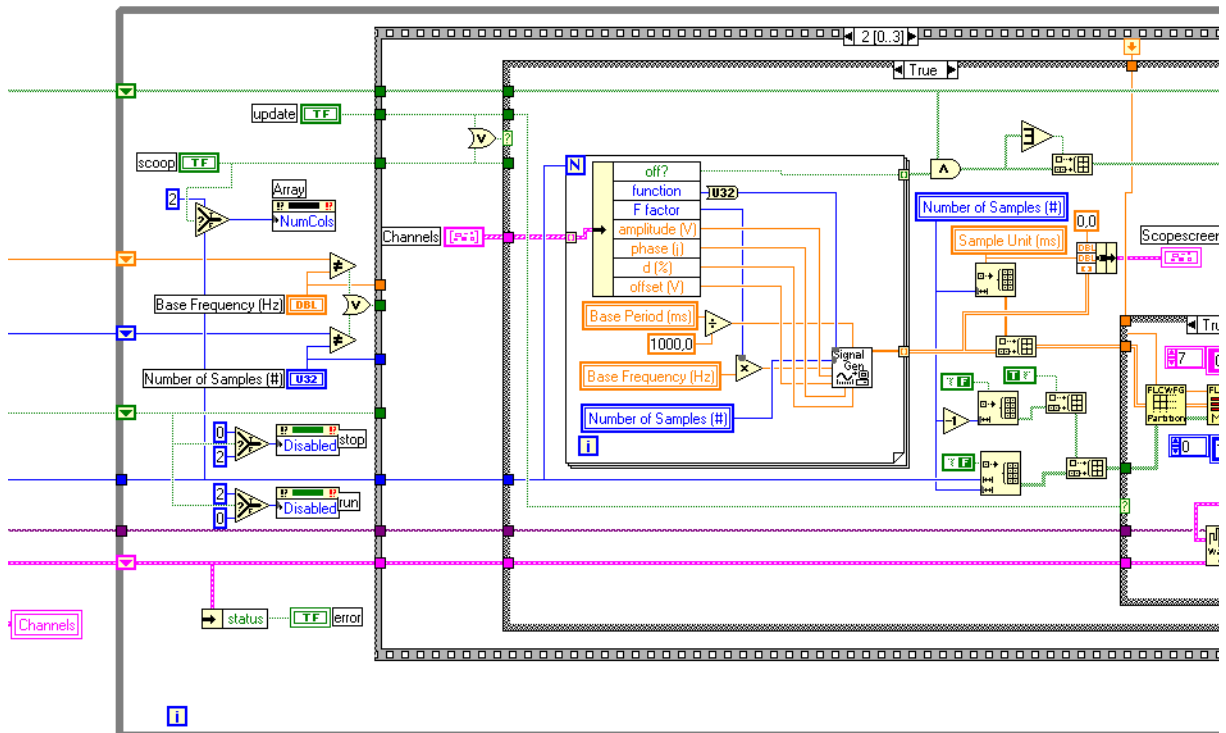


# LabView<sup>®</sup> drivers for WFG 500



## Compatibility and platform issues

We recommend that you use LabView version 5.1.1 (version 5.1 with free upgrade to 5.1.1) or a later version of LabView for your programs. Although all driver VIs are platform independent, some cosmetic changes may occur when you port VIs from one platform to another, especially text may appear differently.

## Installation of the driver package

1. First shut down LabView if it is running.
2. Proceed by copying or moving the folder *FLCWFG 500* to the instrument library (*instr.lib*) of your copy of LabView.
3. Once you restart Labview a menu will be created in the instrument library pallet as well as all submenus.
4. Before you start working with the drivers for the first time, chose *mass compile* from the file menu and select the folder you just installed.
5. That's all.

If this is not the case, you see only question marks in the submenu, then most likely there is an error with the help system or some VIs are not recognized in the driver library, try reinstalling the folder or download a new one. **For the Macintosh:** if you are unable to read any on-line reference help, you probably have two conflicting versions of *Quickhelp*<sup>™</sup> on your computer. Initiate a search and try to remove any version which is not located in LabView's *help* folder. Once you restart LabView the help system should work perfectly, you should now see only icons in the submenus.

If only a few question marks appear then some VIs may have been damaged or deleted. In any case download the latest version and reinstall the folder as described above.

All future versions of the driver library will contain all previous VIs to assure backward compatibility.

## Basic idea behind the driver set

Creating modular programs such as subvi's is something entirely different from creating a stand alone program such as that delivered with your WFG 500. Extreme care has been taken to make all VIs robust but at the same time execution speed has been an important factor. The communication with the waveform generator is based on the VISA system.

As for handling the data in LabView the best way to represent a waveform is in a two dimensional array. In this way one immediately avoids the chance that individual waveforms and timing contain different amounts of data. Another important issue is that one should work with the 'real' values, that is true times, e.g. microseconds, and true voltages. A number of VIs are included that perform conversion of the data to and from this format as well as VIs which provide easy tools to handle the data.

This version of the drivers does not yet support the use of couplings. When you import data from a file the couplings will be ignored.

## Getting started

The first thing is to open the communication channel with the generator, that is initialize the desired serial port. This is done by calling the VI *FLCWFG Initialize.vi* and setting the appropriate parameters. Figure 1. shows you the layout of a basic program.

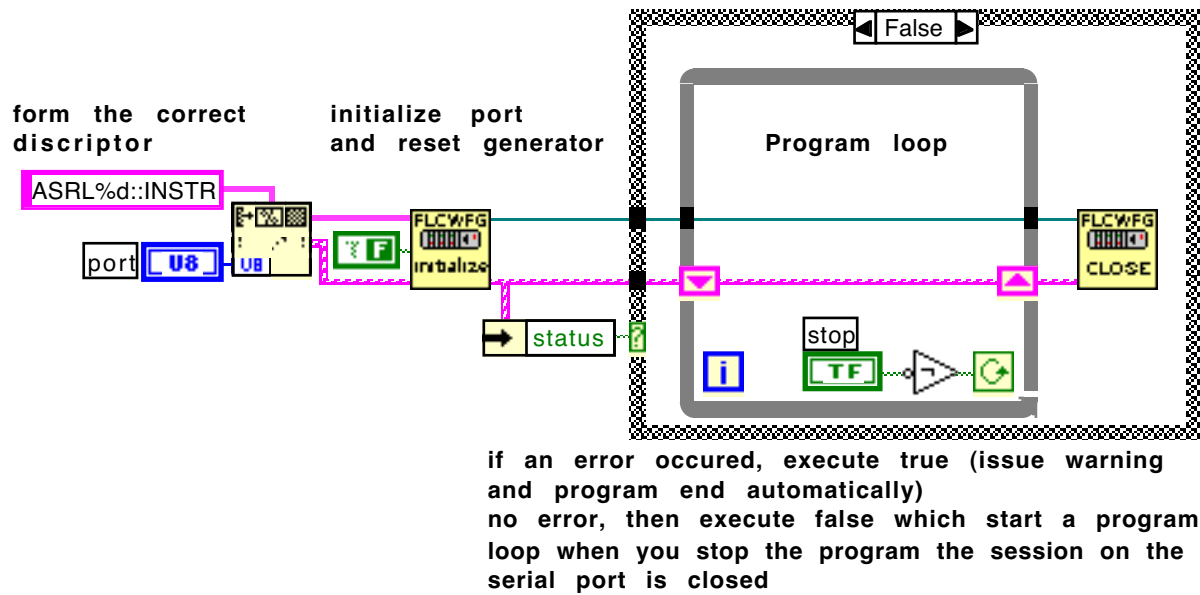


Figure 1. Basic program layout

The error handler contains only ‘serious errors’ that means those that will endanger the communication. Most errors that occur during operation are soft-solved and will not interfere with your program, that is if you do not specifically want them to change your program flow.

