Tech Note: Building a Think & Do™ SECS Equipment Application with TransSECS™

Overview

TransSECS provides a completely graphical environment for building SECS applications. In Run Mode, TransSECS also serves as a SECS *Host Simulator*. This, coupled with VIB and VIBLaces and the Java OPC Gateway, provides a very practical way to build and test SECS interfaces to Think & Do project. As you build the SECS interface in VIBLaces you can test the results against the SECS Host (TransSECS). Essentially, the application you are going to build in VIBLaces with VIB components and TransSECS generated SECS beans is the SECS *Equipment Application*.

More detailed instructions on using TransSECS can be found in the TransSECS User Reference. It is also recommended that you have some familiarity using VIB and VIBLaces. Additional training and tutorials can be arranged from ErgoTech Systems, Inc.

This Tech Note uses the GRPRESS example from Think & Do. The purpose of this lesson is to build a SECS equipment interface to this application. We will start from a pre-built TransSECS project, TnDGRPRESS. This is based on the "SimpleTool" loaded by default by TransSECS. The steps we will follow are:

1. Start Think & Do and start the GRPRESS example
2. Run TransSECS to load the pre-configured GRPRESS SECS project
3. Run VIBLaces to create the equipment application, and add the fundamentals of the SECS interface for diagnostics, sending SECS messages and reports, and starting the equipment
4. Add OPC Beans to the equipment application to send and receive machine status information
5. Connect the OPC data to the SECS messages and test this from the SECS Host simulator (TransSECS). The Think & Do GRPRESS can be run from TransSECS by sending a SECS message to start the ram cycle. A SECS report will be sent to the host from the equipment application when the cycle is complete.

More complex messages and reports can be added as an optional exercise. This Tech Note covers some of the fundamental features of using Think & Do with TransSECS and VIB and is not intended to be a full tutorial.

Installation Note: Do not install TransSECS in a directory with spaces (blanks) in the directory name. Directories with spaces will be detrimental to the code building process. The default locations (ErgoTech/TransSECS or TransSECSEval) have been so designated to avoid this problem.

Getting Started

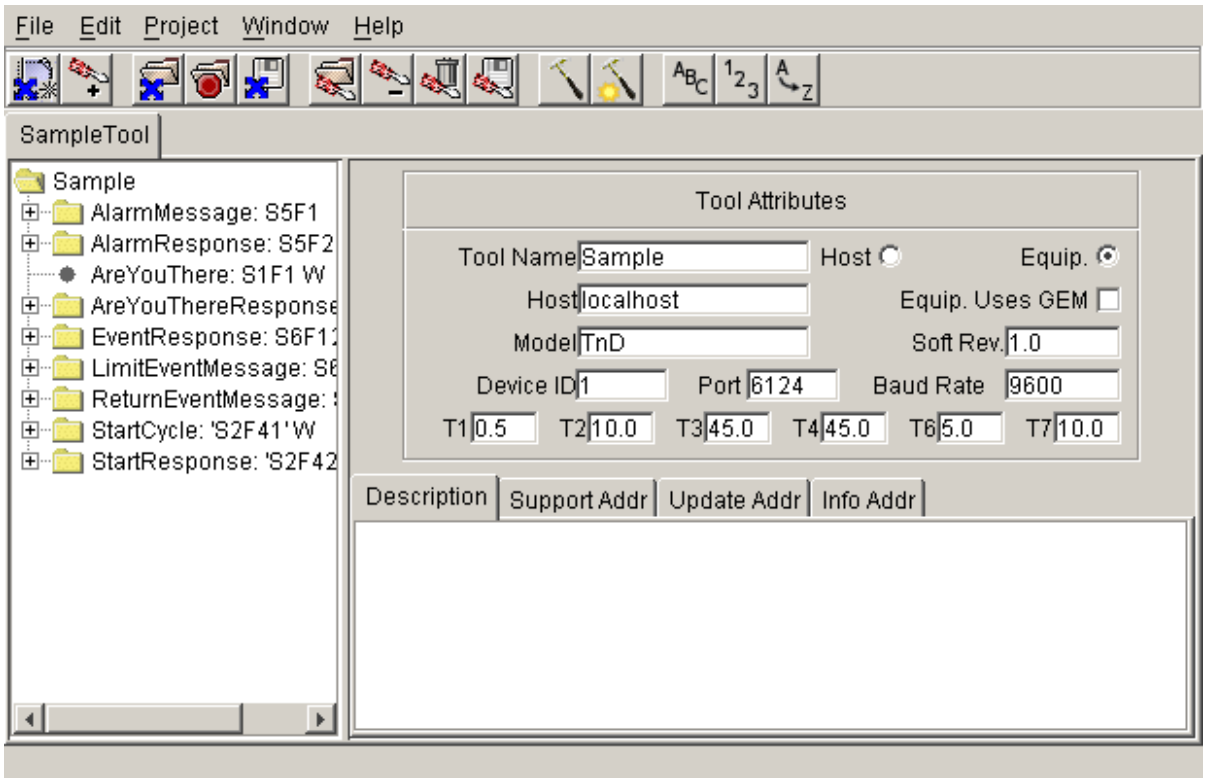
Start Think & Do Run-Time Engine. Find the sample GRPress (GRPress.cfg) and load it into the T&Do RunTime Engine. This is the project for which we will be building a SECS equipment interface.

Find the shortcut to the Java OPC Gateway. Start the OPC Gateway by double clicking on the shortcut or using the Windows Programs “Start” item under ErgoTech. A console window (command shell) will appear saying that the OPC Gateway is started on port 5113. If you are using the Trial version of the OPC Gateway, it will run for one hour after the first OPC connection is made (and then you can restart it).

