

Chapter 6

Extensible Test Format (ETF)

Description of support for the Extensible Test Format

Introduction

The ScanExpress Runner software contains a run-time interpreter to parse and execute Extensible Test Format files. The Extensible Test Format is a simple language that allows the execution of commands outside the normal ScanExpress Runner Boundary-Scan environment. Using ETF makes it possible to control external test equipment such as relay controllers, and digital multi meters. ETF has the ability to execute external Windows and DOS programs from within ScanExpress Runner. It is also possible to capture the output of commands, and display it in a pop-up window during test execution. You can add delays to allow time for external circuitry to initialize and settle. It is even possible to insert documentation into the ETF file using the '#' comment command.

ETF files are created in standard ASCII text format using a program like notepad. ETF files are added into the test plan, just as you would add a CVF, SVF or FPI file. To ScanExpress Runner the ETF file appears to be a single test step, but inside the ETF file can be many individual commands. This allows for a very convenient method of partitioning test steps into logically related sections. For example, if you are performing a variety of analog tests you can place all of the setup initialization, test procedure, and posttest cleanup all in a single ETF file. Or, you may choose to break it up so that each test has its own ETF file, with all of the specific initialization code for that particular test encapsulated. It is even possible to embed CVF files inside an ETF file. In this way you have the combined power of controlling external instruments, initializing circuitry, and then performing test or control methods using boundary-scan technology.

The ETF command parser understands the following commands:

CVF, RUN, SET PIO, SET DMM, READ PIO, READ DMM, WAIT PIO, DELAY, PRINT, COMPARE, POWERSHORT and # (for comment)

Sample ETF File

See Figure 6-1 for a sample ETF file.

This ETF file is used to measure the current draw in a circuit. The digital multi-meter is placed in DC current mode, using the 3.3mA range. The circuit is configured using an external relay controller, and a stimulus is applied to the circuit using Boundary-Scan control registers. The tester waits 500 milliseconds and takes a current reading, saving it in the variable VALUE1. A different stimulus is applied; the tester waits 100 milliseconds, and takes another current reading. The two values are compared, and if the difference is +/- 4 micro amps the test is failed.

