WaveTrak Errors

Introduction

One of the more important aspects of an application is the way it handles situations when things don't work as expected for one reason or another. Because WaveTrak gives you great flexibility, it also increases the chances of anomalous behavior if the wrong instructions are executed at the wrong time. This could range from the benign error, such as attempting to perform an FFT on a wave with a number of points which is not a power of 2, to the more serious, such as attempting to read from an empty NuBus slot, which causes HyperCard to quit immediately (or would completely crash the machine under earlier versions of system 6).

The designers of WaveTrak recognize reliability as the most important quality that a program can possess. As a result, all XCMDs and XFCNs perform extensive error checking to prevent an illegal call from crashing the program, and possibly corrupting the stack. All externals pass an error code back to the calling script in the **XCMDErr** global variable (even though it is called ***XCMD*Err**, XFCNs also pass their errors back in this variable). This global is updated with each call. If the external executed successfully, zero is returned in **XCMDErr**. Otherwise a positive integer, a WaveTrak error code, signals that something went wrong.

There are two types of errors in WaveTrak: non-fatal errors are more like peculiar situations that you should be aware of. Non-fatal errors are silent -- they don't cause the Mac to beep. The external often manages to finish its job and return a valid result. Examples include error 60, which signifies that the pulse width delivered by `AcqWaveTimer` for example was not exactly what you requested. The XFCN still performs the acquisition and generates an approximate pulse, but sets **XCMDErr** to 60 to alert you to the fact that you should check the result returned by the function to find out what the true pulse width really was. Usually it was close enough so there was little reason to abort the entire operation.

Another example is error 17, which is a time-out error. When an XCMD such as `AcqWaveOnTTL` must wait for a TTL pulse that never arrives, it will time out after a certain period and set **XCMDErr** to 17, indicating that, although execution terminated normally, no data were acquired. This is also a silent error, and the Mac doesn't beep. You must therefore check the value of **XCMDErr** to see if the XCMD acquired any new signals.

Tip:

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

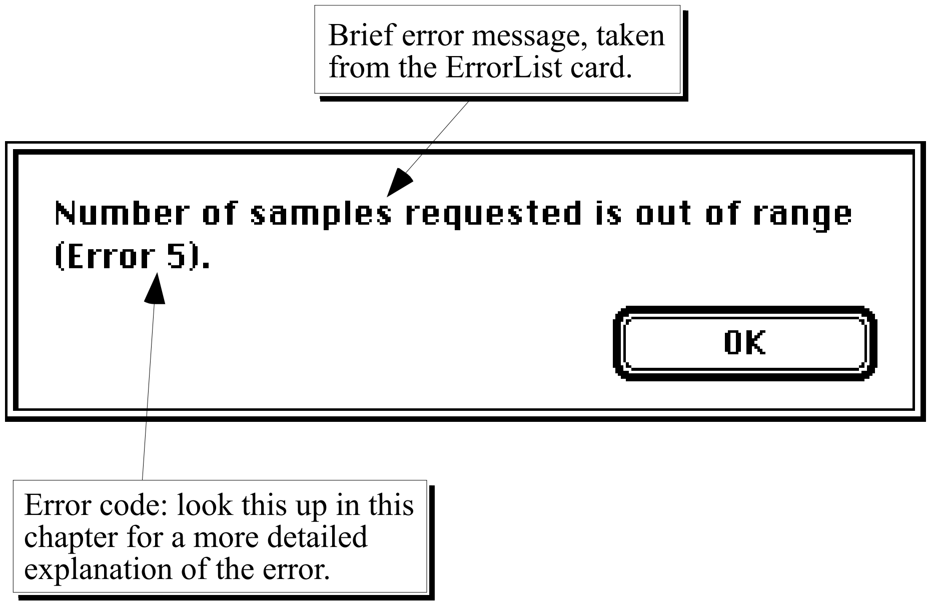
Sometimes an XCMD cannot finish execution and terminates prematurely, generating a fatal error. This type of error would occur, for example, if you pass a negative number for the number of points to be acquired in a wave, or when the program runs out of memory and cannot accumulate any new data. Fatal errors always cause the Mac to beep, to warn the user that something went seriously wrong.

