MATRIXx™

SystemBuild™ HyperBuild User Guide

NATIONAL INSTRUMENTS™

April 2004 Edition
Part Number 370764B-01

Worldwide Technical Support and Product Information

ni.com

National Instruments Corporate Headquarters

11500 North Mopac Expressway   Austin, Texas 78759-3504   USA   Tel: 512 683 0100

Worldwide Offices

Australia 1800 300 800, Austria 43 0 662 45 79 90 0, Belgium 32 0 2 757 00 20, Brazil 55 11 3262 3599,
Canada (Calgary) 403 274 9391, Canada (Ottawa) 613 233 5949, Canada (Québec) 450 510 3055,
Canada (Toronto) 905 785 0085, Canada (Vancouver) 514 685 7530, China 86 21 6555 7838,
Czech Republic 420 224 235 774, Denmark 45 45 76 26 00, Finland 385 0 9 725 725 11,
France 33 0 1 48 14 24 24, Germany 49 0 89 741 31 30, Greece 30 2 10 42 96 427, India 91 80 51190000,
Israel 972 0 3 6393737, Italy 39 02 413091, Japan 81 3 5472 2970, Korea 82 02 3451 3400,
Malaysia 603 9131 0918, Mexico 001 800 010 0793, Netherlands 31 0 348 433 466,
New Zealand 0800 553 322, Norway 47 0 66 90 76 60, Poland 48 22 3390150, Portugal 351 210 311 210,
Russia 7 095 783 68 51, Singapore 65 6226 5886, Slovenia 386 3 425 4200, South Africa 27 0 11 805 8197,
Spain 34 91 640 0085, Sweden 46 0 8 587 895 00, Switzerland 41 56 200 51 51, Taiwan 886 2 2528 7227,
Thailand 662 992 7519, United Kingdom 44 0 1635 523545

For further support information, refer to the *Technical Support and Professional Services* appendix. To comment on the documentation, send email to techpubs@ni.com.

Important Information

Warranty

The media on which you receive National Instruments software are warranted not to fail to execute programming instructions, due to defects in materials and workmanship, for a period of 90 days from date of shipment, as evidenced by receipts or other documentation. National Instruments will, at its option, repair or replace software media that do not execute programming instructions if National Instruments receives notice of such defects during the warranty period. National Instruments does not warrant that the operation of the software shall be uninterrupted or error free.

A Return Material Authorization (RMA) number must be obtained from the factory and clearly marked on the outside of the package before any equipment will be accepted for warranty work. National Instruments will pay the shipping costs of returning to the owner parts which are covered by warranty.

National Instruments believes that the information in this document is accurate. The document has been carefully reviewed for technical accuracy. In the event that technical or typographical errors exist, National Instruments reserves the right to make changes to subsequent editions of this document without prior notice to holders of this edition. The reader should consult National Instruments if errors are suspected. In no event shall National Instruments be liable for any damages arising out of or related to this document or the information contained in it.

EXCEPT AS SPECIFIED HEREIN, NATIONAL INSTRUMENTS MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AND SPECIFICALLY DISCLAIMS ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. CUSTOMER'S RIGHT TO RECOVER DAMAGES CAUSED BY FAULT OR NEGLIGENCE ON THE PART OF NATIONAL INSTRUMENTS SHALL BE LIMITED TO THE AMOUNT THERETOFORE PAID BY THE CUSTOMER. NATIONAL INSTRUMENTS WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY THEREOF. This limitation of the liability of National Instruments will apply regardless of the form of action, whether in contract or tort, including negligence. Any action against National Instruments must be brought within one year after the cause of action accrues. National Instruments shall not be liable for any delay in performance due to causes beyond its reasonable control. The warranty provided herein does not cover damages, defects, malfunctions, or service failures caused by owner's failure to follow the National Instruments installation, operation, or maintenance instructions; owner's modification of the product; owner's abuse, misuse, or negligent acts; and power failure or surges, fire, flood, accident, actions of third parties, or other events outside reasonable control.

Copyright

Trademarks

AutoCode™, MATRIXx™, National Instruments™, NI™, ni.com™, SystemBuild™, and Xmath™ are trademarks of National Instruments Corporation.

Product and company names mentioned herein are trademarks or trade names of their respective companies.

