Application Note

MCU Signal Sight Tool Developer's Guide

Delaney Woodward, Peter Luong, Nima Eskandari

ABSTRACT

Traditional methods of debugging and evaluating embedded applications often involve expensive equipment, and only provide limited visibility into internal signals. Current software alternatives, while cheaper, do not provide the precision necessary to accurately evaluate real-time control applications. This fragmented and resource-intensive debug environment creates a barrier for customers. The MCU Signal Sight tool provides a software-driven, high-performance graphical user interface (GUI) that brings real-time plotting and advanced debugging tools to embedded engineers. MCU Signal Sight enables precise, real-time signal visualization in a familiar interface, with powerful features such as data exporting and real-time statistical analysis. This tool is driven by easily configurable, lightweight software that is ready-to-drop into any software project. MCU Signal Sight integrates seamlessly into any development environment, being available as a standalone tool, integrated within Code Composer Studio™ (CCS), or online through TI's cloud tools.

Table of Contents

List of Figures

Trademarks

Code Composer Studio™ and LaunchPads™ are trademarks of Texas Instruments.
All trademarks are the property of their respective owners.

1 Introduction

MCU Signal Sight is a graphical user interface (GUI) used to plot real-time signals from C2000 microcontroller (MCU) applications on a virtual oscilloscope. The SysConfig tool is employed to autogenerate a custom embedded software library for use inside an embedded application, and a custom GUI application with the related plotting utility. This application note explores the features of the MCU Signal Sight GUI, details through how to implement MCU Signal Sight support into an existing CCS project, and gives an overview of how to use the MCU Signal Sight example provided in C2000ware.

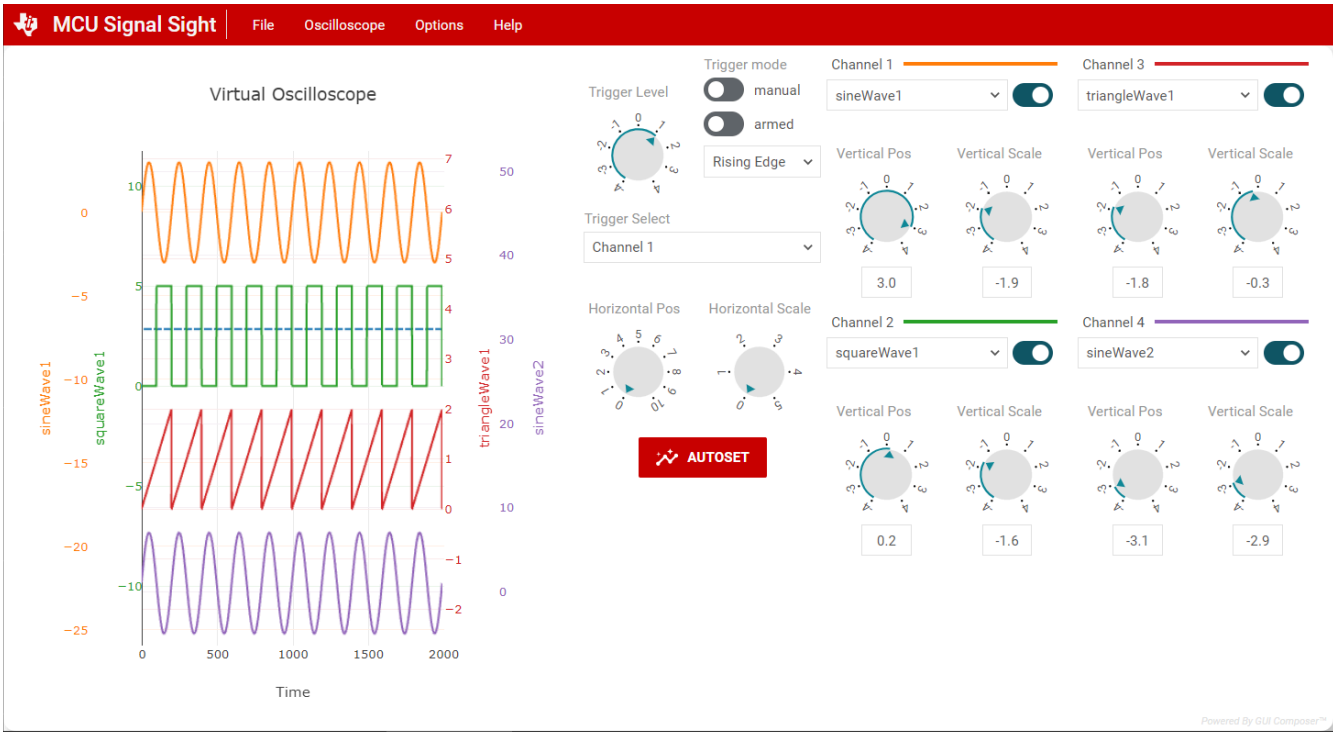

Figure 1-1. MCU Signal Sight GUI Tool Diagram

2 Features

The following features are supported by the MCU Signal Sight tool:

- High speed MCU-to-GUI communication with no debugger connection required
  - Utilizes a UART-to-USB bridge present on all C2000 LaunchPads™ and controlCARDs
- Expanded oscilloscope capabilities with visualization of signals and variables internal to the MCU application software
- Up to four simultaneous plotting channels
- Autogenerated embedded and GUI code based on SysConfig selections
- Minimal CPU bandwidth required for tool execution
  - Enables non-intrusive debugging of an application
- Enhanced analysis of plot signals
  - Manual and armed trigger modes
  - Customizable scope view, buffer size, and sample rate settings
  - Waveform analyzer
  - Autoset graphing view
- CSV file exporting feature
  - Allows for data analysis with external programs in a parsable format

2.1 Software Setup

MCU Signal Sight support can be added to any existing C2000 project that has SysConfig support and uses a C2000ware version of 5.05.000 or newer. The GUI and necessary library files can be fully generated and interfaced with through the integrated CCS environment. Optionally, a standalone version of the MCU Signal Sight GUI can also be downloaded and run independent of a CCS installation.

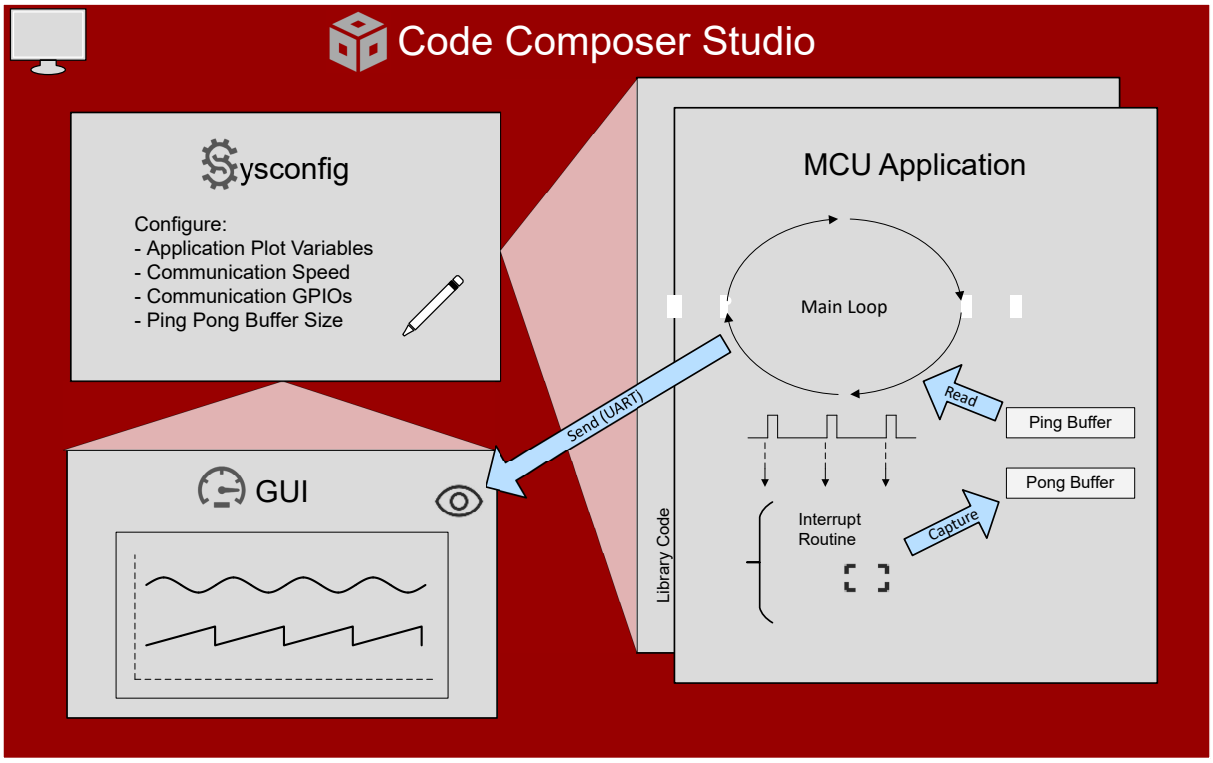

Figure 2-1. Signal Sight Software Diagram

After adding the MCU Signal Sight module in the user's SysConfig (.syscfg) file, the SysConfig tool autogenerates library files for use by the target application, and creates the integrated GUI Composer application with the virtual oscilloscope component.

