

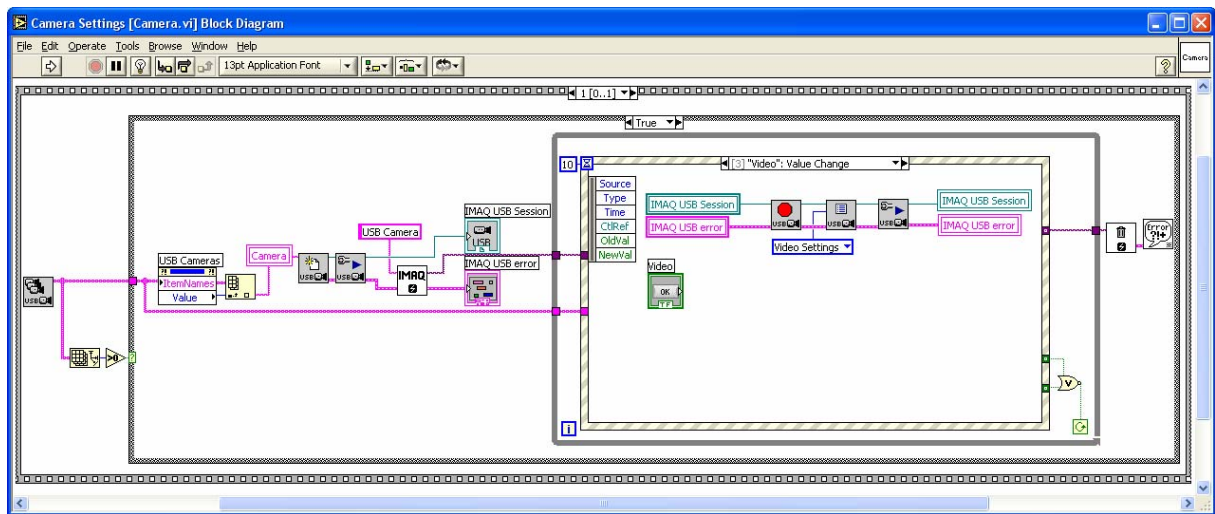


Figure 3.103: Illustration of the code of the function “Camera” that stops the current USB camera session, opens the “Video Settings” window of that camera and starts to play the frames again using the new configuration

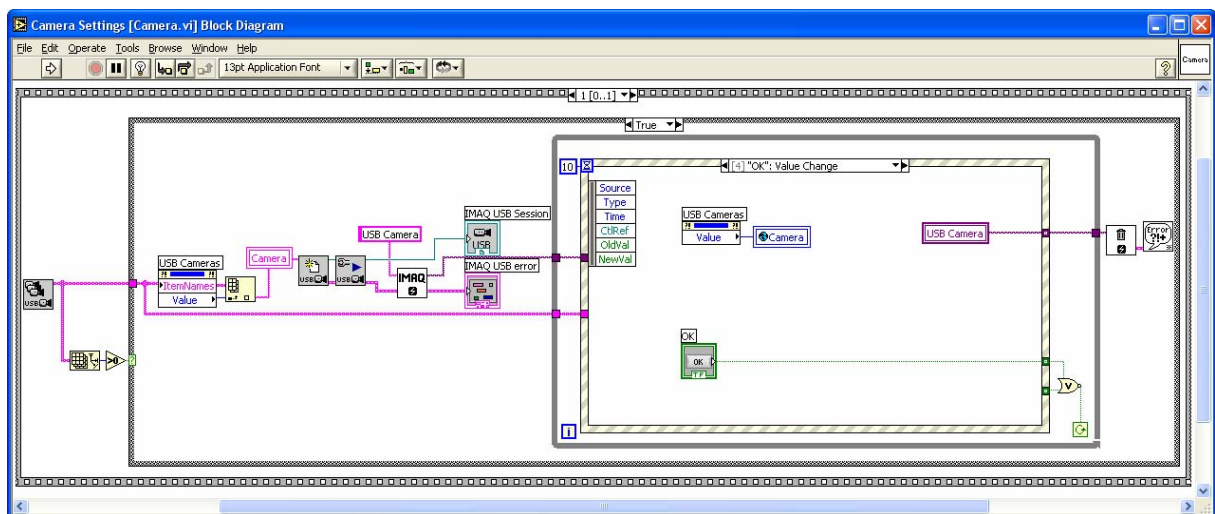


Figure 3.104: Illustration of the code of the function “Camera” that upgrades the value of the global variable “Camera” if the button OK is pressed

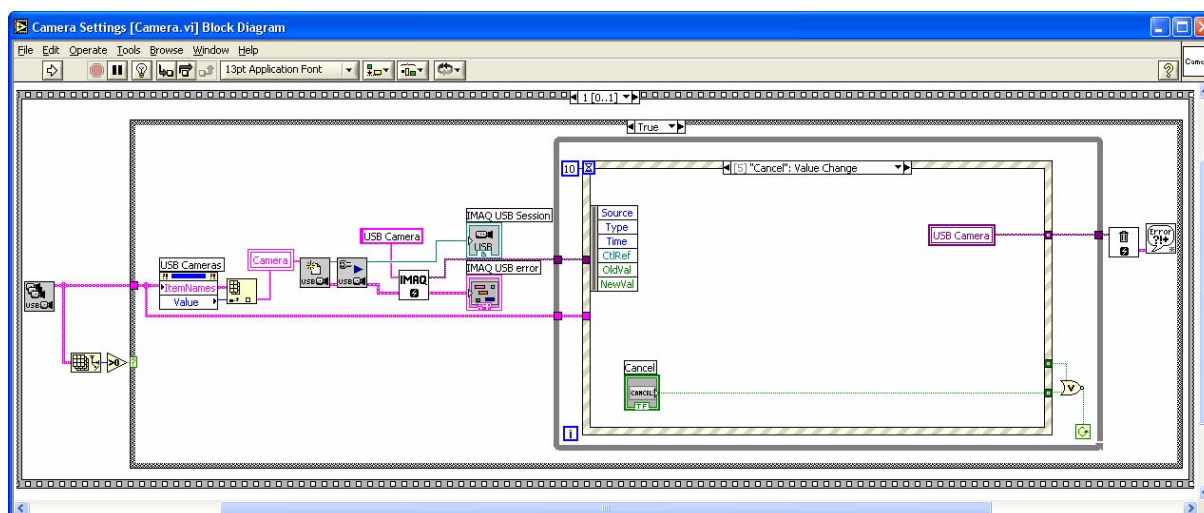


Figure 3.105: Illustration of the code of the function “Camera” that does not upgrade anything if the button Cancel is pressed

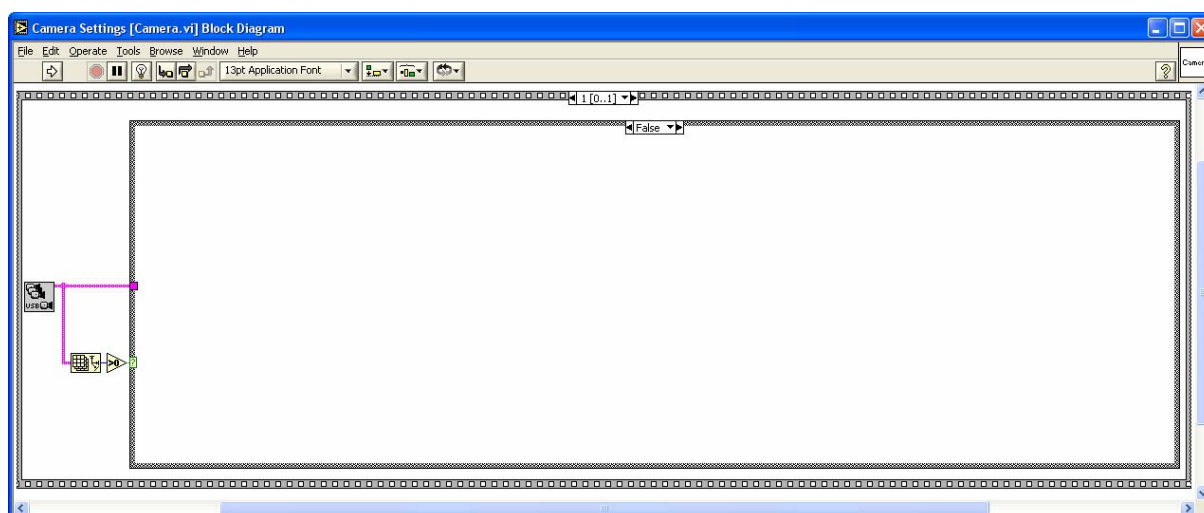


Figure 3.106: Illustration of the code of the function “Camera” that does not execute anything if there is no USB camera plugged to the computer

If none of the items of the “Menu Selection” is selected, a default value equal to “false” is used for keeping the software waiting for an event, without executing any function, as Figure 3.107 shows.

