OPALjr Software

Version 2.0
April 1993

OPALjr MANUAL
TABLE OF CONTENTS - NOTE: This version of the manual does not include the v2.01 corrections. If you require a printed manual with the latest drawings, examples, and corrections, please contact our Applications Group and request a full copy of OPALjr with manual.

1.0 INTRODUCTION
- 1.1 THE OPALjr PACKAGE
- 1.2 HOW OPAL WORKS
- 1.3 THE OPAL CONCEPT

2.0 DOS GRAPHICAL SHELL
- 2.1 COMMAND LINE
- 2.2 MOUSE OPERATIONS
- 2.3 KEYBOARD
- 2.4 HELP MENU
- 2.5 THE VIEW MODULES
- 2.6 MODULES MENU
- 2.7 UTILITIES MENU

3.0 WINDOWS GRAPHICAL SHELL
- 3.1 MOUSE OPERATIONS
- 3.2 HELP MENU
- 3.3 VIEW MODULES
- 3.4 MODULES MENU
- 3.5 UTILITIES MENU
- 3.6 THE WINDOWS MENU

4.0 OPALjr DESIGN MODULES

- 4.1 EQN2JED (Equations to JEDEC)
- 4.2 PAL2GAL
- 4.3 JED2EQN (JEDEC to Equations)

5.0 DESIGN ELEMENTS IN OPALjr

- 5.1 SIGNAL TYPES
- 5.2 SIGNAL EXTENSION

6.0 DESIGN EXAMPLES IN THE EQN FILE FORMAT

- 6.1 SIX-BIT CASCADABLE SHIFT REGISTER
  - 6.1.1 Design
  - 6.1.2 The Basic Equation File
  - 6.1.3 Creating The JEDEC Map
- 6.2 MORE EXAMPLES IN THE EQN FILE FORMAT
  - 6.2.1 6-Bit Cascadable Shift Register
  - 6.2.2 8-Bit Synchronous Cascadable Counter
  - 6.2.3 10-Bit Synchronous Up/Down Counter
  - 6.2.4 16-Bit to 4-Bit Priority Encoder
  - 6.2.5 16-Bit to 1-Bit Multiplexer

6.2.6 Hexadecimal Decoder for a 7-Segment Display
6.2.7 3-Bit Counter With 7-Segment Display Output
6.2.8 15-Bit Up-Down Counter

7.0 EQN FILE FORMAT DESCRIPTION

7.1 VEC FILE
7.2 EQN FILE

8.0 APPENDIX

8.1 DEVICES SUPPORTED
8.1.1 Buried Registers

8.2 USEFUL CONVERSIONS
8.2.1 Conversions of Flip-Flops

9.0 ERROR MESSAGES

9.1 COMMON MESSAGES (0000-0099)
9.2 MESSAGES WHILE READING AN EQN FILE (0100-0199)
9.3 MESSAGES WHILE READING A VEC FILE (0300-0399)
9.4 MESSAGES WHILE READING A JEDEC FILE (0400-0499)
9.5 MESSAGES FROM EQN2JED (1400-1599)
9.6 MESSAGES FROM JED2EQN (1800-1999)
9.7 MESSAGES FROM PAL2GAL (2000-2199)
9.8 MESSAGES FROM JED2XPT (3400-3600)

1.0 INTRODUCTION

The OPALjr software package consists of graphical shell environments for both DOS and Windows, three executable modules (EQN2JED.EXE, JED2EQN.EXE and PAL2GAL.EXE), a set of library device files, a set of examples and this on-disk manual.

The example entry files contain Boolean equations and command labels which are written in the EQN file format. The entry files are used by the module EQN2JED to create JEDEC maps which contain the programming data for a target device. By creating your own entry files, you can use the OPAL software package as part of a complete PLD (Programmable Logic Device) design development system.

The OPALjr software disk is ready to run immediately on an IBM PC or compatible system. The INSTALL program should be used to install the software on your hard disk or another floppy disk.

OPALjr is not copy protected, so you may make working copies for all of your personal computer systems. For additional copies of the software and this manual, or to request information regarding the full OPAL package, contact your local National Semiconductor sales representative. For technical assistance or to report software bugs, contact the appropriate office listed below:

Region Telephone

U.S. and Canada 800-272-9959
Europe +44-793-697472 (U.K.)
Asia +852-733-1869 (Hong Kong)
Japan 03-3299-7001

1.1 THE OPALjr PACKAGE

OPALjr is a functional subset of the full OPAL package. It is designed to:

a) provide free low-level entry design support for all NSC low-density PAL and GAL devices

b) demonstrate the syntax and capabilities of the full OPAL PLD development software package, including support for higher-density MAPL devices.

1.2 HOW OPAL WORKS

The ultimate goal of any PLD software package is to create a JEDEC map of the PLD fuse states that can be downloaded to a device programmer. The JEDEC map contains device-dependent data, such as pin assignment, the number of fuses per product term, the type of feedback, and so on.

Traditional PLD software packages require that you know details of the device architecture before you can create a JEDEC map. Historically, this was acceptable, since the complexity and density of the available programmable logic devices was limited. Devices with greater complexity and density are more difficult to utilize effectively and efficiently.

The current generation of PLD software packages strives to be device-independent; a design can be contructed without any knowledge of the device targeted for the design. The designer need not worry about the architecture of the target device. They are free to concentrate on the design of the system.

The Open Programmable Array Language (OPAL) software package is designed to take this a step further. Instead of only providing device independence, we also supply tool independence. All of the file formats supported (OPAL, PLA, EQN, JEDEC) are completely "open" and compatible with recognized software tools such as the industry-standard ESPRESSO logic minimizer. This approach results in a highly modular software system which gives the designer visibility and control over every step of the logic design and implementation process. The designer has the freedom to alter a design at any step of the compilation, and to utilize familiar tools.

1.3 THE OPAL CONCEPT

The OPAL language is a high-end solution which is included only in the full PAL package. This language gives the designer the choice of using state machine language, multi-level Boolean equations, enhanced truth table functions, or any combination of the three. This gives the designer freedom to choose the implementation which best suits the specific design task.

For a center platform, the logic is compiled into the industry standard Berkeley PLA format, which is the input format for the popular Espresso logic minimizer. At this level, the logic is the most compact. Designers with specific optimization techniques can incorporate those techniques into the software by manipulating the PLA data. Two such tools, developed by National Semiconductor, are the "FITMAPL1" and "FITMAPL2" programs, which will take a generic PLA table and partition it into the individual arrays

present on the MAPL1 and 2 product family. FITMAPL1 and FITMAPL2 and the PLA file format are only available with the full OPAL package.

At the lowest level, the logic can be expressed in the standard sum of products (SOP) format. This format can be compiled into the JEDEC map of a specific PLD. No further logic manipulation is done at this level, so designers who want the highest degree of control over the logic processing can enter designs at this level. However, even at this level, OPAL does provide for automatically mapping the logic into a specific PLD via an automatic pin assignment option. This feature is available in both OPAL and OPALjr.

2.0 DOS GRAPHICAL SHELL (OPAL.EXE)

OPAL-DOS requires DOS version of 2.0 or above to run. OPAL-DOS is a very powerful graphical shell that can be utilized to perform any operation in PLD design provided with OPAL software.

2.1 COMMAND LINE

Synopsis:
opal [filename.ext][options]

where filename is usually the OPAL entry file (optional).

Options:

-h or -? Display help screen.
-m Use minimum shell memory for swapping. This is useful for floppy disk systems.
-n Don't include help file into memory. This is useful for systems with limited memory.
-v Displays version number of the OPAL-DOS shell itself, not the full OPAL software package.
-x Don't show OPAL title screen.

2.2 MOUSE OPERATIONS

The following table shows the different operations which can be performed by the mouse.