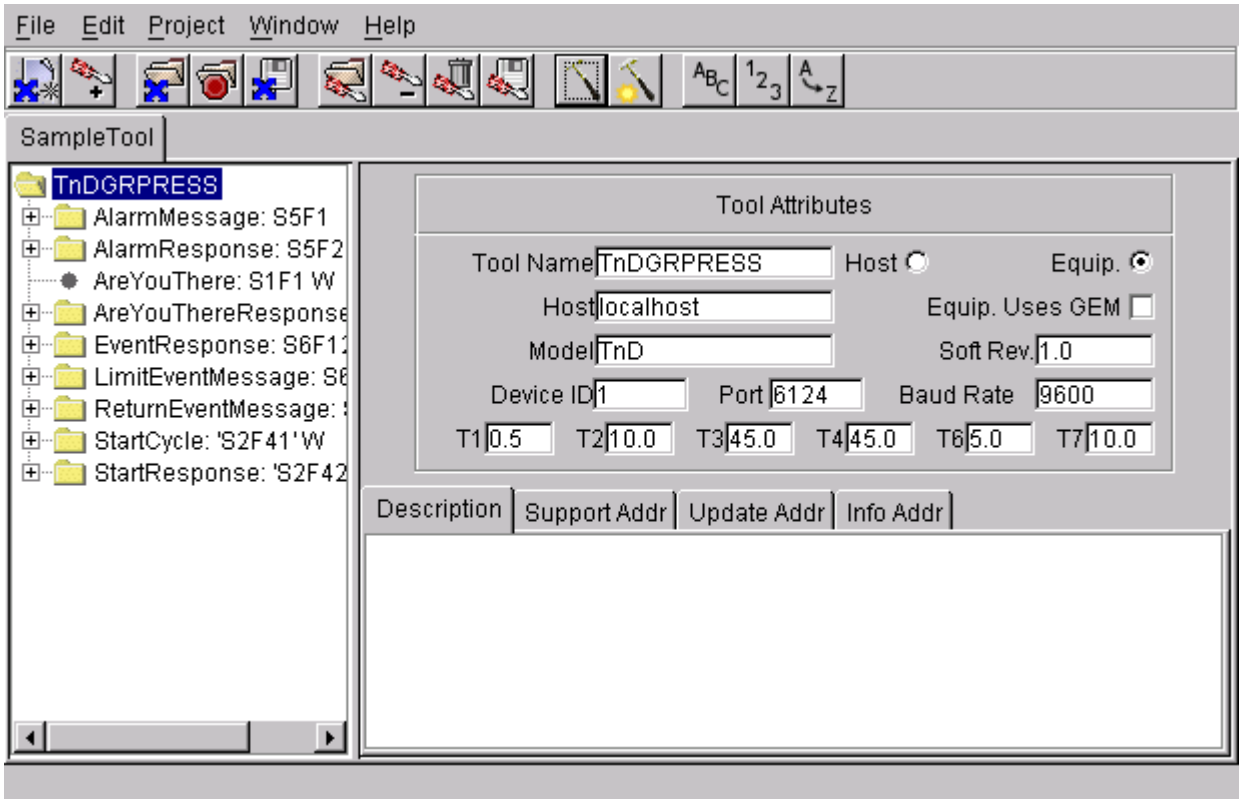
Start TransSECS. You can do this from the desktop shortcut or use the Start menu item. A sample tool, called “Sample”, is automatically loaded and will be the basis of the SECS definitions for this project. An advanced topic follows this tutorial that discusses how to enter the SECS message definitions yourself. You should have a SEMI E5 reference handy to understand how SECS messages are constructed. The Trial version of TransSECS will run up to a specified date.

Running TransSECS

We will begin customization of the Sample Tool for this exercise and save it under a new tool name. First, we will rename the tool. Select the tool node (the line that is the tool name in the tree) and notice that the properties for the tool are displayed in the right hand panel of the TransSECS display. This area is called the “attributes panel” and for the tool will display “Tool Attributes”.



Change “Sample” in the Tool Name to “TnDGRPRESS”. When this tool is saved, it will be called TnDGRPRESS and the JavaBeans generated in TransSECS will be in this designated package name.



The messages have already been set up for this project. To build the message JavaBeans, press the hammer symbol in the toolbar. This will generate the code and compile it.

When the compile stage is complete, the menu item that was a hammer will change to the “Run Mode” icon. TransSECS is now running in “host” mode. In this mode you may send selected messages, and messages may be received and responded to by TransSECS. We do not want to send any messages from TransSECS yet – first we need to build the equipment application in VIBLaces with the message Beans that have just been generated.

Preparing VIBLaces

For the next part of the tutorial we will need to make connections to the Think & Do OPC Server. This will allow us to collect data and add this to the SECS messages which will be sent to the host. For testing purposes, TransSECS will be used as the SECS Host application. We will use VIBLaces to build and test the SECS interface to the equipment. Leave TransSECS running in “Run” mode for the duration of this exercise.

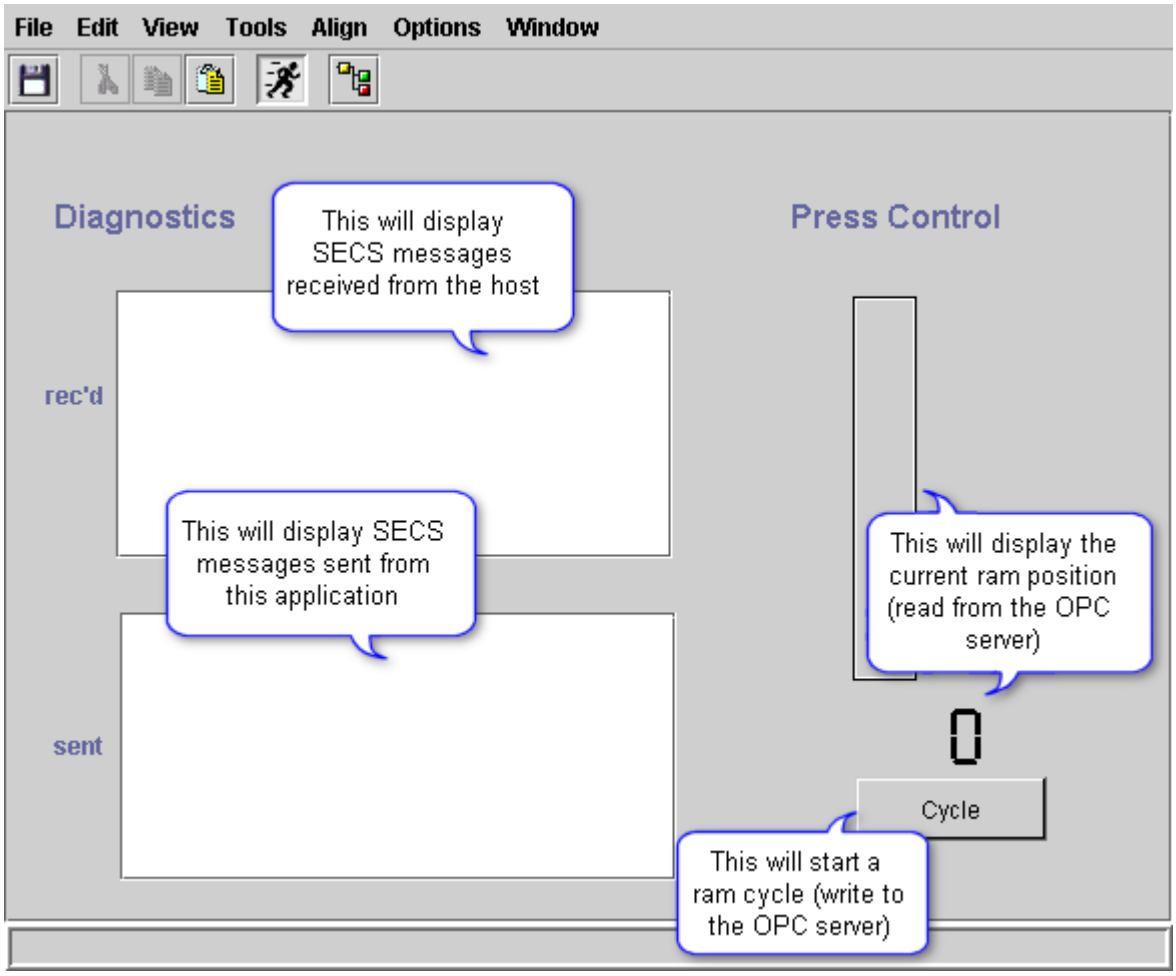
To prepare VIBLaces for the next part of the exercise, we will need to activate the Swing and OPC components. Swing components are special JavaBeans that use Java 2*, so they are not usually available except for expert users. Locate the directory where you installed TransSECS (and VIBLaces). By default this is C:/ErgoTech/TransSECEval, if you are using the Trial software, and C:/ErgoTech/TransSECS if you are using a licensed installation. Find the VIBLaces directory. There is

* Most Web browsers are only JDK1.1 compatible (not Java 2), so the only restriction in using Java 2 (Swing) components is the browser requirements. We need to use a Swing Text Area for this example because it will scroll when the viewing area is full.

a file called “stencils_swing.ini.hide” which needs to be renamed “stencils_swing.ini” so that it will be used by VIBLaces to load the Swing components into the palette. Likewise, there is a file called “stencils_opc.ini.hide” which needs to be renamed “stencils_opc.ini”.

