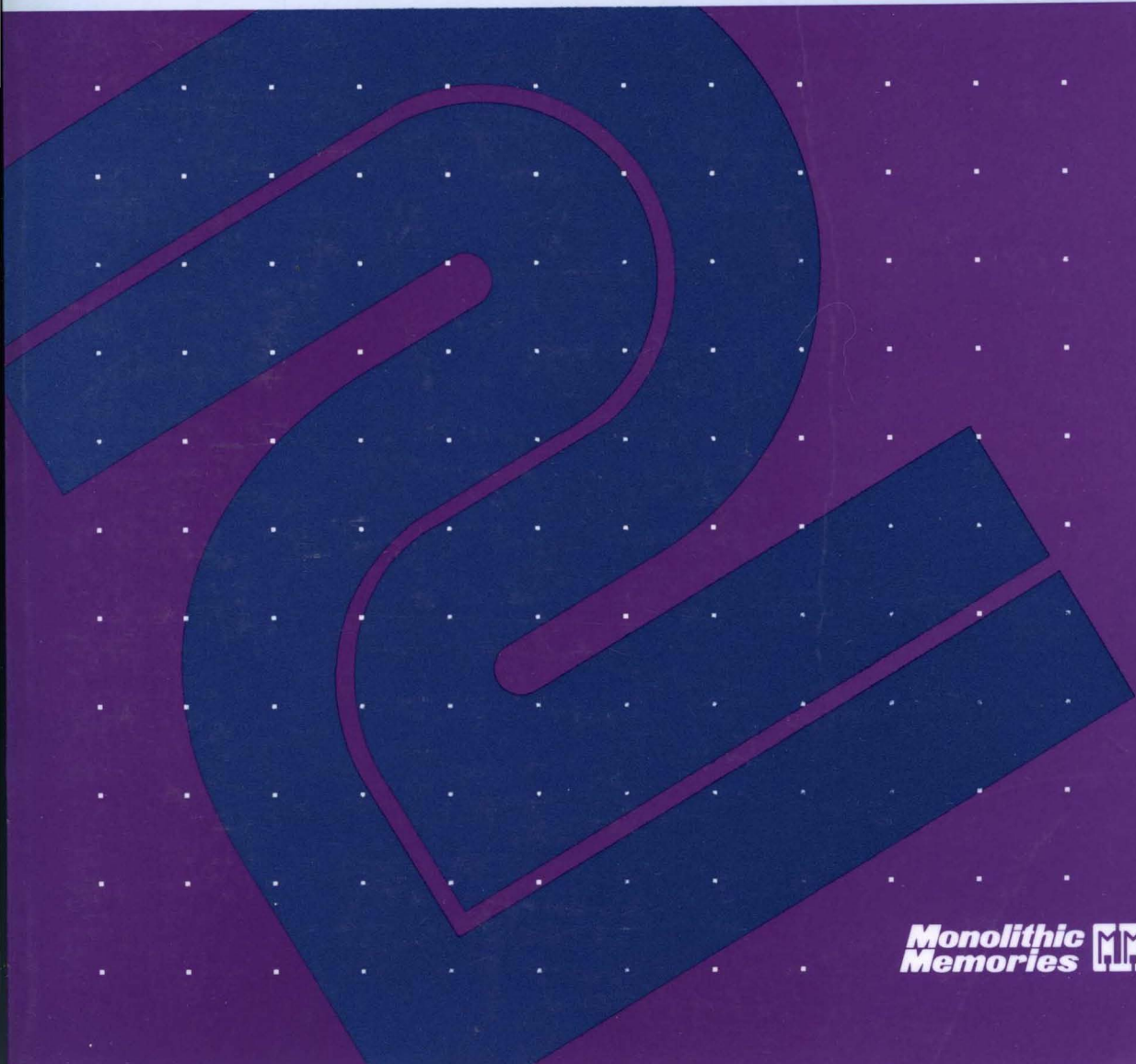
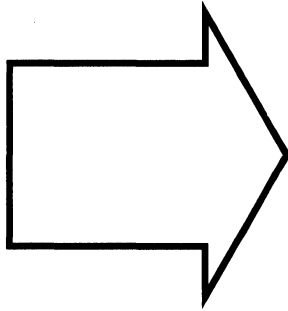


PALASM[®] 2 SOFTWARE




Monolithic Memories 



PALASM 2 USER DOCUMENTATION

VERSION 2, REVISION C, JULY 1987

Table of Contents	vii
Software Errata	ERR
Introduction	1
Installing PALASM [®] 2 Software	2
PDS Syntax	3
Boolean Equation Design	3A
State Machine Design	3B
Simulation	3C
Using PALASM 2 Software	4
Installation and Operation Notes	A
Programmer Notes	B
Device Specific Syntax	C
PAL [®] Design File Library	D
Error Messages	E
Submitting a HAL [®] Design to MMI	F
PALASM 2 Syntax Diagram	G
Supplementary Software	H
JEDEC Standard No. 3	I
Release Notes	RN
Index	IX
PLEASM [™] Manual	

PAL[®], HAL[®], PALASM[®],  and SKINNYDIP[®] are registered trademarks of Monolithic Memories, Inc.

Double-Density PLUS[™] Interface, PLE[™], PLEASM[™], ZHAL[™] AND PROSE[™] are trademarks of Monolithic Memories, Inc.

© Copyright 1978, 1981, 1982, 1984, 1985, 1986, 1987

Monolithic Memories, Inc. • 2175 Mission College Blvd. • Santa Clara, CA 95054-1592
(408) 970-9700 • (910) 970-9700 • (910) 338-2374



PALASM 2 USER DOCUMENTATION

**MONOLITHIC MEMORIES
2175 Mission College Blvd.
Santa Clara, CA 95054-1592
(408) 970-9700**

Version 2

Revision 5C

July 1987

©1987 MONOLITHIC MEMORIES, INC.

Copyright Notice

Copyright 1984,1985, 1986,1987 by Monolithic Memories Inc. The copying and distribution of this manual or the PALASM software is encouraged for the private use of the original purchaser provided this notice is included in all copies. No commercial resale or outside distribution rights are allowed by this notice. This material remains the property of Monolithic Memories Inc. All other rights reserved by Monolithic Memories Inc., 2175 Mission College Blvd., Santa Clara, CA 95054.

Trademarks

PAL, HAL, PLE, ZHAL, PALASM and PLEASM are registered trademarks of Monolithic Memories, Inc.

PROSE is a trademark of Monolithic Memories, Inc.

VAX and VMS are registered trademarks of Digital Equipment Corporation.

TRI-STATE is a registered trademark of National Semiconductor Inc.

IBM, IBM-PC, XT, PC Jr., 3083, PC DOS, and VM/CMS are trademarks of IBM Corporation.

Data I/O and ABEL are registered trademarks of Data I/O Corporation.

UNIX is a trademark of American Telephone and Telegraph.

Omni-Programmer is a trademark of Varix Corporation.

PAL Burner is a trademark of Structured Design Inc.

Wordstar is a trademark of Micro-pro.

MS-DOS is a trademark of Microsoft Inc.

DAISY and DNIX are registered trademarks of DAISY.

DISCLAIMER

Monolithic Memories Inc. makes no representations or warranties with respect to the contents within and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. Further, Monolithic Memories Inc. reserves the right to revise this publication and the product it describes and to otherwise make changes to the product without obligation of Monolithic Memories Inc. to notify any person or organization of such revision or changes.

Preface

Audience

This manual is intended for design engineers who use PALASM™ 2 to program PAL™ devices. The manual assumes that you are familiar with PAL device technology and with PAL device programming concepts.

Using This Manual

This manual steps you through installing PALASM 2 software, and programming a PAL device. We suggest that you work through the examples provided while reading the manual, before proceeding with a design case of your own. All communication with the computer is shown in MONACO typeface.

PALASM 1 Software

In this manual, we refer to the original PALASM software as PALASM 1. This is to distinguish it from the newer PALASM 2 software.

Other Documents

You should have a copy of the PAL Handbook. You should also have vendor manuals for your computer and PAL device programmer. If you are new to PAL devices as well as to PALASM 2 software, you should obtain the booklet *Programmable Logic: A Basic Guide for the Designer*, an excellent introduction to PAL devices that is published by Data I/O Corporation.

Where to Get Help

Monolithic Memories maintains an Applications Hotline to assist in solving engineering related problems. If you are having a problem installing or running PALASM 2 software please call the Hotline at 800-222-9323.

TABLE OF CONTENTS

vii

LIST OF TABLES	xii
LIST OF FIGURES	xiii
SOFTWARE ERRATA	ERR-1

CHAPTER 1 INTRODUCTION

THE PAL DEVICE CONCEPT.....	1-1
Programmable Logic Devices	1-1
INTRODUCING PALASM 2 SOFTWARE	1-6
REFERENCES.....	1-7
SUPPORTED PRODUCTS	1-9
REQUIRED EQUIPMENT.....	1-11
PROGRAM AND FILE SUMMARY.....	1-14
PALASM 2 SOFTWARE PROGRAMS.....	1-14

CHAPTER 2 INSTALLING PALASM 2 SOFTWARE

INTERACTIVE MENU AND NON-MENU MODES	2-2
IBM-PC/XT/AT INSTALLATION.....	2-3
TWIN FLOPPY SET-UP.....	2-4
HARD DISK INSTALLATION.....	2-6
CUSTOMIZE THE INTERACTIVE MENU	2-10
COMPUTER<->PROGRAMMER CONNECTION.....	2-12

CHAPTER 3 PDS SYNTAX

IN THIS CHAPTER.....	3-1
PDS FILE STRUCTURE.....	3-2

CHAPTER 3A BOOLEAN EQUATION DESIGN

DECLARATION SECTION.....	3A-1
Structure.....	3A-2
SYNTAX.....	3A-3
BOOLEAN EQUATION INPUT.....	3A-8
Combinatorial Equations	3A-11
Registered Equations.....	3A-12
Functional Equations.....	3A-14
CHECKLIST FOR BOOLEAN EQUATION DESIGN FILES	3A-22

CHAPTER 3B STATE MACHINE DESIGN

DESIGNING FOR PROSE DEVICES	3B-4
MEALY AND MOORE MACHINES	3B-5
STRUCTURE AND SYNTAX.....	3B-5
DECLARATION SECTION.....	3B-6
STATE SECTION.....	3B-9
STATE MACHINE EQUATIONS.....	3B-14
State Equations.....	3B-16
Output Equations.....	3B-19
POWER_UP Equation.....	3B-21
Condition Equations.....	3B-24

CHAPTER 3C SIMULATION

SIMULATION SYNTAX OVERVIEW	3C-2
DETAILS OF THE SIMULATION SYNTAX.....	3C-4
KEY POINTS TO NOTE.....	3C-11
RULES FOR STATE MACHINE SIMULATION SYNTAX.....	3C-12

CHAPTER 4 USING PALASM 2 SOFTWARE

vii

GENERAL PROCEDURE.....	4-2
DESIGN EXAMPLES	4-10
Boolean Equation Design	4-10
State Machine Design	4-24

APPENDIX A INSTALLATION AND OPERATION NOTES

IBM-PC / DOS 2.10 IMPLEMENTATION.....	A-1
VAX-VMS IMPLEMENTATION.....	A-6
ASCII TAPE INSTALLATION	A-10
VAX-UNIX INSTALLATION	A-12

APPENDIX B PROGRAMMER NOTES

DATA I/O.....	B-1
Using DATA/I/O on VAX-VMS.....	B-6
VARIX OMNI.....	B-9

APPENDIX C DEVICE SPECIFIC SYNTAX

PAL22RX8 AND PAL22V10	
Special Instructions.....	C-2
PAL32VX10	
Special Instructions.....	C-3

APPENDIX D PAL DESIGN FILE LIBRARY

PAL DESIGN FILE LIBRARY.....	D-1
------------------------------	-----

APPENDIX E ERROR MESSAGES

ERRORS REPORTED BY PALASM2	E-1
ERRORS REPORTED BY XPLOT	E-2
ERRORS REPORTED BY MINIMIZE	E-8
ERRORS REPORTED BY PROASM-PROSIM.....	E-17
ERRORS REPORTED BY ZHAL.....	E-30
ERRORS REPORTED BY SIM.....	E-32
ERRORS REPORTED BY JEDMAN.....	E-34

APPENDIX F SUBMITTING A HAL DESIGN TO MMI

MASTER DEVICE.....	F-1
PAL DEVICE DESIGN SPECIFICATION	F-1
FUNCTIONAL TEST VECTORS	F-2

APPENDIX G PALASM 2 SYNTAX DIAGRAM

DEFINITION OF TERMS.....	G-2
SYNTAX DIAGRAM	G-3

APPENDIX H SUPPLEMENTARY SOFTWARE

PALASM.....	H-1
PDSCNVT.....	H-1
DIFFERENCE BETWEEN PALASM 1 AND PALASM 2 SOFTWARE SYNTAX.....	H-2
PC2	H-3

APPENDIX I JEDEC STANDARD NO. 3

vii

PURPOSE.....	I-1
FORMAT DEFINITIONS.....	I-2
OTHER RULES.....	I-10

RELEASE NOTES

IN THIS CHAPTER.....	RN- 1
SUMMARY OF ENHANCEMENTS.....	RN- 2
PALASM 2.22 DISK AND TAPE LAYOUT	RN- 3
FIXED BUGS	RN- 23

INDEX	I-1
-------------	-----

DOCUMENTATION USER RESPONSE FORM

BUG/ENHANCEMENT REPORTS

LIST OF TABLES

1-1	PAL and PROSE Devices Supported by PALASM 2 Software.....	1-10
1-2	PAL Device Programmers Supported by PALASM 2 Software.....	1-13
2-1	Typical Transmission Parameters	2-14
2-2	Command to Set Transmission Parameters.....	2-14
2-3	Commands to Display Programmer Output on Screen	2-15
3-1	PALASM 2 Software Reserved Words	3-3
3A-1	Basic Operators.....	3A-10
3A-2	Results of Polarity Used in Figure 3A-4.....	3A-19
B-1	Recommended Baud Rate.....	B-2
B-2	Device Family Pin Codes For DATA I/O Programmer.....	B-3
D-1	Files Located On Design Examples Disk.....	D-2

LIST OF FIGURES

1-1	Programmable Logic Devices	1-2
1-2	Typical Computer Configuration.....	1-12
1-3	PALASM 2 Software Flow	1-16
2-1	The Main Menu.....	2-8
2-2	MENU.SYS	2-10
3-1	PDS File Structure	3-2
3A-1	Declaration Structure	3A-2
3A-2	Example of the Declaration Section.....	3A-3
3A-3	Overview of Equation Syntax.....	3A-9
3A-4	Pin List and Equations Section.....	3A-18
3A-5	Signal Combinations and Output Polarity.....	3A-20
3A-6	Summary of Output Polarities	3A-21
3B-1	General Synchronous State Machine Architecture	3B-2
3B-2	Moore Behavior	3B-3
3B-3	Mealy Behavior	3B-4
3B-4	Structure of a State Machine Design File	3B-6
3B-5	State Machine Declaration Section.....	3B-8
3B-6	Structure of Default Information	3B-11
3B-7	Syntax for DEFAULT_OUTPUT Statement.....	3B-13
3B-8	Syntax for DEFAULT_BRANCH Statement	3B-14
3B-9	State Machine Equation Operators	3B-15
3B-10	Syntax of State Equations	3B-16
3B-11	Simple State Equation and Diagram	3B-17
3B-12	State Equation with No Local Default.....	3B-18
3B-13	State Equation and Diagram from Traffic Controller Design.....	3B-19
3B-14	Moore Machine Output Equation Syntax.....	3B-20
3B-15	Output Equation Syntax.....	3B-20
3B-16	Mealy Machine Output Equation.....	3B-20
3B-17	Moore Machine Output Equation.....	3B-21
3B-18	Syntax for POWER_UP State Equation.....	3B-22
3B-19	Moore Machine POWER_UP Equation	3B-23
3B-20	Syntax for Mealy Machine POWER_UP Output Equation.....	3B-24
3B-21	POWER_UP State and Output Equations for Mealy Machine.....	3B-24
3B-22	Sample Condition Equations	3B-25
3B-23	Mutually Exclusive Conditions	3B-26
3B-24	Conflicting Conditions.....	3B-26

3C-1	PRLDF Statement	3C-12
4-1	Device Programming Flowchart.....	4-3
4-2	Traffic Intersection	4-24
4-3	Traffic Signal Controller Logic Diagram.....	4-25
4-4	State Diagram of the Traffic Signal Controller.....	4-26
B-1	Data I/O <-> VAX-VMS Cable Connection.....	B-6
I-1	8-Bit Word Definition.....	I-6

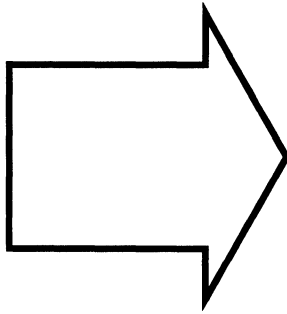


Table of Contents	vii
Software Errata	ERR
Introduction	1
Installing PALASM® 2 Software	2
PDS Syntax	3
Boolean Equation Design	3A
State Machine Design	3B
Simulation	3C
Using PALASM 2 Software	4
Installation and Operation Notes	A
Programmer Notes	B
Device Specific Syntax	C
PAL® Design File Library	D
Error Messages	E
Submitting a HAL® Design to MMI	F
PALASM 2 Syntax Diagram	G
Supplementary Software	H
JEDEC Standard No. 3	I
Release Notes	RN
Index	IX
PLEASM™ Manual	



SOFTWARE ERRATA

ERR

Please read this list of known bugs before you use PALASM 2 software. The following problems occur with PALASM 2.22.

Disk Space

Insufficient disk space causes all PALASM 2 software programs to abort.

Interactive Menu

You require DOS 3.0 or later versions to install and run the PALASM interactive menu. Run the software in non-menu mode if you have earlier versions of DOS.

State Machine Design On A PAL Device

State machine input is being beta tested on PAL devices. Therefore, it may not be fully functional. State machine design entry can, however, be successfully implemented on the PMS14R21 PROSE device.

Output Equations On Mealy and Moore Machines

On a Mealy machine, you must define an output for each state equation. Currently, you must do the same on a Moore machine as well. Otherwise, the software will produce errors.

Run MINIMIZE Before XPLOT

For PAL devices, you must always run the MINIMIZE program before the assembler on a state machine design. If the MINIMIZE program is not run, the assembler, XPLOT, will crash when you try to compile your design.

POWER_UP On PMS14R21

You are allowed only one (1) state branch from the POWER_UP state instead of four.

Brief JEDEC Output

Before running PROASM, the PROSE

- On PMS14R21** assembler, the PALASM2 program asks you if you want a full or brief JEDEC output file. Select full JEDEC output. The PROSE assembler does not produce brief JEDEC output on the PMS14R21.
- PMS14R21 Fuse Information** The PROSE assembler, PROASM, produces an incorrect percentage value for the number of fuses blown in the .XPT output file.
- Parentheses in Boolean Equations** If you use parentheses in your Boolean equations, always run the MINIMIZE program before assembling your design. XPLOT, the assembler, will crash if it detects parenthesized Boolean equations.
- Minimizing XOR Devices** If you use an XOR equation in a Boolean design and run the MINIMIZE program, the XOR will be converted into functionally equivalent AND/OR statements.
- Simulating Files Without Filename Extensions** Your input filename must have an extension, such as <filename.PDS>. Without an extension, the file will be parsed by the syntax checker, PALASM2, but will cause SIM, the simulator, to crash.
- ECL Devices** PALASM 2 software treats I/O pins on ECL devices as outputs only.
- PAL32VX10** In a Boolean equation design on the PAL32VX10, the 3-state I/O pin is defaulted as an output.
- PAL20X8** The PAL20X8 is not a preloadable device. PALASM 2 software allows you to emulate a PRELOAD, but no JEDEC is produced. Also, the device will not be programmed correctly without a PRELOAD.

**PALASM 2 Software
Installation on
VAX-VMS**

The logical directory name PAL2\$DAT must be assigned in the directory that contains the PALASM 2 software. Refer to page A-7 for instructions on assigning PAL2\$DAT.

ERR

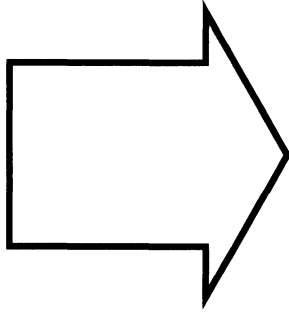


Table of Contents	vii
Software Errata	ERR
Introduction	1
Installing PALASM® 2 Software	2
PDS Syntax	3
Boolean Equation Design	3A
State Machine Design	3B
Simulation	3C
Using PALASM 2 Software	4
Installation and Operation Notes	A
Programmer Notes	B
Device Specific Syntax	C
PAL® Design File Library	D
Error Messages	E
Submitting a HAL® Design to MMI	F
PALASM 2 Syntax Diagram	G
Supplementary Software	H
JEDEC Standard No. 3	I
Release Notes	RN
Index	IX
PLEASM™ Manual	



1. INTRODUCTION

THE PAL DEVICE CONCEPT

A PAL device is a fuse-programmable logic device that can be used to implement custom logic varying in complexity from random gates to complex arithmetic functions. The PAL device implements the familiar sum-of-products logic by using a programmable AND array whose output terms feed a fixed OR array. Since the sum-of-products form can express any Boolean transfer function, the uses of the PAL device are limited only by the number of terms available in the AND and OR arrays. Thus, the PAL device combines much of the flexibility of the PLA with the low cost and easy programmability of the PROM. Moreover, PAL devices come in different sizes so that you can choose the size that is most cost-effective for your applications.

Programmable Logic Devices

PAL devices are one of three main types of programmable logic devices:

- * PAL Programmable Array Logic
- * PROM Programmable Read-Only Memory
- * FPLA Field Programmable Logic Array

All three kinds of devices include an array of AND gates whose outputs feed an OR gate. The differences occur in the location of the programming fuses. Figure 1-1 shows the basic architecture of each kind of programmable logic device.

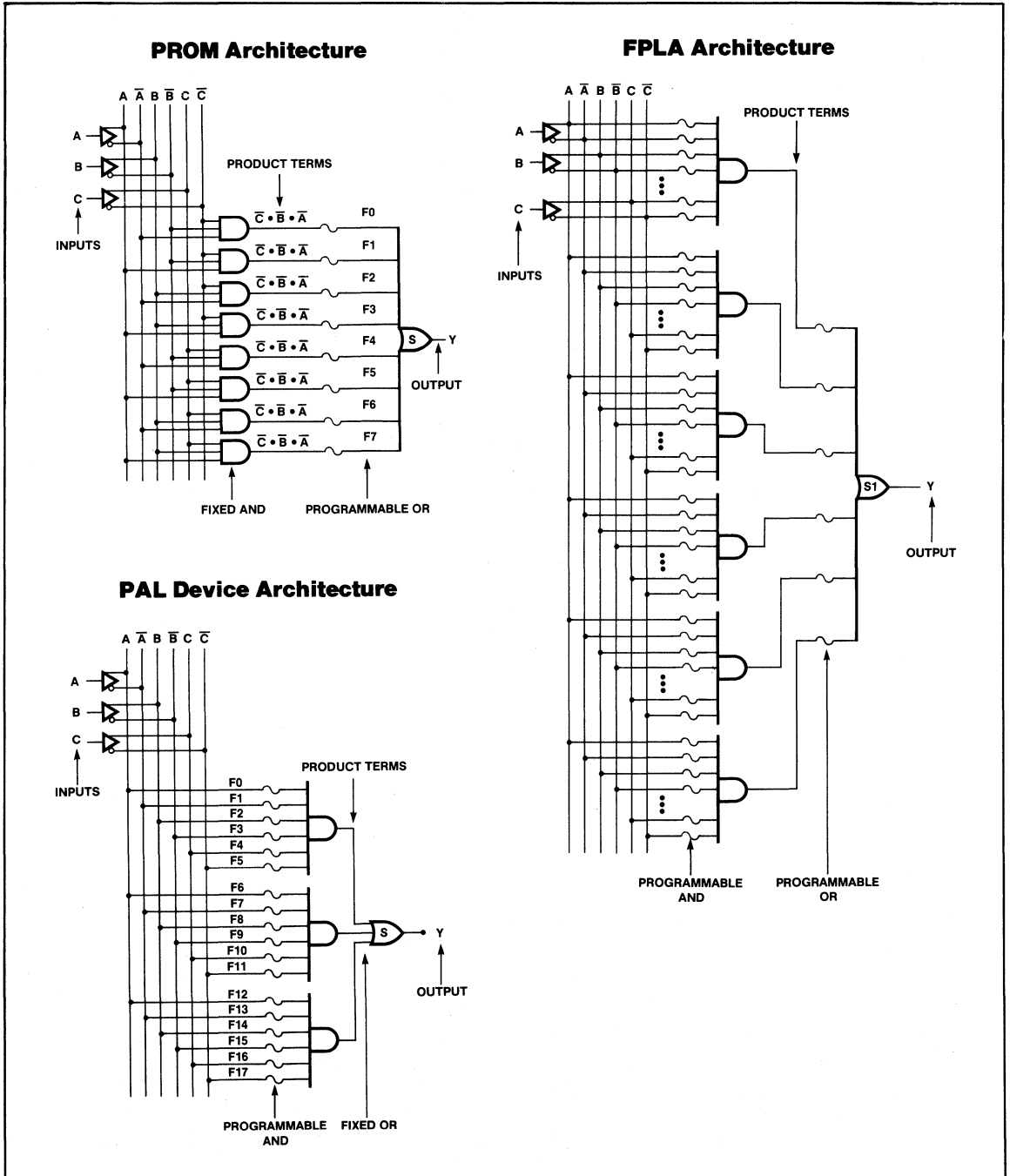


Figure 1-1:
Programmable Logic Devices

PAL devices have fuses on the inputs to the AND gates. By selecting the fuses to blow, you choose product terms available to the OR gate. Usually several product terms are produced and ORed for each output.

PROM devices have fuses on the outputs of the AND gates. All possible product terms are produced; you choose the terms to be applied to each output by deciding which fuses to leave intact.

FPLA devices have fuses on both the inputs and outputs of the AND gates. This is the most general configuration for programmable devices.

1

PAL Device Configurations

The members of the PAL device family combine the following basic configurations: combinatorial arrays, programmable I/O, registered outputs with feedback, exclusive OR (XOR), and programmable polarity.

Combinatorial Arrays

PAL device combinatorial arrays are available in sizes from 64 x 32 (64 input terms maximum and 32 output terms maximum) to 16 x 2. Both active-high and active-low output configurations are available. This wide variety of input/output formats allows you to replace many different-sized blocks of combinatorial logic with single PAL packages.

Programmable I/O

The high-end members of the PAL device family have programmable I/O pins. This allows a product term to directly control an output of the device. The product term is used to enable the three-state buffer that gates the summation term to

the output pin. The output is also fed back into the PAL device as an input. Thus the PAL device drives the I/O pin when the three-state gate is enabled; the I/O pin is an input to the PAL device when the three-state gate is disabled. This feature can be used to allocate available pins for I/O functions or to provide bidirectional output pins for operations such as shifting and rotating serial data.

Registered Outputs with Feedback

Another feature of the high-end members of the PAL device family is registered outputs with registered feedback. Each product term is stored into a D-type output flip-flop on the rising edge of the system clock. The Q output of the flip-flop can then be gated to the output pin by enabling the active-low three-state buffer. The Q output is fed back into the PAL device as an input term. This feedback allows the PAL device to *remember* its previous state, and the PAL device can alter its function based upon that state. This allows the designer to configure the PAL device as a state sequencer that can be programmed to execute such elementary functions as count up, count down, skip, shift, and branch. These functions can be executed by the registered PAL device at rates of up to 65 MHz for *D* PAL devices.

XOR PAL Devices

These PAL devices feature an exclusive OR (XOR) function. The sum of products is segmented into two sums, which are then XORed at the input of the D-type flip-flop. All the features of registered PAL devices are included in the XOR PAL device. The XOR function provides an easy implementation of the HOLD operation used in counters and other state sequencers.

Programmable-Polarity PAL Devices

Programmable-polarity PAL devices allow a single part to function as both an active-low and an active-high part. With the polarity fuses intact, the outputs are active-low; when the polarity fuses are blown, the outputs are active-high. If output 1 (OUT1) is specified as uncomplemented in the pin list, and as uncomplemented in the equations, the output is effectively specified as active-high and the polarity fuse is blown.

The converse is also true. If OUT1 is complemented in the pin list (/OUT1) and complemented in the equation, the fuse will be blown. Whenever the polarities differ between the pin list and the logic equation, the fuse will be left intact.

Hard Array Logic Devices

A hard array logic (HAL™) device is the mask-programmed version of a PAL device. A HAL device is to a PAL device what a ROM is to a PROM. A standard wafer is fabricated up to the metal mask. Then a custom metal mask is used to fabricate aluminum links for a HAL device instead of the programmable titanium-tungsten (Ti-W) fuse array used in a PAL device. The HAL device is a cost-effective solution for large quantities and is unique in that it is a gate array with a programmable prototype. For information on submitting HAL designs to Monolithic Memories, see Appendix F of this manual.

PROSE™ Devices

The PMS14R21 programmable sequencer is the first member of the PROSE (PROgrammable SEquencer) family. The PMS14R21 is a high-speed, 14-input, 8-output state machine. It consists of a 128 x 21 PROM array preceded by a 14H2 PAL array. The PAL array is efficient for a large number of input conditions, while the PROM array is optimal for a large number of

product terms and states. The combination allows a very efficient state machine with a large number of inputs and outputs. For more information on the PMS14R21, refer to the datasheet in the PAL handbook.

INTRODUCING PALASM 2 SOFTWARE

PALASM 2 software is a package that turns PAL device Design Specification (PDS) files into data files for PAL device programmers. PDS is a format for specifying a PAL circuit and for creating inputs to a logic circuit. Using a text editor, you create a PDS file that describes a PAL circuit. PALASM 2 software accepts the file as input and performs a number of functions under your control, including:

- * Assembling PAL Design Specifications
- * Generating PAL device fuse patterns in JEDEC format
- * Reporting errors in syntax and assembly
- * Allowing concise mnemonic names for long, frequently used logic expressions through string substitution

Functional Differences from PALASM 1 Software

We refer to the original PALASM software as PALASM 1 software to distinguish it from the new PALASM 2 software.

PALASM 2 software is quite different from PALASM 1 software in implementation. It is composed of several interacting programs coupled by disk files. (Floppy based files slow interaction. We recommend RAM or hard disks for production use.) The principle benefit of the reorganization is the freedom from fixed limits within the design file.

As mentioned in the feature section, the syntax of PALASM 2

software is significantly different from that of PALASM 1. PALASM 2 software allows description of asynchronous devices like the PAL20RA10 and devices of much higher complexity such as MegaPAL devices.

The current version of PALASM 2 software omits several features provided within PALASM 1. They are:

1

- * Fault coverage prediction for test vectors.
- * Automatic generation of documentation.
- * Device signal/pinout display.
- * Support of the security fuse.
- * Printing of logic equations for each product term in a fuse plot.

Some of these omissions represent a change in philosophy; others will be provided in later versions of the program.

For more detail on PALASM 1 software, refer to Appendix H.

REFERENCES

For further information about using programmable logic devices, see the following references.

Software

1. Birkner, John M., "Macros for Programmable Logic," Wescon Professional Conference Session Record, 1982.
2. Birkner, John M., "High Level Language for Programmable Array Logic," Wescon Professional Conference Session Record, 1981.

3. Birkner, John M., Coli, Vincent J., and Sackett, David M., "Design Your Own Chip With PALASM, Your Personal Computer, and a PROM Programmer," *Machine Design*, July 1983, pp.81-85.
4. Birkner, John M., "CAD Methodology Parallels Advances in Programmable Logic."

PAL Device

1. Birkner, John M. and Coli, Vincent J., "Hard Array Logic Provides New TTL Standards," *Southcon*, 1982.
2. Birkner, John M., "CAD Methodology Parallels Advances in Programmable Logic," *Electronica* 1982.
3. Miller, Warren, "The Philosophies of Fuse Programmable Logic," *Wescon Conference Professional Session Record*, 1982.
4. Miller, Warren, "New Developments in Programmable Logic," *Wescon Conference Professional Session Record*, 1982.
5. "Programmable Logic: A Basic Guide for the Designer," *Data I/O Corporation*.
6. Edwards, E. and Greiner, J., "Programmable Logic Matches Gate-array Density, Eases System Design." *Electronic Design*, June 14, 1984.

PAL Device Applications

1. Coli, Vincent J., "PAL Bumps Eight Chips from Microprocessor Interface," *Electronic Design*, November 25, 1982, pp.180-182.

2. Blasco, Richard W., "PALs Shrink Audio Spectrum Analyzer," *Electronic Design*, August 20, 1980.
3. Coli, Vincent J., "Using a PAL to Emulate the Internal State Counter of the MMI 'S516 LSI Multiplier/Divider," *The Best of the Computer Faires*, Volume VIII, 1983.

SUPPORTED PRODUCTS

With the exception of the PAL16A4 and the PAL16X4 parts, PALASM 2 software supports all Monolithic Memories PAL devices including new PAL products such as RA (Registered Asynchronous), RS (Registered Synchronous), MegaPAL, ZHAL™, as well as the newest and only member of the PROSE family of devices.

Table 1-1 lists PAL and PROSE devices supported by PALASM 2 software.

**Table 1-1:
PAL and PROSE Devices Supported by PALASM 2 Software**

20-Pin Devices	24-Pin Devices	MegaPAL Devices	PROSE Devices
PAL10H8	PAL6L16	PAL32R16	PMS14R21
PAL10L8	PAL8L14	PAL64R32	
PAL12H6	PAL12L10		
PAL12L6	PAL14L8		
PAL14H4	PAL16L6		
PAL14L4	PAL18L4		
PAL16H2	PAL20L2		
PAL16L2	PAL20C1		
PAL16L8	PAL20L8		
PAL16P8	PAL20L10		
PAL16C1	PAL20X4		
PAL16R4	PAL20X8		
PAL16R6	PAL20X10		
PAL16R8	PAL20R4		
PAL16RA8	PAL20R6		
PAL16RP4	PAL20R8		
PAL16RP6	PAL20RA10		
PAL16RP8	PAL20S10		
ZHAL20	PAL20RS4		
	PAL20RS8		
	PAL20RS10		
	PAL22V10		
	PAL10H20P8		
	PAL10H20G8		
	PAL32VX10		
	PAL22RX8		
	ZHAL24		

REQUIRED EQUIPMENT

This section describes computers and PAL device programmers supported by PALASM 2 software and provides information about necessary and optional PALASM 2 programs.

1

Computers

PALASM 2 software operates with no user modification on the following CPUs, provided certain minimum system requirements are satisfied. It is usually provided as an executable program, ready to run on any of these systems:

- | | |
|-----------------|--|
| Minicomputers: | VAX™ under VMS™
VAX™ under UNIX™ (Berkeley 4.2) |
| Microcomputers: | IBM-PC™, -XT™, -AT™
under MS-DOS™ (256K RAM) |
| Workstations: | DAISY™ under DNIX™ 5.1 |

All systems must have a serial port (RS-232) for communication with the PAL device programmer. We also recommend that floppy disk based systems be equipped with two disk drives.

Figure 1-2 shows a typical computer configuration.

Note: Refer to the PALASM 2 software order form for the correct part number of the version of PALASM 2 software designed for your CPU.

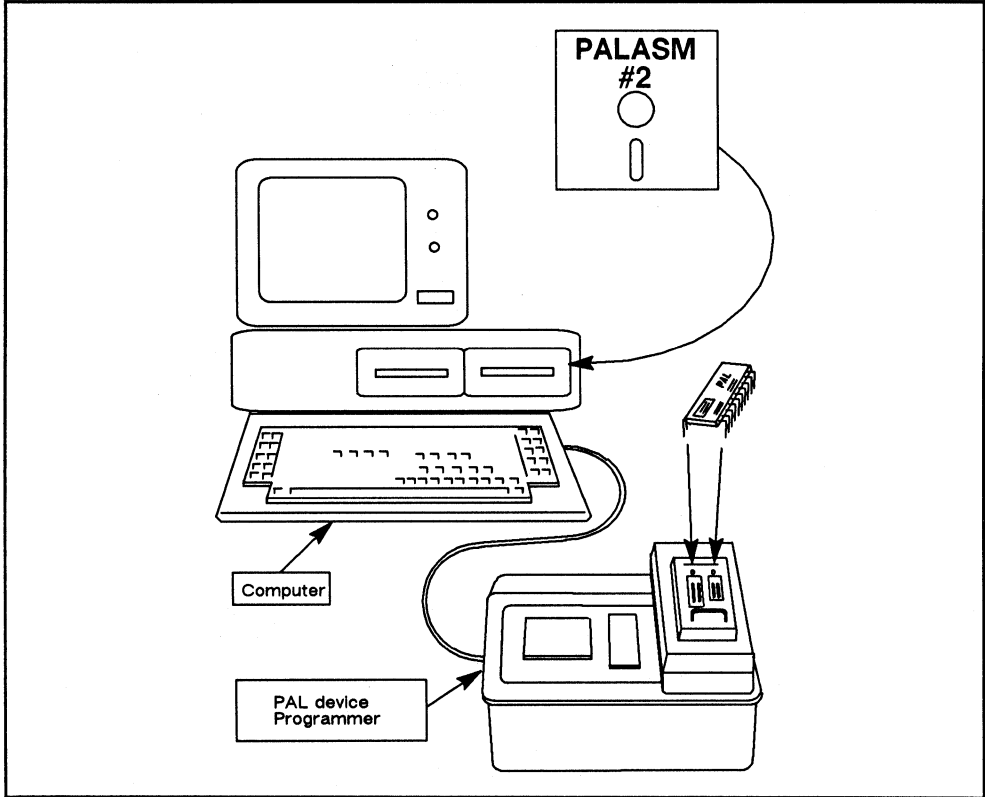


Figure 1-2:
Typical Computer Configuration

PAL Device Programmings

The PAL device programmers supported by PALASM 2 software are shown in Table 1-2.

Table 1-2:
PAL Device Programmers Supported by PALASM 2 Software

Products	Manufacturer
DATA I/O Model 19 with LogicPak	DATA I/O 102525 Willows Rd. N.E
DATA I/O Model 29A or B with LogicPak	P.O. Box 97046 Redmond, WA 98073-9746
Adapter: 303A-002 303A-008 A/B for 32R16 303A-023 A/B for 64R32	
STAG ZL30/PPZ	STAG Microsystems 528-5 Weddell Dr. Sunnyvale, CA 94089
DIGELEC FAM51 or FAM52	DIGELEC 1602 Lawrence Ave. #113 Ocean, NJ 07712
KONTRON MPP80S KONTRON MOD21	KONTRON 1230 Charleston Rd. Mountain View, CA 94039
VARIX OMNI PROGRAMMER	VARIX 1210 East Campbell Road #100 Richardson, TX 75081
STOREY SYSTEMS P240	Storey Systems 3201 North Hwy 67, Suite H Mesquite, TX 75150
VALLEY DATA SCIENCES 160	Valley Data Sciences 2426 Charleston Business Pk Mountain View, CA 94043

1

PROGRAM AND FILE SUMMARY

Following is a summary of all currently available programs. A more detailed description of each program follows this summary.

- | | |
|------------------|--------------------------------------|
| 1. PALASM2 | PALASM 2 syntax parser |
| 2. MINIMIZE | Logic expander and minimizer |
| 3. XPLOT,SIM | PAL device assembler and simulator |
| 4. PROASM-PROSIM | PROSE device assembler and simulator |
| 5. JEDMAN | JEDEC disassembler |
| 6. ZHAL | ZHAL device fit |

Note: The ZHAL utility is not part of the regular package you get when you order PALASM 2 software. You may, however, order the ZHAL program from your local Monolithic Memories sales office.

PALASM 2 SOFTWARE PROGRAMS

The main PALASM 2 software programs are described in the following pages. Figure 1-3 shows the PALASM 2 software processing flow.

PALASM2

PALASM2 is the first program you will use in the PALASM 2 software suite. It reads and validates your input—a PLD device

design specification—for correct design syntax. If an error is detected, the program attempts to indicate where in the input description the error has occurred. Recovery is attempted after each error in order to catch as many errors as possible on a single run. Only if no error is detected is an intermediate specification file generated. This file contains the input specification in a hierarchically structured form to enable easy processing by follow-on programs. Further, it is guaranteed to be syntactically correct. This program recognizes input descriptions for all current PAL devices.

MINIMIZE

The MINIMIZE program gives you the option of automatically reducing your logic equations. Minimization helps to utilize the space on your device more efficiently and is therefore a cost effective feature. This program automatically translates a state machine design file to Boolean equations. Although all PAL devices are supported for logic minimization, the program does not work effectively on Exclusive OR devices. This is because XOR is treated as a complex logic element and is taken out during minimization. A warning message is displayed on the following Exclusive Or devices: PAL22RX8, PAL32VX10, PAL20X10, PAL20X8, PAL20X4.

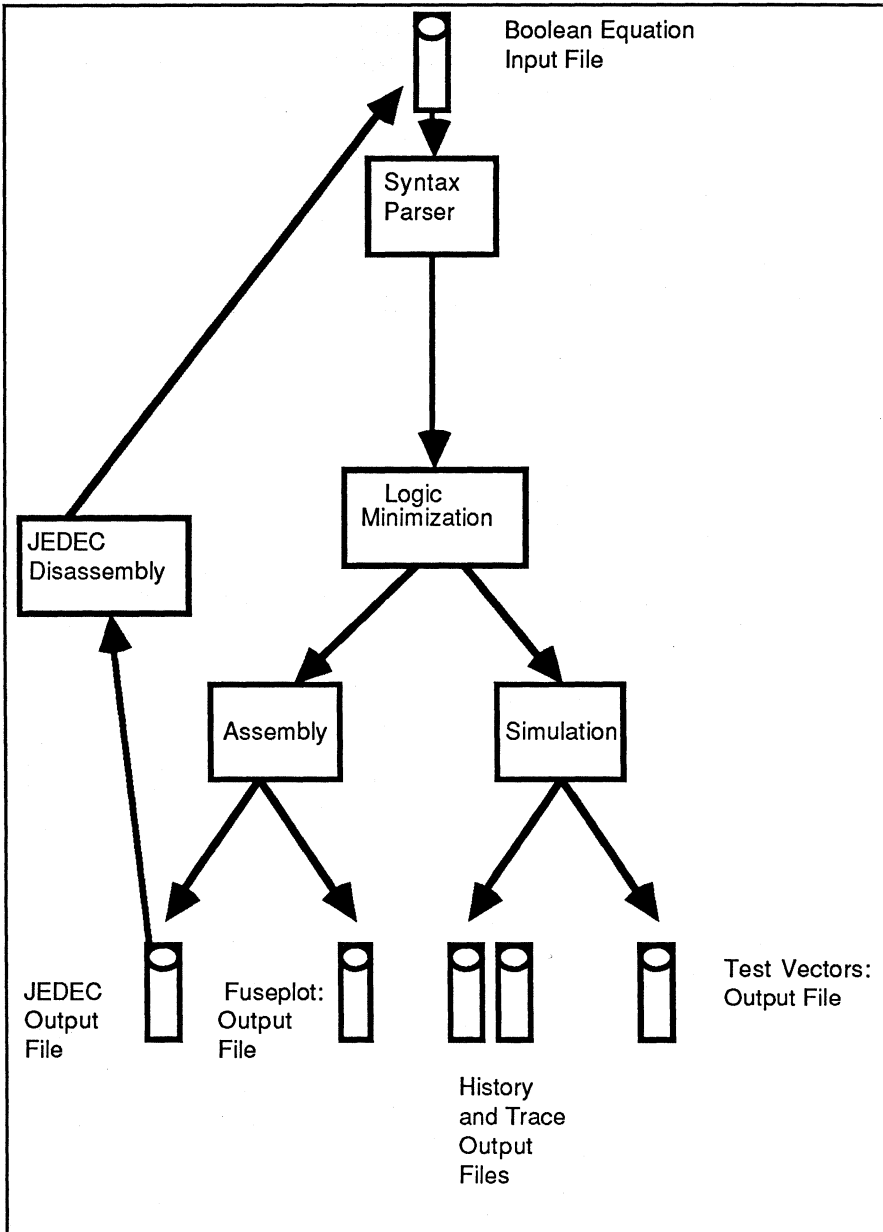


Figure 1-3:
PALASM 2 Software Flow

XPLOT

XPLOT validates the architectural design of an input PAL device description and produces fusemaps and JEDEC data for a specified PAL device. Input is a set of Boolean equations that has been preprocessed by the PALASM2 program. XPLOT checks the equations for consistency and correctness for the specified PAL device. When an error is detected, XPLOT attempts immediate recovery. In this way, XPLOT spots as many errors as possible on each run. Only if no errors are detected will the output fusemaps and JEDEC data be generated. The architectural information for each PAL device is read in from a file containing a profile description for the specific PAL device.

Note: XPLOT will check only valid Monolithic Memories PAL devices.

SIM

SIM checks the functionality of a PAL device design. You will run this program after XPLOT. If the design is architecturally correct, however, you can run SIM directly after the PALASM2 program. SIM reads a special simulation syntax that has been preprocessed by the PALASM2 program. It will simulate the operation of the PAL device you specify, calculating the output values based on input signals through the Boolean equations and any feedback. SIM outputs two files: a history file and a trace file. The history file shows the values of every pin through a simulation sequence. The trace file, which is a subset of the history file, shows only the pins you specify in the simulation syntax. If XPLOT has been run and a JEDEC fuse address file has been created, then SIM will add test vectors to the JEDEC file that duplicate the simulation sequence when the device is tested on a programmer. All JEDEC checksums are recalculated.

Note: SIM tests only valid Monolithic Memories PAL devices.

PROASM-PROSIM

PROASM and PROSIM assemble and simulate PROSE device designs. PROASM accepts both State Machine and Boolean logic designs. Assembly and simulation are both transparent to the user. Please note that the only PROSE device PALASM 2 software currently supports is the PMS14R21.

JEDMAN

JEDMAN offers you the option of disassembling a JEDEC file and generating Boolean equations. You can, in effect, read a fuseplot directly from a programmed device. JEDMAN also recalculates checksums and allows you to convert a PAL22V10 JEDEC file to a PAL32VX10 JEDEC file

ZHAL

The ZHAL program helps you fit your completed PAL device design into a 20-pin or 24-pin ZHAL (Zero-standby-power CMOS HAL) device. If you plan to opt for volume production using Monolithic Memories ZHAL devices, ensure quick turn-around by letting the ZHAL program match your design to make it fit into the device. If your device fits, you may send it to Monolithic Memories for mask processing.

Note: The ZHAL utility is not part of the regular package you get when you order PALASM 2 software. You may, however, order the ZHAL program from your local Monolithic Memories sales office.

Supplementary Software

Following is a summary of all currently available supplementary software. A more detailed description of some of the supplementary programs follows this summary.

1

1. PALASM PALASM 2 interactive menu
2. PDSCNVT PALASM 1 to PALASM 2 syntax conversion
3. PC2 Programmer interface program
4. SCRSIM Simulation waveform generation program
5. VTRACE Sim output files to timing diagrams conversion
6. BINHEX Binary to hexadecimal conversion
7. TIMING Timing diagram entry program
8. PINOUT Pinout Program
9. DECODE Address Decoder Program

PALASM 2 Supplementary Programs

Some of the supplementary programs listed on the previous page are described below. Please note that Monolithic Memories does not support all the programs that reside on the supplementary disk.

PALASM

PALASM is the name of the interactive menu program that is designed to simplify user interface to the software. The program may be installed on an IBM-PC -XT -AT, either twin floppy or hard disk. The user-friendly menu screens display your options on one screen, enable the use of function keys to run all the programs in the software suite, and allow you to view the output as well. The PALASM menu significantly reduces your learning curve since all you need to know is what you want to do, not how to do it. Online help screens and message windows facilitate easy interaction with the software.

PDSCNVT

PDSCNVT allows you to interactively convert PAL device design specifications from the PALASM 1 format to PALASM 2 software. Input is a PALASM 1 formatted specification file, and output is the equivalent design in PALASM 2 software syntax.

PC2

PC2 enables communication between PLD programmers and IBM™ PC machines (-PC, -XT, -AT, etc.). It is a menu-driven multiple-choice program that guides you through various options for programming and checking PLDs.

VTRACE

VTRACE reads the trace output of the PALASM 2 software simulator. The text-formatted data of the trace file is converted into graphic form. VTRACE output looks very much like timing diagrams of the simulation results.

1

Files

Input, output, and intermediate files (files that the software creates but are not visible to the user) are listed below.

1. <filename>.PDS User defined PLD design description input.
2. PALASM2.TRE PLD intermediate design description.
3. <filename>.PDF PLD architecture description data.
4. <filename>.XPT Contains PLD fuse map data.
5. <filename>.JED Contains PLD fuse JEDEC data.
6. <filename>.HST Contains full simulation history data.
7. <filename>.TRF Contains user simulation trace data.
8. <filename>.JDC Contains both PLD fuse JEDEC data and JEDEC test vectors.
9. <filename>.PL2 Contains PDS file reconstructed from JEDEC output.
10. <filename>.JDM Recalculated checksums or PAL22V10 to PAL32VX10 conversions using JEDMAN.

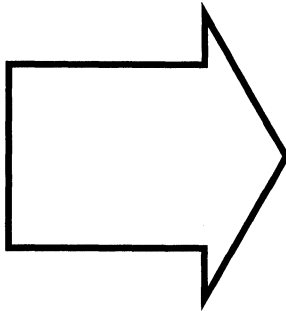


Table of Contents	vii
Software Errata	ERR
Introduction	1
Installing PALASM® 2 Software	2
PDS Syntax	3
Boolean Equation Design	3A
State Machine Design	3B
Simulation	3C
Using PALASM 2 Software	4
Installation and Operation Notes	A
Programmer Notes	B
Device Specific Syntax	C
PAL® Design File Library	D
Error Messages	E
Submitting a HAL® Design to MMI	F
PALASM 2 Syntax Diagram	G
Supplementary Software	H
JEDEC Standard No. 3	I
Release Notes	RN
Index	IX
PLEASM™ Manual	



2. INSTALLING PALASM 2 SOFTWARE

This chapter tells you how to get started. Refer to the appropriate page number for the computer and operating system on which you are running PALASM 2 software.

2

IN THIS CHAPTER

<i>If you have a...</i>	<i>Refer to...</i>
IBM-PC/XT/AT	2-3
VAX-VMS	Appendix A-6
VAX-UNIX	Appendix A-12
VAX-ASCII	Appendix A-10

PALASM 2 software can be run using either an interactive menu interface or in non-menu mode. Decide which mode you would prefer to use. The instructions for installation include the set-up procedure for both modes.

INTERACTIVE MENU AND NON- MENU MODES

PALASM 2 software can be run using the interactive menu or in non-menu mode.

Interactive Menu

We recommend the use of the user-friendly menu interface for both first-time and advanced users. If you are a first-time user, the interface will considerably reduce your learning curve. Advanced users may also find that the menu screens facilitate easy interaction with the software.

The interactive menu resides on your Supplementary diskette. To use the interactive menu, you need to go through a one time installation procedure. Once the program is successfully installed, it will always be the first screen that is called up. You may use the interactive menu only if you have MS-DOS 3.0 or later versions. The interactive menu installation procedure for each computer and operating system is described in the following pages.

Non-Menu Mode

This is recommended for advanced users only. Also, if you do not have MS-DOS 3.0 or later versions of MS-DOS, you must use the software in non-menu mode. In this mode, you merely type commands to activate programs directly from DOS. Instructions for preparing your system to run PALASM 2 software in non-menu mode follow in the next few pages.

IBM-PC/XT/AT INSTALLATION

What You Require

To install PALASM 2 software, you require.

1. An IBM-PC/XT/AT with either twin floppy drives or a hard disk.
2. **Minimum memory of 384K bytes to run the software in menu mode**

or

minimum memory of 256 K bytes to run the software in non-menu mode.

3. **DOS 3.0 or later versions to run the software in menu mode.**
4. An IBM-DOS diskette.
5. PALASM 2 software on regular or high density diskettes.
6. A blank diskette if you are using a twin floppy system.

Note: If your system does not have 384K bytes memory or DOS 3.0 (or later versions), you must use the software in non-menu mode.

<i>If you have a...</i>	<i>Refer to...</i>
Twin Floppy System	Page 2-4
Hard Disk	Page 2-6

TWIN FLOPPY SET-UP

A few simple steps enable you to set-up your twin floppy system to run PALASM 2 software.

Create A Work Diskette

First, you need to create a WORK diskette to store your design files.

1. Insert the IBM-DOS diskette in drive B.
2. Insert a blank diskette in drive A.

Note: If you have a 1.2 megabyte floppy in drive A, remember to use a high density diskette.

