



Figure 3.175: Illustration of the code of the event “f HS” that keeps the same HS camera frame if the typed frame is not under the acceptable limits

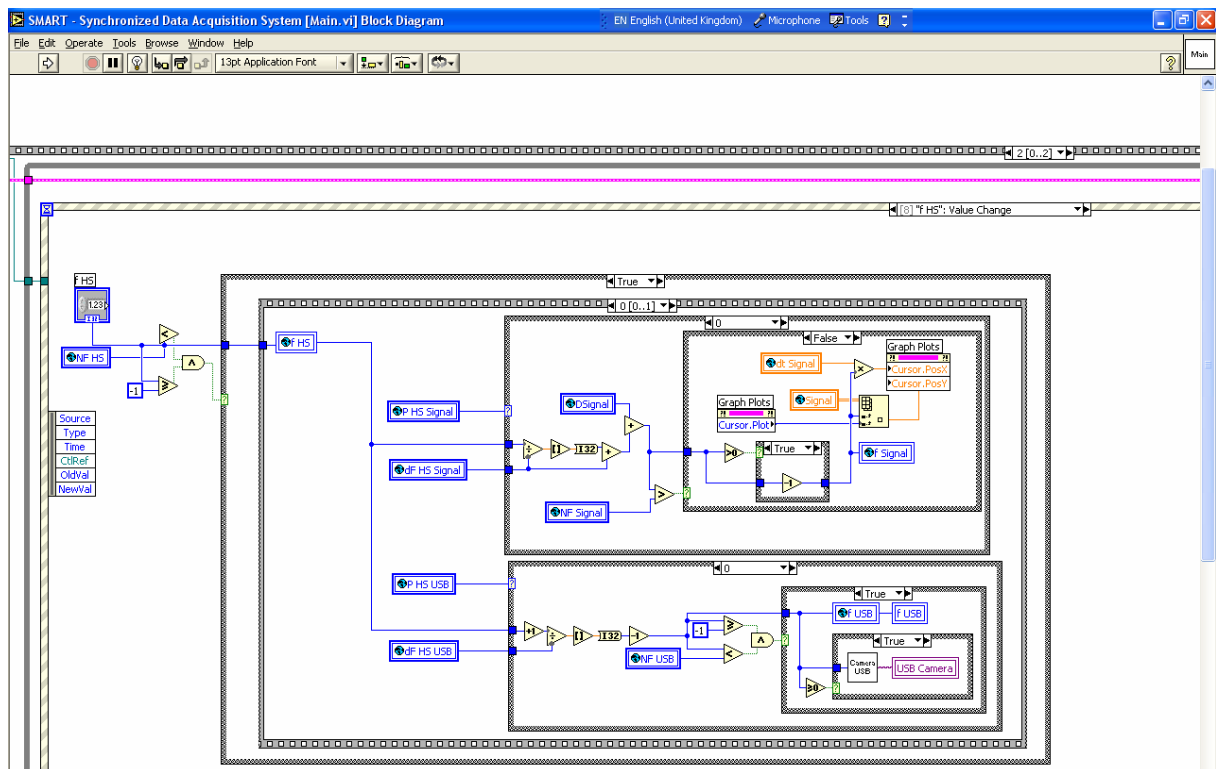


Figure 3.176: Illustration of the code of the event “f HS” that plots an HS camera frame if the typed frame is under the acceptable limits (case 0)

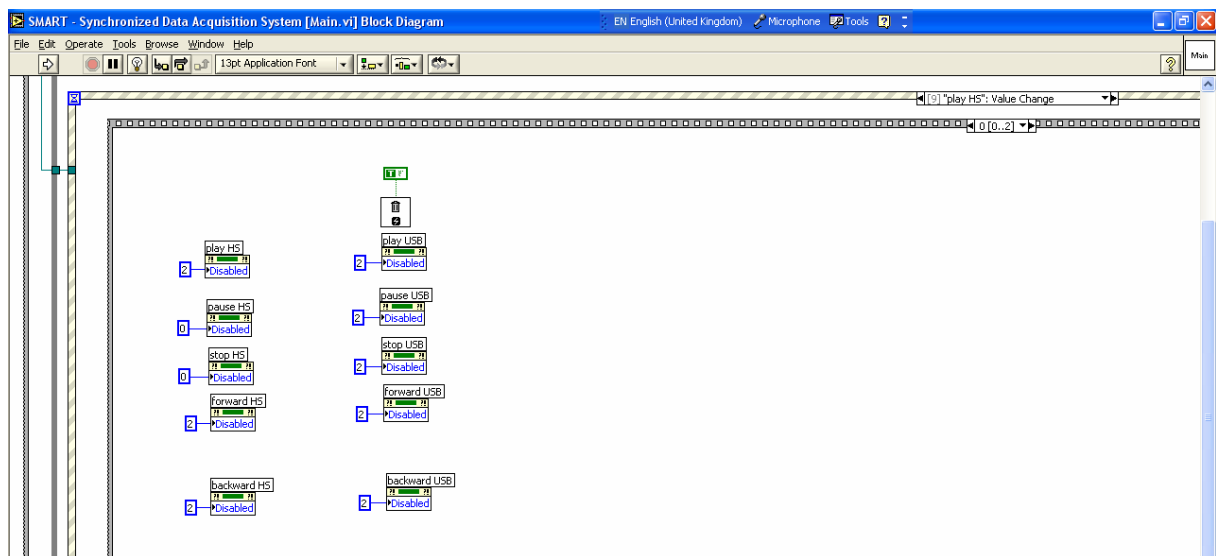


Figure 3.177: Illustration of the code of the event “play HS” that sets the buttons on the main window

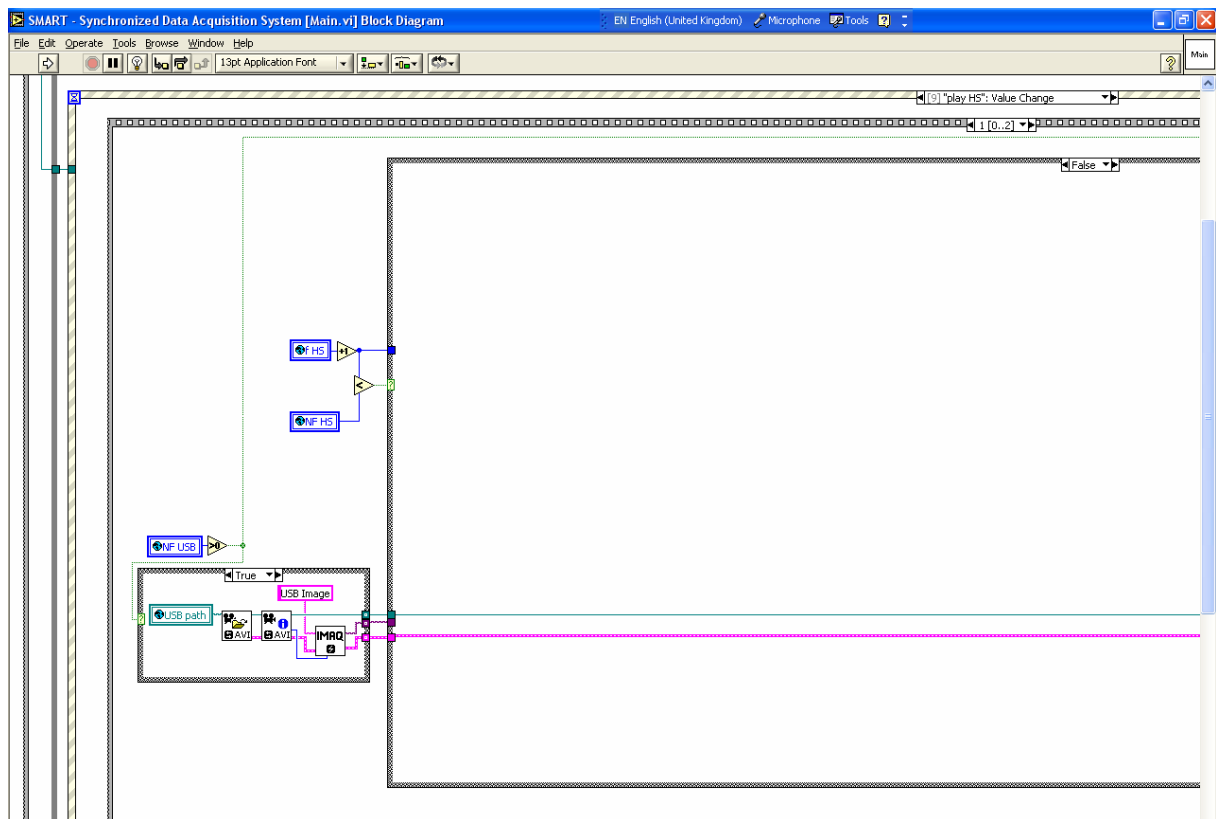


Figure 3.178: Illustration of the code of the event “play HS” that opens the USB camera file

