MATRIXx™

Getting Started Guide

April 2004 Edition
Part Number 371148A-01

Worldwide Technical Support and Product Information

ni.com

National Instruments Corporate Headquarters

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

Worldwide Offices

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

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

Important Information

Warranty

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

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

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

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

Copyright

Trademarks

AutoCode™, 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.

Conventions

The following conventions are used in this manual:

»	The » symbol leads you through nested menu items and dialog box options to a final action. The sequence **File»Page Setup»Options** directs you to pull down the **File** menu, select the **Page Setup** item, and select **Options** from the last dialog box.
	This icon denotes a tip, which alerts you to advisory information.
	This icon denotes a caution, which advises you of precautions to take to avoid injury, data loss, or a system crash.
bold	Bold text denotes items that you must select or click in the software, such as menu items and dialog box options. Bold text also denotes parameter names.
italic	Italic text denotes variables, emphasis, a cross reference, or an introduction to a key concept. This font also denotes text that is a placeholder for a word or value that you must supply.
`monospace`	Text in this font denotes text or characters that you should enter from the keyboard, sections of code, programming examples, and syntax examples. This font is also used for the proper names of disk drives, paths, directories, programs, subprograms, subroutines, device names, functions, operations, variables, filenames, and extensions.
`monospace bold`	Bold text in this font denotes the messages and responses that the computer automatically prints to the screen. This font also emphasizes lines of code that are different from the other examples.
`monospace italic`	Italic text in this font denotes text that is a placeholder for a word or value that you must supply.
Platform	Text in this font denotes a specific platform and indicates that the text following it applies only to that platform.

Contents

Chapter 1
Introduction to the MATRIXx Product Family

Chapter 2
MATRIXx Publications, Help, and Customer Support

Chapter 3
Xmath

Chapter 4
SystemBuild

Chapter 5
AutoCode

Chapter 6
DocumentIt

Appendix A
Technical Support and Professional Services

Index

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. Refer to the *Xmath* section.
- **SystemBuild**—A graphical programming environment that uses a block diagramming paradigm with hierarchical structuring for modeling and simulation of linear and nonlinear dynamic systems. Refer to the *SystemBuild* section.
- **AutoCode**—Template technology used to process SystemBuild model files to produce C or Ada code. Advanced template programming language (TPL) template technology provides a powerful programming capability to tailor the generated code to specialized needs. Refer to the *AutoCode* section.
- **DocumentIt**—TPL template technology (similar to AutoCode) used to capture information from SystemBuild model files and then format it to create documentation. Refer to the *DocumentIt* section.

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

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

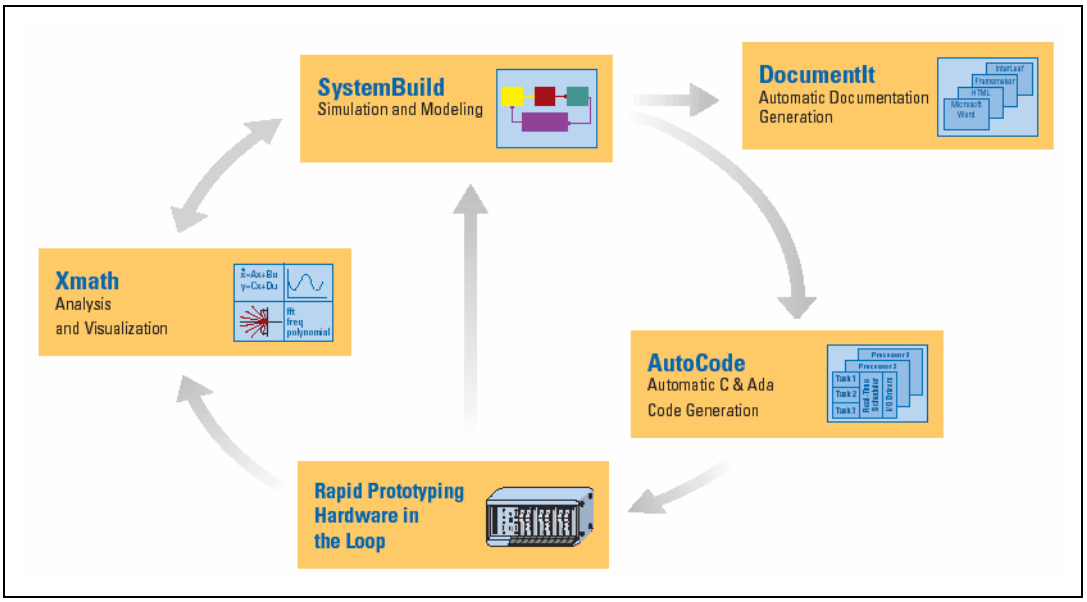


Figure 1-1. MATRIXx Product Family Overview

Table 1-1. Product Dependencies

		Products Needed			
		Xmath	**SystemBuild**	**AutoCode**	**DocumentIt**
Product to Add	**SystemBuild**	X	X	—	—
	AutoCode	X	X	X	—
	DocumentIt	X	X	—	X

Xmath

Xmath software provides a system analysis and visualization software environment with 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 the Xmath Basics section, whereas the *Xmath Help* provides easy access to Xmath commands and functions.

- The Xmath commands support basic operations such as creating, plotting, saving, and loading data, and accessing *Xmath Help*. The *Introduction to Xmath* section of Chapter 3, *Xmath*, describes the capabilities of Xmath and the Xmath modules.
- The 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 aerospace and automotive 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 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, refer to 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, refer to 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, 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, refer to Chapter 6, *DocumentIt*.

2

MATRIXx Publications, Help, and Customer Support

This chapter provides publication conventions and instructions for using MATRIXx online documents and Help. It also contains an annotated list of the online documents, and concludes with directions for obtaining release information and customer support.

- *Online and Printed Book Conventions*
- *Using Online Books*
- *MATRIXx Installation Guides*
- *MATRIXx Getting Started Guide and Master Index*
- *Xmath Books*
- *SystemBuild Books*
- *AutoCode and DocumentIt Books*
- *Using Help*
- *MATRIXx Release Information*

Online and Printed Book Conventions

The MATRIXx online and printed books use several types of conventions: font, format, symbols, mouse, and levels of notes. These conventions are discussed in the sections that follow.

Format Conventions

Xmath output appears in **`Monospace bold`** directly below the `Monospace` input (refer to Example 2-1). If the output is extremely large, continuation marks (… or `:`) are used to indicate continuation, or replace missing parts.

Example 2-1 Xmath Sample Input and Output

```
x=random(2,6)
x (a rectangular matrix) =

  0.827908    0.926234    0.566721    0.571164 ...
  0.559594    0.124934    0.727922    0.267777 ...

x'
ans (a rectangular matrix) =

  0.827908     0.559594
      :            :
      :            :
  0.0568928    0.988541
```

If the input is long, continuing lines of input are indented as shown in Example 2-2.

Example 2-2 Sample Convention for Handling Longer Lines of Code

```
Sys=system(makepoly([1,-1.63,5.5],"s"),
  makepoly([1,2.7,5.6,13.5,8.1],"s"))
Sys (a transfer function) =

           2
          s - 1.63s + 5.5
  -----------------------------
   4      3      2
  s + 2.7s + 5.6s + 13.5s + 8.1

initial integrator outputs
  0
  0
  0
  0

  Input Names
  -----------
  Input 1
```

```
Output Names
------------
Output 1

System is continuous
```

Symbol Conventions

Symbols used in this manual include those shown in Table 2-1:

Table 2-1. Symbol Conventions

Symbol	Use
%	UNIX operating system prompt for C shell. Xmath input shows no prompt, as you will usually be typing in the **Xmath Commands** window command area.
$	UNIX operating system prompt for Bourne and Korn shells.
{ }	Braces denote optional arguments or keywords in Xmath syntax. For example: `[out1,out2]=fun(in1,in2,{in3,keywords}`
[ ]	Brackets indicate that the enclosed information is optional. The brackets are generally not typed when the information is entered.
|	A vertical bar separating two text items indicates that either item can be entered as a value.

Mouse Conventions

This document assumes you have a standard, right-handed two- or three-button mouse. From left to right, the buttons are referred to as MB1, MB2, and MB3 for a right-hand mouse definition; these buttons are right to left for a left-hand mouse definition. For workstations with a two-button mouse, MB1 is the left button and usually the right button behaves as MB3.

All instructions assume MB1 unless otherwise noted. Some common mouse instructions are shown in Table 2-2.

Table 2-2. Common Mouse Instructions

Instruction	Use
click	Press, then quickly release, MB1.
double-click	Rapidly click MB1 twice.
drag	Hold down MB1 while moving the mouse; release the button when the desired result is obtained.

The following mouse-click combinations are useful for selecting text:

- To select a word, point anywhere within the desired word and double-click.
- To select an entire line, point anywhere on the line and triple-click.
- To select all text in an **Xmath** window area, move the cursor into the area and quadruple-click.

Using Online Books

The MATRIXx CD-ROM contains software and documentation in PDF format. With Acrobat Reader you can view, search, and print any document on the MATRIXx Installation.

Using Online Books describes viewing, printing, and searching the online PDF files.

To determine whether your copy of Acrobat Reader includes Search, launch the program, and examine the toolbar (select **Window»Show Tool Bar** if the toolbar is not visible). If Search is installed, the toolbar ends with the four search buttons shown in the *Find and Search in PDF* section.

Viewing, Printing, and Searching PDF Files

To view the documentation, launch Adobe Acrobat Reader with Search (version 3.0 or later); then open the `matrixx.pdf` file.

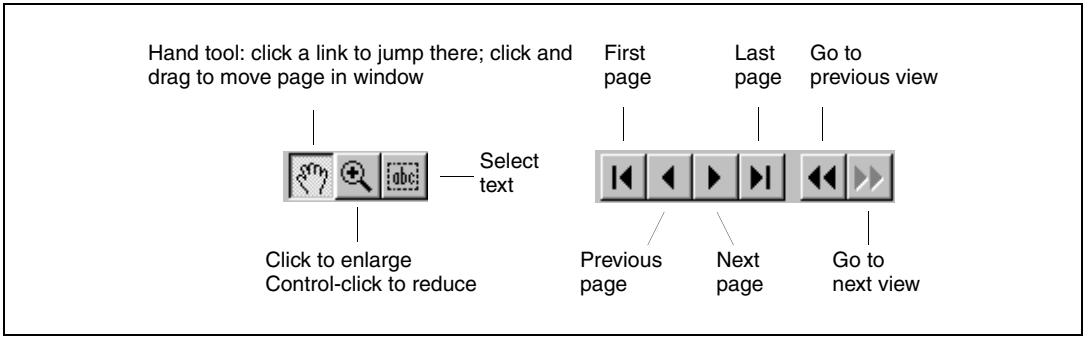
The `matrixx.pdf` file is an overall table of contents for PDF documentation on the CD-ROM. Click any document title to open that document.

Using Acrobat Reader

Each document has a bookmarks pane displayed on the left.

All bookmarks and blue text are hypertext links. To follow a link, be sure the hand tool is selected; then click the bookmark or blue text.

Use the following Acrobat toolbar buttons for browsing and navigation:



Select **Help»Reader Guide** for a detailed description of all Acrobat capabilities.

Bookmarks can contain the following links:

- **Document Title**—Links to the cover of the current document.
- **Contents**—Links to the table of contents for the current document.
- **Chapter and Section Bookmarks**—Links to chapters and sections in the document.
- A plus sign **+** in front of a bookmark means there are sub-bookmarks; click the **+** to expand to lower-level bookmarks.
- **Index**—Links to the index of the current document if one exists.

Pasting Text into Other Applications

To copy examples or text from a PDF document into an application, first click **Tools»Select Text**, or click the **abc** button. Select the text, and then use the usual technique on your platform to copy and paste the text into the target application.

Note In some cases, example text can contain special typeset characters, such as a non-breaking space, or special left- or right-facing delimiters (“_”, ‘_’), and so on, that will not be properly parsed when pasted into an application. If you receive an error message, retype the special characters, and the input will be processed. The following string is an example:

```
plot(a,b, { xlab = “label_string” })
```

If the application does not understand the paired double quotes (“_”) you need to retype them so the input contains straight quotes:

```
plot(a,b, { xlab = "label_string" })
```

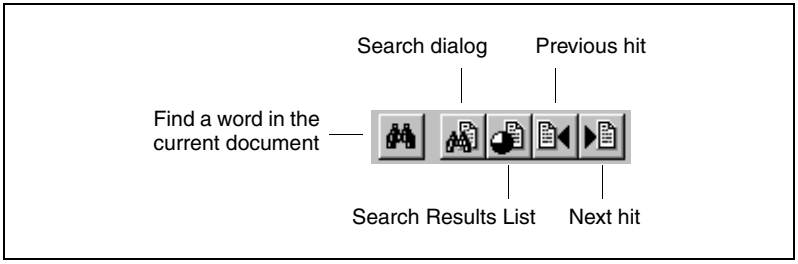
Printing Documents

To print a file, select **File»Print** and then specify the desired pages. Notice that the PDF page numbers appearing at the bottom of the Acrobat screen count every page, including the cover, the table of contents, and so forth. Be sure to use these page numbers (rather than the document page numbers) when printing a range of pages.

Note You must print PDF files on a PostScript printer.

Find and Search in PDF

The Acrobat Reader Find feature locates words or word phrases in the current PDF document.



To use Find, click the plain binoculars on the toolbar, or select **Tools»Find**.

To make a full-text search over all documents on the CD-ROM, use the Search tool. Click the binoculars overlaid on a document on the toolbar, or select **Tools»Search**. In the Search dialog box, enter the word or phrase you want to find, select options as desired, and then click **Search**. Search displays a Search Results List dialog of documents containing the term.

Click any document in the list to open it. All instances of the term on the first page on which it occurs are highlighted.

To find the next or previous occurrence, click the **Next** or **Previous** buttons on the Acrobat toolbar. To return to the Search Results List dialog box, click the **Search Results List** button.

The Search command includes powerful features for expanding a search using automatic word-stemming, a thesaurus to find synonyms, and a sounds like feature. You also can use wild cards in terms, control case matching, and include Boolean connectives.

For further details on Search, select **Help»Plug-In Help»Using Acrobat Search** to read the online guide.

Note The Search feature uses a search index file. When you open `matrixx.pdf` or any other document, Acrobat automatically attaches the required index file.

Acrobat sometimes attaches the index file more than once. Then, the Search Results List dialog contains the same document multiple times. The workaround is as follows:

1. Open the Search dialog, then click the **Indexes** button.
2. Remove *all* indexes by selecting each index in turn and clicking the **Remove** button. When you remove the final index, you are warned that you will not have any indexes; click **OK**. Close the dialog box.
3. Exit and restart Acrobat; then re-open the `matrixx.pdf` file.

MATRIXx Installation Guides

The following documents provide instructions for installing the MATRIXx product family software:

- **(Windows)** *System Administrator Guide (Windows)*—Describes proper setup of a PC running Windows NT/98 for the installation of MATRIXx software products.
- **(UNIX)** *System Administrator's Guide (UNIX)*—Describes proper setup of a UNIX workstation for the installation of MATRIXx software products.
- *Macrovision FLEXlm End User Manual*—Describes FLEXlm from the end-user perspective. It explains how to use the command-line tools that are part of the standard FLEXlm distribution.

MATRIXx Getting Started Guide and Master Index

The following MATRIXx documents provide help for getting started with basic tasks and for finding the information you need.

- *MATRIXx Getting Started Guide*—Describes the MATRIXx product family and provides an introduction to basic tasks and tutorials for using MATRIXx software.
- *Core Documentation Master Index*—The MATRIXx core documentation suite includes the *MATRIXx Getting Started Guide*, and all Xmath, SystemBuild, Autocode, and DocumentIt documents. This document indexes all of the Core Documentation suite except the *Interactive System Identification Module*, *Part 2*, and the *Xμ Module*.

Xmath Books

Xmath software is documented in the *Xmath User Guide* (formerly *Xmath Basics*) and in manuals for each optional Xmath module.

- *Xmath User Guide*—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 (ISIM), 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 ISIM 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 Module*—Describes Xmath functions used for modeling, analysis, and synthesis of linear robust control systems.

SystemBuild Books

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

- *SystemBuild User 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.
- *SystemBuild Aerospace Model Libraries*—Describes libraries of SystemBuild models that were written for the aerospace industry.
- *SystemBuild BlockScript User Guide*—Describes how to write instructions in the BlockScript language. BlockScript is used in SystemBuild with both BlockScript blocks and BetterStateChart blocks.
- *SystemBuild FuzzyLogic Block*—Describes how to use the SystemBuild Fuzzy Logic Block to obtain 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 HyperBuild User 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 Interactive Animation User Guide*—Describes how to create and link Interactive Animation pictures to your model and how to use these pictures for model control and results display at run time. This module is usually considered part of SystemBuild and is now supplied with the standard RealSim package.
- *SystemBuild Neural Network Module*—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 through AutoCode.
- *State Transition Diagram Block*—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 Books

- *AutoCode User Guide*—Describes how to use AutoCode to generate code from a SystemBuild block diagram.
- *AutoCode Reference*—Supplements the *AutoCode User Guide* and provides additional reference information.
- *DocumentIt User Guide*—Describes how to use DocumentIt to generate design documentation from a SystemBuild block diagram.
- *Template Programming Language*—Describes how to write templates using the Template Programming Language (TPL) for AutoCode and DocumentIt.

Using Help

The *MATRIXx Help* is implemented as an HTML fileset linked to MATRIXx using Netscape NetHelp Help engine. For fully functional *MATRIXx Help*, you must use a browser whichsupports JavaScript (Internet Explorer or Netscape). Other browsers can be used to view the help fileset, but they do not interface with MATRIXx to provide context-sensitive help.

The following topics are covered here:

- *Starting the Online Help*
- *Common Startup Questions*
- *Using the MATRIXx Help Window*
- *Navigating Between Topics*
- *Finding Specific Help Topics*
- *Using Help Examples*
- *Using Context-Sensitive Help*

Starting the Online Help

You can open the general MATRIXx Help window in these ways:

- From the command area of the **Xmath Commands** window, type `help`. If you know the name of the command, function, or topic, specify it after the Help command; for example, `help sba`.
- From the **Xmath Commands** window, select **Help»Topics**.
- **(Windows)** Xmath starts a new Internet Explorer and launches the *MATRIXx Help* window directly. **(UNIX)** If Netscape is not running, Xmath launches Netscape and then the *MATRIXx Help* window is spawned from the Netscape session.
- You also can launch the MATRIXx Help window independent of Xmath, assuming the MATRIXx environment variables are properly set and Netscape is on your path. From the operating system command line, type mtxhelp

Note **(Windows)** The Help has its own shortcut on the **Start** menu.

Common Startup Questions

Why Does this Help Topic Look Funny?

In the *MATRIXx Help*, only text for examples and syntax (where returns must be preserved) have a predetermined font (`monospace`). For body text Netscape uses default fonts, or whatever you have selected in the **Netscape File** menu **Preferences** dialog. The Netscape default font is Times. In rare cases, your machine may not have this font loaded, or it may not have the font in the size you have selected, resulting in pages with letters mysteriously missing. Try choosing another font size or another font.