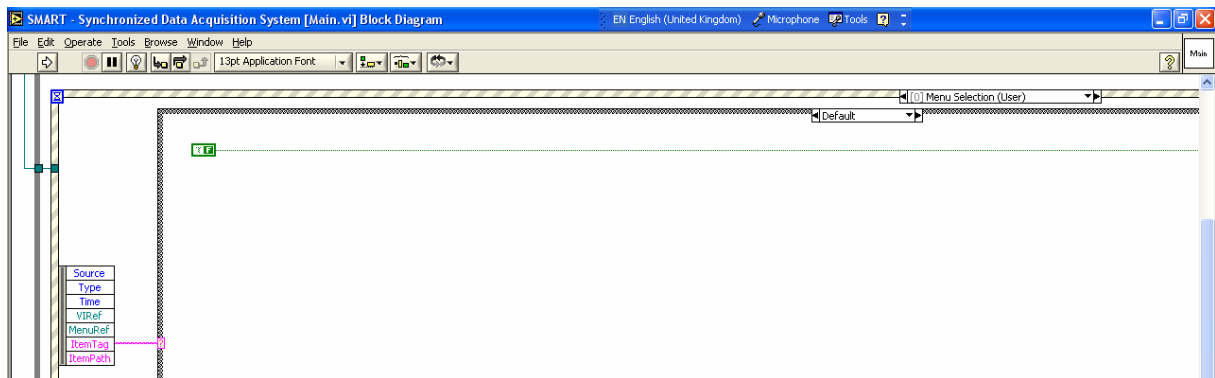


Figure 3.107: Illustration of the code that shows the item “Default” of the “Menu Selection” that does not do anything if no other event happens

The next frame of the sequence of events is “Channels Button”, as Figure 3.108 shows. The window “Channel” (Figure 3.109) is used for selecting the channels of which signals will be plotted during the acquisition. If it is cancelled and there is no signal already acquired, nothing happens. Otherwise, the signals related to the selected channels (variable “User Channels”) are extracted from the buffer of signals (variable “Signal”) and plotted.

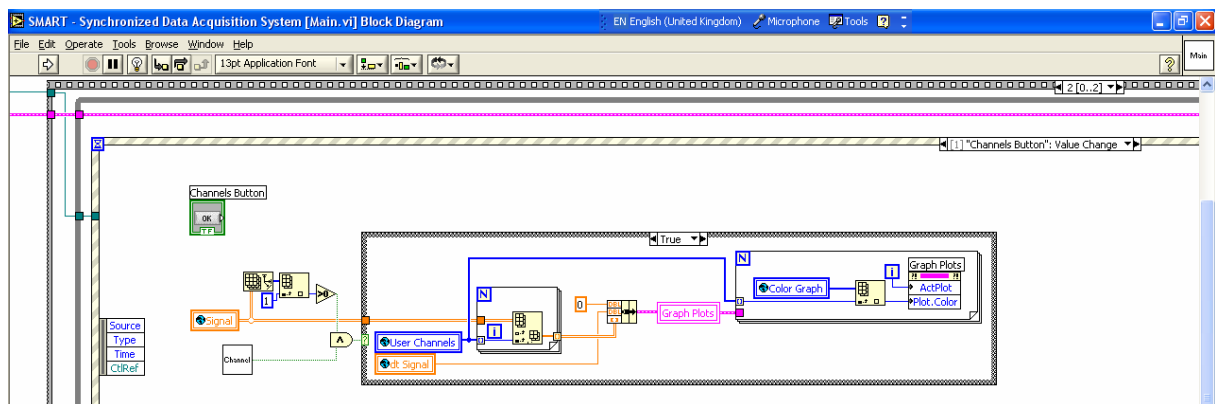


Figure 3.108: Illustration of the code that shows the event “Channels Button” that calls the function “Channel” and sets the colour of the plots on the main window

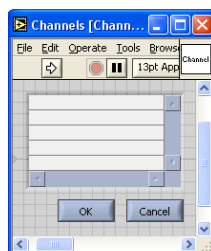


Figure 3.109: Illustration of the window “Channels”, referred in the project as function “Channels”

When the window “Channel” is opened, the last configuration has to be shown. For this purpose, the window “Channel” is firstly cleaned and only the names of the channels are kept, as shown by Figure 3.110. Then, the selected channels receive a “check” symbol indicating that they are selected. A sequence of colours (size equal to the quantity of available channels) is built. Each channel receive a colour which will be the same used for plotting its acquired signal, as Figure 3.111 shows.

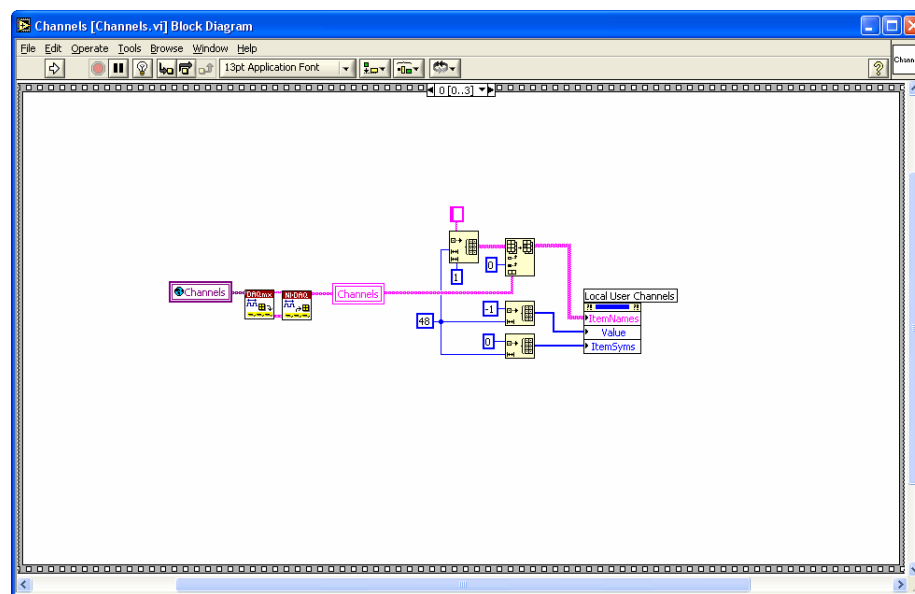


Figure 3.110: Illustration of the code of the function “Channel” that cleans the object “Local User Channels”, responsible for listing the names of the set channels

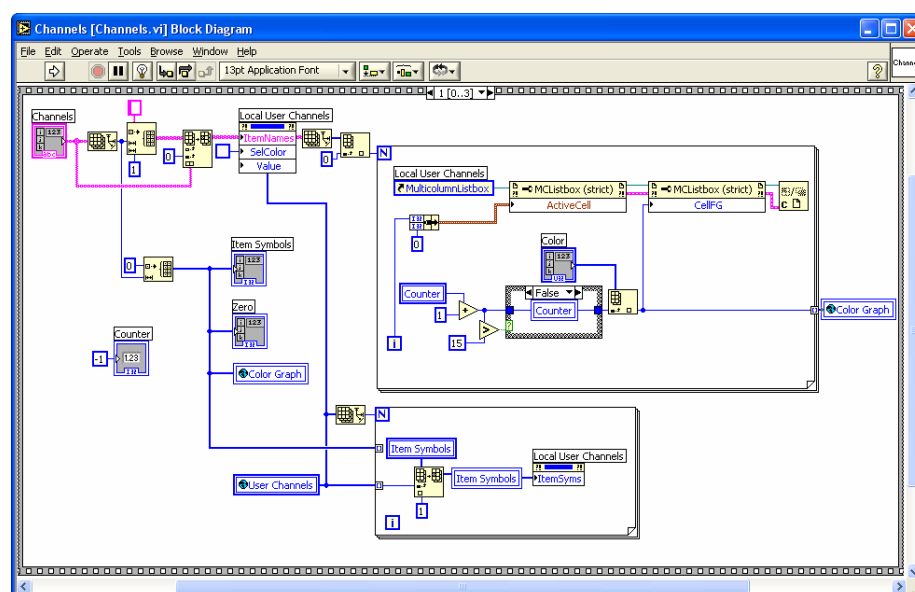


Figure 3.111: Illustration of the code of the function “Channel” that sets the object “Local User Channels”

Then, the current selected channels are saved just in case the selection of channels is cancelled later, as Figure 3.112 illustrates. After that, an event is expected. If an item is selected, a “check” symbol is used. If the “OK” button is pushed, the current “User Channels” are updated. Otherwise, if the “Cancel” button is pushed, the previous configuration is recovered. Figure 3.113, Figure 3.114 and Figure 3.115 show how it works.