Patents

For patents covering National Instruments products, refer to the appropriate location: **Help»Patents** in your software, the `patents.txt` file on your CD, or `ni.com/patents`.

WARNING REGARDING USE OF NATIONAL INSTRUMENTS PRODUCTS

(1) NATIONAL INSTRUMENTS PRODUCTS ARE NOT DESIGNED WITH COMPONENTS AND TESTING FOR A LEVEL OF RELIABILITY SUITABLE FOR USE IN OR IN CONNECTION WITH SURGICAL IMPLANTS OR AS CRITICAL COMPONENTS IN ANY LIFE SUPPORT SYSTEMS WHOSE FAILURE TO PERFORM CAN REASONABLY BE EXPECTED TO CAUSE SIGNIFICANT INJURY TO A HUMAN.

(2) IN ANY APPLICATION, INCLUDING THE ABOVE, RELIABILITY OF OPERATION OF THE SOFTWARE PRODUCTS CAN BE IMPAIRED BY ADVERSE FACTORS, INCLUDING BUT NOT LIMITED TO FLUCTUATIONS IN ELECTRICAL POWER SUPPLY, COMPUTER HARDWARE MALFUNCTIONS, COMPUTER OPERATING SYSTEM SOFTWARE FITNESS, FITNESS OF COMPILERS AND DEVELOPMENT SOFTWARE USED TO DEVELOP AN APPLICATION, INSTALLATION ERRORS, SOFTWARE AND HARDWARE COMPATIBILITY PROBLEMS, MALFUNCTIONS OR FAILURES OF ELECTRONIC MONITORING OR CONTROL DEVICES, TRANSIENT FAILURES OF ELECTRONIC SYSTEMS (HARDWARE AND/OR SOFTWARE), UNANTICIPATED USES OR MISUSES, OR ERRORS ON THE PART OF THE USER OR APPLICATIONS DESIGNER (ADVERSE FACTORS SUCH AS THESE ARE HEREAFTER COLLECTIVELY TERMED "SYSTEM FAILURES"). ANY APPLICATION WHERE A SYSTEM FAILURE WOULD CREATE A RISK OF HARM TO PROPERTY OR PERSONS (INCLUDING THE RISK OF BODILY INJURY AND DEATH) SHOULD NOT BE RELIANT SOLELY UPON ONE FORM OF ELECTRONIC SYSTEM DUE TO THE RISK OF SYSTEM FAILURE. TO AVOID DAMAGE, INJURY, OR DEATH, THE USER OR APPLICATION DESIGNER MUST TAKE REASONABLY PRUDENT STEPS TO PROTECT AGAINST SYSTEM FAILURES, INCLUDING BUT NOT LIMITED TO BACK-UP OR SHUT DOWN MECHANISMS. BECAUSE EACH END-USER SYSTEM IS CUSTOMIZED AND DIFFERS FROM NATIONAL INSTRUMENTS' TESTING PLATFORMS AND BECAUSE A USER OR APPLICATION DESIGNER MAY USE NATIONAL INSTRUMENTS PRODUCTS IN COMBINATION WITH OTHER PRODUCTS IN A MANNER NOT EVALUATED OR CONTEMPLATED BY NATIONAL INSTRUMENTS, THE USER OR APPLICATION DESIGNER IS ULTIMATELY RESPONSIBLE FOR VERIFYING AND VALIDATING THE SUITABILITY OF NATIONAL INSTRUMENTS PRODUCTS WHENEVER NATIONAL INSTRUMENTS PRODUCTS ARE INCORPORATED IN A SYSTEM OR APPLICATION, INCLUDING, WITHOUT LIMITATION, THE APPROPRIATE DESIGN, PROCESS AND SAFETY LEVEL OF SUCH SYSTEM OR APPLICATION.

Conventions

The following conventions are used in this manual:

[ ] Square brackets enclose optional items—for example, [`response`].

» The » symbol leads you through nested menu items and dialog box options to a final action. The sequence **File»Page Setup»Options** directs you to pull down the **File** menu, select the **Page Setup** item, and select **Options** from the last dialog box.

