MATRIXx™

DocumentIt™ User's Guide

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 Any trademark or copyright notices to those entities are no longer valid and any references to those entities as the licensor to the MATRIXx products and related items should now be considered as referring to National Instruments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

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

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

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

NATIONAL INSTRUMENTS™

May 2003 Edition
Part Number 370769A-01

Worldwide Technical Support and Product Information

ni.com

National Instruments Corporate Headquarters

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

Worldwide Offices

Australia 1800 300 800, Austria 43 0 662 45 79 90 0, Belgium 32 0 2 757 00 20, Brazil 55 11 3262 3599,
Canada (Calgary) 403 274 9391, Canada (Montreal) 514 288 5722, Canada (Ottawa) 613 233 5949,
Canada (Québec) 514 694 8521, Canada (Toronto) 905 785 0085, Canada (Vancouver) 514 685 7530,
China 86 21 6555 7838, Czech Republic 420 2 2423 5774, Denmark 45 45 76 26 00,
Finland 385 0 9 725 725 11, France 33 0 1 48 14 24 24, Germany 49 0 89 741 31 30, Greece 30 2 10 42 96 427,
India 91 80 51190000, Israel 972 0 3 6393737, Italy 39 02 413091, Japan 81 3 5472 2970,
Korea 82 02 3451 3400, Malaysia 603 9131 0918, Mexico 001 800 010 0793, Netherlands 31 0 348 433 466,
New Zealand 1800 300 800, Norway 47 0 66 90 76 60, Poland 48 0 22 3390 150, Portugal 351 210 311 210,
Russia 7 095 238 7139, 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 Resources 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™, LabVIEW™, MATRIXx™, National Instruments™, NI™, ni.com™, RealSim™, SystemBuild™, and Xmath™ are trademarks of National Instruments Corporation.

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

Patents

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

WARNING REGARDING USE OF NATIONAL INSTRUMENTS PRODUCTS

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

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

Contents

E Generating Documents Using Microsoft Word

F Generating ASCII Documents

Using This Guide

With SystemBuild™ menu options and this guide, you will soon be able to generate high-quality design documentation automatically from SystemBuild block diagrams.

Organization

This guide provides the information you need to start using DocumentIt™, as well as the more advanced details necessary to customize the generated design documentation. The summaries that follow will tell you *exactly* what to read for the task you want to accomplish.

- Chapter 1, *Introduction*, provides an overview of the Integrated Systems design documentation generation concept, a description of the basic steps required to produce documentation, and a brief example of documentation generation from a SystemBuild model. This chapter, along with Chapter 2, will give you everything you need to know in order to generate documentation using any of the templates described in Appendix B through Appendix E .

  If you want to create custom documentation, refer to Chapter 3 and the Related Publications references provided later in this section.

- Chapter 2, *Invoking DocumentIt*, tells how to generate documents from SystemBuild, the Xmath Commands window, and from the operating system command line.

- Chapter 3, *Customizing the Generated Documentation*, points you to a detailed explanation of template files (`*.tpl`) for formatting a document. Integrated Systems supplies DocumentIt templates (`*.tpl` files) for each of the examples

given in Appendix B through Appendix E. You can copy the `*.tpl` files from any of these examples and modify them using the information in this chapter.

- Appendix A, *DocumentIt Options*, provides a description of the DocumentIt options, and explains how to use an `autostar.opt` file.
- Appendix B, *Generating Documents Using FrameMaker*, provides a description of how to generate a sample design document using DocumentIt and FrameMaker templates.
- Appendix C, *Generating 2167A Documents Using FrameMaker*, provides a description of how to generate a sample design document conforming closely with DOD-STD-2167A using DocumentIt and FrameMaker templates.
- Appendix D, *Generating 2167A Documents Using Interleaf*, provides a description of how to generate a sample design document conforming closely with DOD-STD-2167A using DocumentIt and Interleaf templates.
- Appendix E, *Generating Documents Using Microsoft Word*, provides a description of how to use a template to create a general purpose document in Microsoft Word.
- Appendix F, *Generating ASCII Documents*, provides a description of how to generate a sample design document using a DocumentIt template.

This manual also has an *Index*.

Conventions

This section describes the conventions used in this manual.

Font Conventions

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

%	Represents the C shell system prompt in command examples.
`courier font`	Denotes directory names, file names, menu names, menu options, commands, Template Programming Language (*tpl*) segments, syntax statements, operators, and scope classes within text. This font type is also used in programming examples.
italic	*Italic* is used with the default font for emphasis and publication titles.
`courier italic font`	Denotes placeholders in syntax examples.
`courier bold font`	Denotes command-line input.
Bold Helvetica narrow	Buttons, fields, and icons in a graphical user interface are set in **bold Helvetica narrow** type. Keyboard keys are also set in this type.
<*hll*>	Single-letter abbreviation denoting a high-level language supported by AutoCode: c (C) or a (Ada).
<*HLL*>	Abbreviation for the full name of one of the two high-level languages supported by AutoCode: C or Ada.

Format Conventions

This manual uses two special formatting conventions, one to present code and programming examples, as well as sample procedures, and one to present sample input/output.

Sample Procedures

Example 1 shows the formatting convention for sample procedures and code and programming examples.

EXAMPLE 1: Code, Programming, and Sample Input/Output Example Format

```
#include "XmathLNX.h"

extern void cosine_();

void myfun(nlhs, lhs, nrhs, rhs)
int nlhs, nrhs;
externType **lhs, **rhs;
{
   /* y = cos(x) */

   et_matrix *x, *y;
   x = (et_matrix*)rhs[0];
   y = AllocateMatrix(x->rows, x->columns, 1);
   lhs[0] = (externType*)y;
   cosine_(&x->rows, &x->columns, x->real, y->real);
}

static functionData fdata[] = {
   {"myfun", myfun, 1, 1, 1, 1, (char*)0},
   {0,0,0,0,0,0,0}
};

main(argc, argv)
int argc;
char **argv;
{
   XmathMain(argc, argv, fdata, 1);
   return 0;
}
```

Sample Input/Output

The formatting convention for sample input/output is shown in the next several paragraphs.

Sample input is shown in bold and italic (used for placeholders):

`makeproject` *projname*

Sample output is shown in Example 1 above.

Mouse Conventions

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

Note and Caution Conventions

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

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

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

Environment Variables

At installation an environment variable is defined that points to the location where ISI tools have been installed. For UNIX systems, this variable is `$ISIHOME`. For Windows systems, this variable is `%ISIHOME%`.

How to Access Integrated Systems-Supplied Files

At several places in this manual, you are asked to execute, study, or copy and modify certain files we provide for your use. The paths to the files are specified by environment variables, which are established by the `xmath` startup software for your convenience. For UNIX, the environment variables are:

Variable	Status (as delivered)
`$ISIHOME`	Available from operating system or Xmath Commands window
`$SYSBLD`	Available from Xmath Commands window and SystemBuild Load/Save dialog
`$XMATH`	Available from Xmath Commands window
`$CASE`	Available from Xmath Commands window

To make $SYSBLD, $XMATH, or $CASE available from the operating system, in the Xmath Commands window, type:

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

where `variable` is SYSBLD, XMATH, or CASE, as you require. The Xmath software will display the required path. From the operating system, place this path in a `setenv` statement and execute it.

For information on Windows environment variables, see *Xmath Basics*.

Related Publications

Integrated Systems provides a complete library of publications to support its products. In addition to this guide, publications (from Integrated Systems and other sources) that you may find particularly useful when using DocumentIt include the following:

- *SystemBuild User's Guide*
- *SystemBuild Fuzzy Logic Block User's Guide*
- *Xmath Basics*
- *AutoCode User's Guide*
- *AutoCode Reference*
- *Template Programming Language User's Guide*
- *RealSim User's Guide*
- Military Standard: Defense System Software Development, *DOD-STD-2167A*, February 1988.
- *MML Reference* from Adobe Systems Incorporated

  This manual will help you create custom documentation formats using FrameMaker MML markup language.

Support

You can contact MATRIX$_X$ Technical Support in any of the three ways listed below. When Technical Support responds, you will be given a Call ID specific to the problem you have reported. *Please record the Call ID* and use it whenever you contact Technical Support regarding the issue.

- Submit a problem report via the ISI web site using the following URL:

  `http://www.isi.com/Support/MATRIXx`

  This is the preferred method, as it is the most traceable; your problem report will be automatically entered into our support database.

- Send e-mail to `mx_support@isi.com`. We can serve you better if you e-mail us with details on your configuration and the circumstances under which your problem occurred. We provide an ASCII file that you can use as a template for your e-mail to support; it can be found in:

  MATRIXX`/version/v6support.txt`

  where

  `MATRIXX` is your MATRIX$_X$ product version directory, which is under `ISIHOME`. To determine this directory location, from the Xmath command line, type one of the following commands.

  On the PC:

  ```
  oscmd("echo %MATRIXX%");
  ```

  On UNIX:

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

  If you have Xmath running you can use the following Xmath function call to copy the template to the current working directory:

  ```
  copyfile("$MATRIXX/version/v6support.txt")
  ```

- Call 800-958-8885 (where 1-800 service is available) or 408-542-1930. Telephone support hours are 7:00 a.m. through 5:30 p.m. PST, Monday through Friday. We can respond more efficiently if you are ready to provide the information requested in *MATRIXX*`/version/v6support.txt` at the time you call.

Using the ISI FTP Site

If your problem involves scripts or model file(s), Technical Support may ask you to FTP your files to us for further examination.

1. Connect to the ISI FTP site:

   ```
   ftp ftp.isi.com
   ```

2. Log on as `anonymous`, and supply your email address as the password.

3. Change to the `/incoming` directory:

   ```
   cd /incoming
   ```

4. Use `put` or `mput` to specify the file(s) you are transferring. When the transfer is complete, `quit`.

5. Send an email message to Technical Support that states the Call ID (if available), the *exact* name(s) of the file(s) you put in `/incoming`, and the approximate time you made the transfer; alternatively, call 800-958-8885 (where 1-800 service is available) or 408-542-1930 and provide this information. It will be a minimum of 15 to 20 minutes before the transferred file(s) will pass through the firewall.

1 Introduction

This chapter provides an overview of the MATRIX$_X$® design documentation generation concept, a description of the basic steps required to produce documentation, and a brief example of documentation generation from a SystemBuild™ model. This chapter, along with Chapter 2, describes everything you need to know in order to generate documentation using any of the templates described in Appendix B through Appendix F.

1.1 DocumentIt and the Rapid Prototyping Concept

Conventional real-time system development usually takes place in stages with separate tools for control design, software engineering, data acquisition, testing, and design documentation. The MATRIX$_X$ product family integrates tools for each stage of system development into a single environment. This allows a design to move easily from one stage to the next, making it possible to create a well-documented working prototype early in the design process (Figure 1-1 shows DocumentI™ in the MATRIX$_X$ product line).

Within the MATRIX$_X$ design automation product family, a system model can be built, simulated, analyzed, tested, and debugged using SystemBuild and Xmath®. Real-time code in a high-level language for the model (C or Ada) using AutoCode® and design documentation can be generated automatically with DocumentIt. The generated application code can be evaluated on the host with SystemBuild simulation or run on the RealSim™ controller for hardware in-the-loop testing. Finally, the generated application code can be cross-compiled and linked for implementation on an embedded processor. Documentation can be updated and automatically generated along each step of the process.

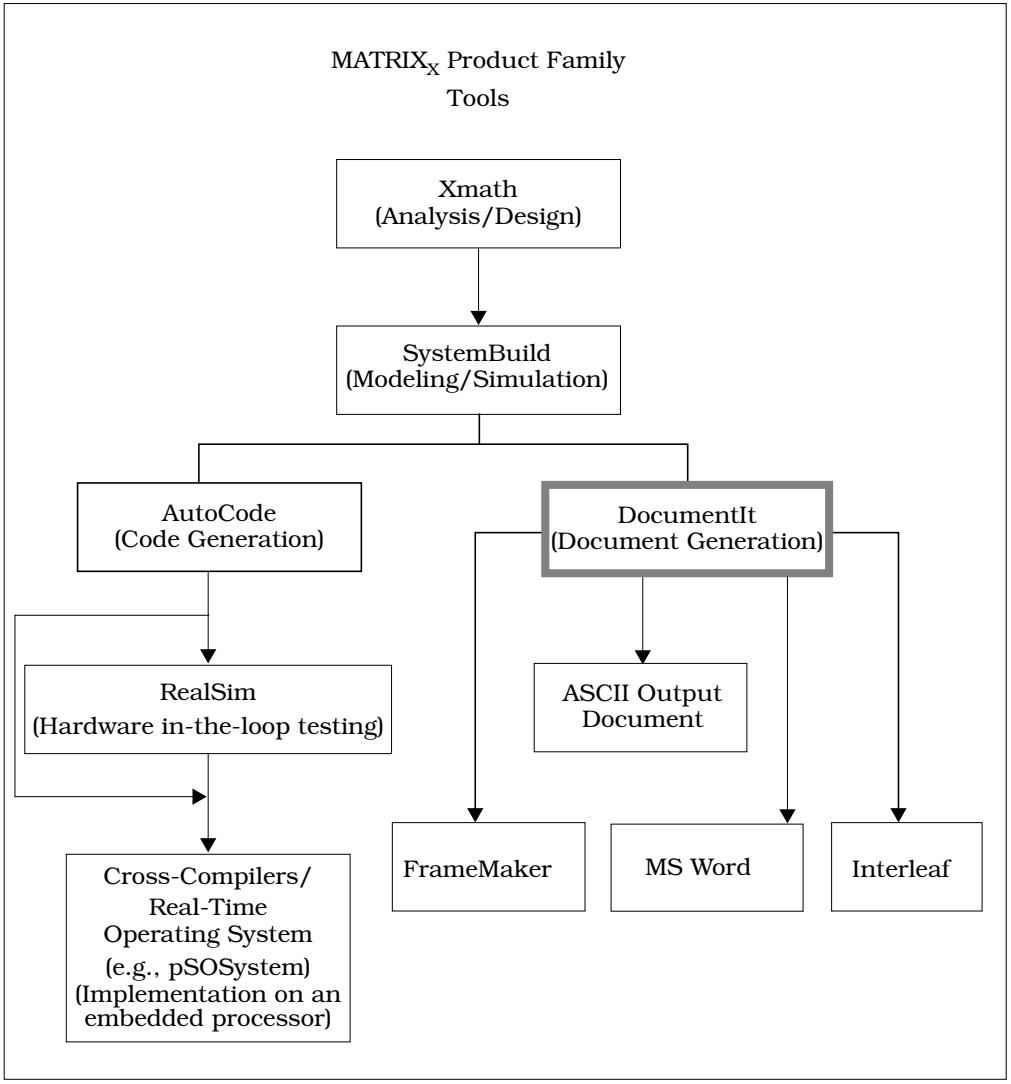


FIGURE 1-1 DocumentIt in the MATRIX$_X$ Product Line

1.2 Automatic Document Generation Process

As an integral part of MATRIX$_X$ Rapid Prototyping concept, DocumentIt lets you generate design documentation from a SystemBuild block diagram model quickly, automatically, and without programming skills. DocumentIt details the software design in a uniform standardized documentation format and fully supports the SystemBuild hierarchical approach to system design.