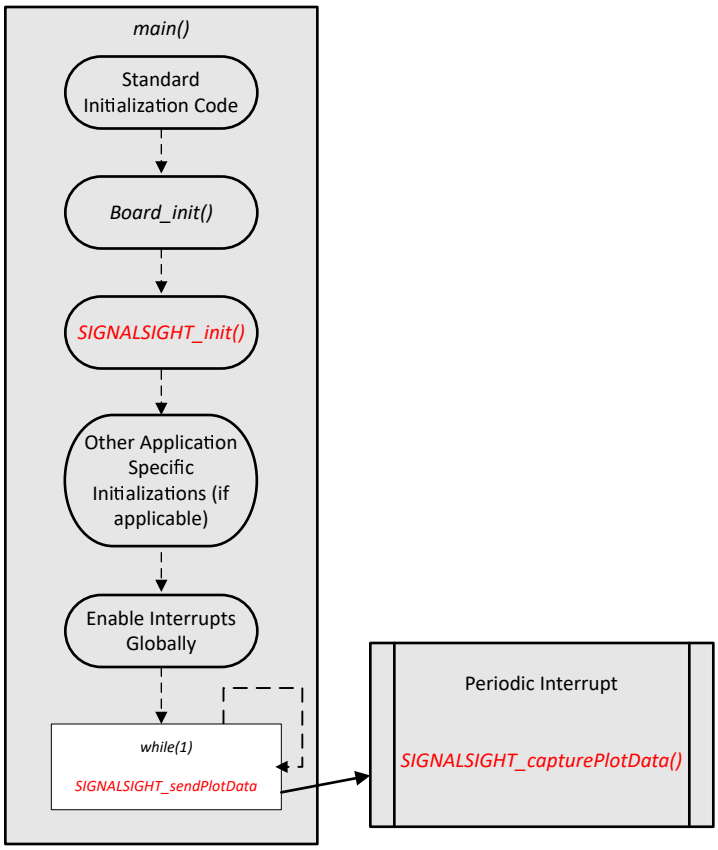

Figure 2-2. Signal Sight Software Flow Diagram

The library files generate three top-level functions inside the library files for use by the target application, which are described below. To access the functions in the user's application, make sure to include the signalsight.h header file with the line: *#include <signalsight/signalsight.h>*

- *SIGNALSIGHT_INIT()* - Signal Sight Initialization Function
  - Call this function once in the initialization code after device initializations, interrupt initializations, and SysConfig initializations (Board_init()), but before enabling interrupts globally.
- *SIGNALSIGHT_sendPlotData()* - Signal Sight Capture Plot Data Function
  - Call this function at the part of the application code where data needs to be sampled. The value of the plot variable is saved in a memory buffer for transmission in the subsequent call of SIGNALSIGHT_sendPlotData(). Generally this is called from a periodic timer or ISR interrupt.
- *SIGNALSIGHT_capturePlotData()* - Signal Sight Send Data Function
  - Call this function in the application where the device sends the buffered data to the GUI by the SCI peripheral. This function is recommended to be placed in a low priority location of the application code, for example, in the main loop of the code, since the CPU can be blocked waiting in this function until there is enough space in the SCI TX FIFO.

2.2 Hardware Setup

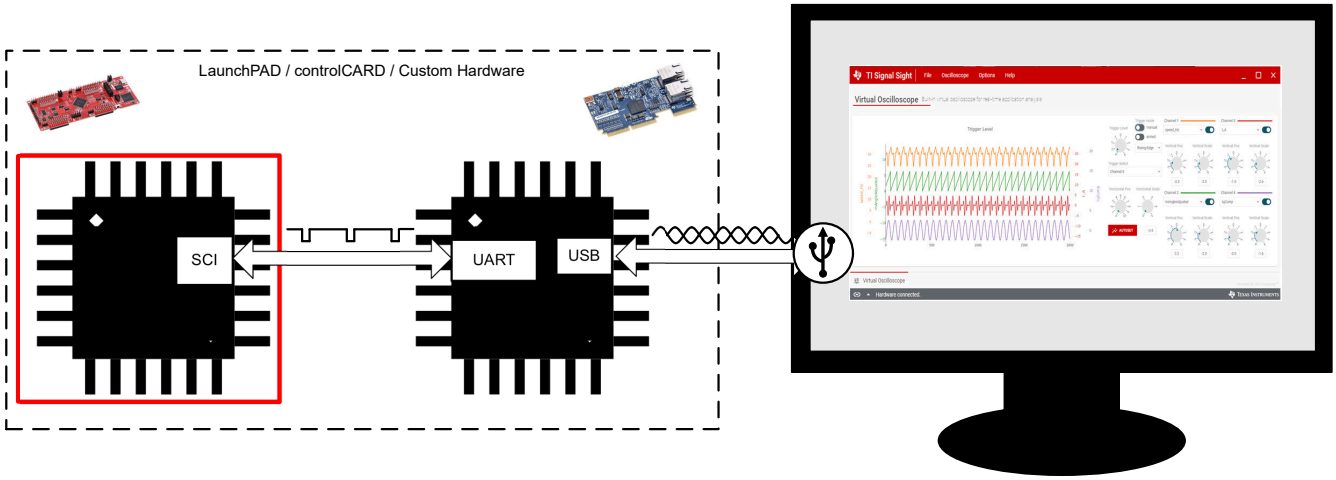

Figure 2-3. MCU Signal Sight Hardware Diagram

The only hardware needed to use the MCU Signal Sight tool is a C2000 device with a SCI peripheral, and some form of a UART-to-USB bridge to communicate with the GUI application on a host PC. The standard C2000 LaunchPads and controlCARDs already contain an on-board bridge device, which in most cases comes flashed with the XDS110 software that implements a UART-to-USB bridge. In the MCU Signal Sight hardware setup, the target MCU utilizes the on-board SCI (UART protocol) peripheral to communicate with the GUI on the PC.

3 Running the C2000ware Example

A basic example called *transfer_ex8_signalsight_basic_demo* using the MCU Signal Sight tool is provided in the C2000ware SDK. Run this example to verify a valid hardware and software setup, as well as learn the features of the tool. Perform the following steps:

1. Launch CCS
2. Click *File* → *Import Projects* → *Browse*
3. Navigate to the example in C2000WARE install path: C2000Ware_VERSION#/driverlib/DEVICE_GPN/examples/CORE_IF_MULTICORE/transfer/ folder.
4. Click *Finish*
5. Right-click on the project and click Properties
   a. Under *General* → *Variables* double-click on *GUI_SUPPORT* and change *Value* to *1*
6. Right-click on the project and click *Debug Project*
7. Press Play in the Debug window
8. Navigate to *View* → *Plugins* → *myMCUSignalSight0*
9. Once the GUI loads, click *Connect*
10. In the GUI window,
    a. Click *Oscilloscope* → *Enable All Channels*
    b. Set the Trigger Mode to *manual*
11. View the sine, triangle, and square waveforms in the Signal Sight interface window

See Section 7 for debugging any issues which the user can experience when following these steps.

4 Adding Signal Sight to a Project

Follow the steps in the below sections to plot the desired application's variables in the Signal Sight virtual oscilloscope.

4.1 SysConfig Steps

The MCU Signal Sight tool is built into the Sysonfig GUI, so having a SysConfig file in the project is required to use the tool. If users already have a SysConfig file (*.syscfg) in the project, then refer to C2000 SysConfig to learn how to add a SysConfig file to an existing project.

1. Double-click the SysConfig file inside CCS to launch SysConfig
2. Scroll down to the *MCU CONTROL CENTER AND TRANSFER* section of modules and click the (+) next to the *MCU Signal Sight* module

Figure 4-1. Add SysConfig Module

3. (Optionally) Customize the name of the GUI module instance in *Name*
4. Input a *Target Buffer Size*. This configures the size of the float array used in device memory to buffer captured data. Larger buffer size selections enable a faster rate of data capture.

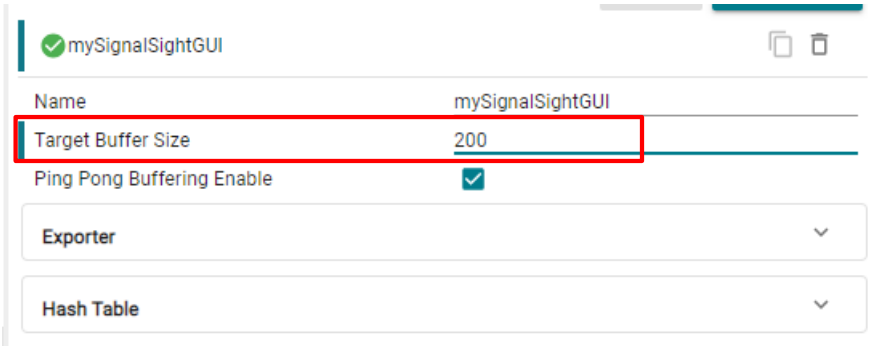

Figure 4-2. Input Buffer Size

Note

For devices with limited on-chip memory or applications with high memory utilization, large buffer sizes can result in a *program does not fit into available memory...* build error, meaning the array cannot fit in the available device memory. Reduce the *Target Buffer Size* value until the build error is no longer present.

5. Open the *Exporter* module drop-down menu inside the *MCU Signal Sight* SysConfig module
   a. (Optionally) Modify the name of the packaging layer in *Name*
6. Open the *SCI Transfer Communication Link* drop-down menu inside the *Exporter* module
   a. (Optionally) Modify the name of the communication layer in *Name*
   b. (Optionally) Modify the SCI(UART) baud rate used by the application in *Baud Rate*. Higher baud rate selections enable a fast rate of data capture.

Note

In some cases, high baud rates can result in a loss of data integrity. This version of the tool has been tested thoroughly at a maximum baud rate of 921600 bits/second. Use caution with all higher baud rate selections.

7. Open the PinMux submodule drop-down menu inside the *Exporter → SCI Transfer Communication Link* submodule

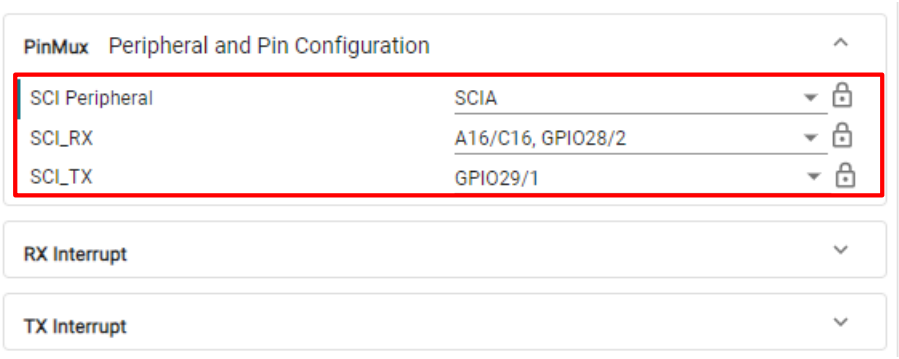

Figure 4-3. Configure SCI GPIOs

   a. Select the desired TX and RX GPIO pins to be connected to the UART-to-USB bridge. If using a LaunchPad or controlCARD, then add board support to the project (see C2000™ SysConfig: Board Support) and select the pins connected to the XDS.

8. Open the *Hash Table* module drop-down menu inside the *MCU Signal Sight* SysConfig module
   a. (Optionally) Modify the name of the hash table in *Name*

9. Add (+) a Hash Table Element for each variable in the user's main application that is enabled for plotting in the virtual oscilloscope. Modify the *Name* of each element to match the variable name in the application.

