Design and Implementation of User Interface for Labview Application

Department of Electrical and Computer Engineering

Chair of Control Systems

RPTU

Table of Contents

1. Introduction

This document outlines the significant changes made to the naming conventions within our project. The primary objective of these changes is to enhance readability, maintainability, and consistency across our codebase. The changes span across various components of our project, including VI, Functions, TCP Client, and Variables. Each section in this document corresponds to these components and provides a detailed overview of the old names, the new names, and a brief description of their functionalities. This will not only simplify the process of understanding the code but also make it easier to debug, test, and enhance the project in the future. Please refer to the table of contents for easy navigation through the document.

1.1. Overview

The Graphical User Interface (GUI) application is the main interface for the user to interact with the control system. It is responsible for displaying the data, sending commands to the microcontroller, and handling errors. The GUI application is built using LabVIEW, a graphical programming language that allows for easy and intuitive development of user interfaces.

1.2. Purpose

The purpose of this document is to provide a comprehensive overview of the use of the changes made to the naming conventions within our project. This document is intended for the use of our team members and any future contributors to the project. It is designed to provide a clear understanding of the old and new names, as well as a brief description of their functionalities.

1.3. Scope

The scope of this document covers the changes made to the naming conventions within the GUI application. The changes are categorized into different sections based on the components of the GUI application, including VI, Functions, TCP Client, and Variables. Each section provides a detailed overview of the old and new names, along with a brief description of their functionalities.

2. Comprehensive Overview of the Control System

The control system is based on the interaction of the controller with various actuators within the

system. The controller has the flexibility to implement different control strategies to deal with the needs of the applications. On the other hand, the controller has the ability to interact with the user through a graphical user interface (GUI) that allows the user to interact with the control system. The GUI is responsible for displaying information to the user, sending commands to the controller, and handling errors.

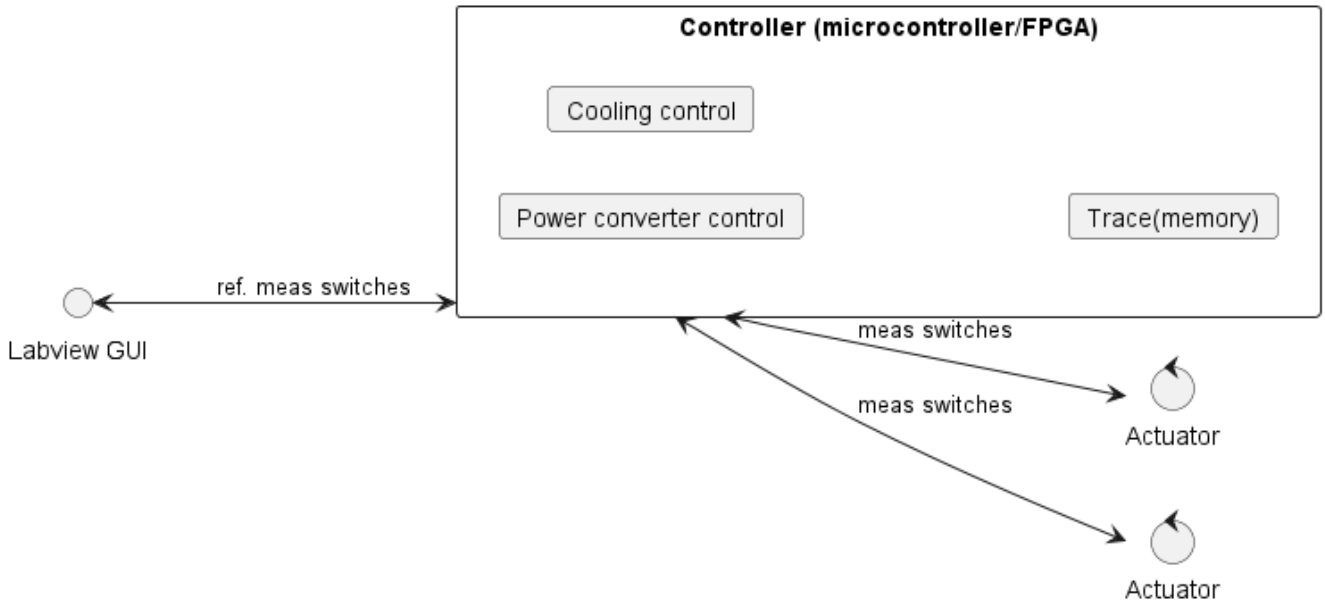


Figure 1. The control system schematic

2.1. Basic Elements

Inside the controller, there are two basic elements with which the GUI interacts:

2.1.1. Control Systems

Different control systems interact with the actuators present in the global system and can be contained within a single controller as shown in the figure

Within the control system, it is divided into two sections, the controller and the hardware.

Controller

In the controller, the following basic elements exist:

- ID
- Enable/disable
- Status
- Controller interface
- Hardware interface

These are defined within the controller's algorithm. Much of the GUI interaction is done with these elements.

On the other hand, the hardware is divided into two sections: - ADCs - PWMs These represent the

inputs and outputs of the controller respectively.

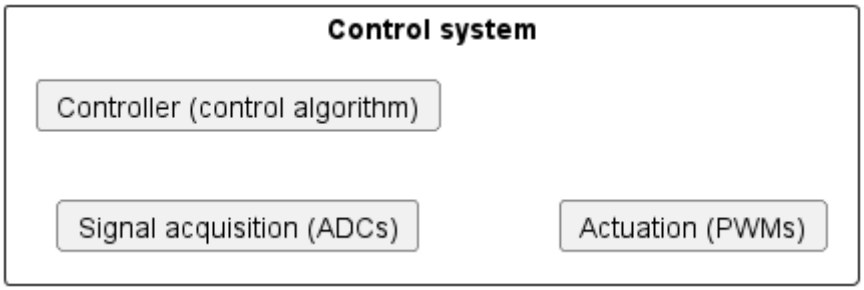


Figure 2. Control system diagram

2.1.2. Traces

Related to the input and output signals that interact with the controller. The information is stored in the controller's RAM and is listed as the different traces that can be visualized later in the GUI.

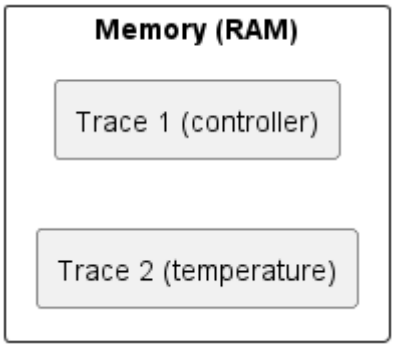


Figure 3. Traces diagram

2.2. Communication with the Controller

As mentioned earlier, the control systems and traces are found within the controller. Therefore, an interface is defined to gain access by sending a message to the controller with a command followed by the data.

Table 1. Basic command structure

CMD	DATA

Following this same pattern, it gives the flexibility that a control system can interact with hardware and controllers independently.

Example: set controller gains of PI controller of control system 1

CMD (controller interface of control system)	DATA (ID of control system, gains)

Example: set PWM frequency of control system 1

CMD(hardware interface of control system)	DATA (ID of control system, frequency)

2.3. TCP protocol/IP in the controller

Definition

TCP/IP, standard Internet communications protocols that allow digital computers to communicate over long distances. The Internet is a packet-switched network, which means that data is divided into smaller packets, sent individually over the network, and then reassembled at the destination. TCP/IP is a suite of protocols that governs the way data packets are transmitted over the Internet. It is the foundation of the Internet and is used by virtually every application that communicates over the network. Source

The controller has a TCP/IP server that allows the GUI to communicate with it. The server listens for incoming connections from the GUI and processes the commands sent by the GUI. The server then sends the appropriate responses back to the GUI.

The TCP/IP server uses a simple command-response protocol to communicate with the GUI. The GUI sends commands to the server, and the server sends responses back to the GUI. The commands and responses are formatted as strings, with each command or response consisting of a command code followed by a set of parameters.

So the communication structure with the controller is as follows:

Table 2. Low level TCP/IP data format

Data size (4bytes,signed)	DATA Command + Data

It should be considered that the controller always replies with a message of at least 4 bytes in size.

Example: get status of control system 1

- *Sent to embededd controller*

Data size(4+4) (4 bytes)	Command - get status (4 bytes)	ID of control system (4 bytes)

- *Reply from embededd controller*

Data size/status (4) (4 bytes, signed)	Status (4 bytes, signed)

The return value is used to indicate status:

- If zero, command was executed successfully
- If positive, there will be a response and the value indicates size of the incoming data
- If negative, there was an error executing the command

3. LabVIEW Project

3.1. Project overview

The GUI application is the main interface for the user to interact with the control system. It is responsible for displaying the data, sending commands to the microcontroller, and handling errors. The GUI application is built using LabVIEW, a graphical programming language that allows for easy and intuitive development of user interfaces.

3.2. LabView software

LabVIEW (Laboratory Virtual Instrument Engineering Workbench) is a system-design platform and development environment for a visual programming language from National Instruments. It is commonly used for data acquisition, instrument control, and industrial automation on a variety of platforms including Microsoft Windows, various versions of Unix, Linux, and macOS. The programming language used in LabVIEW, also referred to as G, is a dataflow programming language. Execution is determined by the structure of a graphical block diagram (the LabVIEW-source code) on which the programmer connects different function-nodes by drawing wires. These wires propagate variables and any node can execute as soon as all its input data become available. Since this might be the case for multiple nodes simultaneously, G is inherently capable of parallel execution. Multiprocessing and multi-threading hardware is automatically exploited by the built-in scheduler, which multiplexes multiple OS threads over the nodes ready for execution.

3.3. General Project Structure

As mentioned earlier, the project is divided into 3 main parts: the graphical interface, the interface controller, and the serial communication controller. The structure of the project in LabVIEW is shown below.

