

**Report describing the open software tool TPQA developed in LabWindows/CVI environment**

|  |
| --- |
|  |

# A2.4.5 - TPQA structure

This report also covers the following activities:

A2.1.1 – Flow chart of TPQA tool

A2.1.2 – Extension for a multiple digitizers

A2.1.4 – Concept of the LV to Octave/Matlab interface

A2.2.2 – Integration of the drivers to the virtual driver

A2.4.2 – TPQA tool structure

A2.4.3 – Acquisition and control module description

A2.4.4 – Processing module description

A3.3.3 – Guidance on integration of new HW

**CONTENTS**

[A2.4.5 - TPQA structure 1](#_Toc5810100)

[1.1 References 3](#_Toc5810101)

[1.2 Overview 4](#_Toc5810102)

[1.3 TPQA Flow Chart 5](#_Toc5810103)

[1.4 TPQA structure in LabWindows/CVI Environment 6](#_Toc5810104)

[1.5 Control and data acquisition module 7](#_Toc5810105)

[1.5.1 Control module 8](#_Toc5810106)

[1.5.2 Acquisition module 9](#_Toc5810107)

[1.5.2.1 Function prototype for LF DMMs 10](#_Toc5810108)

[1.5.2.2 Acquisition module for wideband digitizers 15](#_Toc5810109)

[1.5.2.3 Modular driver design 15](#_Toc5810110)

[1.5.2.4 Virtual driver functions structure 16](#_Toc5810111)

[1.5.2.4.1 ADC session 17](#_Toc5810112)

[1.5.2.5 Virtual driver function reference manual 17](#_Toc5810113)

[1.5.2.5.1 Sessions Configuration 18](#_Toc5810114)

[1.5.2.5.2 Trigger Configuration 19](#_Toc5810115)

[1.5.2.5.3 Synchronization session 20](#_Toc5810116)

[1.5.2.5.4 Initiate sampling 20](#_Toc5810117)

[**1.5.2.5.5** Fetch and store matrix sampled data 21](#_Toc5810118)

[1.5.2.5.6 Function’s list 22](#_Toc5810119)

[1.5.2.5.7 Abort Digitizing Process 23](#_Toc5810120)

[1.5.2.5.8 Clean-up sessions 23](#_Toc5810121)

[1.6 Processing module 23](#_Toc5810122)

[1.6.1 Processing module – LabWindows/CVI environment 23](#_Toc5810123)

[1.6.1.1 TPQA Processing configuration GUI 23](#_Toc5810124)

[1.6.1.2 TPQA post- processing GUI 23](#_Toc5810125)

[1.6.2 Processing module – Matlab component 23](#_Toc5810126)

## References

1. TPQA tool, url: <https://github.com/btrinchera/TPQA>
2. TracePQM, url: <https://www.euramet.org/research-innovation/search-research-projects/details/?page%5BeurametCtcp_project_listResearch%5D=2&eurametCtcp_project_show%5Bproject%5D=1407&eurametCtcp_project%5Bback%5D=450&cHash=69ff67e07cffde667e34af3a7ef39df3>
3. TWM tool, url: <https://github.com/smaslan/TWM>
4. INFO-STRINGS, url: <https://github.com/KaeroDot/info-strings>
5. QWTB toolbox, url: <https://qwtb.github.io/qwtb/>
6. GOLPI interface, url: <https://github.com/KaeroDot/GOLPI>
7. A232 Algorithms exchange format, url:

[https://github.com/smaslan/TWM/tree/master/doc/A232 Algorithm Exchange Format.docx](https://github.com/smaslan/TWM/tree/master/doc/A232%20Algorithm%20Exchange%20Format.docx)

1. A231 Correction Files Reference Manual, url:

[https://github.com/smaslan/TWM/tree/master/doc/A231 Correction Files Reference Manual.docx](https://github.com/smaslan/TWM/tree/master/doc/A231%20Correction%20Files%20Reference%20Manual.docx)

1. A231 Data Exchange Format, url:

[https://github.com/smaslan/TWM/tree/master/doc/A231 Data exchange format and file formats.docx](https://github.com/smaslan/TWM/tree/master/doc/A231%20Data%20exchange%20format%20and%20file%20formats.docx)

1. A331 Installation and Guide\_TPQA, url:

<https://github.com/btrinchera/TPQA/blob/master/doc/A331%20Installation%20and%20Guide_TPQA.docx>

## Overview

TPQA [1] is an open source project that is being developed in scope of EMPIR project TracePQM [2] using the Labwindows/CVI environment. Together with TWM [3] open source project, developed in LabVIEW environment, furnish an open platform to help unexperienced NMIs to speed up in the developing of state-of-the-art standards suitable to perform traceable measurements of electric power and power quality parameters using the concept of waveform digitizing. It is not restricted to power and PQ area but it allows recording and processing of pure and complex voltage and current waveforms.

The TPQA is organized according to the flow chart diagram shown in Figure 0‑1. The whole TPQA application consists of two parts:

1. LabWindows modules (Control and Processing) that controls the instruments, initiates processing and serves as a user interface implemented into a suitable User Interface Guide (GUI).
2. Calculation or Processing module based on the Matlab which performs the processing of post-processing and formatting the data for displaying and generation of the measurement report (summary of the results formatted in compact form), as well as a second processing module based on CVI algorithms for quasi real-time data processing. Note that CVI algorithms are not validated yet and serve only to establish a first processing approach on sampled data.

Further characteristic of the post-processing module can be found in [4] and [7].

The control module consists in several separated processes. Main functions are inserted in TPQA.C source file and declared in TPQA.h file.

## TPQA Flow Chart

The flow chart of TPQA is shown in Figure 0‑1. The meaning of main blocks is reported below.

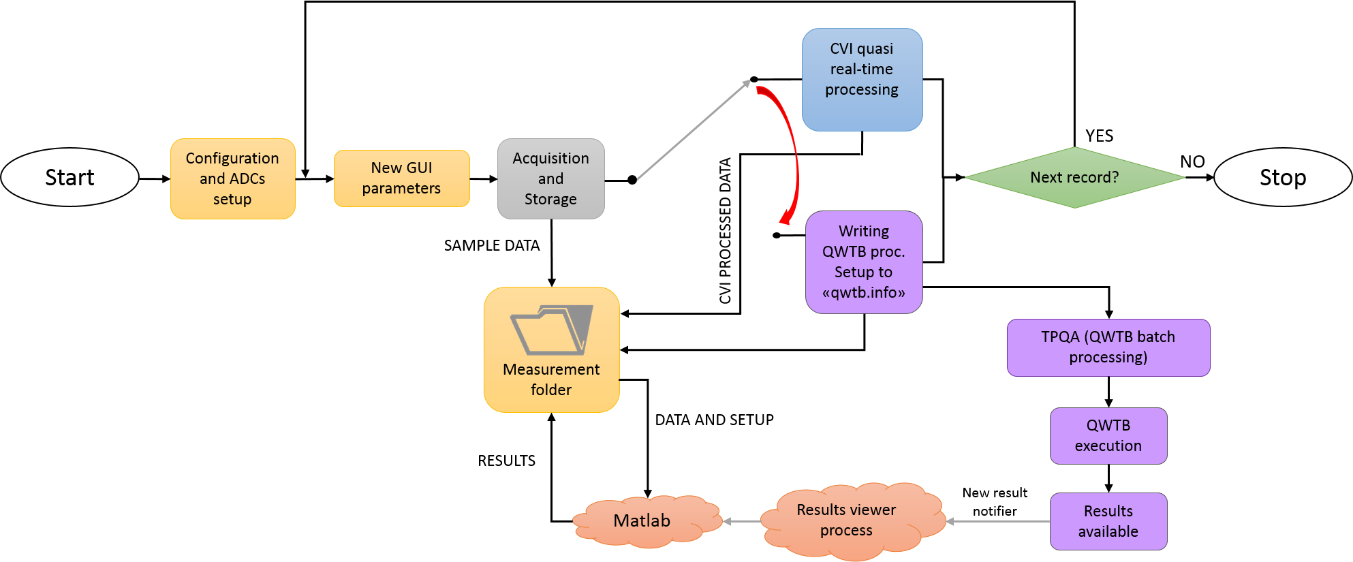


Figure 0.1: TPQA flow chart structure. The coloured frames are used to distinguish the process in which the tasks run.

* **Configuration and ADCs setup**: main process for the configuration of digitizers using specific drivers:
* NI PXI 5922 digitizers using niScope and niTLCK drivers provided by National Instruments. All drivers must be installed on the PC.
* DMMs HP 3458 configured as sampling multimeters (NI-GPIB driver must be installed).
* **New GUI Parameters:** allows to set new digitizing parameters during the sampling process.
* **Acquisition and storage:** enables the possibility of storing sampled and/or processed data through a mechanism based on multiple flags.
* **CVI quasi real time-processing:** CVI algorithms for data processing during the acquisition process.
* **Writing QWTB proc.:** creates measurement process, based on TWM concept of post-processing data [1], and does following: loads correction files; builds measurement sequence; stores acquired data and generates suitable files for further processing.
* **TPQA (QWTB batch processing):** enables GUI panel for post-processing of data and does following: loads selected algorithm’s configuration from QWTB alg. database file.
* **QWTB execution:** when requested by user, initiates processing of all acquired data stored in the folder.
* **Results available and Results viewer process:** initiates refresh of the results view according to the current setup and algorithm selection.

## TPQA structure in LabWindows/CVI Environment

There are two possibilities to test the functionality of TPQA open source project:

1. Developer mode: first the user launches the LabWindows/CVI environment and then opens the file TPQA.CWS. The user can run the project and interact with it through the user interface guide that appears. This mode is useful for expert users, which intend to modify or add new functionality or routines into the project.
2. User mode: After compiling the entire project the executable file TPQA\_32bit.exe will be generated. The user could compile the project also for 64-bit distribution. In this case the environment must be switched to 64 bit distribution and libgen.lib (64-bit ) and libmx64.lib (64-bit) matlab \*.dll files must be inserted into the Libraries folder of the project.

Note that, since the project includes a communication with the Matlab engine the necessary \*.dlls cannot be freely distributed. However the user must install Matlab distribution on local PC and must check careful the path of octprog folder within the config.ini file. By default the path is: twm\_octave\_folder = C:\TPQA\TPQA\_1.1.0\octprog.

The internal hierarchy structure of the TPQA in NI-LabWindows\CVI environment is shown in Figure 0.2.

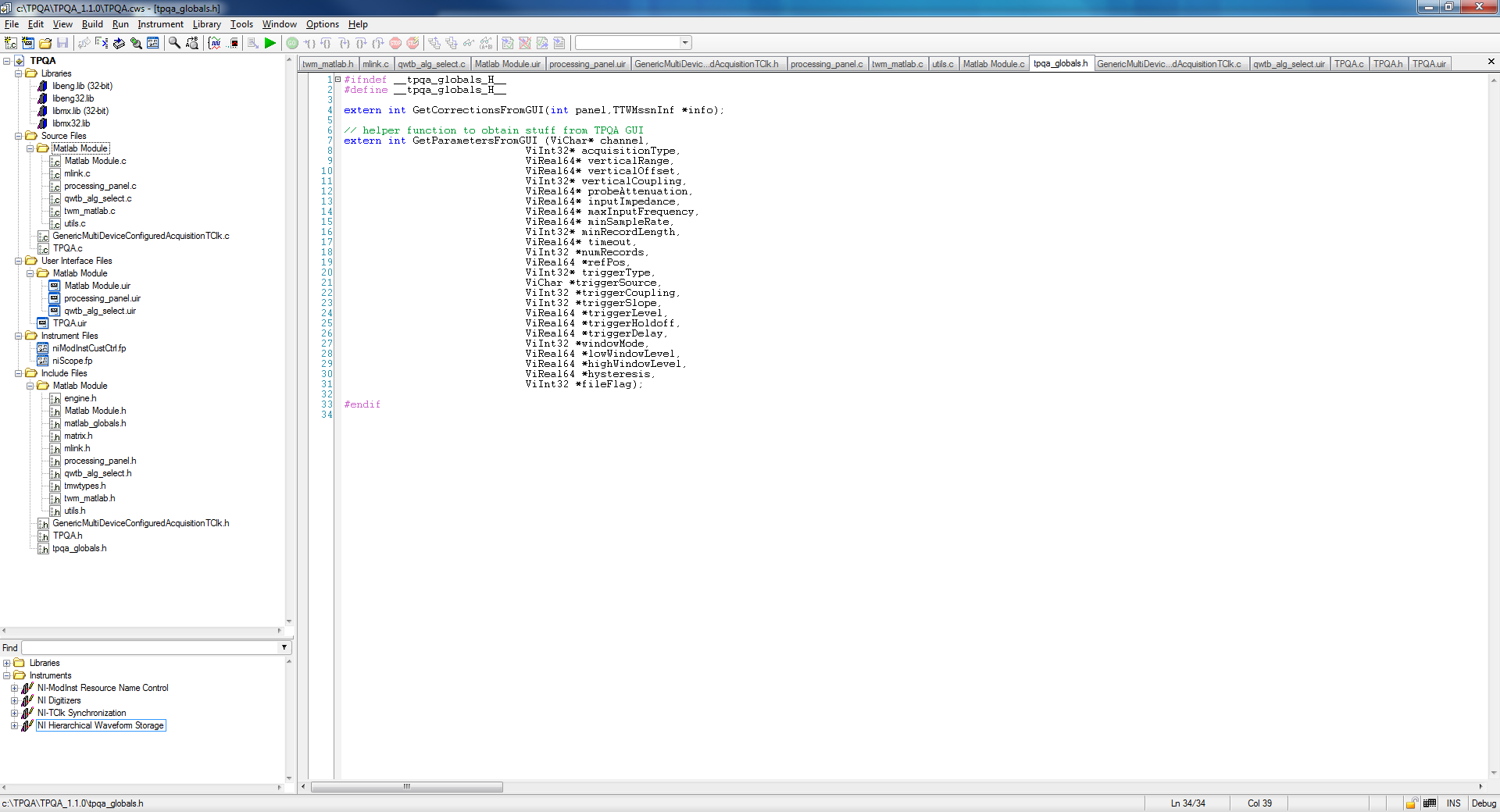


Figure 0.2: Internal TPQA structure shown in Labwindows/CVI.

