Application Note

Altia® Design and FacePlate

Simulation Graphics Software

The Altia and *I-Logix Statemate* Generated Code Connection

Overview

The Altia/Statemate connection is designed to allow the simulation of Statemate models to be driven from Altia panels. A connection to Statemate generated code is also provided.

The Altia/Statemate simulator connection is controlled by a program that has been written using Altia's C/C++ application programming interface (API) and a C API provided by I-Logix for the Statemate simulator. The program connects elements that appear in a Statemate model to external signals in an Altia panel. For connected elements, changes made in the Statemate simulation are sent to the Altia panel and changes made in the Altia panel are sent to the Statemate simulation.

This application note deals exclusively with the process of taking an Altia interface that is connected to a Statemate simulation and converting it to work with Statemate generated code. For information regarding how Altia is first interfaced with a Statemate simulation, please see the tutorial on our web page at http://www.altia.com by choosing Online Support, then Tutorials. A copy is also available in the Altia software stmm/docs directory if you've already installed the Altia/Statemate connection package.

Connection Software Compatibility with Statemate 5.x and 6.x

Altia no longer supports connections for Statemate 5.x or 6.x. You must be using Statemate MAGNUM to use the Altia/Statemate connection described in this document.

The Altia Connection to Statemate Generated Code

After developing a Statemate model and verifying its correctness using the simulator, it is common to transition to generated code. This has two major benefits. First, the model execution performance is significantly improved. Second, the generated code version of the model can run on systems that do not have a Statemate MAGNUM license. In a large organization, this is a significant benefit if several different working groups, possibly in different geographical regions, must have execution privileges.

If an Altia interface has been developed for a Statemate model, distribution to different groups involved in the product development becomes even more desirable because individuals can exercise the model using a "near to real life" interface. For this reason, the Altia/Statemate connection software includes the ability to transition from a simulator connection to a generated code connection with a small amount of effort.

To support the transition, Altia provides a special user_activities.c file in the connection package's stmm/code_link directory. You will find this directory under your Altia product software directory if you've properly installed the Altia/Statemate connection package. The user_activities.c file can be compiled and linked with the code generated by Statemate. When Statemate generates code, it always generates a simple user_activities.c file. Normally, a developer customizes this file if they wish to interface the generated code to external data and/or events (e.g., reading or writing to files or I/O devices).

Altia's user_activities.c file interfaces generated code to Altia and it requires no customizations by the developer. Instead, a simple C header file, bindings.h, is created by the developer (from an easy to follow template) to provide bindings information for Altia's user_activities.c just like the .mxc file provides bindings information for the simulator connection program.

To demonstrate the generated code connection, the Altia/Statemate connection software includes the generated code for the HVAC model in the stmm/demos/hvac_with_codegen sub-directory. This code was generated from the HVAC model using a Compilation Profile named HVAC, which is included in the Statemate HVAC model databank. Altia's special user_activities.c file is pre-installed in the hvac_with_codegen directory along with a bindings.h file that has been edited to work with the HVAC model generated code.

Running the HVAC Generated Code Connection Demonstration

In a command prompt window, you should change directory to the hvac_with_codegen directory located under the stmm/demos Altia/Statemate connection software directory. Alternatively, you can view the contents of the hvac_with_codegen directory in a file browser window. This directory already contains files for a generated code connection in the form of a user_activities.c source file and bindings.h is a C form of the hvac.mxc file (also in the same directory) used by the Altia/Statemate simulator connection program. Like the hvac.mxc file, the bindings.h file is usually specific to an Altia user interface and Statemate model combination and needs to be customized for different user interfaces and models.

The generated code for the HVAC example is already compiled and linked into an executable program. All you have to do is run this executable (hvac.exe on the PC, hvac on UNIX). Based on the information provided by bindings.h, the hvac program (more accurately, the code created when user_activities.c is compiled and linked to create the hvac program) knows how to start the Altia Runtime. The bindings.h file defines the top of the Altia software directory tree (../../..). From this, the location of the Altia Runtime (../../../bin) is determined. The bindings.h file also contains the equivalent of the .mxc file **start** directive that tells the code compiled from user_activities.c to load hvac.dsn from the current working directory into the Altia Runtime.

You may now exercise the generated code from the Altia interface the same way you exercised the model running in the simulator.

If you have the complete Altia Design or FacePlate product installation, you also have the option of starting the Altia Design or FacePlate editor. To do this, modify bindings.h and change the setting of the `start_mode` variable. From a command prompt window, execute make (nmake on the PC) to recompile user_activities.c and relink the hvac program. You may have to remove the object file for user_activities.c (user_activities.o on UNIX or user_activities.obj on PC) before the make to force a recompile because the make files generated by Statemate don't know that the object file for user_activities.c depends on bindings.h. When the make completes, execute the new hvac program. When the editor opens, put it into **Run** mode to interact with the generated code program.

