When a value is received, it will update the amount of colored area of the progress bar depending on the Min and Max values.

Configuration

Hover over parameters in visualizer for more information.

Table 4-22. Value Group

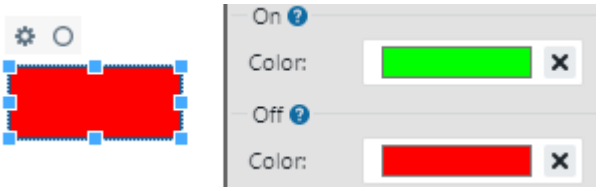
Parameter	Type	Usage
Initial	Numeric	Initial value
Min	Numeric	Minimum value of input
Max	Numeric	Maximum value of input

Table 4-23. Colors Group

Parameter	Type	Usage
Foreground	Color selection dialog	Color of widget foreground.
Background	Color selection dialog	Color of widget background.

4.8.2.1.5. Signal

The Signal widget is a simple color-based ON/OFF indicator.



Data Fields

The Signal widget has a sink data field that accepts all data types. The color of the signal is decided by a boolean evaluation, if the incoming value is a number it is true if it is greater than 0.

Configuration

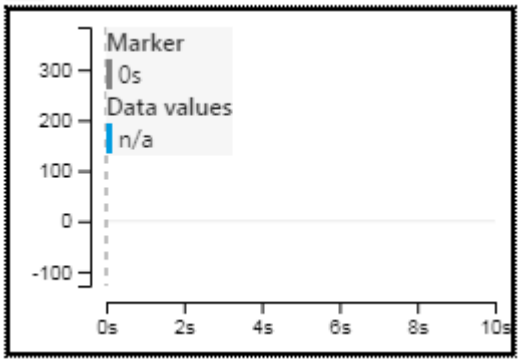
Hover over parameters in visualizer for more information.

Table 4-24. On and Off Group

Parameter	Type	Usage
On Color	Color selection dialog	Color of widget “on” state.
Off Color	Color selection dialog	Color of widget “off” state.

4.8.2.1.6. Graph

The Graph widget plots the incoming data streams in a two-dimensional graph. The graph can be configured to accept zooming and scrolling by mouse interaction or to be static ignoring any mouse interaction.



Data Fields

Add a data source to the plot as with a Time Plot.

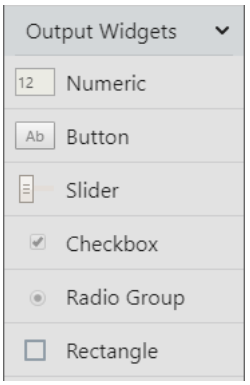
Configuration

See View Data in the Time Plot.

Note: In dashboard Edit Mode, some of the functions and UI elements of the graph widget are disabled in order to minimize interference with mouse events related to positioning the widget. In dashboard View Mode, all graph functions work normally.

4.8.2.2. Output Widgets

Figure 4-65. Output Widgets List



Output widgets will send a data value to the connected sink.

4.8.2.2.1. Numeric

The Numeric widget enables input of numeric values to the dashboard.

Data Fields

The Numeric widget has a source data field. Each time the numerical input value is changed a packet will send a value.

Configuration

Hover over parameters in visualizer for more information.

Table 4-25. Value Group

Parameter	Type	Usage
Initial	Numeric	Initial value
Min	Numeric	Minimum value of input
Max	Numeric	Maximum value of input

Table 4-26. Colors Group

Parameter	Type	Usage
Foreground	Color selection dialog	Color of widget foreground.
Background	Color selection dialog	Color of widget background.

Table 4-27. Font Group

Parameter	Type	Usage
Font Size	Numeric	Enter font size
Font Weight	Dropdown box	Select the font weight (such as bold)

Table 4-27. Font Group (continued)

Parameter	Type	Usage
Font Style	Dropdown box	Select the font style (such as italics)
Font Family	Dropdown box	Select the font family (such as segment)

Table 4-28. Alignment Group

Parameter	Type	Usage
Horizontal	Dropdown box	Select the horizontal alignment of content

4.8.2.2.2. Button

The Button widget will send a value each time it is pressed. The button can either be configured as a normal push button or as a toggle button. The button can have a static text to indicate its functionality.

When it is configured as a toggle button the text will be replaced by ON or OFF depending on the state of the button. To replace the ON/OFF text by something else the Text parameter must be given as '/' delimited text with the first part of the text being the ON state text and the second part the OFF state text. For example, "CLICKED/UNCLICKED."

Data Fields

The Button widget has a source data field. Each time the button is pressed a packet is sent. The value of the packet will always be:

- 0 for a normal push button
- Either 0 for a toggle button in its OFF state or 1 for a toggle button in its ON state. So if a button in the OFF state is pressed, the value will change to the ON state and vice versa.

Configuration

Hover over parameters in visualizer for more information.

Table 4-29. Button Group

Parameter	Type	Usage
Label	Text	Enter label text.
Mode	Dropdown box	Select Fixed or Toggle.
Value	Numeric	If Mode is Fixed, Enter the value sent when button pressed.

Table 4-30. Colors Group

Parameter	Type	Usage
Foreground	Color selection dialog	Color of widget foreground.
Background	Color selection dialog	Color of widget background.

Table 4-31. Font Group

Parameter	Type	Usage
Font Size	Numeric	Enter font size
Font Weight	Dropdown box	Select the font weight (such as bold)
Font Style	Dropdown box	Select the font style (such as italics)
Font Family	Dropdown box	Select the font family (such as segment)

Table 4-32. Alignment Group

Parameter	Type	Usage
Horizontal	Dropdown box	Select the horizontal alignment of content
Vertical	Dropdown box	Select the vertical alignment of content
Line Wrap	Dropdown box	Select conditions to wrap line of content

4.8.2.2.3. Slider

The Slider widget is a linear bar with a movable marker. The marker can be moved to adjust the value of the slider.

By default, the slider sends its values continuously as the slider is moved. However, if the slider option “Send on Release” is checked, then a single value is sent when the mouse button is released and the value has changed from the previous value sent.

Data Fields

The Slider widget has a source data field. When the slider value is changed a packet will send a value.

Configuration

Hover over parameters in visualizer for more information.

Table 4-33. Value Group

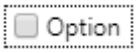
Parameter	Type	Usage
Initial	Numeric	Initial value
Min	Numeric	Minimum value of input
Max	Numeric	Maximum value of input

Table 4-34. Slider Group

Parameter	Type	Usage
Send on Release	Checkbox	Send slider value only when the mouse button is released (and the value has changed from the previous value sent.)

4.8.2.2.4. Checkbox

The Check Box widget will send a value each time its state is changed.



Data Fields

The Check Box widget has a source data field. Every time the state of the widget is changed a value is sent. When the box is checked a 1 is sent and when it is unchecked a 0 is sent.

Configuration

Hover over parameters in visualizer for more information.

Table 4-35. Checkbox Group

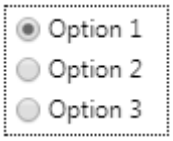
Parameter	Type	Usage
Label	Text	Enter text for the checkbox option

Table 4-36. Font Group

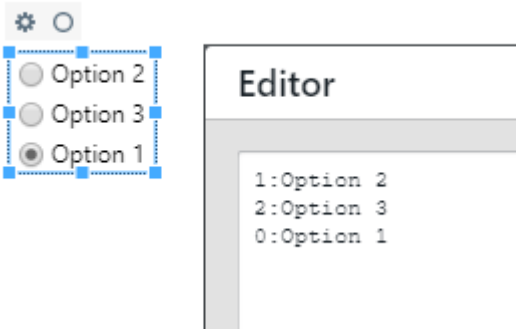
Parameter	Type	Usage
Font Size	Numeric	Enter font size
Font Weight	Dropdown box	Select the font weight (such as bold)
Font Style	Dropdown box	Select the font style (such as italics)
Font Family	Dropdown box	Select the font family (such as segment)

4.8.2.2.5. Radio Group

The Radio Group widget is a group of radio buttons where only one option can be selected at any time. Initially the first option is selected.



To make changes to the list, use the editor under Values.



You can also display the options horizontally.

Data Fields

The Radio Group widget has a source data field. Each time the state of the widget is changed a value is sent.

Configuration

Hover over parameters in visualizer for more information.

Table 4-37. Radio Group

Parameter	Type	Usage
Values	Editor	Edit a list of options to name, add, delete or rearrange options.
Horizontal	Check box	Check to display options horizontally.

Table 4-38. Font Group

Parameter	Type	Usage
Font Size	Numeric	Enter font size
Font Weight	Dropdown box	Select the font weight (such as bold)
Font Style	Dropdown box	Select the font style (such as italics)
Font Family	Dropdown box	Select the font family (such as segment)

4.8.2.2.6. Rectangle

The Rectangle widget sends a value each time it is clicked by the mouse.

Data Fields

The Rectangle widget has a source data field. Each time the element is clicked by the mouse pointer a value is sent.

Configuration

Hover over parameters in visualizer for more information.

Table 4-39. Rectangle Group

Parameter	Type	Usage
Color	Color selection dialog	Color of widget

4.8.2.3. Protocol Session

Widgets in this category are useful during protocol sessions for controlled reading and writing of variables.

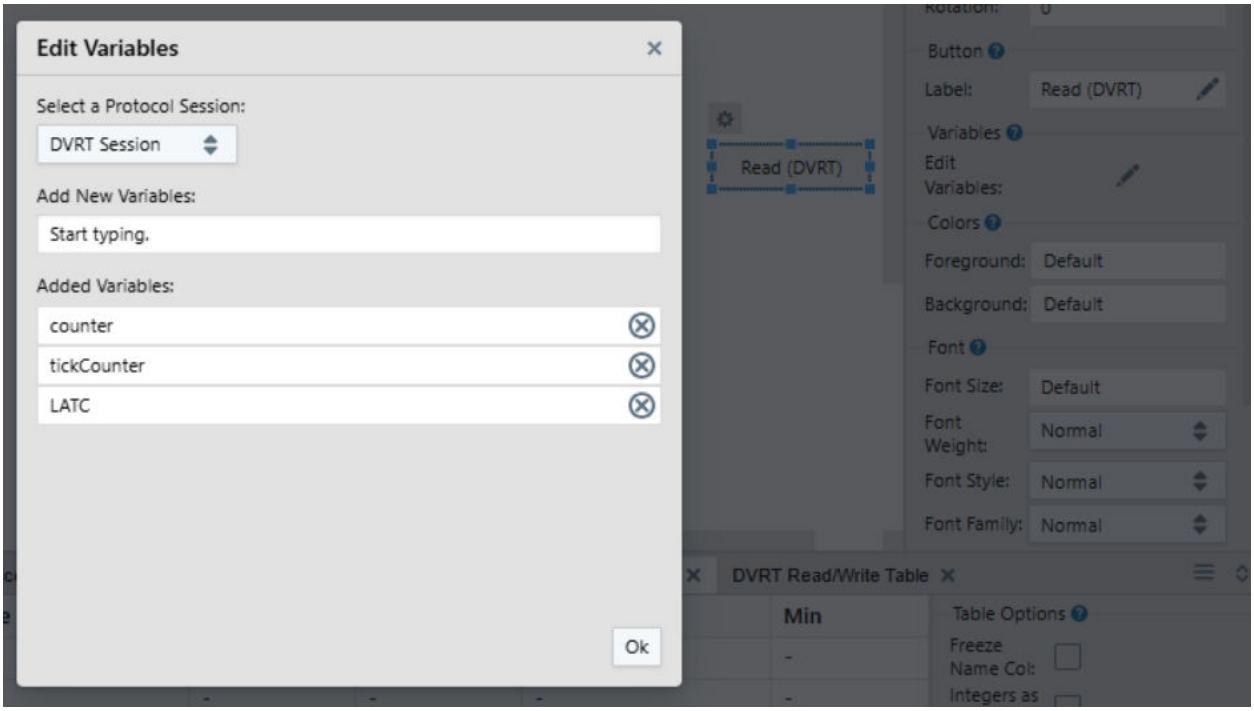
4.8.2.3.1. Read Variables

The Read Variables widget is a button that reads values from variables defined on a protocol session. It can only be used with a protocol session that supports loading an ELF file, which must be loaded before configuring the widget.

Selecting Variables

To select which variables this widget is connected to, press the button in the **Edit Variables** field of the widget configuration. This opens a dialog where protocol session can be selected. If a symbol file has been loaded for this protocol session, the associated variables can be browsed in the dialog by typing its name in the **Add New Variables** field. The selector that opens operates as described in Selecting Symbols. Variables that have been added are shown in the dialog, where they can be removed by pressing the **X** icon. Click **Ok** in the dialog to apply the changes.

Figure 4-66. Selecting Variables in the Edit Variables Dialog



Reading Variables

For input widgets added to the dashboard, variables selected as described above will appear in the sink selection list. Once other widgets are connected, pressing the button will read the symbols from the target and send their values to the connected widgets.

Configuration

The Read Variables widget shares some of the configuration fields with the Button widget. See the tables in Button for details on the fields. In addition, it has an Edit Variables field described in more detail above. Hover over parameters in visualizer for more information.

4.8.2.3.2. Write Variables

The Write Variables widget is a button that writes values to variables defined on a protocol session. It can only be used with a protocol session that supports loading an ELF file, which must be loaded before configuring the widget.

Selecting Variables

See Read Variables

Writing Variables

For output widgets added to the dashboard, variables selected as described above will appear in the source selection list. Once other widgets are connected, make sure those widgets send a value (e.g. by editing their value). Then, pressing the button will write the values to the target.

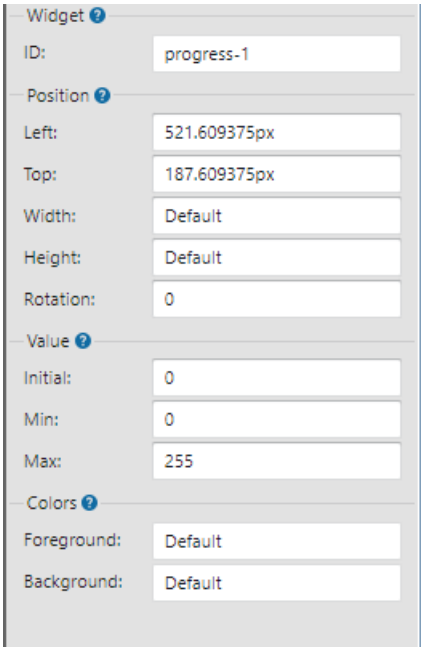
Configuration

See Read Variables

4.8.3. Dashboard Widget Controls (Right) Pane

When a widget in the Dashboard window is selected, the right pane displays controls for specifying widget position, size, and configuration. Depending on the widget selected, the controls will be different.

Figure 4-67. Example Controls - Progress Bar Widget



5. Protocol Comparison

MPLAB Data Visualizer supports three protocols for variable monitoring: Variable Streamer, DVRT, and X2Cscope. This section helps you choose the right protocol for your application.

Variable Streamer

Variable Streamer is the most lightweight protocol option. Variables to stream are predefined at compile time, and the user sends streaming data manually from a location in the application. Data is continuously streamed from the target to the PC as it is sampled. The protocol only supports streaming (one-way communication from target to PC).

The protocol uses UART for communication and has minimal firmware overhead.

Best for:

- Simple data logging with minimal firmware overhead
- Applications where the streamed variables are fixed
- Projects already using MCC with Variable Streamer driver

Refer to Variable Streamers for comprehensive information on this protocol.

DVRT

DVRT provides flexible runtime configuration. Variables can be configured at runtime without code changes - the set of streamed variables and timing parameters can be dynamically updated. Data is continuously streamed to the PC as it is sampled. In addition to streaming, DVRT allows reading and writing individual variables on demand (bi-directional communication).

The protocol has low overhead and uses UART for communication and a timer interrupt for sampling.

Best for:

- General debugging requiring flexible variable selection
- Applications needing continuous real-time streaming
- Quick iteration where streamed variables change frequently

Refer to DVRT Protocol for comprehensive information on this protocol.

X2Cscope

X2Cscope is a runtime debugging tool that uses the LNet protocol. The protocol has low overhead and uses UART for communication and a timer interrupt for sampling.

X2Cscope supports watch and scope functionality. With the scope function, X2Cscope first samples data into an internal buffer on the target MCU, then transfers the complete buffer to the PC. This approach allows capturing high-frequency signals without requiring high-bandwidth continuous communication. The set of monitored variables and timing parameters can be dynamically updated at runtime. Trigger detection is performed on the target MCU itself and controls whether data is actually captured into the buffer on the target.

The watch function reads/writes variables on demand. The MCU only responds when requested, there is no streaming. There is not any need for a buffer on the MCU to do this.

Best for:

- Motor control applications requiring parameter tuning
- Capturing fast signals with precise trigger timing
- Applications where continuous streaming bandwidth is limited

Refer to X2Cscope Protocol for comprehensive information on this protocol.

Quick Reference

	Variable Streamer	SVRT	X2Cscope
Overhead	Minimal	Low	Low
Complexity	Minimal	Low	Low
Streaming configuration	Fixed (compile time)	Flexible (runtime)	Flexible (runtime)
Adjustable data rate	No	Yes	Yes (sample factor)
Can write to the target	No	Yes	Yes
Data capture method	Streaming	Streaming	Buffered
Trigger support	PC-side	PC-side	Target-side
MCC Driver support	Yes	Yes	Yes

6. Variable Streamers

Most communication interfaces use streams of bytes to transfer data. This is enough for single data values of 8-bit precision, but when multiple values are required to be transmitted over the same interface, data must be packed in a protocol. The MPLAB Data Visualizer supports the **Data Stream** protocol.

The Data Stream protocol uses a light-weight framing format to pack several numerical values over one interface. It is only capable of handling incoming data and it only supports synchronous streams (i.e., every data packet must contain one sample from each data stream). **Data Stream Decoder** information resides in the visualizer workspace.

The visualizer data stream module takes an incoming raw data stream and splits it into multiple data streams. The data stream format is specified by the **Variable Streamer** you provide.

6.1. Variable Data Types

Variable Streamers are set up and plotted using a wizard. This set up is saved in the workspace.

Variables defined in a Variable Streamer must be of a type listed in the table below.

Table 6-1. Allowed Data Types

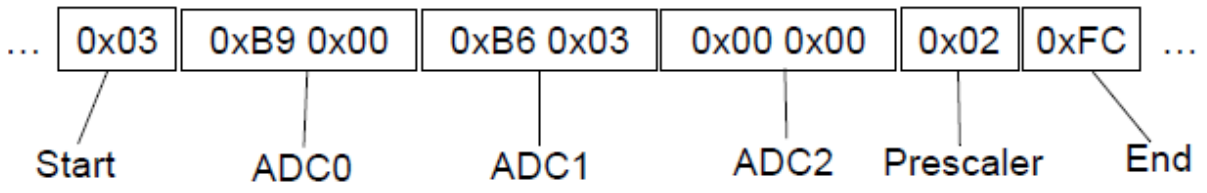
Type	Size (Bytes)
int8	1
int16	2
int32	4
uint8	1
uint16	2
uint32	4
float32	4
float64	8

6.2. Stream Format

The data stream Stream Format is processed in the same order as the variables specified in the Variable Streamer. All data must be given as little endian values, meaning that the lowest byte must be sent first. Additionally, a wrapper consisting of one byte before and one byte after the data stream variables must be added. This wrapper is used by the interpreter to synchronize to the data stream. The start byte can be of an arbitrary but the end byte must be the inverse of the Start byte. The start and end bytes are not defined in the configuration.

The figure below gives an example raw data transmission where ADC0 is 185, ADC1 is 950, ADC2 is 0, and Prescaler is 2.