The 'ErrorList' card contains a list of error codes along with a short description. These entries are used by the `ErrNum` handler in the stack script to put up dialog boxes displaying the short error message. The error code appears in parentheses after the message so you can look it up in this chapter for a more complete explanation and possible remedies. When writing your own scripts, it is a good idea to call `ErrNum` after every XCMD or XFCN to check for any problems. Otherwise, the user will only hear a beep and wonder what's wrong. `Errnum` does nothing if **XCMDErr** = 0, i.e. if the XCMD executed successfully:

Example:

```
global XCMDErr  -- declare as a global in every handler
.
.
-- AcqWave will put any error codes in XCMDErr global
put -1 into npoints  -- force an error
AcqWave
sampleInterval,npoints,startMUX,endMUX,"theWave"
ErrNum XCMDErr       -- report the error
```

The following dialog would be generated by the above code example:


Number of samples requested is out of range (Error 5).

OK
Brief error message, taken from the ErrorList card.
Error code: look this up in this chapter for a more detailed explanation of the error.

Fig. 13-1: Example of an error dialog generated by the `ErrNum` handler.

Below is a complete listing of all WaveTrak errors arranged by error code. A description is included with each, explaining why the problem occurred and what steps you should take to remedy the situation.

List of Error Codes

0: No error.

Execution completed successfully, no errors were generated.

1: Function not available.

This function is not implemented in the current version.

2: Unable to complete clipboard operation.

Clipboard operation (copy, paste or format conversion) could not be completed because the data type was wrong (must be type 'TEXT') or the operation ran out of memory. Place the correct data type on the clipboard, or allocate more memory to HyperCard.

3: Hardware devices are not accessible.

You attempted to call an external that requires the MacADIOS II A/D card. Either the card is not installed, the selected slot is empty or a hardware malfunction occurred. Check to see if the selected slot has a MacADIOS II card installed, and that the card is seated properly in its connector. If you still get this error, it is possible that the card is faulty.

If your serial number is not valid, the MacADIOS II card will not be accessible, and calls to externals requiring the card will also return this error.

4: Sampling interval/rate is out of range.

Sample intervals must be positive and an integral number of microseconds. Acquisition commands also have an absolute lower limit (usually 7 µs for current hardware), and an upper limit of 13107 µs. Check to see that the value you passed is within range.

5: Number of samples requested is out of range.

The number of points (samples) to be sampled per wave (`npoints`) must be > zero and an integer. Check to see that the value you passed is within range.

6: A/D channels are invalid.

You must pass a range of multiplexer channels to acquisition commands specifying which A/D channels are to be sampled. The values must be between 0 and 15, and the starting channel (`startMUX`) must be ≤ the ending channel (`endMUX`).

7: Out of memory: unable to allocate array.

When you launch WaveTrak, a block of memory (i.e. the MultiFinder partition) is allocated by the system for HyperCard, determined by the setting in the Get Info box. After HyperCard loads, WaveTrak uses the remaining free memory (called the heap) for wave arrays and to hold results of intermediate computations. The operation that generated this error could not be completed because not enough memory was available on the heap. Either operate on smaller waves, or quit HyperCard and increase the partition size to allocate more memory.

Tip:

If you are simply doing computations on large waves or wave arrays, and not digitizing signals with the data acquisition board, you can turn on virtual memory under system 7 to allocate more memory to HyperCard if you do not have enough RAM. Operations will be slowed somewhat, but this technique allows you to operate on very large waves.

Note that digitizing signals with virtual memory turned on will work, but you could have gaps in your data during the time the Mac swaps memory pages to and from disk.

8: Out of memory: some waveforms could not be encoded.

The XCMD ran out of memory while encoding a wave. Either repeat the acquisition with less points per wave, or quit HyperCard and increase the partition size to allocate more memory.

9: Invalid TTL bit number: must be 0-7.

The digital input and output ports have eight bits each, numbered 0 to 7. Therefore, the bit number you specify must be between 0 and 7. Check to see that the value you passed is within range.

10: Pulse width out of range or not a multiple of sampling interval.

This error is returned by operations that generate a TTL pulse.

a) For XCMDs generating a pulse from the TTL output port:
Either the pulse width was not a multiple of the sampling interval, was negative, or was too long, extending beyond the end of the sample window (i.e. `sampleInterval` * `npoints`).

b) For XCMDs generating a pulse from the timer chip output:

Pulse width must be a positive integer.

Check to see that the value you passed is within range.