Because DocumentIt uses templates to format documents and select specific contents, you can configure DocumentIt to suit your particular documentation needs. The templates can be readily modified to conform to any documentation standard. For example, DocumentIt can be used to generate the following:

- Detailed description of the design in ASCII format
- Most industry-standard formats
- Tables containing SuperBlock and Block inputs and outputs, State Transition Diagrams, and Global data
- Documents formatted for FrameMaker (including DOD-STD-2167A) and Microsoft Word (or any application that reads RTF)

A typical sequence for using DocumentIt is as follows (this sequence corresponds to the sequence of steps shown in Figure 1-2):

6. Build and Document the Model

   Develop the continuous-time plant model and corresponding discrete-time controller SuperBlocks using SystemBuild block diagrams. The SystemBuild model is built up from a large palette of blocks which combine to describe the way the model works and how it should be controlled. DocumentIt automatically extracts information from the Inputs, Outputs, Document, and Comment fields for SuperBlocks and primitive blocks.

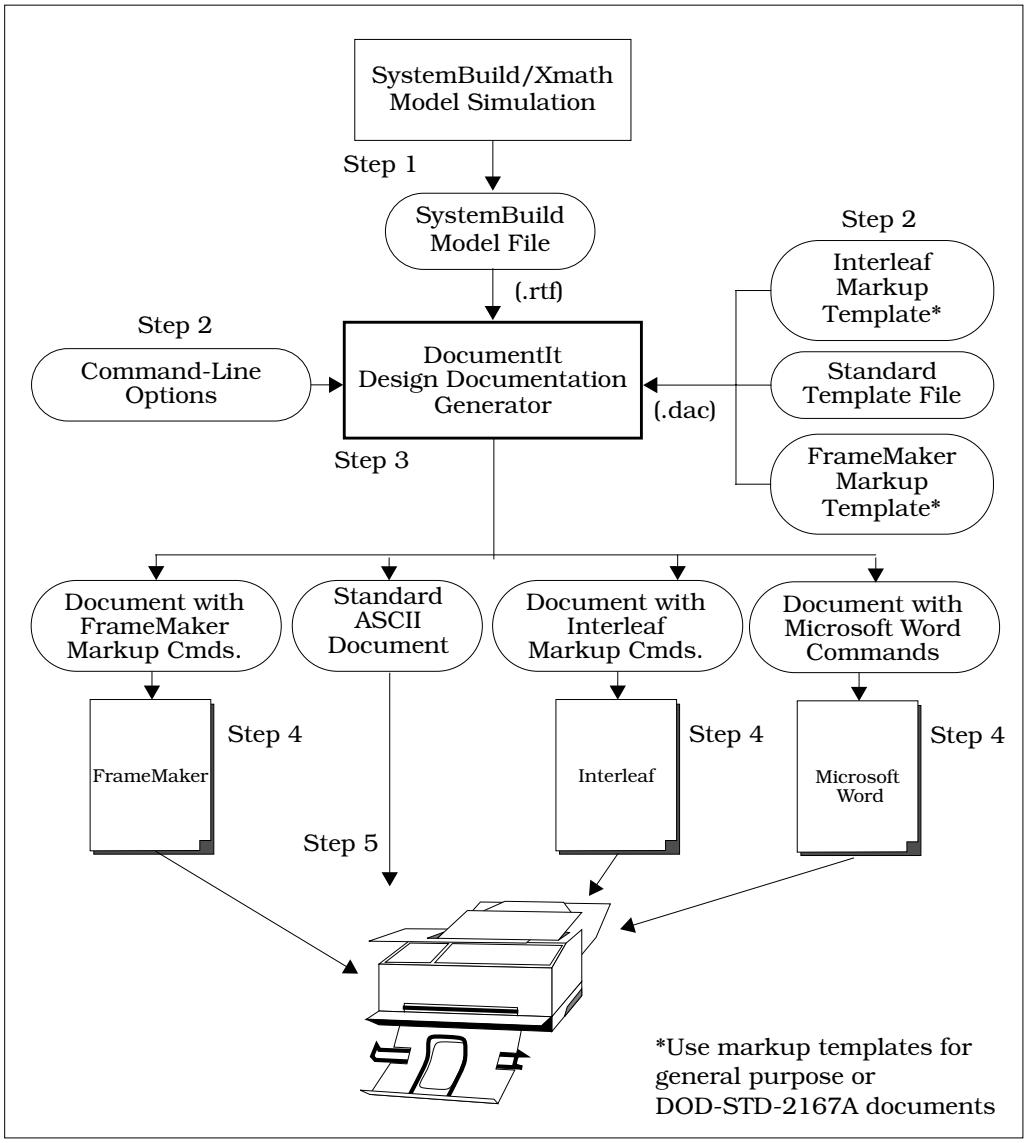


FIGURE 1-2 DocumentIt Automatic Documentation Generation Process

7. Customize documentation generation.

   Tailor your generated design document using the template programming language (TPL) provided in DocumentIt. This programming language lets you implement virtually any template file for a specialized purpose. DocumentIt output in this case is an ASCII text file, which can be used by almost any text processing product. You can edit the file with the host editor or print it as a simple unformatted ASCII file.

   Alternatively, if used with text publishing software, the template file not only contains DocumentIt parameters, but can also contain publishing software commands such as rich text format (RTF) or FrameMaker markup language (MML), which DocumentIt writes directly to the output file. The DocumentIt output ASCII file can then be used as input to the text publishing software for automatic formatting.

8. Generate the design document.

   Invoke DocumentIt from inside SystemBuild, from the Xmath Commands window, or from the operating system command line. DocumentIt loads discrete-time and continuous-time SuperBlocks, reading the associated data to generate the design documentation in accordance with the template instructions.

9. Process documents with markup commands.

   Import DocumentIt document files with FrameMaker, Interleaf, or RTF markup commands into FrameMaker, Interleaf, or Microsoft Word, which automatically formats the document.

10. Print the formatted document.

    Save the model design document in RTF, FrameMaker (MML), or ASCII format, and send it to a printer for hard copy output.

1.3 Cruise Control Example

This section provides an abbreviated illustration of the use of DocumentIt. Figure 1-3 on page 1-6 shows a modified version of the cruise control model. This model is found in the following location:

UNIX	`$CASE/DIT/templates/fmaker/general/Controller_Logic_gen.dat`
Windows	`%CASE%\case\DIT\templates\fmaker\general\Controller_Logic_gen.dat`

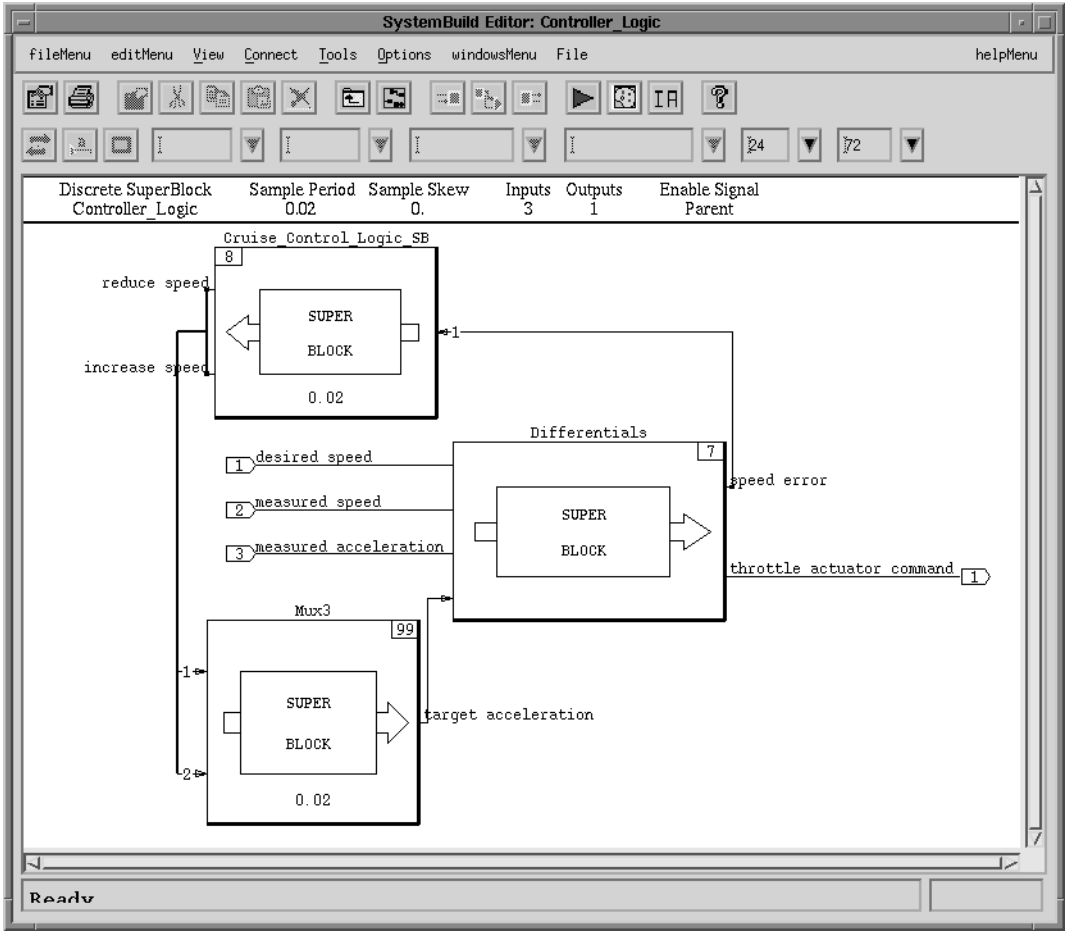


FIGURE 1-3 Cruise Control SystemBuild Diagram

The cruise control model consists of the highest-level SuperBlock, two embedded SuperBlocks, and associated units. Figure 1-4 shows the associated SuperBlock Properties dialog. Click **HELP** for information on each field in the dialog.

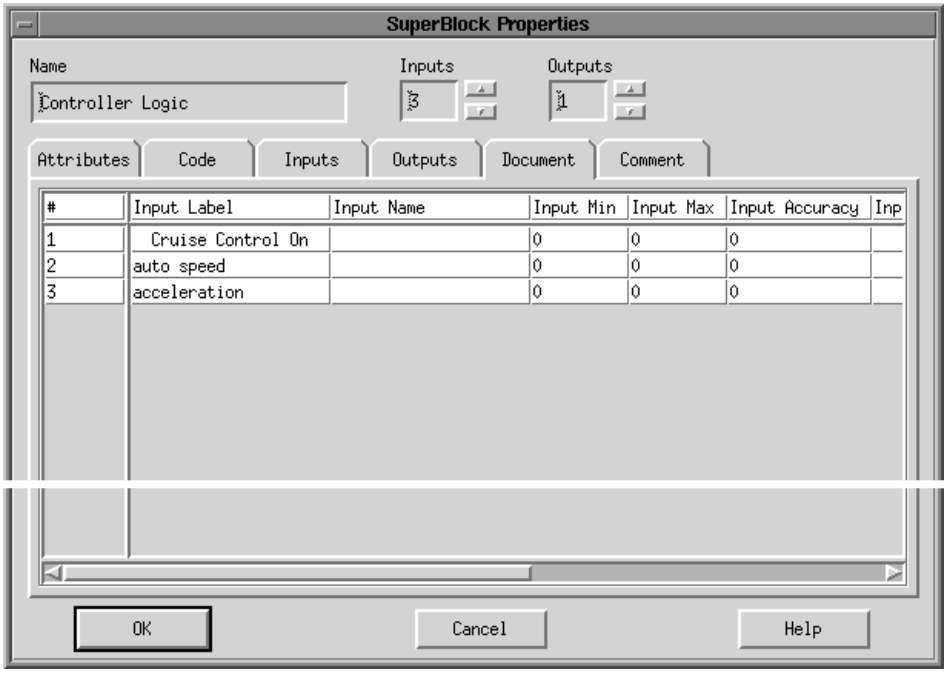


FIGURE 1-4 SuperBlock Properties for Cruise Control Diagram

The information from the SuperBlock Properties dialog can be included in the DocumentIt ASCII output file by using the appropriate keywords in the template file. The excerpt from a template file shown in Example 1-1 produces entries in the DocumentIt ASCII output file shown in Example 1-2.