The User Interface Design

For testing purposes, we are going to build a screen (graphical user interface) that looks like:



To build this screen, start VIBLaces. You will see a blank Design Window. Add two SwingTextArea beans as shown to display the SECS messages. This will form the basic “diagnostics” of the user interface, so you can see the SECS messages being received and sent. Add the three SwingLabels; change their labels to Diagnostics, rec’d and sent. Add one more SwingLabel for the “Press Control” heading.

Add a VIB Box and a SevenSegmentDisplay which will be used to display the current ram position. Set the parameters for the Box as follows: Color For Fill (Red), Fill Direction (Top), Maximum (50), Panel (Etched), Show Label (False). Set the SevenSegment to only display whole numbers: Number of Digits (3), Number of Decimal Digits (0).

Add a Button and change its Label For Off and Label For On to “Cycle”. The button trigger type should be set to “Timed” so that the signal to start the press is sent and then goes low again.

This completes the graphical user interface layout, but now we need to work in the VIBLaces “Diagram” Window to complete the logic for this interface. Save the current VIBLaces screen (it will save as a file with a “.frm” extension). It is a good idea to occasionally save your work.

The Logic Design

The Diagnostics

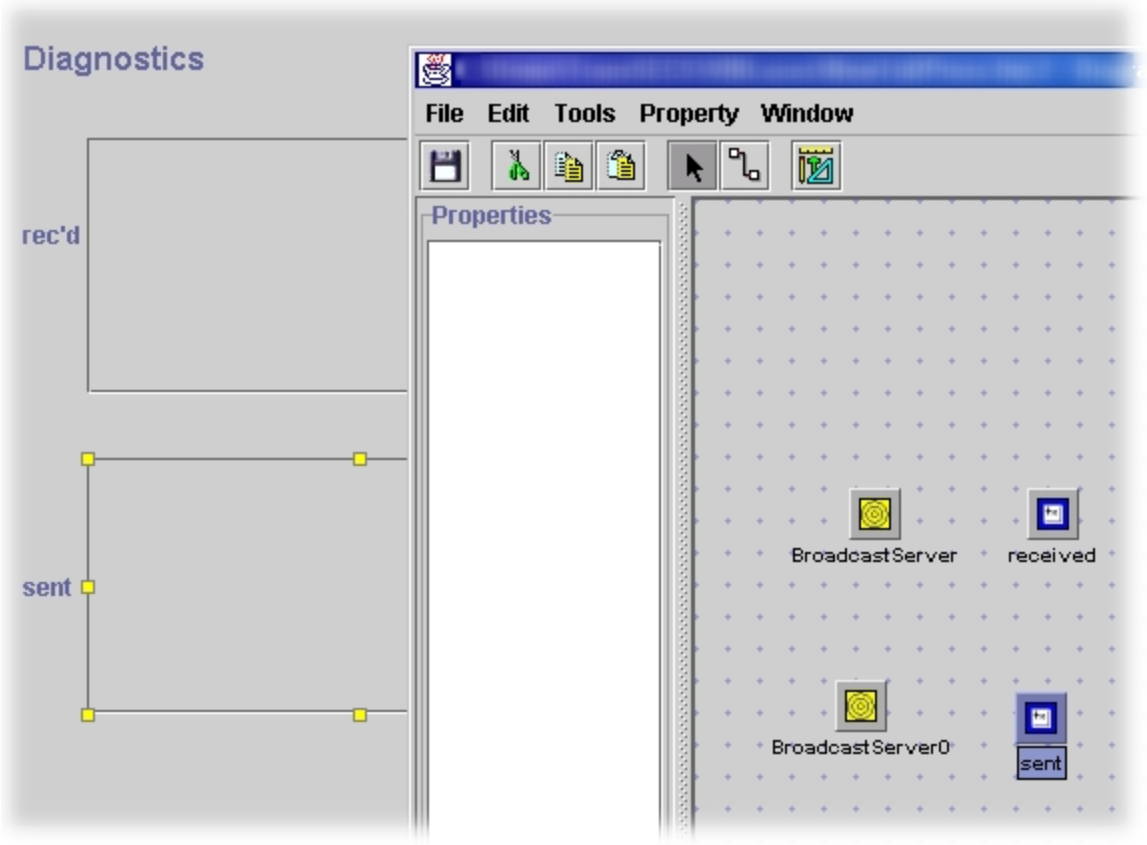
Open the VIBLaces Diagram Window by clicking on the icon that looks like a yellow box wired to a red and green box. The Diagram Window looks similar to the Design Window, but has grid points instead of lines. Select the two SwingTextAreas and “copy to diagram” using the Edit->Copy To Diagram command, or simply type CNTL-D when you have both selected. You will see the two SwingTextArea icons appear in the Diagram Window. In the next illustration you will see that these have been labeled “received” and “sent”. To change the label from the defaults (SwingTextArea and SwingTextArea0), double click on the icon in the Diagram Window and you can enter the name to what you want. To see which icon corresponds to which SwingTextArea in the Design Window, select one of the icons and observe which SwingTextArea is highlighted in the design.

Next, as shown in the figure below, we need two Broadcast Servers which will be used to store the current SECS messages in SML format as they are received from the TransSECS communication layer. Arrange the BroadcastServers to the left of the SwingTextArea icons as shown in the figure. Message logging is fully documented in the TransSECS User Reference. We need to configure the BroadcastServers as for message logging:

- Select the BroadcastServer to the left of the SwingTextArea icon for displaying received messages. Enter “TnDGRPRESS.Passive.InMsg” into the Attached Servers field of this BroadcastServer.

- Select the BroadcastServer to the left of the SwingTextArea icon for displaying sent messages. Enter “TnDGRPRESS.Passive.OutMsg” into the Attached Servers field of this BroadcastServer.

Quick connect the BroadcastServers to their respective SwingTextAreas. To quick connect, select the menu icon that looks like two boxes connected by a single wire. Select the BroadcastServer and drag the wire to the SwingTextArea and release the mouse to make the connection.



Simple SECS Communication Test

We are ready to do a simple SECS communication test. Go to the TransSECS application and select the S1F1 (“Are You There”) message. Click the “Send Message” button at the bottom of the right hand panel. This exercise serves two purposes: it establishes a communication link between the Host (TransSECS) and the Equipment (the VIBLaces application), and it demonstrates that the communication link has been established by the response of the message.