3. Enter

```
B: <CR>
```

4. Enter

```
FORMAT A:/S <CR>
```

Now you have a WORK diskette that contains the COMMAND.COM file.

5. When you see the system message

```
FORMAT ANOTHER?
```

```
Enter
```

```
N
```

Load The Supplementary Software

Now you are ready to load the Supplementary Software.

1. Enter

B: <CR>

2

2. Insert the Supplementary diskette in drive B.
3. If you wish to use PALASM 2 software's **interactive menu**, enter

FLOPPY2 +MENU <CR>

Or, if you wish to use PALASM 2 software in **non-menu mode**, enter

FLOPPY2 NOMENU <CR>

At this point you will see further instructions on your screen. Follow these instructions to complete the installation. Turn to page 2-7 to find out if the menu is properly installed.

HARD DISK INSTALLATION

Follow these steps to install the software on your hard disk.

1. Insert the Supplementary Software disk in drive A. (Use drive A on an AT as well.)

2. Enter

A: <CR>

3. Enter

PAL2INST <CR>

Follow the instructions on your screen to complete the installation procedure.

4. Reboot your system after the installation is complete.

To test that PALASM 2 software is installed in **menu mode**, turn to page 2-7 for further instructions.

To use PALASM 2 software in **non-menu mode**, turn to page 2-9 for further instructions.

Interactive Menu

Run A Test Example

After you have completed the installation procedure, run a test.

1. To call up the program, enter

C: PALASM <CR>

2

2. Now you will see the first screen of the PALASM2 interactive menu.

Next, press <CR> again.

Your screen now displays the main menu as shown in Figure 2-1.

PALASM V2.22 (c)MONOLITHIC MEMORIES,SANTA CLARA,CA 95054 FEB 28,1987

Input PDS file		dummy .pds	Directory		C:\
Device	PAL20RA10	F1	Display Dir	F2	DOS Command
<u>PALASM 2</u>				MENU	
F3	Edit PDS	F4	Program Device		
F5	PALASM	F6	Install Menu	WINDOW	
F7	View Data	F8	Supplementary		

KEY MOVEMENTS

<esc><esc>= exit	PRESS F9 FOR HELP	▼ = next field
<esc><ret>=refresh	→ = previous position	▲ = previous field
 = delete	← = next position	<home> = first field
<ins> = insert		<end> = last field

STATUS : ALL OK

Figure 2-1: The Main Menu

Congratulations! You have successfully installed PALASM 2 software in interactive mode on your hard disk. Remember, to call up PALASM 2 software, type

PALASM

Non-menu Mode

Modify System File

To install PALASM 2 software in non-menu mode, you must modify the AUTOEXEC.BAT file.

1. Open the AUTOEXEC.BAT file.
2. Add the following two lines to the end of the file.

```
PATH <D>:\PALASM2\PAL2;%PATH% <CR>  
DPATH <D>:\PALASM2\PDF;<D>:\PALASM2\MSG;  
      <D>:\PALASM2\SUPL; <CR>
```

3. Reboot the system.

Run A Test

After you have installed PALASM 2 software in non-menu mode, run a test. To activate each of the PALASM 2 programs, you must type the program name. The complete list of programs is given below.

```
PALASM2  
MINIMIZE  
XPLOT  
SIM  
PROASM  
PROSIM  
JEDMAN
```

For a description of each program, refer to page 1-14.

Congratulations! You have successfully installed PALASM 2 software in non-menu mode on your hard disk.

CUSTOMIZE THE INTERACTIVE MENU

You may customize the interactive menu program with a one-time set-up procedure so that from the main menu you can

- * Easily access supplementary programs

This feature enables you to go directly from the main menu to the program of your choice without accessing DOS.

The installation procedure is stored in the file MENU.SYS. Each time you call up the interactive menu, the software reads the file MENU.SYS. Therefore, to customize the interactive menu, you must modify MENU.SYS.

The format of your MENU.SYS file is shown in Figure 2-2.

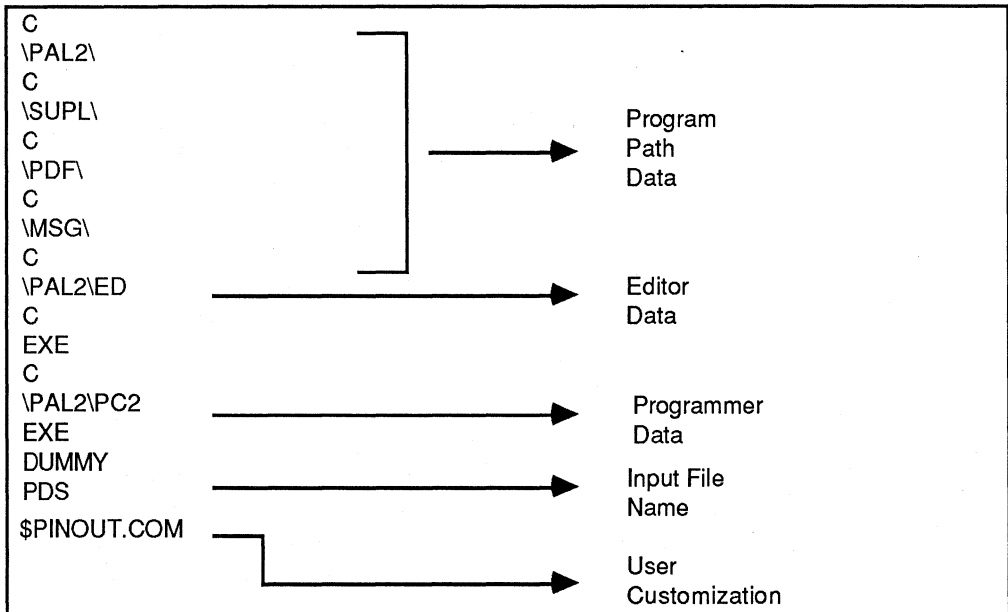


Figure 2-2: MENU.SYS

Any change you make to the file must occur after the Input File Name (see Figure 2-2).

Add a Program to the Supplementary Programs List

2

Supplementary programs supported by Monolithic Memories can be called up from the main menu by choosing the supplementary program option. You may activate more programs from the main menu by adding the program names to the menu system file MENU.SYS. For a complete list of the supplementary programs that you may add to the system file, refer to page 2-12.

The procedure to add a program to the MENU.SYS file follows.

1. Enter the text editor.
2. Open the MENU.SYS file.
3. To the end of the file, add the filename of the Supplementary program in the following format.

`$<FILENAME>`

For example, to add the program PINOUT, you enter

`$PINOUT.COM`

4. Save the file.

That's all. To test the modification, call up the interactive menu program. The F8 option will now allow you to call up the Supplementary program you have added.

Software Programs On The Supplementary Diskette

Following are the programs for which direct access from the main menu is not available. Turn to page 2-11 for the procedure that allows you to call up these programs from the main menu.

SCRSIM.COM	Simulation Waveform Generation Program
VTRACE.COM	A Utility Program To Print Sim Output Files As Timing Diagrams
BINHEX.COM	A Binary To Hexadecimal Conversion Program
TIMING.COM	Timing Diagram Entry Program
PINOUT.COM	Pinout Program
DECODE .COM	Address Decoder Program That Generates PALASM2 Boolean Equations

COMPUTER<-> PROGRAMMER CONNECTION

Because a large number of hardware combinations are possible, this manual cannot give detailed instructions or cabling information. Consult the manuals supplied with the computer and the programmer. Read also the general configuration information below for your type of computer.

IBM-PC

The usual practice with PCs is to connect the programmer to a serial port. In MS-DOS, the serial ports have the device names

COM1: and COM2:.. For computers using add-on boards with serial I/O ports you may need to set switches on the board or the PC's mother board. To use the communication commands in the rest of this manual, you need to know the device name of the port the programmer is connected to.

VAX Minicomputers

2

Many terminals such as the VT-100 have a serial port that echoes the information displayed on the terminal screen. Connect the programmer to this port. (Confirm interface with your computer operations department to avoid damage to the terminal.) When you want the information displayed on the screen to go to the programmer as well, you just have to turn on the programmer and put it in receive mode.

Verifying the Communications Link

To verify the communications link, you can usually send a simple memory dump from the PAL device programmer to the computer. Because PAL device programmers differ in the way that they accept data files, we can only describe a general procedure here. Refer to the manual supplied with the PAL device programmer for specific information.

The general procedure for establishing the communications link is as follows:

1. Set the transmission parameters for the programmer. Refer to the programmer manual for specific information. Typical parameters are shown in Table 2-1.

**Table 2-1:
Typical Transmission Parameters**

Parameter	Typical Setting
baud rate	1200, 2400, or 4800
number of data bits	7
number of stop bits	1
parity	even

- Set the same transmission parameters for the computer. The appropriate command for each operating system is shown in Table 2-2.

**Table 2-2:
Command to Set Transmission Parameters**

Operating System	Command
MS-DOS	MODE (see NOTE following) Example: MODE COM1:4800,N,8,1
VMS	SET TERM

Note: If you are using an IBM PC and a DATA I/O PAL programmer, the supplementary program PC2 is an effective way to connect these systems. PC2 sets the transmission parameters for the IBM PC and establishes the communications link. Appendix A describes how to verify the communications link for this combination of devices.

- Prepare the computer to receive information from the programmer. The easiest way is to set the computer to display on the terminal screen whatever is received on the serial port. The typical procedures are shown in Table 2-3.

Table 2-3:
Commands to Display Programmer Output on Screen

System	Command	Notes
MS-DOS	COPY COM1: CON:	May be COM2: on some systems (See NOTE)
VMS	CREATE<filename>	Captures programmer output in filename. End transmission with <CTRL> Z.

2

Note: If you are using the DATA I/O and IBM PC, use PC2 and do *Select E1* on the DATA I/O. If the DATA I/O menu appears on the IBM screen, the communications link is working.

4. Dump part of the RAM contents of the programmer to the serial port. Nearly all programmers can do this. If the communications link is working, you should see the characters on the computer terminal screen.

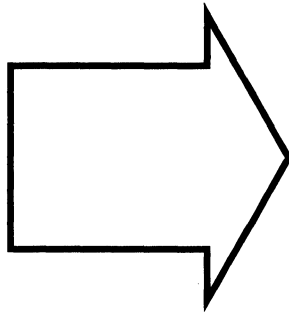


Table of Contents	vii
Software Errata	ERR
Introduction	1
Installing PALASM® 2 Software	2
PDS Syntax	3
Boolean Equation Design	3A
State Machine Design	3B
Simulation	3C
Using PALASM 2 Software	4
Installation and Operation Notes	A
Programmer Notes	B
Device Specific Syntax	C
PAL® Design File Library	D
Error Messages	E
Submitting a HAL® Design to MMI	F
PALASM 2 Syntax Diagram	G
Supplementary Software	H
JEDEC Standard No. 3	I
Release Notes	RN
Index	IX
PLEASM™ Manual	



3. PDS SYNTAX

The first step in using PALASM 2 software is for you to create a design file. To create your design file, you use a text editor such as Wordstar™ or Edlin™. Your design file contains the specifications that PALASM 2 software uses to program a PAL or PROSE device. We refer to the design file as the PAL device Design Specification. It is important to remember the acronym PDS because it is used as the filename extension for your design file. This chapter describes the syntax and structure for each of the two kinds of PDS files that PALASM 2 software accepts: Boolean equation design and state machine design.

3

IN THIS CHAPTER

<i>For a description of...</i>	<i>Refer to page...</i>
Boolean Equation Design	3A-1
State Machine Design	3B-1

PDS FILE STRUCTURE

Your input file may be in Boolean equation design or State Machine design. Both kinds of input files must be named <filename.PDS> and have the following structure.

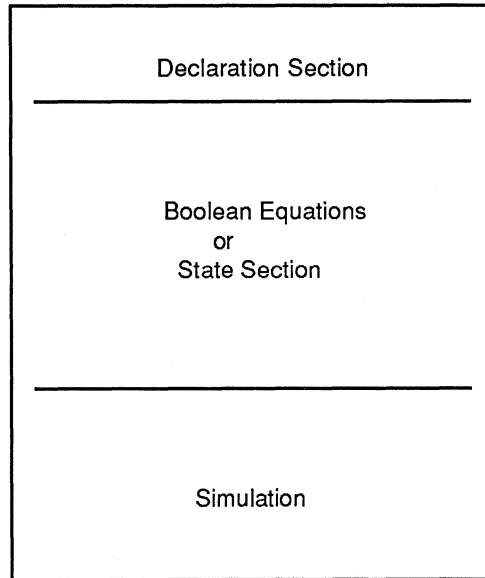


Figure 3-1: PDS File Structure

This chapter provides detail on how to create both kinds of input files using the generic PDS structure shown above. For information on simulation, turn to Chapter 3C, Simulation.

Table 3-1:
PALASM 2 Software Reserved Words

AUTHOR	NC
BEGIN	OR
CHECK	OUTPUT_ENABLE
CHIP	OUTPUT_HOLD
CLKF	PATTERN
CLOCKF	POWER_UP
CMBF	PRLDF
COMPANY	REVISION
CONDITIONS	RSTF
DATE	SETF
DECLARATION	SIMULATION
DEFAULT_BRANCH	STATE
DEFAULT_BRANCH HOLD_STATE	STRING
DEFAULT_BRANCH NEXT_STATE	THEN
DEFAULT_OUTPUT	TITLE
DO	TRACE-OFF
ELSE	TRACE-ON
END	TRST
EQUATIONS	VCC
FOR	WHILE
GND	
IF	
MASTER_RESET	
MEALY_MACHINE	
MOORE_MACHINE	

3

All MMI programmable logic device types are reserved words; for example, PAL16R8 and PMS14R21.

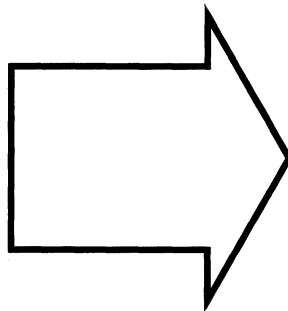


Table of Contents	vii
Software Errata	ERR
Introduction	1
Installing PALASM [®] 2 Software	2
PDS Syntax	3
Boolean Equation Design	3A
State Machine Design	3B
Simulation	3C
Using PALASM 2 Software	4
Installation and Operation Notes	A
Programmer Notes	B
Device Specific Syntax	C
PAL [®] Design File Library	D
Error Messages	E
Submitting a HAL [®] Design to MMI	F
PALASM 2 Syntax Diagram	G
Supplementary Software	H
JEDEC Standard No. 3	I
Release Notes	RN
Index	IX
PLEASM [™] Manual	



3A. BOOLEAN EQUATION DESIGN

DECLARATION SECTION

Function

You use the Declaration section to document information about the designer and part, to define pin assignments, and to define string names.

3A

Structure

The structure of the Declaration section is shown in Figure 3A-1.

Keyword	Data
TITLE	<title of design>
PATTERN	<pattern identification>
REVISION	<revision identification>
AUTHOR	<designer's name>
COMPANY	<company name>
DATE	<date of creation>
CHIP	<chip name> <PAL type> <pin list>
STRING	<string name> <text>

Figure 3A-1: Declaration Structure

CAUTION: CHIP is the only required keyword without which a fuse plot will not be generated. If you omit any of the other key words you will see a warning message, but the compiler will continue to generate a fuse plot.

Figure 3A-2 shows an example of the Declaration section.

```

TITLE      This is an example to
           illustrate the syntax
PATTERN    ABC1234_MMI REVISION  000-
           ABC1234
AUTHOR     John Doe
COMPANY    Monolithic Memories Inc.
DATE       July 5, 1986

CHIP       example_only           PAL20RS10

;PINS 1    2    3    4    5    6
           CLK  ONE /TWO THREE SET RESET

;PINS 7    8    9    10   11   12
           NC   NC   NC  WRITE READ  GND

;PINS 13   14   15   16   17   18
           /OE OUT1 OUT2  NC   NC   NC

;PINS 19   20   21   22   23   24
           /ACK /MMI  NC   NC  MEMADR VCC

STRING INITIALIZE 'RESET * ONE * /TWO'
STRING REQUEST   '/RESET * WRITE * MEMADR'
    
```

3A

Figure 3A-2: Example of the Declaration Section

SYNTAX

The following items are general rules of PDS syntax.

1. Comments may be inserted freely, and must begin with the semi-colon character (;).
2. Line length is 128 characters; all characters beyond this limit are ignored.

3. Data item name length is 24 characters; further characters are ignored.
4. Data identifiers can be any upper or lower case alphanumeric text including spaces, tabs and underscores.
5. **Do not use these special characters:** ! @ # \$ % ^ & * () - = + { } [] " ~ : ' < > ? , . The slash key (/) is used to denote active-low signals.
6. All control characters (including tabs) are treated as a single space.

Aside from these general syntactical rules, there are no special considerations for the use of the keywords TITLE, PATTERN, REVISION, AUTHOR, COMPANY, DATE. The keyword CHIP is a required part of the declaration section of a PDS file. The keyword STRING is an optional part of the declaration section with special syntactical considerations.

CHIP

CHIP is the keyword necessary to start the pin list information.

CHIP <chip name> <device type><pin list>

Example:

```
CHIP OCTAL_LATCH PAL10H20P8 A1, ...VCC
```

Chip Name

A description of the circuit. For example, OCTAL_LATCH.

Any alphanumeric word up to 14 characters containing a letter. The <chip name> is a required parameter and must be provided before the <device type> field is specified.

If you intend routing your design through ZHAL, you may specify the ZHAL device of your choice here. You may specify ZHAL20, ZHAL24, or ZHAL24_20. (ZHAL24_20 utilizes the 24-pin architecture for a 20-pin design.) If you do not spell out a device name in this section of your design file, the ZHAL program will choose an appropriate one for you by default.

Device Type

The device type is the part number of the supported PAL or PROSE device manufactured by Monolithic Memories.

3A

PAL devices with different speed/power options are given the same generic name. For example: PAL16R8, PAL16R8A, PAL16R8A2, PAL16R8A4, and PAL16R8B all have the same generic name PAL16R8.

Pin List

A list of signal names that you assign to the pins of the device. When you are assigning signal names you must keep the following points in mind.

1. Signal names cannot exceed 14 characters in length. The first character must be a letter; the remainder may be letters, numbers, or underscores. For example: 1, 2, 3, ... is an illegal pin list, but p1, p2, p3, ... are legal pin lists.
2. Signals can be specified as active-low or active-high. Active-low signals are preceded by / (/A is an active-low signal).
3. Signal names are separated by spaces or commas.
4. Special pins of the device are assigned special names. The power pin is assigned VCC and the ground pin is assigned GND. These names should come at the appropriate places in the pin list. For example, in PAL20R8, pin number 24 is

VCC, and pin number 12 is GND. If any pin is not used, it must be specified as NC (no connect).

5. Pins are listed in the order expected for DIP (dual in-line package), regardless of whether you are planning to eventually program DIP, LCC or Chip Carrier devices. Any pin reordering for other packages must be done by the programmer or other special fixture. The PAL64R32 device pinout is specified for an 84-pin package.

Note: Do not use the reserved words shown in Table 3-1, or the MMI programmable logic device types as chip signal names.

String

STRING is the keyword that introduces string identifiers in the Declaration section of the PDS file. The keyword must be repeated for each identifier.

STRING <string name> '<text to be substituted>'

Example:

```
STRING LOAD ' LD * /CIN '  
STRING CARRY ' /LD * /SET * /SET * CUP '  
STRING INPUT ' A1 + /A2 + A3 '
```

String Name

The string name is a user-defined name of up to 14 alphanumeric characters. This name has to be unique, which means that it should not be a reserved word, one of the signal names defined in the pin list, or one of the string names used elsewhere.

Text to be Substituted

This is any legal expression specified within single quotes. The text to be substituted can be of any length, must follow the syntax rules of the pin name identifiers, and must be delimited by blanks or tabs.

3A

To eliminate repeated typing of a frequently needed block of text when writing the Functional Description of the circuit, you can declare that block with an alphanumeric identifier. Then, instead of typing the full text you can use the identifier as necessary.

You can also use previously defined string names in the string declaration.

Example:

```
STRING INPUT ' A1 + /A2 + A3 '  
STRING OUTPUT ' LOAD * CARRY '
```

You can currently use a maximum of 20 unique strings within any design file.

String substitution is textual replacement, and the compiler does not try to find any logical meaning to it. You should be very specific in what you want to substitute. Using the example given, if in the equation section there is an occurrence of

/INPUT

Then after substitution the resulting expression will be

$\overline{A1} + \overline{A2} + A3$

not

$\overline{(\overline{A1} + \overline{A2} + A3)}$

because DeMorgan Expansion is not performed on string expressions. If the latter meaning is what you want, your string definition should be

STRING INPUT ' $\overline{(\overline{A1} + \overline{A2} + A3)}$ '.

BOOLEAN EQUATION INPUT

Function

You give all the implementation details of an application in the Equations section of the PDS using Boolean equations. This information will be used to generate the locations of the fuses to be disconnected during programming of the part.

Structure

The beginning of the Equations section is denoted with the keyword EQUATIONS. The remainder of this section consists of Boolean equations that have the general structure:

SIGNAL NAME assignment operator TRANSFER FUNCTION

Depending on the nature of the output signal being described, there are three basic types of equations used:

- * Combinatorial
- * Registered
- * Functional

The syntax of the Equations section, as it is applied to these three equation types, is shown in Figure 3A-3.

CAUTION: Any signal name used in the Equations section must be declared first in the Declaration section.

EQUATIONS
(Keyword marking the beginning of functional description)

Combinatorial Equation:
<signal> = Function (<signal>, <operator>)

Registered Equation:
<signal> := Function (<signal>, <operator>)

Functional Equation:
<signal> . <sfunc> = Function (<signal>, <operator>)

where

<signal>	is the name of a pin from the pin list
<operator>	is a basic operator (/, *, +, :-)
<sfunc>	is a special function associated with the output signal.

3A

Figure 3A-3: Overview of Equation Syntax

The basic operators you use are shown in Table 3A-2. They perform INVERT, AND, OR, and Exclusive-OR operations. These operators can be used to describe any logic function on the right side of the equation using sum-of-products form logic notation.

**Table 3A-1:
Basic Operators**

/ INVERT is used whenever a signal has to be inverted.
It precedes the signal to be inverted.

$/A$ means *not A*

* AND is used when ANDing two or more Boolean variables. The operation of ANDing all the signals results in a product term.

$A * /B * C$ means *A and (not B) and C*

+ OR is used when ORing two or more product terms and/or signals.

$A + /C$ means *A or (not C)*

:+: EXCLUSIVE-OR is used when exclusive - ORing two or more product terms and/or signals.

$A :+: E$ means *A exclusive-or E*

OPERATOR PRECEDENCE: / , * , + , :+:

The specific use of combinatorial, registered, and functional equations will now be described.

Combinatorial Equations

Combinatorial equations are identified by the operator =. Because combinatorial output requires no clock, outputs are based on the inputs.

$$\text{output} = \langle \text{product term} \rangle + \langle \text{product term} \rangle + \dots$$

where

$\langle \text{product term} \rangle$..is composed of $\langle \text{signal} \rangle * \langle \text{signal} \rangle * \dots$
 $\langle \text{signal} \rangle$ is represented by a declared pin name or complement.

3A

Example:

```
CHIP POLARITY_EXMPL PAL16P8
```

```
;PINS 1 2 3 4 5
      A B C D /E
```

```
;PINS 6 7 8 9 10
      /F NC NC NC GND
```

```
;PINS 11 12 13 14 15
      Y /Z W /V NC
```

```
;PINS 16 17 18 19 20
      NC NC NC NC VCC
```

```
EQUATIONS
```

```
Y = A * B + /C * D
/Z = E * F + /F * /E
/W = E
V = /F
```


The signal on the left of = is the output for which the equation is described. This output signal can be active-high (output) or active-low (/output).

On PAL devices with programmable polarity, the polarity fuse is programmed or left intact according to the polarities given on the left side of the equation and those used in the pin list. When these two polarities are the same, the fuse is programmed, giving an active-high output. When the two polarities differ, the fuse is left intact, leaving the output active-low.

In the example, equations for outputs Y and Z have the same polarity as in the pin list. On a programmable-polarity part (eg. PAL16P8), the polarity fuse would be programmed. On an active-low part (eg. PAL16L8), this would be an error.

The outputs W and V have polarities that are the reverse of those specified in the pin list, so the polarity fuse is left intact. The programmable-polarity feature allows you to describe the function in either active-low or active-high state. You do not have to transform the function using De Morgan's theorem. Refer to page 3A-17 for more information on polarity.

Registered Equations

Registered equations are identified by the operator :=.

These equations are described for outputs with a register. For example: Each output of the PAL16RP8 device is a registered output.

output := <product term> + <product term> + ...

where

<product term> ..is composed of <signal> * <signal> *....
<signal> is represented by a declared pin name or complement.

Example:

CHIP POLARITY_EXMPL PAL16RP8

```
;PINS 1 2 3 4 5
      CLK A B C D
```

```
;PINS 6 7 8 9 10
      /E /F NC NC GND
```

```
;PINS 11 12 13 14 15
      Y /Z W /V NC
```

```
;PINS 16 17 18 19 20
      NC NC NC NC VCC
```

EQUATIONS

```
Y:= A * B + /C * D
/Z:= E * F + /F * /E
/W:= E
V:= /F
```

The signal on the left side of := is the output described by the equation.

The clock to the register in most cases is a special clock pin (e.g., on the PAL16RP8 device, pin number 1 is the clock pin). On the PAL20RA10 device the clock is generated by a special product term described in a CLKF functional equation on the PAL20RA10.

The transition at the output of the register takes place on the rising edge of the clock.

This output signal can be active-high (output) or active low (/output).

3A

On programmable-polarity parts, the polarity fuse is programmed or left intact according to the polarity given on the left side of the equation and that used in the pin list. When they are the same, the fuse is programmed, which means the signal is not inverted; when they differ the fuse is left intact, which means the signal is inverted.

The programmable polarity feature allows you to describe the function in either active-low or active-high state. You do not have to transform the function using De Morgan's theorem.

Refer to page 3A-17 for a discussion on polarity.

Functional Equations

Certain PAL devices, such as the PAL20RA10, have the following programmable functions for registers:

- * Set
- * Reset
- * Clock
- * Three-state

These functions are represented by special equations in which the keyword of the special function is a suffix to the signal name:
<output.sfunc> = <product term>

Example:

$$\text{OUT.CLKF} = A * B$$

The left side of the equation identifies the function for the output defined by the right side of the equation. In the example this means that product term $A * B$ controls the Clock function of the output OUT. Because these functions use only a single product term, the OR (+) operation cannot be used. Order of appearance in a PAL Design Specification is not significant for functional equations.

The programmable functions are described below.

**SETF: The programmable
SET Function**

**RSTF: The programmable
RESET Function**

On the PAL20RA10, it is always possible to bypass the register by making the SET and RESET product terms high. There are two ways of doing this. One way is to be explicit, as follows:

```
OUT:=A + /B + D * E      ;Output defined as registered
OUT.SETF = VCC
OUT.RSTF = VCC
OUT.CLKF = GND
```

3A

The other way is to be implicit, as follows:

```
OUT = A + /B + D * E; Output defined as combinatorial
```

In the implicit case, the program XPLOT will take care of the default conditions for SETF, RSTF and CLKF.

In some cases, you might not want to use the SET and RESET functions. Being explicit:

```
OUT := A + /B
OUT.SETF = GND
OUT.RSTF = GND
OUT.CLKF = CLK
```

Being implicit:

```
OUT := A + /B
OUT.CLKF = CLK
```

The program XPLOT will take care of the default conditions and program the appropriate fuses.

Default for PAL20RA10

If the output is defined as combinatorial: the default value is VCC. If the output is defined as registered: the default value is GND.

**CLKF: The Programmable
Clock Function**

If the output is defined as combinatorial, then it has no CLKF. XPLOT will indicate an error if CLKF is defined.

Only the PAL20RA10 has a programmable clock. If the output on the PAL20RA10 is defined as registered, then CLKF must be defined. Otherwise, XPLOT will produce an error. (Define as GND to disable.)

Defaults

GND (clock absent)

**TRST: The Programmable
Three-State Function**

The default for the three-state function is VCC. You can specify this explicitly as:

OUT := A + B
OUT.CLKF = CLK
OUT.TRST = VCC

or implicitly by not specifying the three-state function.

The XPLOT program will program all the fuses.

Defaults

VCC (output enabled)

Polarity

It is important to remember that on most programmable polarity parts, the polarity fuse is located in front of the register and affects the inversion of the data path. The data path is the output of the OR gate through the polarity fuse and into the register. It does not affect the set or reset function of the output.

3A

If no output equation is defined, the polarity fuse is left intact.

Output Polarity

An output can be defined as active-high or active-low. To achieve the desired polarity on an output, the signals in your PDS design file must be defined correctly. We will now look at the factors that determine the polarity of an output.

Two factors determine the polarity of an output:

1. The signal in the pin list.
2. The occurrence of the same signal on the left side of the operator in a Boolean equation.

We will discuss the relative polarity between the signal in the pin list and the same signal in the Boolean equation.

The pin list, which is in the Declaration section of your PDS file, is where you define pin names for the input and output pins on the device. You write Boolean equations in the Equations section of your PDS file.

Figure 3A-4 shows an example of the pin list and Equations section of a PDS file.

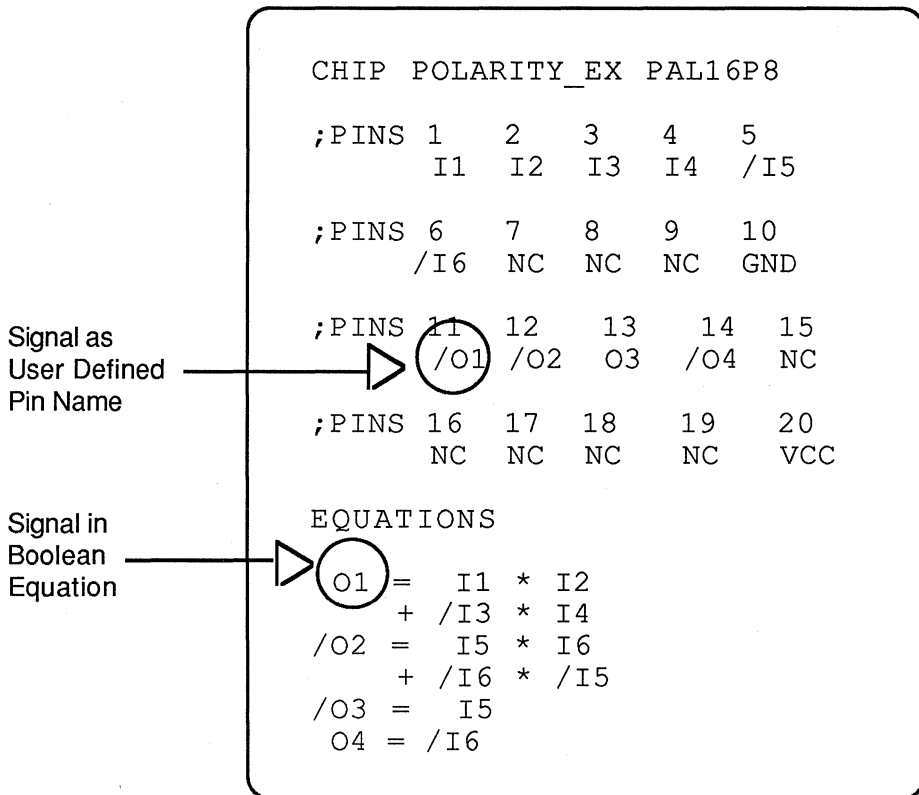


Figure 3A-4:
Pin List and Equations Section

The example in Figure 3A-4 shows that on pin 11, while the signal in the pin list is active-low (/O1), the signal in the Boolean equation is active-high (O1). This results in the output polarity being active-low (/O1). The result is summarized in Table 3A-2.

**Table 3A-2:
Results of Polarity Used in Figure 3A-4**

Output Polarity	Pin List	Boolean Equation
Active-low	Active-low	Active-high
/O1	/O1	O1

We have just seen that the relationship between the signal in the pin list and the signal in the Boolean equation has a direct bearing on the polarity of the output pin. To achieve the desired output polarity, you must define the signals in the pin list and Boolean equation appropriately.

3A

How is this done? The chart in Figure 3A-5 contains every possible combination of signals in the pin list and the Boolean equation along with the resultant polarity of the output. Use the chart as a guide when defining signals in your PDS design file.

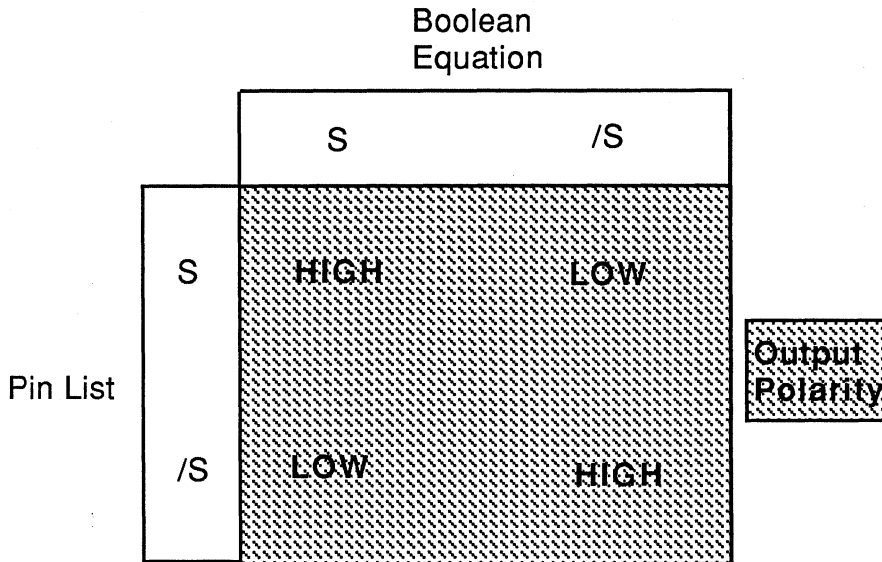


Figure 3A-5:
Signal Combinations and Output Polarity

Let us try and use this chart. If we want the output polarity to be active-high, one possible combination would be /S in the pin list and /S in the Boolean equation.

Note: While any combination in Figure 3A-5 works on a programmable polarity device, PALs such as PAL16L8 and PAL16R4 do not accept the same polarity in both the pin list and the Boolean equation. The combinations these devices accept are /S in the pin list and S in the Boolean equation or vice versa.

Figure 3A-6 summarizes all possible combinations.

```

;PINS  13  14  15  16  17  18
        01  02  /03 /04  05  06

;PINS  19  20  21  22  23  24
        07  08  09  010 011 VCC
    
```

EQUATIONS

;For active-low output

```

/O1:= <expression> ;O1 is high in the pin list,
                ;low in the Boolean equation
    
```

```

O3:= <expression> ;O3 is low in the pin list,
                ;high in the Boolean equation
    
```

;For active-high output

```

O2:= <expression> ;O2 is high in the pin list,
                ;high in the Boolean equation
    
```

```

/O4:= <expression> ;O4 is low in the pin list,
                ;low in the Boolean equation
    
```

3A

Figure 3A-6:
Summary of Output Polarities

**CHECKLIST FOR
BOOLEAN EQUATION
DESIGN FILES**

1. Is the PAL device design file free of control characters such as form feeds, and was it created as a clean ASCII file?
2. Does the keyword CHIP appear before the design name, PAL device type, and pin-list information?
3. Does the keyword EQUATIONS preface all Boolean equations used?
4. Have you defined all strings to be used as logical replacements for terms in the Boolean equations?
5. On 20-pin PAL devices, is GND specified as pin 10 and VCC as pin 20? On 24-pin PAL devices, is GND specified as pin 12 and VCC as pin 24?
6. If you are specifying an active-low PAL device, is the signal name on the left-hand side of the equation the logical opposite of the signal name specified in the pin list? Are the signal names the same for active-high parts?
7. Are you within the maximum number of product terms for any output?
8. Are you specifying .TRST equations for outputs with three-state buffers only?
9. Are you specifying .CLKF equations for PAL20RA10 designs only?
10. Are all comments preceded by a semicolon (;)?

11. Does the last line in your input file terminate with a hard carriage return? (Omitting this carriage return will cause the program to crash.)
12. If you do NOT have any errors during assembly, check your fuse plot and JEDEC output for the following:
 - a) Is the number of product terms per equation correct?
 - b) With xyz.TRST=VCC specified for a three-state buffer, are ALL fuses programmed on the three-state line? If that output is being used as an input, are all fuses intact?
 - c) For PAL devices with programmable polarity, are the polarity fuses correct as expected?
 - d) For PAL devices with product-term sharing, are the sharing fuses correct? Product-term sharing fuses are present in the fuse plots for MegaPAL, PAL20S10, PAL20RS4,8,10 devices as the two unlabeled columns of fuses at the right. They allow a pair of outputs to exclusively share a changeable fraction of the product terms available for the bank.)
 - e) For PAL devices with register bypass (MegaPAL), are the proper outputs bypassed?

3A

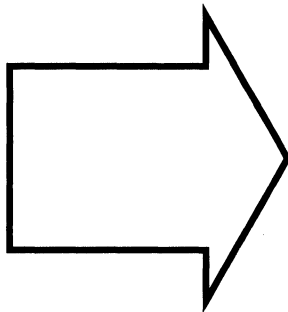


Table of Contents	vii
Software Errata	ERR
Introduction	1
Installing PALASM® 2 Software	2
PDS Syntax	3
Boolean Equation Design	3A
State Machine Design	3B
Simulation	3C
Using PALASM 2 Software	4
Installation and Operation Notes	A
Programmer Notes	B
Device Specific Syntax	C
PAL® Design File Library	D
Error Messages	E
Submitting a HAL® Design to MMI	F
PALASM 2 Syntax Diagram	G
Supplementary Software	H
JEDEC Standard No. 3	I
Release Notes	RN
Index	IX
PLEASM™ Manual	

3B. STATE MACHINE DESIGN

Note: State machine design entry is fully functional on the PMS14R21 PROSE device. Because it is currently being beta released for PAL devices, this chapter does not contain references to PAL devices. Future versions of PALASM 2 software and documentation will include state machine design entry for all Monolithic Memories PAL, PROSE, and PLS devices.

3B

Before discussing PALASM 2 software's state machine syntax, we will define a state machine and its basic operation.

A state machine is a digital device that cycles through a sequence of states in an orderly fashion. A state is a set of values measured at different parts of the circuit.

Figure 3B-1 illustrates how a synchronous state machine is implemented in PALASM 2 software.

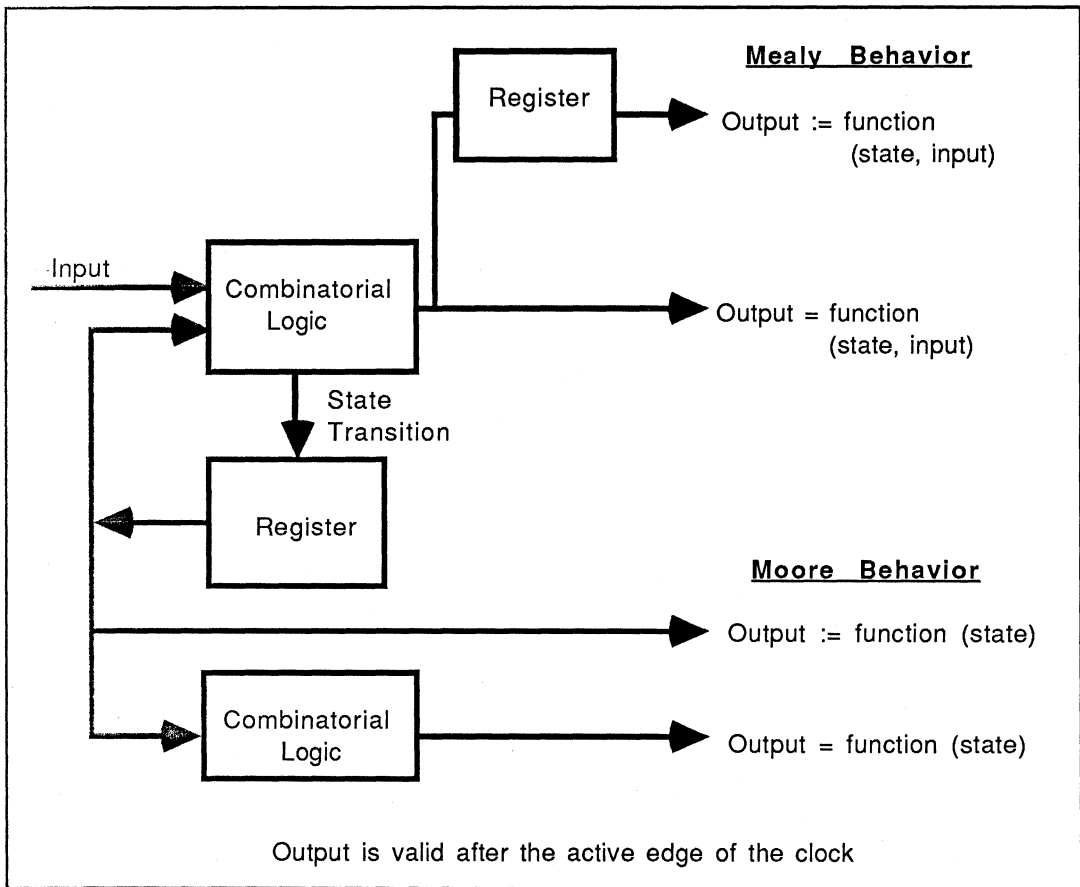
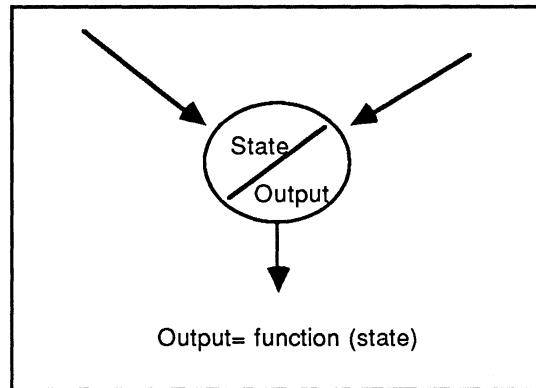


Figure 3B-1:
General Synchronous State Machine Architecture

Notice that two models are available. These are referred to as Mealy and Moore behavior models. When the output is purely a function of the current state, the state machine displays Moore behavior. Refer to Figure 3B-2 for an illustration of Moore behavior.



3B

Figure 3B-2: Moore Behavior

If the output is a function of inputs, the state transition, and the current state, the state machine displays Mealy behavior. In Mealy mode, PALASM 2 software allows either combinatorial or registered outputs to be declared. Refer to Figure 3B-3 for an illustration of Mealy behavior.

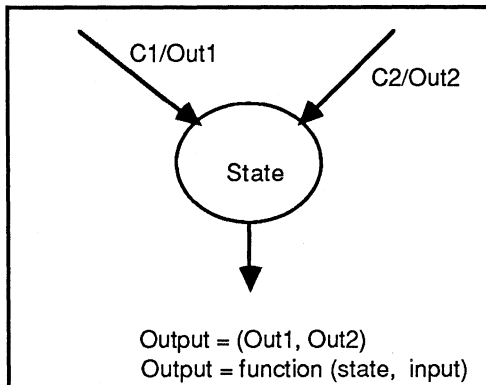


Figure 3B-3: Mealy Behavior

The basic ingredients of a state machine design are:

- * a list of state names
- * a list of conditions that cause state transitions
- * a list of outputs

DESIGNING FOR PROSE DEVICES

Currently, a PROSE device best implements a state machine design. The circuitry in a PROSE device is designed to efficiently implement a state machine definition. State machine designs are not yet fully functional on PAL devices.

MEALY AND MOORE MACHINES

In your design, you need to indicate which type of state machine you will use. Pages 3B-2 to 3B-4 and Figures 3B-1 to 3B-3 demonstrate both Moore and Mealy behavior. Study your device logic diagram and decide which model is best suited to your design.

STRUCTURE AND SYNTAX

Now that we have gone over the concept of a state machine as it is implemented in PALASM 2 software, we will discuss creating a state machine design file. Your design file is created using any text editor.

3B

If you are familiar with the PAL design specification (PDS) using Boolean equations, you will find the structure of the state machine design to be very similar.

A complete state machine design begins on page 4-24. We will step through the sections of a state machine design using parts of the same traffic controller example.

Before you begin to create your design file, study the device's logic diagram in the PAL Handbook. After you have studied and understand the device circuitry, you are ready to begin creating your design file.

Your state machine design must have the structure shown in Figure 3B-4.

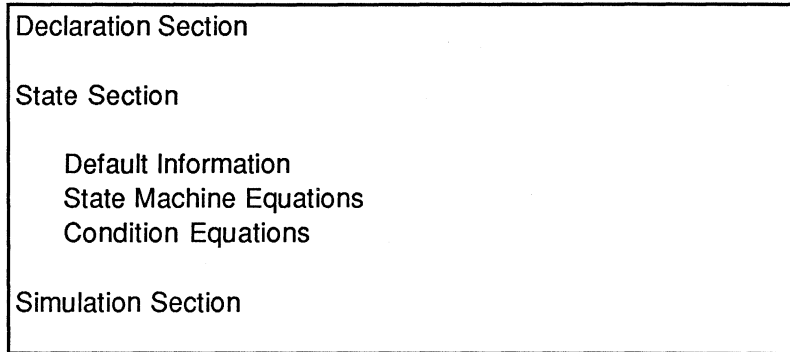
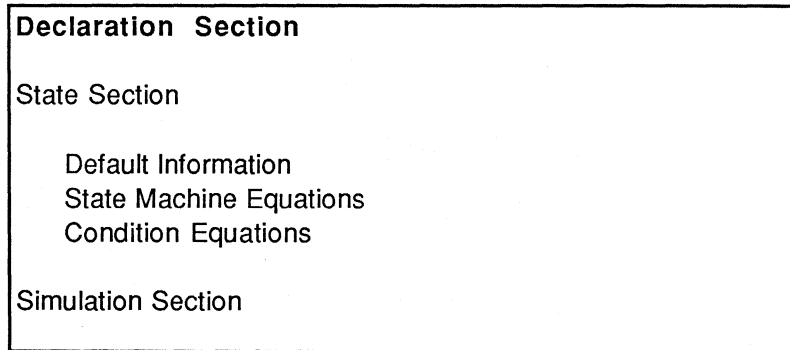


Figure 3B-4:
Structure of a State Machine Design File

We will define each section of the state machine design file beginning with the declaration section.

DECLARATION SECTION



The declaration section consists of names and titles: initial documentation about your design. It also includes some information about the device for which the design is intended.

The structure of the declaration section for both a state machine design and a Boolean equation design is the same. For further detail on the structure of this section, refer to page 3A-1.

Suppose you are designing with the PMS14R21, a PROSE device, to design a traffic controller. Figure 3B-5 shows how the declaration section would look.

3B

```

TITLE      TRAFFIC CONTROLLER
PATTERN    STATE MACHINE
REVISION   1
AUTHOR     JANE ENGINEER
COMPANY    MONOLITHIC MEMORIES
DATE       JANUARY 30, 1987
CHIP       S_MACHINE PMS14R21

;PINS      1      2      3      4      5      6
           CLOCK DCLOCK SEN1 SEN2 I2     I3

;PINS      7      8      9      10     11     12
           I4     I5     I6     I7     SDI    GND

;PINS      13     14     15     16     17     18
           RESET SDO RED1 YEL1 GRN1 RED2

;PINS      19     20     21     22     23     24
           YEL2  GRN2  01    00     MODE  VCC

```

Figure 3B-5:
State Machine Declaration Section

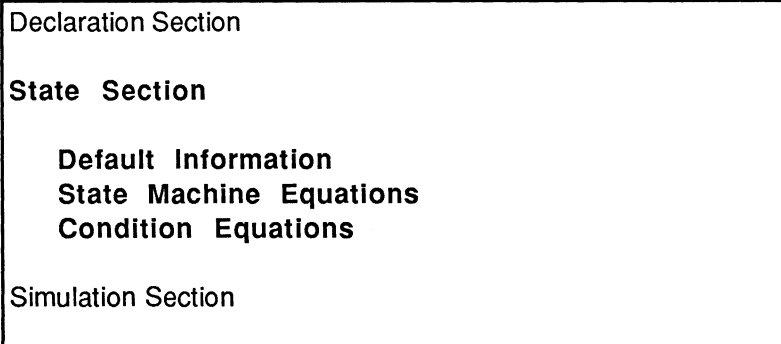
In Figure 3B-5, the CHIP statement consists of the following:

- * Required keyword: CHIP
- * Chip name: S_Machine

- * Device name: PMS14R21
- * Pin list

To completely define the pin list, refer to the circuit design in the PAL Handbook. For each pin you intend using in your design, you must assign a pin name in the Declaration section. Pin names are user defined. For further information on pin names and polarity conventions, refer to pages 3A-5 and 3A-17.

STATE SECTION



3B

The state section follows the declaration section. The information you put in the state section determines how the device is to be physically configured.

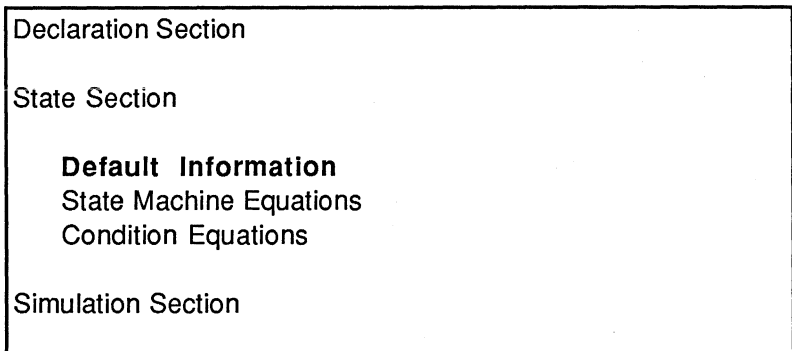
The state section is divided into three parts:

1. **Default Information:** The state section begins with the keyword STATE. This is followed by a block of information that tells the software what kind of machine you are designing and the defaults to use when either the next state or the outputs cannot be determined from the equations.

2. **State and Output Equations:** This part of the state section contains your equations that specify what conditions cause movement from each current state to a next state. You also specify what local default becomes the next state if no condition is defined.
3. **Condition Equations:** The last part of the state section begins with the keyword CONDITIONS. This is followed by condition equations.

We will discuss the three parts of the state section in the following pages.

Default Information



The structure of the default information part of the state section is shown in Figure 3B-6.

STATE (required keyword)		
MOORE_MACHINE	or	MEALY_MACHINE (default)
OUTPUT_ENABLE	or	MASTER_RESET (default)
OUTPUT_HOLD		<output pin name(s)>
DEFAULT_OUTPUT		<output pin name> /<output pin name> %<output pin name>
DEFAULT_BRANCH		<state name>
or		
DEFAULT_BRANCH		HOLD_STATE
or		
DEFAULT_BRANCH		NEXT_STATE

3B

Figure 3B-6:
Structure of Default Information

STATE

STATE is the required keyword marking the beginning of the state section.

MOORE_MACHINE or MEALY_MACHINE

MOORE_MACHINE or MEALY_MACHINE indicate the type of behavior model for which you are designing. Mealy machine is the default.

OUTPUT_ENABLE or MASTER_RESET

On a PROSE device, you have two options for configuring the outputs:

- * OUTPUT_ENABLE (fuse programmable)
- * MASTER_RESET (default)

If you select the fuse programmable OUTPUT_ENABLE option, the Preset/Output Enable (P/E) pin enables the outputs when low, and three-states the outputs when high.

When you select the MASTER_RESET option, the Preset/Output Enable (P/E) pin causes all output registers to go high asynchronously. This puts the device in POWER_UP state. (For more information on POWER_UP, refer to page 3B-21.)

OUTPUT_HOLD

You use OUTPUT_HOLD to list those pin names whose output registers retain their values on a transition from the current state to the next state. This occurs only when no new value can be determined from the output equations.

DEFAULT_OUTPUT

The DEFAULT_OUTPUT statement lists default values for each output pin when no value can be determined from the output equations. The pin can default to high, low, or don't care. The three ways to indicate what default value an output pin will have are listed in Figure 3B-7.

<pin name>	indicates that the pin defaults to high
!<pin name>	indicates that the pin defaults to low
%<pin name>	indicates that the pin defaults to don't care

3B

Figure 3B-7:
Syntax for DEFAULT_OUTPUT Statement

List a pin in either the OUTPUT_HOLD or DEFAULT_OUTPUT statements, but not in both. If a pin is listed in both statements, the software will use the statement appearing last.