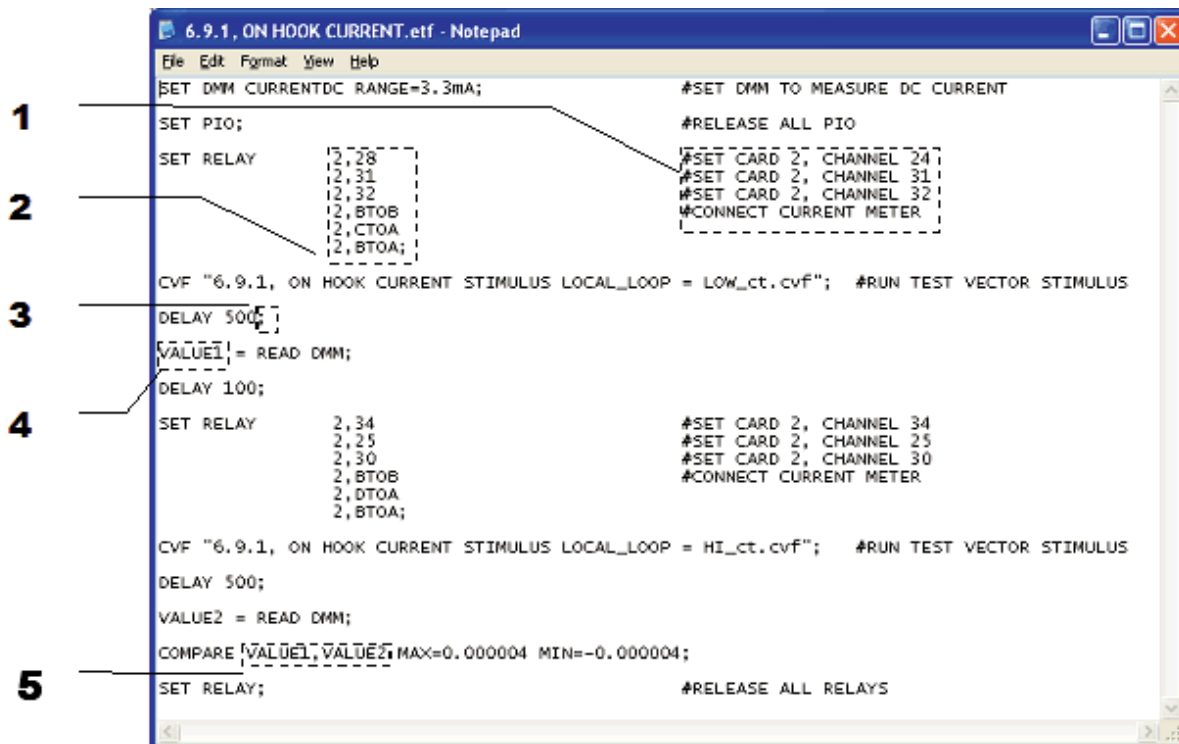


Figure 6-1. Sample ETF file

1. Comments help document the test file.

A comment consists of the '#' character and all of the following characters up to the end of the line. Comments are ignored during processing of the ETF file.

2. A command may be split over multiple lines.

In the example above, the SET RELAY command is spread over several lines. This makes it easier to see which relays are being set. It would also be perfectly valid to place all of the command on one line as follows:

```
SET RELAY 2,28,2,31,2,32,2,BTOB,2,CTOA,2,BTOA;
```

3. Each command must end with a semi-colon: “;”.

The semi-colon designates the end of a command. This allows complex commands to be split over multiple lines, as described in step 2 above.

4. Output from the READ and COMPARE command may be saved into a variable.

As can be seen from above, the results of reading the digital multi-meter are saved to a variable named VALUE1. Variables can hold string or numeric data.

Examples: ExpectedVoltage = 12.0V;

MyOhms = 33k;

A\$ = "This is a string variable";

You can also capture the result of a READ DMM, READ PIO or COMPARE command with a variable.

```
Volts1 = READ DMM;  
Volts2 = READ DMM;  
DeltaVolts = COMPARE Volts1, Volts2;
```

5. The COMPARE and PRINT commands may use a variable as an argument.

Sample Variable Usage

Figure 6-3 below shows an example of using variables to capture the input signals connected to the PIO connectors on the PCI JTAG controller. This test drives bit A4 low using the SET PIO !A4 command. It then waits up to 3 seconds for PIO bit B5 to be high. In the example below this condition succeeds, and then the A4 bit is driven high. The tester will wait up to 3 seconds (the specified time out value) for the B5 bit to go low. As can be seen below, this does not happen within the 3-second time limit, and the WAIT PIO command issues a failure notice. The detailed display shows both the expected and actual values.

At this point the test has failed, but execution will continue to the end of the ETF file. This allows any post-test cleanup to take place. The results of the test are passed back to the ScanExpress Runner main program. (See Figure 6-2 below)

