Customizing the TestExec SL Operator Interface using Visual Basic

Application Note
Overview

This application note describes how users of the Keysight Technologies, Inc. TestExec SL software can customize the operator user interface using Visual Basic.

Operator Interface Overview

Operator interface is the user interface designed for system operators to control the test system in the manufacturing environment. It is usually simplified for non-technical operators and has a lower level of access to the testplan and features of TestExec SL.

Different users will have different requirements for the user interface. TestExec SL offers a customizable user interface that can vary in the appearance, features, interactivity and access levels required. It may range from simple customization to include just basic functions of start and stop buttons, and a pass/fail indicator; or it can be as complicated as to contain multiple-level interfaces for user interactions, with access to external hardware and database sources.

TestExec SL allows you to create custom operator interfaces with Microsoft Visual Basic and other programming platforms that support the ActiveX controls (e.g. Microsoft Visual C++). Generally, using Visual Basic is the preferred method because of its simplicity and flexibility amongst the textual programming environment.

The best way to begin learning how to customize an operator interface is through understanding the working examples. This application note will familiarize you with basic features of the operator interface and some enhancements that you may consider adding to your customized interface.
In order to follow through the example below, you should be familiar with Visual Basic 6 terminology and concept, its integrated development environment (IDE) and the use of ActiveX controls. The operator interface will communicate with TestExec SL through a special set of TestExec SL ActiveX controls.

TestExec SL ActiveX control library is a collection of automation objects with predefined properties and methods. The Visual Basic code controls TestExec SL as a separate process, changing its setting (properties), sending it commands (methods), and waiting for progress to occur (events). Remember to add the “Keysight TestExec SL control library” from the Visual Basic Components List to the toolbox before using them in your interface.

![Figure 1. TestExec SL ActiveX control in Visual Basic](image)

![Figure 2. Hierarchy of TestExec SL ActiveX control collections and objects](image)
Designing the Operator Interface Layout

The first step in customizing an operator interface using Visual Basic is to design its interface layout. To design an effective operator interface, other than knowing the needs of your audience, you also have to consider other design practices. A good interface design layout would have the following characteristics:

- Simple appearance with adequate information
- Logical layout with clear flow of tasks

A simple operator interface should include the basic features of (1) selecting and loading a testplan to test a specific product, (2) running and stopping the testplan, (3) showing the test status of either pass or fail, (4) optionally reporting the test results, and (5) exiting the operator interface. Below is an example showing the layout of a simple operator interface.

This operator interface will populate a list box with the paths of registered testplans. It will then allow the running and stopping of the testplans. The state of testplan execution is maintained in a text box. The testplan report information is maintained in a rich text box. The user is given one aspect of control of the testplan, and that is to enable or disable the reporting of passed tests.

It is important that you define the physical appearance and names of all the controls at this stage prior to coding. Use the following properties window to set the properties of controls in Visual Basic environment.

### Table 1. Properties of controls

<table>
<thead>
<tr>
<th>Control Type</th>
<th>Name</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form</td>
<td>frmSimple</td>
<td>Caption = Simple Keysight TestExec SL operator interface</td>
</tr>
<tr>
<td>Listbox</td>
<td>lstTestplans</td>
<td>N/A</td>
</tr>
<tr>
<td>Command button</td>
<td>cmdLoad</td>
<td>Caption = Load testplan</td>
</tr>
<tr>
<td>Command button</td>
<td>cmdRun</td>
<td>Caption = Run</td>
</tr>
<tr>
<td>Command button</td>
<td>cmdStop</td>
<td>Caption = Stop</td>
</tr>
<tr>
<td>Command button</td>
<td>cmdExit</td>
<td>Caption = Exit</td>
</tr>
<tr>
<td>Textbox</td>
<td>txtTxSLState</td>
<td>Font = 24 point, Text = State, Alignment = Center</td>
</tr>
<tr>
<td>Checkbox</td>
<td>chkReportPasses</td>
<td>Caption = Report passes</td>
</tr>
<tr>
<td>Rich-textbox</td>
<td>rtbReport</td>
<td>Scroll bars = Both</td>
</tr>
<tr>
<td>Txs</td>
<td>testexecsl1</td>
<td>Assume all defaults</td>
</tr>
</tbody>
</table>
Coding the Event Handlers for Your Operator Interface

Upon completion of the interface layout design, the appropriate event handlers have to be created for different controls in the operator interface.

