MATRIXx™

Getting Started (UNIX)

The MATRIXx products and related items have been purchased from Wind River Systems, Inc. (formerly Integrated Systems, Inc.). These reformatted user materials may contain references to those entities. Any trademark or copyright notices to those entities are no longer valid and any references to those entities as the licensor to the MATRIXx products and related items should now be considered as referring to National Instruments Corporation.

National Instruments did not acquire RealSim hardware (AC-1000, AC-104, PCI Pro) and does not plan to further develop or support RealSim software.

NI is directing users who wish to continue to use RealSim software and hardware to third parties. The list of NI Alliance Members (third parties) that can provide RealSim support and the parts list for RealSim hardware are available in our online KnowledgeBase. You can access the KnowledgeBase at `www.ni.com/support`.

NI plans to make it easy for customers to target NI software and hardware, including LabVIEW real-time and PXI, with MATRIXx in the future. For information regarding NI real-time products, please visit `www.ni.com/realtime` or contact us at `matrixx@ni.com`.

NATIONAL INSTRUMENTS™

September 2003 Edition
Part Number 323701A-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 (Montreal) 514 288 5722, Canada (Ottawa) 613 233 5949, Canada (Québec) 514 694 8521, Canada (Toronto) 905 785 0085, Canada (Vancouver) 514 685 7530, China 86 21 6555 7838, Czech Republic 420 2 2423 5774, 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 0 22 3390 150, 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™, DocumentIt™, 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.

Contents

Chapter 4
SystemBuild

Chapter 5
AutoCode

Chapter 6
DocumentIt

Appendix A
Technical Support and Professional Services

About This Manual

This manual acquaints you with the MATRIXx Product Family software. The manual provides an overview to each piece of software and provides tutorials to assist you in learning the product.

This manual is for anyone who wants to learn how to use the MATRIXx Product Family software.

Organization

This manual is organized as follows:

- Chapter 1, *Introduction to the MATRIXx Product Family*, introduces each piece of software in the MATRIXx Product Family.
- Chapter 2, *Available Publications*, lists the MATRIXx Product Family publications available.
- Chapter 3, *Xmath*, provides an overview of Xmath and the Xmath modules, and gives a tutorial.
- Chapter 4, *SystemBuild*, provides an overview to SystemBuild and the SystemBuild modules, and gives a tutorial. It also provides an overview to Interactive Animation and gives a tutorial.
- Chapter 5, *AutoCode*, provides an overview to the AutoCode code generator.
- Chapter 6, *DocumentIt*, provides an overview to the DocumentIt document generator.

Conventions

This section describes the conventions used in this manual.

Font Conventions

Fonts other than the standard text default font are used as follows:

`Courier` — `Courier` is used for command and function names, filenames, directory paths, environment variables, messages and other system output, code and program examples, system calls, and syntax examples.

`bold Courier` — User input (anything you are expected to type in) is set in **`bold Courier`**.

bold Helvetica narrow — Buttons, fields, and icons in a graphical user interface are set in **bold Helvetica narrow** type. Keyboard keys are also set in this type.

italic — *Italic* is used in conjunction with the default font for emphasis, first instances of terms defined in the glossary, and publication titles.

Mouse Conventions

This document assumes you have a standard, right-handed three-button mouse. From left to right, the buttons are referred to as MB1, MB2, and MB3. All instructions assume MB1 unless otherwise noted.

click — Press and quickly release a mouse button. MB1 is assumed if click is used without a button designation. For example, "click on the root window".

double-click — Click MB1 twice in quick succession.

drag — Place the cursor over an object, then hold down MB1 while moving the mouse. Release the button when the desired result is obtained.

Note, Caution, and Warning Conventions

Within the text of this manual, you may find notes, cautions, and warnings. These statements are used for the purposes described below.

Note Notes provide special considerations or details which are important to the procedures or explanations presented.

Caution Cautions indicate actions that may result in possible loss of work performed and associated data. An example might be a system crash that results in the loss of data for that given session.

Warning Warnings indicate actions or circumstances that may result in file corruption, irrecoverable data loss, data security risk, or damage to hardware.

Related Publications

NI provides a library of publications to support its products. Of special interest to the users of this publication are the installation guides:

- *MATRIXx Product Family System Administrator's Guide (UNIX)*
- *MATRIXx Product Family CD-ROM* and booklet for each platform

For additional documentation, see the MATRIXx online Help.

1

Introduction to the MATRIXx Product Family

The MATRIXx Product Family includes the following products:

Xmath—The system analysis environment of the MATRIXx product family. See Chapter 3, *Xmath*.

SystemBuild—A graphical programming environment that uses a block diagramming paradigm with hierarchical structuring for modeling and simulation of linear and nonlinear dynamic systems. See Chapter 4, *SystemBuild*.

AutoCode—Template technology used to process SystemBuild model files to produce C or Ada code. Advanced template programming language (TPL) template technology provides a powerfulprogramming capability to tailor the generated code to specialized needs. See Chapter 5, *AutoCode*.

DocumentIt—TPL template technology (similar to AutoCode) used to capture information from SystemBuild model files and then format it to create documentation. See Chapter 6, *DocumentIt*.

Figure 1-1 shows an overview of the MATRIXx Product Family.

The MATRIXx Product Family core software must be installed according to the *System Administrator's Guide (UNIX)*. All users must have Xmath, as reflected in the product dependencies chart in Table 1-1. AutoCode and DocumentIt users must also have SystemBuild.

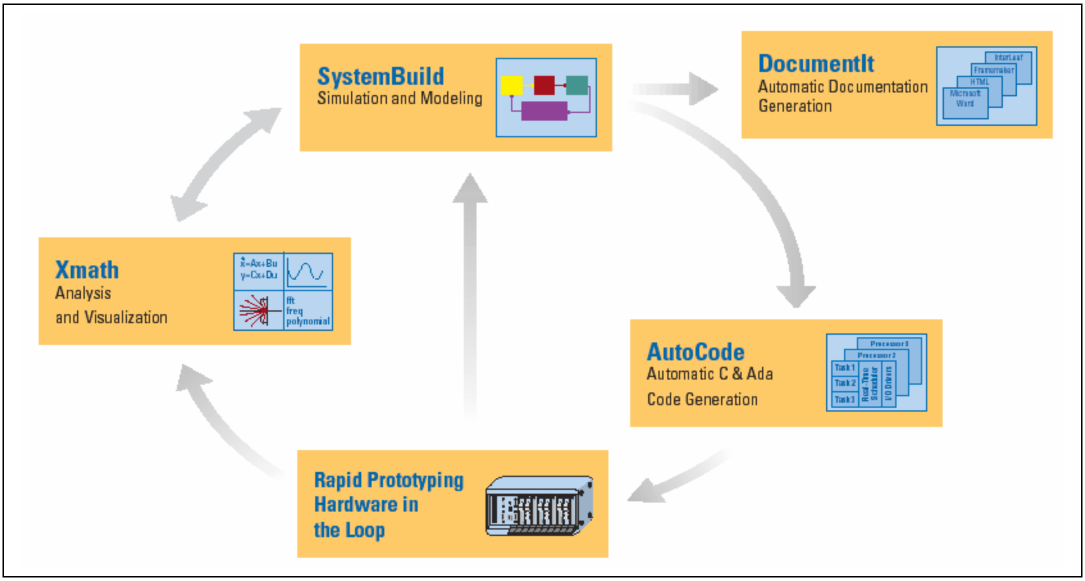

Figure 1-1. MATRIXx Product Family Overview

Table 1-1. Product Dependencies

		Products Needed			
		Xmath	SystemBuild	AutoCode	DocumentIt
Product to Add	SystemBuild	●	●		
	AutoCode	●	●	●	
	DocumentIt	●	●		●

Xmath

Xmath software provides a system analysis and visualization software environment. Xmath has over 700 predefined functions and commands, interactive color graphics, and a programmable graphical user interface (PGUI). The MathScript scripting language simplifies command and function programming. Object-oriented design provides convenient data management and speeds program execution. The structure and capabilities of Xmath are discussed in *Xmath Basics*, while the Xmath online Help provides easy access to Xmath commands and functions.

- Xmath commands support basic operations such as creating, plotting, saving, and loading data, and accessing online Help. The *Introduction to Xmath* section of Chapter 3, *Xmath*, describes the capabilities of Xmath and the Xmath modules.
- Xmath commands provide access to SystemBuild and its related products. Xmath handles data for SystemBuild and all other products in the MATRIXx Product Family.

SystemBuild

SystemBuild visual modeling and simulation software lets you model many kinds of systems, from control loops to complex vehicle applications. You can use SystemBuild to prepare models that can be simulated with the SystemBuild simulator. Built-in simulation tools let you interactively verify, test, and modify system models.

To to create a model, you can use all of the SystemBuild standard and optional features. The optional Interactive Animation (IA) module or the Altia Design module adds the ability to control your model interactively during simulation. With IA, the icons are put in one or more picture files (`.pic`) while Altia images are stored in design files (`.dsn`).

For additional information about SystemBuild, see Chapter 4, *SystemBuild*.

AutoCode

AutoCode is an automatic code generator for SystemBuild models. The AutoCode software processes SystemBuild model files you create and outputs compilable ANSI C or Ada code.

The output code can be compiled to produce a stand-alone real-time executable program suitable for running in a test-bed environment or for use in an embedded real-time system. Using the Template Programming Language (TPL), you can tailor nearly any part of the generated code for special needs. For additional information about AutoCode, see Chapter 5, *AutoCode*.

DocumentIt

DocumentIt is an automated documentation generator for SystemBuild models. This module integrates documentation with SystemBuild design activity for easier and more accurate manuals and reports. Templates are included for FrameMaker, Microsoft Word (PC only), and WordPerfect markup formats. Using TPL, you can capture and tailor any part of the generated document for special documentation standards or other needs.

For additional information about DocumentIt, see Chapter 6, *DocumentIt*.

2 Available Publications

Your MATRIXx Product Family software is shipped with online Help and online manuals. This chapter provides an overview of the manuals.

For additional documentation, see the MATRIXx online Help.

Installation Manuals

The following manual provides instructions for installing the MATRIXx Product Family software:

System Administrator's Guide (UNIX)—Describes proper setup of a UNIX workstation for the installation of the MATRIXx core software products.

Xmath Manuals

The Xmath manuals consist of *Xmath Basics* and a number of other manuals for Xmath modules.

Xmath Basics—Describes Xmath structure and concepts. It provides a tutorial, covers basic features for general Xmath use, and describes advanced Xmath features such as creating a GUI, creating your own MathScript commands, functions, or objects, and linking external programs.

Xmath Control Design Module—Explains the use of the Control Design Module including Linear system representation, building system connections, system analysis, classical feedback analysis, and state-space design. It describes each function in the Control Design Module.

Xmath Interactive Control Design Module—Describes how to use the Interactive Control Design Module (ICDM), which is a tool for interactive design of continuous-time, single-input, linear time-invariant controllers. ICDM uses the Xmath programmable graphical user interface (PGUI or GUI).

Xmath Interactive System Identification Module, Part 1—Describes the Interactive System Identification Module (ISID), which includes system identification, model reduction, and signal analysis tools for identification of linear, discrete time, and multivariable systems.

Xmath Interactive System Identification Module, Part 2—Focuses on a special interactive graphical interface for ISID commands that further simplifies system identification. Various graphical comparison tools allow you to try different identification and validation methods. This interface also supplies plots useful for system identification with the touch of a button.

Xmath Model Reduction Module—Describes the model reduction module (MRM), a collection of tools for reducing the order of systems.

Xmath Optimization Module—Describes nonlinear, quadratic, and linear optimization functions.

Xmath Robust Control Module—Describes the robust control module (RCM), a collection of analysis and synthesis tools that assist in the design of robust control systems.

Xmath Xμ Manual—Describes the Xmath functions used for modeling, analysis, and synthesis of linear robust control systems.

SystemBuild Manuals

The SystemBuild manuals consist of the *SystemBuild User's Guide* and a number of other manuals for SystemBuild modules.

SystemBuild User's Guide—Describes how to use SystemBuild, the graphical modeling and simulation environment, to construct a model for a dynamic system. SystemBuild lets you create custom building blocks, hierarchically organize model subsystems into SuperBlocks, and run system simulations based on the models.

Aerospace Libraries—Describes a library of SystemBuild models that were written for the aerospace industry.

HyperBuild User's Guide—Describes how to decrease the computer simulation time of medium and large SystemBuild models. The bigger and more complex the SystemBuild model, the more significant the increase in simulation speed. HyperBuild achieves this improvement by converting a SystemBuild block diagram into highly optimized C code (called HyperCode) that executes much faster in the simulation engine, which normally interprets the model data. HyperBuild can be used to generate code for continuous SuperBlocks only.

SystemBuild Fuzzy Logic Block User's Guide—Describes how the SystemBuild Fuzzy Logic Block provides a method for employing a fuzzy logic control methodology within SystemBuild for simulation and/or code generation. The Fuzzy Logic Block allows users to implement fuzzy logic decision structures of arbitrary complexity within a standardized block-diagram control-logic structure.

SystemBuild Neural Network Module User's Guide—Describes how the Neural Network Module (NNM) provides users the capability to define, parameterize, and include neural networks as SuperBlocks in a SystemBuild block diagram. Adding neural network technology to the fully integrated block diagram language of SystemBuild includes the capability to simulate your neural network models and to generate embedded code for them via AutoCode.

SystemBuild State Transition Diagram Block User's Guide—Describes the State Transition Diagram (STD) block. This separately licensed block can be obtained from the SystemBuild Palette Browser SuperBlocks menu. The STD block is an interface between a finite state machine and a SuperBlock diagram. In SystemBuild, each state in a finite state machine is graphically rendered as a bubble rather than a block; the STD editor is used to create bubble diagrams.

AutoCode and DocumentIt Manuals

AutoCode User's Guide—Describes how to use AutoCode to generate code from a SystemBuild block diagram.

AutoCode Reference—Supplements the *AutoCode User's Guide* and provides additional reference information.

DocumentIt User's Guide—Describes how to use DocumentIt to generate design documentation from a SystemBuild block diagram.

3

Xmath

Xmath provides tools for performing numerical analysis. You can create, store, plot, and analyze data in Xmath. You can program your own functions, commands, and objects, and also call in externally compiled C or Fortran code. Xmath is also the entry point to SystemBuild and its related products. This chapter gives an overview of Xmath.

Introduction to Xmath

The following subsections introduce the Xmath tools and capabilities.

Data Handling

MathScript, the language of Xmath, allows you to define and manipulate data in the form of numbers, objects, graphs, and text. Xmath provides a graphical user interface to facilitate data management. You can save, load, import, and export data.

Numerical Analysis

Xmath provides an extensive library of commands and basic functions (more than 700), including mathematical functions, filter design functions, and more. Xmath also provides a plotting facility that allows you to modify plot appearance interactively. These are documented in the Xmath online Help.

Each Xmath module includes online Help describing related functions and commands, as well as online and printed versions of a usage manual. Discussions of theory and examples are provided in usage manuals. See the *Xmath Manuals* section of Chapter 2, *Available Publications*, for a list of available manuals.