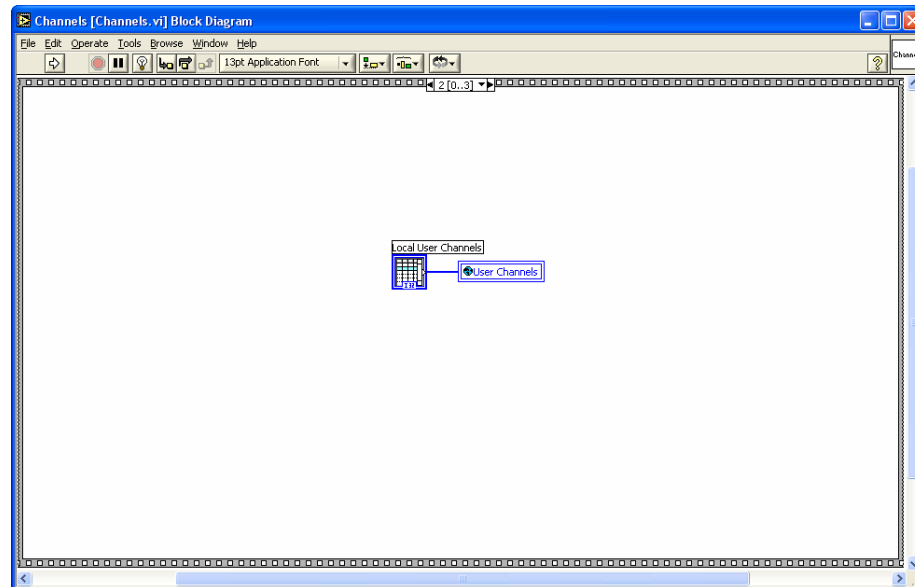


Figure 3.112: Illustration of the code of the function “Channel” that saves the current selected channels just in case the selection of channels is cancelled

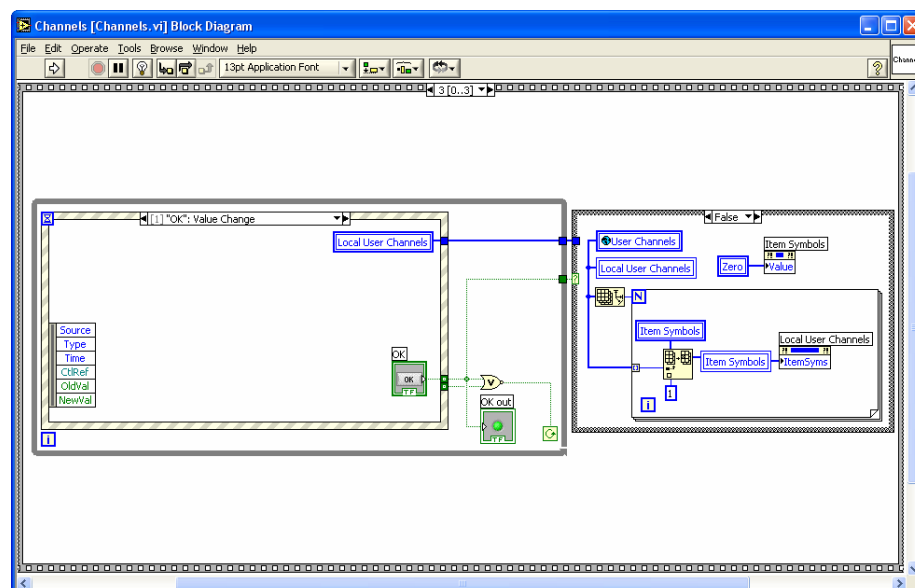


Figure 3.113: Illustration of the code of the function “Channel” that upgrades the global variable “User Channels”, if the button OK is pressed

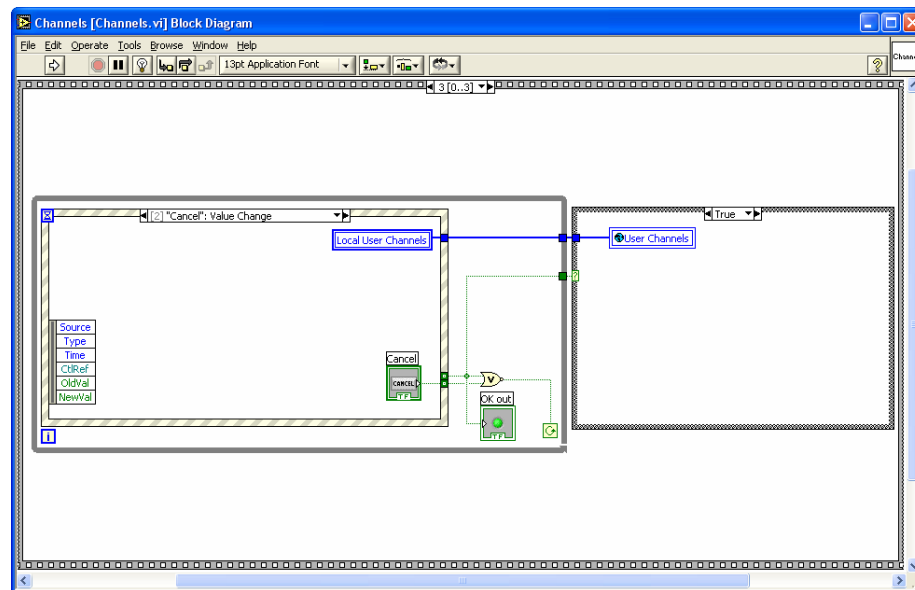


Figure 3.114: Illustration of the code of the function “Channel” that rewrites the original selection of channels on the object “Local User Channels”, if the button Cancel is pressed

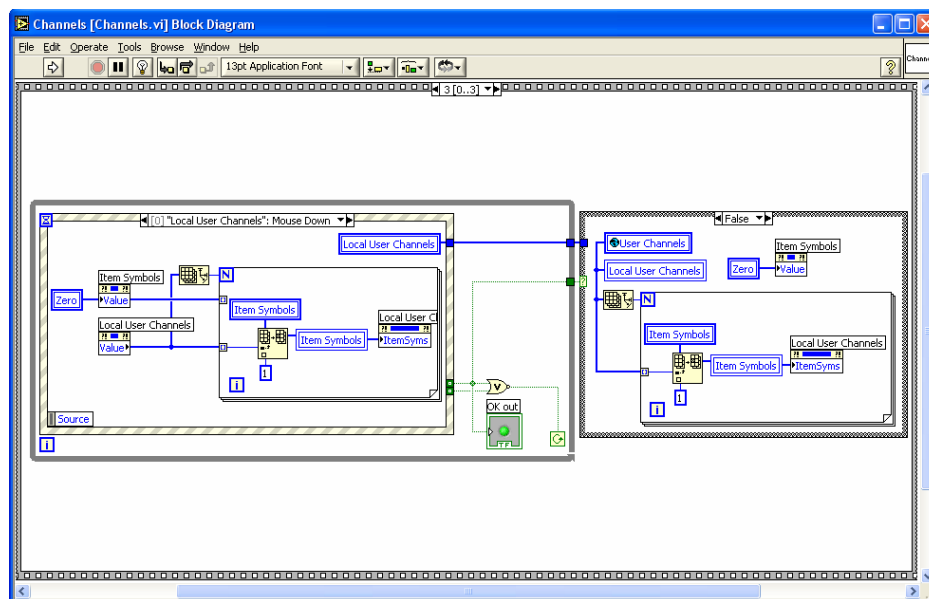


Figure 3.115: Illustration of the code of the function “Channel” that puts a check mark in front of a selected channel, if the object “Local User Channels” is clicked

The next event is “Start”, responsible for doing the data acquisition. First, the current project is deleted, as shown by Figure 3.116. Then the function “Create” (Figure 3.117) checks if txt, xls or avi files, which belong to the current project, already exist. In a positive case, they are also deleted. Eventually, paths for these files are created, as Figure 3.118 shows.