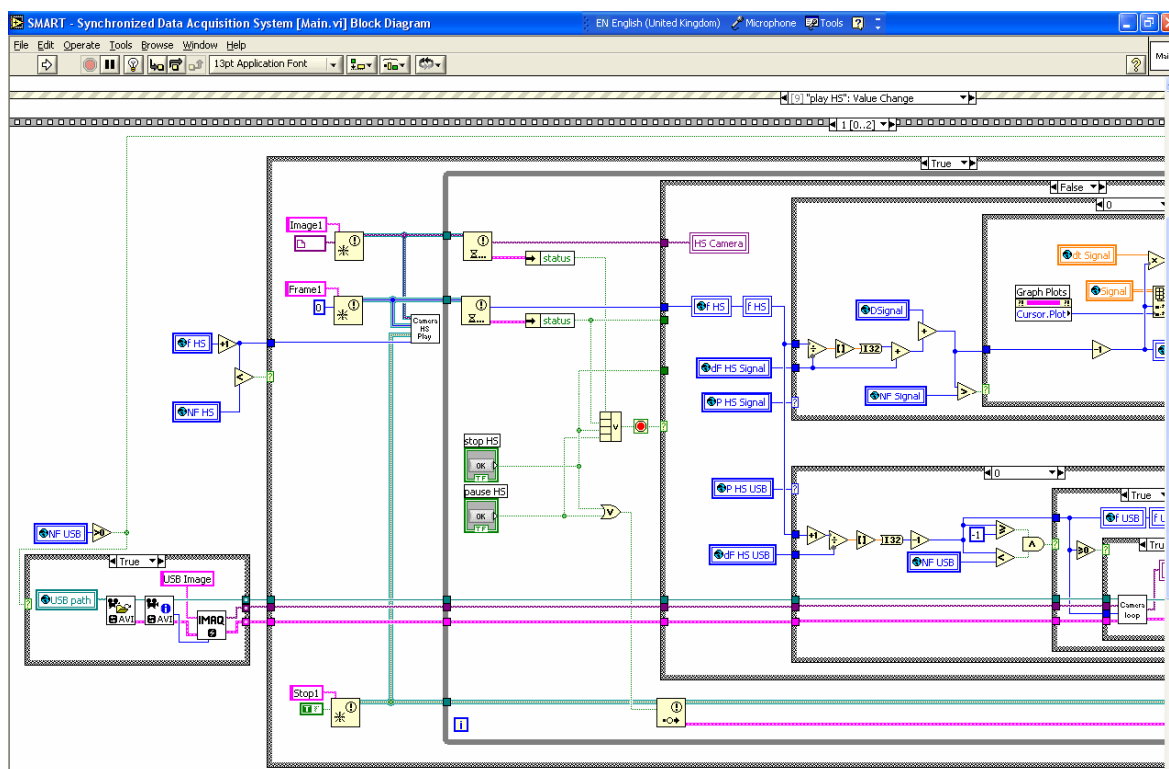


Figure 3.179: Illustration of the code of the event “play HS” that calls the function “Play Camera HS”

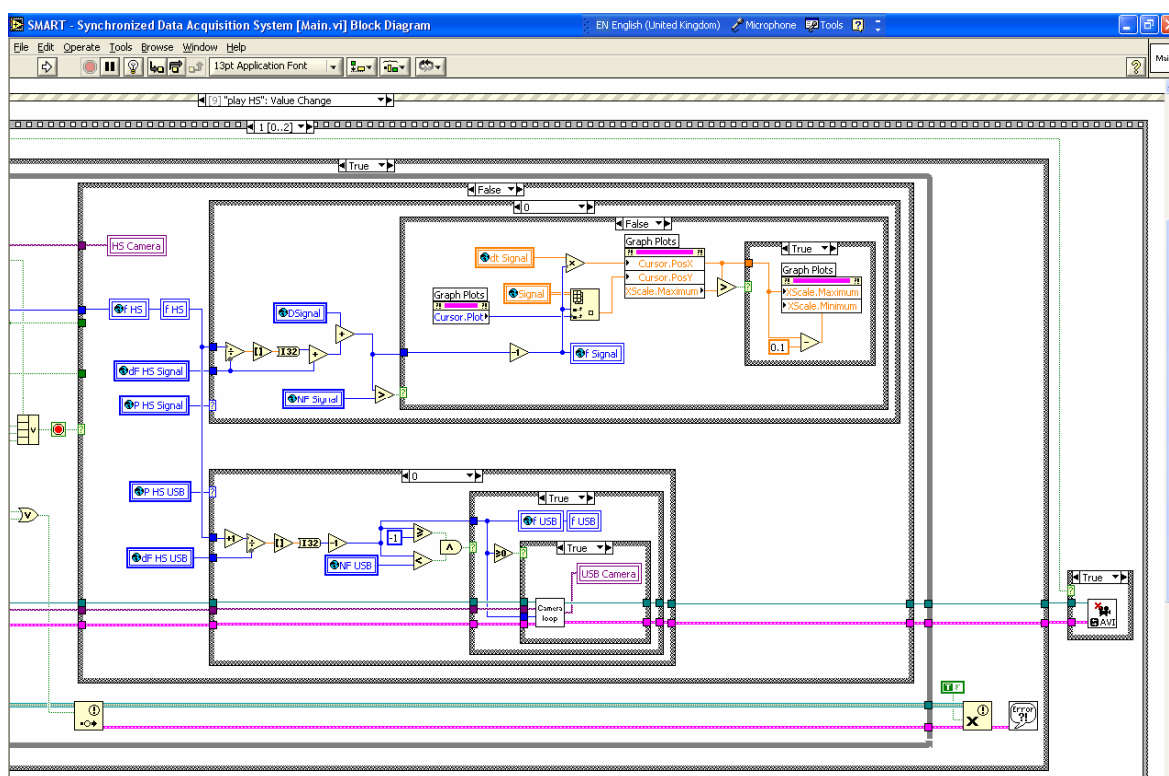


Figure 3.180: Illustration of the code of the event “play HS” that calls the function “Camera loop”

Function “Camera loop” (Figure 3.184) is responsible for playing the USB camera frames, as Figure 3.185 shows. Eventually, all the buttons are enabled again, as illustrated by Figure 3.186.

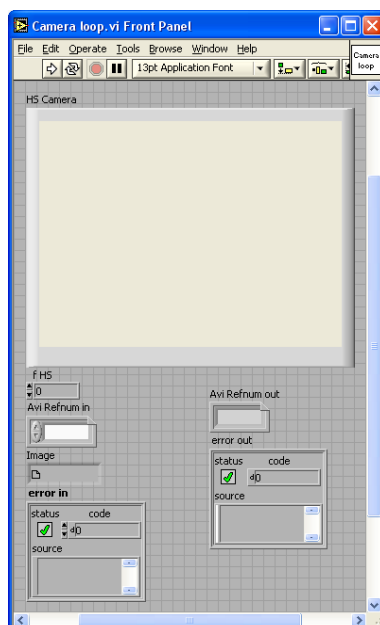


Figure 3.184: Illustration of the front panel of the function "Camera loop"

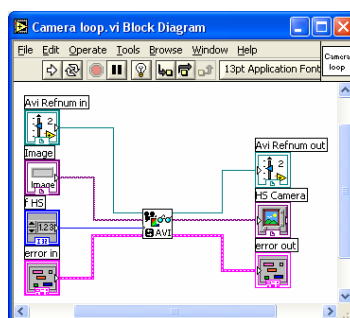


Figure 3.185: Illustration of the code of the function "Camera loop" that plays the USB camera frames

When the button “play USB” is pressed, the current images are disposed and the other buttons related to the cameras are disabled, except for the buttons “pause USB” and “stop USB”, as shown by Figure 3.187. These buttons stop playing the frames. If the execution is paused, the variables “f USB” and “f HS” receive the last played frames. If it is stopped, these variables receive the value -1. In this case, zero will be the first frame of the next execution. Figure 3.193 illustrates this procedure.

Function “Play Camera USB” (Figure 3.191) plays the USB camera frames at a rate of 10fps, as Figure 3.192 shows. The correspondent signal buffer and HS camera frame are

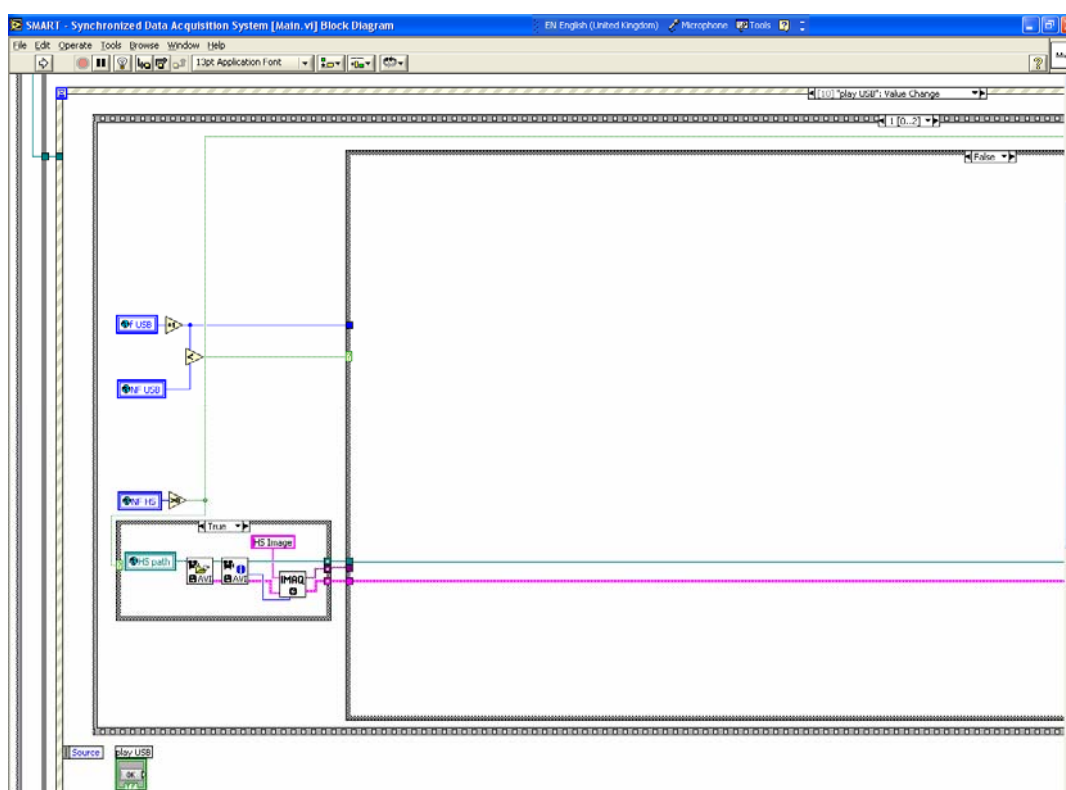


Figure 3.188: Illustration of the code of the event “play USB” that opens the HS camera file