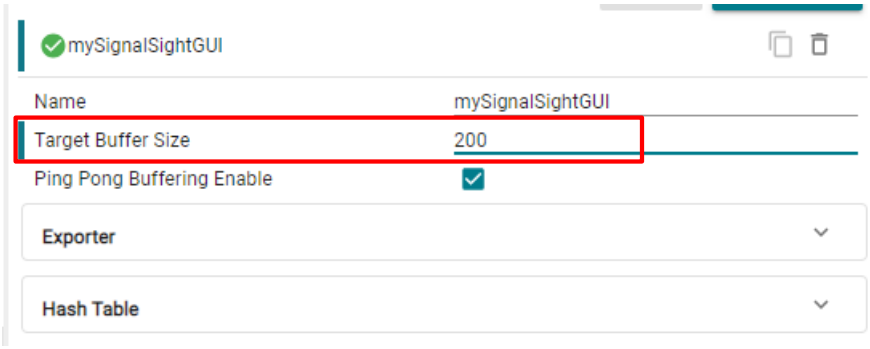

Figure 4-4. Add Hash Elements

Note

The variable name must match to the application variable name exactly and the variable must be a global 32-bit float.

4.2 Target Application Steps

Now that SysConfig selections have been made, SysConfig has auto-generated files containing library functions that can be called inside the user's application code to setup the tool, capture/buffer variable data, and transmit the data to the GUI to be plotted. Below are the steps for implementing these functions in an application.

1. Verify that the standard SysConfig "board.h" header file is included at the top of main C file and a call to *Board_init()* is present at the end of the initialization routine.

```
#include "board.h"
```

```
Board_init();
```

2. Additionally, include the *signalsight/signalsight.h* header file

```
#include <signalsight/signalsight.h>
```

```
//
// Included Files
//
#include <signalsight/signalsight.h>
#include "board.h"
```

Figure 4-5. Target Code Includes

3. After the call to *Board_init()*, add another function call to *SIGNALSIGHT_init()* in the application initialization code. Have this be the last code executed before interrupts are enabled globally.

```
SIGNALSIGHT_init();
```

```
//
// Main
//
void main(void)
{
    //
    // Initialize device clock and peripherals
    //
    Device_init();

    //
    // Disable pin locks and enable internal pull-ups.
    //
    Device_initGPIO();

    //
    // Initialize PIE and clear PIE registers. Disables CPU interrupts.
    //
    Interrupt_initModule();

    //
    // Initialize the PIE vector table with pointers to the shell Interrupt
    // Service Routines (ISR).
    //
    Interrupt_initVectorTable();

    //
    // PinMux and Peripheral Initialization
    //
    Board_init();

    //
    // Initialize Signal Sight tool state
    //
    SIGNALSIGHT_init();
```

Figure 4-6. Target Code Initializations

4. Choose where in the program to read the current values of the plot variables and write these values to the buffer. Usually, this is done periodically in a PWM or CPU timer ISR. Add a function call to *SIGNALSIGHT_capturePlotData()* in this area of the application.

```
SIGNALSIGHT_capturePlotData();
```

```
//
// cpuTimer1ISR - CpuTimer1  ISR every 1 ms
//
__interrupt void
cpuTimer1ISR(void)
{

    //
    // Sample current values of the streaming variables and write them to a
    // temporary buffer
    //
    SIGNALSIGHT_capturePlotData();
```

Figure 4-7. Capture And Buffer Data

5. Choose where in the program to transmit the data in the buffer. Usually this is done in a low priority or background loop since the program can be blocked waiting for the SCI module to transmit the data. Add a function call to *SIGNALSIGHT_sendPlotData()* in this area of the application.

```
SIGNALSIGHT_sendPlotData();
```

```
//
// Enable Global Interrupt (INTM) and real time interrupt (DBGM)
//
EINT;
ERTM;

while(1)
{
    //
    // Send all streaming data from the temporary buffer when full
    //
    SIGNALSIGHT_sendPlotData();
}
```

Figure 4-8. Transmit Buffered Data

4.3 CCS Steps

1. The SysConfig tool autogenerates a bat file (called gui_setup.bat) that copies the autogenerated GUI composer files to the correct location in the CCS installation so that the MCU Signal Sight GUI can be launched from the CCS menu as a plugin. To run this bat file automatically every time to build the project, follow the steps below:
   a. Right-click on the project name and select *Properties*.
   b. Navigate to *General* → *Variables*.
   c. Click (+) and add a variable called *GUI_SUPPORT* and set the value to 1. The *Type* can be left as String.

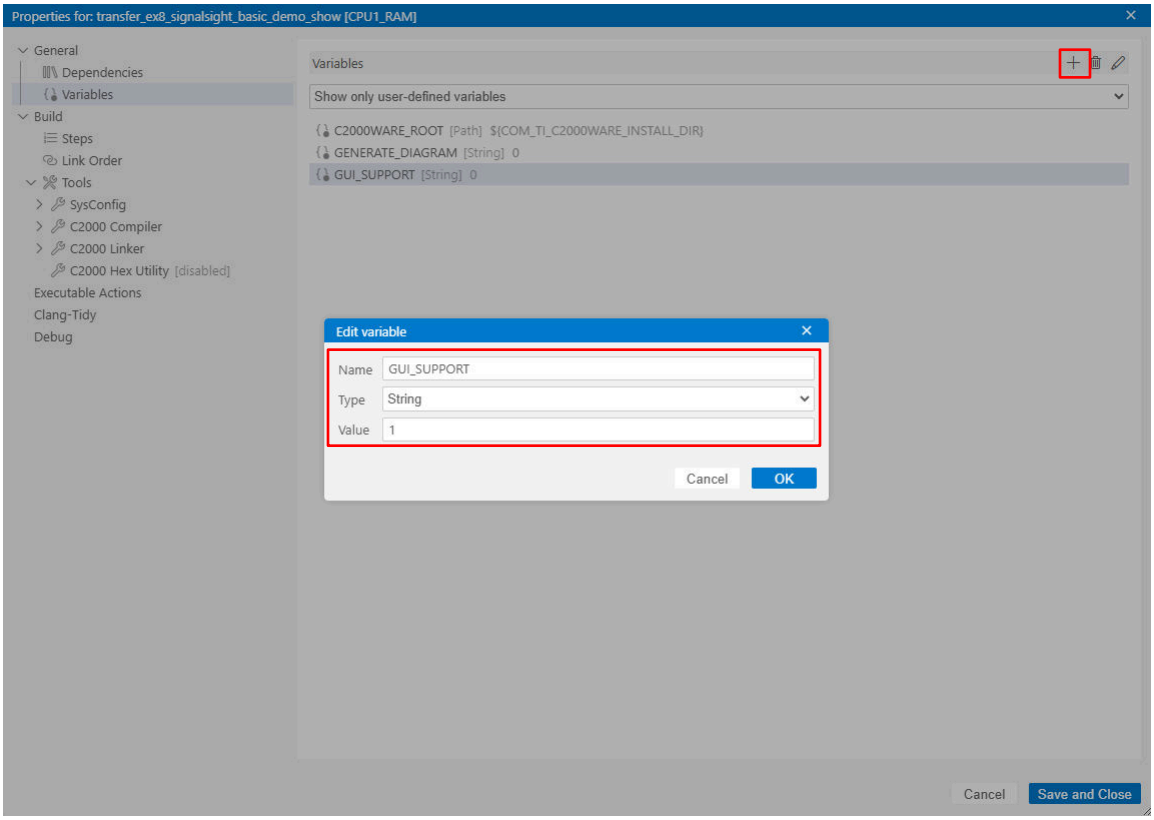

Figure 4-9. Add GUI_SUPPORT User_Defined Variable

   d. Navigate in the Properties menu to *Build* → *Steps*.
   e. Add an entry with the following line:

```
if ${GUI_SUPPORT} == 1 ${BuildDirectory}\syscfg\gui_setup.bat
```

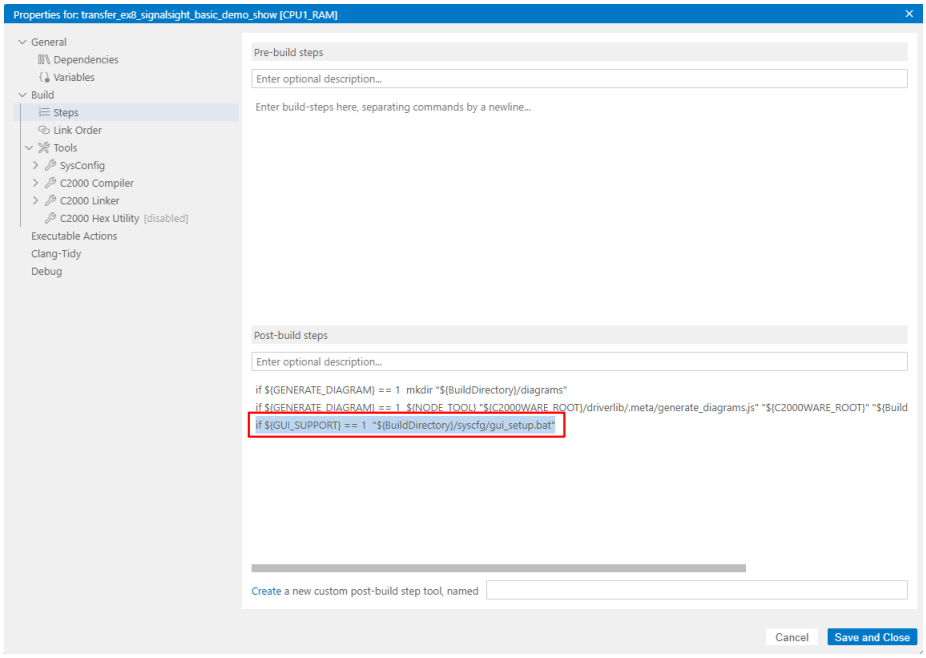

Figure 4-10. Add Post-Build Step

2. At this point, users are ready to build the project with the MCU Signal Sight support added. Right-click on the project name in CCS and select *Build Projects*.

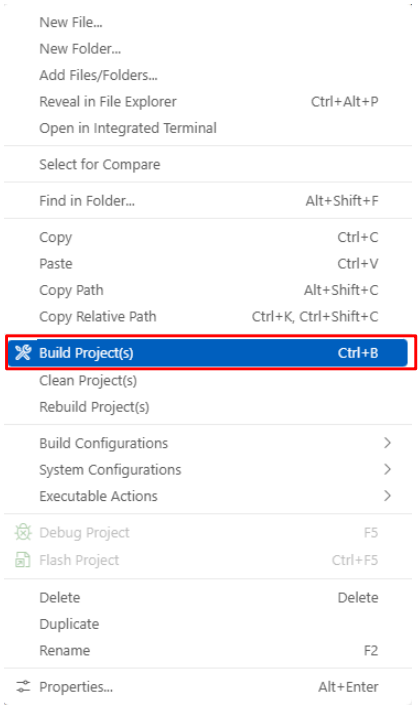

Figure 4-11. Build the CCS Project

3. Once the project build is complete, do a refresh of CCS by navigating to *Help → Reload Window*. This updates the plugins populated in the *View → Plugins* folder of CCS.