Architecture

Xmath's programming language, MathScript, allows users to alter or extend Xmath's functionality. An interactive debugger and a full complement of checking utilities simplify developing scripts to define functions, commands, and objects.

Xmath has an object-oriented structure that makes it unique among numerical analysis tools. This enables efficient numerical handling, including the overloading of operators, and more. Xmath's hierarchical objects greatly reduce the amount of user programming devoted to checking data characteristics.

Xmath includes a fully programmable graphical user interface (PGUI or GUI). This programmable GUI allows you to create and manipulate windows, dialogs, and other user interface tools. Any user can develop convenient user interfaces. You can find instructions for using and building GUI applications in the online Xmath Help under MathScript Programming, Programmable GUI.

MathScript supports calling external routines from within Xmath, or you can call Xmath from your own C programs. The Linked External (LNX) facility uses an interprocess communication (IPC) mechanism for communication between your external routine, which runs as a separate process, and Xmath. You can modify and recompile your routine without exiting Xmath, so that you can use and debug external programs in the same session. The User-Callable Interface (UCI) allows a C program to invoke Xmath as a computational engine. You can invoke Xmath from your C program and pass it values or expressions to evaluate and retrieve results, perform calculations, or plot values. For information on how to create LNXs and UCIs, see the *Xmath Basics* manual.

Getting Started in Xmath

This section assumes that Xmath has been properly installed and configured according to the *System Administrator's Guide* for your platform.

Directories Defined by Environment Variables

The MATRIXx product line is installed in a directory that becomes known as ISIHOME. At that time, the installation process modifies the startup scripts and provides the location of ISIHOME as an environment variable ($ISIHOME) that is known *only* within the MATRIXx environment. Three additional environment variables, also known only within the MATRIXx environment, define three subdirectories of ISIHOME: $XMATH, $CASE, and $SYSBLD. These variables are recognized only in the Xmath command line as environment variables. If you need to use them in the operating system itself, you must specify the full pathname; we indicate such cases with italics: *ISIHOME*, *XMATH*, *SYSBLD*, and *CASE*. If you do not know this pathname, you can determine it by typing the following command within the Xmath command area:

```
oscmd("echo $variable");
```

where $variable is $ISIHOME, $XMATH, $CASE, or $SYSBLD.

These variables are defined whether or not you purchase the entire product line.

Note The environment variables discussed within this section are reserved for the exclusive use of National Instruments, and they are subject to change. Therefore, we recommend that you not use them in your scripts.

Starting Xmath

To start and run Xmath as a background process, type the following command at the system prompt:

```
% xmath &
```

The ampersand (&) directs the process to run in the background. Invoking Xmath from the UNIX command line is convenient for calling Xmath with special startup options, such as the User-Callable Interface (UCI). (For details on UCIs, see *Xmath Basics*.)

Starting Xmath brings up the Xmath Commands window (see the *Getting Acquainted with the Xmath Commands Window* section).

Getting Acquainted with the Xmath Commands Window

The Xmath Commands window, shown in Figure 3-1, comes on view when you invoke Xmath.

You type inputs in the command area at the bottom of the split window. Function outputs and environment status are echoed to the log area above; error messages appear in the message area, the lowest window.

Menu Choices

The menu choices available in the Xmath Commands window are as follows:

- The Edit menu displays commands for editing the Xmath command area: Command Recall, Clear Log Area, Clear Command Area, Clear Message Area, Send Command, and Insert New Line.

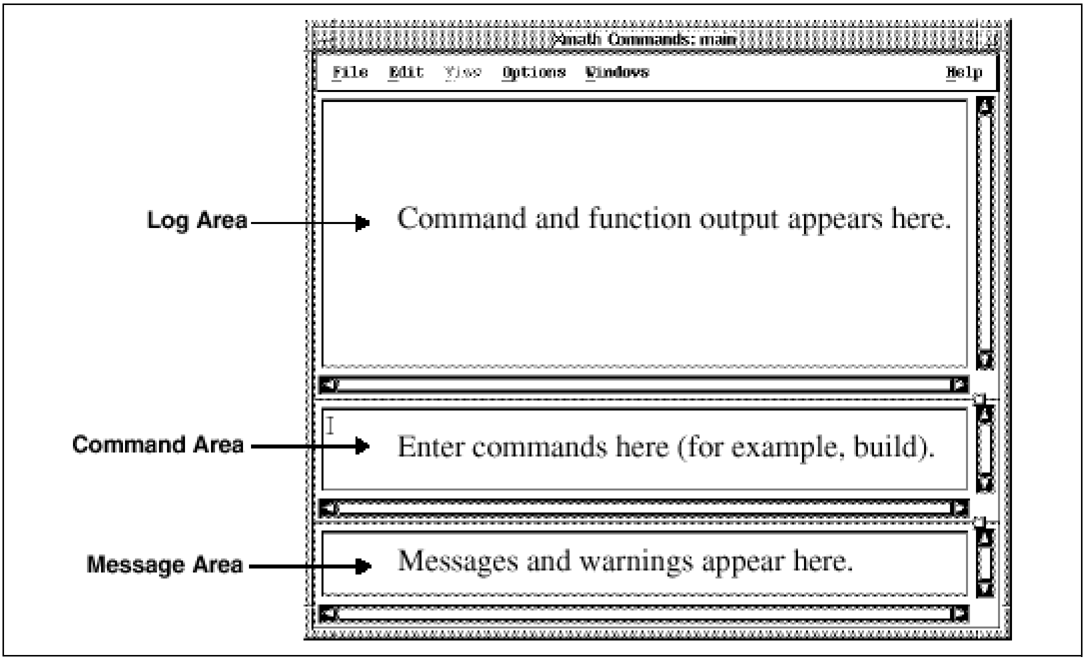

Figure 3-1. Xmath Commands Window

The Edit→Command Recall item lets you recall a command; you can also use the **Ctrl-↑** (previous command) or **Ctrl-↓** (next command) key sequences. Type **@** to display your last 10 commands.

- The View menu is reserved for expansion.
- The Options menu has a Format menu item for selecting the format of numeric values displayed in the Xmath log window.
- The Window menu lets you bring up the Graphics or Palette window or invoke SystemBuild.

Command Modes

Within the Commands window, there are two modes of operation: single-line command mode and multiline command mode.

The default mode is single-line command mode. You follow a command by pressing the **Return** key, which sends the command to Xmath.

The key sequence **Shift-Return** turns multiline mode on and also toggles the mode off. In multiline mode, the **Return** key adds a new line rather than sending the command to Xmath. For example:

`for i=1:10`	Press **Shift-Return** to turn multiline mode on and create a new line.
`i?`	Press **Shift-Return** to add a new line.
`endfor`	Press **Shift-Return** to turn multiline mode off and then Return to send the multiline for loop to Xmath.

Note The above example is not valid for cutting and pasting from online format into Xmath.

Running Demos

For a tutorial of Xmath's basic features, see Chapter 2, *Xmath JumpStart: A Tutorial*, in the *Xmath Basics* manual. For an online demo, click in the command area, and then type:

```
demo
```

This command gives you a choice of several scripts. As a script executes, explanatory text is displayed in the log and messages areas; a Pause dialog pauses the script to give you time to read the text or view a plot. You can move the Xmath Pause dialog so that it does not obscure the Commands window.

Accessing Online Help

Xmath has a comprehensive online Help system.

Note Online Help requires Netscape v3.0.1 or later. For instructions on installing Netscape, see the *Using MATRIXx Version 6.X Online Documentation*.

To access online Help, select Help→Topics from the Xmath Commands window, and the Help screen in Figure 3-2 appears.

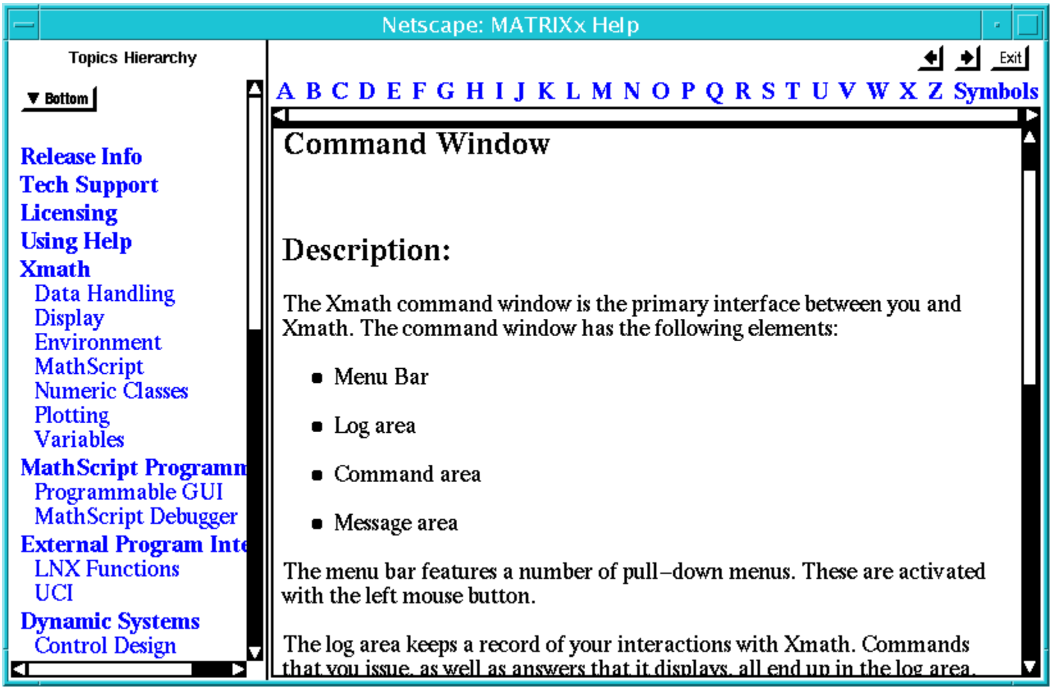

Figure 3-2. MATRIXx Help Screen Showing Topics Hierarchy

To access a topic, click the title in the Topics Hierarchy window, and the topic appears in the text window. Use the scroll bar and various buttons and links to navigate from one topic to another.

To access the index of topics starting with a particular letter, click the appropriate letter at the top of the right pane. The topic appears in the same window.

Stopping Xmath

- To exit from Xmath, type **`quit`** in the Xmath Commands window command area, or select File→Quit from the Commands window menu bar. Xmath prompts you to save the workspace.
- If you want to stop a function from continuing, type **Ctrl-C**, the standard interrupt. Note that **Ctrl-C** cannot interrupt a command in communication with the operating system (`load`, `save`); this includes creating and displaying windows.
- To abort Xmath if the program stops responding (for example, after a system error), type **Ctrl-** (Control backslash). Xmath attempts to save the workspace to a file called `abort.xmd` in the current directory.

Performing Sample Xmath Tasks

This section contains a brief introduction that mixes both basic and advanced features, with emphasis on features unique to Xmath and MathScript. If you have never used a numerical analysis package, you should become familiar with the demos described in the *Running Demos* section before continuing.

In the sections below, inputs are shown in **`bold Courier`**. You can find a description of the action, if applicable, and pertinent Help topics to the right of each input, preceded by the Xmath comment symbol (#). You can also see the *Xmath Basics* for a complete description of each feature used here.

We encourage you to try the examples provided below. You do not need to type comments. Note that most commands and functions can be shortened to as few as four characters.

See the *Starting Xmath* section for information on how to start Xmath.

Creating Data

Creating data is simple. Click in the command area to focus the pointer, and then enter the following MathScript statements to save the variables (comments are for your information):

```
                 # See "punctuation."
a=[1,2,2^2,3^3]  # Define a variable. See "vector" and
                 "operators."
b=1:.1:5         # See "regular vector."
c=sin(b)         # Call a function. See "functions."
```

Xmath provides the ability to save a graph to a variable as follows:

```
graph1=plot(c,{title="Creating the Graph Object
graph1."})
```

We will look at the variable later. To find out more, see the online Help topic "Graph Object" under Xmath (general)→Plotting.

Getting to Know Objects

You have just created two types of numeric objects. Let's identify each object.

```
whatis b         # See "commands" for command calling
                 syntax
whatis c         # See "objects"
```

If you look at the "vector" topic in the online Help, you see that vectors fall in the numeric class. Nonnumeric, or complex, objects are strings or mixtures of strings and numeric objects. Polynomials fall into this category:

```
d=makepoly(a,"d")       # See "makepoly"
e=polynomial(1:3,"d")   # See "polynomial"
```

Xmath's object structure allows you to build mathematical constructs in a natural way. Create a system as follows:

```
sys=system(d,e) # See "system" and "transfer function"
```

Some functions accept only certain objects and return others. For example, `char` accepts an integer and returns a string:

```
str=char(65)
```

The `freq` function accepts a system and returns a parameter-dependent matrix (PDM). A PDM is a special object that stores matrices in relation to an independent parameter or domain. (If you have SystemBuild, note that the simulation output is stored in this format.) The independent parameter is typically time or frequency.

Let's see how PDMs look.

```
f=freq(sys,b)?
g=freq(sys,{fmin=1,fmax=length(f),npts=length(f)})?
```

To create `f`, we specified a vector of frequencies; this became the domain. To create `g`, we let `freq` calculate the frequencies for the domain. Let's compare the two:

```
graph1=plot(f,{rows=2})?
```

```
graph2=plot(g,{row=2})?
```

For more on PDMs, see “pdm” and “PDM object” in online Help. For more on the `plot` function, see “plot” in online Help and *Xmath Basics*.

Saving, Loading, and Printing Data

To list the variables you have created so far, type

```
who
```

Note the sizes (see “who” in online Help for an explanation).

To save everything you have created, type

```
save
```

Xmath saves all data to a file with the default name `save.xmd` in the current working directory. (You may want to specify a filename because `save.xmd` will be overwritten by the next `save` command.) The first of the following two commands saves your variables to a file, and the second uses a wildcard to save a subset of variables to a different file.

```
save "try.xmd"
save "try_2.xmd" g* sys
```

See “save” and “wildcards” in the Xmath online Help.

Type the following command to display your working directory:

```
show directory
```

You can use the Xmath operating system command `oscmd` to list the files you saved:

```
oscmd("ls try*")
```

The operating system should find both `try.xmd` and `try_2.xmd`. If it does, you can delete what you have created in Xmath:

```
delete *
```

Retrieve the second file you saved and use the function `who` to list the variables that you have:

```
load "try_2"
who
```

Graphics

Let's use the variable `sys` again:

```
nyquist(sys)?
```

The function `nyquist` creates a plot; however, the returned value of `nyquist` is not the graphics object. We can save the contents of the Graphics window in one of two ways: select File→Bind to Variable, and specify the variable name `graph3`, or, from the Commands window command line, type:

```
graph3=plot()
```

You now have three graph objects. You can display them in the same manner as you would any variable:

```
graph1
graph2
graph3
```

Printing Graphs

You can print the graph currently displayed in the Graphics window in one of two ways:

- In the Graphics window, select File→Print and fill in the resulting dialog.
- Type

  ```
  hardcopy {color=0}
  ```

  in the Commands window command area. The setting `color=0` ensures that you receive a black and white rather than a color plot, which is the default.