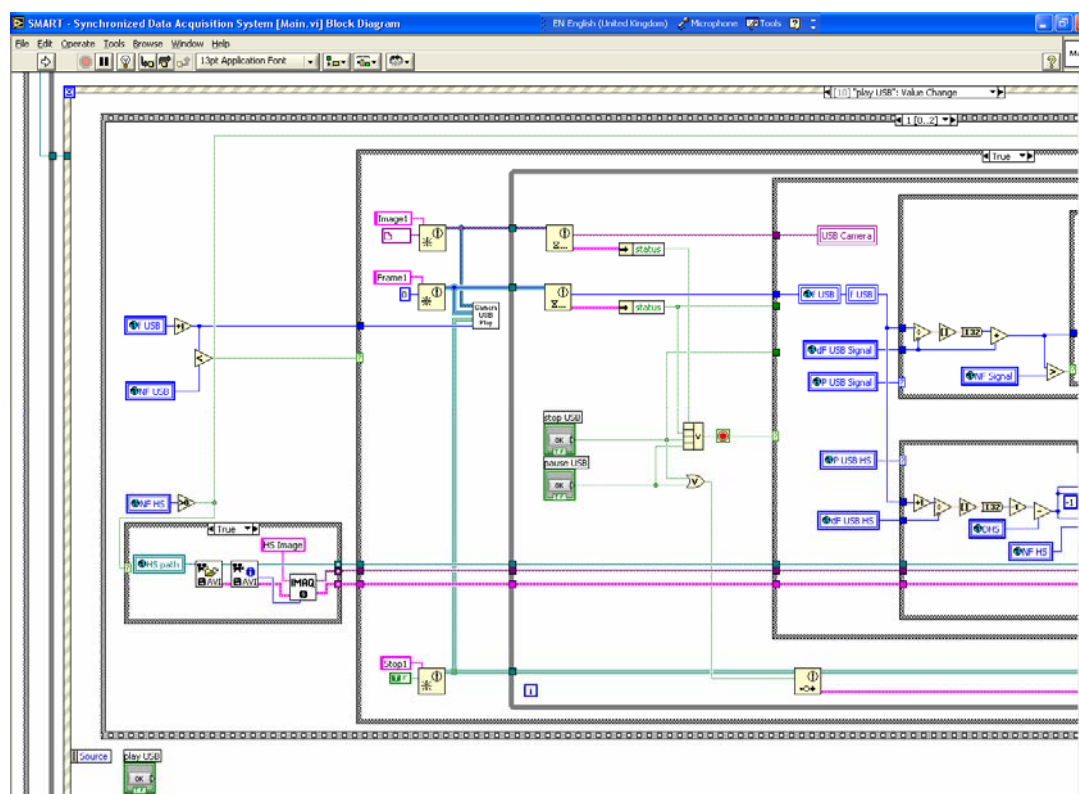


Figure 3.189: Illustration of the code of the event “play USB” that calls the function “Play Camera USB”

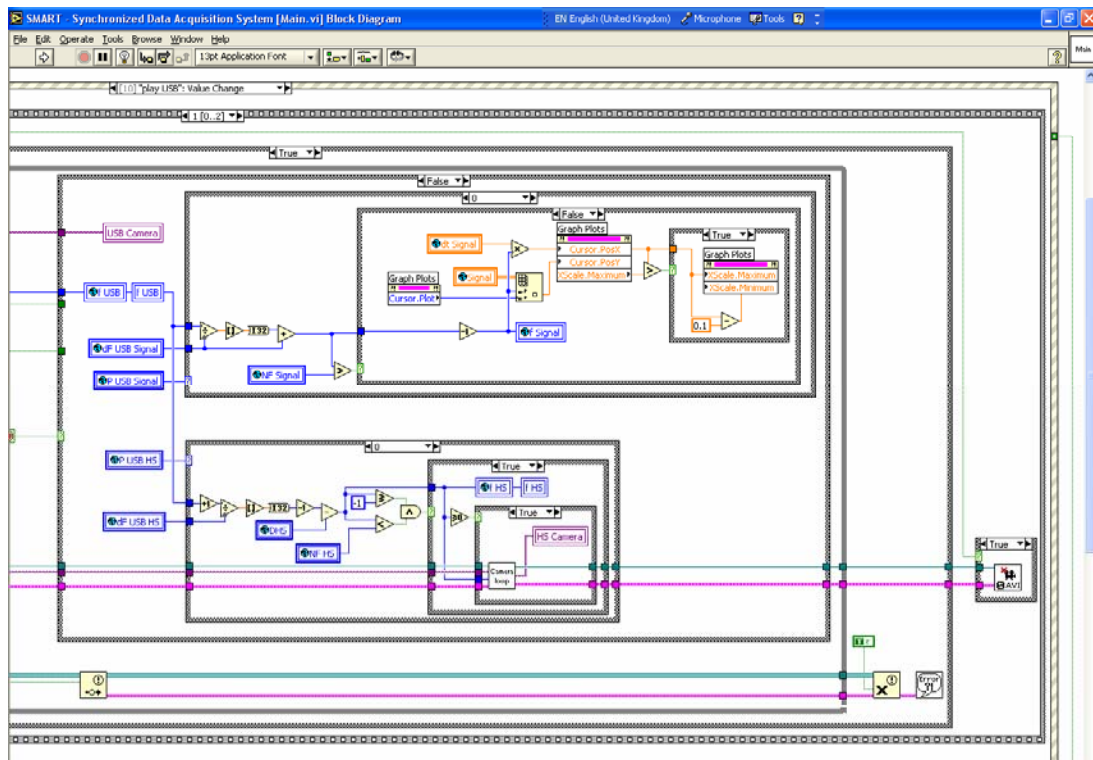


Figure 3.190: Illustration of the code of the event “play USB” that calls the function “Camera loop”

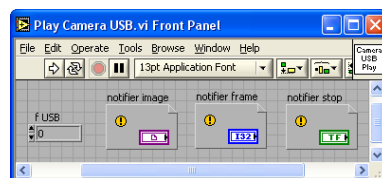


Figure 3.191: Illustration of the front panel of the function "Play Camera USB"

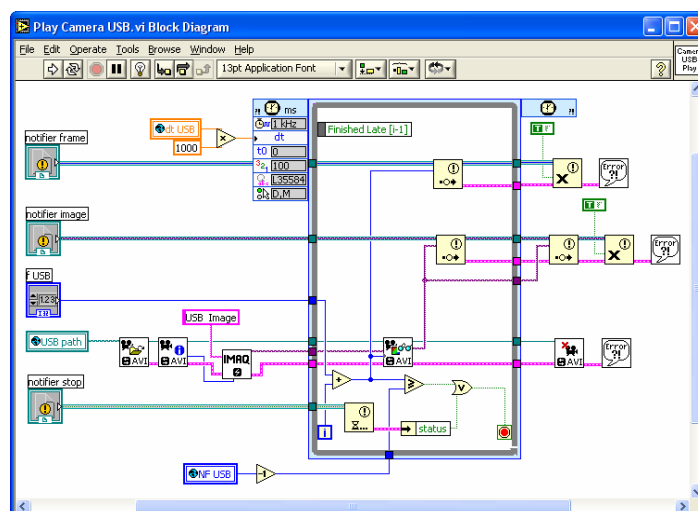


Figure 3.192: Illustration of the code of the function “Play Camera USB” that plays the USB camera frames