After you press the button in TransSECS, you should see something like:

```
SENT:
S1F1 W  .

RECEIVED:
S1F2 <L[2]
  <A 'Ergo'>
  <A '1.3a'>
>  .
```

in the TransSECS message area above the button.

In VIBLaces you should see the S1F1 in the “rec’d” SwingTextArea and the S1F2 reply in the “sent” SwingTextArea.

This message exchange confirms the SECS connection. The next part of this tutorial will cover how to get data from the OPC Server, and then we will discuss how to combine SECS messages with this information. Save the current VIBLaces screen. Both the logic (Diagram Window) and the GUI (Design Window) will both be saved when you save your work.

Using OPC

Adding an OPC Client Bean

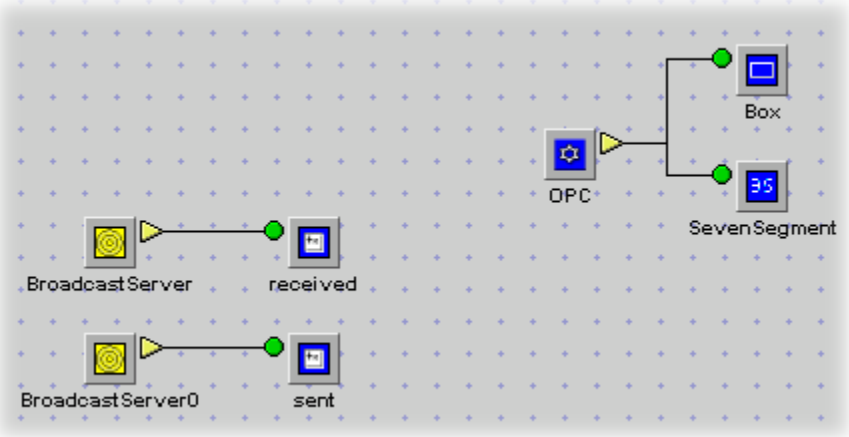
Make sure the OPC Gateway is running. If you start the Trial OPC Gateway, it will time-out in one hour and will need to be restarted. When you run the OPC Gateway you will see a command shell (DOS shell) with a message that says:

```
started web server on port 5113
OPCRemote :  bound in registry
```

You may minimize this window, but do not close it. It must remain running so that the VIB components can make the connection to the Think & DO OPC Runtime.

Since we are going to be using the Think & Do GRPress example, you need to start the Think & Do Run-Time Engine (if you have not already done so), and select the Gear Press Control project. Once you have it running, make the Think & Do Screen about quarter full-size so you can see what is happening and still be able to work with VIBLaces and TransSECS.

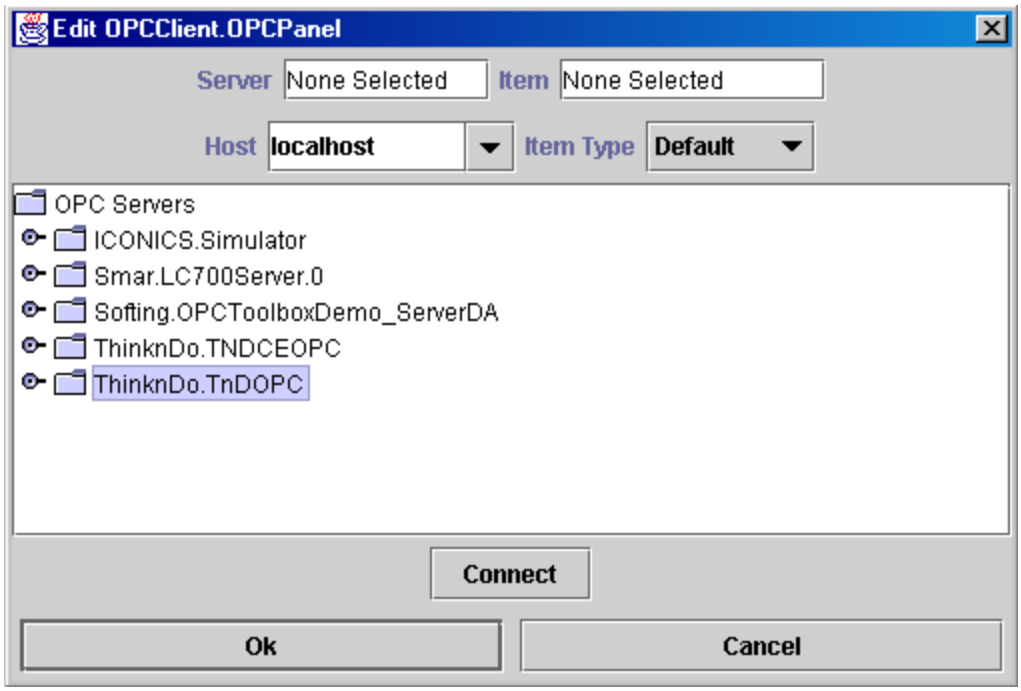
We will make an OPC connection from the Think & Do server to our Box and Seven Segment display to display the ram position. Select the Box and SevenSegment in the Design Window and use Edit->Copy to Diagram (or CNTL-D). You will see the two icons in the Diagram Window. Move them to the approximate location shown in the figure below. Find the OPC Client bean in the palette (under OPC). Drag and Drop the OPC bean to the left of the two graphics icons, as shown. You may quick connect from the OPC bean to the Box and SevenSegment, as shown.



Next we need to configure the OPC bean so that it connects to the ram position tag in the Think & Do server. Select the OPC icon and look at the Properties list. Select the “Browse Servers” property and click on the “…” in the field to the right. This will bring up a browser panel which will be used to select the OPC item for the connection.

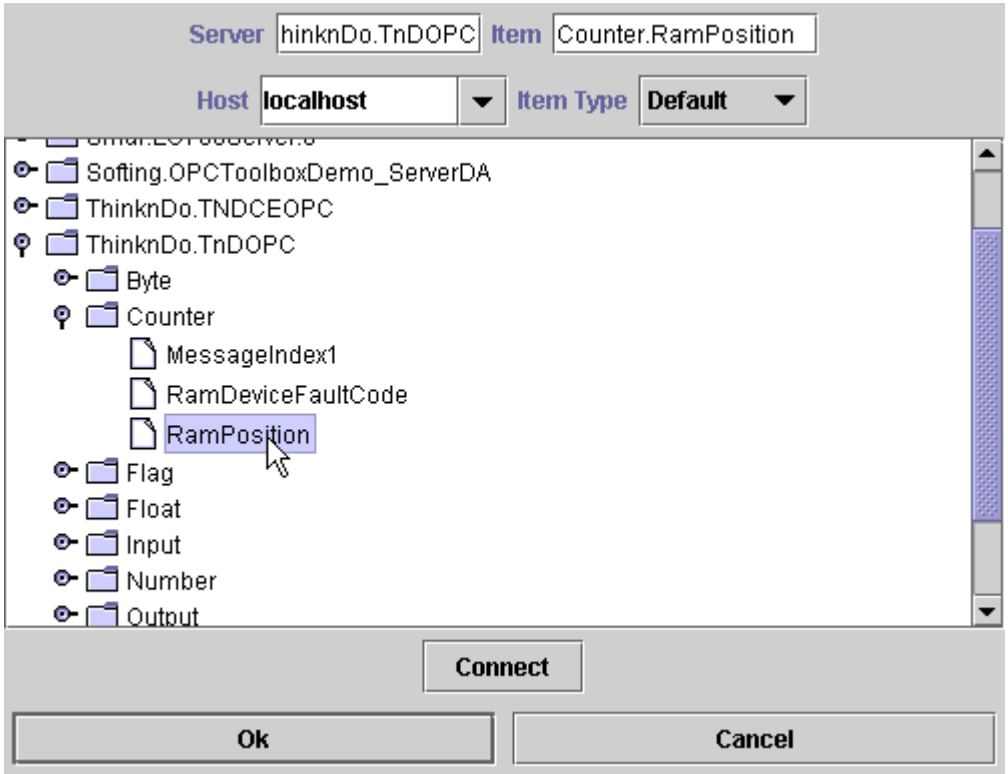
To use the OPC browser, first enter “localhost” into the Host name. You must enter a value into this field even if it displays the correct information. Entering “localhost” tells the browser to look on this local computer for a list of all registered OPC Servers. Before the OPC Servers are listed, you will see a “(Host Not Configured)”. As soon as you enter a valid Host name, this should change to “OPC Servers”, as seen below.