OPERATION	MOUSE ACTION
Select a Menu Item	Click on the item
Toggle a check field in a dialogue box	Click on the field
Display contents of a directory in an open file dialog box	Click on the directory name
Perform an action in a Dialog box	Click on the action button
Display Options of a field in a dialog box	Click on the field
Scroll up/down in a display area	Click on the up/down arrow on top of the scroll bar

Move the cursor	Click on the new location
Scroll down one line at a time in the editor screen or a dialog box listing files or options	Click on the down arrow in the bottom of the scroll bar
Quick Scroll.	Point to the highlighted area of the scroll bar and drag the mouse up or down
Mark a block of data	Point to the beginning of the block and drag the mouse to the end
Select a file in a dialog box.	Double click on the filename, or click on the filename and then click OK button
Select an option from a list in a dialog box	Double click on the option
Select a field to modify in a dialog box	Point to the field.
List possible options in a field in a dialog box	Point to the field and click

2.3 KEYBOARD OPERATION

All actions performed by the mouse can be performed by the keyboard. In this section the functions that can be performed by the keyboard are listed along with the keystrokes to press.

OPERATION	KEYSTROKE
Access the Top-Level menu bar	<F10> or <Alt> <highlighted-letter>
Access a submenu	<F10> Then use right and left arrow keys to move to the item then <Enter>
Select a submenu item	<highlighted-letter> or use up/down arrow keys,then <Enter>
Highlight a field in a dialog box	<Tab> or arrow keys
Select an item in a dialog box	Highlight item then <Enter>
Toggle a check field	<Space>
List options of dialog box field	<Alt-L>

Exit a dialog box
without saving changes <Esc>

2.4 THE HELP MENU

The OPAL-DOS shell provides an online help for all menu items and the editor. To access the help on a specific item, highlight that item and press <F1>. A screen containing an explanation of the item and its operation will appear. To exit a help screen, press <Esc>. In the editor screen, <F1> will display a summary of the editor commands. Some Help screens contains some keywords that contain further details about the subject. To access these topics, use the arrow keys to move to the desired keyword and press <Enter> or click on the highlighted keyword.

2.5 THE VIEW MODULES MENU

This menu is very useful during a design phase, when the user is compiling the source file for the first time. This menu contains selections that allow the user to view the different output files of the last executed procedure.

Any of the following files can be quickly opened and displayed if they were an output or an input of the last executed procedure.

EQN File This is the equation file created by PLA2EQN, or JED2EQN.

JEDEC Map This is the JEDEC file created by EQN2JED, or FITMAPL2.

GAL File This file is created by the PAL2GAL or XOR2GAL modules.

LIS File This is the list file created by EQN2JED or EQN2OPL.

LOG File This is the log file created by OPALPLA.

2.6 THE MODULES MENU

Selecting any of these items will invoke a Module Execution dialog box to prompt for options and parameters. Only EQN and JED modules are available in OPALjr.

2.7 UTILITIES MENUS

All of the selections directly correspond to the modules with the same name except for Device Selection. The Device Selection is actually the same as running EQN2JED with -d@selfile option.

The output will be a log file showing all devices capable of accommodating the design. A dialog box will appear to allow the user to input the equation file name, and to specify one of the four device choices or a file that lists the devices the user wishes to select from.

3.0 WINDOWS GRAPHICAL SHELL (OPALWIN.EXE)

This is an MS-Windows 3.x application and will be referred to it as OPAL-WIN. OPAL-WIN gives the user a complete PLD design environment with the power and ease of MS-Windows. It is assumed that the user is already familiar with the Windows environment. Therefore, details on menu, Window and help operations are not provided except in special cases. The user is advised to refer to the Windows User Manual for more detail.

3.1 MOUSE OPERATIONS

OPAL-WIN is compatible with a mouse. The mouse operations in the OPAL-WIN are the same as those in the Windows environment. Refer to the Windows User Manual for more information.

3.2 HELP MENU

The OPAL-DOS shell provides an online help for all menu items and the editor. To access the help on a specific item, highlight that item and press <F1>. A screen containing an explanation of the item and its operation will appear. To exit a help screen, press <Esc>. In the editor screen, <F1> will display a summary of the editor commands. Some Help screens contains some keywords that contain further details about the subject. To access these topics, use the arrow keys to move to the desired keyword and press <Enter> or click on the highlighted keyword.

3.3 VIEW MODULES MENU

This menu is very useful during a design phase, when the user is compiling the source file for the first time. This menu contains items that allow the user to view the different outputs of the last executed procedure.

EQN File This is the Equation file output by PLA2EQN or JED2EQN.

JEDEC Map This is the JEDEC file output by EQN2JED or FITMAPL2.

GAL File This file is output by the PAL2GAL or XOR2GAL.

LIS This is the list file created by EQN2JED or EQN2OPL.

LOG File This is the log file output by most procedures.

3.4 THE MODULES MENU

This menu contains items that will execute any of the main design procedures. Selecting any of these items will invoke a Module Execution dialog box. Only EQN to JED modules are available in OPALjr.

3.5 THE UTILITIES MENU

This menu contains OPAL modules that are not included in the other menus. Selecting any of these items will invoke a module execution dialog box.

3.6 THE WINDOWS MENU

Unlike OPAL-DOS, OPAL-WIN has the ability to open several files at once. This menu contains functions that are designed to manage multiple windows.

4.0 OPALjr DESIGN MODULES

This chapter contains command line descriptions of the OPAL Jr. modules and utilities.

4.1 EQN2JED (Equations to JEDEC)

EQN2JED converts basic sum-of-products Boolean equations to a device-specific JEDEC file.

The JEDEC file contains all the necessary design details which can be downloaded to a device programmer for programming the target PLD. The JEDEC file is fully compatible with JEDEC standard 3B which is supported by industry-standard device programmers.

A variety of information is stored to the .log file. Primarily, it documents the final fitting of the design with a detailed description of how each pin was used and what label it represents. It also includes information on the utilization of the device resources, and a diagram (DIP or PLCC) of the device and the labels assigned to the exterior pins.

Enhanced support for the GAL16V8 and GAL20V8 devices is incorporated. Different GAL modes are evaluated automatically to fit a design into the device. For a design with only combinatorial outputs, it is not possible to use Medium-PAL mode if more than six outputs are I/Os. This is because there are 6 combinatorial I/Os and 2 dedicated outputs in Medium-PAL mode. Instead Small-PAL mode is used. However, it is possible to use Registered-PAL mode to get more than six combinatorial I/Os for such a design. The trade-off is that the clock and OE pins are not used.

EQN2JED has the ability to perform device selection. When supplied with a device selection file (using the -d @selfile option on the command line, or CHIP design @selfile in the equation file), the module will attempt to fit the equation file design into each of the devices listed in the selection file. This is available for all devices, except for the MAPL devices, which have their own fitter.

There are predefined selection files called:
ALL To attempt to fit into any device,
TTL To attempt to fit into any TTL device,
ECL To attempt to fit into any ECL device,
CMOS To attempt to fit into any CMOS device.

The name of any device that fits will be printed to the screen. The fitting status for every device will be written to the log file. Device utilization statistics and pin assignments will also be written to the log file for any device that fits.

Synopsis:
eqn2jed [options] eqnfile

where eqnfile is the equations file. The default extension is .eqn.

Options:

-b
Select old versions of the 16V8 and 20V8 devices.

-c
Enable the security fuse on the device.

-d device
Override the device name in equation file.

-d @selfile
If the argument following the -d flag begins with the character @, the remaining text is interpreted as a device selection filename, as discussed above. There are predefined selectors called ALL, TTL, ECL, and CMOS. The selfile has a default extension of .sel.

-e errfile
Specify the error log file for error messages instead of directing them to the standard error stream. This is provided since neither DOS nor UNIX C-Shell provides a way to redirect the errors. Default extension is .err.

-f
Turn on the automatic pin assignment switch. The JEDEC file created reflects the pin assignment if successful. The pin list in the original EQN file (if any) is no longer valid. The pin assignment is indicated by the pin out diagram in the log file that is created by EQN2JED. If the -f option fails to assign the pins, it does not mean that there is no solution. In such cases, the designer may have to verify if the design will fit into the PLD and then manually assign the pins before compiling with EQN2JED again.

-g logfile
Specify a different name for the log file; the default extension is .log.

-j
Select JEDEC PLCC package for chip diagram in the log file. For ECL parts, a 24-pin Quad Cerpak is selected.

-k
Select non-JEDEC PLCC package for chip diagram in log file. For ECL parts, a 24-pin Quad Cerpak is selected.

-l listfile
Specify the list file name. The list file contains a listing of the equation source file with all warning and error messages at the corresponding lines. The default extension is .lis.