✎ This icon denotes a note, which alerts you to important information.

bold Bold text denotes items that you must select or click in the software, such as menu items and dialog box options. Bold text also denotes parameter names.

italic Italic text denotes variables, emphasis, a cross reference, or an introduction to a key concept. This font also denotes text that is a placeholder for a word or value that you must supply.

`monospace` Text in this font denotes text or characters that you should enter from the keyboard, sections of code, programming examples, and syntax examples. This font is also used for the proper names of disk drives, paths, directories, programs, subprograms, subroutines, device names, functions, operations, variables, filenames, and extensions.

`monospace bold` Bold text in this font denotes the messages and responses that the computer automatically prints to the screen. This font also emphasizes lines of code that are different from the other examples.

`monospace italic` Italic text in this font denotes text that is a placeholder for a word or value that you must supply.

Contents

Chapter 1
Introduction

Chapter 2
Running HyperBuild from SystemBuild

Chapter 3
Running HyperBuild from Xmath

Chapter 4
Using HyperBuild

Appendix A
Technical Support and Professional Services

Index

1

Introduction

This guide provides a summary of how to use HyperBuild to accelerate the simulation of any SystemBuild model. This includes the following major topics:

- Chapter 1, *Introduction*
- Chapter 2, *Running HyperBuild from SystemBuild*
- Chapter 3, *Running HyperBuild from Xmath*
- Chapter 4, *Using HyperBuild*

Overview

HyperBuild provides a way to decrease the computer simulation time of SystemBuild models, especially medium and large models. The more complex the SystemBuild model, the more significant the decrease in simulation speed when using HyperBuild.

HyperBuild achieves this improvement by converting a SystemBuild block diagram into highly optimized C code (called HyperCode). The HyperCode version of a model executes much faster than the original model itself, because the simulator is interpretive. The simulator normally evaluates a model block-by-block, whereas HyperBuild typically takes an existing SystemBuild model and replaces it with a new model built from HyperBuild blocks, which are derivatives of UserCode Blocks (UCBs). Since the new model has only a single HyperBuild block per subsystem, the simulator can evaluate it quickly. DataStores from the original model are transferred to the new model outside of the HyperCode.

You can use HyperBuild to generate code for most models, including multirate models and those containing procedures or blocks with state events.

HyperBuild provides support for most SystemBuild features, including the following:

- Software constructs
- Models with algebraic loops
- Zero-crossing blocks
- Implicit blocks (including implicit UCBs)
- Read From and Write To Variable blocks
- Resettable integrators
- Procedure SuperBlocks (includes Standard, Startup, Macro, Background, Interrupt, and Inline Procedure SuperBlocks)
- Multirate models
- Condition SuperBlocks

For specific information on any of these features, refer to the *SystemBuild User Guide* or the *MATRIXx Help*. For additional information about HyperBuild performance and limitations, refer to Chapter 4, *Using HyperBuild*.

Simulation Process

Figure 1-1 shows an overview of the HyperBuild simulation process. This figure traces the progress of the imaginary UFO model through the steps of loading, analyzing, simulating, compiling, and relinking. Each of the HyperBuild steps is described in Chapter 2, *Running HyperBuild from SystemBuild*. The SystemBuild process is described in the *SystemBuild User Guide*.