In the next figure, several OPC servers are visible including the Think & Do sever. Select the *ThinknDo.TnDOPC* sever as shown.



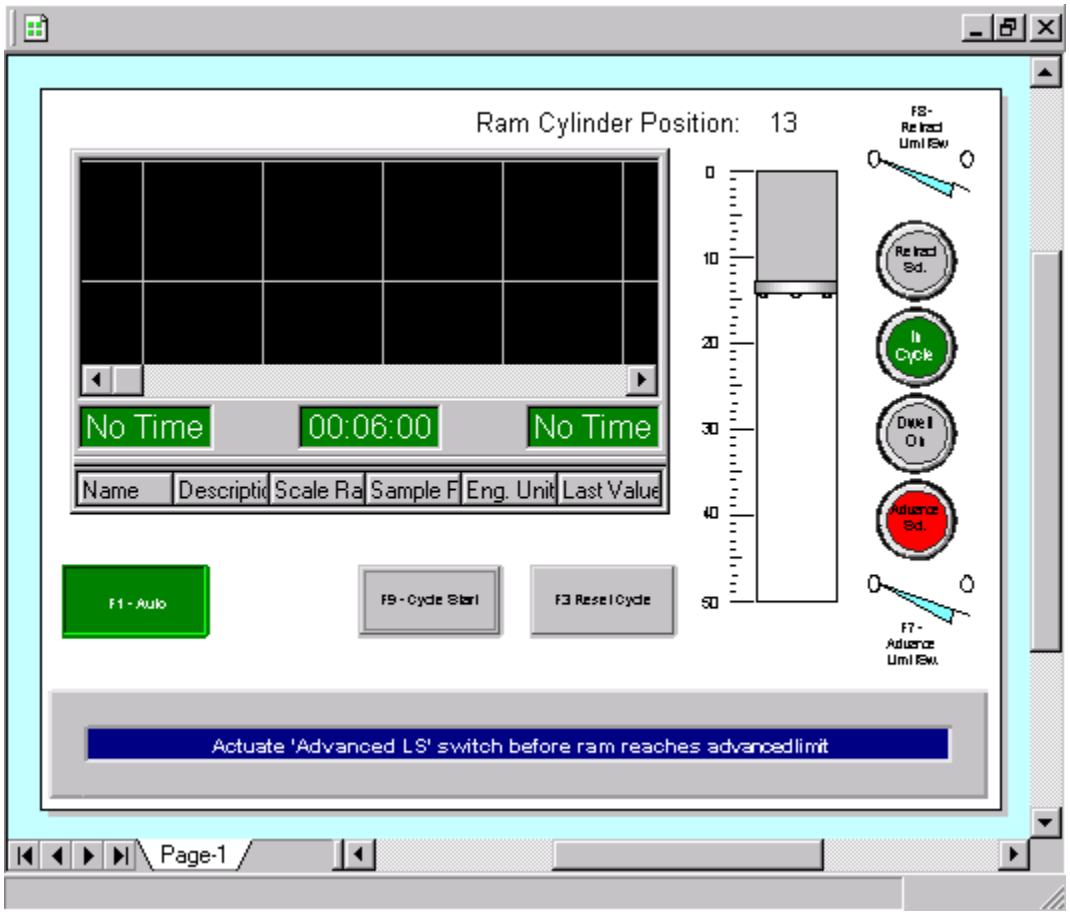
Double click on this server name and the item list will be displayed. The particular item we are interested in, the ram position, is located under “Counters”. Browse through Counters until you find the “RamPosition” item, as shown in the figure below.

Next you must press the “Connect” button at the bottom of the browser. You will be presented with a confirmation panel – just press the “OK” button and the connection will be made.



There is one final configuration step for the OPC bean. We need to give this bean a “tag name” so we can reference the value elsewhere (we will eventually use the data from this server in a SECS message). Select the OPC Client bean and enter “ram_position” in its Tag Name.

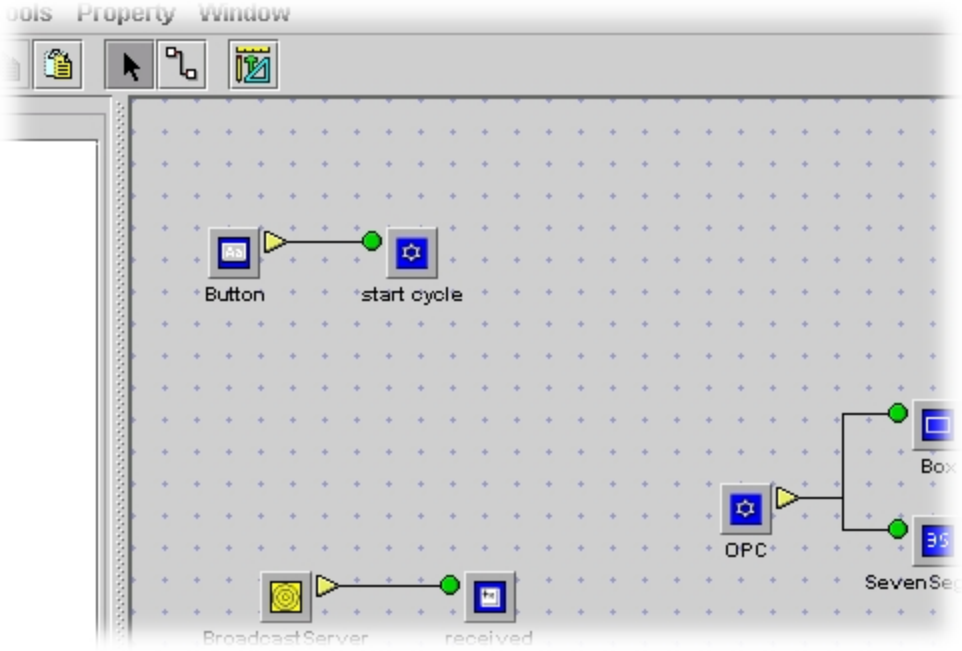
To test the OPC connection, look at the Box and Seven Segment in the Design Window. In the Think & Do press screen, press the button that says “F1 Manual” until it says “F1 Auto” (and is green). Then press the “F9 Start Cycle” button and the ram will start to move. The OPC Client bean will read the OPC Server every 1.0-second. This is controlled by the “timebase” of the OPC Client. You can change this to 0.5 second for a faster update rate.