The project files are arranged as follows:

1. Source Files, containing:

* TPQA.c contains the main and auxiliary functions which allow the various parts of the software to communicate together. Within the same function it was inserted also the communication routine for DMMs LF digitizing multimeters.
* GenericMultiDevice……. .c contains specific sub-routines to communicate with single or multiple NI-5922 digitizers based on the use of NI-Digitizers and NI-TCLK synchronization driver. That is a single file that implement everything we need to communicate with wideband digitizers.
* Matlab Module folder, (developed by CMI) composed by several additional files suitable to establish a communication protocol with the Matlab engine for data processing using QWTB toolbox: Matlab Module.c, Mlink.c, qwtb\_alg\_select.c, Processing\_panel.c, Twm\_matlab.c and Utils.c

1. User Interface Files: containing all the GUI developed to communicate with the macro setups.

* TPQA.uir which contains the main panel and additional GUIs for control and data acquisition modules.
* Matlab Module Folder, (developed by CMI) composed by several GUI, as: Matlab Module.uir, processing\_panel.uir and qwtb\_alg\_select.uir.

1. Instrument Files, containing:

* niScope.fp to communicate with wideband NI 5922 digitizers. For further information about the installation instructions and features please NI-SCOPE Readme guide provided by NI.
* niModInstCustCtrl.fp to search for already connected instruments.

1. Include Files: (\*.h files contains the prototype of \*.c functions and variables definition as well as CVI callback function used by GUIs)

* TPQA.h
* GenericMultiDeviceConfiguredAcquisitionTClk.h
* Tpqa\_globals.h
* Matlab Mudule directory, which containd the following files: engine.h, Matlab Module.h, matlab\_globals.h, matrix.h, mlink.h, processing\_panel.h, qwtb\_alg\_select.h, tmwtypes.h, twm\_matlab.h and utils.h.

## Control and data acquisition module

This module consists of two sub-modules: (i) Control (user interface GUI), (ii) Acquisition. It is invoked starting from the main routine, where the basic prototype is shown in Figure 0.3. Consequently an unified control and data acquisition GUI appears.

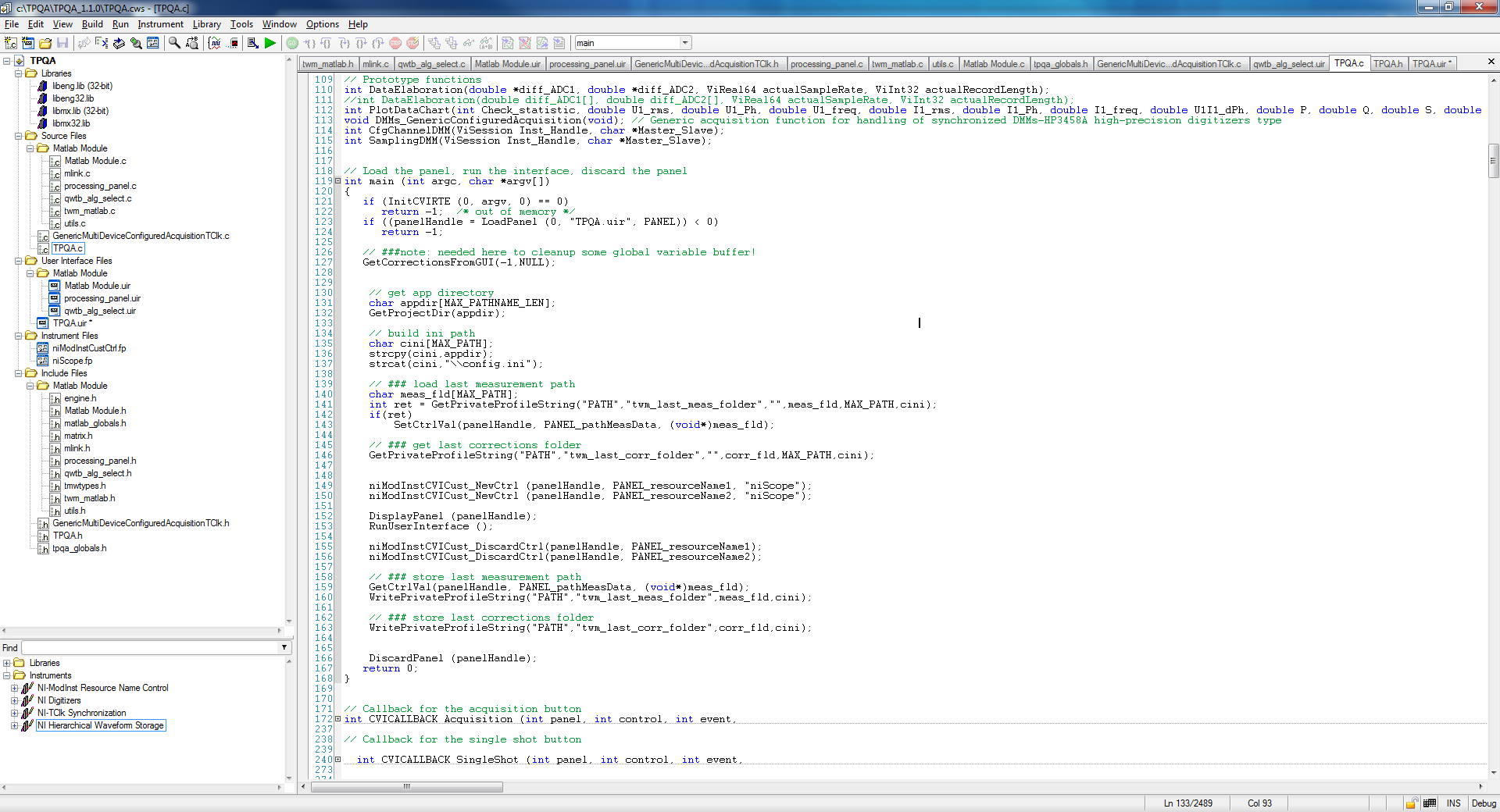


Figure 0.3: Main routine on TPQA

### Control module

The control module is inserted within the main GUI and handles the macro setups developed for traceable power and PQ parameter measurements. It handles four main functions. Figure 0.3 shows the control module on TPQA developed in LabWindows/CVI.

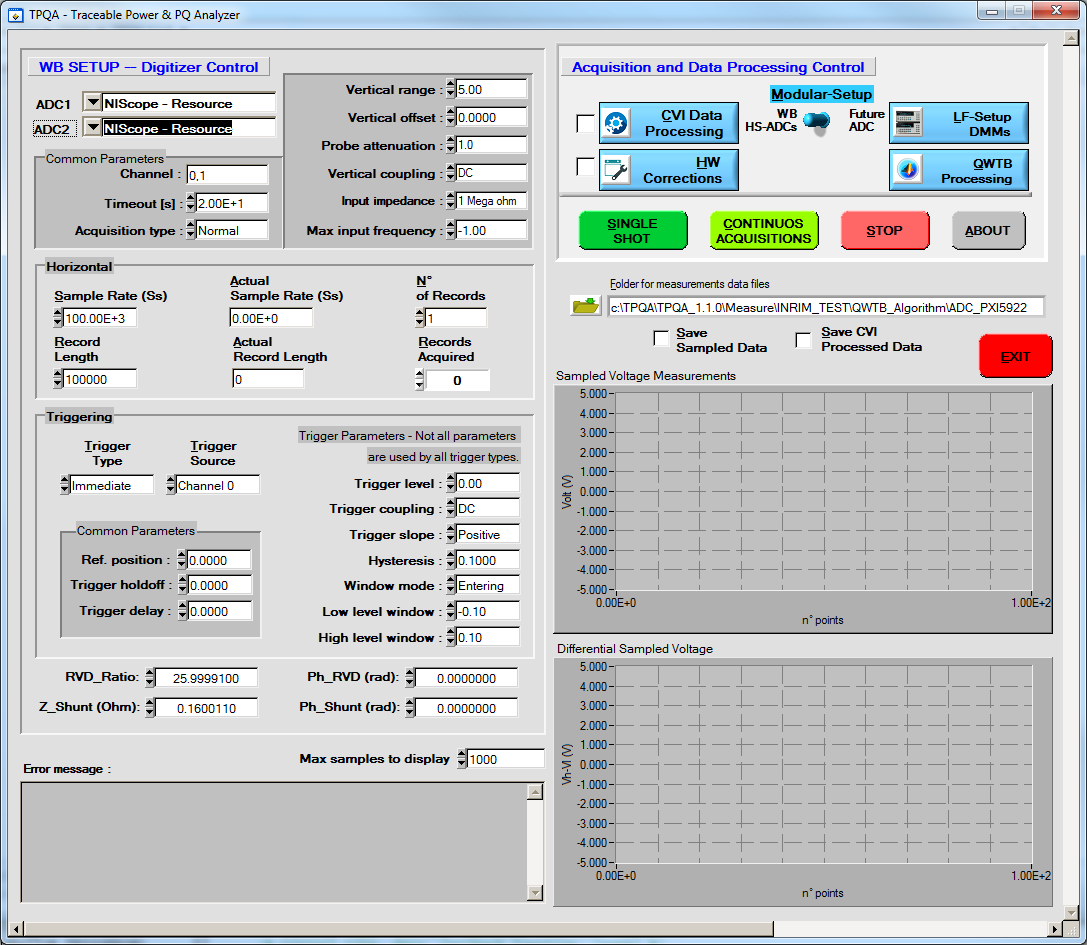
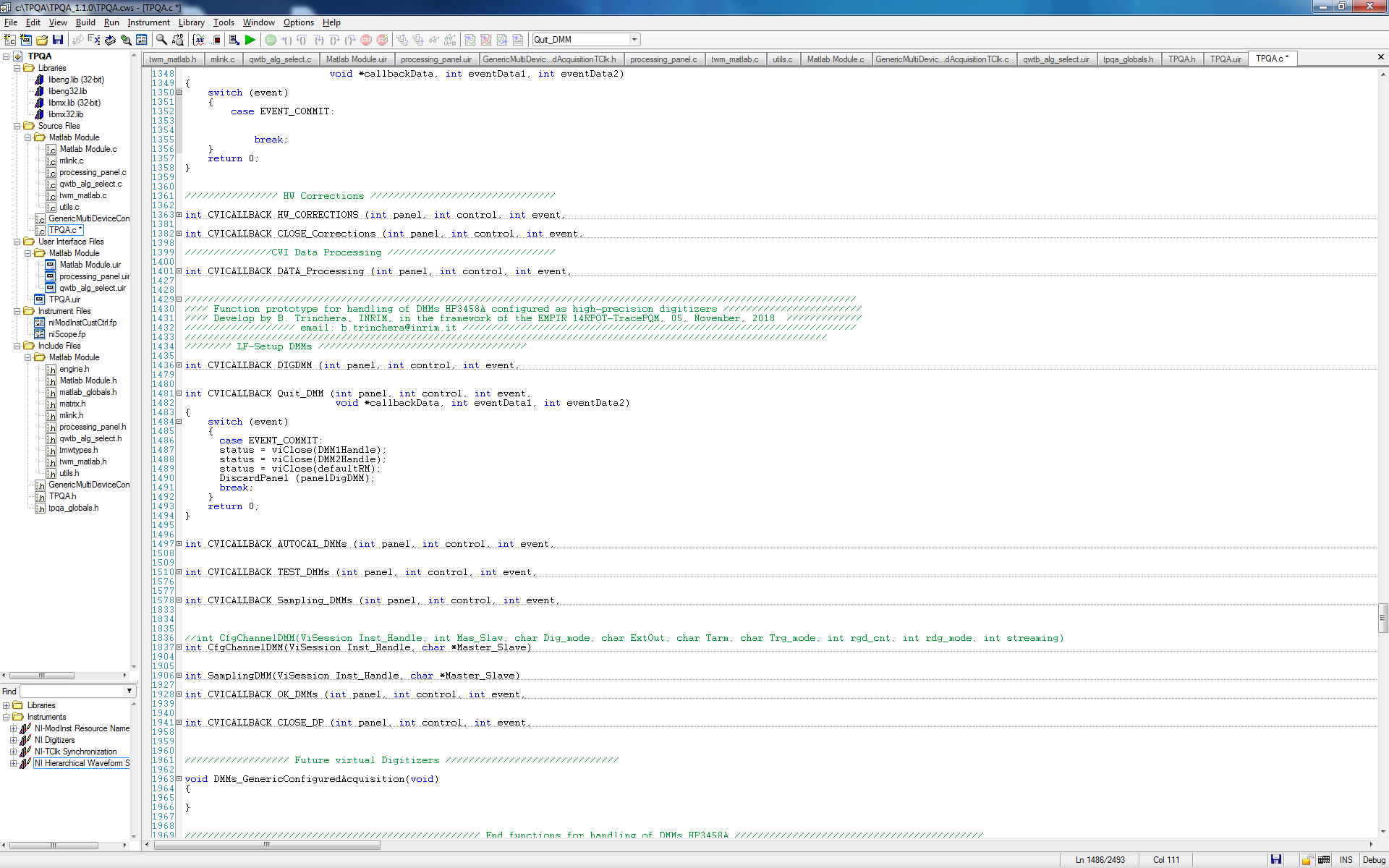


Figure 0.4: Control module of TPQA open source project.