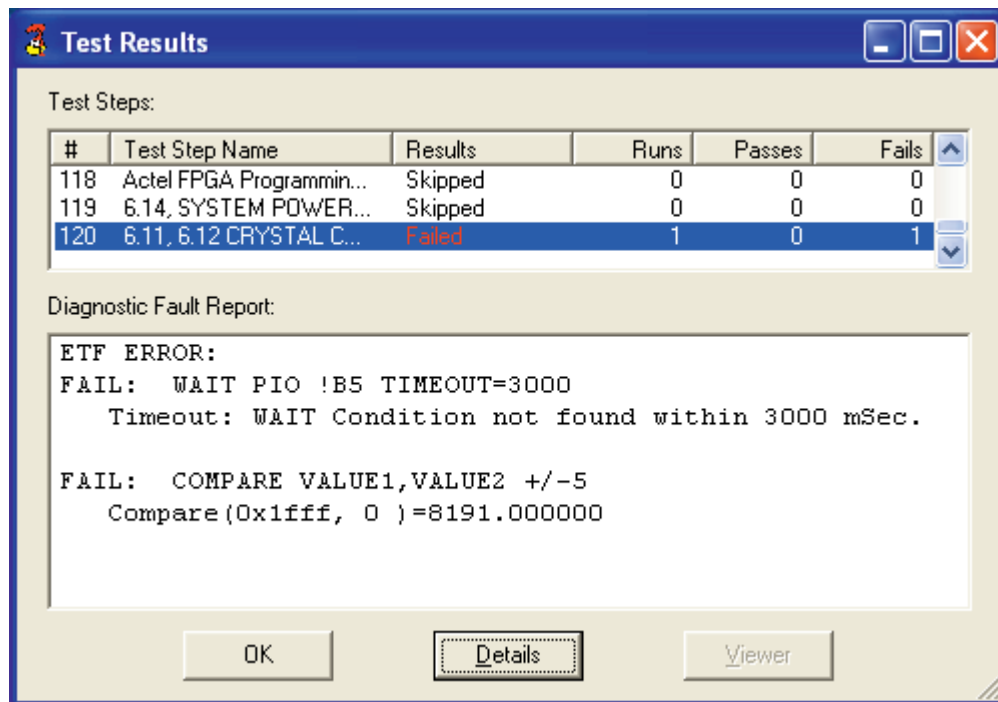


Figure 6-2. WAIT PIO timeout results

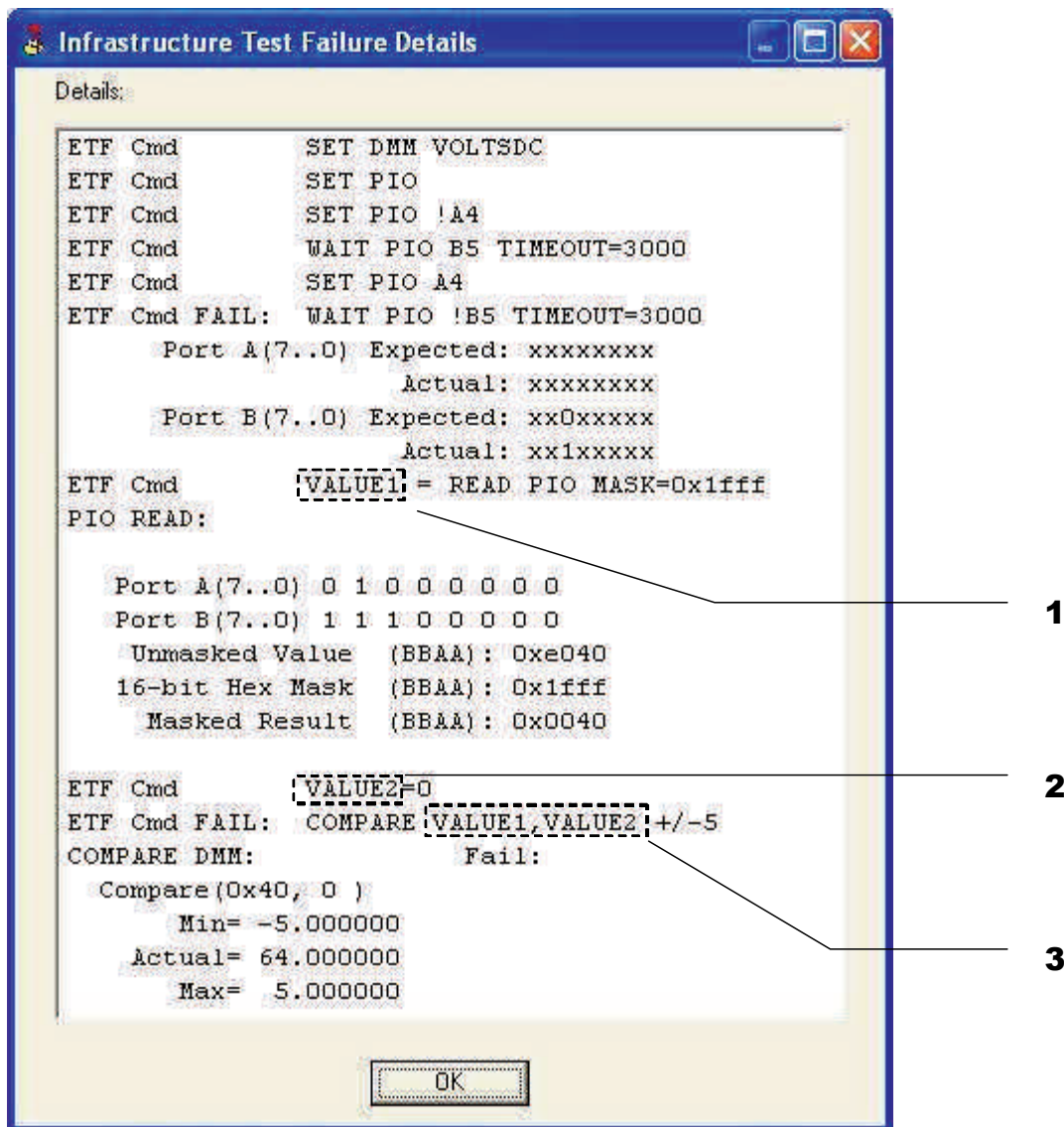


Figure 6-3. ETF Program Variables

1. Variable assigned from READ PIO

The variable VALUE1 is assigned the results of the READ PIO command. This value will be used later in a COMPARE command to verify the contents of VALUE1 fall within a specified range. In this particular case the Parallel I/O bits of the PCI JTAG controller are connected to an external device. The READ PIO command reads the state of these bits, and returns the value as a numerical value.

2. Variable assigned a constant:

The variable VALUE2 is assigned the value of zero. This value will be used as an argument to the COMPARE command.

3. Variables as arguments to the COMPARE command:

The variable VALUE1 and VALUE2 are provided as arguments to the COMPARE command. COMPARE will subtract VALUE2 from VALUE1 and compare it against the specified range. If the result falls within the specified range the COMPARE command will return a “pass” condition, otherwise it will return a “fail” condition.

Sample PRINT Command