Controlling the Press with VIB

The next step is to use the “Cycle” button on the VIB screen to start the press cycle. We need to add one more OPC bean to the logic design to take the Button press and send it to the Think & Do GRPress demo to start the press (just as if you pressed the F9-Start Cycle button on the Think & Do screen).

Select the Cycle Button from the VIBLaces Design Window and “copy” this to the Diagram Window (remember -- ***not*** CNTL-C, CNLT-V) using CNTL-D or Edit->Copy To Diagram. Place it near the upper left hand corner of the logic screen, as shown in the next figure. Add an OPC bean as shown and quick connect from the Button to the OPC bean. Configure the OPC bean to use the ThinknDo item Flag.SimulateCycleStart.



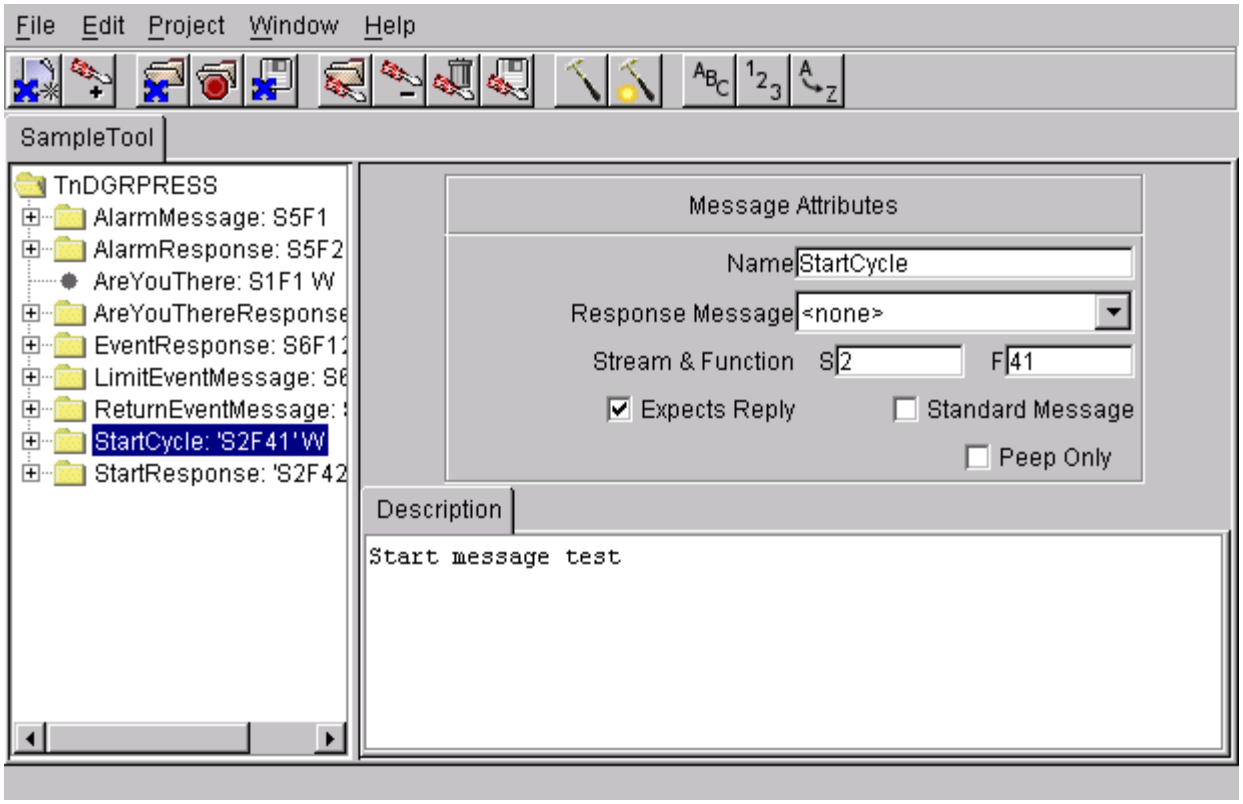
Test this OPC connection by pressing the Cycle Button on the VIBLaces screen. You will need to put VIBLaces into “Run” mode (press the running person button) before you can press the button. When you press the button you should see the Seven Segment display and Box update with the ram position as the Think & Do screen also updates.

The next step is to add SECS messages to the logic so that our SECS Host simulator (TransSECS) can control the press and receive SECS messages from the equipment application.

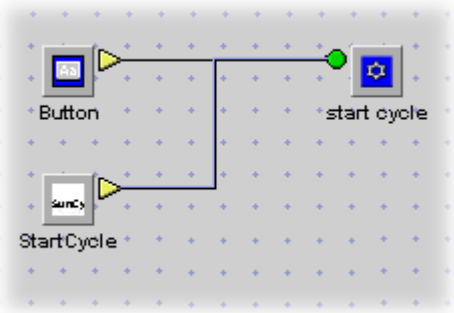
Adding SECS

StartCycle Message

The first SECS message we will add to the VIB logic is the StartCycle. Adding this primary SECS message to the logic will make allow us to respond to the Host's request to start the ram. First, look at the message in TransSECS (you will need to click the Run Mode button in TransSECS to look at the message structure).

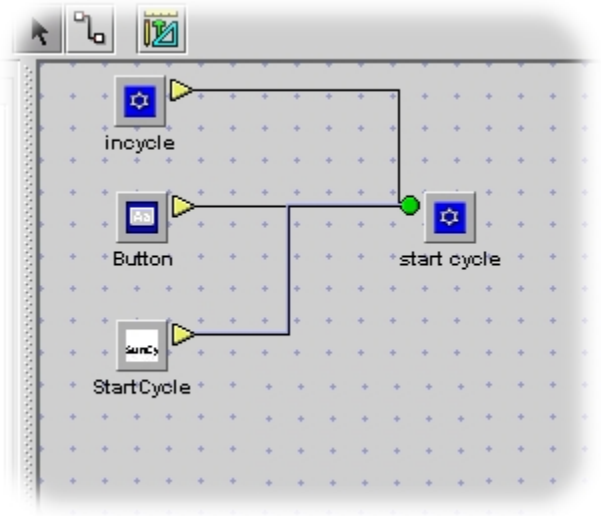


Note that this message "Expects Reply" and has no auto response message. Since it expects a reply, when the Host sends these messages it expects an S2F42 to be sent within the timeout period. We will take care of this later.



We want to trigger the ram cycle when the StartCycle SECS message is sent to the VIB application. All we need to do is add the S2F42 (StartCycle) message to the logic. Position it near the Cycle Button as shown and quick connect it to the OPC bean that starts the ram cycle. You can test this configuration by going to TransSECS and selecting the S2F41 message and sending it (press the Send Message button). Make sure TransSECS is in Run Mode or you won't have the Send Message button present.