It is best if you have Netscape and other color-intensive applications closed before you start MATRIXx. This allows your application to get the colors it needs. It also means that Netscape may not be able to grab the standard colors the help uses. This means hypertext links may not be blue, and so forth. In general, the bad colors do not impair the functionality. For the best results, complete the following steps.

1. Close all applications.
2. Start MATRIXx, but do not launch the Help yet.
3. Start Netscape, and then start *MATRIXx Help* from the Xmath Commands window.

Where Is this File?

The topic filename and relative location are shown at the bottom of every file, just above the copyright, for example, `$XMATH/help/masterIX.doc.html`. You can find the value of the environment variable `$XMATH` from the command area of the **Xmath Commands** window. **(UNIX)** Type `oscmd("env")`. **(Windows)** Type `oscmd("set")`. Locate `XMATH` among the environment variables displayed.

Using the MATRIXx Help Window

Help Window Layout

The MATRIXx Help window includes elements common to many browsers:

- *Frames*
- *Buttons*

Frames

The MATRIXx Help window uses three frames:

- The left frame contains the topics hierarchy. All blue text entries are links to *MATRIXx Help* topics. Topics usually contain lists of pertinent functions and commands.
- The lower right frame displays the current topic. For example, click a subject in the topics hierarchy, and it is displayed in the topic frame.
- The upper right frame displays the letters of the alphabet, and the Symbols topic. These are entry points into the help index. For example, click **D** to display an alphabetized list of topics that start with D.

You can use scrollbars or the **Bottom** and **Top** buttons to navigate within frames. To change the width or height of a frame, click the dividing line between two frames and drag in the direction you want the frame to enlarge/decrease. Alternatively, change the size of the entire window using a method appropriate to your window manager.

Buttons

The **Help** window frame has **Backwards**, **Forward**, and **Exit** buttons. The backwards and forwards arrows move you to the last link visited, or forward in the viewing history to a link you've previously visited. You also can go forward and back from the **Netscape Quick Access** menu. Right-click anywhere within the **Help** window to raise this menu.

Navigating Between Topics

By default, blue text in the *MATRIXx Help* can be used to jump to a related topic. To jump, click the text.

The **Prev** and **Next** buttons, when shown, take you to the previous/next file in the fileset. Because topics are cross-linked, this is not necessarily the next file in the listing that the topic hierarchy shows, because topics are cross-linked.

Topic Groupings

Some topic categories have been grouped into single large topics:

- AutoCode
- Model Reduction Module
- Optimization Module

- Programmable GUI
- Robust Control Module
- RVE
- Simulation
- SystemBuild Utilities

You can access these topics using index and cross reference jumps in the normal way, however:

- **Top** and **Bottom** links lead to the first and last files in the category, not the top or bottom of the individual Help.
- Scroll up or down to navigate through the fileset topics; **Prev** and **Next** are not available.
- When you print a combined fileset, the entire topic category is printed.

Finding Specific Help Topics

- If you know the name of the topic you want to view, go to the command area of the **Xmath Commands** window and type `help`, followed by the name of a command, function, or block. Abbreviation is supported as long as enough characters are supplied to guarantee a unique response. For example,

```
help plot     # raise the help on plot
help algeb    # raise the help on the AlgebraicExpression block
```

- Go to the topics hierarchy and select a general topic. Follow links through the topic hierarchy listings until you find a topic of interest.
- Use the Help Index. Click a letter of the alphabet in the upper right Frame to display all entries that start with that letter. The **Prev** and **Next** buttons to jump to the next alphabetized category.

Using Help Examples

There is a special convention for MATRIXx commands and functions. If a command or function name at the top of the help category is blue, there is a link from the topic name directly to the Examples portion of the help. In some cases, examples are distributed through the topic. This is usually done to provide related discussion, or keyword category grouping.

Command or function Help usually include help examples in the form of Xmath or SBA commands and functions. To test the help examples, use your window manager's copy and paste conventions to copy the example text into the command area of the Xmath Commands window, and then

press <Return>. Usually this involves highlighting some text, using mouse-clicks or menu options to copy the text, and then pasting the text to the command area of the Xmath Commands window.

Note Loss of highlighting is a frequent side-effect of color map conflicts. Netscape's default highlight color in most environments is a pale yellow. If this color is not available it may seem that highlighting is broken when you attempt to highlight text on a white page. In most cases you can assume the highlighting is taking place and carry out your copy and paste operation successfully.

Some help examples consist of Xmath command and function definitions used in conjunction with calls issued from the command area. To test examples where Xmath commands and functions are defined:

- Use a text editor to create a new Xmath command or function file.
- Copy and save the example command or function definition script to the file.
- Name the file `commandName.msc` or `functionName.msf` and save it to a folder included in the lookup path.
- Execute the commands that call the newly defined command or function, by copying them into the Xmath command area.

Using Context-Sensitive Help

The *MATRIXx Help* facility is context sensitive. Clicking the **Help** button or the **?** toolbar button from a specific window or dialog launches Internet Explorer to provide you with information specific to that topic.

For example, to get help on the SuperBlock Editor, go to the Editor and select **Help»Topics**, or click the **?** toolbar button. The Help for the SuperBlock Editor appears. To get help on a given block, open its block dialog—select the block, and then press <Return>—and click the **Help** button. The Help for the active block is displayed.

MATRIXx Release Information

For current MATRIXx release information, refer to the *MATRIXx 7.1 Release Notes*:

- Online books—Click **Release Notes** on the document MATRIXx Bookshelf.
- Online Help—Select the topic **Release Info»Release Notes**.

3

Xmath

Xmath provides tools for mathematical analysis. You can create, store, plot, and explore data in Xmath. You can define your own functions, commands, and objects, and also link in externally compiled C or Fortran code. Xmath is the controlling environment for SystemBuild and related products. This chapter gives an overview of Xmath functionality.

Introduction to Xmath

The following sections introduce the Xmath tools and capabilities:

- *MathScript*
- *Data Handling*
- *Numerical Analysis*

MathScript

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. Refer to the *MathScript Programming, Programmable GUI* topic in the Xmath Help for instructions for using and building convenient user interfaces.

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, refer to the *Xmath User Guide*.

Data Handling

MathScript 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 functions, including mathematical functions and filter design functions. Xmath also provides two plotting facilities, one with an interactive graphics display, and the other integrated with a programmable GUI facility.

Optional Xmath modules contain commands and functions to address special uses. The modules are documented in the Help and each has an online manual. Discussions of theory and examples are provided in the manuals. Refer to Chapter 2, *MATRIXx Publications, Help, and Customer Support*, for a summary of available documentation.

Getting Started in Xmath

This section assumes that Xmath has been properly installed and configured. Refer to the *MATRIXx System Administrator Guide*, (*Windows*) for installation details.

Note Many of the operations described in this guide can be accomplished by alternative methods and shortcuts. To simplify the presentation, only one method is specified in most cases.

Directories Defined by Environment Variables

The MATRIXx product line is installed in a directory known as MTXHOME. The installation process modifies Xmath startup scripts and provides the location of MTXHOME as an environment variable (`%MTXHOME%`) that is known *only* within the MATRIXx environment. Three additional environment variables, also known only within the MATRIXx environment, define three subdirectories of MTXHOME: `%XMATH%`, `%CASE%`, and `%SYSBLD%`.

The MATRIXx environment variables are recognized only in the Xmath command area. If you need to use them elsewhere (for example, in the operating system), you must specify the full pathname. In such cases, we indicate the file location with italics: *MTXHOME*, *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 `%MTXHOME%`, `%XMATH%`, `%CASE%`, or `%SYSBLD%`.

Note The environment variables discussed within this section are subject to change, and therefore, should not be used in scripts.

Setting Your Display Colors

Xmath plots require that the Windows display driver be set to display a minimum of 256 colors.

Default colors for your display windows, borders, and other screen components are established through the **Appearance** tab in the **Settings»Control Panel»Display** selection, just as in other Windows applications.

Starting Xmath

To start and run Xmath, select **Start»Programs»National Instruments»MATRIXx mx_71.1»Xmath**.

Xmath starts and the Xmath Commands window is displayed as shown in Figure 3-1.

The Xmath Commands Window

When you invoke Xmath, the Xmath Commands window is displayed (refer to Figure 3-1).

You type input in the command area. Output, environment status, and error messages are displayed in the log area above.

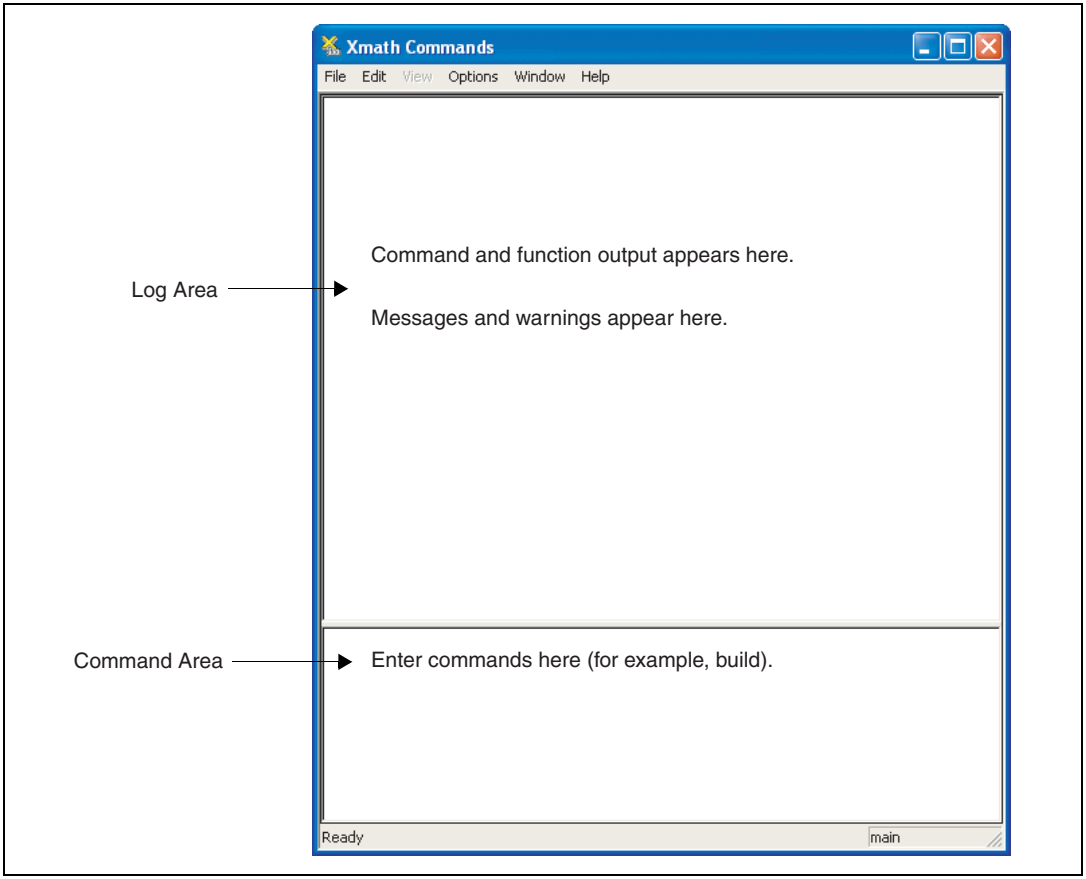


Figure 3-1. Xmath Commands Window

Menu Choices

Menu choices available in the Xmath Commands window are:

- The **Edit** menu displays commands for editing the Xmath command area. Conventional windows selections are Undo, Cut, Copy, and Paste. Xmath adds Clear Log, Clear Command, and Send Command.

- The **View** menu is reserved for expansion.
- The **Options** menu includes a **Font** menu item that allows you to set any TrueType font installed on your Windows system to be used in your Xmath displays. It also has a **Format** menu item for selecting the format of numeric values displayed in the Xmath log window.

Note Xmath provides no capability for saving a font selection between sessions.

- The **Window** menu lets you bring up the Graphics, Palette or Debugger window, or invoke SystemBuild.

Command Window Execution

The Commands window has two command modes: *single-line* and *multiline*.

The default mode is single-line. After typing a MathScript instruction, you press the <Return> key, and the instruction is executed by Xmath.

The key sequence <Shift-Return> toggles the command mode. In multiline mode, the <Return> key adds a new line rather than sending the instruction to Xmath.

For example:

`for i=1:10`	Press <Shift-Return> at the end of the first line to switch to multiline command mode. Press <Return> to add a new line.
`i?`	Press <Return> at the end of the second line to add a new line.
`endfor`	Press <Shift-Return> at the end of the last line to switch to single-line command mode. Press <Return> to send the multiline for loop to Xmath.

Note The previous text is not valid for cutting and pasting from online format into Xmath.

Running Demos

For a tutorial of Xmath's basic features, refer to the *Xmath Jumpstart* section of the *Xmath User Guide*. For an online demo, click in the command area, and then type demo.

You can choose from several example scripts. As a script executes, explanatory text is displayed in the log area; a **Pause** dialog box pauses

the script to give you time to read the text or view a plot. Move the **Xmath Pause** dialog box so that it does not obscure the Commands window.

Accessing Online Help

Xmath has a comprehensive online Help system.

Note *Online Help* requires Internet Explorer. (It does *not* work with Netscape.)

To access *online Help*, select **Help»Topic** from the Xmath Commands window. Two Help windows appear.

To access a topic, click the title in the Topics Hierarchy pane of the window shown in Figure 3-2. The topic (MathScript in the figure) appears in the text window. Use the scroll bar and various buttons and links to navigate from one topic to another.

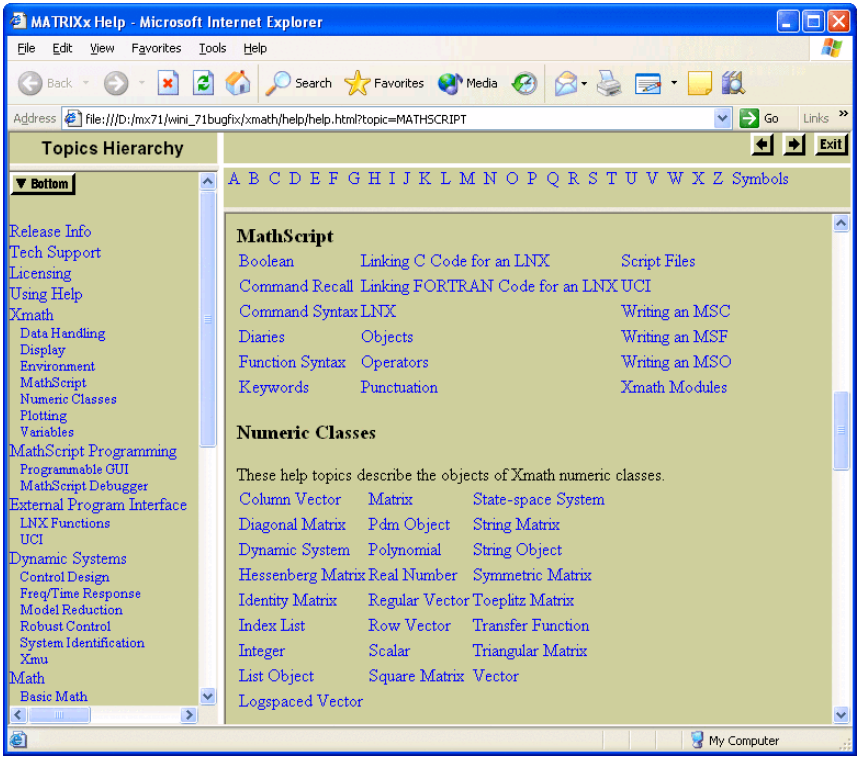


Figure 3-2. MATRIXx Help Window Showing Topics Hierarchy

Figure 3-3 shows the second online Help window with a portion of the *online Help* index on display. Double-click any item to bring up its online Help topic.

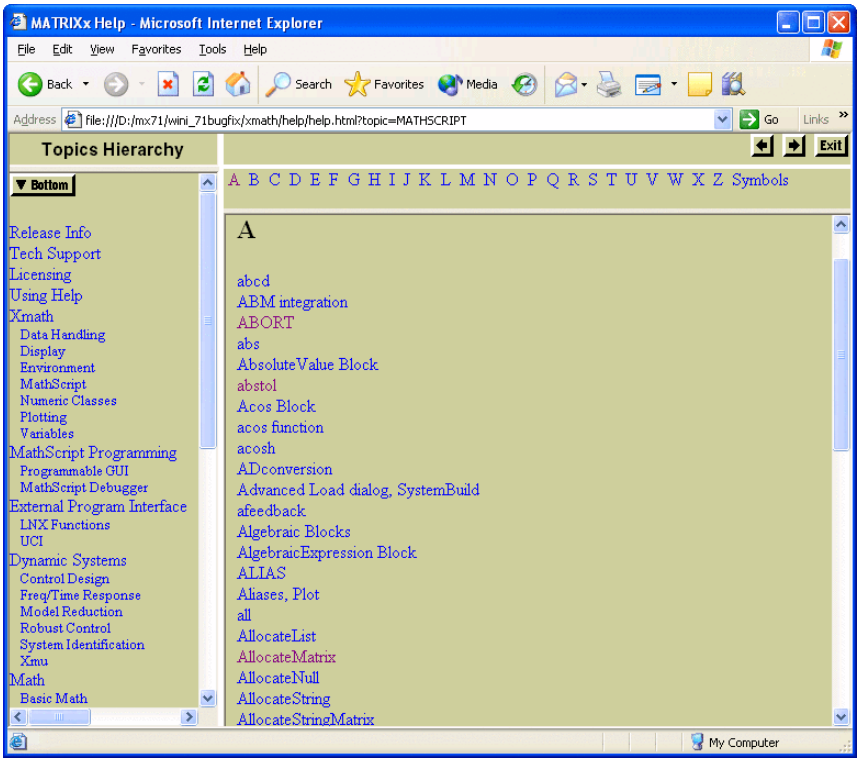


Figure 3-3. MATRIXx Help Window Showing Portion of Index

You can also get the index for any particular letter by clicking that letter at the top of the online Help window shown in Figure 3-3.

Stopping Xmath

- To exit from Xmath type `quit` in the command area or select **File»Exit** from the **Commands** window menu bar. Xmath prompts you to save the workspace.
- To stop an Xmath operation, press <Ctrl-Break>. Notice that <Ctrl-Break> cannot interrupt a process 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), press <Ctrl-Alt-Delete> to display the Task Manager; then select **Xmath Commands** and click the **End Task** button.

Performing Sample Xmath Tasks

This section introduces some basic and advanced Xmath features, including MathScript. If you have never used a mathematical analysis package, become familiar with the demos described in the *Running Demos* section before continuing.

In the following sections, instructions that you enter in the commands area are shown in `monospace`. A description of the instruction, if applicable, and related help topics are found following the Xmath comment symbol (#), to the right of each input.