EXAMPLE 1-1: Template file excerpt

```
@SEGMENT MAIN() STRING S1 @
Software Design Document

@SCOPE SUPERBLOCK 0@

Overview : @S1=user_param("OVERVIEW_s", "SUPERBLOCK")@@S1@
Architecture: @user_param("ARCHITECTURE_s", "SUPERBLOCK")@@S1@
System States and Models: @S1=user_param("SYSTEM_STATES_AND_MODES_s",
 "SUPERBLOCK")@@S1@

Num Ext. In          = @num_sb_in_i@
Num Ext. Out         = @num_sb_out_i@
```

```
Num Super Blocks     = @num_blks_in_sb_i@
Attribute            = @sb_attr_s@
Frequency            = @sb_freq_r@

Has Input Data = @sb_has_in_data_b@
@ENDSEGMENT@
```

EXAMPLE 1-2: DocumentIt ASCII Output File

```
==================================
Software Design Document


Overview :  This cruise controller regulates vehicle speed around a set point.
This cruise controller has three external inputs as follows:

the desired "speed" or velocity of the vehicle,

the current "speed,"

the current "acceleration" of the vehicle.

This cruise controller has one external output, which is the throttle
actuator command.


Architecture:  There are three SuperBlocks in this model.

The SuperBlocks are as follows:

the "Differentials" SuperBlock computes the difference between
desired and measured velocity and acceleration, adds computed target
acceleration to measured acceleration, and outputs a servo-limited
throttle command based upon these differences

the "Cruise Control Logic" SuperBlock determines whether or not
throttle position should be altered, based upon the difference
between actual and desired vehicle velocity, and

the "Mux3" SuperBlock determines the target acceleration based upon
inputs from the control logic.


System States and Models:  This model is enabled and disabled
externally.
```

```
Num Ext. In          = 3
Num Ext. Out         = 1
Num SuperBlocks      = 3
Attribute            = Discrete
Frequency            = 50.0000

Has Input Data = 1
```


Similarly, information from all SuperBlock/block dialogs and input/output description dialogs can be included within an ASCII output file by the appropriate use of the keywords in the template file.

2 Invoking DocumentIt

This chapter tells how to generate documents from SystemBuild, the Xmath Commands window, and from the operating system command line.

2.1 How to Generate Design Documentation

Using DocumentIt, you can generate design documentation from:

- SystemBuild, which lets you automatically generate a real-time file (`.rtf`) and then extract block and SuperBlock documentation from a model, using a Graphical User Interface. For ease of use, this is the recommended method of documentation generation.
- Xmath, which lets you automatically generate an `.rtf` and then extract block and SuperBlock documentation from a model, using an Xmath command. See Appendix A for the Xmath command options.
- The operating system command line, which lets you extract block and SuperBlock documentation from an already-existing `.rtf`, using the `autostar` command from the operating system prompt. See Appendix A for the operating system command-line options.

2.1.1 Generating Documentation From Within SystemBuild

To use DocumentIt while inside SystemBuild,

1. Select a Top-Level SuperBlock. By default, the name of the SuperBlock you select is used for both the RTF file and the documentation output file.
2. Select Tools→DocumentIt from the Catalog Browser to open the Generate Documentation dialog.

3. Select your options. The available options include:

Block Parameters	Select the source of the `%Var` information. The `%Vars` from Xmath option uses the latest values of the Xmath variables. Block Defaults uses the block's default values.
Typecheck	If selected, the Analyzer checks for matching data types for input and output signals. AutoCode generates the appropriate data types in the target language that correspond to the SystemBuild data types. If not selected, input and output signals and parameter data are assumed to be `FLOAT`. In this case, DocumentIt only generates the `RT_FLOAT` data type for signals and data. Default = 0.
Template File	Specify the DocumentIt template `.tpl` or `.dac file` to be used to format the documentation. The standard ASCII template is used by default.
Config File	File containing other DocumentIt command-line options.
MS Word	Enable special prcessing for Microsoft Word RTF file generation. This option is required if you are using the Microsoft Word template as described in *Generating Documents Using Microsoft Word*.

DocumentIt status is displayed in the Xmath Commands window log area.

Platform Specific Differences

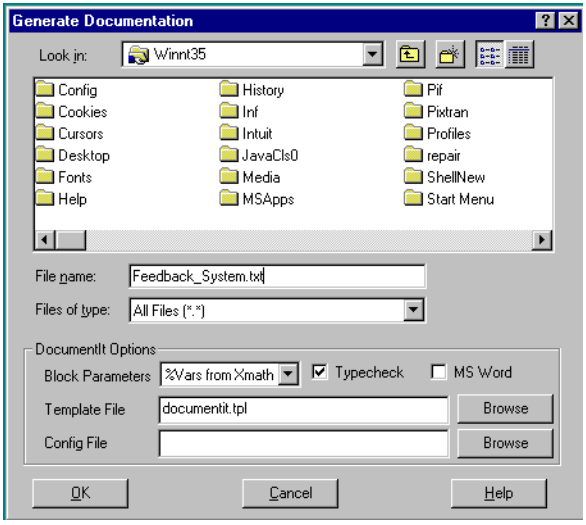
The DocumentIt Generate Documentation dialog has some platform specific features.

Windows

SystemBuild uses the standard Windows selection dialog to display an output file name in the **File name** field.

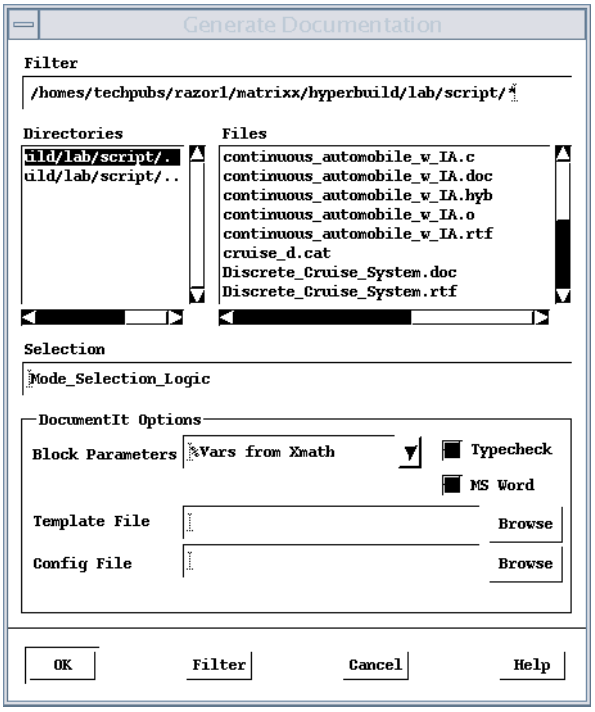
If you want more information on standard portions of the dialog, click on the question mark (?) icon on the dialog's title bar, and then position the question mark cursor over a field or icon and click. The Windows help for that field will be displayed.

Depending on the template file used, the document can be either an ASCII text file or an ASCII file with embedded word processor markup language formatting commands. These commands are word processor specific. Example templates are provided to support FrameMaker, Interleaf, and rich text format (RTF), used by Microsoft Word (and read by other programs).



UNIX

The Generate Documentation dialog uses a standard Motif file selection dialog to specify the name of the output documentation file in the Selection field.



Filter Displays the file selection directory path. Uses standard shell wildcard character(s) to select the files shown in the Files listing. Default value is the current directory with the `*.cat` filter. Pressing **Return** after changing the filter path updates the listings in Directories and Files.

Directories Lists the directories within the current directory. Double-click the directory name to change the current directory.

Files Lists the files in the selected directory that meet the filter criteria. If the rest of the dialog is filled out to your satisfaction, double-click on a file to save to that file name.

Selection Displays the complete path to the file selected in the Files listing. Because this name is used for both the RTF file and the documentation file, any extensions will be stripped before creating the `.rtf` and source file names.

Template Information

Appendix B through Appendix E provide examples of automatic documentation generation using DocumentIt templates, as follows:

- Appendix B describes how to generate a general purpose design document using DocumentIt and FrameMaker templates.
- Appendix C describes how to generate a sample design document conforming closely with DOD-STD-2167A using DocumentIt and FrameMaker templates.
- Appendix D describes how to generate a sample design document generate a sample design document conforming closely with DOD-STD-2167A using DocumentIt and Interleaf templates.
- Appendix E describes how to generate a sample design document using DocumentIt and Microsoft Word templates.
- Appendix F describes how to generate a sample design document in ASCII text file format using a DocumentIt template.

2.1.2 Generating Documentation from the Xmath Commands Window

The `documentit` command lets you process a model to generate design documentation.

Two syntaxes are supported:

```
documentit, {model = name1, file = name2, template=name3}

documentit model, {options}
```

where `name1` identifies the model to be processed for documentation generation. The model can be either:

- A string (in "quotes"), which must be the name of a SuperBlock that exists in the current SystemBuild Catalog. This SuperBlock is analyzed and processed to generate documentation.

- A variable (not in quotes). Variables should be assigned to a string, the string must be the name of a SuperBlock in the current catalog; it is analyzed and processed to generate documentation.

Whenever a file name or other string is included in a command string, it must be enclosed in quotes, but a variable name must *not* be in quotes.

`name2` is the name for the generated output document file. This value must be a string or string variable as well. If the `file=name2` option is not provided, documentation will be generated in the file `mode1.doc`.

`name3` is the name of the template file used to tailor the document. This value must be a string or string variable. If this option is not provided, the template `_documentit.tpl` (in the directory `$CASE/DIT/templates/ascii`) is used.

See Appendix A for a complete list of DocumentIt options (used in the second command syntax).

Examples:

```
documentit "topSB"
```

The system generates a real-time file named *topSB*`.rtf`. It loads this file and processes it to produce the document file. The output file name is *topSB*`.doc`.

```
documentit "topSB", {template="mytemplate"}

documentit, {model = "topSB", template = "mytemplate"}
```

Either syntax processes the SuperBlock *topSB* in the current catalog to produce documentation, using the template file `mytemplate`. The output file name is *topSB*`.doc`.

2.1.3 Generating Documentation from the OS

If a model file already exists, it is also possible to execute DocumentIt from the operating system prompt. The file intended for processing must be a real-time file (`.rtf`). At the operating system prompt, execute the command:

```
% autostar {options} model_file.rtf
```

Many of the options are the same as the fields in the documentation generation dialog. See Appendix A for a complete list of DocumentIt options.

DocumentIt runs, creating a document file. When the operating system prompt returns, the process is complete.

Examples:

% autostar -h

shows a help display.

% autostar -doc -t FM.tpl SysBld_file.rtf

processes the model file `SysBld_file.rtf` using a template file named `FM.tpl` to produce a document file named `SysBld_file.doc`. It assumes the template file named `FM.tpl` exists in your working directory.

NOTE: The DocumentIt real-time file (`.rtf`) input file is not the same as the rich text format file (`.rtf`) word processing output file. Only the file extension is the same.

3 Customizing the Generated Documentation

You can customize DocumentIt-generated output documentation to suit your specific needs by using templates. Template files are ASCII files containing text, interspersed with template command parameters that specify DocumentIt output.

In addition to template command parameters, template files can also use publishing software markup commands (for example, Microsoft Word RTF, Interleaf markup, and FrameMaker MML commands), 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.

For details about how to use TPL for DocumentIt output, see the *Template Programming Language User's Guide*.

A DocumentIt Options

This appendix provides additional information about invoking DocumentIt. Use this appendix together with Chapter 2, *Invoking DocumentIt*.

A.1 Options When Invoking DocumentIt

As described in Chapter 2, DocumentIt can be invoked from the Catalog Browser, the Xmath Commands window, or the operating system command line. Table A-1 lists the various DocumentIt command-line options.

From the Catalog Browser, select Tools→DocumentIt

From the Xmath Commands window, enter the command (see Section 2.1.2):

`documentit` *`options`*

From the operating system command line, enter the command (see Section 2.1.3):

`autostar -doc` *`options`*

TABLE A-1 Command-Line Options When Invoking DocumentIt

Option from Xmath	Option from O/S	Description
`file`	`-o`	A string defining the output file for generated documentation. If no name is supplied, the top level SuperBlock name is used with the `.doc` extension.
See note [a] on page A-3.	`-h`	Obtains a help display.