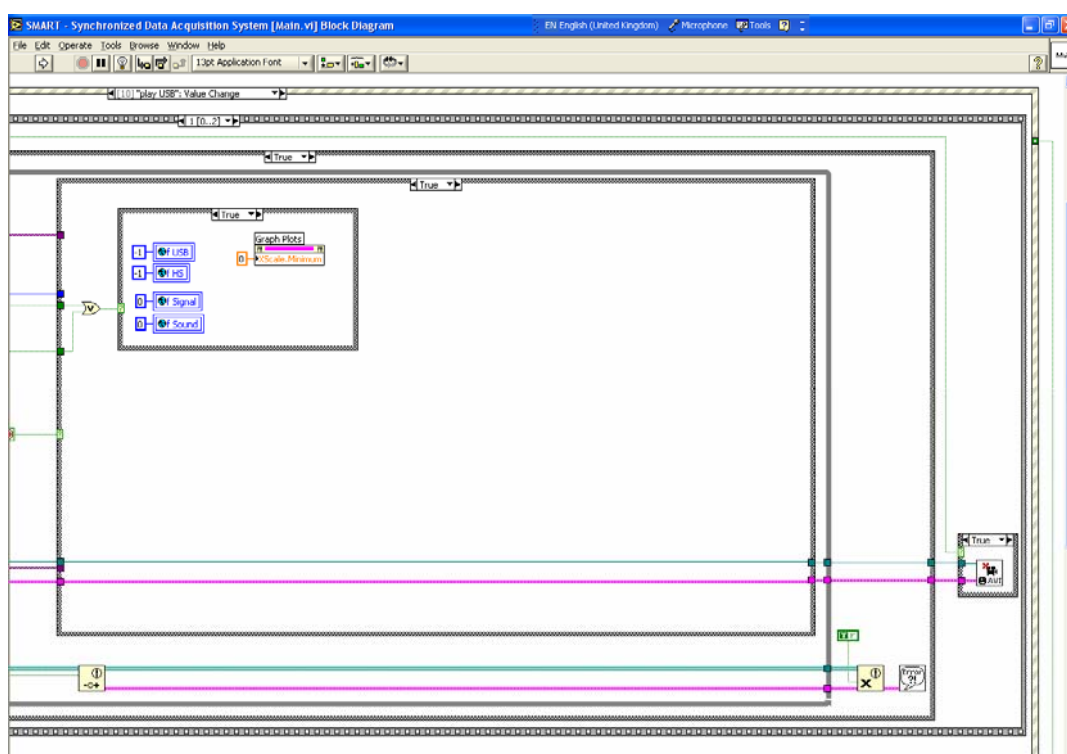


Figure 3.193: Illustration of the code of the event “play USB” that resets the objects on the main window if the acceptable limits are reached

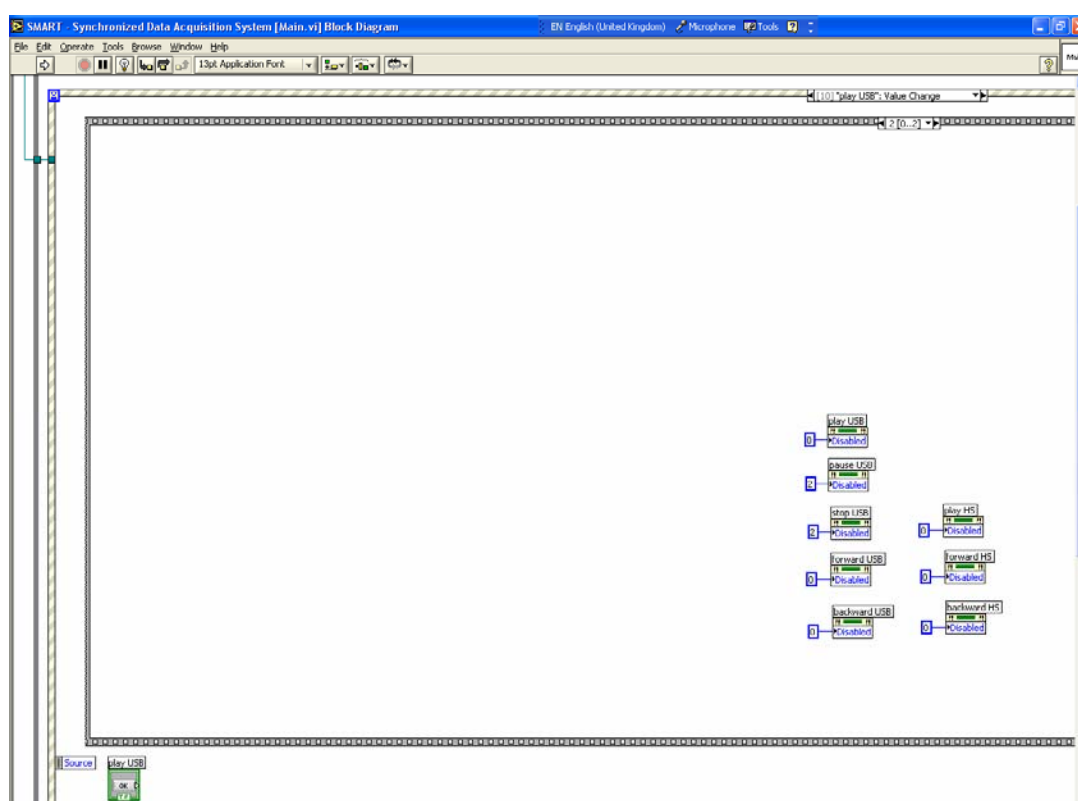


Figure 3.194: Illustration of the code of the event “play USB” that resets the buttons on the main window

Moving the cursor and clicking on the graph, the cameras frames correspondent to the clicked signal value are also shown. The x axis value taken by the cursor is divided by the variables “dt USB” and “dt HS” (they are the inverse of the camera sampling rate) in order to find out the correct frames to be plotted. This procedure is shown by Figure 3.195.

If the window “Main” is closed and an image session had been previously initialized, it is closed as well, as shown by Figure 3.196. All the available images are also disposed, making free the space they occupied in memory.

All the global variables used in the program are shown by Figure 3.197. They are necessary because more than one function needs and modifies those variables.

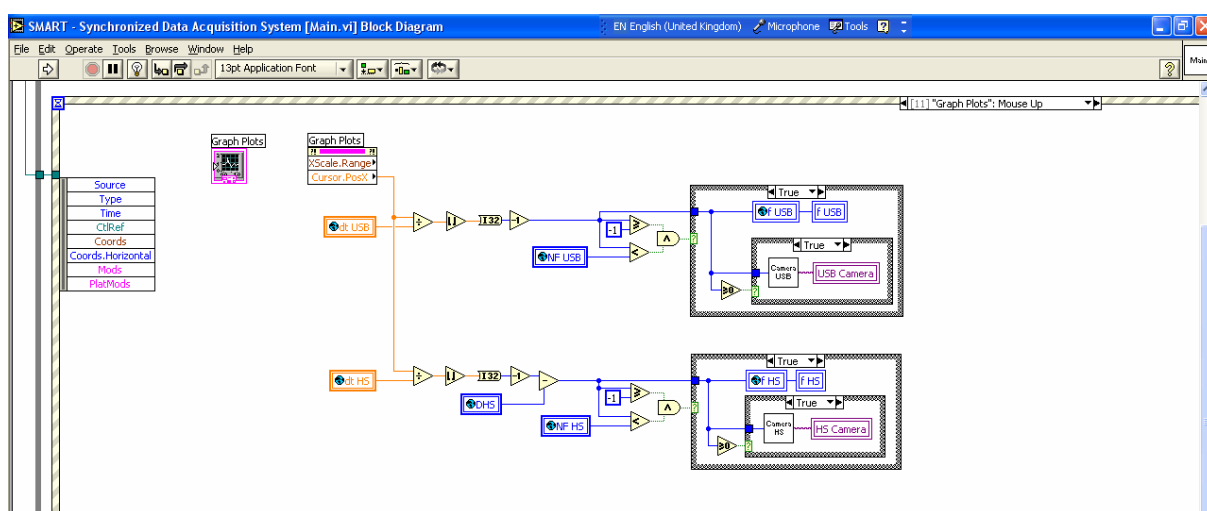


Figure 3.195: Illustration of the code of the event “Graph Plots” that plots the correspondent frames of the cameras on the main window

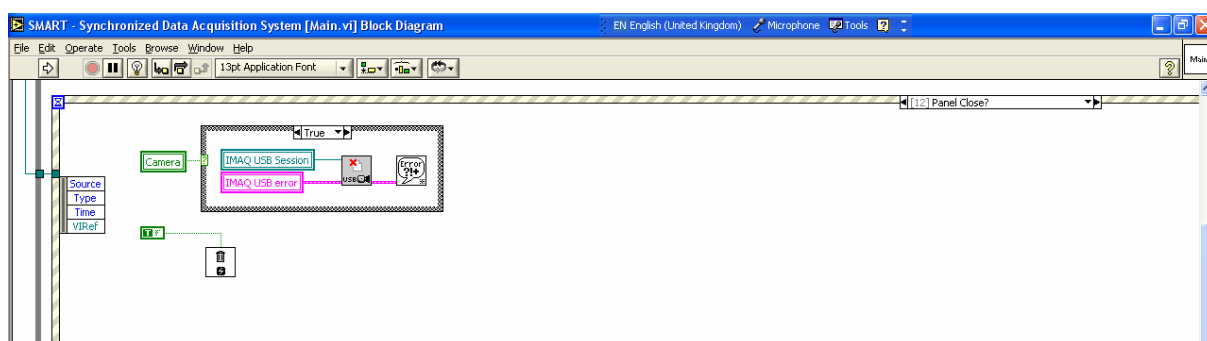


Figure 3.196: Illustration of the code of the event “Panel Close?” that closes an USB camera session if the main window is closed