Figure 6-1. Data Streamer



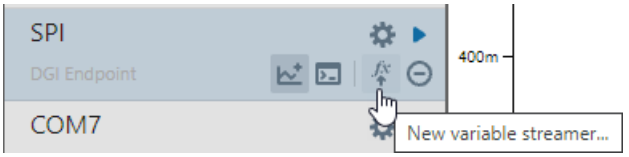
6.3. Variable Streamer Setup and Plot

A Variable Streamer defines the variables that are embedded in a data stream as output from your application. The Variable Streamer is used in the configuration of a Data Stream Decoder instance.

The output from a decoder instance are new data streams that can be visualized using plots on the graph.

6.3.1. Select Data Source

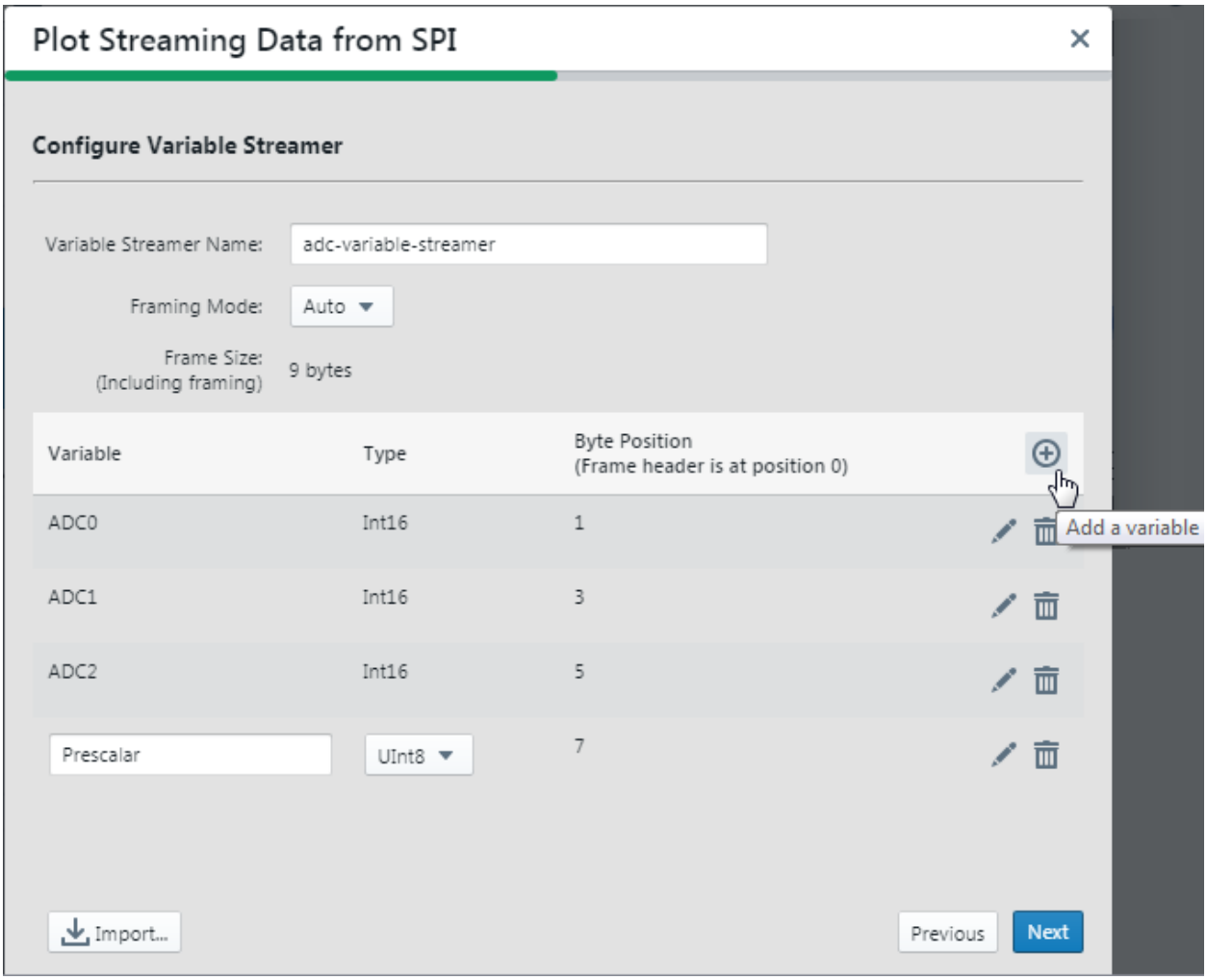
To create a new Variable Streamer, go to the Data Sources pane, **Connections** tab. From the data sources available, select "New Variable Steamer".



The **Plot Streaming Data** wizard will open to the **Configure Variable Streamer** window.

6.3.2. Configure Variable Streamer and Plot

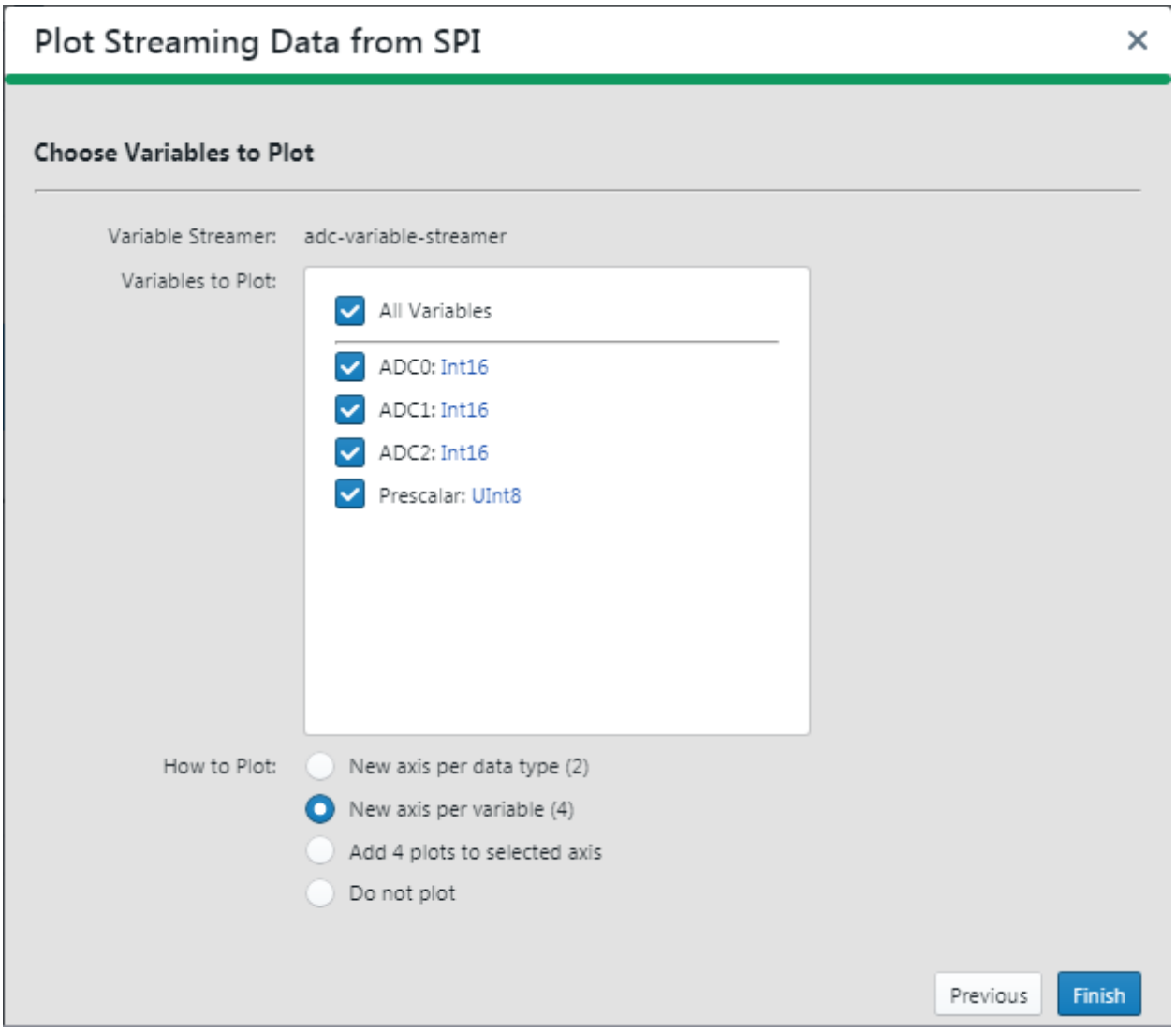
Use the "Configure Variable Streamer" dialog to set up variables. See the table below the figure for descriptions of controls.



Option	Description
Variable Streamer Name	Choose a descriptive name for the variable streamer

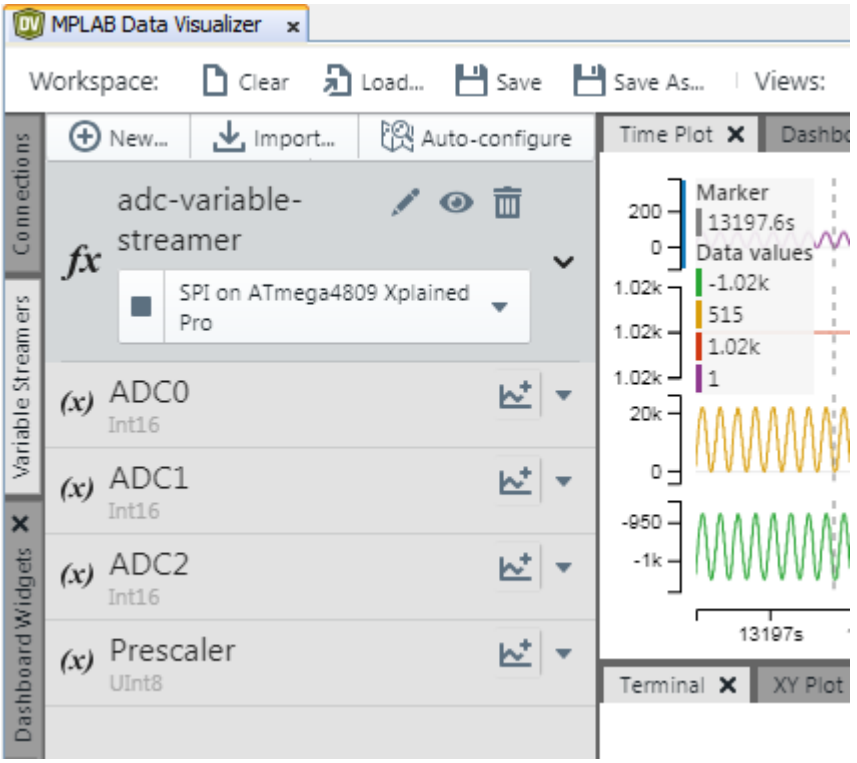
Configure Variable Streamer and Plot (continued)	
Option	**Description**
Framing Mode	If the start byte of data streamer protocol is 0x5F and end byte is 0xA0, “Auto” can be used. For any other start byte and end byte pattern “one’s complement” can be used.
Framing Size	See Stream Format.
Variable	Enter the name of a variable from application code.
Type	See Variable Data Types.
Byte Position	See Stream Format.

Click **Next** to **Choose Variables to Plot**. Select a plotting method and click **Finish**.



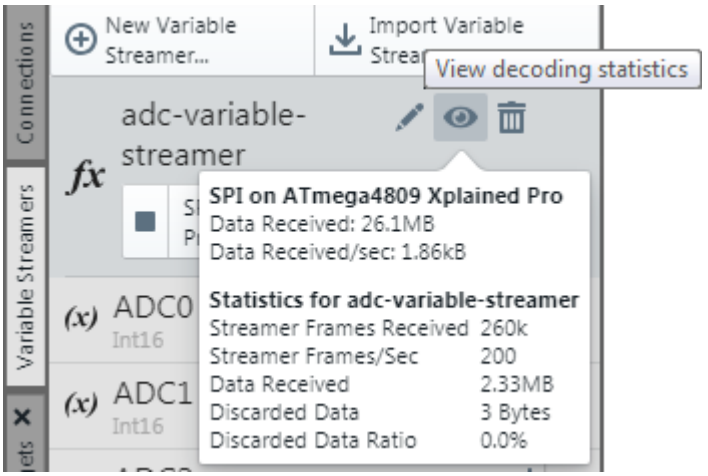
6.3.3. View Variable Streamer Output

The new Variable Streamer will be shown on the **Variable Streamers** tab. To save your setup, save the workspace at the end of your session and then load the workspace for your next session.



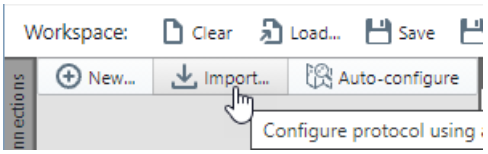
6.3.4. View Statistics

To view statistics on streaming data, click on the eye icon to see the Variable Streamer data.



6.4. Import Variable Streamer

In addition to creating a new variable streamer, you can also import a legacy data stream protocol definitions file (`.ds` or `.txt`).



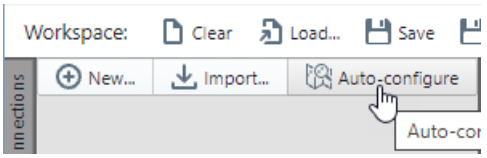
Once imported, you can set up and use the streamer as you would a new or an Auto-Configure Variable Streamer variable streamer. For the later, when the serial source is selected as the input

on the variable streamer, it will attempt to find `.db` and `.sc` files that match the auto-configure ID. On a successful file match, the auto-configure process completes and the dashboard is shown. The variable streamer will be renamed accordingly.

6.5. Auto-Configure Variable Streamer

Open an MPLAB X IDE project that supports Auto-Configure streaming format (see Auto-Configure Example and Format). Program the target with the application.

Open the MPLAB Data Visualizer and click on the Variable Streamer tab. Then click “Auto-Configure” to create an Auto-Configure variable streamer (see figure below). Select a source and then configure the serial port baud rate to match the project settings (auto-detect baud rate not yet supported).



Now start streaming from the serial port on the target. Select the serial source for the Auto-configure variable streamer.

The data visualizer will look for an Auto-configure frame in the serial data selected as the input. The Auto-config frame contains a config ID which will be used to search for files on disk to load. On finding a matching frame, the system searches the disk for three files matching the config ID in the serial stream. The filenames it searches for and tries to load are `<config id>.ds`, `<config id>.sc`, `<config id>.db`.

Search path behavior

For the MPLAB X plugin, it follows the hot project or main project (project context). If there is no project context (e.g. no projects open), it defaults to the `MPLABXProjects` directory. You can override this setting, but it only persists for the current auto-configure entry. A new auto-configure entry will revert to the default behavior of following project context.

For the Standalone DV, it defaults to the user home directory. If you select a different directory, it will use that for new auto-configure entries. This value persists across restarts.

On successful match and file loading

- The auto-configure entry is converted to a variable streamer with fields from the `.ds` file.
- The auto-configure settings will remain visible as information the user can use to orient themselves to the file system location used to load the dashboard.
- The new streamer will be named the same as the project in the case of a project context being available and the parent directory if one is not available.
- The dashboard contents are replaced and the dashboard tab is opened.
- The non-dashboard DV windows and dashboard properties sidebar are collapsed* and the dashboard mode is put in Edit mode.

*unless you have unchecked the “Maximize dashboard” option in the auto-configure options.

Error on file loading

If files are not successfully loaded or there are missing files, an error will show in notifications. You can make adjustments to search path settings as necessary, such as selecting a different project or browsing to a different path.

```
Could not find 3 files named 03EB00000000000000AA5501[.db, .sc, and .ds] in /
Users/User/example-fw-projects/ATtiny817-Xpro-touch-project.X/build.
Try changing the search path.
```

6.5.1. Auto-Configure Example and Format

MPLAB Data Visualizer Auto Configure is similar to Atmel Data Visualizer Auto-Configuration.

For an example application using Auto-configure, as well as Auto-Configure files and formats, see the Atmel Data Visualizer User's Guide (DS40001903) sections "Auto-Configuration Example," "Auto-Configuration Format" and "Signal Connections File Format."

7. DVRT Protocol

DVRT (Data Visualizer Run Time) is a simple bi-directional protocol suited for small microcontrollers. It supports streaming as well as reading and writing of global variables (memory locations) of the target application.

With the DVRT protocol the set of streamed variables or memory locations, as well as the streaming speed, can be updated dynamically at run time. This is different from the Data Streamer protocol, where the variables that are being streamed are defined at compile time.

For details on DVRT protocol options, see DVRT Protocol Options and Status Information.

For current DVRT device support, please refer to Finding Release Notes.

7.1. Adding DVRT Support to a Project

The DVRT implementation in the target relies on a UART in order to communicate with the PC and a timer or a regular callback in order to process commands and stream data in a timely manner. Therefore the project device must have these resources available.

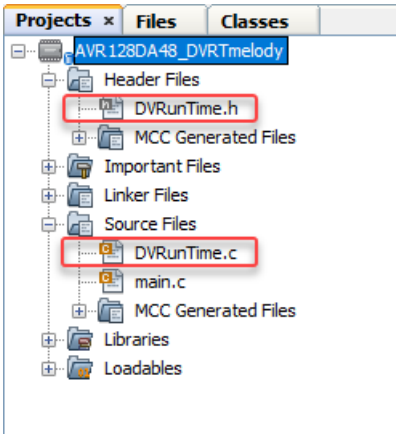
DVRT project examples may be found in zip file(s) on the MPLAB® Data Visualizer webpage under Documentation.

To set up a project for DVRT support:

1. Create a new project or open an existing one to which you want to add DVRT functionality.
   Note: Currently any non-library project can be used for DVRT.
2. Add the following files to your project: `DVRunTime.c`* and `DVRunTime.h`* (see figure below).
3. Build and run the project for debug  to create the ELF file required in the next section.
4. Invoke `DVRT_Initialize()`* at the start of your application in order to initialize the DVRT driver.
5. Invoke `DVRT_Process()`* periodically in your main loop to process DVRT commands (e.g., once per ms).
6. Update the UART `defines` at the end of `DVRunTime.h` in accordance with the UART implementation on the target.

* See examples for content.

Figure 7-1. Example DVRT Project



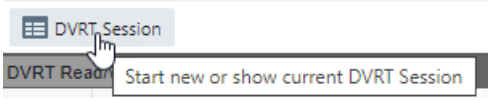
7.2. Loading the DVRT Project ELF file into the Data Visualizer

In order to use the DVRT protocol, the DVRT Project ELF file must be loaded in the MPLAB Data Visualizer. Once the ELF file is loaded, you can browse for global variables (or struct and array elements) and add these to the streaming data table and read/write data tables. The data visualizer will send the location and size of the variable to the target so that it knows what to stream.

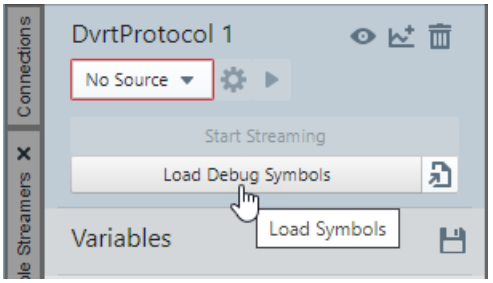
7.2.1. Manually Loading the ELF file in the Standalone Data Visualizer

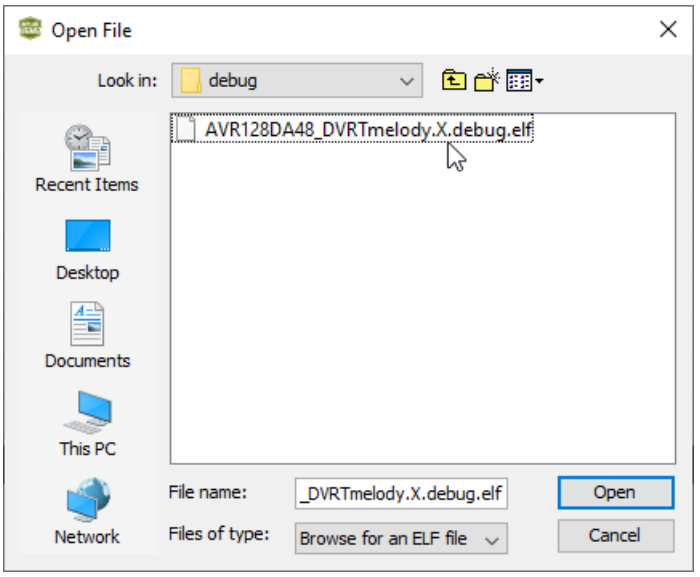
If using the standalone data visualizer:

1. In MPLAB X IDE, ensure a project is open and set up as per Adding DVRT Support to a Project.
2. Click to open the DVRT pane.