-m
Produce a full JEDEC map as output which programs every location in the device. Without this option, unused sections of the device are not programmed, which results in a smaller JEDEC map.

-n
The old log file will be deleted. This is often desirable since the program appends to the log file, if it exists, and it can become quite large after a number of successful calls to the module.

-o outfile
Specify the output JEDEC file name. If no -o option is specified, the input file name is used with the extension .jed.

-r
Retain repeating product terms for PALs.

-s
Shuts off diagnostic messages.

-u ues
Specify the User Electronic Signature as ues for GAL devices in ASCII format (8 bits per character). If only hexadecimal characters are used, it will use a hexadecimal representation (4 bits per character).

-v vecfile
Specify the input test vector file name. The vector file must conform to the JEDEC standard for defining vectors. The vector file is appended to the JEDEC file. Each individual vector is tested to ensure that vector characters are valid for the pins they are assigned to. This feature ensures that vectors are defined correctly, however, it does not perform any simulation to ensure that the vector has the correct output response for the given inputs. The default extension is .vec.

Example:
eqn2jed -f -s -dgal16v8 6_shift

The above command invokes the EQN2JED module. The input is the EQN file called 6_SHIFT.EQN. The output file will be 6_SHIFT.JED, by default. As specified by the options, pin declaration and assignment will be done automatically and no diagnostic messages will be displayed on the screen. Also, the device specified in the equation file will be overridden by the GAL16V8.

4.2 PAL2GAL

The PAL2GAL module converts a PAL JEDEC file into a GAL JEDEC file. PAL2GAL first checks to ensure that there is a pin-to-pin match between the PAL and a supported GAL before it proceeds to do the conversion.

PAL functions that are replaceable by a GAL16V8 are:

PAL10H8 PAL10L8 PAL10P8 PAL12H6
PAL12L6 PAL12P6 PAL14H4 PAL14L4
PAL14P4 PAL16H2 PAL16H8 PAL16L2
PAL16L8 PAL16P2 PAL16P8 PAL16R4
PAL16R6 PAL16R8 PAL16RP4 PAL16RP6
PAL16RP8

PAL functions that are replaceable by a GAL20V8 are:
PAL14H8 PAL14L8 PAL14P8 PAL16H6 PAL16L6
PAL16P6 PAL18H4 PAL18L4 PAL18P4 PAL20H2
PAL20H8 PAL20L2 PAL20L8 PAL20P2 PAL20P8
PAL20R4 PAL20R6 PAL20R8 PAL20RP4 PAL20RP6
PAL20RP8

PAL functions that are replaceable by a GAL22V10 are:

PAL12H10 PAL12L10 PAL12P10
PAL14H8 PAL14L8 PAL14P8
PAL16H6 PAL16L6 PAL16P6
PAL18H4 PAL18L4 PAL18P4
PAL20H2 PAL20H8 PAL20L2
PAL20L8 PAL20P2 PAL20P8
PAL20H10 PAL20L10 PAL20P10
PAL20R4 PAL20R6 PAL20R8

PAL20RP4 PAL20RP6 PAL20RP8
PAL22V10

Note that PALs with registered outputs in the above list have a dedicated OE for tristate control (pin 13). When the design is converted to the GAL22V10, a product term tristate will be used instead; this may produce some timing differences.

Many devices can be converted to either a GAL20V8 or a GAL22V10. The option \EC-g device is provided to specify which device should be used, although it will default to the GAL20V8 if the option is not used.

Synopsis:

pal2gal [options] palfile

where palfile is the PAL JEDEC file to be converted. The default extension is .jed.

Options:

-b
Debug mode to show intermediate diagnostics. This option will show all of the other modules which are executed, as well as keep all intermediate files.

-d device
Specify the PAL device name. Override the device name in PAL JEDEC file (if any).

-e errfile
Specify the error log file for error messages instead of directing them to the standard error stream. This is provided since neither DOS nor UNIX C-Shell provides a way to redirect the errors.

-g device
Specify a GAL device to convert to. This option should be used whenever converting to a GAL22V10.

-o galfile
Specify the output GAL JEDEC file. If the -o option is not specified, the PAL JEDEC filename is used with the extension of .gjd.

-s
Shuts off diagnostic messages.

-v
Do not include vectors in GAL JEDEC file. If this option is not used, then any vectors in the PAL JEDEC file are copied into the GAL JEDEC file. The vectors must conform to the JEDEC standard for defining vectors. The vector file is appended to the GAL JEDEC file. Note that no check is done to ensure the correctness of the vectors.

-u ues Specify the User Electronic Signature as \ECues\EE for GAL devices in ASCII format (8 bits per character). If only hexadecimal characters are used, it will use a hexadecimal representation (4 bits per character).

Example:

pal2gal -d pal12h6 gates

This command invokes the PAL2GAL module. The input file is the PAL JEDEC file gates.jed. the output file will be the GAL JEDEC file with the default name gates.gjd.

the device PAL12H6 specified with the -d option on the command line will override any device name (if any) in the PAL JEDEC file.

4.3 JED2EQN

The JED2EQN module will disassemble a JEDEC file into the corresponding basic Boolean equations.

The labels used in the basic Boolean equations created by JED2EQN contain the pin number preceded by the type of signal. Observing the labels makes it easy to determine if the pin is used as a dedicated input, combinatorial or registered output, and whether or not the output is used as feedback into the device.

Synopsis:

jed2eqn [options] jedfile

where jedfile is the JEDEC file to be disassembled. The default extension is .jed.

It is recommended that the -d option be used to avoid potential conflicts between the device name and any comments which may exist in the JEDEC file. However, if the JEDEC file was created by EQN2JED, this option does not need to be specified. JED2EQN finds this device name in the JEDEC file by searching the header section for the strings PAL, GAL, or MAPL.

JED2EQN attempts to supply meaningful label names for the pins. It does this by examining the JEDEC file for user-supplied labels in the NOTE PINS and NOTE NODES statements. The format for such statements is as follows:

'NOTE' <'PINS' | 'NODES'> [<label> [':' <pin number> '*'

Only pins which are used need to be supplied. Note that in a JEDEC file, any line beginning with N and ending with a * is a comment.

If the number of labels is equivalent to the number of pins on the device and none of the labels have pin numbers assigned to them, it is assumed that the labels have consecutive pin numbers beginning from 1.

Note that JED2EQN does not translate test vectors within the JEDEC file.

Options:

-a
Select the alternate operator set for output:

! = NOT, & = AND, # = OR and $ = XOR.

-d device
Specify the device name. This will override the device name that is in the JEDEC file. There is no standard defined for placing the device name in the JEDEC file.

-e errfile
Specify the error log file for error messages instead of directing them to the standard error stream. This is provided since neither DOS nor UNIX C-Shell provides a way to redirect the errors. Default extension is .err.

-o eqnfile

Specify the output Boolean equations file. If the -o option is not specified, the jedfile name is used with the extension of .eqn.

-s
Shuts off all diagnostic messages.

Example:

jed2eqn -dpal12h6 gates

This command invokes the JED2EQN module. The input is the JED file called gates.jed. The output file will default to gates.eqn. And, the device in the equation file will be
PAL12H6.

5.0 DESIGN ELEMENTS IN OPALjr

In the OPAL design language, all signals must be predefined before they can be used.

5.1 SIGNAL TYPES

There are five types of signals supported by OPAL:

1. Input
2. Output
3. Feedback
4. Node
5. Statebit(not supported by OPALjr.)

Inputs define the signals that go into a design while outputs define the signals that leave a design. Both inputs and outputs are unidirectional signals.

Feedbacks define signals that are both inputs and outputs (bidirectional).

NODE is mainly used for the Input Logic Macro Cells (ILMC) structure of the GAL6001. The ILMC is provided to store a fast changing input for later use by the device. Therefore, the node is an input to the AND array of the device. A feedback is an input to the AND array, and at the same time it is also an output of the OR array. A register will be between the node and the input pin and a register can be between a feedback and the OR array.

5.2 SIGNAL EXTENSIONS

If a designer, for example, defines o1 as output of JK flip-flop he/she can define in the equation block

o1 := i1 * i2 + i3.j.

In this case the software will divide the input functions into j input and k input. It will also use default clock, and reset or preset. On the other hand, if the designer wishes to have complete control over the design he/she can use signal extensions to explicitly define the functions that control the different inputs of the JK flip-flop.