Then, the items of the menu on the window “Main” are disabled, as well as the “Start” button and the buttons related to the cameras. The other objects are reset and the variables related to the acquisition configuration are saved into the project file following the order below:

- order: it is the order of the low-pass digital filter;
- fl: it is the cut-off frequency of the low-pass digital filter;
- Camera: it is the position of the webcam in the listed USB cameras plugged to the computer;
- Trigger: it is the name of the channel used for triggering the high-speed camera;
- Pulses: it is the number of pulses necessary to trigger the high-speed camera (it is always equal to 1);
- Acquisition Mode: it can be continuous or finite, assuming the values 10123 or 10178, respectively;
- Clock Type: it can be internal or external, assuming the values 0 or 1, respectively;
- Active Edge: it can be rising or falling, assuming the values 0 or 1, respectively.
- Sample Time: it is the time in seconds case the acquisition mode is finite; it assumes discrete values between 1 and 60 seconds.
- Rate: it is the sampling rate per channel.
- Clock Source: it specifies the channel for the sample clock if an external clock is used;
- Channels: these are the channels used for doing the acquisition;
- User Channels: these are the selected channels to be plotted;
- Color Graph: these are the colours used for the selected channels.

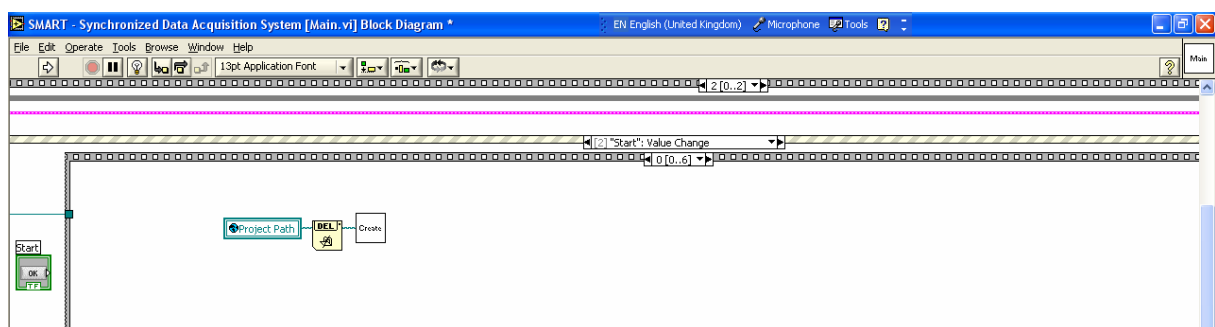


Figure 3.116: Illustration of the first frame of the event “Start” that deletes the current project

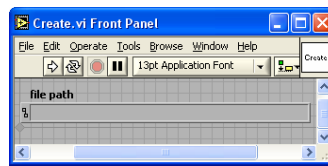


Figure 3.117: Illustration of the front panel of the function "Create"

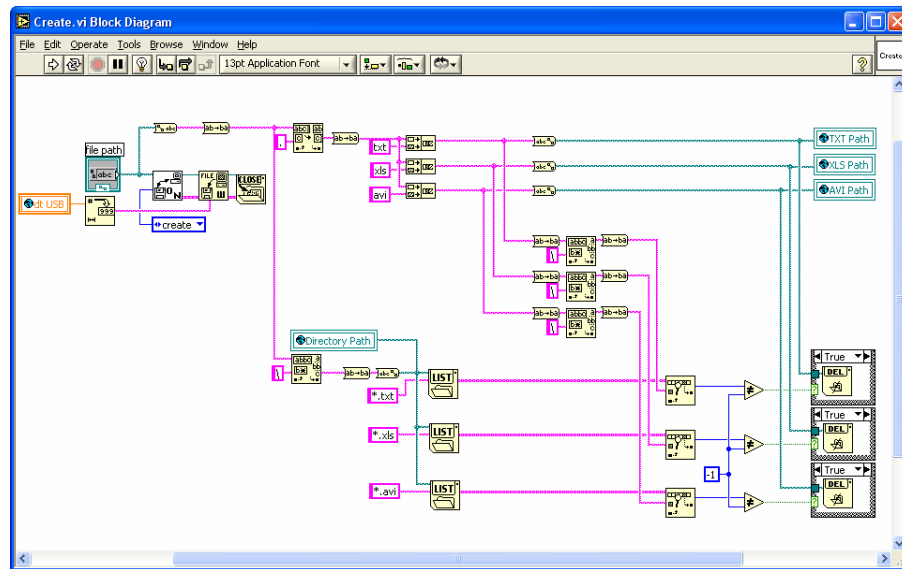


Figure 3.118: Illustration of the code of the function "Create" that creates paths for txt, xls and avi files of the current project

In order to avoid any kind of interference during the acquisition, all the possible combinations of types of execution had to be previously done. Thus, the variable "Execution" was created. If there is no webcam plugged, trigger for the high-speed camera is not solicited and there is at least one selected channel, this variable becomes equal to 3, as Figure 3.119 shows. If there is a webcam plugged, "Execution" becomes equal to 4, as Figure 3.120 illustrates. If the USB camera is not plugged but the trigger is solicited, this variable assumes value 1, as shown by Figure 3.121. If the webcam is plugged, the trigger is solicited and there is at least one selected channel, this variable assumes value 2, as Figure 3.122 shows. The default value for this variable is 0.

[illegible]

Figure 3.120: Illustration of the second frame of the event “Start” that sets the variable “Execution” with value 4

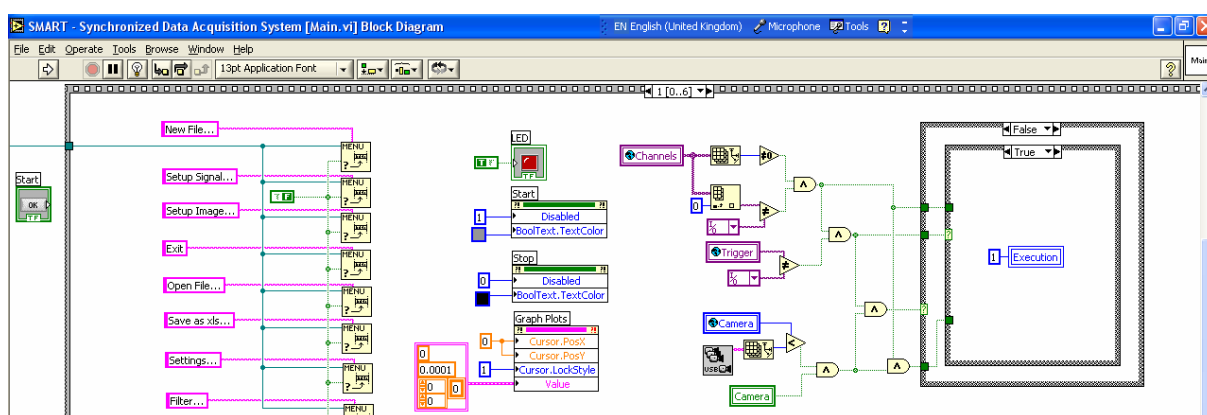


Figure 3.121: Illustration of the second frame of the event “Start” that sets the variable “Execution” with value 1