3. Click **Load Debug Symbols** to find and load the project ELF file.





4. If you decide to load a different ELF file, click on  .

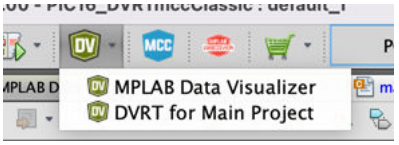
When the ELF is loaded this way, then the ELF path will be monitored for changes. On rebuild, the data visualizer will give the option to reload it. When the ELF is programmed into the target, then symbols will be reloaded and a message displayed when complete.

Note: MPLAB generates different files for production and debug use cases. These files that are located under the `dist/production` or `dist/debug` folders, respectively. Care must be taken so that the ELF file that is loaded into the data visualizer matches the use case.

7.2.2. Automatically Loading the ELF File in the Plugin Data Visualizer

If using the MPLAB X IDE data visualizer plugin:

1. In MPLAB X IDE, ensure a project is open and set up as per Adding DVRT Support to a Project.
2. From the toolbar, find the MPLAB Data Visualizer icon, click on the drop-down menu, and select **DVRT for *Project Name***."



3. If the symbols fail to load, see instructions for manual loading under Manually Loading the ELF file in the Standalone Data Visualizer.

Additionally the following behavior can occur.

If	Then
ELF file does not exist	The project will build in order to produce an ELF file.
A DVRT session is created with a production ELF and you start a debug session.	The debug ELF will replace the DVRT session's ELF file. Hover over the symbol loaded button in the data visualizer in order to see the path when the debug session starts.
A debug ELF is in use for the current DVRT session and you start a production "run" or program action that replaces the debug ELF file.	The production ELF will replace the DVRT session's ELF file once the program/run operation is completed.
A DVRT session is already defined in the data visualizer.	A dialog will be shown to confirm replacement of the existing session. Cancel will have the effect of not changing the data visualizer and confirm will replace the existing session with the new project's ELF path.

7.2.3. Seeing ELF File Name in DVRT Panel

Once the ELF file is loaded, you will see it displayed in the Status section of the DVRT Protocol tab.

Click on the clipboard icon to copy the ELF file path to the clipboard.

If you cannot see the DVRT Protocol Options and Status, click on the DVRT Protocol pane, but not a control, to show these selections. Once you can view Options, you will be able to change them if needed. Mouseover a field to see a pop-up definition.

Figure 7-2. Options and Status Information

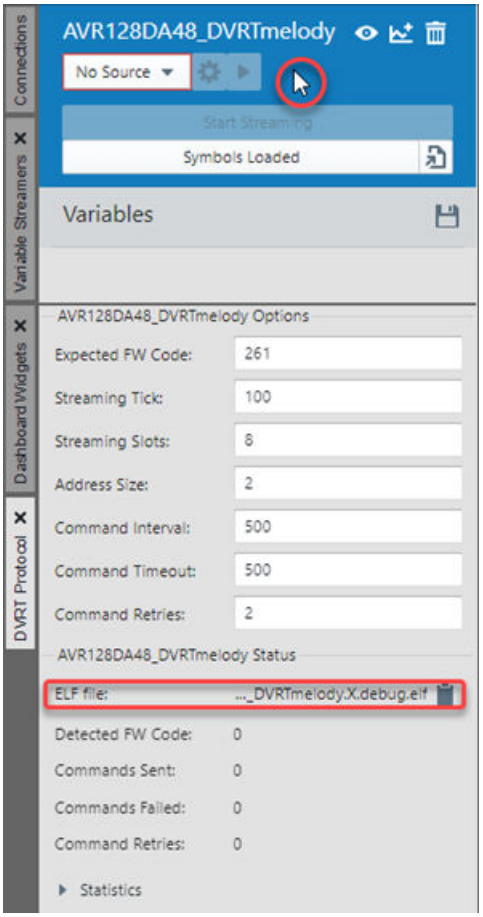
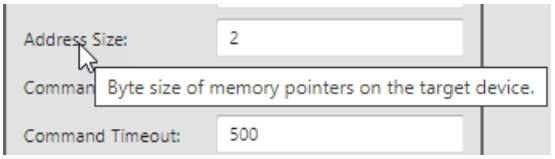


Figure 7-3. Mouseover of Field



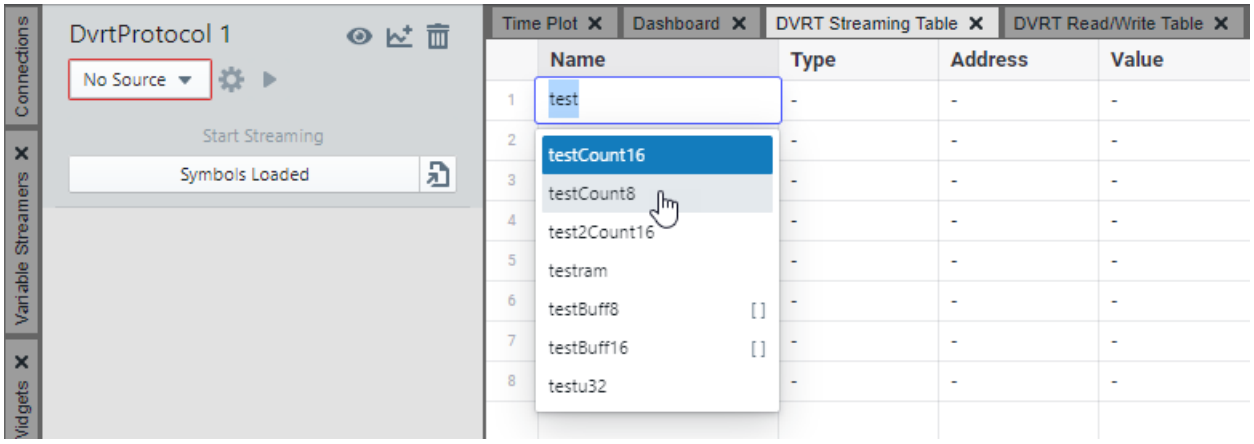
7.3. Selecting Symbols

Once the symbols are loaded, you can select the ones you want to add to the DVRT Streaming Table tab. Click a cell in the Name column and then start typing to drop down a list of symbols.

7.3.1. Selecting Variables

For a variable, click to select the one you want.

Figure 7-4. Select a Variable



7.3.2. Selecting Structs

To select a struct element, first click on the name of the struct and then select the wanted member from the list.

Figure 7-5. Select a Struct

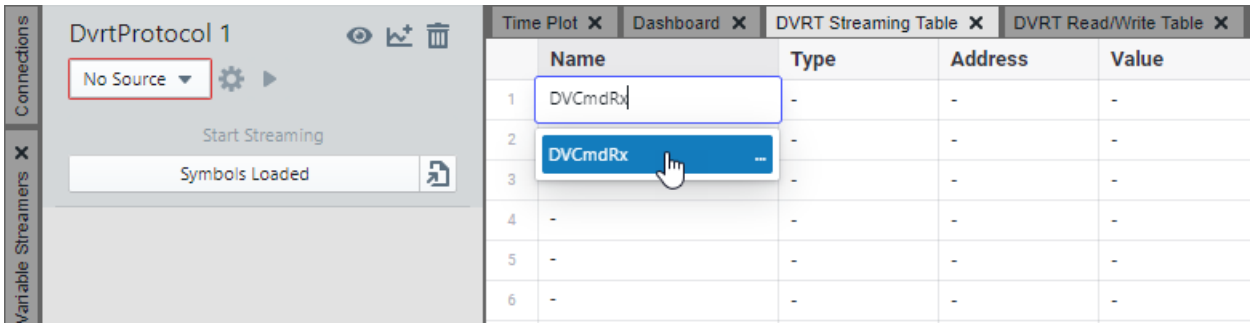
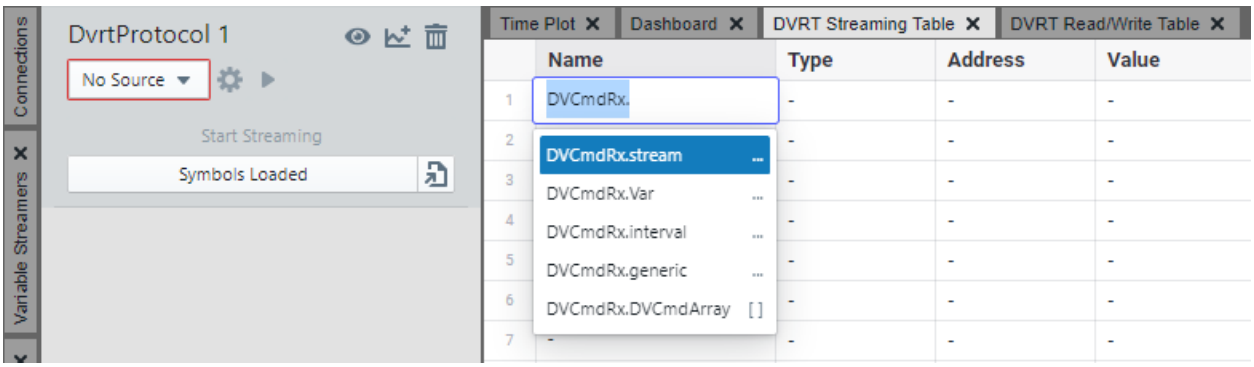


Figure 7-6. Select a Struct Element



7.3.3. Selecting Array Elements

To select an array element, first click on the name of the array and then manually enter the index of the array element (by default 0).

Figure 7-7. Select an Array

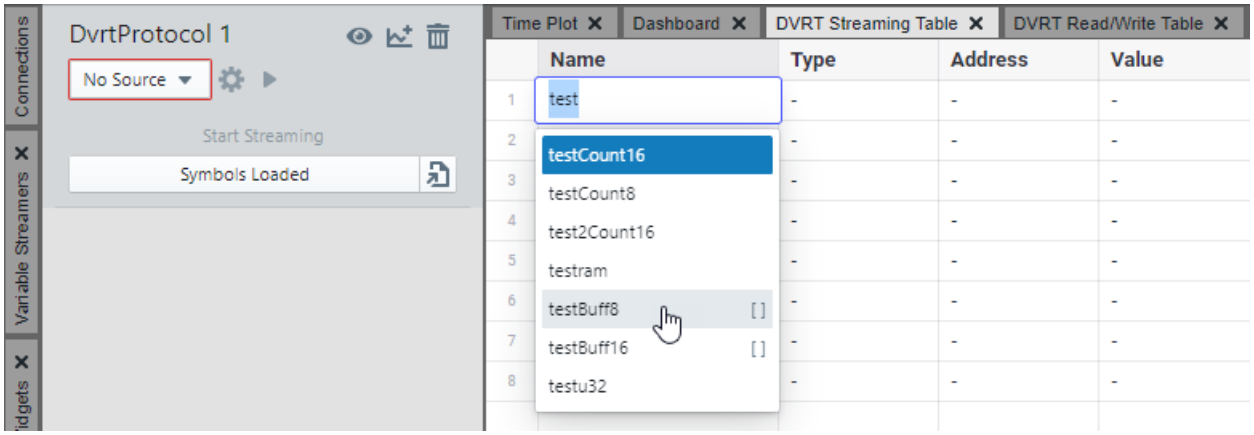
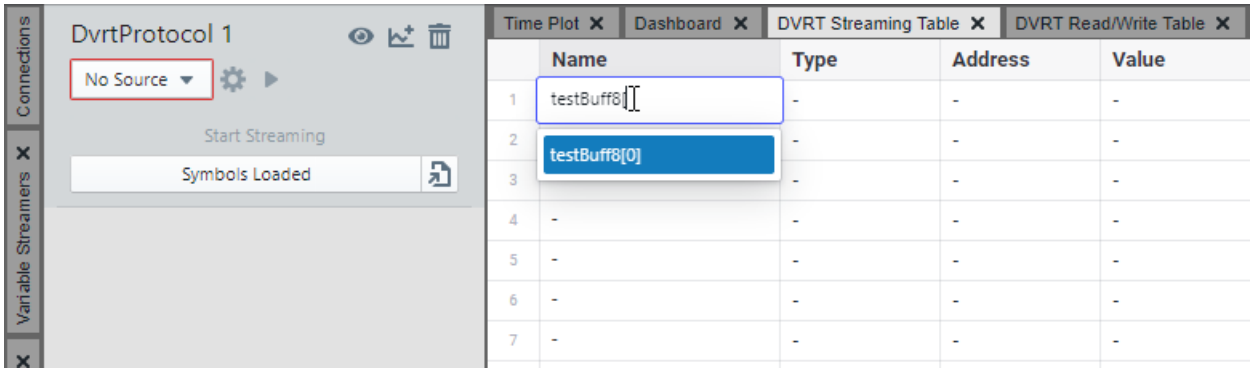


Figure 7-8. Select an Array Element



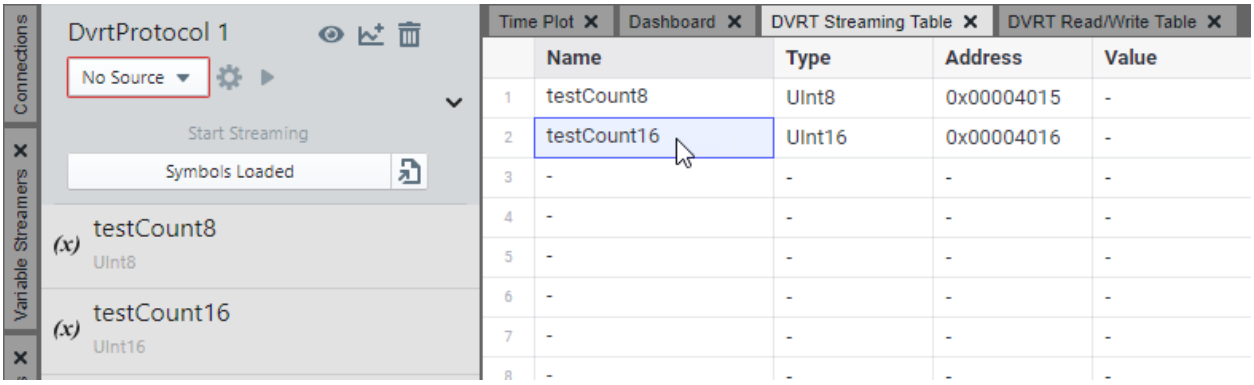
7.3.4. Viewing Selected Symbols

Once you click on the selected symbol, it will appear in the table cell with other row cells populated with related information. Repeat as needed to add other symbols.

Note: If you enter a symbol in a row below an empty one, the entry will snap to row above.

To delete a row, click a cell in that row and then press Delete.

Figure 7-9. Symbols Added

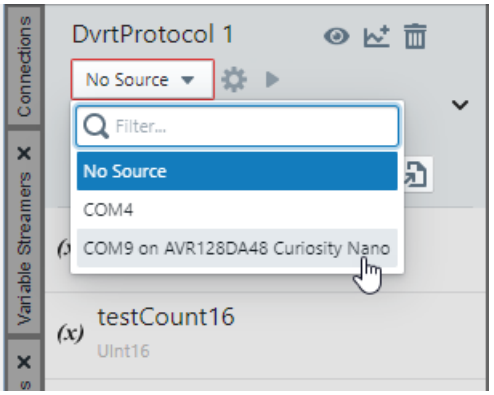


7.4. Streaming Variable Values

7.4.1. Starting Variable Streaming

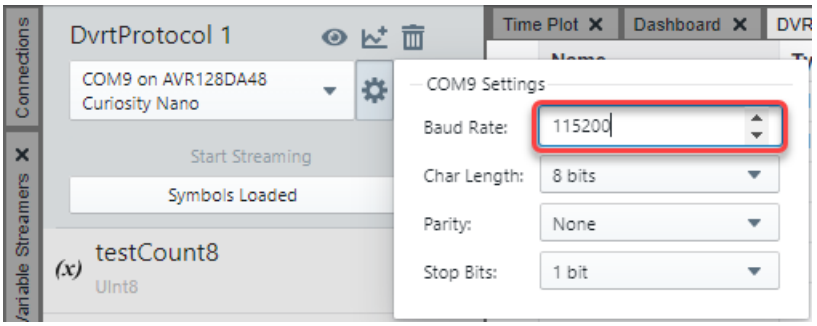
Select the source for streaming.

Figure 7-10. Select the Connection Source



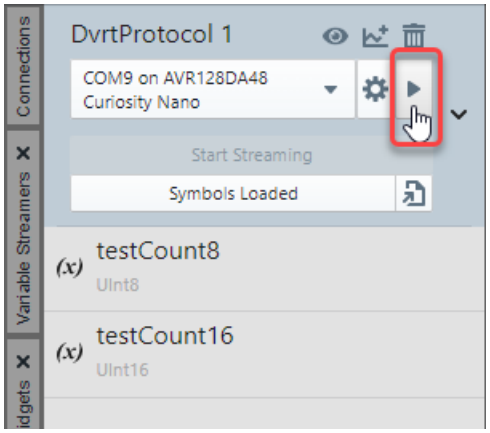
Set up the connection by clicking the gear icon. For DVRT, use the values shown in the figure below.
Note: For all implementations of DVRT, use the Baud Rate of **115200**.

Figure 7-11. Connection Settings for DVRT



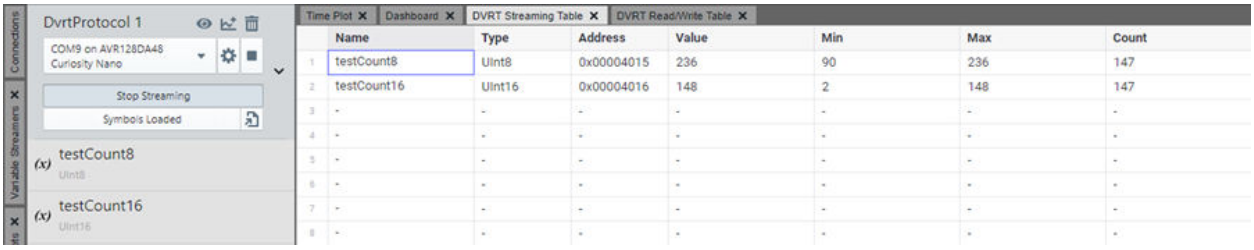
Click the Play button to start streaming. Once streaming has started, a toggle button will be made active where you can **Stop Streaming/Start Streaming**.

Figure 7-12. Play to Start Streaming



See the variable values change during runtime in the table.