TABLE A-1 Command-Line Options When Invoking DocumentIt (Continued)

Option from Xmath	Option from O/S	Description
`model`	See note [b] on page A-1.	A text string representing the name of a SuperBlock in the SystemBuild Editor (e.g., "System").
`msrtf`	`-msw1`	Provides special parsing for Microsoft Word documentation generation. Ignores the `\par` parameter in the rich text format (RTF) when you use `@@` and merges a word when it is divided across two lines.
`options`	`-opt`	A string specifying the name of the options file. Options are entered in the file using the same syntax as if they were specified in an operating-system call to AutoCode (outside Xmath), except that options normally passed as strings must not be quoted. Command-line options override all the options in this file, and the RTF file name cannot be given here. Single line comments are done by using `//` characters. See Section A.2 for more information about the options file.
`rtf`	See note [c] on page A-3.	A string specifying the `RTF` file name. You can use the `RTF` file along with the standalone AUTOCODE command to regenerate code. The `RTF` file is generated using the CREATERTF command.
`tpldac`	`-d`	A string specifying the location of the template `dac` file to be used in document generation. Default file is `$XMATH/../case/DIT/templates/ascii/documentit.dac` (the code template `dac`).
`tplsrc`	`-t`	A string specifying the location of the template file to be used in document generation. The default file is `$XMATH/../case/DIT/templates/ascii/documentit.tpl` (the code template).

TABLE A-1 Command-Line Options When Invoking DocumentIt (Continued)

Option from Xmath	Option from O/S	Description
`typecheck`	See note [a] on page A-3.	Boolean (default=1). When TRUE, this option enables the checking of variable types for the SystemBuild analyzer. Typechecking is always done by default.
`vars`	See note [a] on page A-3.	Boolean (default=1). When TRUE, this flag forces %variables in the Xmath workspace to be used. When FALSE, SystemBuild default block form values will be used.

a. Xmath does not have a `help` keyword, but has the command **`help autocode`** for Netscape help.

b. This Xmath option is used for creating the `.rtf` file. When invoking from the O/S, the `.rtf` file must already exist; therefore, there is no O/S option equivalent.

c. The name of the `.rtf` file must always be specified when invoking from the command line.

A.2 Using the `autostar.opt` File

If you invoke DocumentIt with the same options consistently, you can put these options into an options file, saving a lot of error-prone, repetitive typing each time you invoke Documentit. DocumentIt reads the options file at startup, and performs the options as though you had entered them on the command line. Although you can use an options file whether you invoke DocumentIt from the Xmath Commands window or the operating system command line, the only options that you can specify in the options file are operating system command-line options.

The default options file is `autostar.opt`. If you have an `autostar.opt` file in the current working directory from which you invoke DocumentIt, the options in that file will be executed when you invoke DocumentIt. If you specify an option on the command line that is also in the options file, the command-line option overrides the same option in the options file (see Example A-1 and the paragraphs following).

For different applications, you might need to invoke DocumentIt differently. For this reason, you can have multiple options files. To invoke DocumentIt with an options file other than `autostar.opt`, specify the name of the options file when you invoke DocumentIt (see Example A-2 and the paragraphs following).

Options are entered in the options file using the same syntax as if they were specified in the command line. The exception is that map specifications are not enclosed between quotes. Options can be on one line, separate lines, or a combination. The RTF file name cannot be specified in the options file. Single line comments are preceded and followed by // characters.

Example A-1 shows an options file.

EXAMPLE A-1: Example `autostar.opt` Options File

```
// Sample options file //
-t c386_c860_mb2.tpl
-o myoutput
```

To use this file, invoke DocumentIt as follows:

```
autostar -doc model.rtf
```

This invokes DocumentIt with the `autostar.opt` options file.

Consider the following command:

```
autostar -doc -o myoutput3 model.rtf
```

The options file is again used, but the output file option (`-o`) is specified on the command line, so it overrides the corresponding command in the options file. The output documentation will be in file `myoutput3`, not in file `myoutput` as specified in the options file.

Example A-2 shows an options file called `myopt.opt`.

EXAMPLE A-2: Example Options File Called `myopt.opt`

```
// Sample options file //
-t c386_c860_mb2.tpl
-o myoutput2
```

To use this file, invoke DocumentIt as follows:

```
autostar -doc -opt myopt.opt model.rtf
```

This invokes DocumentIt with the `myopt.opt` options file.

So, if you have both of the above option files (shown in Examples A-1 and A-2) in your directory, invoking autostar without the `-opt` option puts the generated documentation into file `myoutput` (the `autostar.opt` options file is used). Invoking autostar with the `-opt myopt.opt` option as shown above puts the generated documentation into file `myoutput2` (as specified in the `myopt.opt` file).

B Generating Documents Using FrameMaker

This appendix provides instructions for using the FrameMaker documentation example. This includes describing how to:

- Generate documents and encapsulated PostScript files.
- Import encapsulated PostScript files into a generated document automatically and manually.
- Format table data for FrameMaker.

All FrameMaker documentation example files are located in:

```
$CASE/DIT/templates/fmaker/general
```

B.1 Using Your Own Templates

If you want to generate documentation using your own templates, you might consider copying the supplied example files and then modifying them. The purpose of each file is given in Table B-1 on page B-2. FrameMaker markup language (MML) commands are listed in Table B-2 on page B-3.

TABLE B-1 FrameMaker Example Files

Filename	Description
`fmgp.tpl`	This template file determines what information will be extracted from the model to create an ASCII output file. Additionally, `fmgp.tpl` embeds FrameMaker MML commands, which define how the document will be formatted when it is imported into FrameMaker. Finally, the template also embeds an MML command that calls the include file `fmgpinc.mml`, which defines what MML commands FrameMaker should recognize from the FrameMaker template `fmgp.doc`.
`fmgpinc.mml`	This include file specifies what MML commands the FrameMaker template file `fmgp.doc` should recognize. However, the actual format definitions of these commands are not specified in this include file; rather, the format specifications for each MML command are incorporated into `fmgp.doc`. All MML commands listed in this include file are given in Figure B-2. The table also gives a brief description of each MML command as it is defined by `fmgp.doc`.
`fmgp.doc`	This is a supplied FrameMaker template into which you must import an ASCII data file generated by DocumentIt using the DocumentIt template `fmgp.tpl`.
`fmgpTOC.doc`	This FrameMaker template receives the table of contents you generate from `fmgp.doc`. It is important to consider that FrameMaker normally generates a TOC file automatically from a document file (but no format is specified). The purpose of this file is to provide a suitable format for the TOC, rather than requiring you to develop one.
`Controller_Logic_gen.dat`	Controller logic model.

TABLE B-2 FrameMaker MML Commands

Command	Description
<Author>	Command on the title page; formats the name of the author providing the document (see Figure B-2).
<body1>	Normal text paragraph format.
<body2>	Indented text paragraph which can be use to align with the text in a numbered list (first level number).
<body3>	Indented text paragraph which can be use to align with the text in a numbered list (second level number).
<bull1>	First level bulleted list.
<bull2>	Second level bulleted list.
<chap>	Chapter title.
<CoBody>	Name of the company providing the document given on the title page (see Figure B-2).
<Date>	Title page entry for the document date (see Figure B-2).
<DocTitle>	Document title (see Figure B-2).
<EXample>	Numbered title of an example (automatic numbering).
<EXinit>	Numbered title of an example (reset count to 1).
<EXtext>	Text format for programming example.
<FIGcap>	Automatically numbered figure caption.
<head1>	First heading level.
<head2>	Second heading level.
<head3>	Third heading level.
<head4>	Fourth heading level.
<head5>	Fifth heading level.
<list1>	Numeric list (automatic numbering).
<list1init>	Numeric list (automatic numbering reset to 1).
<list2>	Alpha list (automatic numbering).
<list2init>	Alpha list (automatic numbering reset to a).
<NOTE>	Note paragraph with the word NOTE included.
<NOTE+>	Indented paragraph that aligns with the text in the NOTE paragraph (the word NOTE is not included). This is used when a note has a second paragraph.
<Rev>	Revision of the document on the title page (see Figure B-2).
<TBLcap>	Automatically numbered table caption.

B.2 Generating a Sample Document

To generate a sample document using DocumentIt and FrameMaker, follow these steps:

NOTE: If you want to generate a document from your own model, follow the same steps given in this example, but use your own model for document generation.

1. Create a directory in which you want the output document to reside.
2. Copy all files from:

   ```
   UNIX:      $CASE/DIT/templates/fmaker/general
   Windows:    %CASE%\DIT\templates\fmaker\general
   ```

   to the document directory.
3. From the Catalog Browser File menu, select Load.
4. Browse through the document directory and double-click to load the file `Controller_Logic_gen.dat`.
5. In the Catalog pane of the Catalog Browser, select the top-level SuperBlock (in this case, Controller_Logic).
6. From the Catalog Browser, select Tools→DocumentIt. The Generate Documentation dialog appears.
7. In the **File name** field, enter `Controller_Logic.mml`.

   NOTE: An ASCII file with MML markup commands must have a file extension of `.mml` to be formatted by FrameMaker.
8. With the **Block Parameters** combo box, select **%Vars from Xmath**.
9. In the **Template File** field, specify `fmgp.tpl` as the template file.
10. Select the **Typecheck** check box.
11. To generate the MML document file, click **OK**.

B.2.1 Changing the Generic Title Page

The MML file that you have generated (in this case, `Controller_Logic.mml`) includes a generic title page that appears as shown in Figure B-1 after it is imported into FrameMaker. If you want to change the data on the title page, you can accomplish this in three different ways:

- Using a text editor you can edit the `fmgp.tpl` data. Figure B-2 shows the data in the `fmgp.tpl` file that produces the title page shown in Figure B-1. You should make these changes to `fmgp.tpl` *before* you generate the document from SystemBuild. All subsequent documents generated by DocumentIt using `fmgp.tpl` will continue to have your new title page (until you change the contents of `fmgp.tpl` again).
- Using a text editor you can edit the MML output file you have just generated. Figure B-2 shows the data in the example file `Controller_Logic.mml` that produces the title page shown in Figure B-1. If you regenerate the document, you will have to edit this file again.
- If you are familiar with FrameMaker, you can edit the title page after you import the data into FrameMaker.

If your MML output files contain *less than* (<) or *greater than* (>) characters from your model, you must use your text editor to place a *backslash* (\) character in front of these characters. This is required because these characters will otherwise be interpreted as invalid MML commands when the `*.mml` file is imported into FrameMaker.

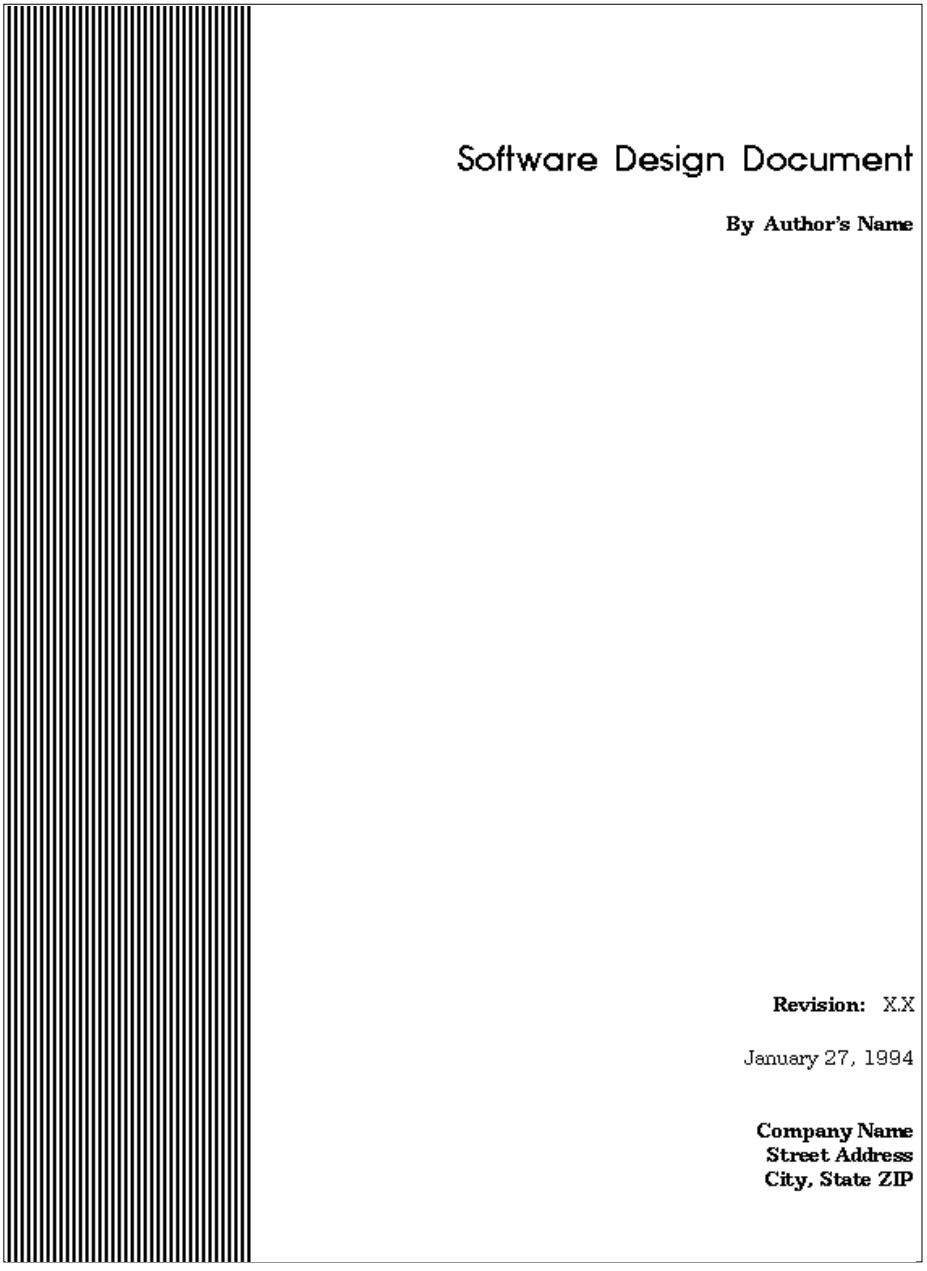


FIGURE B-1 Generic Title Page

```
<DocTitle>
Software Design Document
<Author>
Author's Name
<Rev>
X.X
<Date>
November 17, 1998
<CoBody>
Company Name

Street Address

City, State ZIP
```


FIGURE B-2 Title Page Data in `fmgp.tpl` and `controller_logic_gen.mml`

B.2.2 Generating an `.eps` file from SystemBuild

1. To place an illustration in your document, you must insert an anchored frame where you want the figure to appear. Two anchored frames are inserted into this example so that you can see how they work. For details on `.eps` files, refer to the *Template Programming Language User's Guide*. You can insert an anchored frame in your document in one of the following ways:

   - Using a text editor, insert `@include_img()@` to automatically import the illustration when you generate the document.