11: Illegal pre-trigger time: out of range or not a multiple of sampling interval.

a) `AcqWaveTTL`: pre-trigger time must be ≥ zero, an integral multiple of sample interval, and < sample window time.

b) `AcqWaveTimer`: pre-trigger time must be ≥ zero and < sample window time.

12: Pre-trigger time + pulse width is longer than sample window.

The sum of the pre-trigger time and pulse width was greater than the entire sample window time (= `sampleInterval` * `npoints`). Choose a shorter pre-trigger time, pulse width or both.

13: Invalid D/A channel: must be 0 or 1.

The digital-to-analog output channel must be either 0 or 1.

14: D/A output level is out of range.

The requested analog output level for this digital-to-analog output channel is out of range, given the current D/A configuration and external gain. Choose a lower range or change the D/A output range on the MacADIOS II board.

15: TTL edge must be 0 (falling) or 1 (rising).

The TTL edge parameter must be either 0 (for falling edge) or 1 (for rising edge).

16: Timeout value must be between 1 and 800 seconds.

The value for the time out parameter must be an integer between 1 and 800 seconds.

17: Event was not detected within timeout interval: no samples were acquired.

The TTL edge or analog threshold crossing was not detected within the specified time-out interval. No samples were acquired. This is a silent error.

18: Illegal event list: empty, did not start with time=0, or otherwise invalid.

The event list passed to the initialization function was empty or its first line was not in the form "`0,y`". All event lists must begin by defining a starting value at time = 0. Or the list could not be interpreted; perhaps it contains non-numerical data, or delimiters that were not commas. Check the event list to make sure it is valid.

19: 'leftX' must be less than 'rightX'.

When specifying horizontal limits, `leftX` must be < `rightX`. See the description of the `DrawWaveCoords` XCMD and the Scripting chapter for more details about these two wave descriptors.

20: 'bottomY' must be less than 'topY'.

When specifying vertical limits, `bottomY` must be < `topY`. See the description of the `DrawWaveCoords` XCMD and the Scripting chapter for more details about these two wave descriptors.

21: TTL level must be 0 (low) or 1 (high).

You can only specify a TTL level as either 0 (low) or 1 (high).

22: TTL byte must be between 0 and 255.

The value of a digital byte must be between 0 (all zeroes) and 255 (all ones).

23: Gain of on-board programmable gain amplifier must be 1, 10 or 100.

The current hardware supports gains of 1, 10 or 100 only. Any other value will cause an error.

24: Mode byte must be between 0 and 255.

The value written to the MacADIOS II mode register must be a byte with a value between 0 (all zeroes) and 255 (all ones).

25: XCMD/XFCN failed to get the global: global does not exist or global name was a null string.

The external requires a value from a global variable that it could not find. Make sure the stack has declared and initialized all globals required by this XCMD.

26: Could not find 'WTRK A/D Lib': hardware devices will not be accessible.

The code resources needed to access the hardware on the MacADIOS II card reside in the stack named 'WTRK A/D Lib'. WaveTrak could not find this stack so no hardware devices will be accessible. Make sure that the 'WTRK A/D Lib' stack is in the same folder as the master stack (it cannot be in a subfolder -- it must be at the same directory level).

27: Invalid serial number: hardware devices will not be accessible.

The serial number you entered in the Home Card is invalid, and therefore hardware access is disabled. Make sure you typed in the correct serial number supplied with

your package. If the numbers match, and you still get the error, contact the technical support line for assistance.

28: Invalid slot address: choose another slot from the 'Slot Addresses' card (see 'Go' menu).

The slot selected from the 'Slot Addresses Card' does not exist in this machine. Go to the 'Slot Addresses Card' (under the 'Go' menu) and select a proper slot where the A/D board is installed.

29: Selected slot is empty: choose another slot from the 'Slot Addresses' card (see 'Go' menu).

The slot selected from the 'Slot Addresses Card' is empty. Go to the 'Slot Addresses Card' (under the 'Go' menu) and select the slot where the A/D board is installed.

30: Slot does not contain a recognized acquisition card: choose another slot from the 'Slot Addresses' card (see 'Go' menu).

The slot selected from the 'Slot Addresses Card' contains a card but WaveTrak does not recognize it as a MacADIOS II card. Go to the 'Slot Addresses Card' (under the 'Go' menu) and select the slot where the MacADIOS II card is installed.

31: Demo version: all traces will be discarded when you quit.