DEFAULT_BRANCH

You use DEFAULT_BRANCH to specify the state to which the machine will go when no next state can be determined from your design. Your three options are listed in Figure 3B-8.

DEFAULT_BRANCH <state name>	Go to the state specified
DEFAULT_BRANCH HOLD_STATE	Hold in the current state
DEFAULT_BRANCH NEXT_STATE	Go to the next state listed in the design file

Figure 3B-8:
Syntax for DEFAULT_BRANCH Statement

STATE MACHINE EQUATIONS

Declaration Section
State Section
Default Information
State Machine Equations
Condition Equations
Simulation Section

Now that we have discussed the default information, we will discuss state machine equations. This part of the state section consists of state transition and output equations that provide detailed instructions on how the state machine will operate.

For each state you must define the following:

- * State name

- * Next states that are reachable from that state
- * Conditions under which a transition from that state to the next state can occur
- * Outputs

State Machine Equation Operators

State equations are formed using the operators shown in Figure 3B-9.

->	state transition
+	next condition
+-->	default state transition

3B

**Figure 3B-9:
State Machine Equation Operators**

Rules for State Machine Equations

Remember the following rules when writing your state machine equations:

1. Use the conditions that are defined in the CONDITIONS part of the state section to trigger state or output transitions. Condition equations follow the state and output equations in the state section of your design file.
2. **Do not use parentheses in your equations to group either state names or conditions.**

State machine syntax uses three types of equations:

- * State Equations
- * Output Equations
- * Condition Equations

State Equations

State equations describe the transitions from the named state to the next state. They may also detail the conditions that trigger transitions.

The syntax for a state equation is shown in Figure 3B-10.

`<STATE>:= <CONDITION1> -> <STATE1> + ... -> <STATE>`

Figure 3B-10:
Syntax of State Equations

Figure 3B-11 illustrates a simple state equation with a state diagram.

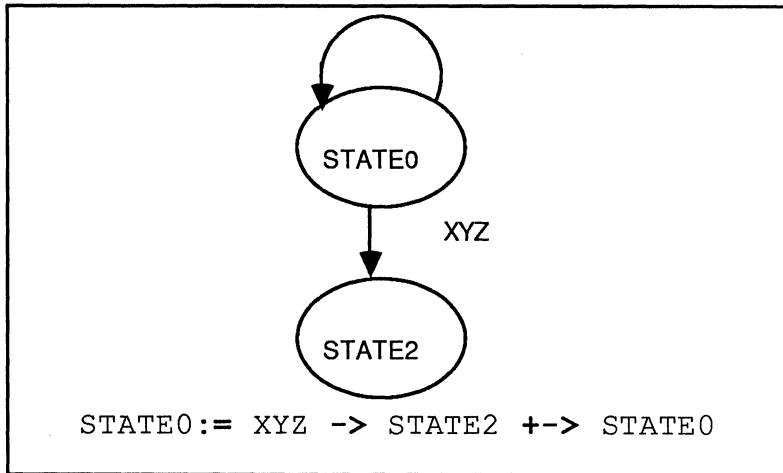
**3B**

Figure 3B-11:
Simple State Equation and Diagram

The equation in Figure 3B-11 says: when in STATE 0, if condition XYZ is true, go to STATE 2 ; otherwise stay in STATE 0. STATE 0 is the local default next state.

The equation and diagram in Figure 3B-12 show an example that does not specify a local default next state.

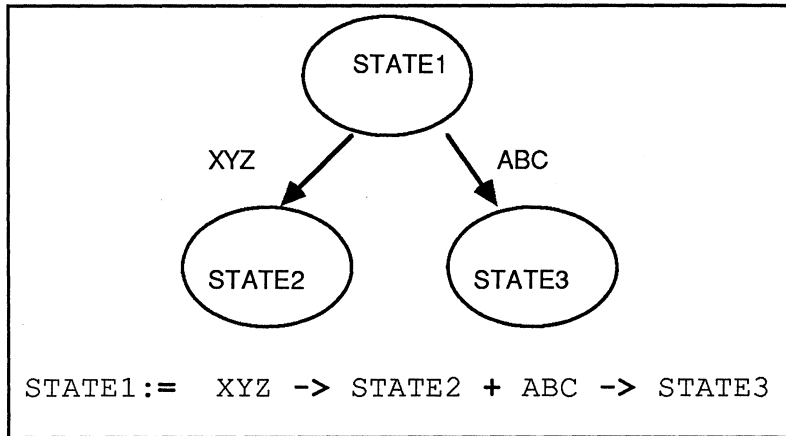


Figure 3B-12:
State Equation with No Local Default

The equation in Figure 3B-12 says: when in STATE 1, if condition XYZ is true, go to STATE 2, or if condition ABC is true, go to STATE 3. Notice the absence of a local default next state. This indicates the presence of a DEFAULT_BRANCH statement in the default information section. The equation implies that when neither condition XYZ nor ABC is true, go to the DEFAULT_BRANCH next state.

Figure 3B-13 shows a diagram for an equation taken from the traffic controller design starting on page 4-24.

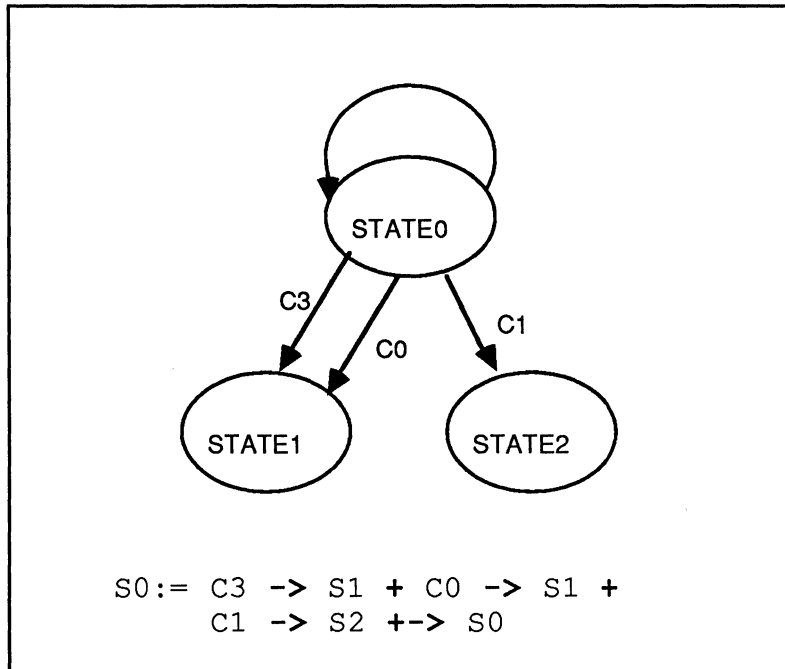
**3B**

Figure 3B-13:
State Equation and Diagram from Traffic Controller Design

The equation in Figure 3B-13 says: when in state 0 (S0), if condition C3 is true, go to state 1; or if condition C0 is true, go to state 1; or if condition C1 is true, go to state 2; otherwise, stay in state 0.

Output Equations

Output equations describe the outputs you expect from each state (Moore state machine) or the outputs you expect on transitions to next states (Mealy state machine).

Moore Machine: Output:=Function (State)

Mealy Machine: Output:=Function (State, Transition)

The syntax for a Moore machine output equation is given in Figure 3B-14.

$$\langle \text{STATE.OUTF} \rangle := \langle \text{OUTPUT1} \rangle \dots * \dots \langle \text{OUTPUT2} \rangle$$

Figure 3B-14:
Moore Machine Output Equation Syntax

The syntax for a Mealy machine output equation is given in Figure 3B-15.

$$\langle \text{STATE.OUTF} \rangle := \langle \text{CONDITION1} \rangle \rightarrow \langle \text{OUTPUT1} \rangle + \langle \text{CONDITION} \rangle \rightarrow \langle \text{OUTPUT2} \rangle + \dots \langle \text{OUTPUTS1,2,..} \rangle$$

Figure 3B-15:
Output Equation Syntax

Figure 3B-16 shows a diagram and equation for outputs on transitions, and is therefore meant for a Mealy state machine.

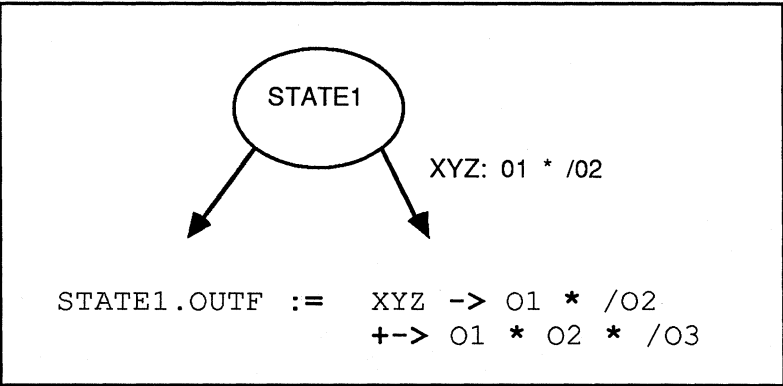


Figure 3B-16:
Mealy Machine Output Equation

The equation in Figure 3B-16 says: on a transition from STATE1, if condition XYZ is true, output O1 will be high and output O2 will be low; and since nothing is defined for output O3, it will be determined by the OUTPUT_HOLD and DEFAULT_OUTPUT statements. If condition XYZ is false, outputs O1 and O2 will be high, and output O3 will be low.

Figure 3B-17 shows an output equation and diagram for a Moore machine taken from the traffic controller example starting on page 4-24.

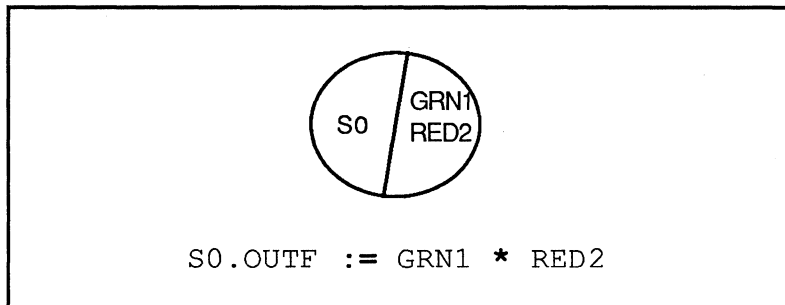
**3B**

Figure 3B-17:
Moore Machine Output Equation

The equation in Figure 3B-17 says: the output expected for state 0 is GRN1 and RED2 are true.

POWER_UP Equation

The POWER_UP equation is the first equation of your set of state and output equations.

The PROSE machine goes to the POWER_UP state upon initialization. The purpose of the POWER_UP equation is to specify what state the machine will enter during POWER_UP

initialization. This is the same state the machine enters when the P/E pin is asserted low. (See page 3B-11.)

Figure 3B-18 shows the syntax for the POWER_UP state equation.

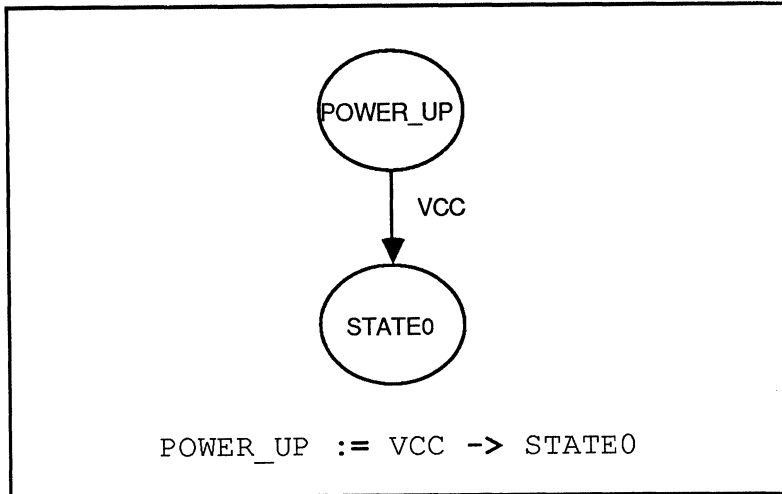
POWER_UP := VCC -> <state name>

**Figure 3B-18:
Syntax for POWER_UP State Equation**

You will notice that only one next state is specified. Currently, PALASM 2 software allows only one next state from POWER_UP.

POWER_UP For Moore And Mealy Machines

On a Moore machine, the POWER_UP equation is a state equation. Figure 3B-19 shows an example of a POWER_UP state equation for a Moore machine.

**3B**

**Figure 3B-19:
Moore Machine POWER_UP Equation**

In the example, the pin VCC defines an unconditional transition to the specified state.

On a Mealy machine, you need to have a state and an output equation for POWER_UP. The state equation is exactly like the POWER_UP equation for a Moore machine.

A Moore machine does not require an output equation for the POWER_UP state.

A Mealy machine requires an output equation to specify what output you need on the transition to the next state. The syntax for a Mealy machine POWER_UP output equation is shown in Figure 3B-20.

```
POWER_UP.OUTF := <OUTPUT>...*...<OUTPUT>...
```

Figure 3B-20:
Syntax for Mealy Machine POWER_UP Output Equation

Both POWER_UP state and output equations for a Mealy machine are shown in Figure 3B-21.

```
POWER_UP := VCC -> STATE0  
POWER_UP.OUTF := O1 * O2 */O3
```

Figure 3B-21:
POWER_UP State and Output Equations for Mealy Machine

Condition Equations

Declaration Section

State Section

Default Information

State Machine Equations

Condition Equations

Simulation Section

The third part of the state section is introduced by the required keyword **CONDITIONS** and follows the state and output equations. Condition equations define conditions used in the state section. These conditions are used to determine whether transitions occur (Moore or Mealy) and whether outputs change (Mealy only).

Figure 3B-22 shows condition equations from the traffic controller design starting on page 4-24.

CONDITIONS	
C0	= /SEN1 * /SEN2
C1	= /SEN1 * SEN2
C2	= SEN1 * /SEN2
C3	= SEN1 * SEN2

3B

Figure 3B-22:
Sample Condition Equations

This block begins with the keyword **CONDITIONS**.

Next, each condition used in the state section is defined in a condition equation. Each equation sets a condition equal to a sum-of-products of device inputs.

Rules for Condition Equations

1. Do not use parentheses to group terms in your condition equations.
2. The conditions that govern either state or output transitions from the same current state must be mutually exclusive. If they are not mutually exclusive, the transitions will conflict.

An example of mutually exclusive conditions is shown in Figure 3B-23.

$\begin{aligned} \text{COND1} &= I1 * I2 * I3 \\ \text{COND2} &= /I1 * I4 \end{aligned}$
--

**Figure 3B-23:
Mutually Exclusive Conditions**

Conflicting conditions occur when two or more conditions may be true at the same time. An example of conflicting conditions is shown in Figure 3B-24. Notice that conditions COND1 and COND2 are not mutually exclusive; both conditions may simultaneously be true when I1 is high.

$\begin{aligned} \text{COND1} &= I1 * I2 * I3 \\ \text{COND2} &= I1 * I4 \end{aligned}$

**Figure 3B-24:
Conflicting Conditions**

Where conflicting condition equations exist and both conditions become true, conflicting transitions result and the next state or output will be undefined.

SIMULATION

Declaration Section

State Section

Default Information
State Machine Equations
Condition Equations

Simulation Section

The simulation section is the last part of the design specification. Simulation is discussed in Chapter 3C.

3B

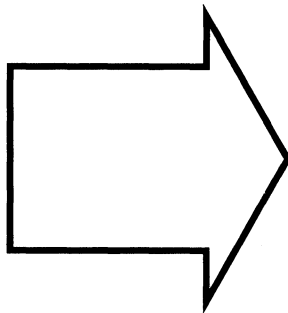


Table of Contents	vii
Software Errata	ERR
Introduction	1
Installing PALASM® 2 Software	2
PDS Syntax	3
Boolean Equation Design	3A
State Machine Design	3B
Simulation	3C
Using PALASM 2 Software	4
Installation and Operation Notes	A
Programmer Notes	B
Device Specific Syntax	C
PAL® Design File Library	D
Error Messages	E
Submitting a HAL® Design to MMI	F
PALASM 2 Syntax Diagram	G
Supplementary Software	H
JEDEC Standard No. 3	I
Release Notes	RN
Index	IX
PLEASM™ Manual	

3C. SIMULATION

After you have defined the logic of your function in terms of equations, you need to be able to verify that the equations do implement the required function. Hence, simulation is a very important part of any design cycle. The basic feature of any simulation is the ability to give a trial set of input values to your design and check the resulting outputs for correctness. PALASM 2 software provides you with this ability.

In general, logic simulation can be described using

- * PRLDF, SETF, CLOCKF, TRACE_ON, TRACE_OFF, and CHECK commands.
- * a FOR <variable> DO loop to iterate a set of commands a fixed number of times.
- * a WHILE <condition> DO loop also to iterate a set of commands till the condition is false.
- * an IF <condition> THEN ELSE for conditional branching.

PALASM 2 software has an *Event-Driven Simulator* supporting all the different PAL device architectures, both asynchronous and synchronous. The program is designed so that internal events generated by asynchronous/synchronous feedbacks and external events you generate are simulated in a very realistic way. Oscillatory conditions are also detected and reported to the user. Conflict in the expected and the actual value of any signal is an error, which is detected by the simulator and reported to the user. The simulation continues from that point using the actual value of the signal.

The PALASM 2 software language has simulation commands which are English-like words, thereby making the simulation

specification very natural to read and understand. Facilities exist for iterative looping, conditional branching, setting of signals, checking of signal values, and selective observation of signals. All these commands will be explained in this section. All the simulation results are stored in two files: a history file (<filename>.HST) and a trace file (<filename>.TRF). The history file contains the values of all the signals from the start of simulation to the end. The trace file contains the values of the signals mentioned in TRACE_ON statements and between these and following TRACE_OFF statements.

The simulation results are organized in a horizontal format resembling a timing diagram. Each page contains 40 vectors. A maximum of 512 vectors are allowed with this release of the simulator. Corresponding to each SETF and CLOCKF statement in the simulation a **g** or **c** appears on the horizontal axis in the result files. A CLOCKF statement causes the clock to go L to H to L. The **c** appears over the final L. This helps you to identify the vector corresponding to the SETF or CLOCKF statement.

SIMULATION SYNTAX OVERVIEW

Our basic philosophy in writing the simulation language was to make it easy for you to describe a function in a natural way, so that you could in turn find it easy to comprehend the behavior of the design from the simulation specification. PALASM 2 software simulation language is divided into two sections:

- * Directives
- * Structured Control

Simulation directives are commands to establish circuit inputs, clock waveforms, check for circuit outputs and capture time response waveforms as you find necessary.

The simulation section is introduced by the required keyword SIMULATION.

Simulation Directives-Syntax:

```
PRLDF      <signal list>
SETF       <signal list>
CLOCKF     <clock pin>
CHECK      <signal list>
TRACE_ON   <signal list>
TRACE_OFF
```

Structured Control Constructs-Syntax:

3C

- * FOR <variable>:= <lower limit> TO <upper limit> DO
 BEGIN <statements> END
- * WHILE <condition> DO
 BEGIN <statements> END
- * IF <condition> THEN
 BEGIN <statements> END
 ELSE
 BEGIN <statements> END

The structured control constructs are used to build up sequences of operations that repeat or are modified as a result of particular logic values or conditions. They provide the basic looping and decision branching of structured high-level programming languages. <Condition> is a Boolean expression or a mathematical equality. The condition is true if the Boolean expression is asserted or the mathematical equality is satisfied.

DETAILS OF THE SIMULATION SYNTAX

Each of the simulation statements is described below along with some related items necessary for their proper use.

PRLDF

PRLDF < output pin list >

Example:

```
PRLDF 01 /02 /03
```

The PRLDF statement is used primarily to assign logical values to, or initialize, register outputs on preloadable PAL devices. The arguments to PRLDF are registered output pin names. Uncomplemented names cause high, while names preceded by / cause low logic values to be assigned to the registered output pins.

This command affects the flow of simulation differently according to the way each registered PAL device has its preload configured. For PAL devices that have a dedicated preload pin, PRLDF successively disables the outputs, enables preload, loads the registers with the required logic values, disables preload and finally enables the outputs. For PAL devices that have their registers preloaded with supervoltages, PRLDF loads the output registers, with a P inserted in the clock field of the JEDEC vector. Simulation continues with the new values. For registered PAL devices that cannot be preloaded, PRLDF provides a convenient way of initializing registers to desired values. Since this cannot be done in hardware, no more test vectors are generated in the JEDEC file. Simulation results will, however, appear in the trace and history files.

Points to Note

1. Only registered output pin names are valid arguments to the PRLDF command.
2. The register outputs are preloaded, not the output pins.
3. When PRLDF is used on a state machine, both the state and its outputs must be preloaded.
4. On certain PAL devices such as the PAL20X4, the A version of the part preloads with supervoltages, while the standard version does not. In these cases, preloading is performed by the software, and a warning message is issued to you.

3C**SETF**

SETF <signal list>

Example:

```
SETF A /OE B /RESET /D0 D1 D2
```

The signal is set high (H) if it is not preceded by / otherwise it is set low (L). In the above example A, B, D1, and D2 are all set to H and OE, RESET, and D0 are all set to L.

The signal should only be set if you want a change from the previous value. The simulator always remembers the last value of all the signals. At the start of simulation, all signals are assumed to have a don't care value (X).

Every time a SETF statement is executed, a vector is generated and all the equations that are affected are evaluated. Any internally generated events are also detected and evaluated. With some activities, many more vectors can be generated by a single SETF statement than with others, because of feedbacks and asynchronous events. The simulator continues this process of generating vectors and evaluating equations until the system

stabilizes, that is, until there are no more changes in the output signals or no events are generated. If the system fails to stabilize after ten iterations, then an oscillatory condition is detected, and the simulation halts.

CLOCKF

CLOCKF <list of clock signals>

Example:

```
CLOCKF CLK1 CLK2
```

The **CLOCKF** statement has the list of clock signals (dedicated clock pins) to which a clock pulse is to be applied. Only the clock pins of the device can be used in the **CLOCKF** statement; any other pin is an illegal signal for this statement.

Each **CLOCKF** statement corresponds to a pulse going from low to high to low. Thus two vectors are generated, and during the positive edge transition, the new value of the registers being clocked is transferred to the output. No action takes place for the registers that are not clocked.

At every **CLOCKF** statement, internally generated events and asynchronous events are detected; and if they are present, more vectors are generated. The operation of **CLOCKF** is similar to the **SETF** statement except that **CLOCKF** goes through a pulse rather than a level.

Using the **SETF** statement, initialize the clock pin to low before using **CLOCKF**:

```
SETF <clock pin or /clock pin>
```

If the clock pin has a high value at the first **CLOCKF** statement, an error occurs.

CHECK

CHECK <signal list>

Example:

```
CHECK Q0 /Q1 /Q2
```

CHECK lets you keep track of simulation results. The signals in the CHECK statement are the high or low output signals that you want to check. In the above example, you want to check if Q0 is high and Q1 and Q2 are low. Again, a signal without / is to be checked for high and a signal with / is to be checked for low.

Whenever a CHECK statement is executed, the simulator compares the actual value and the expected value of a particular signal. If they are equal then no action is taken. Otherwise, an error is reported and the simulator continues assuming the actual value. CHECK reports the error placing a ? in the vector where the error occurred as well as a vector number. The history and trace files will contain the ? at this particular location.

3C

CHECK is a powerful statement that should be used at important points in your simulation for debugging your design.

TRACE_ON

TRACE_ON <signal list>

Example:

```
TRACE_ON /OE SET RESET D0 D1 D2 D3 /Q0  
/Q1 /Q2
```

This statement contains the signals that you want to have listed in the trace file. The signals will be listed in the same order and with the same polarity as present in the TRACE_ON statement. This list of signals will be active until the next TRACE_OFF statement or until the end of the simulation specification. New signals can

be traced on after the TRACE_OFF statement. This statement helps you group the signals more naturally for debugging purposes. For example, all control signals can be grouped together, then all data signals can be grouped together, and then all output signals can be grouped together. This makes the observation of the results in the trace file very easy.

TRACE_OFF

TRACE_OFF

This statement traces off all the signals mentioned in the latest TRACE_ON statement. After this statement, no more results are added to the trace file until the next TRACE_ON statement is executed. Thus none of the results between the current TRACE_OFF statement and the next TRACE_ON statement are displayed in the trace file.

This breaks your results into time frames which is critical for debugging. It should be remembered that the history file contains all the information generated from the start of simulation to the end of simulation. The signals are in the same order and of the same polarity as in the pin list of the CHIP statement.

FOR Loop

```
FOR <index var> := <lower limit> TO <upper limit> DO
  BEGIN
    <statements>
    ...
  END
```

Example:

```
FOR J:= 3 to 8 DO
  BEGIN
    SETF A /B CLOCKF clk
  END
```

The FOR loop allows a very powerful, repetitive execution of statements. Many statements can be embedded in a FOR loop—even another FOR statement with a different indexing variable. You can generate many vectors just by increasing the limits of this loop. The <lower limit> should be less than or equal to the <upper limit>. All the limit values should be greater than or equal to zero. You cannot use negative values for the limits. The loop is not executed if the conditions expressed in the limits are equal.

IF...THEN...ELSE Loop

<pre>IF <cond> THEN BEGIN <Statements> END ELSE BEGIN <Statements> END</pre>	or	<pre>IF <cond> THEN BEGIN <Statements> END</pre>
--	----	--

3C

Example:

```
IF J= 5 THEN
  BEGIN
    CHECK Q0
  END
ELSE
  BEGIN
    CHECK /Q0
  END
```

There are two variations of this statement. In the first, there is an ELSE clause, and in the second there is no ELSE clause. If the <condition> is true the THEN clause is executed; otherwise the ELSE clause is executed. If there is no ELSE clause, then the simulation executes the next statement after the IF

statement. Condition expressions cannot contain nested parentheses.

The <condition> can be any mathematical equality (=, >, <, >=, <=, <>)

Example 1:

```
IF (I<2) THEN
```

The <condition> can also be any Boolean expression.

Example 2:

```
IF (DRDY * /CLR) THEN
```

In Example 1, the condition of I less than 2 is checked. In Example 2, the expression (DRDY * /CLR) is evaluated; if it is true then the condition is true.

WHILE...DO Loop

```
WHILE<condition>DO  
  BEGIN  
    <statements>  
  END
```

The WHILE loop allows a repetitive execution of statements that may be controlled by evaluation of logic conditions present within the PAL device. Many statements can be embedded in a WHILE loop including even other looping constructs. The WHILE loop is used to iterate a set of commands until the condition is false.

The <condition> can be any Boolean expression of logic signals.

KEY POINTS TO NOTE

1. All signals are assumed to be don't care at the start.
2. Initialize all your control signals (like three-state, Preload and Clock) to their default values. For example: if the 3-state pin is not initialized to the default condition, then the simulation will give erroneous results.
3. If the 3-state pin is, say, /OE then SETF OE will enable the outputs and SETF /OE will three-state the outputs.
4. If the 3-state pin is, say, OE then SETF /OE will enable the outputs and SETF OE will three-state the outputs.
5. Points 3 and 4 also apply to the preload pins.
6. When using PAL20RA10 remember the following points:

```

Q0 := A * B
Q0.CLKF = CLK
Q0.RSTF = RESET
Q0.SETF = SET
SETF SET /RESET      ;The register Q0 is set to H and so
                      ; the output pin will go L.
SETF RESET /SET      ;The register Q0 is set to L and so
                      ; the output pin will go H.

```

The data path of this PAL device is treated in the normal way because the polarity fuse is in front of the register. Any difference in the polarity between the signal in the pin list and the left side of the equation is taken care of by the simulator.

It is only in the case of functional events that the value of the register and the value of the pin are inverted.

RULES FOR STATE MACHINE SIMULATION SYNTAX

These rules show the difference between Boolean equation and state machine simulation syntax.

1. In a Boolean equation design, you use the PRLDF, CHECK, TRACE_ON, WHILE and IF statements to reference the value of an output. In a state machine design, you may use these statements to reference states *as well as outputs*.

Let us look at an example of a PRLDF statement used to reference a state as well as outputs from that state. In Figure 3C-1, note that the same statement is used to PRLDF a state and the outputs from that state.

```
PRLDF STATE_ONE 01 /02 /03
```

Figure 3C-1:
PRLDF Statement

This brings us to rule #2.

2. When you PRLDF a state, remember to PRLDF the outputs associated with that state in the *same* PRLDF statement. Use one PRLDF statement for one state and as many signals as fit within 128 characters. The more PRLDF statements you use, the more vectors the software creates.
3. The two history output files from simulation are <filename>.TRF (if TRACE_ON is used) and <filename>.HST. On the PMS14R21, these files contain information that is not found in the output files of a Boolean equation design: the state of the machine at each point in the simulation.

For a complete state machine design example that includes simulation, refer to the traffic controller specification starting on page 4-24.

3C

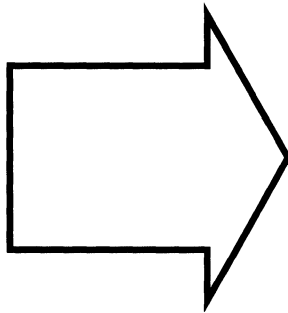


Table of Contents	vii
Software Errata	ERR
Introduction	1
Installing PALASM [®] 2 Software	2
PDS Syntax	3
Boolean Equation Design	3A
State Machine Design	3B
Simulation	3C
Using PALASM 2 Software	4
Installation and Operation Notes	A
Programmer Notes	B
Device Specific Syntax	C
PAL [®] Design File Library	D
Error Messages	E
Submitting a HAL [®] Design to MMI	F
PALASM 2 Syntax Diagram	G
Supplementary Software	H
JEDEC Standard No. 3	I
Release Notes	RN
Index	IX
PLEASM [™] Manual	

4. USING PALASM 2 SOFTWARE

This chapter describes how to complete the process of creating a PAL circuit after you have written your PAL device Design Specification (PDS). In addition, a few example PDS files are provided at the end of this chapter. A good way to begin using the program is by trying a few example files.

Note: Example PDS files are available on an IBM-PC disk. If you don't have a copy, contact your local Monolithic Memories sales office.

The procedure described in this chapter refers to non-menu mode. Menu users will find the function key options clearly defined on the main menu.

4

GENERAL PROCEDURE

The steps in using PALASM 2 software to create a circuit are as follows:

1. Create the PAL Design Specification (PDS) file. Consult chapter 3A of this manual for detailed instructions.
2. Run the parser.
3. Run the minimization program (optional).
4. Assemble the JEDEC file.
5. Run the simulation program.
6. Download the file to the programmer.
7. Program and test the part.

Figure 4-1 shows a flowchart of the process.

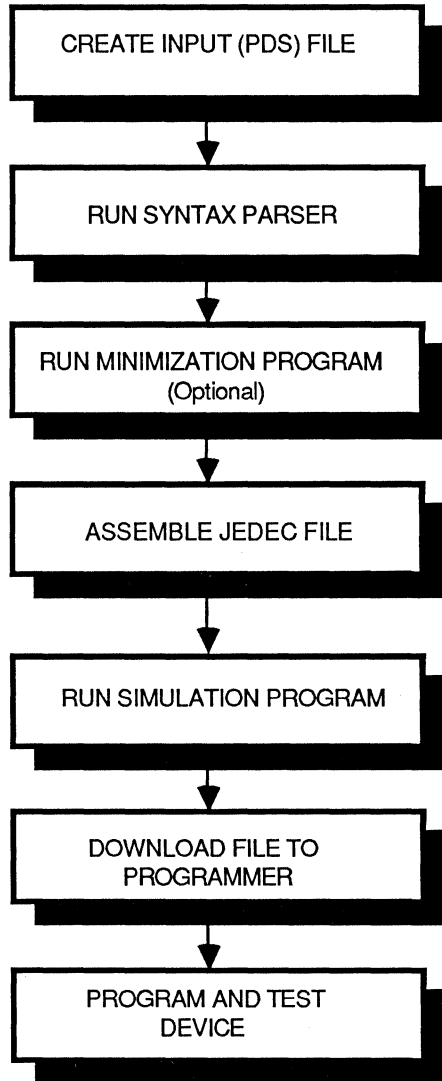


Figure 4-1:
Device Programming Flowchart

Step 1: Create the PDS File

You must describe the circuit you want as a series of Boolean equations in sum-of-products form, or in a set of state machine equations. Part of this process is deciding on the number of inputs and outputs that the circuit requires. Once you have this information, choose a PAL device with the right number of inputs, outputs, and product terms.

Using a text editor, create a PAL device design specification (PDS) file. The file name must adhere to the conventions of the operating system you are running PALASM 2 software on and should end with the suffix **.PDS**.

Note: If you create your PDS file using a word processor (such as WORDSTAR), use it in non-document mode. You must create a *clean* ASCII file free of embedded control characters.

Step 2: Run The Parser

1. Insert the disk containing the executable files in drive B.
Insert the disk containing your PDS file in drive A.

Make sure that the operating system is looking at both drives for command files. The MS-DOS command

```
PATH A:\;B:\;
```

will take care of this requirement. (If you are using a hard disk, specify drive C instead of B.)

2. Enter PALASM2 <CR

The program will prompt you to give the name of your PDS file.

3. Enter <filename.ext> <CR>

Enter the name of your PDS file. (Full path names are not supported by the software. Copy the file to your directory.)

4. Enter Y <CR> (Create error file.)
5. Enter Y <CR> (Echo PDS to terminal.)
6. Enter N <CR> (Establish compatibility with PDSCNVT-generated files.)
7. Enter N <CR> (Generate a brief fuse plot.)

The system then checks the syntax of the design file. PALASM 2 software creates an intermediate file, PALASM2.TRE, on the default drive. Syntax errors are listed to the terminal (and to PALASM2.ERR if 4 above is answered YES), along with an explanation of locations and likely causes. Brief fuse plots do not contain entries for phantom fuses and unprogrammed product terms.

4

Step 3: (Optional) Logic Minimization

Performing this step will automatically minimize your logic equations and convert state machine language to Boolean equations. Minimization helps you better utilize the space on your device.

(PALASM automatically minimizes state machine designs on the PMS14R21.)

Enter MINIMIZE <CR> (run Minimize software)

Note: PALASM 2 software currently supports minimization for a limited number of devices. Please refer to Table 1-1 for a list of supported devices.

Step 4: Assemble the JEDEC File

Enter XPLOT <CR> (run fuse plot software)

The system now produces the fuse plot and JEDEC output files. XPLOT reads the intermediate file PALASM2.TRE and processes each of the equations sequentially. If any inconsistencies between the design file and the PAL device architecture are detected, messages will be displayed. The resulting XPLOT and JEDEC file will be listed by name and may be used to program PAL devices.

Optional Step: Disassemble the JEDEC File

(This step can be run either before or after Step 5: Simulation. The program, however, does not currently disassemble the simulation part of the JEDEC file.)

Enter JEDMAN <CR> (run JEDEC disassembly program)

The system prompts you to enter your JEDEC filename. The filename should have a .JED or .JDC extension. The next prompt asks you if you wish to enter a *new* JEDEC filename. This gives you the option of manipulating a JEDEC file that has been used for another device.

The program presents you with three options:

* *Disassembly*

This option enables you to disassemble a JEDEC file. Your output is a Boolean equation .PDS file.

- * *Recalculation of checksums*

You can manipulate a JEDEC file directly using this option.

- * *Conversion of PAL22V10 JEDEC to PAL32VX10 JEDEC*

The program JEDMAN takes a PAL22V10 JEDEC file and converts it to a PAL32VX10 JEDEC file.

Step 5: Simulation

Enter SIM <CR> (run simulation software)

The system next produces the history (.HST) and trace (.TRF) files indicated by simulation commands and directives of the PDS file. SIM also reads PALASM2.TRE, but it requires the full data structure to begin device simulation. The simulator also reads the JEDEC file produced by XPLOT and add simulation vectors at the end, updating the necessary checksums (OUTPUT in filename JDC).

4

Step 6: Download the File to the Programmer

Refer to Appendix B and your programmer manual.

Step 7: Program and Test the Part

Refer to your programmer manual.

Optional Step: Route Logic

Enter ZHAL <CR>

The program attempts to route the logic equations in a ZHAL array. If the process is successful, an appropriate message is given. If the process fails, you see the message `FORCE ROUTE FAILURE` along with the probable cause of failure.

The message that appears on your screen when the process fails also indicates remedies, such as rearranging the device pinout. ZHAL devices have a limited number of product terms; hence the reason for failure is often an overlap of product terms between adjacent pins indicated. ZHAL processing can be done for 20-pin and 24-pin devices. By default, the software counts the pins indicated in your design file and chooses the appropriate device. You can, however, specify the device of your choice in your design file.

Note: The ZHAL utility is not part of the regular package you get when you order PALASM 2 software. You may, however, order the ZHAL program from your local Monolithic Memories sales office.

Error Messages

Most problems with design file syntax are caught by the front-end PALASM 2 software program. Messages such as `Malformed Variables` are given along with a line number location. The line number can be identified by observing the echo output listing generated by the program, if this option has been selected.

You might also find it useful to have a printed copy of error messages. You can make one by diverting the program output to a file and then printing that file.

Other errors in PDS syntax are only diagnosed during processing by one or more back-end programs. Supplying too many or too few pin names, extra product terms, or invalid pin usage are only diagnosed by the XPLOT program. You will need to run each of these programs to catch all potential error conditions. Appendix E lists all error messages generated by the program.

All intermediate files generated by PALASM 2 software are directed to listed drive (A:). For the input file 4CNT, the intermediate files generated by PALASM 2 software are as follows:

4CNT.PDS	Original user input file
4CNT.XPT	Fuse plot
4CNT.JED	JEDEC file with all checksums inserted
4CNT.HST	Complete Simulation History file
4CNT.TRF	Selective Trace of Simulation
PALASM2.TRE	Intermediate design description
4CNT.JDC	Complete JEDEC file containing simulation vectors

4

Note: For designs involving the MegaPAL64R32, an additional file with extension .VRX is generated. It contains the specific fuse plot to be downloaded to the VARIX programmer. A full (not brief) fuse plot must be specified for VARIX downloading.

DESIGN EXAMPLES

Boolean Equation Design

For an example of a state machine design, turn to page 4-24.

The following examples were generated from the 4CNT.PDS supplied on the demonstration examples disk. (Contact your local Monolithic Memories sales office if you don't have a copy of the examples disk.) The input file, which includes simulation equations, is followed by history, trace, XPLOT, and JEDEC files.

Input File

```
Title      4Bit_Counter
Pattern    4cnt.pds
Revision   B
Author     J. Engineer
Company    Monolithic Memories Inc., Santa Clara, CA
Date       1/14/85
```

```
CHIP 4BitCounter PAL16RP4
```

```
;PINS  1   2   3   4   5
        CLK UP  AI  BI  CI

;PINS  6   7   8   9  10
        DI CLR LOAD NC  GND

;PINS  11  12  13  14  15
        /OC NC  NC  D   C

;PINS  16  17  18  19  20
        B   A  NC  NC  VCC
```

EQUATIONS

```

A := /A*/B*/C*/D*/UP*/LOAD*/CLR      ;When CLR=1, A=0.
+ /A* B* C* D* UP*/LOAD*/CLR        ;Else it will count
+ A* B* /D* /UP*/LOAD*/CLR          ;UP or DOWN.
+ A*/ B* C* UP*/LOAD*/CLR
+ A* /C* UP*/LOAD*/CLR
+ A* D* /UP*/LOAD*/CLR
+ LOAD*/CLR* AI                      ;New value is loaded
                                       ;when LOAD=1, CLR=0.

B := /B*/C*/D*/UP*/LOAD*/CLR        ;When CLR=1, B=0.
+ /B* C* D* UP*/LOAD*/CLR          ;Else it will count.
+ B* C*/D* /LOAD*/CLR
+ B*/C* UP*/LOAD*/CLR
+ B* D*/UP*/LOAD*/CLR
+ LOAD*/CLR* BI                      ;New value is loaded
                                       ;when LOAD=1, CLR=0

C := /C*/D*/UP*/LOAD*/CLR          ;When CLR=1, C=0.
+ /C* D* UP*/LOAD*/CLR            ;Else it will count.
+ C*/D* UP*/LOAD*/CLR
+ C* D*/UP*/LOAD*/CLR
+ LOAD*/CLR* CI                      ;New value is loaded
                                       ;when LOAD=1, CLR=0.

D := /D* /LOAD*/CLR                 ;Count
+ LOAD*/CLR* DI                     ;New value is loaded
                                       ;when LOAD=1, CLR=0.

```

4

SIMULATION

```

TRACE_ON AI BI CI DI LOAD CLR UP A B C D
SETF LOAD /CLR AI BI CI DI OC      ;Load all registers
CLOCKF CLK                          ;to HIGH and count up

SETF CLR                             ;Clear all registers
CLOCKF CLK

```



```

SETF /CLR UP /LOAD           ;Start counting up

FOR I:= 1 TO 16 DO           ;Count up 16 clock
BEGIN                         ;cycles
CLOCKF CLK
END

SETF LOAD /CLR /UP AI BI CI DI ;Load all registers
CLOCKF                       ;to HIGH and count
SETF /LOAD                   ;down
FOR I:= 1 TO 16 DO           ;Count down 16 clock
BEGIN                         ;cycles
CLOCKF CLK

SETF LOAD CLR AI /BI CI /DI  ;Test setting LOAD
CLOCKF CLK                   ;and CLR on at the
                             ;same time.

```

```
SETF /OC
```

```
TRACE_OFF
```

;The 4-bit counter counts up or down and has the
;clear and load capability. The clear operation
;overrides count and load. The counter counts up
;whenever CLR=low, LOAD=low, and UP=high. It counts
;down whenever CLR=low, LOAD=low, and UP=low.

XPLOT File

The following fuse plot of the 4Bit_Counter is produced by PALASM 2 software. It shows the location and condition of each fuse in the device.

X means the fuse is intact.
- means the fuse is programmed.

PALASM XPLOT, V2.22 - MARKET RELEASE (11-19-86)
 (C) - COPYRIGHT MONOLITHIC MEMORIES INC, 1986

Title 4Bit_Counter
 Pattern 4cnt.pds
 Revision B
 Author Mehrnaz Hada, Bill Hollo
 Company Monolithic Memories Inc.
 Date 1/14/85

4

PAL16RP4
 4BITCOUNTER

			11	1111	1111	2222	2222	2233
0123	4567	8901	2345	6789	0123	4567	8901	
0	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX
1	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX
2	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX
3	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX
4	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX
5	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX
6	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX
7	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX

```

 8 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
 9 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
10 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
11 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
12 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
13 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
14 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
15 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

16 -X-- ---- ---X ---X ---X -X-X -X-- ----
17 X--- ---- ---X --X- --X- -XX- -X-- ----
18 -X-- ---- --X- --X- ---- -X-- -X-- ----
19 ---- ---- --X- ---X ---- -XX- -X-- ----
20 X--- ---- --X- ---- ---X -X-- -X-- ----
21 ---- ---- --X- ---- --X- -X-X -X-- ----
22 ---- X--- ---- ---- ---- -X-- X--- ----
23 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

24 -X-- ---- ---- ---X ---X -X-X -X-- ----
25 X--- ---- ---- ---X --X- -XX- -X-- ----
26 ---- ---- ---- --X- --X- -X-X -X-- ----
27 X--- ---- ---- --X- ---X -X-- -X-- ----
28 -X-- ---- ---- --X- ---- -XX- -X-- ----
29 ---- ---- X--- ---- ---- -X-- X--- ----
30 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
31 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

32 -X-- ---- ---- ---- ---X -X-X -X-- ----
33 X--- ---- ---- ---- ---X -XX- -X-- ----
34 X--- ---- ---- ---- --X- -X-X -X-- ----
35 -X-- ---- ---- ---- --X- -XX- -X-- ----
36 ---- ---- ---- X--- ---- -X-- X--- ----
37 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
38 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
39 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

40 ---- ---- ---- ---- ---- -X-X -X-- ----
41 ---- ---- ---- ---- X--- -X-- X--- ----
42 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

```

```
43 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
44 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
45 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
46 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
47 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

48 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
49 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
50 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
51 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
52 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
53 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
54 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
55 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

56 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
57 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
58 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
59 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
60 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
61 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
62 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
63 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
```

4

```
OUTPUT PINS: 11111111
              23456789
```

```
POLARITY FUSE: XX----XX
```

```
TOTAL FUSES BLOWN: 548
```

History File

PALASM SIMULATION, V2.22 - MARKET RELEASE (11-19-86)
 (C) - COPYRIGHT MONOLITHIC MEMORIES INC, 1986
 PALASM SIMULATION HISTORY LISTING

Title : 4Bit_Counter
 Pattern : 4cnt.pds
 Revision : B
 Author : Mehrnaz Hada, Bill Hollo
 Company : Monolithic Memories Inc.
 Date : 1/14/85

4BITCOUNTER

Page : 1

	g	cg	cg	c	c	c	c	c	c	c	c	c	c
CLK	XHHL LHHL LH	HLHL LHHL LH	LHHL LHHL LH	LHHL LHHL LH	LHHL LHHL LH	LHHL LHHL LH	LHHL LHHL LH	LHHL LHHL LH	LHHL LHHL LH	LHHL LHHL LH	LHHL LHHL LH	LHHL LHHL LH	LHHL LHHL LH
UP	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH
AI	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH
BI	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH
CI	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH
DI	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH
CLR	LLLLHHHLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL
LOAD	XXXXXXXXHLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL
GND	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL
/OC	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL
D	XXHHHLLLL	HHLLLHHL	LLHHLLHH	HLHHLLHH	HLHHLLHH	HLHHLLHH	HLHHLLHH	HLHHLLHH	HLHHLLHH	HLHHLLHH	HLHHLLHH	HLHHLLHH	HLHHLLHH
C	XXHHHLLLL	LLLHHHHL	LLLLHHHH	LLLLHHHH	LLLLHHHH	LLLLHHHH	LLLLHHHH	LLLLHHHH	LLLLHHHH	LLLLHHHH	LLLLHHHH	LLLLHHHH	LLLLHHHH
B	XXHHHLLLL	LLLLLLLLL	HHHHHHHH	HHHHHHHH	HHHHHHHH	HHHHHHHH	HHHHHHHH	HHHHHHHH	HHHHHHHH	HHHHHHHH	HHHHHHHH	HHHHHHHH	HHHHHHHH
A	XXHHHLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL	LLLLLLLLL
VCC	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH	XXXXXXXXHH

4BITCOUNTER

Page : 2

	c	c	c	c	c	cg	cg	c	c	c	c	c	c
CLK	H	L	H	H	L	H	H	L	H	H	L	H	H
UP	H	H	H	H	H	H	H	H	H	H	H	H	H
AI	H	H	H	H	H	H	H	H	H	H	H	H	H
BI	H	H	H	H	H	H	H	H	H	H	H	H	H
CI	H	H	H	H	H	H	H	H	H	H	H	H	H
DI	H	H	H	H	H	H	H	H	H	H	H	H	H
CLR	L	L	L	L	L	L	L	L	L	L	L	L	L
LOAD	L	L	L	L	L	L	L	L	L	L	L	L	L
GND	L	L	L	L	L	L	L	L	L	L	L	L	L
/OC	L	L	L	L	L	L	L	L	L	L	L	L	L
D	H	H	H	L	L	H	H	H	L	L	L	L	H
C	H	H	H	L	L	L	L	L	L	L	L	L	L
B	L	L	L	H	H	H	H	H	H	H	H	H	H
A	H	H	H	H	H	H	H	H	H	H	H	H	H
VCC	H	H	H	H	H	H	H	H	H	H	H	H	H

4BITCOUNTER

Page : 3

	c	c	c	c	c	c	c	c	c	g	cg
CLK	HHLHHLHHLH	HLHHLHHLHH	LHHLHHLHHL	LHHL							
UP	LLLLLLLLLL	LLLLLLLLLL	LLLLLLLLLL	LLLL							
AI	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHH							
BI	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	LLLL							
CI	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHH							
DI	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	LLLL							
CLR	LLLLLLLLLL	LLLLLLLLLL	LLLLLLLLLL	HHHH							
LOAD	LLLLLLLLLL	LLLLLLLLLL	LLLLLLLLLL	HHHH							
GND	LLLLLLLLLL	LLLLLLLLLL	LLLLLLLLLL	LLLL							
/OC	LLLLLLLLLL	LLLLLLLLLL	LLLLLLLLLL	LLLLH							
D	HLLLHHHLLL	HHLLLHHHL	LLHHLLLHH	HLLZ							
C	LLLHHHHHH	LLLLLLHHH	HLLLLLLLHH	HLLZ							
B	LLLHHHHHH	HHHHHLLL	LLLLLLLHH	HLLZ							
A	HHHLLLLLLL	LLLLLLLLLL	LLLLLLLHH	HLLZ							
VCC	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHH							

Trace File

PALASM SIMULATION, V2.22 - MARKET RELEASE (11-19-86)

(C) - COPYRIGHT MONOLITHIC MEMORIES INC, 1986

PALASM SIMULATION SELECTIVE TRACE LISTING

Title : 4Bit_Counter
 Pattern : 4cnt.pds
 Revision : B
 Author : Mehrnaz Hada, Bill Hollo
 Company : Monolithic Memories Inc.
 Date : 1/14/85

4BITCOUNTER

Page : 1

	g	cg	cg	c	c	c	c	c	c	c	c	c	c
AI	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH
BI	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH
CI	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH
DI	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH
LOAD	HHHHHHHHLL	LLLLLLLLLL	LLLLLLLLLL	LLLLLLLLLL	LLLLLLLLLL	LLLLLLLLLL	LLLLLLLLLL	LLLLLLLLLL	LLLLLLLLLL	LLLLLLLLLL	LLLLLLLLLL	LLLLLLLLLL	LLLLLLLLLL
CLR	LLLLHHHHLL	LLLLLLLLLL	LLLLLLLLLL	LLLLLLLLLL	LLLLLLLLLL	LLLLLLLLLL	LLLLLLLLLL	LLLLLLLLLL	LLLLLLLLLL	LLLLLLLLLL	LLLLLLLLLL	LLLLLLLLLL	LLLLLLLLLL
UP	XXXXXXXXHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH
A	XXHHHHLLLL	LLLLLLLLLL	LLLLLLLLLL	LLLLLLLLLL	LLLLLLLLLL	LLLLLLLLLL	LLLLLLLLLL	LLLLLLLLLL	LLLLLLLLLL	LLLLLLLLLL	LLLLLLLLLL	LLLLLLLLLL	LLLLLLLLLL
B	XXHHHHLLLL	LLLLLLLLLL	LLLLLLLLLL	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH	HHHHHHHHHH
C	XXHHHHLLLL	LLLHHHHHHL	LLLLHHHHHH	LLLLHHHHHH	LLLLHHHHHH	LLLLHHHHHH	LLLLHHHHHH	LLLLHHHHHH	LLLLHHHHHH	LLLLHHHHHH	LLLLHHHHHH	LLLLHHHHHH	LLLLHHHHHH
D	XXHHHHLLLL	HHHLLLHHHL	LLHHHLLLHH	LLHHHLLLHH	LLHHHLLLHH	LLHHHLLLHH	LLHHHLLLHH	LLHHHLLLHH	LLHHHLLLHH	LLHHHLLLHH	LLHHHLLLHH	LLHHHLLLHH	LLHHHLLLHH

4BITCOUNTER

Page : 2

	c	c	c	c	c	cg	cg	c	c	c	c	c	c
AI	H	H	H	H	H	H	H	H	H	H	H	H	H
BI	H	H	H	H	H	H	H	H	H	H	H	H	H
CI	H	H	H	H	H	H	H	H	H	H	H	H	H
DI	H	H	H	H	H	H	H	H	H	H	H	H	H
LOAD	L	L	L	L	L	L	L	L	L	L	L	L	L
CLR	L	L	L	L	L	L	L	L	L	L	L	L	L
UP	H	H	H	H	H	H	H	L	L	L	L	L	L
A	H	H	H	H	H	H	H	H	H	H	H	H	H
B	L	L	L	L	L	L	L	L	L	L	L	L	L
C	H	H	H	L	L	L	L	L	L	L	L	L	L
D	H	H	L	L	L	L	L	L	L	L	L	L	L

4BITCOUNTER

Page : 3

	c	c	c	c	c	c	c	c	c	c	g	cg
AI	H	H	H	H	H	H	H	H	H	H	H	H
BI	H	H	H	H	H	H	H	H	H	H	L	L
CI	H	H	H	H	H	H	H	H	H	H	H	H
DI	H	H	H	H	H	H	H	H	H	H	L	L
LOAD	L	L	L	L	L	L	L	L	L	L	H	H
CLR	L	L	L	L	L	L	L	L	L	L	H	H
UP	L	L	L	L	L	L	L	L	L	L	L	L
A	H	H	H	L	L	L	L	L	L	L	H	H
B	L	L	L	H	H	H	H	L	L	L	L	L
C	L	L	L	H	H	H	H	H	L	L	L	L
D	H	L	L	L	H	H	L	L	L	L	H	H

JEDEC File

PALASM XPLOT, V2.22 - MARKET RELEASE (11-19-86)
 (C) - COPYRIGHT MONOLITHIC MEMORIES INC, 1986

Title : 4Bit_Counter
 Pattern : 4cnt.pds
 Revision : B
 Author : Mehrnaz Hada, Bill Hollo
 Company : Monolithic Memories Inc.
 Date : 1/14/85

```

PAL16RP4
4BITCOUNTER*
GO*F0*
L0512 10111111111011101110101010111111*
L0544 01111111111011011101100110111111*
L0576 10111111110111011111101110111111*
L0608 11111111101111011111100110111111*
L0640 01111111101111111110101110111111*
L0672 11111111101111111011010101111111*
L0704 11110111111111111111101101111111*
L0768 10111111111111101110101010111111*
L0800 01111111111111101101100110111111*
L0832 11111111111111011101101010111111*
L0864 01111111111111011110101110111111*
L0896 10111111111111011111100110111111*
L0928 11111110111111111111101101111111*
L1024 101111111111111111110101010111111*
L1056 011111111111111111110100110111111*
L1088 011111111111111111101101010111111*
L1120 101111111111111111101100110111111*
L1152 11111111111101111111101101111111*
L1280 11111111111111111111010101111111*
L1312 111111111111111110111101101111111*
L2048 00111100*
V0001 CX111101XN0XXHHHXXN*
  
```

```

V0002 CX111111XN0XXLLLLXXN*
V0003 C1111100XN0XXHLLLXXN*
V0004 C1111100XN0XXLHLLXXN*
V0005 C1111100XN0XXHHLLXXN*
V0006 C1111100XN0XXLLHLXXN*
V0007 C1111100XN0XXHLHLXXN*
V0008 C1111100XN0XXLHHLXXN*
V0009 C1111100XN0XXHHHLXXN*
V0010 C1111100XN0XXLLLHXXN*
V0011 C1111100XN0XXHLLHXXN*
V0012 C1111100XN0XXLHLHXXN*
V0013 C1111100XN0XXHLLHXXN*
V0014 C1111100XN0XXLLHHXXN*
V0015 C1111100XN0XXHLHHXXN*
V0016 C1111100XN0XXLHHHXXN*
V0017 C1111100XN0XXHHHHXXN*
V0018 C1111100XN0XXLLLLXXN*
V0019 C0111101XN0XXHHHHXXN*
V0020 C0111100XN0XXLHHHXXN*
V0021 C0111100XN0XXHLHHXXN*
V0022 C0111100XN0XXLLHHXXN*
V0023 C0111100XN0XXHLLHXXN*
V0024 C0111100XN0XXLHLHXXN*
V0025 C0111100XN0XXHLLHXXN*
V0026 C0111100XN0XXLLLHXXN*
V0027 C0111100XN0XXHHHLXXN*
V0028 C0111100XN0XXLHHLXXN*
V0029 C0111100XN0XXHLHLXXN*
V0030 C0111100XN0XXLLHLXXN*
V0031 C0111100XN0XXHLLLXXN*
V0032 C0111100XN0XXLHLLXXN*
V0033 C0111100XN0XXHLLLXXN*
V0034 C0111100XN0XXLLLLXXN*
V0035 C0111100XN0XXHHHHXXN*
V0036 C0101011XN0XXLLLLXXN*
V0037 00101011XN1XXZZZZXXN*
C446C*
9E50

```

State Machine Design

Traffic Controller Example

We have used sections of the state machine traffic controller as examples in Chapter 3B: State Machine Design. You will now see how the entire example is created. (Parts of this example also appear in Monolithic Memories' Programmable Logic Handbook.)