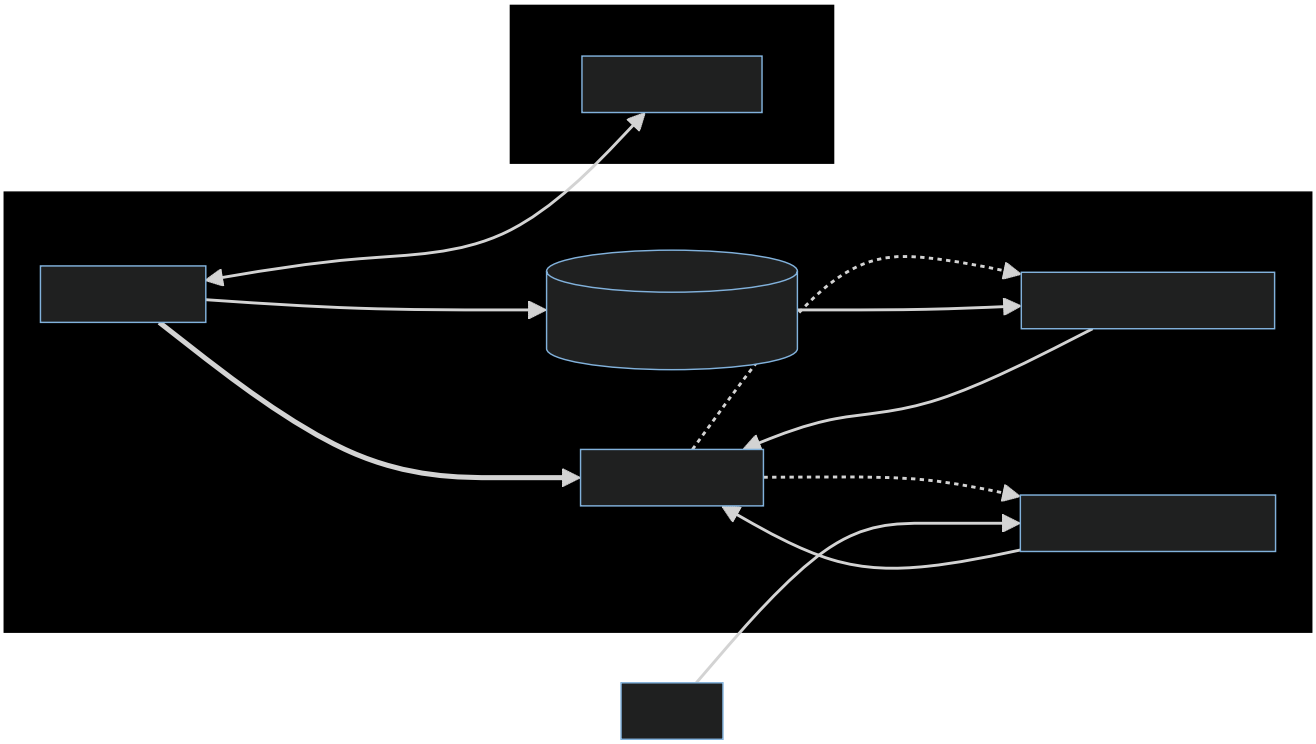


Figure 4. Project Structure of the GUI application

3.4. LabView Project Structure

The graphical interface was developed within the LabVIEW environment, specifically in a LabVIEW project. A LabVIEW project is a file that contains all the files necessary to develop an application in LabVIEW. It helps manage and streamline the development process, especially for complex systems that may include multiple VIs (Virtual Instruments), libraries, shared variables, hardware interfaces, and other resources.

Within the LabVIEW project, the sub-blocks are organized in the following way:

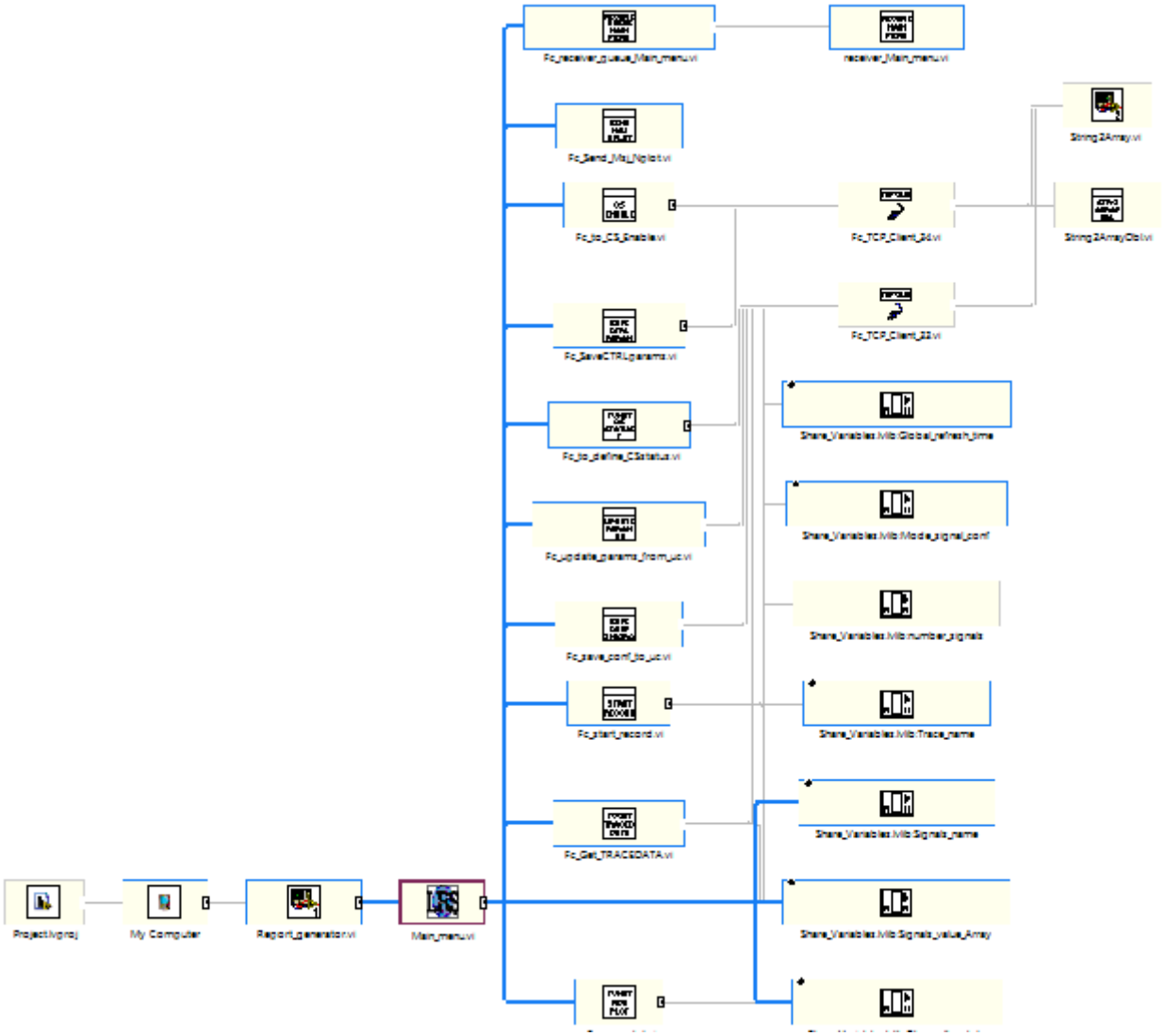


Figure 5. VI Hierarchy of the LabView Project

Where the main sub-blocks are:

- **Main_menu.vi:** It is the main menu of the graphical interface, in it are the access buttons to the different functionalities of the graphical interface.
- **Window_plot.vi:** It is the sub-block responsible for the real-time data visualization.
- **Call_signal_fromHW.vi:** It is the sub-block responsible for requesting the data from the Main_menu signals to Window_plot.
- **Call_signal_fromfile.vi:** It is the sub-block responsible for data from a csv file to Window_plot.
- **Shared_variables.lvlib:**

It is the sub-block responsible for communication between the different sub-blocks of the graphical interface through shared variables. All window_plot share the same variables of this sub-block. Except Main_menu.vi, the other sub-blocks (or subVIs) are called by Main_menu.vi and can be reentrant, meaning they are cloned with the purpose of being able to load multiple windows for real-time data visualization.

3.5. Main_Menu.vi

The main menu function acts as the focal point for navigating and utilizing the diverse range of features and capabilities within our application. When users initiate the application, they encounter a user-friendly menu interface, offering a variety of options that cater to their specific requirements.

Figure 6. Main_menu.vi icon

Key functionalities include:

- Initiating window plots for data visualization.
- Establishing communication with microcontrollers.
- Implementing robust error handling mechanisms

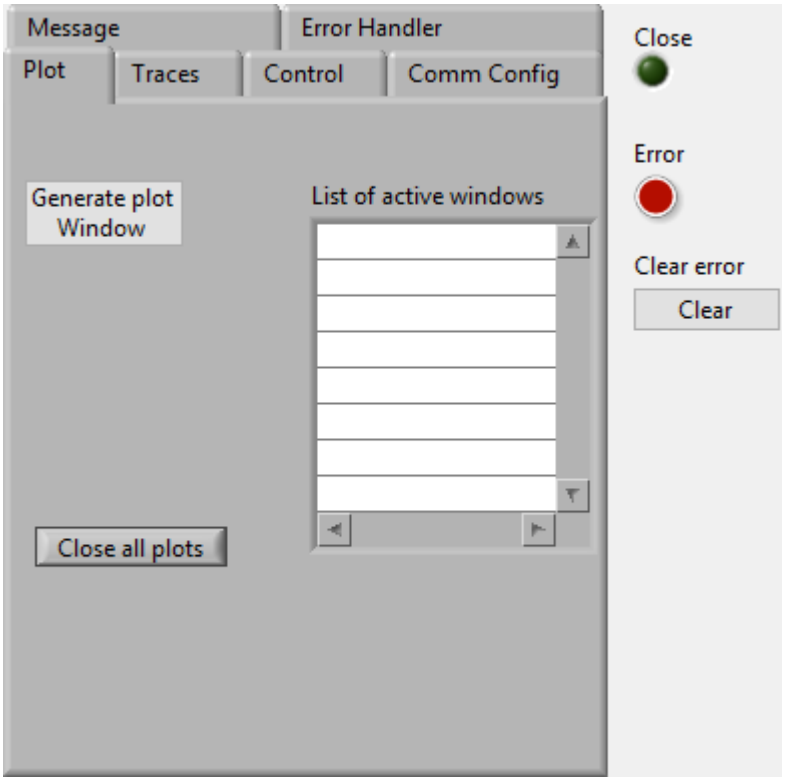


Figure 7. Main Menu front panel

3.6. Window_plot.vi

This window has the capability to visualize and analyze the graphs of the stored signals from the Main_menu.vi. The user can select the desired signal to be displayed on the graph, and the graph

will be updated in real-time as new data is received. The user can also adjust the scale of the graph to better visualize the data.