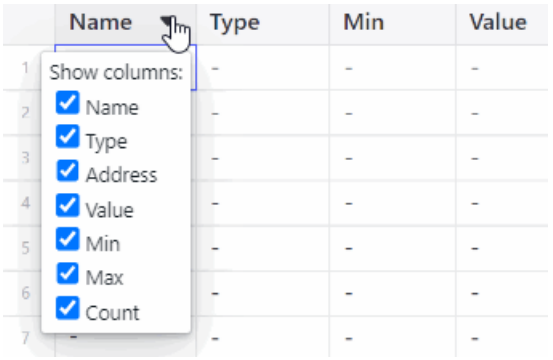
Figure 7-13. Streaming Int Values



7.4.2. Streaming Table Appearance Configuration

Columns of the streamer table are individually hide-able. Hovering over any header cell will show a clickable "caret down" symbol. Clicking this produces a popover menu with toggle buttons for showing each column, in the default column order. All columns are shown by default, and deselecting one removes it from the table.

Figure 7-14. Hiding Table Columns



If you want change value format, in this case view Int values in hex, you can use the controls on the right to select available options.

Figure 7-15. Format Hex

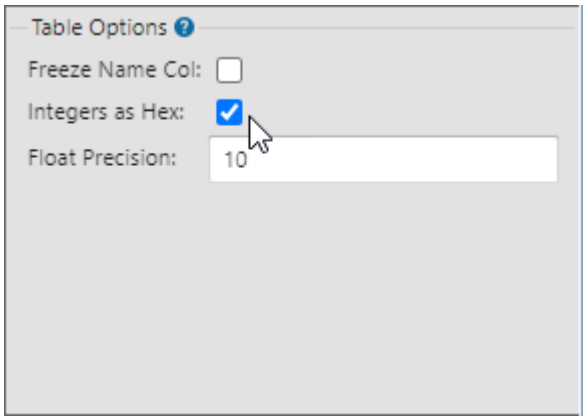
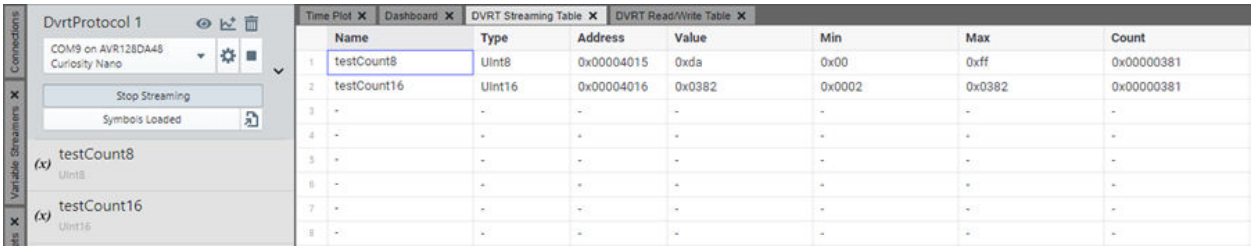


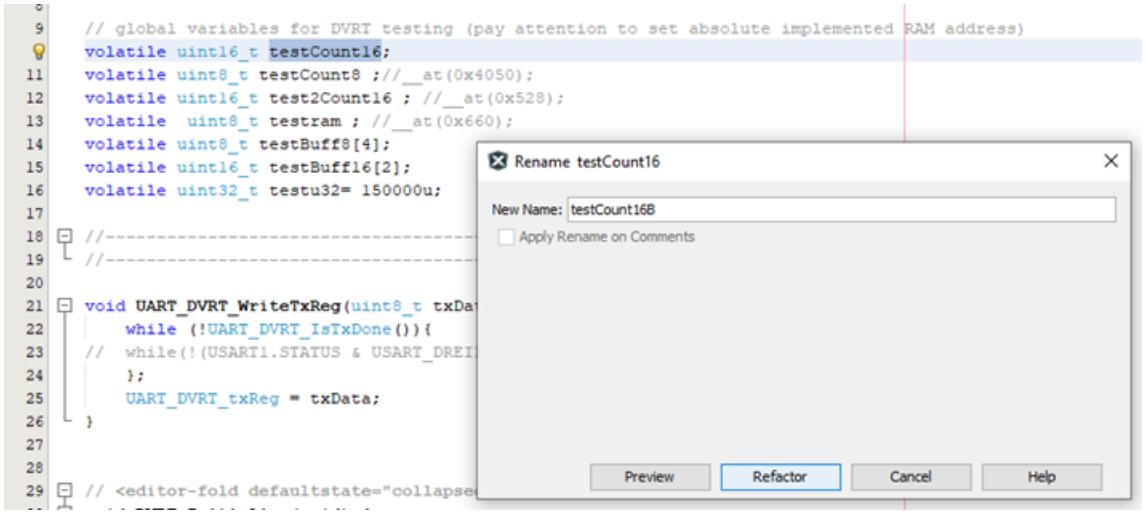
Figure 7-16. Streaming Int Values in Hex



7.5. Changing a Variable Name and Streaming Again

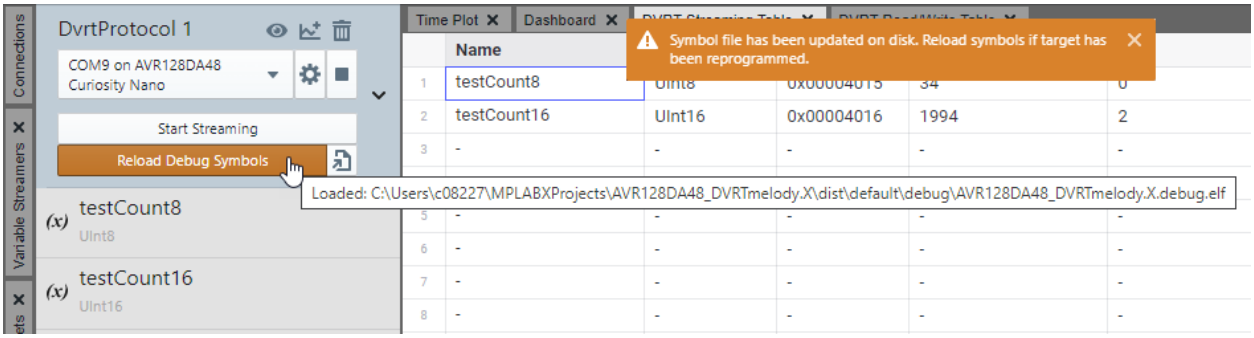
Open the file `DVRunTime.c`. Change the name of the variable `testCount16` to `testCount16B` by using *Refactor>Rename*. Rebuild the project to create a new ELF file.

Figure 7-17. Rename a Variable



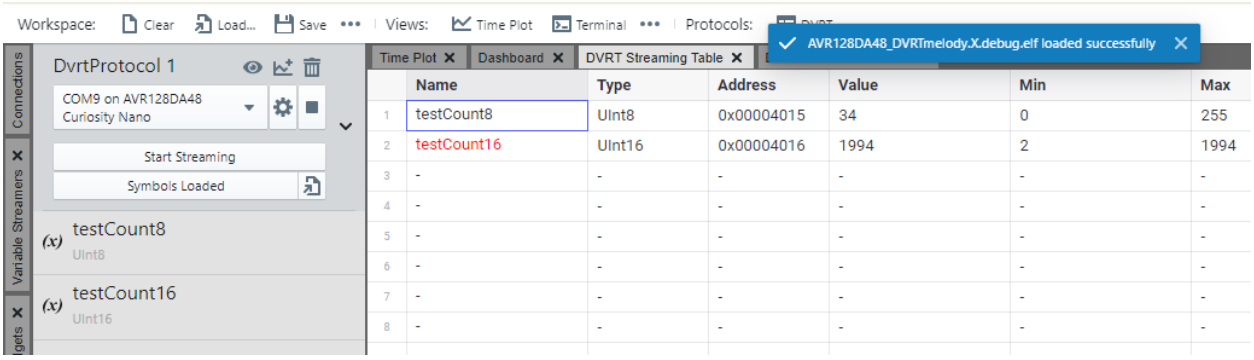
Now return to the Data Visualizer. Because the DVRT session is in synch with the ELF file, you will see a message stating that the ELF file has been updated and that symbols need to be reloaded.

Figure 7-18. Reload Debug Symbols



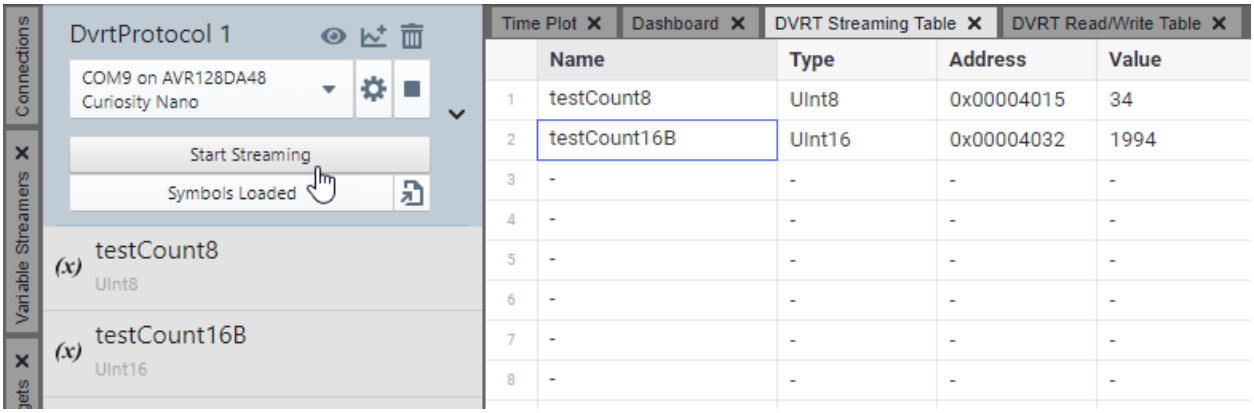
Click **Reload Debug Symbols**. Once the new ELF file is loaded, the previous variable name will be in red because it is no longer found.

Figure 7-19. Variable Not Found



Change the variable name to the new name and see the streaming values for this variable appear.

Figure 7-20. Renamed Variable Added

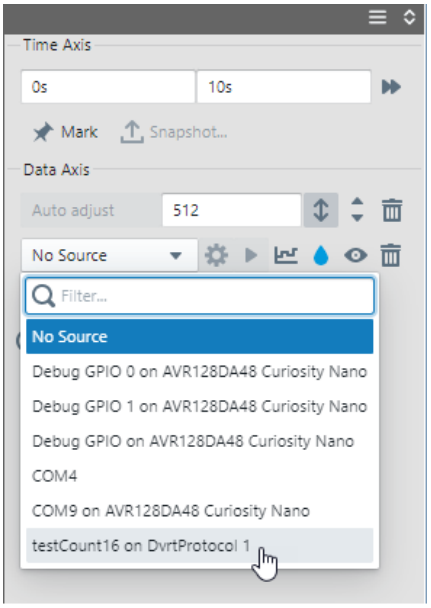


7.6. Visualize Streaming Data

Variable data shown on the DVRT Streaming Table may be visualized as other streaming data is visualized: on time plots, in the terminal, on the Dashbord, or on an XY plot. For example, to show data on a Time Plot, select the variable as a source (see the figure below).

For more information, see Visualization.

Figure 7-21. Time Plot DVRT Streaming Variable



7.7. Read and Write Variables in Firmware

The DVRT Read/Write Table tab is used to read and write individual variables, not streaming.

- Edit fields the same way as in the DVRT Streaming Table.
- Under Actions, click to read variable value from target. The Value will be displayed in the next column.
- Under Value, double click to edit value and then hit **Return** or click elsewhere to write the value to the target.

Figure 7-22. Read Value from Target

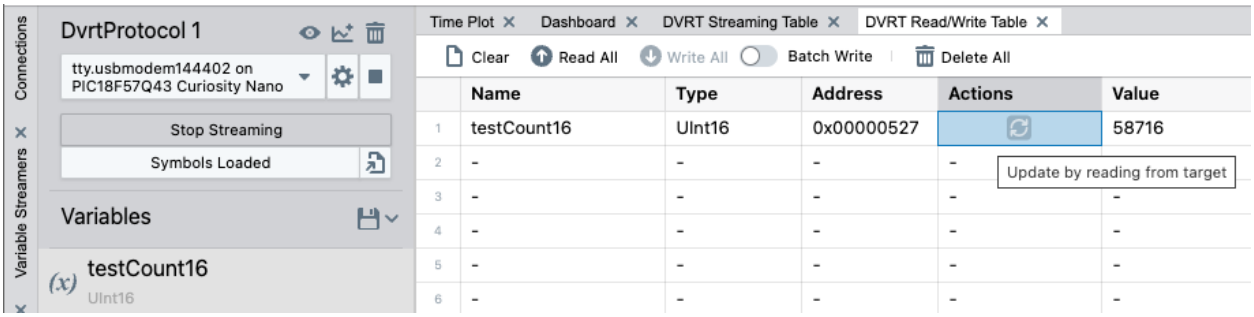
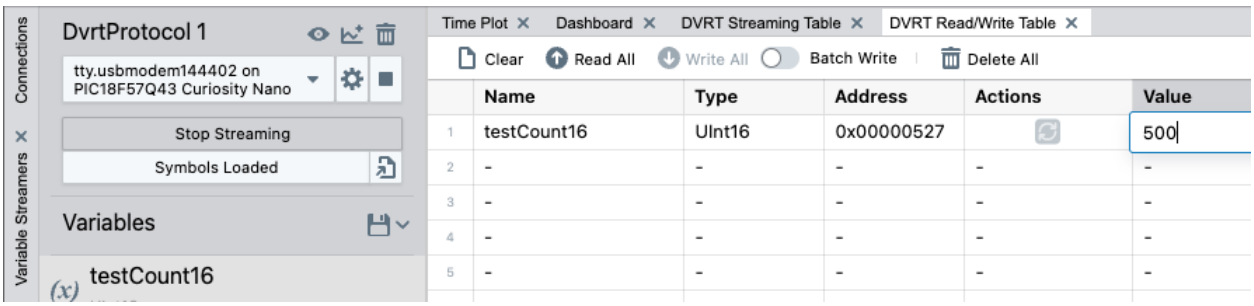


Figure 7-23. Write Value to Target

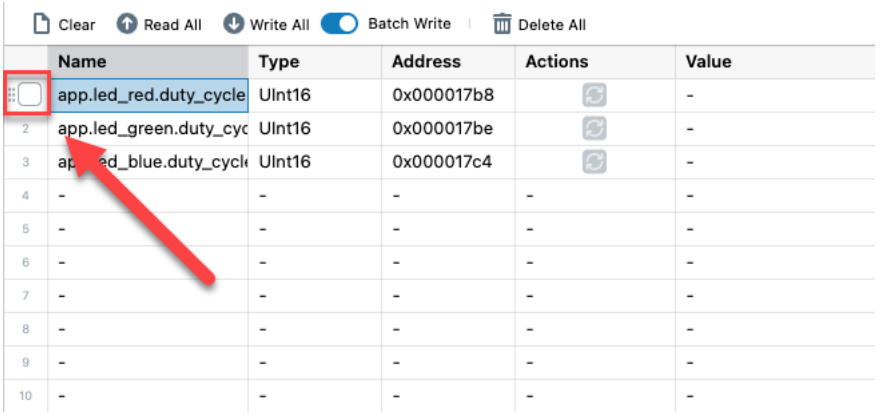


7.8. Batch Read or Write Multiple Variables

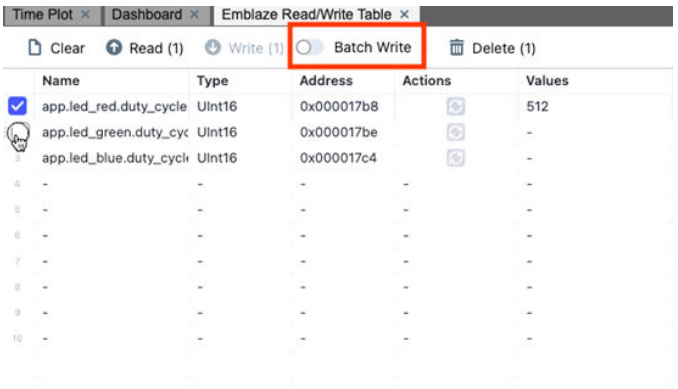
Read/Write Table Toolbar Button Behavior

Batch reads and writes are useful when multiple values need to be read or written in sequence, such as for configuring a specific test or development scenario. For batch writes, feedback from the target will indicate the success or failure of each individual write command. For batch reads, these are controlled by the table row selections and **Read** toolbar button.

Multiple rows may be selected or deselected by moving the mouse cursor over the leftmost column in the table, holding the **ctrl** key (Windows/Linux) or the **cmd** key (macOS), and clicking the checkbox so that it adds or removes a check mark. Rows may be rearranged by clicking and dragging the move handle in the leftmost column, next to the checkbox.



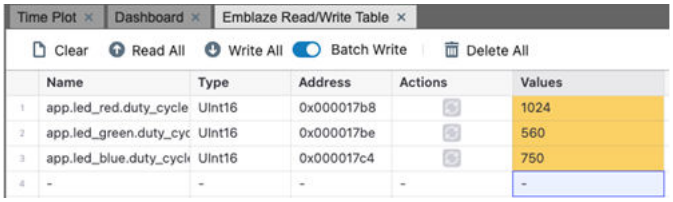
If multiple rows are selected, automatically enable batch write:



The Read/Write toolbar button titles indicate how many rows are selected. When no rows are selected, the behavior of these buttons is to operate on all rows.

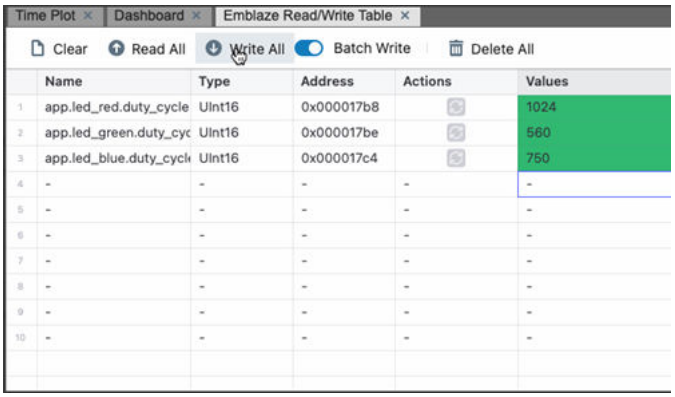
Read/Write Table Cell Behavior

When a value is changed in batch write mode, the background color of the cells changes to yellow to indicate pending writes.

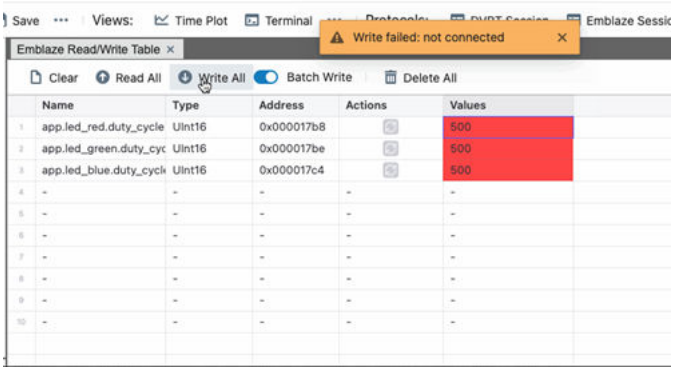


When the **write** button is pressed in batch write mode, write commands will be issued for each selected row in ascending order from top to bottom sequentially (table row index 1 to N).

When a value is written, the cell will turn green to indicate success or red to indicate failure.



In the event of a write failure, a popup will show an error message. Refer to the **Messages** view for error details.



When batch mode is disabled by toggling the switch to off, it will clear the cell colors and revert pending cell values to the last read values (non-batch mode).

7.9. Live Updates and Scaled Values

This section covers advanced Read/Write table features.

Automatic Updates (Live Polling)

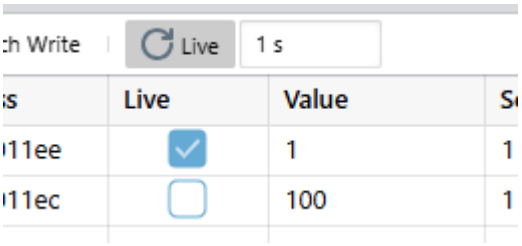
The Read/Write table supports automatic periodic updates for watch variables.