A. Loading registered testplan

A list box is added in the interface to display a list of registered testplan for user’s selection. All testplans are registered in the preferences.upf file prior to running the operator interface. This allows the developer to control and ensure that only designated testplans are accessible by the operators.

On the initial load of this interface, the coding will check the Registered Testplan collection of TestExec SL, and use the results to populate the list box lstTestplans.

A LoadTestplan button is added to allow users to control the testplan loading. The return value of the LoadTestplan method can be used to update the interface based on BeforeTestplanLoad and AfterTestplanLoad events.

After the testplan is loaded, the chkReportPasses checkbox status and testplan state information should be updated on the interface to reflect the correct status of the just loaded testplan.

```vbnet
Private Sub Form_Load()
    Dim RegisteredTestplanInstance As Object
    Set RegisteredTestplanInstance = TestExecSL1.RegisteredTestplans(1)
    For Each RegisteredTestplanInstance In TestExecSL1.RegisteredTestplans
        lstTestplans.AddItem RegisteredTestplanInstance.Path
    Next RegisteredTestplanInstance
End Sub

Private Sub TestExecSL1_AdviseClearReport()
    rtbReport.Text = ""
End Sub

Private Sub cmdLoadTestplan_Click()
    TestExecSL1.LoadTestplan lstTestplans.Text
End Sub

Private Sub TestExecSL1_AfterTestplanLoad(ByVal Path As String)
    txtTxSLState.Text = “Loaded”
        chkReportPasses.Value = vbChecked
    Else
        chkReportPasses.Value = vbUnchecked
    End If
    chkReportPasses.Enabled = True
    cmdRun.Enabled = True
    cmdRun.SetFocus
End Sub
```
B. Running and stopping testplan

In all operator interfaces, after selecting and loading the testplan, Run and Stop buttons are the necessary functions to control the states of testplan execution. Below are the source codes assigned to the Run and Stop buttons.

For actual applications, users may consider adding coding to check the testplan state (e.g. running or not running) before executing the .Run and .Stop TestExec SL methods. Furthermore, users may also add the Abort button to immediately terminate a running testplan under certain urgent situations. Aborting a testplan is different from stopping a testplan where all hardware and software states will still remained active as when it is terminated. However, the latter can be coded with proper test procedures to end a testplan.

C. Displaying the state of testplan execution

The next important task is to display the information of the testplan state (e.g. testing, passed, failed, etc.) on the operator interface.

Users can acquire the state information directly through TestExec SL state model and can customize the display information such as the display text, its properties and any animation effects. These can be coded for BeforeTestplanBegin and AfterTestplanStop events in TestExec SL. We use the return parameter Reason in the latter to set the text and color of the txtTxSLState text box.
D. Showing the testplan report

Typically, to display testplan report information, a rich text box will be used instead of just a standard text box. This is because a standard text box has limited size, which is not appropriate for long reports. A check-box is included in the interface to offer the user the option to display the detailed report information.

E. Exiting the operator interface

Last but not least, a complete operator interface should have an exit button to unload the application.

Private Sub chkReportPasses_Click()
If chkReportPasses.Value = vbChecked Then
Else
End If
End Sub

Private Sub TestExecSL1_ReportMessage(ByVal Message As String)
    rtbReport.SelLength = 1000000
    rtbReport.SelText = Message
End Sub

Private Sub cmdExit_Click()
    Unload Me
End Sub

Additional Enhancement Features for Operator Interface

What constitutes a basic operator interface is usually not sufficient for an actual manufacturing environment due to limited functionality. More event management coding is needed to make the interface more robust. For example, the LoadTestplan button should only be enabled when a testplan is selected.

More often than not, an operator interface also requires some error trapping routines and needs to be more informative and interactive. Additional information (e.g. company logo, progress bar, and test yield information) will certainly improve the usability of the interface.

Besides the appearance and layout, useful prompts and status information are also vital to an operator interface for improving the user interactivity. Direct and precise messages are essential to guide the operators on their tasks. In some cases, keyboard shortcuts are also practical to have on top of mouse navigation features.

One of the main reasons to create a customized interface is to control the user access to the test system. The level of access is very much dependent on the user types – operator, supervisor, developer and administrator. TestExec SL allows you to create a single interface with multiple logins for different level of access.

Additional Enhancement Features for Operator Interface

Accessing External Hardware Resources

Other than communicating with TestExec SL controls, the operator interface can also be customized to access external hardware resources for partially or fully automated manufacturing lines. You may want to control switches and sensors using a digital I/O card or a bar code reader through RS-232 serial interface. In the TypicalOpUI example, you will learn how to access the external bar code reader from the operator interface. It synchronizes the testplan with bar code reader events.