Each button is linked to specific callback functions, generated automatically by the software environment, as follows:

* Main four control button for specific routines entitled **LF-Setup DMMs, QWTB Processing, CVI Data Processing and HW Corrections** . The callback function for each button are shown in Figure 0.5.



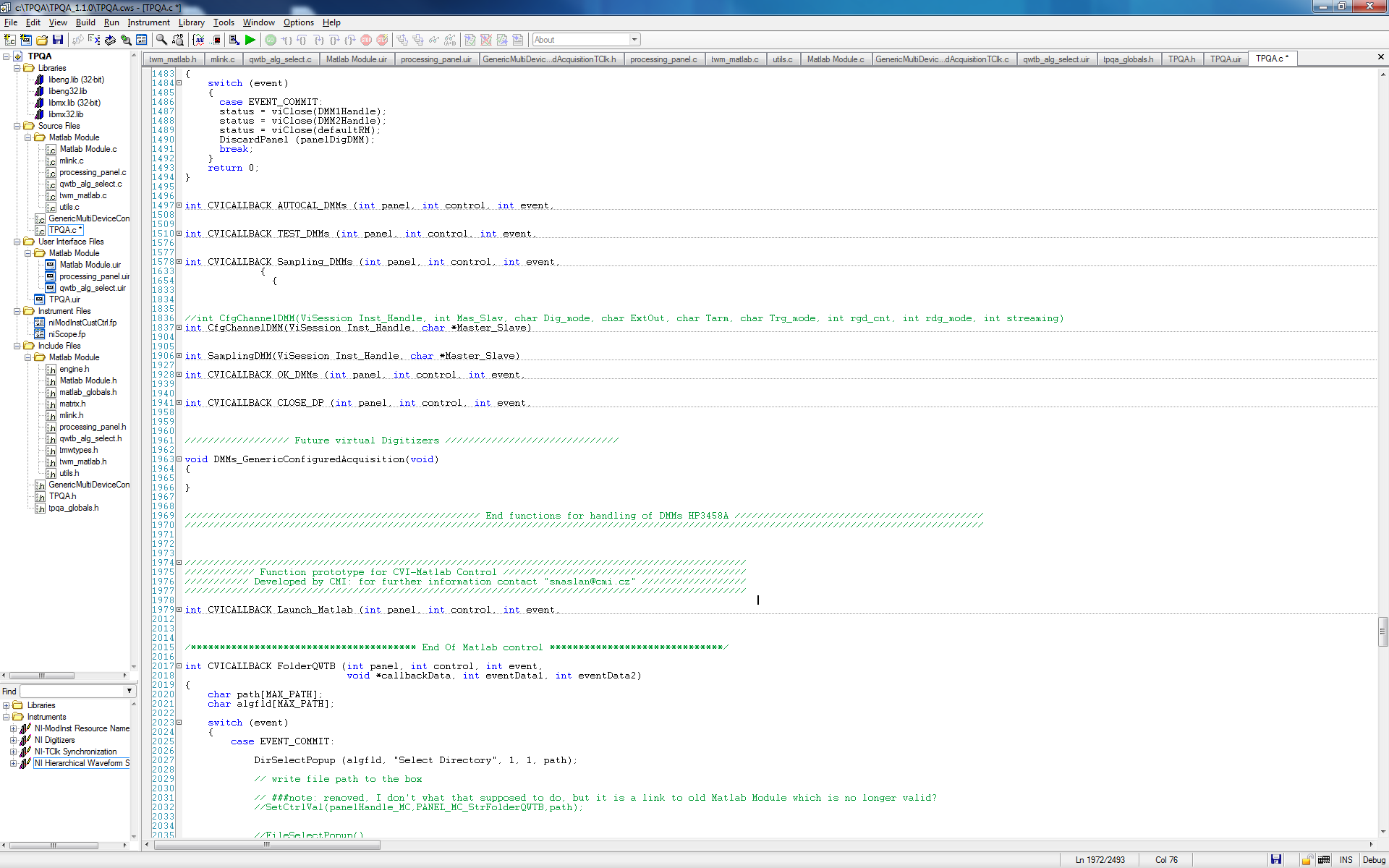


Figure 0.5: Callback functions for each command button.

### Acquisition module

Acquisition module runs in a separate process (see Figure 0‑1). It is composed of two separated GUIs developed for handling of wideband digitizers as PXI-5922 and low speed but high precision DMMs such as HP3458A. Each GUI controls the acquisition parameters of the digitizers. The instrument driver is integrated into the upper level software to create a virtual generic digitizer by means of two control modules.

Figure 0.6 shows the control modules for wideband and LF digitizers. A detailed description of all TPQA control buttons and parameters are given in A331 [8].

|  |  |
| --- | --- |
|  |  |

Figure 0.6: Control module for wideband and DMMs digitizers.

#### Function prototype for LF DMMs

The Figure 0.7 shows the function prototype for handling DMMs connected via GPIB IEEE4882 using VISA drivers. To allow the acquisition of sampled data with DMMs digitizers, in the \*.c file of TPQA there are several functions, some of these are CVICALLBACK functions. These are functions that start to run when their correspondent button is pressed (see [8]). In Figure 0.8 is possible to see a prototype of the functions set to acquire with DMMs digitizers.

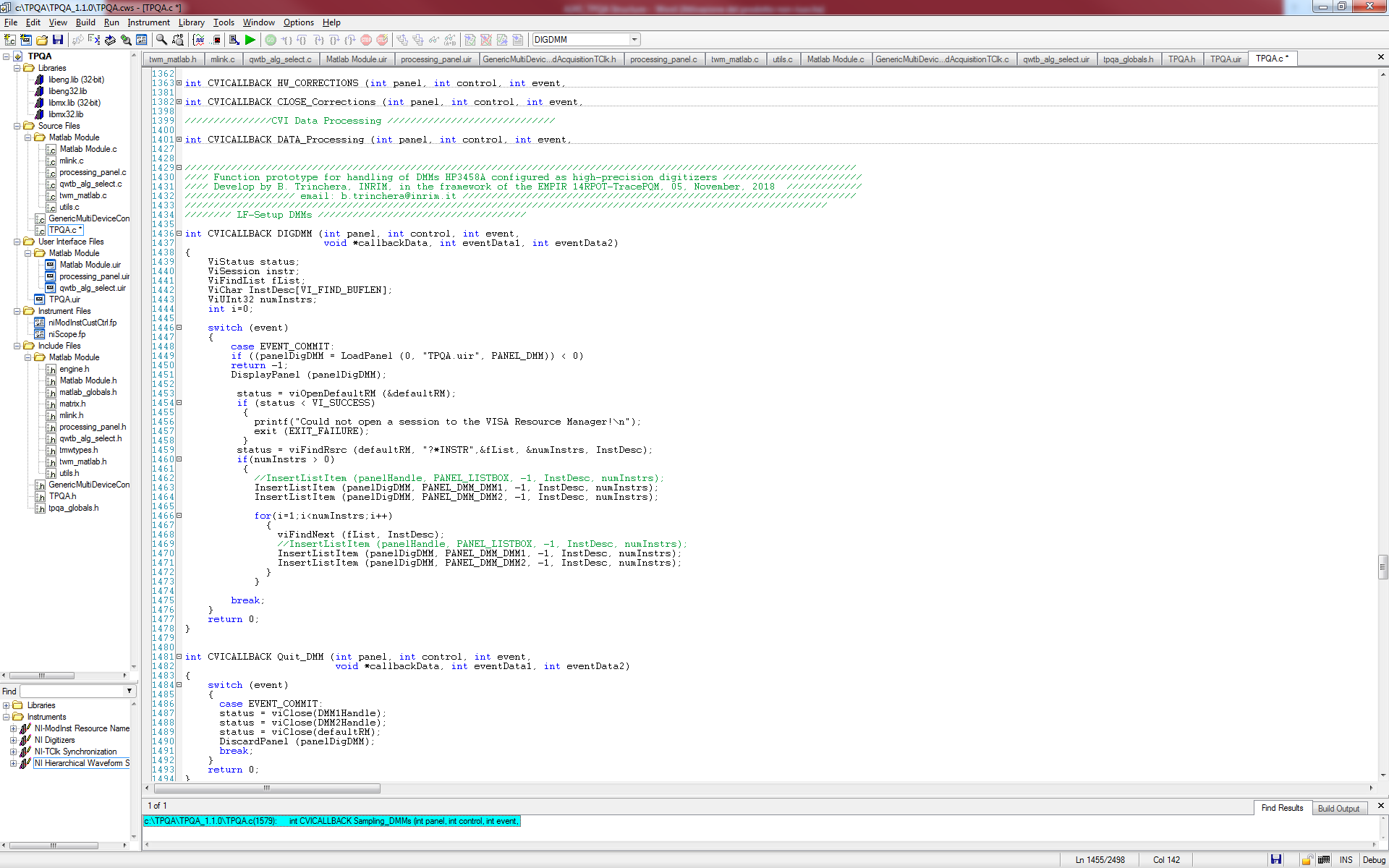


Figure 0.7: Function prototype for handling HP3458A configured as high-precision digitizers.

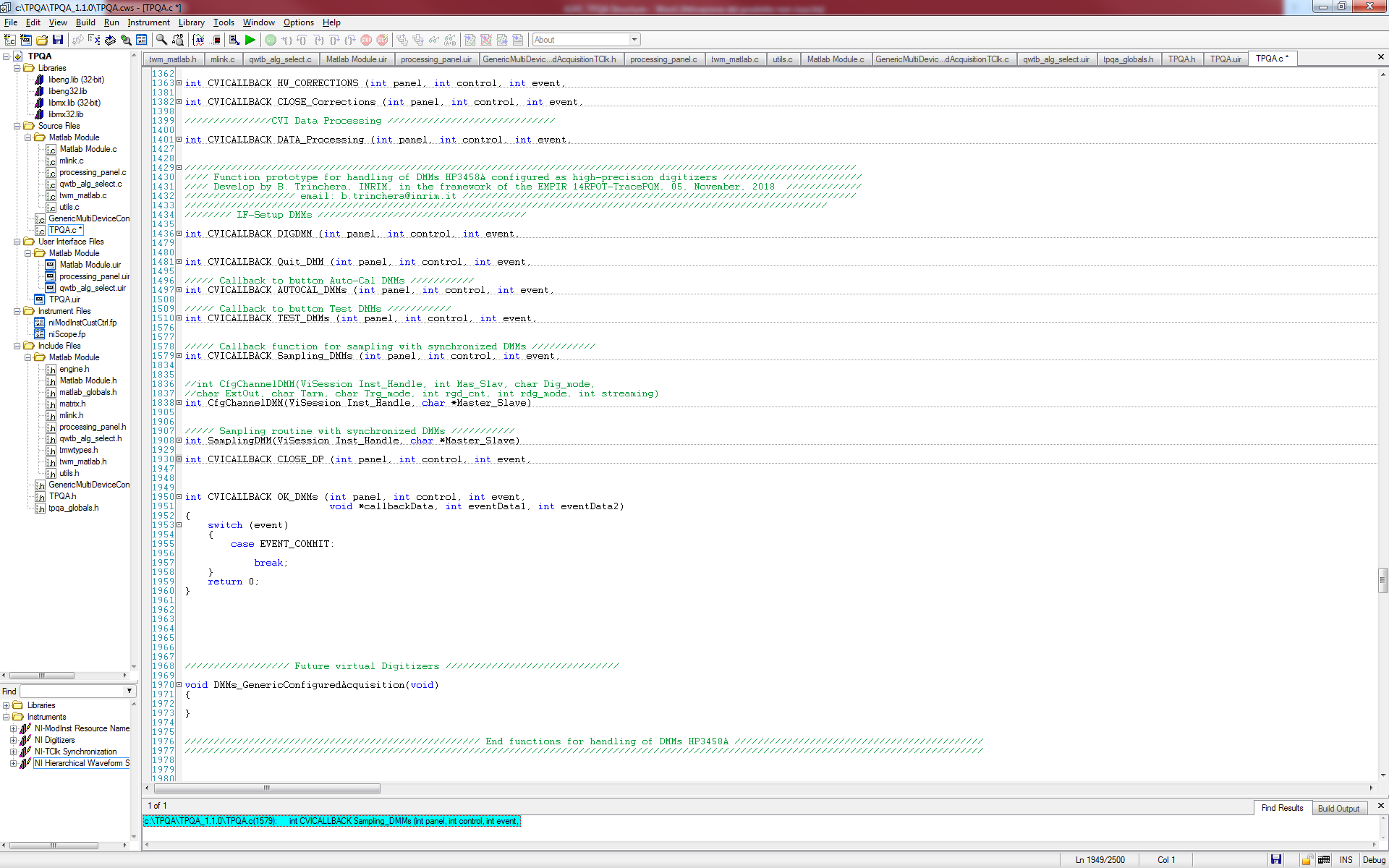


Figure 0.8: Set of Callback functions developed for handling DMMs HP3458A.