The traffic intersection we are designing for is illustrated in Figure 4-2.

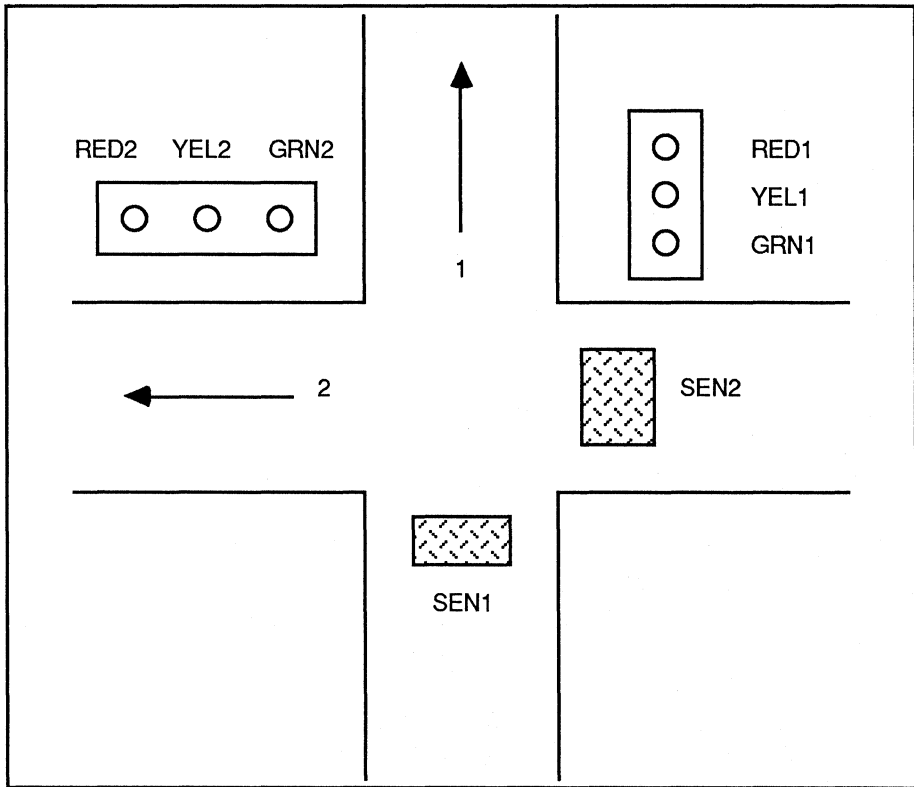
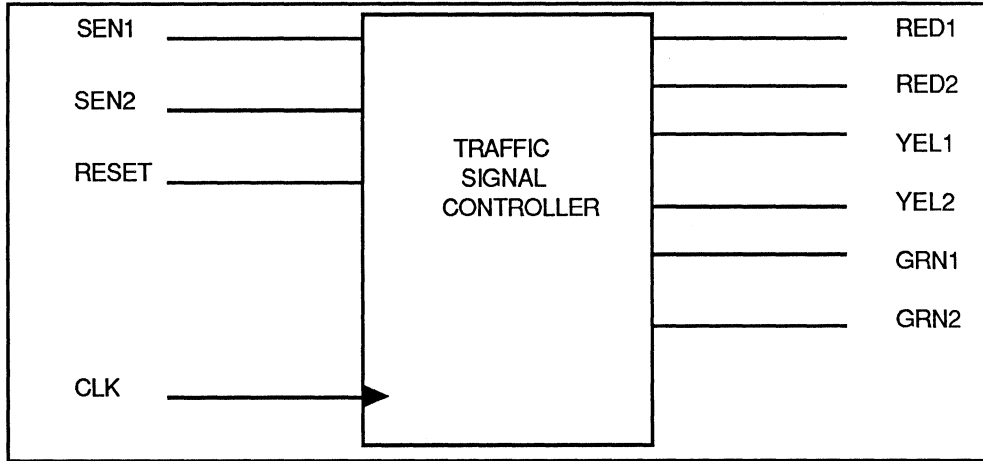


Figure 4-2:
Traffic Intersection

Figure 4-2 shows two one-way streets: direction 1 and direction 2. Each direction has a signal consisting of red, yellow, and green lamps. These lamps are activated with active-high signals: RED1, YEL1, GRN1 for direction one; RED2, YEL2, GRN2 for direction 2. Also, each direction has a sensor that provides an active-high signal indicating the presence of a vehicle. Our controller is to manage this intersection with the sensors as inputs and the lamps as outputs.



4

Figure 4-3:
Traffic Signal Controller Logic Diagram

The traffic signal controller logic diagram in Figure 4-3 includes the system clock (CLK) and an initialize or reset signal (RESET). RESET drives the controller to the initial state you define in your design.

In order to set up our design file, we need to spell out the specifics of the controller. Figure 4-4 illustrates the traffic controller with a state diagram.

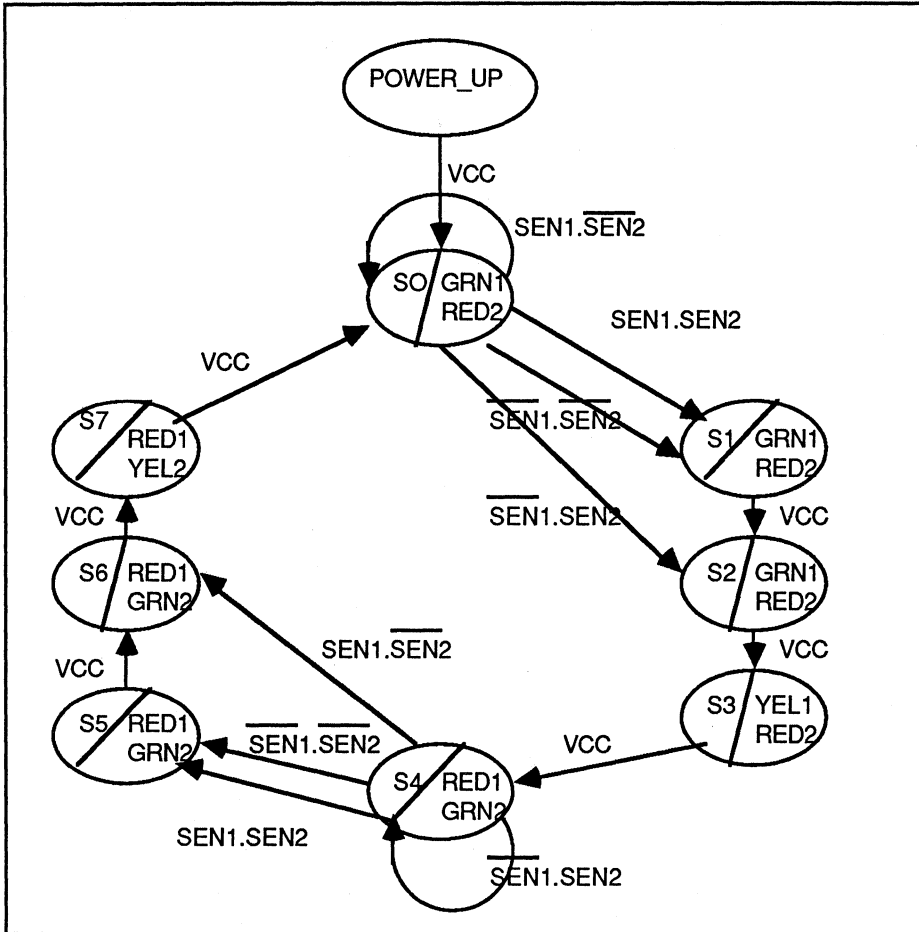


Figure 4-4:
State Diagram of the Traffic Signal Controller

**Traffic Controller Design
Specification**

The complete design file for the state machine traffic controller is given below.

```
TITLE          TRAFFIC CONTROLLER
PATTERN        STATE MACHINE
REVISION       1
AUTHOR         J. ENGINEER
COMPANY        MONOLITHIC MEMORIES
DATE           JANUARY 30, 1987
```

```
CHIP           S_MACHINE  PMS14R21
```

```
;PINS  1      2      3      4      5      6
        CLOCK DCLOCK SEN1  SEN2  I2     I3

;PINS  7      8      9     10     11     12
        I4     I5     I6     I7     SDI    GND

;PINS  13     14     15     16     17     18
        RESET  SDO   RED1   YEL1   GRN1   RED2

;PINS  19     20     21     22     23     24
        YEL2   GRN2   01     00     MODE   VCC
```

```
STATE
MOORE_MACHINE
MASTER_RESET
DEFAULT_OUTPUT /RED1 /YEL1 /GRN1 /RED2 /YEL2
              /GRN2
```



```

POWER_UP := VCC -> S0
S0 := C3 -> S1
      + C0 -> S1
      + C1 -> S2
          +-> S0
S1 := VCC -> S2
S2 := VCC -> S3
S3 := VCC -> S4
S4 := C3 -> S5
      + C0 -> S5
      + C2 -> S6
          + -> S4
S5 := VCC -> S6
S6 := VCC -> S7
S7 := VCC -> S0

```

```

S0.OUTF := GRN1 * RED2
S1.OUTF := GRN1 * RED2
S2.OUTF := GRN1 * RED2
S3.OUTF := YEL1 * RED2
S4.OUTF := RED1 * GRN2
S5.OUTF := RED1 * GRN2
S6.OUTF := RED1 * GRN2
S7.OUTF := RED1 * YEL2

```

CONDITIONS

```

CO = /SEN1 * /SEN2
C1 = /SEN1 * SEN2
C2 = SEN1 * /SEN2
C3 = SEN1 * SEN2

```

SIMULATION

TRACE_ON CLOCK SEN1 SEN2 RED1 YEL1 GRN1 RED2 YEL2
GRN2

SETF RESET /CLOCK

CLOCKF CLOCK ;STATE TRANSITION ONLY ON 1ST CLOCK

CHECK /RED1 /YEL1 GRN1 /YEL2 /GRN2 RED2

SETF /SEN1 /SEN2

CLOCKF CLOCK

CLOCKF CLOCK

CHECK /RED1 /YEL1 GRN1 RED2 /YEL2 /GRN2

CLOCKF CLOCK

CHECK /RED1 YEL1 /GRN1 RED2 /YEL2 /GRN2

CLOCKF CLOCK

CHECK RED1 /YEL1 /GRN1 /RED2 /YEL2 GRN2

CLOCKF CLOCK

CHECK RED1 /YEL1 /GRN1 /RED2 /YEL2 GRN2

CLOCKF CLOCK

CHECK RED1 GRN2

CLOCKF CLOCK

CHECK RED1 YEL2

CLOCKF CLOCK

CHECK /RED1 /YEL1 GRN1 RED2 /YEL2 /GRN2

SETF /SEN1 SEN2

CLOCKF CLOCK

CHECK /RED1 /YEL1 GRN1 RED2 /YEL2 /GRN2

```
CLOCKF CLOCK
CLOCKF CLOCK
SETF SEN1 /SEN2
CLOCKF CLOCK
CLOCKF CLOCK
CHECK YEL2 RED1
CLOCKF CLOCK
CHECK GRN1 RED2
```

```
TRACE_OFF
```

The file TRAFFIC.PDS is the name of the traffic controller design file. After running PALASM 2 software, we get several output files. The following pages show the history file that the simulator produces.

HISTORY FILE

```
PROSIM, V2.22 - MARKET RELEASE (11-19-86)
(C) - COPYRIGHT MONOLITHIC MEMORIES INC, 1986
```

```
Title      :TRAFFIC CONTROL
Pattern    :TRAFFIC.PDS
Revision   :A
Author     :J. Engineer
Company    :MONOLITHIC MEM
Date       :3/31/86  [9/30]
```

```
S_MACHINE
```

Page : 1

	s	c	s	c	c	c	c	c	c	c	s	c	c					
CLOCK	L	H	H	L	L	H	H	L	H	L	L	H	H	L	H	H	L	H
DCLK	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
SEN1	X	X	X	X	L	L	L	L	L	L	L	L	L	L	L	L	L	L
SEN2	X	X	X	X	L	L	L	L	L	L	L	L	L	L	L	L	L	L
I2	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
I7	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
SDI	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
GND	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L
RESET	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
SDO	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
RED1	H	H	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L
YEL1	H	H	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L
GRN1	H	H	H	H	H	H	H	H	H	H	L	L	L	L	L	L	L	L
RED2	H	H	H	H	H	H	H	H	L	L	L	L	L	L	L	L	L	L
YEL2	H	H	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L
GRN2	H	H	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L
O1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
O0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
MODE	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
VCC	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
STATE	P	P	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
	O	O	0	0	0	0	0	1	1	1	2	2	2	3	3	3	3	4
	W	W																
	E	E																
	R	R																
	—																	
	U	U																
	P	P																

4

S_MACHINE
Page : 2

	C	S	C	C	C
CLOCK	L	L	H	H	L
DCLK	X	X	X	X	X
SEN1	L	H	H	H	H
SEN2	H	L	L	L	L
I2	X	X	X	X	X
I3	X	X	X	X	X
I4	X	X	X	X	X
I5	X	X	X	X	X
I6	X	X	X	X	X
I7	X	X	X	X	X
SDI	X	X	X	X	X
GND	L	L	L	L	L
RESET	H	H	H	H	H
SDO	X	X	X	X	X
RED1	H	H	H	H	H
YEL1	L	L	L	L	L
GRN1	L	L	L	L	L
RED2	L	L	L	L	L
YEL2	L	L	L	L	L
GRN2	H	H	H	H	H
O1	X	X	X	X	X
O0	X	X	X	X	X
MODE	X	X	X	X	X
VCC	H	H	H	H	H
STATE	S	S	S	S	S
	4	4	4	6	6
	7	7	7	7	7
	0	0	0	0	0

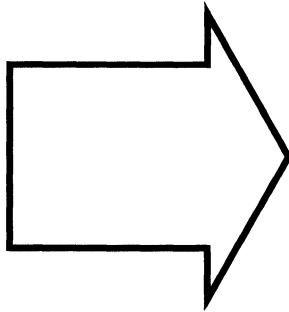


Table of Contents	vii
Software Errata	ERR
Introduction	1
Installing PALASM [®] 2 Software	2
PDS Syntax	3
Boolean Equation Design	3A
State Machine Design	3B
Simulation	3C
Using PALASM 2 Software	4
Installation and Operation Notes	A
Programmer Notes	B
Device Specific Syntax	C
PAL [®] Design File Library	D
Error Messages	E
Submitting a HAL [®] Design to MMI	F
PALASM 2 Syntax Diagram	G
Supplementary Software	H
JEDEC Standard No. 3	I
Release Notes	RN
Index	IX
PLEASM [™] Manual	

APPENDIX A INSTALLATION & OPERATION NOTES

IBM-PC / DOS 2.10 IMPLEMENTATION

Using PALASM 2 Software

Before starting you must have a work disk that has been created using the procedure in Chapter 2.

The example below walks you through the programming of a PAL device from start to finish. It assumes that you are working with a 20-pin PAL device, an IBM-PC and a Data I/O Model 29.

1. Using a text editor, create your PAL device design file, making certain that it follows the conventions stated in Chapter 3, or copy example 8COUNT.PDS from the Examples disk to your work disk. (Contact your local Monolithic Memories sales office if you do not have a copy of the Examples disk.)

If you are using Wordstar as your text editor, a word of caution: Wordstar files created in document mode have control characters which PALASM 2 software may not recognize. A quick and simple test is to TYPE (DOS command) your file on to the screen. If you see strange characters, PALASM 2 software will have trouble understanding the file. Although this is not a foolproof way of catching all control characters, it un.masks most of them.

A

EDLIN, which is available on your DOS disk, is a simple and effective line editor that can be used to create straight ASCII files free from control characters.

2. To start, insert the disk containing the program PALASM.EXE into drive B and the disk with your WORK diskette into drive A.

Reboot the machine. Now type:

```
PALASM2 <CR>
```

In response to the first prompt, type:

```
A: <filename>.PDS
```

This indicates that the input file, which is a design file describing an 8-bit counter, is located on the disk in drive A. Answer the four Y/N questions as explained in Chapter 4 of this manual.

3. Insert the disk containing the XPLOT.EXE program in drive B. and enter

```
A:>XPLOT
```

4. Insert the disk containing PC2 into drive B and enter

```
A:>PC2
```

5. Run PC2 (PALSETUP mode) to make certain the communications parameters specified are correct. For the Data I/O programmer, the recommended parameters are 4800 baud rate, 7 data bits, 1 stopbit and even parity. In any event, due to limitations of the PC, do not exceed 4800 baud.

6. Return to the communications part of PC2. Before pressing F1, check to see if proper communication has been set up with the programmer. While downloading, the most common problem is establishing that the Data I/O and PC are actually communicating. To test this quickly, on the Data I/O, press COPY from RAM to PORT START. If communication has indeed been established, the contents of RAM on the Data I/O will now be dumped on the screen. You can now be sure that the proper handshaking is taking place.
7. Next, manually enter the family pin code listed in Appendix B of this manual. Using the DATA I/O keyboard, press <COPY> from <DEVICE> to <RAM> on the Data I/O and, when the programmer asks for the family pin code on the window, key in the 4 digit code for the device you are using. You will find the proper codes in the Data I/O manual.
8. Press <SELECT> <E> <START> on the Data I/O keyboard. This tells the programmer to expect a JEDEC input file. Push <F1> on the IBM-PC to begin transmission. (Provide a filename if asked.)



The clock face on the window of the Data I/O should begin to turn, indicating data transfer. After transmission has been completed, verify that the checksum on the Data I/O is all right (no error status indicated.)

9. If you do not wish to use PC2, a quick and dirty way to set up the communications parameters on the PC is to type the following with the DOS disk in drive A:

```
MODE COM1:4800,E,7,1,P
```

This has the same effect as running PC2 in setup mode. Next, assuming the file you wish to download is <filename>.JED, instead of running PC2 in communications mode, type

```
COPY <filename>.JED COM1:
```

Restrictions

1. The XPLOT program needs access to PAL device Definition Files (PDF) in order to work with the specific architectural dependencies of individual part types. The program first searches the default directory to locate them and then on drive B. This will work if you place your editor, design files and scratch files on A (the default drive) and the executable and PDF files on drive B (two drive system). The PATH command should be used to allow finding the executable files on drive B as in

```
A> PATH A:\;B\;
```

If you have an IBM-XT, place all files in a single directory on the fixed disk and substitute drive C for B in the PATH command. That directory should be your default location. If the system cannot find the PDF files, it will prompt you to insert a disk into drive B.

2. Do not terminate the program abnormally by pressing CTRL-C, BREAK, etc. when you are sending the output to a file. If the session is terminated using CTRL-C, it may result in lost files on your disk, since your output file will not have been properly closed.

Suppose your output file on drive A is called OUTPUT.PRN. If you abort your program using CTRL-C, when you look at the directory, it will indicate that your output file contains zero blocks:

```
A> DIR <CR>  
OUTPUT.PRN 0 09-15-83 1:47p
```

Your disk, however, will contain lost files. To recover any lost files or clusters, run CHKDSK with the /F switch using the IBM 2.0 boot disk.

Type:

```
A> CHKDSK B:/F
1 lost clusters in 1 chain
Convert lost chains to files
(Y/N)? Y
362496 bytes total disk space
0 bytes in 1 hidden files
216064 bytes in 2 user files
1024 bytes in 1 recovered files
145408 bytes available on disk
```

MS-DOS names the recovered file, FILE000.CHK. You can then delete OUTPUT.PRN and rename FILE000.CHK as you please.

3. The disk in drive A: should have a COMMAND.COM on it and be formatted as a system disk. Otherwise, DOS will prompt you to insert a COMMAND.COM disk after each program executes.
4. All the software must be from the same release. Running different software versions on the same data files will cause the programs to generate a PDF failure message.

**A**

PALASM 2 Software Files

PALASM 2 software is distributed on 5 1/4 inch floppy disks. The actual allocation of files and file count on a disk may vary with the specific release of the program due to space considerations. In general expect files with the suffix .PDS to be located on the Examples Disk and files with the suffix .PDF on the Executable Disks.

VAX-VMS IMPLEMENTATION

Installation of Software

1. A directory or sub-directory should be created to contain the PALASM 2 software, if one does not already exist. For ease of identification name it [PALASM]; all future references in this document will use [PALASM] as the directory name.

```
$ CREATE/DIR [PALASM]
```

2. Move to directory [PALASM]

```
$ SET DEFAULT [PALASM]
```

3. Create a command procedure START.COM with the following steps:

```
$ IN_DEVICE := "MTA0:"
$ ALLOCATE 'IN_DEVICE' IN_VOLUME
$ MOUNT/DENSITY=1600/OVERRIDE=
  IDENTIFICATION'IN_DEVICE'
$ COPY/LOG 'IN_DEVICE'INSTALL.COM *
$ DISMOUNT/NOUNLOAD IN_VOLUME
$ DEALLOCATE 'IN_DEVICE'
$ EXIT
```

where MTA0: is the name of the magnetic tape drive.

4. Remove the ring from the magnetic tape, load it on the required drive, and set it on-line.
5. Execute the START command procedure:

```
$ @START
```

The command procedure will now read the file `INSTALL.COM` from the magnetic tape in the `[PALASM]` directory.

6. To install all the PALASM 2 software, execute the command procedure `INSTALL.COM`. Type

```
$ @INSTALL
```

The following message appears

```
INPUT (SUB)DIRECTORY NAME TO CONTAIN  
PALASM 2 SOFTWARE? :
```

Choose a name for the directory that will contain PALASM 2 software (e.g., `PALASM`) and type it at the prompt.

Next, assign the logical directory name `PAL2$DAT` to the directory you chose for PALASM 2 software. Type

```
$ASSIGN <directory containing PALASM 2 software>  
PAL2$DAT<CR>
```

This command procedure will read all the necessary files from the magnetic tape and from the PALASM 2 software executables.

7. To make the PALASM 2 software generally available to users, either all users can separately execute the PALASM 2 software set-up command procedure in the user login file, or the VAX system manager can execute this in the VAX system login file.

Be prepared to answer the following two questions:

- A. Directory input name (e.g., `[PALASM]`)
- B. Magnetic tape drive name

A

To execute the PALASM 2 software set-up procedure input the command

```
$ @DRA:[PALASM]PAL2ASS DRA:[PALASM]
```

where DRA: is the disk drive name.

8. Now input the help request to display what has been set up:

```
$ PAL2HLP
```

Notes on Software Support Procedures

The following procedures may be performed on the VAX-VMS system only.

1. The command procedure file PAL2ASS.COM contains various assignments which the software will use to find run-time files and to give you easy access to PALASM 2 software, command files, test files and documentation. If desired you can place the different elements of the PALASM 2 software suite in multiple directories, but the assignments in the set-up file must be changed to reflect the different directory structure. A single directory structure has been used here to simplify description of operations.
2. A set of PALASM 2 software example test files has been included in your package. A list of the examples plus a description can be obtained with the input command

```
$ PAL2LIS
```

To run any of the examples, a command procedure is included to execute the syntax checker and to produce fuse maps. To execute the command procedure, input the command

```
$ PAL2EX <example name>
```

3. A simplified batch procedure that runs various components of the PALASM 2 software can be executed by using the input command

```
$ PAL2
```

4. A command has been included to display a list of all the assignments and commands that are set up on execution of the PALASM 2 software set-up procedure. To see this information input the command

```
$ PAL2HLP
```

5. To link each of the programs in the PALASM 2 software suite you will use a set of commands, one for each program, which are also shown on input of the help command (PAL2HLP).
6. For users who have source files, these files have the extensions *<program name>.VMS*. To compile and link these source programs a set of command procedures exist, one for each program:

```
BLDCVT.COM.... compile & link PDSCNVT  
BLDFEP.COM.... compile & link PALASM2  
BLDXPT.COM.... compile & link XPLOT  
BLDSIM.COM.... compile & link SIM  
BLDZHL.COM.... compile & link ZHAL  
BLDJED.COM.... compile & link JEDMAN  
BLDPRA.COM.... compile & link PROASM  
BLDPRS.COM.... compile & link PROSIM
```

These procedures all execute the program VXPASCAL.COM when compiling. Edit this program if you want to alter compile times

A complete list of all the files on the VAX-VMS tapes can be



found in the Release Notes at the end of this manual.

ASCII TAPE INSTALLATION

Tape Description

Your ASCII magnetic tape, which contains all the files required by PALASM 2 software, has been formatted in the following manner:

- * ASCII characters
- * no tape label
- * tape density = 1600
- * record size = 132 bytes
- * block size = 6600 bytes (50 records)

The first file on the tape contains a list of file names. (See Release Notes.) In order to process this tape, you must provide a utility to read the first file off the tape, extract the file names and then copy each file from the tape renaming each with the correct file name. The first file on the tape corresponds to the first file name on the file name list file. Please note that the source code on the ASCII tape is in a non-compilable form and needs modification for the computer environment you are using.

Reading In an ASCII Tape On a VAX-VMS System

The following command file can be used to read in a PALASM 2 software ASCII tape on a VMS system:

```
$ ON CONTROL_Y THEN GOTO DONE
```

```

$ ON SEVERE_ERROR THEN GOTO DONE
$ IN_DEVICE := "MTA0:"
$ ALLOCATE 'IN_DEVICE' IN_VOLUME
$ MOUNT/FOREIGN/RECORDSIZE=132/
  BLOCKSIZE=6600 'IN_DEVICE'
$ COPY/LOG 'IN_DEVICE' FILE.LST
$ OPEN/READ/ERR=DONE IPF FILE.LST
$ READ/END=DONE IPF FNAM
$ LP1:
$ READ/END=DONE IPF FNAM
$ LG0='F$LOCATE(" ", FNAM) '
$ FILNAM := 'F$EXTRACT(0, LG0, FNAM) '
$ COPY/LOG 'IN_DEVICE' 'FILNAM'
$ GOTO LP1
$ DONE
$ CLOSE IPF
$ DISMOUNT/NOUNLOAD IN_VOLUME
$ DEALLOCATE 'IN_DEVICE'
$ EXIT

```



File Description

Following is a description of the files on this tape.

Extension/Name	Description
*.SRC	Source code for PALASM 2 software
*.INC	The global-include file data for PALASM 2 software
*.PDF	PALASM 2 software run-time data files
INSTALL.COM	VAX-VMS command procedure to install a PALASM 2 software VAX-VMS tape on a VAX computer
README.ASC	Installation documentation
PAL2HLP.FAC	Help facility information
PAL2ASS.COM	System logical assignments needed to run PALASM 2 software on VAX-VMS
PAL2.COM	Procedure to execute PALASM 2

BLD*.COM	software on the VAX-VMS VAX-VMS procedures to compile each source module
LNK*.COM	VAX-VMS procedures to link each program
VXPASCAL.COM	VAX-VMS Pascal compile module procedure
*.PDS	PALASM 2 software design examples
PAL2EX.LIS	List describing all the design examples
PAL2EX.COM	VAX-VMS command procedure to execute the design examples

VAX-UNIX INSTALLATION

What You Require

- * PALASM 2 software on a VAX-UNIX magnetic tape
- * Berkeley 4.2 UNIX operating system
- * Berkeley Pascal compiler (or any compatible compiler)

Software Installation

To install PALASM 2 software, follow these steps

1. Create a directory or sub-directory for PALASM 2 software. Name it PALASM so that it is easily identifiable. To accomplish this, enter

```
$ mkdir palasm
```

2. To move to the PALASM directory, type

```
$ cd palasm
```

3. Remove the ring from the magnetic tape, load it on the appropriate drive, and set it online.
4. To read the PALASM 2 software files from the magnetic tape to the PALASM directory you have just created, type the command

```
$ tar xu
```

5. To complete the installation procedure, you must execute the command procedure INSTALL.COM. Type

```
$ install.com
```

6. To make the PALASM 2 software available to general users, each user must set a \$PATH variable that searches for commands in the PALASM directory.

A

Or

The UNIX system manager must place the commands in a system directory.

Your PALASM 2 software is now ready for use on the UNIX operating system. Below are some additional pointers.

Command Procedures

- * To access the PALASM 2 example design files (PDS files) along with a brief description of each, enter the command

```
$ cat pal2lis.doc
```

To successfully run the example PDS files, you must execute a command procedure. This procedure enables you to run the syntax checker and to produce fusemaps. Enter

```
$ pal2ex.com <filename>
```

- * You can run various parts of PALASM 2 software in batch mode by executing the command

```
$ pal2.com
```

- * For a listing of all the commands that were set up on execution of the PALASM 2 installation procedure, type

```
$ cat pal2hlp.fac
```

- * To create each of the programs in the PALASM 2 software suite, type

```
makeall
```

Note: Remember to type commands in lowercase. Filenames are read only if they are in the following format: <filename.EXT>. Notice that the filename is in lowercase, but the extension is in uppercase. The JEDMAN output files, however, are exceptions. Both filename and extension must be in uppercase. For example: <JEDMAN.EXT>.

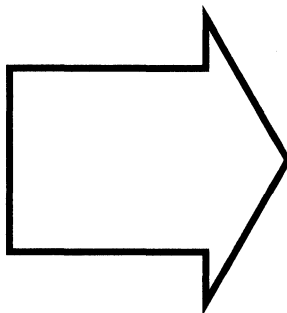


Table of Contents	vii
Software Errata	ERR
Introduction	1
Installing PALASM [®] 2 Software	2
PDS Syntax	3
Boolean Equation Design	3A
State Machine Design	3B
Simulation	3C
Using PALASM 2 Software	4
Installation and Operation Notes	A
Programmer Notes	B
Device Specific Syntax	C
PAL [®] Design File Library	D
Error Messages	E
Submitting a HAL [®] Design to MMI	F
PALASM 2 Syntax Diagram	G
Supplementary Software	H
JEDEC Standard No. 3	I
Release Notes	RN
Index	IX
PLEASM [™] Manual	



APPENDIX B PROGRAMMER NOTES

DATA I/O

Helpful Hints

DATA I/O's EPROM revision changes Monolithic Memories manufacturer code from "95" to "22" in the JEDEC *family code*. Please check that your P/T adapter 303a-002 is rev 06 or later.

Firmware updates accept a checksum of 0000 as valid and space before the L field data. Both of these were flagged as errors by old Data I/O firmware: Please update your Logic Pak to rev 04 or later. You will also notice higher programming yields from this update.

B

Downloading

These notes are to be used in conjunction with the Data I/O model 29 A/B programmer and PALASM 2 software.

1. Create a JEDEC file using PALASM 2 software. The JEDEC file produced by the XPLOT backend module has .JED as the extension to its filename.
2. Using a screwdriver, set the programmer to the desired baud rate. Twist the circular notch located at the bottom left

of the back of the machine until the white arrow points to the letter C. This is the recommended baud rate (4800 baud). Codes for the other baud rates are listed in Table B-1.

Table B-1:
Recommended Baud Rate

Code	Baud rate
5	300
7	1200
A	2400
C	4800

3. Power on the Data I/O. When it finishes its self-test, press <COPY> from <DEVICE> to <RAM> <START>. The programmer will then ask for the device family pin code. The codes for Monolithic Memories devices are listed in Table B-2.

Table B-2:
Device Family Pin Codes for DATA I/O Programmer

Device	Family Pin Code
PAL10H8/H8-2	2218
PAL10L8/L8-2	2213
PAL10H20G8	2242
PAL10H20P8	2242
PAL12H6/H6-2	2219
PAL12L10	2201
PAL12L6/L6-2	2214
PAL14H4/H4-2	2220
PAL14L4/L4-2	2215
PAL14L8	2202
PAL16C1/C1-2	2221
PAL16H2/H2-2	2222
PAL16L2/L2A	2216
PAL16L6	2203
PAL16L8/L8-2/L8-4/L8A	2217
PAL16L8B	3017
PAL16P8/16RA8	2230
PAL16R4/R4-2/R4-4/R4A	2224
PAL16R6/R6-2/R6-4/R6A	2224
PAL16R8/R8-2/R8-4/R8A	2224
PAL16RP4	2231
PAL16RP6	2231
PAL16RP8	2231
PAL18L4	2204
PAL20C1	2212
PAL20L10	2206

B

Table B-2 (Continued):
Device Family Pin Codes for DATA I/O Programmer

Device	Family Pin Code
PAL20L2	2205
PAL20L8	2226
PAL20S10	2243
PAL20R4	2227
PAL20R6	2227
PAL20R8	2227
PAL20RA10	2245
PAL20RS10	2244
PAL20RS4	2246
PAL20RS8	2244
PAL20X10	2223
PAL20X4	2223
PAL20X8	2223
PAL22RX8	2278
PAL22V10	4628
PAL32R16	2247
PAL32VX10	2277
PAL 6L16	2248
PAL64R32	2284
PAL8L14	2249
PMS14R21	2258

4. Next, run PC2 (PALSETUP mode) on your PC. Use the following RS-232 parameters: 4800 baud, 7 data bits, 1 stop bit and even parity.

These parameters are stored in the PC2.DAT file. If you do not wish to run PALSETUP, a quick and dirty way to modify the parameters is to edit the PC2.DAT file directly.

5. Run PC2 next. You can specify the filename by using function key F9. When it prompts you for the name of your input file, respond with the <filename>.JED generated by the PALASM2 program.
6. After specifying the filename, PC2 blanks the screen and indicates its command options: F1 (for downloading), F2 (PALSETUP mode), F3 (VIEW toggle), F4 (CAPT mode), F9 (filename) & F10 (return to DOS). Do NOT begin downloading yet.
7. To verify communications press <SELECT> <E> <1> <START>. This will produce the Data I/O operations menu on your screen. (You will have to enter the FAMILY/PIN code from the IBM-PC keyboard if it has not been entered on the programmer keyboard.)

If you are not successful in getting the Data I/O to talk to the PC, repeat the steps above. If problems still exist, confirm the cabling between the RS232 ports.

8. Next, press <SELECT> <E> <START> on the Data I/O. The window of the programmer will display a clock face. Press F1 (IBM-PC). Downloading will begin and the clock face should start to turn.
9. When downloading is completed, press F10 to exit. The Data I/O should indicate a 4-digit fuse checksum on its display if the process has been successful. You can now program the part.

B

Using DATA I/O on VAX-VMS

Figure B-1 illustrates the cable connections that must be made between a VAX-VMS system and a Data I/O programmer, plus an example command procedure that will enable you to program on VAX-VMS systems.

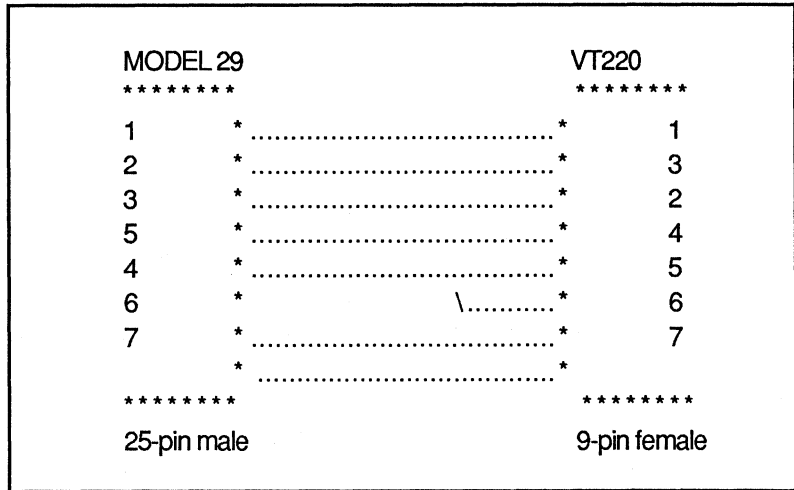


Figure B-1:
Data I/O <-> VAX-VMS Cable Connection

To download from the VAX, do the following:

A. Model 29

1. Copy DEV to RAM with appropriate family and pinout code.
2. Select EB and start.

**VT220: VAX-VMS Program
To Make DATA I/O
Connection**

Monolithic Memories recommends the use of the following program, and would like to acknowledge John Carvalho of RCA who created it.

1. Type @PALCOPY PALNAME.JED where PALCOPY is the name of this command procedure file.

```
$ON CONTROL_Y THEN EXIT
$ON ERROR THEN EXIT
$IPFNAM := 'P1'
$IF P1 .NES. "" THEN GOTO FNAME1
$INQUIRE P1 "PLEASE ENTER THE NAME
$OF THE FILE TO BE TRANSMITTED"
$IF P1 .EQS. "" THEN GOTO PALXIT
$FNAME1:
$IPFNAM := 'P1'
$LN = 'F$LEN(IPFNAM) '
$DOT = 'F$LOCATE(".",IPFNAM) '
$IF DOT.EQ.LN THEN
$IPFNAM:="' 'IPFNAM'.JED"
$XMTFIL:
```

2. Set up the VT220 for write through printer mode.

```
$WRITE SYS$OUTPUT "TRANSMITTING
$FILE" 'IPFNAM' ""
$WRITE SYS$OUTPUT "[<ESC> 5I"
$TYPE 'IPFNAM'
$WRITE SYS$OUTPUT "<ESC> [4I"
```

B

3. Drop the VT220 out of write-through mode.

```
$WRITE SYS$OUTPUT "DOWNLOAD
$COMPLETE"
$OPEN/READ INDATA 'IPFNAM'
$LINE1 = ""
$LINE2 = ""
$LINE3 = ""
$RDLOOP:
$READ/END_OF_FILE=DONE INDATA LINE3
$LINE1 = LINE2
$LINE2 = LINE3
$GOTO RDLOOP
$DONE:
$CLOSE INDATA
$WRITE SYS$OUTPUT "THE CHECKSUM IS:"
      `LINE1' ----`LINE2'"
$PALXIT:
```

VARIX OMNI

Helpful Hints

For designs involving the PAL64R32 MegaPAL device, an additional file with extension .VRX is generated. It contains the specific fuse plot to be downloaded to the VARIX programmer. A full (not brief) fuse plot must be specified for VARIX downloading when using MegaPAL devices.

Downloading

To begin, connect the Varix programmer to your PC. You must have the parallel port card supplied by Varix installed in the PC. Once the card is installed, connect the Varix to the PC with the cable supplied by Varix (observe the proper connector labels: the cable is not symmetric). *You should also have revision 3.16a or later OMNI software for programming the PAL64R32, and revision 3.18e or later OMNI software for programming the PAL32R16 and all 20-pin and 24-pin PAL devices.*

B

1. Turn on power to the programmer. Plug the PAL32R16 or PAL64R32 adapter into the largest socket on the programmer. No adapter is required for programming 20-pin or 24-pin PAL devices.
2. Insert the Omni software diskette supplied by Varix in drive A. Insert the diskette with the PALASM 2 fuse plot in drive B. But if you are using OMNI software revision 3.16a. Insert the diskette with JEDEC files in drive B if you are using OMNI software revision 3.18e.

Note: The file extensions of JEDEC files must be .JED; otherwise OMNI software will refuse to accept them, or will treat the files as if they are fuse plots. You will need to rename any .JDC files you wish to use.

3. Type OMNI <CR> . The following message is displayed:

```
Omni-Programmer for the IBM PC  
Version x.yy Copyright (c)1984  
Varix Corporation
```

```
Please select target device (or type  
? for list):
```

In response to this message you can either enter ? to get a list of the part codes, or a specific code such as 64R32.

For OMNI software revision 3.16a: The help menu is displayed after the part code is entered.

For OMNI software revision 3.18e: The command prompt * is displayed. Type H or M to get either the brief or the detailed help menu respectively. Type H PROGRAM to get the help menu for programming commands.

Note: The Varix software displays * as a command prompt. Thus when a command has finished executing, the * is displayed.

4. Plug a device into the proper adapter socket.
5. Verify that the part is blank.

For OMNI software revision 3.16a: Type V B

For OMNI software revision 3.18e: Type F X 0 130 to fill the memory buffer with X; then type V 0 130 to verify that the part on the socket is blank.

6. Load the buffer storage in the Varix programmer with the fuse data stored in the file that was created in PALASM 2.

For OMNI software 3.16a: Enter L <filename>.VRX <CR>.

For OMNI software 3.18e: Enter L <filename>.JED <CR>.

Note: You should substitute the name of the file containing XPLOT in drive B: for filename in the command above. You don't have to specify drive B in the command: it is already assumed by the software.

7. Display the contents of the Varix buffer starting at the "first parameter" product term, for "second parameter" product terms following. The example is set up to display the entire contents of a buffer containing a 32R16 or 64R32 XPLOT. Enter D 0 128 for 32R16, or D O 256 for 64R32. When this command is executed the screen will display the product terms contained within the buffer.

For OMNI software 3.16a: Enter D 0 256 <CR>.

For OMNI software 3.18e: Enter D 0 128 <CR>.

8. Program the PAL device. The reasoning for the parameters is the same as in step 7; thus, to program an entire 32R16, enter D 0 128; enter D 0 256 to program a 64R32.

For OMNI software 3.16a: Enter P 0 256 <CR>.

For OMNI software 3.18e: Enter P 0 128 <CR>.

9. Program the flush (bypass) fuses of the PAL device.

For OMNI software 3.16a: Enter P F <CR>.

For OMNI software 3.18e: Enter P F <CR>.

10. Program the polarity fuses of the PAL device.

For OMNI software 3.16a: Enter P O <CR>.

For OMNI software 3.18e: Enter P O <CR>.

11. Verify that the PAL device has been programmed properly.

B

The parameters follow the same reasoning as in the D and P commands.

For OMNI software 3.16a: Enter V 0 256 <CR>.

For OMNI software 3.18e: Enter V 0 128 <CR>.

While this command is executing you will see the product term numbers displayed as each is verified as correct. If an inconsistency occurs between what is in buffer storage and what is in the device, the buffer and device product terms will be displayed. If no inconsistencies are discovered, you can assume that the part has been programmed properly.

12. To upload the contents of the device in the socket to the buffer RAM of the programmer:

For OMNI software 3.16a: Enter R 0 256 <CR>

For OMNI software 3.18e: Enter R 0 128 <CR>.

13. To observe the contents:

For OMNI software 3.16a: Enter D 0 256 <CR>.

For OMNI software 3.18e: Enter D 0 128 <CR>

14. To exit the OMNI software:

For OMNI software 3.16a: Enter X <CR>.

For OMNI software 3.18e: Enter QUIT <CR>.

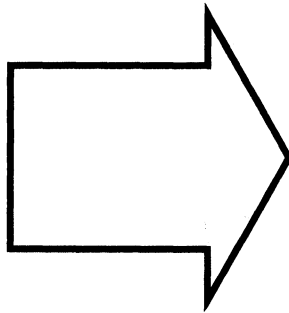


Table of Contents	vii
Software Errata	ERR
Introduction	1
Installing PALASM [®] 2 Software	2
PDS Syntax	3
Boolean Equation Design	3A
State Machine Design	3B
Simulation	3C
Using PALASM 2 Software	4
Installation and Operation Notes	A
Programmer Notes	B
Device Specific Syntax	C
PAL [®] Design File Library	D
Error Messages	E
Submitting a HAL [®] Design to MMI	F
PALASM 2 Syntax Diagram	G
Supplementary Software	H
JEDEC Standard No. 3	I
Release Notes	RN
Index	IX
PLEASM [™] Manual	

APPENDIX C DEVICE SPECIFIC SYNTAX

The unique architectures of the PAL22RX8, the PAL22V10, and the PAL32VX10 devices require special syntax considerations. This appendix provides special instructions for designing with the three devices. Note that only Boolean equation design instructions are included. PALASM 2.22 is being tested for state machine design entry for PAL devices.

If your regular package of PALASM 2.22 software does not contain a design examples disk, contact your local Monolithic Memories sales office for a disk of design examples.

The EXAMPLES.DAT directory contains two subdirectories of design examples:

- * State
- * Boolean

Refer to Appendix D for a list of design examples and descriptions of the designs.

C

PAL22RX8 AND PAL22V10: SPECIAL INSTRUCTIONS

The following special syntax considerations are the same for the PAL22RX8 and PAL22V10.

Special Syntax for PAL22RX8 and PAL22V10

The pin list for a PAL22RX8 includes the pin names you would expect for this 24-pin PAL device, followed by a name used to specify the global .SETF and .RSTF functions of the registers.

Example:

```
CHIP INPUT_OUTPUT PAL22RX8
CLK I2 I3  $\bar{I}4$  I5 I6 I7 I8 I9 I10 I11 GND
I13 I14 O15 O16 O17 O18 O19 O20 O21 O22
I23 VCC
GLOBAL
```

On PAL22RX8, there is only one Exclusive-Or gate per output. Each output equation can therefore contain at most one Exclusive-Or function (of two terms). If you use the Exclusive-Or in your equation, then you must not also use it as a polarity inverter. Thus, if you define an output with an Exclusive-Or, the polarity of the output must be the opposite of that given in the pin list. (e.g. /O15 := ... if O15 is the name used in the pin list.)

On both the PAL22RX8 and the PAL22V10 devices, you can use the name given in the pin list and .SETF or .RSTF to specify the global SET and RESET functions.

GLOBAL.SETF = product_term Specifies the global
SET function

GLOBAL.RSTF = product_term Specifies the global
RESET function

Note: You are allowed to use only one product term for these
functions

PAL32VX10: SPECIAL INSTRUCTIONS

The PAL32VX10, also a 24-pin device, has associated with it a number of architectural features unique in the PAL device family. Each output has a programmable flip-flop, which can be configured as a J-K, S-R, T or D type. Each register can be buried so that its contents cannot be observed directly on the output pin. Each output has an internal Exclusive-OR gate which can either be used as such or as a polarity inverter, depending on the application. The XOR gate also allows the user to create the various flip-flop types. In addition, you can set outputs as Registered or Combinatorial (and set the register type) dynamically, specifying the option you want through a product term.

Along with these special features of the PAL32VX10 architecture, there are some special rules which need to be observed while using PALASM 2 software on this device.

Special Syntax for PAL32VX10

The pin list for a PAL32VX10 includes the expected pin names followed by a name used to specify the global .SETF and .RSTF functions of the buried registers. This is followed by ten names used to specify the buried registered nodes located at the /Q

C

outputs of the buried registers at pins 14 through 23 respectively.

Example:

```
CHIP INPUT_OUTPUT PAL32VX10
```

```
CLK I2 I3 I4 I5 I6 I7 I8 I9 I10 I11 GND  
I13 O14 O15 O16 O17 O18 O19 O20 O21 O22  
O23 VCC  
GLOBAL R14 R15 R16 R17 R18 R19 R20 R21  
R22 R23
```

(The relationship between the buried registered node names and output names in the pin list is such that R14 corresponds to O14, etc.)

Output Equations- Buried Registered Node

When the output you want is a registered function of product terms, you must define it this way at the buried registered node. If you want the output to be used only for feedback, you do not need to also define it at the output node. However, if you want the output to be visible at the output pin, you must also define it at the output node by specifying either $Onn := Rnn$ or $Onn := /Rnn$, depending on your polarity preference.

Note: that the output node must be defined as a function of the buried registered node when one has been defined.

Output Equation- Output Node

When the output you want is a combinatorial function of product terms, you must define it this way at the output node. In this case, the buried registered node must not be defined.

Because there is only one AND/OR array for each output, you can define either the buried registered node or the output node as a sum of products, but not both. Follow the rules above for defining registered or combinatorial outputs, and you should meet this requirement.

The PAL32VX10 has only one Exclusive-Or gate per output. Each output equation can therefore contain at most one Exclusive-Or function (of two terms). If you use the Exclusive-Or in your equation, then you must not also use it as a polarity inverter. Thus, if you define a buried registered node or output node with an Exclusive-Or, the polarity of the node must be the opposite of that given in the pin list. (e.g. 'R14 := ...' if 'R14' is the name used in the pin list.)

You may use the name given in the pin list and .SETF or .RSTF to specify the global SET and RESET functions.



GLOBAL.SETF = product_term Specifies the global SET function.

GLOBAL.RSTF = product_term Specifies the global RESET function.

Note: You are allowed to use only one product term for these functions.

.CMBF Function

After defining an output node either as a function of a buried registered node or as a function of product terms, you can use the .CMBF function to override the definition selected using the :=, = convention for registered or combinatorial output. This provides for dynamic selection of registered or combinatorial output.

O_{nn}.CMBF = VCC

specifies a fixed combinatorial output.

O_{nn}.CMBF = GND

specifies a fixed registered output.

O_{nn}.CMBF = product_term

specifies a dynamically selected registered or combinatorial output, depending on whether the result is high or low.

Example:

```
O14 := <product_term>
O14.CMBF = VCC
```

This is a registered output, but is defined in terms of a combinatorial equation. You need not use the .CMBF if you do not want dynamic selection.

Note: You are allowed to use only one product term for these functions.

Note: The Examples disk contains a complete PAL32VX10 design specification example.

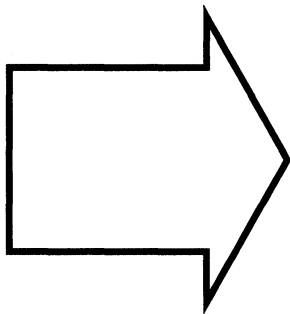


Table of Contents	vii
Software Errata	ERR
Introduction	1
Installing PALASM [®] 2 Software	2
PDS Syntax	3
Boolean Equation Design	3A
State Machine Design	3B
Simulation	3C
Using PALASM 2 Software	4
Installation and Operation Notes	A
Programmer Notes	B
Device Specific Syntax	C
PAL [®] Design File Library	D
Error Messages	E
Submitting a HAL [®] Design to MMI	F
PALASM 2 Syntax Diagram	G
Supplementary Software	H
JEDEC Standard No. 3	I
Release Notes	RN
Index	IX
PLEASM [™] Manual	

APPENDIX D PAL DESIGN FILE LIBRARY

(Files Located on Design Examples Disk)

If your regular package of PALASM 2.22 software does not contain a design examples disk, contact your local Monolithic Memories sales office for a disk of design examples.