This is a demo version. Any new traces acquired during this session will be discarded when you quit.

32: Invalid D/A bit resolution: 'DACbits' global does not contain a legal value.

The global variable **DACbits** contains the resolution and binary encoding format of the D/A converter. The global currently contains an illegal value (usually should be -12). Check to see that you didn't overwrite this global inadvertently.

33: Invalid A/D bit resolution: 'ADCbits' global does not contain a legal value.

The global variable **ADCbits** contains the resolution and binary encoding format of the A/D converter. The global currently contains an illegal value (usually should be -12). Check to see that you didn't overwrite this global inadvertently.

34: Global does not contain a valid wave.

You attempted to pass a wave in a global variable. The value was not a legal WaveTrak wave. Make sure that you really copied a wave into the global.

35: GetGlobal returned a NIL handle.

The XCMD required the value of a global variable, but HyperCard couldn't find it. Make sure the stack has declared and initialized all globals required by this XCMD.

36: Number of vertices must be > 1.

The number of vertices passed to build a polygon (such as with `CopyBigPICT`XCMD) must be > 1. Check to see that the `nVertex` parameter is valid.

37: X calibration must be ≥ 0.

The size of the X calibration bar must be ≥ zero.

38: Y calibration must be ≥ 0.

The size of the Y calibration bar must be $\geq$ zero.

39: Threshold is out of range.

The requested analog threshold was beyond the current binary range of the A/D converter, and can therefore never be reached. Check to see that the conversion of the threshold parameter from real units to binary is accurate.

40: List of global names was invalid (empty, too few items or non-alphanumeric characters).

The XCMD expected a list of global names (usually in `gList`). The list was either empty, contained too few names for the number of waves to be returned, or some of the items contained characters which are illegal in HyperCard variable names. Check the instructions that compiled the `gList`.

A common error that is difficult to find at first glance is to inadvertently add a space between variable names in `gList`. For example:

```
-- space before w4 is OK here because HyperCard
ignores
-- spaces in variable declarations and parameter
lists
global w0,w1,w2,w3, w4,w5,w6,w7
.
.
-- space here will generate an error 40, because
AcqWave
-- will attempt to find a global named ' w4' (with a
-- leading space), which is an illegal variable
name.
put "w0,w1,w2,w3, w4,w5,w6,w7" into gList
.
.
AcqWave sampleInterval,npoints,startMUX,endMUX,gList
```

The space in the second line is not ignored because it is part of a quoted string, rather than just a list of variables. Remember that all characters count when enclosed within double quotes.

41: Illegal value passed for baseline.

The baseline parameter was not a legal floating point number.

42: An integer overflow occurred: result is OK but out-of-range values were clipped.

A math operation was instructed to return an integer wave type (e.g. `resultType` = -12), but some points in the result were beyond the allowable range (for instance, some values were > 2047 for `resultType` = -12). The out-of-range elements were clipped to the maximum or minimum allowable values. The result is otherwise valid. This is a silent error.

43: Illegal result type: must be an integer in -16 to +16 range, or 'F'.

The result type requested of a math function must be an integer in the range from -16 to +16, or "F".

44: A divide by zero error occurred: the calculation was aborted and the result is undefined.

One of the points in the divisor was zero. The operation was aborted and the result is undefined.

45: Illegal wave segment: startTime and/or endTime parameters are invalid.

When defining a wave *segment* (as in `Trim`) with `startTime` and `endTime` parameters, either one of the parameters was negative, or `startTime` was > `endTime`, or `endTime` extended beyond the end of the wave. Check to see that these parameters define a valid wave segment.
A special case allows `endTime` to be -1, which means 'go to the end of the wave'.

46: Constant is not a legal floating point number.

The constant value passed to this function was not a legal floating point number.

47: Illegal global name: name was too long or contained non-alphanumeric characters.

The global name passed as one of the parameters (either alone or part of a `gList`) was empty, too long or contained non-alphanumeric characters. The most common cause is passing `theWave` (by value) instead of `"theWave"` (by name). Make sure you pass the *name* of the global where appropriate.

48: Can't pass resultType = 0 here.

You requested that the same data type be returned as the original wave, but there was no original wave in this context. For example, you can't pass `resultType` = 0 to `InitWaveK`, because there is no wave from which to determine the data type. Pass a valid integer (e.g. -12) or floating point type (e.g. "F").