To understand better the task that each function does, below will be given a detailed description of them:

* **CfgChannelDMM:** is a function that is callback when the user presses the button to configure the channels for the acquisition, in fact this function allows to configure the digitizer channels;

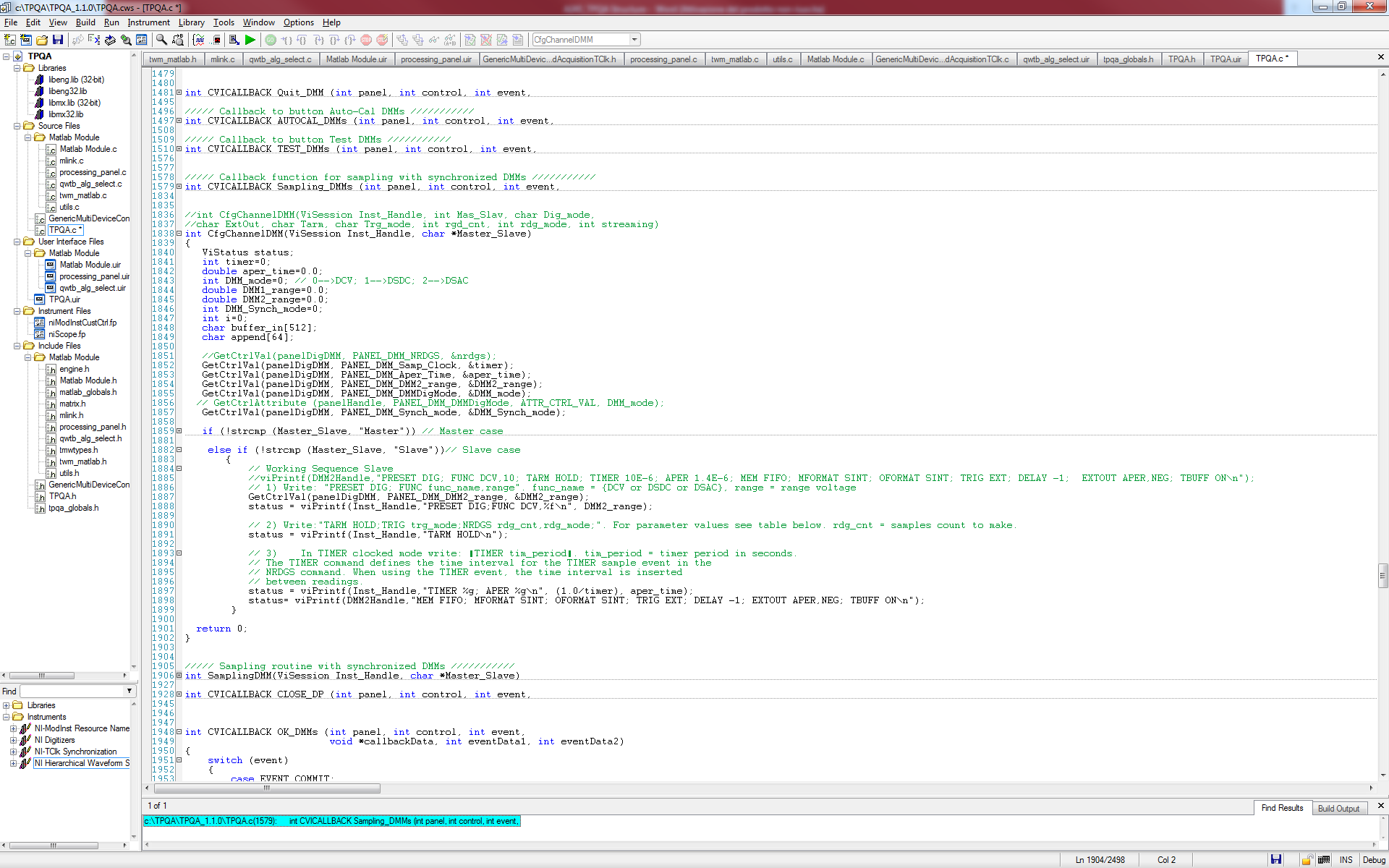


Figure 0.9: Function used to configure the acquisition channels

* **CVICALLBACK Sampling\_DMMs**: is a function that is callback when the user presses the button to start the sampling of data. Within it is employed the function that processes the sampling data ( SamplingDMM function).

|  |  |
| --- | --- |
|  |  |
|  |  |

Figure 0.10: CVICALLBACK Sampling\_DMMs function.

* **Sampling function**: is the function that deals with to implement the sampling routine, it is shown in the Figure 0.11.

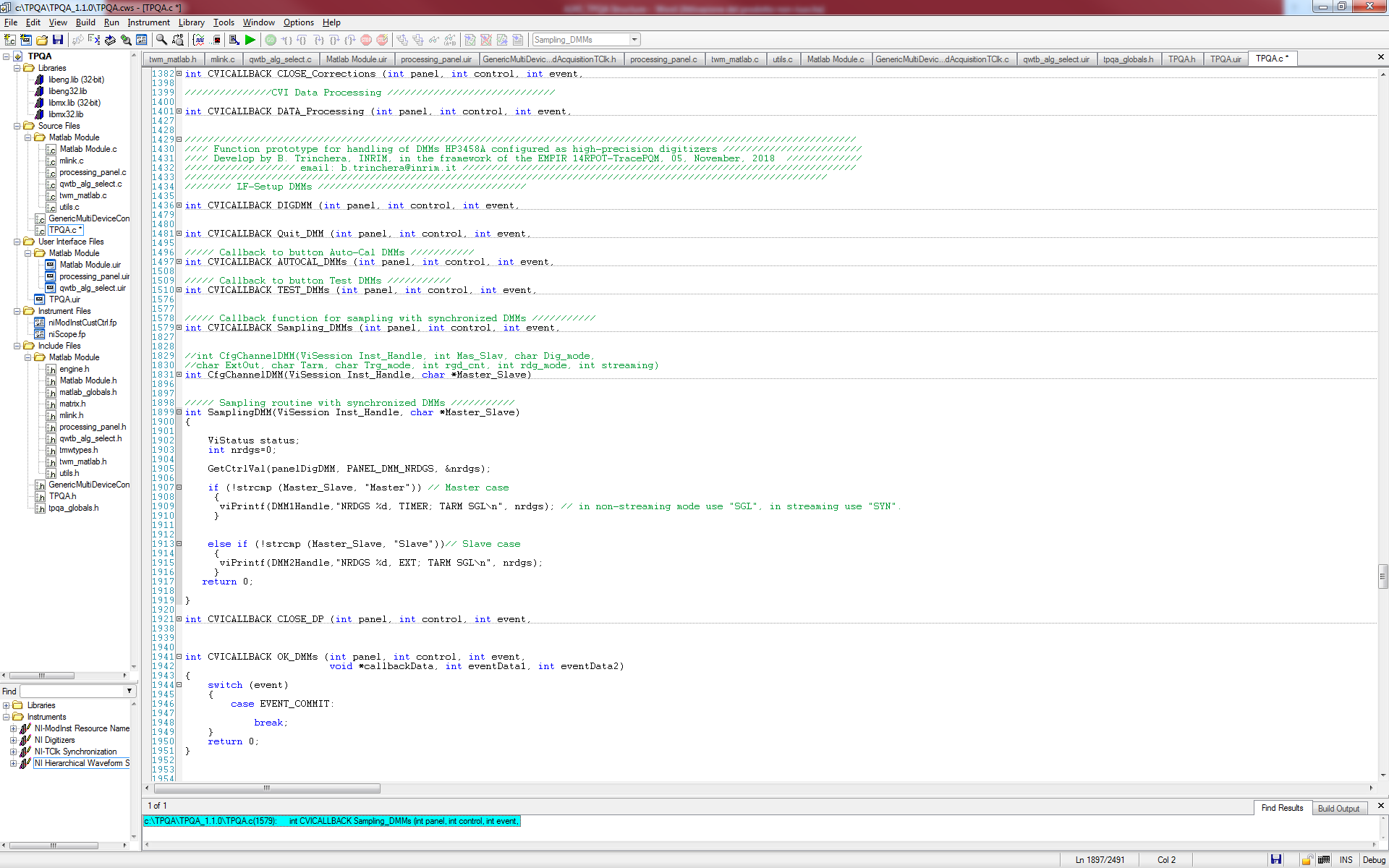


Figure 0.11: SamplingDMM function.

* **CVICALLBACK CLOSE\_DP**: is a function that is callback when the user presses the button to close the sampled data processing session. Using this function, how is possible to see in Figure 0.12, the panel where are processed the sampled data and any of its child panels are removed from memory and them off to the screen.

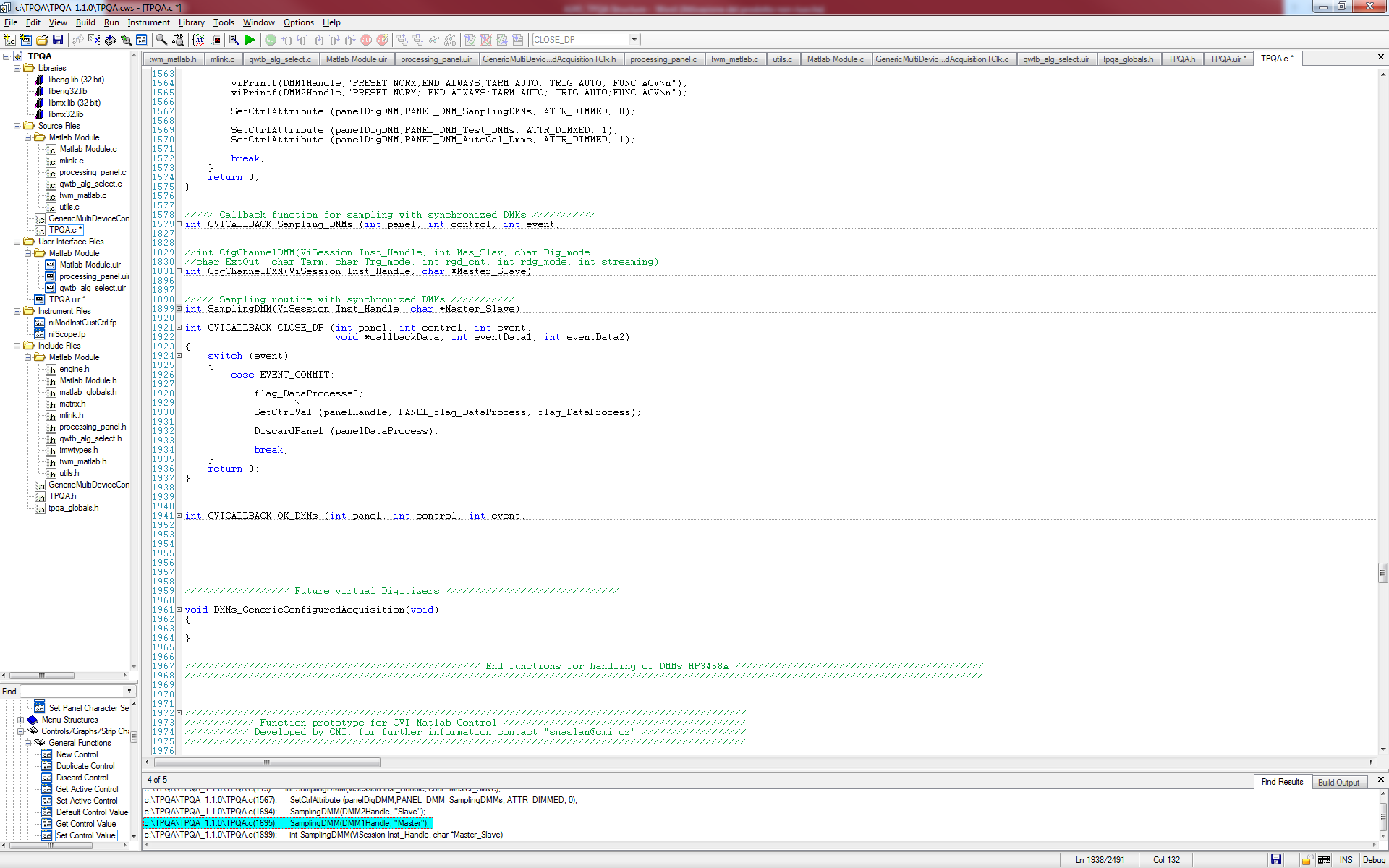


Figure 0.12: CVICALLBACK CLOSE\_DP function.