After you have decided to quit the program you must close the VISA session of the serial port.

You can control more than one generator from the same LabView program. All you have to do is initialize two (or more) serial ports and separate the operations with them.

## Importing data from files

The VI library contains a file filter *FLCWFG File Filter.vi* which extracts all relevant information from files created with the normal WFG programs on Mac and PC, however the current version does not extract the coupling data. Most users have no use for all of this information in a LabView program, therefore a translator is provided, *FLCWFG File Data Translator.vi* that will create a cluster containing all necessary information for handling the waveforms. It is such a cluster which is “send ready”, that is all information to unambiguously send the data to the

generator. As mentioned above, the best way to handle the real data is in a 2D array, *FLCWFG Unmold Data.vi* does just this starting from the full data cluster. Figure 2. shows the call chain as explained above.

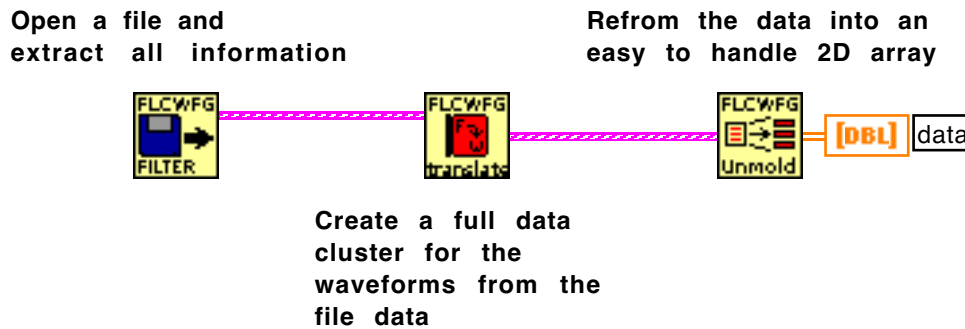


Figure 2. Call chain for extracting data

Once you have a 2D array you are ready to perform all sorts of operations on it. In addition to the data array, an array containing all trigger-bit and control-bit information with the same dimension as the data array is available as well as arrays containing the waveform names and their channel data and the time unit used for these waveforms, e.g. 0,1  $\mu$ s for the 10 MHz clock.

The data array is structured as rows, that is the first row (first index=0) contains the timing data in microseconds, all subsequent rows contain the voltage data of a waveform. The above layout of the data array makes it possible to access timing and individual waveforms subsequently by wiring it to a for loop. Should you wish to access a column of pulses simultaneously you can first transpose the array and then wire it to a for loop. To view the data in a graph use *FLCWFG Build Graph Data.vi* to obtain the correct LabView format for graphs, set the graph to 'square interpolation' with the jump on the first point (or copy it from the VI).

## Handling waveform data

One of the major differences with working with the 2D arrays and the full data clusters is that the timing is in true values. We recommend that you use microseconds throughout your programs and convert appropriately when desired. In doing so one will encounter situations where a pulse as entered in the 2D array can no longer be represented by a single slot on the generator. As an example consider that you have selected the 10MHz clock and want to create a pulse of 5 ms that is 50000 clock units, then one needs to split this pulse into two slots. As a user you do not want to be bothered with this and hence *FLCWFG Partition Raw data.vi* does the job for you.

A situation as described above can appear quite easily when you scale timing data, which can be done with *FLCWFG Scale Data.vi*. This VI takes a set of scaling factors and changes the 2D data accordingly, but be sure to partition the data before you mold it back into a full data cluster which is needed for downloading operations. Figure 3. shows you a call chain describing the above situation.

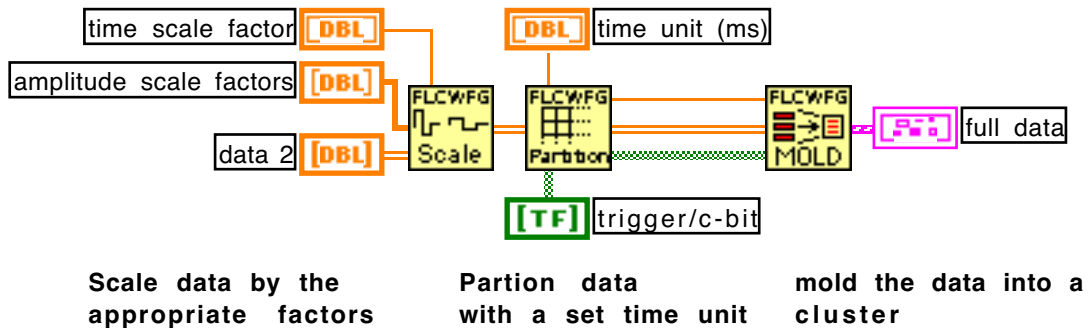


Figure 3. Call chain for scaling and molding

### Loading waveforms into the generator

Once you have composed a full data cluster ready to send, you use *FLCWFG Download Waveforms.vi* to get them into the generator. Everything will be taken care of to get all information into the generator, including timing. An important part of the information is the assignment of the channels to which each waveform should be written. This is an array of integer elements of a size equal to the amount of waveforms. The integer itself is a translation of the bit pattern to a number. This means that you add powers of two to control the channels, e.g.

- 2 (binary 00000010) means channel 2,
- 3 (binary 00000011) means channel 1 and 2,
- 4 (binary 00000100) means only channel 3 etc.

A lot can happen when you handle data and related information, therefore one should be careful when making changes. A dedicated VI is included to perform a standard check of the waveform cluster. *FLCWFG Check Waveforms.vi* will also warn you when actions have been taken to correct errors or will prompt you for setting the channel data.

As with all VISA based communication you need to wire the descriptor and preferentially also the error cluster. Figure 4. shows a call chain for initializing, sending and closing.

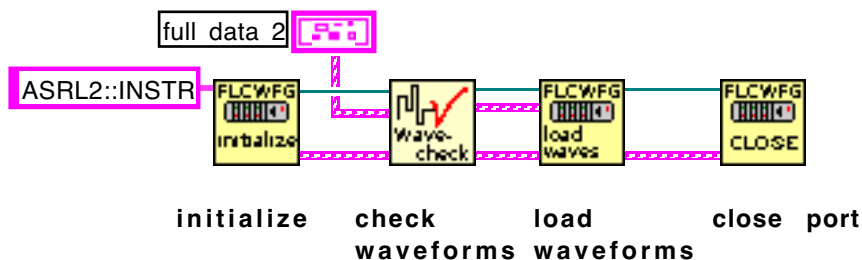


Figure 4. Call chain for sending data

For running and stopping the waveforms one just calls the subvi's *FLWFG Run.vi* and *FLCWG Stop.vi* respectively.

Setting the clock is usually done during the download action when the VI decides from the time unit which clock should be set. You can change the clock using the VI *FLCWFG Clock Setup.vi* and adjust the software clock using *FLCWFG Set Software Clock.vi* should this be necessary.

As a default the run mode is set to continuous and internal triggering, using the clock VI one can change the triggering or run mode. The VIs *FLCWFG Normal Burst.vi* and *FLCWFG Burst and Invert.vi* set the appropriate burst mode.