49: Hardware has been disabled by the hardware lockout/option key feature.

Hardware access has been disabled because the 'Hardware Lockout' box in the System Parameters card is checked, or because you held down the option key while launching WaveTrak. Uncheck the 'Hardware Lockout' box, quit HyperCard and re-launch WaveTrak to re-enable the hardware.

50: No floating point math chip installed.

This operation cannot proceed without a 68881/882 floating point coprocessor.

51: Number of cycles parameter must be > 0.

Check to see that the `cycles` parameter passed to `InitWaveSin` or related commands was > zero.

52: Peak amplitude parameter is not a legal floating point number.

Check to see that the `pkAmpl` parameter passed to `InitWaveSin` or related commands was a valid floating point number.

53: Phase parameter is not a legal floating point number.

Check to see that the `phase` parameter passed to `InitWaveSin` or related commands was a valid floating point number.

54: Offset parameter is not a legal floating point number.

Check to see that the `offset` parameter was a valid floating point number.

55: Floor parameter is not a legal floating point number, or was ≥ 0: floor must be negative.

The `floor` parameter required by functions using dB levels (such as `AmplSpectrum`) must be negative. Check to see that it was a legal floating point number.

56: The number of points in the wave must be an integral power of 2.

This operation uses the FFT (Fast Fourier Transform) to transform the wave into the frequency domain. The FFT can only process arrays with a number of points which is an integral power of 2 (e.g. 512, 1024, 2048 and so on). Check to see that the wave you passed to this XCMD conforms to this restriction. This is why it's a good idea to always set `npoints` to an integral power of 2 in case you may want to perform filtering or spectral analysis on the data later.

57: An illegal counter/timer channel was requested.

The AM9513 has five counter/timer channels, numbered from 1 to 5. Some operations need dedicated channels internally so you may not select these in the parameter list. Check the description for each XCMD/XFCN for restrictions. In any case, the timer channel number must be between 1 and 5.

58: Period must be > 0. Period must be > sample window time for averaging functions.

The `period` parameter must be > 0. Furthermore, averaging functions (such as `AvgWaveTimer`) must have a period that is > sample window time (i.e. `sampleInterval` * `npoints`).

59: Pulse width cannot be ≥ period.

Check to see that the pulse width is < period.

60: The period and/or pulse width delivered differed slightly from what was requested.

This silent error informs you that, although the operation completed successfully, the period and/or pulse width requested in the parameter list differed slightly from the true signal that was generated, as a result of limited resolution of the AM9513 counter/timer chip.

61: Waves passed in a list of globals must be of the same type for this operation.

This operation requires that all waves passed in the global list be of the same data type.

62: Waves passed in a list of globals must have the same number of points for this operation.

This operation requires that all waves passed in the global list have the same number of points.

63: Number of averages must be > 0.

The number of waves to be averaged cannot be negative. Check the `nAvg` parameter.

64: Operation was aborted with Cmd-period.

The operation was prematurely terminated with the command-period key combination. Depending on the function, the results may or may not be valid.

65: Illegal window type.

The `Window` XFCN currently supports Hanning (1), Parzen (2) and Welch (3) windows. You requested a window that is not implemented.

66: Illegal frequency parameter.

One of the frequency parameters was invalid (either fractional, negative or not an integer).

67: fHi must be > fLo.

When specifying cutoff frequencies for linear rolloff filters, `fHi` must be greater than `fLo`.

68: Some frequency parameters were above the Nyquist limit: only valid parameters were used.

Sampling theory tells us that the highest frequency component that can be faithfully captured in a digitized wave is one half the sampling rate; this frequency is called the *Nyquist frequency*. Therefore, specifying a filter cutoff frequency above the Nyquist limit makes no sense. This error tells you that at least one of your frequency parameters was beyond the Nyquist limit. This is a silent, non-fatal error, which indicates that the filter was still applied, but the illegal frequency was replaced by the highest legal value i.e. the Nyquist frequency.

Example:

You sample a signal at 10 µs/point (= 100 kHz). The Nyquist frequency is therefore 50 kHz. You now pass the wave for filtering to `FilterWaveFFTloLin` with `fLo` = 20000 and `fHi` = 70000. `fHi` is above the Nyquist frequency, so the filter is applied with `fHi` = 50000 instead.