1. Check the **Live** checkbox next to variables you want to update.
2. Configure the sample interval in the panel settings.
3. Click the **Live** button in the toolbar to start updating periodically.

Only variables marked as Live are updated, minimizing communication overhead.

Click the **Live** button again to stop polling.

The sample interval controls the update rate for Live variables. Configure based on how quickly the monitored values change and your bandwidth constraints.



Scaled Values

The Watch table has the possibility to calculate a scaled value for each variable row, by specifying custom Scale and Offset factors in the columns with the same names in the Watch table.

The resulting scaled value is the result of the expression:

```
Scaled Value = Raw Value * Scale + Offset
```

It is possible to edit the scaled value in the table. The raw value will then be updated accordingly.

Value	Scale	Offset	Scaled Value
1	2	10	12.00000000
100	1	0	100.0000000

Use Cases

Scaled values are useful for:

- Converting raw ADC values tp engineering units (voltage, current, etc.)
- Applying calibration offsets

7.10. DVRT Protocol Options and Status Information

To view and change DVRT Protocol Options and Status, click on the DVRT Protocol pane, but not a control, to show these selections. Mouse over a field to see a pop-up definition.

Figure 7-24. Options and Status Information

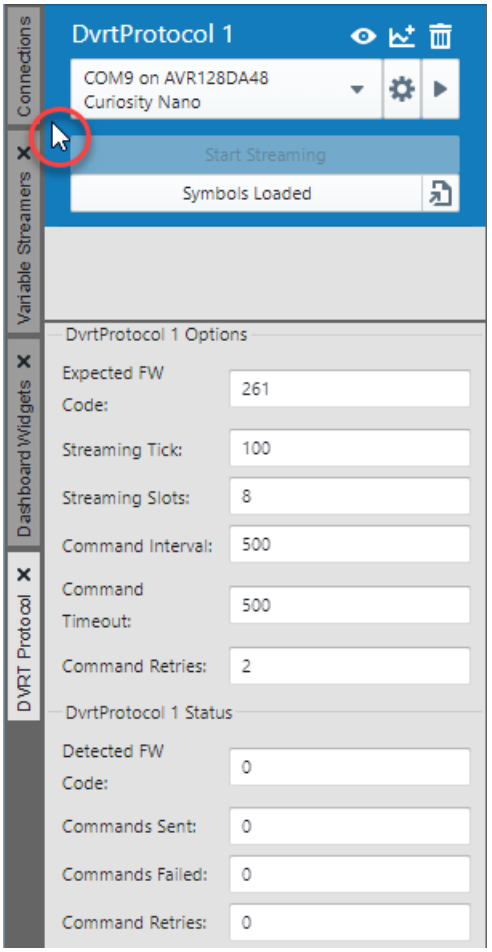
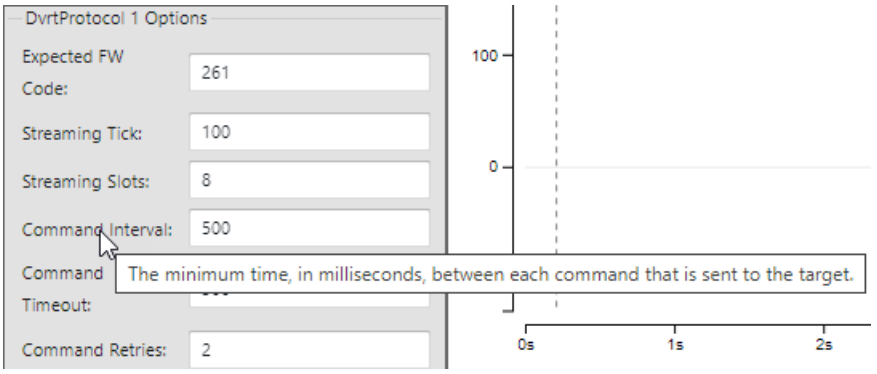


Figure 7-25. Mouseover of Field



8. X2Cscope Protocol

X2Cscope is a runtime debugging tool that enables live variable monitoring and manipulation on Microchip microcontrollers. It communicates with the target using the LNet protocol over standard interfaces such as UART, allowing developers to inspect and modify firmware variables without halting execution.

The protocol has low overhead and uses UART for communication and a timer interrupt for sampling.

Key Features

- Scope function: Capture buffered data on the target for visualization.
- Watch function: Read and write firmware variables with optional periodic updates.
- Target-side triggers: Configure trigger conditions that are evaluated on the MCU itself.

When to Use X2Cscope

X2Cscope is particularly suited for:

- Motor control applications requiring real-time parameter tuning.
- Power management systems needing runtime diagnostics.
- Capturing data in applications that benefit from target-side buffering.

For a comparison with other protocols, see Protocol Comparison.

Scope Function

With the scope function, X2Cscope first samples data into an internal buffer on the target MCU, then transfers the complete buffer to the PC. This approach allows capturing high-frequency signals without requiring high bandwidth continuous communication. The set of monitored variables and timing parameters can be dynamically updated at runtime.

Trigger detection is performed on the target MCU itself and controls whether data is actually captured into the buffer on the target. This provides precise triggering on high-speed events.

See Scope Channels and Scope Triggers for details.

Watch Function

The watch function reads and writes variables on demand. The MCU only responds when requested - there is no streaming. There is no need for a buffer on the MCU to do this. This is suitable for configuration parameters and slow-changing values.

See Watch Table.

Related Resources

x2cscope.github.io/

8.1. Adding X2Cscope Support to a Project

To use X2Cscope with MPLAB Data Visualizer, your firmware project must include the X2Cscope library and make periodic calls to its API functions.

Prerequisites

- X2Cscope firmware library for your target MCU family.
- A free UART peripheral for communication.
- A timer or periodic interrupt for sampling.

Adding X2Cscope Support

For information about how to create a code project for X2Cscope, see: x2cscope.github.io/docs/firmware/X2CscopeFirmware.html.

The screenshots in this section are based on this code example for dsPIC33CK: mplab-discover.microchip.com/com.microchip.ide.project/com.microchip.mplabx.project.dspic33ck-lvmc-x2cscope-blinky

There are 3 main interfaces that must be integrated into your application.

X2Cscope_init(): Must be called before using other X2Cscope interfaces. When using code configurator, like MCC or Harmony, this will be done automatically. The X2Cscope initialize routine connects the serial peripheral interfaces, initializes LNET protocol, buffers and functions.

X2Cscope_communicate(): Handles the communication with the client on the PC side. This function is not very time critical. This can be executed in a low level task in an RTOS environment, or in the IDLE loop with OS free application. This must be done manually as it highly depends on the final application implementation.

X2Cscope_update(): This is the sample point of the scope, therefore this must be called with the same fixed periodicity as selected in the X2Cscope interface on the PC side. Other wise the time domain will not match in the scope view. If you select a variable for the scope channel on the PC side, then this function will start to stream the values of the selected variables to a reserved RAM buffer. Once the buffer is full the buffer content is transferred to the PC.

Building the Project

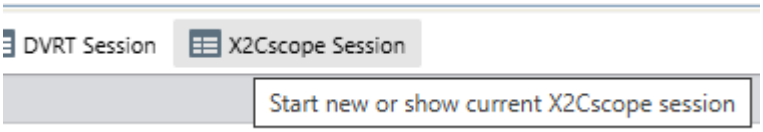
Build your project to generate the ELF file. The ELF file contains the symbol information that MPLAB Data Visualizer uses to identify variables in your firmware.

8.2. Starting an X2Cscope Session

This section describes how to create an X2Cscope session and connect to your target device.

Creating an X2Cscope Session

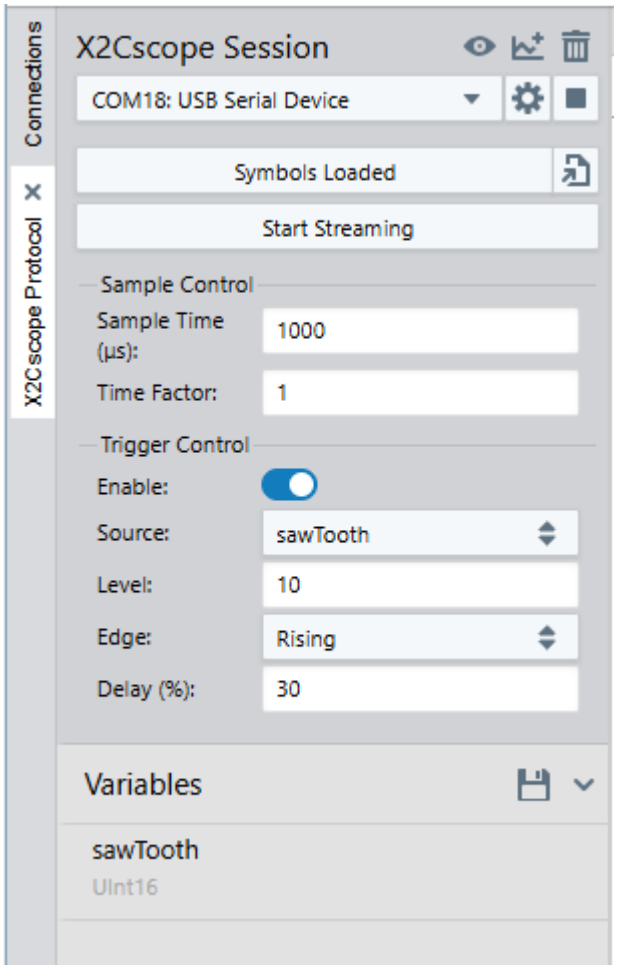
Click the **X2Cscope Session** button in the main Data Visualizer toolbar to create a new X2Cscope session. This opens the X2Cscope panel where you can load symbols, configure the connection, and add variables.



The X2Cscope Panel

The X2Cscope panel contains:

- Connection source selector: For configuring the data source (COM port).
- Load Debug Symbols button: For loading the project ELF File.
- Start/Stop Streaming button: For controlling scope capture.
- Scope timing configuration: Sample time and sample time factor settings.
- Trigger configuration: Variable, type, level, and delay settings.



Connecting to the Target

Selecting the Serial Port

Navigate to the X2Cscope Protocol panel.

Select the COM port connected to your target's UART:

1. Click the connection source selector.
2. Choose the appropriate serial port.

Connection Settings

Configure the baud rate to match your target's UART configuration:

- Baud rate: 115200 (default)
- Data bits: 8
- Stop bits: 1
- Parity: None

For detailed configuration options, see X2Cscope Configuration Options.

8.3. Loading the X2Cscope Project ELF File

To use X2Cscope, the project ELF file must be loaded into MPLAB Data Visualizer. The ELF file contains symbol information that identifies the names, addresses and types of variables in your firmware.

Loading the ELF File

To load an ELF file:

1. Open the X2Cscope panel.
2. Click **Load Debug Symbols**.
3. Select your project's ELF file.

The symbol status indicator shows when symbols are loaded.

The ELF path is monitored for changes. If the ELF file is modified (e.g. on rebuild), the Data Visualizer will prompt you to reload it.

Detailed Instructions

The procedures for loading ELF files are identical for X2Cscope and DVRT. For detailed instructions on:

- Loading ELF files manually in standalone or MPLAB X IDE plugin.
- Reloading after code changes.
- Handling missing symbols.

See Section Loading the DVRT Project ELF file into the Data Visualizer. Follow those same procedures, substituting "X2Cscope" for "DVRT" where applicable.

8.4. Browsing and Selecting Symbols

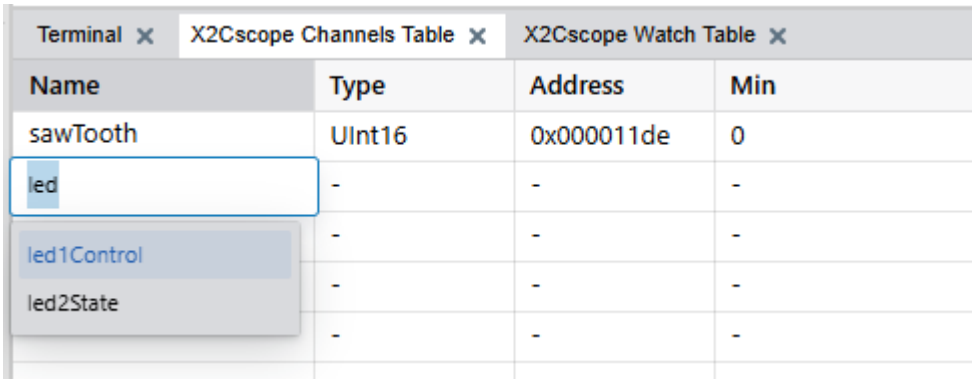
Once the ELF file is loaded, you can browse and select variables to add to the X2Cscope, Scope Channels table, or Watch table (described in the following sections).

The symbol browser provides access to global variables, struct or union members and array elements defined in your firmware.

Selecting Variables

To add a variable to a table:

1. Click a cell in the **Name** column of the table.
2. Start typing to filter the list of available symbols.
3. Select the desired variable from the drop-down.



The selected symbol appears in the table cell with related information (address, type, size) populated automatically.

You can also select struct or union members and individual array elements.

Detailed Instructions

The symbol browser works identically for both X2Cscope and DVRT protocols. For detailed step-by-step instructions on:

- Selecting struct/union members or array elements.

- Managing and reordering table entries.
- Removing variables.

See Selecting Symbols in the DVRT Protocol documentation. The symbol browser interface and procedures are the same for X2Cscope.

8.5. Scope Channels

The Scope Channels table specifies the variables that are captured by the firmware. In X2Cscope terms, a single variable is a Channel.

How Buffered Capture Works

Unlike continuous streaming (DVRT), X2Cscope uses a buffered capture approach:

1. The target MCU samples selected variables at a fixed rate into a user defined buffer. The buffer is defined by the user and a pointer to the buffer is passed during initialization.
2. When the buffer is full, sampling stops.
3. The buffer contents are transferred to the PC for display.
4. The process repeats for the next capture.

This method allows capturing high-frequency signals without requiring high-bandwidth continuous communication.

Adding Scope Channels

Select variables using the symbol browser as described in the Selecting Symbolssection. Up to 8 variables may be selected for the scope function. The number of variables and their sizes (record size) along with the buffer size and the scope configuration determine the duration of one capture cycle.

Figure 8-1. X2Cscope Channels Table Tab

Terminal | X2Cscope Channels Table | X2Cscope Watch Table

Name	Type	Address	Min	Max	Count
sawTooth	UInt16	0x000011de	0	719	2856
led2State	UInt8	0x000011e0	0	1	2856
myStruct.sinus	Float32	0x000011d6	-100.00	100.00	2856
-	-	-	-	-	-
-	-	-	-	-	-
-	-	-	-	-	-
-	-	-	-	-	-
-	-	-	-	-	-

Configuring Scope Sample Time

The **Scope Sample Time** specifies the period between calls to `X2CScope_Update()` in your firmware:

1. Enter the sample period in the Scope configuration (e.g. 1 ms for 1 kHz sampling).
2. Ensure this value matches your firmware's actual update rate.

Important:

This setting tells the PC what rate the firmware is using. It does not change the firmware's sampling rate.

Sample Time Factor

The **Sample Time Factor** reduces the effective sampling rate:

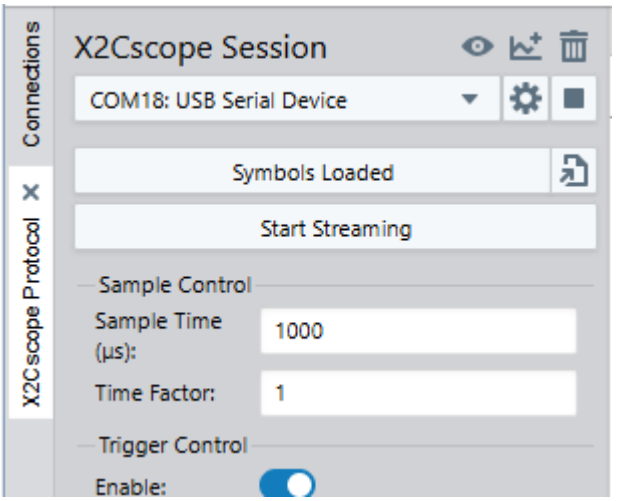
- A factor of **1** captures every sample.
- A factor of **N** captures every Nth sample.

For example, with a 1 ms scope sample time and factor of 10, the effective capture interval is 10 ms. This is useful for:

- Capturing slower phenomena over a longer duration.
- Reducing the amount of data when high time resolution isn't needed.
- Extending capture time without modifying firmware buffer size.

Controlling Data Capture

Figure 8-2. X2Cscope Session



Start and Stop Streaming

1. Click **Start Streaming** to begin continuous capture. The scope repeatedly fills the buffer, transfers data and displays results.
2. Click the button again to stop.

Data capture can run continuously or be controlled by triggers.

Captures repeat automatically, providing updated views of ongoing signals. Single shot mode is currently not supported by the Data Visualizer.

Capture Duration

The total capture duration depends on several factors:

```
Duration = (Buffer Size / Record Size) x Scope Sample Time x Sample Time Factor
```

Where:

- Buffer Size: Determined by firmware configuration (in bytes).
- Record Size: Sum of selected variable sizes.
- Scope Sample Time: Period between sampling.
- Sample Time Factor: Sample frequency divisor.

8.6. Scope Triggers

X2Cscope supports target-side triggers, allowing the MCU to detect trigger conditions and control when data capture occurs. This provides precise triggering on high-speed events that would be difficult to detect on the PC side.

The trigger configuration is located in the X2Cscope Protocol session panel together with the scope configuration.

The values used when configuring trigger settings (trigger level for instance) must be in range of the variable selected as a trigger source, other wise the trigger will not hit but rather loop forever.

Trigger Modes

When triggers are disabled:

- Data capture begins immediately when started.
- The scope updates as soon as the buffer is full.

Use this mode for continuous monitoring without waiting for specific events.

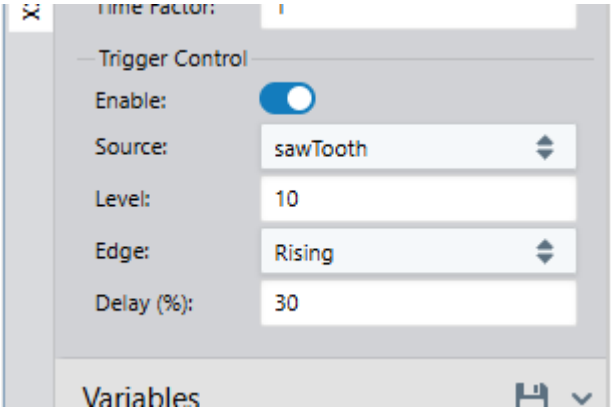
When a trigger is configured:

- Data capture is controlled by detection of a trigger condition.
- Data is captured from the time of the trigger event or as determined by the Trigger Delay setting (see below).

Configuring a Trigger

1. Select the trigger variable from your scope channels. Choose a variable that clearly indicates the event of interest.
2. Choose the trigger type:
   - Rising Edge: Triggers when the value crosses the trigger level going up.
   - Falling Edge: Triggers when the value crosses the trigger level going up.
3. Set the trigger level (the threshold value that the variable must cross to trigger). The level must be within range of the selected variable.

Figure 8-3. Trigger Level



Trigger Delay (Pre/Post Trigger)

The trigger delay controls where the trigger point appears in the captured data, ranging from -50% to 50%:

Delay Setting	Trigger Position	Use Case
0%	Start of capture	See what happens after the trigger.