Most of the other VIs work behind the scenes. If you would like to see which VIs are called during the execution of your program, open the *FLCWFG Log Global.vi*. Each VI sends a text to this log when it is executed along with error information if applicable.

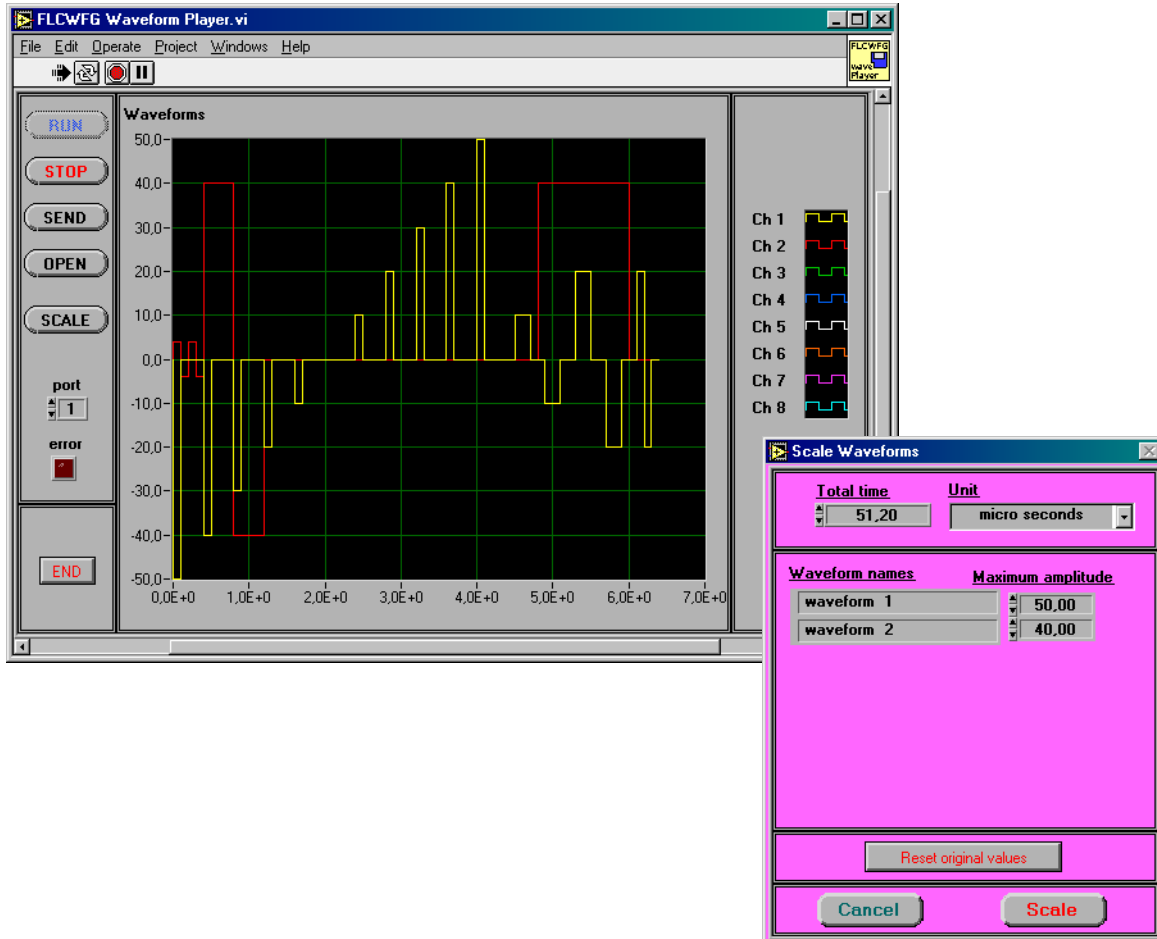
# Application examples

**FLCWFG Waveform Player.vi**

**FLCWFG Frequency Generator.vi**

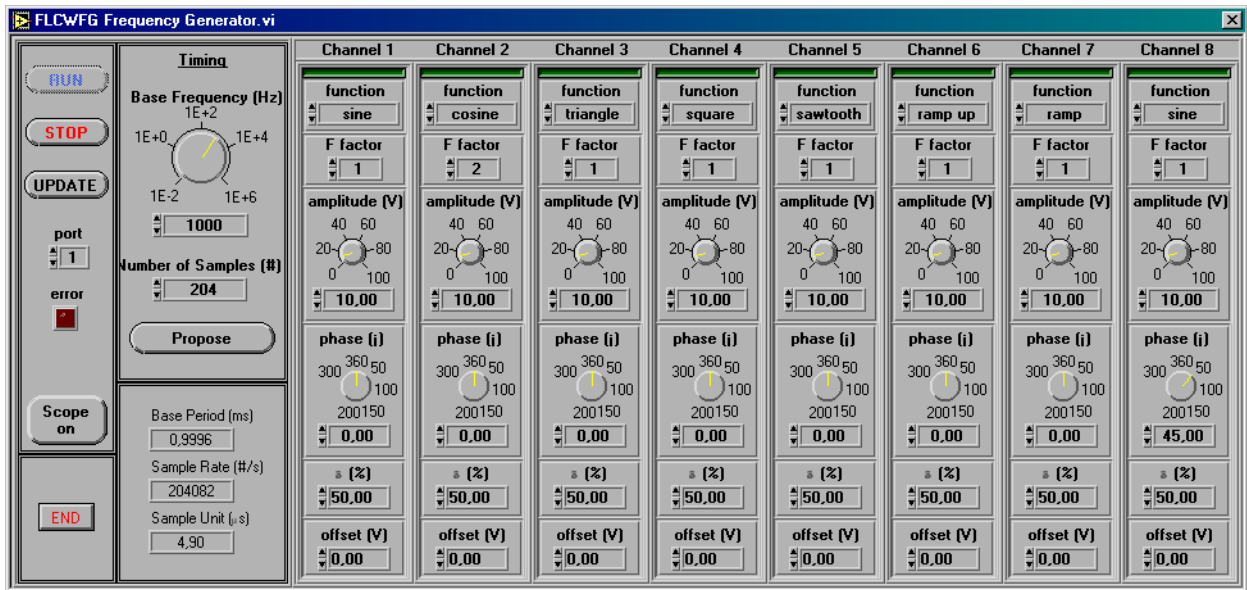
**FLCWFG Waveform Editor.vi**

## FLCWFG Waveform Player.vi

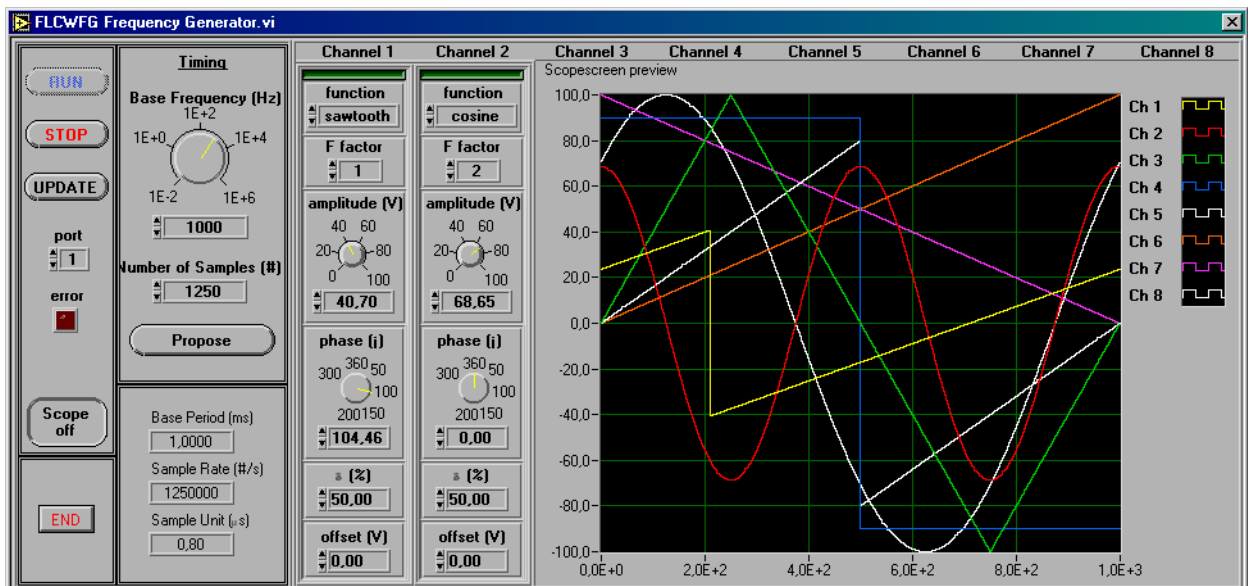


*FLCWFG Waveform Player.vi* is a program that enables you to open a file created with a normal WFG500 program (Mac or PC), send it to the generator, run it and rescale timing and waveform amplitudes.

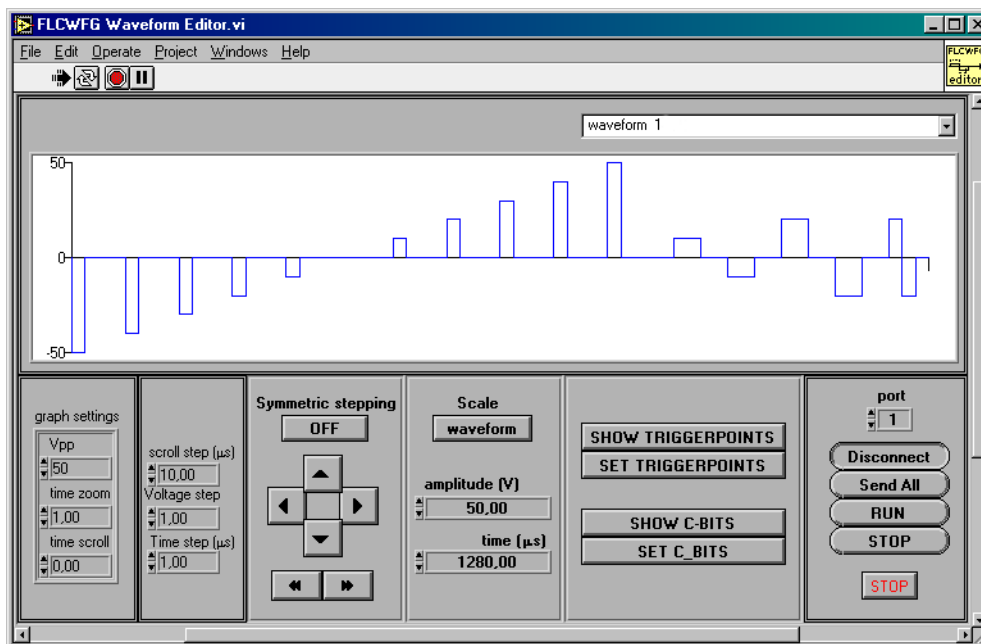
## FLCWFG Frequency Generator.vi



*FLCWFG Frequency Generator.vi* is a much more elaborate program. It simulates a frequency/phase generator using the WFG 500. Before it runs it checks which channels are available. You will see a green indicator above those channels. For each channel you can select the desired waveform such as sine, triangle, square etc. and its parameters. The frequency is set on the Timing panel together with the number of samples you wish to use. Keep in mind that larger amounts of samples not only take longer time to generate, but also to send. As all channels have a common time base, the frequency difference between them must be an integral number. You can set this using the frequency factor of each channel. Next, you can control all amplitudes separately, as well as the phase and offset. The duty cycle control works with the square wave only. A trigger point is set at the beginning of the base period. The button *Propose* will give you the best approximation of the desired timing. To send the information to the generator click the button *Update*. Should you want to see the data in a graph click *Scope on*.



## FLCWFG Waveform Editor.vi



*FLCWFG Waveform Editor.vi* is a rather elaborate program. It allows for selecting pulses on a graph using a mouse. If you click a pulse it will be selected and show up red. If you click and drag you can select a range of pulses. When you hold down shift and then click a pulse you add the just selected pulse to the already existing selection, shift click drag adds a range of pulses. If you shift click an already selected pulse it will be removed from the selection. Clicking left from the voltage axis will deselect all. The *amplitude* and *time* controls contain the total time of the waveform and the maximum amplitude encountered in it if *scale* is set to “Waveform”. If you chose “selection” for *scale*, then the value of the first pulse in the selection will be visible. Changing anything in these two controls will immediately scale either the entire waveform or just the selection. To the left of the scaling section you find stepper control buttons. The individual step sizes are listed to the left of the stepper buttons and can be altered at your convenience. An extra option is included to allow for symmetric stepping, this is similar to scaling but will reduce or increase the amplitude by the step value. If this option is off, then the step value is just added or subtracted from the voltage data. To find the trigger positions click *Show Trigger Points*, to find c-bit positions in the current waveform click *Show C-bits*. To set the trigger points over the entire selection on the graph click *Set Trigger Points*, and similarly, if you wish to set the c-bits click *Set C-bits*.

In order to send anything to the generator you must first connect to it, so click *Connect*. If you are connected, then the button shows *Disconnect* and the buttons *Send All*, *Run* and *Stop* are enabled. If this is not the case then an error has occurred during connecting. To end the program click *Stop*. Any changes you make will immediately be sent to the generator, in this way you can easily optimize time and voltage settings for individual pulses.

Scroll the window left to find controls to change the graph settings; these include the maximum amplitude shown in the graph, the time zoom and the time scroll.

# Alphabetic index

## FLCWFG Add At Index.vi

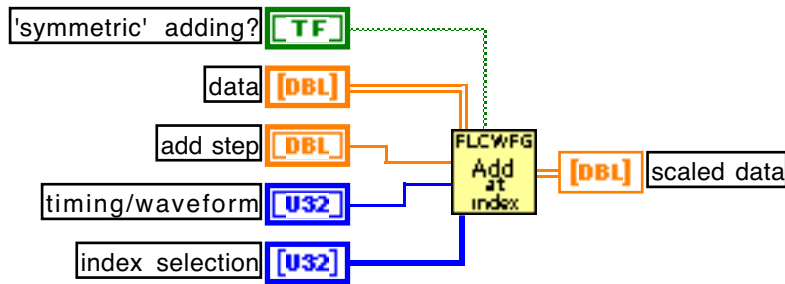


Figure 5. FLCWFG Add At Index.vi

This VI will add the value of **add step** to the data in the selected pulses. Selection of the timing or the waveform is done by setting the desired index through **timing/waveform**, 0 for the timing or the index corresponding to the waveform in the 2D **data** array. **index selection** is an array containing the indices of the pulses you which to alter. The option **'symmetric' adding?** controls how the addition is done. If it is set to *true*, the VI will first check the value at the index, if that is negative the step value will be subtracted if positive it will be added. If you set this option to *false* the value is simply added. **scaled data** returns the modified 2D data array.