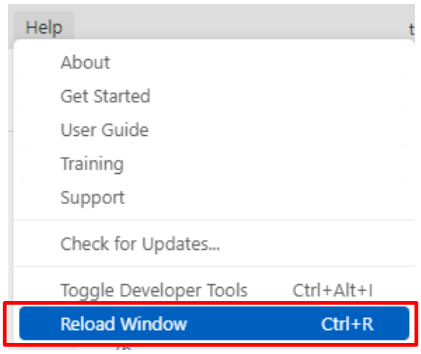

Figure 4-12. Reload the CCS Window

4. Load the application code onto the device by doing either right-click → *Debug Project* (starting a CCS debug session) or *Run* → *Flash Project* (loading the program without starting a debug session).
5. Navigate to *View* → *Plugins* → {GUI Name set in 3} to launch the MCU Signal Sight GUI.
6. Click the *SELECT COM PORT* button and follow the steps below.
   a. Select the COM port from the *Port* drop-down menu that matches the COM port used by the user's board in the PC Device Manager (Enter *Device Manager* into the PC Search bar). For most LaunchPads and controlCARDs, the COM port number is under the name *Ports (COM & LPT)* → *XDS110 Class Application/User UART (COM#)*.

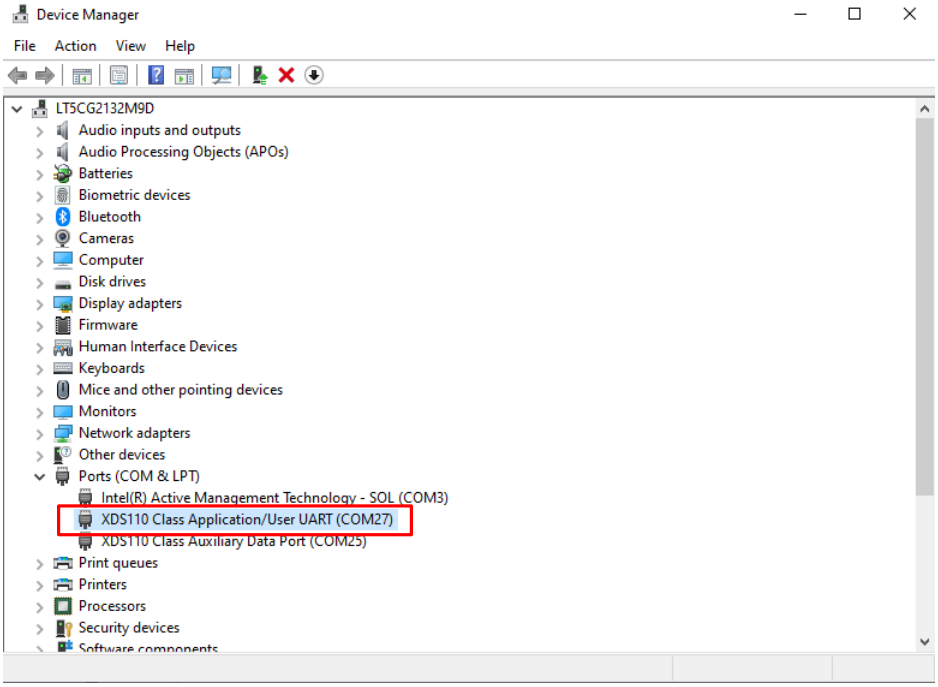

Figure 4-13. COM Port Number From Device Manager

   b. Select or input the *Baud Rate* matching the input chosen in 2.
   c. Click *OK*.
7. Click *CONNECT* in the GUI and wait a few seconds for the initial communication handshake to take place.
8. Toggle on a channel and see the data begin plotting in the window.

Note

If any data being plotted in the window, then check the trigger settings and verify that the values of the variables in the code are actively changing.

See Section 7 if there are any issues with these steps.

5 Navigating the Signal Sight GUI

Signal Sight provides a user-friendly GUI which closely mimics the familiar interface of a standard oscilloscope while offering various enhancements and features. The sections below describe the steps for leveraging the most commonly used features of the tool.

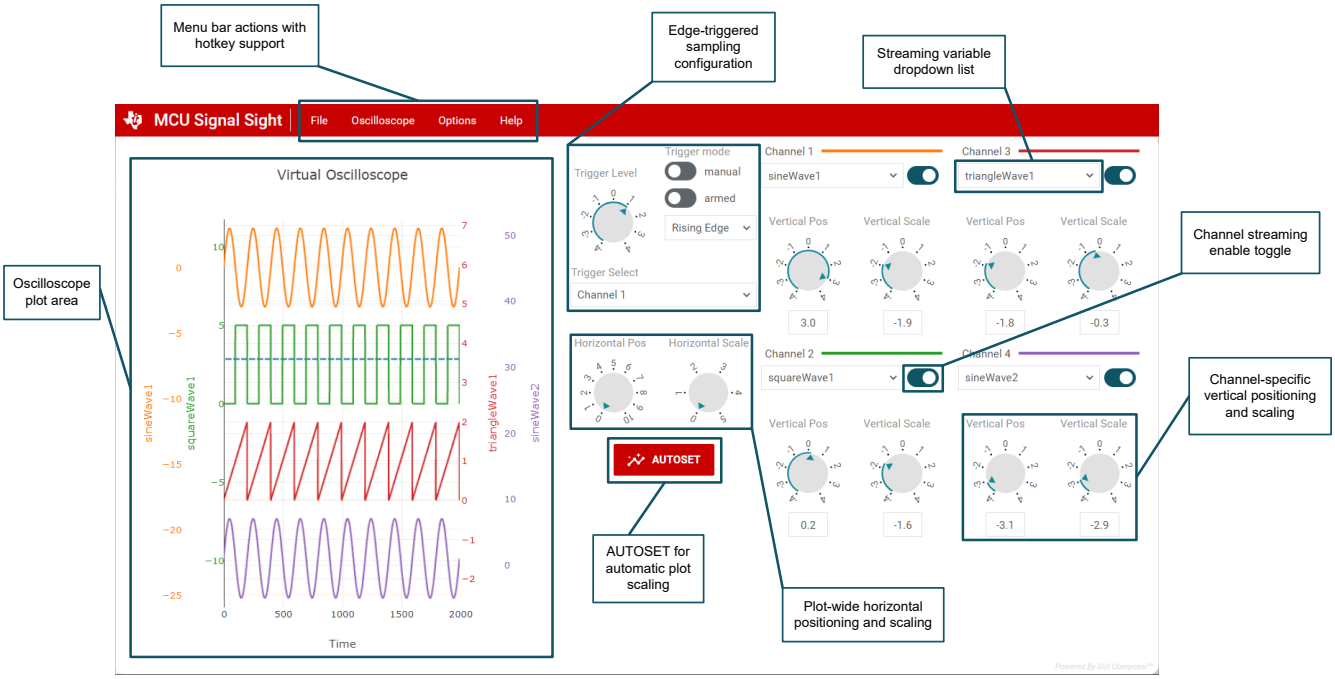

Figure 5-1. Signal Sight GUI Interface

5.1 Verifying Target Connection

Upon launching the Signal Sight GUI, a modal appears providing instructions on connecting to the target MCU device. Depending on the debug environment and if the tool has been used before, various different messages can be shown.

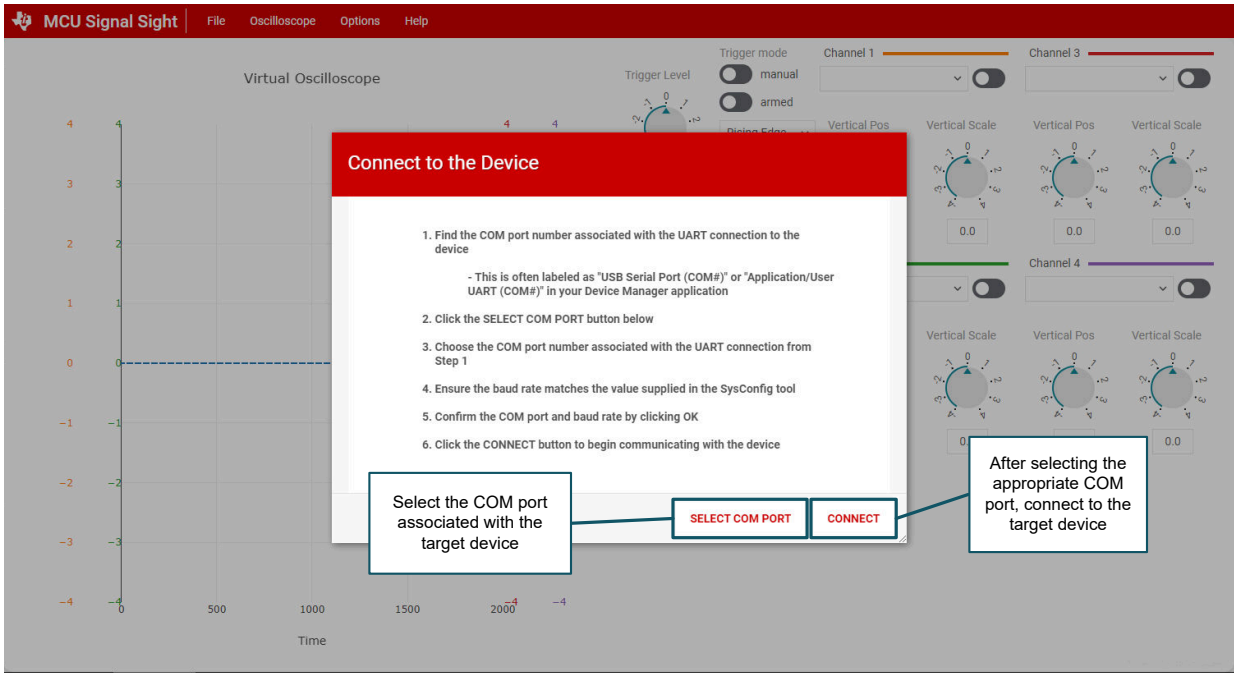

Figure 5-2. CCS Environment

When debugging within the CCS environment, a modal describing the steps to connect to the target are shown. Follow the steps and click *Connect* to establish the connection with the target device.