     You must edit the `.eps` file generated from SystemBuild in the text editor and change the first line as follows:

     `%!PS-ADOBE-1.0`
     to
     `%!PS-ADOBE-3.0 EPSF-3.0`

   - Using a text editor, you can insert the TPL function `@small_frame()@` (5" x5" frame) or `@large_frame()@` (7" x5" frame) in `fmgp.tpl` where you want the figure to appear. You should make these changes to `fmgp.tpl` *before* you generate the document from SystemBuild. After the document is processed and imported into FrameMaker, a blank frame will be placed into

your document in the location you specified. Refer to your FrameMaker documentation to use the *capture* feature to capture and import the figure that you want into the anchored frame.

- Using a text editor you can insert the MML command

  ```
  <AFrame <BRect 0 0 h" w">>
  ```

  into the ASCII output file you have just generated (where the variables `h` and `w` are the height and width of the frame, respectively). After the document is imported into FrameMaker, a blank frame will be placed into your document in the location you specified. Refer to your FrameMaker documentation to use the *capture* feature to capture and import the figure that you want into the anchored frame.

- If you are familiar with FrameMaker, you can create an anchored frame and import the figure you want directly into your FrameMaker document.

  i. Highlight the top level SuperBlock (Controller_Logic).

  ii. Select File→Page Setup to make any adjustments to the image before creating the graphics file. In this case, accept the defaults.

  iii. Select File→Print to File.

  iv. From the format pull-down, select **EncapPostScript**.

  v. From the output pull-down, select **Separate files**.

  vi. Click **OK**.

NOTE: If the `.eps` illustration is too large for the anchored frame, use the FrameMaker Graphics Scale option to adjust the size.

2. From the system prompt in the `document` directory, start FrameMaker.
3. From FrameMaker, open `fmgp.doc`. You will find a blank file, with the exception of vertical lines on the left side of the page.
4. From the pull-down menu, select File→Import.
5. Select **Copy File into Document**.
6. Select `Controller_Logic.mml` and click **OK**.

7. After the import is complete, you will still have a blank page, except for a single paragraph text symbol at the top of the page somewhat hidden in the top portion of the vertical lines). Select the paragraph symbol and delete it.

   The title page of the *Software Design Document* fills the page. (Do not be concerned if the vertical lines do not remain intact; they will be restored as soon as you page through the document.)

NOTE: Do not save this file until you are told to do so.

B.2.3 Putting Information in Table Format

1. Page through the document to ensure it meets your needs. If you have information that you want to be in *table* format, follow the steps below, beginning with a. If you have no tables, skip to step 2 on page B-10.

   a. Select all the data you want to include in the table, but do not include the last paragraph symbol in the last row of table data (see Figure B-3 for an example of exactly what information you need to select for reformatting).

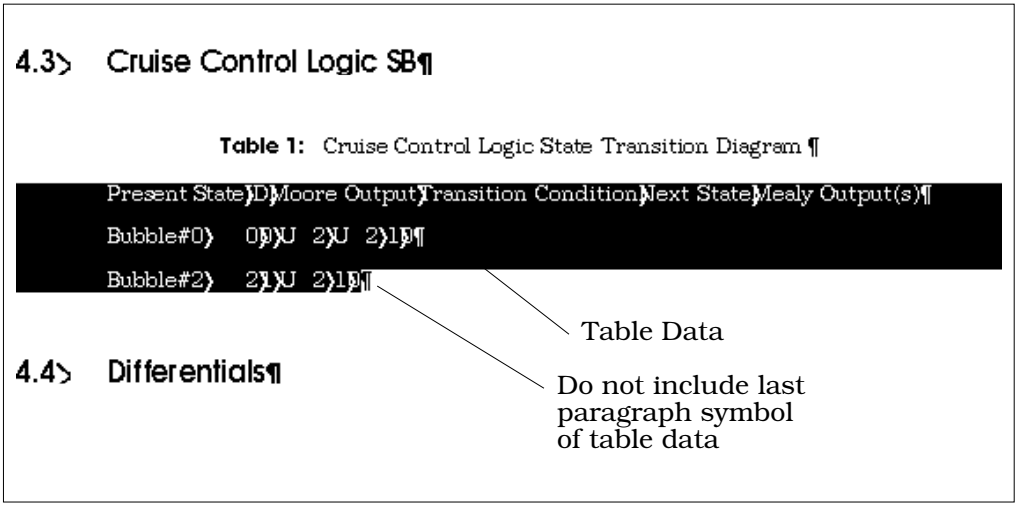


FIGURE B-3 Marking Data for Table Formatting

   b. From the pull-down menu, select Table→Convert to Table. When the Convert to Table dialog appears, leave the settings intact and click **OK**.

   c. From the pull-down menu, select Table→Resize Columns. When the Resize Columns dialog appears, click on **By Scaling to Widths Totalling**; change the value

to 6.25" and click **OK**. Make sure that the whole table is selected when you do this. At this point you can resize individual columns if necessary.

You can also click and drag to highlight one or more table columns. Click to grab the left column handles, and pull out or push in to resize the column.

d. Delete any extraneous paragraph symbols that may have appeared below the table during the conversion process (this is necessary due to the way FrameMaker converts tables).

e. Perform steps `a` through `d` above for each table you have in the document.

2. From the pull-down menu, select File→Generate.

3. Click **List Table of Contents**. Then, click **Generate**.

4. Leave all the current settings in the Set Up Table of Contents dialog intact and click **OK**.

5. The `fmgpTOC.doc` file now appears on the screen with a complete list of sections and subsections. However, since FrameMaker lists all items in the order in which they appear in a document, some minor formatting is required for the list of figures and tables. Complete these tasks as follows:

a. Page down through the TOC. Using the Cut and Paste features under the Edit pull-down menu, put all figure references at the end of the Table of Contents. Do the same thing for all table references, so they follow the figures.

b. At the first *figure* entry (now near the end of the Table of Contents), put the cursor in front of the `Figure 1` entry.

c. Type in the word "Figures" and press **ENTER**.

d. Click once on the word "Figures."

e. Click the Paragraph Catalog symbol in the upper right corner of the page to display the Paragraph Catalog. In the catalog, select the paragraph type `TabFigTC`.

The heading for the list of figures is now formatted correctly.

f. Repeat this process for the list of tables.

6. Select the `fmgp.doc` frame. From the File pull-down menu, select `Save As` and assign a new file name to your document file.

7. (Optional) To print the document, select File→Print.

8. Select the `fmgpTOC.doc` frame. From the File pull-down menu, select `Save As`. You must specify the identical path and file name as you did in the previous step for your new document file, only append "TOC" to the file name before the extension (see the example that follows).

Original File Name	→	New File Name
`fmgp.doc`	→	`FMsample1.doc`
`fmgpTOC.doc`	→	`FMsample1TOC.doc`

   This scheme allows you to generate TOCs in the future (when you modify your document) without having to construct a new TOC template.

9. You now have a new document and your original templates have remained intact.

NOTE: See *Using Your Own Templates* on page B-1 for information about using your templates instead of the supplied templates.

C Generating 2167A Documents Using FrameMaker

This appendix describes how to use the FrameMaker 2167A example. This includes describing how to:

- Generate documents and encapsulated PostScript files.
- Import encapsulated PostScript files into a generated document automatically and manually.
- Format table data for FrameMaker.

All FrameMaker 2167A example files are located in:

`$CASE/DIT/templates/fmaker/milstd` (Sun)

C.1 Using Your Own Templates

If you want to generate documentation using your own templates, you might want to consider copying the supplied example files and then modifying them. The purpose of each file is given in Table C-1 on page C-2. FrameMaker markup language (MML) commands are listed in Table C-2 on page C-3.

TABLE C-1 FrameMaker 2167A Example Files

Filename	Description
`fmmil.tpl`	This template file determines what information will be extracted from the model to create an ASCII output file. Additionally, `fmmil.tpl` embeds FrameMaker MML commands, which define how the document will be formatted when it is imported into FrameMaker. Finally, the template also embeds an MML command that calls the include file `fmmilinc.mml`, which defines what MML commands FrameMaker should recognize from the FrameMaker template `fmmil.doc`.
`fmmilinc.mml`	This include file specifies what `MML` commands the FrameMaker template file `fmmil.doc` should recognize. However, the actual format definitions of these commands are not specified in this include file; rather, the format specifications for each MML command are incorporated into `fmmil.doc`. All MML commands listed in this include file are given in Table C-2. The table also gives a brief description of each MML command as it is defined by `fmmil.doc`.
`fmmil.doc`	This is a supplied FrameMaker template into which you must import an ASCII data file generated by DocumentIt using the DocumentIt template `fmmil.tpl`.
`fmmilTOC.doc`	This FrameMaker template receives the table of contents you generate from `fmmil.doc`. It is important to consider that FrameMaker normally generates a TOC file automatically from a document file (but no format is specified). The purpose of this file is to provide a suitable format for the TOC, rather than requiring you to develop one.

TABLE C-2 FrameMaker 2167A MML Commands

Command	Description
<Author>	Last command on the title page; formats the name of the company providing the document (see Figure C-2).
<body1>	Normal text paragraph format.
<body2>	Indented text paragraph which can be use to align with the text in a numbered list (first level number).
<body3>	Indented text paragraph which can be use to align with the text in a numbered list (second level number).
<bull1>	First level bulleted list.
<bull2>	Second level bulleted list.
<CDRL>	Title page CDRL sequence number (see Figure C-2).
<chap>	Chapter heading level.
<Client>	Title page entry for the name of the client for who the document was prepared (see Figure C-2).
<contract>	Title page entry for the contract number (see Figure C-2).
<DocTitle>	Document title (see Figure C-2).
<EXample>	Numbered title of an example (automatic numbering).
<EXinit>	Numbered title of an example (reset count to 1).
<EXtext>	Text format for programming example.
<FIGcap>	Automatically numbered figure caption.
<head1>	First heading level (e.g., 1.1).
<head2>	Second heading level (e.g., 1.1.1).
<head3>	Third heading level (e.g., 1.1.1.1).
<head4>	Fourth heading level (e.g., 1.1.1.1.1).
<head5>	Fifth heading level (e.g., 1.1.1.1.1.1).
<list1>	Numeric list (automatic numbering).
<list1init>	Numeric list (automatic numbering reset to 1).
<list2>	Alpha list (automatic numbering).
<list2init>	Alpha list (automatic numbering reset to a).
<NOTE>	Note paragraph with the word NOTE included.
<NOTE+>	Indented paragraph that aligns with the text in the NOTE paragraph (the word NOTE is not included). This is used when a note has a second paragraph.
<PrepBy>	Title page entry (see Figure C-2).
<PrepFor>	Title page entry (see Figure C-2).
<Rev>	Revision and date entry on the title page (see Figure C-2).
<TBLcap>	Table caption.

TABLE C-2 FrameMaker 2167A MML Commands (Continued)

Command	Description
<TitlePar1>	Title page entries (see Figure C-2).
<TitlePar2>	Title page CSCI name and system name entries (see Figure C-2).

C.2 Generating the 2167A Document

To generate the sample 2167A document using DocumentIt and FrameMaker, follow these steps:

NOTE: If you want to generate a document from your own model, follow the same steps given in this example, but use your own model for document generation.

1. Create a directory in which you want the output document to reside.
2. Copy all files from:

   ```
   UNIX:       $CASE/DIT/templates/fmaker/general
   Windows:    %CASE%\DIT\templates\fmaker\general
   ```

   to the document directory.
3. From the Catalog Browser File menu, select Load.
4. Browse through the document directory and double-click to load the file `Controller_Logic_mil.dat`.
5. In the Catalog pane of the Catalog Browser, select the top-level SuperBlock (in this case, Controller_Logic).
6. From the Catalog Browser, select Tools→DocumentIt. The Generate Documentation dialog appears.
7. In the File Name field, enter `Controller_Logic_mil.mml`.

   NOTE: An ASCII file with MML markup commands must have a file extension of `.mml` in order to be formatted by FrameMaker.
8. With the **Block Parameters** combo box, select **%Vars from Xmath**.
9. In the **Template File** field, specify `fmgp.tpl` as the template file.
10. Click the **Typecheck** check box.