You will see the message sent in TransSECS, and received (in the rec'd diagnostic box) in the VIB GUI. With this configuration you cannot send the S2F41 message again from TransSECS because the OPC Server item controlling the start cycle needs to be cycled low to restart the cycle. We can solve this problem by adding another input to the start cycle OPC bean to tell it when the cycle is complete.



The figure to the left has another OPC bean added to it to read the “InCycle” status of the ram. When the ram is moving the “InCycle” item is high (Boolean true) and then it goes low (Boolean false) when the ram is finished. This is what we need to reset the start cycle signal from the Host to the Think & Do start cycle simulator.

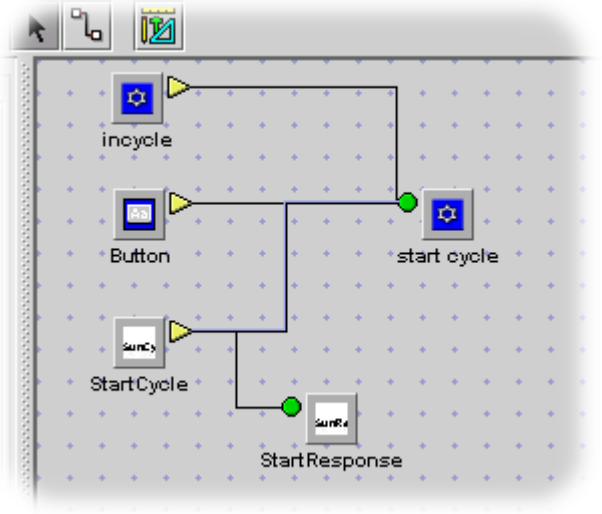
Configure the newly added OPC bean to use the Output.InCycle item of the ThinknDo OPC server. Also change its Tag Name to “in_cycle” (we will use this later). Now when you send the S2F41 message from the Host (from TransSECS) the ram will start again.

Adding a SECS Response Message

There is one more SECS message that we need to add to the logic. We alluded to this when we looked at the definition for the S2F41 message in TransSECS. The Host expects a reply when this message is sent; the reply is already defined in the sample, S2F42. Up to this point, if you look in the received messages Diagnostics in the VIB screen, you will see that for each time we were sent an S2F41, we received an S9F9 after a timeout period. This was sent by the Host, indicating a timeout condition.

All we need to do is to add an S2F42 message to the logic, triggered by the receipt of the S2F41 message. This is show in the figure to the right. The StartCycle message bean is quick connected to the StartResponse bean.

Now when the Host sends the S2F41, our application sends back an S2F42 and we no longer receive the S9F9 (timeout) messages.