* **CVICALLBACK TEST\_DMMs**: is a function that is callback when the user presses the button to Test DMMs digitizers. This function is mainly constituted to three steps:
* **Open Visa session:** used to establish a communication session with a device and creates an Instrument Handle that is used by all other VISA operations to perform operations on that session.
* **Call device clear:** used to send a command to clear the device. For GPIB devices, this operation sends a GPIB clear.
* **Set Visa:** used to set a specified attribute for the given object.

The function is shown in Figure 0.13.

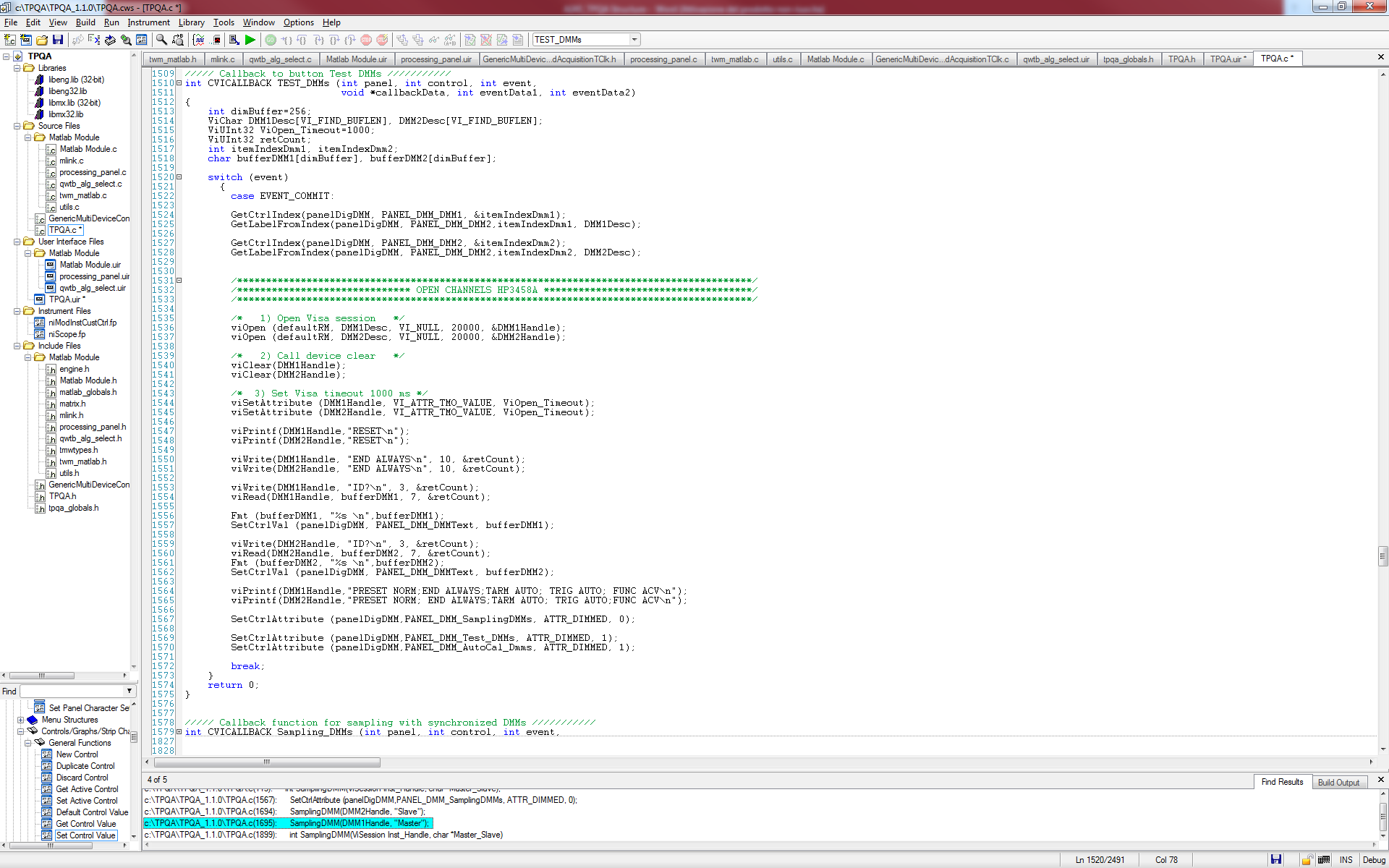


Figure 0.13: CVICALLBACK TEST\_DMMs function.

* **CVICALLBACK Quit\_DMM:** is a function that is callback when the user presses the button to close the LF DMMs digitizers session. Using this function, how is possible to see in Figure 0.14, the LF DMMs digitizers panel and any of its child panels are removed from memory and them off to the screen.

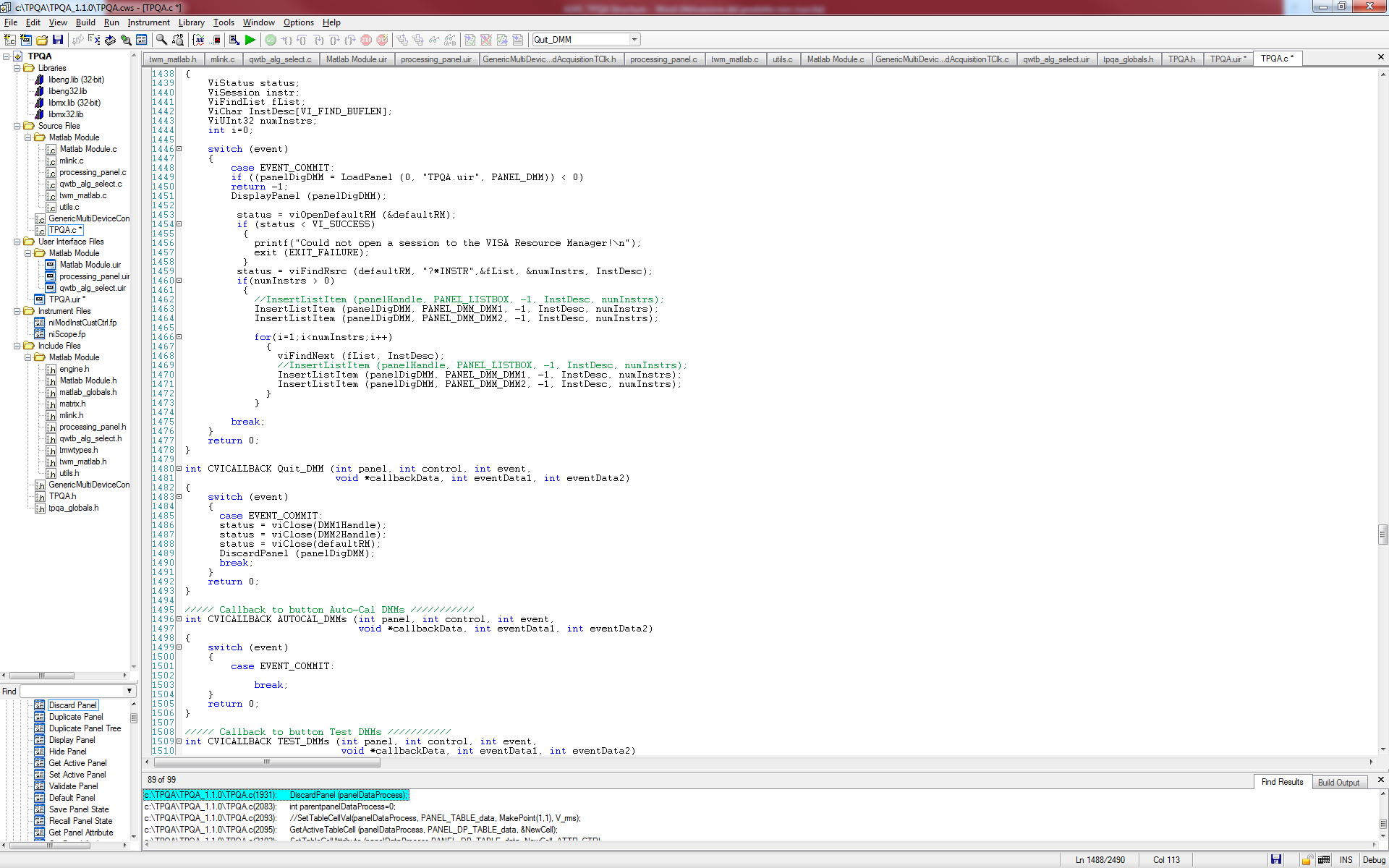


Figure 0.14: CVICALLBACK Quit\_DMM function.

#### Acquisition module for wideband digitizers

Description to be done…

#### Modular driver design

The concept of the modular driver design has been extensively described in TWM. The key idea is the Acquisition module that does not access the drivers of the particular instruments directly, because each digitizer requires completely different approach. So it was decided to insert a command translation layer between the acquisition module and the drivers of physical instruments. This layer was called **Virtual digitizer**. All HW specific function calls of each digitizer are translated to a universal format and merged into a few basic VI functions which are, for the acquisition module, identic for any digitizer no matter how different is the HW control implementation inside. The basic block diagram of the TPQA in current version is shown in Figure 0.15.

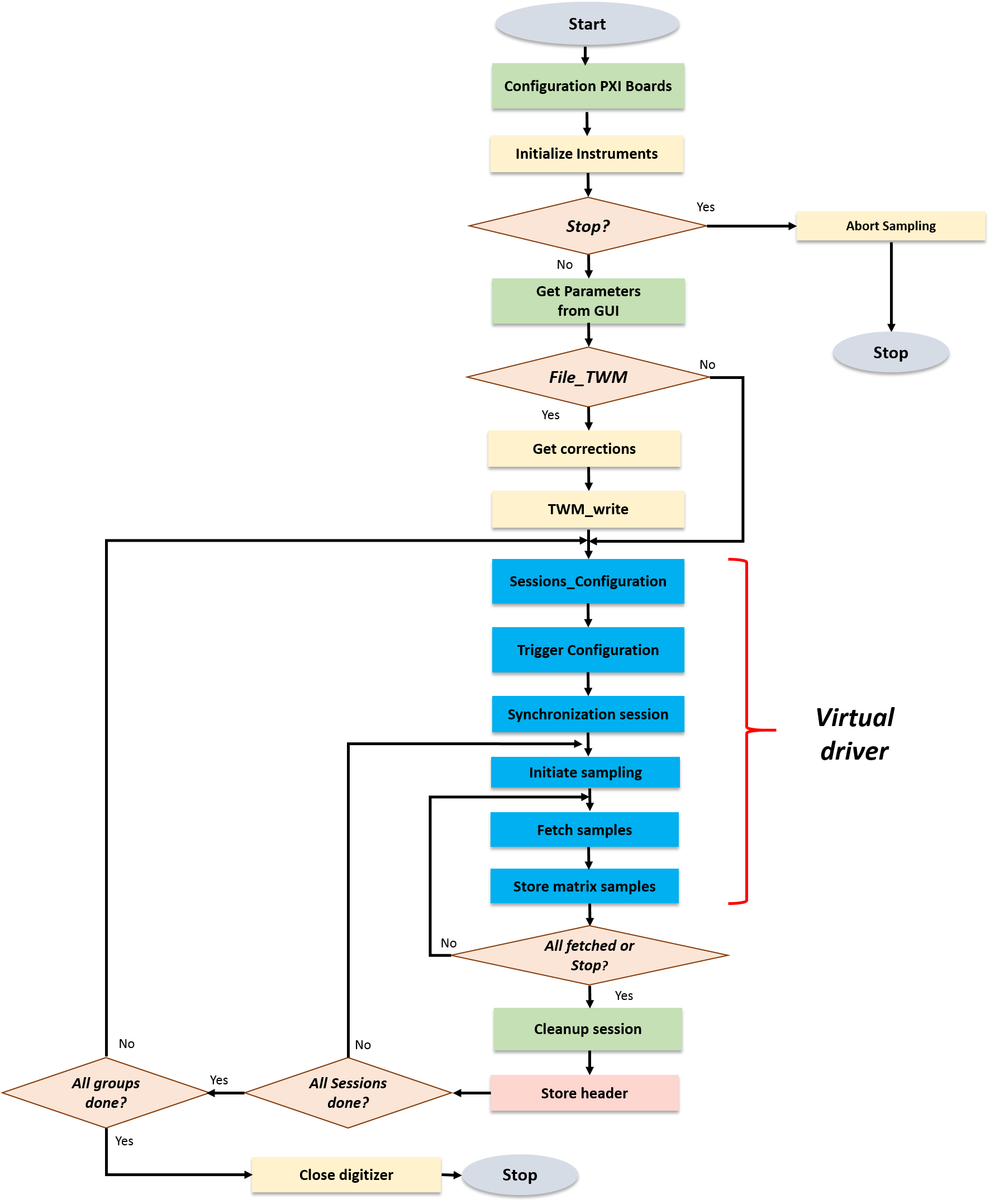


Figure 0.15: Virtual digitizer driver structure and data flow for TPQA open software tool project.

#### Virtual driver functions structure

The concept of the virtual driver has been developed in order to avoid the direct access directly the drivers of particular instruments.

It was verified that accessing directly of instrument drivers requires a different approach for each particular ADC board. It was substituted such an approach with a different mechanism which uses a \*.c function translator able to interact as an interface layer between the data acquisition module and the physical instrument drivers. These \*.c functions represent the virtual driver.