Figure 8. Window_plot.vi icon

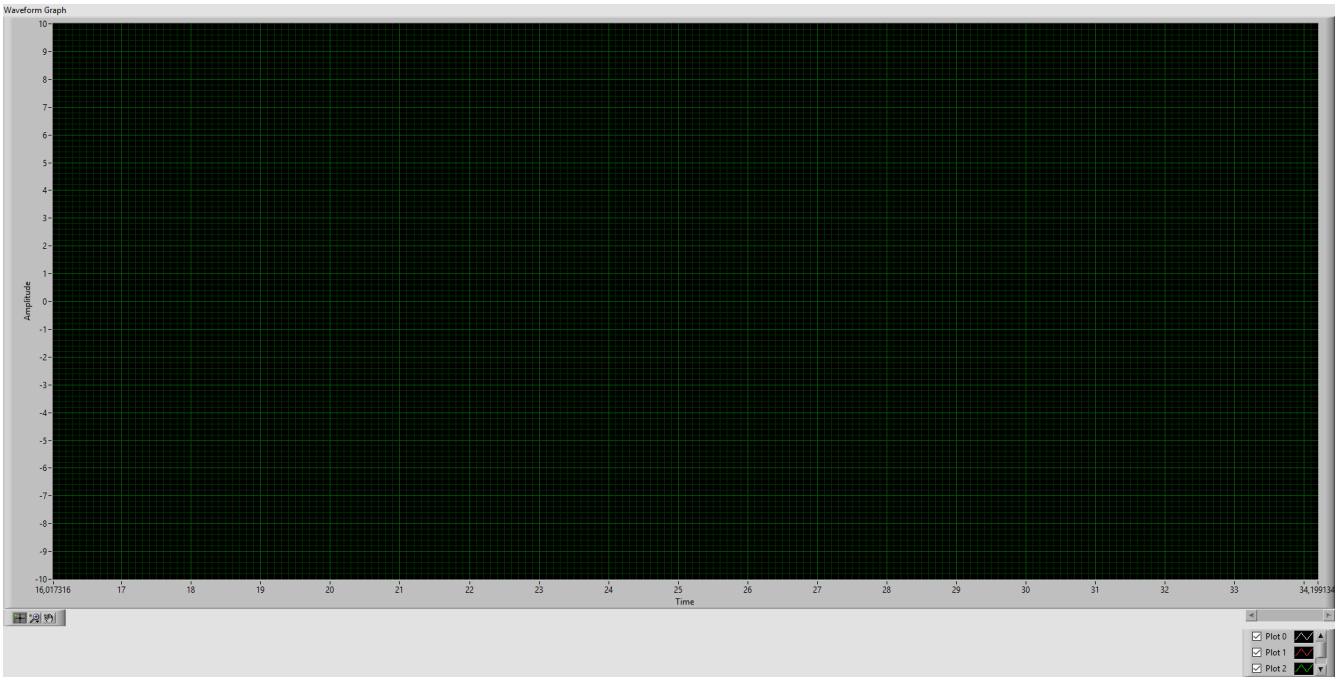


Figure 9. Window_plot front panel

3.7. Call_signal_fromHW.vi

Through this function you can select the stored signals from the Main_menu. This sub-block can be called at any time by any Window_plot.

In the following diagram, the inputs and outputs of this sub-block can be observed.

Figure 10. Call_signal_fromHW.vi icon with inputs and outputs

Thus, in the front panel, the list of signals stored in the Main_menu.vi is displayed, from which the desired one can be selected to be visualized in the Window_plot.vi.

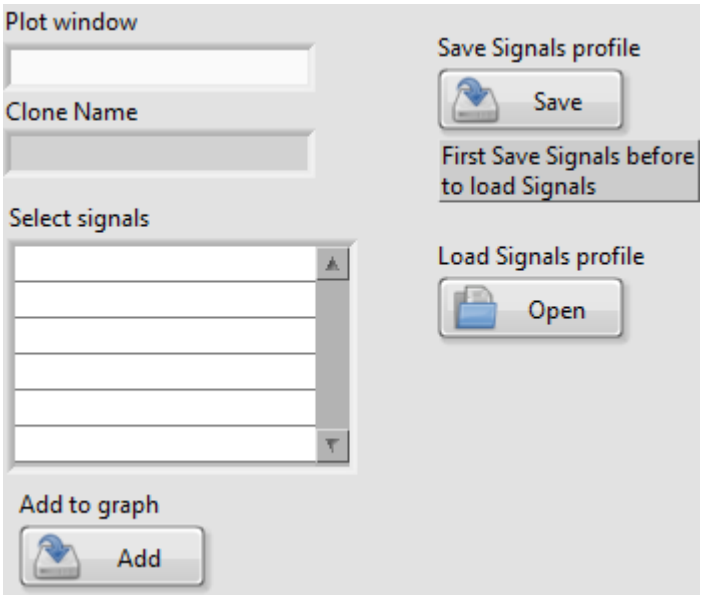


Figure 11. Call_signal_fromHW.vi front panel

3.8. Call_signal_fromfile.vi

This function is responsible for reading data from a CSV file and displaying it on the graph. The user can select the desired file from the file browser and the data will be displayed on the graph in real-time.

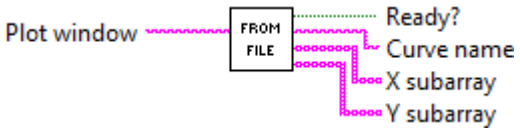


Figure 12. Call_signal_fromfile.vi icon with inputs and outputs

Through this function you can select the signals stored in a CSV file.

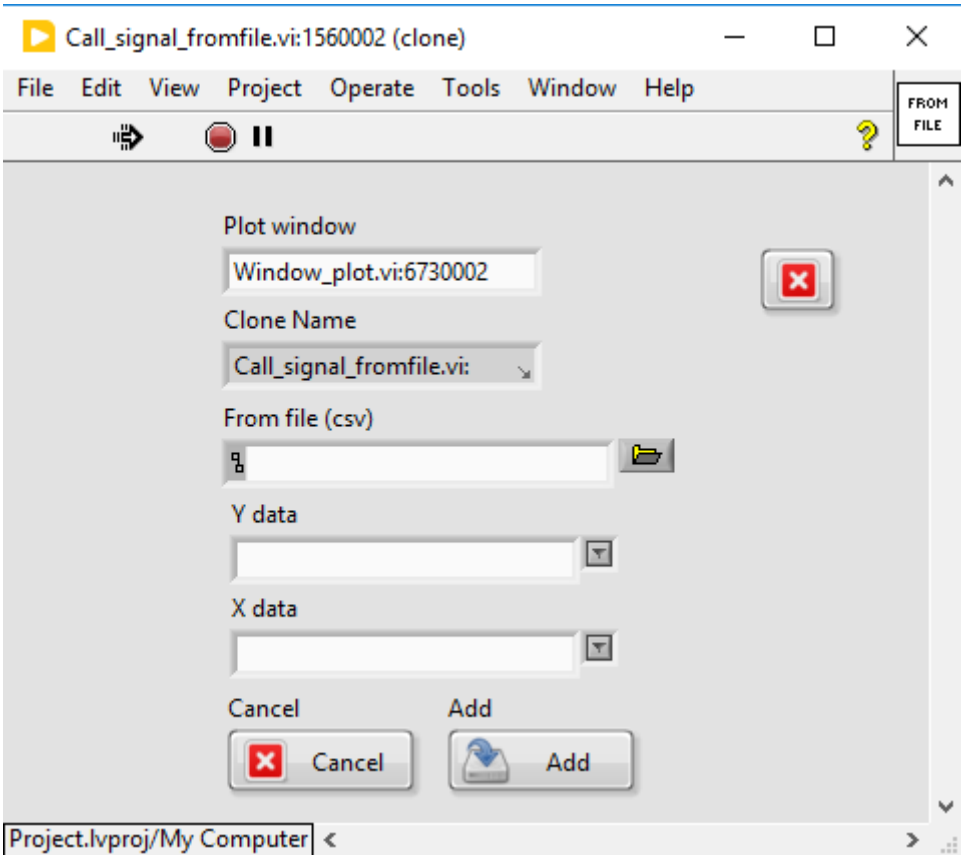


Figure 13. Call_signal_fromfile.vi front panel

3.9. Shared_variables.lvlib

It is the sub-block responsible for communication between the different sub-blocks of the graphical interface through shared variables.

LabVIEW provides access to a wide variety of technologies for creating distributed applications. The shared variable simplifies the programming necessary for such applications. This article provides an introduction to the shared variable and includes a discussion of its features and performance.

Using the shared variable, you can share data between loops on a single diagram or between VIs across the network. In contrast to many other data-sharing methods in LabVIEW, such as UDP/TCP, LabVIEW queues, and Real-Time FIFOs, you typically configure the shared variable at edit time using property dialogs, and you do not need to include configuration code in your application.

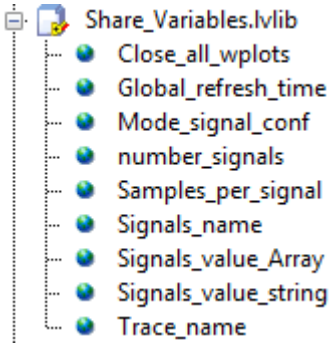


Figure 14. Shared_variables.lvlib in LabVIEW project

4. Getting started of GUI Application

4.1. Overview

The proper way to use the application is by running the executable file GUI_App.exe. This executable was created from the LabVIEW project and includes all the functionalities of the graphical interface.

You can find this executable in the GUI_App folder within the LabVIEW directory. It is generated through the Build Application option in the LabVIEW project.

4.2. Prerequisites

- NI LabVIEW Runtime 2022 Q3 Patch 1 (64-bit). Labview Runtime
- Access to the GUI App in the GUI_App folder

4.3. Requirements installation

- Labview Runtime
- Current Version in LabVIEW Development: LabVIEW 2022 Q3 Patch 1 (64-bit)
- Download GUI App in the GUI_App folder

4.3.1. Usage

- Run the executable file GUI_App

4.4. How to run the GUI application

After starting the executable file GUI_App.exe, the graphical interface of the application will be displayed. In which you can view the different options of the application (Main_menu.vi).