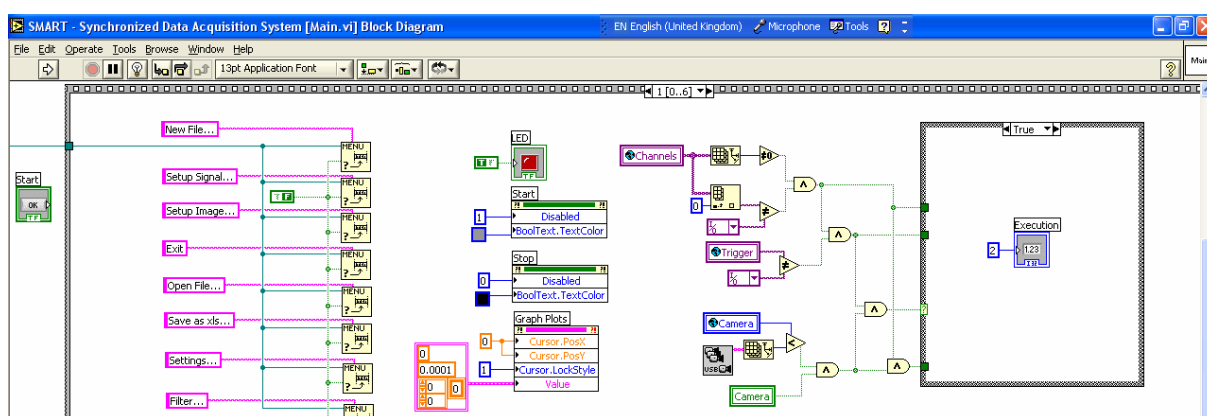


Figure 3.122: Illustration of the second frame of the event “Start” that sets the variable “Execution” with value 2

After setting the variable “Execution”, if its value is equal to 0, nothing happens, as Figure 3.123 shows. Otherwise, the type of execution is selected and the acquisition starts. The program is divided into 3 threads: one responsible for webcam images acquisition, another one responsible for reading data from the acquisition board and a main one responsible for synchronizing the two first threads and plotting the signals and images on the window “Main”. The communication among them is done by notifiers. This procedure is shown by Figure 3.124.



Since a webcam frame is acquired every 0.1 second (10fps), a buffer of signals has to be read into the same interval of time. Function "Image1" (Figure 3.125) is responsible for doing the acquisition when the variable "Execution" is equal to 1. After setting the necessary variables (Figure 3.126) and checking the acquisition mode, which can be finite or continuous, the data acquisition starts, as Figure 3.127, Figure 3.128 and Figure 3.129 show. If the acquisition is continuous, the button "Stop" has to be pushed in order to stop the acquisition. If the acquisition is finite, the button "Stop" has to be pushed or the "Sample Time" has to be reached.

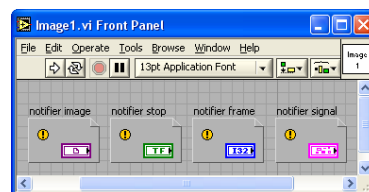


Figure 3.125: Illustration of the front panel of the function "Image1"

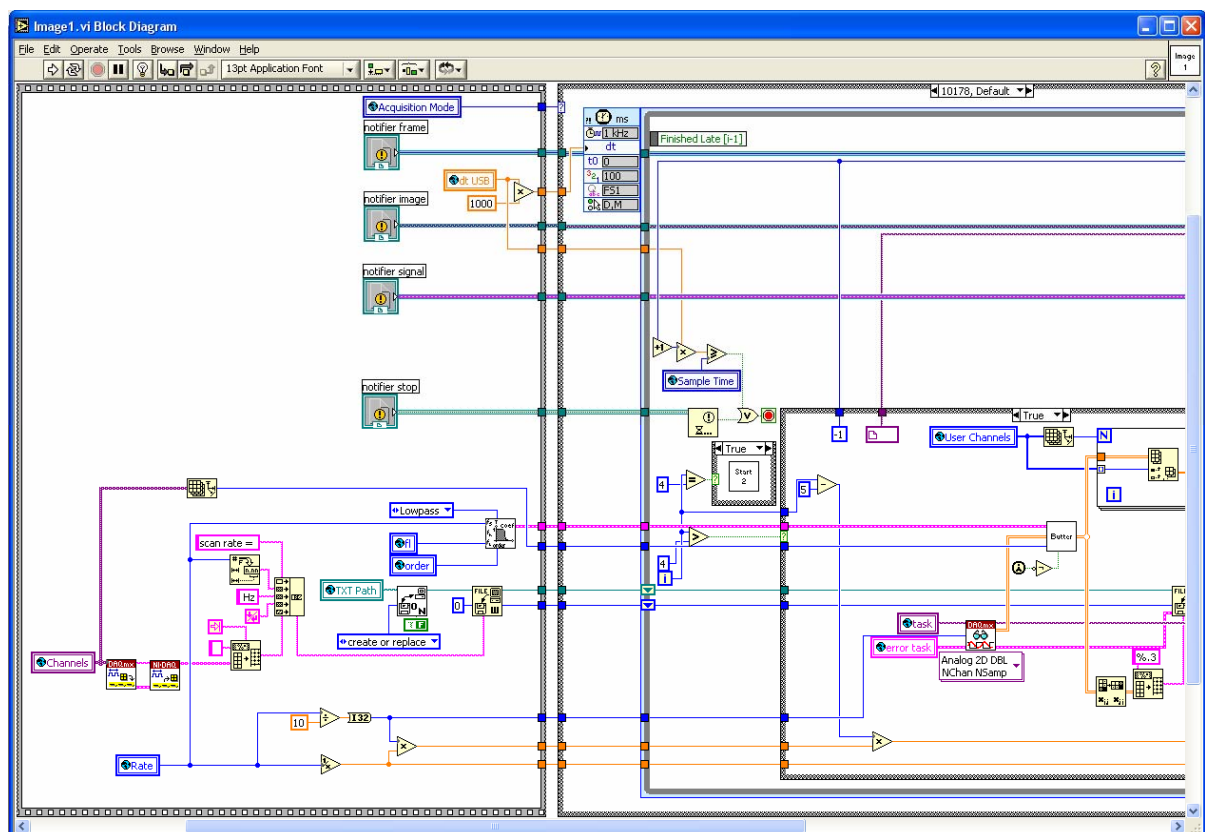


Figure 3.126: Illustration of the code of the function "Image1" that sets the inputs to be used by the functions during the data acquisition

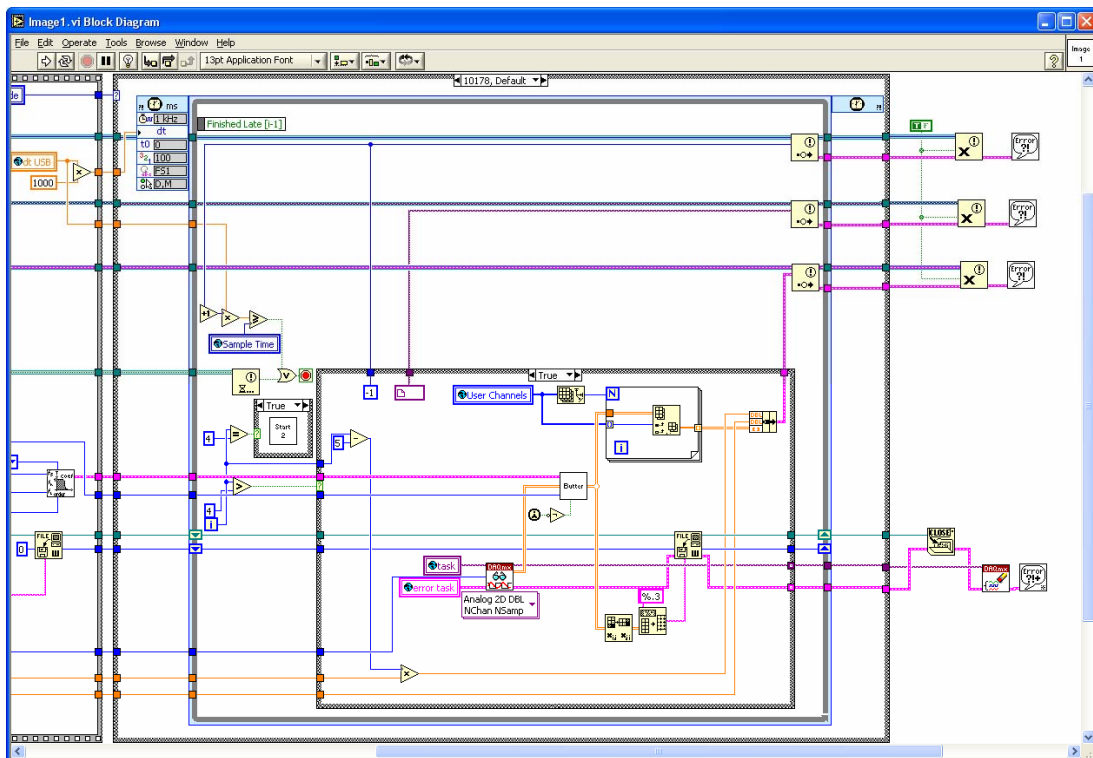


Figure 3.127: Illustration of the code of the function "Image1" that calls functions "Start2" and "Butterworth" and acquires data during a determined interval of time