Try some of the following examples. You do not need to type comments. If you are accessing this document online, the instructions can be copied and pasted to the command area for execution.

Note Command and function names can be shortened to a unique opening substring (as few as four characters).

If Xmath is not running, refer to the *Starting Xmath* section.

Creating Data

The first step in mathematical analysis is usually creating some data. Enter the following MathScript statements to create and 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 as data to a variable:

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

For more information on graph objects, refer to the *Graph Object* help topic by selecting **Xmath»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.
```

Look at the *vector* topic in the help. `vector` is a numeric class. Nonnumeric, or complex, objects are strings or combinations 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 a certain type object and return another type object. 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. In SystemBuild, simulation output is a PDM. 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 information on PDMs, refer to the *pdm* and *PDM object* topics in the *Xmath Help*. For more on the `plot( )` function, refer to the *Xmath User Guide* and the plot topic in the *MATRIXx Help*.

Saving, Loading, and Printing Data

To list the variables you have created so far, type:

```
who
```

Note the sizes. Refer to the *who( )* topic of the *Xmath 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 wild card to save a subset of variables to a different file.

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

Refer to the *save* and *wild cards* topics of the *Xmath 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("dir 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

Use the variable `sys` again:

```
nyquist(sys)?
```

The function `nyquist( )` creates a plot; however, the output of `nyquist( )` is not the graphics object. To save the contents of the Graphics window, use one of the following methods:

- In the Xmath Graphics window, select **File»Bind to Variable**, and then specify the variable name `graph3`
- From the Xmath Commands window command line, type:

  ```
  graph3=plot()
  ```

You now have three graph objects: `graph1`, `graph2`, and `graph3`. You can display them in a manner analogous to other variables:

```
graph1
graph2
graph3
```

Printing Graphs

To print the graph currently displayed in the Graphics window, use one of the following methods:

- In the Xmath Graphics window, select **File»Print** and fill in the resulting dialog box.
- In the Xmath Commands window, enter `hardcopy {color=0}`.

  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. Open **Control Panel»System** and examine the environment variables. If you need additional help, refer to the *Xmath User Guide*.

Use `hardcopy` to save your graphics to a PostScript (`.ps`) file and then submit the file to the printer with a standard command. For example:

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

From a DOS Command window prompt window, type

```
copy file.ps path_to_printer
```

MathScript

MathScript, the language of Xmath, defines statements, constructs, punctuation, functions (MSFs), commands (MSCs), and objects (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 contents as shown in Example 3-1.

Example 3-1 cdown.msf

```
#{
     cdown counts from the integer input down to 1
     and displays the square root of each count.
     cdown outputs a vector of the square roots

}#
function [out]=cdown(c)

if is(c,{integer, min=1}) then
    out=[];
    display "************"
    for i=[c:-1:1]
      display "SQRT(" + string(i) + ") = " +
string(sqrt(i))
      out=[out,sqrt(i)];
    endfor
else
    error("cdown accepts positive integers only","C",c)
endif

endfunction
```

Save your file in the current directory for Xmath (or any directory in the lookup path), and return to Xmath. To see your current lookup path:

```
show path
```

To add a new directory to the path:

```
set path "directory path specification"
```

Call `cdown( )` with valid and invalid inputs:

```
cdown(5)
```

```
cdown(-5)
```

```
cdown("what, me worry?")
```

The Xmath Debugger

The Xmath debugger helps you to debug MathScripts you write—MSFs, MSCs, and MSOs. You can control the Xmath debugger interactively from the Debugger window as shown in Figure 3-4, or from the Command area in the Commands window. This section describes these interfaces.

Starting the Debugger

Debug mode starts under three circumstances:

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

  `debug` *`script_name`*

  The debugger opens automatically on the first executable line in the script.
- A script contains a syntax error, for example, an error in punctuation, such as a missing brace: `plot(a,{xlab="A missing brace").`
- A script 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 operation + does not accept an integer and a string:

  ```
  x=5 + "hello"
  ```

Normally, when an error is detected in a script, Xmath automatically displays the error in the debugger window and sets the interpreter to debugging mode. To prevent the interpreter from going into debugging mode, execute the command:

```
set debugonerror off
```

Xmath 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**, show in Figure 3-2, or type the equivalent commands in the commands area to preform debugging tasks.

In the command window, 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, try the debugger:

```
cdown(2)
```

Notice the difference in the status bar at the bottom of the Xmath window. You are now in debug mode, and the function that you are debugging shows on the right side. 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`. Notice that you travel through the for loop two times, and the debugger notifies you 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, refer to the *MathScript Programming» MathScript Debugger* topic in the *Xmath Help*.

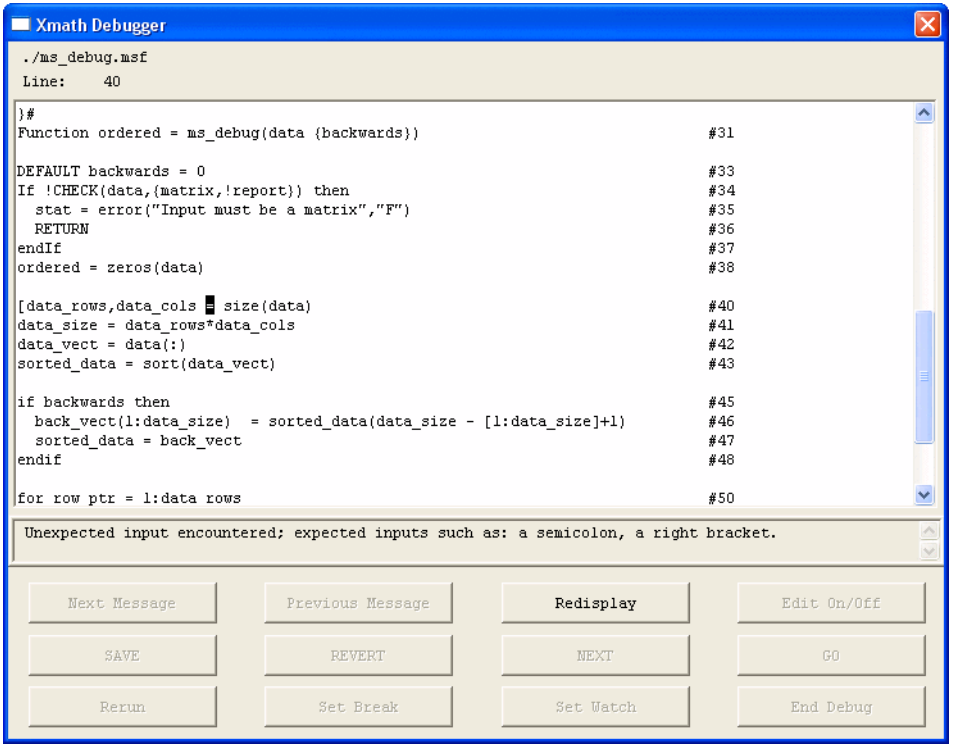


Figure 3-4. Xmath Debugger Window

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 a MathScript, 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. However, one limitation of some editors is that they do not support line numbers. You can use the editor's find feature to locate the error by copying the line containing the error from the debugger to the search field. To avoid this inconvenience, you can use an ASCII editing program that supports line numbering.

Xmath Plotting

Xmath provides a choice of three basic plotting functions:

- The `plot( )` function provides an easy to learn syntax for 2D and 3D plotting in an interactive graphics window. For a quick, interactive look at your data, and for 3D plotting, the `plot( )` function is a good choice.
- The `uiPlot( )` function provides full featured 2D plotting integrated with an extensive programmable GUI facility. If you want more control over the formatting of your 2D graphics, or the ability to integrate plots with your own interactive Xmath PGUI tools, then `uiPlot( )` has the power you need.
- The `plot2d( )` function provides quick access to advanced formatting features of the `uiPlot` function, while avoiding the cost of constructing a programmable GUI tool. Use `plot2d( )` to obtain highly-customized 2D graphics without writing a PGUI tool.

Exploring Additional Topics

There are many more topics to explore in Xmath. For additional information, refer to the *MATRIXx Help* and the *Xmath User Guide*.

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 the SuperBlock editor to build block diagram models, and then test them with SystemBuild Simulator and additional analysis tools. This chapter presents an overview of SystemBuild, as well as a tutorial to guide you through some of the most common SystemBuild tasks.

Additional information about SystemBuild is available:

- The *SystemBuild User Guide* details use of the SystemBuild SuperBlock Editor and the SystemBuild Simulator. It also contains a comprehensive guide to terms, concepts, and keyboard and mouse actions, as well as several chapters on special topics.
- The extensive SystemBuild block library and other technical reference topics are documented in the *MATRIXx Help*.
- The *SystemBuild Books* section of Chapter 2, *MATRIXx Publications, Help, and Customer Support*, lists additional SystemBuild publications.

Introduction to SystemBuild

This section introduces fundamental SystemBuild concepts, tools, and functions.

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

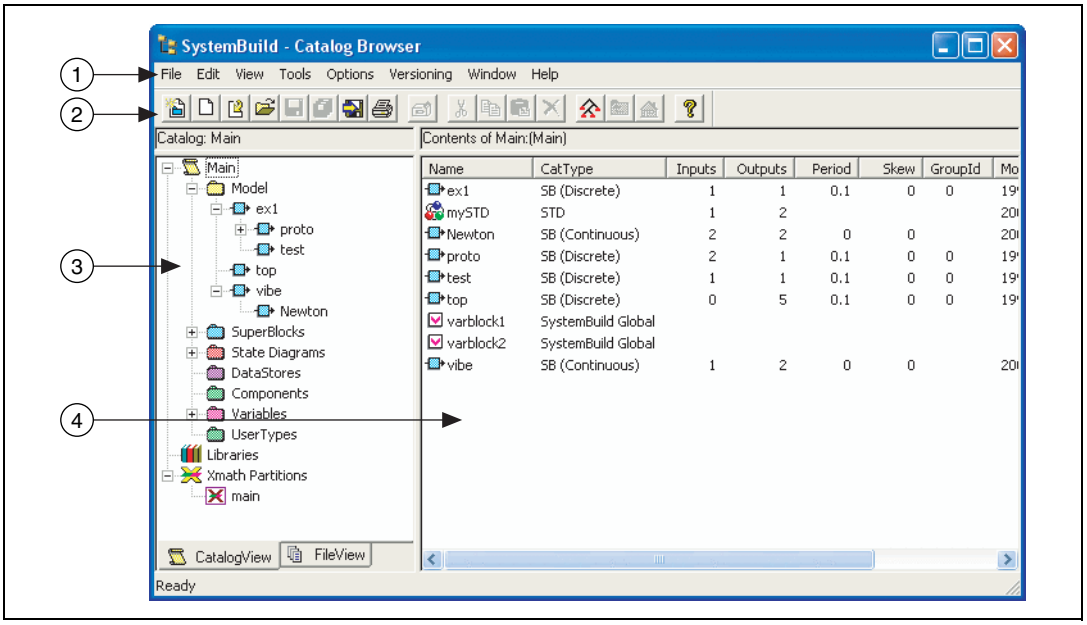
Table 4-1. Definition of Key SystemBuild Terms

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

Catalog Browser

The Catalog Browser is used to manage your SystemBuild models. You use the Catalog Browser to save and load model catalogs composed of SuperBlocks and State Transition Diagrams. It also can be used to view currently loaded SuperBlocks and State Transition Diagrams, create new SuperBlocks and State Transition Diagrams, and to select SuperBlocks and State Transition Diagrams for editing, as well as other functions.

The Catalog Browser, shown in Figure 4-1, contains a menu bar and a toolbar with buttons that are shortcuts to menu operations. The main portion of the Catalog Browser is divided into two *panes*. The left pane displays a hierarchical catalog tree of different types of objects (for example, SuperBlocks). The hierarchy 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 in the left pane.



1 Menu Bar	4 Catalog Tree
2 Toolbar	5 Contents List

Figure 4-1. Catalog Browser

SuperBlock Editor

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

SystemBuild supports the use of up to 20 Editor windows simultaneously. Each Editor window can display the contents of one SuperBlock. The **Window** menu available on both the Catalog Browser and the SuperBlock Editor facilitates switching between the Catalog Browser to select a SuperBlock for editing, and an Editor window to do the editing.

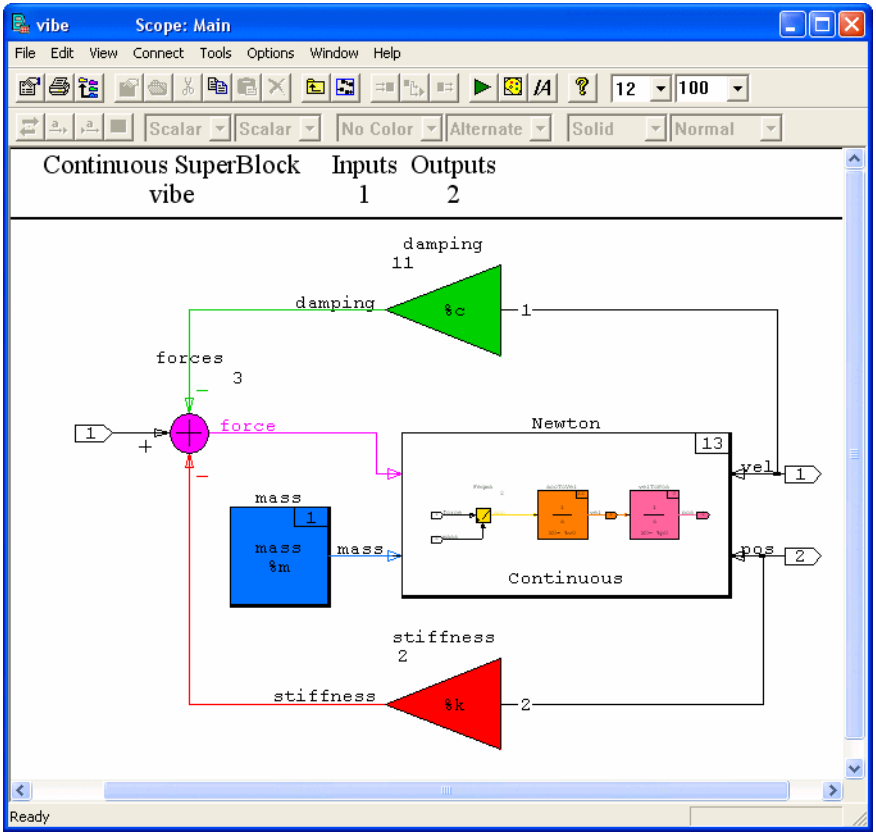


Figure 4-2. SystemBuild Editor

SystemBuild Palette Browser

The SystemBuild Palette Browser, as shown in 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 feature is customizable. The *SystemBuild User Guide* describes several methods for adding custom blocks and custom palettes.

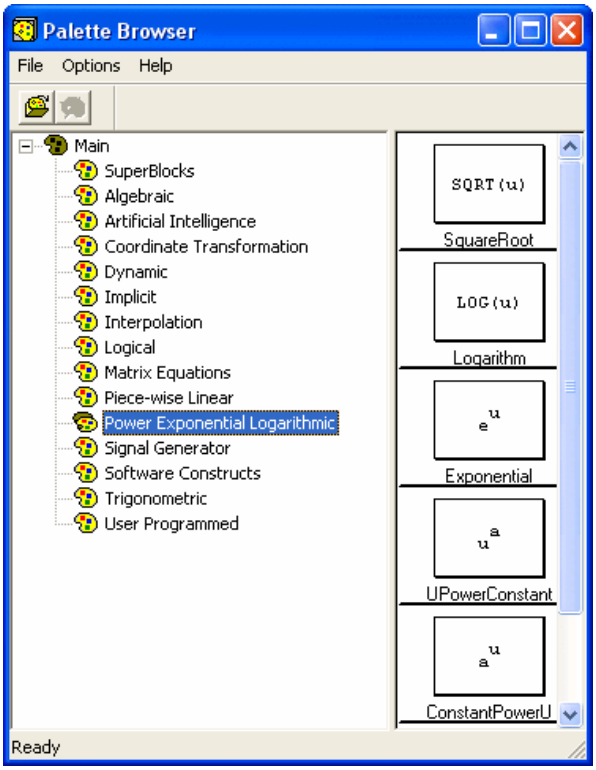


Figure 4-3. Palette Browser

SystemBuild Simulator

The SystemBuild Simulator facilitates 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. Interactive and command based simulation interfaces are provided.

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

Two- and Three-Button Pointing Devices

Workstation users of SystemBuild, accustomed to the three-button pointing device, or mouse, may find themselves on a Windows machine with only a two-button mouse. The Microsoft Windows convention for the two-button mouse is that any operation that you perform using the middle mouse button on the workstation can be performed on a PC by using the right mouse button and pressing the <Ctrl> key. For example, to connect blocks, one of the most common functions of SystemBuild, click the right mouse button in each block while pressing the <Ctrl> key.

Specifying an ASCII Text Editor

You may need a text editor for entering text in some block properties dialog boxes. Each tab that requires text has a drop-down combo box that allows you to select a text editor.

To customize the editor selections available, complete the following steps.

1. Edit the *SYSBLD*\etc\user.ini file.
2. Refer to the *SystemBuild User Guide* for details.

To change the default editor in Windows operating systems, from Windows Explorer, select **Control Panel»System**. In the **System** dialog box, select the **Environment** tab. At the bottom, type EDIT_COMMENT in the **Variable** text field; type the path and filename of your editor in the **Value** text field. Alternatively, you can type the following command in a DOS Command window:

```
set EDIT_COMMAND=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 Notepad, which is a simple, menu-driven ASCII text editor available on every Microsoft Windows system.

SystemBuild Optional Modules

This section describes optional modules available for SystemBuild.

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 with AutoCode. The module supports both training (offline) and learning (real-time) modes of operation.

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.

SystemBuild HyperBuild Module

The HyperBuild Module accelerates SystemBuild simulations. Using the HyperBuild Module, the larger and more complex the model, the more significant the increase in simulation speed. You can invoke the tool from the Xmath command line, for scripting batch builds and simulations, or the SystemBuild menu. The module operates by converting a SystemBuild block diagram into highly-optimized C code, which executes much faster than the original model because the original simulator is interpretive. HyperBuild can be used to generate code for most models, including multirate models and those containing procedures or blocks with state events

SystemBuild Interacitve Animation Module

The SystemBuild Interactive Animation (IA) Module provides the ability to create, edit, and operate displays for interacting with and monitoring SystemBuild-based models. Interactive animated displays may run in conjunction with workstation-based simulations or real-time simulations. The IA Module offers multiple palettes of predefined icons that can be easily customized for your specific application, or you can quickly build your own. IA icons can be integrated with existing SystemBuild models to provide quick feedback for model debugging, or combined with other icons to produce control panels for monitoring and interacting with your models.