Note To use the `hardcopy` command to print directly, the environment variable $XMATH_PRINT must be defined. If you need further help, see the *Xmath Basics* manual.

Alternatively, you can save your graphics to a `.ps` file and then submit the file to the printer with a standard UNIX command. For example:

```
hardcopy graph3, file="graph3.ps", {color=0}
```

From a terminal window, type:

```
lpr -P printer_name graph3.ps
```

Using MathScript

MathScript, the language of Xmath, defines statements, constructs, punctuation, functions (MSFs), commands (MSCs), and operators (MSOs). You can use MathScript to create your own functions and commands. Open a text editor, and create a file named `cdown.msf` (`.msf` corresponds to MathScript function), with the contents shown in Example 3-1. Note that the filename must be all lowercase.

Example 3-1: cdown.msf

```
#{
   Counts down from the integer supplied to 1.
}#
function [out]=cdown(c)

if is(c,{integer})then
   for i=[c:-1:1]
      display "*** " + string(i) + " ***"
   endfor
   display "Ignition!"
else
   error("positive integers only; try again","C")
endif

endfunction
```

Save your file in the current directory for Xmath, and return to Xmath. Try calling `cdown` with good and bad inputs:

`cdown(5)`
`cdown(2.2)`

Using the Xmath Debugger

You can control the Xmath debugger interactively from the Debugger window (see Figure 3-3) or from the command area in the Commands window. This section describes these two interfaces.

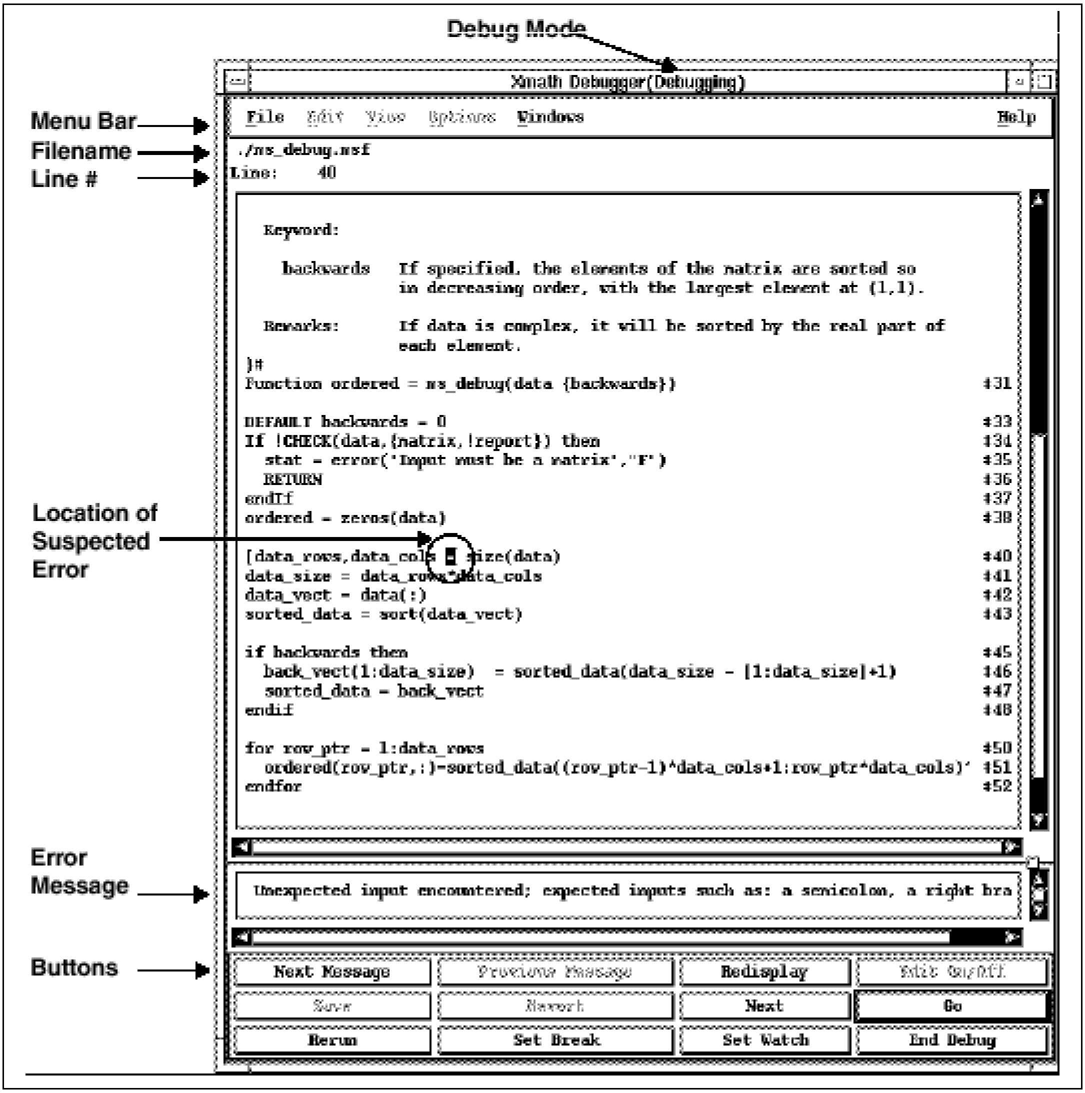

Figure 3-3. Xmath Debugger Window in Debug Mode

Starting the Debugger

Debug mode starts under three circumstances:

- A call is made to a program that is set up for debugging—that is, you execute the debug command:

  `debug` ***`program_name`***

  The debugger opens automatically on the first executable line in the script whenever you call the entity.
- A program contains a syntax error (for example, an error in punctuation, such as a missing brace: `plot(a,{xlab="A missing brace").`
- A program contains a run-time error. A run-time error occurs when an instruction is impossible to process. The following statement would cause a run-time error because the objects are incompatible: `x=5 + "hello".`

Normally, when an error is detected in a MathScript entity, Xmath automatically displays the error in the debugger window and sets the interpreter to debugging mode. When you execute the command, `set debugonerror off`, Xmath still displays the debugger window, but the interpreter does not go into debugging mode.

Using the Debugger

You can provide instructions to the Xmath debugger to debug MathScripts you write (MSFs, MSCs, and MSOs) interactively from the Debugger window or from the Xmath command line in the Commands window. You can use the buttons, such as **Set Break** and **Set Watch**, shown in Figure 3-3, or type the equivalent commands in the commands area to perform debugging tasks.

In the command window, let's start the debugger by typing:

`debug cdown`

The debugger sets a break at the first line of executable code—in this case, line 6. Now that a break point is set, let's try the debugger:

`cdown(2)`

Look at the difference in the command area prompt. You are now in debug mode; you can step through the code and examine local or global variables. Type **`next`** to continue until you reach the first line of the `for` loop. Click **Set Watch** on the variable `i`, or type:

`set watch i`

Click **Next** or type **`next`**. Note that you travel through the `for` loop two times, and you are notified when `i` is incremented.

You can examine variables local to the function. In the command area, type:

```
who
i?
```

When you fall out of the loop, type **`next`**, or **`go`** to run the function through to the end.

For additional information, see the MathScript Programming→MathScript Debugger online Help topic.

Exiting the Debugger

When you reach the end of the MathScript, you automatically exit debug mode. Type **`abort`** in the Commands window to exit debug mode before completing the script.

Correcting Errors during Debugging

When you are in the process of developing, you can open your file in an ASCII text editor and fix problems that the debugger locates. After you save your file, you can restart the script, and start debugging again. The debugger identifies the locations of errors by means of program line numbers; one limitation of some editors is that they do not support line numbers. Instead, you can use the `find` feature to locate the error by copying the information in error from the debugger to the **search** field. To escape the limitation entirely and run with program line numbers as the debugger provides them, you can use an ASCII editing program that supports line numbering.

Exploring Additional Topics

Beyond the brief introduction in this chapter, there are many more topics to explore. For example, your test MSF contains a simple `for` loop. You may want to explore this and other common scripting tasks, such as indexing in the online Help. You might make changes to a sample file and use the Xmath debugger to correct them. For additional information, see online Help or *Xmath Basics*.

4

SystemBuild

SystemBuild is a graphical programming environment that uses a block diagram paradigm with hierarchical structuring for modeling and simulation of linear and nonlinear dynamic systems. You can use SystemBuild to build models, and then test the models by using the SystemBuild Simulator and analysis tools. This chapter presents an overview of SystemBuild.

While you are working through this chapter, remember that there are additional sources of information available to you. The *SystemBuild User's Guide* covers the SystemBuild Editor and Simulator in detail, including several chapters on special SystemBuild topics. The extensive SystemBuild block library and other technical subjects are covered in the SystemBuild online Help.

Introduction to SystemBuild

This section introduces you to some of the SystemBuild tools and functions. For additional information about specific features, see the *SystemBuild User's Guide* or the SystemBuild online Help.

Key Terms

Table 4-1 provides key terms and their definitions. These terms are used throughout this guide and in other SystemBuild documentation.

Table 4-1. Definition of Common Terms

Term	Definition
SuperBlock	The basic hierarchical object in SystemBuild, which serves as a container blocks and defines the environment in which they operate.
Block	The basic functional element of SystemBuild. A set of blocks are used to model a control or real-time system.

Table 4-1. Definition of Common Terms (Continued)

Term	Definition
Internal Connection	Signals and data are passed between blocks using connections that appear as lines in the diagram within the Editor window. Internal connections pass data between blocks within the same SuperBlock.
External Connection	Connections between the SuperBlocks of a model and between the SuperBlocks and the outside world.

For a guide to other terms, concepts, and keyboard and mouse actions see the *SystemBuild User's Guide*.

SystemBuild Catalog Browser

The SystemBuild Catalog Browser is a tool that manages your SystemBuild model. You use the Catalog Browser to save and load models, view currently loaded SuperBlocks, create new SuperBlocks, and select SuperBlocks for editing, as well as other functions. The Catalog Browser, shown in Figure 4-1, contains a menu bar and toolbar (with buttons that are shortcuts to menu operations). The main portion of the Catalog Browser is divided into two *panes*. The left pane represents a hierarchical structure of different types of catalog objects (for example, SuperBlocks). The hierarchical structure provides compartmentalization of models and allows you to build and visualize extremely large models; it also provides for reuse of elements of a diagram. The right pane contains the contents of the catalog object selected from the left pane.

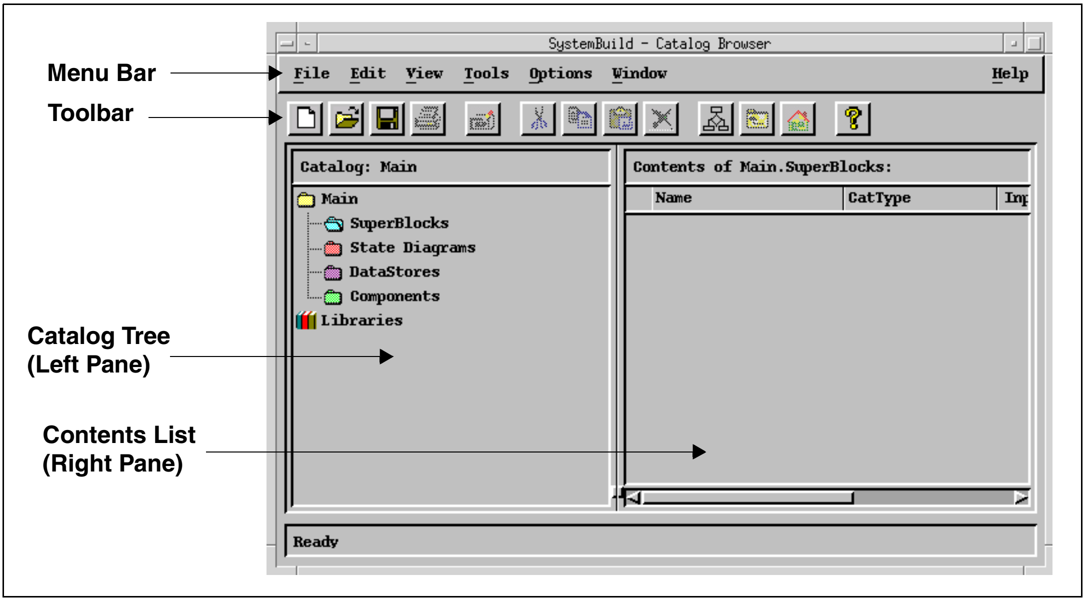

Figure 4-1. Catalog Browser

SystemBuild Editor

The SystemBuild Editor (also known as the Editor window or Editor) offers a user-friendly graphical model design/programming environment, which allows you to construct continuous-time, discrete-time, and hybrid systems of arbitrary complexity. You use the Editor window (see Figure 4-2) to edit the contents of your model.

SystemBuild supports the use of up to 20 Editor windows at once. Each Editor window can display one SuperBlock or state transition diagram at a time. You can switch frequently between the Catalog Browser to select a SuperBlock to edit and an Editor window to do the actual editing; one mechanism for accomplishing the window switching is using the Window menu available on both the Catalog Browser and the Editor window.

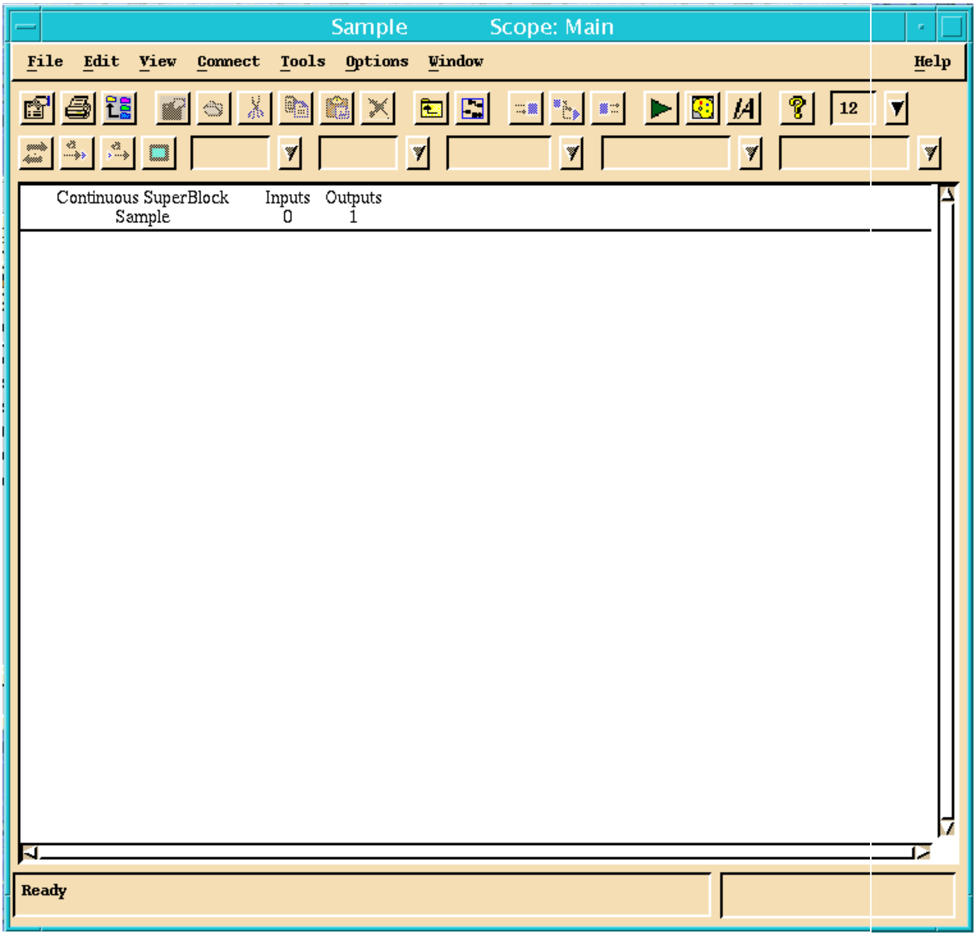

Figure 4-2. SystemBuild Editor

SystemBuild Palette Browser

The SystemBuild Palette Browser (see Figure 4-3) provides a choice of over 80 block types, including dynamic systems, algebraic and logical functions, signal generators, piecewise linear functions, trigonometric and exponential functions, and user-programmable blocks. The palette is fully customizable. The *SystemBuild User's Guide* provides several methods for adding custom blocks and custom palettes.