It is possible to pause a test and display a message in a pop-up window (see Figure 6-4 below). Execution will halt until the pop-up window is closed. This is accomplished using the PRINT command.

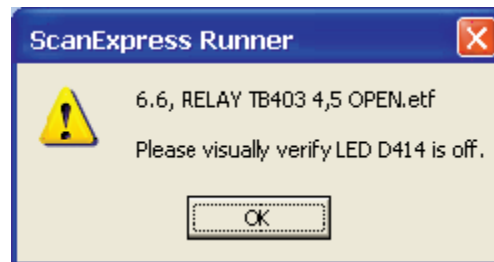


Figure 6-4. PRINT pop-up window

In Figure 6-5 below the PRINT command activates a pop-up window with the message “Please visually verify LED D414 is off.” Test execution will halt until the user closes the window by clicking on ‘OK’ or the ‘X’ (See Figure 6-4).

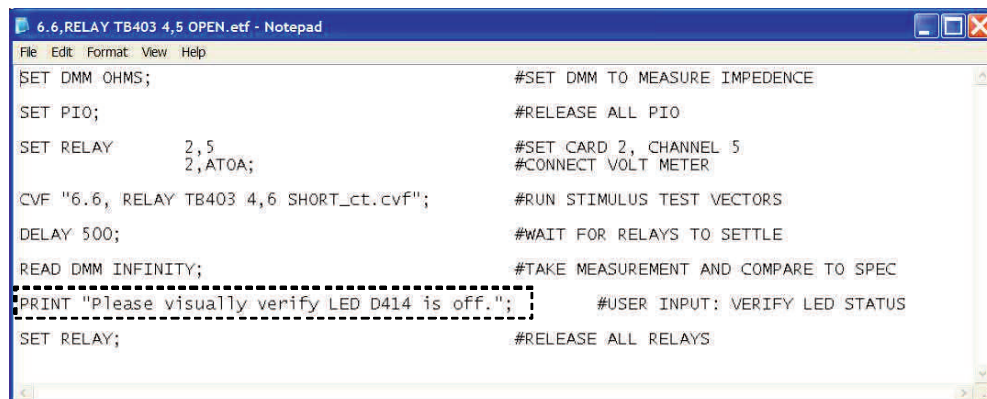


Figure 6-5. Sample PRINT command in ETF file

It is also possible to print the contents of a variable.

Example: MyMessage\$ = "This is a message that will print in a Windows Message Box";
 PRINT MyMessage\$;

PASS or FAIL Test Results

ETF files (test steps) return a result code (PASS or FAIL) similar to the other types of test steps. An ETF file is considered to pass if every command within it executes successfully and without error. Conversely, the ETF test step is considered to fail if any one or more of the commands within the ETF file fail. See Figure 6-6 for an example of an ETF test step reporting a failure to ScanExpress Runner.