SystemBuild Aerospace Libraries Module

The SystemBuild Aerospace Libraries Module provides a set of SystemBuild models which are used extensively in the aerospace industry. Aerospace Library models are pre-built, user-accessible SystemBuild block diagrams designed to be easily incorporated into your aerospace-related block diagrams. These models include an atmospheric model, a spherical gravity and fifth-order gravity model, and a Six-Degree-of-Freedom library containing a Single Rigid Body Dynamics Model which utilizes quaternion algebra. In addition, the included Attitude Geometry Library contains a library of SuperBlocks commonly used in attitude dynamics, including 3D vector and matrix manipulations, quaternion manipulations, cross-product attitude determination equations, and three-axis rotation equations.

Altia Design for SystemBuild

Altia Design for SystemBuild allows you to create new graphical panels with the state-of-the-art Altia Design Editor. This package includes an editor, runtime engine and numerous libraries of components for quickly creating a user interface to SystemBuild simulations. In addition to supplied components, you can modify existing components, import images and create your own custom components in the editor with no programming, and even import existing Interactive Animation Picture files and convert them to Altia Design files. With these features, you can quickly create a user interface prototype for product simulations that will be very similar to your final production user interface.

Starting and Exiting SystemBuild

This section describes how to start and exit SystemBuild.

Note Many of the operations described in this guide can be accomplished by alternative methods and shortcuts. To simplify the presentation, only one method is specified in most cases.

Starting SystemBuild

To start SystemBuild:

1. If Xmath is not currently running, start Xmath as described in the *Starting Xmath* section of Chapter 3, *Xmath*.
2. Type the following in the command area of the Xmath Commands window:

   ```
   build
   ```

   After a short time, SystemBuild is loaded and the Catalog Browser window is displayed.

Exiting SystemBuild

To exit SystemBuild:

1. Select **File»Exit** from the Catalog Browser.
2. SystemBuild asks if you want to save your work before exiting; if you answer yes, the **Save As** dialog box appears.

Basic SystemBuild Tasks

This section describes tasks performed in basic SystemBuild use.

Note Many of the operations described in this guide can be accomplished by alternative methods and shortcuts. To simplify the presentation, only one method is specified in most cases.

Creating a New SuperBlock

The Catalog Browser can be used to create a new top-level SuperBlock. If this window is not activated, click the Catalog Browser's window frame to select it, or select **Window»Catalog Browser** from the Editor.

To create a new top-level SuperBlock and define its properties, complete the following steps.

1. Select **File»New»SuperBlock**.

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

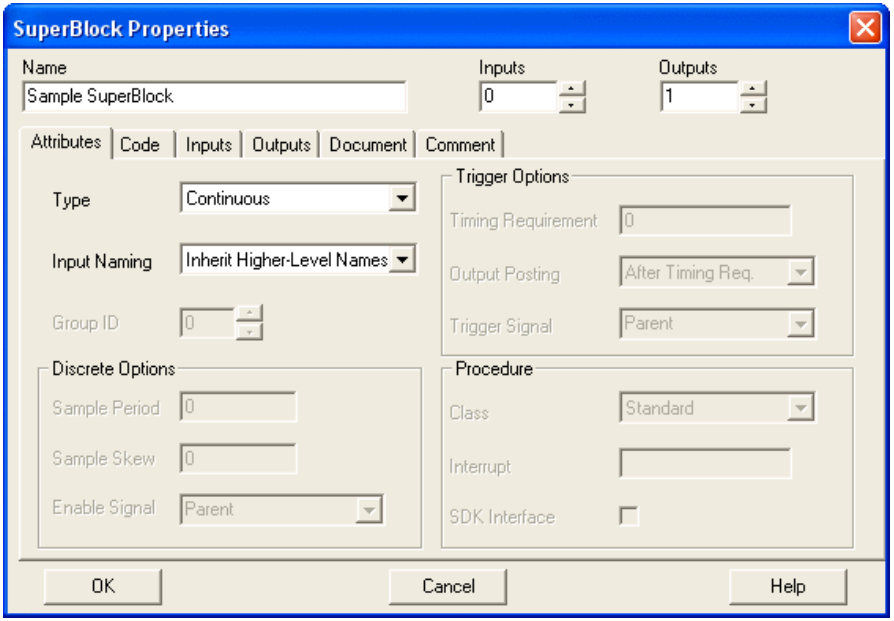


Figure 4-4. SuperBlock Properties Dialog Box for Creating a New SuperBlock

2. With the **SuperBlock Properties** dialog box (refer to Figure 4-4), complete the following steps.

   a. Click the **Name** edit field, and type:

      ```
      Sample SuperBlock
      ```

   b. In the **Outputs** field, set the number of outputs to `1`.

   c. Click **OK** to verify creation of the SuperBlock.

      The SystemBuild Editor (or Editor window) now appears; it contains an 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 in a SuperBlock by dragging it from the Palette Browser into the Editor window, as shown in Figure 4-5.

To create a new block in a SuperBlock, complete the following steps.

1. With your SuperBlock displayed in the Editor window, select **Window»Palette Browser** to open the Palette Browser.

   Reposition your windows so that both the Palette Browser and the Editor are visible.
2. Click the Algebraic palette in the Palette Browser.
3. Move the cursor over the Gain block icon. Press and hold down MB1. (❶)
4. While holding down MB1, drag the cursor into the Editor window. (❷)
5. With the cursor within the Editor window, release MB1 to complete the drag-drop operation. (❸)

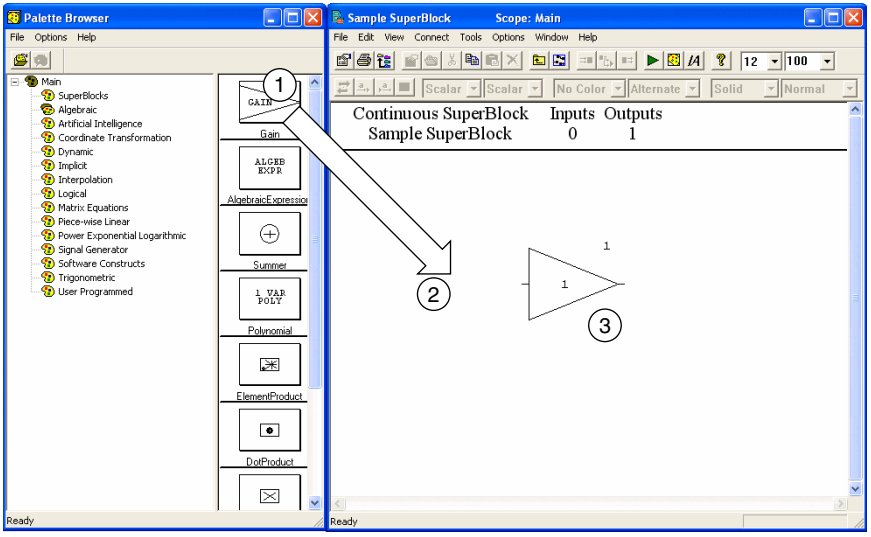


Figure 4-5. Creating a New Block Using the Palette Browser

Loading a Model File

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

To load the `pred_prey` catalog file from the Xmath command area type:

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

`$SYSBLD` is an environment variable that specifies your SystemBuild directory, defined automatically when you start Xmath. Environment variables are recognized only on the Xmath command line. SystemBuild catalogs also can be loaded from the Catalog Browser, but there you must specify the full pathname of the catalog directory.

After the load completes, the Catalog Browser lists the contents of the model (refer to Figure 4-6).

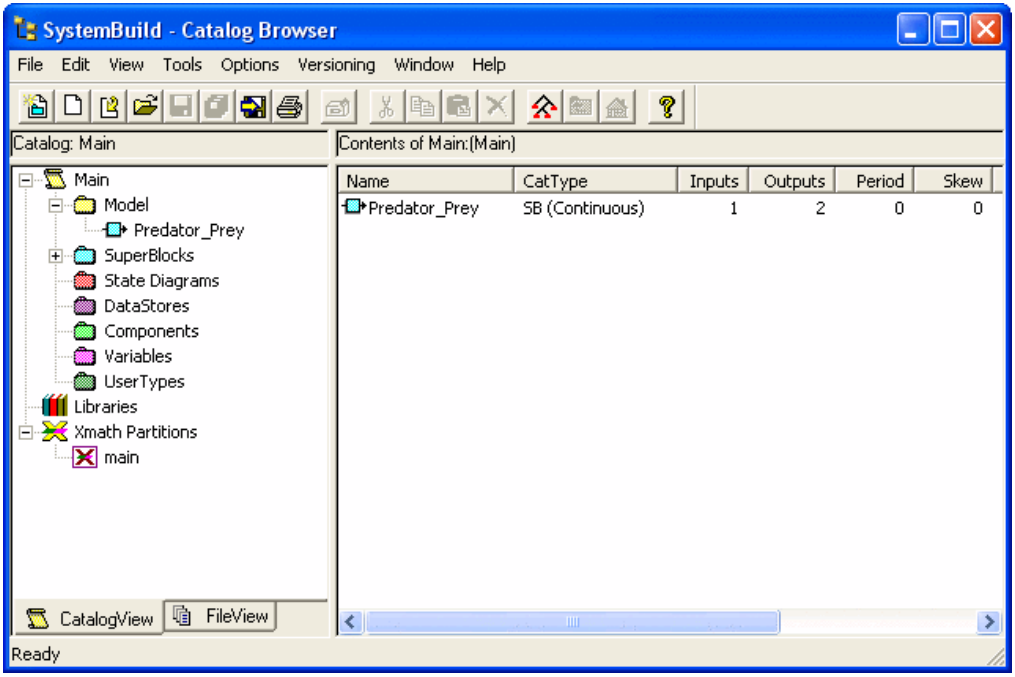


Figure 4-6. Predator-Prey Model Loaded into Catalog Browser

Opening a SuperBlock in the Editor

After loading a model, you can open a SuperBlock in the editor to edit or view it, by completing the following steps:

1. If needed, load the predator-prey model as described in the *Loading a Model File* section.
2. To see a list of all SuperBlocks currently loaded, click in the left pane on the **SuperBlocks** node. A listing of the SuperBlocks appears 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, as shown in Figure 4-7.

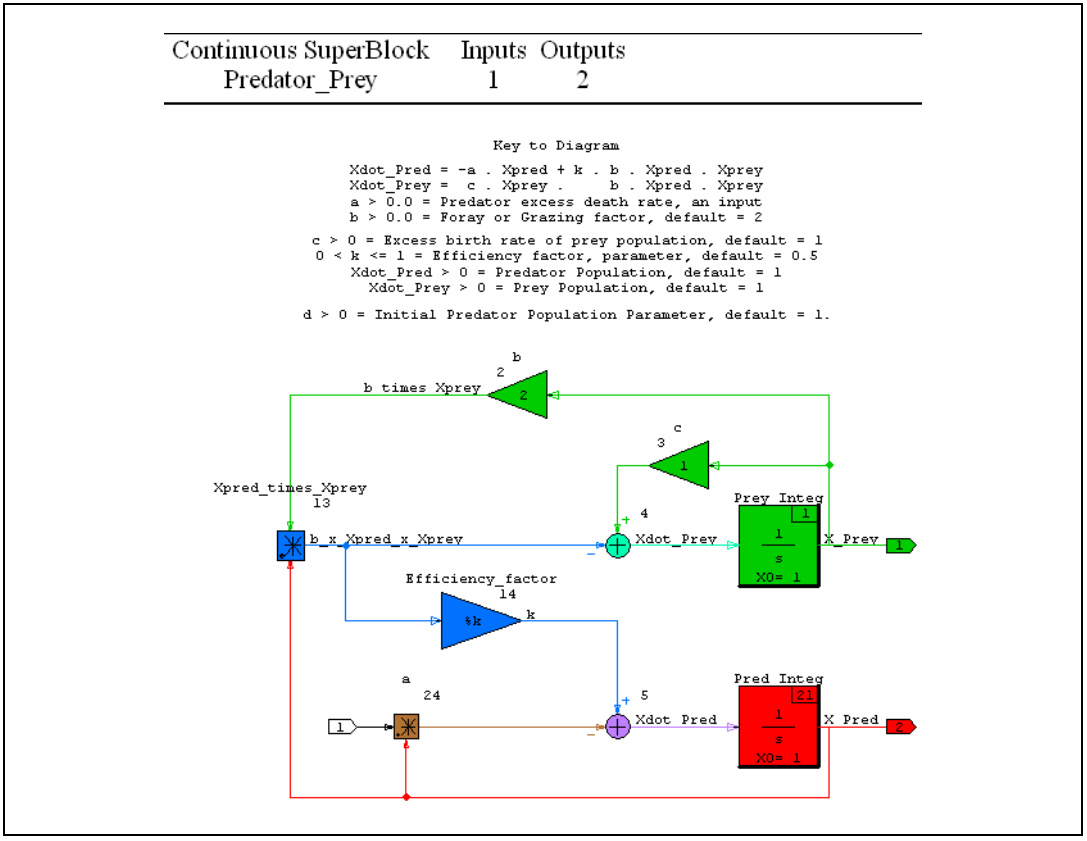


Figure 4-7. Predator_Prey SuperBlock Displayed in Editor Window

Simulating the Model from the Xmath Commands Window

After a model is loaded, you can simulate it. This section describes performing a simulation directly from the Xmath command area and also from the SystemBuild Editor. Load the predator-prey model as described in the *Loading a Model File* section.

To simulate the predator-prey model from the Xmath commands area, complete the following steps:

1. Activate the Xmath window by clicking the Xmath window frame.
2. Click in 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 simulation output plot, which appears in a separate window, is shown in Figure 4-8.

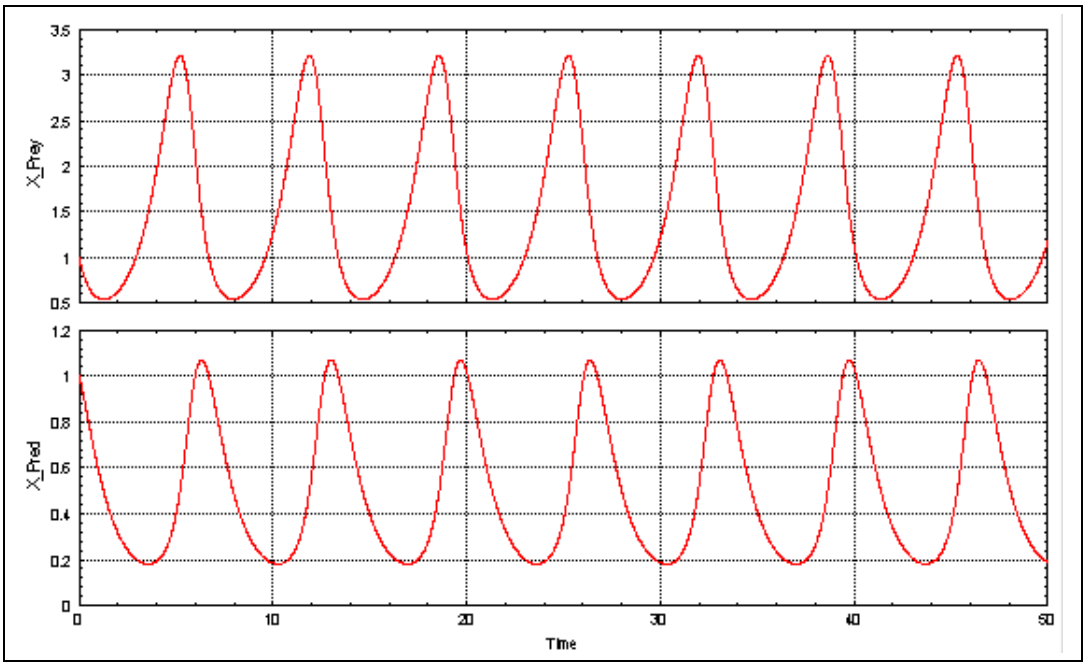


Figure 4-8. Plot of a Predator-Prey Simulation Output

Complete the following steps to simulate the model from the SystemBuild Editor.

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

1. Activate the Xmath window by clicking the Xmath window frame.
2. Click in 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. Activate the Predator_Prey SuperBlock SystemBuild Editor. Refer to the *Opening a SuperBlock in the Editor* section for more information.
6. In the SystemBuild Editor, select **Tools»Simulate** from the pull-down menu.

   The **SystemBuild Simulation Parameters** dialog box appears, as shown in Figure 4-9.

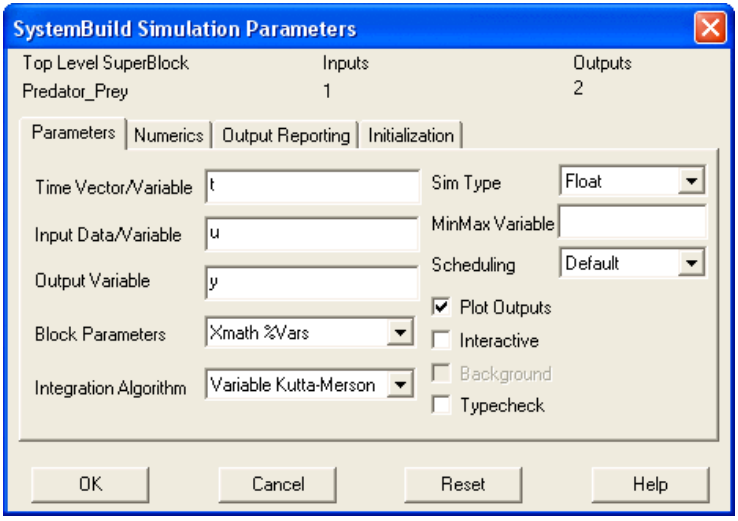


Figure 4-9. SystemBuild Simulation Parameters Dialog Box

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

   You can monitor the log area of the Xmath window as the model is analyzed and simulated. The simulation output plot appears in a separate window, as shown in Figure 4-8.

Deleting a SuperBlock

To delete a SuperBlock, complete the following steps:

1. From the Catalog Browser (either pane, provided the SuperBlock names appear), select a SuperBlock.
2. Select **Edit»Delete**.

Caution Deletion of SuperBlocks *cannot* be undone.

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

Navigating a SuperBlock Hierarchy

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

You need a fresh start for this exercise, complete the following steps:

1. Delete all of the SuperBlocks you may have created, or exit the Catalog Browser (**File»Exit**), and restart SystemBuild.
2. Load a model with a SuperBlock hierarchy:

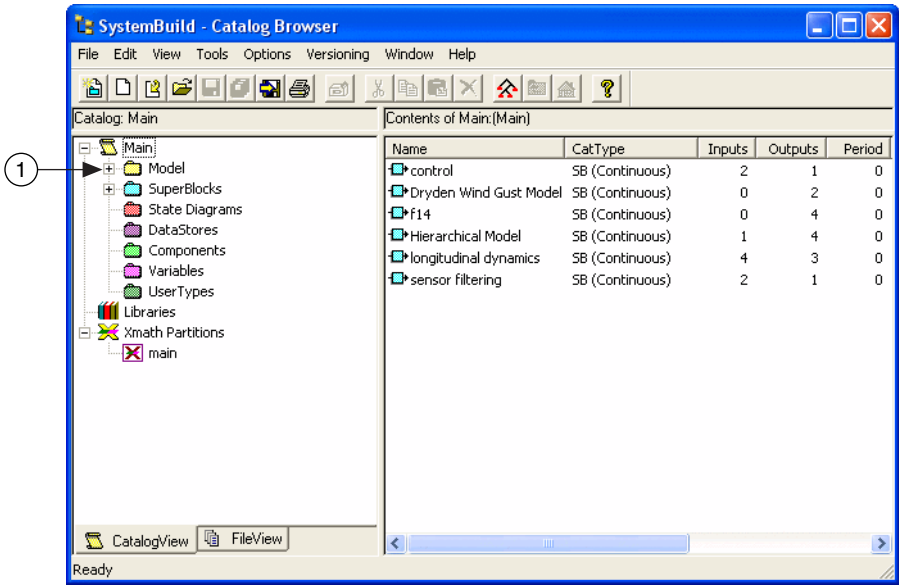
   From the Xmath command area, enter:

   ```
   load "$SYSBLD\demo\f14_demo\f14new.cat";
   ```

Navigating with the Catalog Browser

When you navigate with the Catalog Browser, 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 the types of catalog objects (SuperBlocks, and so forth) in the left pane. Click the **Model** folder icon to see a full list of model objects, including SuperBlocks, in the right pane, as shown in Figure 4-10.