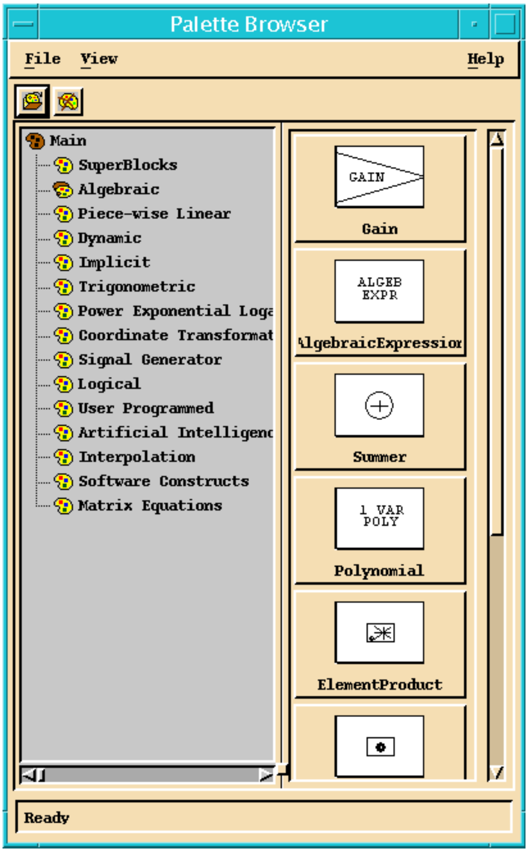

Figure 4-3. Palette Browser

SystemBuild Simulator

The SystemBuild Simulator provides a means of simulating your block diagram model under user-defined conditions. The Simulator provides flexibility in algorithms for integration, data input methods, model timing, and other areas. Both interactive and noninteractive simulation interfaces are provided for maximum utility.

Simulation in an interactive mode lets you interact with the model and monitor outputs of your blocks during simulation. You can debug your models with interactive capabilities such as block stepping or time stepping.

You can also change the values of block parameters during simulation by using the Run-Time Variable Editor (RVE).

Additional Functions

SystemBuild has a large number of additional functions that are not covered in this brief introduction. For example, the `lin` function performs linearization of single-rate and multi-rate models, the and `trim` function finds the trimmed input, state, and output values for equilibrium points of a system.

SystemBuild and its modules (see the *SystemBuild Optional Module Overview* section) are documented in several publications. The *SystemBuild Manuals* section of Chapter 2, *Introduction to the MATRIXx Product Family*, lists the SystemBuild publications that are available.

Specifying an ASCII Text Editor

You need a text editor for several tabs on both the SuperBlock Properties dialog and the block properties dialog. You choose the editor directly on the Comments tabs. To customize the editor selections available on the Comments tabs, enter or change one or more entries for `TextEditorItem = CommentEditor` in the `SYSBLD/etc/user.ini` file (see the *SystemBuild User's Guide* for details).

The text editor for other tabs is controlled by the environment variable `$EDIT_COMMENT`; if you want to change the default editor, change the environment variable:

```
setenv EDIT_COMMENT Editor_name
```

You must restart Xmath before the new editor becomes available.

If no text editor is specified by the `$EDIT_COMMENT` environment variable, the default is `vi`.

SystemBuild Optional Module Overview

This section describes optional modules offered with SystemBuild.

State Transition Diagram Block

State transition diagrams (STD) offer the capability to design and implement finite state machines. A mathematically rigorous implementation of finite state machines is supported by simulation, AutoCode code generation, and DocumentIt.

Fuzzy Logic Block

The Fuzzy Logic Block module lets you design and implement fuzzy logic real-time applications that are fully supported by SystemBuild, AutoCode, and DocumentIt.

Neural Network Module

The Neural Network Module lets you define, parameterize, and include neural networks as SuperBlocks in a SystemBuild block diagram. Adding neural network technology to the fully integrated block diagram language of SystemBuild includes the capability to simulate your neural network models and to generate embedded code for them via AutoCode. The module supports both training (offline) and learning (real-time) modes of operation.

Starting and Exiting SystemBuild

There are several methods of both starting and exiting SystemBuild; we are showing you one method for starting and one method for exiting below.

Starting SystemBuild

To start SystemBuild:

1. If you have not already started Xmath, type the following from the UNIX command prompt:

   ***ISIHOME*/bin/xmath**

   where *ISIHOME* is the MATRIXx installation folder (see the *Directories Defined by Environment Variables* section of Chapter 3, *Xmath*).

2. When Xmath comes up, type the following in the command area of the Xmath Commands window:

   build

   After a short time, SystemBuild is loaded; the Catalog Browser comes on view and the Editor window is iconized.

Exiting SystemBuild

To exit SystemBuild:

Select File→Exit from the Catalog Browser.

SystemBuild asks if you want to save your work before exiting; if you answer yes, the Save dialog appears.

Sample SystemBuild Tasks

This section describes a few routine operations.

Creating a New SuperBlock

You must use the Catalog Browser to create a new SuperBlock. If this window is not on top of all other windows, click on the Catalog Browser's window frame to raise the window to the top of all other windows or select Window→Catalog Browser from the Editor.

To create a new SuperBlock and define its properties:

1. Select File→New→SuperBlock or click the **New SuperBlock** toolbar button in the Catalog Browser to create a new SuperBlock (see Figure 4-4).

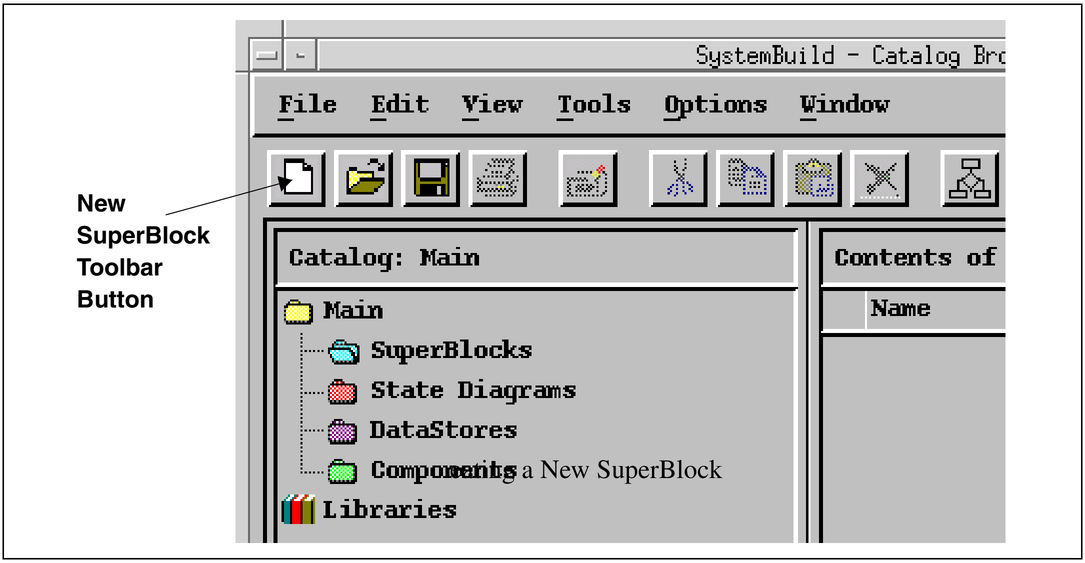

Figure 4-4. Creating a New SuperBlock

The SuperBlock Properties dialog appears. Use this dialog to define the attributes of the SuperBlock, such as its name, type (for example, continuous or discrete), and number of inputs and outputs.

2. Within the SuperBlock Properties dialog, shown in Figure 4-5, perform the following steps:
   a. Click within **Name** edit field, and type the name of the new SuperBlock within the **Name** edit field:

      `Sample SuperBlock`

   b. In the **Outputs** field, set the number of outputs to `1`.
   c. Click **OK** to complete the creation of the SuperBlock.

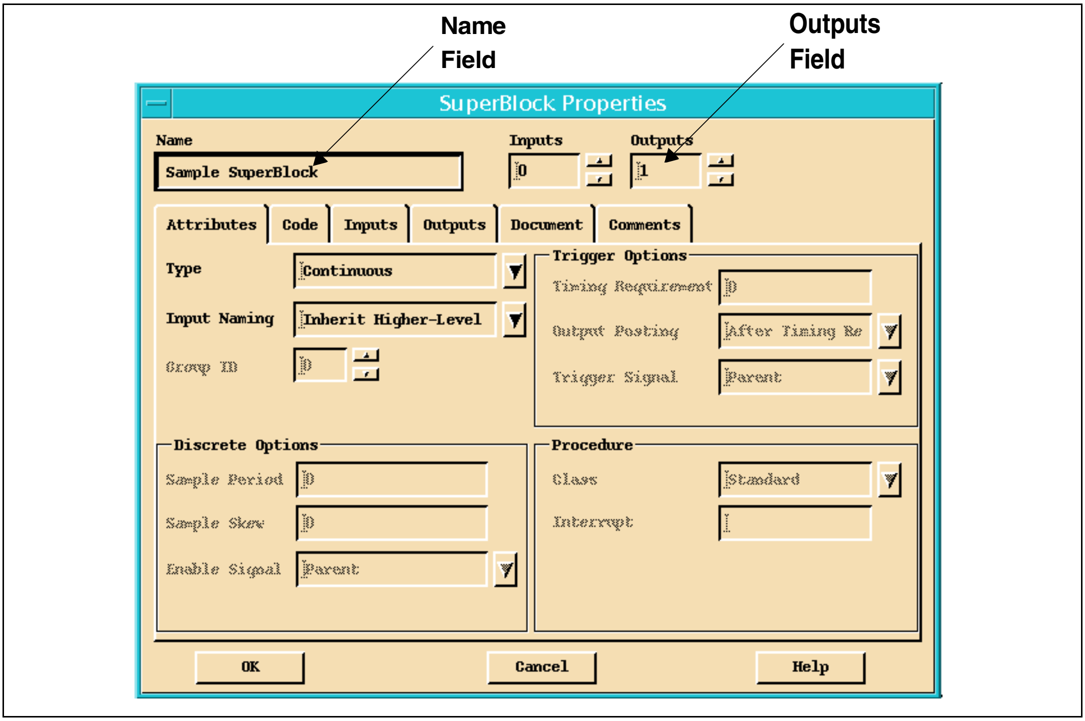

Figure 4-5. SuperBlock Properties Dialog for Creating a New SuperBlock

The SystemBuild Editor (or Editor window) (see Figure 4-2) now appears; it contains the Info Bar, which displays the SuperBlock name (Sample SuperBlock), type (continuous), and other relevant information (0 inputs and 1 output) about the current SuperBlock.

3. Click **OK** to complete the creation of the SuperBlock.

   The SystemBuild Editor (or Editor window) (see Figure 4-3) now appears; it contains the Info Bar, which displays the SuperBlock name (Sample SuperBlock), type (continuous), and other relevant information (0 inputs and 1 output) about the current SuperBlock.

Creating a New Block in a SuperBlock

You create a new block using both the Palette Browser and the Editor window. You drag a block from the Palette Browser into the Editor window to create a new block. Follow these steps (depicted in Figure 4-6):

1. Create a new SuperBlock if you haven't already using the steps outlined in the *Creating a New SuperBlock* section.
2. Select Window→Palette Browser to open the Palette Browser.

Note You may want to move your windows around so that both are visible.

3. Click the Algebraic palette in the Palette Browser.
4. Move the mouse cursor over the gain block. Press and hold down MB2. (❶)
5. While holding down MB2, drag the mouse cursor into the Editor window. (❷)
6. With the mouse cursor within the Editor window, release MB2 to complete the drag-drop operation. (❸)

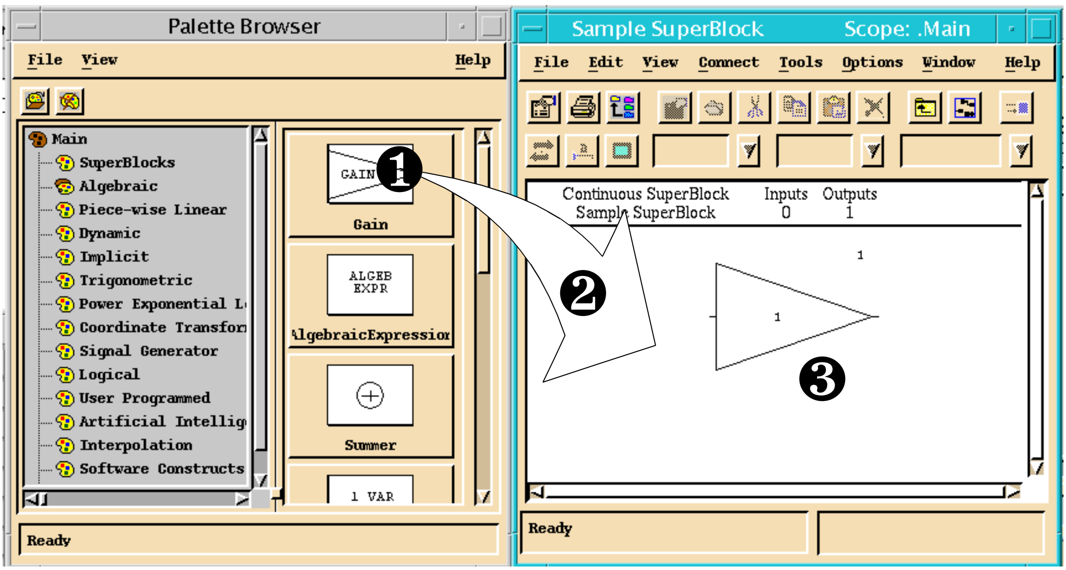

Figure 4-6. Creating a New Block Using the Palette Browser

Loading a Model File

Loading a model file opens a previously saved SystemBuild diagram. Once loaded, you can then edit or simulate that model.