## FLCWFG Assign Channels.vi

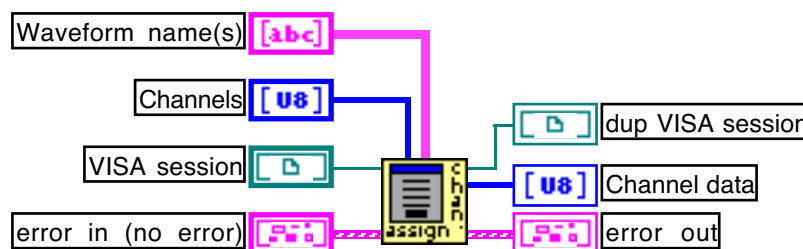


Figure 6. FLCWFG Assign Channels.vi

When this VI is called a pop up window is invoked, prompting you to set the channel data. The VI checks with the generator through the **VISA session** which channels are available and will not execute if an error is reported in the **error in (no error)** cluster. The array **Waveform name(s)** is used to set the possible choices. Through the array **Channels** you can pre-load a certain selection into the window. After the pop up window has closed, the user's choice is returned through the array **Channel data**, which can immediately be used to replace the one in the full data cluster.

## FLCWFG Build Graph Data.vi



Figure 7. FLCWFG Build Graph Data.vi

Starting from a 2D **data** array, this VI will create suitable data structure for visualization in a graph plot. **graph data** is an array of clusters, each representing a plot specified by two arrays of x and y data, respectively. The interpolation style of the plots in the graph must be set to *square* with the y-change on the first point. For the ease of use you can copy the graph from the panel or select *create indicator* from the pop-up menu of this VI in the block diagram.

## FLCWFG Build Picture Data.vi

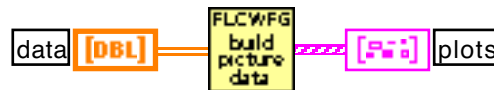


Figure 8. FLCWFG Build Picture Data.vi

Starting from a 2D **data** array, this VI will create suitable data structure for visualization in a picture. **plots** is an array of clusters, each representing an individual plot through an array of points, where a point is a cluster of an x- and a y-value. This conversion is necessary if a picture and its options are to be used in a program.

## FLCWFG Build Timing String.vi

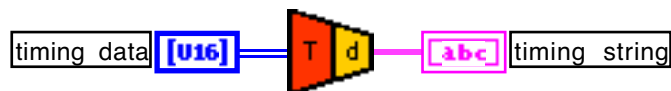


Figure 9. FLCWFG Build Timing String.vi

**timing data** is a 2D array which contains unitized values representing the timing and triggers to be sent to the generator. This structure of the data is used in the full data clusters. **timing string** returns a correct translated bit stream that is ready for sending to the generator.

*Note:*

*you should normally have no need to use this VI directly, as it is called by higher level VIs.*

## FLCWFG Build Waveform String.vi

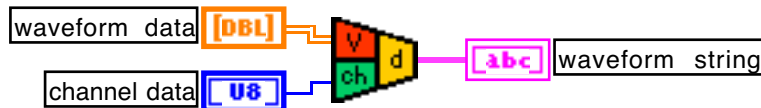


Figure 10. FLCWFG Build Waveform String.vi

**waveform data** is a 2D array containing all information about a waveform’s pulses and c-bits to be sent to the generator. This data structure is partially used in the full data clusters. **channel data** is the value that controls to which channels the waveform should be sent. **waveform string** returns the translated bit stream ready for sending to the generator.

*Note:*

*you should normally have no need to use this VI directly, as it is called by higher level VIs.*

## FLCWFG Burst and Invert All.vi

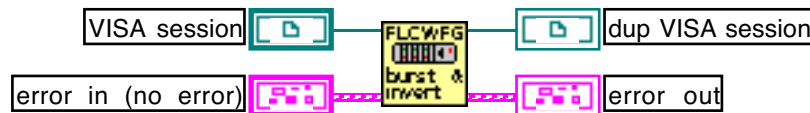


Figure 11. FLCWFG Burst and Invert All.vi

With this VI you can set the mode to *burst and invert all*. If an error is reported in the **error in (no error)** cluster the VI will not execute and just pass along the error cluster and the duplicate for the VISA communication.

## FLCWFG Clock Pop Up.vi



Figure 12. FLCWFG Clock Pop Up.vi

This VI calls a pop up window in which you can set the time unit. The control **Boolean** is only added for program flow control and has no function. The indicator **time unit (ms)** returns the set value in *MICROSECONDS!* (generic creation of names in the block diagram does not support multiple fonts).

### FLCWFG Clock Setup.vi

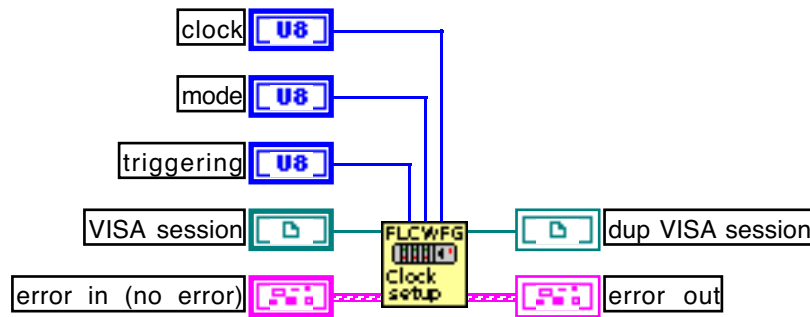


Figure 13. FLCWFG Clock Setup.vi

You can use this VI if you need to set up the clock yourself, usually this done during the downloading action by the VI *FLCWFG Download Waveforms.vi*. **clock** is a menu ring with the following values: [0=10MHz, 1= 1MHz, 2=software clock, 3=external clock]. **mode** is a menu ring with values: [0=contiuous, 1=burst]. **triggering** is a menu ring with values: [0=internal, 1=external]. If an error is reported in the **error in (no error)** cluster the VI will not execute and just pass along the error cluster to **error out** and the duplicate for the VISA communication.

### FLCWFG Close.vi

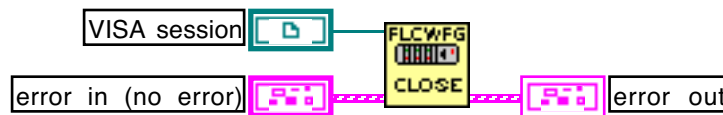


Figure 14. FLCWFG Close.vi

Calling this VI will close the wired **VISA session**.

### FLCWFG Download Waveforms.vi

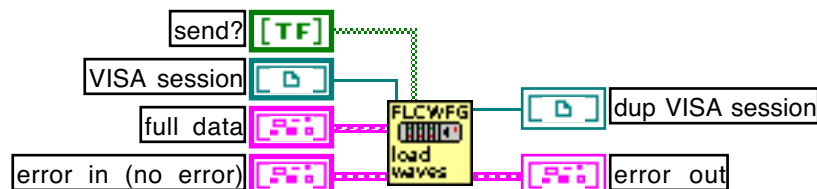


Figure 15. FLCWFG Download Waveforms.vi

Use this VI to download waveform data as a **full data** cluster to the generator. With the array **send?** you can control the individual sending of the waveforms. This array must have number of waveforms plus one elements to be valid, otherwise or if empty all waveforms and timing

are sent. A value of *true* in the array will allow for sending, the first element controls the timing. If an error is reported in the **error in (no error)** cluster the VI will not execute and just pass along the error cluster to **error out** and the duplicate for the VISA communication.