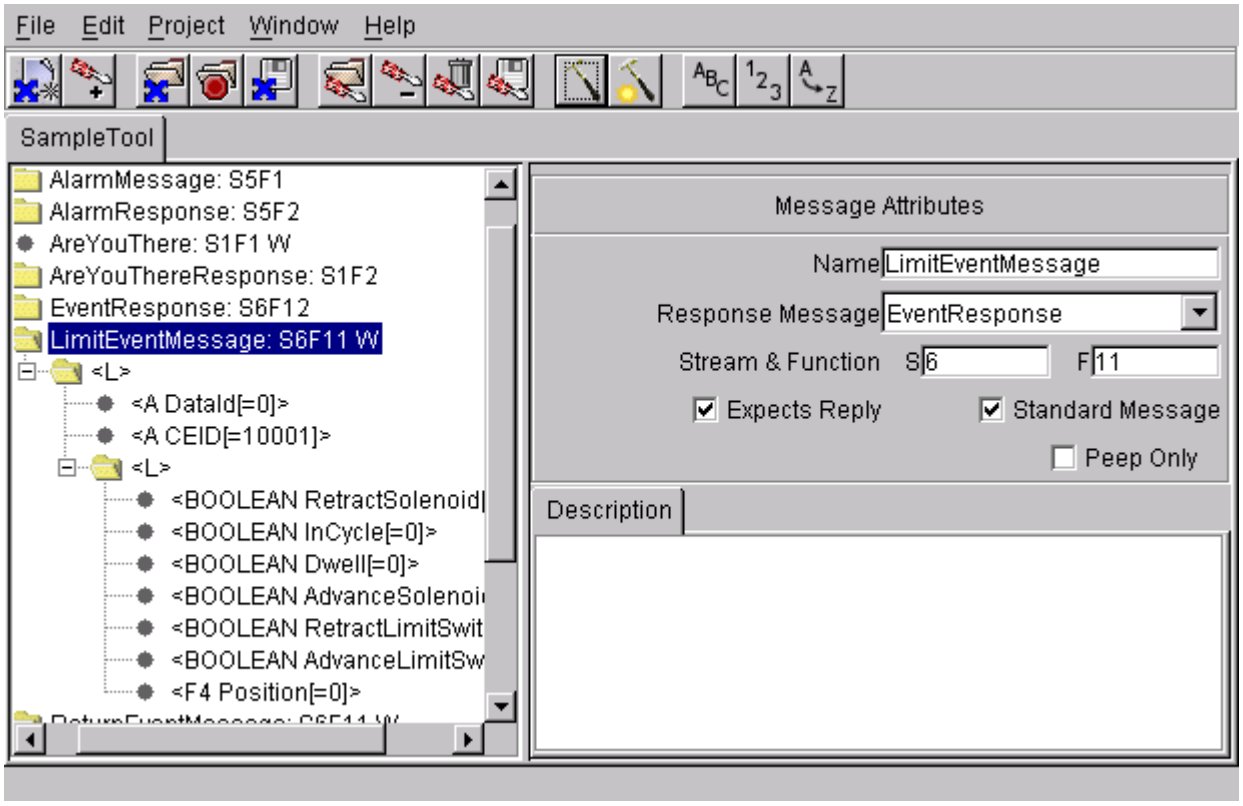


A SECS Event Message

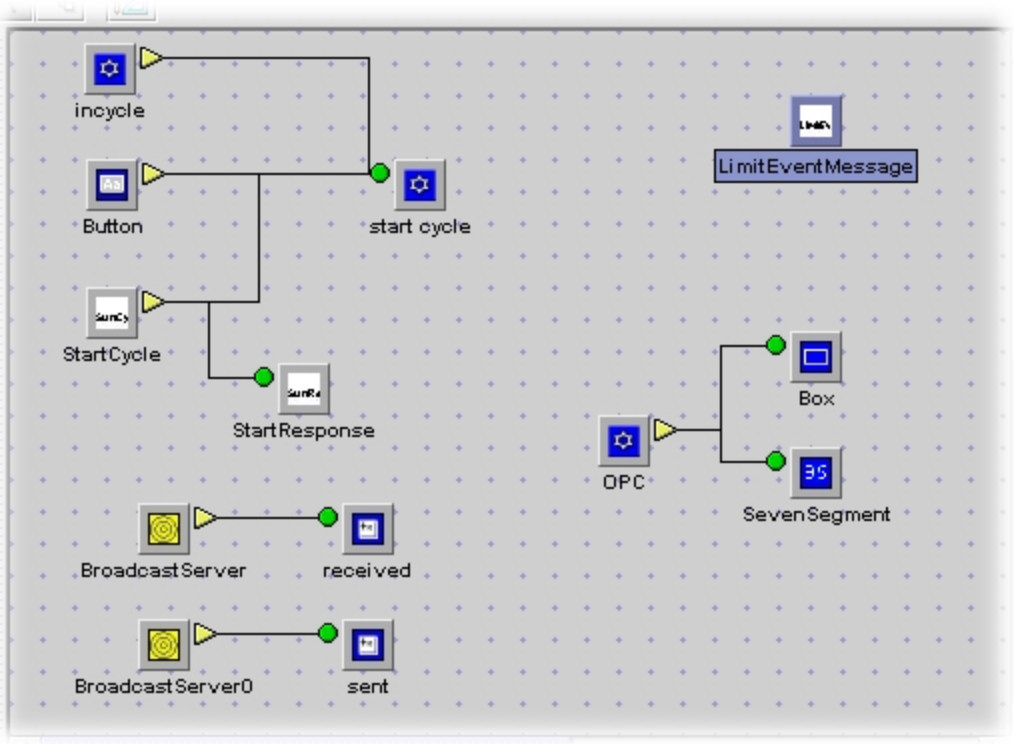
In this next example we will add data from the OPC Server to a SECS message that is sent to the Host. We will add an S6F11 message to the VIB logic and configure it by combining data from the Think & Do OPC Server. This report message will be sent to the Host every time the ram completes a cycle.

Before we start, take TransSECS out of Run Mode and look at the pre-defined S6F11 messages. There are two defined S6F11 messages, and we will use the one called *LimitEventMessage*. Select

this message. Notice that this message *Expects Reply* and has an auto-response message selected (EventReponse). The auto-repsonse means that the Host will automatically send the S6F12 (EventResponse) when it receives an S6F11. While the S6F11 is selected, expand the items on the message tree and notice that there are a number of data attributes in the list. All are BOOLEAN (such as RetractSolenoid, InCycle, Dwell, etc.) except for one F4 data type (a Float) for "Position".



You will see shortly that you will be able to connect a VIB data source (i.e., an OPC bean) to any or all of the data items in the list. Put TransSECS back into Run Mode and reselect the S2F41 message (so we are ready to start the ram again).

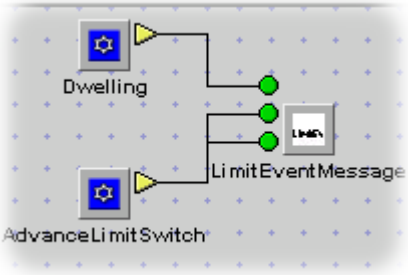


Add the S6F11 (LimitEventMessage) bean to the logic as shown in the next figure above.

Select the S6F11 bean and look at the properties. You will see each of the data items as a “Server List”. These Server Lists fields can be used to connect to data sources using tag names. Recall that when we configured the OPC bean to display the ram position we entered “ram_position” in its *Tag Name* property. We also entered “in_cycle” for the Tag Name for the OPC server that was connected to the InCycle OPC data item. These two tag names can be used to fill in the appropriate fields, as shown in the figure below.

Property Name	Property Value
Attached Servers	
Tag Name	
AdvanceLimitSwitch Server List	
AdvanceSolenoid Server List	
CEID Server List	
DataId Server List	
Dwell Server List	
InCycle Server List	in_cycle
Position Server List	ram_position
RetractLimitSwitch Server List	
RetractSolenoid Server List	

Customize...



We can add more OPC beans to collect the information for the other fields. These can be attached as tag names or quick connected. In the illustration to the left, two more OPC servers have been added. These two beans use the Flag. DwellingAtAdvancedPosition and Flag.RamAtAdvancedLimiteSwitch OPC items. The OPC bean that reads the Dwelling condition also triggers the message (it is quick connected to the “Data Connection”). You can also test the message by triggering it with a regular VIB Button at any time.

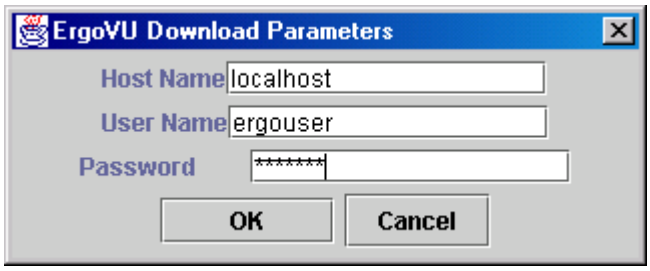
Deployment

The equipment application you have designed and tested in VIBLaces must be generated and deployed to be of general use. It is highly recommended that you use ErgoVU to deploy the application. ErgoVU will run the logic you designed in the Diagram Window continuously, while the user interface is available either as a front-end display from ErgoVU or as an applet in a Web browser with a Java 2 plug-in (if we had not used Swing components, any Web browser could be used to view the applets).

To deploy an ErgoVU application, first start ErgoVU on your system. The Trial version of ErgoVU will only run for two hours, and then needs to be restarted. ErgoVU is started by the shortcuts on your desktop or by using the Windows Start Menu (under ErgoTech). Starting ErgoVU also starts a web server on your system. If you already have a Web server on your computer you will either need to temporarily stop it, or reconfigure ErgoVU's web server to use a port other than the default (port 80). More about reconfiguration can be found in the ErgoVU documentation. The following deployment steps assume that you do not need to reconfigure ErgoVU's web server

When you first start ErgoVU you will see a "partial" screen view that is blank. This partial screen will be filled with your user interface after we have generated the code and downloaded it to ErgoVU.

To generate the ErgoVU code, use the Code Wizard option at the top of the Diagram Window. Chose "ErgoVU Application" and then press the "Next" button. In the next panel, you can leave everything with the defaults for testing purposes. Press "Finish" and the code will be generated. After the code generation, you will see an FTP download panel for ErgoVU. Fill it in with the following: "localhost" for the host name, "ergouser" for the user name, and "default" for the password. Press OK to download the application to ErgoVU.



You will see the partial screen fill with your user interface. There is one more important step that we must take to test the application in ErgoVU. First, we must close VIBLaces and ErgoVU. This is because both VIBLaces and ErgoVU are trying to make connections to the Host application. VIBLaces has had the connection for some time, while we were building and testing the equipment application. There can only be one connection to the host at a time (with the same device id). So we must stop VIBLaces and restart ErgoVU to let ErgoVU use this connection to the Host. To stop ErgoVU you must close both command shells that are started when ErgoVU starts. One is the TiniHttpServer (the web server) and one is ErgoVU.

After you have stopped ErgoVU (and VIBLaces), restart ErgoVU. ErgoVU will automatically load the user interface applet called NoName (that is why we used the defaults). You may resize this partial screen view as desired. The user interface is not really necessary for this SECS application since the logic makes the OPC connections and handles the SECS messages. If you want to you can use the user interface to monitor the SECS messages (with the Diagnostics text areas), or to start the ram manually with the Cycle button.

This concludes the Think & Do OPC and TransSECS tutorial. You should read the examples in the TransSECS User Reference and the regular VIB and VIBLaces guides to learn more about these tools and the VIB components.