Although the error is non-fatal, the fact that you requested an illegal cut-off frequency indicates that another part of your script may be wrong. Check the instructions that generated the out-of-range frequency parameter.

69: Illegal value passed for roll-off parameter.

The rolloff parameter passed to a logarithmic roll-off filter was not a number.

70: f3dB rounded down to zero: next highest frequency was used instead.

Because log roll-off filters determine the attenuation of frequencies beyond the 3 dB point using the *ratio* of frequencies with respect to the 3 dB point, `f3dB` cannot be zero. If you pass a value in `f3dB` which is non-zero but small enough, it could round down to zero given the wave's sample interval. This is illegal, and the next highest frequency component (in this case the fundamental frequency) was substituted instead. *The smallest 3 dB frequency that can be passed to an FFT-based log roll-off filter is the fundamental frequency*.

71: System 7.x required for this operation.

Certain functions require features found only in system 7.0 or later. You must upgrade your system software to take advantage of this function. The vast majority of WaveTrak functions do not require system 7. We have found system 7 to be extremely robust, and more reliable that several system 6 releases. We recommend that all users upgrade to system 7 and HyperCard 2.1. Future releases of WaveTrak will rely more and more on features found only in system 7.0 and later.

72: A file-related operating system error occurred (file was open, busy, locked, etc...).

An operation on a disk file failed because the file was busy, locked, could not be found, the disk was full and so on. Make sure the file is in a location where WaveTrak can find it, is not currently being used by another application, and that you have enough free space on your disk.

73: Analog prepulse time must be > 0 and a multiple of the sample interval.

The specified width of the analog prepulse was negative or was not an integral multiple of the sampling interval. Check the `prePulse` parameter in `AcqWaveDAC`.

74: Analog prepulse + pulse width time must be less than the window time.

The sum of the width of the analog prepulse and the width of the analog pulse was greater than the sample window. `prePulse` + `pulseWidth` must be < `sampleInterval` * `npoints`.

75: Illegal value passed for hysteresis parameter.

The value passed for `hysteresis` was not a legal floating point number.

76: Error in pulse width was too large and timer was not armed.

The actual pulse width that could have been generated would have differed by more than 30% from what was requested, and therefore the pulse (train) was not initiated. Check the possible resolution from the function result and adjust the new pulse width to minimize the discrepancy (see the `StartPulseTrain` XFCN description).

77: Too many events were detected and some were not reported.

`WaveToEventList` and related functions can only return 400 events at a time. The reported analysis is correct but was terminated prematurely. This limit will usually be sufficient for most requirements, but if you really need to determine more events than this, you can analyze smaller segments (using `Trim`) and adjust the reported times according to the segment extracted.

78: Sorry, there is not enough memory to continue. WaveTrak will quit so you can allocate more memory.

You must allocate at least 1 MB to HyperCard for WaveTrak to run properly, otherwise cards will refuse to open and you will get strange error messages directly from HyperCard. If you attempt to start WaveTrak with less than the required amount of memory, you will be warned with this error message and returned to the Finder. Single-click on the *HyperCard application icon* (not the WaveTrak icon) to select it, then choose the 'Get Info' item under the 'File' menu and increase the memory partition to *at least* 1000K, and preferably more. Double-click on the WaveTrak icon to restart it.

Tip:

The more RAM you allocate to HyperCard, the faster WaveTrak (and other stacks for that matter) will run. For optimal results, allocate 2.5 MB (2500 K) or more if you can.

79-91: Reserved.

92: Format for XY or Y table is incorrect.

An attempt to convert an XY or Y ASCII table into a wave using the `XYTableToWave` XFCN failed because the format of the table was incorrect. Either delimiters were missing or incorrect (only commas, tabs and spaces are recognized), lines (including the last one) were not terminated with carriage return

characters, or the values contained illegal characters. Check the format of your table for the above errors and try again.

In Summary

- WaveTrak informs the user of anomalies or malfunctions by returning errors. Errors can be non-fatal (and silent, where the Mac does not beep), usually allowing the XCMD to complete execution successfully, or fatal, signaling a serious problem (usually a software error).

- Errors are represented by numbers called *error codes*. Each XCMD returns error codes in the global **XCMDErr**. A value of zero means that the operation completed successfully.

- Error codes and brief descriptions appear in the 'ErrorList' card.

- Look up the error code in this chapter for a more complete description of the error and the necessary remedy.