11. To generate the MML document file, click **OK**.

C.2.1 Changing the Generic Title Page

The MML file that you have generated (in this case, `controller_logic_mil.mml`) includes a generic title page that appears as shown in Figure C-1 after it is imported into FrameMaker. If you want to change the data on the title page, you can accomplish this in three different ways:

- Using a text editor you can edit the `fmmil.tpl` data. Figure C-2 shows the data in the `fmmil.tpl` file that produces the title page shown in Figure C-1. You should make these changes to `fmmil.tpl` *before* you generate the document from SystemBuild. All subsequent documents generated by DocumentIt using `fmmil.tpl` will continue to have your new title page (until you change the contents of `fmmil.tpl` again).
- Using a text editor you can edit the MML output file you have just generated. Figure C-2 shows the data in the example file `controller_logic_mil.mml` that produces the title page shown in Figure C-1. If you regenerate the document, you will have to edit this file again.
- If you are familiar with FrameMaker, you can edit the title page after you import the data into FrameMaker.

NOTE: If your output files contain *less than* (<) or *more than* (>) characters from your model, you must use your text editor to place a *backslash* (\) character in front of these characters. This is required because these characters will otherwise be interpreted as invalid MML commands when the `*.mml` file is imported into FrameMaker.

FIGURE C-1 Generic Title Page

```
<Rev>
Revision X.X: September 9, 1999
<DocTitle>
Software Design Document
<TitlePar1>
For The
<TitlePar2>
CSCI NAME
<TitlePar1>
Of
<TitlePar2>
System Name
<contract>
XXXXXXX
<CDRL>
XXXXXXXXXX
<PrepFor>
Prepared for:
<Client>
Contracting Agency Name, Department Code
<PrepBy>
Prepared by:
<Author>
Integrated Systems, Inc.
```

FIGURE C-2 Title Page Data in `fmmil.tpl` and `controller_logic_mil.mml`

C.2.2 Generating an `.eps` file from SystemBuild

To place an illustration in your document, you must insert an anchored frame where you want the figure to appear. Two anchored frames are inserted into this example so that you can see how they work. For details on `.eps` files refer to the *Template Programming Language User's Guide*. You can insert an anchored frame in your document in one of the following ways:

- Using a text editor, insert `@include_img()@` to import the illustration automatically when you generate the document.

You must edit the `.eps` file generated from SystemBuild in the text editor and change the first line as follows:

```
%!PS-ADOBE-1.0
to
%!PS-ADOBE-3.0 EPSF-3.0
```

- Using a text editor, you can insert the tpl function

  `@small_frame()@` (5" x5" frame) or `@large_frame()@` (7" x5" frame)

  in `fmgp.tpl` where you want the figure to appear. You should make these changes to `fmgp.tpl` *before* you generate the document from SystemBuild. After the document is processed and imported into FrameMaker, a blank frame will be placed into your document in the location you specified. Refer to your FrameMaker documentation to use the *capture* feature to capture and import the figure that you want into the anchored frame.

- Using a text editor you can insert the `MML` command `<AFrame <BRect 0 0 h" w">>` into the ASCII output file you have just generated (where the variables `h` and `w` are the height and width of the frame, respectively). After the document is imported into FrameMaker, a blank frame will be placed into your document in the location you specified. Refer to your FrameMaker documentation to use the *capture* feature to capture and import the figure that you want into the anchored frame.

- If you are familiar with FrameMaker, you can create an anchored frame and import the figure you want directly into your FrameMaker document.

  a. Highlight the top level SuperBlock (Controller_Logic).

  b. Select File→Page Setup to make any adjustments to the image before creating the graphics file. In this case, accept the defaults.

  c. Select File→Print to File.

  d. From the format pull-down, select **EncapPostScript**.

  e. From the output pull-down, select **Separate files**.

  f. Click **OK**.

  NOTE: If the `.eps` illustration is too large for the anchored frame, use the FrameMaker Graphics Scale option to adjust the size.

The following message appears in the SystemBuild menu bar to confirm that a hard copy .eps file was generated:

```
5 SuperBlock(s) hardcopied into file Controller_Logic.eps
```

1. From the system prompt in the `document` directory, invoke FrameMaker.
2. From FrameMaker open `fmmil.doc`. You will find a blank file.
3. From the pull-down menu, select File→Import.
4. Click on Copy File into Document.
5. Select `Controller_Logic_mil.mml` and click **OK**.
6. After the import is complete, you will still have a blank page except for a single paragraph text symbol at the top of the page. Select the paragraph symbol and delete it. The *Software Design Document* title page will fill the page.

NOTE: Do not save this file until you are prompted to do so.

C.2.3 Putting Information in Table Format

Page through the document to ensure it meets your needs. If you have information that you want to be in *table* format, follow the steps below, beginning with `a`. If you have no tables, skip to step 7 on page C-10.

1. Select all the data you want to include in the table, but do not include the last paragraph symbol in the last row of table data (see Figure C-3 for an example of exactly what information you need to select for reformatting).

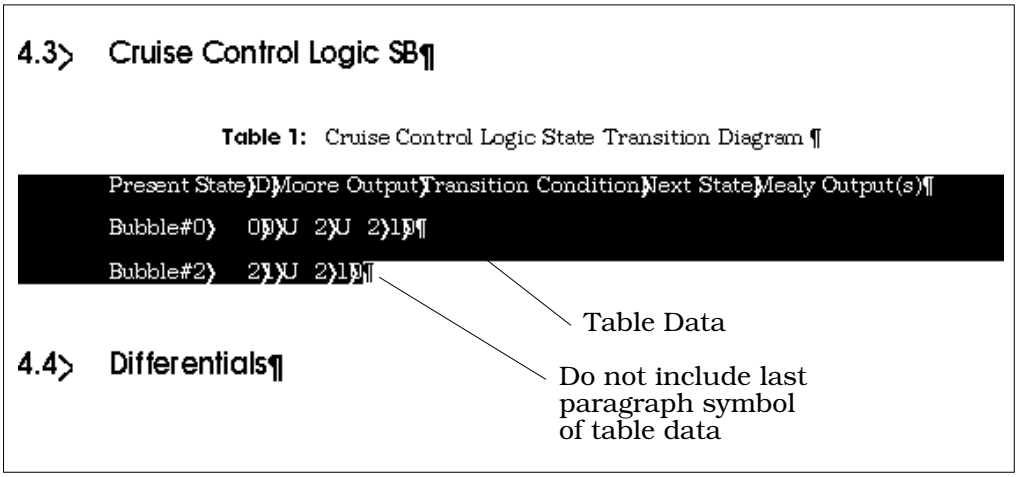


FIGURE C-3 Marking Data for Table Formatting

2. From the pull-down menu, select Table→Convert to Table. When the Convert to Table dialog appears, leave the settings intact and click **OK**.

3. From the pull-down menu, select Table→Resize Columns. When the Resize Columns dialog appears, click on **By Scaling to Widths Totalling**; change the value to 6.25" and click **OK**. Make sure that the whole table is selected when you do this. At this point you can resize individual columns if necessary.

4. You can also click and drag to highlight one or more table columns. Click the left mouse to grab the left column handles, and pull out or push in to resize the column.

5. Delete any extraneous paragraph symbols that may have appeared below the table during the conversion process (this is necessary due to the manner in which FrameMaker converts tables).

6. Perform the above steps for each table you have in the document.

7. From the File pull-down menu, select Generate.

8. Click **List Table of Contents**, and then click **Generate**.

9. Leave all the current settings in the Set Up Table of Contents dialog intact and click **OK**.

10. The file `fmmilTOC.doc` now appears on the screen with a complete list of sections and subsections. However, since FrameMaker lists all items in the order in which they appear in a document, some minor formatting is required for the list of figures and tables. Complete these tasks as follows:

    a. Page down through the TOC. Using the `Cut` and `Paste` features under the Edit pull-down menu, put all figure references at the end of the Table of Contents. Do the same thing for all table references, so they follow the figures.

    b. At the first *figure* entry (now near the end of the Table of Contents), put the cursor in front of the `Figure 1` entry.

    c. Type in the word "Figures" and press **Enter**.

    d. Click once on the word "Figures."

    e. Click on the Paragraph Catalog symbol in the upper right corner of the page (this will display the Paragraph Catalog). In the catalog, click on the paragraph type `TabFigTC`. The heading for the list of figures is now formatted correctly.

    f. Repeat this process for the list of tables.

11. Select the `fmmil.doc` frame. From the File pull-down menu, select `Save As` and assign a new file name to your document file. If you wish, you can print the file at this time.

12. Select the `fmmilTOC.doc` frame. From the File pull-down menu, select `Save As`. Now, you must specify the identical path and file name as you did in the previous step for your new document file, only append "TOC" to the file name before the extension (see the example that follows).

Original File Name	→	**New File Name**
`fmmil.doc`	→	`FMsample1.doc`
`fmmilTOC.doc`	→	`FMsample1TOC.doc`

    This scheme allows you to generate TOCs in the future (when you modify your document) without having to construct a new TOC template.

13. You now have a new document and your original templates have remained intact. You can close your files now if you want.

NOTE: See *Using Your Own Templates* on page C-1 for information about using your templates instead of the supplied templates.

D

Generating 2167A Documents Using Interleaf

Here are instructions for using the Interleaf 2167A example. This tutorial tells how to:

- Generate documents and encapsulated PostScript files.
- Import epsf files into a generated document automatically and manually.
- Format table data for MS Word.

All Interleaf 2167A example files are located in:

`$CASE/DIT/templates/ileaf/milstd` (Sun)

D.1 Using Your Own Templates

If you want to generate documentation using your own templates, you might want to consider copying the supplied example files and then modifying them. The purpose of each file is given in Table D-1 on page D-2. Interleaf markup commands are listed in Table D-2 on page D-2.

TABLE D-1 Interleaf 2167A Example Files

Filename	Description
`ilmil.tpl`	This template file determines what information will be extracted from the model to create an ASCII output file. Additionally, `ilmil.tpl` embeds Interleaf markup commands, which define how the document will be formatted when it is opened under Interleaf. Finally, the template also embeds a markup command that calls the include file `ilmilinc`, which defines how the document will be formatted by Interleaf.
`ilmilinc`	This include file specifies the markup commands that Interleaf will use to format the ASCII output file generated by DocumentIt. All markup commands listed in this include file are given in Figure D-2. The table also gives a brief description of each command as it is defined by `ilmilinc`.
`Controller_Logic_mil.dat`	Controller logic model.

TABLE D-2 Interleaf Markup Commands

Command	Description
<"Author">	Command on the title page that formats the name of the company providing the document (see Figure D-2).
<"body1">	Normal text paragraph format.
<"body2">	Indented text paragraph which can be use to align with the text in a numbered list (first level number).
<"body3">	Indented text paragraph which can be use to align with the text in a numbered list (second level number).
<"CDRL">	Title page CDRL sequence number (see Figure D-2).
<"chap">	Chapter heading level.
<"Client">	Title page entry for the name of the client for who the document was prepared (see Figure D-2).

TABLE D-2 Interleaf Markup Commands

Command	Description
<"contract">	Title page entry for the contract number (see Figure D-2).
<"DocTitle">	Document title (see Figure D-2).
<"dummy">	First paragraph on the title page. This is an empty place holder required to provide the correct spacing from the top of the page.
<"FIGcap">	Automatically numbered figure caption.
<"head1">	First heading level (e.g., 1.1).
<"head2">	Second heading level (e.g., 1.1.1).
<"head3">	Third heading level (e.g., 1.1.1.1).
<"head4">	Fourth heading level (e.g., 1.1.1.1.1).
<"head5">	Fifth heading level (e.g., 1.1.1.1.1.1).
<"list1">	Upper case Alpha list (automatic numbering).
<"list2">	Numeric list (automatic numbering).
<"list3">	Lower case alpha list (automatic numbering).
<"NOTE">	Note paragraph with the word NOTE included.
<"NOTE+">	Indented paragraph that aligns with the text in the NOTE paragraph (the word NOTE is not included). This is used when a note has a second paragraph.
<"PrepBy">	Title page entry (see Figure D-2).
<"PrepFor">	Title page entry (see Figure D-2).
<"Rev">	Revision and date entry on the title page (see Figure D-2).
<"row">	Table row.
<"TBLcap">	Table caption.
<"TBLhead">	Column headings in a table.
<"TitlePar1">	Title page entries (see Figure D-2).
<"TitlePar2">	Title page CSCI name and system name entries (see Figure D-2).

D.2 Generating the 2167A Document

To generate the sample 2167A document using DocumentIt and Interleaf, follow these steps:

NOTE: If you want to generate a document from your own model, follow the same steps given in this example, but use your own model for document generation.

1. From your system prompt, go to your Interleaf desktop directory (do not invoke Interleaf) using the `change directory` command.
2. Copy all files from:

   ```
   UNIX:     $CASE/DIT/templates/fmaker/general
   WIN95/NT:  %CASE%\DIT\templates\fmaker\general
   ```

   to the document directory.
3. From the Catalog Browser File menu, select Load.
4. Browse through the document directory and double-click to load the file `Controller_Logic_mil.dat`.
5. In the Catalog pane of the Catalog Browser, select the top-level SuperBlock (in this case, Controller_Logic).
6. From the Catalog Browser, select Tools→DocumentIt. The Generate Documentation dialog will appear.
7. In the File Name field, enter `Controller_Logic_mil.asc`.
8. With the **Block Parameters** combo box, select **%Vars from Xmath**.
9. In the **Template File** field, specify `ilmil.tpl` as the template file.
10. Click the **Typecheck** check box.
11. To generate the ASCII document file, click **OK**.