1 Expand/Collapse Indicator

Figure 4-10. Catalog Browser after Loading the f14 Model

Notice the expand/collapse indicator for the Model folder (left pane). Plus (+), indicates that the folder is collapsed, and at least one additional hierarchical layer can be expanded.

To navigate from the Catalog Browser, complete the following steps:

1. Expand the hierarchy of the Model 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, as shown in Figure 4-11.
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.

   To edit another SuperBlock, return to the Catalog Browser, and double-click any SuperBlock in the right pane.

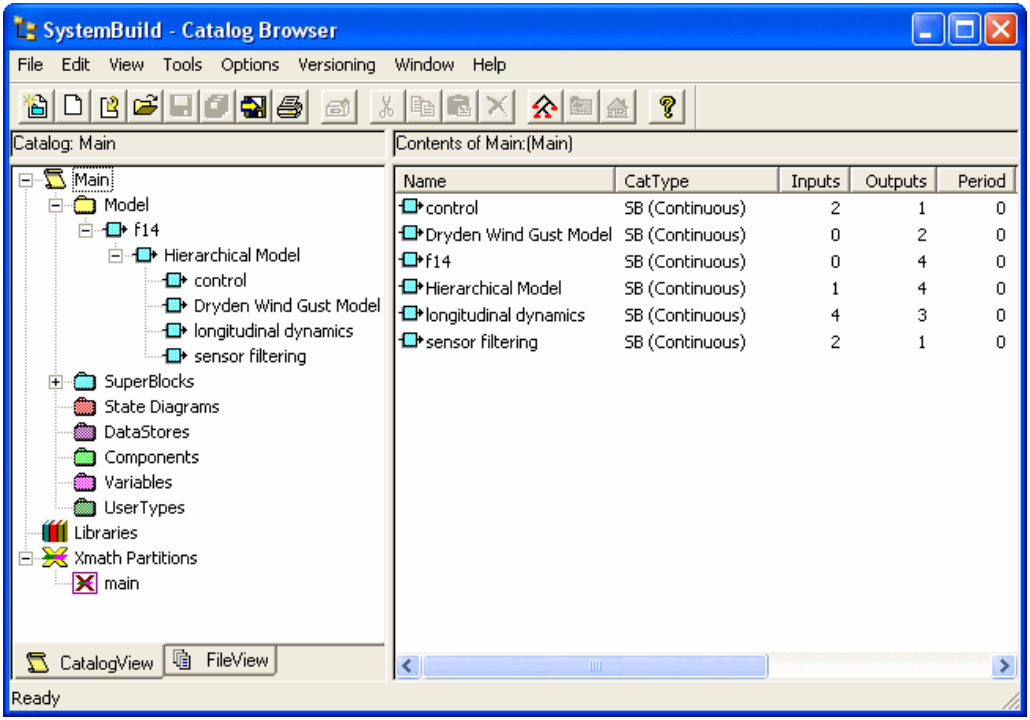


Figure 4-11. Expanded Hierarchy of the F14 Model

Navigating from the Editor Window

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

Complete the following steps to navigate from the Editor window:

1. From the Catalog Browser, open the `sensor filtering` SuperBlock (refer to step 1 through step 3 in the *Navigating with the Catalog Browser* section).
2. With the **Editor** window, select **View»Parent»Hierarchical Model** to view this SuperBlock's parent.

   The **Editor** window now displays the SuperBlock named `Hierarchical Model`.
3. To move down the SuperBlock hierarchy in the Editor window, click the desired SuperBlock icon, then select **Edit»Open**.

   The Editor window now displays the selected SuperBlock.

Note The MATRIXx demo package includes a simulation of the F14 model.

To run the demo simulation, complete the following steps:

1. Enter the `demo` command from the Xmath Commands window.

   The **Xmath Demos** dialog box appears.

2. Enable the **SystemBuild Demos** radio button and click **OK**.

   If the **Save SysBld & Xmath workspace** dialog box appears, enable the **Yes->Save** radio button and click **OK**.

   The **SystemBuild Demos** dialog box appears.

3. Enable the **F14 Jet Simulation** radio button and click **OK**.

   Follow the prompts to run the demo.

Printing from the Editor Window

SystemBuild provides the standard **Windows Print** dialog box for printing from the Editor window on Windows operating systems.

Complete the following steps to print the contents of an **Editor** window, complete the following steps:

1. Select **File»Print**.
2. Printing uses the settings you last made in the **Page Setup** dialog box, also available from the **File** menu. The default printer is specified by the `$PRINTER` environment variable.

SystemBuild Tutorial

This section presents basic procedures used in the design, development, and simulation of SystemBuild block diagrams:

- *Designing a Block Diagram*
- *Creating and Editing a Block Diagram*
- *Simulating a SuperBlock*
- *Encapsulating a SuperBlock*

Note This tutorial is designed to lead you through the construction of a basic SystemBuild block diagram. If you want to examine the completed models, the block diagram constructed in this tutorial can be loaded into the Catalog Browser by executing the following command from the Xmath Commands window.

```
load "$SYSBLD\examples\gs_tutorial\vibe1.cat";
```

The `vibe1` catalog contains a solution of the basic spring-mass damper constructed in the first four sections of the SystemBuild tutorial.

Designing a Block Diagram

To develop a useful block diagram representation of a physical system, you need to know the following:

- The analytic behavior of the physical system components.
- How block diagram elements correspond to physical system components.
- How to use SystemBuild editors and dialog boxes to develop a block diagram.

The Spring-Mass Damper Model

In this tutorial, you develop a block diagram representation for a physical system incorporating a spring-mass damper. The following well-known equation is derived by making standard assumptions about the system behavior:

$$F(t) = m\ddot{x}(t) + c\dot{x}(t) + kx(t) \qquad (4\text{-}1)$$

Equation 4-1 follows the common convention of indicating time derivatives by dots. *F* denotes the external force applied to a mass *m*. The spring introduces a force proportional to and opposite its elongation. The scalar value *k* depends on the spring, and is called its *stiffness*. Motion of the mass is damped by a force proportional to and opposite its velocity. The scalar value *c* is referred to as the *damping* constant.

Equation 4-1 provides a mathematical model that can be represented directly by a SystemBuild block diagram. Alternatively, the system analysis that produced the equation can be used to develop a block diagram. But first you need to know how SystemBuild blocks model physical systems.

SystemBuild Block Basics

SystemBuild block diagrams are composed of interconnected blocks. Most blocks in a block diagram receive input signals and produce output signals. A signal is a scalar value that can vary over time. The process performed by a given block to produce its outputs may depend on user-defined properties.

Blocks are combined in a block diagram by connecting outputs to inputs. A group of interconnected blocks can collectively define a SuperBlock. Most SuperBlocks also have input and output signals connected to one or more of its blocks. A SuperBlock can be simulated by specifying each of its input signals with respect to some time vector. A SuperBlock can be a building block in a higher-level SuperBlock. A SuperBlock hierarchy is defined in this manner.

In this tutorial, you use several basic SystemBuild block types to develop the block diagrams of a SuperBlock hierarchy. The introductory block information included here is intended to motivate that development; it is not a complete description of the capabilities and options of the block types.

Constant Block

A Constant block has no input signal. Its output signal is a constant.

ElementDivision Block

The output signal of an ElementDivision block is input signal #1 divided by input signal #2.

Integrator Block

The output signal of an Integrator block is the integration over time of its input signal. The initial value of the output signal of an Integrator can be defined, and an Integrator can be triggered to reset its output signal to a specified value.

Gain Block

The output signal of a Gain block is its input signal multiplied by a constant.

Summer Block

The output signal of a Summer block is the sum of its input signals. Each input signal has a sign which determines if it added to, or subtracted from the output signal.

ElementProduct Block

The output signal of an ElementProduct block is the product of its input signals.

ZeroCrossing Block

During a simulation, a ZeroCrossing block detects the instant at which its input signal crosses zero. The simulation is recomputed to include this additional time point. The ZeroCrossing block output signal toggles between zero and one at such crossings.

Getting Started on a Design

Identifying the inputs and outputs of the top-level SuperBlock is a good way to start a design. For the spring-mass damper, there is one input—the external force applied to the mass. The outputs of a top-level SuperBlock are the signals you choose to monitor for the purpose of observing the behavior of the modeled system. Position and velocity of the mass are useful choices here.

Next, make a rough plan for modeling and connecting elements of the physical system. For the spring-mass damper, you have force acting on a mass. That determines an acceleration that can be integrated to give velocity and position. The velocity and position can be used to determine the spring and damping components of the force.

In developing a block diagram, try to identify subsystems that can be built independently. Simulating submodels can help to verify that a complex SuperBlock hierarchy simulation is valid. Also, by encapsulating the subsystems in your area of interest, you can optimize reuse in subsequent designs.

In this tutorial, you start building the spring-mass damper by modeling force acting on a mass.

Creating and Editing a Block Diagram

A block diagram is the graphical representation of a SystemBuild model. To create and edit a new block diagram, complete the following steps:

1. Create a SuperBlock.
2. Add blocks to the block diagram of the SuperBlock.
3. Edit block properties.
4. Connect blocks to each other.
5. Connect blocks to SuperBlock inputs and outputs.
6. Save the SuperBlock.

Note Many of the operations described in these procedures can be accomplished by alternative methods and shortcuts. To simplify the instructions, only one method is specified in most cases.

Creating a SuperBlock

To create a new SuperBlock called `vibe`, complete the following steps:

1. Launch Xmath as described in the *Starting Xmath* section of Chapter 3, *Xmath*.
2. In the Xmath command area, type:

   ```
   build
   ```

   After loading, the SystemBuild Catalog Browser is displayed. If necessary, click the bar at the top of the window to make it active and bring it to the front.
3. In the Catalog Browser, create a new SuperBlock by selecting **File»New»SuperBlock**.

   The **SuperBlock Properties** dialog box is displayed. The **Attributes** tab is selected and all properties of the SuperBlock are set to their default values. Figure 4-12 shows the **SuperBlock Properties** dialog box as it appears during step 4.

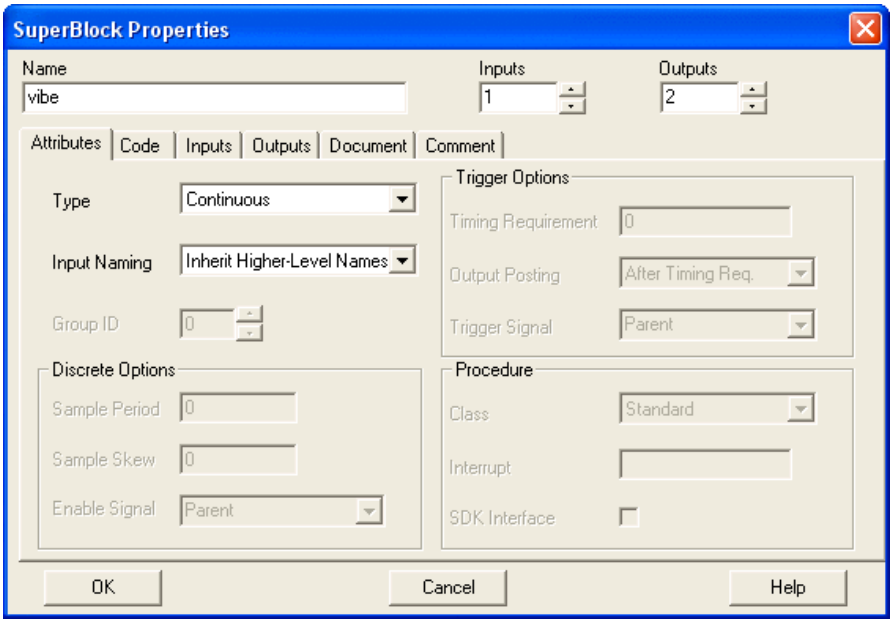


Figure 4-12. SuperBlock Properties Dialog Box

4. Complete the following steps in the **SuperBlock Properties** dialog box:
   a. Click the **Name** field; name the new SuperBlock `vibe`.
   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 **2**.
   e. Click **OK** to accept current values and close the dialog box.

   A SuperBlock Editor is displayed for the new SuperBlock. The information bar at the top contains the type, name, and number of inputs and outputs. The area in which block diagrams are constructed is empty, as shown in Figure 4-13.

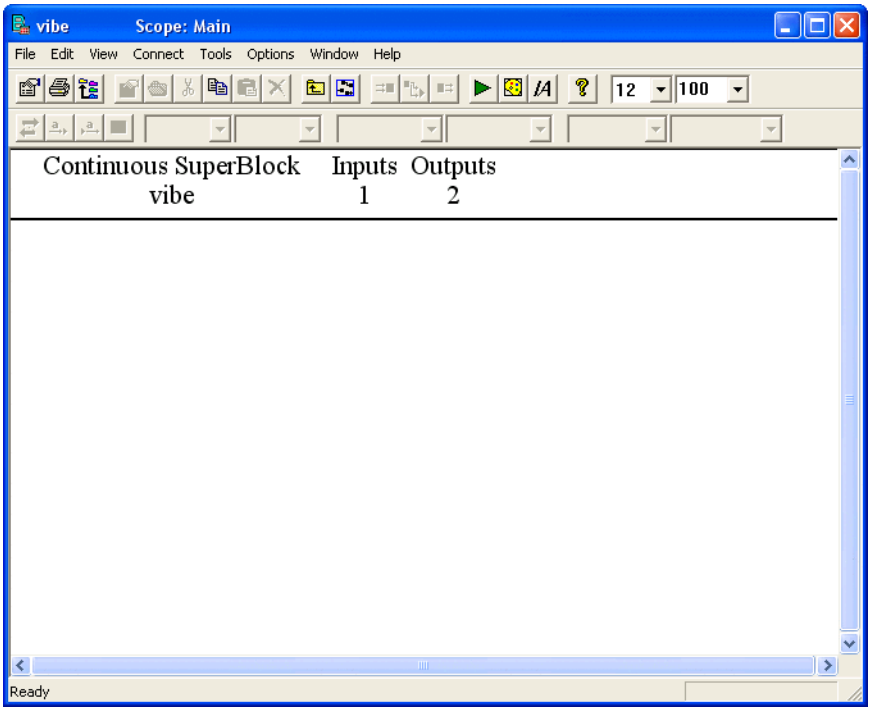


Figure 4-13. SuperBlock Editor (Initial View)

Adding Blocks to the Block Diagram

This section teaches how to add four blocks to the block diagram of the `vibe` SuperBlock: a Constant block, an ElementDivision block, and two Integrator blocks.

Complete the following stesp to add a Constant block to the block diagram:

1. Open the Palette Browser by selecting **Window»Palette Browser**. Position your windows so that the Palette Browser is alongside the Editor.
2. In the Palette Browser, select the **Matrix Equations** palette.
3. With MB1, drag and drop a Constant block from the **Matrix Equations** palette into the Editor.

In the same manner, add an ElementDivision block from the **Algebraic** palette, and two Integrator blocks from the **Dynamic** palette.

Blocks are positioned in a block diagram by dragging with MB1. Position the blocks as in Figure 4-14.

Note The block ID is displayed in the upper right corner of each block. Your ID numbers might not match those in Figure 4-14. To add color to a block, left-click block (ID=12) and change the Display Toolbar Combo Box **NoColor** to **Orange**. Change the color of Block (ID=3) to **Pink**, Block (ID=2) to **Yellow** and Block (ID=1) to **Blue**.

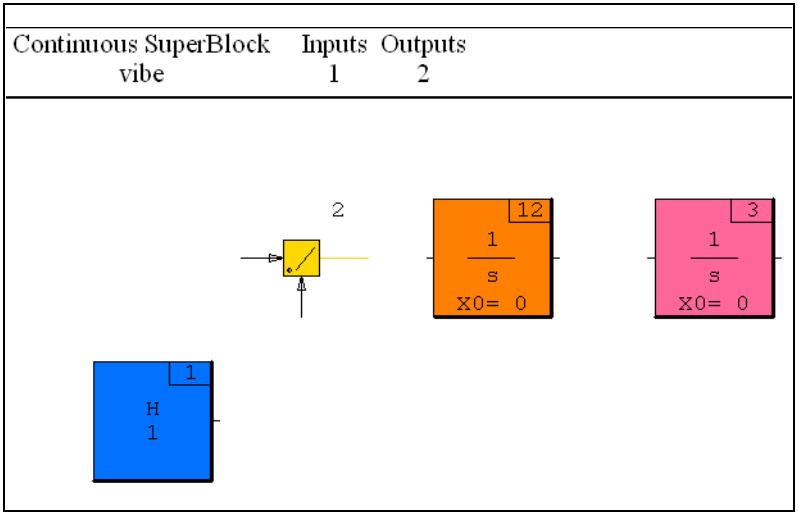


Figure 4-14. Adding Blocks to vibe

Editing Block Properties

Each block in a block diagram contains properties that can be edited to adjust aspects of its performance. This section teaches you to open the **Block Properties** dialog box for each block in `vibe` and edit various properties.

To edit block properties for the Constant block, complete the following steps:

1. In the Editor, move the mouse cursor over the Constant block (the left-most block as shown in Figure 4-14) and press the <Enter> key.

   The **Constant Block** properties dialog box is displayed. The **Parameters** tab is selected and all properties are set to their default values. Figure 4-15 shows the **Constant Block Properties** dialog box as it appears during step 3.