Scope Triggers (continued)		
Delay Setting	**Trigger Position**	**Use Case**
-50%	Before capture window	See what happens some time after the trigger.
50%	Middle of capture window	See context before and after the trigger.

Adjust the delay to capture the relevant portion of data around your event of interest.

Trigger Configuration Example

To capture data before and after the value of a selected variable exceeds a certain value:

1. Add the trigger variable and other relevant variables to the scope channel table.
2. Select the variable to trigger on in the trigger settings.
3. Select **Rising Edge** trigger type.
4. Set trigger level to the threshold (e.g. 10.0).
5. Set delay to 50% to see data before and after the trigger condition.
6. Click **Start Streaming** and wait for the trigger condition to occur.

Configure the time plot to show the scope variables. The time plot will show the captured data when the trigger condition has been met and the data has been transferred to the PC.

Tips and Troubleshooting

Start with **continuous capture** (trigger disabled) - it's simpler and always provides updates. Use triggered capture when looking for specific events or faults.

Issue	Solution
Never triggers	Verify variable crosses threshold; check edge direction.
Incomplete capture	Adjust pre/post trigger delay.

8.7. Visualizing Scope Data

Captured scope data can be visualized using the standard visualization tools in MPLAB Data Visualizer.

X2Cscope Panel

Variables added to the Scope Channels table are also listed in the Variables section of the X2Cscope panel. From here, variables can be quickly added to visualizations using shortcut buttons.

It is also possible to drag a variable from this area to the desired plot or terminal.

Time Plots

To display scope data on the time plot:

1. Open the **Time Plot** panel if it is not already opened.
2. In the data source selector on the right, find the X2Cscope Scope Channels variables.
3. Select the variable to plot. The plot updates after each capture completes.

Multiple scope variables can be plotted together for comparison. Add separate axis or plot variables on the same axis as a preferred.

See View Data in the Time Plot for more information.

Terminal Output

Scope data can also be displayed in the terminal for numerical inspection. Select the X2Cscope scope variables as terminal data sources.

See View Data as Text in the Terminal for more information.

Update Rate

Unlike continuous streaming protocols, X2Cscope visualizations update after each complete buffer capture. This means the plot and terminal shows the data of the captured time window on each update, rather than scrolling/updating in real-time.

8.8. Watch Table

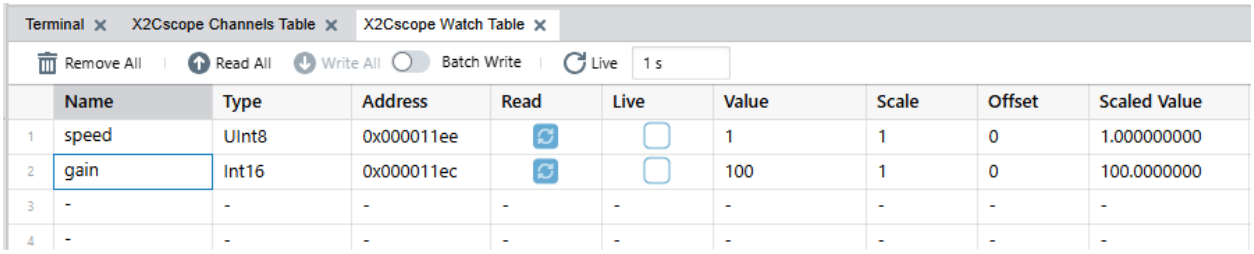
The X2Cscope Watch Table lets you read and write individual variable values on demand. Unlike scope channels, the watch table fetches values only when requested, making it suitable for configuration parameters and slow-changing values.

Overview

The watch table displays a list of selected variables with their current values. This is similar to a debugger's watch window, but operates while the target continues running.

Variables added to the watch table cannot be selected as data sources for the time plot or the terminal.

Figure 8-4. X2Cscope Watch Table



Adding Variables

Select variables using the symbol browser as described in Browsing and Selecting Symbols.

Variables can be reordered by clicking and dragging the move handle in the leftmost column.

Reading and Writing Variables

The Watch Table reads and writes variables only on demand.

- Under **Actions**, click to read the variable value from target. The Value will be displayed in the next column.
- Under **Value**, double-click to edit value and then press **Enter** or click elsewhere to write the value to the target.

After a write operation, the cell will turn green to indicate success or red to indicate failure. In case of failure, check the Messages view for error details.

Batch Read or Write Multiple Variables

Batch operations allow reading or writing multiple variables in sequence, useful for configuring test scenarios or saving/restoring configurations. Multiple rows can be selected using checkboxes in the leftmost column. When batch write mode is enabled, edited cells turn yellow (pending), then green (success) or red (failure) after writing.

For detailed instructions on batch operations, see Section Read and Write Variables in Firmware in the DVRT Protocol documentation. The interface and procedures are identical for X2Cscope.

Automatic Updates (Live Polling)

X2Cscope supports automatic periodic updates for watch variables. Enable the Live checkbox for variables you want to monitor, configure the sample inter val and click the Live toolbar button to start periodic updates.

For detailed instructions on live updates, see Live Updates and Scaled Values in the DVRT Protocol documentation.

Scaled Values

The Watch table can calculate scaled values using custom Scale and Offset factors:

```
Scale Value = Raw Value * Scale + Offset
```

This is useful for converting raw values to engineering units (e.g. ADC counts to voltage) or applying calibration offsets.

For detailed instructions on scaled values and examples, see Live Updates and Scaled Values in the DVRT Protocol documentation.

8.9. X2Cscope Configuration Options

This section covers LNet protocol settings and configuration options for X2Cscope connections in MPLAB Data Visualizer.

Connection Settings

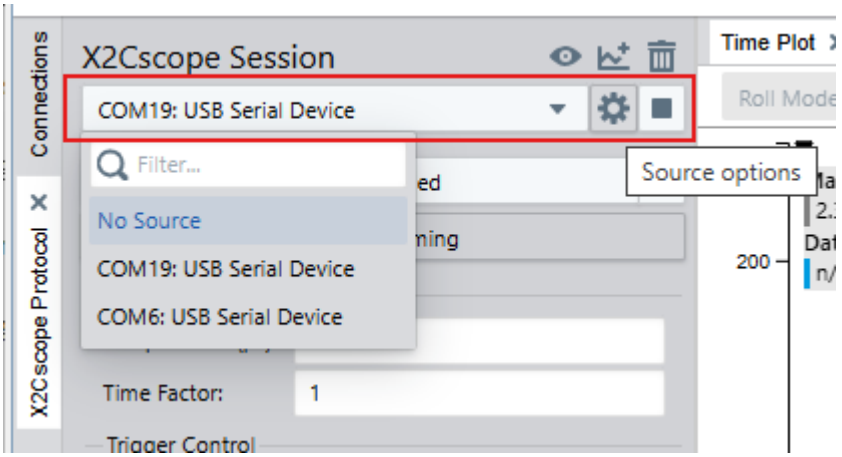
Navigate to the X2Cscope Protocol panel. Select the COM port connected to your target's UART:

1. Click the connection source selector.
2. Choose the appropriate serial port.

Configure the baud rate to match your target's UART configuration:

- Baud rate: 115200 (default)
- Data bits: 8
- Stop bits: 1
- Parity: None

Figure 8-5. USB Serial Device



Scope Configuration

Scope Sample Time: This setting indicates the time between calls to `X2CScope_Update()` in the firmware.

Time factor: This setting configures a divisor for how often a call to X2CScope_Update in the firmware will produce a data sample. 1 means ever y sample, 2 means ever y second sample, and so on.

See Scope Channels for details on sample time and sample time factor settings.

Saving and Loading the Configuration

X2Cscope configuration parameters, along with the content in the X2Cscope Channels and X2Cscope Watch tables, are saved in the Workspace file. The X2Cscope configuration is saved together with other settings like time plot and terminal settings, panel layout etc.

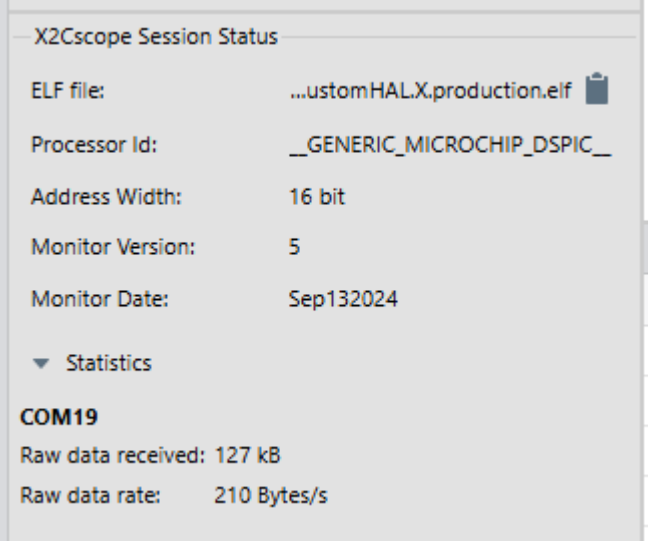
The workspace file can be loaded at a later time to recreate the setup.

Status Information

When connected, the target may provide device information:

- Device type/identifier
- Firmware version

This information helps verify you are connected to the correct target.



Troubleshooting

Issue	Possible Cause	Solution
Frequent timeouts	Slow main loop	Ensure `X2CScope_Communicate` is called frequently.
Checksum errors	Baud rate mismatch	Verify baud rate matches firmware configuration.
No response	Wrong COM port	Check port selection in Device Manager.
Intermittent connection	Electrical noise	Use shorter cables, add filtering.
Wrong time scale	Sample time mismatch	Verify scope sample time matches firmware.

9. Log and Save Data To a File

The MPLAB Data Visualizer offers two methods for saving data to disk. The Log Data to File feature enables continuous logging of data to file, as the data is being captured.

The snapshot function (see Saving Already Captured Data) enables selecting and exporting a range of data that has already been captured and plotted in the time plot.

9.1. Data Logging

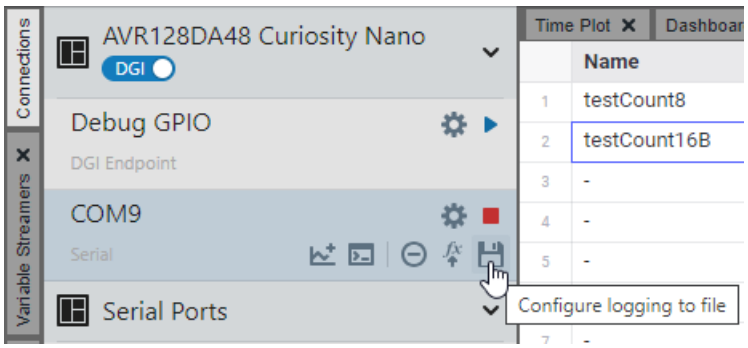
Logging captured data to disk is intended to work with any data source with these exceptions/notes:

- Power DGI data logging is not supported
- Debug GPIO data is logged as the combined bus value of the GPIO lines

9.1.1. Configure, Enable and Disable Logging

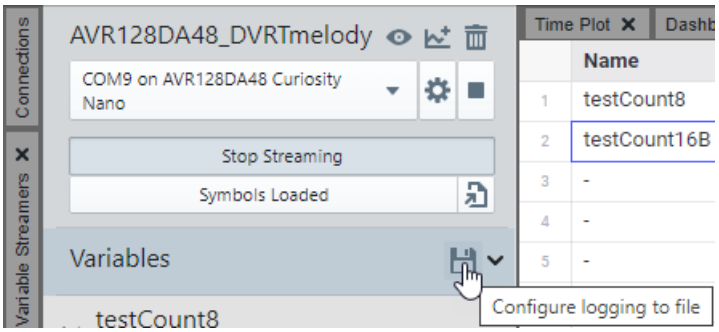
To configure logging for DGI or COM connections on the **Connections** tab, click on the disk button  in the button row that is displayed when hovering over a connection's section in the panel.

Figure 9-1. Configure Logging - Connections



To configure logging for the **Variable Streamers** or **DVRT Protocol** tab, click on the disk button in the Variables section header.

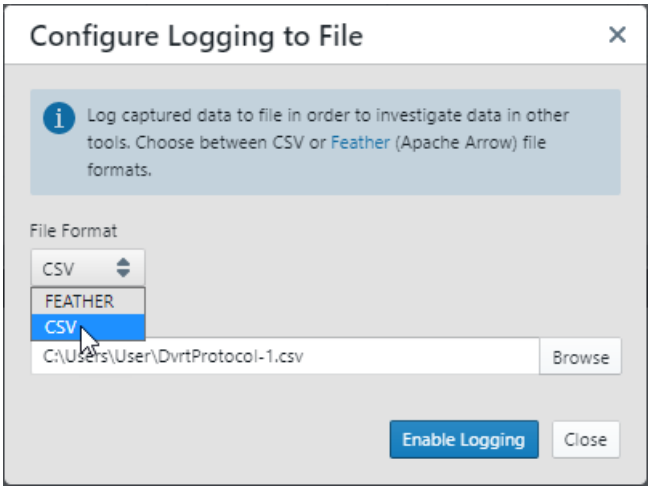
Figure 9-2. Configure Logging - Variable Streamers or DVRT Protocol



Once the disk button is clicked, the **Configure Logging to Files** dialog will open.

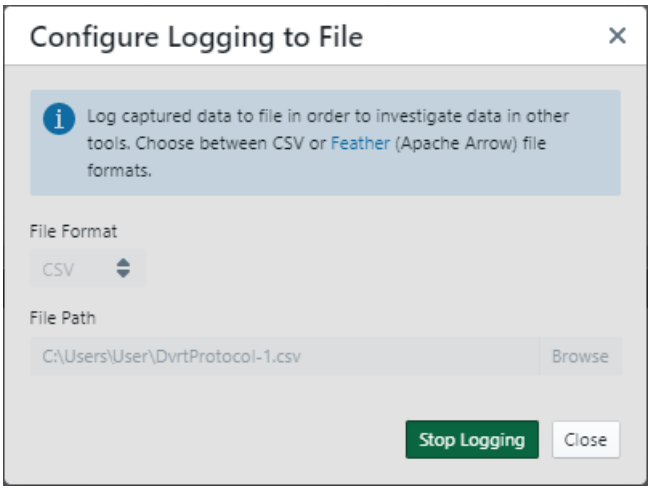
1. Select the file format. Options are **CSV** or **Feather**. Click the link on the dialog for more information on Feather format. Also see About Log Files.
2. Select the file path. A default path is selected, but you may enter or browse to another.
3. When done, click **Enable Logging** to begin logging data or **Close** to save your settings but not enable logging.

Figure 9-3. Log File Configured in Dialog



4. If **Enable Logging** was selected, the dialog button will now display **Stop Logging** and the input fields will be disabled. Click the button to stop logging or click **Close** to close the dialog.

Figure 9-4. Logging Enabled in Dialog



5. If the dialog has been closed and logging is still enabled, the disk button will be green. To stop logging, click on the button.

Figure 9-5. Green Disk - Connections

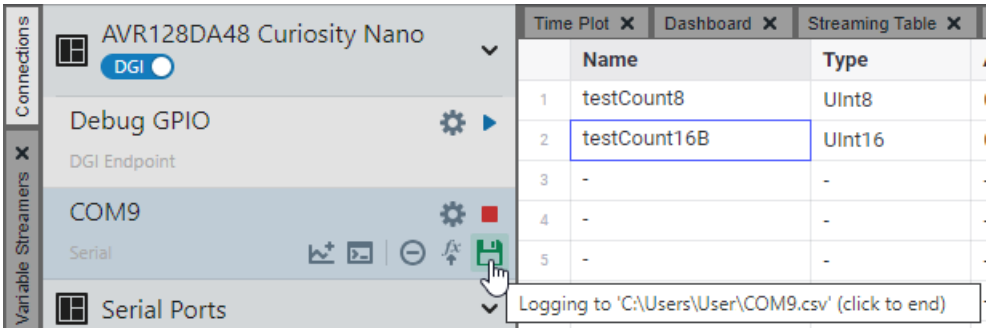
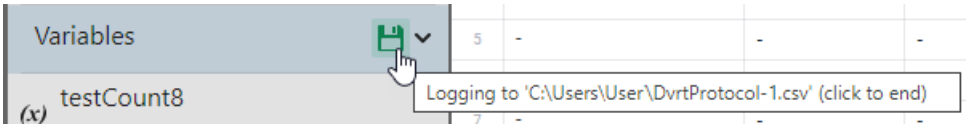
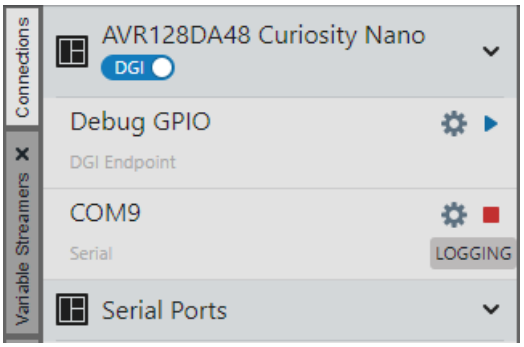


Figure 9-6. Green Disk - Variable Streamers or DVRT Protocol



In addition, on the **Connections** tab, the label **Logging** is displayed when logging and capturing are both active.

Figure 9-7. Logging - Connections



9.1.2. About Log Files

Data can be buffered in memory before it is logged to disk. Any buffered data is written to disk when:

- logging is stopped
- the data source is removed
- the data visualizer workspace is cleared or the data visualizer is closed down

The actual creation of the log file on disk can be delayed as described above. If no data was captured while logging was enabled, no file is created.

When logging ends and the file on the disk is closed, a small message is displayed in the UI:

If a problem occurred, a corresponding warning message will be displayed.

Formats available for storing logged data into a file are discussed in the following sections.

9.1.2.1. Log File: CSV Format

CSV (comma-separated values) is a format available for storing logged data into a file.

For CSV, the output file will contain a header row listing the names of the respective data columns, followed by rows containing the actual data. A comma is used to separate values for the individual columns.

Example: CSV file produced for the DVRT session shown in DVRT Protocol.

```
timestamp,demo_uint8,demo_float
1019.643276,24,1217187.0
1019.7512378,34,1217199.5
1019.7632439,45,1217213.2
1019.8713035,55,1217225.8
1019.979245,65,1217238.2
1020.0872369,75,1217250.8
1020.1952683,85,1217263.2
```

```
1020.3032743,95,1217275.8
1020.411237,105,1217288.2
```

Timestamps describe the time since the data visualizer backend process was started, in seconds. They are represented as decimal numbers in the CSV format.

9.1.2.2. Log File: Feather Format

Feather (Apache Arrow) is a format available for storing logged data into a file.

Feather data is ZSTD compressed and is directly consumable by several external tools and scripting languages such as Python (see example below.) For more information see arrow.apache.org/ and, for example, arrow.apache.org/cookbook/py/io.html#reading-a-feather-file for Python examples.

Example 2: Dump of a Feather file using `feather-check.py`.

```
c:\temp>python feather-check.py DvrtProtocol-1.feather
Length: 2
Reading DvrtProtocol-1.feather as pyarrow table
pyarrow.Table
timestamp: double not null
demo_uint8: uint8 not null
demo_float: float not null
----
timestamp:
[[4458.8244819,4458.9324957,4459.0405149,4459.1484745,4459.256488,...,4482.0444293,4482.152426
6,4482.260458,4482.3684368,4482.4764697]]
demo_uint8: [[151,161,172,182,192,...,9,19,29,39,49]]
demo_float:
[[1617665.8,1617678.2,1617692,1617704.5,1617717,...,1620368.2,1620380.8,1620393.2,1620405.8,16
20418.2]]
Total number of rows: 219
```

feather-check.py

```
import sys
import pyarrow.feather as pf

print("Length: " + str(len(sys.argv)))

if (len(sys.argv) < 2):
    print("Usage: feather-check.py <file_name>")
    sys.exit(1)

fileName = sys.argv[1]
print("Reading " + fileName + " as pyarrow table")

table = pf.read_table(fileName)

print(table)
print("Total number of rows: " + str(len(table)))
```

Timestamps describe the time since the data visualizer backend process was started, in seconds. They are represented as 64-bit floating point data in the Feather format.