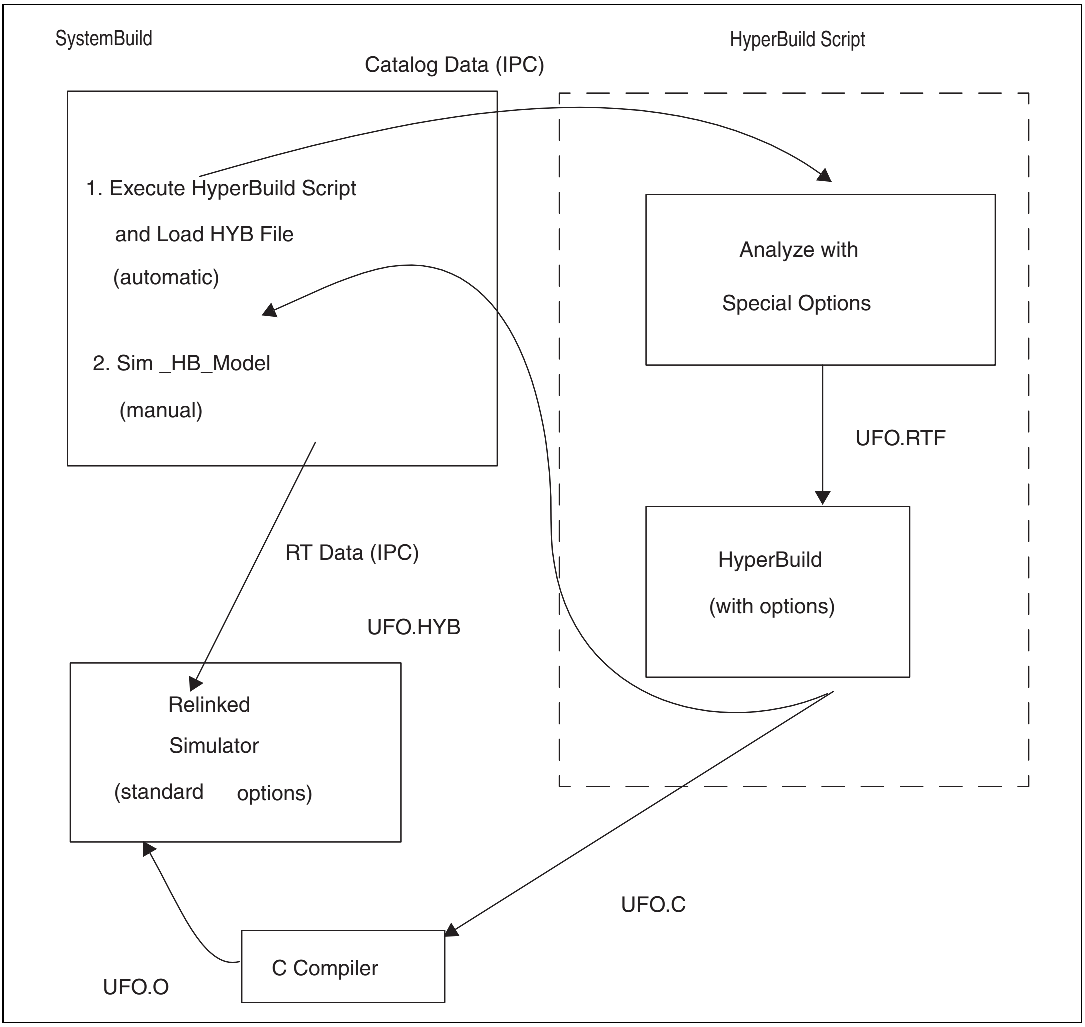


Figure 1-1. HyperBuild Simulation Process

2

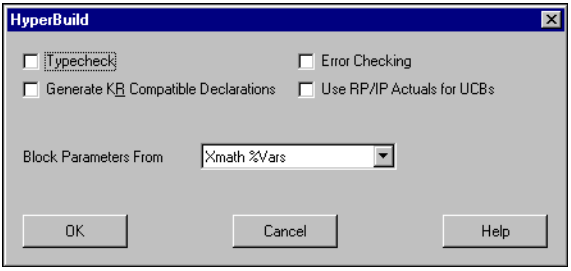
Running HyperBuild from SystemBuild

Running HyperBuild and Generating HyperCode

To run HyperBuild and generate HyperCode, complete the following steps:

1. Select a top-level SuperBlock of your model from the SystemBuild Catalog Browser and then select **Tools»HyperBuild**.

   The HyperBuild dialog box appears.



□ Typecheck

□ Generate KR Compatible Declarations

□ Error Checking

□ Use RP/IP Actuals for UCBs

Block Parameters From: Xmath %Vars

2. From this dialog box, select the options you want and click **OK**.

   HyperBuild generates HyperCode and creates a catalog in your working directory named *blockname*.hyb.

   The available options include:

   Typecheck — Enables data type checking in the generated code. Default is unchecked.

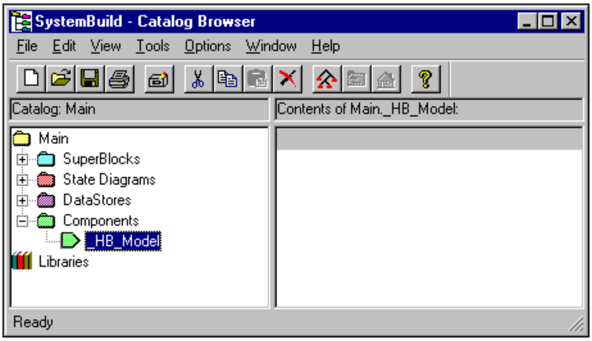
   Generate KR Compatible Declarations — Enables the generation of Kernigan and Ritchie style C code declarations. Default is unchecked.