The EXAMPLES.DAT directory contains two subdirectories of design examples:

- * State
- * Boolean

Table D-1 lists all examples in the order they appear in the subdirectories.

D

Descriptions of most of the examples follow the table.

Table D-1:
Files Located On Design Examples Disk

File Name	Description	Device
32VX10.PDS	Example of PAL32VX10 Design Specification File	PAL32VX10
<i>State Subdirectory</i>		
STRAFFIC.PDS	Traffic Signal Controller	PMS14R21
PRODCDER.PDS	QIC-02 Command Decoder	PMS14R21
QAM3.PDS	ENCODER.	PAL20R8
4WAYTRAF.PDS	4-Way Traffic Light Controller	PMS14R21
<i>Boolean Subdirectory</i>		
9BITCNT.PDS	9-Bit Counter	PAL20X10
8COUNT.PDS	8-Bit Counter	PAL20X8
DCOUNT.PDS	5-Bit Down Counter	PAL20RA10
CONTROL.PDS	DEC PDP-11 Unibus Interrupt Controller	PAL20RA10
OCTCOMP.PDS	Octal Comparator	PAL16C1
3TO8DMUX.PDS	3-8 Demultiplexer	PAL16R8
CRT.PDS	CRT Controller Logic	PAL20RA10
PORT.PDS	7-Bit I/O Port with Handshake Logic	PAL20RA10
BARREL.PDS	Barrel Shifter	PAL64R32
4CNT.PDS	4-Bit Counter	PAL16RP4
4-16DEC.PDS	4 - 16 Decoder	PAL6L16
10COUNT.PDS	10-Bit Counter	PAL20RS10
ADREG16.PDS	16-Bit Addressable Register	PAL32R16
MEMIO.PDS	PC I/O Mapper	PAL8L14
UPCOUNT.PDS	5-Bit Up Counter	PAL20RA10
FLIPFLOP.PDS	Basic Flip Flops	PAL16RP8

Table D-1 (Continued):
Files Located On Design Examples Disk

File Name	Description	Device
MEMORY.PDS	Memory Handshake Logic	PAL16RP6
ARBITER.PDS	3-Bit Arbiter	PAL20RA10
LINK.PDS	Serial Data Link Controller	PAL20RA10
LATCH.PDS	Octal Latch	PAL10H20P8
9BITREG.PDS	9-Bit Register	PAL20X10
10BITREG.PDS	10-Bit Register	PAL20X10
BTRAFFIC.PDS	Traffic Signal Controller	PAL16RP8
DCODER.PDS	QIC-02 Command Decoder	PAL20L8
K7ENC1.PDS	1/2 Rate Convolution Code Encoder	PAL32VX10
K7ENC2.PDS	1/2 Rate Convolution Code Encoder	PAL22V10
COUNTER .PDS	Counter	PAL22V10
B8ZSA.PDS	Encoder PAL-A	PAL16R8
B8ZSB.PDS	Encoder PAL-B	PAL16R6
B8ZSC.PDS	Encoder PAL-C	PAL16R6
CRC6.PDS	Error Detection PAL	PAL20R6
DEC_R8.PDS	Decoder PAL	PAL16R8
SUPER.PDS	Super Frame PAL for T1 Interface	PAL16R6
FDP.PDS	T1 Frame Detection PAL for T1 Interface	PAL16L8
SYNC.PDS	T1 Frame Sync PAL for T1 Interface	PAL20R4
V32_2.PDS	Trellis Encoder	PAL20RS8
TREL12.PDS	Trellis Encoder	PAL20RS8
UDCOUNT.PDS	10-Bit Loadable, Even Boundary Up/Down Counter	PAL20X10
9BCASC1.PDS	9-Bit Cascadable Counter, Look Ahead Carry	PAL20X10

D

Table D-1 (Continued):
Files Located On Design Examples Disk

File Name	Description	Device
9BCASC2.PDS	9-Bit Cascadable Counter, MSD	PAL20X10
VIDEO.PDS	Video Shift Register. with Attributes	PAL20X8
BBENCODE.PDS	Manchester Encoder for.Byte and Bit Inputs	PAL22V10
PIPELINE.PDS	Pipeline Controller for Instruction Registers.	PAL16R8D
LIFORAM2.PDS	LIFO RAM Controller Pattern 02 of 2 (8K DEEP)	PAL20X8
180DEGC.PDS	180 -Degree Up/Down Counter	PAL20X10
8BAPPREG.PDS	8 -Bit Successive Approximation Register	PAL20RS10
128LRAM.PDS	LIFO RAM Controller for 128-Deep Stack	PAL32VX10
PORTADPT.PDS	M68020 32-/16-/8-Bit Port Adaptor	PAL20RA10
VIDSREG.PDS	Video Shift Register PAL 3 of 3	PAL32VX10.
VLSYNCG.PDS	Video Line Sync Generator	PAL32VX10
LIFORAM3.PDS	RAM-Based LIFO	PAL22RX8

32VX10 SYNTAX EXAMPLE

32VX10.PDS

PAL32VX10

Example on how to program Boolean equations for a PAL32VX10 part.

STATE SYNTAX EXAMPLES

STRAFFIC.PDS Traffic Signal PMS14R21
 Controller

PRODCDER.PDS QIC-02 Command PMS14R21
 Decoder

This application implements a QIC-02 command decoder using a PROSE device, a PAL device, and a PLE device.

QAM3.PDS ENCODER. PAL20R8

Note: Use this file with PALASM 2 software version 2.23 or later. Otherwise, errors will be detected with the .OUTF equations.

4WAYTRAF.PDS 4-Way Traffic Light PMS14R21
 Controller

D

BOOLEAN EQUATIONS EXAMPLES

9BITCNT.PDS 9-Bit Counter PAL20X10

The 9-bit synchronous counter has parallel load, increment, and hold capabilities. The carry out pin (/CO) shows how to implement a carry out using a register by anticipated one count before the terminal count if counting and the terminal count if loading.

Operations Table

/OC	CLK	/LD	D8-D0	Q8-Q0	Operation
H	X	X	X	Z	HI-Z
L	L	X	X	Q	Hold
L	C	L	D	D	Load
L	C	H	X	QPLUS 1	Increment

8COUNT.PDS 8-Bit Counter PAL20X8

This 8-bit up/down counter has the hold and load capabilities. It sets all the outputs high if SET=high. It loads new value when SET=low and LOAD=high. Else it counts up if UP=high and counts down if UP=low.

DCOUNT.PDS 5-Bit Down Counter PAL20RA10

CONTROL.PDS DEC PDP-11 Unibus Interrupt Controller PAL20RA10

OCTCOMP.PDS Octal Comparator PAL16C1

The octal comparator establishes when two 8-bit data strings (A7-A0) and (B7-B0) are equivalent (EQ=H) or equivalent (NE=H).

3TO8DMUX.PDS 3-8 Demultiplexer PAL16R8

The 3-to-8 demultiplexer with control storage provides a conventional 8-bit demux function combined with control storage functions: load true, load complement, hold, toggle, polarity, clear and preset. Five inputs (/LD, /CLR, /PR, POL, TOG) select one of six operations. The six operations are summarized in the following operations table:

Control		Functions			Polarity		Inputs	Outputs	Operation
/OC	CLK	/CLR	/PR	/LD	POL	TOG	ABC	Q7-Q0	
H	X	X	X	X	X	X	X	Z	HI-Z
L	C	L	X	X	X	X	X	L	Clear
L	C	H	L	X	X	X	X	H	PRESET
L	C	H	H	L	H	X	I	MUX	Load true
L	C	H	H	L	L	X	I	/MUX	Load COMP
L	C	H	H	H	X	L	X	Q	Hold
L	C	H	H	H	X	H	X	/Q	Tog polarity

CRT.PDS CRT Controller Logic PAL20RA10

PORT.PDS 7-Bit I/O Port with Handshake Logic PAL20RA10

BARREL.PDS Barrel Shifter PAL64R32

The 16-bit barrel shifter will shift 16 bits of data (D15-D0) a number of locations into the output pins, as specified by the binary encoded input. Inputs are shown by D. Si are shift amount inputs and Qj are outputs. 16 product terms in each output pair are directed to one output; thus only 16 out of 32 output pins are used.

D

4CNT.PDS 4-Bit Counter PAL16RP4

The 4-bit counter counts up or down and has the clear and load capability. The clear operation overrides count and load. The counter counts up when CLR=low, LOAD=low, and UP=high.

4-16DEC.PDS 4 - 16 Decoder PAL6L16

The 4 to 16 decoder, decodes four binary decoded inputs into one of 16 mutually exclusive outputs, whenever the two enable lines EN1 and EN2 are high. When one or both of the enable lines are low the outputs are all set to high values.

10COUNT.PDS 10-Bit Counter PAL20RS10

The 10-bit counter increments on the rising edge of the clock input (CLK), if CNT input is high. The outputs are HIGH-Z when the enable line (/OE) is high and enabled when the enable line (/OE) is low. The counter is cleared (all lows) if CLR=HIGH.

ADREG16.PDS 16-Bit Addressable Register PAL32R16

The 16-bit addressable register loads one of 16 registers selected by ADDR[0..3] with data input, DATA.

MEMIO.PDS PC I/O Mapper PAL8L14

Personal computers which are hardware compatible with the ubiquitous IBM PC share this I/O map.

UPCOUNT.PDS 5-Bit Up Counter PAL20RA10

FLIPFLOP.PDS Basic Flip Flops PAL16RP8

**MEMORY.PDS Memory Handshake Logic
PAL16RP6**

ARBITER .PDS 3-Bit Arbiter PAL20RA10

LINK.PDS **Serial Data Link** **PAL20RA10**
Controller

LATCH.PDS **Octal Latch** **PAL10H20P8**

The octal latch is an 8-bit latch with load, hold and clear capability. Clear sets all outputs to low and overrides hold. Load operation loads inputs (D0-D7) into the latch. The hold operation holds the previous values of (Q0-Q7).

9BITREG.PDS **9-Bit Register** **PAL20X10**

This is a design of a 9-bit register with parallel load and hold capabilities. The operations of this register are summarized in the following operations table:

/OC	CLK	/LD	D8-D0	Q8-Q0	Operation
H	X	X	X	Z	HI-Z
L	1	H	X	Q	Hold
L	1	L	D	D	Load

10BITREG.PDS **10-Bit Register** **PAL20X10**

D

The 10-bit register loads the data (D9-D0) on the rising edge of the clock(CLK) into the register(Q9-Q0). The data is held in the register until the next positive edge of the clock.

/OC	CLK	D9-D0	Q9-Q0	Operation
H	X	X	Z	HI-Z
L	C	D	D	Load
L	L	X	Q	Hold

BTRAFFIC.PDS **Traffic Signal Controller** **PAL16RP8**

DCODER.PDS **QIC-02 Command Decoder** **PAL20L8**

This PAL is part of the QIC-02 command sequencer design. The primary purpose of this PAL is to encode 8-bit commands into 4-bit command codes. This PAL is also used to encode tape drive status signals and to select the drive number.

K7ENC1.PDS **1/2 Rate Convolution Code Encoder** **PAL32VX10**

1/2 rate convolution code encoder, constraint length ($k=7$). This PAL 32VX10 design implements a high speed convolutional encoder with a constraint length $k=7$ and rate = 1/2. This encoder is used commonly in conjunction with a Viterbi, trellis decoding algorithm. Applications include geostationary satellite communication, high speed local loop bypass networks etc.

K7ENC2.PDS **1/2 Rate Convolution Code Encoder** **PAL22V10**

Convolution code encoder, constraint length ($k=7$). This PAL 22V10 design implements a high speed convolutional encoder with a constraint length $k=7$ and rate = 1/2. This encoder is used commonly in conjunction with a Viterbi, trellis decoding algorithm. Applications include geostationary satellite communication, high speed local loop bypass networks etc.

COUNTER .PDS **Counter** **PAL22V10**

Simulation of counter equations are a combination of active-high or -low and register or combinatorial. Preload and global reset functions are included.

B8ZSA.PDS **Encoder PAL-A** **PAL16R8**

B8ZSB.PDS **Encoder PAL-B** **PAL16R6**

B8ZSC.PDS **Encoder PAL-C** **PAL16R6**

CRC6.PDS **Error Detection PAL** **PAL20R6**

The CRC-6 PAL performs error detection on a serial data stream. CRC-6 PAL supports the T1 Fe standard for error detection. The CRC result can be output either in serial or in parallel.

DEC_R8.PDS **Decoder PAL** **PAL16R8**

SUPER.PDS **Super Frame PAL for T1 Interface** **PAL16R6**

This PAL counts the T1 Frames and controls the Signal Bits extraction process, including Fly Wheeling. It also provides various other signals which indicate the frames with signal bits. The counter is reset with either RSTB or when frame detection is SUNK and frame 1 occurs from two different sources (FRM1 & SOF).

FDP.PDS **T1 Frame Detection PAL for T1 Interface** **PAL16L8**



This PAL monitors 12 193rd bits in the incoming T1 NRZ data stream. It detects any valid Frame Pattern (start of any Frame) and the start of Frame 1.

SYNC.PDS **T1 Frame Sync PAL for T1 Interface** **PAL20R4**

This PAL decides whether the T1 Interface is in Frame Sync, Sync, or Out of Sync. It controls the Frame Sync process.

V32_2.PDS **Trellis Encoder** **PAL20RS8**

This PAL performs the signal mapping onto the 32 state constellation according to CCITT V.32, 9600 bps specification.

TREL12.PDS **Trellis Encoder** **PAL20RS8**

This PAL performs the signal mapping onto the 32 state constellation according to CCITT V.32, 9600 bps specification.

UDCOUNT.PDS **10-Bit Loadable,
Even Boundary
Up/Down Counter** **PAL20X10.**

9BCASC1.PDS **9-Bit Cascadable
Counter,.Look
Ahead Carry** **PAL20X10**

9BCASC2.PDS **9-Bit Cascadable
Counter, .MSD** **PAL20X10**

VIDEO.PDS **Video Shift Register.
with Attributes** **PAL20X8**

BBENCODE.PDS **Manchester Encoder
for.Byte and Bit Inputs** **PAL22V10**

PIPELINE.PDS **Pipeline Controller
for Instruction
Registers.** **PAL16R8D**

LIFORAM2.PDS **LIFO RAM Controller
Pattern 02 of 2
(8K DEEP)** **PAL20X8**

180DEGC.PDS **180-Degree Up/
Down Counter** **PAL20X10.**

8BAPPREG.PDS	8-Bit Successive Approximation Register	PAL20RS10.
128LRAM.PDS	LIFO RAM Controller for 128-Deep Stack	PAL32VX10
PORTADPT.PDS	M68020 32-/16-/8-Bit Port Adaptor	PAL20RA10
VIDSREG.PDS	Video Shift Register PAL 3 of 3	PAL32VX10.
VLSYNCG.PDS	Video Line Sync Generator	PAL32VX10
LIFORAM3.PDS	RAM-Based LIFO	PAL22RX8

Provides control and addressing for a PAL22RX8 32-location-deep RAM-based LIFO.

D

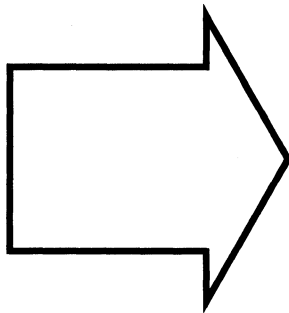


Table of Contents	vii
Software Errata	ERR
Introduction	1
Installing PALASM [®] 2 Software	2
PDS Syntax	3
Boolean Equation Design	3A
State Machine Design	3B
Simulation	3C
Using PALASM 2 Software	4
Installation and Operation Notes	A
Programmer Notes	B
Device Specific Syntax	C
PAL [®] Design File Library	D
Error Messages	E
Submitting a HAL [®] Design to MMI	F
PALASM 2 Syntax Diagram	G
Supplementary Software	H
JEDEC Standard No. 3	I
Release Notes	RN
Index	IX
PLEASM [™] Manual	

APPENDIX E

ERROR MESSAGES

ERRORS REPORTED BY PALASM2

Error Number	Error Message
1.....	Unexpected EOF
2.....	Identifier name is too long
3.....	Illegal character
4.....	Malformed STRING declaration
5.....	Variable is already declared
6.....	Incomplete or invalid operator
7.....	Bad for loop variable usage
8.....	Malformed expression
9.....	Expected an identifier
10.....	Expected an operator
11.....	Expected =, too many strings
12.....	Illegal variable usage
13.....	Undeclared variable
14.....	Malformed variable
15.....	Unexpected part type
16.....	Unbalanced parens or brackets
17.....	Malformed statement
18.....	Missing keyword END
19.....	Missing keyword or section SIM
20.....	Missing keyword CHIP
21.....	This array already declared
22.....	Expected keyword BIN
23.....	Expected a =
24.....	Title not specified

E

- 25.....Pattern not specified
- 26.....Revision not specified
- 27.....Author not specified
- 28.....Company not specified
- 29.....Date not specified
- 30.....Missing keyword EQUATION
- 31.....VCC, GND, NC not allowed here
- 32.....Dangling equation found
- 33.....Arch. defined as Mealy already
- 34.....Arch. defined as Moore already
- 35.....Pin defined as Reset already
- 36.....Pin defined as Enable already
- 37.....Missing keyword STATE
- 38.....Missing keyword DEFAULT_BRANCH
- 39.....Missing Argument
- 40.....Redundant definition
- 41.....Improper usage of setf or rstf. For
32VX10, proper usage is:
GLOBAL.setf/rstf.

ERRORS REPORTED BY XPLOT

XPLOT can detect a total of 52 errors (plus one warning) that are identified uniquely by numbers. Depending on the type of error, an error message alone appears on the terminal, or the name of the pin where the error occurred is also reported. Once an error is caught for an equation, further processing of that equation is terminated, and the next equation is processed. The format for error reporting is as follows:

ERROR NUMBER : error number

PIN :<name of the pin where the error
 occurred>

ERROR MESSG. :error message explaining the type of error

All the errors detected are device-related errors such as extra pins, conflicting use of pins, extra number of product terms, etc.

Warning Number	Warning Message
1.....	CLOCK FOR THIS PIN ASSUMED TO BE GND

Error Number	Error Message
0.....	Unexpected end of input TRE or PAL file, File invalid

This is caused by one of three possible conditions:

A. XPLOT has been run on a design that has syntax errors detected by the frontend. (PALASM2.TRE will contain the word ERROR).

B. The PDF or TRE file contains a different version number from that stored within the XPLOT program. (Re-run the PALASM 2 software frontend or move the PDF file from the distribution disk to where the XPLOT program can locate it).

C. The PDF or TRE file is corrupted by bad disk media or a disk error. (Get a new, freshly formatted disk and repeat processing.)



- 1.....This output is registered but another output of same bank is used as combinatorial.
- 2.....This output is combinatorial but another output of same bank is used as registered.
- 3.....This output is combinatorial but has SET function defined
- 4.....This output is combinatorial but has RESET function defined
- 5.....This output is combinatorial but has CMBF function defined
- 6.....This output is a combinatorial but has CLOCK function defined
- 7.....This output is using more than 8 product terms
- 8.....These outputs are using more than 16 product terms
- 9.....The polarity of the output on the left side of the equation should be opposite to the polarity in the pinlist
- 10.....SET function has more than one product term
- 11.....This output is used as combinatorial but is defined as registered output.
- 12.....This output is used as registered but is defined as a combinatorial output.
- 13.....This output is a registered but has no CLOCK function defined for it.
- 14.....RESET function has more than 1 product term
- 15.....THREE STATE function has more than 1 product term
- 16.....CMBF function has more than 1 product term
- 17.....CLOCK function has more than 1 product term
- 18.....This output is using more than 4 product terms

- 19.....This output is using more than 7 product terms
- 20.....These outputs are using more than 8 product terms
- 21.....These outputs are using more than 14 product terms
- 22.....The polarity of the output on the left side of the equation should be same to the polarity in the pinlist
- 23.....This output is using more than 2 product terms
- 24.....This output cannot be used as a feedback signal
- 25.....Cannot define a simple/functional equation for an input pin
- 26.....This output is using more than 3 product terms
- 27.....This output is using more than one XOR (:+) term
- 28.....This output is using more than two product terms per one XOR term
- 29.....This output is using more than four product terms per one XOR term
- 30.....This output is using more than 16 product terms
- 31.....Pins allowed: XX Pins used: XX
- 32.....Undefinable/unusable pin in this equation
- 33.....Simple/Functional equation already defined
- 34.....Cannot define functional equation for this output
- 35.....The product term XXX cannot be edited
- 36.....Pins 23/14 do not have XOR function in PAL20X8
- 37.....Pins 23/22/21/14/15/16 do not have XOR fnc. in PAL20X4
- 38.....Programmable Combinatorial (.CMBF) function not supported



- 39.....XNOR (:*) is not supported in any PAL device
- 40.....XOR (:+) is not a legal operator for non-XOR PAL devices. e.g. PAL16R8,PAL20S10 etc...
- 41.....The functional equation for this pin is using a registered equation(:=) instead of combinatorial equation (=).
- 42.....PAL16A4 and PAL16X4 are not supported by XPLOT
- 43.....PAL16C1 and PAL20C1 cannot have more than one equation defined for an output
- 44.....Exceeded the maximum number of tokens used per equation
- 45.....This output is using more than 1 product term
- 46.....This output is using more than 6 product terms
- 47.....An output pin in the bank of this pin was previously used as combinatorial. This pin is now being used as feedback. Output pins in combinational bank cannot be used as feedback.
- 48.....An output pin in the bank of this pin was previously used as feedback. This pin is now being used as combinatorial output. Output pins used as feedback have to be registered.
- 49A20RA10 Architectural/Application Errors

3: This output is combinatorial but has SET function defined

4: This output is combinatorial but has RESET function defined

5: This output is combinatorial but has CMBF function defined

6: This output is combinatorial but has CLOCK function defined

13: This output is registered but has no CLOCK function defined

- 49BCannot have same polarity and also :+ term in an equation
- 50AToken GND is not defined correctly in pin list
- 50BOnly one side of :+ gate can have more than one product term
- 51AToken VCC is not defined correctly in pin list
- 51BProduct terms already used by external node equation
- 52.....Product terms already used by internal node equation
- 53.....Registered equation for an OUTPUT pin must only be in terms of its internal node. Use internal nodes to define registered equations.
- 54.....TRST equations should be defined only as active high
- 55.....SETF equations should be defined only as active high
- 56.....RSTF equations should be defined only as active high
- 57.....Setting >TRST=GND in a 32VX10 means that you cannot define the output, since you have turned it off.
- 58.....Proper usage is GLOBAL>RSTF = . . .
- 59.....Using Xor not allowed for 22V10 part
- 60.....This output is using more than 10 product terms.



- 61.....This output is using more than 12 product terms
- 62.....This output is using more than 14 product terms.
- 63.....Can not use XOR in Tristate equation
- 64.....This part is not supported yet
- 65.....Above PDF file not found
- 66.....On IBM/PC, you cannot have the file name without extension - more than 8 characters.
- 67.....On vax, you cannot have filename without extension - more than 11 characters
- 68.....State Machine syntax is not supported by Xplot. Please run Expander Program

Note: For error number 68, run MINIMIZE program, not Expander.

ERRORS REPORTED BY MINIMIZE

Error Number	Error Message
0010.....	Simulation boolean equation missing.
0020.....	Boolean equation missing.
0030.....	Invalid operator in boolean equation.
0040.....	Relational expression missing left-hand side.
0050.....	Relational expression missing right-hand side.
0060.....	Invalid operator in boolean equation.
0070.....	Relational expression missing left-hand side.
0080.....	Relational expression missing right-hand side.
0090.....	Invalid operator in boolean equation.

- 0100.....Relational expression missing left-hand side.
- 0110.....Relational expression missing right-hand side.
- 0120.....Invalid operator in boolean equation.
- 0130.....Relational expression missing left-hand side.
- 0140.....Relational expression missing right-hand side.
- 0150.....Invalid operator in boolean equation.
- 0160.....Relational expression missing left-hand side.
- 0170.....Relational expression missing right-hand side.
- 0180.....Invalid operator in boolean equation.
- 0190.....Relational expression missing left-hand side.
- 0200.....Relational expression missing right-hand side.
- 0210.....Logical operator missing left-hand side.
- 0220.....Logical operator missing right-hand side.
- 0230.....Logical operator missing left-hand side.
- 0240.....Logical operator missing right-hand side.
- 0250.....Logical operator missing left-hand side.
- 0260.....Logical operator missing right-hand side.
- 0270.....Logical operator missing left-hand side.
- 0280.....Logical operator missing right-hand side.
- 0290.....Logical operator missing operand.
- 0300.....Logical unary operator has additional right-hand side.
- 0310.....Simulation vcc has bad syntax.
- 0320.....Simulation gnd has bad syntax.
- 0330.....Simulation pin has bad syntax.
- 0340.....Simulation state has bad syntax.
- 0350.....Simulation condition has bad syntax.
- 0360.....Illegal Simulation boolean equation.
- 0370.....Illegal boolean equation.
- 0410.....Missing integer equation.
- 0420.....Binary operator missing left-hand side.



0430.....	Binary operator missing right-hand side.
0440.....	Binary operator missing left-hand side.
0450.....	Binary operator missing right-hand side.
0460.....	Binary operator missing left-hand side.
0470.....	Binary operator missing right-hand side.
0480.....	Binary operator missing left-hand side.
0490.....	Binary operator missing right-hand side.
0500.....	Integer constant has bad syntax.
0510.....	Integer iterator has bad syntax.
0520.....	Illegal integer equation encountered.
0560.....	While statement missing boolean equation.
0570.....	If statement missing Simulation boolean equation.
0580.....	If statement missing Begin.
0590.....	If statement has second clause, which is missing Begin.
0600.....	If statement's second clause has bad syntax.
0610.....	For statement missing registered assignment.
0620.....	For statement missing Iterator.
0630.....	For statement missing To.
0640.....	For statement's To is missing left-hand side.
0650.....	For statement's To is missing right-hand side.
0660.....	For statement's Iterator has bad syntax.
0670.....	For statement's To has bad syntax.
0680.....	Simulation pin has bad syntax.
0690.....	Simulation state has bad syntax.
0700.....	Simulation condition has bad syntax.
0710.....	Simulation not missing element.
0720.....	Simulation not pin has bad syntax.
0730.....	Simulation not pin has bad syntax.
0740.....	Simulation not condition has bad syntax.
0750.....	Simulation not condition has bad syntax.
0760.....	Simulation list: not with state.
0770.....	Simulation don't care missing element.

0780.....	Simulation don't care pin has bad syntax.
0790.....	Simulation don't care pin has bad syntax.
0800.....	Simulation don't care condition has bad syntax.
0810.....	Simulation don't care condition has bad syntax.
0820.....	Simulation don't care with illegal element.
0830.....	Illegal Simulation equation.
0860.....	State output equation missing outf.
0870.....	State output equation missing State name.
0880.....	State output equation missing condition.
0890.....	State output equation has bad syntax.
0900.....	State output equation has bad syntax.
0910.....	State output equation has bad syntax.
0920.....	State transition equation missing State.
0930.....	State transition equation missing condition.
0940.....	State transition equation has bad syntax.
0950.....	State Transition equation has bad syntax.
1010.....	Pin equation missing left-hand side.
1020.....	Pin equation missing right-hand side.
1030.....	Pin equation has bad syntax.
1040.....	Pin equation pin has bad syntax.
1050.....	Pin equation not has bad syntax.
1060.....	Pin equation Setf has bad syntax.
1070.....	Pin equation Clk has bad syntax.
1080.....	Pin equation Set has bad syntax.
1090.....	Pin equation Reset has bad syntax.
1100.....	Pin equation Trst has bad syntax.
1110.....	Pin equation Outf has bad syntax.
1120.....	Illegal pin definition.
1160.....	State or missing left-hand side.
1170.....	State or missing right-hand side.
1180.....	State or not connected to a branch
1190.....	State default branch missing left-hand side.
1200.....	State default branch missing right-hand side.

- 1210.....State default branch not connected to a default.
- 1220.....State next branch missing left-hand side.
- 1230.....State next branch missing right-hand side.
- 1240.....State and missing left-hand side.
- 1250.....State and missing right-hand side.
- 1260.....State don't care missing pin.
- 1270.....State don't care has bad syntax.
- 1280.....State don't care has bad syntax.
- 1290.....State not missing pin.
- 1300.....State not has bad syntax.
- 1310.....State not has bad syntax.
- 1320.....State pin has bad syntax.
- 1330.....Illegal default for output or state value equation.
- 1340.....Illegal default for state transition equation.
- 1350.....State's state name has bad syntax.
- 1410.....Default branch has bad syntax.
- 1420.....Default branch has bad syntax.
- 1430.....Default branch has bad syntax.
- 1440.....Default branch has bad syntax.
- 1450.....Default branch has bad syntax.
- 1460.....Default branch has bad syntax.
- 1470.....Default branch has bad syntax.
- 1480.....Illegal default branch.
- 1490.....Default branch missing state.
- 1500.....Output hold has bad syntax.
- 1510.....Default output has bad syntax.
- 1520.....Output hold pin has bad syntax.
- 1530.....Illegal pin in output hold.
- 1540.....Default output not missing pin.
- 1550.....Default output pin has bad syntax.
- 1560.....Default output pin has bad syntax.
- 1570.....Default output don't care missing pin.
- 1580.....Default output pin has bad syntax.
- 1590.....Default output pin has bad syntax.
- 1600.....Default output pin has bad syntax.
- 1610.....Illegal element in default-output.
- 1620.....Illegal state value equation.

- 1660.....State value equation missing State.
- 1670.....State value equation missing Bin or
output list.
- 1680.....State value equation with Bin missing pin
list.
- 1690.....State value equation has bad syntax.
- 1700.....State value equation has bad syntax.
- 1710.....State value equation has bad syntax.
- 1720.....State value equation is missing a pin.
- 1730.....State value equation has bad syntax.
- 1740.....State value equation has bad syntax.
- 1750.....Illegal state value equation element.
- 1760.....State value equation missing pins.
- 1770.....Mealy-machine has bad syntax.
- 1780.....Mealy-machine has bad syntax.
- 1790.....Moore-machine has bad syntax.
- 1800.....Moore-machine has bad syntax.
- 1810.....Master reset has bad syntax.
- 1820.....Master reset has bad syntax.
- 1830.....Output-enable has bad syntax.
- 1840.....Output-enable has bad syntax.
- 1860.....Equal sign defines illegal equation.
- 1870.....Registered assignment defines illegal
equation.
- 1880.....Equation of unknown type.
- 1890.....Condition equation missing left-hand
side.
- 1900.....Condition equation missing right-hand
side.
- 1910.....Condition equation has bad syntax.
- 1920.....Condition equation has bad syntax.
- 1930.....Equal sign missing arguments.
- 1940.....Colon equals missing arguments.
- 1950.....Asked to parse Null equation.
- 1960.....Too many state tokens, power up cannot
be stored.
- 2010.....General output error.
- 2020.....Output file write error occurred.
- 2030.....Output file not open.

2040.....	Output file already open.
2050.....	Output file cannot be opened.
2210.....	Illegal Simulation element.
2220.....	Illegal Simulation command type.
2230.....	ISimulation state not active.
2240.....	Simulate Section: Conditions aren't allowed...yet.
2310.....	Txtfile cannot be opened.
2320.....	Nothing in txtfile.
2330.....	Message not in txtfile.
2340.....	End of txtfile reached before message.
2350.....	Number withough message in txtfile.
2360.....	Went over maximum number for txtfile.
2510.....	Undefined value for state bit.
2520.....	State Value Equations must use exactly the same state bits.
2530.....	Not enough columns for tree.
2710.....	S-equation memory retrieve failed.
2720.....	partial memory retrieve failed.
2730.....	products memory retrieve failed.
2740.....	State value memory retrieve failed.
2750.....	State value additional memory retrieve failed.
2760.....	Next branch memory retrieve failed.
2770.....	Next branch additional memory retrieve failed.
2780.....	State name memory retrieve failed.
2790.....	State name additional memory retrieve failed.
2800.....	Output branch memory retrieve failed.
2810.....	Output branch additional memory retrieve failed.
2820.....	State output memory retrieve failed.
2830.....	State output additional memory retrieve failed.
2840.....	Condition memory retrieve failed.
2850.....	Condition additional memory retrieve failed.
2860.....	Part element memory retrieve failed.

- 2870.....Part name memory retrieve failed.
- 2880.....Sop equation memory retrieve failed.
- 2890.....Pin memory retrieve failed.
- 3010.....Can't initialize design input file.
- 3020.....Can't open design input file.
- 3030.....Design input file equation read error.
- 3040.....Can't rename design output file.
- 3050.....Part not supported.
- 3060.....Prose device selected, no processing performed.
- 3070.....Can't rename prose design file.
- 3080.....Can't delete temporary file.
- 3090.....IDevice: Minimize will remove XOR from equation for 'X' part.
- 3210.....Illegal boolean equation.
- 3220.....More signals than products count.
- 3230.....Conditions: Circular condition encountered.
- 3240.....Illegal condition encountered.
- 3250.....Equation: Illegal state.
- 3260.....Equation: Illegal state. State is unreachable.
- 3270.....Equation: Can't process simulate boolean equation.
- 3410.....Minimizer: Minimization failed.
- 3510.....File: Read Error. Part definition file (PDF).
- 3520.....File: Open Error. Part definition file (PDF).
- 3610.....Equations: Illegal equation in equations section.
- 3620.....Equations: Illegal equation suffix.
- 3630.....Equations: Multiple equations for a node.
- 3640.....Equations: Illegal registered equation.
- 4010.....State: Illegal equation in state section.
- 4020.....State Transition: Last equation has a default of next state.
- 4030.....State: Both mealy and moore specified (default: mealy).
- 4040.....State: Moore machine with output conditions.

ERROR MESSAGES

4050.....	State Value: Multiple equations for a state.
4060.....	State Transition: Multiple equations for a state.
4070.....	State Transition: Multiple defaults for a state.
4080.....	State Output: Multiple equations for a state.
4090.....	State Output: Multiple defaults for a state.
4100.....	Conditions: Illegal equation in conditions section.
4110.....	Conditions: Multiple equations for a condition.
4120.....	State List: Illegal entry. (Entry must be a pin.)
4130.....	State List: Multiple occurrences of a pin.
4140.....	State: Unreached state(s).
4150.....	State: Need to automatically assign a new state bit.
4160.....	State Transition: Default Branch is the only transition.
4170.....	State: Unspecified state transition equation.
4180.....	State: Unable to automatically assign a new state bit.
4190.....	State: Device selected isn't supported for state machines.
4200.....	State Transition: Undefined default next state.
4210.....	State Output: Registered and combinatorial equations are mixed.
4220.....	State Output: Hold specified for a combinatorial equation.
4230.....	State: Moore Machine with combinatorial outputs.
4240.....	State Transition: Conditions overlap.
4250.....	State Output: Conditions overlap.
4500.....	Minimizer: Memory allocation failed. Block requested too large.

- 4501.....Minimizer: Memory allocation failed. Out of memory.
- 4502.....Minimizer: Memory free failed.
- 4510.....Minimizer: No solution possible (check equations).
- 4511.....Minimizer: Inconsistent Frequency Table.
- 4520.....Minimizer: Cube not covered in REDUCE.
- 4530.....Minimizer: split_select index error.
- 4531.....Minimizer: split_two_select index error.
- 4532.....Minimizer: large_cube_index -- bestindex -1.
- 4540.....Minimizer: cover capacity exceeded.
- 4550.....Minimizer: ON and OFF sets aren't orthogonal (check equations).

ERRORS REPORTED BY PROASM-PROSIM

Error Number	Error Message
101	Logically there are <integer value> product terms which could not fit into this device.
102.....	Logically there are <integer value> states which could not fit into this device.
103.....	Logically there are more than <integer value> conditions which are impossible to fit into this device.
104.....	Logically there are more than <integer value> states which are impossible to fit into this device.
105.....	Only one machine type is allowed in the state section.
106	For MOORE machine, there is only one output in each state. State <string> has more than one output.

E

- 111.....Number of pins in CHIP section does not match number of pins in actual device.
- 112.....Position of pin <pin name> should be VCC in CHIP section.
- 113.....Position of pin <string> should be GND in CHIP section.
- 114.....PROASM does not support this device, only PROSE device.
- 117.....<string>.OUTF is specified as combinatorial output, MUST be REGISTERED.
- 118Unrecognizable next state in state equation <string>.
- 119POWER.OUTF is not defined in this MEALY machine. Output pattern is <string> when you clock this device after power up or reset.
- 120.....Unrecognizable output in state output equation <string>.
- 121.....Unrecognizable condition in state equation <string>.
- 122.....Only output pin is allowed in DEFAULT_OUTPUT statement. Pin <string> is not.
- 123Only output pin is allowed in OUTPUT_HOLD statement. Pin <string> is not.
- 124Only HOLD_STATE, or NEXT_STATE, or a specific state name is allowed after DEFAULT_BRANCH statement.
- 125.....No .CLKF/.RSTF/.SETF/.TRST are allowed for state <string>.
- 126Illegal output equation for state <string>. Only STATE.OUTF is allowed in the left hand side.
- 127More than 1 local default next state in state <string>.
- 128No default next state in state equation <string>.

- 129Only one next state is allowed after POWER_UP state. For MEALY model, one output pattern is also allowed.
- 130.....Illegal state equation for state <string>.
- 131.....State <string> has more than 4 next states.
- 132.....No .CLKF/.RSTF/.SETF/.TRST/.OUTF are allowed after POWER_UP state, no '/' is allowed before it either.
- 133.....Illegal state name usage, /<string> is not allowed.
- 134.....State <string> has more than 4 conditions.
- 135.....Unmatched condition in state <string>'s output and transition equations.
- 136.....State <string> output has more than 4 conditions.
- 137.....Only Output pin is allowed in the right hand of state.outf equation. Pin <string> is not an output pin.
- 138.....Conflict output specified for pin <string> in state.outf equation.
- 139More than one POWER_UP specified in STATE section.
- 140.....State <string> is defaulted to be the state after power-up.
- 141.....Only sum of products is allowed in the right hand side of condition equation <string>.
- 142.....PROASM does not support DeMorganizing condition. So /<string> is not allowed.
- 143.....No .CLKF/.RSTF/.SETF/.TRST/.OUTF are allowed for condition <string>.
- 144.....Only combinatorial condition is allowed, ':=' is not allowed in condition <string>.
- 145.....Only input pin is allowed in the right hand of condition equation. Pin <string> is not a input pin.



- 146.....Not enough products available in processing condition <string>.
- 161.....Current device doesn't have enough PAL product terms for this design.
- 162.....Current device doesn't have enough PROM locations for this design.
- 171Overlapping decision in state <string>.
- 172.....Condition <string> overlaps with its following condition(s).
- 173.....Fatal overlapping decision in state <string>.
- 174.....Same condition <string> causes branching to 2 different states.
- 181.....There is error in accessing the PALASM2.TRE file.
- 182.....End of section is encountered unexpectedly.
- 183.....No more memory left to read in the PALASM2.TRE file.
- 184.....A previous PALASM2.TRE file has been opened and has not been closed.
- 185.....PALASM2.TRE file has never been successfully opened.
- 191.....Prose Definition File open/close error.
- 192.....Data accessing error in Prose Definition File.
- 196.....Xplot/Jedec output file open/close error.
- 197.....Xplot/Jedec write error.
- 198Xplot/Jedec output file name length is more than <integer value>.
- 600.....Simulator message not implemented yet.
- 605.....Error while initializing the error program.
- 610.....Simulator encountered a system allocation error while attempting to create a dynamic data structure.
- 615.....An error occurred while opening the intermediate design file. The cause is probable either incompatible simulator

- and frontend programs or intermediate file does not exist.
- 616.....An error occurred reading from intermediate design file within the state section.
- 617.....An error occurred reading from intermediate design file within the equation section.
- 618An error occurred reading from intermediate design file within the simulation section.
- 620.....An error occurred within the text handler.
- 621.....Text message file index not found. Probable cause is incorrect message file for this version of the simulator.
- 625.....Simulator could not locate PDF file for device
- 626.....Simulator detected an error while reading from PDF file. Simulator detected an unknown error when reading from PDF file.
- 630.....Internal error detected within history/trace program. Partial History Output.
- 631Internal error detected within history/trace program. Cannot Re-Initialize History.
- 632.....Internal error detected within history/trace program. Partial Trace Output.
- 633.....Internal error detected within history/trace program. History Names Not Terminated.
- 634.....Internal error detected within history/trace program. Trace Names Not Terminated.

- 635.....Internal error detected within history/trace program. More values written to output than there are names.
- 636.....Internal error detected within history/trace program. Less values written to output than there are names.
- 637.....Simulator could not open history file.
- 638.....Simulator could not open trace file.
- 646Error occurred while creating history file name.
- 647Error occurred while creating trace file name.
- 648Error occurred while creating JEDEC input file name.
- 649.....Error occurred while creating JEDEC output file name.
- 650.....Simulator cannot construct file name; name is too long.
- 651.....Simulator encountered an unknown device type for the PROSE device.
- 652.....More than one clock is defined in the PROSE PDF file. Extra clock pins will be ignored.
- 653More than one initialize/output enable pin is defined in the PROSE PDF file. Extra pins will be ignored.
- 654.....Value of condition in state equation is not boolean value.
- 654Condition will be ignored.
- 655.....Three-State control does not evaluate to a boolean value. Control will be ignored.
- 656.....Condition in WHILE/IF construction does not evaluate to a boolean value. Construction will be ignored.
- 657.....Trace Qualifier does not evaluate to a boolean value. Trace will be displayed.
- 658.....Register asynchronous load does not evaluate to a boolean value. Load will be ignored.

- 659.....Register asynchronous reset does not evaluate to a boolean value. Reset will be ignored.
- 660.....Register asynchronous set does not evaluate to a boolean value. Set will be ignored.
- 661Register clock does not evaluate to a boolean value. Register will be clocked with a new value.
- 662.....Unexpected token found when evaluating internal PROSIM equation. This error may indicate internal PROSIM failure and should be reported to the factory.
- 663.....Simulation cycle prematurely terminated. Circuit did not settle before simulation cycle limit exceeded.
- 664.....Signal value does not compare with internal simulation value.
- 665.....Trace output is already on. The TRACE_OFF command will be assumed immediately before this TRACE_ON command.
- 666.....Unexpected TRACE_OFF. Trace output was not active.
- 667.....Simulator can not SETF a state register. You should use PRLDF to set state value.
- 668.....Signal can not be set through the SETF command.
- 669.....Signal not found or is not a boolean signal in SETF command.
- 670.....There is not a state register in the simulator to preload.
- 671.....Signal can not be set through the PRLDF command.
- 672.....Signal not found or is not a boolean signal in PRLDF command.

- 673.....There is not a state register in the simulator to check state value against.
- 674Signal not found or is not a boolean signal in CHECKF command.
- 675.....There is not a state register in the simulator to qualify state name against.
- 676.....Signal not found or cannot be traced in a TRACE_ON command.
- 677The TRACE_OFF command does not allow arguments.
- 678.....Unexpected intermediate file data structure encountered during the FOR command.
- 679.....Unexpected intermediate token encountered in FOR command.
- 680.....Unexpected intermediate file data structure encountered during construction of boolean/arithmetic expression.
- 681.....Unexpected intermediate token encountered while constructing an arithmetic expression.
- 682Unexpected intermediate token encountered while constructing a boolean expression.
- 683There is no state register defined within the simulator to test in a boolean equation.
- 684Simulator detected an error when writing to an history or trace file.
- 685.....Internal error detected within history/trace program. Unknown Object.
- 686.....Unexpected intermediate file data structure encountered during construction of terminal expression.
- 687Unexpected intermediate file token encountered while constructing a terminal expression.

- 688.....Multiple prefixes are not allowed for signal or state names.
- 689.....Multiple suffixes are not allowed for signal or state name
- 690.....Expected signal usage is not consistent with its earlier definition
- 690.....definition
- 691.....Simulator encountered integrity check failure while reading from intermediate page file.
- 692.....Simulator does not have enough space for state names.
- 693.....State output element either has a suffix or is an unexpected state value.
- 694Unexpected intermediate file data structure encountered during construction of state output element.
- 695.....Unexpected intermediate file data structure encountered during construction of next state equation.
- 696.....Unexpected intermediate file token encountered while constructing next state equation.
- 697.....Unexpected intermediate file data structure encountered during construction of next state or state output equations.
- 698.....Unexpected intermediate file data structure encountered during construction of left hand side of next state or state output equations.
- 699.....Multiple state suffixes found while constructing state equation.
- 700.....Unexpected intermediate file token encountered while constructing left hand side of a state equation.
- 701.....Unexpected combination of token on left hand side of state equation. The simulator only accepts state := next state

- equation or \ state. OUTF = state output equation.
- 702..... Unexpected internal file token encountered in state section.
- 703..... Multiple models for state equations can not be specified.
- 704..... Simulator cannot implement both initialize and enable models.
- 705..... Too many state output default values are specified. Extra default will be ignored.
- 706..... Moore model only accepts default output states. Other specifications will be ignored.
- 707..... Too many next state default values are specified. Extra default will be ignored.
- 708..... Last next state equation must have an explicit default state specified. Unknown next state will be assumed.
- 709..... Unexpected internal token encountered.
- 710..... Unexpected intermediate file data structure encountered during construction of signal equation.
- 711..... Unexpected intermediate file token encountered while constructing signal equation.
- 712..... Simulator does not recognize the signal equation type. The simulator currently supports only the following forms: signal = equation.
- 713 Signal already defined.
- 714..... Simulator does not support registered equations. The simulator currently supports only the following forms signal = equation.
- 715..... Signal does not have a definition for this signal. Signal cannot be set because it is not at the device boundary.

- 716.....Buried registers can never be preloaded.
- 717.....Unexpected intermediate file token encountered within a default branch command.
- 718Simulator does not have a definition for this signal. It will use a boolean don't care value in its stead.
- 719Unexpected intermediate file token encountered while constructing simulation command.
- 720.....Prefixes (/ or %) should not be used on arguments to the OUTPUT_HOLD command. The prefixes are being ignored.
- 721Simulator can not CLOCKF a state register. You should use PRLDF to set state value.
- 722.....Signal can not be set through the CLOCKF command.
- 723.....Signal not found or is not a boolean signal in CLOCKF command.
- 724.....CLOCKF command without any arguments. Command will not affect simulation since there is not signal specified to clock.
- 725.....Simulation section not found in PROSE device specification.
- 726.....Buried state output can never be preloaded.
- 727TRACE_ON prefix (probably %) not supported by simulator. Prefix will be ignored.
- 728Prose part description file (PDS file) and part definition file (PDF file) pin counts do not agree.
- 729The Prose definition file (PDF file) defines more than one serial diagnostic input/output pin, serial diagnostic clock,

- serial mode control or device clock pin.
This in not supported in the present
version of the simulator.
- 730.....The Prose definition file (PDF file) did
not define either the serial diagnostic
input/output pin, serial diagnostic clock,
serial mode control or device clock pin.
This in not supported in the present
version of the simulator.
- 731.....The simulator only supports register
outputs on output pins of a Prose
device.
- 732.....The simulator does not allow signals to
be defined to VCC, GND and NC pins.
- 733.....The simulator does not allow definition
on either the serial diagnostic
input/output pin, serial diagnostic clock,
or the serial mode control.
- 734.....The simulator detected an unknown pin
type in the Prose definition file (PDF
file).
- 735.....The simulator does not allow a defined
signal (an output) on an input pin.
- 736.....The simulator does not allow a
undefined signal (an input) on an output
pin.
- 737.....The simulator has output more than 512
JEDEC vectors.
- 738.....The simulator has detected an error
during the device simulation and is
stopping the generation of JEDEC test
vectors.
- 739The simulator has detected too many
state bits for this version of the simulator.
- 740.....The trace output was active and there
was not a TRACE_OFF command at the
end of the simulation. The simulator will
assume a TRACE_OFF.

- 741 Unexpected end of file encountered while reading the JEDEC input file.
- 742 Too many state value bits were encountered while reading the JEDEC input file.
- 743 Unexpected delimiter between the state value and the state name encountered while reading the JEDEC input file.
- 744 The simulator could not locate a state name defined within the JEDEC input file with the state names defined within the Prose device specification.
- 745..... An illegal fuse value was encountered while reading the
- 745..... JEDEC input file.
- 746 The JEDEC input file and the Prose definition file do not describe the same device.
- 747 The simulator attempted to open the JEDEC input file twice.
- 748 The simulator could not close the JEDEC input file.
- 749 The simulator could not the JEDEC output file.
- 750..... The simulator could not close the JEDEC output file.
- 751 The simulator has produced a JEDEC test vector which is too long.
- 752 An address defined within the JEDEC input file does not match the calculated internal value of this address.
- 753 The simulator has encountered an illegal JEDEC file format when copying the JEDEC input file to the JEDEC output file.

**ERRORS
REPORTED BY
ZHAL**

Error Number	Error Message
124.....	Make sure that your design passes XPLOT
3424.....	Declared device pin/node count from PDS file is <integer value>
3428.....	Max device pin/node count exceeded
3440	Force Route Failure: force route range is full for device pin <integer value>

Possible solutions:

- 1)Try swapping pins around
- 2)Interleave large and small equations on the output pins (Adjacent pins with large equations often fail to route)
- 3)Reduce equation size by removing redundancy if possible
- 4)May be possible to hand route while keeping present device pinout

3448	Force Route Failure: equation product term too large for pin <integer value>
3452	Equation product term may contain a maximum of <integer value> signals
3456	Force Route Failure: there is no space available for equation on device pin <integer value>. Remaining routing range is insufficient

Possible solutions:

- 1)Try swapping pins around

2) Interleave large and small equations on the output pins (Adjacent pins with large equations often fail to route)

3) Reduce equation size by removing redundancy if possible

4) May be possible to hand route while keeping present device pinout

- 3460 Force route range does not exist Illegal equation on device pin <integer value> Design may be illegal for target device Make sure that your design passes XPLOT
- 3488 Error selecting enable condition for pin <integer value>
- 3492 Force Route: failed to place signal for device pin <integer value> into array

Possible causes and solutions:

1) Maximum fan out reached on the signal. Use new signal or hand route

2) Error in PDF: consult FAE or factory

- 3520 Illegal use of VCC or GND keywords in equation for device pin <integer value>
- 3524 Maximum number of product terms exceeded on device pin <integer value>
- 3528 Maximum number of product terms allowed in an equation is <integer value>
- 3532 Illegal device pin: pin <integer value> does not exist on target device
- 3536 Error in array column allocation

Error was found on signal for device pin <integer value>



Possible problems and solutions:

1) High probability that there are too many signals in the equation set. Modify equation to reduce number of signals used

2) Signal has no connect to array such as GND or VCC. Remove signal from equation set.

Note: a* /a=GND, a + /a=VCC

3600..... Illegal condition in the PDF file for device pin <integer value>
4000..... Maximum usage exceeded, <integer value>, of input signal
4004 Illegal ZHAL device
5000..... Message file cannot be accessed
5004..... No match for error code in message table

Note: The above list consists of user errors only. Other errors you might see are likely to be internal, for which you should contact your local FAE or the Monolithic Memories factory.