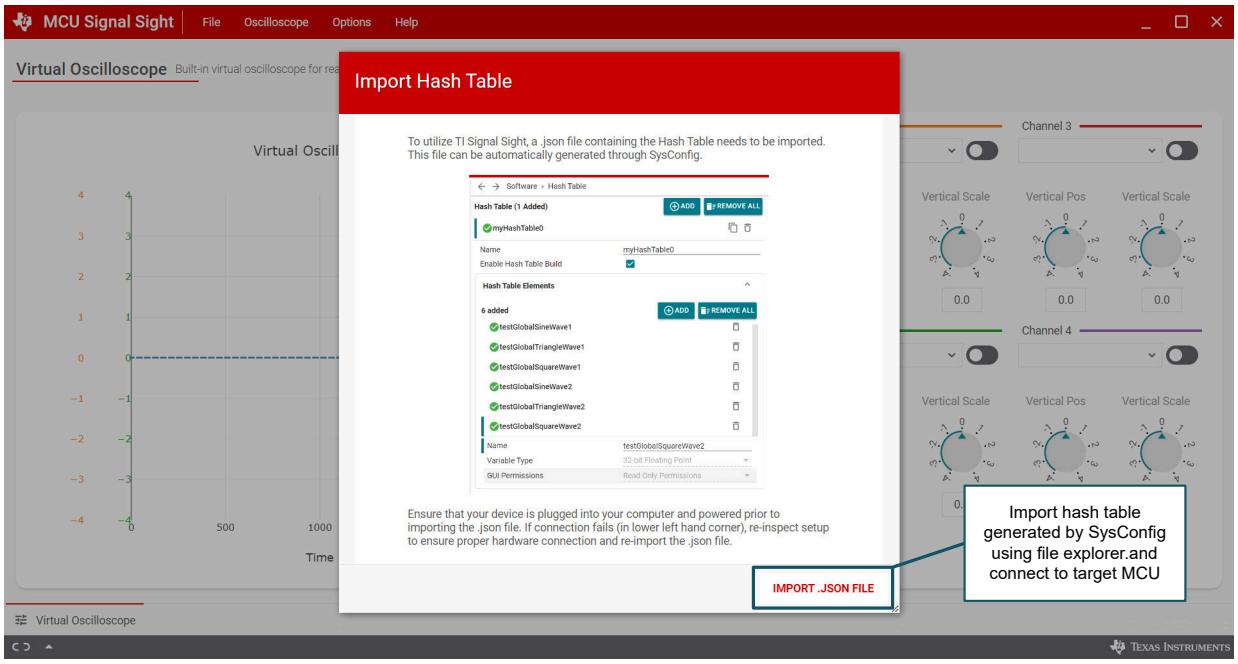

Figure 5-3. Standalone GUI Environment

When using the Signal Sight GUI in a standalone environment, the hash file generated by the SysConfig tool needs to be provided externally. To do this, select the *IMPORT .JSON FILE* button and navigate through the file explorer. Import the desired .json file. Once imported, the GUI issues a connection request to the connected MCU device. Once connected, the modal dismisses.

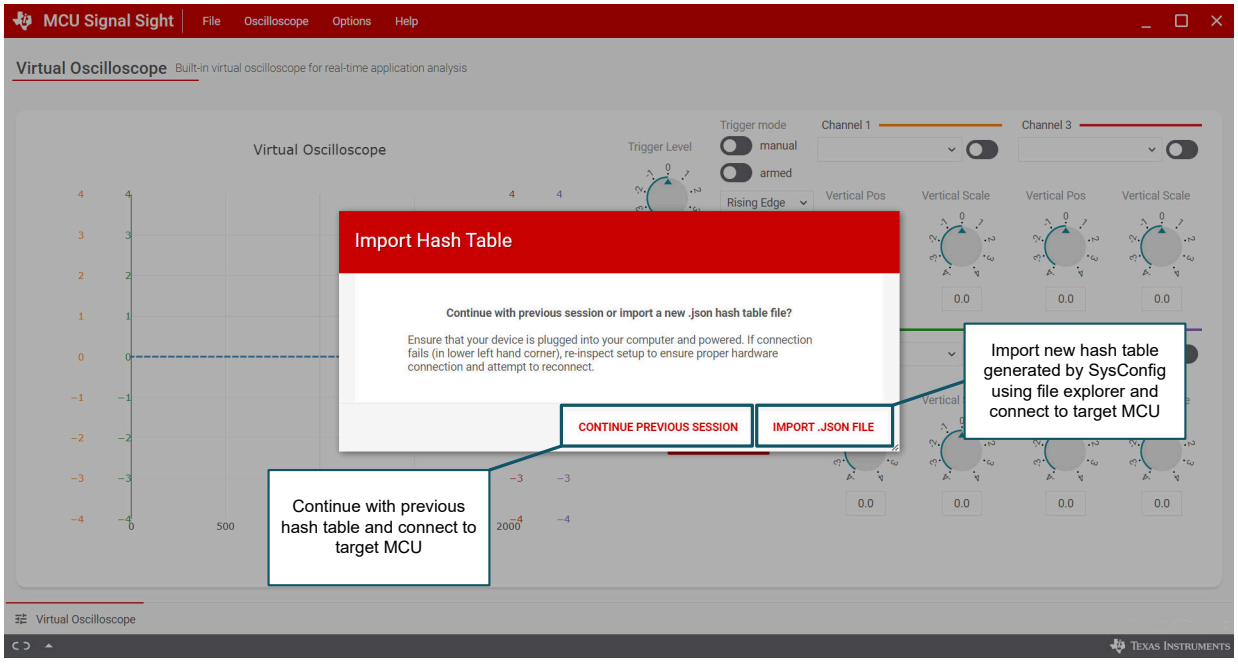

Figure 5-4. Restore Previous Session

On subsequent uses of the tool, the modal prompts the user to continue with a previous Signal Sight session. If continuing a previous session, then click the *CONTINUE PREVIOUS SESSION* button loads the previously

used hash table and attempt to connect to the device. If wanting to debug with a new project, then click the *IMPORT .JSON FILE* prompts the user to import a hash table file and connect to the target MCU.

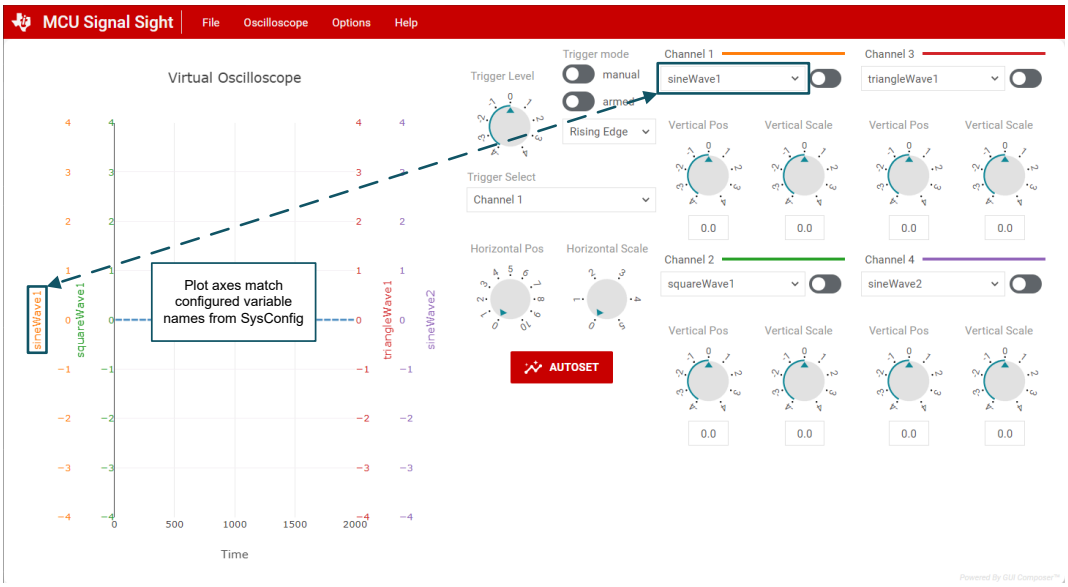

Figure 5-5. Signal Sight Interface

When the Signal Sight tool successfully connects to the target MCU, the oscilloscope interface is exposed. The first four streaming variables from the hash table are automatically populated onto the drop-down menus of Channels 1-4. The streaming variables are also displayed on the plot axes on the graph.

5.2 Enabling Data Streaming

Data streaming requests are initiated directly from the PC to the target MCU. To start the data streaming, click the switch corresponding to the input channel that is wanting to be viewed. To switch the specific variable that is being streamed, click the drop-down menu and select the desired streaming variable from the list.

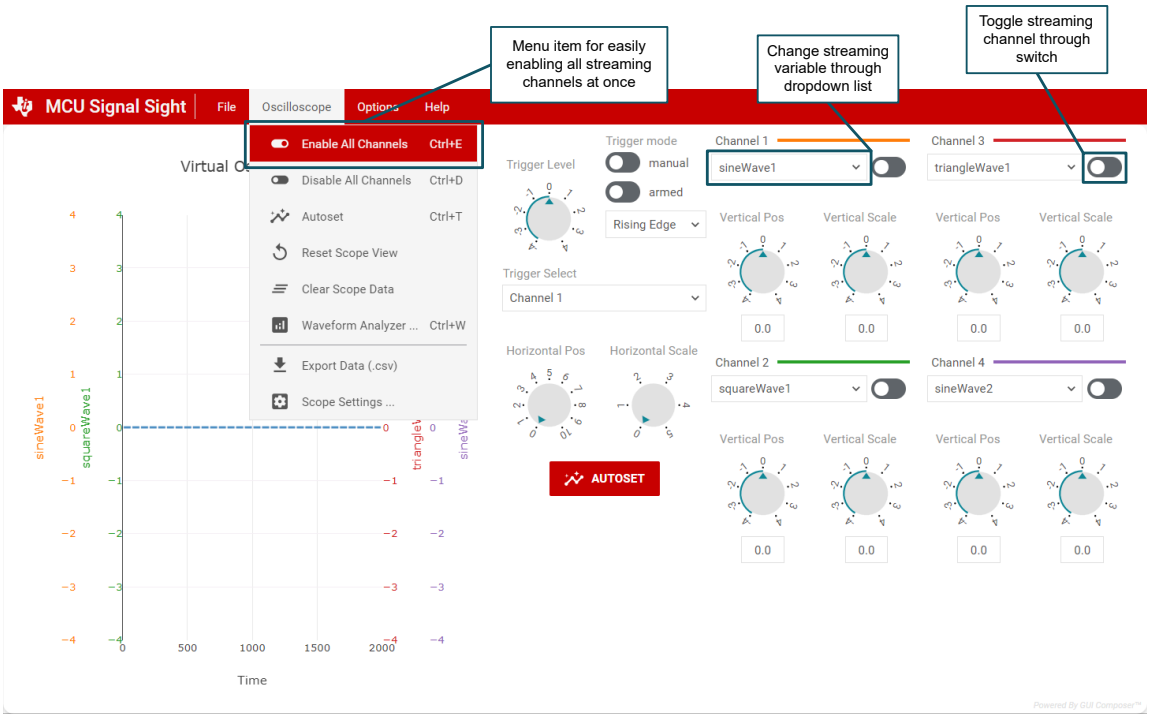

Figure 5-6. Enabling Data Streaming

Signal Sight features menu actions for starting and stopping the data stream. To quickly enable all four channels on the plot, click *Oscilloscope* → *Enable All Channels*. This automatically set the trigger mode to auto to enable continuous streaming. To disable data streaming, click *Oscilloscope* → *Disable All Channels*.