For example:
o1.j = defines the function to j input of the flip-flop.
o1.k = defines the function to the k input of the flip-flop.
o1.c = defines the function of the clock input of the flip-flop.

o1.re = defines the function to the reset input of the flip-flop.
o1.oe = defines the function to the tri-state control of
the output.

Recognized Dot Extensions

EXTENSION	FUNCTION
AP	Asynchronous Register Preset
AR	Asynchronous Register Reset
C	Clock Input
CE	Clock Enable for DE Flip-flop
D	D Input to D or DE Flip-flop
FB	Pre-buffer Feedback
J	J Input to JK Flip-flop
K	K Input to JK Flip-flop
LE	Latch Enable Control for Latch
OE	Output Enable
PIN	Post-buffer Feedback
PR	Register Preset
RE	Register Reset
REG	Input to Generic Flip-flop
SP	Synchronous Register Preset
SR	Synchronous Register Reset

Register Types Versus Valid Extensions

REGISTER TYPE	VALID EXTENSIONS
COM	OE, FB, PIN
REG	OE, REG, C, PR, RE, AP, AR, SP, SR, FB, PIN
JK	OE, J, K, C, PR, RE, AP, AR, SP, SR, FB, PIN
D	OE, D, C, PR, RE, AP, AR, SP, SR, FB, PIN
DE	OE, D, CE, C, PR, RE, AP, AR, SP, SR, FB, PIN
LCH	OE, LE, C, PR, RE, AP, AR, SP, SR, FB, PIN

6.0 DESIGN EXAMPLES IN THE EQN FORMAT

Summary of Examples and Their Features

Filename (*.eqn)	Section Reference	Features
6_shift	6.1	Low Complexity CombinatorialSignals Output Enables Common Subexpressions Device and Pin Independent
8_count	6.2.2	Low Complexity Registered and CombinatorialSignals XOR Equations Mixed Polarities PAL20X8 Device
10_count	6.2.3	Medium Complexity

Registered and
Combinatorial Signals
Output Enables
XOR Equations
Mixed Polarities
PAL20X10 Device

16_4_pri 6.2.4 Low Complexity
GAL20V8 Device

16_1_mux 6.2.5 Low Complexity
PAL20C1 Device

hex_7seg 6.2.6 Medium Complexity
Mixed Polarities
Output Enables
PAL16L8 Device

3b_7seg 6.2.7 Medium Complexity
Common Subexpressions
Registered and
Combinatorial Signals
GAL22V10 Device

15_count 6.2.8 High Complexity
UES Specification
JK Type Registers
MAPL128 Device

6.1 SIX-BIT CASCADABLE SHIFT REGISTER

This example will show a few of the software features and the usage of the module, EQN2JED.

```
CHIP 6_shift unknown
      ; Device not selected yet

sr   sl d5 d4 d3 d2 d1 d0
            ; Declare order of inputs
rilo q5 q4 q3 q2 q1 q0 liro
            ; Declare order of outputs

      @define Hold   "/sr * /sl"
      @define Right  " sr * /sl"
      @define Left   "/sr *  sl"
      @define Load   " sr *  sl"

EQUATIONS

/q0 := /q0 * Hold  +  /q1   * Right  +  /liro * Left  +  /d0 * Load
/q1 := /q1 * Hold  +  /q2   * Right  +  /q0   * Left  +  /d1 * Load
/q2 := /q2 * Hold  +  /q3   * Right  +  /q1   * Left  +  /d2 * Load
/q3 := /q3 * Hold  +  /q4   * Right  +  /q2   * Left  +  /d3 * Load
/q4 := /q4 * Hold  +  /q5   * Right  +  /q3   * Left  +  /d4 * Load
/q5 := /q5 * Hold  +  /rilo * Right  +  /q4   * Left  +  /d5 * Load

/liro = /q0
liro.oe = Right

/rilo = /q5
```

```
rilo.oe = Left


Basic Equation File for the Six-bit Shift Register PAL16R6
EQN2JED - Boolean Equations to JEDEC file assembler
Copyright (R) National Semiconductor Corporation 1990,1991
Assembled from "6_shift.eqn". Date: 4-3-91


      6-Bit Cascadable Shift Register
      National Semiconductor OPAL Example
*
QF2048*QP20*F0*
L0000
01111011111111111111111111111111
11111111111111101111111111111111*
L0256
10111010111111111111111111111111
01111011111011111111111111111111
10110111111111101111111111111111
01110111111111111111111110111111*
L0512
10111011111011111111111111111111
01111011111111111110111111111111
10110110111111111111111111111111
01110111111111111111101111111111*
L0768
10111011111111101111111111111111
01111010111111111111111111111111
10100111111111111111111111111111
01110111111111111111111111111011*
L1024
10111011111111111110111111111111
01111011111111111111111011111111
10110111111011111111111111111111
01110111111111111101111111111111*
L1280
10111011111111111111111011111111
01111011111111111111111101111
10110111111111111110111111111111
01110111111110111111111111111111*
L1536
10111011111111111111111111101111
01111011111111111111111111111110
10110111111111111111111011111111
01110111101111111111111111111111*
L1792
10110111111111111111111111111111
11111111111111111111111111101111*
C6743*

JEDEC Map File for a Six-bit Shift Register on the PAL16R6



Behavior of the Shift Register

SR      SL        ACTION
0       0         Hold current value
0       1         Shift left
1       0         Shift right
1       1         Load from D5-D0
```

6.1.1 Design

The shift register example shown below has four modes of operation (hold, shift right, shift left, or load) controlled by two input signals (SR, SL). It has six registered outputs which show the current register value, and two I/O pins for the shift in and shift out, depending on the direction of shift. It also has six input pins so that a new value can be loaded into the register.The operation of the shift register is controlled by the input signals SR and SL. Table10-1 shows the behavior of the design.

6.1.2 The Basic Equation File

With these requirements in mind, the basic sum-of-products Boolean equation file can be written.

The source file begins with the design header, which usually describes the design and its author. The header is different from comments in two ways. First, the header will be reproduced in the JEDEC file, whereas comments are ignored. Secondly, the header is any text prior to the CHIP keyword, and comments are denoted by either beginning with a semicolon (;) or enclosed within curly braces ({ and }).

NOTE: Don't use the keyword chip in the header or comments.--

The next portion is the declaration block, which begins with the CHIP keyword, then the name of the design (for documentation purposes), followed by the device name, if any. In this case, a device has not yet been selected, so unknown is entered so as not to forget to specify the -d devicename option to EQN2JED later on. Note that the comment on this line will not be passed to the JEDEC file.

The declaration block specifies a required device and/or assigns labels to specific pins. The next portion lists the pins which are used. By including all of the pins, including GND and VCC, in the proper order, we can force signals to specific pins. In this case, not all pins were included (e.g., CLK, GND, and VCC), so EQN2JED treats it as a partial pin list. A partial pin list specifies the order in which you would like the pins to appear; this is useful for buses or groups of related signals, such as in this example.

The label @define is used for declaring common subexpressions. In this case, it would be unclear and tedious to type /sr * /s1 in every equation. Using the @define statement as shown, we only need to type Hold. Since the @define statement simply replaces the symbol with the supplied string, particular care should be taken with OR-terms, which have higher precedence than AND-terms.

In addition, it is possible to specify a user electronic signature (@ues) for GAL devices, and declare input identifiers to be latched (@ilch) or registered (@ireg) on the GAL6001 within the declaration section. This is discussed in Appendix C and shown in the included examples.

The basic equation block, which starts with the keyword EQUATIONS and continues through the end of the file, is where the design is described. It is described using standard sum-of-products notation (XOR is also valid). The list of operators is described in Section C.5, and is mostly self-explanatory.

Of particular note are the operators = and :=. The former describes a combinatorial (asynchronous) assignment, and the latter is for registered (synchronous with the clock) assignment. Both are used in this design. Registered outputs are necessary since we wish to hold the register value (q5 - q0), and combinatorial I/O pins (for rilo and liro) are used for cascading since we desire an asynchronous signal.

The designer, knowing that this device would probably be programmed onto a small PAL device (which often have active-low outputs), decided to declare the

outputs as active low by placing a / in front of each output equation. If the design were to be programmed on a device with active-high outputs, these symbols would not be included.