ERRORS REPORTED BY SIM

Error Number	Error Message
0.....	Unexpected end of input TRE or PAL file, file invalid
1.....	SETF/TRACE_ON/ CHECK/TRACE_OFF should have parameters
2.....	Illegal parameter

- 3.....Illegal parameter after expansion
- 4.....Cannot SETF a combinatorial output
- 5.....This output is SETF without preload and
TRI-STATE functions active
- 6.....Can not CLOCKF this pin
- 7.....The value of the clock pin is high and
CLOCKF is asserted
- 8.....TRACE_ON command should follow
TRACE_OFF command
- 10.....This is not a functional equation
- 11.....Undefined equation
- 12.....SET/RESET function asserted at the
same time PRELOAD function
- 13.....SYSTEM OSCILLATION
- 14.....This pin cannot be used as an input pin
- 15/16.....Cannot have simple/functional equation
for this pin
- 17.....Number of vectors exceed maximum
vectors (=512)
- 18.....No simulation commands present
- 19.....Cannot have input pins as arguments of
CHECK statements
- 20.....Output XX is SETF without tristate
function active
- 21.....Combinatorial output cannot be
preloaded
- 22.....Input pin cannot be preloaded. Use the
SETF command
- 23.....CLOCK pin cannot be preloaded. Use
the CLOCKF command
- 24.....PRLDF statement must have arguments
- 25.....All arguments to the PRLDF command
are invalid
- 26.....Simulator does not support this device.
- 27.....Cannot SETF a buried register node in
32VX10
- 28.....Cannot CLOCKF a clock pin which is
also used as input

- 29.....GLOBAL
Set/Reset both are high for above output
- 30.....The functional equation for this pin is using a registered equation (:=) instead of combinatorial
- 31.....Simple equation can not be defined for GLOBAL pin
- 32.....Can not SETF above pin
- 33.....Global pin can not be preloaded
- 34.....Set or Reset is always high
- 35.....Output pin cannot be preloaded.
Preload can only be done on the buried register nodes
- 36.....Set/Reset are both high for above output
- 37.....Buried register node value missing.
32VX10 preload requires all buried nodes to be specified
- 38.....Missing PALASM2.TRE, cannot simulate file. Please run your file through PALASM2 to generate PALASM2.TRE
- 39.....Missing .PDF file. Please copy the .PDF file into this directory and run again

**ERRORS
REPORTED BY
JEDMAN**

Error Number	Error Message
10.....	Input JEDEC file cannot be opened.
20.....	PDF file PALPDF.PDF cannot be opened.
30.....	Part name is incorrect.
40.....	Input file is not a valid JEDEC file.
50.....	Input file terminated prematurely.

- 60.....Error file cannot be created.
- 70.....Output JEDEC file cannot be created.
- 80.....Output PALASM2 file cannot be
created.
- 90.....Illegal data in fuse field.
- 95.....All the JEDEC fuse data must appear in
one block.
- 100.....No fuses specified in the input file.
- 105.....Fuse number has been defined twice.
- 110.....Product term SHARING is not allowed.
- 115.....Fuse number exceeds the fuse count
permitted.
- 120.....Data array fuse buffer size exceeded.
- 130.....The input file is not a valid 22V10
JEDEC file.
- 135.....Device name entered should be 22V10
for the option chosen.
- 145.....Non-recoverable errors detected.
Program terminated.
- 150.....Invalid field identifier.
- 160.....Field not valid in this context.
- 165.....Integer expected in the current field.
- 170.....Illegal specification of - <string> - field.
- 180.....Incorrect number of fuses in the QF
field.
- 10.....Input JEDEC file cannot be opened.
- 20.....PDF file PALPDF.PDF cannot be
opened.
- 30.....Part name is incorrect.
- 40.....Input file is not a valid JEDEC file.
- 50.....Input file terminated prematurely.
- 60.....Error file cannot be created.
- 70.....Output JEDEC file cannot be created.
- 80.....Output PALASM2 file cannot be
created.
- 90.....Illegal data in fuse field.
- 95.....All the JEDEC fuse data must appear in
one block.
- 100.....No fuses specified in the input file.

E

- 105.....Fuse number has been defined twice.
- 110.....Product term SHARING is not allowed in.
- 115.....Fuse number exceeds the fuse count permitted.
- 120.....Data array fuse buffer size exceeded.
- 130.....The input file is not a valid PAL22V10 JEDEC file.
- 135.....Device name entered should be PAL22V10 for the option chosen.
- 140.....Non-recoverable errors detected. Program terminated.
- 150.....Invalid field identifier .
- 160.....Field - <string> - not valid in this context.
- 165.....Integer expected in the current - <string> - field.
- 170.....Illegal specification of - <string> - field.
- 180.....Incorrect number of fuses in the QF field.
- 18.....Found - <integer value> - .
- 181.....Expected - <integer value> - .
- 190.....Incorrect number of pins in the QP field.
- 190.....Found - <integer value> - .
- 191.....Expected - <integer value> - .
- 210.....Incorrect fuse checksum in the input file.
- 210.....Found - <string> - .
- 211.....Expected - <string> - .
- 220.....Invalid data - <string> - in vector field.
- 225.....Simulation code generation not currently supported.
- 230.....Super-voltage will be replaced by "PRLDF".
- 240.....Incorrect number of logic values in vector.
- 240.....Found - <integer value> - entries.
- 241.....Expected - <integer value> - entries.
- 250.....Vector limit - <integer value> - exceeded.
- 260.....Incorrect transmission checksum.
- 260.....Found - <string> - .

- 261.....Expected - <string> -.
- 270.....Invalid transmission checksum in the
input file.270 350 PROASM
Warnings issued. Check the input
JEDEC file.
- 500.....PROASM Program successfully
completed. Check output files.
- 750.....PROASM Input file <string>, Warnings
<integer value>
- 760.....Errors <integer value>

E

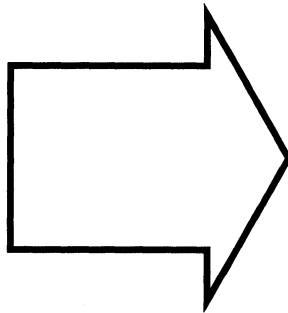


Table of Contents	vii
Software Errata	ERR
Introduction	1
Installing PALASM® 2 Software	2
PDS Syntax	3
Boolean Equation Design	3A
State Machine Design	3B
Simulation	3C
Using PALASM 2 Software	4
Installation and Operation Notes	A
Programmer Notes	B
Device Specific Syntax	C
PAL® Design File Library	D
Error Messages	E
Submitting a HAL® Design to MMI	F
PALASM 2 Syntax Diagram	G
Supplementary Software	H
JEDEC Standard No. 3	I
Release Notes	RN
Index	IX
PLEASM™ Manual	

APPENDIX F SUBMITTING A HAL DESIGN TO MONOLITHIC MEMORIES

In order for Monolithic Memories to produce your HAL or ProPAL devices, you need to provide Monolithic Memories Inside Sales with a Master Device, a PAL device Design Specification in PALASM or ABEL format, and Functional Vectors. Once these materials have been received, a Monolithic Memories Product Marketing Engineer will work with you to make sure that any problems are quickly resolved, and that production begins as soon as possible.

MASTER DEVICE

The master device is a programmed part you provide as a reference point for Monolithic Memories testing.

F

PAL DEVICE DESIGN SPECIFICATION

You can provide the PAL device Design Specification (Logic Equations) may be provided through the following means:

- 1: Floppy disk or magnetic tape in formats readable by the computers in Monolithic Memories (VAX, IBM-PC, or IBM-3083).
- 2: A printed list of the logic equations, verified by PALASM 2 software or ABEL. The listing should be dated and signed by its originator.

Upon receipt of your design submission Monolithic Memories will verify the PAL device Design Specification and generate ProPAL sample devices. The samples are sent to you for confirmation of the pattern. (This step can be waived if you provide logic equations on magnetic media, and stipulate that matching the submitted master device is sufficient confirmation.)

FUNCTIONAL TEST VECTORS

The functional vectors should consist of 20 to 40 vectors that represent the actual operation of the device. They are used to initiate Monolithic Memories' test vector generation (TGEN). The functional vectors should initialize the device to a known state within a specified number of clock cycles.

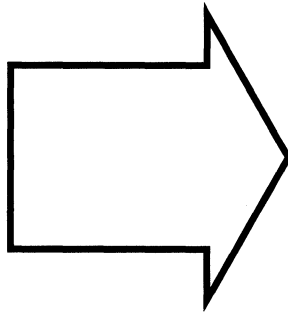
TGEN test vector generation is considered complete when fault grading reaches a minimum of 90%. If this is not achieved, you will be contacted for testability enhancements or a waiver.

Upon successful verification of the equations and vectors, Monolithic Memories will send the LOGIC DESIGN/SAMPLE APPROVAL form to you for sign off. You will at this time have the option of reviewing the final test vectors. If you elect this option, your approval of the TGEN-generated test pattern is confirmed when you sign the CUSTOMER TEST PATTERN APPROVAL form.

Note: Should the test pattern simulation by PALASM 2 software differ from the simulation by TGEN, the TGEN simulation will be used.

F

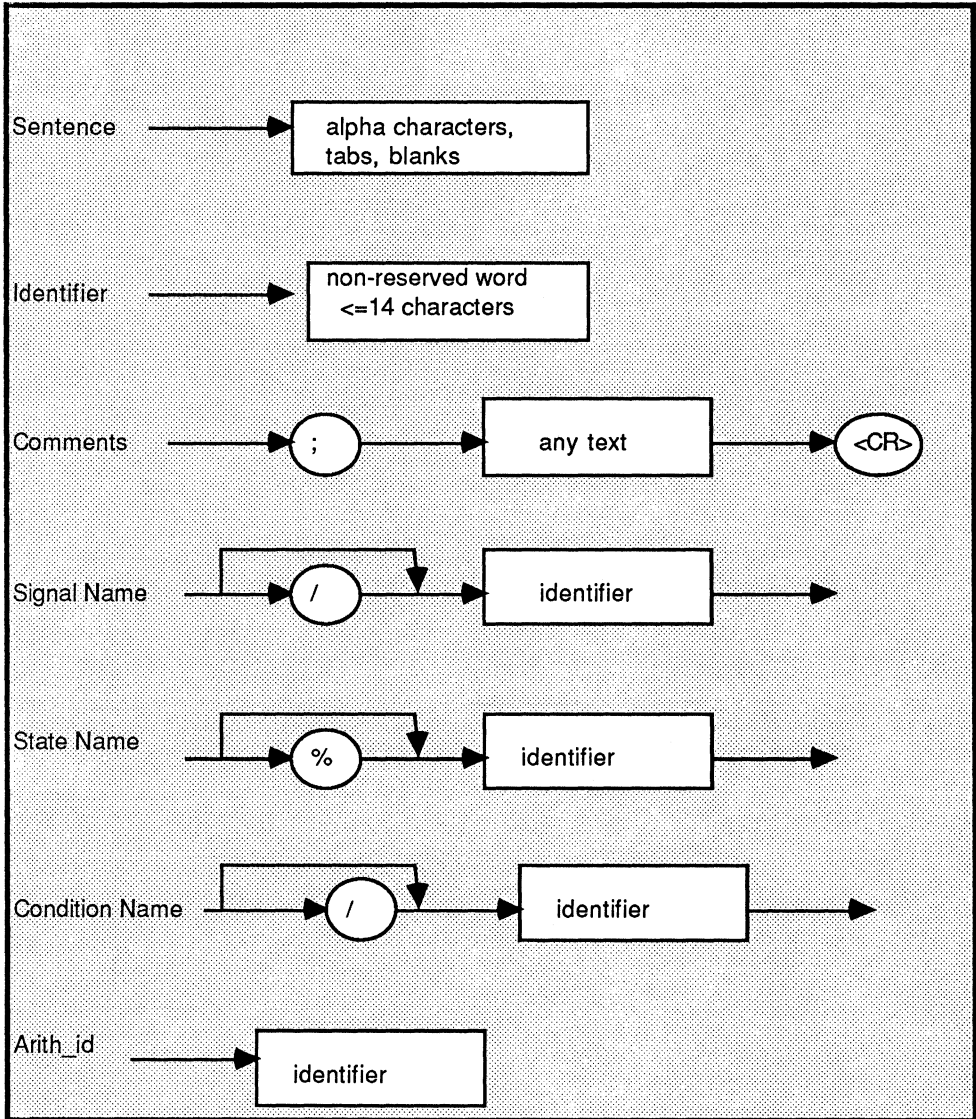
Table of Contents	vii
Software Errata	ERR
Introduction	1
Installing PALASM® 2 Software	2
PDS Syntax	3
Boolean Equation Design	3A
State Machine Design	3B
Simulation	3C
Using PALASM 2 Software	4
Installation and Operation Notes	A
Programmer Notes	B
Device Specific Syntax	C
PAL® Design File Library	D
Error Messages	E
Submitting a HAL® Design to MMI	F
PALASM 2 Syntax Diagram	G
Supplementary Software	H
JEDEC Standard No. 3	I
Release Notes	RN
Index	IX
PLEASM™ Manual	



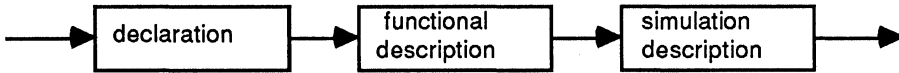
APPENDIX G PALASM 2 SOFTWARE SYNTAX DIAGRAMS

G

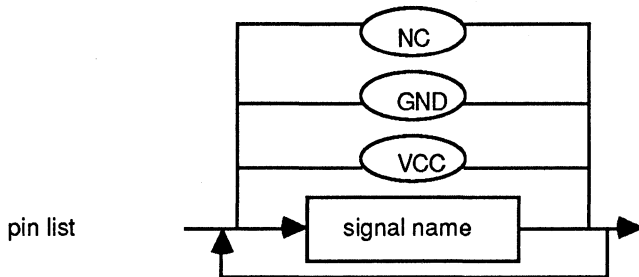
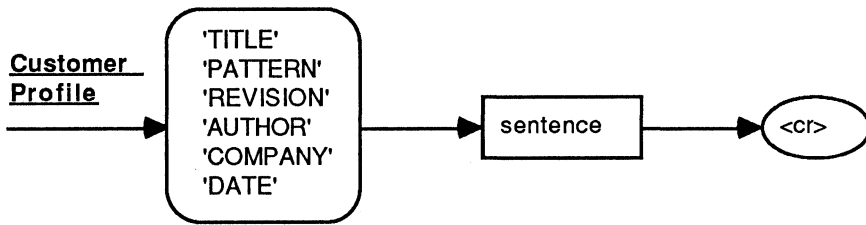
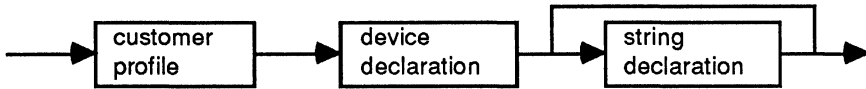
Definition of Terms



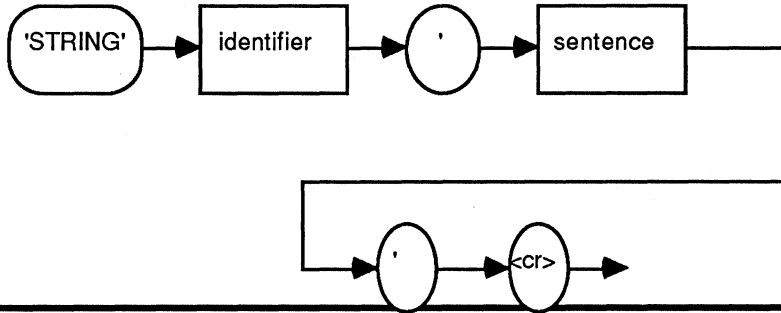
Syntax Diagram



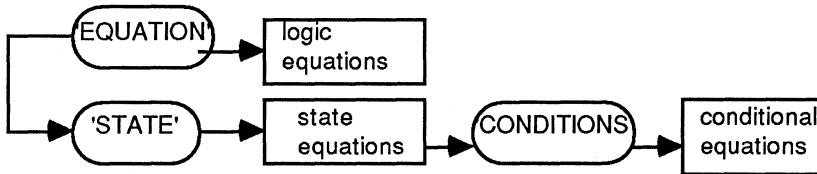
Declaration



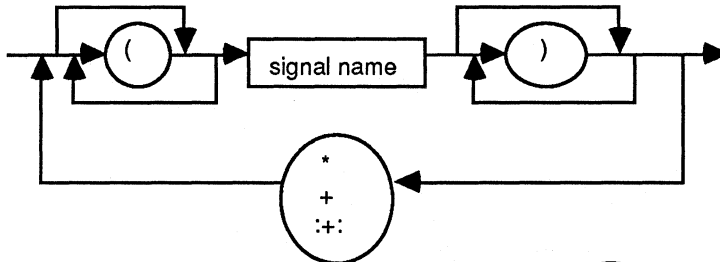
String Declaration



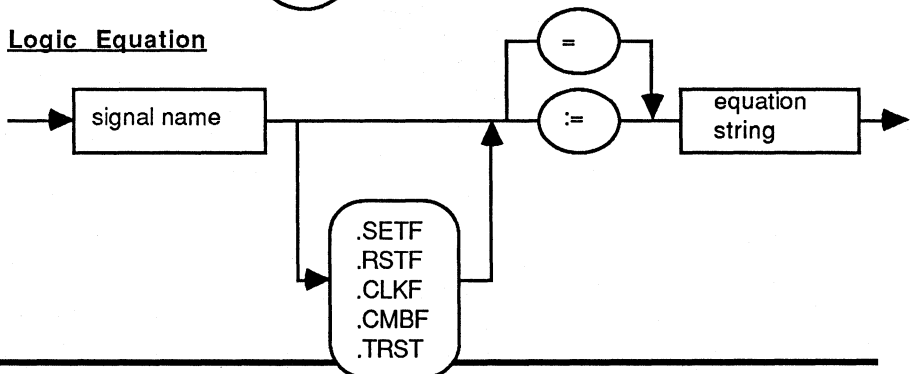
Functional Description



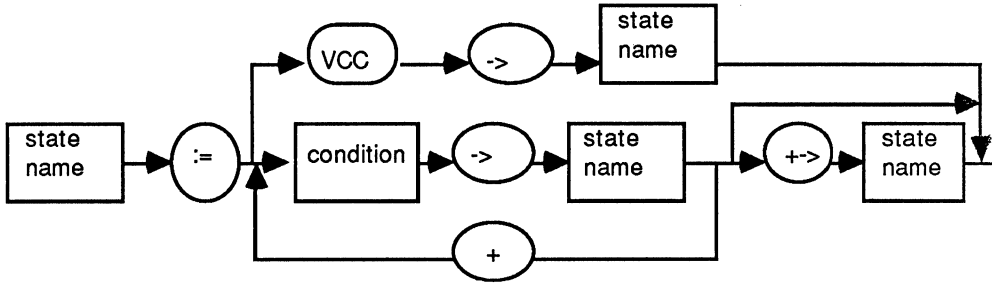
Equation String



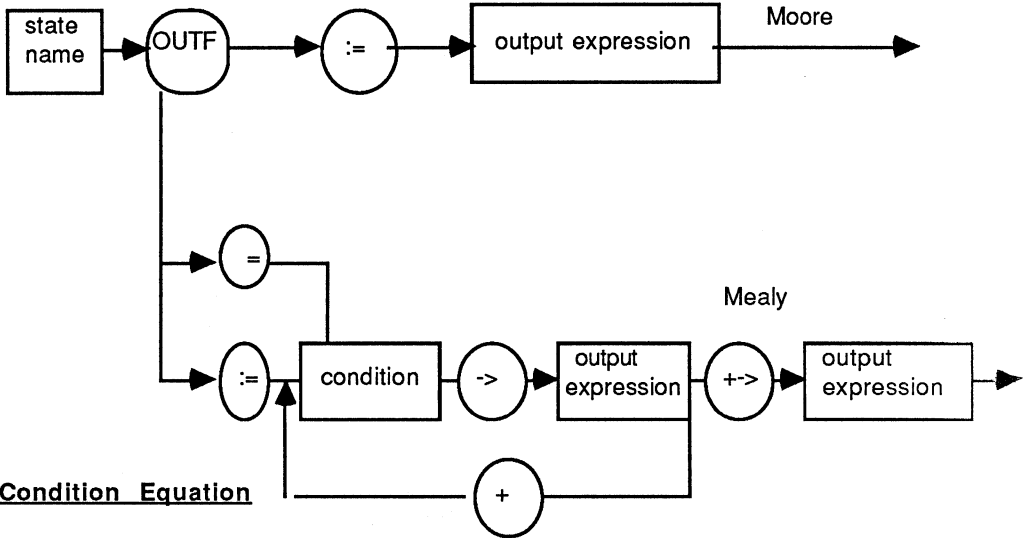
Logic Equation



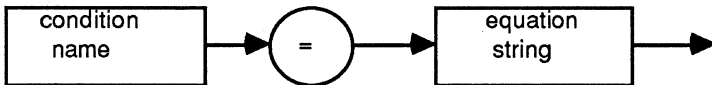
State/Output Equation



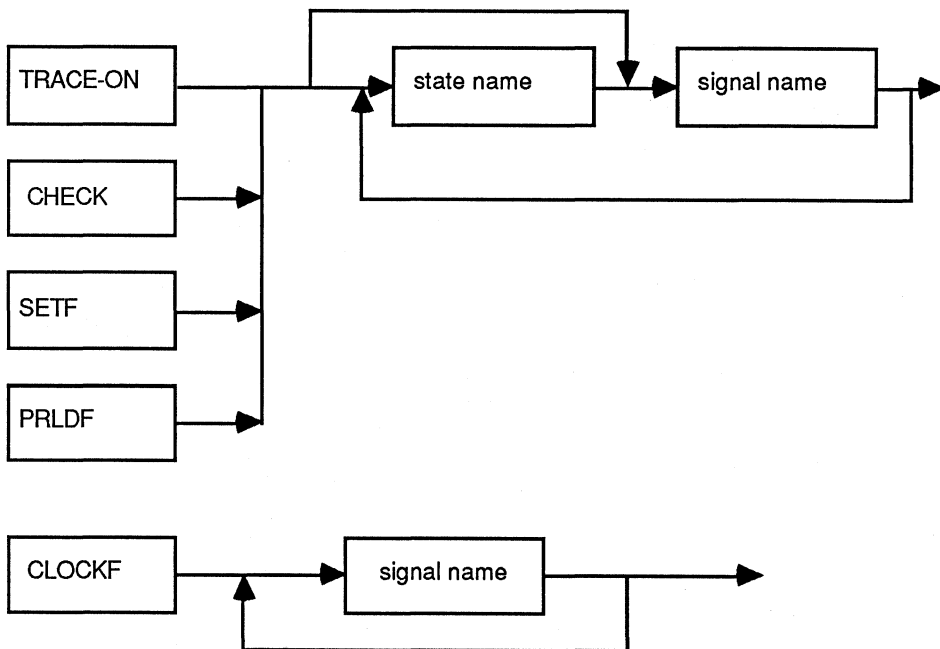
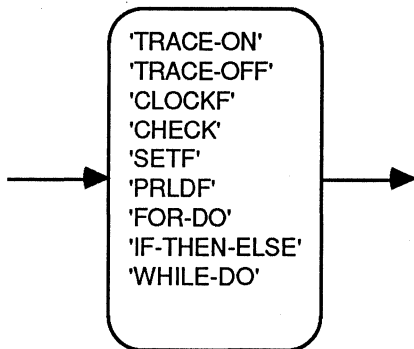
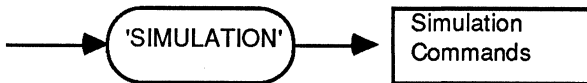
State Equation



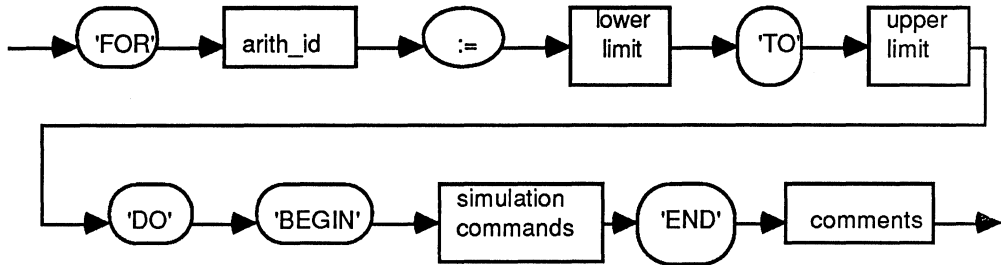
Condition Equation



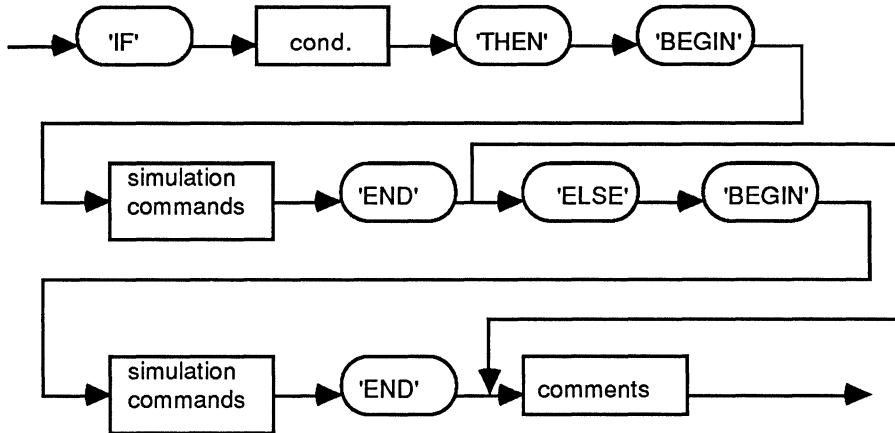
Simulation Description



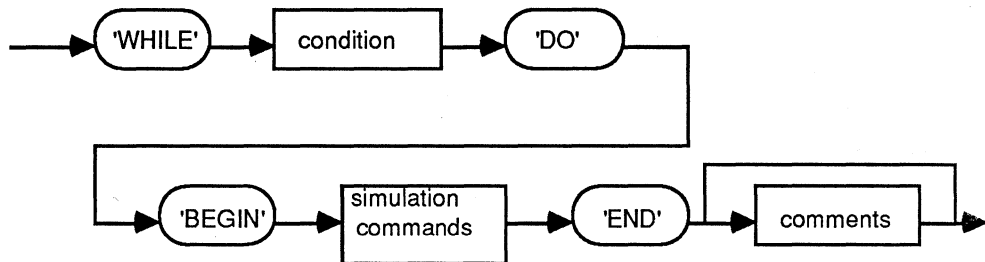
'FOR...DO'



'IF...THEN...ELSE'

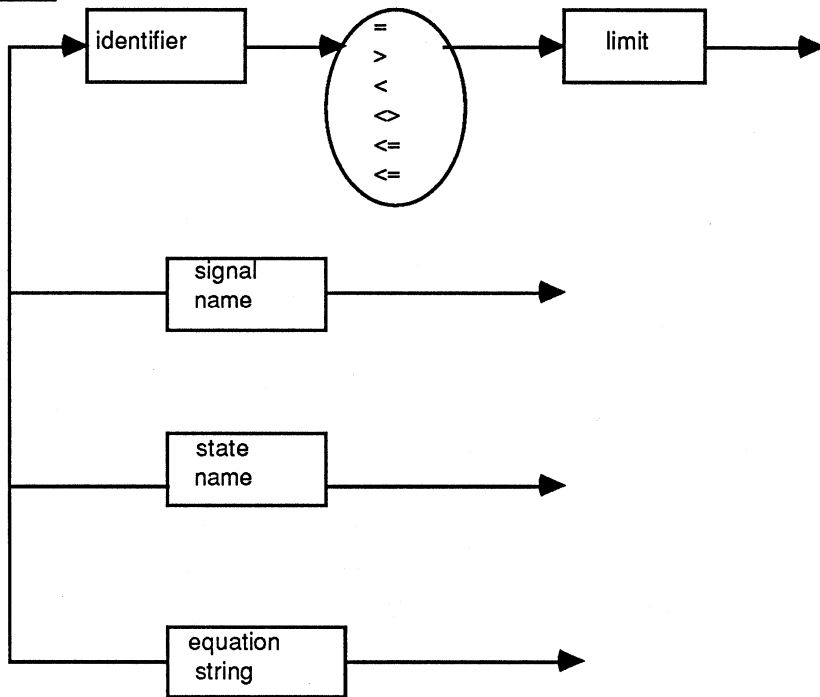


'WHILE...DO'



G

Condition



Lower Limit
Upper Limit

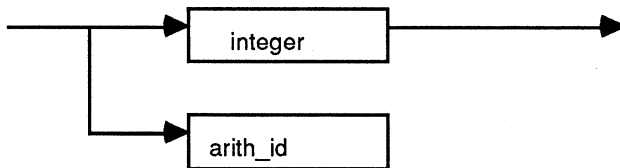
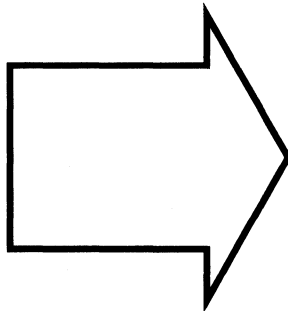


Table of Contents	vii
Software Errata	ERR
Introduction	1
Installing PALASM® 2 Software	2
PDS Syntax	3
Boolean Equation Design	3A
State Machine Design	3B
Simulation	3C
Using PALASM 2 Software	4
Installation and Operation Notes	A
Programmer Notes	B
Device Specific Syntax	C
PAL® Design File Library	D
Error Messages	E
Submitting a HAL® Design to MMI	F
PALASM 2 Syntax Diagram	G
Supplementary Software	H
JEDEC Standard No. 3	I
Release Notes	RN
Index	IX
PLEASM™ Manual	



APPENDIX H SUPPLEMENTARY SOFTWARE

The software programs described in this chapter reside on your Supplementary disk. Some of these programs are not supported by Monolithic Memories.

PALASM

This version of PALASM 2 software contains a new menu interface designed for ease of use. It is especially helpful for beginning users of PALASM 2 software. The program is on the Supplementary diskette and you need to install it before it is ready for use.

The program is called PALASM. For both the installation procedure and details on how to operate the new Menu program, turn to Chapter 2: Installing PALASM 2 Software.

Note: The old Menu program that was available in version 2.21 and all earlier versions of PALASM 2 software has been replaced by the new enhanced Menu program. The new program is available on PALASM 2.22 and all later versions.

PDSCNVT

PDSCNVT converts PALASM 1 software designs into PALASM 2 software syntax.

H

To call up the program, type

PDSCNVT

Next, supply the desired file name for conversion. The extension .PDS has been reserved for the output file; hence some other extension should be used for the input file: .PAL is suggested.

PDSCNVT will successfully handle most syntax differences, but not items 2 and 3 listed in this section. Manual correction is the only solution. You should also correct information for the customer profile section.

Conversion of function tables is also not fully automated. PDSCNVT does not know any details of PAL device architecture, such as which pins are inputs or outputs; hence it does not produce the correct SETF and CHECKF statements. If you want to use simulation, you must manually uncomment (remove semicolons from) those lines of the simulation section in the output file that contain executable commands. You might begin by removing the semicolon before the word SIMULATION. Also, you must edit your file to remove all the output pins from the SETF statements. It is a good idea to add CHECK statements in your design file, wherever it is appropriate.

DIFFERENCES BETWEEN PALASM 1 AND PALASM 2 SOFTWARE SYNTAX

PALASM 2 software language syntax is not a true superset of the original PALASM software syntax. In several areas, logically correct designs will have to be manually edited to update them to the new syntax. This job, while not difficult, presents several issues:

1. PALASM 1 accepts 8 character names; PALASM 2 software accepts 14 .
2. Characters in the PAL device type beyond the output designator (eg: PAL16R8A) were ignored by PALASM 1 software. They are significant and must be removed before processing PALASM 2 software design files.
3. Purely numeric (1,2,5) pin identifiers were accepted in PALASM 1, but are not legal in PALASM 2 software. A letter (A-Z) is required as part of any pin name.
4. The IF (condition) was used to enable 3-state output controls on PALASM 1 software combinational equations. PALASM 2 software uses signal.TRST syntax for this condition.
5. FUNCTION TABLE and DESCRIPTION were keywords in PALASM 1 and marked separate sections of the design file. PALASM 2 software uses other key words (CHIP, SIMULATION) to parse the design file. Any of these four terms should be avoided as user pin names.
6. PALASM 1 software used a fixed truth-table format for describing inputs and outputs of chips during testing. PALASM 2 software has a more powerful structured syntax for describing how to do the testing. Function tables are not currently accepted by PALASM 2 software.

PC2

Downloading of files to PAL device programmers can be accomplished by a variety of methods on the IBM-PC. The simplest way is to use a copy command with the output device set to the port in use. Another method is to use a bidirectional communications program such as PC2. PC2 replaces its predecessors PALCOMM and PALSETUP. It offers

H

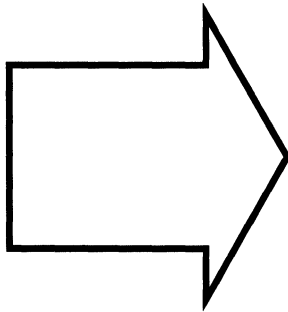
the combined features of both programs along with a few enhancements. The program is function key driven as follows:

- F1: Send file
- F2: PALSETUP mode
- F3: VIEW switch toggle
- F4: CAPT switch toggle
- F9: New File Name
- F10: Exit

The two switches (F3 and F4) enable 2 special modes of operation. VIEW allows you to see the JEDEC file as it is downloaded to the programmer. CAPT saves any data sent over the communications line to a disk file. It is especially useful to save dumps of device fuse plots produced by the programmer for later printing, or comparison.

When you press F2 the program begins the PALSETUP dialog to configure the communications port. The current default data is displayed. If no defaults are present on the disk, the program creates a default parameter file (PC2.DAT) and allows you to alter them. On the screen, the current parameters (COMM port, baud rate, stop bits, etc.) are highlighted, and you can select an input value by pressing any key on the keyboard. Press the value (stop bits) has been selected, you are asked to confirm the information. If confirmed, the data is saved in PC2.DAT. PC2 allows you to change file names for downloading at any time during the session, and to download as many files as desired. If no name is available for the downloading, a prompt is provided to ask you for the name. Checks insure that the file exists where indicated and is accessible.

Table of Contents	vii
Software Errata	ERR
Introduction	1
Installing PALASM®2 Software	2
PDS Syntax	3
Boolean Equation Design	3A
State Machine Design	3B
Simulation	3C
Using PALASM 2 Software	4
Installation and Operation Notes	A
Programmer Notes	B
Device Specific Syntax	C
PAL® Design File Library	D
Error Messages	E
Submitting a HAL® Design to MMI	F
PALASM 2 Syntax Diagram	G
Supplementary Software	H
JEDEC Standard No. 3	I
Release Notes	RN
Index	IX
PLEASM™ Manual	



APPENDIX I

JEDEC STANDARD

NO. 3A

What follows is the JEDEC (Joint Electronic Devices Engineering Council) standard format for transfer of data between data preparation systems and programmable logic device programmers (from JEDEC Council Ballot JCB-82-2, formulated under the cognizance of JC-42.1 Committee on Bipolar Memory Standardization).

PURPOSE

The purpose of this Standard is to define a data transmission format for transferring information between a data preparation system and a device programmer. The information to be transferred is divided into five categories:

1. The design specification identifier.
2. The device to be programmed.
3. Fuse links that must be blown to implement the design specification.
4. Information to perform a structured functional test.
5. Other information.

This Standard is intended to be applicable to all programmable logic devices.

FORMAT DEFINITIONS

The Design Specification Identifier

The design specification identifier will consist of:

1. An ASCII (STX) (02 Hex).
2. User's name and company.
3. The design specification's date, part number and revision.
4. The manufacturer's part number of the programmable device.
5. Other information.
6. An asterisk.

An (STX) begins the transmission and is followed by ASCII characters representing the information defined above. An * terminates the identification information.

Example:

```
( STX ) J. ENGINEER LOGIC CO. 2-2-80  
D.S. 018-1563A
```

```
/ /
```

```
P.N. LOG UAL
```

```
20L10* Design  
/
```

/Spec. /

Part Number

/

Part No.

Revision

The Device to be Programmed

An optional field is defined to specify the device being programmed. The device can be specified by an ASCII code. The code is preceded by a D and terminated by an asterisk (*).

Example:

(STX) J. ENGINEER LOGIC CO. 2-2-80
D.S.

018-1563A

P.N. LOG

UAL20L10*D7503*

Device Code

Fuse Information

Each device fuse link will be assigned a decimal number. These numbers can be shown on the logic diagram of the device's data sheet. Each numbered fuse will have two possible states. A zero will specify a low-resistance link and a one will specify a high-resistance link.

Fuse information can be presented in three fields. The first "F" field is optional. This field can be used to define the default state for links that are not defined in the second "L" field. If the "F" field is not used, all device links must be defined in the "L" field. The third, optional "C" field, defines a checksum.

The default state (0 or 1) for all links not defined in any "L" field is preceded by an "F." This chosen state is followed by an asterisk (*).

Example:

```
( STX ) J. ENGINEER LOGIC CO. 2-2-80  
D.S. 018-
```

```
1563A
```

```
Font = F7>
```

```
P.N. LOG
```

```
UAL20L10*D7503*
```

```
FO*
```

```
Default
```

```
State
```

The specific link information is preceded by an "L." The "L" is followed immediately by a variable length decimal number that indicates the starting fuse number of a string of data. The first 1 or 0 is preceded by a space, and the data string is terminated by an *. Each data string can be any convenient length, and any number of data strings can be specified. If the state of a link is specified 2 or more times, the last state replaces preceding entries.

Example:

(STX) J. ENGINEER LOGIC CO. 2-2-80
D.S.

018-1536A

P.N. LOG UAL

20L10

*7503*FO*LO 01010101*

L8 01010111*L1000

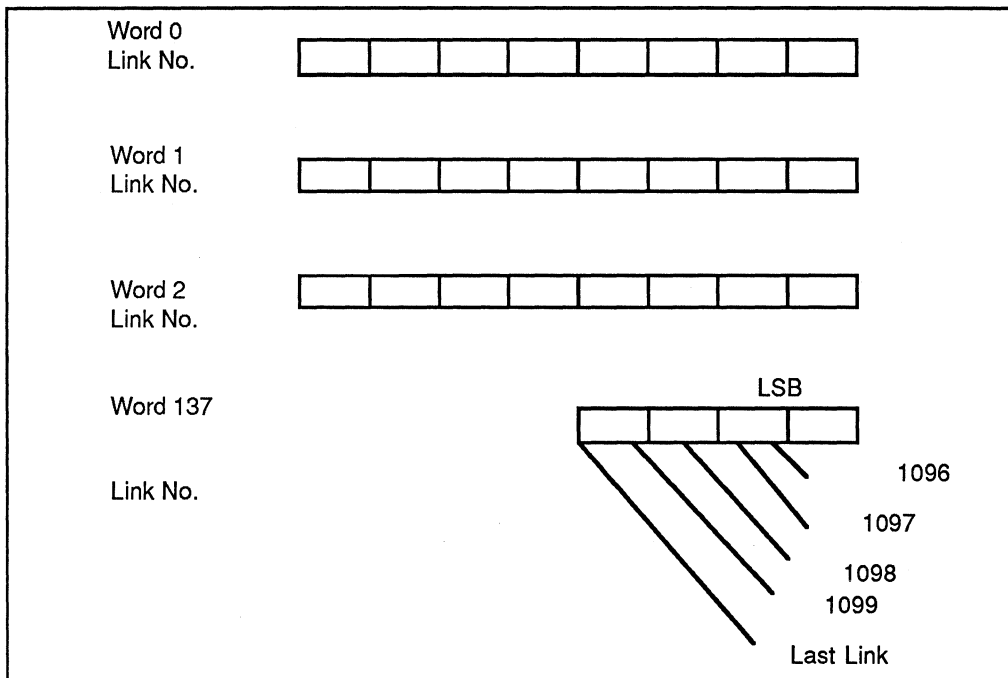
0101*

Link Information

For this example, fuses 16 through 999 will be assigned state 0, and fuses 1004 through 1639 will be assigned state 0, due to the "F" field default state. When the "F" field is used, the receiving equipment must know the total number of fuse links.

An optional field "C" is reserved for the link information checksum. This link information checksum is computed by performing a 16 bit addition of 8 bit words constructed from the specified state of each link in the device. The 8 bit words are defined as shown in the following diagram.





**Figure I-1:
8-Bit Word Definition**

The word encompassing the last link is constructed by setting zeros for all bit locations more significant than the last link. The 16-bit sum is expressed as 4 ASCII HEX characters. A "C" precedes the four hex characters. The last character is followed by an "".

Example:

(STX) J. ENGINEER LOGIC CO. 2-2-80
D.S.

018-1563A

P.N. LOG UAL

20L10*D7503*FO*LO 01010101*

L8 01010111*L1000

0101*CF38A*

Checksum

Structured Functional Test Information

An optional field is defined to specify one of several test conditions for each pin of a device. A test vector is defined as N test conditions, where N is the pin count of the device.

The ASCII variables representing test conditions are:

- 0 - Drive input low
- 1 - Drive input high
- 2 - Drive input to supervoltage #2
- 3-9 - Drive input to supervoltage #3-7
- C - Drive input low, high, low
- K - Drive input high, low, high
- N - Power pins and outputs not tested
- L - Test output low
- H - Test output high
- Z - Test input or output for high impedance

F - Float input or output

The C and K driving signals are presented after other inputs are stable. The L, H, Z tests are performed after all inputs have stabilized, including C and K.

A test vector must be preceded by a V. The V is followed immediately by a variable length decimal number denoting the test vector number. This number is followed by a space, then the test variables. The first variable in the vector is for pin 1 and the last for pin N. The vectors will be applied to the device under test in numeric order. If any vectors are specified 2 or more times, the data in the last vector replaces previous data.

Example:

```
( STX )   J. ENGINEER LOGIC CO. 2-2-80
D.S.
```

```
018-1563A
```

```
P.N.     LOG   UAL
```

```
20L10#D7503*FO*LO 01010101*
```

```
L8 01010111*L1000
```

```
0101*CF38A*
```

```
V1
```

```
C1011101010NHLHHHLLLLLN*V4
```

```
C1111011011N
```

```
HHHHLHHHLHLN*
```

Structured Test Vectors

The test vector V field can be preceded by an optional P field. The P field is used to specify the correspondence of test vector variables to the device pin numbers. A "P" followed by a space, then each pin number separated by a space are listed in the desired order. The "P" field is terminated by an asterisk (*).

Example:

```
P 13 14 15 16 17 18 19 20 21 22 23 24  
12 11 10 9 8 7 6 5 4 3 2 1*
```

The first variable in a test vector V will be presented to the first pin listed in the P field. The second variable will go to the second pin listed, etc.

If the P field is not transmitted prior to the first V field, the standard definition is assumed, where pin 1 corresponds to the first variable and pin N the last variable in a test vector.

Other Information

Additional optional fields may be defined using the letters G, S, R, M, Q and T. Each field must begin with one of the above letters and be terminated with an asterisk (*). No other restrictions are applied. Therefore, multiple letters could be used to specify any number of optional fields.



Ending the Format

Example:

(STX) J. ENGINEER LOGIC CO. 2-2-
80 D.S.

018-1563A

P.N. LOG UAL

20L10*D7503*FO*LO 01010101*

L8 01010111*L1000

0101*CF38A*

V1

C1011101010NHLHHHLLLLLN*V4

C1111011011N

HHHHLHHHLHLN* (ETX) AB15

Sum-check

OTHER RULES

Carriage Returns and Line Feeds

Carriage return characters and line feed characters can be placed anywhere in the format to obtain maximum visibility.

Transportability

All receiving machines should have a "Kernel" mode to ignore all optional fields; then the actual programming data will be transportable. For example, as allowed in the format, optional fields can be sent to specify additional checksums. A receiving machine in the "kernel" mode could ignore this information yet receive the link information required to program the device. If the optional "F" field is used to avoid transmitting link data, transportability could be lost. Therefore, whenever practical, data should be transmitted for all links of the device.

Legal Characters

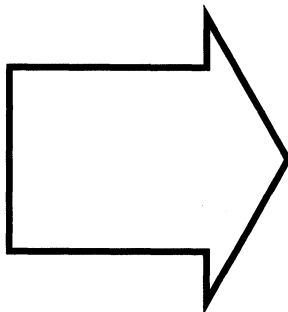
Only the following ASCII characters are legal. Any other characters present in the file may result in invalid operation. Receiving machines should be designed to ignore illegal characters and transmitting machines should avoid sending illegal characters.

STX	02	HEX	START OF TEXT
ETX	03	HEX	END OF TEXT
LF	0A	HEX	LINE FEED
CR	0D	HEX	CARRIAGE RETURN

All printable 20 Hex through 7E Hex inclusive characters.



Table of Contents	vii
Software Errata	ERR
Introduction	1
Installing PALASM® 2 Software	2
PDS Syntax	3
Boolean Equation Design	3A
State Machine Design	3B
Simulation	3C
Using PALASM 2 Software	4
Installation and Operation Notes	A
Programmer Notes	B
Device Specific Syntax	C
PAL® Design File Library	D
Error Messages	E
Submitting a HAL® Design to MMI	F
PALASM 2 Syntax Diagram	G
Supplementary Software	H
JEDEC Standard No. 3	I
Release Notes	RN
Index	IX
PLEASM™ Manual	





RELEASE NOTES PALASM 2.22

IN THIS CHAPTER

- * PALASM 2.22: Summary of Enhancements
- * PALASM 2.22: Disk and Tape Layout
- * PALASM 2.22: List of Fixed Bugs

RN

SUMMARY OF ENHANCEMENTS

Following is a summary of enhancements in the current release: **PALASM 2.22**.

Release Date: March 2, 1987

- * Complete software support for PROSE
- * State Machine design entry
- * Logic Minimization
- * JEDEC disassembly
- * New easy-to-use menu interface
- * Supported on the VAX-UNIX operating system
- * Supported on DAISY workstations
- * Available on high density disks

PALASM 2.22 DISK AND TAPE LAYOUT

The disk and tape layout outlined in the following page can be used to check whether you have all the necessary files to successfully run the 2.22 version of PALASM 2 software. Below is the disk layout. If you need the tape layout, turn to RN-8.

Disk Layout

Your package should contain the following IBM disks.

- Disk #1 of 6: PALASM2.EXE, SIM.EXE.
- Disk #2 of 6: XPLOT.EXE, JEDMAN.EXE.
- Disk #3 of 6: MINIMIZE.EXE.
- Disk #4 of 6: PROASM.EXE, PROSIM.EXE.
- Disk #5 of 6: Supplementary Software.
- Disk #6 of 6: PALASM 2 Examples

To get a listing of the files on each of your disks, insert the disk into one of the drives in your IBM-PC/AT/XT and type

DIR

Each disk should contain the following files:

RN

Disk #1 of 6:
PALASM2.EXE SIM.EXE

(C)-COPR.TXT	README
P10H8.PDF	P10L8.PDF
P12H6.PDF	P12L10.PDF
P12L6.PDF	P14H4.PDF
P14L4.PDF	P14L8.PDF
P16C1.PDF	P16H2.PDF
P16L2.PDF	P16L6.PDF
P16L8.PDF	P16P8.PDF
P16R4.PDF	P16R6.PDF
P16RA8.PDF	P16RP4.PDF
P16RP6.PDF	P16RP8.PDF
P18L4.PDF	P20C1.PDF
P20L10.PDF	P20L2.PDF
P20L2.PDF	P20L8.PDF
P10H20G8.PDF	P10H20P8.PDF
P20R4.PDF	P20RA10.PDF
P20RS10.PDF	P20X10.PDF
P20X4.PDF	P20X8.PDF
P22RX8.PDF	P32R16.PDF
P32VX10.PDF	P64R32.PDF
P6L16.PDF	P8L14.PDF
P22V10.PDF	P16R8.PDF
PALASM2.EXE	SIM.EXE

50 Files

Disk #2 of 6: XPLOT.EXE
JEDMAN.EXE

(C)-COPR.TXT	README
P10H8.PDF	P10L8.PDF
P12H6.PDF	P12L10.PDF
P12L6.PDF	P14H4.PDF

P14L4.PDF	P14L8.PDF
P16C1.PDF	P16H2.PDF
P16L2.PDF	P16L6.PDF
P16L8.PDF	P16P8.PDF
P16R4.PDF	P16R6.PDF
P16RA8.PDF	P16RP4.PDF
P16RP6.PDF	P16RP8.PDF
P18L4.PDF	P20C1.PDF
P20L10.PDF	P20L2.PDF
P20L2.PDF	P20L8.PDF
P10H20G8.PDF	P10H20P8.PDF
P20R4.PDF	P20RA10.PDF
P20RS10.PDF	P20X10.PDF
P20X4.PDF	P20X8.PDF
P22RX8.PDF	P32R16.PDF
P32VX10.PDF	P64R32.PDF
P6L16.PDF	P8L14.PDF
P22V10.PDF	P16R8.PDF
XPLOT.EXE	PALPDF.PDF
JEDMAN.MSG	JEDMAN.EXE
JEDMAN.DOC	

53 Files

RN

Disk #3 of 6:
PROASM.EXE
PROSIM.EXE

(C)-COPR.TXT	README
PROASM.EXE	PROSIM.EXE
PMS14R21.PDF	PROSE.MSG

6 Files

Disk #4 of 6:
MINIMIZE.EXE

(C)-COPR.TXT	README
PALPDF.PDF	MINIMIZE.EXE
MINIMIZE.MSG	

5 Files

Disk #5 of 6:
SUPPLEMENTARY
SOFTWARE

README.1ST	PDSCNVT.EXE
PALASM.COM	P2_1.SCR
VTRACE.COM	SETUP.BAT
PRINTER.SET	FXASCII.EXE
TRACE.DOC	TRACE.BAT
PC2.EXE	PC2.DAT
BINHEX.COM	TIMING.DOC
TIMING.COM	RIPPLE.SIG
PINOUT.DOC	PINOUT.COM
SCRSIM.COM	CONFIG.SYS
DECODE.DOC	DECODE.COM
P2_2.SCR	P2_3.SCR
P2_4.SCR	DPATH.COM
DUMMY.PDS	AUTOEXEC.BAT
AUTOFLOP.BAT	FLOPPY2.BAT
MENU.SYS	PAL2INST.COM

31 Files

Disk #6 of 6:
PALASM 2 Examples

Refer to Appendix D for the filenames of the PALASM 2 software examples on this disk.

RN

Tape Layout

The PALASM 2 software is available on magnetic tape media for the following computers and operating systems:

- * VAX-VMS
- * VAX-UNIX

For a complete listing of the files on the VAX-UNIX tape, turn to RN-20.