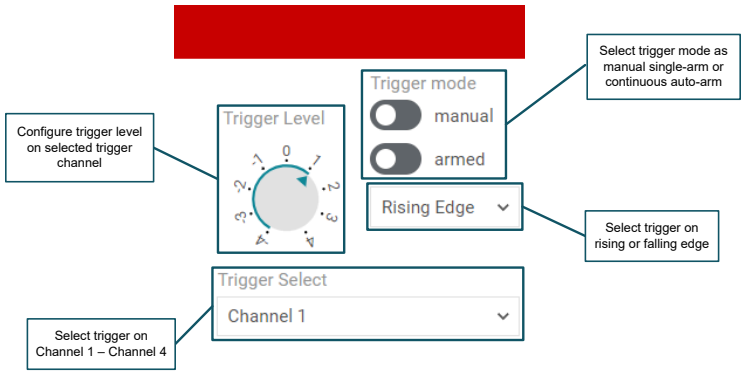

Figure 5-7. Trigger Settings

Like a physical scope, the virtual oscilloscope plot is level-triggered. That means that data is not displayed on the plot until the incoming data points intersect with the scope's trigger level. In the trigger settings, the user can configure the trigger settings, such as the trigger level, the channel being triggered, and the trigger edge (either rising or falling).

There are also various trigger modes available. The top switch controls the trigger repetition and the bottom switch controls the arming of the trigger. If the top switch is in the off position (to the left), then the scope trigger is in manual single-arm mode. This mode is intended for capturing single plots of data. Once the bottom switch is turned on (to the right), the scope trigger looks for the edge on the incoming data stream. When the edge is seen, the scope display starts plotting the incoming data. Once the plot is completely filled with data, the bottom switch automatically switches off and the trigger is disabled.

If the top switch is in the on position (to the right), then the scope trigger is in continuous arm-mode. This mode is intended for running the scope continuously and viewing the data over extended periods of time. The scope trigger continuously samples the incoming data stream and plot onto the graph without needing to manually rearm the trigger. To disable this mode, simply toggle off the top switch.

5.3 Adjusting Plot View

When data is streamed from the MCU device to the PC, the data points are populated on the graph.

For convenience, an *Autoset* button is provided which automatically determines the best settings for the vertical scaling of all scope channels. This Autoset button can be accessed from the main interface or through the Oscilloscope menu in the menubar.

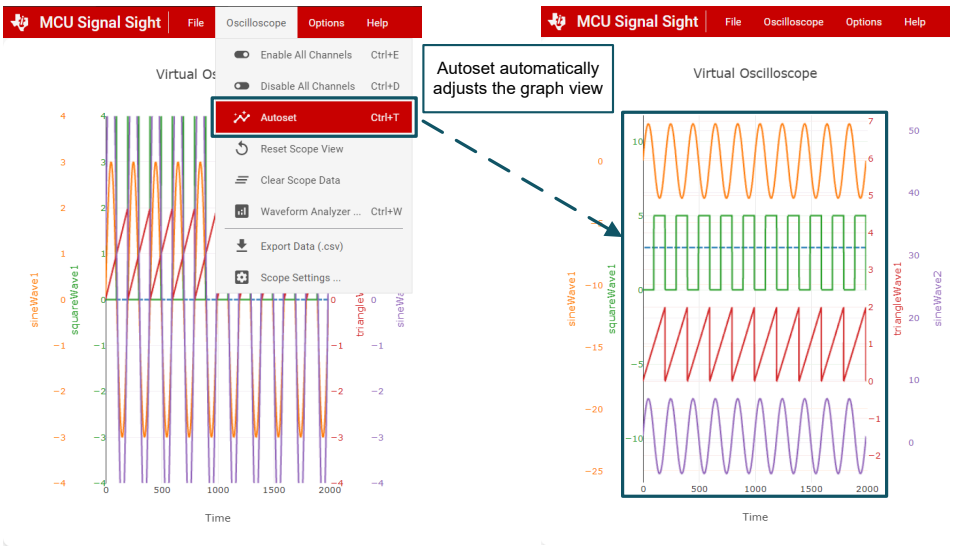

Figure 5-8. Autoset Plot View

The Signal Sight interface features knobs for manually adjusting the view of the plot. The Horizontal Pos knob controls the positioning of the x-axis. At 0, the view of the plot shows the incoming data traced out as received by the tool. At 10, the view of the plot shows the previously buffered data that was cleared from the plot. The Horizontal Scale knob adjusts the scaling of the x-axis and can be used to zoom into parts of the graph. The scaling is logarithmic. At 0, the full buffer is in the frame of the plot. At 1, only half of the buffer is in view of the plot. At 2, only a quarter of the buffer is displayed.

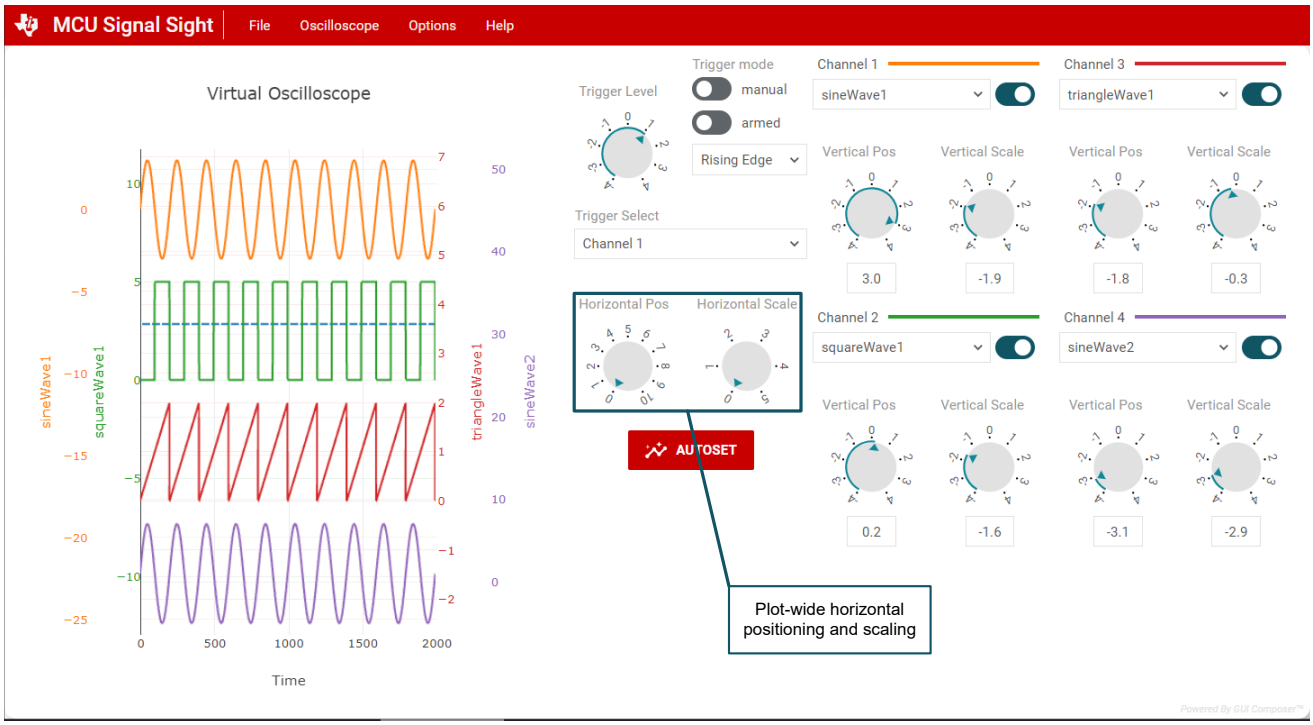

Figure 5-9. Horizontal Knobs

Each input channel has an individual vertical positioning and vertical scaling knobs. These controls allow the user to adjust the view of specific plots on the graph. The knobs have similar functionality as the horizontal knobs described previously.

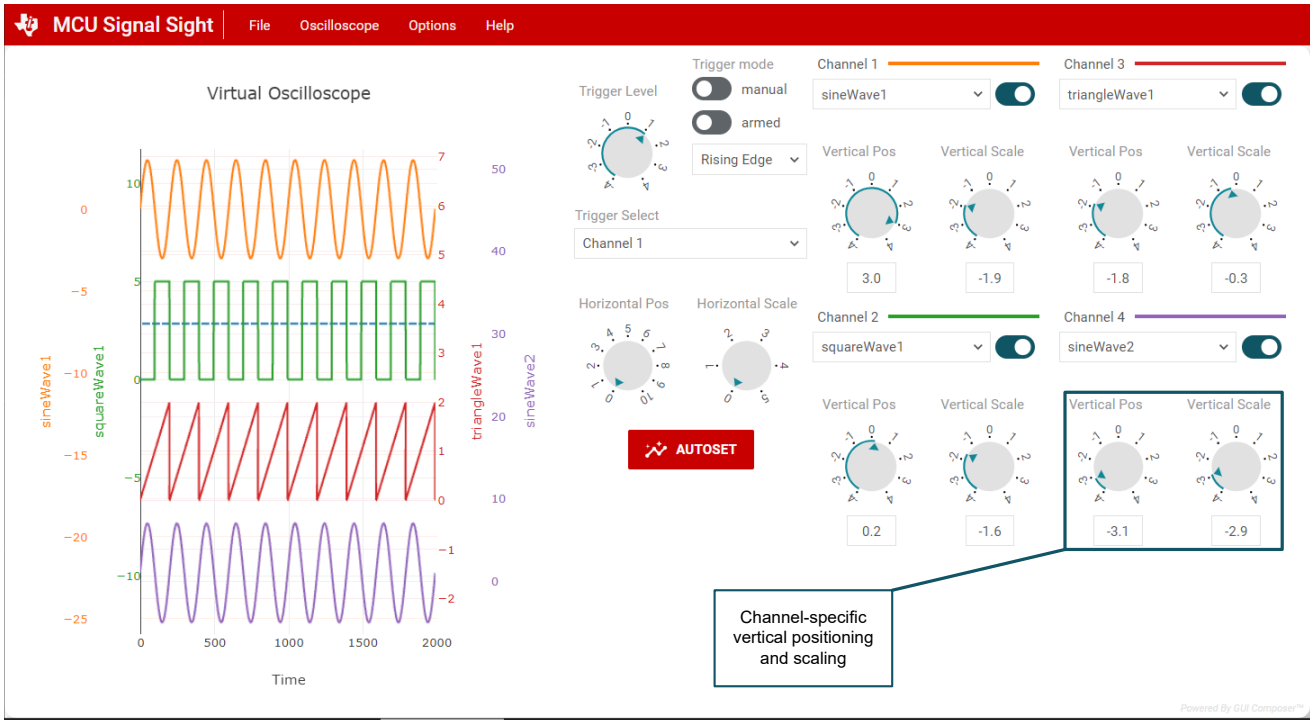

Figure 5-10. Vertical Knobs

To easily reset the scope view to the default view configuration, click *Oscilloscope → Reset Scope View*.