4.4.1. Through Main_menu.vi

The first step will be to navigate to Comm Config (communication configuration) and define the IP address and communication port with the microcontroller.

Click on the Save option, where the port and IP address will be saved. Subsequently, the application will verify if there is a connection with the microcontroller. If the microcontroller is connected, a check can be seen in the Connected box.

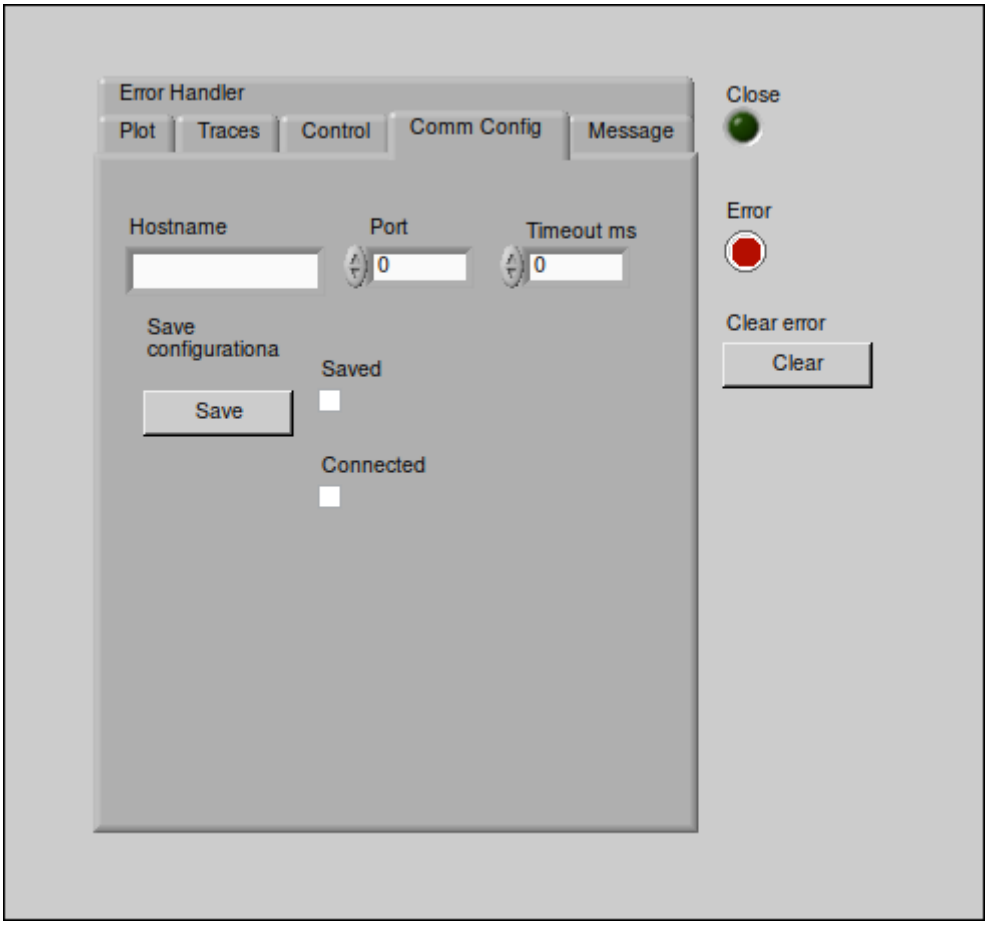


Figure 15. Communication Configuration tab in Main_menu.vi

Having a connection, you can move to the control tab where you can view the current control set. With the "CS enable" and "CTRS enable" buttons, they activate the control system and control respectively.

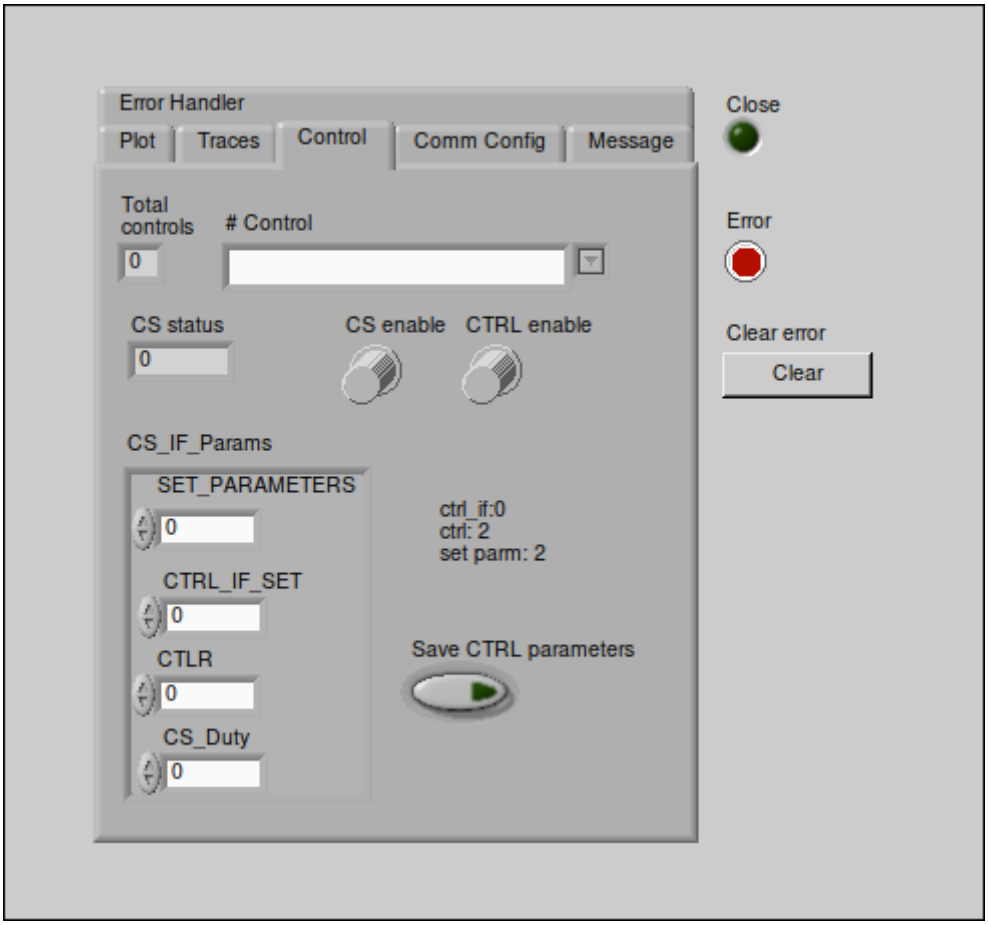


Figure 16. Control tab in Main_menu.vi

In the Traces tab, you can see the name of the current traces system and samples per signal. It is determined if the data is obtained continuously or only once in the "Global Mode" option and the refresh time in "Refresh Time". Having the options ready, click on the "Start recording" button to start obtaining the data.

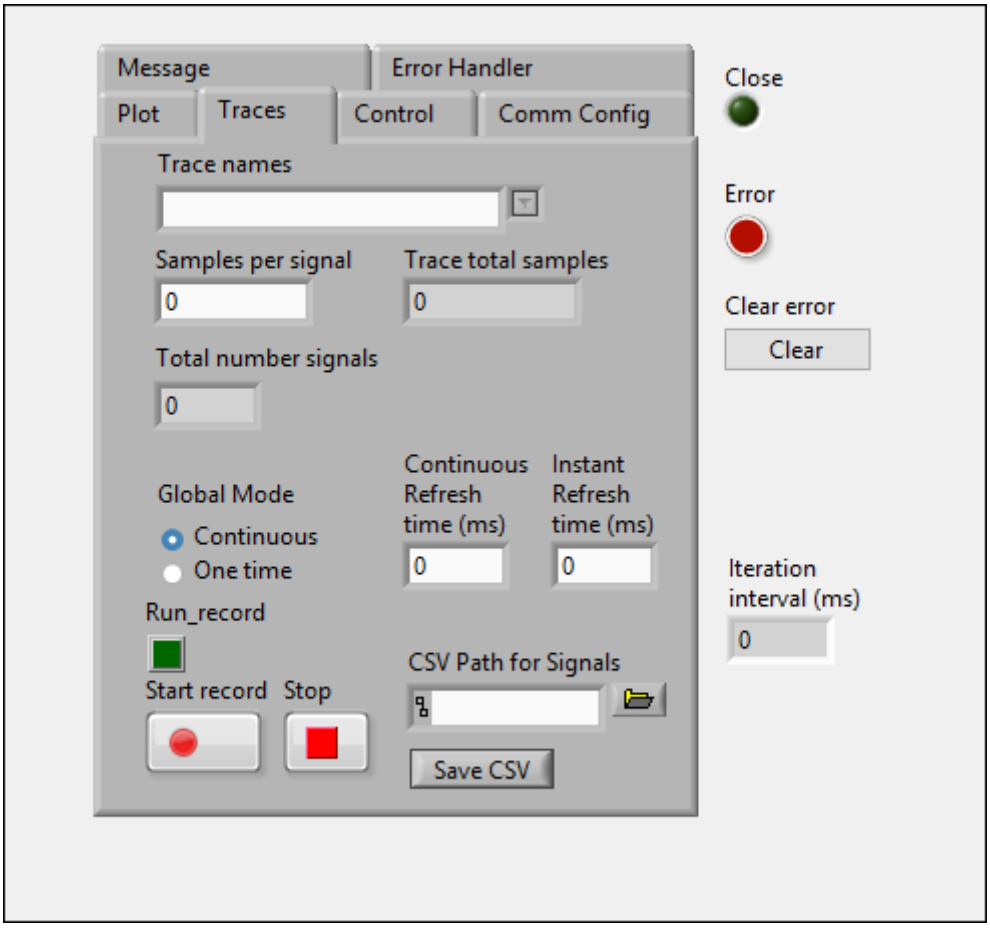


Figure 17. Traces tab in Main_menu.vi

Next, go to the Plot tab, where you can see the possible active windows and there is a button to close them ("Close all plots") in a single instant. Clicking on the "Generate plot window" button displays a window with the graphs of the obtained signals.