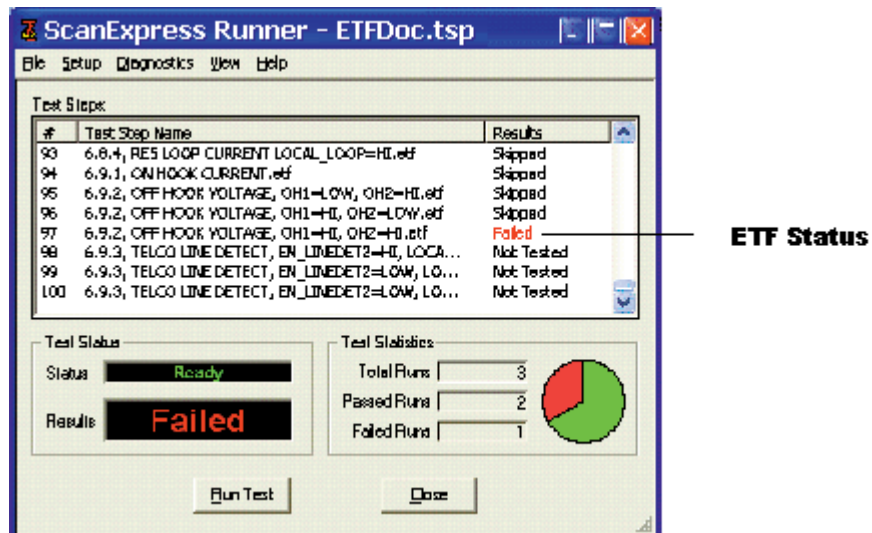


Figure 6-6. ETF test step results: Main Window

The COMPARE and READ commands have the ability to PASS/FAIL a test based on an expected value and tolerance, or a min and max value. The RUN command returns a PASS/FAIL based on the DOS Error Level or Windows Return Code generated during execution. Additionally the CVF command has the ability to PASS/FAIL a test based on the results of the boundary-scan operations.

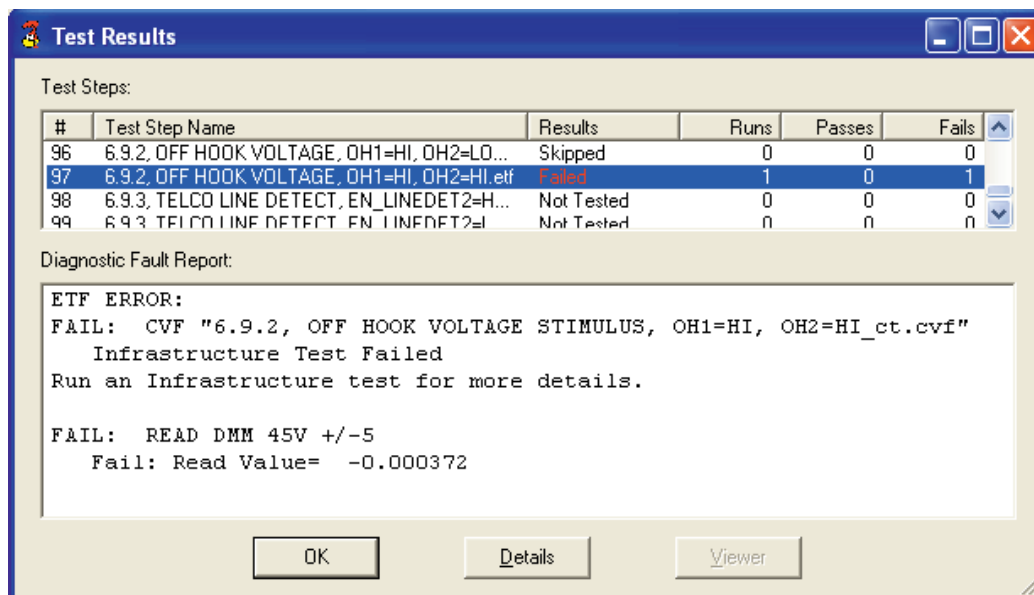


Figure 6-7. ETF test step Result Dialog

These results are passed back to the main ScanExpress Runner executive. This allows the ETF file to act just like any other test step in a test plan. As with any other test step, the results dialog will

contain status information for any selected test step. If the test step was a success, the Diagnostic Fault Report will be empty. If there were one or more failures in the ETF test step, the Diagnostic Fault Report will contain a description of each failure. See Figure 6-7 for the example Results Dialog that matches the main screen shown in Figure 6-6 above.