Below are the three kinds of VAX-VMS tapes available:

- * Source
- * Executable
- * ASCII

The lists of files on each of the three tapes are given in the following pages:

VAX-VMS Source Files

Directory MSAO: []

INSTALL.COM	README.DOC
PAL2HLP.FAC	PAL2ASS.COM
PAL2.COM	LNKCVT.COM
LNKFEP.COM	LNKJED.COM
LNKPRA.COM	LNKPRS.COM
LNKSIM.COM	LNKXPT.COM
LNKZHL.COM	CONVINC.ENV
CSOPENINC.ENV	CTRE.ENV
CTXTIO.ENV	EXTINC.ENV

RELEASE NOTES

FNXINC.ENV	JEDFNX.ENV
JEDGLOBS.ENV	JEDINC.ENV
PAL2INC.ENV	PDFONLY.ENV
PSFUSINC.ENV	PSINC.ENV
PSPDFINC.ENV	PSXPTINC.ENV
PXSIMINC.ENV	SIMINC.ENV
XPLOTINC.ENV	Z24.ENV
ANDAR.OBJ	CONVINC.OBJ
CSOPEN.OBJ	CSOPENINC.OBJ
CTRE.OBJ	CTRESUB.OBJ
CTXTIO.OBJ	CTXTSUB.OBJ
EXTINC.OBJ	EXTPDF.OBJ
FNXINC.OBJ	INIT.OBJ
JEDCONV.OBJ	JEDDISASM.OBJ
JEDFNX.OBJ	JEDFUNC.OBJ
JEDGLOBS.OBJ	JEDINC.OBJ
JEDMAN.OBJ	JEDPARSE.OBJ
JEDWRITE.OBJ	PAL2INC.OBJ
PALASM2.OBJ	PARSE1.OBJ
PARSE2.OBJ	PARSE3.OBJ
PARSE4.OBJ	PARSE5.OBJ
PDFONLY.OBJ	PDSCNVT.OBJ
PSASM.OBJ	PSASSIGN.OBJ
PSBUILD.OBJ	PSCHECK.OBJ
PSENABLE.OBJ	PSFUSE.OBJ
PSFUSINC.OBJ	PSINC.OBJ
PSINIT.OBJ	PSJEDEC.OBJ
PSMAIN.OBJ	PSOUTPUT.OBJ
PSPDF.OBJ	PSPDFINC.OBJ
PSSUMMARY.OBJ	PSWGT.OBJ
PSXPLOT.OBJ	PSXPTINC.OBJ
PXANAL.OBJ	PXARCH.OBJ
PXCONSTR.OBJ	PXCYCLE.OBJ
PXDEBUG.OBJ	PXERROR.OBJ
PXEXEC.OBJ	PXINPUT.OBJ
PXMAIN.OBJ	PXMEMORY.OBJ
PXOUTPUT.OBJ	PXPDF.OBJ
PXSIM.OBJ	PXSIMINC.OBJ
PXSIMULA.OBJ	PXSRTSTP.OBJ

RN

RELEASE NOTES

PXSUPPORT.OBJ	PXSYNTH.OBJ
PXTRE.OBJ	PXTXTIO.OBJ
PXVECTOR.OBJ	SANDAR.OBJ
SCAN.OBJ	SCHD.OBJ
SCMD.OBJ	SDB.OBJ
SEXT.OBJ	SFEQ.OBJ
SFNODE.OBJ	SFUNC.OBJ
SHIST1.OBJ	SIM.OBJ
SIM1.OBJ	SIMINC.OBJ
SINIT.OBJ	SINT.OBJ
SMISC.OBJ	SPRIM.OBJ
SPRLD.OBJ	SRX.OBJ
STR.OBJ	STRING.OBJ
SVX.OBJ	TSTDUMP.OBJ
XDB.OBJ	XFEQ.OBJ
XFNODE.OBJ	XM1.OBJ
XM2.OBJ	XM3.OBJ
XM4.OBJ	XM5.OBJ
XPLOT.OBJ	XPLOTINC.OBJ
Z24.OBJ	Z24MAIN.OBJ
ZCOMPRESS.OBJ	ZDAA.OBJ
ZERROR.OBJ	ZFORCE.OBJ
ZGLBINIT.OBJ	ZHAL.OBJ
ZINPUTLOG.OBJ	ZIOPRO.OBJ
ZMBC.OBJ	ZOUTPUTLOG.OBJ
ZPDFPROC.OBJ	ZPROPHYSICAL.OBJ
ZREADPRO.OBJ	ZTRAVERSE.OBJ
ZWRITEPRO.OBJ	P10H20G8.PDF
P10H20P8.PDF	P10H8.PDF
P10L8.PDF	P12H6.PDF
P12L10.PDF	P12L6.PDF
P14H4.PDF	P14L4.PDF
P14L8.PDF	P16C1.PDF
P16H2.PDF	P16L2.PDF
P16L6.PDF	P16L8.PDF
P16P8.PDF	P16R4.PDF
P16R6.PDF	P16R8.PDF
P16RA8.PDF	P16RP4.PDF
P16RP6.PDF	P16RP8.PDF

P18L4.PDF	P18P8.PDF
P20C1.PDF	P20L10.PDF
P20L2.PDF	P20L8.PDF
P20R4.PDF	P20R6.PDF
P20R8.PDF	P20RA10.PDF
P20RS10.PDF	P20RS4.PDF
P20RS8.PDF	P20S10.PDF
P20X10.PDF	P20X4.PDF
P20X8.PDF	P22RX8.PDF
P22V10.PDF	P32R16.PDF
P32VX10.PDF	P64R32.PDF
P6L16.PDF	P8L14.PDF
PALPDF.PDF	PMS14R21.PDF
Z20SRC.PDF	Z20TAB.PDF2
Z24OBJ.PDF	Z24SRC.PDF
Z24TAB.PDF	ZHAL20.PDF
ZHAL24.PDF	JEDMAN.MSG
PROSE.MSG	ZHAL24.MSG
PAL2EX.LIS	PAL2EX.COM
10BITREG.PDS	10COUNT.PDS
20RA10.PDS	3TO8DMUX.PDS
416DEC.PDS	4CNT.PDS
8COUNT.PDS	9BITCNT.PDS
9BITREG.PDS	ADREG16.PDS
ARBITER.PDS	BARREL.PDS
CONTROL.PDS	CRT.PDS
DCOUNT.PDS	FLIPFLOP.PDS
LATCH.PDS	LINK.PDS
MEMIO.PDS	MEMORY.PDS
OCTCOMP.PDS	P7000.PDS
P7004.PDS	PORT.PDS
TRAFFIC.PDS	UPCOUNT.PDS
VXPASCAL.COM	BLDCVT.COM
BLDFEP.COM	BLDJED.COM
BLDPRA.COM	BLDPRS.COM
BLDSIM.COM	BLDSUB.COM
BLDXPT.COM	BLDZHL.COM
CONVINC.INC	CSOPENINC.INC
CTRE.INC	CTXTIO.INC

RN

EXTINC.INC	FNXINC.INC
JEDFNX.INC	JEDGLOBS.INC
JEDINC.INC	PAL2INC.INC
PDFONLY.INC	PSFUSINC.INC
PSGLOBAL.INC	PSINC.INC
PSPDFINC.INC	PSXPTINC.INC
PXGLOBAL.INC	PXSIMINC.INC
SIMINC.INC	XPLOTINC.INC
Z24.INC	Z24GLOBAL.INC
ANDAR.VMS	CSOPEN.VMS
CTRESUB.VMS	CTXTSUB.VMS
EXTPDF.VMS	INIT.VMS
JEDCONV.VMS	JEDDISASM.VMS
JEDFUNC.VMS	JEDMAN.VMS
JEDPARSE.VMS	JEDWRITE.VMS
PALASM2.VMS	PARSE1.VMS
PARSE2.VMS	PARSE3.VMS
PARSE4.VMS	PARSE5.VMS
PDSCNVT.VMS	PSASM.VMS
PSASSIGN.VMS	PSBUILD.VMS
PSCHECK.VMS	PSENABLE.VMS
PSFUSE.VMS	PSINIT.VMS
PSJEDEC.VMS	PSMAIN.VMS
PSOUTPUT.VMS	PSPDF.VMS
PSSUMMARY.VMS	PSWGT.VMS
PSXPLOT.VMS	PXANAL.VMS
PXARCH.VMS	PXCONSTR.VMS
PXCYCLE.VMS	PXDEBUG.VMS
PXERROR.VMS	PXEXEC.VMS
PXINPUT.VMS	PXMAIN.VMS
PXMEMORY.VMS	PXOUTPUT.VMS
PXPDF.VMS	PXSIM.VMS
PXSIMULA.VMS	PXSRTSTP.VMS
PXSUPPORT.VMS	PXSYNTH.VMS
PXTRE.VMS	PXTXTIO.VMS
PXVECTOR.VMS	SANDAR.VMS
SCAN.VMS	SCHD.VMS
SCMD.VMS	SDB.VMS
SEXT.VMS	SFEQ.VMS

SFNODE.VMS	SFUNC.VMS
SHIST1.VMS	SIM.VMS
SIM1.VMS	SINIT.VMS
SINT.VMS	SMISC.VMS
SPRIM.VMS	SPRLD.VMS
SRX.VMS	STR.VMS
STRING.VMS	SVX.VMS
TSTDUMP.VMS	XDB.VMS
XFEQ.VMS	XFNODE.VMS
XM1.VMS	XM2.VMS
XM3.VMS	XM4.VMS
XM5.VMS	XPLOT.VMS
Z24.VMS	Z24MAIN.VMS
ZCOMPRESS.VMS	ZDAA.VMS
ZERROR.VMS	ZFORCE.VMS
ZGLBINIT.VMS	ZHAL.VMS
ZINPUTLOG.VMS	ZMBC.VMS
ZOUTPUTLOG.VMS	ZPDFPROC.VMS
ZPROPHYSICAL.VMS	ZTRAVERSE.VMS

Total of 368 files, 3524 blocks.

VAX-VMS Executable Files

Directory MSA0:[]

INSTALL.COM	README.DOC
PAL2HLP.FAC	PAL2ASS.COM
PAL2.COM	LNKCVT.COM
LNKFEP.COM	LNKJED.COM
LNKPRA.COM	LNKPRS.COM
LNKSIM.COM	LNKXPT.COM
LNKZHL.COM	CONVINC.ENV
CSOPENINC.ENV	CTRE.ENV
CTXTIO.ENV	EXTINC.ENV

RN

FNXINC.ENV	JEDFNX.ENV
JEDGLOBS.ENV	JEDINC.ENV
PAL2INC.ENV	PDFONLY.ENV
PSFUSINC.ENV	PSINC.ENV
PSPDFINC.ENV	PSXPTINC.ENV
PXSIMINC.ENV	SIMINC.ENV
XPLOTINC.ENV	Z24.ENV
ANDAR.OBJ	CONVINC.OBJ
CSOPEN.OBJ	CSOPENINC.OBJ
CTRE.OBJ	CTRESUB.OBJ
CTXTIO.OBJ	CTXTSUB.OBJ
EXTINC.OBJ	EXTPDF.OBJ
FNXINC.OBJ	INIT.OBJ
JEDCONV.OBJ	JEDDISASM.OBJ
JEDFNX.OBJ	JEDFUNC.OBJ
JEDGLOBS.OBJ	JEDINC.OBJ
JEDMAN.OBJ	JEDPARSE.OBJ
JEDWRITE.OBJ	PAL2INC.OBJ
PALASM2.OBJ	PARSE1.OBJ
PARSE2.OBJ	PARSE3.OBJ
PARSE4.OBJ	PARSE5.OBJ
PDFONLY.OBJ	PDSCNVT.OBJ
PSASM.OBJ	PSASSIGN.OBJ
PSBUILD.OBJ	PSCHECK.OBJ
PSENABLE.OBJ	PSFUSE.OBJ
PSFUSINC.OBJ	PSINC.OBJ
PSINIT.OBJ	PSJEDEC.OBJ
PSMAIN.OBJ	PSOUTPUT.OBJ
PSPDF.OBJ	PSPDFINC.OBJ
PSSUMMARY.OBJ	PSWGT.OBJ
PSXPLOT.OBJ	PSXPTINC.OBJ
PXANAL.OBJ	PXARCH.OBJ
PXCONSTR.OBJ	PXCYCLE.OBJ
PXDEBUG.OBJ	PXERROR.OBJ
PXEXEC.OBJ	PXINPUT.OBJ
PXMAIN.OBJ	PXMEMORY.OBJ
PXOUTPUT.OBJ	PXPDF.OBJ
PXSIM.OBJ	PXSIMINC.OBJ
PXSIMULA.OBJ	PXSRTSTP.OBJ

RELEASE NOTES

PXSUPPORT.OBJ	PXSYNTH.OBJ
PXTRE.OBJ	PXTXTIO.OBJ
PXVECTOR.OBJ	SANDAR.OBJ
SCAN.OBJ	SCHD.OBJ
SCMD.OBJ	SDB.OBJ
SEXT.OBJ	SFEQ.OBJ
SFNODE.OBJ	SFUNC.OBJ
SHIST1.OBJ	SIM.OBJ
SIM1.OBJ	SIMINC.OBJ
SINIT.OBJ	SINT.OBJ
SMISC.OBJ	SPRIM.OBJ
SPRLD.OBJ	SRX.OBJ
STR.OBJ	STRING.OBJ
SVX.OBJ	TSTDUMP.OBJ
XDB.OBJ	XFEQ.OBJ
XFNODE.OBJ	XM1.OBJ
XM2.OBJ	XM3.OBJ
XM4.OBJ	XM5.OBJ
XPLOT.OBJ	XPLOTINC.OBJ
Z24.OBJ	Z24MAIN.OBJ
ZCOMPRESS.OBJ	ZDAA.OBJ
ZERROR.OBJ	ZFORCE.OBJ
ZGLBINIT.OBJ	ZHAL.OBJ
ZINPUTLOG.OBJ	ZIOPRO.OBJ
ZMBC.OBJ	ZOUTPUTLOG.OBJ
ZPDFPROC.OBJ	ZPROPHYSICAL.OBJ
ZREADPRO.OBJ	ZTRAVERSE.OBJ
ZWRITEPRO.OBJ	P10H20G8.PDF
P10H20P8.PDF	P10H8.PDF
P10L8.PDF	P12H6.PDF
P12L10.PDF	P12L6.PDF
P14H4.PDF	P14L4.PDF
P14L8.PDF	P16C1.PDF
P16H2.PDF	P16L2.PDF
P16L6.PDF	P16L8.PDF
P16P8.PDF	P16R4.PDF
P16R6.PDF	P16R8.PDF
P16RA8.PDF	P16RP4.PDF
P16RP6.PDF	P16RP8.PDF

RN

RELEASE NOTES

P18L4.PDF	P18P8.PDF
P20C1.PDF	P20L10.PDF
P20L2.PDF	P20L8.PDF
P20R4.PDF	P20R6.PDF
P20R8.PDF	P20RA10.PDF
P20RS10.PDF	P20RS4.PDF
P20RS8.PDF	P20S10.PDF
P20X10.PDF	P20X4.PDF
P20X8.PDF	P22RX8.PDF
P22V10.PDF	P32R16.PDF
P32VX10.PDF	P64R32.PDF
P6L16.PDF	P8L14.PDF
PALPDF.PDF	PMS14R21.PDF
Z20SRC.PDF	Z20TAB.PDF
Z24OBJ.PDF	Z24SRC.PDF
Z24TAB.PDF	ZHAL20.PDF
ZHAL24.PDF	JEDMAN.MSG
PROSE.MSG	ZHAL24.MSG
PAL2EX.LIS	PAL2EX.COM
10BITREG.PDS	10COUNT.PDS
20RA10.PDS	3TO8DMUX.PDS
416DEC.PDS	4CNT.PDS
8COUNT.PDS	9BITCNT.PDS
9BITREG.PDS	ADREG16.PDS
ARBITER.PDS	BARREL.PDS
CONTROL.PDS	CRT.PDS
DCOUNT.PDS	FLIPFLOP.PDS
LATCH.PDS	LINK.PDS
MEMIO.PDS	MEMORY.PDS
OCTCOMP.PDS	P7000.PDS
P7004.PDS	PORT.PDS
TRAFFIC.PDS	UPCOUNT.PDS

Total of 238 files, 1334 blocks.

ASCII Files

FILE.LST	README.ASC
PAL2.COM	PAL2ASS.COM
PAL2EX.COM	PAL2HLP.FAC
PAL2EX.LIS	10BITREG.PDS
10COUNT.PDS	20RA10.PDS
3TO8DMUX.PDS	416DEC.PDS
4CNT.PDS	8COUNT.PDS
9BITCNT.PDS	9BITREG.PDS
ADREG16.PDS	ARBITER.PDS
BARREL.PDS	CONTROL.PDS
CRT.PDS	DCOUNT.PDS
FLIPFLOP.PDS	LATCH.PDS
LINK.PDS	MEMIO.PDS
MEMORY.PDS	OCTCOMP.PDS
P7000.PDS	P7004.PDS
PORT.PDS	TRAFFIC.PDS
UPCOUNT.PDS	BLDCVT.COM
BLDFEP.COM	BLDJED.COM
BLDPRA.COM	BLDPRS.COM
BLDSIM.COM	BLDSUB.COM
BLDXPT.COM	BLDZHL.COM
LNKCVT.COM	LNKFEP.COM
LNKJED.COM	LNKPRA.COM
LNKPRS.COM	LNKSIM.COM
LNKXPT.COM	LNKZHL.COM
VXPASCAL.COM	P10H20G8.PDF
P10H20P8.PDF	P10H8.PDF
P10L8.PDF	P12H6.PDF
P12L10.PDF	P12L6.PDF
P14H4.PDF	P14L4.PDF
P14L8.PDF	P16C1.PDF
P16H2.PDF	P16L2.PDF
P16L6.PDF	P16L8.PDF
P16P8.PDF	P16R4.PDF
P16R6.PDF	P16R8.PDF

RN

RELEASE NOTES

P16RA8.PDF	P16RP4.PDF
P16RP6.PDF	P16RP8.PDF
P18L4.PDF	P18P8.PDF
P20C1.PDF	P20L10.PDF
P20L2.PDF	P20L8.PDF
P20R4.PDF	P20R6.PDF
P20R8.PDF	P20RA10.PDF
P20RS10.PDF	P20RS4.PDF
P20RS8.PDF	P20S10.PDF
P20X10.PDF	P20X4.PDF
P20X8.PDF	P22RX8.PDF
P22V10.PDF	P32R16.PDF
P32VX10.PDF	P64R32.PDF
P6L16.PDF	P8L14.PDF
PALPDF.PDF	PMS14R21.PDF
JEDMAN.MSG	PROSE.MSG
ZHAL24.MSG	CONVINC.INC
CSOPENINC.INC	CTRE.INC
CTXTIO.INC	EXTINC.INC
FNXINC.INC	JEDFNX.INC
JEDGLOBS.INC	JEDINC.INC
PAL2INC.INC	PDFONLY.INC
PSFUSINC.INC	PSGLOBAL.INC
PSINC.INC	PSPDFINC.INC
PSXPTINC.INC	PXGLOBAL.INC
PXSIMINC.INC	SIMINC.INC
XPLOTINC.INC	Z24.INC
Z24GLOBAL.INC	INIT.SRC
PALASM2.SRC	PARSE1.SRC
PARSE2.SRC	PARSE3.SRC
PARSE4.SRC	PARSE5.SRC
PDSCNVT.SRC	SCAN.SRC
STRING.SRC	ANDAR.SRC
XDB.SRC	XFEQ.SRC
XFNODE.SRC	XM1.SRC
XM2.SRC	XM3.SRC
XM4.SRC	XM5.SRC
XPLOT.SRC	SANDAR.SRC
SCHD.SRC	SCMD.SRC

RELEASE NOTES

SDB.SRC	SEXT.SRC
SFEQ.SRC	SFNODE.SRC
SFUNC.SRC	SHIST1.SRC
SIM.SRC	SIM1.SRC
SIMSHELL.SRC	SINIT.SRC
SINT.SRC	SMISC.SRC
SPRIM.SRC	SPRLD.SRC
SRX.SRC	STR.SRC
SVX.SRC	CSOPEN.SRC
CTRESUB.SRC	CTXTSUB.SRC
UNXLIBNAM.SRC	EXTPDF.SRC
JEDCONV.SRC	JEDDISASM.SRC
JEDFUNC.SRC	JEDMAN.SRC
JEDPARSE.SRC	JEDWRITE.SRC

RN

VAX-UNIX

The complete listing of files on the VAX-UNIX tape is given below:

README.DOC	INSTALL.COM
PAL2LIS.DOC	PAL2EX.COM
PAL2.COM	PAL2HLP.FAC
MAKEFILE	
JEDMAN.MSG	PROSE.MSG
ZHAL24.MSG	
P10H20G8.PDF	P10H20P8.PDF
P10H8.PDF	P10L8.PDF
P12H6.PDF	P12L10.PDF
P12L6.PDF	P14H4.PDF
P14L4.PDF	P14L8.PDF
P16C1.PDF	P16H2.PDF
P16L2.PDF	P16L6.PDF
P16L8.PDF	P16P8.PDF
P16R4.PDF	P16R6.PDF
P16R8.PDF	P16RA8.PDF
P16RP4.PDF	P16RP6.PDF
P16RP8.PDF	P18L4.PDF
P20C1.PDF	P20L10.PDF
P20L2.PDF	P20L8.PDF
P20R4.PDF	P20R6.PDF
P20R8.PDF	P20RA10.PDF
P20RS10.PDF	P20RS4.PDF
P20RS8.PDF	P20S10.PDF
P20X10.PDF	P20X4.PDF
P20X8.PDF	P22RX8.PDF
P22V10.PDF	P32R16.PDF
P32VX10.PDF	P64R32.PDF
P6L16.PDF	P8L14.PDF
PALPDF.PDF	PMS14R21.PDF
ZHAL20.PDF	ZHAL24.PDF

RELEASE NOTES

10BITREQ.PDS	10COUNT.PDS
20RA10.PDS	3TO8DMUX.PDS
416DEC.PDS	4CNT.PDS
8COUNT.PDS	8LATCH.PDS
9BITCNT.PDS	9BITREQ.PDS
ADREQ16.PDS	ARBITER.PDS
BARREL.PDS	CONTROL.PDS
CRT.PDS	DCOUNT.PDS
FLIPFLOP.PDS	LATCH.PDS
LINK.PDS	MEMIO.PDS
MEMORY.PDS	OCTCOMP.PDS
P7000.PDS	P7004.PDS
PORT.PDS	TRAFFIC.PDS
UPCOUNT.PDS	ZHAL.O
Z24MAIN.O	ZCOMPRESS.O
ZDAA.O	ZERROR.O
ZFORCE.O	ZGLBINIT.O
ZMBC.O	ZPDFPROC.O
ZPROPHYSICAL.O	ZTRAVERSE.O
EXTPDF.O	JEDCONV.O
JEDDISASM.O	JEDFUNC.O
JEDMAN.O	JEDPARSE.O
JEDWRITE.O	
PSASM.O	PSASSIGN.O
PSBUILD.O	PSCHECK.O
PSDUMP.O	PSENABLE.O
PSFUUSE.O	PSINIT.O
PSJEDEC.O	PSMAIN.O
PSOUTPUT.O	PSPDF.O
PSSUMMARY.O	PSWGT.O
PSXPLOT.O	
PXANAL.O	PXARCH.O
PXCONSTR.O	PXCYCLE.O
PXERROR.O	PXEXEC.O
PXINPUT.O	PXMAIN.O
PXMEMORY.O	PXOUTPUT.O
PXPDF.O	PXSIM.O
PXSIMULA.O	PXSRTSTP.O
PXSUPPORT.O	PXSYNTH.O

RN

PXTRE.O	PXTXTIO.O
PXVECTOR.O	
CSOPEN.O	CTRESUB.O
CTXTSUB.O	UNXLIBNAM.O
PALASM2.O	INIT.O
STRING.O	SCAN.O
PARSE1.O	PARSE2.O
PARSE3.O	PARSE4.O
PARSE5.O	
SIM.O	SINIT.O
SDB.O	SFNODE.O
SFEQ.O	SMISC.O
SCMD.O	STR.O
SIM1.O	SINT.O
SINT.O	SEXT.O
SHIST1.O	SPRIM.O
SCHD.O	SFUNC.O
SANDAR.O	SPRLD.O
SRX.O	SVX.O
XPLOT.O	ANDAR.O
XDB.O	XFNODE.O
XFEQ.O	XM1.O
XM2.O	XM3.O
XM4.O	XM5.O

FIXED BUGS

Since the last release, several bugs have been fixed. We appreciate your effort in reporting problems and recommending enhancements. For your convenience, we have included a self addressed Bug/Enhancement Report in the manual. If you need more forms, call our customer support hotline.

PALASM 2.22 fixed bugs are listed below. If you call in with questions about any of these bugs, please refer to the Bug Report number given above each bug description.

Fixed Bugs: PALASM

Bug Report #: 805

System: IBM-PC

Version: 2.19

Reported by: Leonard Cardoza

Date received: 4/14/86

Description: If XPLOT is run after running a PROSE design through PALASM2, a hard crash results. However, if a non-PROSE device is run through PALASM2, and PROASM-PROSIM are run next, a hard crash does not occur.

Bug Report #: 807

System: IBM-PC

Version: 2.19

Reported by: Leonard Cardoza

Date received: 4/16/86

Description: If a pin name contains a _ character, PALASM2 goes into an endless loop.

RN

Bug Report #: 911
System: VAX-VMS
Version: 2.21
Reported by: Leonard Cardoza
Date received: 8/12/86
Description: On a PAL22V10 test, the XPLOT fusemap contained 2 rows of fuses printed together at the bottom of the output file, and none at the top. These correspond to the Set and Reset lines and should be located at the top and bottom.

Bug Report #: 912
System: VAX-VMS
Version: 2.21
Reported by: Leonard Cardoza
Date received: 8/12/86
Description: On a PAL22V10 test, columns of the XPLOT fuse array shifted by one position from where they should be. This causes the print-out to start from the second column instead of the first.

Bug Report #: 919
System: VAX-VMS
Version: 2.21
Reported by: Marilyn Gleason
Date received: 8/20/86
Description: Changes made to PAL22RX8 and PAL22V10 parts cause the line numbers on the .JED and .JDC files on the PAL32VX10 device to differ from the released version's saved output.

Bug Report #: 928
System: IBM-PC
Version: 2.21
Reported by: Leonard Cardoza
Date received: 8/20/86
Description: If no Declaration section is included in the .PDS file, the PALASM2 program goes into an endless loop.

Bug Report #: 597
System: VAX-VMS
Version: 2.15
Reported by: Marilyn Gleason
Date received: 10/2/85
Description: On a PAL20C1 test, when the chip name is longer than fourteen characters, the PASCAL error message *Array Index Value Is Out Of Range* is displayed.

Bug Report #: 714
System: VAX-VMS
Version: 2.18
Reported by: Jo-Ning Ta
Date received: 1/28/86
Description: On Prose devices, the PALASM front end does not accept a design file where the entire Equations section is missing or blank.

Bug Report #: 833
System: IBM-PC
Version: 2.20
Reported by: D. McCarthy
Date received: 5/15/86
Description: If a filename contains 12 characters, DOS truncates the name. The PALASM2 program, however, *does* accept 12-character filenames, but XPLOT does not.

Bug Report #: 863
System: IBM-PC, VAX-VMS
Version: 2.20
Reported by: Robert Steggles
Date received: 6/19/86
Description: Using the keyword EQUATIONS for state transition conditions is confusing. Instead, create a new sub-section in the state block called STATE_CONDITIONS.

RN

Bug Report #: 873
System: IBM-PC
Version: 2.20
Reported by: J. Huston
Date received: 8/19/86
Description: If the software being used does not contain the correct PDF file, XPLOT and SIM display the error *Unexpected End of .TRE or .PAL file*. It would be clearer if the message said *Missing File: <filename>*.

Bug Report #: 883
System: IBM-PC
Version: 2.20
Reported by: Robert Steggles
Date received: 7/2/86
Description: Although the PALASM2 program accepts an input file from a sub directory (for example
C: /EX2 /LINK .PDS)
XPLOT and SIM crash and display a *Range or Nil Check* error message.

Bug Report #: 885
System: IBM-PC
Version: 2.20
Reported by: Leonard Cardoza
Date received: 7/7/86
Description: On PAL22RX8 and PAL32VX10 devices, errors such as PIN.SETF, PIN.RSTF, PIN.CLKF=P-TERM, and GLOBAL.CLKF, GLOBAL.TRST, GLOBAL=P-TERM are not detected by XPLOT.

Bug Report #: 913
System: IBM-PC
Version: 2.21
Reported by: Chris Jay
Date received: 8/13/86
Description: On the PAL32VX10 device, the error message *Setting .TRST = VCC in a 32VX10 means that...* is both incorrect and invalid. The message should say *Setting .TRST = GND...*

Bug Report #: 917
System: VAX-VMS
Version: 2.21
Reported by: Leonard Cardoza
Date received: 8/15/86
Description: On a PAL22V10.test a stack dump caused equations to abort, and O15.TRST and O18.TRST equations were accepted when they contained the illegal character :+.

Bug Report #: 949
System: VAX-VMS
Version: 2.21
Reported by: Nancy Hensley
Date received: 8/28/86
Description: An incorrect part name (16RP8 instead of 16RA8) caused a Pascal dump.

Bug Report #: 954
System: VAX-VMS
Version: 2.21
Reported by: Marilyn Gleason
Date received: 9/3/86
Description: On PAL32VX10, the 25th pin name accepts the word GLOBAL, but does not accept any other name. However, on PAL22RX8 and PAL22V10 other names are accepted.

Bug Report #: 970
System: VAX-VMS
Version: 2.21
Reported by: Nancy Hensley
Date received: 9/8/86
Description: If a zero length file is specified when the PALASM2 program is run, a Pascal dump occurs.

RN

Bug Report #: 990
System: IBM-PC, VAX-VMS
Version: 2.21
Reported by: S.Scard
Date received: 9/22/86
Description: When XPLOT is run on a PAL32VX10 design file, with I/O pin as an input *OUTPUT.TRST = GND* in the hard code, and with no other output equations, the message *Setting .TRST = GND ... Cannot Define The Output...* is displayed.

Bug Report #: 998
System: IBM-PC, VAX-VMS
Version: 2.21
Reported by: Marilyn Gleason
Date received: 9/25/86
Description: On all devices, the EQUATIONS section should be optional.

Bug Report #: 1005
System: VAX-VMS
Version: 2.21
Reported by: Marilyn Gleason
Date received: 10/2/86
Description: When a State Machine file that does not contain a Simulation section is run through the PALASM2 program, error messages are displayed.

Bug Report #: 1010
System: VAX-VMS
Version: 2.21
Reported by: Nancy Hensley
Date received: 10/7/86
Description: On a PAL part with a state machine design, an error occurs if a condition or equation with a slash (/) before the equals (=) sign is used. The message *Missing Keyword Equation* is displayed.

Bug Report #: 1013
System: VAX-VMS
Version: 2.21
Reported by: Nancy Hensley
Date received: 10/13/86
Description: If you attempt to use a Bin command in the Equations section of a file, a PALASM2 dump occurs.

Bug Report #: 1017
System: VAX-VMS
Version: 2.21
Reported by: Nancy Hensley
Date received: 10/16/86
Description: Using XPLOT, a Bin command in Boolean equations causes a Pascal dump.

Bug Report #: 656
System: VAX-VMS
Version: 2.18
Reported by: Mike Gzowski
Date received: 12/10/85
Description: When the keyword EQUATIONS is missing, an error message should be displayed.

Bug Report #: 840
System: IBM-PC
Version: 2.19
Reported by: Howard Tang
Date received: 5/12/86
Description: The Stag programmer does not accept the JEDEC files generated by PALASM2. It displays a checksum error when the JEDEC files are loaded in.

RN

Bug Report #: 959
System: VAX-VMS
Version: 2.21
Reported by: Marilyn Gleason
Date received: 10/31/86
Description: On the PALASM2.G03 test, an error message about an extra parenthesis in a Simulation WHILE loop is displayed. This occurs when the number of parentheses is accurate.

Bug Report #: 1035
System: VAX-VMS
Version: 2.21
Reported by: K. Toland
Date received: 11/6/86
Description: On PAL32VX10, PALASM2 does not accept any other pin name besides GLOBAL on the Global pin.

Bug Report #: 860
System: IBM-PC
Version: 2.20
Reported by: Leonard Cardoza
Date received: 6/6/86
Description: The JEDEC standard states that the Device Type field is obsolete. Programmers other than Data I/O cannot handle Device Types. Why not eliminate the D line from the JEDEC file completely?

Bug Report #: 869
System: IBM-PC, VAX-VMS
Version: 2.20
Reported by: Marilyn Gleason
Date received: 10/31/86
Description: PALASM 2 software should support PRLD family codes on parts that are both preloadable and non-preloadable.

**Fixed Bugs:
SIMULATOR**

Bug Report #: 770

System: IBM-PC

Version: 2.19

Reported by: Leonard Cardoza

Date received: 3/19/86

Description: On PAL16R8, SETF CLK and SETF /CLK=L on the next line causes CLK=H in outputs instead of CLK=L. Since the outputs then are not accurate, the device fails the verify test on the programmer.

Bug Report #: 826

System: IBM-PC

Version: 2.19

Reported by: B. Peeters

Date received: 5/7/86

Description: When the user sets a clock pin at SETF, and then tries to clock the clock pin, incorrect output is placed in the JEDEC file.

Bug Report #: 896

System: IBM-PC

Version: 2.20

Reported by: Marilyn Gleason

Date received: 10/31/86

Description: On PAL16RA10 and PAL16RA8 tests, the design will not verify as the Output Enable and Three-state equations are evaluated incorrectly.

RN

Bug Report #: 898
System: IBM-PC
Version: 2.20
Reported by: Leonard Cardoza
Date received: 7/24/86
Description: PAL20RA10 and PAL16RA8 do not simulate correctly with three-states of an output used as an input. In the .HST and .JDC file, the pin goes from Z to H, L or 0, 1 when SETF'D.

Bug Report #: 899
System: IBM-PC, VAX-VMS
Version: 2.20
Reported by: Leonard Cardoza
Date received: 7/24/86
Description: On PAL32VX10, when an output is set to three-state, any other combinatorial output set equal to that first output should go to X. Currently, they retain their old values in .HST and .JDC.

Bug Report #: 900
System: IBM-PC, VAX-VMS
Version: 2.21
Reported by: Leonard Cardoza
Date received: 7/23/86
Description: On a PAL32VX10 test, buried registers were preloaded. When an input was changed, however, the buried register's output in the .HST file seemed to change without the register having been clocked.

Bug Report #: 908
System: IBM-PC
Version: 2.21
Reported by: Chris Jay
Date received: 8/8/86
Description: On PAL32VX10, when CLK is used as an input – not as a clock – the output is combinatorial. When you SETF CLK or SETF /CLK, however, vectors are not generated.

Bug Report #: 909
System: VAX-VMS
Version: 2.21
Reported by: Chris Jay
Date received: 8/8/86
Description: On PAL32VX10, the clock input from the fuse array was gated to a combinatorial output. When enabled, that output was half the actual clock frequency, although it should have been the same.

Bug Report #: 910
System: VAX-VMS
Version: 2.21
Reported by: Leonard Cardoza
Date received: 8/12/86
Description: On a PAL22V10 test, you find invalid check clashes because of the second assertion of the global SETF line. Also, SET and RESET lines for pins I3 and I4 are both high. The register should be set correctly, instead of changed to X.

Bug Report #: 964
System: IBM-PC
Version: 2.21
Reported by: B. Hsiang
Date received: 9/3//86
Description: .JDC test vectors in PAL20X10 contain 0,1 for outputs instead of L,H. This causes the programmer to attempt to apply voltages to outputs.

Bug Report #: 971
System: VAX-VMS
Version: 2.21
Reported by: S. Kuang
Date received: 9/8/86
Description: The preloaded values in the PAL16RP8 did not stay after the preload process was done.

RN

Bug Report #: 974
System: IBM-PC, VAX-VMS
Version: 2.21
Reported by: S. Scard
Date received: 9/9/86
Description: The PAL16R8 vectors contain 0,1 instead of L,H.

Bug Report #: 984
System: IBM-PC, VAX-VMS
Version: 2.21
Reported by: Marilyn Gleason
Date received: 9/10/86
Description: PAL20RA10, PAL20X10, PAL20X4, and PAL20X8 will not verify because the output pins in the JEDEC file are Z's when the output enable pin is 0. The history file is incorrect and seems to be missing a column.

Bug Report #: 758
System: IBM-PC
Version: 2.19
Reported by: Leonard Cardoza
Date received: 2/26/86
Description: You cannot write a PAL20RA10 where .TRST = LOW output. All outputs stay in Z state after .TRST goes low even when they are SETF in SIM.

Bug Report #: 875
System: IBM-PC
Version: 2.20
Reported by: Chris Jay
Date received: 6/24/86
Description: Test vector generation in the PAL24X family is incorrect. The message says that only the B parts are preloadable. In fact, both the A and the B parts are preloadable and only the standard parts are not preloadable.

Bug Report #: 897
System: IBM-PC, VAX-VMS
Version: 2.21
Reported by: Leonard Cardoza
Date received: 7/23/86
Description: The file PORT.PDS leaves the outputs enabled when the OE pin is H. However, it reverts to the enabled condition for unknown reasons.

Bug Report #: 915
System: VAX-VMS
Version: 2.21
Reported by: Leonard Cardoza
Date received: 8/15/86
Description: On a PAL22V10 test, an incorrect error message *No Clock Pin Defined For This Device...* was displayed. A more accurate error message would be *Cannot CLOCKF this pin.*

Bug Report #: 916
System: VAX-VMS
Version: 2.21
Reported by: Leonard Cardoza
Date received: 8/15/86
Description: On PAL22V10, after doing the PRLDF of pins O14, O15, O16...023, a message should appear which tells the user that the JEDEC output switched off.

Bug Report #: 918
System: VAX-VMS
Version: 2.21
Reported by: Nancy Hensley
Date received: 8/15/86
Description: On a PAL22RX8 test, a loop occurs in SIM. The repeated message is Warning #16: *O15 SET RESET Both Are High For Above Output.*

RN

Bug Report #: 925
System: VAX-VMS
Version: 2.21
Reported by: Nancy Hensley
Date received: 8/20/86
Description: On PAL22V10 and PAL22RX8, SIM displays a message saying that SET or RESET is always high, and then aborts.

Bug Report #: 931
System: VAX-VMS
Version: 2.21
Reported by: Marilyn Gleason
Date received: 8/27/86
Description: PAL22V10 and PAL22RX8 need to have the PRELOAD implemented in the code.

Bug Report #: 943
System: VAX-VMS
Version: 2.21
Reported by: Marilyn Gleason
Date received: 8/27/86
Description: On PAL20X4, PAL20R6, PAL20R4, PAL16R8, PAL16R6, and PAL16R4 warning message number seven is displayed. It says *This PAL Preloads Only The B Version Of The Part For PRLDF Statements*. The message should be more generic.

Bug Report #: 944
System: VAX-VMS
Version: 2.21
Reported by: Marilyn Gleason
Date received: 8/27/86
Description: On PAL20R8, PAL20R6, PAL16R8, PAL16R6 and PAL16R4, warning message #8 is displayed. It says *Registers Initialized And JEDEC Vector Generation Turned Off*. The JEDEC vector generation should not, in fact, be turned off.

Bug Report #: 945
System: VAX-VMS
Version: 2.21
Reported by: Marilyn Gleason
Date received: 8/27/86
Description: The PRLDF commands should work for PAL32VX10.

Bug Report #: 947
System: IBM-PC
Version: 2.21
Reported by: Barry Seidner
Date received: 8/28/86
Description: The simulator output for the three-state condition is incorrect.

Bug Report #: 953
System: VAX-VMS
Version: 2.21
Reported by: Marilyn Gleason
Date received: 8/29/86
Description: SIM produced a Pascal dump when the back end program PALASM2.TRE is missing. An error message telling the user that the file is missing should be displayed.

Bug Report #: 969
System: IBM-PC
Version: 2.21
Reported by: Theresa Shafer
Date received: 9/4/86
Description: On PAL32R16, error #16: *System Oscillation* occurs after the SETF /CLR, but before the FOR loop. This happens when all outputs with feedbacks are registered.

RN

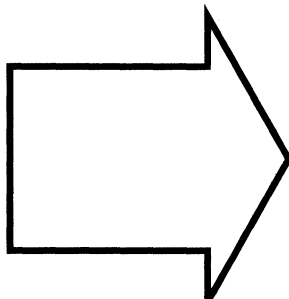
Bug Report #: 989
System: VAX-VMS
Version: 2.21
Reported by: Marilyn Gleason
Date received: 9/18/86
Description: On a 32R16 test, the part failed to verify on the programmer. This occurs when the second output enable is High and the first is Low. The output pins for the second output enable bank should be 1's and 0's. Instead, they appear as H's in the JEDEC file.

Bug Report #: 995
System: VAX
Version: 2.21
Reported by: Marilyn Gleason
Date received: 9/24/86
Description: A test on PAL32VX10 produces error #36: *SET/RESET Are Both High For The Above Output*. This error message is unwarranted, since both *can* be High on this part.

Bug Report #: 996
System: IBM-PC
Version: 2.21
Reported by: H. Nguyen
Date received: 9/24/86
Description: On PAL16R6, using an output pin with PIN.TRST in the equation, and no equation for the output pin causes Sim to hang.

Bug Report #:1034
System: IBM-PC
Version: 2.21
Reported by: Marilyn Gleason
Date received: 11/11/86
Description: When the Simulator detects GLOBAL.SETF or a GLOBAL.RSTF equations, it displays error #16: *Cannot Have Simple/Functional Equation For This Pin (Global)*.

Table of Contents	vii
Software Errata	ERR
Introduction	1
Installing PALASM®2 Software	2
PDS Syntax	3
Boolean Equation Design	3A
State Machine Design	3B
Simulation	3C
Using PALASM 2 Software	4
Installation and Operation Notes	A
Programmer Notes	B
Device Specific Syntax	C
PAL® Design File Library	D
Error Messages	E
Submitting a HAL® Design to MMI	F
PALASM 2 Syntax Diagram	G
Supplementary Software	H
JEDEC Standard No. 3	I
Release Notes	RN
Index	IX
PLEASM™ Manual	



INDEX

AND

- AND array 1-1
- AND gates 1-1
- AND operator 3A-9

ASCII Tape

- how to install the software A-10

AUTOEXEC.BAT 2-9

BINHEX 1-19; 2-12

Boolean

- Boolean transfer function 1-1
- design file
 - declaration section 3A-1
 - equations 3A-8
 - syntax 3A-3
- generating equations from a JEDEC file 1-18

CHECK 3C-1, 3, 7

CHIP 3A-2, 4; 3B-8

CLKF 3A-14

CLOCKF 3C-1, 2, 3, 6

Combinatorial Arrays 1-3

Combinatorial Equations 3A-11

Computer<->Programmer Connection

- IBM-PC 2-12
- VAX 2-13

Declaration Section 3A-1

- Boolean 3A-3
- State 3B-6

DECODE 1-19; 2-12

DEFAULT_BRANCH 3B-14

DEFAULT_OUTPUT 3B-13

EQUATIONS **3A-8**
Error Messages
 description **4-8**
 JEDMAN **E-34**
 MINIMIZE **E-8**
 PALASM2 **E-1**
 PROASM-PROSIM **E-17**
 SIM **E-32**
 XPLOT **E-2**
 ZHAL **E-30**
Examples
 design examples **D-1**

FOR...DO Loop **3C-1**
FPLA
 Field Programmable Logic Array **1-1**
Functional Equations
 CLKF **3A-16**
 RSTF **3A-15**
 SETF **3A-15**
 TRST **3A-16**
Functional Equations **3A-14**
Fusemaps (See XPLOT)

HAL Device
 description of **1-5**
 submitting a HAL design to MMI **F-1**

IBM-PC
 computer<->programmer connection **2-12**
IBM-PC/DOS 2.10 Implementation **A-1**
IF...THEN...ELSE Loop **3C-1**
Installation
 hard disk **2-6**
 how to install on the IBM-PC/XT/AT **2-3**
 how to install PALASM 2 software **2-1**
 interactive menu **2-7**
 non-menu mode **2-9**
 twin floppy system **2-4**

Interactive Menu
description of 2-2
description of PALASM 1-20
how to customize 2-10
how to install 2-7
Interactive Menu (See also PALASM) 1-20
INVERT operator 3A-9

JEDEC
download the file to the programmer 4-7
how to assemble 4-6
how to disassemble 4-6
JEDEC Standard No. 3A I-1
JEDMAN
convert PAL22V10 to PAL32VX10 4-7
description of 1-14, 18
disassemble JEDEC 4-6
error messages E-34
recalculate checksums 4-7

MASTER_RESET 3B-12
MENU.SYS 2-10
MINIMIZE
description of 1-14, 15
error messages E-8
how to use 4-5
Moore and Mealy machines 3B-12

Non-menu Mode
description of 2-2
how to install 2-9

Operator Precedence 3A-10
OR
OR array 1-1
OR gate 1-1
OR operator 3A-9
OUTPUT_ENABLE 3B-12
OUTPUT_HOLD 3B-12

- PAL Device
 - combinatorial arrays **1-3**
 - concept **1-1**
 - devices supported by PALASM 2 software **1-9**
 - product-term sharing **3A-23**
 - programmable I/O **1-3**
 - programmable polarity **1-5; 3A-12, 14**
 - references **1-8**
 - registered outputs with registered feedback **1-4**
 - XOR **1-4**
- PALASM (See also Interactive Menu)
 - description of **1-19**
- PALASM 1 Software **1-6**
 - differences from PALASM 2 software **1-6; H-2**
- PALASM 2 Software
 - asynchronous devices **1-7**
 - differences from PALASM 1 software **1-6; H-2**
 - how to install PALASM 2 software (See Installation)
 - how to use **4-1**
 - assemble the JEDEC **4-6**
 - create the PDS file **4-4**
 - detailed instructions **4-9**
 - disassemble the JEDEC **4-6**
 - minimization **4-5**
 - run the parser **4-4**
 - input and output files **1-21**
 - installation and operation
 - detailed instructions **A-1**
 - interactive menu **H-1**
 - introduction **1-6**
 - program and file summary **1-14**
 - references **1-7**
 - required equipment
 - computersComputers **1-11**
 - programmers **1-12**
 - reserved words **3-3**
 - simulation **3C-1**
 - supported products **1-9**
- PALASM2
 - description of **1-14**
 - error messages **E-1**

- PC2
 - description of 1-19, 20; H-3
 - how to use A-2
 - how to use with Data I/O programmer B-5
- PDS
 - create the PDS file 4-4
 - example files 4-1
 - file structure 3-2
 - introduction to 1-6
 - introduction to 3-1
 - using a text editor 4-4
- PDSCNVT
 - description of 1-19, 20
- Pin List 3A-5, 3B-9
- PINOUT 1-19; 2-12
- PMS14R21 1-5
- Programmable Polarity 1-5
- Polarity 3A-17
- POWER_UP 3B-21
- PROASM-PROSIM
 - description of 1-14, 18
 - error messages E-17
- PRLDF 3C-1, 3, 4, 12
- Product-term Sharing 3A-23
- Programmable I/O 1-3
- Programmable Logic Devices 1-1
- Programmable Polarity 3A-12, 14
- Programmer
 - communication between PLD programmers and IBM™ PC
(See PC2)
 - Data I/O B-1
 - how to connect to VAX-VMS B-7
 - how to use with VAX-VMS B-6
 - device family pin codes B-3
 - how to program and test the part 4-7
 - manufacturer addresses 1-13
 - programmers supported 1-13
 - VARIX OMNI B-9
- Programming a Device
 - IBM-PC<->Data I/O A-1
 - MegaPAL B-9
- PROM
 - Programmable Read-Only Memory 1-1
- PROSE™ Devices 1-5

Registered Equations **3A-12**
Registered Outputs with Feedback **1-4**
Release Notes **RN-1**
Required Equipment
 computers **1-11**
 PAL device programmers **1-12**
RSTF **3A-15**

SCRSIM **1-19; 2-12**
SETF **3A-15; 3C-1, 2, 3, 5**

SIM

 description of **1-14, 17**
 error messages **E-32**

Simulation

 commands

 CHECK **3C-1**
 CLOCKF **3C-1**
 PRLDF **3C-1**
 SETF **3C-1**
 TRACE_OFF **3C-1**
 TRACE_ON **3C-1**

 constructs **3C-3**

 FOR loop **3C-8**
 IF...THEN...ELSE loop **3C-9**
 WHILE...DO loop **3C-10**

 introduction **3C-1, 2**

 output files

 history file **3C-2**
 trace file **3C-2**

 run simulation software **4-7**

 syntax

 CHECK **3C-7**
 CLOCKF **3C-6**
 differences between Boolean and state machine **3C-12**
 FOR...DO loop **3C-1, 3, 8**
 IF...THEN...ELSE loop **3C-1, 3, 9**
 overview **3C-2**
 PRLDF **3C-4**
 SETF **3C-5**
 TRACE_OFF **3C-8**
 TRACE_ON **3C-7**
 WHILE DO loop **3C-10**
 WHILE...DO loop **3C-1, 3**

- SIMULATION 3C-3**
- STATE 3B-11**
- State Machine
 - creating a design file
 - DECLARATION section 3B-6**
 - Mealy and Moore machines 3B-5**
 - STATE section 3B-9**
 - structure and syntax 3B-5**
 - equations
 - condition equations 3B-25**
 - description 3B-14**
 - operators 3B-15**
 - output equations 3B-19**
 - POWER_UP 3B-21**
 - rules 3B-15**
 - state equations 3B-16**
 - Mealy and Moore Behavior 3B-2**
- STRING 3A-6**
- Supplementary Software
 - PALASM**
 - interactive menu H-1**
 - PDSCNVT H-1**
- Supported Products **1-9**
- Syntax
 - diagrams G-1**
 - special syntax
 - PAL22RX8 C-2**
 - PAL22V10 C-2**
 - PAL32VX10 C-3**

- Three-state **3A-16**
 - programmable I/O 1-3**
- TIMING 1-19; 2-12**
- TRACE_OFF 3C-1, 2, 3, 8**
- TRACE_ON 3C-1, 2, 3, 7**

- VAX
 - computer<->programmer connection 2-13**
- VAX-UNIX
 - command procedures A-13**
 - how to install the software A-12**

VAX-VMS

- how to connect to Data I/O **B-7**
- how to install the software **A-6**
- software support procedures **A-8**

Volume Production

- see HAL **F-1**

VTRACE

- description of **1-19, 21**

VTRACE **2-12**

WHILE...DO Loop **3C-1**

XOR

- devices **1-4**
- minimizing **1-15**
- XOR operator **3A-9**

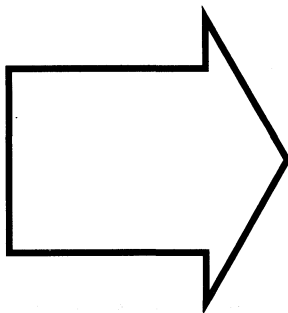
XPLOT

- description of **1-14**
- error messages **E-2**

ZHAL

- description of **1-14, 18**
- error messages **E-30**
- how to order the ZHAL program **1-14**
- how to use **4-7**

Table of Contents	vii
Software Errata	ERR
Introduction	1
Installing PALASM® 2 Software	2
PDS Syntax	3
Boolean Equation Design	3A
State Machine Design	3B
Simulation	3C
Using PALASM 2 Software	4
Installation and Operation Notes	A
Programmer Notes	B
Device Specific Syntax	C
PAL® Design File Library	D
Error Messages	E
Submitting a HAL® Design to MMI	F
PALASM 2 Syntax Diagram	G
Supplementary Software	H
JEDEC Standard No. 3	I
Release Notes	RN
Index	IX
PLEASM™ Manual	





PLEASM™ MANUAL

Version 1.2H

PLEASM software documentation follows. Your software package does not include PLEASM software. If you use PLE devices, please contact your local Monolithic Memories representative for a copy of PLEASM software.

Table of Contents

CHAPTER 1: INTRODUCTION	1-1
CHAPTER 2: PROMs vs. PLEs	2-1
2.1 The PROM as a Memory Element	2-1
2.2 The PROM as a Programmable Logic Element ...	2-3
CHAPTER 3: PLE DEVICES SUPPORTED	3-1
CHAPTER 4: RUNNING PLEASM	4-1
CHAPTER 5: PLEASM - AN EXAMPLE	5-1
CHAPTER 6: THE SYNTAX OF PLEASM	6-1
APPENDIX A: PLEASM ON THE VAX11/VMS	A-1
A.1 PLEASM on the VAX/VMS	A-1
A.2 PLEASM on the IBM PC DOS 2.0	A-5
APPENDIX B: PROM/PLE PROGRAMMER INFORMATION	B-1
APPENDIX C: PLEASM ERROR MESSAGES	C-1
APPENDIX D: PLE DESIGN FILE LIBRARY	D-1
APPENDIX E: HELP!! AND WHERE TO GET IT	E-1
APPENDIX F: USER CUSTOMIZATION	F-1

Introduction

PLEASM (Programmable Logic Element ASseMbler) is a software package developed by Monolithic Memories, Inc. used for designing with PROMs as Programmable Logic Elements (PLE™s). PLEASM is a FORTRAN IV program which assembles and simulates PLE Design Specifications. It also generates programming formats for direct download to PROM programmers and can therefore be regarded as a tool that reduces the design-to-production time considerably.

Key Features

- Assembles Logic or Arithmetic equations into a PROM truth table.
- Provides INTEL HEX and ASCII HEX programming formats along with the hex check sum.
- Programming formats can be directly downloaded to standard PROM programmers.
- Simulates the Function Table, in the design equations.
- Reports design errors.

The purpose of this manual is to aid the user in running PLEASM and getting to know and understand all its capabilities. It begins with an article that describes the wide range of PROM applications and the motivation for developing a software tool to aid in designing these applications. The next section states the system requirements for PLEASM. The next two sections give a detailed account of how to run the program with an explanation of what each of the options accomplish. This is strengthened by an example where all the operations have been performed on an input file that has the PLE specifications for basic logic gates. The PLEASM syntax is described next. This is best understood if this section is read in accompaniment with some of the design examples that come with the PLEASM program. A detailed PLE applications handbook is available.

The Appendix gives some machine-specific information about PLEASM along with details on PROM programmers, the PLE design files supplied as examples, and user customization. An important part of the Appendix is the section on the errors detected by PLEASM. This should be very useful when creating your own PLE designs.

The new PLEASM user should read Chapters 3 to 5 initially and then try to run the demonstration examples. Once the user is ready to create his own design, Chapters 2 and 6 will provide ideas and details of the specification format. The Appendices serve to provide additional supporting information on a number of subtleties the user should be aware of in fully utilizing the software.

PROMs vs. PLEs

PROMs have grown steadily in size and speed since their introduction in 1971, when Monolithic Memories introduced the world's first 1K bit bipolar PROM. Today, 16K and 32K PROMs are readily available and their speeds have improved such that the maximum address time (address to output) of these devices is down to 30–40 nanoseconds, over the complete operating temperature and VCC range. This means that PROMs can be used effectively in both high speed memory and logic replacement applications.

The PROM implements a sum-of-products Boolean transfer function in which any possible input (address) combination can be transferred to any output variable (data out). Figure 2-1 shows logical structure of a typical PROM. The input Fixed-AND array is a decoder and the output Programmable-OR array is a decoder. It is this decoder area that is field programmable to implement any Boolean transfer function.

Each output of the AND array is connected to an input of the OR array by thin metal wire (e.g. titanium-tungsten, nichrome, platinum-silicide) which can be selectively removed from the circuit by passing a current through it. This is referred to as "programming" or "blowing" a "fuse".