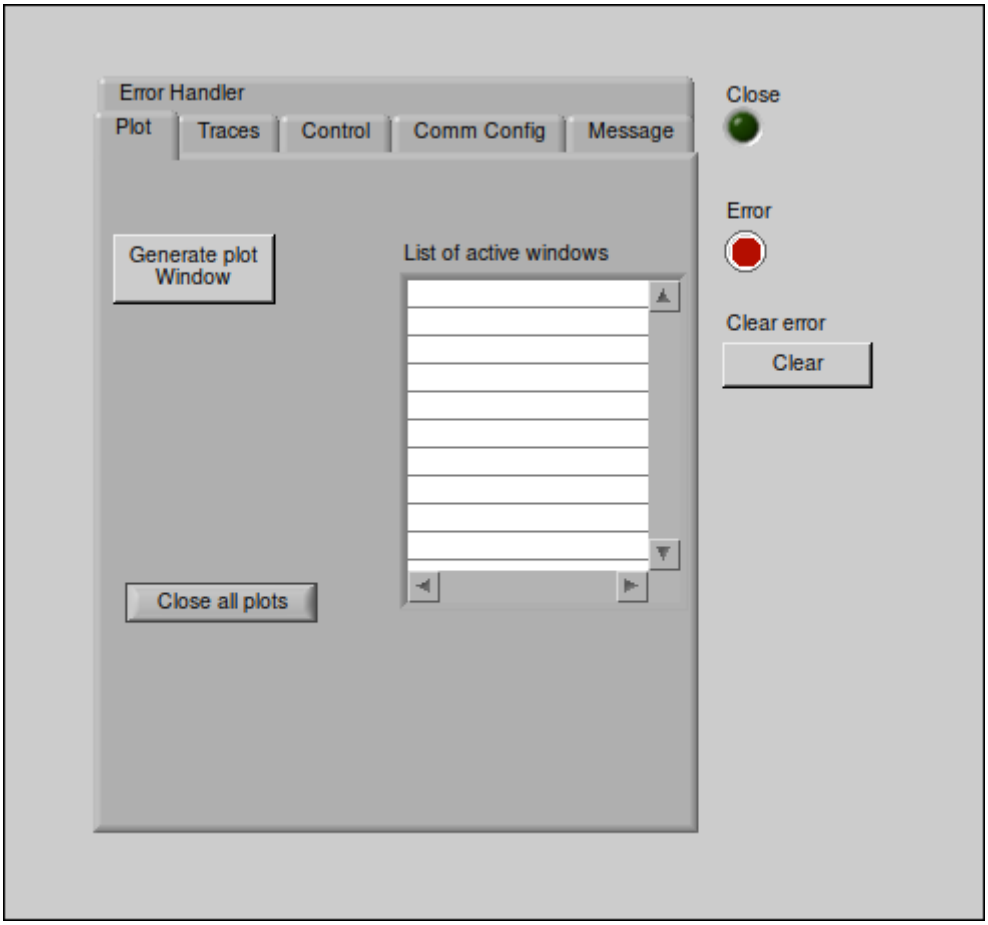


Figure 18. Plot tab in Main_menu.vi

4.4.2. Through Window_plot.vi

After having an active graph window, the obtained signals can be visualized. A graph showing the signal amplitude on the Y-axis and time on the X-axis.

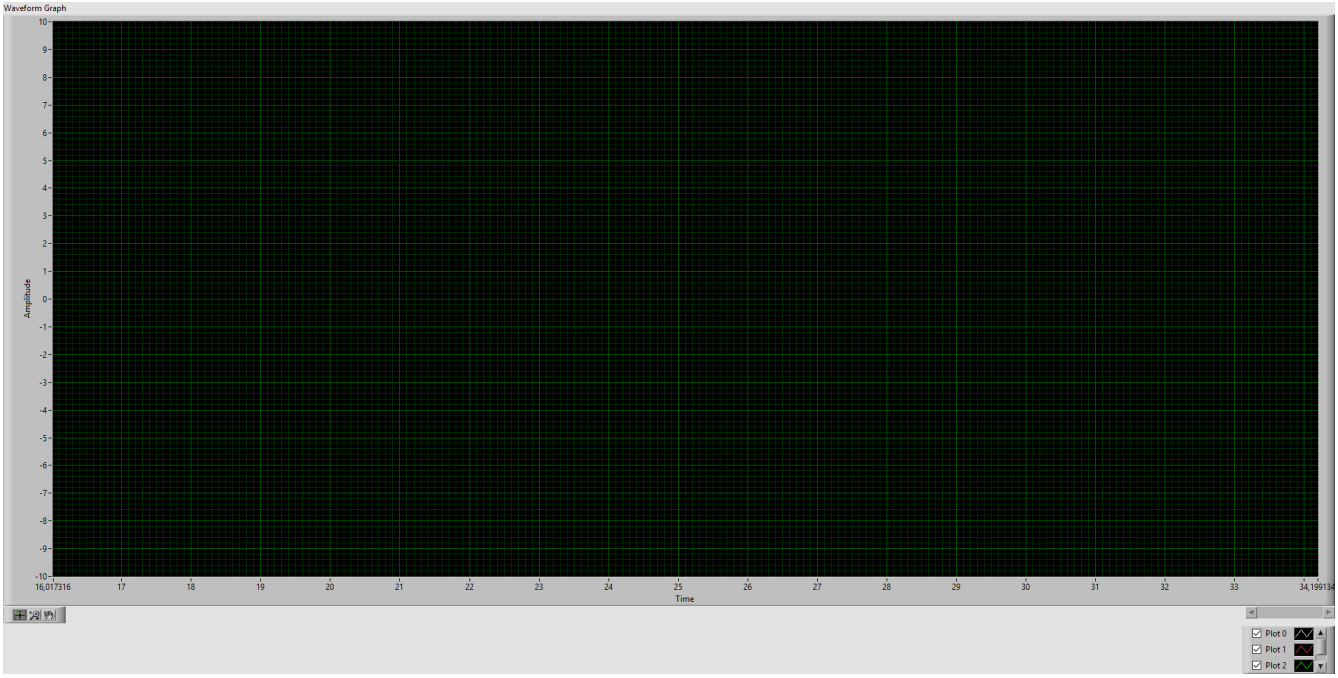


Figure 19. Window_plot front panel

Thus, in the upper left part of the window, you can proceed to request the required signal through

the following two options. In the option that says "New" allows you to choose between two options:

- Get signal from HW: Allows you to select the signal stored in the Main_menu.vi. This signal was obtained through communication with the microcontroller.
- Get signal from file: Allows you to select the signal stored in a csv file.

At the moment of selecting the signal, it will be displayed on the graph and its name will be displayed in the lower right part of the window, in the "Plot legend" table

Figure 20. Table of plot legend.vi

4.4.3. Close Window_plot.vi

The simplest way to close the graph window is to click on the "Close" button in the upper right part of the window. Or also by clicking on the "Close all plots" button in the Main_menu.vi window.

5. Main_menu.vi in detail

5.1. Overview

Main_menu.vi is the main VI of the application. It is the first VI that is called when the application is launched. It is responsible for creating the main window of the application and for managing the different sub-VIs that are called from the main window. The Main_menu.vi is divided into two main sections: the front panel and the block diagram. The front panel is the user interface of the VI, where the user can interact with the application. The block diagram is the code that controls the behavior of the VI. Thus, the elements found in the front panel of Main_menu.vi will be described below.

In the front panel, the following elements are found:

5.2. Elements

- **Error** Indicator: This indicator is responsible for displaying the errors that have been generated in the application. Each time an error is generated in the application, this indicator is activated. When this happens, you have to move to the "Error Handler" tab to see the error that has been generated. On the other hand, press the "Clear error" button to clear the errors that have been generated in the application and resume the operation of the application. Since the application stops when an error is generated.
- **Clear error** Button: This button is responsible for clearing the errors that have been generated in the application. Each time this button is pressed, the errors that have been generated in the application are cleared.

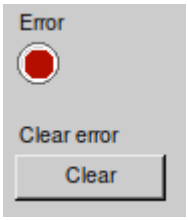


Figure 21. Indicator "Error" and button "Clear error"

5.3. Tab Error Handler

In this tab, the error that has been generated in the application is displayed. In addition, the error code and the error description are displayed. To clear the generated error, you must press the **Clear error** button located to the right of the 'Error Handler' tab."

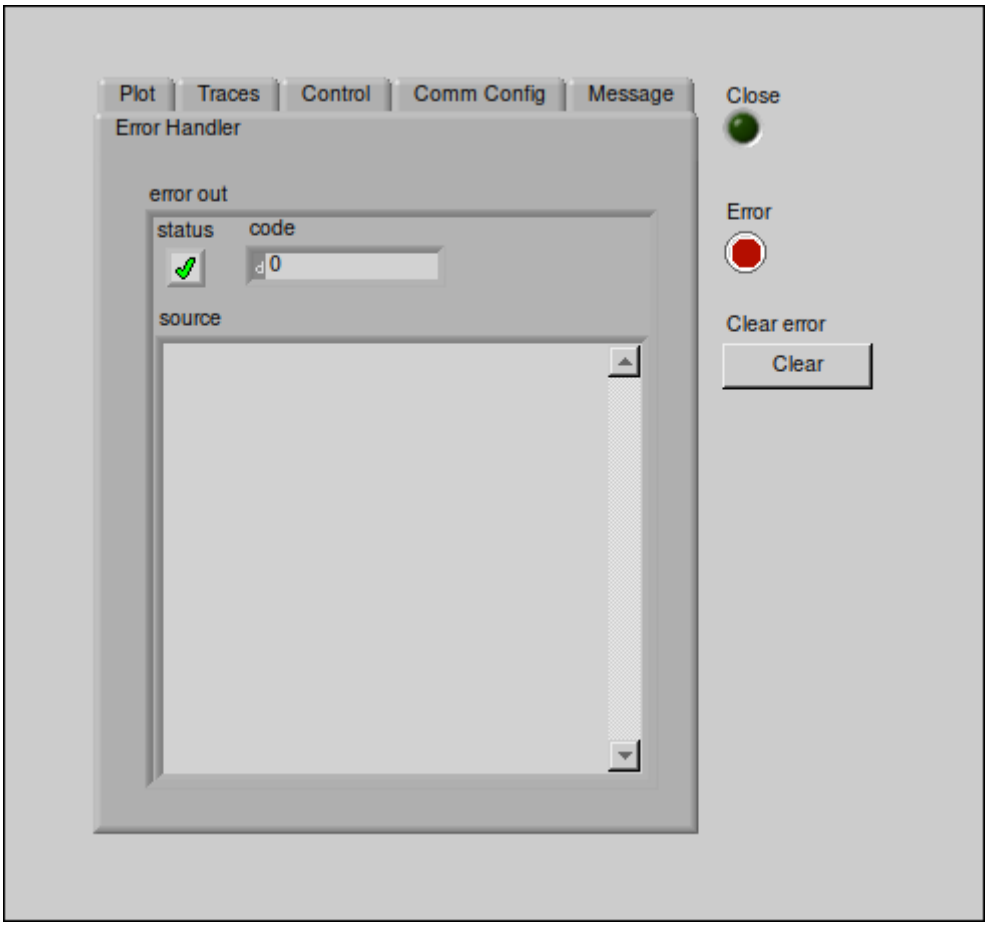


Figure 22. Error Handler tab in Main_menu.vi

5.4. Tab Plot

In this tab you can see the possible active windows in the table **List of active windows**. In the bottom, there is a button to close them **Close all plots** in a single instant. Clicking on the **Generate plot window** button displays a window with the graphs of the obtained signals.