In TPQA open software project such an interface is composed for a specific board, e.g. PXI-

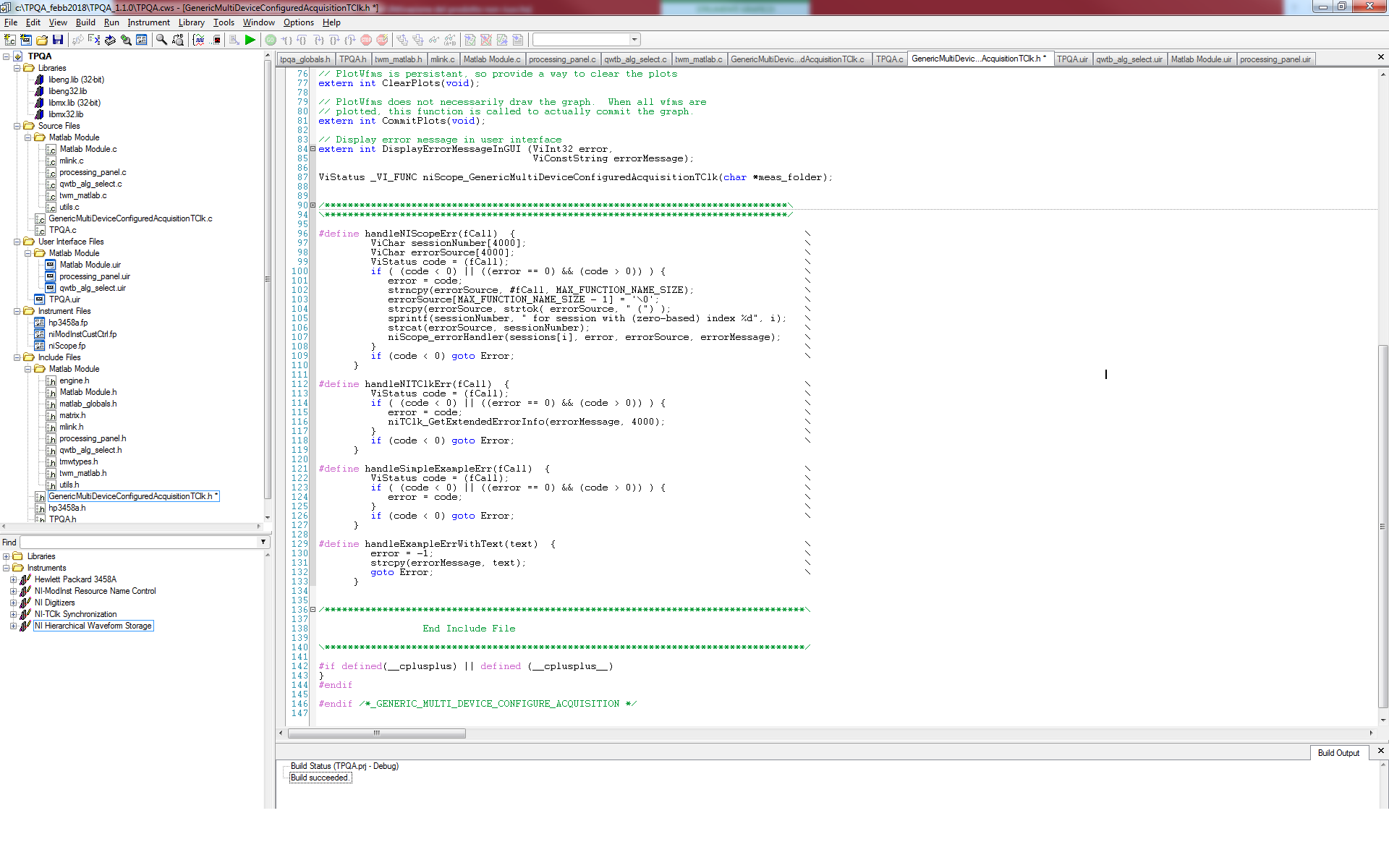
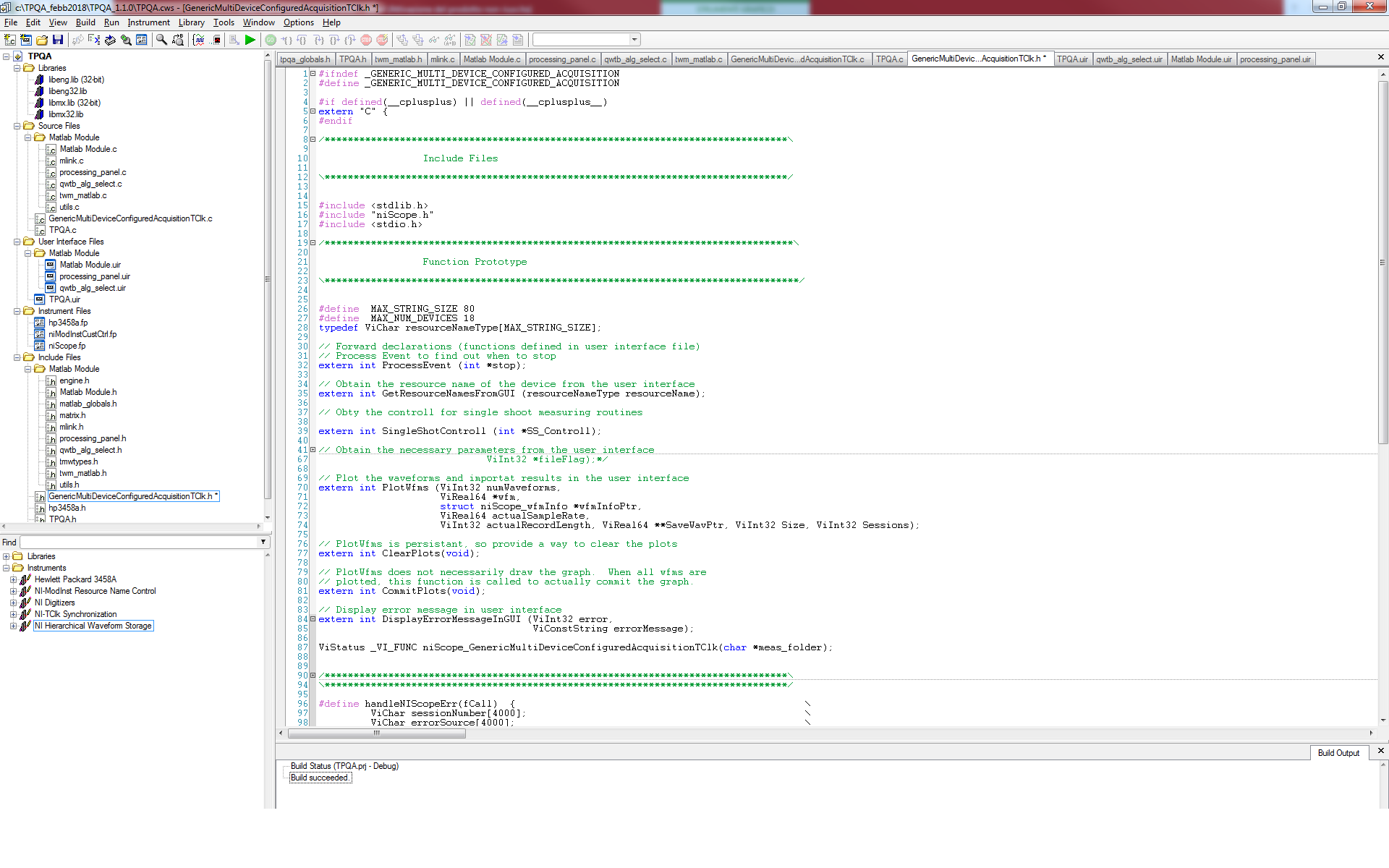
NI-5922 digitizer, by two files a (\*.c file and a header (\*.h)) file:

* *GenericMultiDeviceConfiguredAcquisitionTClk.c;*
* *GenericMultiDeviceConfiguredAcquisitionTClk.h.*

Only some remarks to take into account regard to the data storing of sampled data when the digitizer runs continually. The solution adopted aims to collect the sampled data and then stored them directly on hard drive while the ADCs runs continually. Furthermore, the TPQA program allows to the user to change the parameters in real time. For this reason the developed TPQA code has some difference with the TWM structure.

In Figure 0.16 is shown the \*.h file (header file) employed as translator for the virtual driver used with ni.Scope driver.

Figure 0.16 Header structure of the translator used with ni.scope driver.



##### ADC session

#### Virtual driver function reference manual

Within a virtual driver are employed several functions. An example of virtual driver developed for use with ni.Scope driver is shown in Figure 0.17.

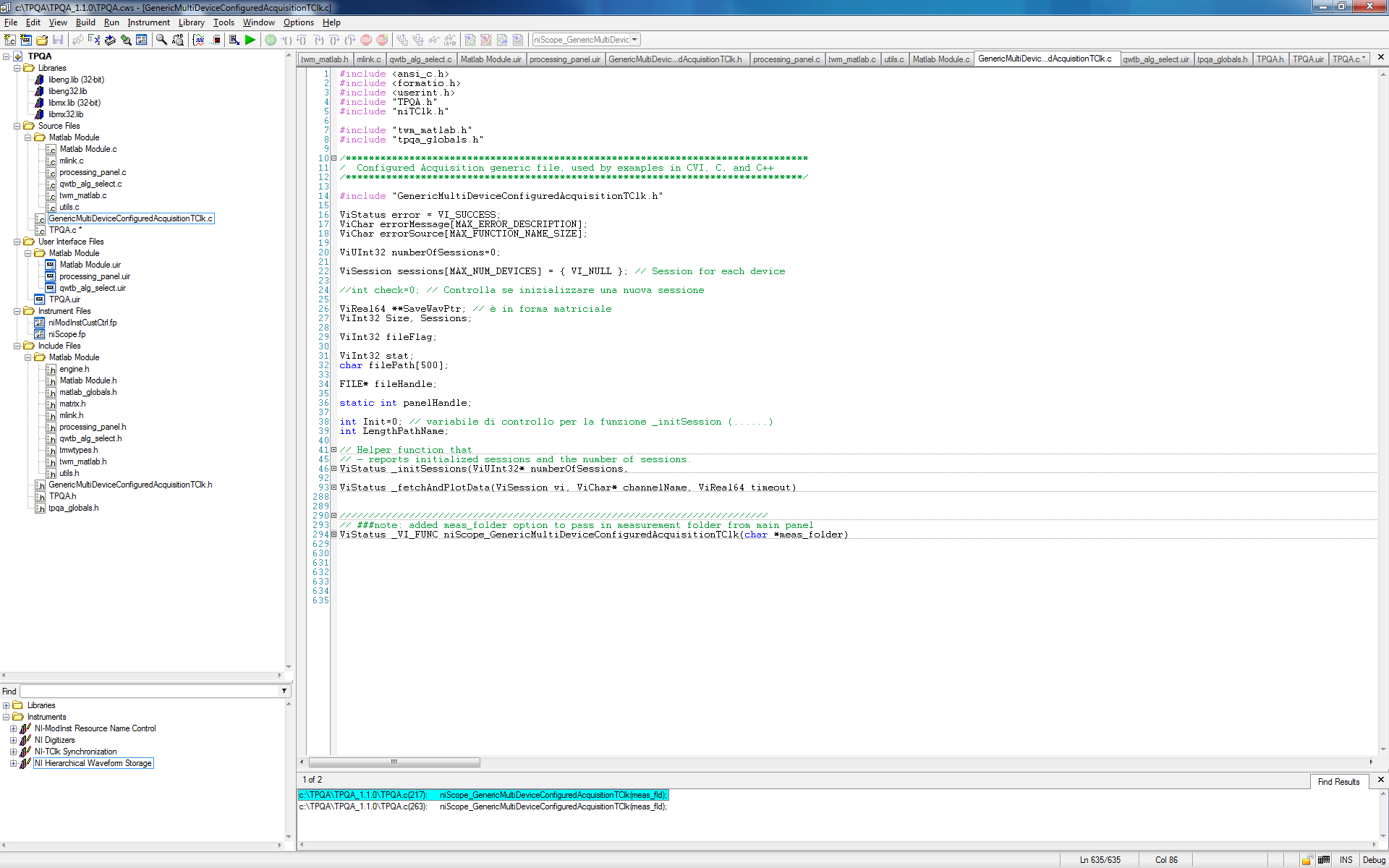


Figure 0.17 Prototype of \*.c translator developed for LabWindows/CVI to be used with niscope driver.

The virtual driver called **Vi\_Status \_Vi\_FUNC ni Scope\_GenericMultiDeviceConfiguredAcquisitionTClk (char\*meas\_folder)** is mainly composed by functions that deal with of:

* Session configuration;
* Synchronization of session;
* Trigger configuration;
* Initiate sampling;
* Fetch and store matrix sampled data;
* Abort digitizing process;
* Clean-up sessions.

In the following sessions will be explain the tasks of these several functions.