There are two methods in writing Visual Basic codes to interact with the external hardware resources from operator interface. The first method is to access the hardware directly via an I/O interface and the second method is to interact with TestExec SL, which in turn controls hardware via its standard I/O interface.
Conclusion

This application note explains the basic features of a typical operator interface and serves as a good introduction to learning the development of an operator interface for beginners. For more advanced learning and customization of the operator interface, we recommend you to refer the TypicalOpUI sample operator interface which comes together with TestExec SL installation. It includes several layers of error checking, extensive management of the state of controls and also support of external hardware for manufacturing automation. You can use the sample source code almost immediately for your project with minimal modification.

<table>
<thead>
<tr>
<th>Testplan-level events</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdviseClearReport</td>
</tr>
<tr>
<td>Indicates that report output is being cleared at the beginning of a run of the testplan. Typically triggers once at the beginning of a run of the testplan, even if the testplan will loop.</td>
</tr>
<tr>
<td>AfterTestplanLoad</td>
</tr>
<tr>
<td>Indicates that a new testplan has successfully been loaded.</td>
</tr>
<tr>
<td>AfterTestplanPause</td>
</tr>
<tr>
<td>Indicates that TestExec SL has entered a paused state, and returns the reason why the testplan paused.</td>
</tr>
<tr>
<td>AfterTestplanStop</td>
</tr>
<tr>
<td>Indicates that TestExec SL has halted, and returns the reason why the testplan stopped.</td>
</tr>
<tr>
<td>AfterTestplanUnload</td>
</tr>
<tr>
<td>Indicates the testplan has been unloaded.</td>
</tr>
<tr>
<td>BeforeTestplanBegin</td>
</tr>
<tr>
<td>Indicates that a pass through the testplan sequence is about to begin. Also occurs when testplan execution resumes via calling the Continue method.</td>
</tr>
<tr>
<td>BeforeTestplanLoad</td>
</tr>
<tr>
<td>Triggered in response to a LoadTestplan method.</td>
</tr>
<tr>
<td>ReportMessage</td>
</tr>
<tr>
<td>Indicates a new “block” of report output has arrived.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test-level events</th>
</tr>
</thead>
<tbody>
<tr>
<td>AfterTestDone</td>
</tr>
<tr>
<td>Triggered after the current test finishes executing.</td>
</tr>
<tr>
<td>BeforeTestBegin</td>
</tr>
<tr>
<td>Triggered before the next test in the testplan begins executing.</td>
</tr>
<tr>
<td>ReportMessage</td>
</tr>
<tr>
<td>Indicates a new “block” of report output has arrived.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Miscellaneous events</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdviseUpdate</td>
</tr>
<tr>
<td>Triggers to let the operator interface update its display in a way that does not interrupt the critical timing of a testplan.</td>
</tr>
<tr>
<td>UserDefinedMessage</td>
</tr>
<tr>
<td>Triggers to notify the operator interface that a user-defined message has arrived.</td>
</tr>
</tbody>
</table>

For more information and other literature, please go to www.keysight.com/find/testexec and www.keysight.com/find/fftforum.
Evolving Since 1939

Our unique combination of hardware, software, services, and people can help you reach your next breakthrough. We are unlocking the future of technology. From Hewlett-Packard to Agilent to Keysight.

For more information on Keysight Technologies’ products, applications or services, please contact your local Keysight office. The complete list is available at: www.keysight.com/find/contactus

myKeysight

www.keysight.com/find/mykeysight
A personalized view into the information most relevant to you.

www.keysight.com/find/emt_product_registration
Register your products to get up-to-date product information and find warranty information.

Keysight Services

www.keysight.com/find/service
Keysight Services can help from acquisition to renewal across your instrument’s lifecycle. Our comprehensive service offerings—one-stop calibration, repair, asset management, technology refresh, consulting, training and more—helps you improve product quality and lower costs.

Keysight Assurance Plans

www.keysight.com/find/AssurancePlans
Up to ten years of protection and no budgetary surprises to ensure your instruments are operating to specification, so you can rely on accurate measurements.

Keysight Channel Partners

www.keysight.com/find/channelpartners
Get the best of both worlds: Keysight’s measurement expertise and product breadth, combined with channel partner convenience.

www.keysight.com/find/ftforum