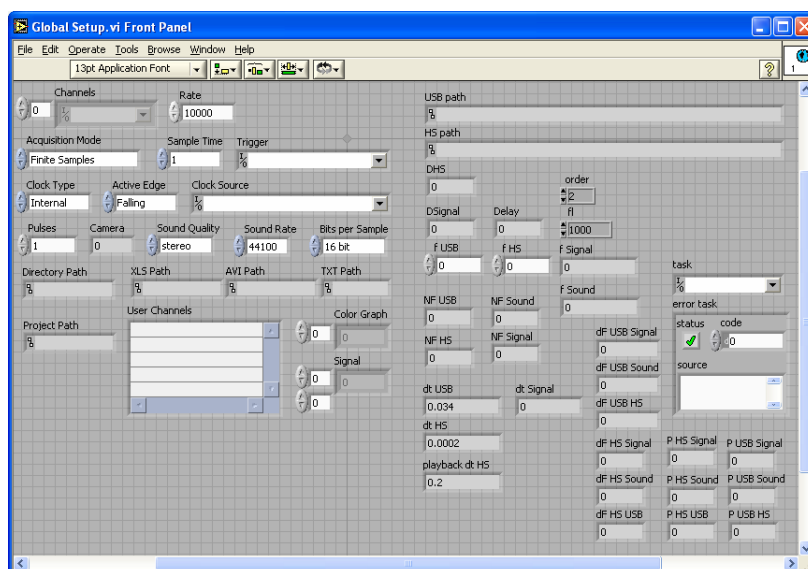


Figure 3.197: Illustration of the front panel that shows all the global variables

Each function created is a thread. It means sequences of code to be executed following priorities, which can be: background (lowest), normal, above normal and time critical (highest). The window “Main” and the functions “Image1”, “Image2”, “Image3”, “Image4”, “Start1” and “Start2” have time critical priority. These functions are responsible for the data acquisition and need to occupy more times the PC processor, in order to provide a deterministic execution. The other functions have normal priority.

LabVIEW also permits to set the “Preferred Execution System”, which can be: user interface, standard, instrument I/O, data acquisition, other1, other2, same as caller. This tool helps to organize the software structure. All the created windows are set as “user interface”. The functions previously cited are set as “data acquisition”. All the other ones are set as “same as caller”.

Figure 3.198 shows how these parameters can be set by the windows “VI Properties” of LabVIEW.

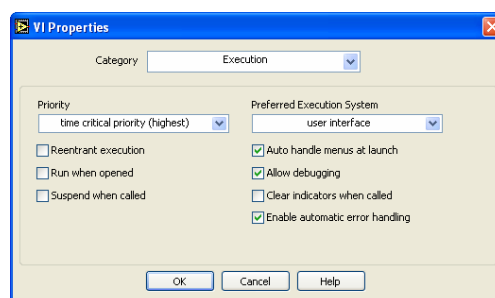


Figure 3.198: Illustration of the window “VI Properties” that sets the priority and type of execution of the functions created in the software

The diagram shown by Figure 3.199 summarizes the software. It is divided into 3 parts: Menu, Acquisition and Analysis and contains the created functions and their hierarchy. The full lines mean that the following function belongs to the previous one. And the traced lines mean that the following function does not belong to the previous one, but it is executed in that order. Figure 3.200 presents the software's data flow.

After doing the code, it was necessary to build a stand alone application (the executable version). LabVIEW offers this option and permits to create an installer as well. For this purpose, it is necessary to add the files "ImaqDirectShowDLL.dll" and "NIVisSvc.dll" which can be found into the folder C:\Windows\system32.

Before installing the created application, it is necessary to install the National Instruments Vision Assistant 7.1 that has a separated license and contains the video functions used by the application. Besides that, it is also necessary to install the NI-IMAQ for USB Cameras (can be downloaded from www.ni.com for free), which contains the USB functions used for acquiring frames of USB cameras.

This final version was obtained after many trial versions. In order to have synchronism among signals and images, an ideal situation would be to provide an instantaneous trigger for the high-speed camera, although, this is not possible by software. Thus, a suitable solution was to provide a deterministic delay when the trigger is produced. Then, it would be possible to calculate the correct high-speed camera frame.

A real-time version was considered. For this purpose, it was installed the LabVIEW Real Time 7.1 (RTX). This version of LabVIEW runs the applications using a real-time extension for Windows XP. This extension is another operating system, which means another kernel, it is called RTX and it is provided by Venturcom. An academic license of LabView does not provide RTX, so it has to be purchased apart and it is quite expensive. The found solution was to contact Venturcom and ask a trial version of RTX 5.5 (later versions are not acceptable for LabVIEW Real Time 7.1). It is very important to uninstall service pack 2 before installing RTX 5.5, otherwise nothing works. It is also necessary to install DAQmx Base 1.5.1 (can be downloaded from www.ni.com). This is the package used for programming data acquisition in a real-time environment. The following steps have to be executed in order to make LabVIEW Real-Time (RTX) works:

- Install RTX 5.5 (if the LabView license does not include it)
- Install LabView 7.1 (if it was not previously installed)
- Install LabView 7.1 Real Time (RTX)
- Install DAQmx Base 1.5.1 (it contains the functions for data acquisition in Real-Time)

- Update device driver (the driver of the acquisition board driver has to be changed to RTX supported)