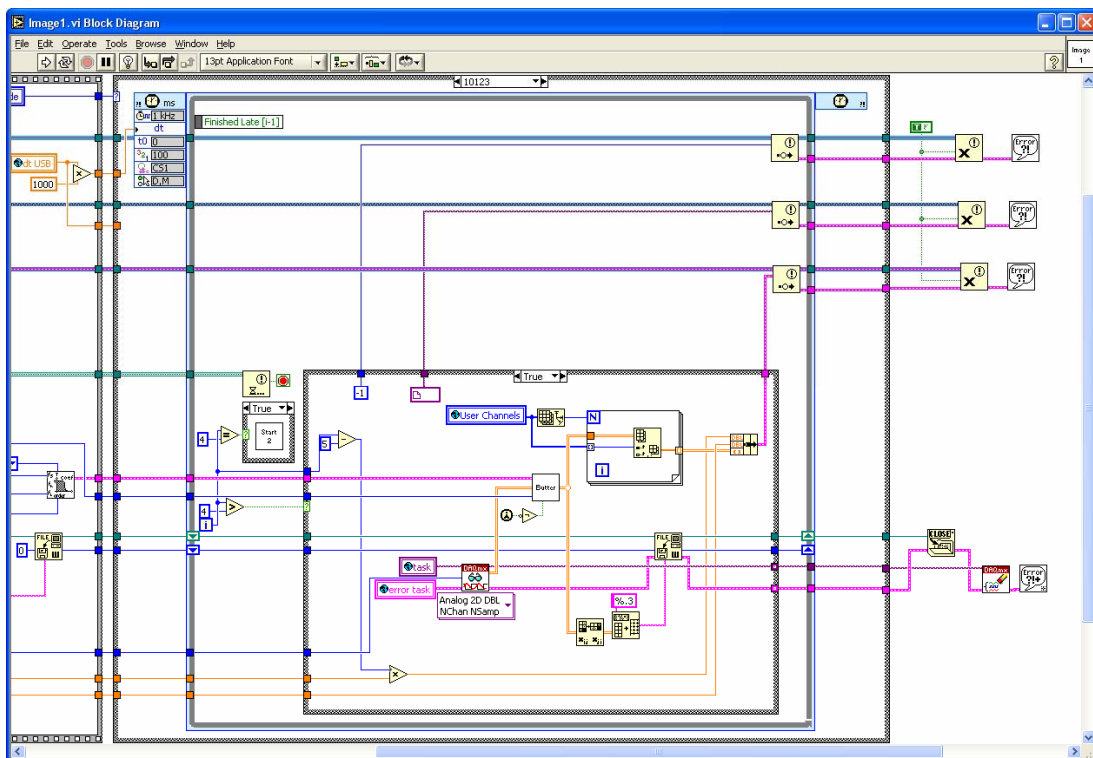


Figure 3.128: Illustration of the code of the function "Image1" that calls functions "Start2" and "Butterworth" and acquires data until the button Stop of the main window is pressed

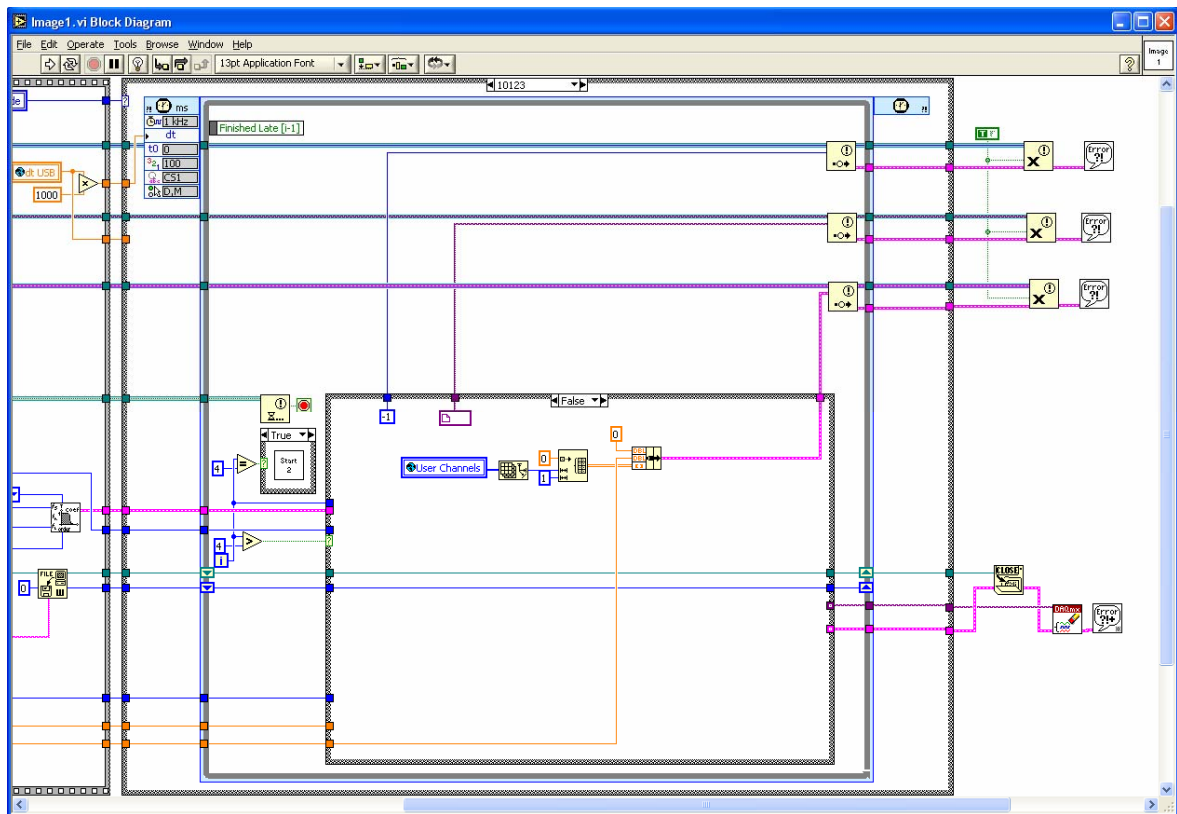


Figure 3.129: Illustration of the code of the function "Image1" that waits for the fifth USB camera frame to start the data acquisition

Function "Start2" (Figure 3.130) starts to acquire the signals and shoots the trigger in order to start the HS camera image acquisition, as shown by Figure 3.131 and Figure 3.132. Being the delay between these two moments deterministic, it is possible to calculate the correct frame to be plotted, during a frame-by-frame analysis.

The function "Butterworth" (Figure 3.133) is a Butterworth low-pass digital filter which filters the acquired data. LabVIEW can filter only one signal each time. Thus, 16 filters were put in parallel and can filter up to 16 channels at the same time, as shown by Figure 3.134.