Note You can only load a model file from Xmath or the Catalog Browser.

Loading a File from the Xmath Command Area

To load a file from the Xmath command area in the Commands window:

Enter the following command:

```
load "$SYSBLD/demo/predprey_demo/pred_prey.cat";
```

where `$SYSBLD` is your SystemBuild directory defined as an environment variable automatically when you start Xmath. After the load completes, the Catalog Browser lists the contents of the model (see Figure 4-7).

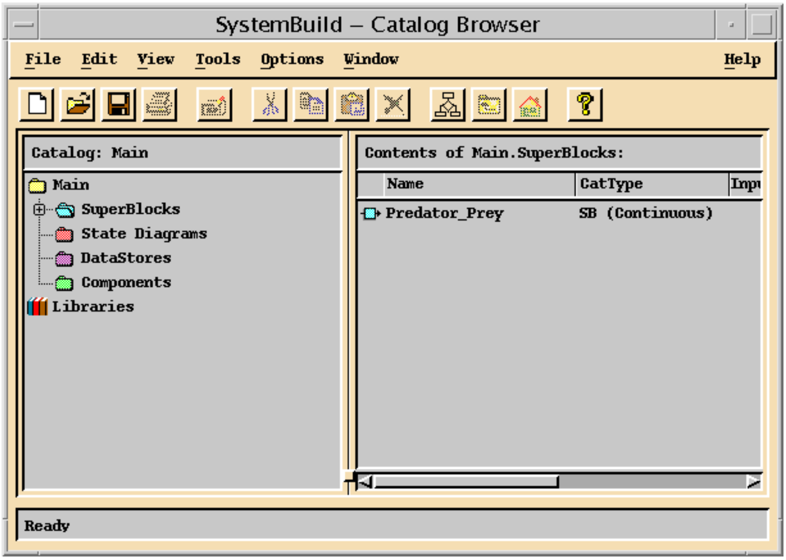

Figure 4-7. Predator-Prey Model Loaded into Catalog Browser

Note The Xmath command line is the only place that can recognize the environment variable itself. For the other methods that follow, you must know that full pathname of the SystemBuild directory.

Loading a File from the Xmath Commands Window

To load a file from the Xmath Commands window:

1. Select File→Load.
2. From the Load dialog, load the following file:

   `SYSBLD/demo/predprey_demo/pred_prey.cat`

   where *SYSBLD* is the SystemBuild distribution directory located under *ISIHOME* for the particular version of MATRIXx software (see the *Directories Defined by Environment Variables* section of Chapter 3, *Xmath*).

   You can navigate the directories and files using the directory list or type the filename directory into the **Selection** field (see Figure 4-8).

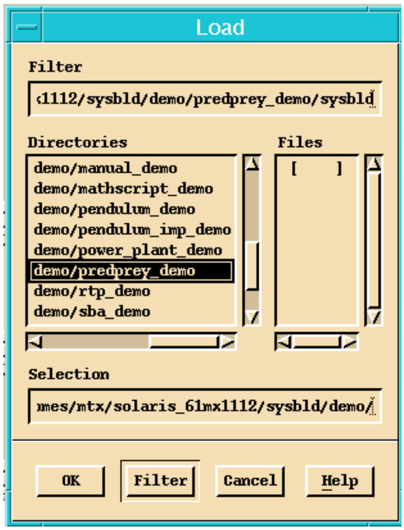

Figure 4-8. Xmath's Load Dialog

3. Click **OK** to load the file.

   After the load completes, the Catalog Browser lists the contents of the model (see Figure 4-7).

Loading a File from the Catalog Browser

To load a file from the Catalog Browser:

1. Select File→Load.

   A standard Motif-application Load dialog appears (see Figure 4-9) in which you can select a file in several ways.

2. Enter the following filename in the **Selection** field:

   ***SYSBLD*/demo/predprey_demo/pred_prey.cat**

   where *SYSBLD* is the SystemBuild distribution directory located under *ISIHOME* for the particular version of MATRIXx software (see the *Directories Defined by Environment Variables* section of Chapter 3, *Xmath*).

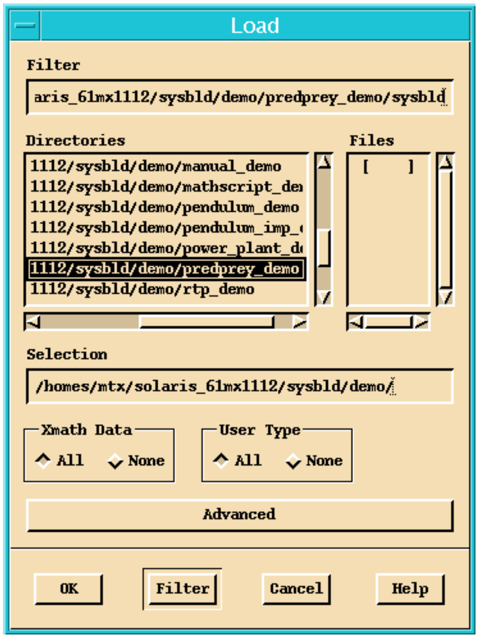

Figure 4-9. Catalog Browser's Load Dialog

3. Click **OK** to load the file.

Note The default settings in this Load dialog include no Xmath data.

After the load completes, the Catalog Browser lists the contents of the model (see Figure 4-7).

Opening a SuperBlock in the Editor

After loading a model, you need to open a SuperBlock in the editor so you can edit or view the SuperBlock.

1. Load the predator-prey model as described in the *Loading a Model File* section.
2. In the Catalog Browser, click in the left pane on top of the `Main` folder to see all of the SuperBlocks currently in the browser.

   All of the SuperBlocks should appear in the right pane.
3. In the right pane, double-click the **Predator_Prey** SuperBlock.

   This opens an Editor window and displays the contents of the SuperBlock (see Figure 4-10).

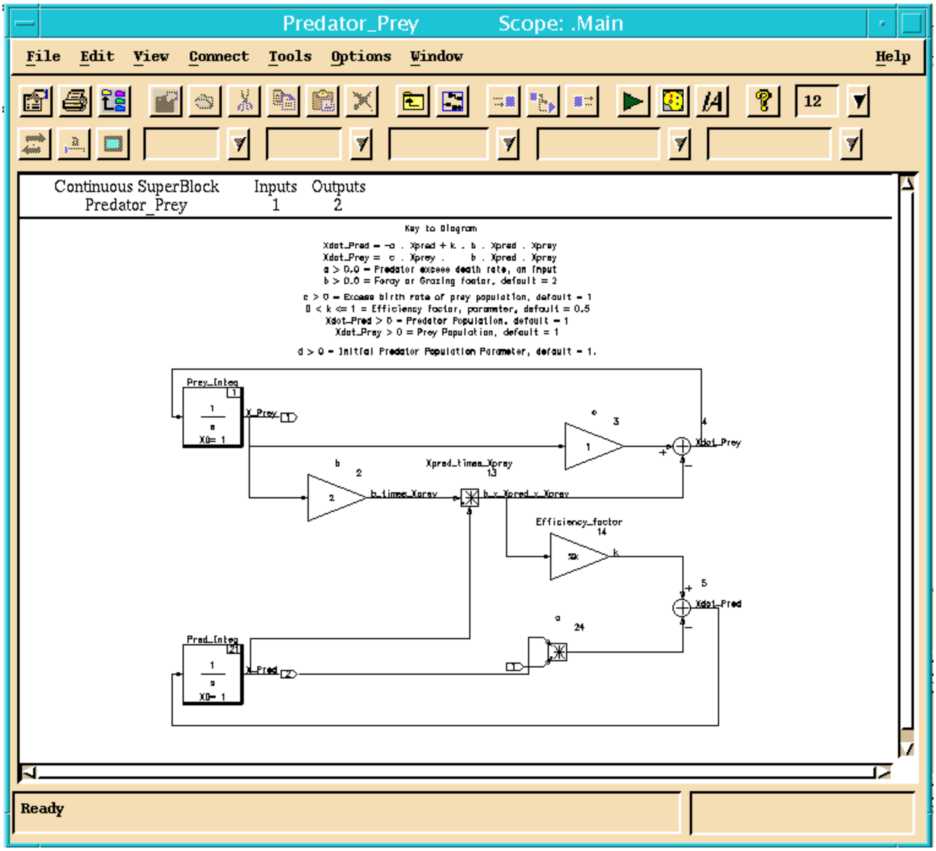

Figure 4-10. Predator_Prey SuperBlock Opened within Editor Window

Simulating the Model from the Xmath Commands Window

After the model is loaded, you can simulate it. There are many ways to simulate a model; this section illustrates a simulation directly from the Xmath command area and from the SystemBuild Editor. For this example, load the predator-prey model and follow one of the methods below.

To simulate the model from the Xmath commands area:

1. Activate the Xmath window by clicking on the Xmath window's frame.
2. Click within the Xmath command area.
3. Create a time vector and assign the input vector to a variable:
   ```
   t=[0:.01: 50]';
   u=ones(t);
   ```
4. Input the value of the efficiency factor `k`:
   ```
   k=.333;
   ```
5. In the Xmath command area, type:
   ```
   y=sim("Predator_Prey",t,u,{graph});
   ```
   Watch the log area of the Xmath window as the model is analyzed and simulated. The plot, which appears in a separate window, looks like Figure 4-11.

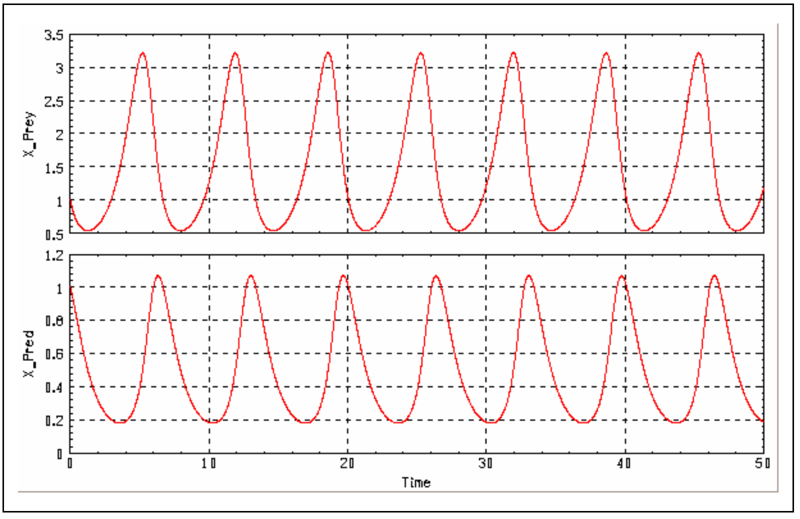

Figure 4-11. Plot of the Predator-Prey Example

To simulate the model from the SystemBuild Editor:

Note If you performed the simulation from the Xmath command line above and haven't deleted the variables, you can start at step 5.

1. Activate the Xmath window by clicking on the Xmath window's frame.
2. Click within the Xmath command area.
3. Create a time vector and assign the input vector to a variable:

   ```
   t=[0:.01: 50]';
   u=ones(t);
   ```

4. Input the value of the efficiency factor `k`:

   ```
   k=.333;
   ```

5. Bring up the SystemBuild Editor with the Predator_Prey SuperBlock on view (see the *Opening a SuperBlock in the Editor* section as necessary).
6. In the SystemBuild Editor, select Tools→Simulate from the pull-down menu.

   The Simulation dialog appears (see Figure 4-12).

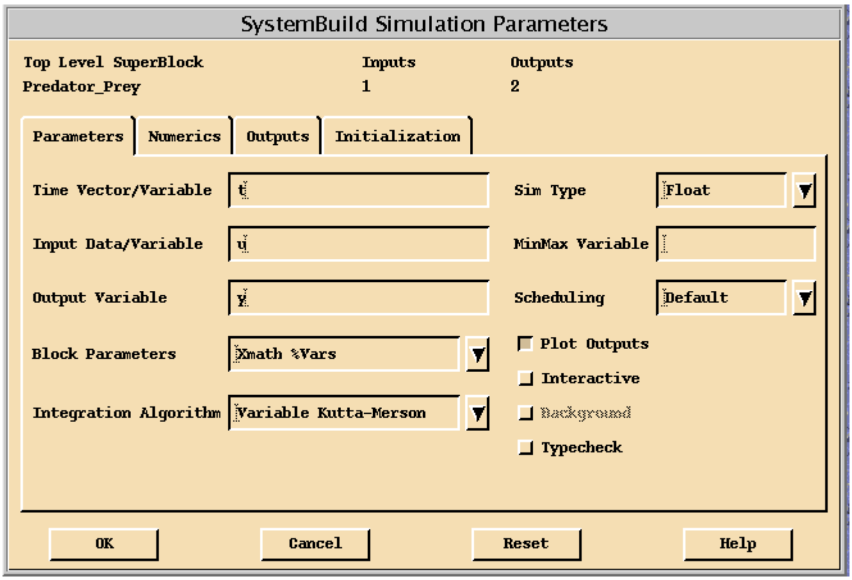

Figure 4-12. SystemBuild Simulation Parameters Dialog

7. As indicated in the SystemBuild Simulation Parameters dialog above, enter **t** in the **TimeVector/Variable** field, **u** in the **Input Data Variable** field, and **y** in the **Output Variable** field; click the **Plot Outputs** box, and click **OK**.

   Once again, you can watch the log area of the Xmath window as the model is analyzed and simulated. The plot appears in a separate window (see Figure 4-11).

Deleting a SuperBlock

Deleting a SuperBlock is an easy process. From the Catalog Browser (either pane, provided the SuperBlock names appear), select a SuperBlock. Then select Edit→Delete.

Once you delete a SuperBlock, it is no longer visible to the Catalog Browser. Any SuperBlock that references the deleted SuperBlock contains the "Undefined" SuperBlock indicator.

Caution Deletion of SuperBlocks cannot be undone.

Navigating a SuperBlock Hierarchy

The use of hierarchy in your SystemBuild models is crucial to the successful implementation of a system. As mentioned before, you use SuperBlocks to create a model hierarchy. This section presents some of the methods for navigating up and down a SuperBlock hierarchy.

Before we get started, we need to start fresh. So, delete all of the SuperBlocks you may have created, or just exit the Catalog Browser (File→Exit), and then restart SystemBuild.

Next, load a model with some hierarchy in it. Following the instructions in the *Loading a Model File* section, load the following file:

`SYSBLD/demo/f14_demo/f14new.cat`

where *SYSBLD* is the SystemBuild distribution directory located under *ISIHOME* for the particular version of MATRIXx software (see the *Directories Defined by Environment Variables* section of Chapter 3, *Xmath*).

Note This section does not simulate this model. If you are interested in this model, from the Xmath command line, type **demo**, and then follow the subsequent dialogs to select the SystemBuild demos.

Navigating from the Catalog Browser

To navigate from the Catalog Browser, you use the left pane of the browser to expand and collapse SuperBlocks within the tree.

