WaveTrak Globals

Function of Globals

A global variable is one that is accessible from any script in the stack, provided it is declared a global at the beginning of the handler. There are several important uses for globals. WaveTrak needs a number of default variables to run the environment, such as how many points to plot when drawing a wave, or what the current A/D sampling rate is set to. This information is used and changed in many places in the stack and must always be current. The best way is to store this data in a global variable so any script can have access to it. The second important use arises from the fact that the only way that externals, on which WaveTrak depends heavily, can return large amounts of data is through global variables. XFCNs can only return a single value, so that a list (or *array*) of globals is an efficient way for XCMDs to return many waves acquired from the A/D converter. This chapter describes some of the more important global variables that you are most likely to use when scripting your own functions.

Remember to declare these variables as global at the start of every *handler*, or you will instead access a *local* variable of the same name, which will not contain the information you expect.

Alphabetical Listing

ADCbits

Stores the number of bits of resolution for the currently active A/D converter. The same convention is used as for standard WaveTrak data types (see the Scripting chapter). The on-board A/D converter has 12 bits of resolution and is factory jumpered to signed, two's complement coding (range -2048 to 2047), so **ADCbits** is set to -12 at startup. If you enter different defaults for the A/D converter configuration in the System Parameters card (see Fig. 6-8), **ADCbits** will be updated accordingly. You will commonly use this global when calling the `translateToReal` and `translateToBinary`

handlers.

bottomY

This is one of the standard wave descriptors, and is discussed in detail in the Scripting chapter.

DACbits

As with **ADCbits**, **DACbits** stores the number of bits of resolution for the currently active D/A converter. The on-board converter has 12 bits of resolution, and is usually jumpered for signed, two's complement encoding (range -2048 to 2047). Entering different defaults for the D/A converter configuration in the System Parameters card (see Fig. 6-8), will update **DACbits** accordingly. **DACbits** is used directly by XCMDs that generate analog levels for calculating the required binary code. You will normally never need to access this global in your scripts.

DACGainTable

This global contains two lines corresponding to the gain applied to the D/A outputs external to the MacADIOS II board. It is a copy of the 'Ext gain' column in the 'External D/A gain' table in the System Parameters card (Fig. 6-8). For example, if your D/A converter is configured to output ±10 V and you externally attenuate D/A output 0 by a factor of 10, line 1 in the 'External D/A gain' table would be 0.1 and line 1 in **DACGainTable** would also be 0.1 (remember that line 1 in **DACGainTable** corresponds to D/A channel 0). This allows scripts to accurately and automatically generate analog levels even with external gain (or attenuation). Here is an example from the 'Write DAC' button:

Example:

```
global DACGainTable
put 0 into DACchannel      -- the D/A channel
put 1000 into DAClevel     -- the required output
voltage (mV)

-- adjust DAC level w.r.t external DAC gain
get line (DACchannel+1) in DACGainTable
put round(DAClevel/it) into adjDAClevel

-- output the analog level
WriteDAC DACchannel,adjDAClevel
```

It's a good idea to always adjust the output level in accordance with the **DACGainTable**, even if you don't have any external gain. This way, if you add gain later, all you have to do is update the table in the System Parameters cards, and all your scripts will behave accurately. Otherwise, you would have to search for all scripts that generate analog outputs and correct them individually.

DACmax, DACmin

These globals hold the maximum and minimum full scale output voltage of the D/A converter (in mV). The present hardware can output either ±10 V (DACmax=10000, DACmin=-10000) or 0 to +10V (DACmax=10000, DACmin=0), depending on the jumper settings. These globals are updated depending on the settings in the System Parameters card, and are used directly by all XCMDs that generate analog levels for computing the required binary code. You will normally never need to access these globals in your scripts.

dispRect

Short for 'display rectangle', this global defines the rectangle which determines where waves will be drawn by `DrawWaveCoords` and related XCMDs. **dispRect** consists of a comma-delimited list of four integers in the order left, top, right, bottom corresponding to two points defining the display

rectangle. **dispRect** is initialized to 12,47,375,289 at startup. You can pass any rectangle you wish and the drawing XCMDs will automatically re-size the plots to fill it. All zooming, cursor readout and low resolution PICT export functions will automatically recognize the new display

size. If you change the **dispRect** coordinates to generate a wave plot of a different size on a custom card, be sure to restore the original values before leaving this card, or waves subsequently drawn in trace cards will not match the background frame (see the technical note below).