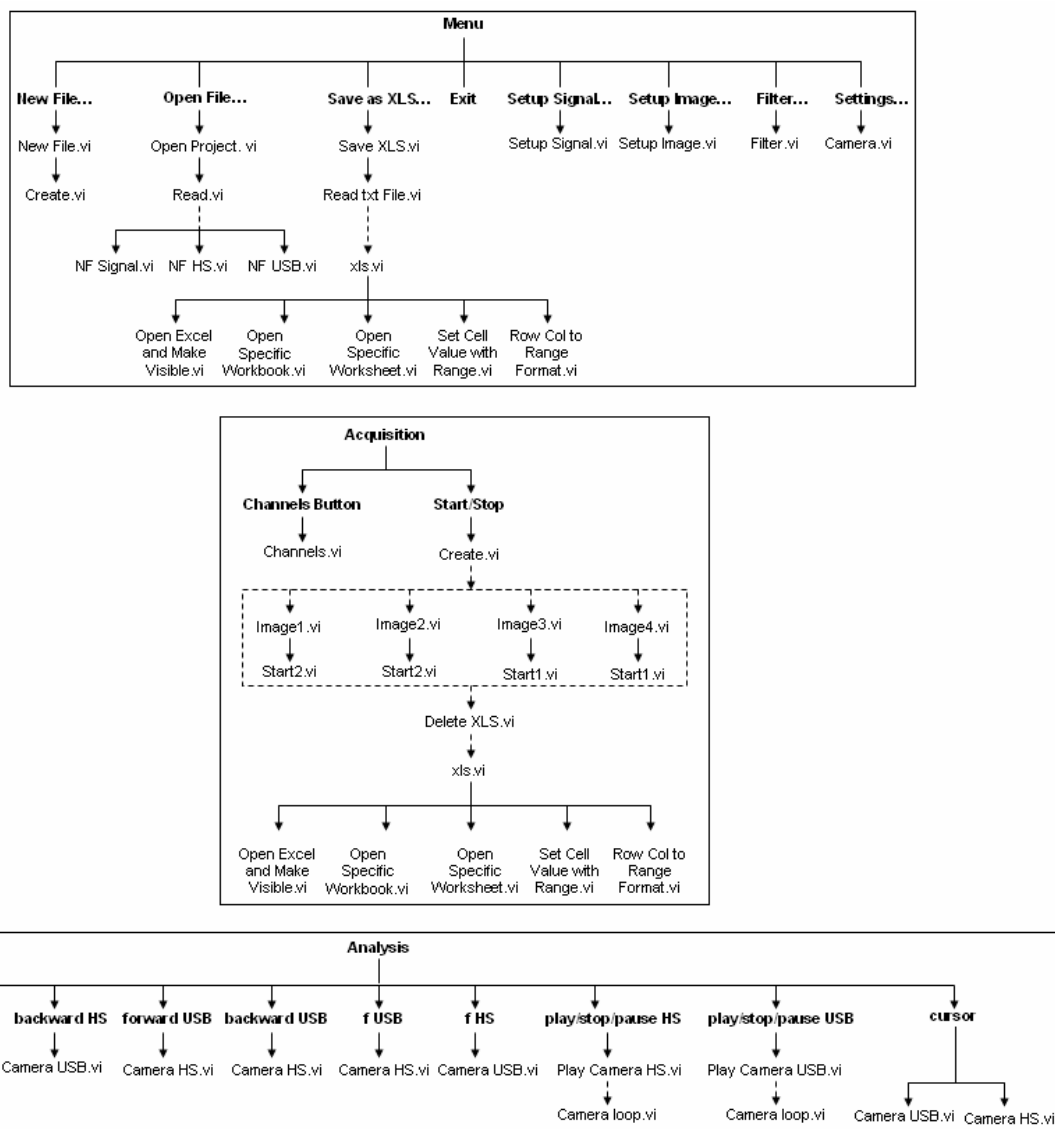
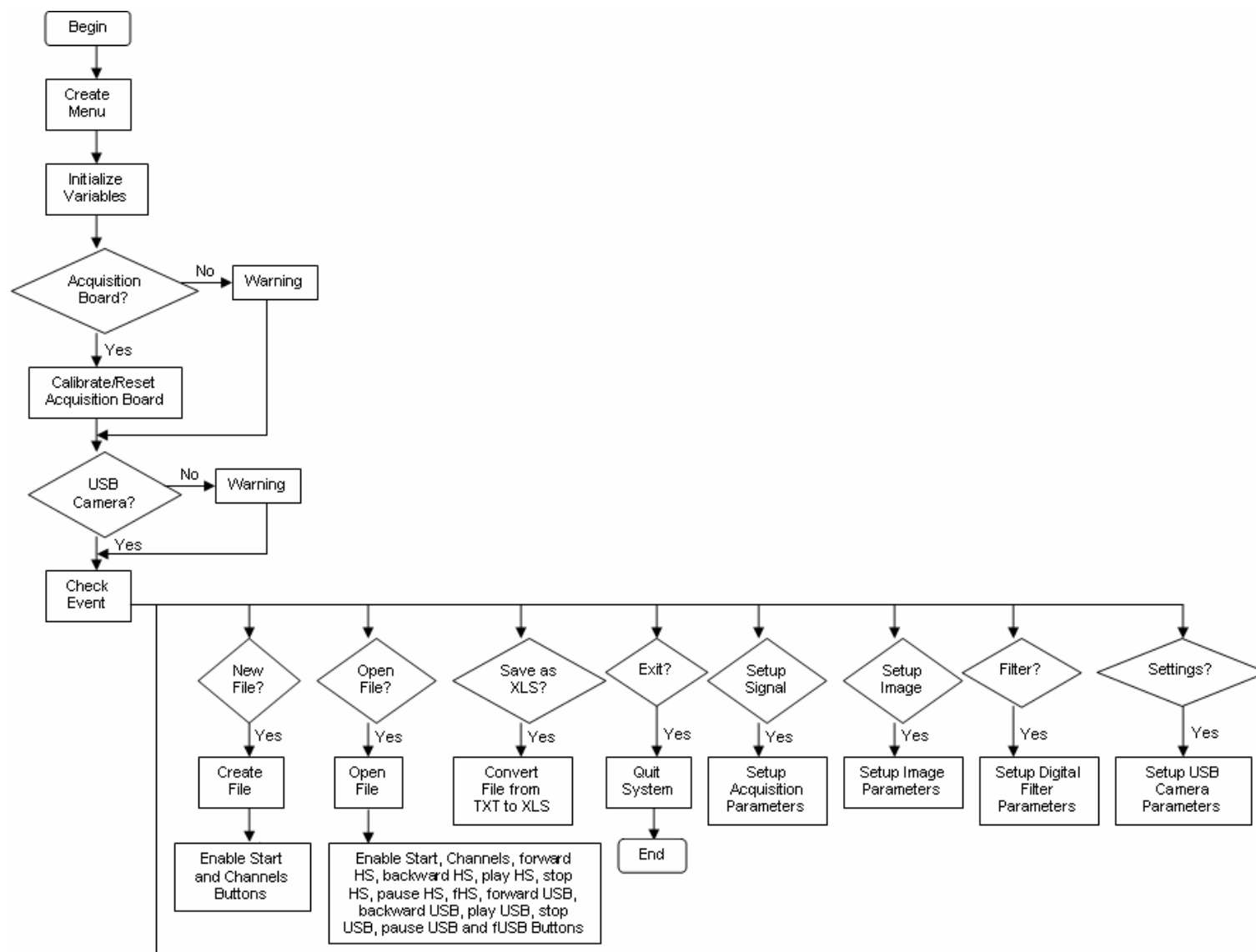
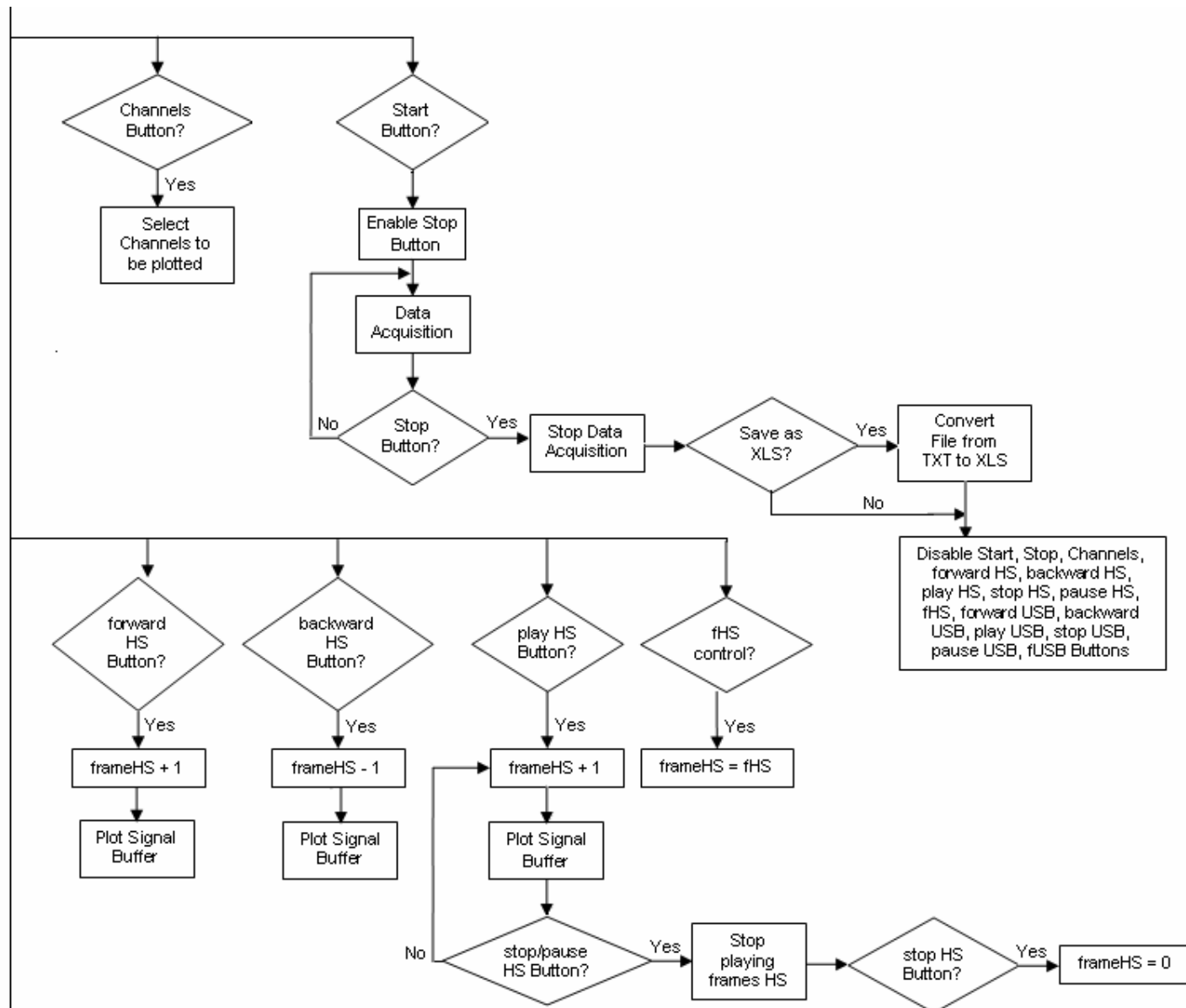


Figure 3.199: Illustration of a diagram that summarizes how the software works





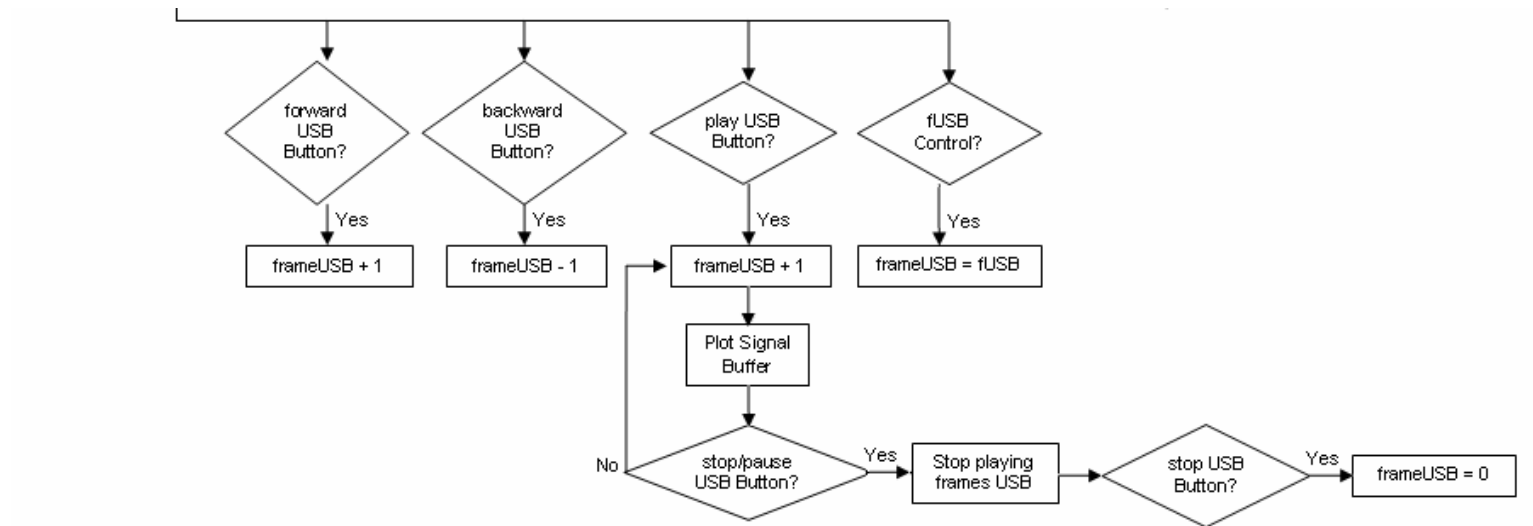


Figure 3.200: Illustration of the software's data flow

The problems in using this configuration are many. First of all, the DAQmx Base does not have all the functions of the DAQmx and the entire application has to run by the same kernel, what makes impossible to maintain all the previous code resources. There are no available drivers for PCMCIA acquisition boards, which means that a laptop could not be used for doing the acquisition. Finally, there are no available drivers for USB cameras, so the webcam could not be integrated to the system. Those are the main reasons to not use the real-time version of LabVIEW.