After the f14 model is loaded, the Catalog Browser displays only the types of catalog objects (SuperBlocks, State Diagrams, and so forth) in the left pane (see Figure 4-13). Notice that the folder icon next to SuperBlocks is open, indicating that the SuperBlocks are listed in the right pane.

Notice the expand/collapse indicator for the SuperBlocks folder in the left pane. The indicator displays a plus (+), indicating that the folder is collapsed.

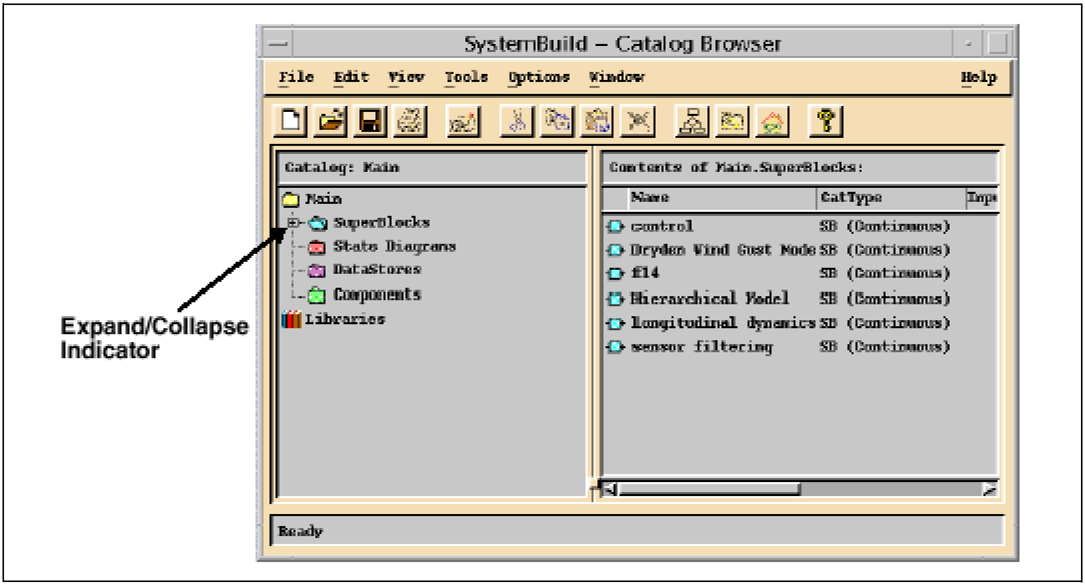

Figure 4-13. Catalog Browser After Loading the f14 Model

To navigate from the Catalog Browser:

1. Expand the hierarchy of the SuperBlocks folder in the left pane by double-clicking the folder or by single-clicking the expand indicator, the plus (+) sign.

2. Continue expanding each level of the model (see Figure 4-14).

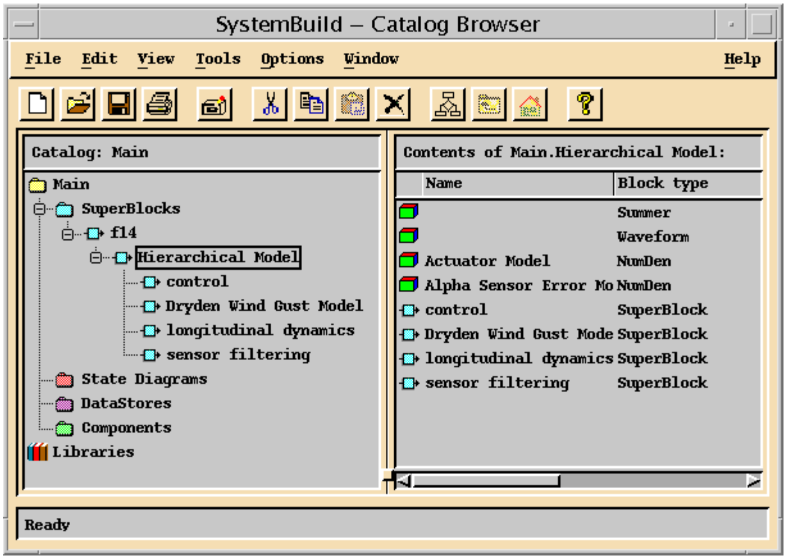

Figure 4-14. Expanded Hierarchy for f14 Model

3. Open the SuperBlock named `sensor filtering` by double-clicking the SuperBlock in the right pane.

   An Editor Window appears with the selected SuperBlock on view. You can edit the contents of the SuperBlock.

4. To navigate to another SuperBlock, go to the Catalog Browser; find the SuperBlock you want to edit, and then open it.

Navigating from the Editor Window

You can navigate up and down a hierarchy from within the Editor window using menu items.

To navigate from the Editor window:

1. From the Catalog Browser, open the `sensor filtering` SuperBlock (see steps 1–3 above).

2. Within the Editor window, select View→Parent to view this SuperBlock's parent.

   Notice now that the Editor is displaying a different SuperBlock named `Hierarchical Model`.

3. To move down the hierarchy in the Editor window, select the desired SuperBlock; then select Edit→Open.

   The Editor now displays the selected SuperBlock.

Printing from the Editor Window

To print the contents of an Editor window, select File→Print. You can change the print command in the dialog. Printing uses the settings you made last in the Page Setup dialog, also available from the File menu. The default printer is specified by the `$PRINTER` environment variable.

You can change the print command and its option(s) that appear in the Print dialog to any print command that is available on your system; you can also specify the list of printers available in the *SYSBLD*`/etc/user.ini` file (see the *SystemBuild User's Guide* for details).

SystemBuild Tutorial

Now that you have a basic understanding of Xmath and SystemBuild, you are ready to build and simulate a model. This section presents a basic tutorial to take you through the basic process from model conception to model simulation.

Building a Block Diagram

A block diagram is the graphical representation of the model using blocks and connections. At a high level, building a block diagram is basically a three-step process:

1. Create a new SuperBlock.
2. Create blocks within the diagram with appropriate block parameters.
3. Connect the blocks together.

In this tutorial, you build a simple spring-mass damper model and then simulate it. This model has the following equations:

$$\begin{aligned} F(t) &= m\ddot{x} + c\dot{x} + kx \\ x &\rightarrow x_1 \\ \dot{x} &\rightarrow x_2 \\ \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \frac{1}{m}[F(t) - cx_2 - kx_1] \end{aligned} \tag{4-1}$$

The block diagram for this model appears in Figure 4-15.

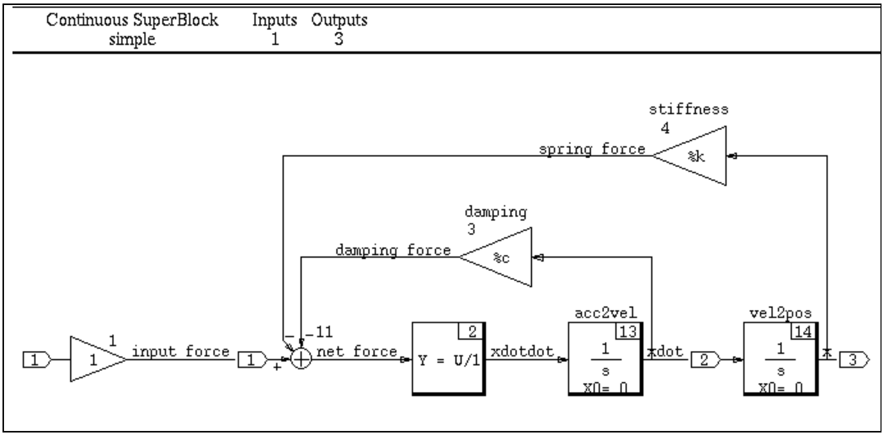

Figure 4-15. Spring-Mass Damper Block Diagram

Creating a New SuperBlock

To create a new SuperBlock called `simple`:

1. Start Xmath as described in the *Starting Xmath* section of Chapter 3, *Xmath*.
2. To invoke SystemBuild, at the Xmath command line (bottom window), enter the following command:

   `build`

   The SystemBuild Catalog Browser loads. If necessary, click on the bar at the top of the window to move it to the front of the display.
3. In the Catalog Browser, create a new SuperBlock by clicking the **New SuperBlock** toolbar button or selecting File→New→SuperBlock.

The SuperBlock Properties dialog appears (see Figure 4-16). Initially, the Attributes tab is active in the dialog, and all of the properties of a SuperBlock are set to their default values.

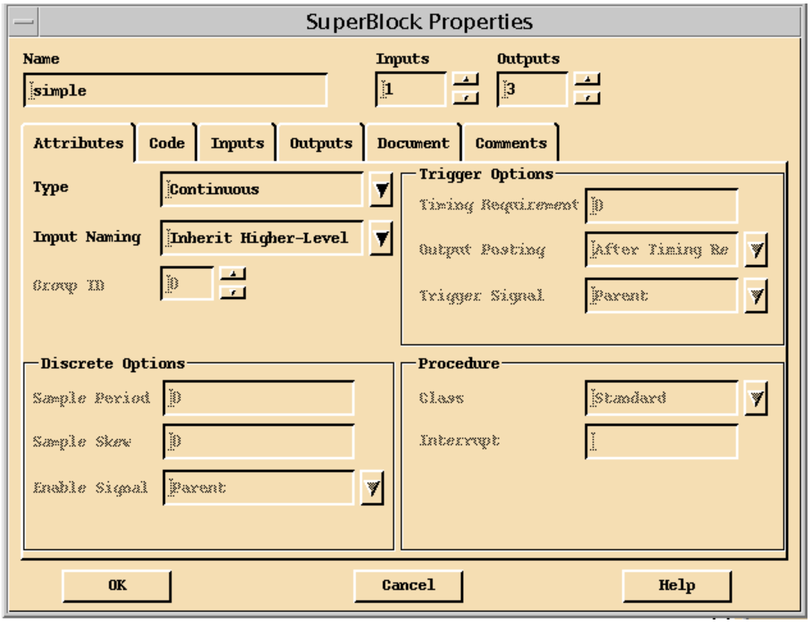

Figure 4-16. SuperBlock Properties for Sample SuperBlock

4. As shown in the dialog in Figure 4-16, perform the following steps:
   a. Name the new SuperBlock **`simple`**.
   b. Verify that the **Type** of the SuperBlock is `Continuous`.
   c. Set the number of **Inputs** to **`1`**.
   d. Set the number of **Outputs** to **`3`**.
   e. Click **OK** to complete the process.

      The SystemBuild Editor window appears; it is empty except for the information bar at the top, which contains the information that you provided above.

Creating and Placing Blocks

The SystemBuild Editor is on view with the SuperBlock properties of `simple` defined. In this section, you create and place all blocks in this model.

Creating the First Block in Your SuperBlock

To create a block into your new SuperBlock `simple`:

1. Open the Palette Browser by using one of the following methods:
   - Click the **Palette Browser** toolbar button
   - Select Window→Palette Browser
   - Double-click in the open space of the Editor

   You might want to move your windows so that the Palette Browser is alongside the Editor window.
2. In the Palette Browser, select the Algebraic palette as shown in Figure 4-3.
3. Using MB2, drag and drop a gain block from the Palette Browser into the editor (see Figure 4-6).

   It is now time to edit the properties of the gain block.
4. In the Editor, select the gain block by moving the mouse cursor over the block and then clicking MB1.

   The block is now highlighted with a border around the block. Notice that more buttons in the toolbar are enabled.
5. Bring up the Block Properties dialog by one of the following methods:
   - Select the **Block Properties** toolbar button (fourth button from left)
   - Select Edit→Block Properties
   - Click the block, and press the **Return** key.

The Block Properties dialog (see Figure 4-17) comes on view.

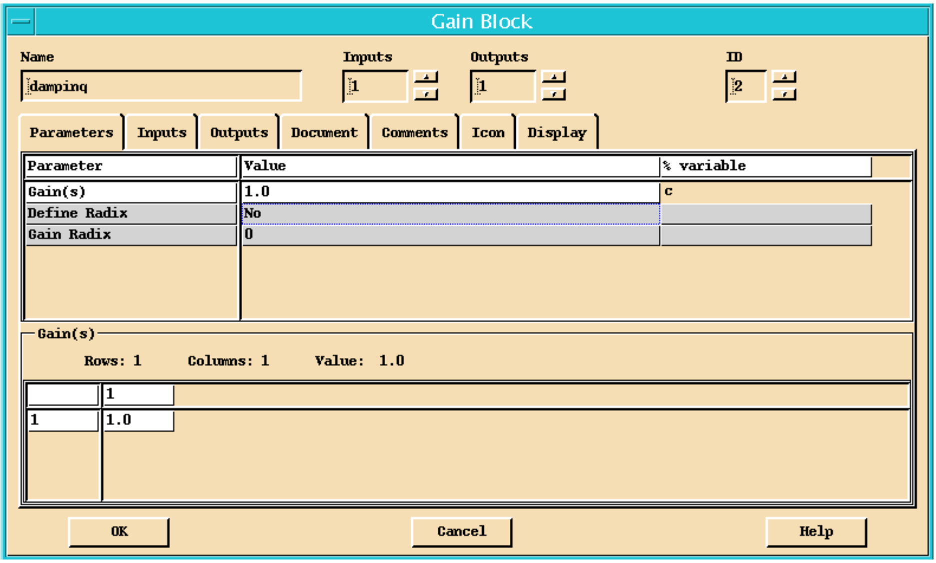

Figure 4-17. Block Properties Dialog (Gain Block)

6. Fill in the Block Properties dialog as follows:
   a. Click in the **Name** field, and enter the name **`damping`**.
   b. From the Parameter's tab, enter the name **`c`** in the **% variable** column of the **Gain(s)** row.

      A `% variable` is a method for entering parameter values into your model via variables in the Xmath workspace. This method allows you to change values in the model between or during simulations without editing the model.
   c. Click the Outputs tab.

      This changes the dialog to contain a table relevant to the block's outputs.
   d. Click within the cell corresponding to output #1, **Output Label**, and enter **`damping force`** (see Figure 4-18).

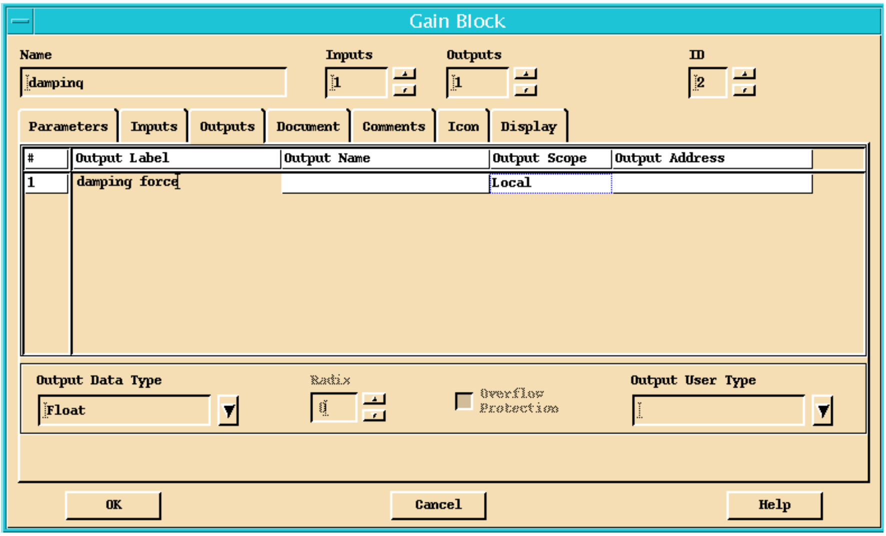

Figure 4-18. Gain Block (Outputs Tab)

e. Click the Display tab (see Figure 4-19), and enable the **Show Output Labels** check box.