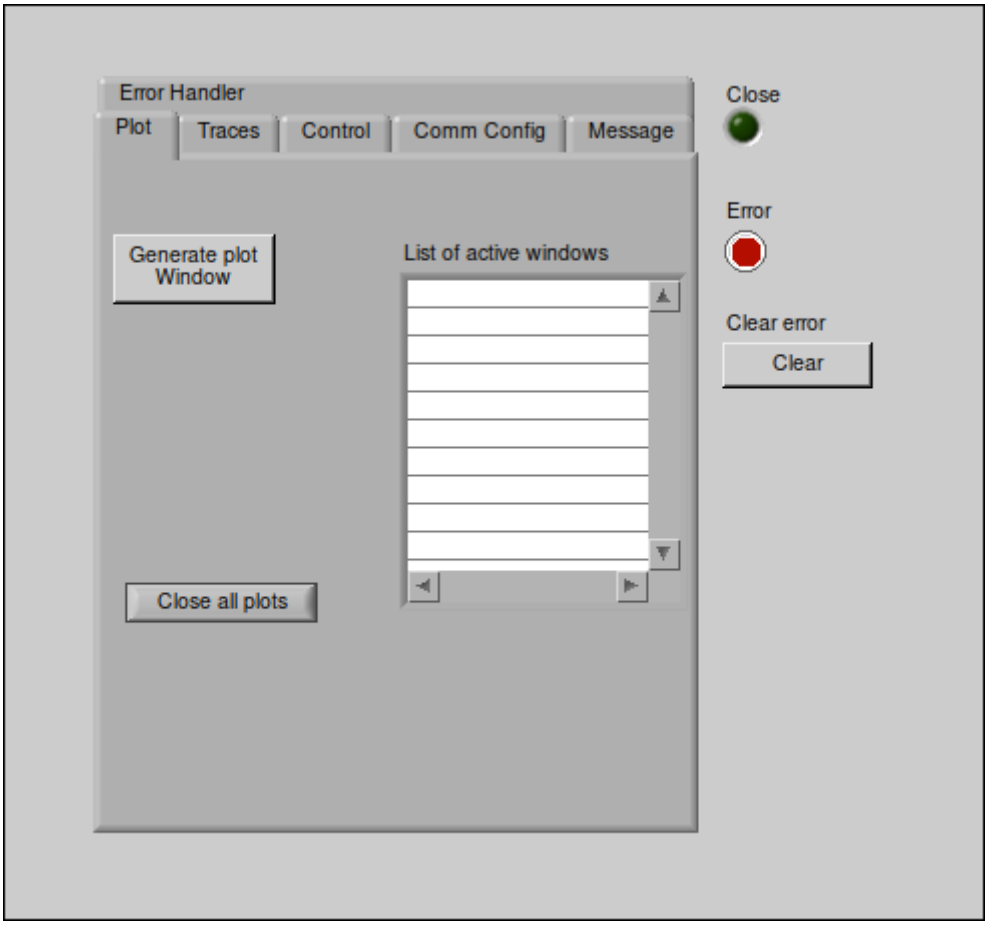


Figure 23. Plot tab in Main_menu.vi

5.5. Tab Traces

In the Traces tab, you can see the name of the current traces system in the microcontroller **Trace names**.

In **Samples per signal** is visualized the current samples per signal that are being obtained and can be set the quantity to get from the microcontroller.

The indicator **Trace total samples** shows the total samples that are being obtained from the microcontroller. And the indicator **Qty of signals** shows the quantity of signals that are being obtained from the microcontroller.

In **Global Mode** is determined if the data is obtained continuously or only once.

The **Refresh time (ms)** determines the time that the application waits to obtain the data again. In the current version, there are two types of Refresh time, which correspond to either Continuous mode or Instant mode and are taken into consideration depending on the selected Global Mode. However, for it to work, the Start record button must be pressed.

Finally, the **Start record** button to start obtaining the data from microcontroller depending on the previous configurations. It is possible to stop the data acquisition by pressing again the **Start record** button. Changing set up values will be applied when the data acquisition pressing again the **Start record** button. The indicator **Run_record** allow to be aware of the status of the data acquisition.

The button **CSV** is used to save the all the data obtained in a CSV file.

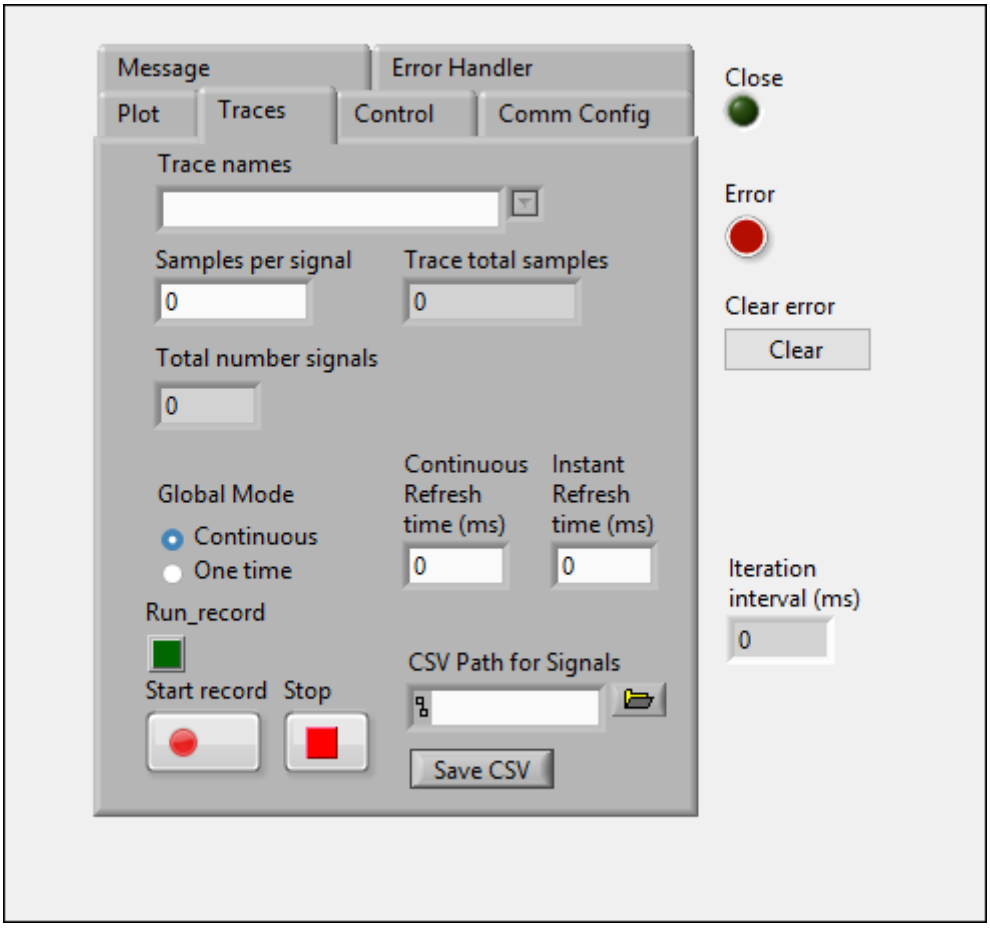


Figure 24. Traces tab in Main_menu.vi

5.6. Tab Control

In the # **Control** selector, the available controls to be selected are displayed. In turn, in **Total controls**, the number of preset controls in the control system (CS) is displayed.

In **CS status**, the status of the control system is displayed. The value will depend on the assigned control system configuration. *Note: Check the control system configuration to interpret its status.*

Next, the **CS enable** button enables the control system, indicating a color change in the button. Just like the **CTRL enable** button activates the selected control.