2. Click the **Name** field; name the Constant block `mass`.

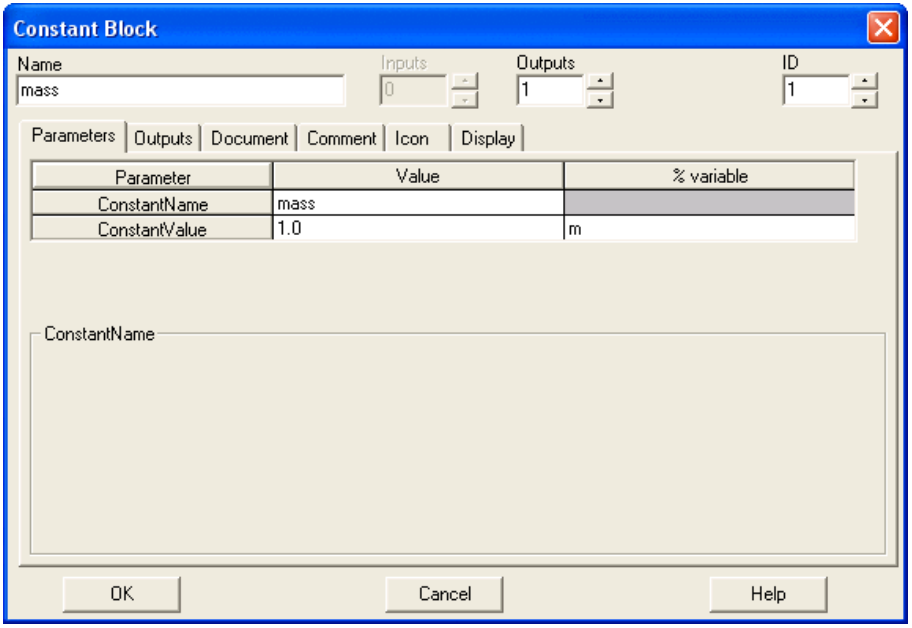


Figure 4-15. Constant Block Properties Dialog Box: Parameters Tab

3. Verify that the **Parameters** tab of the **Block Properties** dialog box is selected, as shown in Figure 4-15.
   a. Locate the field in the **Parameter** table, in the **Value** column, and the **ConstantName** row. Click that field and replace `H` with the name `mass`.
   b. Locate the field in the **Parameter** table, in the **% variable** column, and the **ConstantValue** row. Click that field and type `m`.

Note The *% variable* property allows you to set parameters of your model by assigning values to corresponding variables in the Xmath workspace. Before simulating this model, you set the mass parameter in the Xmath command window by typing an assignment to the Xmath variable `m` (refer to step 3 of the *Simulating a SuperBlock* section). In this way you can simulate the model with different parameter settings without editing the block diagram.

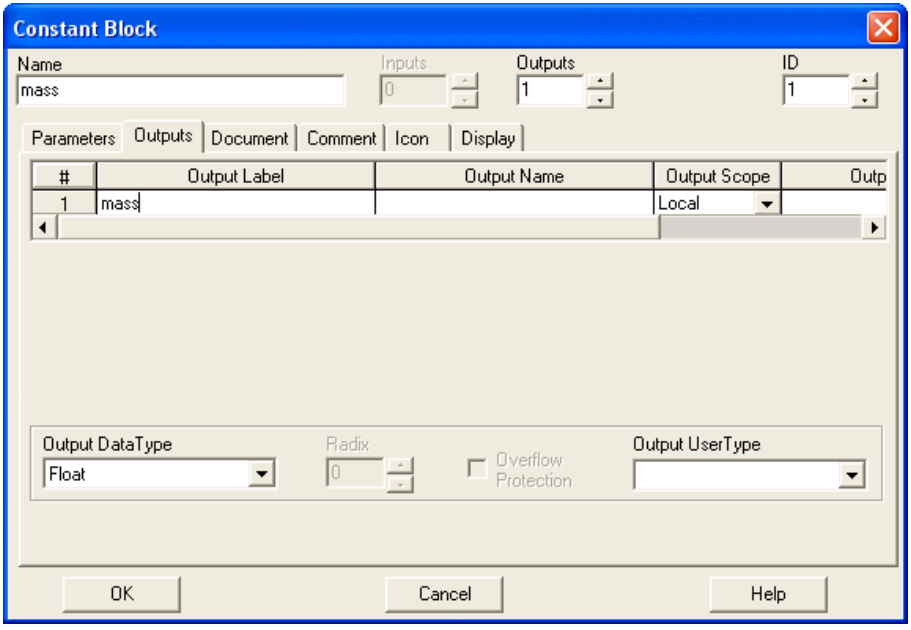


Figure 4-16. Constant Block Properties Dialog Box: Outputs Tab

4. Click the **Outputs** tab. (refer to Figure 4-16).

   Locate the field in the **Outputs** table, in the **Output Label** column, and row **1**. Click that field and type `mass`.
5. Click the **Display** tab (refer to Figure 4-17).

   Enable the **Show Output Labels** checkbox.

Note When the **Show Output Labels** checkbox is enabled for a block, the block's output labels are displayed on its output connectors.

6. Click **OK** to accept current values and close the dialog box.

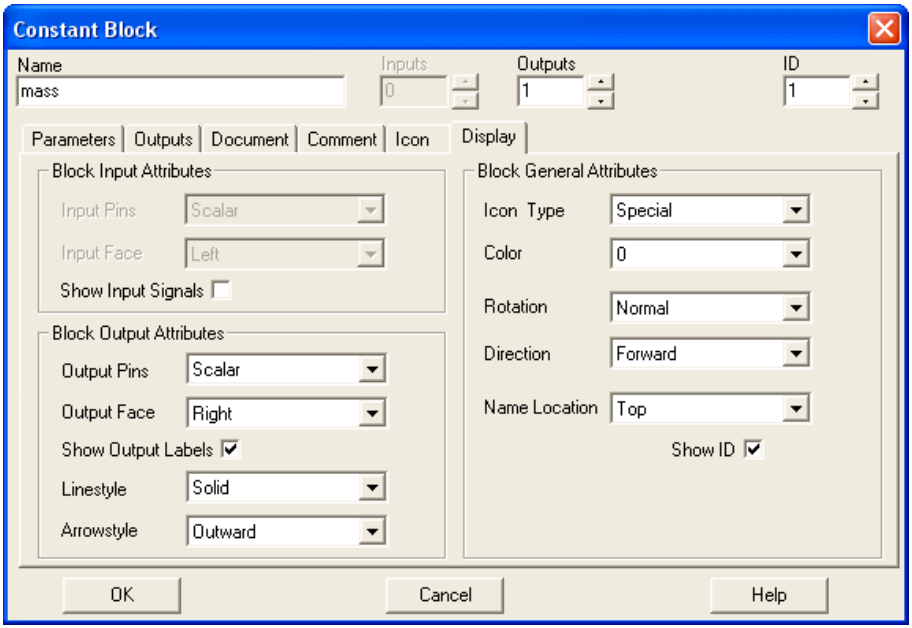


Figure 4-17. Constant Block Properties Dialog Box: Display Tab

Complete the follow instructions to edit block properties for the remaining blocks in `vibe`. You give each block a name, label its outputs, and enable the **Show Output Labels** checkbox. For additional detail, refer to steps 2, 4, and 5 for editing properties in a Constant block.

You also set additional properties as noted in the instructions.

To edit block properties for the ElementDivision block, complete the following steps:

1. In the Editor, move the cursor over the ElementDivision block (second block from the left as shown in Figure 4-14) and press the <Enter> key.

   The **ElementDivision Block** properties dialog box is displayed. The **Inputs** tab is selected and all properties are set to their default values. Figure 4-18 shows the **ElementDivision Block** properties dialog box as it appears during step 3.

2. Name the block `Feqma`.

3. `Feqma` divides force by mass to calculate acceleration. Naming block inputs can make block connections easier.

   In the **Input Name** column: name input 1 `force`; name input 2 `mass`.

4. Click the **Outputs** tab and set the **Output Label** to `acc`.
5. Click the **Display** tab and enable the **Show Output Labels** checkbox.
6. Click **OK** to accept current values and close the dialog box.

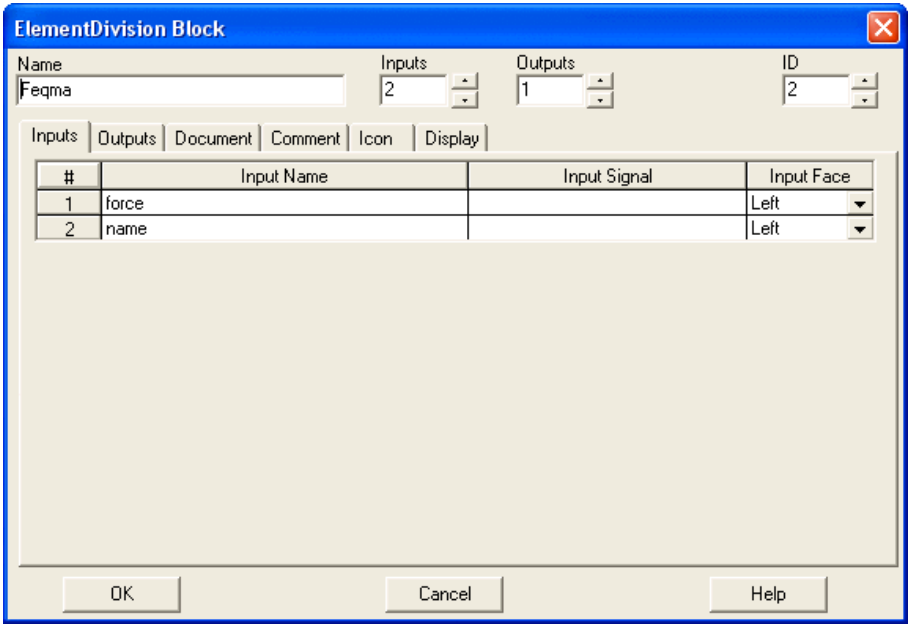


Figure 4-18. ElementDivision Block Properties Dialog Box: Inputs Tab

Complete the following steps to edit block properties for the first Integrator block.

1. In the Editor, move the cursor over the first Integrator block, second block from the right as shown in Figure 4-14, and press the <Enter> key.

   The **Integrator Block** properties dialog box is displayed. The **Parameters** tab is selected and all properties are set to their default values. Figure 4-19 shows the **Integrator Block** properties dialog box as it appears during step 3.

2. Name the block `accToVel`.

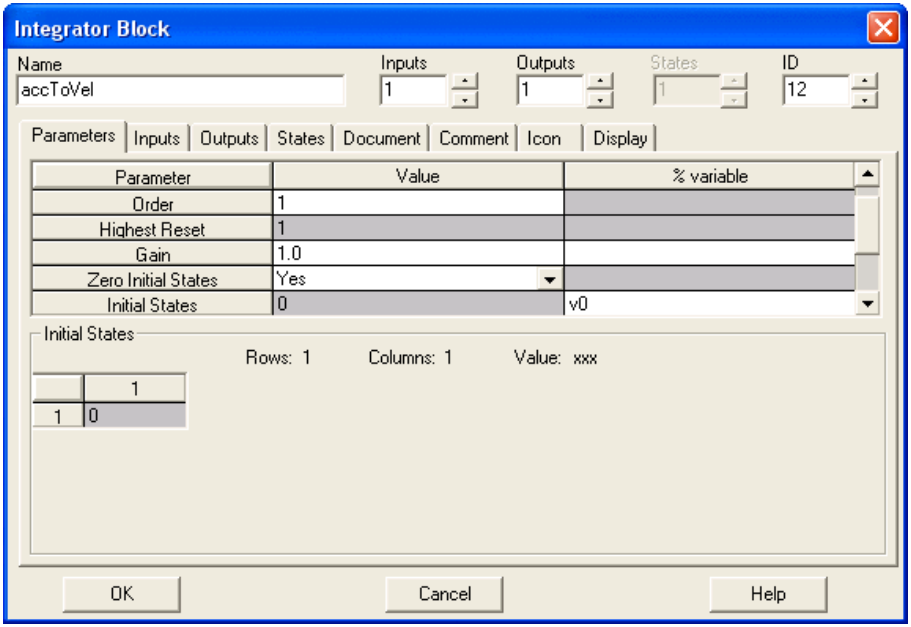


Figure 4-19. Integrator Block Properties Dialog Box: Parameters Tab

3. Verify that the **Parameters** tab of the **Block Properties** dialog box is selected, as shown in Figure 4-19.

   Scroll down in the **Parameter** table to access the field in the **% variable** column, and the **Initial States** row. Click that field and type `v0`.

4. Click the **Outputs** tab and set the **Output Label** to `vel`.
5. Click the **Display** tab and enable the **Show Output Labels** checkbox.
6. Click **OK** to accept current values and close the dialog box.

Repeat steps 1 through 6 for the second integrator block. Name it `velToPos`. Give it an **Initial States %** variable named `p0`. Set the **Output Label** to `pos`.

After editing block properties, your block diagram should resemble Figure 4-20. The short lines extending out from the sides of the blocks are input and output *pins*. Each output pin is now labeled.

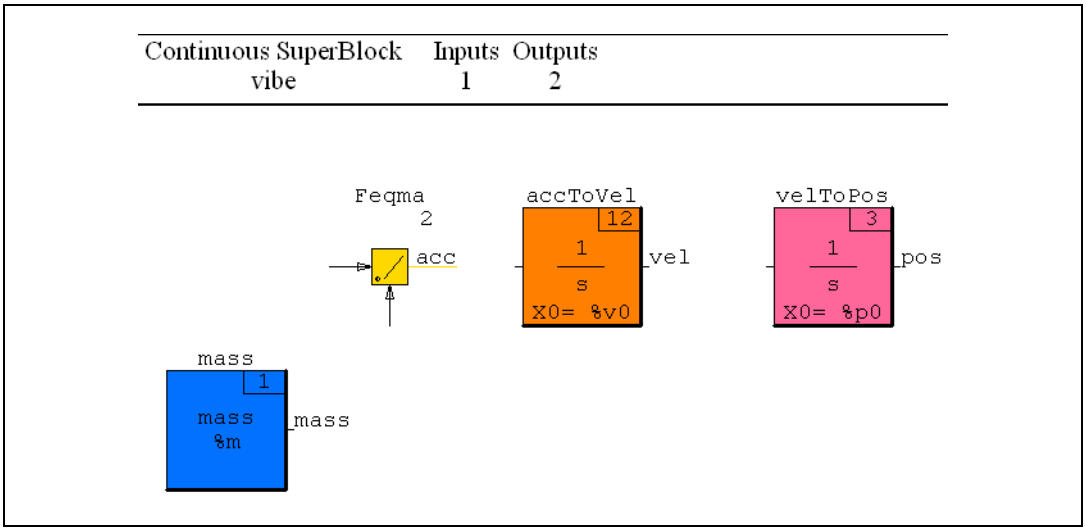


Figure 4-20. Block Diagram after Editing Block Properties

Connecting Blocks

When two blocks are connected, an output signal of one becomes the input signal of the other (a block cannot be connected to itself). Connections between the blocks of a SuperBlock are called *internal connections*. An internal connection is *directed* from an output pin of the *source* block to an input pin of the *destination* block.

Note A source block output pin can be connected to zero, one, or more than one destinations. However, a destination block input pin can be connected to at most one source: either a source block output pin or a SuperBlock input.

Complete the following steps to connect the Constant block to the AlgebraicExpression block.

1. Click the Constant block with MB2 (middle mouse button).
2. Click the AlgebraicExpression block with MB2.

   The Connection Editor is displayed, as shown in Figure 4-21.

   - Source block information is displayed on the left.
   - Destination block information is displayed on the right.
   - Output labels and input names are displayed when available.

- Data types are displayed when available; `F` (Float) is the default.
- Source block outputs are numbered in a column opposite the numbered inputs of the destination block. Click the numbers with MB1 to add and delete connections.

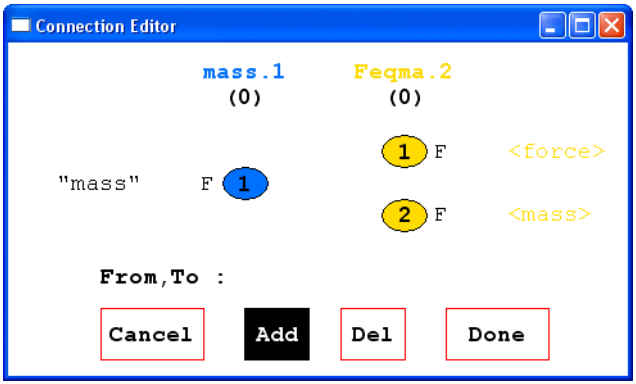


Figure 4-21. Connection Editor

3. Verify that the **Add** button is highlighted. With **MB1**, click output **1** (left column), and click input **2** (right column). A line is drawn to indicate the connection as shown in Figure 4-22.
4. Click **Done** to accept the connection and close the dialog box.

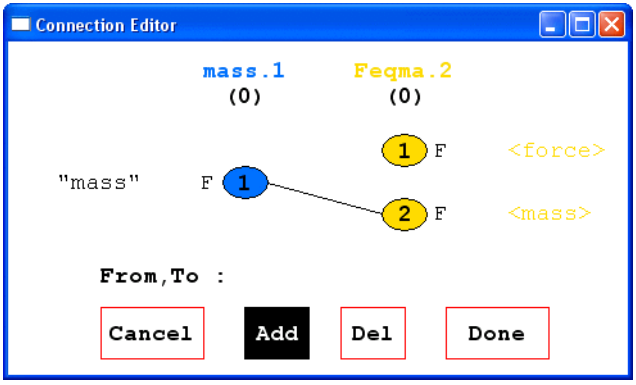


Figure 4-22. Adding a Connection with the Connection Editor

Complete the internal connections as shown in Figure 4-23. In each case, use MB2 to click the source block, and then the destination block.

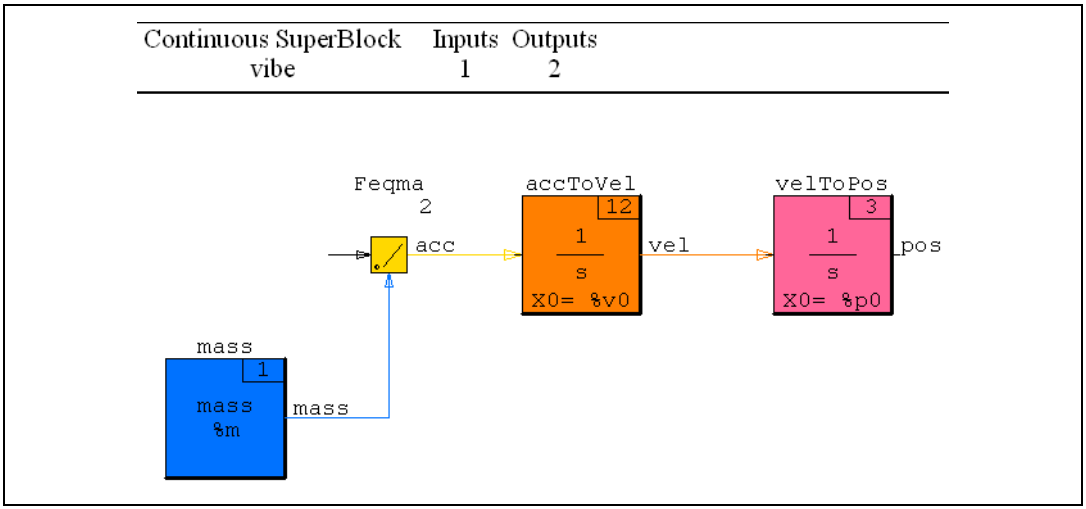


Figure 4-23. Block Diagram after Internal Connections

Note When only one possible connection can be made between the source and destination blocks, the connection is made without displaying the Connection Editor.