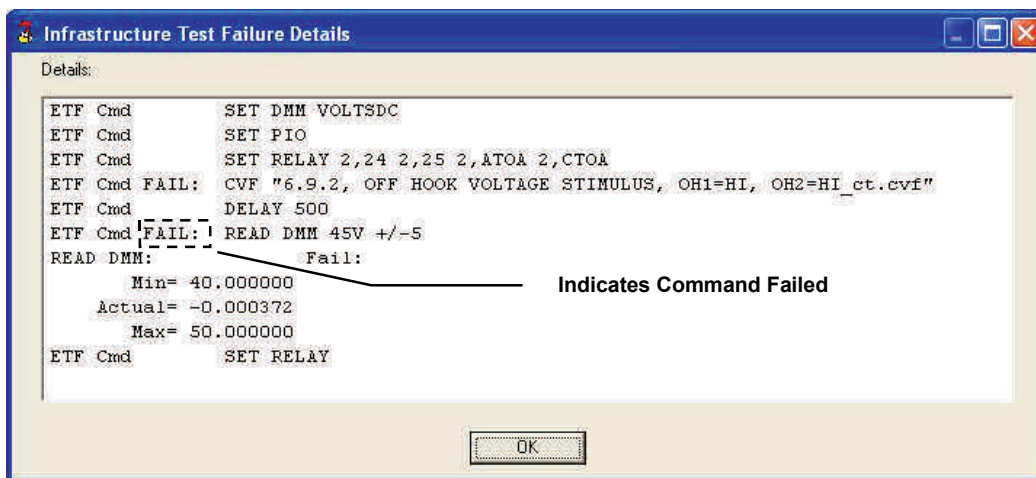


Figure 6-8. ETF test step failed: Details Dialog

Detailed information is gathered during execution of an ETF test step. This detailed information is available regardless of the pass/fail condition of the ETF file. At a minimum, the detailed information will contain an entry for each command executed in the ETF file. Comments and blank lines are ignored, and not reported. Figure 6-8 refers to an ETF file with failures. This is the same test step depicted in Figure 6-6 and Figure 6-7 above. Figure 6-9 is an example of an ETF test step that executed without failure.