##### Sessions Configuration

The session configuration is mainly composed by these functions:

* **GetParametersFromGUI ()**: a prototype is shown in Figure 0.18 and the function has as input variables all parameters shown in TPQA panel. Through this function the algorithm read all parameters insert by user;

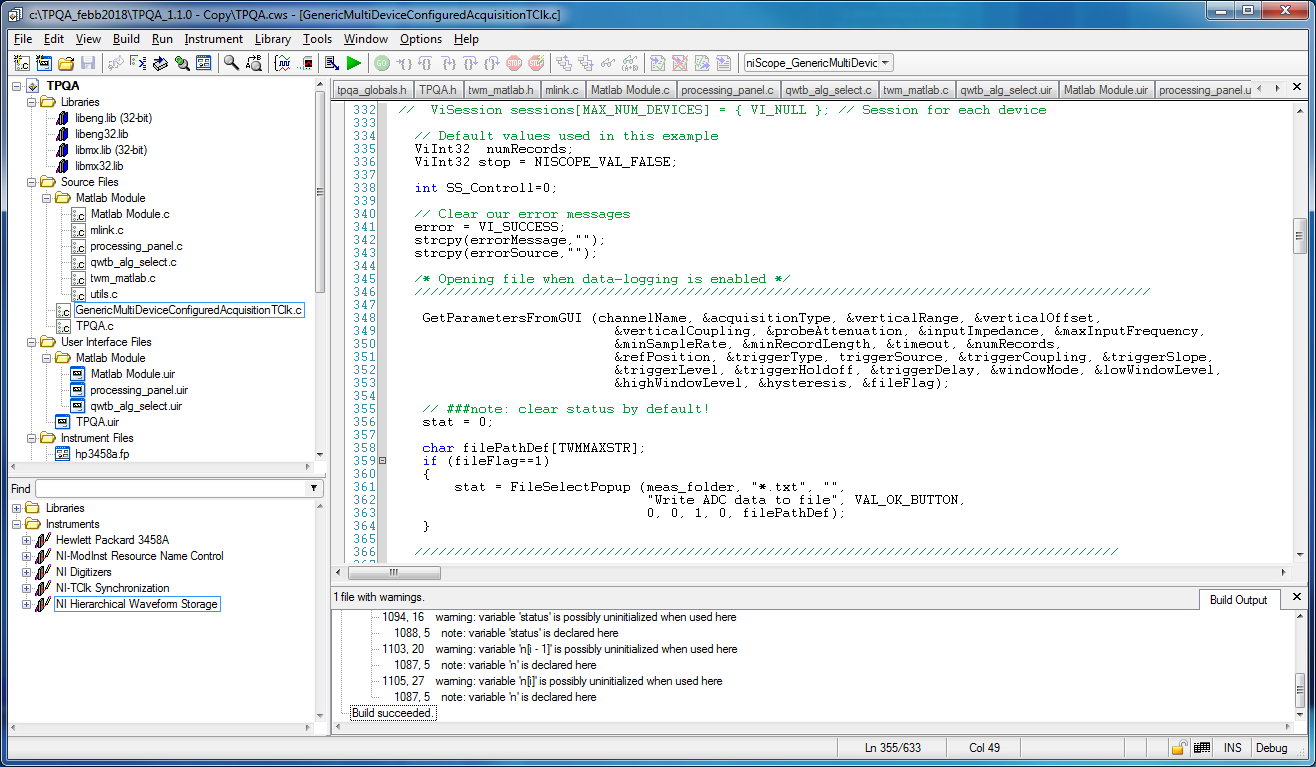


Figure 0.18: GetParametersFromGUI function.

* **GetCorrectionsFromGUI()**, this function is shown in Figure 0.19 and through it away the algorithm read the hardware corrections uploaded by user;

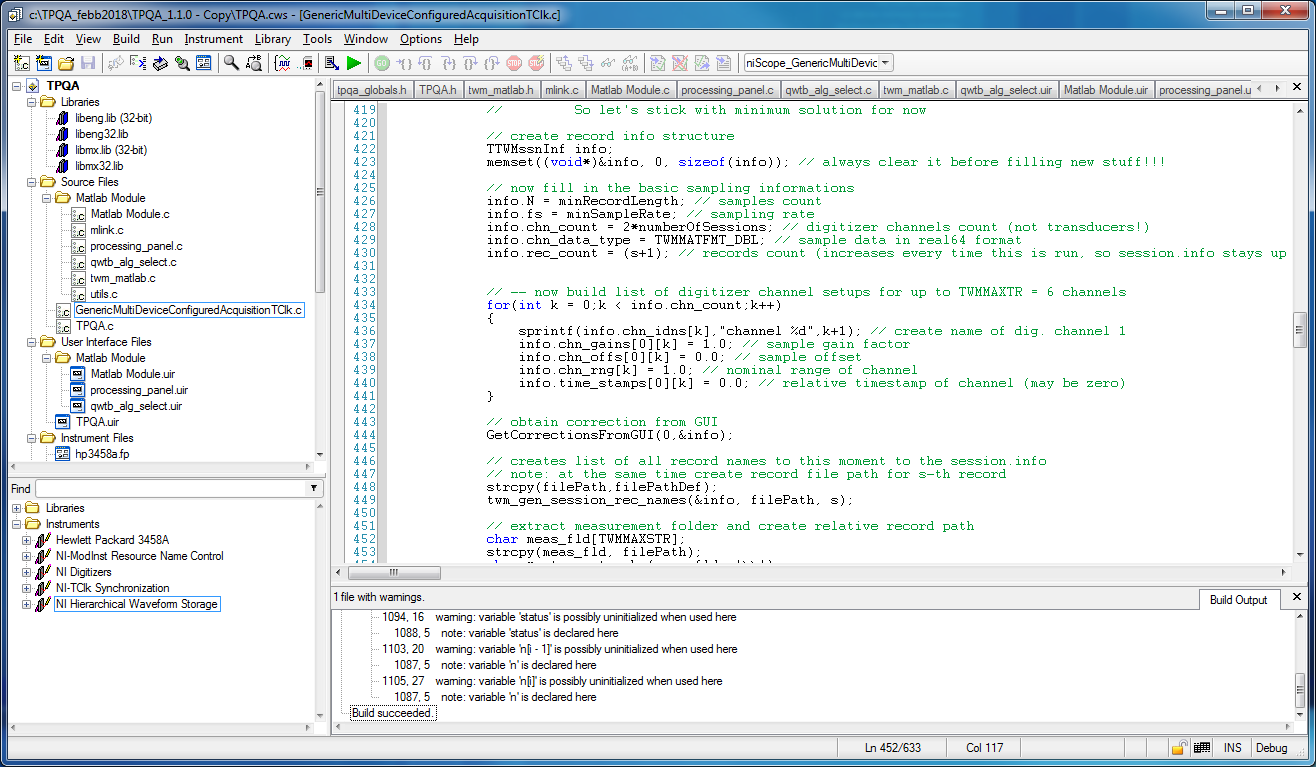


Figure 0.19: GetCorrectionsFromGUI function.

* **twm\_write\_session ()**, for the generation of measurement session for use with QWTB post-processing tool.

##### Trigger Configuration

To configure common properties of trigger are employed the following functions: TODO-Specify better the input and output of the functions

1. **handleNIScopeErr(niScope\_ConfigureTriggerEdge())**;
2. **handleNIScopeErr(niScope\_ConfigureTriggerHysteresis())**;
3. **handleNIScopeErr(niScope\_ConfigureTriggerDigital ())**;
4. **handleNIScopeErr(niScope\_ConfigureTriggerWindow ())**;
5. **handleNIScopeErr(niScope\_ConfigureTriggerImmediate ())**.

The first four functions listed above are used to configure common properties for analog triggering, while the last one is used when the user insert an immediate trigger. In Figure 0.20 is shown the mode to configure the trigger type using a switch case.

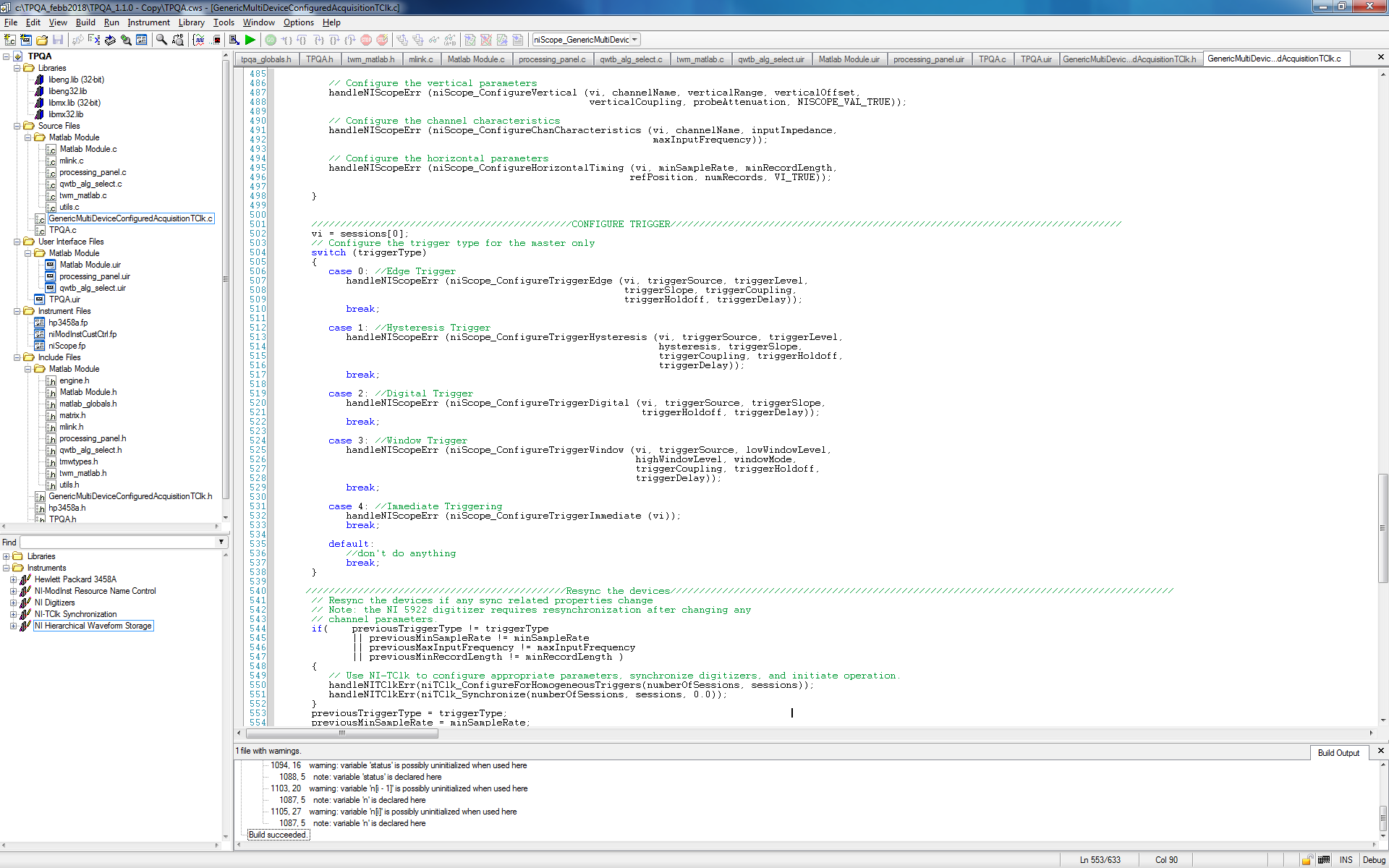


Figure 0.20: Trigger configuration code.

##### Synchronization session

The functions used to synchronize the devices are (see Figure 0.21): TODO-Specify better the input and output of the functions

* 1. **handleNITClkErr(niTClk\_ConfigureForHomogeneousTriggers())**, this function is used to configure the attributes for the reference clocks, start triggers, reference triggers, script triggers, and pause triggers.
  2. **handleNITClkErr(niTClk\_Synchronize())**, this function synchronizes the TClk signals on the given sessions.

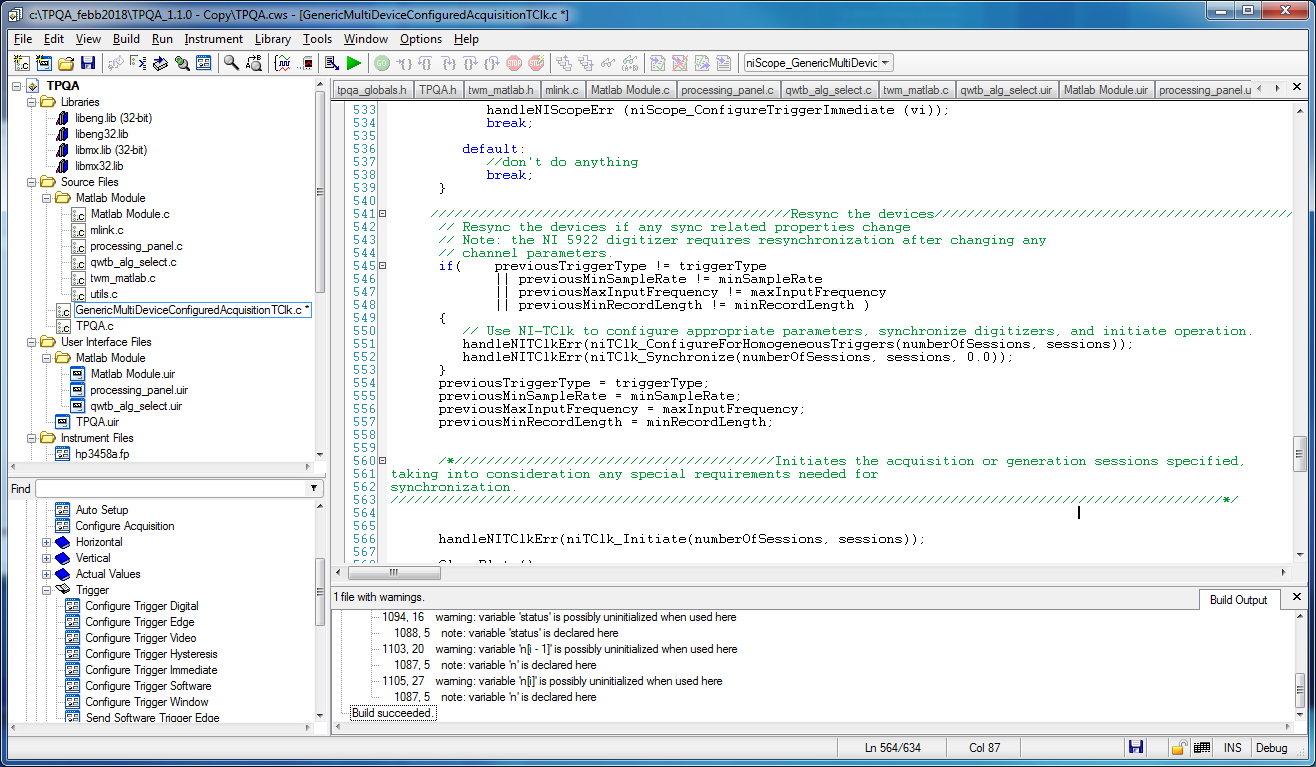


Figure 0.21: Synchronization session code.