Connecting SuperBlock Inputs and Outputs

Connections from SuperBlock inputs to blocks, and from blocks to SuperBlock outputs are called *external connections*. An external input connection is directed from a SuperBlock input to an input pin of the destination block. An external output connection is directed from an output pin of the source block to a SuperBlock output.

Note A SuperBlock input can be connected to zero, one, or more destination blocks. However, a SuperBlock output must be connected to exactly one source block.

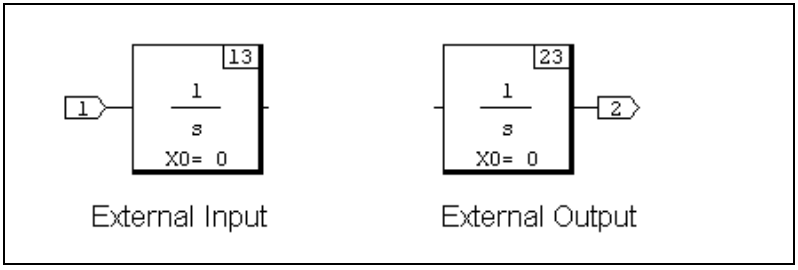


Figure 4-24. External Input and Output Flags

External input and output connections are represented in a block diagram by flags containing the number of the corresponding input or output. Figure 4-24 shows typical external connection flags: a SuperBlock input 1, and a SuperBlock output 2.

This section describes how to connect the external input and output signals for the `vibe` SuperBlock.

Complete the following steps to connect the `vibe` SuperBlock input to the `Feqma` ElementDivision block:

1. Move the cursor to an open space in the editor (not over any block), and click with MB2.
2. Click the ElementDivision block with MB2.

   The Connection Editor is displayed. The layout of information is analogous to that displayed for internal connections, as shown in Figure 4-21.
3. Verify that the **Add** button is highlighted. With MB1, click external input **1** (left column), and click block input **1** (right column). A line is drawn to indicate the connection.
4. Click **Done** to accept the connection and close the dialog box.

To connect the `accToVel` Integrator block to the `vibe` SuperBlock output 1, complete the following steps:

1. Click the `accToVel` Integrator block with MB2.
2. Move the cursor to an open space in the editor (not over any block), and click with MB2.

   The Connection Editor is displayed. The layout of information is analogous to that displayed for internal connections.
3. Verify that the **Add** button is highlighted. With MB1, click block output **1** (left column), and click external output **1** (left column). A line is drawn to indicate the connection.
4. Click **Done** to accept the connection and close the dialog box.

Repeat steps 1 through 4 to connect the `velToPos` Integrator block to `vibe` SuperBlock output 2.

When complete, your block diagram should resemble Figure 4-25.

Note The current block diagram models force acting on a mass. The spring-mass damper model will be completed in the *Encapsulating a SuperBlock* section.

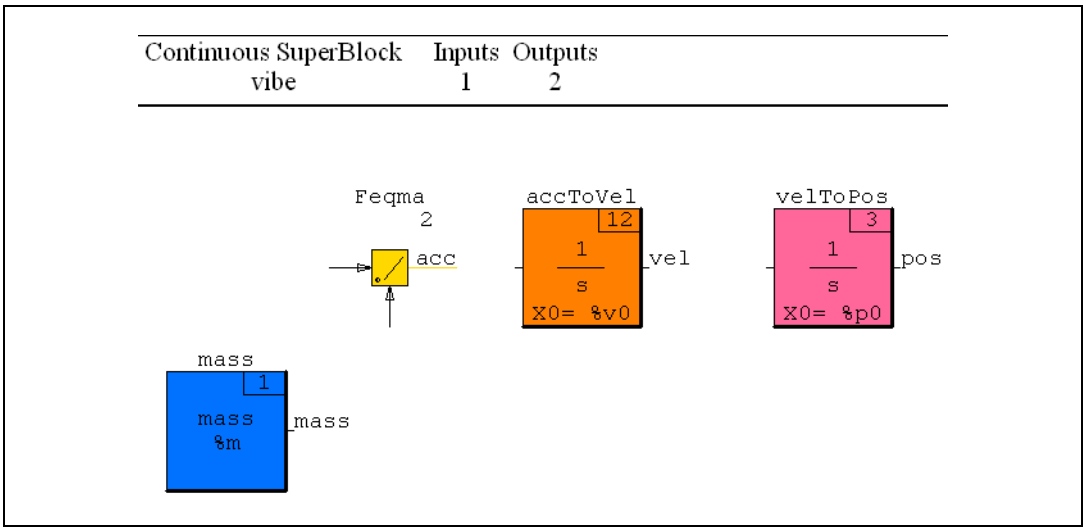


Figure 4-25. Block Diagram after External Connections

Saving a SuperBlock

To avoid loss of your work, you should save your block diagrams at regular intervals during development. This section describes how you save the `vibe` SuperBlock.

Note SuperBlocks can be saved individually, or grouped in catalogs. The method described here saves all SuperBlocks in the current catalog of the Catalog Browser to a single catalog file.

Complete the following steps to save the `vibe` SuperBlock to a catalog file.

1. Make the Editor the active window and update the `vibe` SuperBlock by selecting **File»Update**.
2. Make the Catalog Browser the active window and refresh its contents by selecting **View»Update**. A list of SuperBlocks in the current catalog is displayed on the right.
3. Select **File»Save As**.

   The **Save As** dialog box is displayed. Figure 4-26 shows the **Save As** dialog box as it appears during step 5.
4. Select a directory in which to save SuperBlock catalogs.

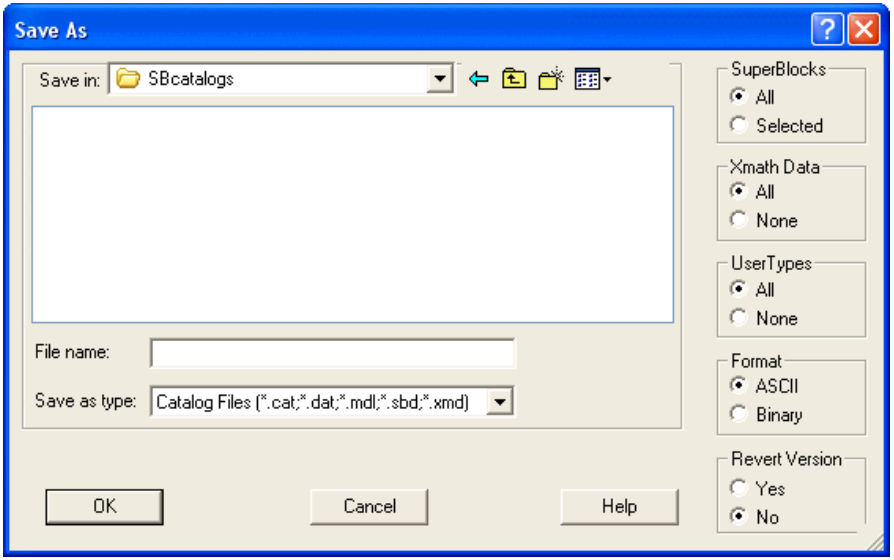


Figure 4-26. Save As Dialog Box

5. Click the **File name** field and type `vibe.cat`, as shown in Figure 4-26.

Note If you type a file name with no extension, the default extension `.dat` is appended to the file name when the file is saved.

6. Click **OK** to save the file and close the dialog box.

Simulating a SuperBlock

This section describes how to simulate the SuperBlock `vibe`. The current model applies an input force to a mass. The resulting velocity and position of the mass are output.

Simulations proceed with respect to a user-defined time vector. Because `vibe` is a *continuous* SuperBlock, its simulation algorithms are relatively independent of the granularity of the time vector sequence. However, the input and output of a continuous simulation are indexed by its time vector, so the granularity must accommodate those factors.

To simulate `vibe`, an input force vector is specified whose elements correspond to those of the time vector. Here, you model the force of gravity. Therefore, the input signal is constant with respect to time.

There are three % variable parameters to define: the mass `m`, the initial velocity `v0`, and the initial position `p0`. In this tutorial, you assign values with MKS units.

Complete the following steps to simulate the SuperBlock `vibe` from the Xmath command area.

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

   ```
   t=[0:0.01:5]';
   ```

Note Remember to type the semicolon to suppress output.

   Time must be specified in a column vector. You have created a regular column vector with 501 elements: `0,0.01,0.02,…,4.99,5.00`. The simulation will proceed for 5 seconds, with a computational granularity of 0.01 seconds.

2. Define an input force vector (MKS gravity is approximately –9.8 meters/sec^2):

   ```
   u=-9.8*ones(t);
   ```

   The input variable `u` is constructed to have the same dimensions as the time vector `t`. Here, `u` is a column vector of 501 elements.

3. Specify a mass of 1 kilogram, starting at rest and position 0:

   ```
   m=1;
   v0=0;
   p0=0;
   ```

4. Execute the simulation:

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

   Information about the simulation is displayed in the Xmath log area. When the simulation completes, velocity and position are plotted in an Xmath Graphics window, as shown in Figure 4-27.

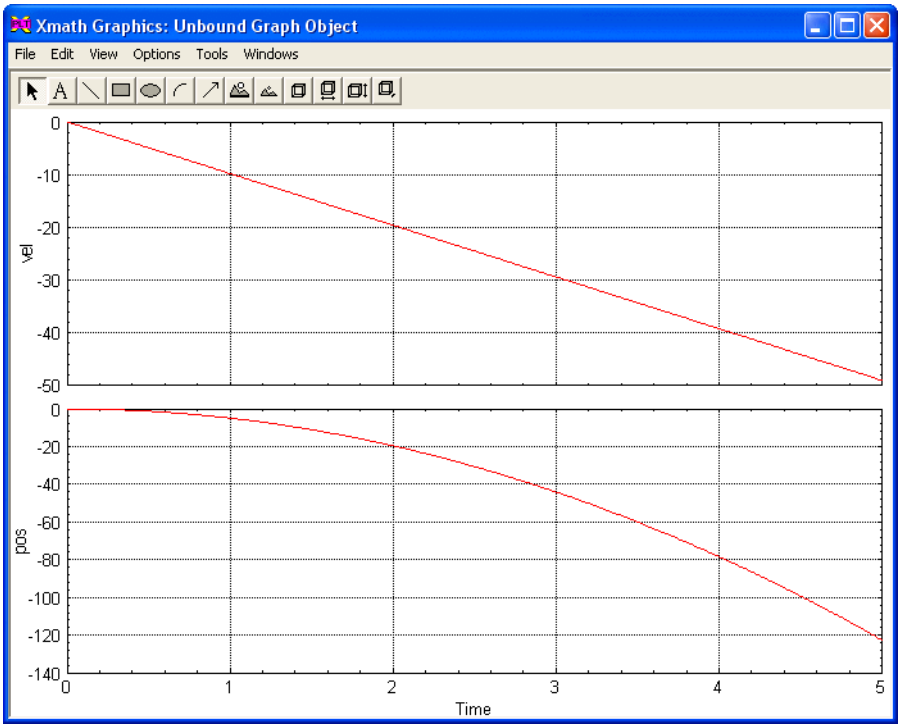


Figure 4-27. Simulation Results

The simulation result `y` is a pdm consisting of 501 row vectors. Each vector has two scalar elements—a velocity and a position. You can inspect the velocity and position values at a given time by computing its position in the pdm. For example, to see the result at time 3.83 seconds, type:

```
y(3.83*100+1)
```

Exercise the model by simulating other values for the parameters `m`, `v0`, and `p0`. You also can change the input force. Try `u=cos(4*t);`.

Encapsulating a SuperBlock

This section describes how to encapsulate a SuperBlock and develop a hierarchical block diagram.

Complete the following steps to encapsulate a SuperBlock.

1. Make the Editor the active window and update the `vibe` SuperBlock by selecting **File»Update**.

2. Select the ElementDivision block and both Integrator blocks of the block diagram by holding down the <Ctrl> key and clicking each in turn with MB1. A heavy rectangular border indicates that a block is selected.
3. Select **Edit»Make SuperBlock**. The block diagram now shows a new SuperBlock block, along with the `mass` Constant block.

Complete the following steps to edit properties of the new SuperBlock block.

1. In the Editor, move the cursor over the SuperBlock block and press the <Enter> key.

   The **SuperBlock Block** properties dialog box is displayed. The **Parameters** tab is selected and all properties are set to their default values. Figure 4-28 shows the **SuperBlock Block** properties dialog box as it appears during step 3.
2. Click the **Name** field and replace the default name with `Newton`.

3. Click the **Display** tab, as shown in Figure 4-28.

   Locate the drop-down combo box labeled **Icon Type**. Change its value to `User`.

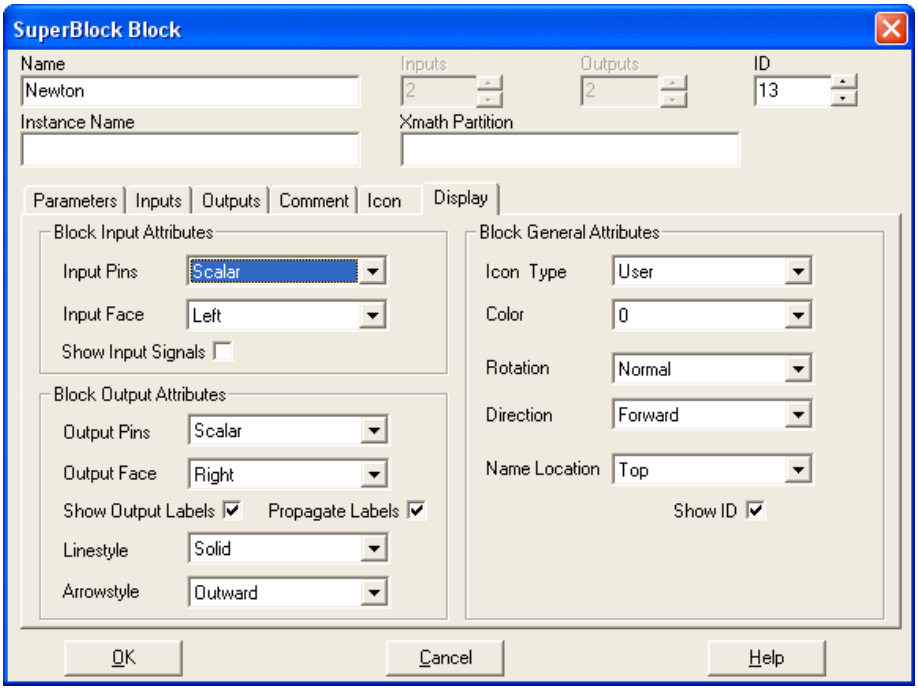


Figure 4-28. SuperBlock Block Properties: Encapsulation

4. Click **OK** to accept current values and close the dialog box.

The Superblock editor now displays the `mass` Constant block, and the `Newton` SuperBlock block. Position and resize the blocks so that they resemble the block diagram displayed in Figure 4-29. Recall, blocks are positioned by dragging with MB1.

To enlarge the `Newton` SuperBlock block, move the cursor over the block, and press <E> twice.

To reduce the `mass` Constant block, move the cursor over the block, and press <R> twice.

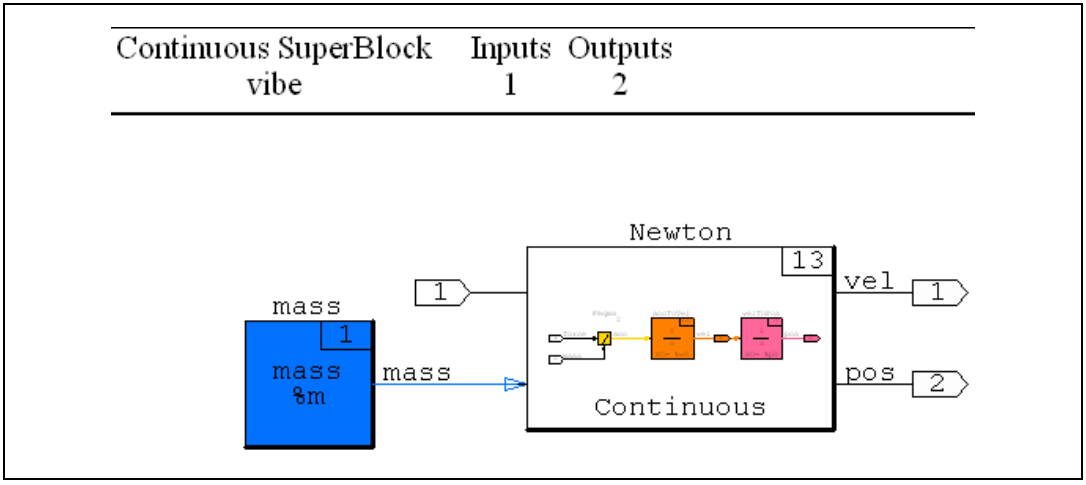


Figure 4-29. Newton SuperBlock Block

To complete a block diagram representation of a damped spring, a few more blocks are needed. A solution is shown in Figure 4-30. Two Gain blocks have been added. One produces the damping force by multiplying velocity by the damping constant. Position and stiffness are used by the other to determine the spring force. These forces are subtracted from the `vibe` input force by a Summer block that passes the resultant force to the Newton SuperBlock block.

Complete the following steps to implement the damped spring representation.

1. Add blocks to the block diagram.

   Open and position the Palette Browser. Select the **Algebraic** palette. With MB1, drag and drop a Gain block and a Summer block into the Editor.

2. Flip, reduce, and duplicate the Gain block.

   Move the cursor over the Gain block and press the <F>, <R>, and <D> keys (one key at a time).

   You now have two gain blocks that have been flipped horizontally. Input pins are now on the right; output pins are on the left.