Feather Metadata

For schema root and field meta information, MPLAB Data Visualizer uses this convention to avoid collisions with user defined values:

`{ "data-visualizer": [UTF-8 encoded JSON string object] }`

At the schema root level, there will be a meta info object with a data-visualizer key with JSON values:

- encoding: `{ version: int; type: "basic" }`
- timestamp: `{ format: ["integer", "float"]; units: "seconds"; encoding: ["relative", "delta"]; resolution: float }`
- time-origin-ms-since-epoch: long

For protocol fields, meta information will have a data-visualizer key with JSON values:

- `{ units: string }`

Meta Examples

Schema root meta:

```
"data-visualizer": {"encoding": {"version": 1, "type": "basic"}, "time-origin-ms-since-epoch": 1731113302508, "timestamp": {"format": "integer", "units": "seconds", "encoding": "relative", "resolution": 1e-06}}
```

Timestamp (1 µs resolution integer encoding) field meta:

```
"data-visualizer": {"timestamp": { "format": "integer", "units": "seconds", "encoding": "relative", "resolution": 1e-06}}
```

Voltage field meta:

```
"data-visualizer": {"units": "V" }
```

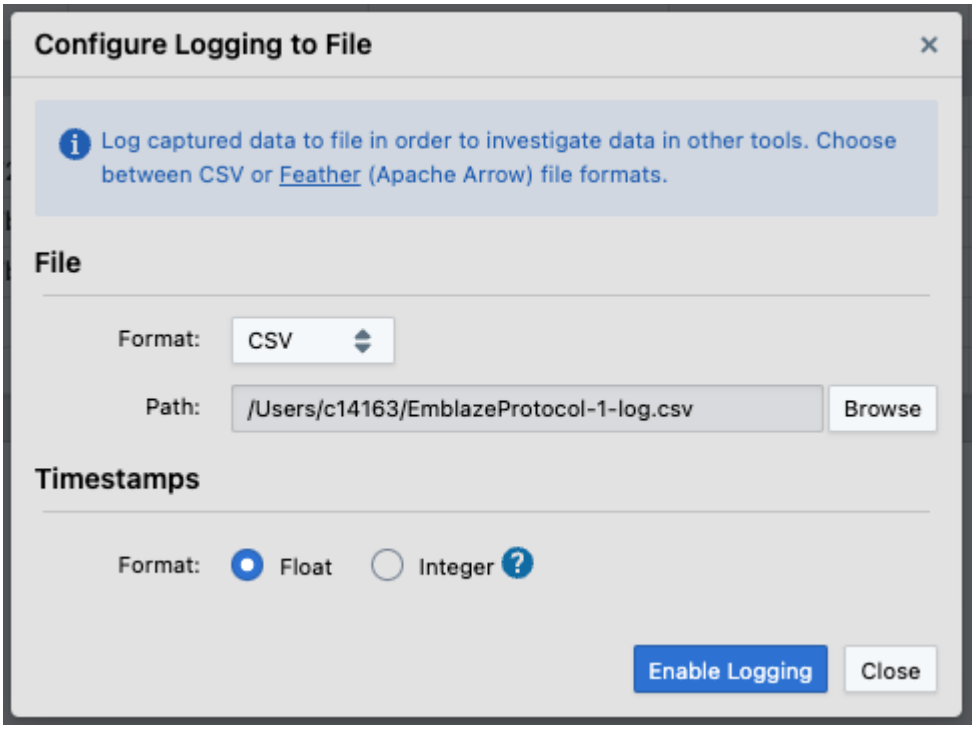
The next section contains more information about integer timestamp configuration and example Python code for decoding.

9.1.3. Optimizing File Sizes

Integer Timestamps

Timestamps are formatted as float values by default. When floating point timestamps are logged during a longer running or high data rate capture, they take up a lot of disk space and are hard to compress. This results in large file sizes. Configuring the data logger to format timestamps as integers can result in smaller files.

To navigate to the timestamp configuration section, go to the log file configuration dialog:



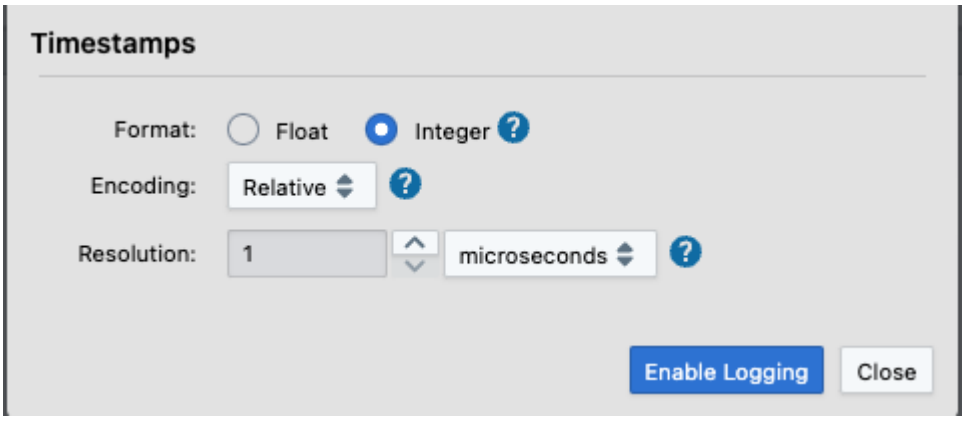
The default configuration values for Integer change when timestamp format integer is first selected for a given data source or frame:

- Encoding: Relative
- Timestamps Resolution: 1

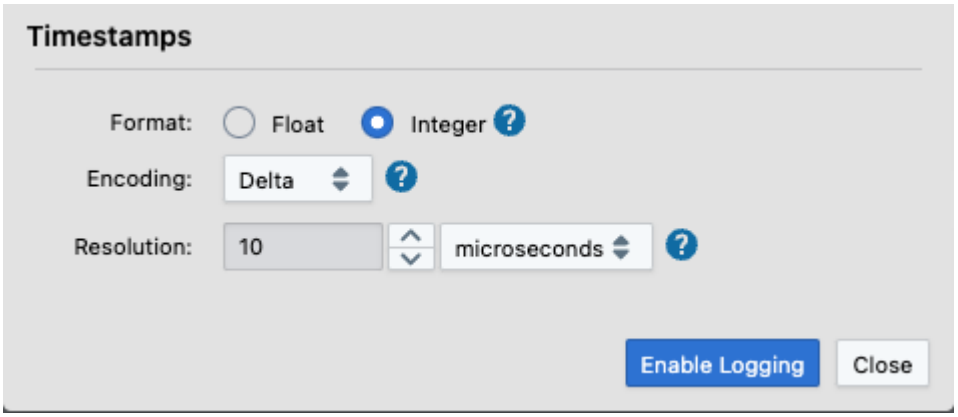
- Resolution Unites: Microseconds

These are the two resulting modes and their functionality:

- Relative: Each timestamp is converted to an integer and calculated as an offset to the first timestamp in the log file. Then, it is encoded according to the timestamp resolution and units values. For example, timestamp values of [33.000001, 33.000002, 33.000003] will be encoded as [0, 1, 2]



- Delta: Each timestamp is converted to an integer and calculated as an offset to the previous timestamp in the log file. Then, it is encoded according to the timestamp resolution and units values. For example, timestamp values of [33.000001, 33.000002, 33.000003] will be encoded as [0, 1, 1]



Integer Timestamp Decoding Example

This code example shows how to decode integer timestamps as floating point values from a feather log:

```
import sys
import numpy as np
from pyarrow import feather as pf
import json

def read(filename):
  table = pf.read_table(filename)
  df = table.to_pandas()
  print(df)

  timestamps = df['timestamp']
  meta = json.loads(table.schema.metadata[b'data-visualizer'])
  timestamp_meta = meta['timestamp']
  format=timestamp_meta['format']
  if format == 'integer':
    encoding = timestamp_meta['encoding']
    factor = float(timestamp_meta['resolution'])
    if encoding == 'delta':
      timestamps = np.cumsum(timestamps)*factor
```

```
    elif encoding == 'relative':
      timestamps = np.multiply(timestamps, factor)

    print('decoded timestamps\n{}'.format(timestamps))

if __name__ == "__main__":
  if (len(sys.argv) < 2):
    print("Usage: decode_timestamps.py <file_name>")
    sys.exit(1)

  filename = sys.argv[1]
  print("Reading " + filename + " as pyarrow table")
  read(filename)
```

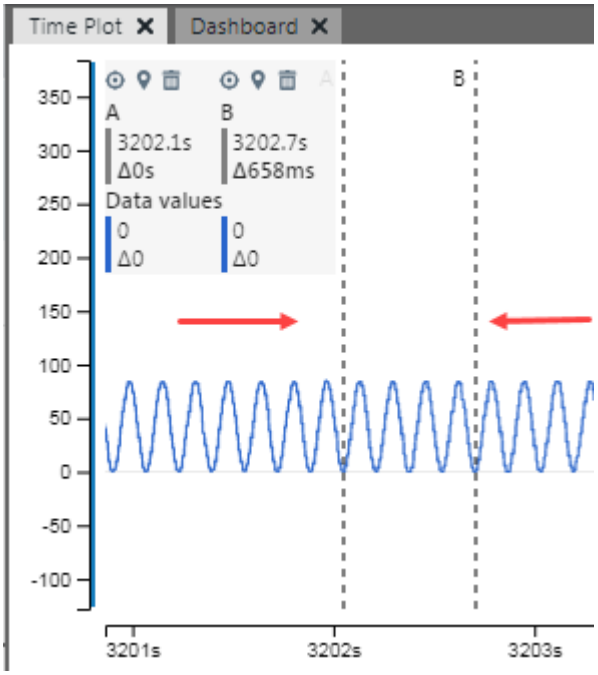
9.2. Saving Already Captured Data

The snapshot function of the time plot supports selecting and exporting a range of data that has already been captured. The data snapshot can be output to a text file in CSV or JSON format. To capture the plot data, use the **Mark Range** and **Save Snapshot** buttons in the time plot toolbar. The **Save Snapshot** button will be disabled when there are no sources plotted.

Figure 9-8. Mark and Snapshot Buttons

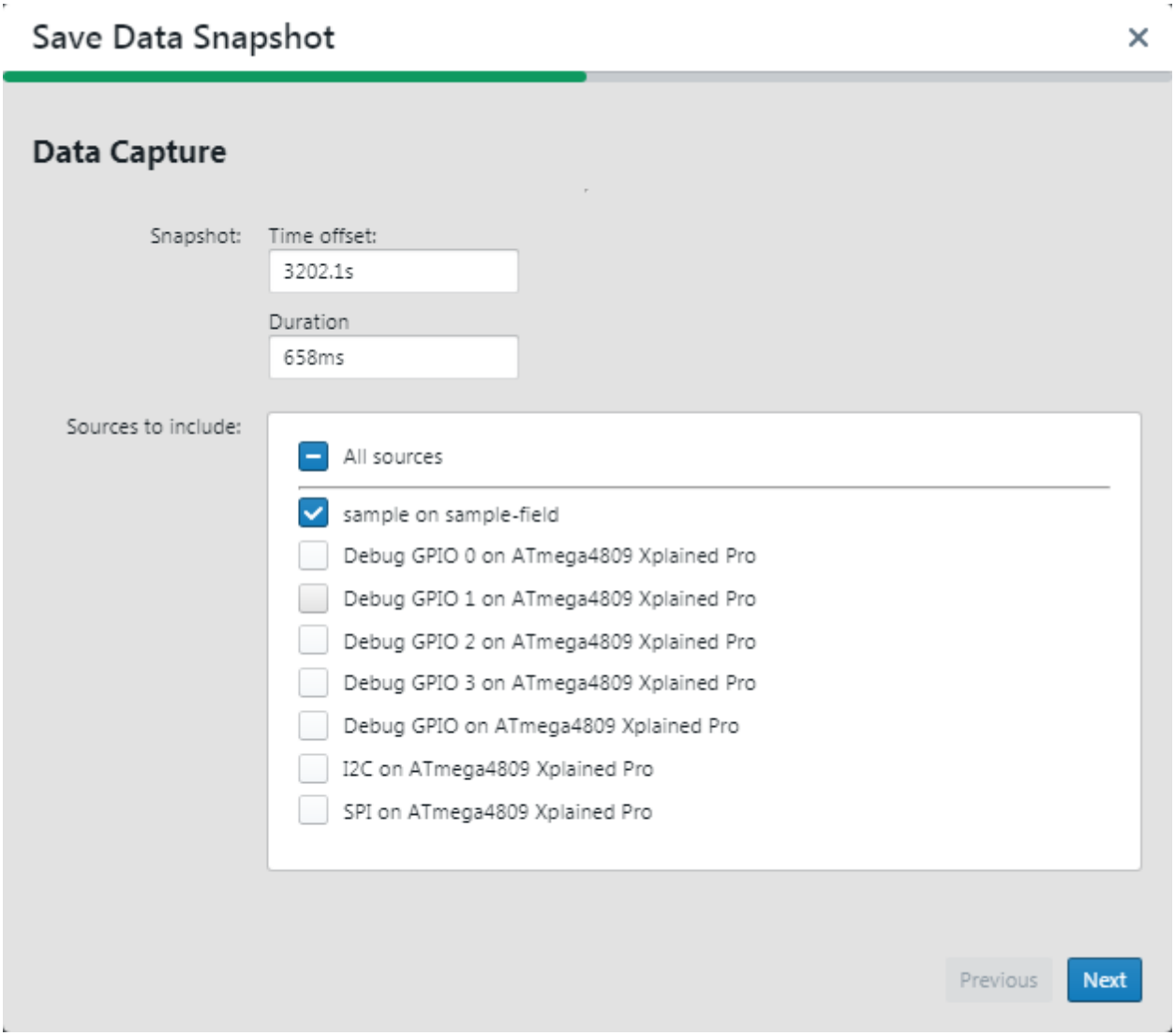
To capture a snapshot of plot data, first click on **Mark** to halt plot scrolling and set cursors (A and B) to mark a range in the plot window. The marked range will cover about 70 per cent of the visible plot. Adjust the cursors as desired, as shown in the figure.

Figure 9-9. Mark for Snapshot Area



Then click **Snapshot** to open the Snapshot wizard to select data and format output.

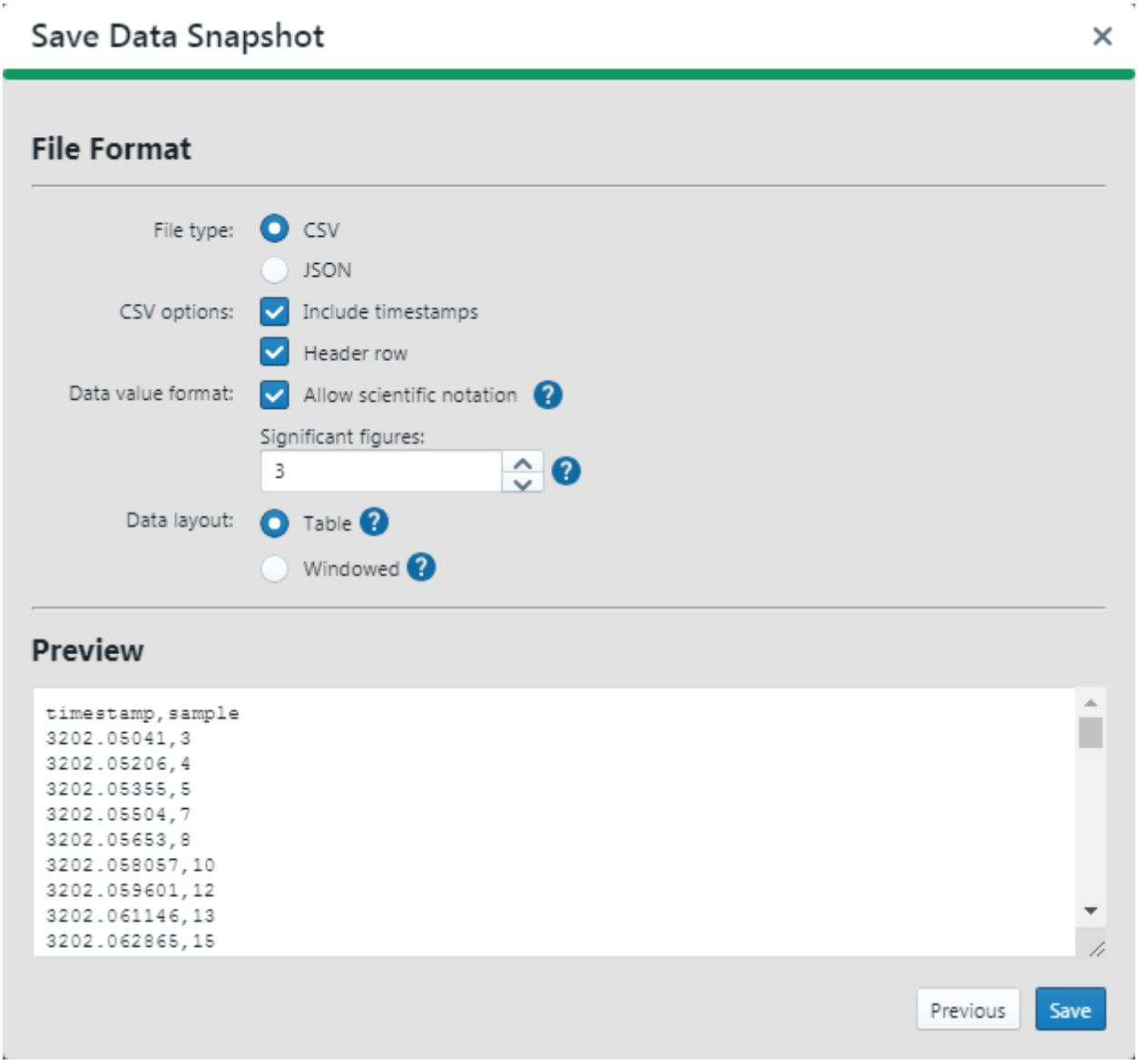
Figure 9-10. Snapshot - Data Capture



The Snapshot time offset and duration are set by the location of the cursors. Snapshot data sources may be chosen from the available list.

Note: The start time shown is the internal data timestamp that corresponds to the start of the range that is marked in the plot. By default, the time plot's time axis and cursor boxes show time offsets, not data timestamps. Toggle the **Show Timestamps** option in the time plot settings to also show timestamps in the plot.

Figure 9-11. Snapshot - File Format



Select one available output file type. Other options are determined by this selection. To learn more about each option, hover over the question mark icon next to the option for a tooltip.

The Preview window will show you the result of selections made. When you are done, click **Save**. By default the file is saved to your User directory.

10. Troubleshooting

See an MPLAB Data Visualizer component below to find tips on troubleshooting related issues. See also the general “Special Considerations” section.

10.1. Data Streaming

SPI over DGI

If the data waveform doesn't match expectations or can't be decoded by the Data Stream Decoder, try enabling “Force synchronization on CS.” Sync issues are especially common when starting a debug session on an Xplained Pro development kit and then streaming over SPI from the same kit.

Plotting Resources

If your machine is having trouble plotting multiple data streams, such as voltage, current, and power:

- Stop the data stream when performing analysis.
- Zoom out on the time plot while live data is scrolling.
- Select fewer plot sources by using the controls on the right pane.
- Remove one or more plots.

Each of these suggestions cuts down on the system resources required.

10.2. Data Stream Decoder

No Data Input

Make sure that your data source is connected and transmitting data. An easy way to verify this is to plot the raw data stream or display the data in the Terminal. If there are no values in the plot or in the terminal, then data is likely not being received.