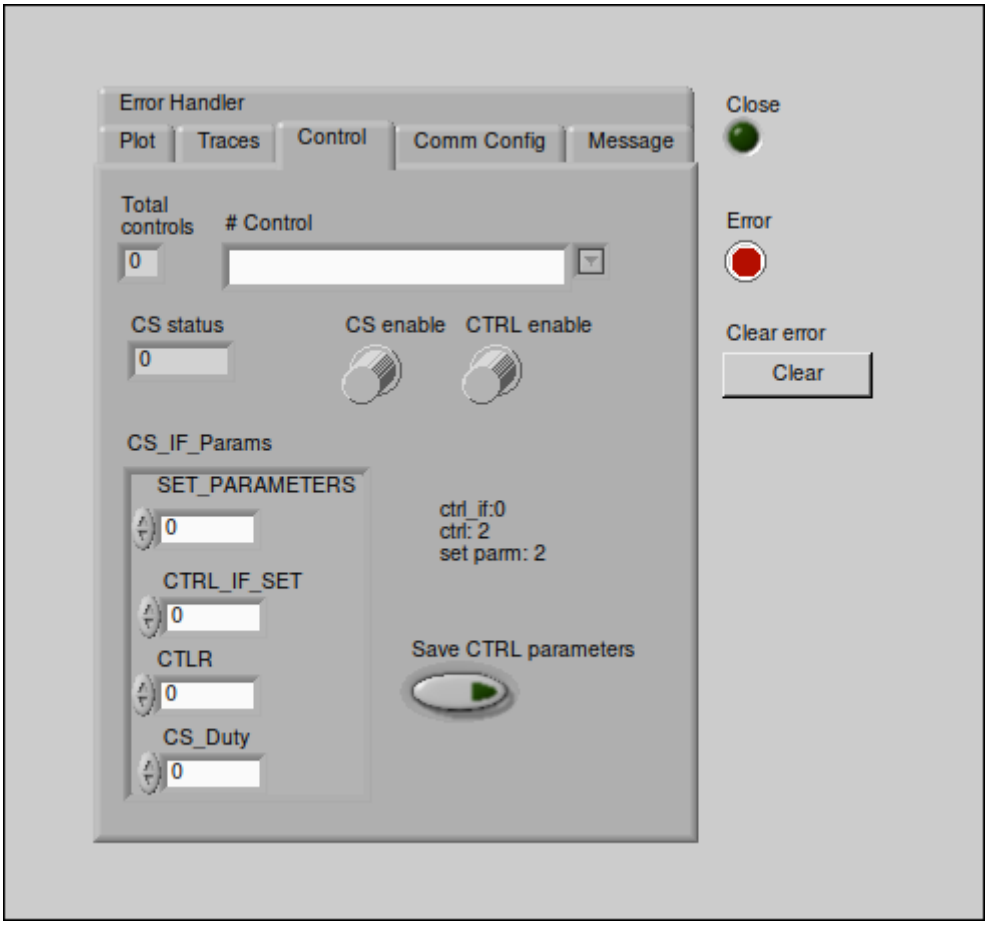


Figure 25. Control tab in Main_menu.vi

5.6.1. Section CS_IF_Params

In the **CS_IF_Params** block, there are four numeric controls that allow you to configure the parameters of the control interface. These parameters are: - SET_PARAMETERS - CTRL_IF_SET - CTRL - CS_DUTY

Upon setting the values of these parameters, the **Save CTRL parameters** button must be pressed so that the changes are applied to the microcontroller. This action can be performed at any time as it represents the sending of a data packet to the microcontroller as shown in the following table.

Table 3. Message CS_IF_Params command structure

SET_PARAMETERS	CTRL_IF_SET	CTRL	CS_DUTY

It should be clarified that these parameters will depend on the configuration developed in the controller. It will require reviewing the documentation of the microcontroller development.

5.7. Tab Comm Config

The Comm Config tab represents the section to configure the communication between the microcontroller and the application. In this section, the following control elements are found:

- **Hostname**: Indicator to display the host name or IP Address to communicate.
- **Port**: Selector to choose the communication port.

- **Timeout ms**: Indicator to display the communication wait time.
- **Save configuration**: Button to save the communication configuration.

Clicking on the **Save** button, port and IP address will be saved in the application and the **Saved** box will have and check mark.

Subsequently, the application will verify if there is a connection with the microcontroller. If the microcontroller is connected, a check mark will be in the **Connected** box.

Figure 26. Communication Configuration tab in Main_menu.vi

5.8. Tab Message

This tab is only used to send messages and data to the active windows in the application. Only the messages can be viewed within the block diagram of the respective active window.

Just for development purposes

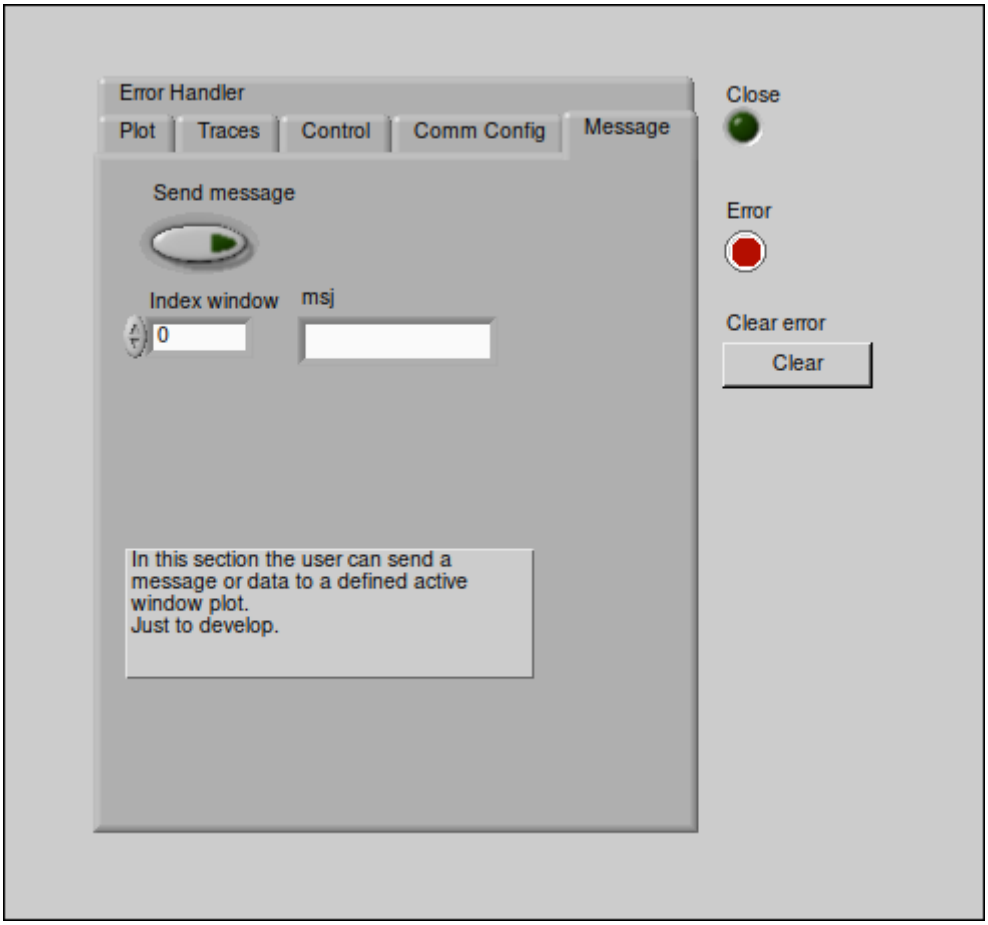


Figure 27. Message tab in Main_menu.vi

6. Window_plot.vi in detail

6.1. Overview

This VI is used to plot data in a window. It is a simple wrapper around the XY Graph control where the signal data is plotted.

6.2. Elements

6.2.1. Waveform Graph

The waveform graph displays one or more plots of evenly sampled measurements.

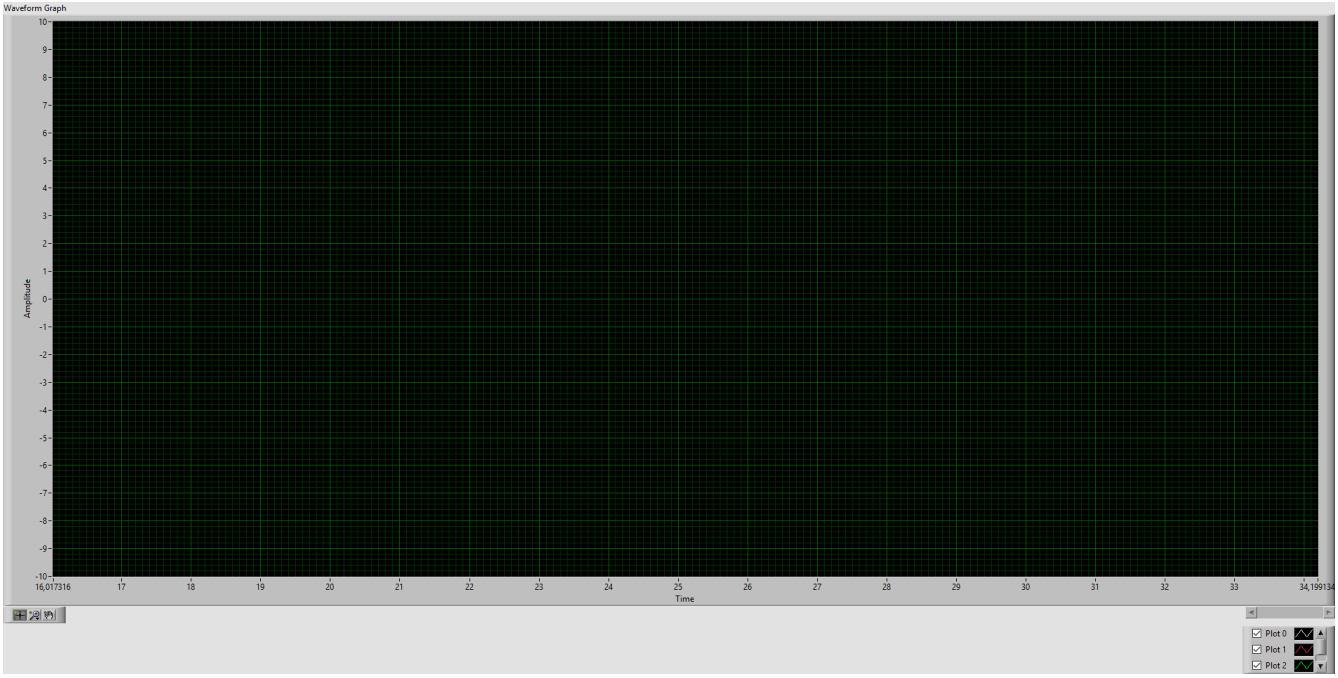


Figure 28. Window_plot front panel

Properties

The waveform graph has the following properties:

- Graph Pallet

The Graph Palette in LabVIEW is a set of tools and options that allow you to customize the appearance and behavior of a graph or chart. It is typically available when you right-click on a graph or chart control in the front panel window Graph Palette Documentation.

Figure 29. Graph Pallet

Here are some of the features you can access from the Graph Palette:

Cursor Movement Tool: No function.

[Cursor] | *../documentation_log/graphs_doc_vi/graph_pallet/noloc_graph_cursor.gif*

Zoom: Zooms in and out of the display.

[Zoom] | *../documentation_log/graphs_doc_vi/graph_pallet/noloc_graph_resize.gif*

Zoom to Rectangle: Zooms in on a rectangle that you draw on the display area.

[Zoom to Rectangle] | *../documentation_log/graphs_doc_vi/graph_pallet/noloc_zoom_torectangle.gif*

X-zoom: Zooms in along the x-axis.

[X-zoom] | *../documentation_log/graphs_doc_vi/graph_pallet/noloc_zoom_xzoom.gif*

Y-zoom: Zooms in along the y-axis.