### FLCWFG End Data Transfer.vi

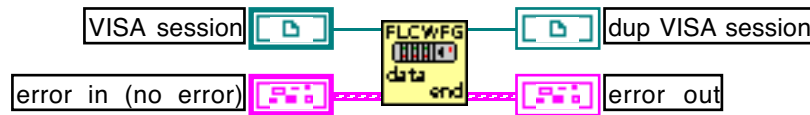


Figure 16. FLCWFG End Data Transfer.vi

This VI ends a data transfer.

*Note:*

*you should normally have no need to use this VI directly, as it is called by higher level VIs.*

### FLCWFG Errorcode Interpretation.vi

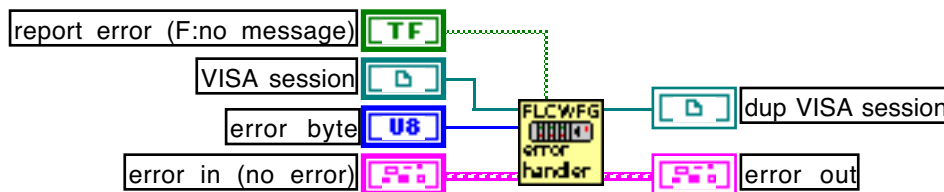


Figure 17. FLCWFG Errorcode Interpretation.vi

This VI is used to check for errors and alleviate them if possible. An error is reported in a message window if **report error (F:no message)** is set to *true* but is always listed in the Log. The **error byte** is the last byte returned by most commands.

*Note:*

*you should normally have no need to use this VI directly, as it is called by higher level VIs.*

### FLCWFG File Data Translator.vi



Figure 18. FLCWFG File Data Translator.vi

Use this VI to obtain a **full data** cluster from the **file data** cluster which is obtained from *FLCWFG File Filter.vi*.

## FLCWFG File Export.vi

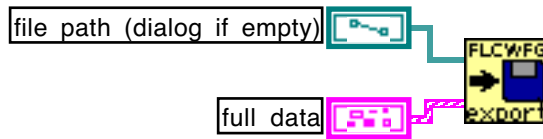


Figure 19. FLCWFG File Export.vi

With this export tool you can save a **full data** cluster as a WFG-style file which can be read by the other programs. If **file path** is left empty or unwired a dialog will prompt you to select or create a file. If you replace an existing file remember that any coupling data that was in the file will be lost.

## FLCWFG File Filter.vi



Figure 20. FLCWFG File Filter.vi

When you want to import data from an existing WFG-style file, use this VI. If you leave **file path** empty or unwired a dialog will appear. If you cancel this dialog, **Canceled?** is *true* and **file data** is empty, otherwise it contains all information EXCEPT couplings.

## FLCWFG Find Index by Time.vi

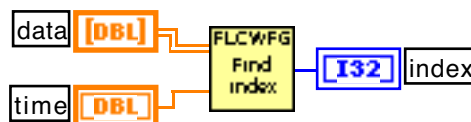


Figure 21. FLCWFG Find Index by Time.vi

This is a versatile searching tool. It will return the **index** of the pulse corresponding to the elapsed **time** in the waveform 2D **data** array. It is for instance used to find the selected pulse from the mouse position in a picture graph.

## FLCWFG Frequency generator.vi



Figure 22. FLCWFG Frequency Generator

This is an application program that simulates a frequency/phase generator.

## FLCWFG Initialize.vi

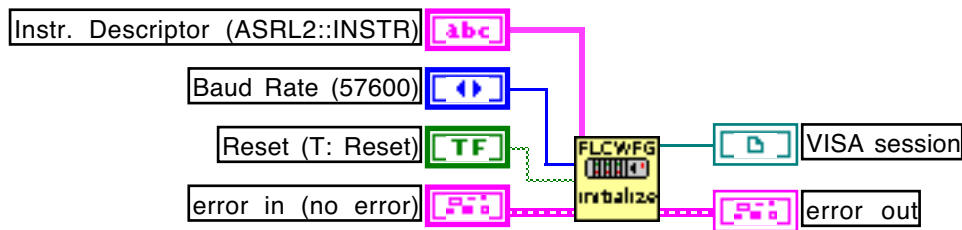


Figure 23. FLCWFG Initialize.vi

Calling this VI will initialize a serial port specified through the **Instr. Descriptor**. The syntax for this string is “ASRL\*::INSTR” where \* represents the port number, for a PC this is equal to the com\* port, for the mac 1 represents the modem port and 2 the printer port. **Baud Rate** is the serial bitspeed your generator uses, for the WFG 500 this is 57600 and you may leave this control unwired for earlier generators you must select 19600. If **Reset** is *true* a software controlled reset will occur before the generator will be used.

## FLCWFG Log Global.vi



Figure 24. FLCWFG Log Global.vi

This is a text window that lists the port and the VI that is executed using that port. Most communication VIs in this library send a string to this log when they are executed. In addition errors are also listed in this log and whether or not they are alleviated are lead to a communication stop. Open the panel of this VI to see the log.

## FLCWFG Make Picture.vi

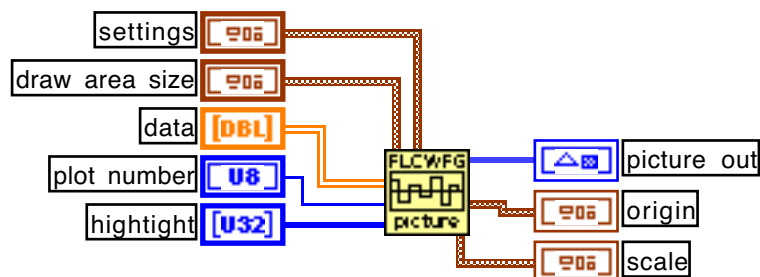


Figure 25. FLCWFG Make Picture.vi

You can make a picture of a 2D **data** array that you can display in **picture out**. **draw area size** is a cluster containing width and height, it can be obtained from the attribute node of a picture control or indicator (this allows for rescaling of the actual picture control or indicator). **plot number** controls which waveform should be visible in the picture. With the array **hightight**

you can specify a selection that will be shown in red on the blue plot. **settings** is a cluster containing values for the maximum voltage on the y-axis, the time zoom factor and the time scroll factor. **origin** is a cluster containing the pixel values of the origin of the graph, **scale** is a cluster containing the x- and y-scales of the actual picture created. These outputs can be used to transform pixel data as obtained from the mouse attribute in to real time voltage values with *FLCWFG Mouse Position to VT.vi*.

### FLCWFG Mold Data.vi

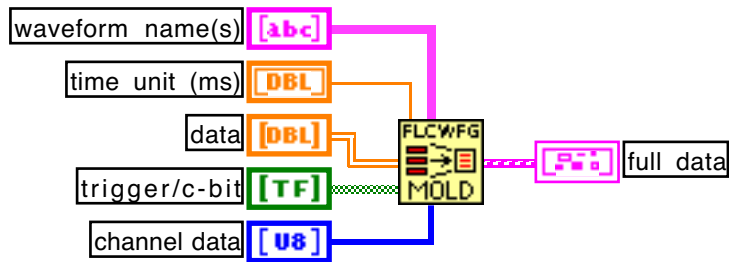


Figure 26. FLCWFG Mold Data.vi

If you have a 2D **data** array that you wish to prepare for sending use this VI to create the **full data** cluster needed for downloading. **waveform name(s)** contains the names of the waveforms, **channel data** is the array with values controlling to which channels each waveform should be sent. The **time unit** controls the clock setting. Be sure to correctly partition the data with the time unit using *FLCWFG Partition Raw Data.vi* before calling this VI. a final check before downloading is also advisable using *FLCWFG Waveform Check.vi*.