Technical note:

The solid frame enclosing the wave display in all trace cards is actually a rectangle painted in the trace *background* (whereas the tick marks, which are in color, are re-drawn each time). If you change the standard **dispRect** coordinates you must erase the old frame and manually paint a new rectangle in the trace background corresponding exactly to the new **dispRect** coordinates. Otherwise, the plotted waves will not be in register with the background frame.

dotFlag

Used by `DrawWaveCoords` and related XCMDs to determine whether waves should be drawn as dots only (**dotFlag** = TRUE) or if points should be connected by lines (**dotFlag** = FALSE). The boolean value of this global is toggled by the 'Dots only' menu item in the Scope and trace cards.

f3dB, fHi, fLo

Digital filters supplied with WaveTrak support linear or logarithmic rolloffs. These globals, which are preset from the 'Digital Filters' card in a graphical fashion, determine the cut-off frequencies (in Hz) for various filters. They offer a convenient way for the user to enter filter characteristics that will be consistent throughout the stack. See the description of various filtering XFCNs in the chapter on XCMDs and XFCNs for more details. See also the **rolloff** global.

FIRCoeffs

WaveTrak implements a finite impulse response (FIR) digital filter with the `FilterWaveFIR` XFCN (see the chapter on XCMDs and XFCNs for more

details). The coefficients describing the characteristics of the filter are passed in a global as the second parameter to `FilterWaveFIR`. You can paste coefficients into a field in the Digital Filters card, and for convenience, WaveTrak stores these coefficients for you in the global **FIRCoeffs**. This allows you to have access to the filter you stored in the Digital Filters card from anywhere in the stack by passing this global.

FSTable

This very important variable holds all the information required to translate binary codes from the A/D converter to real voltage values. It contains a current copy of the 'Full-scale' column in the 'External A/D gain' table located in the System Parameters card (Fig. 6-8). The default value contains 8 lines (corresponding to A/D channels 0 to 7), with three comma-delimited items per line.

Tip:

You can add eight more lines to this global to represent A/D channels 8 to 15 if you wish. The standard WaveTrak defaults assume that the converter is jumpered for differential input, which results in eight input channels.

Items 1 and 2 define the minimum and maximum full-scale input for that channel, taking into account the setting of the on-board programmable gain amplifier, external gain (entered in the 'Ext. gain' column of the 'External A/D gain' table) and the converter input polarity (± 10V or 0 to +10V input). Item 3 identifies the unit of measure. The 'External A/D gain' table and the **FSTable** global are automatically updated at startup and whenever you change any of the relevant parameters in the System Parameter card. This global is used extensively to map converter codes (which are always binary integers) to real units.

Example 1:

In the simplest case, the A/D converter is configured to accept a ±10 V full scale input, the on-board amplifier is set to unit gain (x1) and you have no amplification externally. What will be the corresponding **FSTable** entry for channel 3?
With no gain, the real full scale input corresponds to the converter's full-scale input, i.e. ±10 V. Therefore, line 4 in **FSTable** (remember, line 1 for channel 0) will contain "-10000,10000,mV". The last item tells us that the values are expressed in millivolts. Assuming a 12-bit converter, this indicates that when the converter outputs a code of -2048 (the most negative 12-bit integer), it actually had -10000 mV at the input of A/D channel 3, and for a code of +2047, it had +10000 mV. When you acquire a trace, the appropriate line from **FSTable** is copied into the 'Hardware Parameters' field in the new trace card.