   Error Checking — Enables error checking in the generated code. Default is unchecked.

   User RP/IP Actuals for UCBs — Provides for a selection between RP/IP (real parameter/integer parameter) hardcoded numbers and %var encoding for UCBs. Default is unchecked.

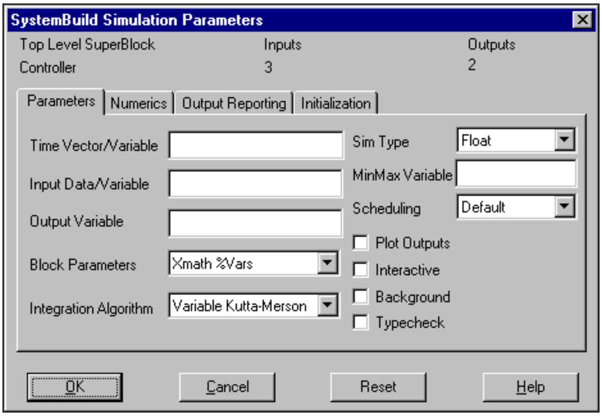
Block Defaults or Xmath %Vars	Provides a choice between putting default variable block parameters or %vars into the HyperBuild code. Default is Xmath %Vars.

SystemBuild automatically loads the `*modelname*.hyb` model after it is generated as the component _HB_Model.



Loading and Using HyperCode

After you have run HyperBuild on your model, you select the **_HB_Model** component. Then select **Tools»Simulate** from the Catalog Browser. The SystemBuild Simulation Parameters dialog box appears.



Top Level SuperBlock	Inputs	Outputs
Controller	3	2

Time Vector/Variable

Input Data/Variable

Output Variable

Block Parameters: Xmath %Vars

Integration Algorithm: Variable Kutta-Merson

Sim Type: Float

MinMax Variable

Scheduling: Default

□ Plot Outputs

□ Interactive

□ Background

□ Typecheck

To simulate your HyperCode, enter the following required parameters:

Parameters tab: Time Vector/Variable	Requires a time column vector (for example, `[0:0.1:300]'`).
Parameters tab: Input Data/Variable	Required variable for models with 1 or more inputs (for example, `u`).

To make the best use of your HyperCode, select from the following options:

Parameters tab: Output Variable	Keep at a minimum for best performance.
Parameters tab: Sim type	Select **Float** for best performance. **Note**: If the original model uses fixed-point types, these will be simulated last in the HyperBuild object, not by the simulator. Therefore, select Float even if there are fixed-point types within the HyperCode.
Parameters tab: Scheduling	Select **actiming** if you plan to use AutoCode to generate C or Ada code.
Numerics tab: Time Delay Buffer	Set to a low number to minimize delays (20 is the default).
Initialization tab: Remake sim executable	Leave unchecked to minimize simulation time.

Note The `LINKHYPER` Xmath command is no longer supported. HyperCode is linked to the simulator immediately after generation.