Decoder Mismatch

If you're getting a Variable Streamer mismatch error, make sure that the variables in the Variable Streamer match the incoming data exactly. The decoder expects a start of frame (SoF) byte, followed by a sequence of bytes that match the fields defined in the Variable Streamer, and an end of frame (EoF) byte which is a one's complement of SoF. The following situations would prevent a data packet from being decoded:

- Either the SoF or EoF bytes are not present
- Any of the fields are missing from a data packet transmission
- The size of the variable doesn't match that in the Variable Streamer

Each variable must have the number of bytes that match the data type. For instance, `Int8` would be one byte and `Int16` would be 2 bytes.

Sometimes the mismatch error will clear itself even though there’s still a mismatch between the data frame sent by the target and the Variable Streamer in DV. This can happen when there is a small difference in byte lengths between the data frames. Check each field size to be sure they match.

[Start of Frame character][Data Field 1 * data type size in bytes][Data Field 2 * data type size in bytes]...[Data Field n * data type size in bytes][End of Frame character]

Example:

For two `Int16` fields, x and y, the protocol byte sequence would look like this:

[SoF][x1][x0][y1][y0][EoF]

A data stream that matches this protocol might look like this:

[0x3][0x34][0x12][0x78][0x56][0xFC]

The decoded x and y values would be 0x1234, 0x5678.

10.3. Special Considerations

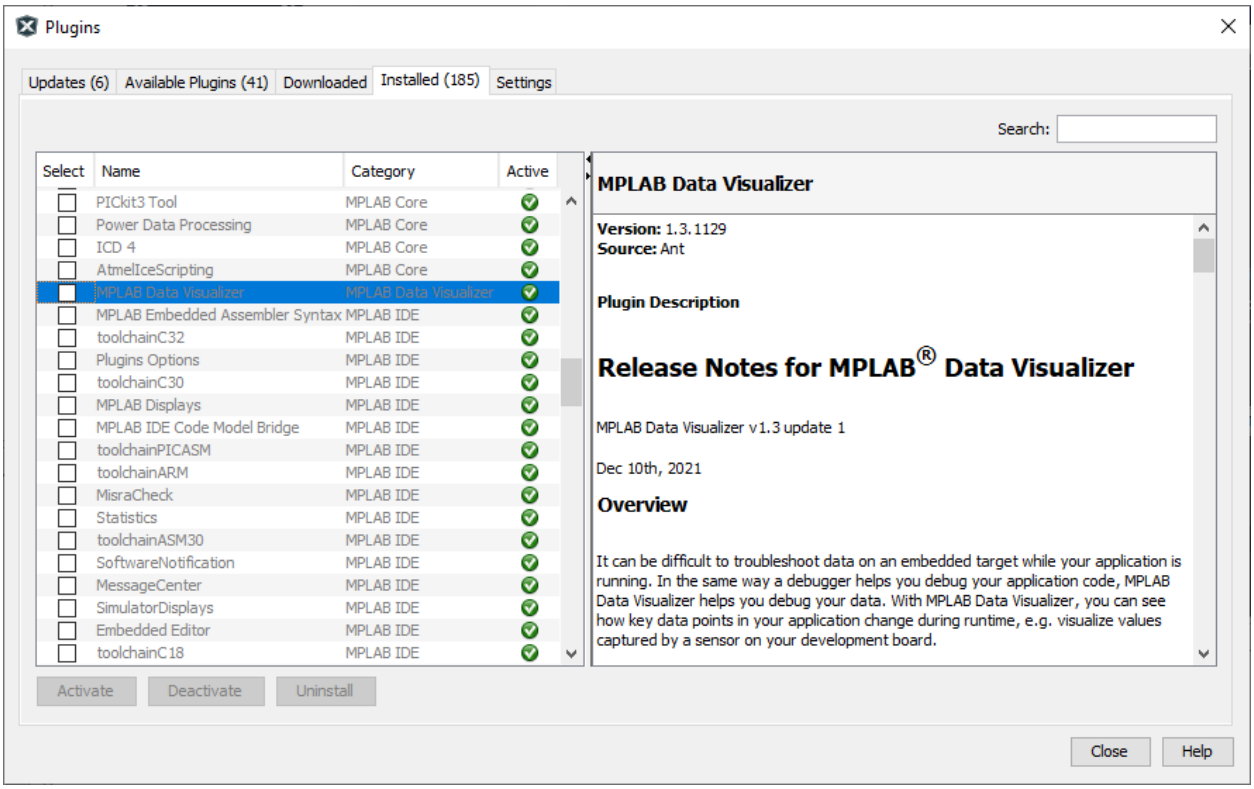
- When streaming timestamped data over DGI from an Xplained Pro or Curiosity Nano kit, too high data rates can lead to buffer overflows and reduced stability of operation. This also applies to Debug GPIO and Power Data. The Data Visualizer UI will display a warning when the kit or tool indicates a buffer overflow condition. The sample rate at which buffer overflows are likely to occur depends on the PC and the system load. For GPIO it could be down to a few kHz sample rate at system load. For other interfaces it is typically higher.
- To get the expected result when streaming data, it is important to configure the target system properly. Study the user guide of the target development board or tool carefully as it can contain important information.
- The standalone MPLAB Data Visualizer may interrupt an active debug session on a development kit on start up. If a debug session is in progress in MPLAB X on a DGI-enabled kit when MPLAB Data Visualizer standalone is started, the debug session may be interrupted and the kit might not show up as a connection in the standalone version. The workaround is to close MPLAB X and unplug and plug in the kit. For simultaneous debugging and data visualization, please use the MPLAB X plugin version of MPLAB Data Visualizer (*Window>Debugging>Data Visualizer*).

11. Finding Release Notes

Release notes for the MPLAB Data Visualizer can be found:

- for the plugin, under *Tools>Plugins>Installed>MPLAB Data Visualizer*.
- for the standalone program, under the Help menu.

Figure 11-1. Plugin Release Notes



12. Example of Plotting Data - Code Listing

Example Header and C code for the ATmega4809 Xplained Pro project may be found in the following sections.

Note: Care should be taken when copying across PDF pages, as the page footer may appear in the code listing. Instead copy from WebHelp version of documentation.

12.1. C Header Code

configure.h

```
/*
 * File:   configure.h
 * Author: Microchip Technology Inc.
 *
 * Created on September 20, 2018, 11:00 AM
 */

#ifndef CONFIGURE_H
#define CONFIGURE_H

#ifdef __cplusplus
extern "C" {
#endif

void initializePeripherals();

#ifdef __cplusplus
}
#endif

#endif  /* CONFIGURE_H */
```

memutil.h

```
/*
 * File:   memutil.h
 * Author: Microchip Technology Inc.
 *
 * Created on September 19, 2018, 1:03 PM
 */

#ifndef MEMUTIL_H
#define MEMUTIL_H

#ifdef __cplusplus
extern "C" {
#endif

#define LEN(a) (sizeof(a) / sizeof(*a))

#ifdef __cplusplus
}
#endif

#endif /* MEMUTIL_H */
```

pins.h

```
/*
 * File:   pins.h
 * Author: Microchip Technology Inc.
 *
 * Created on September 19, 2018, 11:22 AM
 */

#ifndef PINS_H
#define PINS_H

#ifdef __cplusplus
extern "C" {
```

```
#endif

#define MISO_PIN  5
#define MOSI_PIN  4
#define CS_PIN  3
#define SCK_PIN  6

#ifdef __cplusplus
}
#endif

#endif  /* PINS_H */
```

spi.h

```
/*
 * File:   spi.h
 * Author: Microchip Technology Inc.
 *
 * Created on September 19, 2018, 11:21 AM
 */

#ifndef SPI_H
#define SPI_H

#ifdef __cplusplus
extern "C" {
#endif

void init_spi0(void);

void select_dgi_spi(void);
void deselect_dgi_spi(void);
void tx_spi0(uint8_t tx_usart1);
void tx_string_spi0(char* tx_string);
void tx_data_spi0(uint8_t tx_byte[], int length);

#ifdef __cplusplus
}
#endif

#endif  /* SPI_H */
```

timer_callback.h

```
/*
 * File:   timer_callback.h
 * Author: Microchip Technology Inc.
 *
 * Created on September 19, 2018, 11:15 AM
 */

#ifndef TIMER_CALLBACK_H
#define TIMER_CALLBACK_H

#ifdef __cplusplus
extern "C" {
#endif

void timer_callback();

#ifdef __cplusplus
}
#endif

#endif  /* TIMER_CALLBACK_H */
```

12.2. C Source Code

configure.c

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/cpufunc.h>
#include "timer_callback.h"
```

```
#include "spi.h"
#include "pins.h"

void init_sysclock(void);
void init_tcb0(void);

#define PORTB_PIN2CTRL  _SFR_MEM8(0x0432)

void initializePeripherals() {
    init_sysclock();
    init_tcb0();
    init_spi0();

    PORTB_PIN2CTRL |= 0x8;
}

void init_sysclock(void)
{
    CPU_CCP = 0xD8;
    CLKCTRL_MCLKCTRLB = 0x00;
}

void init_tcb0(void)
{
    TCB0.CTRLA = TCB_CLKSEL_CLKDIV2_gc;  // base clock 16Mhz / 2 = 8 MHz
    TCB0.CTRLB = 0x00;  // all the defaults
    TCB0.CCMP = 7999;  // 8Mhz / 8000 = 1 kHz
    TCB0.INTCTRL = 0x01;  // enable interrupt
    TCB0.CTRLA |= 0x01;  // enable timer
}
```

main.c

```
/*
 * File:   main.c
 */

#include <stdio.h>
#include <stdlib.h>
#include "configure.h"
#include <avr/interrupt.h>

int main(int argc, char** argv)
{
    initializePeripherals();
    sei(); //set global interrupt flag

    while(1) ;
}
```

sine_app.c

```
/*
* There are two waveforms in this application:
* 1. sine wave
* 2. triangle wave
*
* There are two global variables for control in this application:
*  - amp_factor - this defines the amplitude of the waveform
*  - wave_select - this defines the waveform selection.
*    It can either be 0 for sine or 1 for triangle.
*/

#include <stdio.h>

#include "timer_callback.h"
#include "spi.h"
#include "memutil.h"

int amp_factor = 1;
int wave_select = 0;
int counter = 0;

int sine[] = {
    0x2b,0x2d,0x30,0x32,0x35,0x38,0x3a,0x3d,
    0x3f,0x41,0x43,0x46,0x48,0x49,0x4b,0x4d,
```

```
    0x4e,0x50,0x51,0x52,0x53,0x54,0x54,0x55,
    0x55,0x55,0x55,0x55,0x54,0x54,0x53,0x52,
    0x51,0x50,0x4e,0x4d,0x4b,0x49,0x48,0x46,
    0x43,0x41,0x3f,0x3d,0x3a,0x38,0x35,0x32,
    0x30,0x2d,0x2b,0x28,0x25,0x23,0x20,0x1d,
    0x1b,0x18,0x16,0x14,0x12,0xf,0xd,0xc,
    0xa,0x8,0x7,0x5,0x4,0x3,0x2,0x1,
    0x1,0x0,0x0,0x0,0x0,0x0,0x1,0x1,
    0x2,0x3,0x4,0x5,0x7,0x8,0xa,0xc,
    0xd,0xf,0x12,0x14,0x16,0x18,0x1b,0x1d,
    0x20,0x23,0x25,0x28,0x2b
};

int tri_1k[] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1};

struct
{
    int cnt;
    int *amp;
} waveform[] =
{
    {LEN(sine), sine},
    {LEN(tri_1k), tri_1k}
};

void timer_callback()
{
    uint8_t sample = (amp_factor * waveform[wave_select].amp[counter]) & 0x7F;
    if (++counter >= waveform[wave_select].cnt) {
        counter = 0;
    }

    select_dgi_spi();
    tx_spi0(0x03);
    tx_spi0(sample);
    tx_spi0(0xFC);
    deselect_dgi_spi();
}
```

spi.c

```
#include <avr/io.h>
#include "spi.h"
#include "pins.h"

void init_spi0(void)
{
    VPORTA.DIR |= (1 << MOSI_PIN) | (1 << SCK_PIN);
    VPORTF.DIR |= (1 << CS_PIN);

    SPI0.CTRLA = 0 << SPI_CLK2X_bp | 0 << SPI_DORD_bp | 1 << SPI_MASTER_bp |
SPI_PRESC_DIV64_gc;
    SPI0.CTRLB = (1 << SPI_SSD_bp);  // disable SS#
    SPI0.CTRLA |=  1 << SPI_ENABLE_bp;
}

void select_dgi_spi(void)
{
    VPORTF.OUT &= ~(1 << CS_PIN);
}

void deselect_dgi_spi(void)
{
    VPORTF.OUT |= (1 << CS_PIN);
}

void tx_spi0(uint8_t tx_byte)
{
    uint8_t tx_rdy = 0;

    SPI0.DATA = tx_byte;
    while(!tx_rdy)
        tx_rdy = (SPI0.INTFLAGS & SPI_IF_bm );
}

void tx_string_spi0(char* tx_string)
{
```

```
    while (*tx_string)
        tx_spi0(*(tx_string++));
}

void tx_data_spi0(uint8_t tx_byte[], int length)
{
    while (length--)
        tx_spi0(*(tx_byte++));
}
```

timer_loop.c

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/cpufunc.h>
#include "timer_callback.h"

unsigned char period = 1;
unsigned char tick = 0;

ISR(TCB0_INT_vect)
{
    TCB0.INTFLAGS = 0x01;

    if (++tick > period) {
        tick = 0;
        timer_callback();
    }
}
```

13. Revision History

The following is a list of changes by version to this document.

13.1. Revision I (June 2026)

Updated Curiosity Nano Explorer section with note about the Power Monitor.

Updated Power Monitor Settings table with additional information under Usage.

Updated Widget Overview in the Dashboard Widgets Tab:

- Created new Protocol Session section
- Created new Read Variables section
- Created new Write Variables section

Modified titles and content in Configuring Widgets section and Sending Values section.

Updated content in Connections Tab section.

Modified titles and content in Connections Tab section.

Created new chapter for X2Cscope Protocol.

Created new chapter for Protocol Comparison.

13.2. Revision H (August 2025)

In the External Connections section:

- Introduction was updated with new information relation to the Curiosity Nano Explorer
- Section Curiosity Nano Explorer was added
- Numerous figure titles were added

In the Visualization section:

- A note was added to the Graphic Widget section
- In section 4.2.3, some new features were added to the table
- Sections 4.7.1.1.1-8 were moved up one level in the topic hierarchy
- Numerous figure titles were added

13.3. Revision G (January 2025)

In the 3. External Connections section:

- 3.1 Connection Options was modified to add more context about source options.
- The screenshot was updated to reflect the new options in 3.2 Data Gateway Interface (DGI)

In the 4. Visualization section, the entire 4.3 Time Plot section has been restructured to reflect the new Time Plot and accompanying features:

- 4.3.1.1 Time Plot Mode
- 4.3.1.2 Running and Stopped Mode
- 4.3.1.3 Panning and Zooming
- 4.3.1.4 Add Data Axes
- 4.3.1.5 Configure Data Axes and Plots
- 4.3.1.6 View Plot Values
- 4.3.1.7 Automatically Adjust Data Axes
- 4.3.1.8 Marking Data and Saving a Data Snapshot

- 4.3.1.9 Inspecting Power Data and PC Profiling
- 4.3.1.10 Other Time Plot Settings

In the 6. DVRT Protocol section, new subsections were added:

- 6.4 Stream Variable Values was developed to add new subsections
- 6.7 Read and Write Variables in Firmware
- 6.8 Batch Read or Write Multiple Variables
- 6.10 Comparison of the Data Streamer and DVRT Protocols

In the 7. Log and Save Data to a File Section, new subsections were added:

- 7.1 Data Logging
- 7.2 Saving Already Captured Data
- 7.1.2.2 Log File: Feather Format was developed further to add more content about feather metadata

13.4. Revision F (February 2024)

- **4.2.1 Terminal Window**: History feature added to Line input.
- **4.7.1 Dashboard Window**: Added details on connecting a widget to a source.
- **4.7.1.1.4 Rotating Widgets**: Widgets may now be rotated around the center of the widget.
- **4.7.2.1 Sending Values from Output Widget to Input Widget**: In addition to other sources, an output widget may be connected to an input widget.
- **4.7.2.2.1 Label**: Additional label formatting.
- **4.7.2.3.2 Button**: Button press function update.
- **4.7.2.3.3 Slider**: “Send on Release” option added.
- **6.2.3 Seeing ELF File Name in DVRT Panel** and **6.8 DVRT Protocol Options and Status Information**: Content combined; name of ELF file added under Status.
- **7. Log and Save Data To a File**: New section on feature to log data values and then save values to a file.

13.5. Revision E (January 2023)

- **2.2.1.1 Toolbar Controls**: Updates to controls structure.
- **2.2.1.2 Notifications and Messages**: Added new section.
- **4. Visualization**: Added information on the “Start visualizing data” dialog.
- **4.1.1 Terminal Window**: Added information about updates to this window when used for input to target.
- **4.2.2 Connections Tab**: Added a note that stopping Time Plot streaming does not stop streaming for Widget Plots.
- **4.5.3.1 MPLAB Touch Plugin Example**: Changed this section to be more relevant to XY Plot. Also referenced discussion in Developer Help.
- **6. DVRT Protocol**: New chapter for the run-time configurable Data Visualizer Run Time (DVRT) protocol for 8- and 16-bit targets.

13.6. Revision D (January 2022)

- DGI Debugger Profiling with PC Sampling available using the MPLAB Data Visualizer plugin with SAM and AVR devices.
- Noise in power plots addressed.

13.7. Revision C (November 2021)

- Dashboards with widgets for streaming data to and from the target.
- Support for importing Atmel Data Visualizer dashboard files (associated with Touch example projects).
- Support for Atmel Data Visualizer Auto configuration.

13.8. Revision B (May 2021)

- Power Monitor support: Power connection under DGI and related three power plots when activated; Power Analysis feature under Time Plot hover menu.
- Save a plot Snapshot to a CSV or JSON file.
- Plugin support available. Select *Tools>Plugins* from either the MPLAB X IDE or standalone visualizer application.
- Graph tab renamed Time Plot tab.
- XY Plot tabbed window added for machine learning support.
- Views menu icon added from which to select as the active window Time Plot, XY Plot, Terminal, or plugin window.

13.9. Revision A (June 2020)

First release of this document (PDF and WebHelp.)

Initial Release (December 2019)

Initial release of this document as WebHelp.

Microchip Information

Trademarks

The “Microchip” name and logo, the “M” logo, and other names, logos, and brands are registered and unregistered trademarks of Microchip Technology Incorporated or its affiliates and/or subsidiaries in the United States and/or other countries (“Microchip Trademarks”). Information regarding Microchip Trademarks can be found at https://www.microchip.com/en-us/about/legal-information/microchip-trademarks.

Legal Notice

This publication and the information herein may be used only with Microchip products, including to design, test, and integrate Microchip products with your application. Use of this information in any other manner violates these terms. Information regarding device applications is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. Contact your local Microchip sales office for additional support or, obtain additional support at www.microchip.com/en-us/support/design-help/client-support-services.

THIS INFORMATION IS PROVIDED BY MICROCHIP “AS IS”. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL, OR CONSEQUENTIAL LOSS, DAMAGE, COST, OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP’S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION.

Use of Microchip devices in life support and/or safety applications is entirely at the buyer’s risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip products:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner, within operating specifications, and under normal conditions.
- Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip products are strictly prohibited and may violate the Digital Millennium Copyright Act.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is “unbreakable”. Code protection is constantly evolving. Microchip is committed to continuously improving the code protection features of our products.