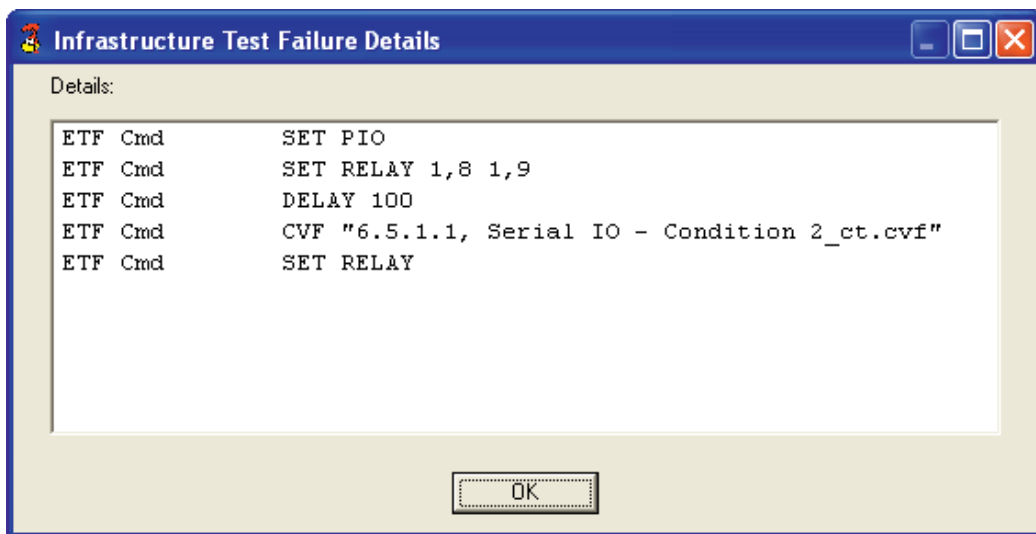
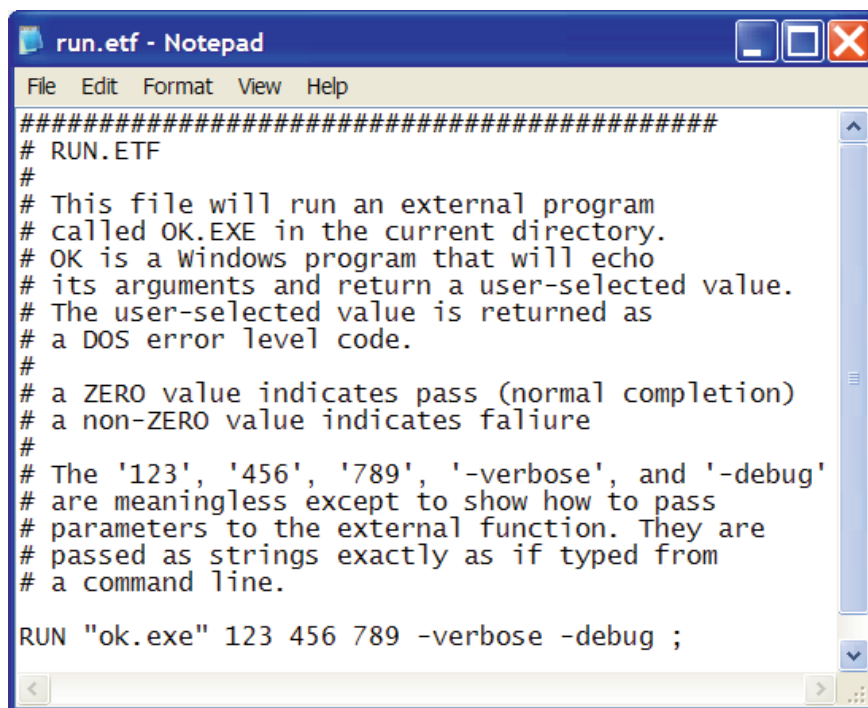


Figure 6-9. ETF test step passed: Details Dialog

RUN Command

The RUN command executes an external program. The program may be a Windows program, a DOS program, or a BATCH command file. Execution of the current test step will pause until the external program is complete and exits. The external program may return a value. If the value is non-zero the RUN command will report a FAIL condition to ScanExpress Runner.



```
run.etf - Notepad
File Edit Format View Help
#####
# RUN.ETF
#
# This file will run an external program
# called OK.EXE in the current directory.
# OK is a Windows program that will echo
# its arguments and return a user-selected value.
# The user-selected value is returned as
# a DOS error level code.
#
# a ZERO value indicates pass (normal completion)
# a non-ZERO value indicates failure
#
# The '123', '456', '789', '-verbose', and '-debug'
# are meaningless except to show how to pass
# parameters to the external function. They are
# passed as strings exactly as if typed from
# a command line.

RUN "ok.exe" 123 456 789 -verbose -debug ;
```

Figure 6-22. RUN Command ETF file

Figure 6-22 shows an ETF file that issues a run command. The first argument to the run command is the fully qualified path of the program to execute. The quotes around the file name are optional unless the file name or path contains spaces in which case the quotes are required. If you leave the quotes off a file name that contains spaces the parser will take everything up to the first space as the file specifier. Everything following the space will be treated as an argument. Surrounding the file name and path with quotes forces the ETF parser to treat them as a single argument.

The program C:\BIN\OK.EXE is a very simple Windows program that displays the arguments used to call it. It also returns a user-selectable value upon exit. This provides a very simple method to see the RUN command in action. Figure 6-23 shows the OK program after we called it with the RUN command. Notice that the arguments are the same as in the ETF file with one exception. The `-verbose` and `-debug` arguments are now capitalized! That is because the ETF parser is case insensitive and converts its arguments to upper case internally. The file name is still in lower case because it was surrounded by quotes.