5.4 Menu Bar Actions and Hotkeys

The Signal Sight GUI features a number of features and quick actions which can be accessed through the menubar located on the top of the screen. The table below describes the available menu items and functions.

Menu	Menu Item	Menu Action	Hotkey
File	Import Hash Table JSON	Prompts the user to import a different hash table to use the GUI with a different project	
	Restart	Restarts the GUI	Ctrl + R
	Exit	Closes the GUI	Ctrl + X
Oscilloscope	Enable All Channels	Toggles on all the scope channels and begins streaming data	Ctrl + E
	Disable All Channels	Toggles off all scope channels	Ctrl + D
	Autoset	Automatically determines the best settings for vertical scaling of all scope channels	Ctrl + T
	Reset Scope View	Returns the scope plot scaling to the default values	
	Clear Scope Data	Clears all data in the scope plot buffer	
	Waveform Analyzer	Opens the Waveform Analyzer menu	Ctrl + W
	Export Data	Exports all of the data shown in the scope plot to a .csv file for external analysis	
	Scope Settings	Opens the scope settings menu	
Options	Serial Port Settings	Opens the serial port settings menu	
	Settings	Opens the Signal Sight settings menu	
Help	About	Displays information about the application	

5.5 Advanced Features

The Signal Sight GUI features various additional features on top of the basic data streaming capabilities to enable streamlined debugging.

5.5.1 Waveform Analyzer

The Signal Sight GUI features a dedicated signal analysis tool called Waveform Analyzer. This tool provides quick statistical analysis on the incoming data displayed on the scope plot. Waveform Analyzer displays key information such as the minimum datapoint, maximum datapoint, and root mean square (RMS). To access the Waveform Analyzer tool, click *Oscilloscope* → *Waveform Analyzer*.

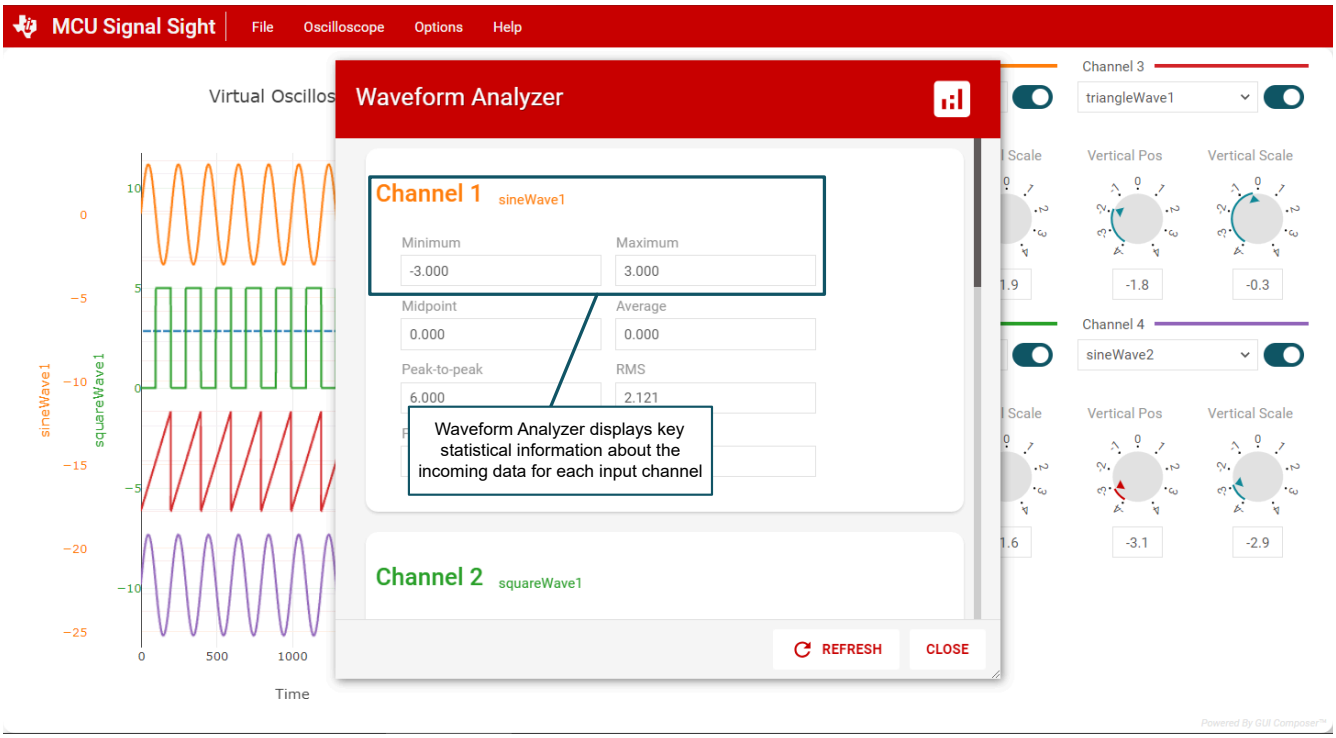

Figure 5-11. Waveform Analyzer Tool

5.5.2 Scope Settings

The virtual oscilloscope plot can be further customized from the standard view by adjusting the parameters available in the scope settings menu. To access the scope settings menu, click *Oscilloscope* → *Scope Settings*. This menu features settings for changing the buffer size and sampling rate. This also allows the user to edit the vertical scaling past the settings allowed by the channel knobs. This can be useful if the incoming data waveform is too large to be shown on the graph with the default settings.

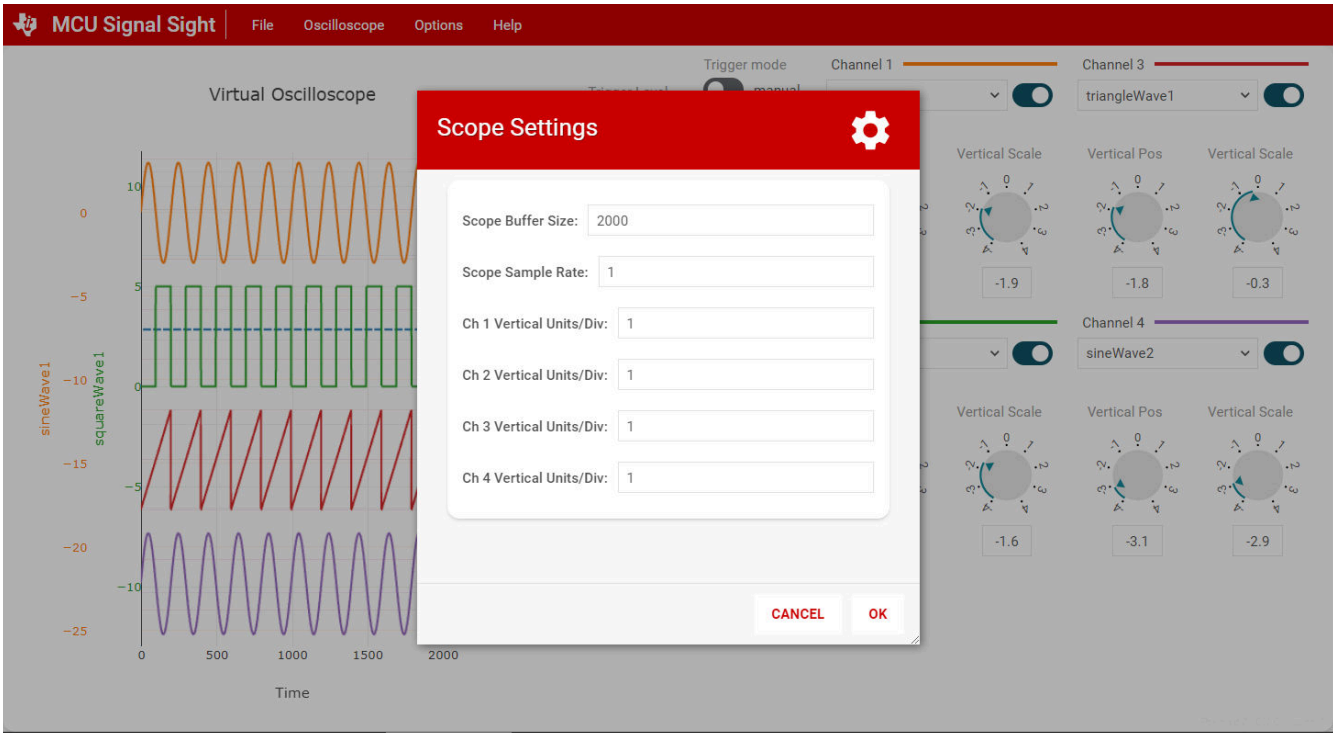

Figure 5-12. Scope Settings Menu

To capture more datapoints on the plot, edit the Scope Buffer Size parameter in the scope settings menu. This allows the user to increase the number of datapoints before the scope view is cleared.