##### Initiate sampling

TODO-Specify better the input and output of the functions

The function used to initiate sampling is called **handleNITClkErr(niTClk\_Initiate())** and it is the function that initiates the acquisition or generation session. In the code this function has been used as shown in Figure 0.22.

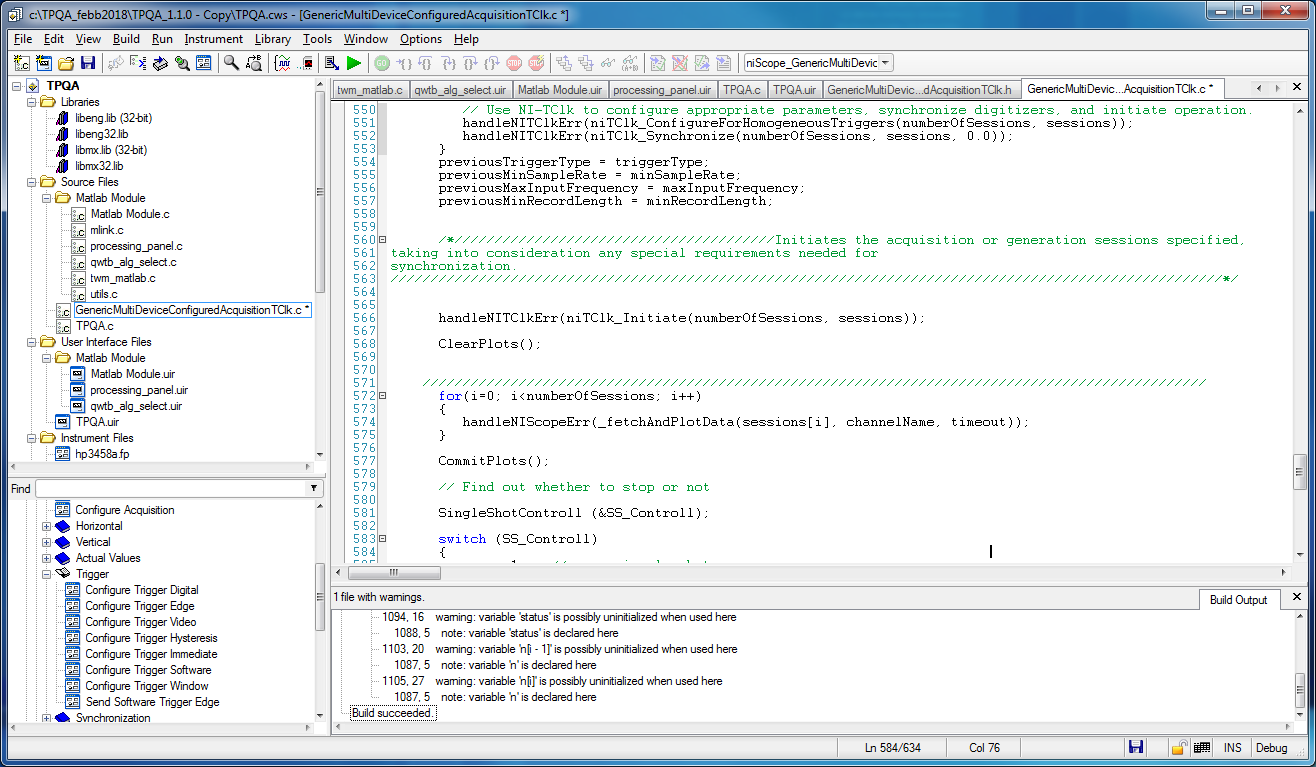


Figure 0.22: Initiate sampling function.

##### Fetch and store matrix sampled data

TODO-Specify better the input and output of the functions

**HandleNIScopeErr(\_fetchAndPlotData())** is function used to fetch the acquisition data and then plot them. In the code this function has been used as shown in the figure below.



Figure 0.23: Fetch and store sampled data.

The final aim of this function is to plot the sampled data (waveforms acquired). To reach this goal there is a function recalled in the **fetchAndPlotData()** function, that allows to have an interface between the virtual digitizers (**GenericMultiDeviceConfiguredAcquisitionTClk()** function) and main callback function (**CVICALLBACK Acquisition()**). This function just described is **PlotWfms ()** function and it is employed within the primary TPQA software open tool, that is TPQA.c file.

In this file the virtual drive goes in action when the **CVICALLBACK Acquisition()** function is recalled. This happens when the user through the TPQA main panel presses the **CONTINOUS ACQUISITION** button.

After that the acquisition session is concluded and then in the TPQA.c file the **PlotWfms ()** is callbacks, as is shown in Figure 0.24.

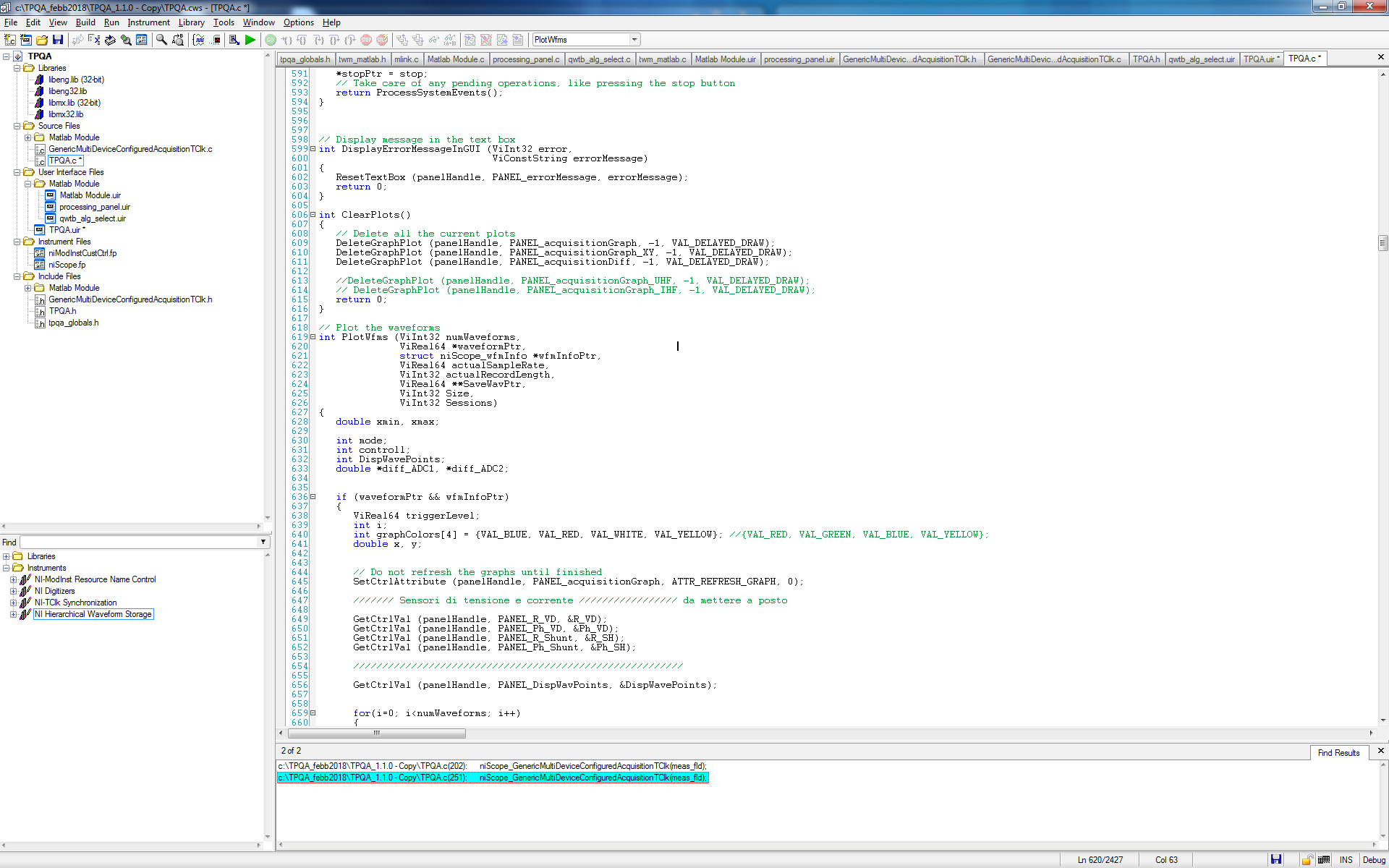


Figure 0.24 PlotWfms function.

The PlotWfms function is constituted from different input variables that will be described in the following:

* *ViInt32 numWaveforms*: is the number of waveforms recorded by acquisition units. In particular the number is two if there is just one single board, instead the number becomes four when there are two boards and so on*;*
* *ViReal64 \*waveformPtr*: is the variable used to allocated the information of the waveforms, for example the number of waveforms;
* *struct niScope\_wfmInfo \*wfmInfoPtr*: is a matrix (*struct*) that contains all values of waveforms recorded;
* *ViReal64 actualSampleRate:* is the sample rate returned from the digitizer, i.e. the digitizer returns a value of sample rate that is closer to that to which it can work;
* *ViInt32 actualRecordLength:* is the array that contains the length of recorded??
* *ViReal64 \*\*SaveWavPtr:??*
* *ViInt32 Size:* is the number of ADC channels;
* *ViInt32 Sessions:* is thenumber of ADC boards.

With all information on virtual drive given in the Flow chart LabWindowsTM/CVI environment section, the users can to start to acquaint themselves with TPQA software open tool and, if they want, start to integrate new digitizers.

##### Function’s list

TODO-Specify better the functions

**[GetParametersFromGUI]**

**Class="Function"**

**Prototype="int GetParametersFromGUI(char \*channel, long \*acquisitionType, double \*verticalRange, double \*verticalOffset, long \*verticalCoupling, double \*probeAttenuation, double \*inputImpedance, double \*maxInputFrequency, double \*minSampleRate, long \*minRecordLength, double \*timeout, long \*numRecords, double \*refPos, long \*triggerType, char \*triggerSource, long \*triggerCoupling, long \*triggerSlope, double \*triggerLevel, double \*triggerHoldoff, double \*triggerDelay, long \*windowMode, double \*lowWindowLevel, double \*highWindowLevel, double \*hysteresis, long \*fileFlag);"**

**[\_fetchAndPlotData]**

**Class="Function"**

**Prototype="long \_fetchAndPlotData(unsigned long vi, char \*channelName, double timeout);"**

**[GetCorrectionsFromGUI]**

**Class="Function"**

**Prototype="int GetCorrectionsFromGUI(int panel, TTWMssnInf \*info);"**

**[niScope\_GenericMultiDeviceConfiguredAcquisitionTClk]**

**Class="Function"**

**Prototype="long niScope\_GenericMultiDeviceConfiguredAcquisitionTClk(char \*meas\_folder);"**

##### Abort Digitizing Process

TODO

##### Clean-up sessions

TODO

## Processing module

TODO

### Processing module – LabWindows/CVI environment

TODO-only functions

#### TPQA Processing configuration GUI

TODO-only functions

#### TPQA post- processing GUI

### Processing module – Matlab component

The Matlab component of the processing module is standalone set of m-functions. They are executed from TWM via the GOLPI interface [4]. From typical user point of view the only four functions are to be called directly:

1. “qwtb\_exec\_algorithm.m” for execution of the processing on the measurement session.
2. “qwtb\_get\_results\_info.m” to get information about available results in the session.
3. “qwtb\_get\_results.m” to load and format results for displaying in matrix form.
4. “qwtb\_plot\_result.m” to load and display results as a graph.

The rest of the m-functions are either sub-functions of abovementioned or special functions that are rarely used directly. Only the top level functions will be described in following sections.