### FLCWFG Mouse Position to VT.vi

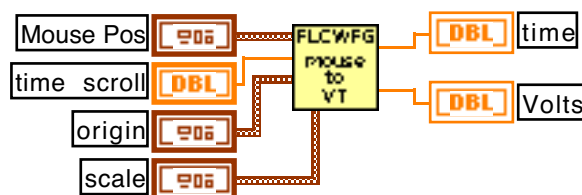


Figure 27. FLCWFG Mouse Position to VT.vi

**Mouse position** is obtained through the attribute node of a picture control or indicator. **time scroll** should not be used as shifting of the origin is already taken into account in the **origin** cluster. **origin** and **scale** are obtained from *FLCWFG Make Picture.vi* and give scale and position of the axis system. The outputs **time** and **voltage** give the values associated with the mouse position in the picture drawn.

### FLCWFG Normal Burst.vi

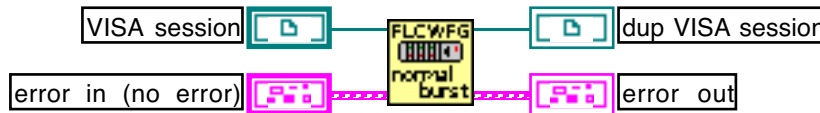


Figure 28. FLCWFG Normal Burst.vi

This VI will set the mode to *normal burst*. If an error is reported in the **error in (no error)** cluster the VI will not execute and just pass along the error cluster to **error out** and the duplicate for the VISA communication.

### FLCWFG Partition Raw Data.vi

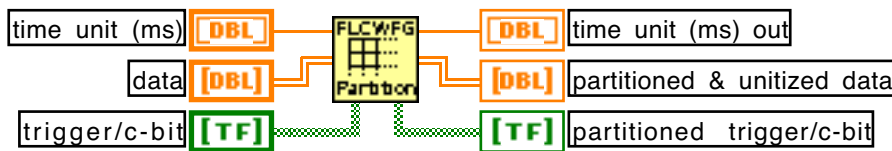


Figure 29. FLCWFG Partition Raw Data.vi

By using this VI you can let the computer decide if extra slots are needed to represent the pulses in the generator given the desired **time unit** (IN MICROSECONDS!). The 2D **data** and 2D **trigger/c-bit** arrays will be changed accordingly. This enables you to use true time values in microseconds throughout your programs. Call this VI before you mold the data into a full data cluster.

### FLCWFG Pop Up Time Unit.vi



Figure 30. FLCWFG Pop Up Time Unit.vi

This is a pop up window that will ask you to specify the time unit in a case where the software clock is used. Appropriate values are 1 to 35 ms.

*Note:*  
you should normally have no need to use this VI directly, as it is called by higher level VIs.

### FLCWFG Ready Query.vi

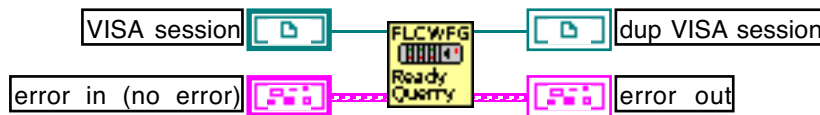


Figure 31. FLCWFG Ready Query.vi

This VI performs a ready query on the generator.

Note:

*you should normally have no need to use this VI directly, as it is called by higher level VIs.*

### FLCWFG Reset.vi

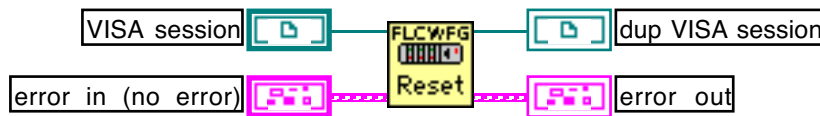


Figure 32. FLCWFG Reset.vi

This VI will perform a software controlled reset of the generator.

### FLCWFG Run.vi

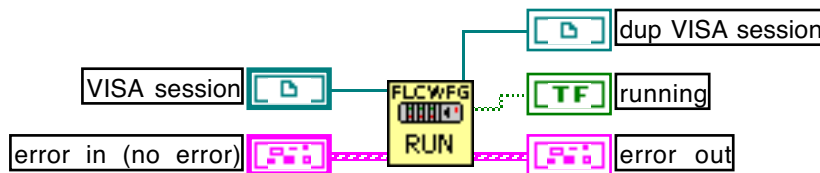


Figure 33. FLCWFG Run.vi

With this VI you can start the run procedure on the generator. If the generator was not ready to run a message will appear. The output **running?** returns the actual state of the generator.

### FLCWFG Scale At Index.vi

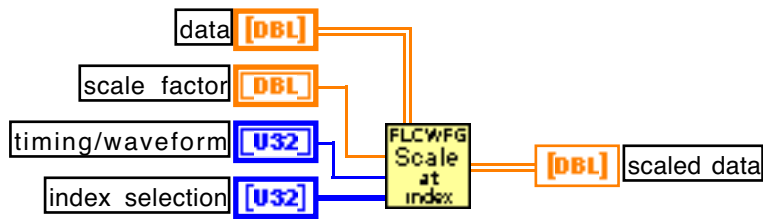


Figure 34. FLCWFG Scale At Index.vi

The 2D **data** array you provide will be scaled at the selected pulses by the specified **scale factor**. The control **timing/waveform** should be set to 0 to address the timing all values actually correspond to the first index of the date array. With the array **index selection** you can specify a number of pulses which should be scaled. The VI returns the **scaled data**.

### FLCWFG Scale Data.vi

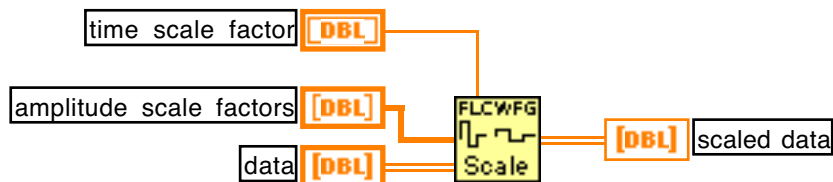


Figure 35. FLCWFG Scale Data.vi

With this VI you can scale entire waveforms and timing. You provide a 2D **data** array and an array of **amplitude scale factors** for the waveforms in addition to a **time scale factor**. Any scale factor of 1 has no effect and, hence, no scaling will be performed for that waveform. The VI returns a **scaled data** array.

### FLCWFG Scale Waveforms Pop Up.vi



Figure 36. FLCWFG Scale Waveforms Pop Up.vi

This VI invokes a pop up window in which you can specify a scale factor for each waveform in the **full data** cluster, in addition to a time scale factor. The scaled data is returned through **scaled full data**.