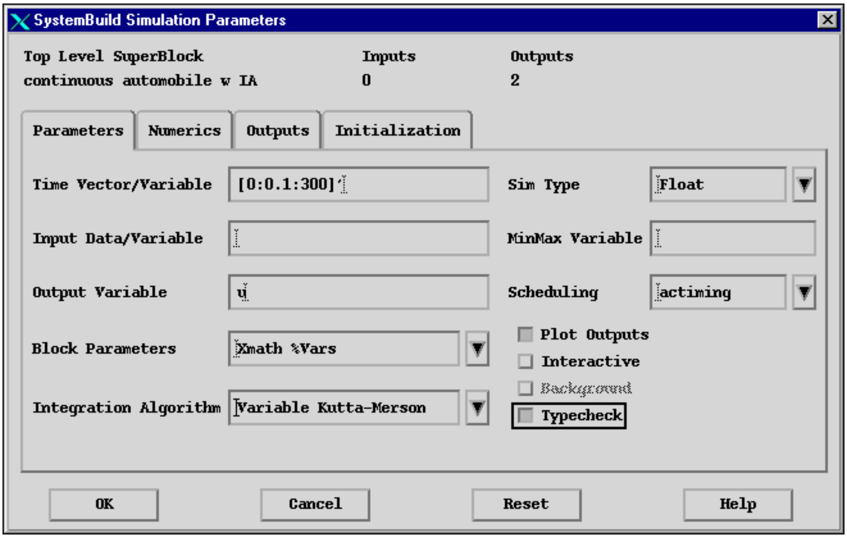
Simulation Example

As an example, load the super_cruise model from the *`MTXHOME`*`/sysbld/demo` directory, select a top level block such as the **continuous automobile with IA** block, and then simulate the model with and without HyperBuild.

Simulation Without HyperBuild

To simulate a model without HyperCode, complete the following steps:

1. Select **Tools»Simulate**, which opens the SystemBuild Simulation Parameters dialog box.
2. In the **Parameters** tab, set the **Time Vector/Variable** to `[0:0.1:300]'`, enter an **Output Variable** of `u`, and select any of the optional checkboxes such as **Typecheck** and **Plot Output**.



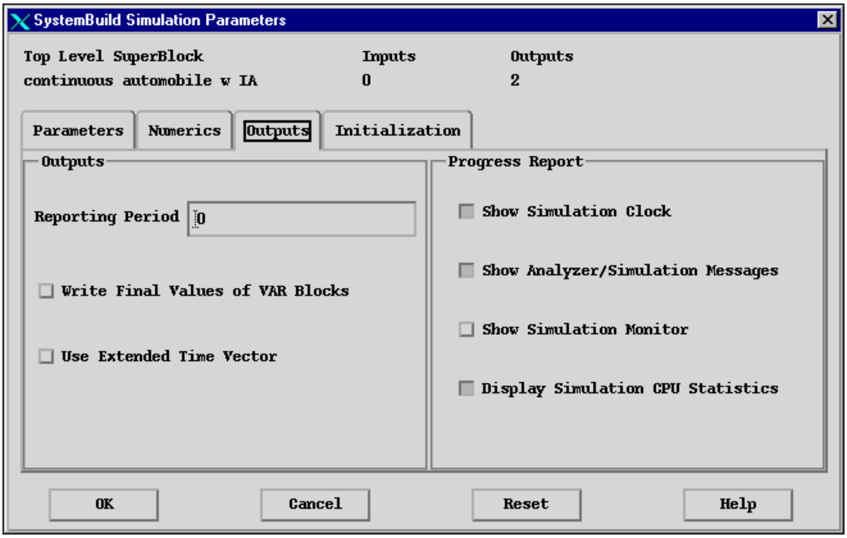
SystemBuild Simulation Parameters

Top Level SuperBlock: continuous automobile w IA
Inputs: 0
Outputs: 2

Time Vector/Variable: [0:0.1:300]'
Input Data/Variable:
Output Variable: u
Block Parameters: Xmath %Vars
Integration Algorithm: Variable Kutta-Merson
Sim Type: Float
MinMax Variable:
Scheduling: actiming

□ Plot Outputs
□ Interactive
□ Background
□ Typecheck

3. Select **Show Simulation Clock** (optional), **Show Analyzer/ Simulation Messages** (optional), and **Display Simulation CPU Statistics** (to display comparative results) from the **Outputs** tab.



SystemBuild Simulation Parameters

Top Level SuperBlock: continuous automobile w IA
Inputs: 0
Outputs: 2

Outputs

Reporting Period: 0

□ Write Final Values of VAR Blocks

□ Use Extended Time Vector

Progress Report

□ Show Simulation Clock

□ Show Analyzer/Simulation Messages

□ Show Simulation Monitor

□ Display Simulation CPU Statistics

After the simulation, you see output similar to the following:

```
Using variable step Variable Kutta-Merson integration.
Continuous states are solved explicitly.
Continuous outputs are solved explicitly.

Initialization Complete.
The solution is   1% complete.
The solution is   2% complete.
The solution is   5% complete.
The solution is  10% complete.
The solution is  20% complete.
The solution is  40% complete.
The solution is  60% complete.
The solution is  80% complete.
elapsed:  0 00:00:01.07   cpu:  0 00:00:01.06
The solution is 100% complete.
```

Simulation With HyperBuild

Using the same model and simulation parameters as in the *Simulation Without HyperBuild* section, you can compare your simulation time using HyperBuild.