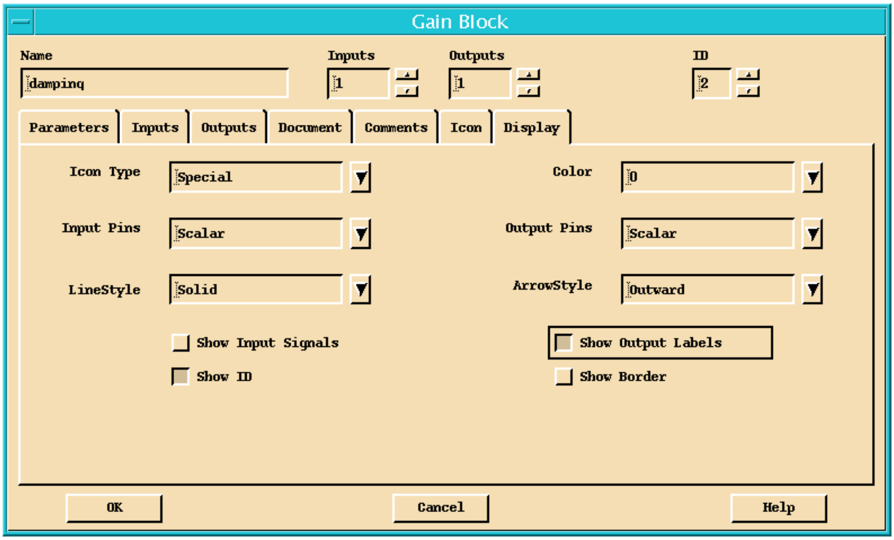

Figure 4-19. Gain Block (Display Tab)

f. Click **OK** to complete creating the block.

Note The gain block is directed left to right. For a better layout of the diagram, we can change its direction to be right to left. This is often needed on feedback paths in block diagrams.

7. (Optional) To flip this block on the diagram, perform one of the following:
   - Select the block and then select Edit→Flip Horizontal.
   - Move the mouse cursor over the block and press **f**.

Creating the Remaining Blocks in the SuperBlock:

You now need to create all of the remaining blocks for this model. You repeat most of the steps listed in the *Creating the First Block in Your SuperBlock* section except that the block types, parameters, and labels are different. Table 4-2 is a summary of the information related to each of the blocks, including the one you've already created. For some blocks, you have to switch between tabs in the Block Properties dialog to enter data in the appropriate fields.

Note Flip the damping and stiffness blocks.

Table 4-2. Block Definitions for Spring-Mass Block Diagram

Palette Name	Block Type	Parameter Tab	Outputs Tab	Display Tab	Code Tab
Algebraic	Gain (flip)	**Name**: damping **Gain %Var**: c	**Output label**: damping force	**Show Output Labels** enabled	—
Algebraic	Gain (flip)	**Name**: stiffness **Gain %Var**: k	**Output label**: spring force	**Show Output Labels** enabled	—
Algebraic	Gain	**Gain Value**: 1	**Output label**: input force	**Show Output Labels** enabled	—

Table 4-2. Block Definitions for Spring-Mass Block Diagram (Continued)

Palette Name	Block Type	Parameter Tab	Outputs Tab	Display Tab	Code Tab
Algebraic	Summer	**Inputs**: 3 **Outputs**: 1 **Number of branches**: 3 **Signs**: [–1, –1, +1]	**Output label**: net force	**Show Output Labels** enabled **IconType**: Special	—
Algebraic	Algebraic Expression	ParamValue row: **Value**: 1 **%Var**: m Press **Control-P**.	**Output label**: xdotdot	**Show Output Labels** enabled	Y=U/P; See note below.
Dynamic	Integrator	**Name**: acc2vel	**Output label**: xdot	**Show Output Labels** enabled	—
Dynamic	Integrator	**Name**: vel2pos	**Output label**: x	**Show Output Labels** enabled	—

Note For the code in the algebraic expression, `Y=output`, `U= input`, `P= parameter value` that you input under the Parameters tab, SystemBuild substitutes the value of the parameter into your equation in the block. You use **Control-P** to transfer the value of the parameter into the Xmath variable `m`. You might also want to check the online Help for "AlgebraicExpression."

After creating all of those blocks, arrange your diagram so that it looks like Figure 4-20. Note that the block IDs might be different, and that is expected.

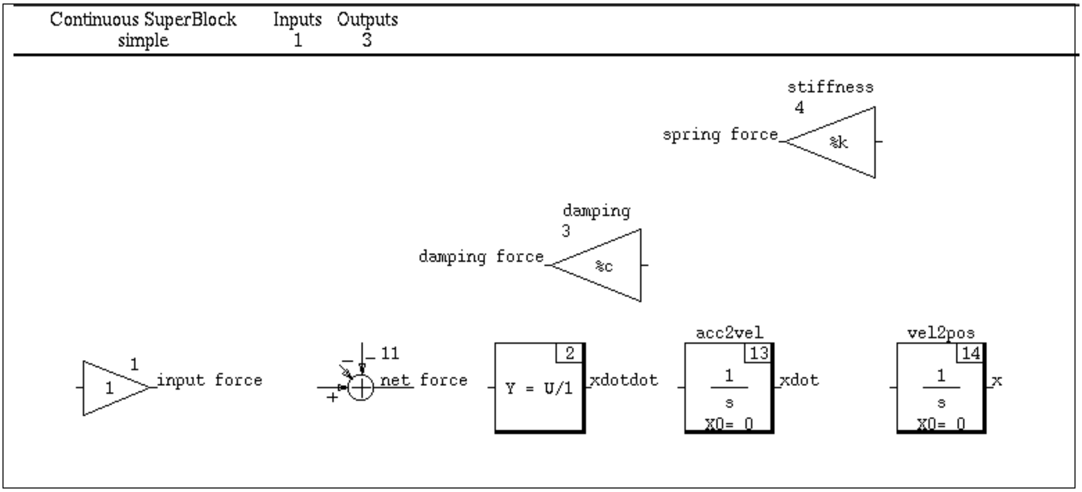

Figure 4-20. Sample Diagram without Connections

Connecting Blocks

Connecting the blocks completes the process of building a SystemBuild model. This section assumes you have successfully created the blocks in the diagram and that your diagram looks like Figure 4-20.

Making Internal Connections on the Diagram

To make internal connections on the diagram:

1. Click the source block with the middle mouse button.

   Start with the gain block that has the `input force` output label.

2. Click the destination block with the middle mouse button.

   The summer block is the target for the first block.

The Connection Editor, shown in Figure 4-21, appears.

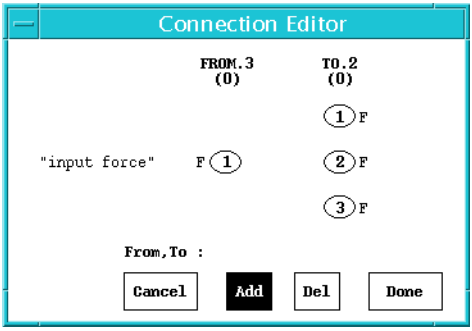

Figure 4-21. Connection Editor

In the Connection Editor, the FROM or source block (the ID of the first block you clicked on) appears on the left side of the window while the TO or destination block (the ID of the second block you clicked on) appears on the right side.

- All of the source block's outputs as well as all the destination block's inputs are listed; the number in each box represents the channel number of the signal for that block
- Output labels and input names, if any, are displayed.
- The datatype of each signal is abbreviated; in this case F, indicates Float, the default datatype.

3. Click the appropriate channel of the source block and the appropriate channel of the destination block, and then click **Done**.

   SystemBuild makes the connection using the **Add** button selection, which is the default.

4. Continue to connect the blocks as shown in Figure 4-15.

Note When connecting a block with one output to another with one input, SystemBuild makes the connection automatically without displaying the Connection Editor.

Making External Connections on the Diagram

External connections represent the interface between a SuperBlock and its outside world. External connections are represented on block diagrams with a special symbol or flag. The number within a flag is the channel number of the signal it represents.

The spring-mass damper model (see Figure 4-15) has one external input and three external outputs in the tutorial. Figure 4-22 shows some of the external I/O flags from this model. Notice that there is only one way to represent an external input connection, while there are two possible ways to represent an external output connection. The flag shown for external output 2 indicates that the same block output is connected to an external output and to another block.

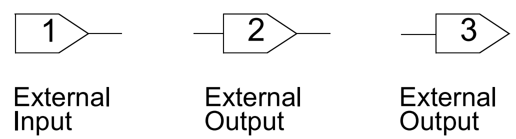

Figure 4-22. External I/O Flags from Spring-Mass Damper Model

Note You can connect a single external input to as many block inputs as needed, but only one signal can be connected to a single external output in each SuperBlock.

The process of making an external connection is similar to making an internal connection. Just as you selected a source and destination block for internal connections, you select a source and destination for external inputs and outputs. The external source (input) or destination (output) is any open area or blank space in the SuperBlock. So, instead of middle-clicking on a block for your source (external input) or destination (external output), you middle-click on the open space.

To make external input connections:

1. Move the mouse cursor to the open space so that it is not over a block, and click the middle mouse button.
2. Move the mouse cursor over the destination block and click the middle mouse button.

   For this model, the external input connection connects to the gain block with the output label `input force`.

Note When there is only one external input, SystemBuild makes the connection automatically without the use of the Connection Editor.

3. Connect the external input to the appropriate channel for the destination block.
4. Continue to make any other external input connections using the same methodology.

   This example has only one external input connection.

To make external output connections:

1. Move the mouse cursor over the source block, and click the middle mouse button.

   For this model, the first external output is the gain block with the output label `input force`.
2. Move the mouse cursor over an open space in the diagram, and then middle-click.

   The Connection Editor comes on view.

Note When there is only one external output, SystemBuild makes the connection automatically without the use of the Connection Editor.

3. Connect the source block's appropriate channel to the appropriate external output channel and click **Done**.
4. Repeat the previous steps for the remaining external output connections.

   The sample has two additional external connections from the integrator blocks.

   When complete, your diagram should be similar to Figure 4-15.

Saving the Tutorial Model

After you have completed the diagram, we recommend that you save your work. As with many things in SystemBuild, there are many ways to save the model. You can save the model using Xmath or from the Catalog Browser. We'll use the Catalog Browser method. (See the *SystemBuild User's Guide* for other methods.)

To save the model using the Catalog Browser:

1. Update the current SuperBlock in the Editor Window into the Catalog Browser by selecting File→Update.
2. Switch to the Catalog Browser and refresh its contents: View→Update.

   These two steps update the contents of the catalog and show those changes in the Catalog Browser's panes.
3. Select File→Save As in the Catalog Browser to open the Save dialog.
4. Select a directory.

Warning You must select a directory where you have write permission to save the file.

5. Enter the name **`simple.cat`**.

Note The filename extension of a SystemBuild save file is not predefined, meaning that you can use any extension you want. Some common extensions include `.cat`, `.sbd`, `.mdl`, and `.dat`.

6. Click **OK**.

Simulating the Model

With the model completed, you can now simulate it. We describe the method for executing the simulation from the Xmath command area.

To simulate the model from the Xmath command area:

1. Define a time vector as an Xmath variable. In the Xmath command area, type:

   ```
   t = [0:0.01:4]';
   ```

   Notice that the time variable must be a column vector.

   You can use any variable name, but we use our convention of using `t` for time, `u` for inputs, and `y` for outputs.
2. Next, define the input vector of magnitude one:

   ```
   u = ones(t);
   ```

   The input variable must also be a column vector with a one-to-one correspondence with the time vector.

3. Enter the %variable values in Xmath:

```
k = 1000;
c = 2;
m = 1;
```

4. Finally, execute the simulation (see the *Simulating the Model from the Xmath Commands Window* section):

```
y = sim("simple", t, u, {graph});
```

After the simulation completes, you should see the results plotted (see Figure 4-23).

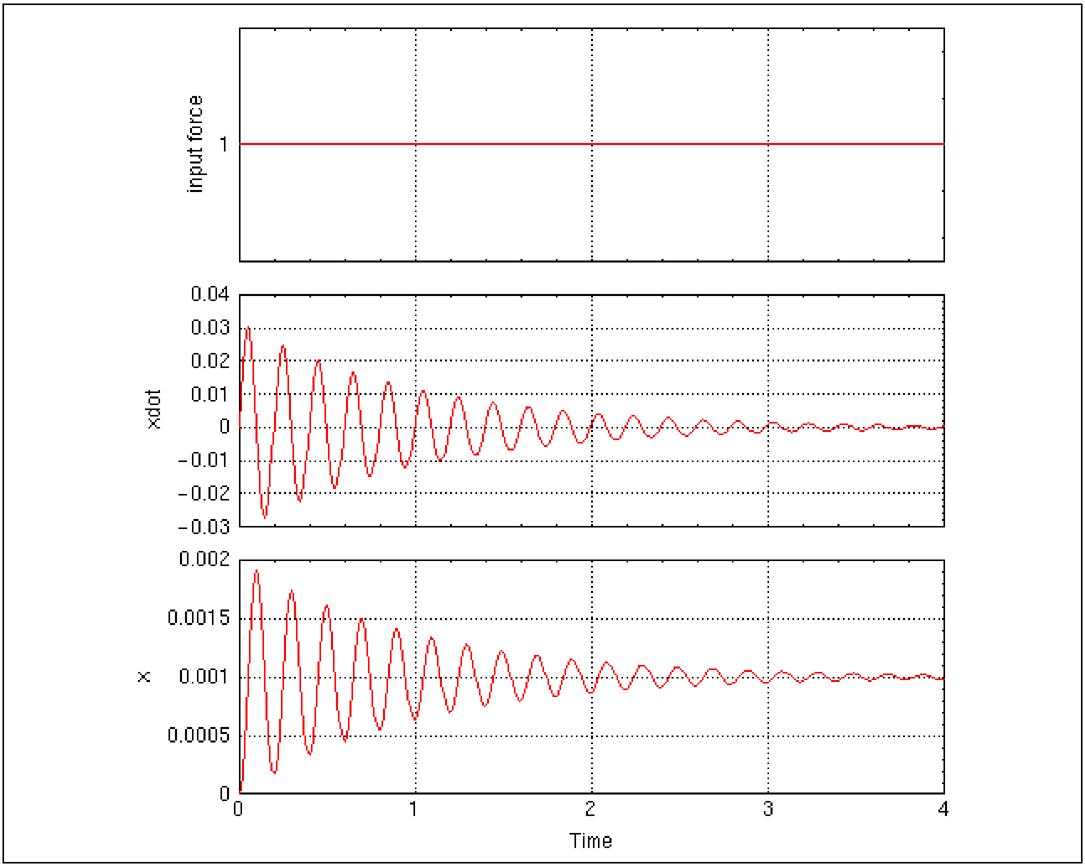

Figure 4-23. Simulation Results

5 AutoCode

The AutoCode software lets you generate ANSI C or Ada code automatically from SystemBuild models.

