Custom Error Handling with TestPoint

Application Note 017 with three associated example programs.

TestPoint is commonly used for manufacturing tests, where limit or range testing is required. A practical example is a test that is comprised of several steps and may take several minutes to complete. Typically, tests that catch the most common failures are run first, along with tests that may cause problems "downstream". When a problem is detected you may want to exit the test immediately if further testing would be unproductive, or you may want to exit to investigate the fixture or make an adjustment.

Another example is a single test comprised of a loop with a high iteration count. If an error is found the first time through a loop that will normally go 1000 iterations, you don't want to force the operator to wait through 999 additional tests.

The best technique for exiting an Action List or to break out of a loop is to create an error condition that can be handled by TestPoint's Error handler object. There are no user-defined errors but we can create one of TestPoint's runtime errors deliberately and use it to trigger any set of actions.

For example, if the close bracket, ']', is the first character in a math object's formula, TestPoint runtime error 65 will be generated immediately. This feature can be exploited to add custom error handling to a TestPoint application. In essence, we can borrow error number 65 for own purposes. To do so, the code will programmatically set the formula of the math object (drag the formula field from the math object's settings panel to an action list).

TP017a.tst illustrates this first concept. The value from a slider object is tested for greater than 1value. If this is true, the program writes the ']' character to the math objects formula field. This will generate a runtime error and the code will branch to TestPoint's Error-Handler object. Appropriate action can be taken with the commands placed onto the action list of the Error-Handler object.

TP017b.tst illustrates a typical use of the error handler. In this case an error is detected within a loop, where a calculation is based on a user input. As long as the calculated values are within range, the display shows the in range message. If the user input creates an out-of-range condition, the loop halts and displays an out-of-range message with the loop count and calculated value that created the out-of range condition.

In this case, the application stops when an error is detected. However, the Error-Handler object has options to continue, stop, retry or post a message when an error is detected so that many alternative actions are available. Alternative actions could include automatically changing the test parameters and restarting the test, changing the test sequence, or requesting new test parameters, among others.

TP017c.tst illustrates a method for enclosing the error handler within a user-defined object (UDO). This is generally the best approach if the error handler is used frequently because it can be added to the stock list, the details of it's implementation are hidden, and the user defined object provides convenient drop-down-list choices when placed in an Action List. Consult the TestPoint documentation or other example programs for more information on creation of UDOs.