6.1.3 Creating The JEDEC Map

The EQN2JED module assembles the basic Boolean equation file, as shown above, and generates a JEDEC map file. The command to execute this is as follows:

eqn2jed [options] eqnfile

The default extension of the eqnfile is .eqn. Many flags and options can also be included to override default values or direct the process (refer to Section 9.4 for more details). In this example, the equation file name is 6_SHIFT.EQN. Since the EQN source file has no device specified, a -d option with a device name must be supplied on the command line. This is because the JEDEC map is device specific. A PAL16R6 is the target device. Pin numbers were also not defined in the original source file, so the -f flag must be specified to enable an automatic pin assignment. Since we supplied a partial pin list, the -f option will attempt to assign pins in the order listed. Therefore, to assemble this example, type in the following:

eqn2jed -f -dpal16r6 6_shift

The output of this process is the standard JEDEC map file that has the extension .jed by default. It is listed above.

In addition to the JEDEC file, this process will append design documentation to the .log file. This file contains the pin types and labels, and it will document the usage of product resources. A listing file with the extension .lis will only be generated if specified with the -l option. It contains a source listing excluding comments. It will also show all warnings and error messages in their appropriate location.

6.2 MORE EXAMPLES IN THE EQN FILE FORMAT

In this section all examples in the distribution disk in the EQN file format will be briefly discussed. The user can view these files directly and experiment with them as suggested here. Table10-2 provides a summary of the examples and their illustrated features.

6.2.1 6-Bit Cascadable Shift Register

This is the best EQN format example provided with the package and should be understood before examining the more complex examples or designing your own files. This example uses both the = (combinatorial) and := (registered) assignments, as well as the output enable signal.This example does not have a device or pin numbers declared. An appropriate device for this design is the PAL16R6.

To create a JEDEC map, type:

eqn2jed -f -d pal16r6 6_shift

6.2.2 8-Bit Synchronous Cascadable Counter

This design is an eight bit up-down counter design which can be cascaded with similar devices. This example illustrates combinatorial and registered signals, mixed signal polarities, and the XOR operator. Pin assignments have already been provided, but can be overridden with the -f option. Note that if the pin assignments are overridden, the pin polarity information will be kept in tact.

The device name is also specified. Specifying a device in the equation file, avoids having to type -d device_name for every compilation. This is useful if the device to be programmed is known.

This example is designed for the PAL20X8. The compiler EQN2JED can be run without any flags: eqn2jed 8_count

To force the design to be placed on another device, simply specify the -d option, which will override any device name in the source file:
eqn2jed -f -d pal20r8 8_count

Executing this command will cause an error declaring that the design will not fit in the selected device. This is due to the fact that the design utilizes nine outputs, and the PAL20R8 only has eight.

6.2.3 10-Bit Synchronous Up/Down Counter
This design is a more complex version of the 8-bit counter. It has more functionality, but lacks a carry out signal due to the limits of the device. The designer also used the mixed polarities slightly differently than the designer of the 8-bit counter.Since both the device and pin list have been specified, this design can be compiled with: eqn2jed 10_count

6.2.4 16-Bit to 4-Bit Priority Encoder
This is a straightforward design. An encoder is a circuit that has a number of inputs, and encodes whichever line is high into its binary representation. A priority encoder also handles the case where multiple signals may be high simultaneously; it encodes the highest one.

The only trick in this design is the equation for Y3, the highest bit. The desired equation would be:

Y3 = D15 + D14 + D13 + D12 + D11 + D10 + D9 + D8

but this would take eight product terms if programmed in that manner, so DeMorgan\EDs Law was applied to the equation so that it would fit into smaller devices. A similar decision was made with the ANY signal.

This design is compiled by: eqn2jed 16_4_pri

6.2.5 16-Bit to 1-Bit Multiplexer
This is the simplest included design example, implementing a 16-bit to 1-bit multiplexer via combinatorial logic.

To compile this design, type: eqn2jed 16_1_mux

6.2.6 Hexadecimal Decoder for a 7-Segment Display
No set of examples is complete without this design for a 7-segment LED driver for hexadecimal inputs. This is a combinatorial design, utilizing mixed polarities and output enable equations.

To run this example, type: eqn2jed hex_7seg

6.2.7 3-Bit Counter With 7-Segment Display Output
This design is a three bit up-down counter which has a set of outputs for a 7-segment LED. This example illustrates additional complexity, combinatorial and registered signals, and the use of @define for common subexpressions. By specifying a specific device in the equation file, we also avoid having to type -d device_name for every compilation.

This example is designed for the GAL22V10. The compiler EQN2JED should be run with the -f flag to assign the pins, since the pins were left unassigned:
eqn2jed -f 3_7seg

To force the design to be placed on another device, simply specify the -d option, which will override any device name in the source file: eqn2jed -f -d pal20r8 3_7seg

Executing this command will cause an error declaring that the design will not fit in the selected device. This is due to the fact that the design utilizes ten outputs, and the PAL20R8 only has eight.

6.2.8 15-Bit Up-Down Counter

This design is a 15-bit up-down counter design which uses four state bits for a state-machine-like design. It shows the use of JK type registers and the @ues directive. Since it is such a large design, the MAPL128 device, which is an ideal part for large state machines, was selected. Note that the pins have been assigned within the file, and that both the state bits and the page bits have been assigned to buried registers. The advantage of buried registers is that external pins are not wasted for unneeded signals.This example was compiled into the EQN file format from an original design using a state machine in the OPAL file format.

Since this design has been specifically written for the MAPL128, neither a device nor automatic pin assignment need to be specified on the command line. In general, if you are using a PLCC packaged device, it may be useful to specify the -j option to get a PLCC instead of a DIP diagram. For devices such as the MAPL128, which are available only in the PLCC package, the default is to use the PLCC diagram.

eqn2jed 15_count

After this command has been executed, examine the log file, which will be named 15_COUNT.LOG in the same directory. From the information presented in the log file, one can see that the design utilized the device\EDs external pins reasonably well, using seven out a possible twenty-six. However, the file also shows that the internal product terms were not utilized nearly as well, making use of only twenty-five percent of the available terms. A designer will often try to fit the final design to many devices, and choose the cheapest or fastest of those available.

7.0 FILE FORMAT DESCRIPTIONS

7.1 VEC FILE

A VEC file is given the default extension of ".vec". It is used as an intermediate file for the design test vectors. The first line in the file is usually "$OPAL VECTORS". Although this line is optional, the _OPAL software will always generate it. This line indicates that the file was translated from a vector block in the OPAL design language.

The second line is the $PIN statement which defines the pin labels for each column of the vector table that is to follow. The $PIN statement begins with the keyword "$PIN", followed _by a number to indicate the number of labels. This number is followed by a list of that many signal labels.

The optional $BUS statements come next. A $BUS statement is used to declare a related group of signals. The $BUS statement begins with the keyword "$BUS", followed by the bus label. Following this is a number to indicate the number of labels in the bus, followed by a list of that many signal labels.

After the $PIN statement (or $BUS statements, if present), there may be an optional VNUM statement which declares the number of test vectors in the file.

An optional $PROBE statement can be used to re-specify the order of the pins/nodes or to specify extra signals to be included in the circuit file .PROBE statement. If the $PROBE statement does not exist, the labels in the $PIN statement will be used.

The remainder of the file is made up of the test vectors. The allowable characters and format of the test vectors is identical to the JEDEC Test

Vector Format.

The VEC file format test vectors are very similar to the JEDEC test vectors with only one difference. This difference is that the JEDEC format requires that every pin on the device PIN _be included in every test vector. For example, there must be 20 test conditions defined within every vector for a 20 pin device.

The VEC file format only includes the symbols thatyou are interested in testing. Two special symbols can be used to repeat test vectors. The $LOOP and $END statements to define a section of test vectors to be repeated.

7.2 EQN FILE

The EQN file format is a simple sum-of-products equation format for describing Boolean logic. It has three components

1. Header
2. Declaration block
3. Basic equation block

There are four reserved keywords that cannot be used as identifiers in an EQN file:

CHIP
EQUATIONS
GLOBAL
STRING

7.2.2 COMMENTS

Comments can be freely inserted anywhere in the source file. There are two types of comments.