[Y-zoom] | *../documentation_log/graphs_doc_vi/graph_pallet/noloc_zoom_yzoom.gif*

Zoom In about Point: Autoscales the chart to fit in the display area.

[Zoom In about Point] | *../documentation_log/graphs_doc_vi/graph_pallet/noloc_zoom_tofit.gif*

Zoom Out about Point: Zooms in on a point.

[Zoom Out about Point] |
../documentation_log/graphs_doc_vi/graph_pallet/noloc_zoom_outaboutpoint.gif

Zoom to Fit: Zooms out from a point.

[Zoom to Fit] | *../documentation_log/graphs_doc_vi/graph_pallet/noloc_zoom_inaboutpoint.gif*

Panning Tool: Repositions the plot on the display.

[Panning Tool] | *../documentation_log/graphs_doc_vi/graph_pallet/noloc_graph_repo.gif*

- Table of plot legend

The plot legend in LabVIEW is a graphical feature that provides information about the different plots or data series displayed in a graph or chart. It helps users distinguish between multiple data sets and understand what each plot represents. Here's a detailed overview of the table of plot legend in LabVIEW:

Figure 30. Table of plot legend

Components of the Plot Legend

Plot Name: Displays the name or label of each plot. Can be customized to provide meaningful names that describe the data.

Line Style and Color: Shows the style (solid, dashed, dotted) and color of the line associated with each plot. Helps visually differentiate between multiple plots on the same graph.

Point Style: Indicates the style of data points (circle, square, triangle) used in the plot. Useful for distinguishing between data sets that might otherwise look similar.

Visibility Toggle: Allows users to hide or show specific plots without removing the data from the graph. Useful for focusing on specific data sets while temporarily ignoring other

6.2.2. Offset Control

The offset control is used to set the offset of the signal. The first rectangle is used to select the signal to be offset. And follows the slider to set the offset value.

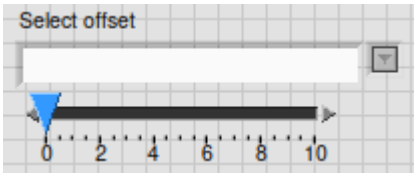


Figure 31. Offset Control

When this control is used, the signals stop for interpretation. The only way to resume data acquisition is by accessing the *Configuration→Local Mode & Refresh Time menu.*

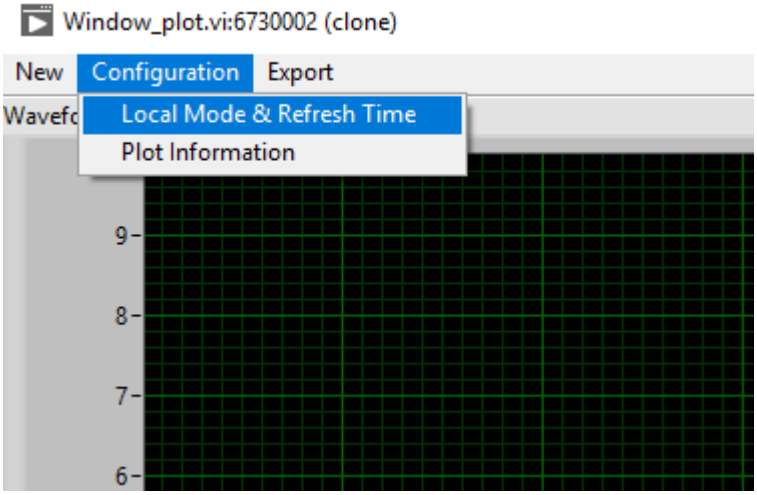


Figure 32. Path to Local mode & Refresh Time option.

In that section, you can resume data acquisition. And also it can be defined the local refresh time of the graph.

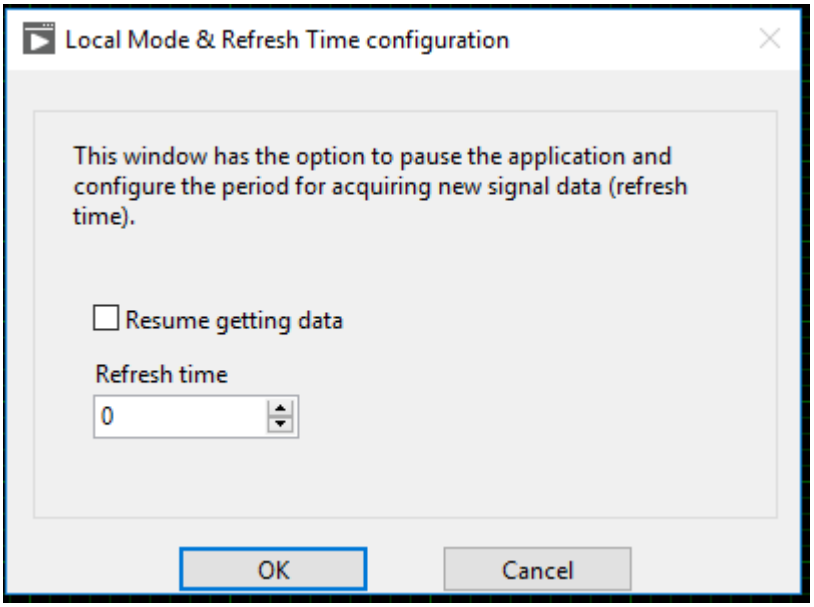


Figure 33. Local mode & Refresh Time window

6.2.3. Get signal from HW

The Get signal from HW is used to get the signal from the hardware.

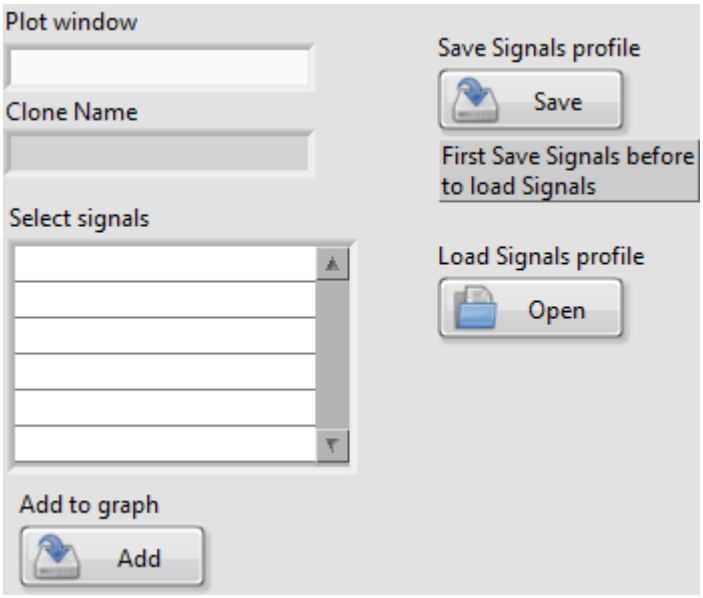


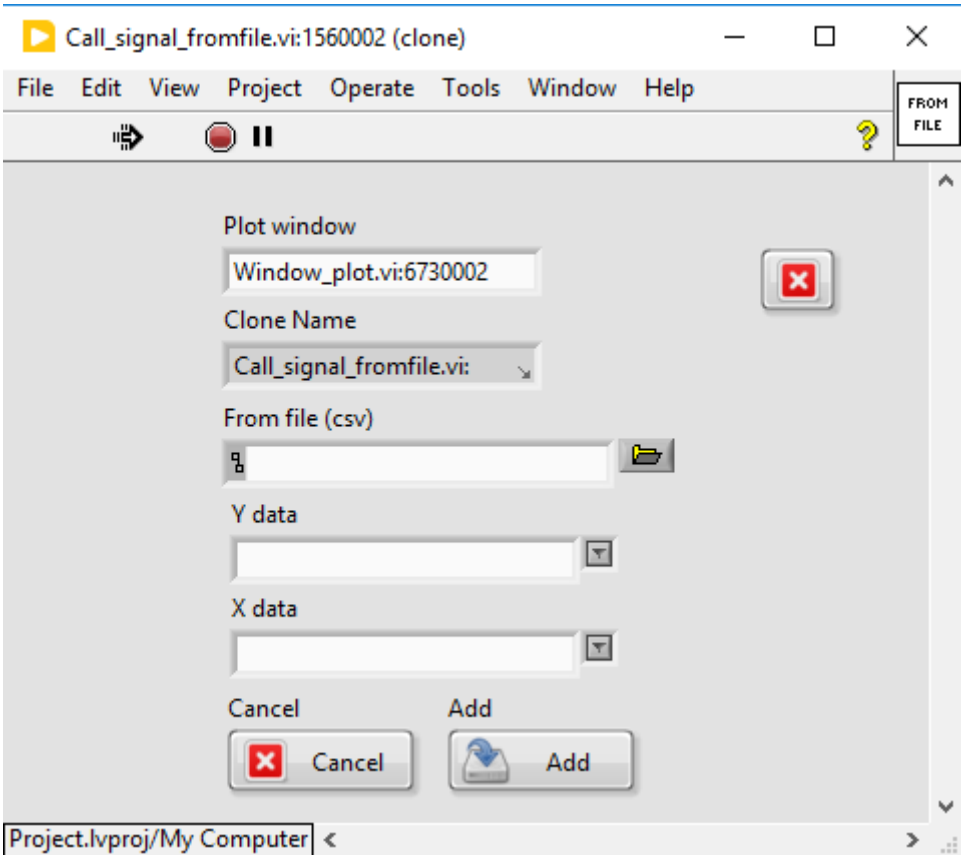
Figure 34. Get signal from HW window

Features of Get signal from HW:

- Plot window: Shows the current name of the caller window_plot.vi.
- Clone Name: Shows the name of the clone version of Call_signal_fromfile.vi.
- Select signal: Select the signals to be plotted. The signals can be select individually or many at once holding keys *Shift* + *Click* or *Ctrl* + *Click* .
- Save Signals profile: Save the signals profile in the temporary file that list the signals to be plotted for the next time the window_plot.vi is called.
- Load Signals profile: Load the signals previously saved.

6.2.4. Get signal from file

The Get signal from file is used to get the signal from a csv file.



Features of Get signal from HW:

- Plot window: Shows the current name of the window.
- Clone Name: Shows the name of the clone version of the window_plot.vi .
- From file (csv): Select the csv file to be plotted.
- Y data: Select the column to be plotted as Y.
- X data: Select the column to be plotted as X.