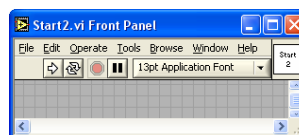


Figure 3.130: Illustration of the front panel of the function "Start2"

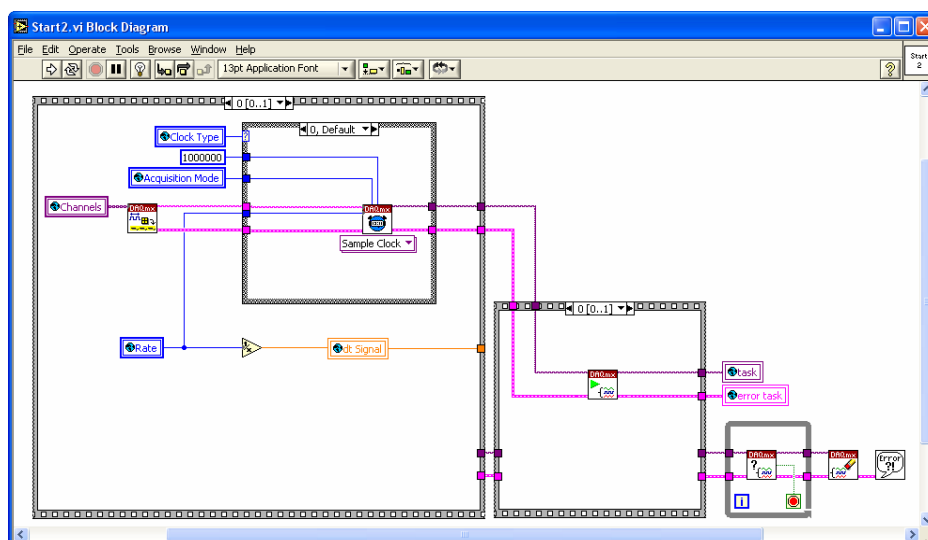


Figure 3.131: Illustration of the code of the function "Start2" that starts reading the inputs of the acquisition board

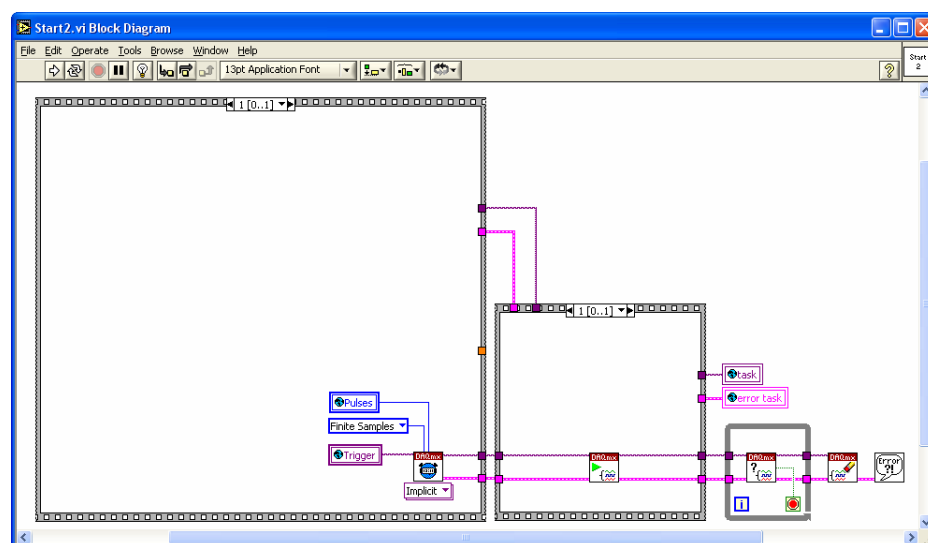


Figure 3.132: Illustration of the code of the function "Start2" that shots the trigger of the HS camera

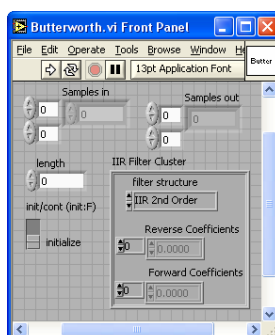


Figure 3.133: Illustration of the front panel of the function “Butterworth”

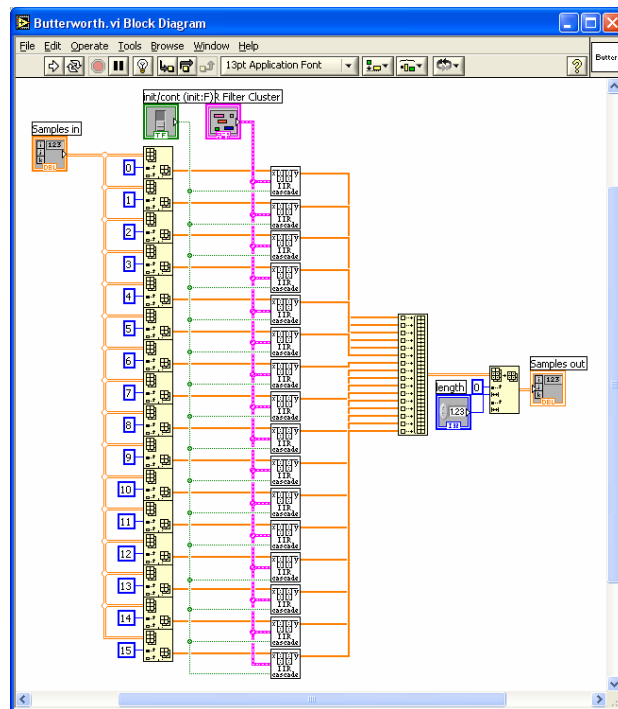


Figure 3.134: Illustration of the code of the function “Butterworth” that filters a data buffer of each input channel

If the variable “Execution” is equal to 2 (Figure 3.135), the function “Image2” (Figure 3.136) does the acquisition. In this case, the USB camera frames are also acquired. Since the webcam presented an irregular behaviour when acquiring the first frames, the acquisition actually starts after disposing the first five ones, as shown by Figure 3.137, Figure 3.138, Figure 3.139 and Figure 3.140. It assures a regular quantity of brightness and a rate of 10fps for all frames. This procedure is done even when no webcam image is acquired, what provides a similar behaviour for all types of executions.

If the variable “Execution” is equal to 3 (Figure 3.141), the acquisition is done by function “Image 3” (Figure 3.142). In this case, only the signals are acquired and the HS camera is not triggered, as Figure 3.143, Figure 3.144, Figure 3.145 and Figure 3.146 show. Thus, the function “Start1” (Figure 3.147) is used for starting the data acquisition, as shown by Figure 3.148 and Figure 3.149.

If the variable “Execution” is equal to 4 (Figure 3.150), the acquisition is done by function “Image 4” (Figure 3.151). In this case, the signals are acquired as well as the USB camera frames, but the HS camera is not triggered, as Figure 3.152, Figure 3.153, Figure 3.154 and Figure 3.155 show.

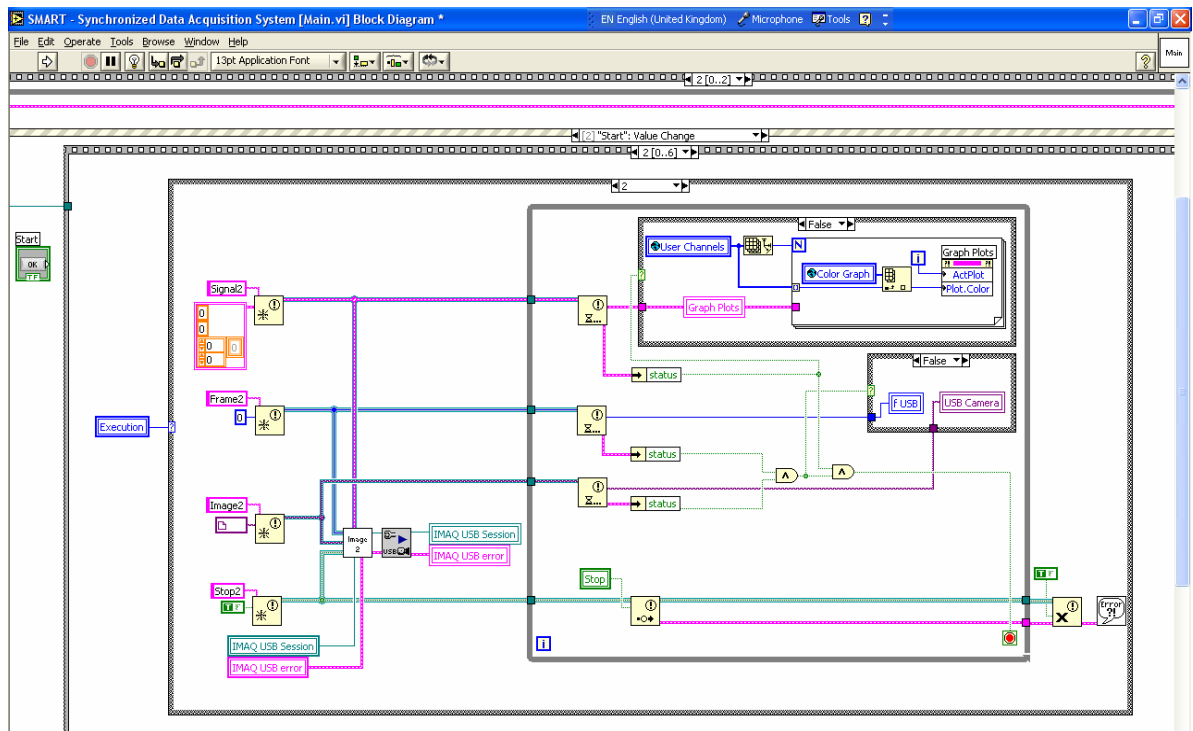


Figure 3.135: Illustration of the third frame of the event "Start" that calls the function "Image2" and plots the acquired data