1. begins with a semi-colon ";" and ends with the end-of-line character.
2. begins with a "{" and ends with a "}."

7.2.3 HEADER

The header is any text before the "CHIP" keyword, with the exception of comments. This will be passed on to a similar header (if one is available) of the output file.

7.2.4 DECLARATION BLOCK

The declaration block begins with the "CHIP" keyword, followed by a chip name and a device name. Following that is an optional pin list and directives. The declaration block ends when the EQUATIONS keyword is encountered.

<declaration> := CHIP <chip name> <device name>
[<pin list>]
[<directives>]

The "CHIP" keyword indicates the end of the header (if any) and the beginning of the declaration block. The chip name is a name for the design. The device name is the target device for the logic design.

An optional list of pin identifier names follows the device name. This is usually a list of pin names, which can also indicate if a pin is not connected ("NC"), power ("Vcc"), or ground ("GND"). After the physical pins, the buried pins (if there are any on the device, are listed. If a specific pin number should be assigned to a label, it can be specified with the "=" character.

An optional inverting operator ("/" or "!") may precede a label to indicate negative polarity.

If there is no pin list, or a partial pin list, or a pin list (complete or partial) with some of the labels having pin numbers assigned, then it is an error unless an automatic pin assignment feature is invoked to assign the pins automatically.

Note that some devices (GAL6001 and MAPL128) have buried registers, which are feedbacks without an external pin. Any assigned pin numbers greater than the number of external pins on the device are assumed to be buried.

7.2.5 DIRECTIVES

One or more optional directives may follow the pin list. Directives are declared after the pin list and before the "EQUATIONS" keyword. There are five directives being used currently.

1. STRING Defines common sub-expressions to help clarify or shorten the equations in the equations block.

2. @define Equivalent with STRING. Particular care should be taken with the define statement. Using OR or XOR terms in the replacement string may not produce the expected behavior, since @define blindly substitutes the replacement string for the symbol and the OR operator takes precedence over the AND operator. In addition, using an inverting operator ("/" or "!") in front of the @define label will only invert the first symbol, not the entire replacement string

3. @ues Defines a user electronic signature.

4. @ilch Declares identifiers with latched inputs.

5. @ireg Declares identifiers with registered inputs.

7.2.6 BASIC EQUATIONS BLOCK

The basic equations block begins with the "EQUATIONS" keyword and is followed by one or more equations through end of file.

The equations assign the value of an expression to a signal. The equation assigns a value to the righ hand side (RHS) of the assingment operator to the output/feedback signal on the left hand side (LHS) of the assignment operator. Two different assignment operators are used: "=" for combinatorial assignment and "=" for registered assignment.

The LHS of the equation consists of a signal name. This could be followed by an optional signal extension or preceded by an inversion symbol ("/" or "!"). A label that is an input only cannot appear on the LHS of the equation.

The RHS of an equation is a logical expression that evaluates to a single value. The logical expression must be described using the sum-of-products form, but must be reduced. The use of parentheses for grouping is not permitted.

VALID OPERATORS IN AN EQUATION BLOCK

OPERATOR	EXAMPLE	OPERATION	PRECEDENCE
/	/signal	invert	4 (highest)
!	! signal	invert	4
*	A * B	and	3
&	A & B	and	3
|	A | B	or	2
+	A + B	or	2
^	A ^ B	xor	1
$	A $ B	xor	1
:+:	A:+:B	xor	1
=	A = B	assignment	0
:=	A:=B	registered assignment	0 (lowest)

8.0 APPENDIX

8.1 DEVICES SUPPORTED

The following sections list the devices which are supported by the OPAL package. To use these devices within a design, enter the device name exactly as shown below. Shortened forms of the name can be used if there is only one version of that device; for example, 16R8 can be used instead of PAL16R8. Note that some devices have both a PAL and GAL version, so you should specify which device you intend.

EECMOS GAL DEVICES

GAL16V8
GAL16V8A
GAL16V8QS
GAL20V8
GAL20V8A
GAL20V8QS
GAL20RA10
GAL22V10
GAL22CV10
GAL6001

Most GAL16V8 and GAL20V8 devices manufactured prior to March 1991 are slightly different than those currently manufactured by National Semiconductor. When using these older devices, the EQN2JED -B switch should be used. To determine the status of any old devices, contact your National representative.

TTL PAL Devices

PAL16R4
PAL16R6
PAL16L8
PAL16R8

PAL20R4
PAL20R6
PAL20L8
PAL20R8

ECL PAL Devices
PAL1016C4
PAL10016C4
PAL1016P4A
PAL10016P4A
PAL1016P8
PAL10016P8
PAL1016PE8
PAL10016PE8
PAL1016RD8
PAL10016RD8
PAL1016RM4A
PAL10016RM4A

EECMOS MAPL Devices

MAPL128
MAPL144

Other Supported PAL Device Architectures

PAL10H8
PAL10L8
PAL10P8
PAL12H6
PAL12L6
PAL12P6
PAL12H10
PAL12L10
PAL12P10
PAL14H4
PAL14L4
PAL14P4
PAL14H8
PAL14L8
PAL14P8
PAL16C1
PAL16H2
PAL16L2
PAL16P2
PAL16RP4
PAL16H6
PAL16L6
PAL16P6
PAL16RP6
PAL16H8
PAL16P8
PAL16RA8
PAL16RP8
PAL18H4
PAL18L4
PAL18P4
PAL20C1
PAL20H2
PAL20L2
PAL20P2

PAL20X4
PAL20RP4
PAL20RP6
PAL20X8
PAL20H8
PAL20P8
PAL20RP8
PAL20X10
PAL20H10
PAL20L10
PAL20P10
PAL20RA10
PAL22V10

8.1.2 Buried Registers
A number of devices supported by OPAL have buried registers. Whenever an appropriate signal has been specified as buried and there are no pins available, the signal may be assigned to a buried register. In some cases, such as for the page bits on the MAPL devices, the designer may wish to assign a signal to a specific buried register.

Each buried register has been assigned a number which can be treated as if it were a pin within OPAL.The buffered input registers in the MAPL2 devices are used only in double-buffered input mode and refers to the output of the D-register nearer to the external pin.

GAL6001

Buried Registers (SLMC) 7:0
25, 26, 27, 28, 29, 30, 31, 32
Buried Feedback from pins 14:23
33, 34, 35, 36, 37, 38, 39, 40, 41, 42

MAPL128

Buried Registers 0:7
29, 30, 31, 32, 33, 34, 35, 36
Buried Page Bits 0:2
37, 38, 39

MAPL144
Buried Page Bits 0:2
45, 46, 47

8.2 USEFUL CONVERSIONS

8.2.1 Conversion of Flip-Flops
The table below lists forms which might be found in other languages and their conversion into a register-independent OPAL expression.

In addition, if the register type is known and is a DE or JK register, the designer may want to convert a register-independent expression into its register-dependent form for maximal implementation efficiency.

Register Conversions

OPAL EXPRESSION	D/DE FORM	J/K FORM
s := A;	s.d = A; s.ce = 1;	s.j = A;

```
s := A + B*s;       s.d = A;          s.j = A;
                 s.ce = A + !B;     s.k = !(A + B);

s := A + C*!s;                      s.j = A + C;
                                 s.k = !A;

s := A + B*s + C*!s;                s.j = A + C;
                                 s.k = !(A + B);

s := A $ s;                         s.j = A;
                                 s.k = A;

s := (A + B*s) $ s;                 s.j = A;
                                 s.k = A + B;

s := (A + C*!s) $ s;                s.j = A + C;
                                 s.k = A;

s := (A + B*s + C*!s) $ s;          s.j = A + C;
                                 s.k = A + B;
```

Note that the last four equations synthesize a toggle flip-flop from a D flip-flop, so they allow conversions from T to JK registers as well.

For the full generalization, we can say:

```
s := (A + B*s + C*!s) $ D*s;
s.j = A + C;
s.k = !(A + B)*!D + (A + B)*D;
```

which reduces to:

```
s := A*!D + (!A*!B*D + B*!D)*s + (A + C)*!s
s.j = A + C;
s.k = !A*!B*!D + A*D + B*D;
```

In other words, the functions SET, RESET, TOGGLE, and HOLD can be given as:

```
SET    = A*!D + !A*B*C*!D + !A*!B*C*D;
RESET  = !A*!B*!C*!D + !A*B*!C*D;
TOGGLE = !A*!B*C*!D + A*D + !A*B*C*D;
HOLD   = !A*B*!C*!D + !A*!B*!C*D;
```

And from this you can synthesize the equations for any flip-flop, with any polarity of inputs.