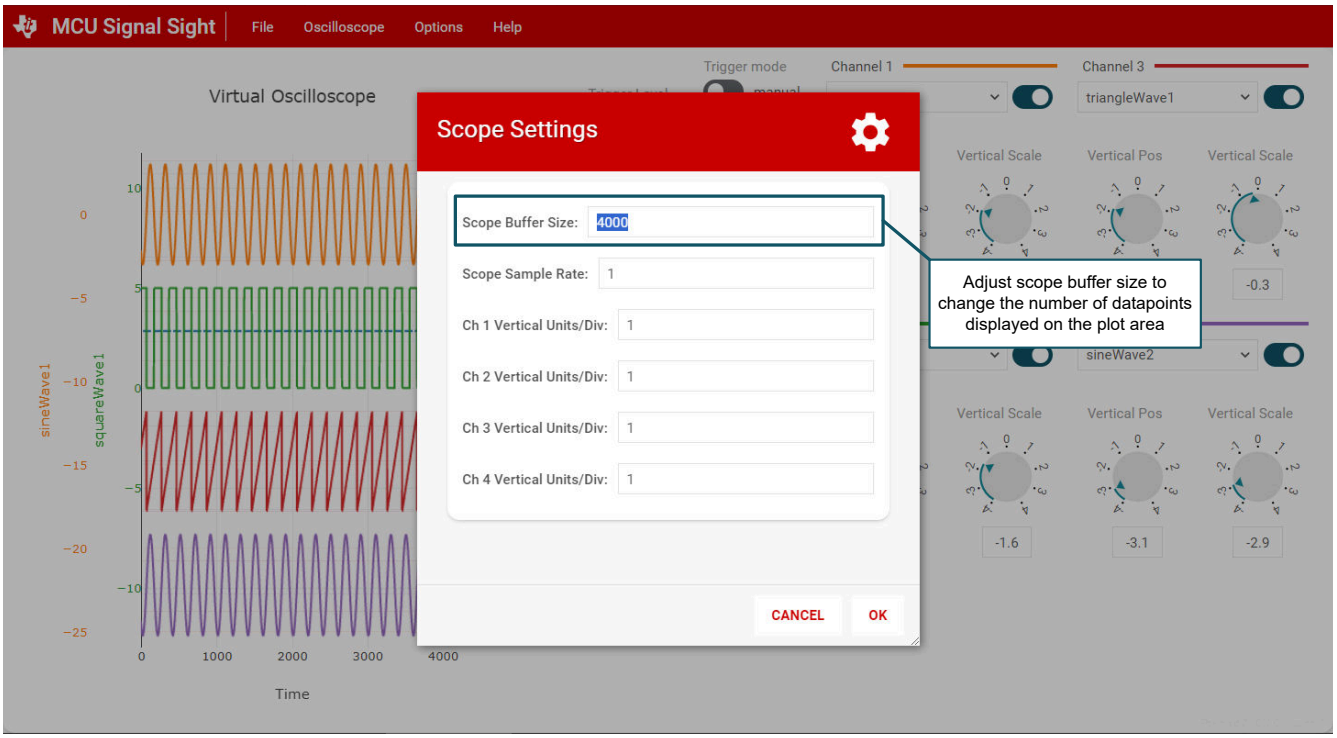

Figure 5-13. Changing Scope Buffer Size

5.5.3 Data Export

To easily access the datapoints being received from target MCU, the user has the option to download the data shown on the scope plot. To export the data to a .csv file, click *Oscilloscope → Export Data (.csv)*.

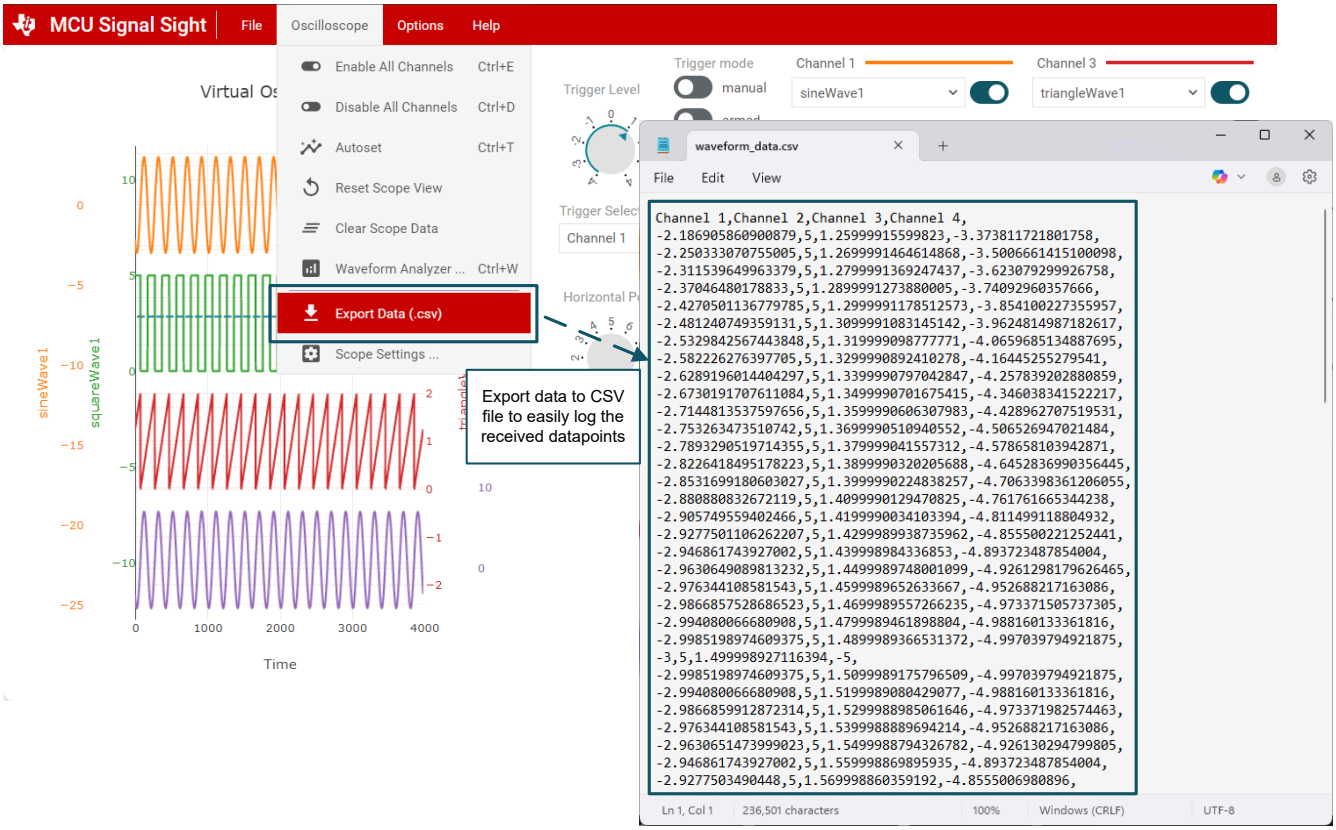

Figure 5-14. Exporting Signal Sight Data to .csv File

6 About the Tool

Plot Refresh Rate

Currently, the only communication peripheral supported by the tool is SCI (UART), so the plotting speed is dependent on the maximum SCI baud rate on the given C2000 device. See the relevant device data sheet for the maximum SCI baud rate.

Plot Variables

The current revision of this tool only supports plotting global variables with a 32-bit float type. Support for variables of signed or unsigned types and variables inside of structures is planned for future releases.

7 Troubleshooting Guide

See the FAQ below for suggestions to different troubleshooting scenarios.

- The GUI name does not appear in the *View → Plugins* folder
  - Check 1: Verify that the SysConfig and C2000Ware products are set to the expected versions in the project *Properties → General → Dependencies* .
  - Check 2: Navigate to the projects *Properties → General → Variables → {drop-down menu}Show both user-defined and system variables → SYSCONFIG_TOOL* variable. Verify that the path to the "sysconfig_cli.bat" file is located inside the current CCS installation. The path needs to look something like C:\ti\ccsxxxx\ccs\utils\sysconfig_x.xx.x\sysconfig_cli.bat.
- Data does not appear on the plot after I toggle on a channel
  - Check 1: Verify that the trigger channel and level is set to a signal that passes through (based on the nature of the data being plotted) the trigger and set the trigger mode to *Manual*.
  - Check 2: Verify that the SS_currentStreamState variable is being updated when a channel is toggled on.
    - Connect to the target code via CCS.
    - In the watch window, add the variable SS_currentStreamState.
    - If the value of this variable remains as *NO_UPDATE*, then there is an issue with the UART communication.
  - Check 3: For LaunchPads/controlCARDs - verify that the XDS, UART switch is in the correct position and the correct GPIOs are configured in the SysConfig SCI PinMux selection from SysConfig Step 7a.
  - Check 4: Verify that the Baud rate setting in the GUI *Options → Serial Port Settings* matches the baud rate selection in Sysconfig entry *MCU Signal Sight → Exporter → SCI Transfer Communication Link → Baud Rate* from SysConfig Step 6b.
- The plot appears to be missing/skipping over data from one or more channels
  - The GUI has a built in feature to skip capturing data samples whenever the buffer is full.
    - Connect to the target code via CCS (make sure to run the target code before clicking *Connect* in the GUI.
    - In the watch window, add the variable `SS_streamDataSkips` and `SS_streamDataSkips2` if ping pong buffering is enabled.
    - If the values of the skip variables are incrementing significantly, then the software is skipping capturing data samples. Users can either:
      - Increase the buffer size from SysConfig Step 4. This allows more data to be placed in the buffer before becoming full.
      - Decrease the data capturing frequency in the application code (for example: decreasing the frequency of the CPU timer ISR from Step Target Step 4). This gives the SCI module more time to empty the buffer before more data is added.
      - Increase the baud rate from SysConfig Step 6b. This allows the SCI to transmit data from the buffer faster, and therefore empty the buffer faster.
- GUI plot appears too small or channel knobs/Autoset button are missing from the screen
  - The virtual oscilloscope feature is designed to be viewed on a screen with a 100% scale. Users can modify this in the PC settings. For Windows, the option is in *Settings → Display → Scale and Layout*.
- Data plotting is suspended unexpectedly
  - Reconnect to the target by clicking *Options → Serial Port Settings* and selecting *Connect*. This resets the streaming state on the target side.
- For any bugs found in the tool, post on the E2E forum.

8 Summary

Seamlessly integrate the MCU Signal Sight GUI into an existing embedded C2000 project to gain visibility into the signals in applications and streamline the debugging process. With separate data buffering and transmission, MCU Signal Sight support can provide real-time captures of data without hindering CPU performance. Unlike traditional debugging methods that require costly hardware and offer limited signal visibility, or existing software implementations that lack precision for real-time applications, MCU Signal Sight provides a comprehensive, software-based implementation.

9 References

- Texas Instruments, MCU Control Center Tool Developer's Guide, application note
- Texas Instruments, C2000 Real-Time Microcontrollers Peripherals, user's guide
- Texas Instruments, C2000 SysConfig, application note
- Texas Instruments, DLT Developer's Guide With Tooling, application note

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265