Example 2:

The A/D converter is again configured to accept a ±10 V full scale input, but now the on-board amplifier is set to a gain of 10. You additionally have an amplifier with a gain of 50 between your system and the connector on the A/D board.
You must update the values in the System Parameters card. Set the 'Onboard A/D gain' pop-up menu to 10, and click on the cell corresponding to A/D channel 3 in the 'Ext. gain' column. Enter 50 in the dialog box. You have now given WaveTrak all the information it needs to determine what the true voltage was *at the source* (you really don't care what the voltage was at your external amplifier output, or at the input of A/D chip), from any binary code generated by the A/D converter. You will notice that the corresponding 'Total gain' is now 500 (10 x 50), and the full-scale input is "-20,20,mV", which was immediately copied to **FSTable**. This makes sense because if your source puts out +20 mV, your amplifier will boost this to +1000 mV (50 x 20 mV) and the on-board amplifier will further amplify the signal to +10000 mV (10 x 1000 mV), which, not surprisingly, corresponds to the positive full-scale input of the A/D chip (±10 V). The net result as far as you are concerned, is that when the converter outputs a code of 2047, you know your source was at +20 mV (and at -20 mV for a code of -2048). All other values can be computed by the linear relationship determined by these two points. The `translateToReal` handler in the stack script was designed to perform just such a transformation automatically, and is a good example of how a linear transfer function is used to translate binary codes to real units.

Globals

gList

This global (short for 'global List') contains a list of *names* of *globals* where waves are stored. When WaveTrak redraws a wave(s) on the screen, or exports a wave as a PICT or table, it looks for the name(s) of the globals in **gList**, that contain the waves. This creates a mechanism whereby you can insert the names of your own globals containing your own waves, and immediately take advantage of all of WaveTrak's standard drawing and exporting functions (the Scripting chapter gives detailed information about this mechanism). Each time you open a new trace card, WaveTrak copies the wave into a global called **theWave**, and logs the name of the global into **gList**, to indicate that all of its standard functions should be performed on a single wave which is stored in **theWave**:

```
global theWave,gList
.
.
put bg fld "data" into theWave  -- copy data to a global
put "theWave" into gList        -- log name of global in gList
.
.
```

Example:

You are in a trace card and you write a button that generates two sine waves. You want WaveTrak to automatically draw them, zoom them, and export them.

```
global gList,w0,w1
.
.
-- put 1st sine wave into global w0
put InitWaveSin (npoints,cycles,pkAmpl,phase,offset,resultType)¬
into w0
-- put 2nd sine wave into global w1
put InitWaveSin (npoints,cycles,pkAmpl,phase,offset,resultType)¬
into w1
-- log the names of the globals in gList
put "w0,w1" into gList
-- the standard ReplotWave and all other handlers will know that
-- they should handle two waves, in w0 and w1
ReplotWave
```

Once you execute this script, WaveTrak's standard menus such as 'Copy as Big PICT' will know that they should handle the waves whose names you put in **gList**.

Note:

In the current release of WaveTrak, only drawing and zooming of multiple waves simultaneously is supported (i.e. up to 16 names can be logged in **gList**). Other functions such as exporting will only recognize the *first* wave in **gList** and ignore any others.

HardDiskName

Contains the full pathname of the folder where WaveTrak resides, including the last colon. Use for file related functions, when you want to place a new file in the current folder.