Creating Your Own Altia Connection to Statemate Generated Code

After an Altia user interface design has been successfully connected to a Statemate model running in the simulator, you can easily connect the same design to the code generated for the Statemate model using the following steps as a guide:

1. Copy the user_activities.c and altia.h files from the Altia/Statemate connection stmm/code_link directory to the directory containing the generated code for your own Statemate model.

A SPECIAL NOTE IF YOU HAVE GENERIC CHARTS IN YOUR STATEMATE MODEL

You should copy your existing user_activities.c file to user_activities.c_temp if user_activities.c_temp does not already exist. If it already exists, no extra steps are necessary. You can also create a user_activities.c_temp automatically by simply regenerating code from Statemate.

It is important to have a copy of this generated version of user_activities.c renamed to user_activities.c_temp. The .c files generated for Generic Charts (g_*.c) need to reference its contents. The Altia version of user_activities.c contains special C preprocessor lines to reference user_activities.c_temp when user_activities.c is referenced by a Generic Chart .c file.

2. The user_activities.c file requires no modifications; however, you must create a bindings.h file specific to your Altia user interface design and Statemate model. Open your Altia user interface design file into Altia Design or FacePlate, choose the **Statemate** menu, and select the **Create bindings.h** option.

   This creates a bindings.h file in the same directory as your Altia user interface design file. Copy the bindings.h file from this directory to the directory containing the Statemate generated code if the directories are different.

   From the Statemate generated code directory, open bindings.h into a text editor window. For example, on the PC you could use Notepad or Wordpad. Confirm the path to your Altia user interface design file by studying the line that looks like:

   ```
   static char *start_design_file = "tutorial.dsn";
   ```

   If the design file (tutorial.dsn in this example) is in the same directory as the Statemate generated code or you plan to copy the design file and its associated .rtm file to the same directory, then make no changes. Otherwise, add the complete path to the design file. For example, on UNIX, you might replace tutorial.dsn with /projects/tutorial/tutorial.dsn. On the PC, you might replace tutorial.dsn with c:\projects\tutorial\tutorial.dsn. Save the file if you make changes to it.

   In a later step when you make the sources into a program, you may have to edit this file again if you get one or more "undeclared identifier" errors. See step 4 for more details.

3. You are now ready to modify the Makefile Settings for your Statemate Compilation Profile so that you can make a program that incorporates the code from Altia's user_activities.c file and the new bindings.h file. Open your Compilation Profile in Statemate (hint: you begin by choosing the **C Code Generation** icon from the main window of Statemate MAGNUM). From the **Options** menu in the Statemate code generation window, choose the **Makefile Settings…** option.

   In the **Libraries:** field of the Makefile Settings dialog, UNIX users should add:

   ```
   /usr/altia/lib/liblan.a
   ```

   PC users should add:

   ```
   c:\usr\altia\lib\ms32\liblan.lib
   ```

   If the Altia product software is NOT installed in /usr/altia on UNIX or c:\usr\altia on the PC, change the path accordingly. Press the **OK** button in the Makefile Settings dialog after the changes are complete.

   You must generate new code from Statemate to incorporate the new changes into the generated Makefile file. To do this, open the **Compile** menu from the Statemate code generation window and choose the **Generate Code…** option.

4. Now you are ready to make the program from Statemate. In the Statemate code generation window, open the **Compile** menu and choose the **Make Code…** option. A new window will display the progress of the make as each source file is compiled and the object files are linked into a program.

NOTE

If there are "undeclared identifier" errors from bindings.h, you must edit the bindings.h file and make some changes. Such errors indicate that one or more of the Statemate variable names defined in the file's bindings[ ] array require special extensions like acy, ev1, or co1.

The special extensions that Statemate has given to these variable names can only be determined by searching the .h header files referenced by all.h. For example, if an error occurs because POWER is undeclared, do a search of the .h files for POWER. You can use a file browser's Find or Search utility. On UNIX, you can also use grep. You might find that Statemate has assigned a name like POWERev1 to the variable. You would then edit bindings.h and replace any occurrence of &POWER with &POWERev1. After making changes, save the file and choose **Make Code…** again from the Statemate **Compile** menu.

Repeat the edit, save, and make until you have all of the undeclared identifier errors resolved. Now that bindings.h is customized, be careful not to overwrite it if you do another **Create bindings.h** from Altia's **Statemate** menu. This will permanently remove all of your changes.