Another possibility was studied, the LabVIEW Real-Time (ETS). This version executes the application on hardware targets running the real-time operating system of Venturcom Phar Lap Embedded Tool Suite (ETS). It provides a real-time operating system that runs on NI RT Series hardware to meet the requirements of embedded applications that need to behave deterministically or have extended reliability requirements. This solution was not suitable because one of the objectives of this work was to build a flexible application to run on laptops, not an embedded one.

3.3 Calibration

Calibration is necessary in order to achieve higher measurement accuracy when the work environment changes. There are two types of calibration, device calibration and channel calibration. Device calibration consists of verifying the measurement accuracy of a device and adjusting for any measurement error. It means to measure the performance of the device and compare these measurements to the published specifications. During calibration, voltage levels or other signals are supplied and read using external standards, and then the device calibration constants are adjusted. The new calibration constants are stored in the memory (EEPROM). These calibration constants are loaded from memory as needed to adjust for the error in the measurements taken by the device.

One type of device calibration is the external calibration, which is typically performed by a metrology lab, which uses a high-precision voltage source to verify and adjust calibration constants. This procedure replaces all calibration constants in the EEPROM and is equivalent to a factory calibration. Because the external calibration procedure changes all EEPROM constants, it invalidates the original calibration certificate. If an external calibration is done with a National Institute of Standards and Technology (NIST) certified voltage source, a new NIST traceability certificate can be issued. It should be done at a regular interval as defined by the measurement accuracy requirements of the application. National

Instruments recommends a complete calibration at least once every year. This interval can shorten to 90 days or six months.

Another type of device calibration is the self-calibration which does not require any external connections. It adjusts the calibration constants with respect to an onboard reference stored on the device. The new calibration constants are defined with respect to the calibration constants created during the external calibration to ensure that the measurements are traceable to these external standards. The new calibration constants do not affect the constants created during an external calibration because they are stored in a different area of the device memory. A self-calibration can be performed at any time to adjust the device for use in environments other than those in which the device was externally calibrated.

Channel calibration is necessary in applications where the highest degree of accuracy is critical, but it does not replace device calibration. Channel calibration compensates for various errors, including those introduced by cabling, wiring, and sensors.

SMART does a self-calibration every time the system is initialized. The other types of calibration have to be done if a high accuracy of the measurements is necessary. For welding parameters measurement, the self-calibration is usually enough.

3.4 System configuration

The first step to configure the system is to create virtual channels through the Measurement & Automation Explorer (MAX), which can be downloaded from www.ni.com, for free. This application also contains the drivers for the NI acquisition boards. Thus, it is advisable to download its last version, which is supposed to be more complete. Figure 3.201 illustrates how to create the virtual channels.

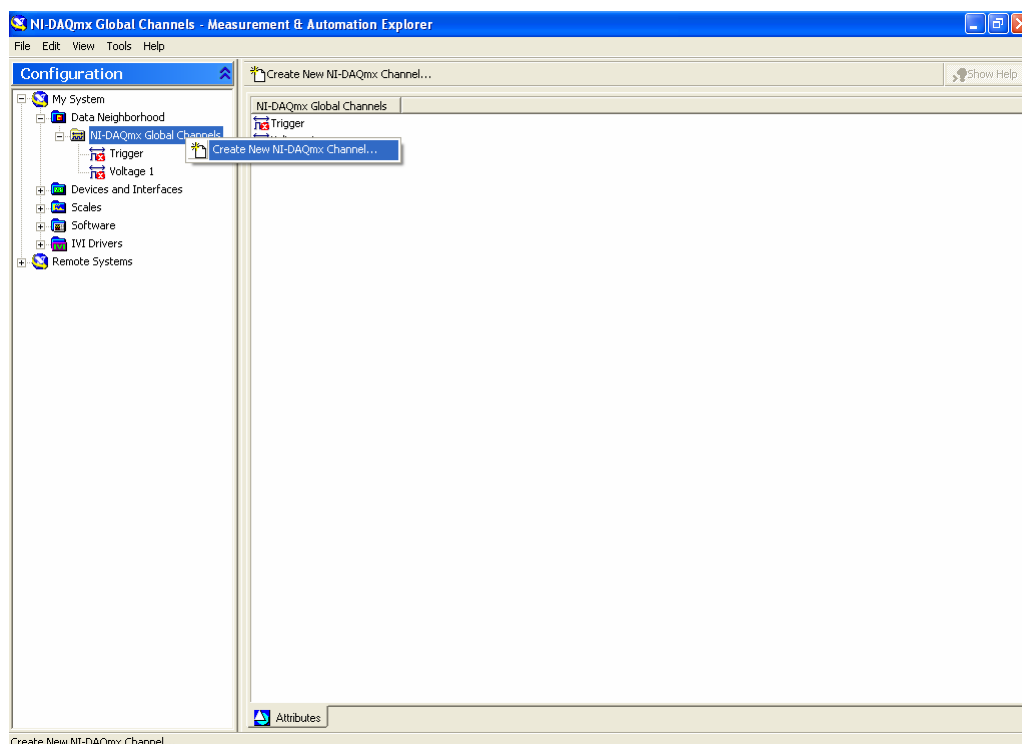


Figure 3.201: Illustration of the panel “Measurement and Automation Explorer” that creates a Virtual Channel

The measurement type has to be selected as Figure 3.202 shows. Voltage and current signals are both analog inputs. Figure 3.203 shows how the standard configuration for voltage signals was done. Since the output of the current sensor is a voltage signal, current signals have to be configured as voltage inputs. In this case, a scale according to the specifications of the sensor has to be build. Figure 3.204 and Figure 3.205 show how to create a scale. Figure 3.206 shows how the standard configuration for current signals was done.

The configuration of the voltage and current terminals are non-referenced single-ended (NRSE) because the measurement of these signals is made with respect to the same node, but the potential at this node can vary with respect to the measurement system ground (since different types of sensors are used). This type of configuration makes possible to use a greater number of inputs.

In order to measure temperature, virtual channels were configured as thermocouples of R/S type, as Figure 3.207 illustrates. In this case, the configuration of the terminals is differential because the thermocouple outputs are differential as well.

A virtual channel for the external trigger is actually a pulse output, as Figure 3.208 shows. Figure 3.209 illustrates its standard configuration.

When the software runs for the first time, it takes a while to calibrate the system and reset the acquisition board. After that, the next step is to configure the software. First, it is necessary to create a new file or to open an existent one, as Figure 3.210 illustrates. After doing this, the name of the current file appears in the box “Project”.

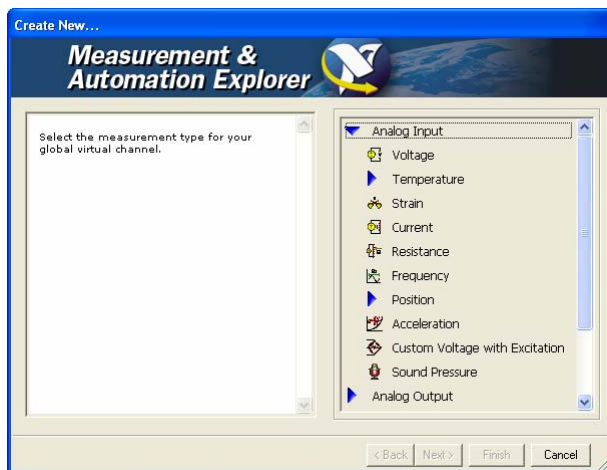


Figure 3.202: Illustration of the panel “Measurement and Automation Explorer” that chooses the measurement type for voltage and current signals