9.0 ERROR MESSAGES

There are three types of messages:

1.Error messages, which begin with the character E. These are problems which will cause the program to fail or will result in incorrect output.

2.Warning messages, which begin with the character W. These are messages which alert the designer to possible problems in the design or in operation of the program.

3.Information messages, which begin with the character I. These are messages which give the designer additional information about the processing of the design or operation of the program.

The four digits following the message type identifier are a unique number which indexes additional information in the following error list. In addition, the error number should be specified in any software bug reports.

The typical user will initially create an OPAL file and use the OPAL modules to automatically generate the PLA, EQN, and JEDEC files. Since these files were generated automatically, file format errors should not occur. This class of errors contain the statement, If the file was automatically generated, please contact your representative. If you get one of these errors, and you did not edit the file in any way, you should report the problem to the appropriate National Semiconductor representative listed at the front of the manual.

9.1 COMMON MESSAGES (0000-0099)

E0001: Cannot open file XXX.--
The indicated file XXX could not be opened. If XXX is an input file to read from, check to see if it exists. If it exists, you should check the file protection settings which allow you to access it. If XXX is an output file to write to, possible problems could be a full disk drive, invalid directory path or file protection that will not allow file to be written to. On DOS, you should also check your FILES value in the system file config.sys.

E0002: Unable to allocate memory.--
The program tried to allocate memory dynamically but was unable to do so. This indicates either not enough memory or an internal error. If you are compiling a small to moderate design or are on a machine with ample memory, please contact your representative listed at the front of this manual. Otherwise, attempt the design on a machine with more memory.

E0003: Unable to write to disk.--
An internal program file could not be written to. The disk drive being written to could be full or there may be problems in closing the file. It is possible that you do not have write permission to the file or directory. On a floppy disk, check the write protect tab.

E0004: Invalid/unsupported device name XXX.--
XXX could be a misspelled device name or one that is not supported. Another possible reason is a missing library file for XXX. You should check that XXX is a supported device.

E0006: Wrong library version number #.## for device XXX.--
The version number in the library file for the device XXX is wrong. See I0007 for the expected version number. Try recopying the library files from your original release diskette, particularly if the expected version number is higher than the incorrect library number. If this does not correct the problem, contact your representative.

I0007: Expected library version number is #.## for device XXX.--
The version number in the library file for the device XXX is wrong. The wrong version number is printed by message E0006.

E0010: Unexpected EOF encountered in file XXX.--
An unexpected EOF is encountered while reading the file XXX. This could be caused by a syntax error. Specify the source listing option on the next recompilation, and examine it for more information.

E0012: Unknown option -X on command line.--
The option -X on the command line is not recognized by the program. Check the list of command line options for this program.

E0013: Expect an argument after -X option on command line.--
An argument is expected after the - X option on the command line. Check the list of command line options to see what type of argument this option requires.

E0014: Input file name missing in command line.--

Every program requires the input file to be specified on the command line. Although the extension will default to a certain extension if not specified, the base file name must be given. If you did supply it, check the syntax of your command line against the list of command line options.

E0015: Input file already defined - XXX.--
You already specified a different input file name. The programs cannot process more than one source file at a time. If you did supply only one file name, check the syntax of your command line against the list of command line options.

E0016: All switches must be separated by a space.--
Although some programs allow you to specify a number of options with only a single dash (-), this program does not. Please give each option its own dash and separate all options and arguments by a space.

E0018: Path name is too long.--
The complete file name (including the path) for OPL2PLA exceeds 80 characters.

E0020: The subprogram XXX did not complete execution.--
The subprogram XXX could not be completely executed. Either there is not enough memory to execute the module XXX or error(s) occurred during the execution of the module. If the problem cannot be solved by examining the messages produced by XXX, please contact your representative.

E0021: Invalid/Unsupported "OPTION" "XXX" on command line.--
The command line specified the option OPTION with an illegal argument of XXX. The argument is either invalid or unsupported within the module. You should also check the spelling of the argument.

E0100: Unknown internal error.--
An unknown internal error has occurred. Do not continue to use this program and report the problem to your representative listed at the front of this document. Please save the input files that you were using so that we may more accurately find your problem.

9.2 MESSAGES WHILE READING AN EQN FILE (0100-0199)

E0101: (Line ##): Chip keyword previously declared.--
The chip keyword is declared more than once. Either delete or comment out the undesired line.

E0102: (Line ##): Invalid device name XXX.--
XXX could be a misspelled device name or one that is not supported. Check the list of supported devices, and contact your representative if you suspect a problem.

E0103: (Line ##): @define keyword found before chip keyword.--
All @ directives must be declared in the declaration section. The declaration section is defined as the area following the chip keyword and before the equation keyword.

E0104: (Line ##): @define keyword found after equations keyword.--
All @ directives must be declared in the declaration section. The declaration section is defined as the area following the chip keyword and before the equation keyword.

E0105: (Line ##): Missing chip keyword.--
The chip keyword is missing. The chip and equation keyword are the only keywords that must be present in an EQN file. You may have attempted to use the wrong type of file as input.

E0106: (Line ##): Missing equations keyword.--
The equation keyword is missing. The chip and equation keyword are the only keywords that must be present in an EQN file.

E0107: (Line ##): Too many labels in the pin list or missing equations keyword.--
The number of labels is more than the number of device pins. Another possible reason could be a missing equation keyword. Check your source file and the pin-out diagram for the device in question.

E0108: (Line ##): Invalid character X in identifier YYY.--
An valid identifier consists of one or more alphanumeric, underscore _ or tilde ~ characters. Remove the offending character.

W0109: (Line ##): 0 is an identifier, NOT logic zero (gnd).--
The keyword gnd is used to represent logic zero. The character 0 will be treated like any other identifier. If this is not the desired behavior, replace the 0 with gnd.

W0110: (Line ##): 1 is an identifier, NOT logic one (vcc).--
The keyword vcc is used to represent logic one. The character 1 will be treated like any other identifier. If this is not the desired behavior, replace the 1 with vcc.

E0111: (Line ##): Invalid dot extension XXX.--
The dot extension XXX is either not supported or misspelled. Consult the list of valid dot extensions.

E0112: (Line ##): Invalid assigned pin ## for label XXX.--
The assigned pin ## for label XXX is invalid. The label attributes do not match those of the assigned pin. For example, an active-high output is assigned to an active-low output.

E0113: (Line ##): Incomplete or missing pin list.--
The number of labels in the pin list is less than the number of device pins or the pin list is completely missing. Specify the automatic pin fitting option if you do not wish to assign the pins yourself.

E0114: (Line ##): Keyword XXX used as identifier.--
The keyword XXX is used as an identifier. Either rename the symbol, or check the syntax of your source file.

E0115: (Line ##): @define definition must be enclosed in "s.--
The @define statement begins with the keyword @define followed by a label and a string enclosed in "s. It must also be on a single line.

W0116: (Line ##):Empty @define string for label XXX.--
An empty string is defined for the label XXX. Recheck your design to see if this is what you intended.

E0117: (Line ##): @define statement must end with a ".--
There is a missing " at the end of the defined string. The statement must be on a single line.

E0118: (Line ##): Previous equation ends with XXX. XXX is ignored.--
An unexpected end of equation was found. Check your source file with the EQN file format.

E0119: (Line ##): Unexpected character X ignored.--
An unexpected character X is encountered while reading the EQN file. The character is ignored in an attempt to continue reading the EQN file and catch

as many errors as possible. If this character is in error, remove it from the source file, otherwise check your source file with the EQN file format.--

E0120: (Line ##): Identifier XXX is undefined.--
The identifier XXX is not defined in the declaration section. Either you misspelled the identifier or forgot to declare it.

W0121: (Line ##): Only first label in "@define XXX" is inverted.--
The @define string is directly substituted for the @define label XXX. A NOT operator in front of a @define label inverts the first label in the @define string. It does not perform the inversion of the entire @define string. This is probably not the desired behavior.