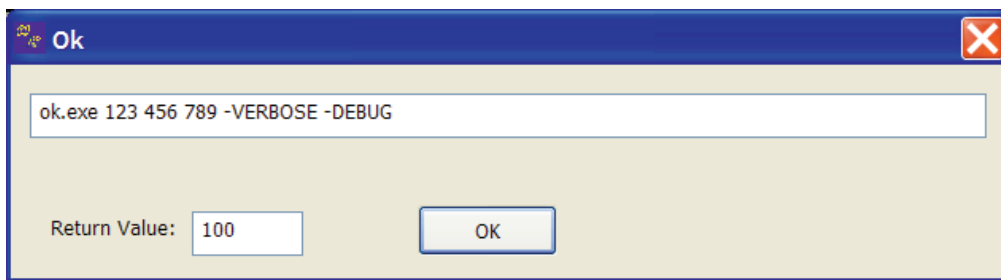


Figure 6-23. Display of the OK program

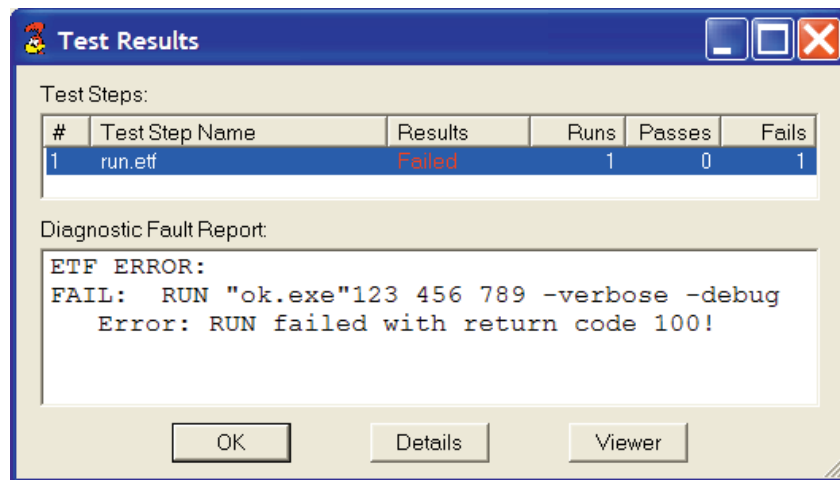


Figure 6-24. RUN command Test Results Dialog - FAIL

Figure 6-24 shows what happens when the external command returns a non-zero status code. In the example above the OK program returns a value of 100. The RUN command determines that the result code is not zero and issues a FAIL condition to ScanExpress Runner. The Test Results dialog shows the failed RUN command, as well as the return code value.

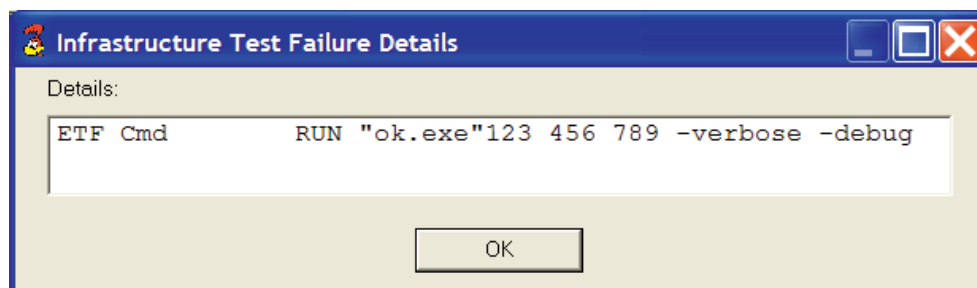


Figure 6-25. Details dialog after OK returns a zero.

Figure 6-25 displays the Details dialog after the OK program returns zero for the status return code. Even though the Test Results dialog has no information for a successfully PASSED test step, the Details dialog will still contain each ETF command processed.

RUN

This command will load and execute an external Windows or DOS program. ETF file parsing and execution will halt until the external program has completed. The external program may return an error status in the DOS ERRORLEVEL. If the return status is non-zero, the ETF file will be treated as a “Fail” condition. A zero return status indicates a “Pass” condition.

Usage:

RUN *command*

Return Value:

Pass or Fail.

Parameters:

command

The name and parameters for an external Windows or DOS program.

Example:

RUN C:\Boards\NewProduct\externaltest.exe /t /b /delay=100; # Run my test program