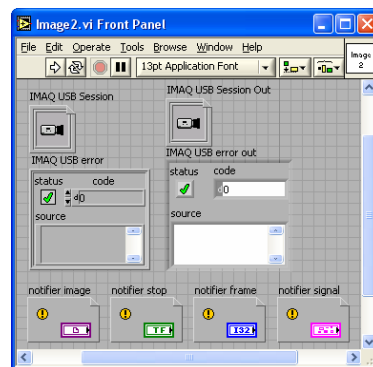


Figure 3.136: Illustration of the front panel of the function "Image2"

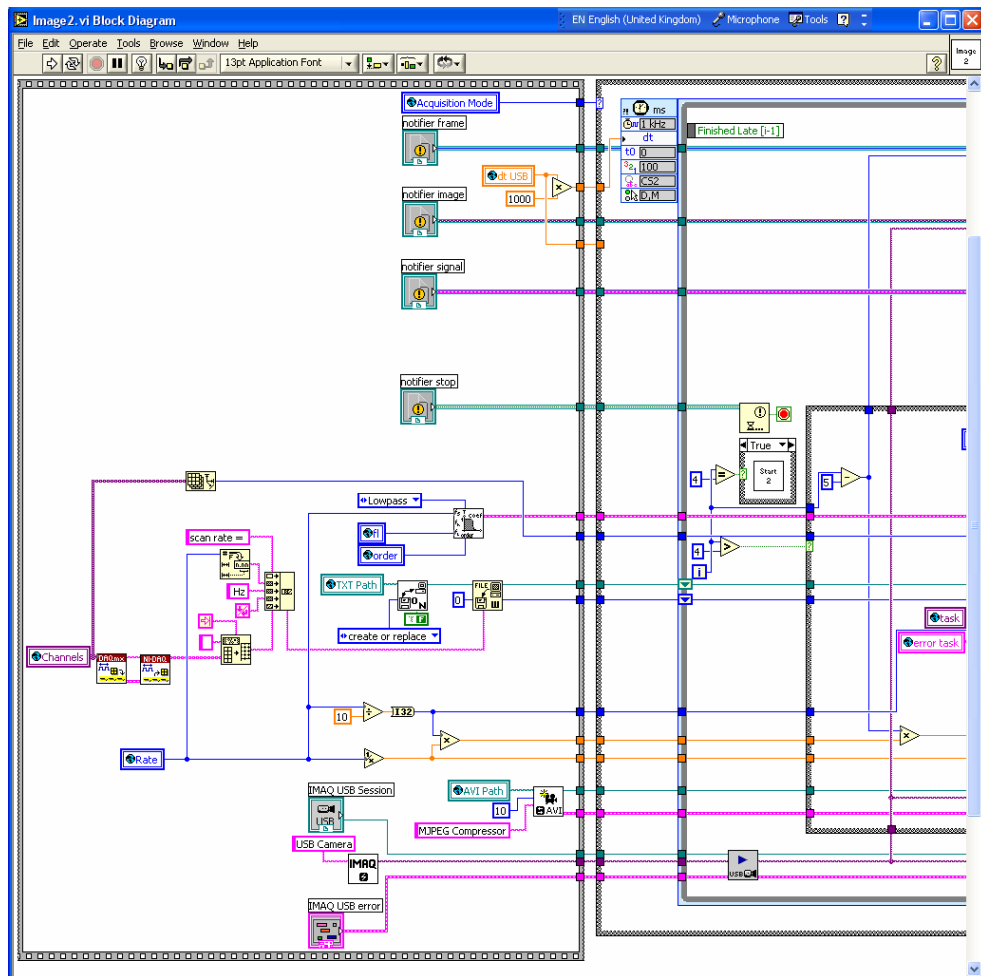


Figure 3.137: Illustration of the code of the function "Image2" that sets the inputs to be used by the functions during the data acquisition



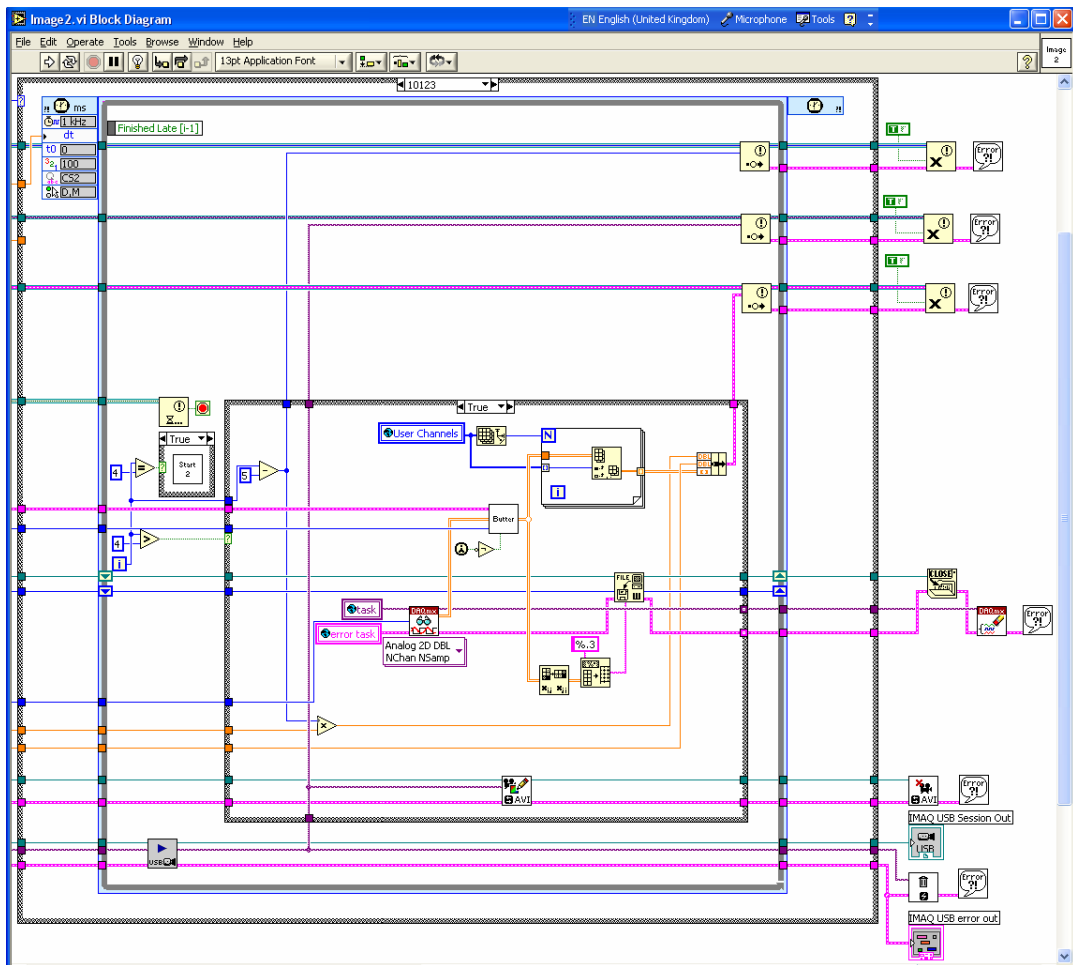


Figure 3.139: Illustration of the code of the function "Image2" that calls functions "Start2" and "Butterworth" and acquires data until the button Stop of the main window is pressed