2.1 The PROM as a Memory Element

Because of their high speeds, bipolar PROMs are ideal for use in systems requiring fast Address-to-Data access times. PROM applications can be found in both the data and control paths of a system.

In data paths, PROMs are used mainly as storage elements to implement different table look-up applications such as trigonometric functions, signal processing coefficients, bootstrapping and initialization programs, etc. In particular, the PROM can be used to advantage in the design of digital filters and Fast Fourier Transforms. In character generator applications, the PROM user has the flexibility of modifying the conventional fonts to his/her particular requirements.

FUNCTION					F1 F2 F3 F4				
ADDRESS	A0	A1	A2	A3	OUTPUT	O1	O2	O3	O4
0	0	0	0	0		1		1	
1	1	0	0	0		1		1	
2	0	1	0	0		1		1	
3	1	1	0	0		1		1	
4	0	0	1	0				1	
5	1	0	1	0				1	1
6	0	1	1	0				1	1
7	1	1	1	0		1		1	
8	0	0	0	1				1	
9	1	0	0	1				1	1
10	0	1	0	1				1	1
11	1	1	0	1		1		1	
12	0	0	1	1				1	
13	1	0	1	1				1	
14	0	1	1	1				1	
15	1	1	1	1			1	1	

Figure 2-1. Typical PROM Structure

In control paths, PROMs are used mainly to store microprograms. Microprogrammed controllers may be simple PROM-register finite state machines or they may be complex microprogrammed CPUs, where the complete instruction set of the system resides in PROM. It is thus possible, by using PROMs for microprogramming, to use the same hardware to emulate the characteristics of various processors.

Since most memory applications involve storing PROM memory data in a temporary register before it is used (pipelining), this has spawned a new generation of PROMs with on-board D-type edge-triggered registers. These registered PROMs operate faster than discrete PROM-register combinations, and the registered PROMs also occupy less space.

2.2 The PROM as a Programmable Logic Element

The PROM implements a sum-of-products Boolean transfer function so that any function of x inputs and y outputs may be generated in a PROM with x addresses and y data outputs. Figure 2-2 shows the combinatorial functions available in a hypothetical 16 X 4 PROM.

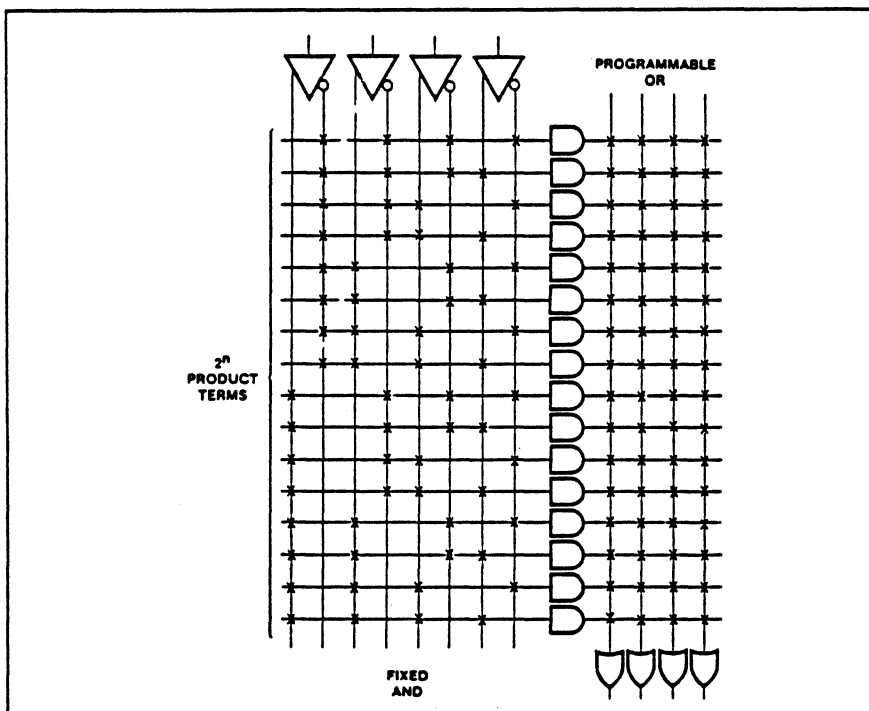


Figure 2-2. Combinatorial Functions Available in a Hypothetical 16x4 PROM

The AND-OR structure of the PROM can be viewed as a two-level logic circuit. The fixed AND plane contains all possible input combinations. Each input combination is a product term and it is connected to the output in the OR plane.

In terms of a PLE circuit, a product term is the equivalent of an AND gate equal in size to the number of inputs. Each output is equivalent to an OR gate connected to all the AND gates. Programming a fuse blows this connection between the AND gate and the OR gate. The PROM thus conveniently implements combinatorial logic when a large number of input combinations are required or a large number of product terms per output is desired.

Most applications of PLEs are in synchronous control systems where they replace random logic or customise logic functions. In data paths, they are used to generate complex functions such as pseudo random number generators, ALU operations, multiplications, reciprocals, etc.

PLE DEVICES SUPPORTED

This version of PLEASM supports all of the present members of the PLE family. Several soon to be released PLEs* are also supported. Supported PLEs are summarized in the Selection Guide below. Consult the PLE Data Sheet for pinouts and detailed specifications.

Part Number	Pins	Inputs	Outputs	Product Terms	Output Registers	tPD Max	ICC Max
PLE5P8	16	5	8	32		25ns	125mA
PLE5P8A	16	5	8	32		15ns	125mA
*PLE5P16	24	5	16	32		15ns	180mA
*PLE5P16-2	24	5	16	32		30ns	100mA
*PLE6P16	24	6	16	64		15ns	200mA
PLE6P16-2	24	6	16	64		30ns	110mA
PLE8P4	16	8	4	256		30ns	130mA
PLE8P8	20	8	8	256		28ns	140mA
PLE9P4	16	9	4	512		35ns	130mA
PLE9P8	20	9	8	512		30ns	155mA
PLE10P4	18	10	4	1024		35ns	140mA
PLE10P8	24	10	8	1024		30ns	160mA
PLE11P4	18	11	4	2048		35ns	150mA
PLE11P8	24	11	8	2048		35ns	185mA
PLE12P4	20	12	4	4096		35ns	175mA
PLE12P8	24	12	8	4096		40ns	190mA
*PLE13P8	24	13	8	8192		45ns	190mA
PLE9R8	24	9	8	512	8	15ns	180mA
PLE10R8	24	10	8	1024	8	15ns	180mA
PLE11RA8	24	11	8	2048	8	15ns	185mA
PLE11RS8	24	11	8	2048	8	15ns	185mA

Note: All outputs are three-state. Only Commercial Specifications are listed. Clock to output times are given for Registered PLEs.

* Contact MMI for availability of the following PLE parts: PLE5P16, PLE5P16-2, PLE6P16, PLE6P16-2, and PLE13P8. The specifications for these parts are preliminary.

USER'S GUIDE TO PLEASM—PLE ASSEMBLER VERSION 1.2H

To get started with PLEASM, turn the computer ON. Check your directory to make sure you have the files mentioned in Appendix A. Once this has been verified, run the program as explained below with one of the example files. Our suggestion is to start with the file P5000.PLE which contains the PLE Design Specifications for the basic logic gates. Once the capabilities of the program have been understood, you can work with any of the other examples or attempt to create your own designs.

USING PLEASM

Type the system's execute command to run the program. PLEASM will respond....

```
MONOLITHIC MEMORIES PLEASM(tm) VERSION 1.2H  
(C) COPYRIGHT 1984 MONOLITHIC MEMORIES
```

```
WHAT IS THE SOURCE FILENAME (d:filename.ext) ?:P5000.PLE
```

At this point enter the name of the file containing the specifications for the PLE being designed. If you are using this package for the first time, we suggest you try out one of the design examples that was sent along with PLEASM. PLEASM next prompts you for the name of the file you could have the output sent to, defaulting to the console....

```
OUTPUT FILENAME - PRESS <ENTER> FOR NO OUTPUT FILE ?:
```

If you press <enter>/<return> the output will be sent to the console after each operation. At this point, the input file is read, and a count of lines and characters in the file is written out to the screen. The next prompt is for the operation you want performed and is....

```
E=ECHO INPUT  S=SIMULATE  T=TRUTH TABLE  B=BRIEF TABLE  
H=HEX TABLE  I=INTEL HEX  A=ASCII HEX    C=CATALOG  Q=QUIT
```

```
ENTER OPERATION CODE:C
```


You can now enter the appropriate operation code, IN UPPER CASE!!.

The various options are briefly discussed below.

E ECHO INPUT. Prints the input PLE specifications file. Useful as a ready reference while working interactively with PLEASM.

S SIMULATE. Exercises the logic values in the optional function table in the logic equations provided. Errors in the function table are detected along with fairly explicit diagnostic messages. An important point to note is that all "don't care" conditions are treated as low logic values. This option can be successfully invoked only when a function table is present in the input specifications.

T TRUTH TABLE. Prints out the entire binary truth table for all the input variables in the PLE by substitutions into the Boolean equations specified. The output has a tabular format for ease of reading. The program also provides a hex checksum for the entries in the truth table at the end.

B BRIEF TABLE. Prints out the truth table only for the used input addresses in the PLE, again by substitutions into the Boolean equations. The output is tabulated as before, this time with a partial hex checksum corresponding to the possibly shorter table.

H HEX TABLE. Prints out the entire truth table as before, except the inputs and outputs are translated into hex. Also generates a tabular format with a hex checksum.

I INTEL HEX . Generates the Intel Hex format for PROM programmers for both 4- and 8-bit data downloading. The format is shown below....

```
:AABBBB00CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCDD
```

```
:      Starting colon marker
AA     Record length in hex
BBBB   Record starting address in hex
C..C   Data
DD     Hex checksum
```

Here all data is sent in streams of 16 8-bit bytes starting at 0000H. 4-bit data is padded up with zeros in the most significant four places. The checksum is the negative of the sum of all 8-bit bytes starting at "AA" up to "DD", modulo 256. Transmission is terminated by the string ":00000001FF".

A ASCII HEX SPACE. Generates the ASCII Hex format for PROM programmers for 4- and 8-bit data downloading. The format is shown below....

```

A
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB .
C
A --> Record start character (STX)
BB --> Data byte
C --> End of text character (ETX)
    
```

Data is sent in streams of 16 8-bit bytes separated by spaces. An execute character, the ASCII period ".", is sent at the end of each stream of data. 4-bit data is padded up with zeros in the most significant 4 places. In addition, a hex checksum is passed at the end of the transmission.

C CATALOG. Prints a one-line description of each option provided by PLEASM. Always displays to the console.

CATALOG OF OPERATION CODES:

MONOLITHIC MEMORIES PLEASM(tm) VERSION 1.2H

PLEASM --PLE ASSEMBLER-- PROVIDES THE FOLLOWING OPTIONS :

- C CATALOG** - PRINTS THE PLEASM CATALOG OF OPERATIONS
- E ECHO INPUT** - PRINTS THE PLE DESIGN SPECIFICATIONS
- T TRUTH TABLE** - PRINTS THE ENTIRE TRUTH TABLE
- B BRIEF TABLE** - PRINTS ONLY USED ADDRESSES IN THE TRUTH TABLE
- H HEX TABLE** - PRINTS THE TRUTH TABLE IN HEX FORM

- S SIMULATE** - EXERCISES THE FUNCTION TABLE IN THE LOGIC EQUATIONS

- I INTEL HEX** - GENERATES INTEL HEX PROGRAMMING FORMAT
- A ASCII HEX** - GENERATES ASCII HEX PROGRAMMING FORMAT

- Q QUIT** - EXITS PLEASM



Q QUIT. Exits the PLEASM program and prompts for restarting with another input specifications file.

5. PLEASM - An Example

ECHO

Echoes the input PLE Design Specifications file. This will help verify that the input file has been read in correctly.

ENTER OPERATION CODE: E

```
PLE5P8                               PLE DESIGN SPECIFICATION
P5000                                VINCENT COLI 01/12/84
BASIC GATES
MMI SANTA CLARA, CALIFORNIA
```

```
.ADD I0 I1 I2 I3 I4
.DAT 01 02 03 04 05 06 07 08
```

```
01 = I0                               ; BUFFER
02 = /I0                              ; INVERTER
03 = I0 * I1 * I2 * I3 * I4          ; AND GATE
04 = I0 + I1 + I2 + I3 + I4          ; OR GATE
05 = /I0 + /I1 + /I2 + /I3 + /I4     ; NAND GATE
06 = /I0 * /I1 * /I2 * /I3 * /I4     ; NOR GATE
07 = I0 :+: I1 :+: I2 :+: I3 :+: I4   ; EXCLUSIVE OR GATE
08 = I0 **: I1 **: I2 **: I3 **: I4    ; EXCLUSIVE NOR GATE
```

FUNCTION TABLE

```
I0 I1 I2 I3 I4 01 02 03 04 05 06 07 08
```

```
;INPUT  - -  OUTPUTS FROM BASIC GATES  - -
;01234  BUF INV AND OR NAND NOR XOR XNOR  COMMENTS
-----
LLLLL   L  H  L  L  H  H  L  L  ALL ZEROS
HHHHH   H  L  H  H  L  L  H  H  ALL ONES
HLHLH   H  L  L  H  H  L  H  H  ODD CHECKERBOARD
LHLHL   L  H  L  H  H  L  L  L  EVEN CHECKERBOARD
-----
```

DESCRIPTION

THIS EXAMPLE ILLUSTRATES THE USE OF PLEs TO IMPLEMENT THE BASIC GATES: BUFFER, INVERTER, AND GATE, OR GATE, NAND GATE, NOR GATE, EXCLUSIVE OR GATE, AND EXCLUSIVE NOR GATE.

NOTE ALSO THAT THREE-STATE OUTPUTS ARE PROVIDED WITH ONE ACTIVE LOW OUTPUT ENABLE CONTROL (/E).

PLEASM GENERATES THE PROM TRUTH TABLE FROM THE LOGIC EQUATIONS AND SIMULATES THE FUNCTION TABLE IN THE LOGIC EQUATIONS.

SIMULATE

This option verifies that the output entries in the Function Table are correct for the given Boolean equations and input vectors. Any discrepancy between the expected output value as given in the Function Table and the output value as computed from the Boolean equations is flagged as an error. The following are acceptable input entries in the Function Table:

H - High level
 L - Low level
 X - Irrelevant

ENTER OPERATION CODE: S

FUNCTION TABLE

IO I1 I2 I3 I4 O1 O2 O3 O4 O5 O6 O7 O8

;	INPUT	- - OUTPUTS FROM BASIC GATES - -								
;	O1234	BUF	INV	AND	OR	NAND	NOR	XOR	XNOR	COMMENTS
LLLLL	L	H	L	L	H	H	L	L	L	ALL ZEROS
HHHHH	H	L	H	H	L	L	H	H	H	ALL ONES
HLHLH	H	L	L	H	H	L	H	H	H	ODD CHECKERBOARD
LHLHL	L	H	L	H	H	L	L	L	L	EVEN CHECKERBOARD

PASS SIMULATION

TRUTH TABLE

Generates an exhaustive binary truth table for all the given inputs by substitution in the Boolean equations.

ENTER OPERATION CODE: T

BASIC GATES

```
.ADD I0 I1 I2 I3 I4  
.DAT O1 O2 O3 O4 O5 O6 O7 O8
```

ADD	A0	A1	A2	A3	A4	O0	O1	O2	O3	O4	O5	O6	O7
0	L	L	L	L	L	L	H	L	L	H	H	L	L
1	H	L	L	L	L	H	L	L	H	H	L	H	H
2	L	H	L	L	L	L	H	L	H	H	L	H	H
...
29	H	L	H	H	H	H	L	L	H	H	L	L	L
30	L	H	H	H	H	L	H	L	H	H	L	L	L
31	H	H	H	H	H	H	L	H	H	L	L	H	H



BRIEF TABLE

Prints the truth table for only the used input and output pins.

ENTER OPERATION CODE: B

.ADD I0 I1 I2 I3 I4

.DAT O1 O2 O3 O4 O5 O6 O7 O8

BASIC GATES

ADD	A0	A1	A2	A3	A4	O0	O1	O2	O3	O4	O5	O6	O7
0	L	L	L	L	L	L	H	L	L	H	H	L	L
1	H	L	L	L	L	H	L	L	H	H	L	H	H
2	L	H	L	L	L	L	H	L	H	H	L	H	H
.
.
29	H	L	H	H	H	H	L	L	H	H	L	L	L
30	L	H	H	H	H	L	H	L	H	H	L	L	L
31	H	H	H	H	H	H	L	H	H	L	L	H	H

HEX TABLE

Generates the truth table with input and output vectors translated into hex.

ENTER OPERATION CODE: H

BASIC GATES

```
.ADD I0 I1 I2 I3 I4  
.DAT 01 02 03 04 05 06 07 08
```

ADD	HEX ADDRESS	HEX DATA
0	000	0032
1	001	00D9
2	002	00DA
.	.	.
.	.	.
29	01D	0019
30	01E	001A
31	01F	00CD



INTEL HEX

Generates the Intel Hex programming format for downloading to a PROM programmer.

ENTER OPERATION CODE: I

```
:1000000032D9DA19DA191AD9DA191AD91AD9DA1940  
:10001000DA191AD91AD9DA191AD9DA19DA191ACD54  
:00000001FF
```

ASCII HEX

Generates the ASCII Hex Space programming format for downloading to a PROM programmer, with the required <STX>, <SOH>, AND <ETX> control characters delimiting the transmission.

ENTER OPERATION CODE: A

^B^A

32 D9 DA 19 DA 19 1A D9 DA 19 1A D9 1A D9 DA 19 .
DA 19 1A D9 1A D9 DA 19 1A D9 DA 19 DA 19 1A CD .

^C

OOF3C

QUIT

Exits PLEASM and prompts for restarting with another input file.

ENTER OPERATION CODE: **Q**

RESTART PLEASM(Y/N) ?:

PLE DESIGN SPECIFICATION

The PLE Design Specification is the input file used with PLEASM. It is also the recommended data sheet format for describing the function of a PROM, once it has acquired the unique personality of a particular fuse pattern. This format for creating an input specifications file can be best understood by studying the examples that come along with this package. The format for the PLE Design Specification is:

Line 1 PLE part number, starting in column 1, followed by the words *PLE DESIGN SPECIFICATION*.

Line 2 User's part number followed by the designer's name and the date, starting in column 1. This may be an identifier that defines the application and is used for reference.

Line 3 Device application name, starting in column 1.

Line 4 User's company name and address, starting in column 1.

Lines 2-4 are without formal rules and are provided for documentation purposes.

Line 5 Address pin list, prefixed by *.ADD*, starting in column 1. The pin list should be ordered LSB first.

Line 6 Data pin list, prefixed by *.DAT*, starting in column 1. The pin list should be ordered LSB first.

The pin list is a sequence of symbolic names separated by one or more spaces on one or more lines in the order of the device pin numbers. Each symbolic name is unique (except the unused pins which may have the same name). All pins including power and ground must be named. Names may use any printable characters except the operators ';', '.', ',', '=', '*', and '+'. The prefix '/' is used to logically complement the name.

Line m EQUATIONS

The transfer function of the device is expressed in the following form:

SYMBOL = EXPRESSION

The following terms are used to construct the equations:

SYMBOL Pin Name with optional prefix '/'

EXPRESSION A sequence of SYMBOLS separated by operators.

OPERATORS (in hierarchy of evaluation)

; Comment follows

. Dot operator (pin list or arithmetic operator follows)

ADD Address pins (Inputs)

DAT Data pins (Outputs)

, Delimiter, separates binary bits (MSB first)

= Equality (combinatorial)

BOOLEAN OPERATORS

/ Complement, prefix to a pin name

* AND (PRODUCT)

+ OR (SUM)

:+ XOR (EXCLUSIVE OR)

:* XNOR (EXCLUSIVE NOR)

ARITHMETIC OPERATORS

- .*. Multiply (Arithmetic multiplication)
- +. Plus (Arithmetic addition)

Line n FUNCTION TABLE

The function table begins with the key word *FUNCTION TABLE*, starting in column 1 of a line following the equation list. It is followed by a pin list which may be in a different order and polarity from the pin list in Lines 5 and 6. The pin list is followed by a line of dashes (-'s) which is in turn followed by a list of vectors, one vector per line. One state must be specified for each pin name and optionally separated by spaces.

A vector is a sequence of states listed in the same order as the function table's pin list and followed by an optional comment. The vector list is followed by another dashed line.

An important restriction is that blank lines are not permitted in the body of the function table. To separate logically distinct parts of the function table, however, comments can be used. In other words, blank lines are permitted with a semicolon (;) in the first column. Additionally, comments can be placed in this line. Extra blank lines might result in the simulator scanning past the end of the function table and detect "-" as an error symbol, resulting in failure to pass simulation.

A function table is optional and need be present only for simulation to be performed. The keyword *FUNCTION TABLE*, however, is necessary in the input file. This should start in column 1, and should follow the equation list.

Definition of Function Table States:

Symbol	Definition	Input	Output
H	HIGH LEVEL	Drive High	Test High
L	LOW LEVEL	Drive Low	Test Low
X	IRRELEVANT	Don't Care Condition	Do Not Test

Line o DESCRIPTION

This begins with the keyword *DESCRIPTION*, starting in column 1. The device operation and application are described here. All the lines following the keyword *DESCRIPTION* are treated as comments. Even though the lines following may be blank, the keyword *DESCRIPTION* is necessary in the input file for assembly.

An informal grammar for the input specifications file is given below:

```

PLE_SPECIFICATIONS_FILE →      PLE_TYPE_LINE
                                PLE_PATTERN_LINE
                                APPLICATION_DESCRIPTION_LINE
                                COMPANY_INFORMATION_LINE
                                ADDRESS_PIN_LIST
                                DATA_PIN_LIST
                                EQUATION_LIST
                                FUNCTION_TABLE
                                [FUNCTION_TABLE_PIN_LIST]
                                [FUNCTION_TABLE]
                                DESCRIPTION
                                [COMMENTS]

PLE_TYPE_LINE → PLE<PLE PART NUMBER> ... PLE DESIGN SPECIFICATION

PLE_PATTERN_LINE → <REFERENCE NUMBER FOR APPLICATION, AUTHOR'S NAME>

APPLICATION_DESCRIPTION_LINE → <APPLICATION NAME>

COMPANY_INFORMATION_LINE → <COMPANY NAME AND ADDRESS>

ADDRESS_PIN_LIST → .ADD <PIN NAME LIST FOR INPUT PINS>

DATA_PIN_LIST → .DAT <PIN NAME LIST FOR OUTPUT PINS>

EQUATION_LIST → <LIST OF BOOLEAN EQUATIONS>

FUNCTION TABLE
    
```

FUNCTION_TABLE_PIN_LIST → <LIST OF PINS TO BE SIMULATED>

FUNCTION_TABLE → H|L|X <ENTRIES DEFINING LOGIC VALUES FOR THE PINS>

DESCRIPTION

COMMENTS → <COMMENTS DESCRIBING APPLICATION>

where, [...] denotes an optional expression,

<.> denotes an informal representation for the expression,

'__' denotes a keyword,

and '|' denotes the "OR" operator.

IMPORTANT NOTES ON PLEASM INPUT SPECIFICATIONS

1. The specifications file must contain the keywords *FUNCTION TABLE* and *DESCRIPTION* starting in column 1 for assembly to occur.
2. All responses to PLEASM prompts should be in upper case.
3. The input specifications file should be entirely in upper case.
4. The number of characters in the file, the number of lines in the file, and the number of characters in each line should be within the limits set in the I/O initialization package. The current limits set are 9999 chars/file, 250 lines/file, and 80 chars/line.

A.1 PLEASM ON THE VAX11/VMS

The following files should be in your tape if you have the Load/Go System:

PLEASM.EXE. This is the executable file that can be invoked to assemble your input PLE specifications.

P5000.PLE through P5032.PLE. These are the example files containing some applications. They are useful for studying how the input file should be written, and can be run with PLEASM to provide an on-line demonstration of how the program works. Details on the contents of these files can be found in Appendix D.

IOINIT.FOR. User customization package for array dimensions and I/O.

If you have ordered the Development system, you should have in addition the files:

PLEASM.FOR. This file contains the FORTRAN source for the PLEASM program and can be compiled by any FORTRAN compiler you have at your disposal.

IOLIB.FOR. This file contains the I/O features that make this version of PLEASM compatible with that on the IBM PC. This will have to be linked in with the main program during compilation.

Unloading Your Magnetic Tape under VAX/VMS

Helpful Hints

- The volume is labelled PLEASM and is recorded in Files-11 format, 1600 BPI and 9-track magnetic tape.
- For the neophytes who wish to learn everything there is to learn about mag tapes and more, Digital Equipment Corporation has published The Magnetic Tape Users' Guide (Order No. AA-M539A-TE).

- Your tape drive goes by many different names. For example, MTA0: or MSA0:. To list all devices on your installation, type SH DEV M<cr> when you see the \$ prompt.

After loading your tape on the drive, when you see the \$ prompt, type the following:

\$ ALL MTA0: TAPE

This allocates space for TAPE on device MTA0.

\$ MOUNT/OVER=IDENT TAPE

This mounts the tape and overrides any tape labels. Operator privileges are sometimes required to do this.

**\$ CREATE/DIR [.PLEASM]
\$ SET DEFAULT [.PLEASM]**

This sets up the directory appropriately.

\$ COPY TAPE:*..*;* [*]*

This copies the tape files to your directory.

\$ DISMOUNT TAPE

Dismounts tape and wraps things up.

DISREGARD THIS MARKED SECTION IF YOU HAVE THE LOAD/GO SYSTEM !!

Compile and link the source program using the following sequence of commands, to create the executable version:

**\$ FORTRAN PLEASM,IOLIB
\$ LINK PLEASM,IOLIB**

Using PLEASM

Create your PLE Design Specification file using one of your system's editors. Then type the following:

```
$ RUN PLEASM
```

The program PLEASM should now run. At this point, refer to Chapter 4 for step-by-step instructions on how to use the program.

Dumping a File From the VAX11/VMS to the Data I/O

Cable Connections

The RS-232C cable that connects the VAX-11 to the Data I/O has lines 2 and 3 reversed. The only other pins that must be connected are pins 1 and 7.

Operating Procedures

1. Turn Data I/O power off.
2. Connect the Data I/O programmer to the modem and VT100 terminal as shown in Fig. A-1.1.
3. Turn the Data I/O programmer on.
4. Press the "SELECT" (Data I/O).
5. Enter "EB" (Data I/O)
6. Press the "START" (Data I/O).
7. Type "TYPE FILENAME.DAT" (VT100).
8. Press "RETURN" on the VT100.
9. Disconnect VT100 terminal from the modem and Data I/O.

10. Reconnect Data I/O to VT100 as shown in Fig. A-1.2.
11. Press the "SELECT" (Data I/O).
12. Enter "E1" (Data I/O).
13. Press "START" (Data I/O).
14. Use VT100 keyboard in order to communicate with the Data I/O.

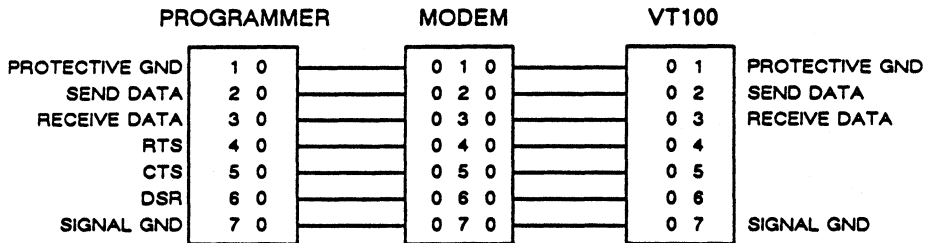


Figure A-1.1. Downloading from Host (VAX-11) to Programmer (VAX Talking to Programmer and VT100)

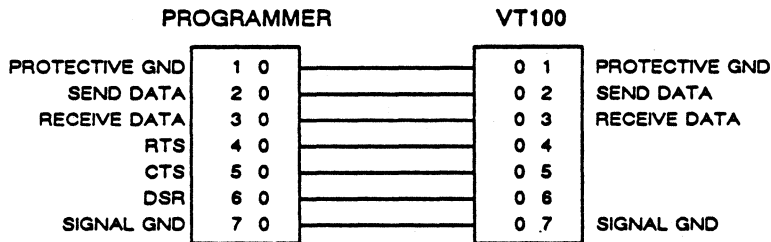


Figure A-1.2. Using Programmer as Host

A.2 PLEASM ON THE IBM PC DOS 2.0 (OR ABOVE)

Please note that for the IBM PC, system requirements are as follows:

- 8088 based microprocessor system
- 64K bytes minimum of memory
- MS-DOS (PC-DOS) operating system
- Optional text printer
- 1 disk drive.

The IBM PC version comes with a diskette which contains the following files:

Disk #1: PLEASM.EXE (if the Load/Go system has been ordered)
PLEASM.FOR (if the Development system was ordered)
P5000.PLE - P5032.PLE
PALCOM.EXE&PALSETUP.EXE

PLEASM.EXE is the executable version of PLEASM.PLEASM.

PLEASM.FOR is the FORTRAN source for the program.

P5000.PLE-P5032.PLE contain the example applications.

PALCOM.EXE is the program used for downloading.

PALSETUP.EXE allows the user to specify communications protocol.

DISREGARD THIS MARKED SECTION IF YOU HAVE THE LOAD/GO SYSTEM !!

.....

To create the executable version for the source program you have to compile and link the source file using any FORTRAN compiler/linker you have at your disposal. The one recommended is the Supersoft FORTRAN compiler which was used during the development and testing of this program. For this compiler the sequence of commands to create the executable file would be (with the compiler/linker in drive B: and the source in drive A:):

```
A> B:SFOR PLEASM.FOR PLEASM.REL
A> B:CNV PLEASM.REL PLEASM.OBJ/R
A> B:LINK S+SEMU+@PLEASM.RSP,PLEASM,NUL,SFLIB+MLIB,,
```

This creates the executable file PLEASM.EXE.

.....

To use PLEASM on the IBM PC, two steps are necessary:

1. Create and edit the PLE Specification File.
2. Run PLEASM.

To run PLEASM with Disk #1 in drive A, type the following when you see the A> prompt:

```
A> PLEASM
```

At this point, PLEASM begins running. For step-by-step instructions on how to use the program, please refer to Chapter 4.

General

1. Do NOT terminate the program abnormally by pressing CTRL-C, CTRL-BREAK, etc. when you are sending the output to a file rather than the screen. Use instead the QUIT option to terminate your session. If the session is terminated using CTRL-C, this may result in lost files on your disk. This is because your output file will not have been properly closed.
2. The approximate run-times for the programs vary depending on the size of the PLE and the length of the input specifications file. Simulation takes between 1 and 3 minutes, while generating programming formats could take anywhere between 3 and 10 minutes.

**DATA I/O MODEL 19 PROGRAMMER WITH UNIPAK/
UNIPAK2**

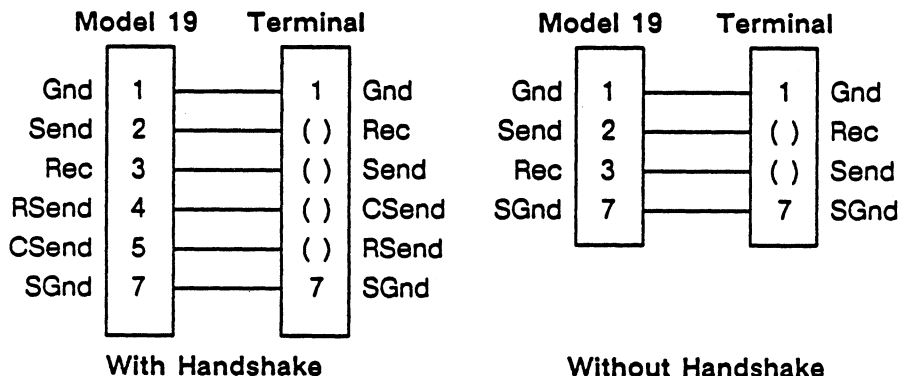
Key Features

- Accepts PLE HEX programming format.
- Allows user limited communication via RS232 interface to computer development system.

Using the DATA I/O Model 19 with UniPak/Unipak2

RS232 Serial Interface

Prior to powering up the DATA I/O it is important that the RS232 interface be connected to the host computer correctly. The correct interface is described below:



Turn on the DATA I/O programmer. After it finishes its self check routine, type the following key sequence on the DATA I/O keyboard:

- <LOAD> - Select device type
- <SELECT>
- <D1>
- <START> - Prepares DATA I/O to receive data via RS232 interface port

The DATA I/O Model 19 is now ready to accept data from the computer development system. After the file is entered, the data may be reviewed on the terminal by typing the following on the DATA I/O:

- <KEYBOARD>
- <ENTER> - Sends data to remote terminal

For additional information please refer to the DATA I/O Model 19 users manual.

DATA I/O MODEL 29A WITH UNIPAK/UNIPAK2

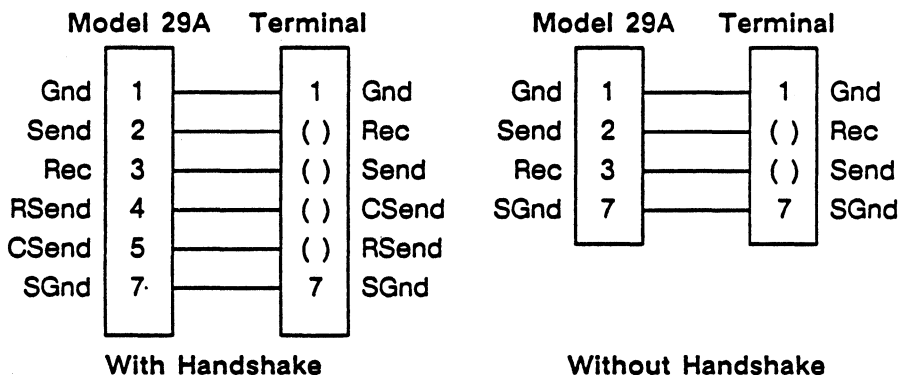
Key Features

- Accepts PLE HEX programming format.
- Allows interactive communication via RS232 interface to computer development system.

Using the DATA I/O Model 29A with UniPak/Unipak2

RS232 Serial Interface

Prior to powering up the DATA I/O it is important that the RS232 interface be connected to the host computer correctly. The correct interface is described below:



Turn on the DATA I/O programmer. After it finishes its self check routine, type the following key sequence on the DATA I/O keyboard:

<COPY>
<DEVICE>
<RAM>
<FFCC> - Enter family and pin code

- <SELECT>
- <F7> - Configure for HEX input format

- <COPY>
- <PORT>
- <RAM> - Prepares DATA I/O to receive data via RS232 interface port

The DATA I/O Model 29A is now ready to accept data from the computer development system. After the file is entered, the data may be reviewed on the terminal by typing the following on the DATA I/O:

- <COPY>
- <PORT>
- <RAM> - Send data to remote terminal

Interactive communication may be achieved by typing on the DATA I/O:

- <SELECT>
- <FB> - Enable output port for interactive communication

For additional information please refer to the DATA I/O Model 29A users manual.

DIGELEC MODEL UP-803 WITH FAM 12

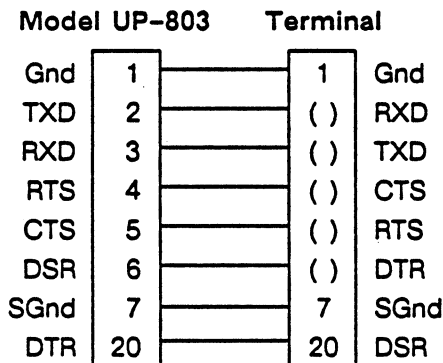
Key Features

- Accepts PLE HEX programming format.
- Allows limited communication via RS232 interface to computer development system.

Using the DIGELEC Model UP-803 with FAM 12

RS232 Serial Interface

Prior to powering up the DIGELEC it is important that the RS232 interface be connected to the host computer correctly. The correct interface is described below:



Insure that the lab switch is set, the serial communication switch setting is used, matching baud rates are used, and ASCII HEX format is specified.

PROM/PLE Programmer Information

Turn on the DIGELEC programmer. Prepare the DATA TRANSFER function of the UP-803 as follows:

SOURCE : SERIAL INPUT
DESTINATION : RAM
DESTINATION INITIAL ADDRESS : XXXX
BLOCK LENGTH : YYYY (FFFF if unknown)
Press the <EXECUTE> key on the UP-803.

The DIGELEC Model UP-803 is now ready to accept data from the computer development system. After the file is entered, the data may be reviewed on the remote terminal by selecting the following on the DIGELEC:

SOURCE : RAM
DESTINATION : SERIAL OUTPUT
Press the <EXECUTE> key on the UP-803.

For additional information on the use of the DIGELEC Model UP-803 please refer to the users manual.

KONTRON MODEL MPP-80S

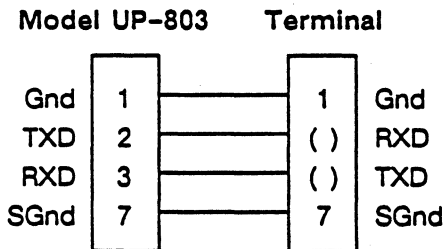
Key Features

- Accepts PLE Intel HEX programming format.
- Allows limited communication via RS232 interface to computer development system.

Using the KONTRON Model MPP-80S

RS232 Serial Interface

Prior to powering up the KONTRON it is important that the RS232 interface be connected to the host computer correctly. The correct interface is described below:



The baud rate, parity, and number of start/stop bits should be the same for the programmer and terminal.

Turn on the KONTRON programmer. Prepare the DATA TRANSFER function of the MPP-80S by performing the following:

- Select proper device type
- Press the white <IN> key (Select input port)
- Press the grey <A> key (Select port A)
- Enter <40> for Intel HEX format
- Press the grey <ENTER> key

The KONTRON Model MPP-80S is now ready to accept data from the computer development system. After the file is entered, the data may be reviewed on the remote terminal by selecting the following on the KONTRON:

- Press the white <OUT> key (Select output port)
- Press the grey <A> key (Select port A)
- Enter output format desired
- Press the grey <ENTER> key

For additional information on the use of the KONTRON model MPP-80S please refer to the users manual.

PLEASM ERROR MESSAGES

PLEASM detects and reports the following errors encountered while running the program. These include incorrect file naming, syntax errors in the input specifications file, and errors in simulation.

1. Non-existent filename specified in response to the query for the name of the source file.

DISK I/O ERROR - MAYBE WRONG FILENAME ???

2. Open file named in response to query for the name of the output file.

DISK I/O ERROR - MAYBE WRONG FILENAME ???

3. Input file size in number of characters exceeds the maximum dimensions specified in the initialization subroutine.

TOO MANY CHARACTERS IN INPUT FILE

This will probably lead to a run-time error with an output message appropriate to your system.

4. Keyword FUNCTION TABLE missing in input specifications.

*** KEYWORD "FUNCTION TABLE" MISSING. ASSEMBLY TERMINATED

5. Keyword DESCRIPTION missing in input specifications.

*** KEYWORD "DESCRIPTION" MISSING. ASSEMBLY TERMINATED

6. Invalid PLE name in line 1 of the input specifications.

PLE PART TYPE PLE\$\$\$\$ IS INCORRECT

7. Number of pin names specified in the address list exceeds the number of input pins available in the PLE being used.

*** TOO MANY PIN NAMES IN INPUT PIN LIST ***

8. Number of pin names specified in the data list exceeds the number of output pins available in the PLE being used.

*** TOO MANY PIN NAMES IN OUTPUT PIN LIST ***

9. A pin name specified in the equations or in the function table pin list does not match any of the names declared in the address and data pin lists.

ERROR SYMBOL = \$\$\$\$\$\$\$

10. The "SIMULATE" option has been invoked without a function table being present in the input specifications.

FUNCTION TABLE MUST BE SUPPLIED IN ORDER TO PERFORM SIMULATION

11. The number of pin names in the function table pin list exceeds the number of pins being used in the PLE.

*** TOO MANY PIN NAMES IN FUNCTION TABLE PIN LIST ***

12. A symbol other than H(high), L(low), or X(don't care) has been entered in the function table.

ERROR SYMBOL **\$** IN LINE \$\$\$ OF FUNCTION TABLE

13. Simulation error caused by an entry in the function table (expected) not agreeing with that evaluated from the Boolean equations (actual).

FUNCTION TABLE ERROR IN LINE \$\$\$ PIN = \$\$\$\$\$\$\$\$ EXPECTED \$ ACTUAL \$

The offending line in the function table is then printed out with a question mark in place of the incorrect entry.

PLEASM Error Messages

14. Overall count of function table errors if there are one or more simulation errors.

ERRORS IN FUNCTION TABLE = \$\$\$\$

15. PLEASM could not generate HEX programming formats for the PLE being used.

PLEASM DOES NOT SUPPLY HEX PROGRAMMING FORMAT FOR \$\$\$\$ BY \$\$ PLE DEVICES

Appendix D: PLE Design File Library

File Name	Page No.	Title	PLE Type
P5000.PLE	4-4	Basic Gates	PLE5P8
P5001.PLE	4-6	Memory Address Decoder	PLE8P8
P5002.PLE	4-8	6-bit True/Invert and Clear/Set Logic Function	PLE8P8
P5003.PLE	4-10	Expandable 3-to-8 Demultiplexer	PLE5P8
P5004.PLE	4-12	Dual 2:1 Multiplexer	PLE10P4
P5005.PLE	4-13	Quad 2:1 Multiplexer with Polarity Control	PLE10P4
P5006.PLE	4-15	Hexadecimal to Seven Segment Decoder	PLE5P8
P5007.PLE	4-18	5-bit Binary to BCD Converter	PLE5P8
P5008.PLE	4-19	4-bit BCD to Gray Code Converter	PLE5P8
P5009.PLE	4-21	4-bit Gray Code to BCD Converter	PLE5P8
P5010.PLE	4-22	8-bit Priority Encoder	PLE8P4
P5011.PLE	4-24	4-bit Magnitude Comparator	PLE8P4
P5012.PLE	4-25	6-bit Magnitude Comparator	PLE12P4
P5013.PLE	4-26	8-bit Barrel Shifter	PLE8P8
P5014.PLE	4-29	4-bit Right Shifter with Programmable Output Polarity	PLE11P4
P5015.PLE	4-32	8-bit Two's Complement Conversion	PLE8P8
P5016.PLE	N/A	Binary to Dual-digit 7 Segment Display Decoder	PLE5P8
P5017.PLE	4-53	Arithmetic Logic Unit	PLE12P8
P5018.PLE	4-41	4-bit Multiplier Look-up Table	PLE8P8
P5019.PLE	4-56	Seven 1-bit Integer Row Partial Products Adder	PLE8P4
P5020.PLE	4-57	Five 2-bit Integer Row Partial Products Adder	PLE10P4
P5021.PLE	4-58	Four 3-bit Integer Row Partial Products Adder	PLE12P8
P5022.PLE	4-59	Three 4-bit Integer Row Partial Products Adder	PLE12P8
P5023.PLE	4-42	Arc Tangent Look-up Table	PLE5P8
P5024.PLE	4-44	Hypotenuse of a Right Triangle Look-up Table	PLE5P8
P5025.PLE	4-47	Perimeter of a Circle Look-up Table	PLE5P8
P5026.PLE	4-50	Period of Oscillation for a Mathematical Pendulum Look-up Table	PLE5P8
P5027.PLE	4-34	A Portion of Timing Generator for PAL Array Programming	PLE5P8
P5028.PLE	4-37	Timing Generator for PAL Security Fuse Programming	PLE5P8
P5029.PLE	N/A	6809 Address Decoder	PLE8P4
P5030.PLE	N/A	4-bit Magnitude Comparator with Polarity Control	PLE9P4
P5031.PLE	N/A	Memory Address Decoder with 16 Chip Enables	PLE5P16
P5032.PLE	N/A	4-to-16 Demultiplexer	PLE6P16

Note 1: Pattern numbers correspond to files found on this disk with the same name and "PLE" extension (i.e., P50XX.PLE).

Note 2: Page number references to the section and page number where the PLE design specification is located in the PLE Handbook (First Edition).

**HELP!!
AND WHERE TO GET IT**

If you have any questions or problems, help is available from MMI in Santa Clara, California at the following telephone number:

(408) 970-9700, Extension 6197

When you call, ask for PLEASM Applications Support. Don't forget to take note of what version of PLEASM you have (Version 1.2H) and state your question or problem as accurately as possible. PLEASM Applications Support is also available from our Field Application Engineers. Consult the PLE Handbook or MMI Databook for the telephone number of your local MMI FAE.

Additional copies of PLEASM can be ordered directly from MMI through the IdeaLogic™ Exchange:

Monolithic Memories, Inc.
IdeaLogic Exchange
Mail Stop 09-07
2175 Mission College Boulevard
Santa Clara, California 95054
(408) 970-9700, Extension 6085

Input/Output

Prior to running the program, the user has access to an I/O package which can be used to specify I/O unit numbers and array sizes without having to recompile the program. The variables associated with this I/O routine are explained below....

CPG(6000)	Maximum number of characters permitted in the input specifications file
LLN(250)	Maximum number of lines permitted in the input specifications file
CLN(250)	Maximum number of characters permitted per line in the input file
CONINP	The Logical Unit Number for <READ>'s from the console
CONOUT	The Logical Unit Number for <WRITE>'s to the console
FILINP	The Logical Unit Number for <READ>'s from a named file
FILOUT	The Logical Unit Number for <WRITE>'s to a named file

The Data Set Reference Numbers for INPUT and OUTPUT files are then assigned as variables at the beginning of the program, as follows....

INPUT	PLE DESIGN SPECIFICATION	RPD=FILINP
INPUT	OPERATION CODES	ROC=CONINP
OUTPUT	ECHO AND TRUTH TABLE	POF=CONOUT
OUTPUT	HEX AND BINARY PROGRAMMING FORMATS	PDF=CONOUT/FILOUT
OUTPUT	PROMPTS AND ERROR MESSAGES	PMS=CONOUT/FILOUT

To perform the customization corresponding to your needs, you need to have access to the source code which means that you should have the Development system. These changes can help reduce the memory requirements and facilitate I/O.

The dimension of arrays CPG, LLN, and CLN can be modified by making the appropriate changes in the subroutine IOINIT located at the top of the source code. This could be done to reduce memory requirements, since the arrays are statically allocated during compilation.

The logical unit numbers for the console are fixed for any given system and must be changed accordingly. For example, the logical unit number for reading from the console with VAX/VMS FORTRAN is 5 and for writing to the console is 6. These numbers are different with Supersoft FORTRAN used on the IBM-PC.

The logical unit numbers for writing and reading to and from files can be normally allocated to be between 0 and 19 excluding those reserved for the console.

PALASM® Software Bug/Enhancement Report

Name: _____ Phone: _____

Company Name: _____

Address: _____

City: _____ State: _____ Zip: _____

Date: ___/___/___

Software Version/Revision No.: _____

Computer/Operating System: _____

PAL® Device Programmers: _____

Observed Bug (Include Software, Hardware, Documentation): _____

Suggested Solution/Enhancement: _____



NO POSTAGE
NECESSARY IF
MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

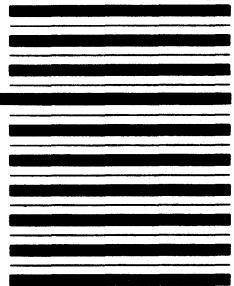
FIRST CLASS PERMIT NO. 523 Santa Clara 95054-1592

Postage will be paid by addressee

MONOLITHIC MEMORIES INC.

2175 MISSION COLLEGE BLVD.
SANTA CLARA, CA 95054-1592

ATTENTION: PLD Software
MAIL-STOP 10-46



Documentation User Response Form

Name: _____ Date: ___/___/___

Company: _____ Revision No.: _____

1. On the following scale, please rate the PALASM[®]2 documentation:

	POOR	AVERAGE	GOOD	
5	4	3	2	1

2. What do you like most about the documentation?

3. What do you dislike most about the documentation?

4. Check the items below that you would like included in the next revision:

- A simple step-by-step tutorial for the first time user.
- A reference section organized alphabetically for the advanced user.
- A quick reference card.
- An introduction to PAL[®] devices.
- More examples.
- More graphics.

5. Please add any suggestions to improve the existing documentation:



NO POSTAGE
NECESSARY IF
MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

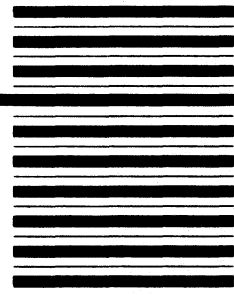
FIRST CLASS PERMIT NO. 523 Santa Clara 95054-1592

Postage will be paid by addressee

MONOLITHIC MEMORIES INC.

2175 MISSION COLLEGE BLVD.
SANTA CLARA, CA 95054-1592

ATTENTION: PLD Software
MAIL-STOP 10-46



Monolithic Memories

Corporate Sales Offices

Americas

Monolithic Memories, Inc.
2175 Mission College Blvd.
Santa Clara, CA 95054-1592
Phone (408) 970-9700
TWX (910) 338-2374
TWX (910) 338-2376
TWX (910) 338-2405
Fax (408) 988-4254

France

Monolithic Memories France S.A.R.L.
Silic 463
F 94613 Rungis Cedex
France
Phone (33)-1-46-87-34-62
Telex 202146
Fax (33)-1-45-60-57-25

Japan

Monolithic Memories Japan KK
5-17-9 Shinjuku-Ku
Shinjuku
Tokyo 160
Japan
Phone 81-3-207-3131
Telex 232-3390 MMIKKJ
Fax 81-3-207-3130

United Kingdom

Monolithic Memories, Ltd.
Monolithic House
1 Queens Road
Farnborough, Hants
England GU146DJ
Phone 0252-517431
Telex 858051 MONO UKG
Fax 44-252-521041

Singapore

Monolithic Memories Singapore Pte., Ltd.
19 Kepple Road 11-06
Jit Poh Building
Singapore 0208
Phone 65-2257544
Telex RS55650 MMI RS
Fax 2246113

Germany

Monolithic Memories, GmbH
Mauerkircherstr 4
8000 Munich 80
West Germany
Phone 49-89-984961
Telex 5-24385 MONO D
Fax 983162

Hong Kong

Monolithic Memories (HK) Ltd.
Room 1901, Henan Building
90-92 Jaffe Road
Wanchai
Hong Kong
Phone (852)5-201-838
Fax (852)5-202-186

Domestic Sales Offices

Monolithic Memories, Inc.

4040 Moorpark Avenue, No. 216
San Jose, CA 95117
(408) 249-7766

Monolithic Memories, Inc.

1800 East McFadden, No. 110
Santa Ana, CA 92705
(714) 543-8664

Monolithic Memories, Inc.

One Energy Center, No. 250
300 East Shuman Blvd.
Naperville, IL 60566
(312) 961-9200

Monolithic Memories, Inc.

12801 N. Central Exp., Suite No. 530, 35
Dallas, TX 75243
(214) 991-1491

Monolithic Memories, Inc.

40 Speen Street
Framingham, MA 01701
(617) 875-7373

Monolithic Memories, Inc.

280 Regency Ridge, Suite 1000
Dayton, OH 45459
(513) 439-0470

Monolithic Memories, Inc.

5 Greentree Centre, Suite 204
Marlton, NJ 08053
(609) 985-1444

Monolithic Memories, Inc.

6575 The Corners Pkw., Suite 304
Norcross, GA 30092
(404) 447-0006

Other Technical Support Offices

Canoga Park, CA — (818) 341-7257

Technical Application Hotline: 800-222-9323

Applications for any integrated circuits contained in this publication are for illustration purposes only and Monolithic Memories makes no representation or warranty that such applications will be suitable for the use specified without further testing or modification. Devices sold by Monolithic Memories are covered by the warranty and patent indemnification provisions appearing in its Terms of Sale only. Monolithic Memories makes no warranty, express, statutory, implied, or by description regarding the information set forth herein or regarding the freedom of the described devices from patent infringement, in any described application or otherwise and no licenses of any kind are granted or implied under any patent or other right. Monolithic Memories makes no warranty of merchantability or fitness for any purposes. Monolithic Memories reserves the right to discontinue production and change specifications and prices at any time and without notice. Monolithic Memories products are intended for use in normal applications. Applications requiring unusual environmental requirements, or reliability applications, such as medical life-support or life-sustaining equipment are specifically **NOT** recommended without written Monolithic Memories' acceptance and additional processing for such applications.

Copyright © 1987 Monolithic Memories, Inc. 2175 Mission College Blvd., Santa Clara, CA 95054-1592. All rights reserved. The material contained herein may not be reproduced, adapted, merged, translated, stored, or used without the prior written consent of the copyright owner.