D.2.1 Changing the Generic Title Page

12. The ASCII file that you have generated (in this case, `controller_logic_mil.asc`) includes a generic title page that appears as shown in Figure D-1 after it is opened from Interleaf. If you want to change the data on the title page, you can accomplish this in three different ways:

    - Using a text editor you can edit the `ilmil.tpl` data. Figure D-2 shows the data in the `ilmil.tpl` file that produces the title page shown in Figure D-1. You should make these changes to `ilmil.tpl` *before* you generate the document from SystemBuild. All subsequent documents generated by DocumentIt using `ilmil.tpl` will continue to have your new title page (until you change the contents of `ilmil.tpl` again).

    - Using a text editor you can edit the ASCII output file you have just generated. Figure D-2 shows the data in the example file `controller_logic_mil.asc` that produces the title page shown in Figure D-1. If you regenerate the document, you will have to edit this file again.

    - If you are familiar with Interleaf, you can edit the title page after you have opened the file from Interleaf.

13. From the system prompt, invoke Interleaf.

14. From Interleaf open `Controller_Logic_mil.asc`.

15. After the file is opened, you will see that it is completely formatted.

16. If the new document meets your specifications, you can save and print it.

NOTE: See *Using Your Own Templates* on page D-1 for information about using your templates instead of the supplied templates.

You can use the DocumentIt template `ilmil.tpl` and its associated include file `ilmilinc` in precisely the same manner for your own models to generate a document using Interleaf.

To generate documentation using your own templates, copy the supplied example files and modify them. The purpose of each file is given in Table D-1 on page D-2.

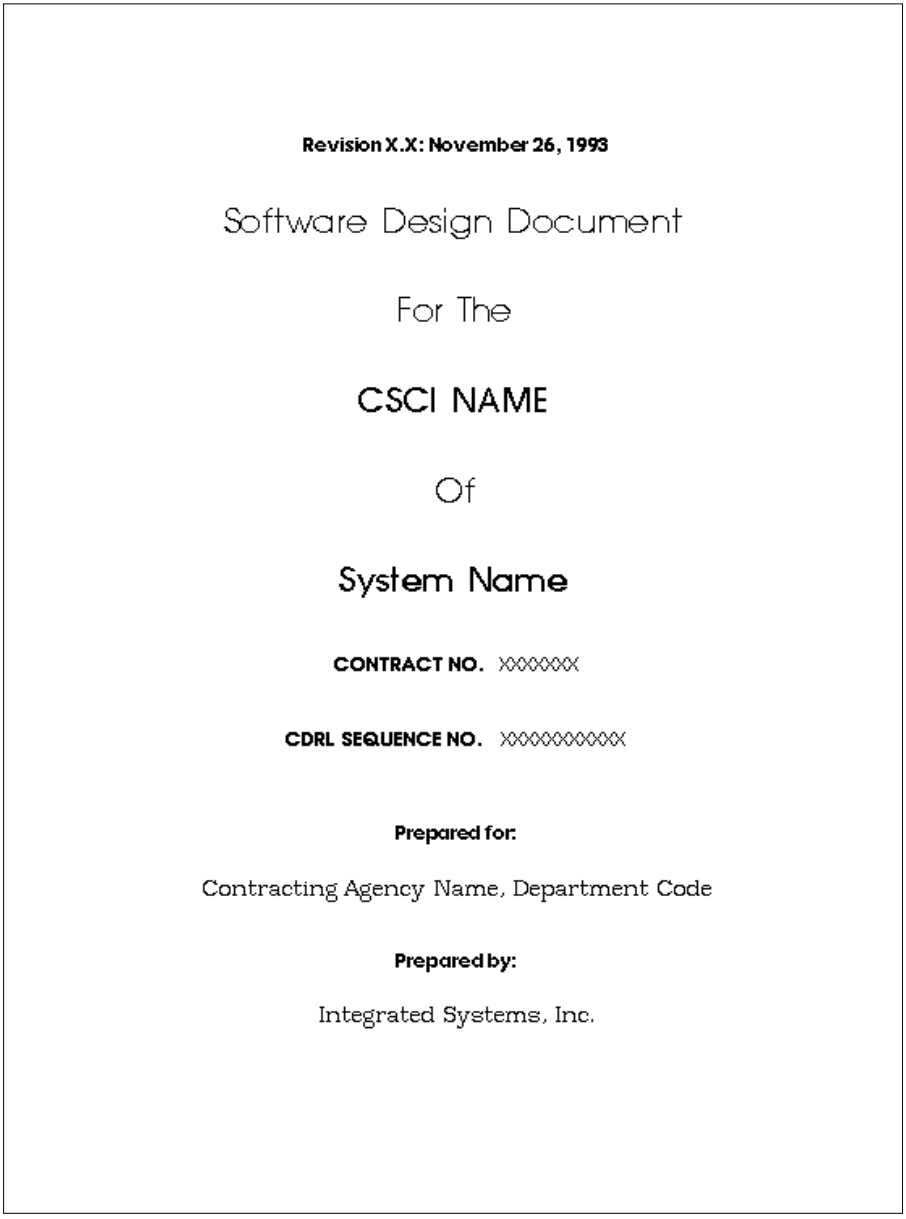


FIGURE D-1 Generic Title Page

```
<“Rev”>
 Revision X.X: November 26, 1993

<“DocTitle”>
 Software Design Document

<“TitlePar1”>
 For The

<“TitlePar2”>
 CSCI NAME

<“TitlePar1”>
 Of

<“TitlePar2”>
 System Name

<“contract”>
 XXXXX.XX

<“CDRL”>
XXXXXX.nnn

 <“PrepFor”>
 Prepared for:

<“Client”>
Contracting Agency Name, Department Code

<“PrepBy”>
 Prepared by:

<“Author”>
 Integrated Systems, Inc.
```

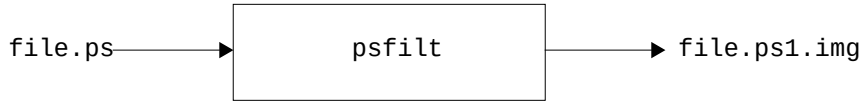
FIGURE D-2 Title Page Data in `ilmil.tpl` and `controller_logic_mil.doc`

D.2.2 Adding a Plot or SystemBuild Diagram to an Interleaf Document

17. To place a plot or SystemBuild diagram in your Interleaf document, perform the following procedure:

    a. Use the `hardcopy` command to create a postscript file of each plot or SystemBuild diagram you want to include.

    b. You must then convert each postscript file to an Interleaf image file. The following diagram illustrates the conversion process (using `psfilt`) for each postscript file.



    Run each postscript file through the Interleaf `psfilt` filter by selecting custom→misc→psfilt.

    NOTE: If the `psfilt` filter is not set up, consult your Interleaf manual for instructions on how to set it up.

18. Use the `include_img()` segment to add the image to your generated document. (Consult Chapter 3 of the TPL User's Guide for information about `include_img()`.) Perform the following procedure:

    a. At the bottom of the `$ISIHOME/`*`platform`*`/case/DIT/templates/ileaf/milstd/ilmil.tpl` template file is a `SEGMENT include_img()` line. Example D-1 on page D-9 shows a portion of this `SEGMENT` section (details of the frame itself are omitted).

    Include this `SEGMENT` section in your own tpl file.

EXAMPLE D-1: Example of `include_img()` in tpl File

```
@SEGMENT include_img()@

<Frame,

    Name =                      “floating”,
    Placement =                 Following Anchor,
    Horizontal Alignment =      Center,
    Width =                     6.50 Inches,
    Width =                     Page Without Margins,
    Height =                    9 Inches,
    Height =                    Page Without Margins,
    Page # =                    Autonum,
    Diagram =

V11,
(g9,1,0,
(i18,1,0,,0,0,8.5333325,0,0,10.9999992,0,0,2559,3299,1,0,7,0,0,8.5333325,10.999
 9992,673657859,1,2559,0,0,3299,0,0,2559,3299,0,0,0,0,0,2559,3299,1,255,
 @sb_mangled_name_s.ps1.img,2,639,0,0,824,0,0,2559,3299,0,0,0,0,0,486,674,1,1,
3157  (X0,3157,FBC8FCFCFCFCFCFCFCFCFCFCFCFCFCFCFCFCFCFCFCFCFCFCFCFCFCFCFCFCFCFCF
CFCFC

                                        .     (rest
                                        .      of
                                        .      frame)

   FC91F9C706070EC7010AFCA1E50114FCA1EB0128FCA1F029FC9AC9E12D71FC9AC9DE5D79FCA1
   CAC1C896DFE4FF))
(E16,0,0,,5,1,1,0.0533333,1,15,0,0,1,0,0,0,1,5,127,7,0,0,7,0,1,1,0.0666667,0.06
 66667,6,6,0,0.0666667,6))>

@ENDSEGMENT@
```

b. Scope on a SuperBlock. The `@include_img()@` statement in the `SEGMENT` section will include the currently scoped SuperBlock's image file in your generated document.

If you wish, you can customize the image in the `include_img()` segment (see Example D-1). Consult your Interleaf manual. You should be familiar with Interleaf markup language.

c. To incorporate a plot, make a copy of the `include_img()` segment (Example D-1), and replace `@sb_name_s@.psl.img` (highlighted in Example D-1) with the name of your plot image file.

E Generating Documents Using Microsoft Word

This appendix describes how to use the Microsoft® Word documentation example. This includes describing how to:

- Generate documents and encapsulated PostScript™ files.
- Import encapulated PostScript (`.eps`) files into a generated document automatically and manually.
- Format table data for Microsoft Word.

All Microsoft Word documentation example files for Windows platforms, including a sample file that demonstrates these tasks, are located in the `%CASE%\DIT\templates\msword\general\mswgp.wrd` file.

E.1 Using Your Own Templates

To customize your generated document, make a copy of this file in your local directory and make appropriate changes. You must regenerate your document, every time you make changes to `mswgp.tpl` and `mswgp.wrd`.

If you want to generate documentation using your own templates, consider copying the supplied example files and then modifying them. The purpose of each file is given in Table E-1 on page E-2.

NOTE: Your templates must be in rich text format (RTF), with TPL commands included. Do *not* make a template in Microsoft Word format.

The `mswgp.tpl` and `mswgp.wrd` files are in the `%CASE%\DIT\templates\msword\general` directory. The `Controller_Logic_gen.dat` file is in the `%CASE%\DIT\templates\ascii` directory.

TABLE E-1 Microsoft Word Documentation Example Files

Files in %CASE%\DIT\templates\msword\general:	
mswgp.tpl	The template file you must specify when using DocumentIt to generate an ASCII output file for use with the Word general purpose documentation. It determines what information is extracted from the model to create an ASCII output file.
mswgp.wrd	This is a template file included in the mswgp.tpl file. This file specifies RTF commands, which are recognized by Word. It contains the template TPL command to search the model for SuperBlocks and Blocks, and has RTF commands embedded with template TPL commands to generate the output document. You can load this file into Word to change the template program or font settings for the title/section names. The changes appear when you regenerate the document. Load this document into MS Word, to see what the generated document will look like.
Files in %CASE%\DIT\templates\ascii:	
Controller_Logic_gen.dat	Controller logic model.

E.2 Generating a Sample Document

To generate a sample document using DocumentIt and Microsoft Word, follow these steps:

NOTE: If you want to generate a document from your own model, follow the same steps given in this example, but use your own model for document generation.

1. Create a directory where you want the output document to reside.
2. Copy all files from:

   %CASE%\DIT\templates\msword\general

   to the document directory.

3. From the Catalog Browser select File→Load.
4. Browse through the document directory and double-click to load the file `Controller_Logic_gen.dat`.
5. In the Catalog pane of the Catalog Browser, select the top-level SuperBlock (in this case, Controller_Logic).
6. From the Catalog Browser, select Tools→DocumentIt. The Generate Documentation dialog appears.
7. In the **File name** field, enter `Controller_Logic.doc`. This output file is created in your working directory unless you specify another path.
8. With the **Block Parameters** combo box, select **%Vars from Xmath**.
9. In the **Template File** field, specify `msgwp.tpl` as the template file.
10. Click the **Typecheck** check box.
11. Click the **MS Word** check box.
12. To generate the document file, click **OK**.

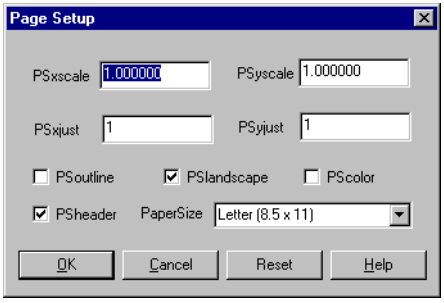
If you have any graphics files to include in the document, follow the steps below, beginning with Section E.2.1. If you have no graphics to include, skip to step 1 on page E-5.

NOTE: After you generate a document file, you may need to or want to make format changes to select desired fonts, font sizes, and other typographic choices.

E.2.1 Generating an `.eps` file from SystemBuild

To generate an `.eps` file from SystemBuild, take these steps:

1. Highlight the top level SuperBlock (Controller_Logic).
2. Select File→Page Setup to make any adjustments to the image before creating the graphics file. In this case, accept the defaults.