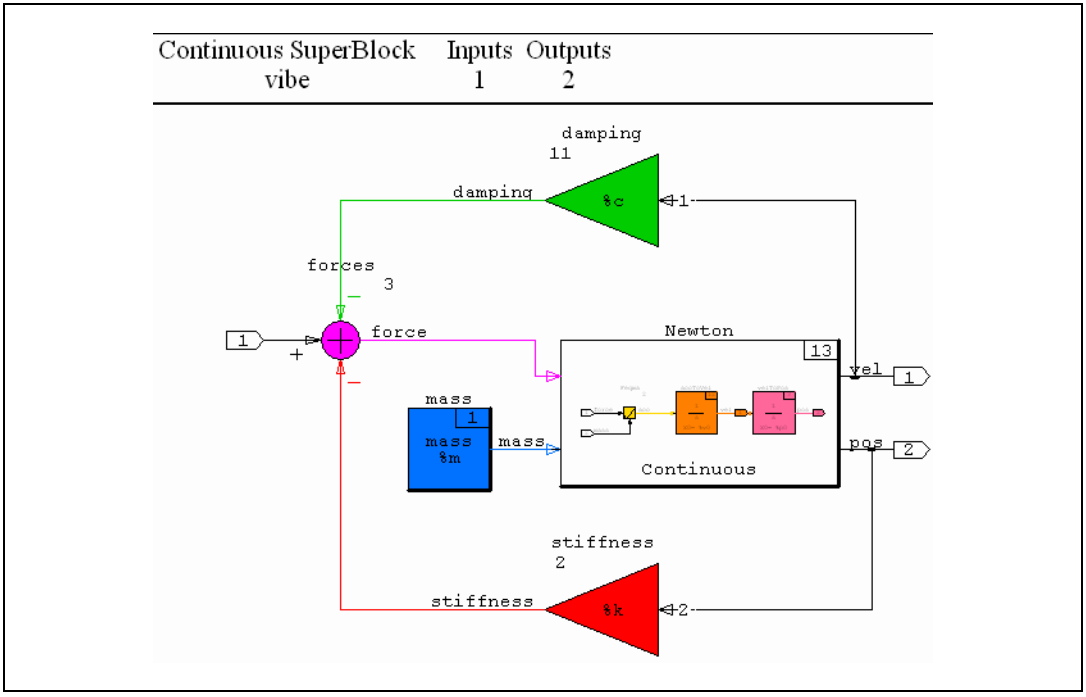


Figure 4-30. Block Diagram for Damped Spring

3. Position the new blocks as shown in Figure 4-30.
4. Edit properties for the Summer block.
   a. Move the cursor over the Summer block and press the <Enter> key.

      The **Summer Block** properties dialog box is displayed. The **Parameters** tab is selected and all properties are set to their default values. Figure 4-31 shows the **Summer Block** properties dialog box as it appears during step e.

   b. Name the block `forces`.
   c. Change the number of inputs to `3`.
   d. In the **Parameter** table, click in the field in the **Value** column, and the **Number Branches** row. Change its value to `3`.
   e. In the **Parameter** table, click in the field in the **Value** column, and the **Signs(+1,–1)** row. Change its value to `-1,1,-1`.

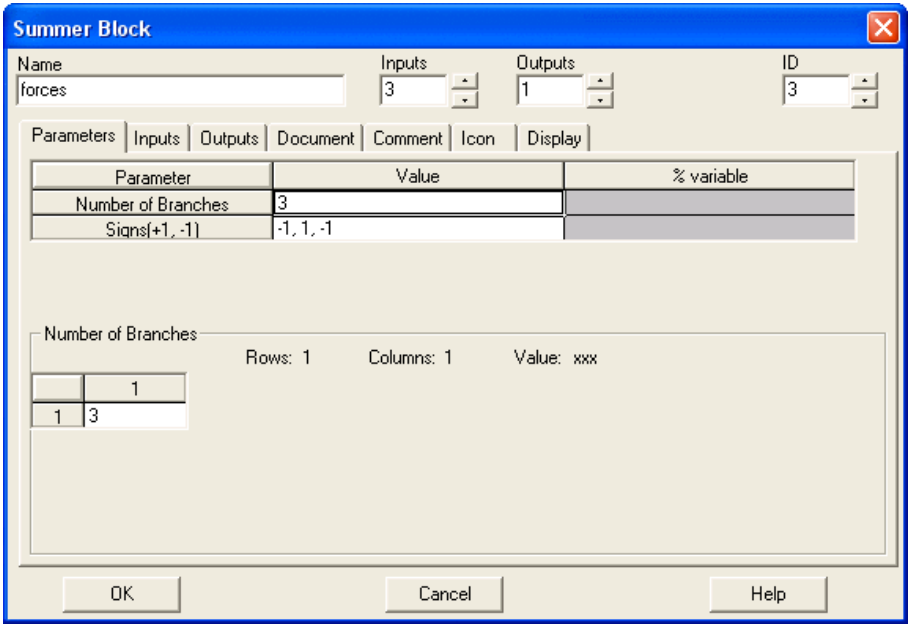


Figure 4-31. Summer Block Properties Dialog Box

f. Click the **Inputs** tab and name the inputs: `damping`, `external`, and `stiffness`.

g. Click the **Outputs** tab and label the output: `force`,

h. Click the **Display** tab, change the value of **Icon Type** to `User`, and enable the **Show Output Labels** checkbox.

i. Click **OK** to accept current values and close the dialog box.

5. Edit properties for the Gain blocks.

   You give each Gain block a name, a % variable assignment, label its output, and enable the **Show Output Labels** checkbox. For additional detail, refer to the instructions for setting these properties in a Constant block, in the *Editing Block Properties* section.

   a. Name the upper Gain block `stiffness`, give it a % variable named `k`, label its output `stiffness`, and enable the **Show Output Labels** checkbox.

   b. Name the lower Gain block `damping`, give it a % variable named `c`, label its output `damping`, and enable the **Show Output Labels** checkbox.

6. Make block diagram connections as shown in Figure 4-30. For additional detail, refer to the *Connecting Blocks* and *Connecting SuperBlock Inputs and Outputs* sections.
7. Save the current catalog. For additional detail, refer to the *Saving a SuperBlock* section.

Note The current tutorial block diagrams can be loaded into the CatalogBrowser by executing the following command from the Xmath Commands window.

```
load "$SYSBLD\examples\gs_tutorial\vibe1.cat";
```

The `vibe1` catalog contains a solution of the basic spring-mass damper constructed in the first four sections of the SystemBuild tutorial.

Complete the following steps to simulate the block diagram representation of the damped spring.

1. In the Xmath command area, verify or reenter the original values for `t`, `u`, `m`, `p0`, and `v0`:

   ```
   t = [0:0.01:5]';
   u = -9.8 * ones(t);
   m = 1;
   v0 = 0;
   p0 = 0;
   ```

2. Define values for the stiffness `k` and damping constant `c`:

   ```
   k = 100;
   c = 1;
   ```

3. Execute the simulation:

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

   When the simulation completes, velocity and position are plotted in an Xmath Graphics window, as shown in Figure 4-32.

   Exercise the model by simulating other values for the parameters and the input force. Try `u = zeros(t); p0 = 1;`.

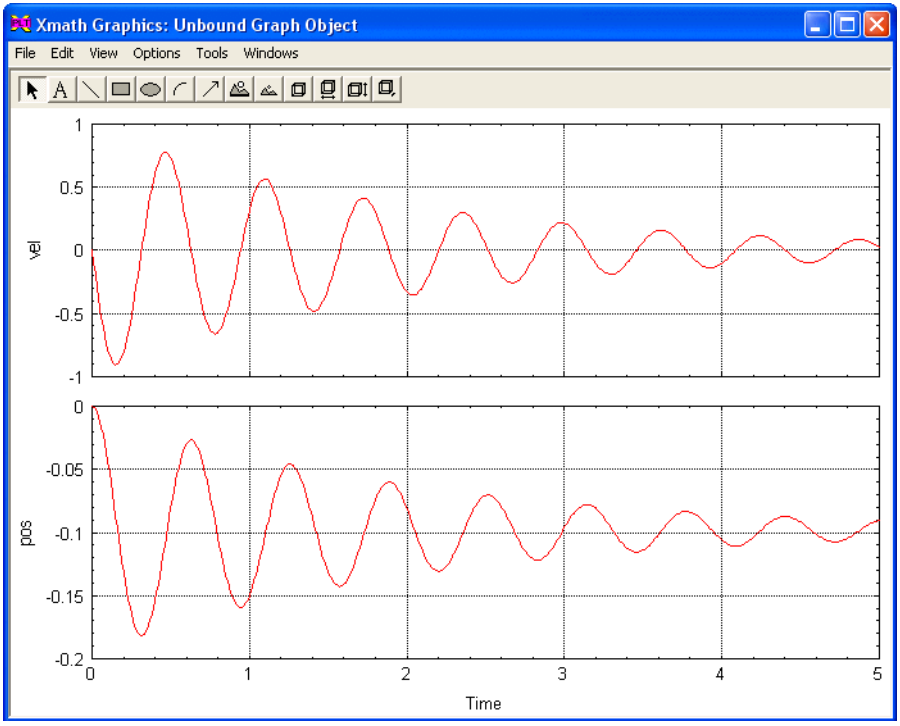


Figure 4-32. Damped Spring Simulation Plot

Exercise

While using the Palette Browser, you have probably noticed many more pre-defined block types. Complete the following steps to read brief descriptions of SystemBuild block types.

1. In the Xmath command area, enter:

   ```
   help blocks
   ```

2. When the Blocks topic appears, scroll down to the tables that list the block types by palette, and read the descriptions there. For more detailed information about block types, follow the table links.

5

AutoCode

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 and can be targeted to run on computers or on an actual controller. The default target is a stand-alone simulation that you can execute on your computer. You can load the results of the simulation back into Xmath for analysis.

Generating Non-Customized Code

Complete the following steps to generate code for the Discrete Cruise System SystemBuild model.

1. If Xmath is not currently running, start Xmath as described in the *Starting Xmath* section of Chapter 3, *Xmath*.
2. Verify that you have write permission for the current directory, and that it is where you want to save your code. If not, enter the follwing command from the Xmath command window, substituting your directory name:

   ```
   set directory ="write_enabled_directory"
   ```

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

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

Note Environment variables are only recognized on the Xmath command line. For other loading methods, you must know the full pathname of the SystemBuild directory.

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

Note You must generate code from a top level SuperBlock.

5. From the Catalog Browser, select **Tools»AutoCode** to bring up the **Generate Real-Time Code** dialog box, as shown in Figure 5-1.
6. Enter a name in the **File name** field, or accept the default, `Discrete_Cruise_System`.

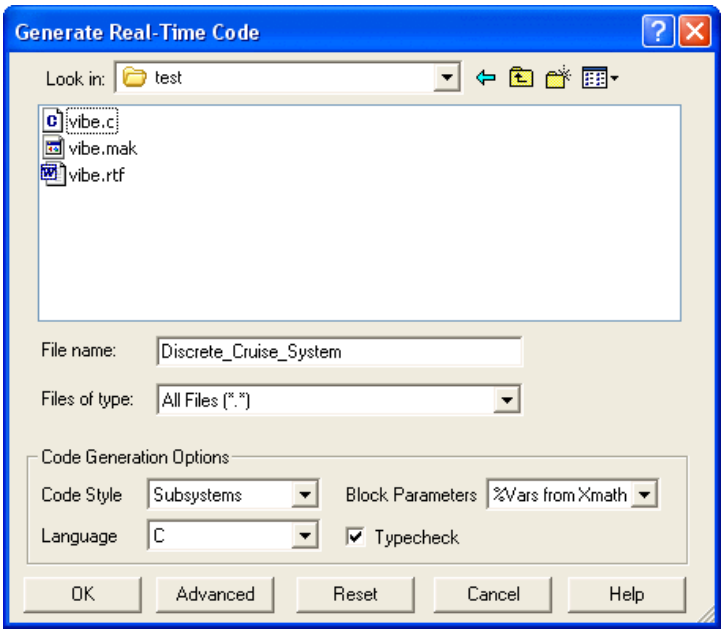


Figure 5-1. Generate Real-Time Code Dialog Box

7. Click **OK** to start the code generation process.
8. Activate the Xmath Commands window to monitor the progress of the code generation.
9. After the code generation is complete, look for a statement similar to the following in the Xmath log area:

   `Output generated in` *`your_directory`*`\Discrete_Cruise_System.c.`

   `Code generation complete.`

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

`oscmd ("type` *`your_directory`*`\Discrete_Cruise_System.c")`

Generating Customized Code

To customize your AutoCode output, click **Advanced** on the **Generate Real-Time Code** dialog box; this opens the **Advanced** dialog box, as shown in Figure 5-2.

You can use the **Advanced** dialog box from the **AutoCode Code Generation** dialog box 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. Numerous templates are available, including one to customize the generated code for the pSOSystem real-time operating system. For more information on templates, refer to the *Template Programming Language User Guide*.
- 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 also can specify indentation between levels, as well as set a number of other parameters.
- The **IALG (Integration Algorithms) Options** tab lets you select an integration 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. Refer to the *AutoCode Reference* manual 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.

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

For information about compiling, executing, and using the generated code, refer to the *AutoCode User Guide*. For information about autocode keywords, refer to the *AutoCode* topic in the *MATRIXx Help*.

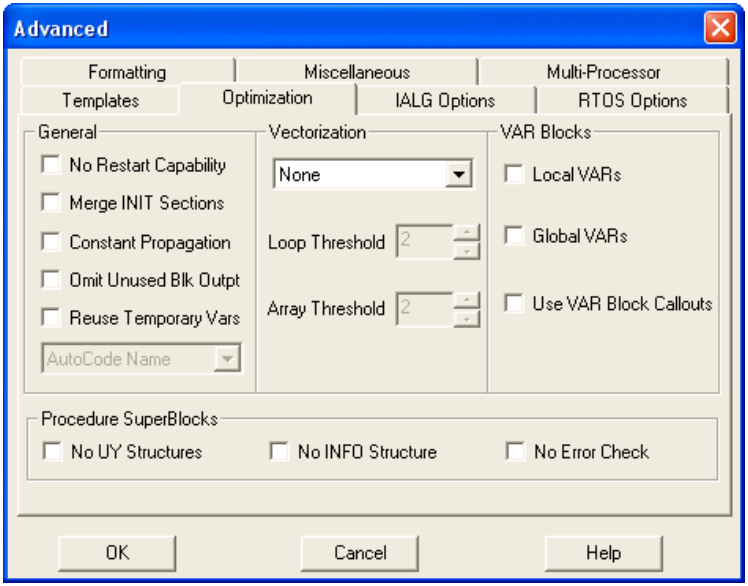


Figure 5-2. Advanced Dialog Box

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 box.

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

Generating Non-Customized Documentation

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

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

   ```
   set directory = "your_directory"
   ```

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

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

Note Environment variables are recognized only in the Xmath command area. For loading with other methods, you must know the full pathname of the SystemBuild directory. Refer to 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 box, as shown in Figure 6-1.

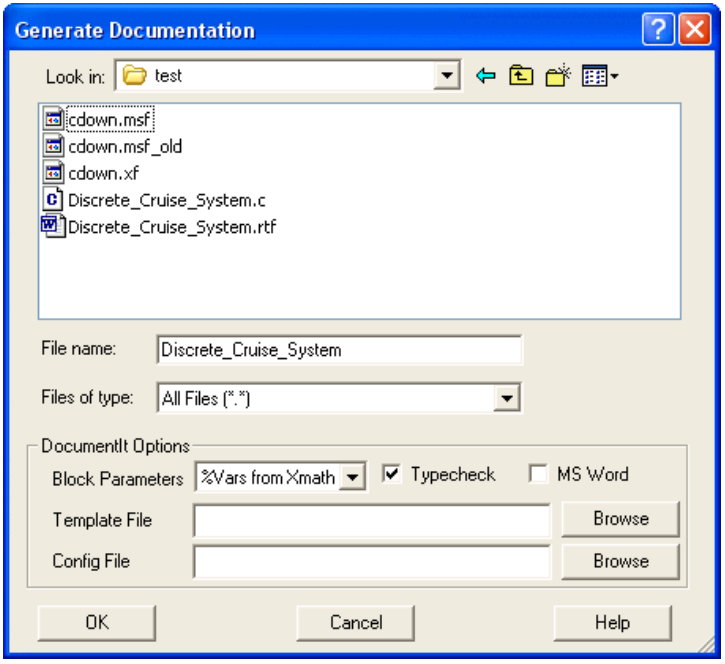


Figure 6-1. Generate Documentation Dialog Box

5. Choose a directory, and enter a name in the **File name** 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. Select the **Xmath Commands** window to monitor the progress of the documentation generation.
8. After 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 a `.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 ("type
   your_directory\Discrete_Cruise_System.doc")
   ```

Figure 6-2 provides a samping of DocumentIt output from the `Discrete_Cruise_System.doc` document.

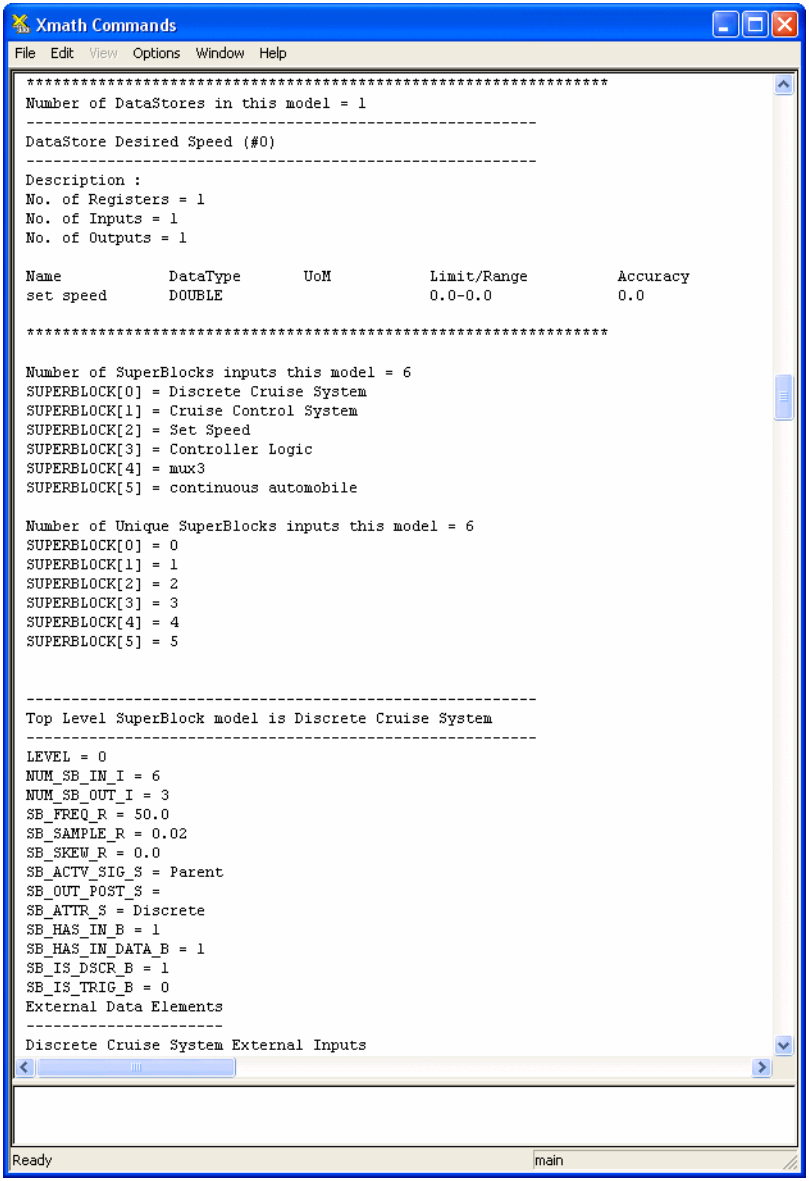


Figure 6-2. Sample of DocumentIt Output

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 also can place publishing software markup commands (for example, FrameMaker, Microsoft Word, 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. Refer to the *Template Programming Language User Guide* for more information.

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

A

Technical Support and Professional Services

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

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

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

Index

G

H

I

K

L

M

N

O

P

R

S

T

U

W

X