WARNING

WHENEVER YOU CHANGE bindings.h AND YOU ARE READY TO DO A MAKE, YOU ALSO MUST REMOVE THE OBJECT FILE FOR user_activities.c (user_activities.o on UNIX or user_activities.obj on PC) and the tmp_out_lib library (tmp_out_lib.a on UNIX or tmp_out_lib.lib on PC) IF THEY EXIST. This will force a recompile of user_activities.c. The make files generated by Statemate do not know that the object file for user_activities.c depends on bindings.h. You can add the following line to the end of the User_Makefile:

```
user_activities.o:      bindings.h
```

On the PC, it would be:

```
user_activities.obj:    bindings.h
```

This line tells make that user_activities.o(bj) depends on bindings.h so if bindings.h is newer, user_activities.c needs to be recompiled. Even with this change, you will still need to manually remove the tmp_out_lib library if it exists.

5. If the make completes successfully, you will have a new program in the current directory that is ready for execution. The name of the program depends on the name of the Statemate Compilation Profile used to generate the code. On UNIX, the program will have no extension. On the PC, the program will have a .exe extension. The new program contains the Statemate generated code as well as the Altia user_activities.c file and bindings.h information. Because the bindings.h file is linked directly into the program, you ***must*** remake the program and restart it if changes are made to bindings.h.

6. If the start_design_file variable in your bindings.h file refers to a non-empty string, the initialization portion of user_activities.c will attempt to start Altia Runtime and request it to open the design file designated in the string. For this to work properly, the Altia Runtime executable (altiart.out on UNIX or altiart.exe on the PC) must be accessible to the generated code program. If the full Altia Design product software is installed in /usr/altia on Unix or \usr\altia on the PC, Altia Runtime is automatically accessible because it resides in /usr/altia/bin or \usr\altia\bin. If the full Altia FacePlate product software is installed in /usr/altiafp on UNIX or \altiafp on the PC, Altia Runtime is also automatically accessible. If this is not the case, it may be easiest to simply copy the Altia Runtime executable to the local directory. You can copy this program from the Altia/Statemate connection stmm/code_link directory.

   On UNIX systems, you must also copy fonts.ali from the stmm/code_link directory if it is present. On PC systems, you must copy fonts.ali and colors.ali from the directory. Failure to copy these files, if they are present, along with the Altia Runtime executable will result in unusual looking fonts and possibly unusual colors when a design opens into the Altia Runtime window(s).

   Another alternative to copying the runtime files to the local directory is to set the ALTIAHOME variable in the bindings.h file if you have a full Altia product software installation. For example, if the Altia software bin directory is /opt/altia/bin, set ALTIAHOME to /opt/altia.

   If you choose to disable the start_design_file variable in bindings.h by setting it to an empty string (static char *start_design_file = “”;), you must manually start Altia Runtime or the Altia editor before executing the generated code program.

   In any case, if you make changes to bindings.h, be careful not to overwrite it if you do another **Create bindings.h** from Altia’s **Statemate** menu. This will permanently remove all of your changes.

7. After starting the program, you can exit it by closing the associated Altia window (for an Altia Runtime session, a **Ctrl+C** key sequence within the Altia window will close Altia Runtime). If you start the program from a command prompt window, you can also stop it with a **Ctrl+C** in the command prompt window.

Packaging Files for Running on Other Systems

To run the generated program on other compatible systems, you need the generated program, Altia Runtime files, and Altia user interface design files.

As an example, the files needed to run the HVAC demonstration on any Windows NT or 2000 machine are:

hvac.exe altiart.exe colors.ali fonts.ali hvac.dsn hvac.rtm

For one of the supported UNIX platforms, the files are:

hvac altiart.out fonts.ali hvac.dsn hvac.rtm

Files Summary

The Altia HVAC Statemate directories and files for the HVAC demo model are:

stmm/demos/hvac_with_codegen/*

The Altia Runtime program and support files are:

stmm/code_link/altiart.out (altiart.exe on the PC)

stmm/code_link/fonts.ali
stmm/code_link/colors.ali (PC only)

The Altia/Statemate simulator connection program and support files are:
stmm/bin/*

The Altia/Statemate generated code link sources are:
stmm/code_link/user_activities.c stmm/code_link/altia.h

The Altia/Statemate documentation files are:
stmm/docs/*

Additional Questions?

If you have additional questions, feel free to contact an Altia Technical Support Representative at 719-598-4299 in the United States or send email to support@altia.com. If you are located outside of the U.S., contact your Altia software distributor.