In this example, you again use the super_cruise model and complete the following steps from the SystemBuild - Catalog Browser.

1. Load (or open) the super_cruise model from the `*MTXHOME*/sysbld/demo` directory.
2. Select the **continuous automobile with IA** block.
3. Select **Tools»HyperBuild** to create the _HB_Model component.
4. Simulate the model with HyperBuild by clicking **_HB_Model** and then selecting **Tools»Simulate**.

The output from the simulation with HyperBuild is identical to the “without HyperBuild” example except for the elapsed and cpu times.

```
elapsed:  0 00:00:00.54   cpu:  0 00:00:00.52
```

Because the super_cruise model is a small demonstration model, you do not see the full power of the tool. However, you see about a 50% improvement in the simulation speed even with this small model.

Note If you simulate a model and then load another model, you may need to delete some of the generated files (for example, `ucbhook.*`) before simulating the _HB_Model component again.

3

Running HyperBuild from Xmath

Running HyperBuild from the Xmath Commands Window

To run HyperBuild from the Xmath Commands window, you can use the following HyperBuild command:

```
hyperbuild "topSB", {keywords}
```

where *topSB* (a text string) is the name of the top SuperBlock of the model and keywords are described in Table 3-1.

Table 3-1. HyperBuild Keyword Options

Keyword	Type	Description
`errorcheck`	Integer	Enables error checking in the generated code. Default = 0.
`file`	File	Output file to receive generated code. **Note**: The keyword has no suffix.
`typecheck`	Boolean	Enables data type checking in the generated code. Default = 0.
`ucbparams`	Boolean	Lets the user configure UCBs to use either RP/IP (real parameter/integer parameter) hardcoded numbers or %var encoding.
`vars`	Boolean	If 1 (default), put `%Variables` into the HyperBuild code.

The `hyperbuild` command generates three files: the C file (*topSB*`.c`), the model file containing the _HB_Model component (*topSB*`.hyb`), and the real-time file (*topSB*`.rtf`).

For example, to generate HyperCode for the super_cruise model, enter the following command sequence from the Xmath Commands window:

```
load file = "cruise_d.cat";
main.t = [0:0.1:300]';
hyperbuild "continuous automobile w IA", {typecheck};
```

where `continuous automobile with IA` is a top-level SuperBlock. HyperBuild code is generated for the SuperBlocks contained in this *`topSB`*. The _HB_Model component that is generated contains UCBs that reference the HyperCode.

Linking a Simulation (Optional)

With HyperBuild, you can generate HyperCode and simulate the _HB_Model component directly, or you can use the `linksim` command to link your code with other UserCode Block files.

The `linksim` command has the following syntax:

```
linksim model, {csource, fsource}
```

where `model` is a string specifying the SystemBuild model to search for UserCode Block source files, `csource` is a string specifying the C source files to be compiled and linked into the simulator, and `fsource` is a string specifying the FORTRAN source files to be compiled and linked into the simulator.

Note File names must be separated by spaces, and files specified in the UCB dialog boxes need not be included.

The `linksim` command compiles and links UserCode Block source files and creates a UCB shared library (`simucb.*`, where the extension is platform dependent), in your local directory. For Windows, the extension is `.dll`. For Solaris, SGI, and IBM platforms, the extension is `.so.1.0`. For Compaq Tru64 platforms, the extension is `.so`. For HP platforms, the extension is `.sl`.

Source files can be specified in the parameter dialog of each UserCode Block reference in a model (no more than one file reference per UserCode Block reference). The specified SystemBuild model is searched for source files listed in any UCB, or referenced by the values of the `csource` and `fsource` values set using `setsbdefault`.

Simulating Linked HyperCode or linksim Output

To simulate linked HyperBuild code (the _HB_Model component) or the output from `linksim`, use the following `sim` command (for example):

```
outputPDM = sim ("_HB_Model", InputPDM, {keywords})
```

where `inputPDM` is a PDM with domain as a time vector, the range is the input, and keywords are as described in the *MATRIXx Help*.