Example:

If WaveTrak is in a folder called "my WTRK folder" on your hard disk called "myBigHD", **HardDiskName** would contain "myBigHD:my WTRK folder:". Now you want to write the contents of a variable called `Report` to a new file in the current folder:

```
global HardDiskName
put "myFile" into shortFname              -- name of file
put HardDiskName & shortFname into fName  -- full path name
open file fName
write Report to file fName
close file fName
```

HardwareOK

This global contains TRUE if the hardware check performed at startup found a recognized acquisition card in the selected slot. If something went wrong, **HardwareOK** will contain FALSE, indicating that you cannot access any of the hardware devices on the A/D card.
This global is commonly used to check whether or not to proceed with a hardware-dependent operation, such as `AcqWave`.

leftX

This is one of the standard wave descriptors, and is discussed in detail in the Scripting chapter.

maxPlotPoints

Determines the number of points that will be plotted in the display windows in both the Scope and trace cards. To maximize drawing speed, not all points in your wave are plotted on the screen when your wave is first drawn. One side-effect is that very fast transients (narrow pulses or frequency spectra of very pure sine waves, for

instance) may be missed in the normal view. Zooming in will of course reveal all of the wave's details. If you expect many sharp spikes in your data and want to make sure they appear in the normal view, you can increase the number of points WaveTrak plots in the display windows. The trade-off is slower drawing speed. The default is 500 points, which is fast and represents most data well. The 'Display Points...' menu item in the Analysis menu (Scope and trace cards only) allows you to manually reset the value of **maxPlotPoints**.

npoints

This global contains the most current entry (either from the System Parameters card or the Scope card) for the number of points to be sampled per wave. It is used by all acquisition commands.

openedOnce

This global is set to TRUE the first time the stack's `openStack` handler executes. When you open the stack again (when returning from the summary stack for instance), the value of **openedOnce** is checked (it will retain its value across stacks, until you quit HyperCard) and certain initializations are not repeated. Use this global if you want to perform some of your own initializations only once at startup.

rightX

This is one of the standard wave descriptors, and is discussed in detail in the Scripting chapter.

rolloff

The slope of logarithmic roll-off filters, in dB/octave. Positive values will attenuate, and negative values will emphasize components beyond the **f3dB** frequency (see also **f3dB, fHi, fLo** in this chapter).

Globals

rootID

Short ID number of the last root card you visited. This number is updated each time you open a new root card. Traces use this number to access data from the parent root card.

sampleInterval

This global contains the most current entry (either from the System Parameters card or the Scope card) for the sample interval (the number of μs between samples) used by all acquisition commands.

shiftTraceStep

If you hold down the shift key while pressing the navigation arrow buttons in the trace cards, you will jump by **shiftTraceStep** traces, instead of jumping to the next (or previous) trace. **shiftTraceStep** is initialized to 10 at startup.

theWave

Global receiving contents of background 'data' field when you open a new trace.

timeStamp

The current time and date (from the standard HyperCard function `the seconds`) is copied into **timeStamp** just before a call to an acquisition command to accurately log the time the wave was captured. This global is then used by `newTrace` to copy the acquisition time into the required fields in the new trace.

Example:

```
global timeStamp
put the seconds into timeStamp
-- acquire the wave
AcqWave sampleInterval, npoints, startMUX, endMUX,"theWave"
-- execution time of other code here could make time stamp
-- inaccurate if it were read only by newTrace below
.
.
newTrace  -- uses the value previously stored in timeStamp
```

topY

This is one of the standard wave descriptors, and is discussed in detail in the Scripting chapter.

TTLpulse

Line 5 in the Hardware Parameters field (System Parameters card) lets you define a default onset and width (both in μs) for TTL pulses (Fig. 6-7). This line is copied into **TTLpulse** for general access throughout the stack.

Example:

```
global TTLpulse
global sampleInterval,npoints
global theWave

put 1 into timerChannel
-- extract parameters from TTLpulse global
put item 1 in TTLpulse into preTrig    -- pulse onset
put item 2 in TTLpulse into pulseWidth -- pulse width

put 0 into startMUX
put startMUX into endMUX

-- acquire a wave, generate a TTL pulse according to TTLpulse global
get AcqWaveTimer (sampleInterval,npoints,startMUX,endMUX,¬
timerChannel,preTrig,pulseWidth,"theWave")
```

w0..w15

Sixteen globals declared for you at startup, to be used to store waves and build wave arrays.