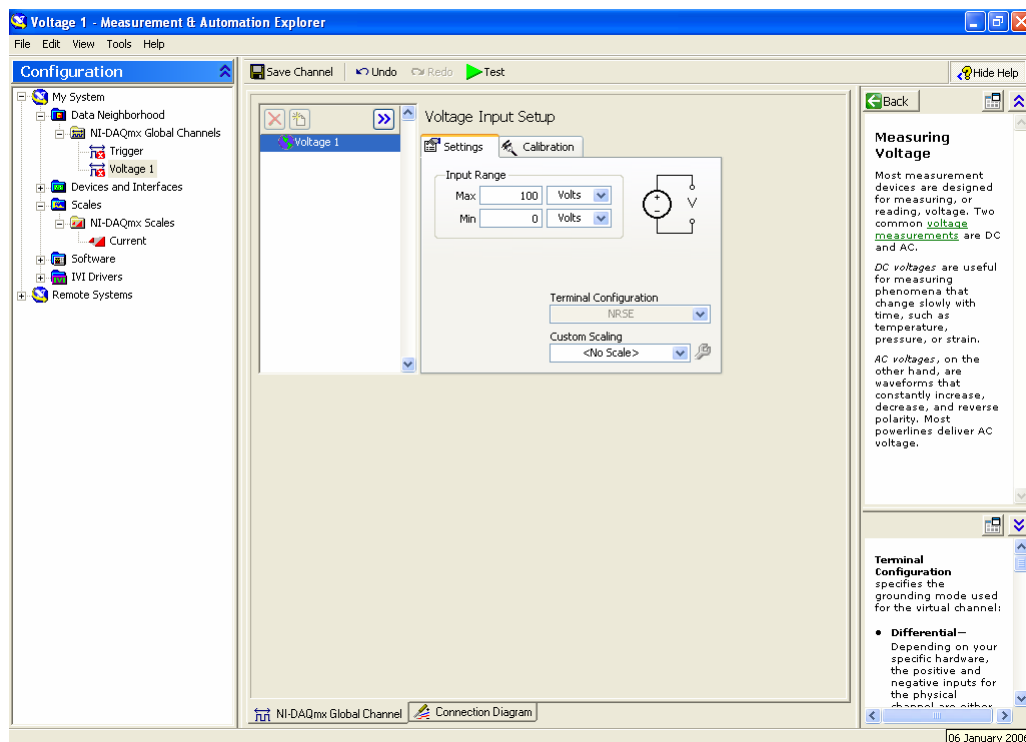


Figure 3.203: Illustration of the panel “Measurement and Automation Explorer” that configures the voltage signals

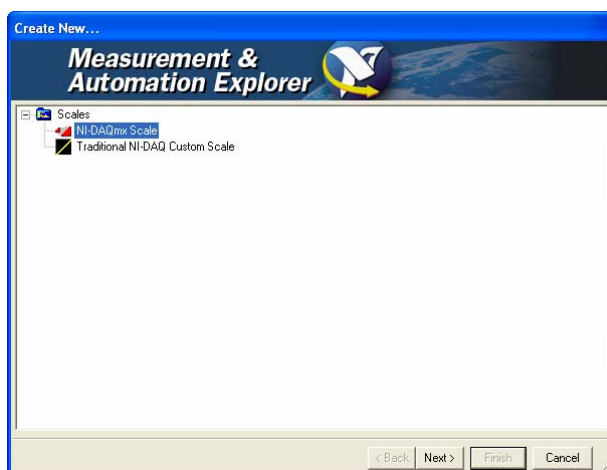


Figure 3.204: Illustration of the panel “Measurement and Automation Explorer” that creates a NI-DAQmx Scale

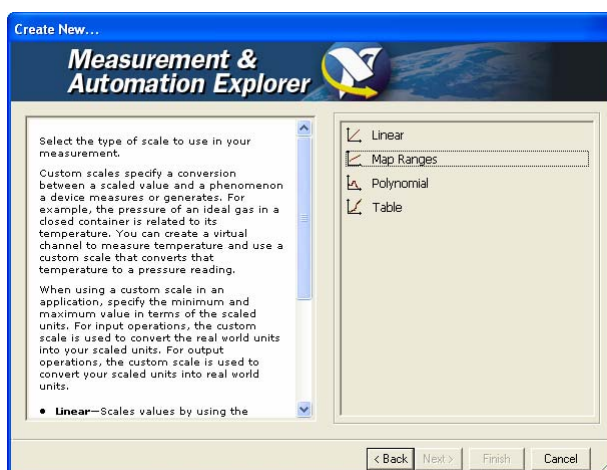


Figure 3.205: Illustration of the panel “Measurement and Automation Explorer” that selects a type of scale

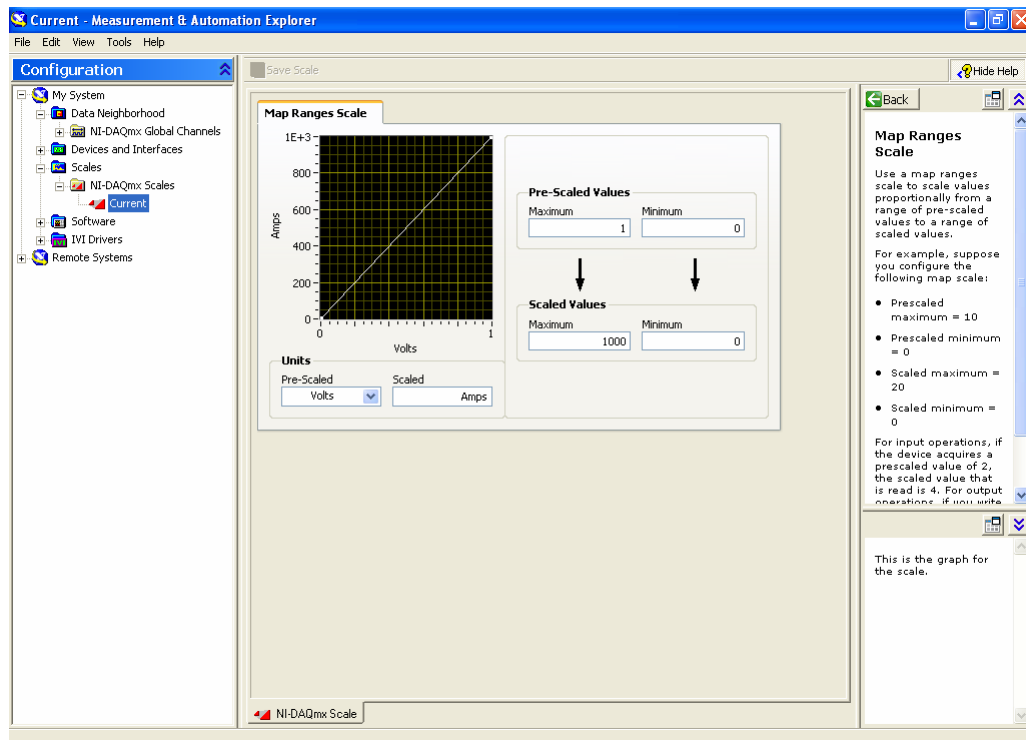


Figure 3.206: Illustration of the panel “Measurement and Automation Explorer” that configures a scale to be used for current signals

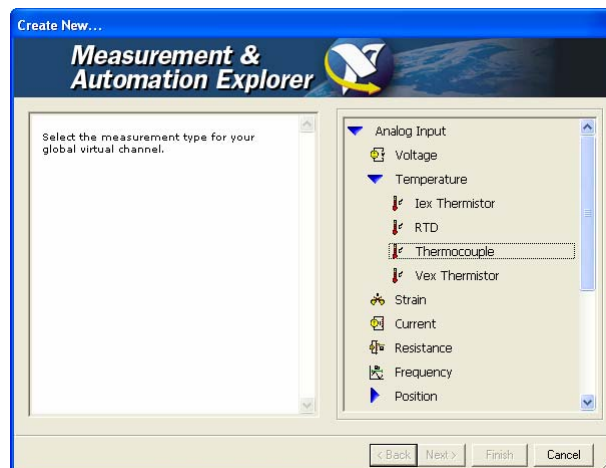


Figure 3.207: Illustration of the panel “Measurement and Automation Explorer” that configures the measurement type of the temperature signals

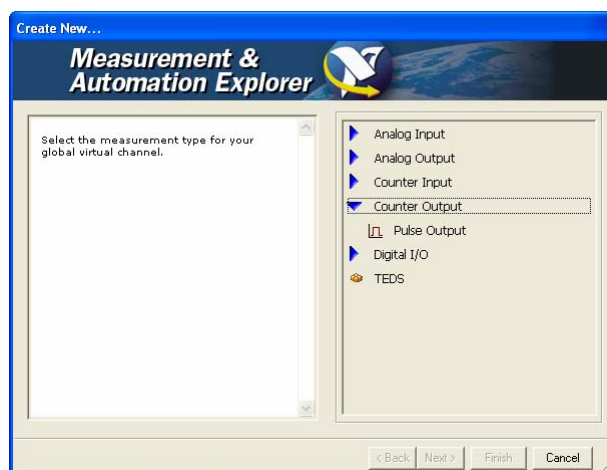


Figure 3.208: Illustration of the panel “Measurement and Automation Explorer” that configures the measurement type of the external trigger

If after opening an existent file a new one is created, the previous parameters are kept, otherwise the default ones last. It is possible to change signal, image and filter parameters, as Figure 3.211 shows. In order to change the signal parameters, the desired channels have to be selected among the created virtual channels. The sampling rate per channels has also to be defined (it is recommended 20kHz or above, since there is a low-pass filter with cut-off frequency equal to 10kHz attached to each input connector). If the acquisition mode is Finite Samples, the Sample Time has to be set (the minimum value is 1 second and the maximum value is 60 seconds). If the acquisition mode is Continuous, it is advisable to worry about the space available in the HD of the computer, what limits the acquisition time.