Note Simulation of a HyperBuild model is exactly the same as simulation of any other SystemBuild model and is described fully in the *MATRIXx Help*.

Running HyperBuild from a Script

The `hyperbuild` command is especially useful if you want to run a script to perform your HyperCode creation and simulation. For example, the following script loads the super_cruise model, runs HyperBuild, and then simulates the `continuous automobile with IA`. This script also simulates the _HB_Model component:

```
load file = "cruise_d.cat";
pause 6;
main.t = [0:0.1:300]’;
hyperbuild "continuous automobile w IA", {typecheck};
[,main.ycsim] = sim("continuous automobile w IA",main.t,
{simtimer, typecheck});
[,main.ychyp] = sim("_HB_Model",main.t, {simtimer,
typecheck});
```

For additional information about writing Xmath scripts, refer to the *Xmath User Guide*.

4

Using HyperBuild

Usage Notes

When you are using HyperBuild, give special consideration to the following:

- If your regular SystemBuild simulation takes more than a few minutes, or if you want to run it repeatedly, NI recommends the use of HyperCode.
- Only one HyperCode model at a time can be linked with the simulation engine.
- Because the HyperCode SuperBlock _HB_Model is automatically generated, the best way to change initial states, connections, or parameters is to edit the original SuperBlock in the SystemBuild model and then regenerate the HyperCode.
- The generated code has comments that indicate the original SuperBlock name, block name, and block type for each section of code.
- UserCode Blocks (UCBs) are allowed in HyperBuild models, just as in conventional SystemBuild models. If you use the `UPDUSR` interface to UCBs, you must specify all required source code files as `csource` in Xmath, by using the `SETSBDEFAULT` command, and specify a positive integer for the function name in the block dialog box. UCBs created prior to version 4.0 are in this format.

Performance Results

Performance results will vary, but simulation speeds using HyperBuild have been tested at seven to 10 times the speed of simulations without HyperBuild. You can improve performance results by simplifying your model in the following ways:

- Avoid implicit outputs and algebraic loops if possible.
- Minimize the number of inputs and outputs.

- Use procedure states instead of task states where possible in discrete subsystems.
- Use floating-point signals instead of other types of signals for subsystem inputs and outputs.

For additional information on improving performance, refer to the *Code Optimizations* chapter of the *AutoCode Reference*.

Limitations

HyperBuild has the following limitations:

- Algebraic loops within procedures are supported but discouraged for efficiency reasons.
- MathScript blocks are not supported.
- %var data from the Xmath stack are re-initialized. HyperCode takes the %var values that are set within Xmath at the time of code generation. To change these values, you must change them in Xmath and regenerate the code.

For specific information on SystemBuild topics, refer to the *SystemBuild User Guide* or the *MATRIXx Help*.

A

Technical Support and Professional Services

Visit the following sections of the National Instruments Web site at ni.com for technical support and professional services:

- **Support**—Online technical support resources at ni.com/support include the following:
  - **Self-Help Resources**—For immediate answers and solutions, visit the award-winning National Instruments Web site for software drivers and updates, a searchable KnowledgeBase, product manuals, step-by-step troubleshooting wizards, thousands of example programs, tutorials, application notes, instrument drivers, and so on.
  - **Free Technical Support**—All registered users receive free Basic Service, which includes access to hundreds of Application Engineers worldwide in the NI Developer Exchange at ni.com/exchange. National Instruments Application Engineers make sure every question receives an answer.
- **Training and Certification**—Visit ni.com/training for self-paced training, eLearning virtual classrooms, interactive CDs, and Certification program information. You also can register for instructor-led, hands-on courses at locations around the world.
- **System Integration**—If you have time constraints, limited in-house technical resources, or other project challenges, NI Alliance Program members can help. To learn more, call your local NI office or visit ni.com/alliance.

If you searched ni.com and could not find the answers you need, contact your local office or NI corporate headquarters. Phone numbers for our worldwide offices are listed at the front of this manual. You also can visit the Worldwide Offices section of ni.com/niglobal to access the branch office Web sites, which provide up-to-date contact information, support phone numbers, email addresses, and current events.

Index