XCMDErr

This global is used by all externals to inform you of any errors that might have occurred during the operation, by returning an error number. *Every XCMD and XFCN updates the value of **XCMDErr** with every call*, so don't expect this global to retain its value after calling an external. A value of zero means that execution of the XCMD or XFCN completed successfully. Any other positive integer indicates an error; these are documented in the chapter on WaveTrak errors. The 'ErrorList' card contains a list of error codes and a short description. These entries are used by the `ErrNum` handler to put up dialog boxes displaying the short error message. When writing your own scripts, it is a good idea to call `ErrNum` after every XCMD or XFCN to inform the user of any problems. Otherwise, the user will only here a beep and wonder what's wrong. `Errnum` does nothing if **XCMDErr** = 0.

Example:

```
global XCMDErr  -- declare as a global in every handler
.
.
-- AcqWave will put any error codes in XCMDErr global
AcqWave sampleInterval, npoints, startMUX, endMUX,"theWave"
ErrNum XCMDErr  -- report the error, if any
```

Remember to declare **XCMDErr** as a global in *every* handler.

xMag

Horizontal magnification factor used by `DrawWaveCoords` for zooming in and out. Each time you click the mouse in the display window while holding down the option key, the horizontal dimension will be magnified by a factor of **xMag**. If you click the mouse while holding down both the option and shift keys, the horizontal dimension will shrink by the same factor. This global is set to 2.0 at startup. The 'X mag...' menu item under the Analysis menu (Scope and trace cards only) allows you to manually reset the value of this global. Enter a value of 1 to prevent magnification in the horizontal direction. **xMag** must be ≥ 1.0 (see also **yMag**).

Xunit

This is one of the standard wave descriptors, and is discussed in detail in the Scripting chapter.

XYCoordXformat, XYCoordYformat

Whenever WaveTrak must display or export a real value (which may have a fractional component), it looks to these two globals for instructions on how many digits to include after the decimal point. The default values are `"%.0f"` and `"%.1f"`, respectively, meaning that any X coordinates will be displayed without fractional components (zero digits after the decimal point), and Y coordinates will have 1 digit after the decimal point (go to any trace card and observe the cursor readout). In general, these formatting strings are as follows:

`"%.nf"`

where `n` determines the number of digits after the decimal point.

You can reset these formatting strings as you wish. For example, if you chose to display the cursor readout in ms and volts, instead of µs and mV, you would probably want to include 3 digits for the X values (`"%.3f"`) and 4 digits for the Y values (`"%.4f"`) to obtain the same precision in the readout. Otherwise, the display would be rounded to the nearest millisecond and tenth of volt. Many of the numerical export functions such as `CopyAsXY` and `CopyAsY` use these globals when generating their tables.

Technical note:

C programmers will recognize these strings as standard format specifiers. In fact, these globals are passed directly in `sprintf` calls. For example:

```
i=sprintf(Cstr,"%.1f",yFloat);
```

where `yFloat` is a single precision float

You can put any valid C-type format specifier into **XYCoordXformat** or **XYCoordYformat** to customize your output.

yMag

Vertical magnification factor used by `DrawWaveCoords` for zooming in and out. Each time you click the mouse in the display window while holding down the option key, the vertical dimension will be magnified by a factor of **yMag**. If you click the mouse while holding down both the option and shift keys, the vertical dimension will shrink by the same factor. This global is set to 2.0 at startup. The 'Y mag...' menu item under the Analysis menu (Scope and trace cards only) allows you to manually reset the value of this global. Enter a value of 1 to prevent magnification in the vertical direction. **yMag** must be ≥ 1.0 (see also **xMag**).

Yunit

This is one of the standard wave descriptors, and is discussed in detail in the Scripting chapter.

In Summary

- Globals are variables accessible from any script in a stack. If you want to access a global variable, you must declare it as such at the beginning of *each* handler.

- WaveTrak globals hold important system parameters. They also allow XCMDs and XFCNs to return multiple results (such as wave arrays).