You can generate code from the Catalog Browser in SystemBuild or use the `autocode` Xmath command. The generated code represents a complete implementation of the model. The generated code can be targeted for and run on other computers or an actual controller. The default target is a stand-alone simulation that you can execute on your computer; you can then load the results of the simulation back into Xmath for analysis.

Generating Non-Customized Code

To generate code for the sample Discrete Cruise System model, follow the steps below. We assume that you have Xmath running on your terminal.

1. Make sure you are in a directory where you have write permission for saving your code. If not, enter the command below from the Xmath command window, substituting your directory name:

   `set directory ="`*`your_working_directory`*`"`

2. From the Xmath command line, type the following command to load the model:

   `load "$SYSBLD/demo/cruise_demo/cruise_d.cat";`

Note The Xmath command line is the only place that can recognize environment variables. For loading with other methods, you must know that full pathname of the SystemBuild directory.

3. From the SystemBuild Catalog Browser, select the Discrete Cruise System SuperBlock.

Note You must generate code from a top level SuperBlock.

4. From the Catalog Browser, select Tools→AutoCode to bring up the Generate Real-Time Code dialog (see Figure 5-1).

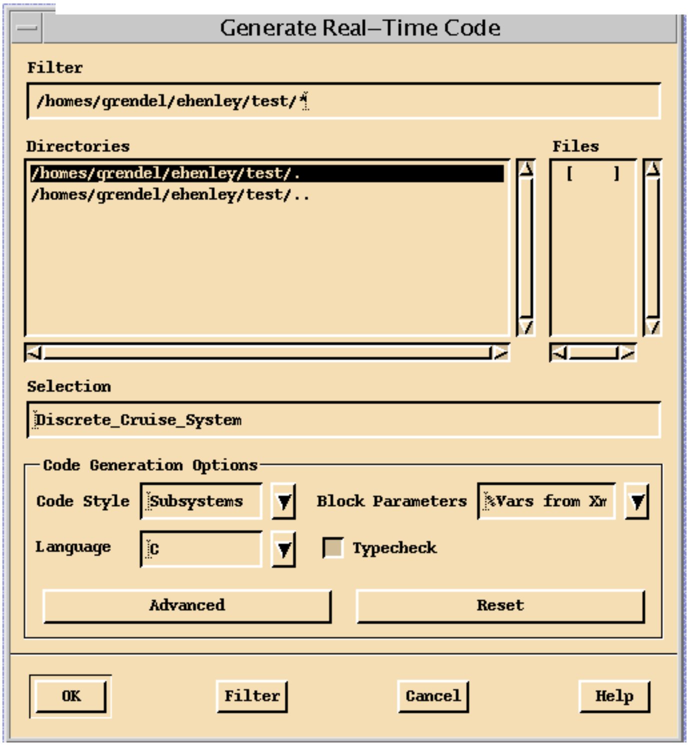

Figure 5-1. Generate Real-Time Code Dialog

5. Enter a name in the **Selection** field or accept the default, `Discrete_Cruise_System`.
6. Click **OK** to start the code generation process.
7. Raise the Xmath Commands window to monitor the progress of the code generation.
8. Once the code generation is complete, look for a statement similar to the following in the Xmath log area:

```
Output generated in Discrete_Cruise_System.c.
Code generation complete.
```

9. (Optional) Display the output file in the Xmath Output window by entering a command similar to the following in the Xmath Command window:

```
oscmd ("more Discrete_Cruise_System.c")
```

Customizing Generated Code

To customize your AutoCode output, click **Advanced** on the Generate Real-Time Code dialog; this brings up the Advanced dialog (see Figure 5-2).

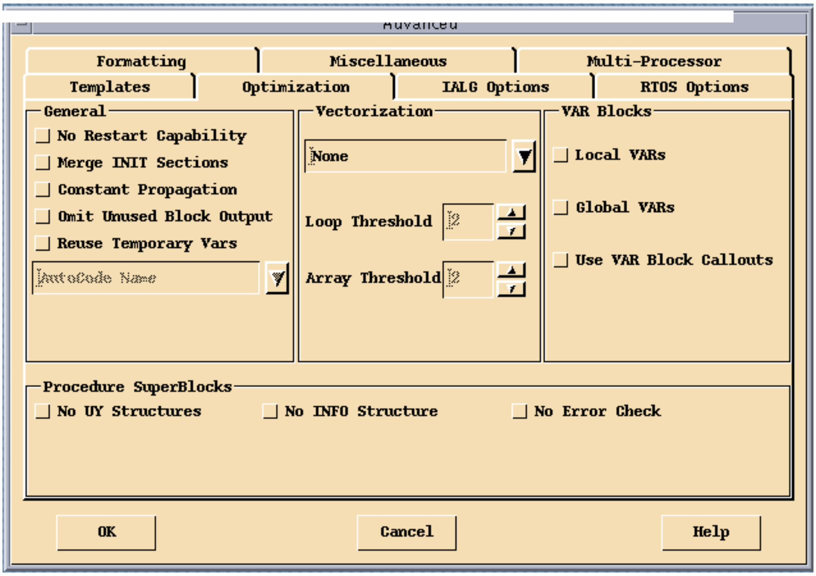

Figure 5-2. Advanced Dialog

You can use the Advanced dialog from the AutoCode Code Generation dialog or use keywords with the `autocode` Xmath command to customize the generated code as follows:

- Specifying a template file on the Templates tab allows you to control the formatting of the output of AutoCode to meet a variety of software needs; you can modify the overall architecture of generated code, customize the scheduler, modify data structures and external I/O calls, add user code, and so forth. Using the Template Programming Language (TPL), you can tailor any part of the code except the hierarchy logic and the elementary blocks.
- Formatting options (Formatting tab) let you set maximums, such as the number of significant digits, the length of variable names, and columns per row. From here, you can also specify indentation between levels, as well as set a number of other parameters.
- The IALG (Integration Algorithms) Options tab lets you select an algorithm such as Euler or Runge Kutta.

- The Multi-Processor tab lets you specify a processor, startup, background, interrupt, skew, priority, or map file.
- The Optimization tab lets you make general, vectorization, and VAR block settings that affect code size and efficiency (see the *Autocode Reference* for details).
- The Miscellaneous tab lets you select an options file, the type of scheduler, output scope control, and various other settings.
- The RTOS (real-time operating system) Options tab lets you specify a configuration file and set additional options.

Once you have customized your settings, you click **OK** in the Advanced dialog; then you generate code by clicking **OK** in the Generate Real-Time Code dialog.

For information about compiling, executing, and using the generated code, see the *AutoCode User's Guide.* For information about `autocode` keywords, see the Xmath online Help.

6

DocumentIt

The DocumentIt software generates block-level documentation for SystemBuild models. The DocumentIt software extracts the parameters of the SuperBlocks and elementary blocks in your model and any comments you have entered for each block; it then formats the documentation according to guidelines you define. You can generate documentation from the Xmath command area or from the SystemBuild Catalog Browser. You can invoke controls as arguments from the command area or make choices in a user dialog.

This chapter provides an introduction to using DocumentIt. For a complete description, see the *DocumentIt User's Guide*.

Generating Non-Customized Documentation

To generate documentation for the sample Discrete Cruise System model, follow the steps below. We assume that you have Xmath running on your terminal.

1. Make sure you are in a directory where you have write permission for saving your code. If not, enter the command below from the Xmath command window, substituting your directory name:

   set directory ="*your_working_directory*"

2. From the Xmath command line, type the following command to load the model:

   load "$SYSBLD/demo/cruise_demo/cruise_d.cat";

Note The Xmath command lineis the only place that can recognize environment variables. For loading with other methods, you must know that full pathname of the SystemBuild directory. See the *Directories Defined by Environment Variables* section of Chapter 3, *Xmath*, for additional information.

3. From the SystemBuild Catalog Browser, select the Discrete Cruise System SuperBlock.

Note You must generate documentation from a top level SuperBlock.

4. Select Tools→DocumentIt to bring up the Generate Documentation dialog (see Figure 6-1).

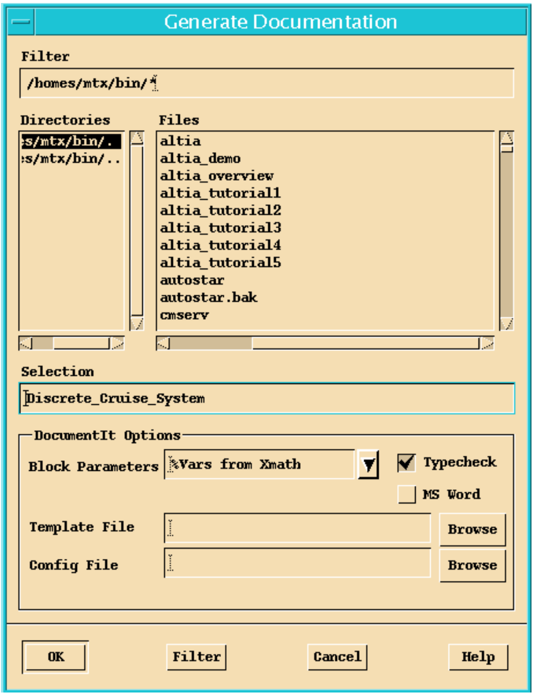

Figure 6-1. Generate Documentation Dialog

5. Choose a directory, and enter a name in the **Selection** field or accept the default, `Discrete_Cruise_System`.

Note You do not need to supply the extension. DocumentIt supplies the default, `.doc`, for you.

6. Click **OK** to start the document generation process.
7. Raise the Xmath Commands window to monitor the progress of the documentation generation.
8. Once the document generation is complete, look for a statement similar to the following in the Xmath Log window:

```
Documentation generation complete.
Document generated and saved in file:
Discrete_Cruise_System.doc.
```

Note The .doc file is in ASCII format. The current defaults also produce an RTF file, which contains Microsoft Word markup commands.

9. (Optional) Display the output file in the Xmath Output window by entering a command similar to the following in the Xmath Commands Window:

   oscmd ("more Discrete_Cruise_System.doc")

   Example 6-1 provides an example of DocumentIt output; this example is the first two pages of output from the Discrete_Cruise_System document.

Generating Customized Documentation

You can customize documentation generated with DocumentIt by using templates. Template files are ASCII files containing text, interspersed with template command parameters that specify DocumentIt output. The TPL programming language lets you modify the templates to control the output of DocumentIt to meet a variety of needs. Various templates are available.

In addition to template command parameters, you can also place publishing software markup commands (for example, FrameMaker or WordPerfect markup commands) in template files, which DocumentIt writes directly to the ASCII output file. The markup commands automatically format the document when it is imported into the corresponding publishing software. See the *Template Programming Language User's Guide*.

Unlike AutoCode, DocumentIt does not have a dialog box for advanced features.

Example 6-1. Example of DocumentIt Output

```
**********************************************************************
|                DocumentIt (TM) Document Generator latest
|             INTEGRATED SYSTEMS INC.,   SUNNYVALE, CALIFORNIA
**********************************************************************


rtf filename           : Discrete_Cruise_System.rtf
Filename               : Discrete_Cruise_System.doc
Dac filename           : /homes/mtx/solaris_latest/xmath/../case/DIT/
templates/ascii/documentit.dac

Generated on           : Fri Jun 19 10:54:59 1998
Dac file created on    : Wed May  6 02:49:00 1998
```

```
*****************************************************************

Number of DataStores in this model = 1

---------------------------------------------------------
DataStore Desired Speed (#0)
---------------------------------------------------------
Description :
No. of Registers = 1
No. of Inputs = 1
No. of Outputs = 1

Name            DataType      UoM          Limit/Range          Accuracy
set speed        DOUBLE                      0.0-0.0               0.0

*****************************************************************
Number of SuperBlocks inputs this model = 6
SUPERBLOCK[0] = Discrete Cruise System
SUPERBLOCK[1] = Cruise Control System
SUPERBLOCK[2] = Set Speed
SUPERBLOCK[3] = Controller Logic
SUPERBLOCK[4] = mux3
SUPERBLOCK[5] = continuous automobile

Number of Unique SuperBlocks inputs this model = 6
SUPERBLOCK[0] = 0
SUPERBLOCK[1] = 1
SUPERBLOCK[2] = 2
SUPERBLOCK[3] = 3
SUPERBLOCK[4] = 4
SUPERBLOCK[5] = 5

-----------------------------------------------------------
Top Level SuperBlock model is Discrete Cruise System
-----------------------------------------------------------
LEVEL = 0
NUM_SB_IN_I = 6
NUM_SB_OUT_I = 3
SB_FREQ_R = 50.0
SB_SAMPLE_R = 0.02
SB_SKEW_R = 0.0
SB_ACTV_SIG_S = Parent
SB_OUT_POST_S =
SB_ATTR_S = Discrete
SB_HAS_IN_B = 1
SB_HAS_IN_DATA_B = 1
SB_IS_DSCR_B = 1
SB_IS_TRIG_B = 0
External Data Elements
```

```
----------------------
Discrete Cruise System External Inputs
------------------------------------------

ID       Desc             DType       UoM        Limit/Range      Accuracy

set                       DOUBLE                 0.0-0.0          0.0
resume                    DOUBLE                 0.0-0.0          0.0
disengage                 DOUBLE                 0.0-0.0          0.0
brake                     DOUBLE                 0.0-0.0          0.0
manual throttle           DOUBLE                 0.0-0.0          0.0
incline                   DOUBLE                 0.0-0.0          0.0

Discrete Cruise System External Outputs
------------------------------------------

Internal Data Elements
----------------------

LEVEL = 1

Cruise Control System Inputs
---------------------------------

ID       Desc             DType       UoM        Limit/Range      Accuracy

set                       DOUBLE                 0.0-0.0          0.0
resume                    DOUBLE                 0.0-0.0          0.0
disengage                 DOUBLE                 0.0-0.0          0.0
brake                     DOUBLE                 0.0-0.0          0.0
auto speed                DOUBLE                 0.0-0.0          0.0
acceleration              DOUBLE                 0.0-0.0          0.0
```

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 include the following:
  - **Self-Help Resources**—For immediate answers and solutions, visit our extensive library of technical support resources available in English, Japanese, and Spanish at `ni.com/support`. These resources are available for most products at no cost to registered users and include software drivers and updates, a KnowledgeBase, product manuals, step-by-step troubleshooting wizards, conformity documentation, example code, tutorials and application notes, instrument drivers, discussion forums, a measurement glossary, and so on.
  - **Assisted Support Options**—Contact NI engineers and other measurement and automation professionals by visiting `ni.com/support`. Our online system helps you define your question and connects you to the experts by phone, discussion forum, or email.
- **Training**—Visit `ni.com/training` for self-paced tutorials, videos, and interactive CDs. 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.