3. Select File→Print to File.
4. From the format pull-down, select **EncapPostScript**.
5. From the output pull-down, select **Separate files**.
6. Click **OK**.
7. By default DocumentIt automatically includes all the SuperBlock diagrams in the generated document. This is done by inserting template function `@include_img@` under every SuperBlock scope where the image is to be included. The template `mswgp.tpl` is written to handle only importing `.eps` files into the Word document. The generated document has links to the `.eps` files in your local directory. So every time you update the graphics file, it will be automatically reflected in your generated document. For details on `.eps` files refer to the *Template Programming Language User's Guide*.

`@include_img()@` is a template TPL segment defined in `mswgp.tpl`. This segment contains the RTF commands to include a graphics file named `@sb_mangled_name_s@.eps`. The `@sb_mangled_name_s@` token is filled in with the appropriate SuperBlock when you scope to a particular SuperBlock. For details on scoping SuperBlocks, refer to the *Template Programming Language User's Guide*.

A typical example to include SuperBlock 1 graphics into your document would look like this:

```
@SCOPE SUPERBLOCK 1@
@include_img()@
```

E.2.2 Changing the Generic Title Page

1. The document file that you have generated (in this case, `Controller_Logic.doc`) includes a generic title page that appears after it is imported into Microsoft Word (as shown in Figure E-1 on page E-6). If you want to change the data on the title page or change the template TPL commands, you can accomplish this in three different ways (see *Using Your Own Templates* on page E-1):

   - If you are familiar with Microsoft Word, load the file `mswgp.wrd`. Word interprets the RTF in this file and displays the template program commands along with chapter titles, section headings. you can edit the title page in the first page and save the document (after you open the document in Word). All subsequent documents generated by DocumentIt using `mswgp.wrd` will continue to have your new title page (until you change `mswgp.wrd` again).

   - Using a text editor you can edit the `mswgp.wrd` file. If you are familiar with RTF format, you can search for title generation and modify it there. You should make these changes to `mswgp.wrd` *before* you generate the document from SystemBuild. All subsequent documents generated by DocumentIt using `mswgp.wrd` will continue to have your new title page (until you change `mswgp.wrd` again).

   - Using a text editor you can edit the output document you have just generated. If you are familiar with RTF format, you can search for title generation and modify it there. If you regenerate the document, you will have to edit this file again.

2. Generate a new DocumentIt output file.

3. Start Microsoft Word.

4. From the `File` pull-down menu, select `Open`.

5. In the Open dialog, select `Controller_Logic.doc` and click **`OK`**. You can verify your changes in the new output file.

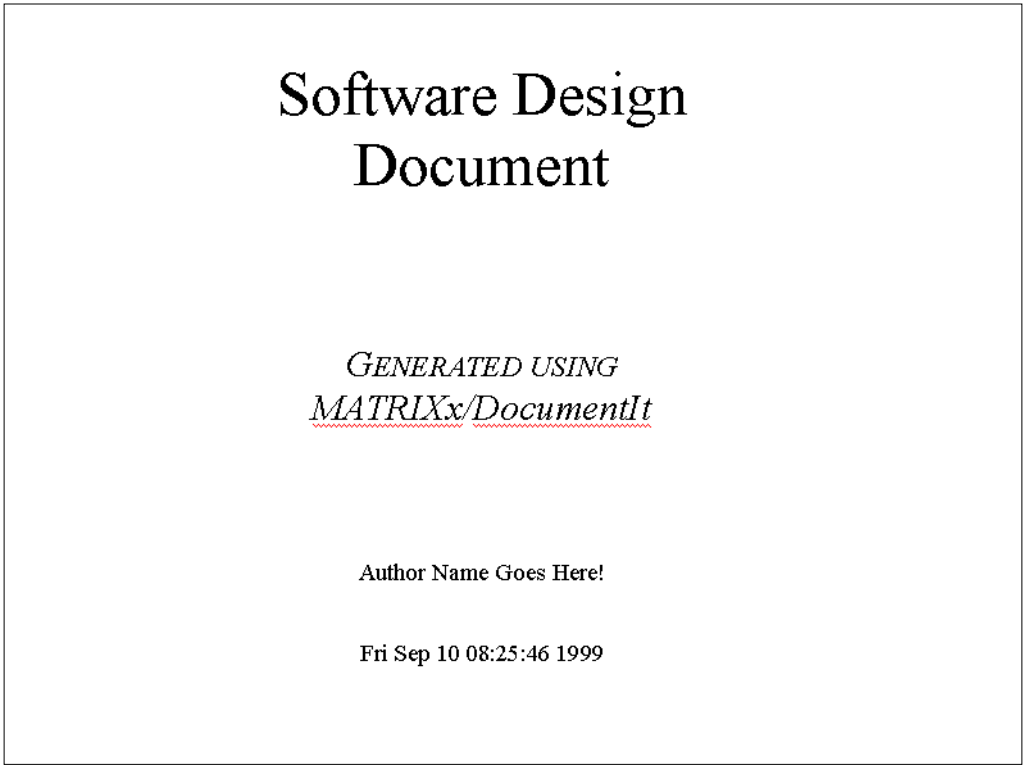


FIGURE E-1 Generic Title Page

If you have graphics files you want to include in the document, see *Importing Graphics*. If you have tabular information, see *Creating Tables*.

E.2.3 Importing Graphics

1. To place an illustration in your document manually, you must insert an anchored frame in the page where you want the figure to appear. You can insert an anchored frame in your document by selecting the Insert pull-down menu. A blank frame will be placed into your document in the location you specified.

2. Now you can select Picture under the Insert pull-down menu. There you can select the graphics file you want to import. Click **Link to File** in the dialog, to create a link to your graphics file. Refer to your Microsoft Word documentation for more information about using the Insert utility to insert and import pictures.

E.2.4 Creating Tables

You can use DocumentIt to format tabular data to open as formatted tables when opened in Microsoft Word. These tables display as "standard" Microsoft Word tables. To see an example, generate the example document, and then examine the `mswgp.wrd` template file using Microsoft Word. For information on how to create tables, see the procedure in the next section.

E.2.5 Converting Tabular Data to Tables

To set up tabular data so the DocumentIt template converts them to Microsoft Word tables, the input data must have Tab characters (>) separating columns and end-of-paragraph characters (¶) separating rows. Follow these steps to create a table.

1. Select all the data you want to include in the table, but do *not* include the last paragraph symbol in the last row of table data (see Figure E-2 for an example of exactly what information you need to select for reformatting).

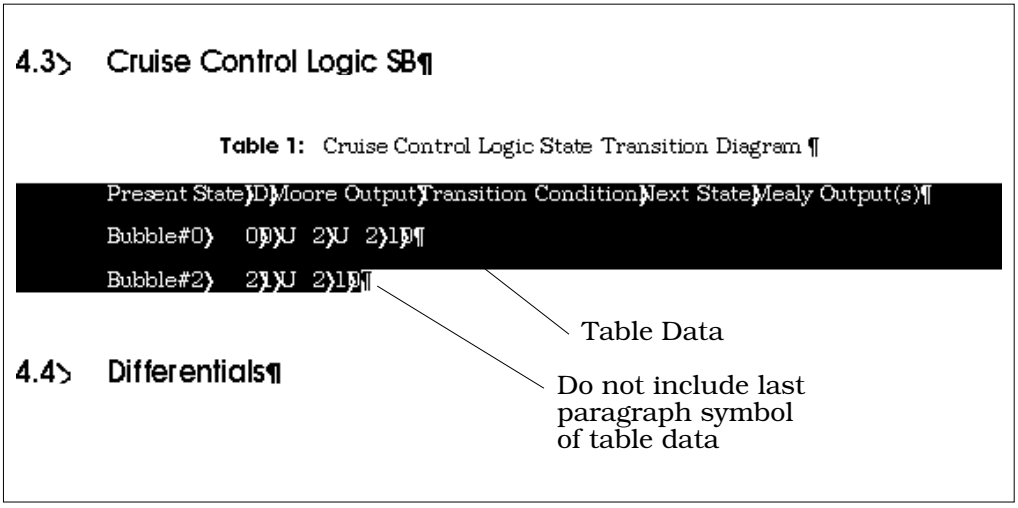


FIGURE E-2 Marking Data for Table Formatting

2. From the pull-down menu, select Table→Convert Text to Table. When the Convert Text to Table dialog appears, select **AutoFormat** and select "Grid 5" under that and then leave the settings intact and click **OK** for the AutoFormat and click **OK** for the Text to Table dialog. You can choose any design under the AutoFormat.

3. Delete any extraneous paragraph symbols that may have appeared below the table during the conversion process (this is necessary due to the way Microsoft Word converts tables).

4. Perform the above steps for each table you have in the document. If you are going to choose the same Format for all the tables, then follow step a, and select Repeat Text to Table under the Edit pull-down menu; this will repeat the macro for the table conversion.

5. Save your document.

E.2.6 Creating a Table of Contents

The DocumentIt template is unable to generate a table of contents (TOC); however, the template does create sections within the document so that it is easy to have Microsoft Word generate the TOC.

1. Go to the Table of Contents page in the document and place the cursor on that page.

2. From the pull-down menu, select Insert→Index and Tables. When the dialog appears, click **List Table of Contents**.

3. Leave all the current settings in the Set Up Table of Contents dialog intact and click **OK**.

   Microsoft Word fills up the page with the Table of Contents. As Word lists all items in the order in which they appear in a document, some minor formatting is required for the list of figures. Repeat this process for the list of tables.

4. From the pull-down menu, select File→Save to save the document.

F

Generating ASCII Documents

This appendix describes how to use the ASCII document example. All ASCII document example files are located in:

```
UNIX:     $CASE/DIT/templates/ascii/general
Windows:   %CASE%\DIT\templates\ascii\general
```


The files in this directory are:

`documentit.tpl`	This is the supplied default template file that is used by DocumentIt to generate an ASCII output file when no other template file is specified.
`Controller_Logic_gen.dat`	Controller logic model

NOTE: If you want to generate a document from your own model, follow the same steps given in the example that follows, but use your own model for document generation.

To generate the sample ASCII document using DocumentIt, follow these steps:

1. Create a directory for the output document (for example, `docout`).
2. Copy the template files from:

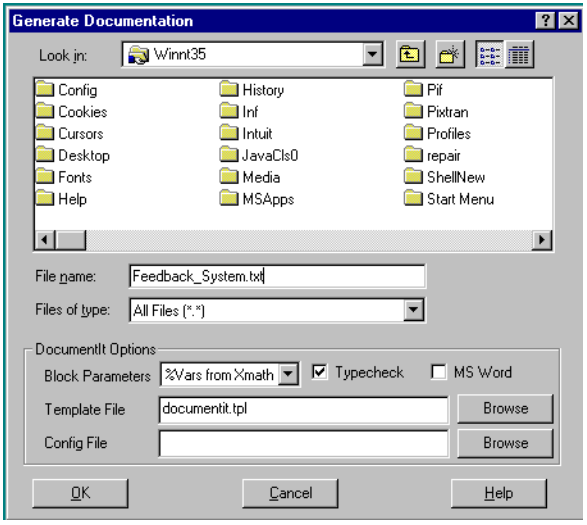
   ```
   UNIX:     $CASE/DIT/templates/ascii/general
   Windows:   %CASE%\DIT\templates\ascii\general
   ```

   to the `document` directory.
3. From the Catalog Browser File menu, select Load.

4. Browse through the document directory and double-click to load the file `Controller_Logic_gen.dat`.

5. In the Catalog pane of the Catalog Browser, select the top-level SuperBlock (in this case, Controller_Logic).

6. From the Catalog Browser, select Tools→DocumentIt. The Generate Documentation dialog appears.



7. In the **File name** field, enter a name for your document (for example, `Controller_Logic_gen.txt`).

8. With the **Block Parameters** combo box, select **%Vars from Xmath**.

9. In the **Template File** field, specify `documentit.tpl` as the template file.

10. Select the **Typecheck** check box.

11. To generate the ASCII document file, click **OK**.

You can use the default template `documentit.tpl` to generate an ASCII document for any valid model.

To generate documentation using your own templates:

- Copy `documentit.tpl` file.

- Modify it to your specifications using the TPL programming information described in the *Template Programming Language User's Guide*. You can also embed markup language commands to import the ASCII files into your software publishing application, or print the document.

Index

I

M

O

P

R

S

T

U

V

W

Technical Support and Professional Services

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

- **Support**—Online technical support resources include the following:
  - **Self-Help Resources**—For immediate answers and solutions, visit our extensive library of technical support resources available in English, Japanese, and Spanish at ni.com/support. These resources are available for most products at no cost to registered users and include software drivers and updates, a KnowledgeBase, product manuals, step-by-step troubleshooting wizards, conformity documentation, example code, tutorials and application notes, instrument drivers, discussion forums, a measurement glossary, and so on.
  - **Assisted Support Options**—Contact NI engineers and other measurement and automation professionals by visiting ni.com/support. Our online system helps you define your question and connects you to the experts by phone, discussion forum, or email.
- **Training**—Visit ni.com/training for self-paced tutorials, videos, and interactive CDs. You also can register for instructor-led, hands-on courses at locations around the world.
- **System Integration**—If you have time constraints, limited in-house technical resources, or other project challenges, NI Alliance Program members can help. To learn more, call your local NI office or visit ni.com/alliance.

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