### FLCWFG Set At Index.vi

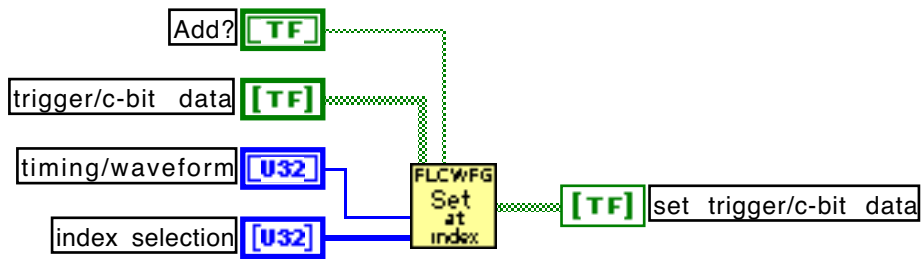


Figure 37. FLCWFG Set At Index.vi

Using this VI you can set the trigger data if you select the timing in the **timing/waveform** control or the c-bit data for the selected waveform. If the option **Add?** is set to *true* the pulses in **index selection** will be set to *true* in addition to those already present in the **trigger/c-bit data**. If this option is set to *false* only the pulses in the index selection will be set, effectively erasing all previous trigger or c-bit information. The VI returns the **set trigger/c-bit data** array.

### FLCWFG Set Software Clock.vi

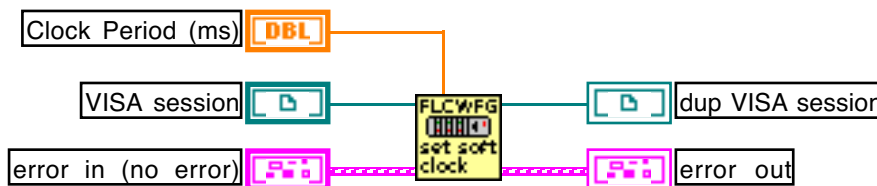


Figure 38. FLCWFG Set Software Clock.vi

With this VI you can set the software clock in multiples of milliseconds from 1 to 35 ms. Setting of the clock is usually done by *FLCWFG Download Waveforms.vi*.

### FLCWFG Status Request.vi

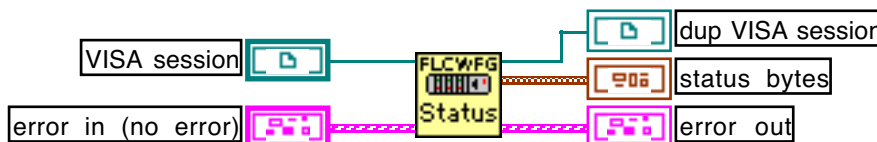


Figure 39. FLCWFG Status Request.vi

This VI issues a status request to the generator. **status bytes** is a cluster containing the value of each byte. In normal programs you do need this VI, information about the status bytes is found in the WFG 500 manual.

## FLCWFG Stop.vi

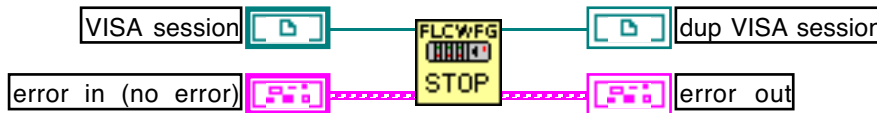


Figure 40. FLCWFG Stop.vi

This VI will stop a running generator.

## FLCWFG T-Data To Word.vi

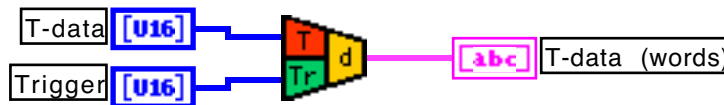


Figure 41. FLCWFG T-Data To Word.vi

This VI performs the translation of the timing data taking into account the trigger data. A bit pattern is obtained. This VI is called by *FLCWFG Build Timing String.vi*.

Note:

*you should normally have no need to use this VI directly, as it is called by higher level VIs.*

## FLCWFG Unmold Data.vi

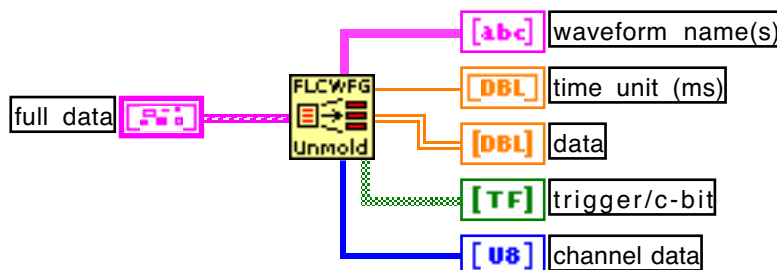


Figure 42. FLCWFG Unmold Data.vi

Use this VI to transform a **full data** cluster, as obtained from *FLCWFG File Data Translator.vi*, into a 2D **data** array with corresponding 2D **trigger/c-bit** array. It also returns arrays for **waveform name(s)** and **channel data**.

### FLCWFG V-Data To Word.vi

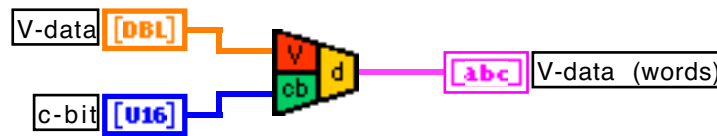


Figure 43. FLCWFG V-Data To Word.vi

This VI performs the translation of the voltage data taking into account the c-bit data. A bit pattern is obtained. This VI is called by *FLCWFG Build Waveform String.vi*.

Note:

*you should normally have no need to use this VI directly, as it is called by higher level VIs.*

### FLCWFG Waveform Check.vi

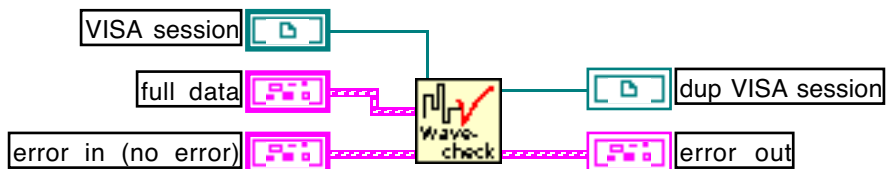


Figure 44. FLCWFG Waveform Check.vi

This VI performs a number of standard checks on the **full data** cluster. It will spot any problems and take action accordingly. It will display a message for each encountered problem and its solution. If the channel data makes no sense, a pop up window will appear in which you can solve the problem.

### FLCWFG Waveform Editor.vi



Figure 45. FLCWFG Waveform Editor.vi

This is an application example that enables you to make selections on a graph, scale and step values dynamically and set trigger or c-bit information.

## FLCWFG Waveform Player.vi



Figure 46. FLCWFG Waveform Player.vi

This is an application example with which you can open a file, scale it and send it to the generator.

## FLCWFG Write & Read.vi



Figure 47. FLCWFG Write & Read.vi

This VI enables a robust write - read cycle on the serial port to the generator.

*Note:*

*you should normally have no need to use this VI directly, as it is called by higher level VIs.*

## FLCWFG Write To Log.vi

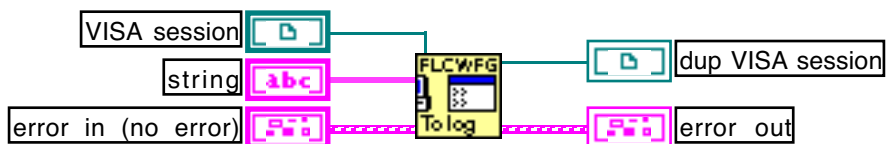


Figure 48. Write To Log.vi

This VI writes **string** to the *FLCWFG Log Global.vi*. Open the front panel of this global to see the messages.

## FLCWFG PREF\_U.llb



Figure 49. FLCWFG PREF\_U.llb

These VIs are used inside the *FLCWFG Initialize.vi*. You should not use these VIs directly.

*Note:*

*you should normally have no need to use this VI directly, as it is called by higher level VIs.*