E0122: (Line ##): Expecting : after :+. :+ is ignored.--
A semicolon is expected after the string :+ is encountered. The :+: is the exclusive-OR operator in the EQN format. If you did not intend to use the XOR operator, check your syntax against the EQN file format.

E0123: (Line ##): Expecting = or + after :. : is ignored.--
An = or + character is expected after a semicolon : is found. := is the registered assignment operator. :+: is the exclusive-OR operator. If you did not intend either of these operators, check your syntax against the EQN file format.

E0124: (Line ##): Mismatched braces for comments.--
A comment begins with a { and end with a }. The ending brace } is missing. Edit the source file to correct the problem.

E0125: (Line ##): Invalid character X for start of comment.--
Internal error. A ; or { character is expected for the start of a comment. If there is not an obvious error with your source file, please contact your representative.

E0126: (Line ##): UES exceeds MM ASCII (NN HEX) characters for device XXX.--
The UES string for the device XXX cannot be more than MM ASCII characters or more than NN HEX characters. An ASCII character takes up 8 UES bits while a HEX character takes up 4 UES bits. This limitation is device dependent.

E0127: (Line ##): Unexpected EOF encountered.--
An unexpected EOF is encountered while reading the EQN file. This could be caused by a syntax error. Please check your source file against the EQN file format.

E0128: (Line ##): Unassigned labels found and the "-f" option was not used.--
Only some of the labels have been assigned pin numbers in your source file. To assign the others, either do it manually or specify the -f option on the command line.

E0129: Expect a number after XXX keyword.--
A number is expected after XXX keyword which represents the number of labels in the list. XXX is either @ILCH or @IREG. Place the correct number there in accordance with the EQN file format.

W0130: OR/XOR operator in @define statement.--
An OR (+) or XOR (:+:) operator was used in the @define block. The @define statement performs an exact substitution of the defined string for the defined word. No logical grouping is implied or enforced. The designer should be wary of the precedence of AND over OR and XOR after the string has been substituted into the equations.

E0133: Unable to select a device from file XXX.--
When the argument following the -d option begins with @, the string following the @ character is a file containing a list of device names to select from.

The most likely cause of this error message is that the device selection file XXX is empty.

E0134: Pin assignment operator @ is not supported; use =.--
Some equations file formats allow a pin assignment which look like pinName @ P15, which is not supported in our format. Please change this to pinName = 15.

9.3 MESSAGES WHILE READING A VEC FILE (0300-0399)

E0301: Missing $VNUM number.--
A number indicating the number of vectors in the file is expected after the $VNUM keyword. Either delete the optional $VNUM line or insert the correct number of vectors which follow.

W0302: Extra $VNUM ignored.--
More than one $VNUM encountered in the vector file. The first one will be the valid number of vectors. You should delete the incorrect line.

E0303: Missing $PIN number.--
A number indicating the number of labels in the pin list is expected after the $PIN keyword. Count the number of labels and insert the correct number. Please see the documentation of the VEC file format.

E0304: Unknown or unsupported keyword XXX.--
The keyword XXX is either unknown or not supported. Please see the documentation of the VEC file format.

E0305: file_name: Not an intermediate vector file.--
file_name is not an intermediate vector file.

E0306: Incorrect number of labels in $PIN declaration.--
The number of labels does not match the number after the $PIN keyword. Count the number of labels and correct the number specified after the $PIN keyword.

W0307: Vector span more than one line.--
The vector span more than one line. This could indicate a possible error in the vector file. Please consult the VEC file format description.

E0308: Illegal character X in vector.--
There is an illegal character X in the vector. Please consult the VEC file format description.

E0309: Missing $PIN statement in file XXX.--
The $PIN statement is required. Please consult the VEC file format description.

E0310: P field must be defined before any V fields in vector file XXX.--
Since the P field indicates the order of the test conditions in the vectors, it must come before any V fields which defines the vectors. Please consult the VEC file format description.

E0311: Missing pin numbers in P Field of vector file XXX.--
The P field indicates the order of the test conditions in the vectors. The P field begins with a P followed by a list of space-separated numbers equal to the number of pins for the device and ends with a *. The list of numbers in the P field must match the number of pins for the device.

E0312: Invalid character ? in P Field of vector file XXX.--
The P field indicates the order of the test conditions in the vectors. The P field begins with a P followed by a list of space-separated numbers equal to the number of pins for the device and ends with a *. Only numbers are expected in the P Field.

E0313: Missing * to end the P Field in vector file XXX.--
The P field indicates the order of the test conditions in the vectors. The P field begins with a P followed by a list of space-separated numbers equal to the number of pins for the device and ends with a *. The * is required as a delimiter to indicate the end of the P field.

E0314: Missing * to end vector ## of
vector file XXX.--
The V field begins with a V followed by a vector number. This is followed by a list of the test conditions equal to the number of pins for the device. Finally a * is used as a delimiter to end the V field. The * is missing for vector ##.

E0315: Missing $BUS number.--
A number indicating the number of signals in the bus is missing. Please refer to the VEC file format description.

E0316: Unknown/Bad vector format in file XXX.--
The file XXX does not conform to the VEC file format or the JEDEC Test Vector format. Check the file to make sure that is your vector file.

E0317: Missing $LOOP number.--
The number indicating how many times a vector or group of vectors must be iterated through is missing.

E0318: Mismatched $END and $LOOP statements.--
The number of $END and $LOOP keywords in the vector file should be the same.

E0319: Missing $PROBE number.--
A number indicating the number of labels in the probe list is expected after the $PROBE keyword. Count the number of labels in the statement and insert the correct number.

9.4 MESSAGES WHILE READING A JEDEC FILE (0400-0499)

W0401: Missing/invalid pin number for label label in file filename.--
A pin number is expected after the label label in the JEDEC file. This pin number is either missing or invalid for the device.

W0402: Default labels will be used for equations.--
The program attempts to extract label names from the NOTE PINS or NOTE NODES statements in the JEDEC file. If it is not successful for any reason, default labels will be created instead.

W0403: More than one label assigned to pin ## in file XXX.--
The pin number ## has been assigned to more than one label within the JEDEC file XXX. This is illegal.

E0404: Incorrect QF value for XXX. Correct value is ####. --
The QF field specifies the maximum number of fuses in the JEDEC fusemap. The QF value is incorrect for the device XXX, and should be #### for that device. You have probably specified an incorrect device on the command line. If the file and device type were automatically generated, please contact your representative.

E0405: Fuse number in L field is greater than QF value. --
The QF value specifies the maximum number of fuses in the JEDEC fusemap. It is an error if a fuse number in an L field exceeds the QF value. You have probably specified an incorrect device on the command line. If the file and device type were automatically generated, please contact your representative.

E0406: Incorrect QP value for XXX. Correct value is ##.--

The QP field specifies the number of pins for the device XXX. The QP value is incorrect in the JEDEC map, and should be ## for that device. You have probably specified an incorrect device on the command line. If the file and device type were automatically generated, please contact your representative.

E0407: Illegal character X in JEDEC fusemap ignored.--
The illegal character X in the JEDEC fusemap was found and is being ignored. Only 0s and 1s are allowed in the JEDEC fusemap (or L field).

E0408: Illegal macrocell fuse configuration in JEDEC map.--
The macrocell fuse configuration in the JEDEC map is incorrect. Please check the JEDEC file for possible missing macrocell fuses.

E0409: No fuse number found in JEDEC file XXX.--
There is no fuse map information (L fields) found in the JEDEC file. This indicates an incorrect JEDEC file or some non-standard file format. Go back to the program that originally created the JEDEC map.

E0410: Illegal JEDEC field character X (##).--
An illegal JEDEC field character X (decimal value ##) was found in the JEDEC file. See JEDEC Standard 3-A for the valid list of field characters. If you are unable to determine the problem, contact your representative.

E0411: F Field must appear before L Field in JEDEC file.--
The F field indicates what the default fuse state is if the fuse is not specified in the JEDEC file. It must appear before any of the L fields which define the fuse states.

9.5 MESSAGES FROM EQN2JED (1400-1599)
E1401: Missing equations filename on command line.--
The input equations filename is missing on the command line. The default extension is .eqn.

W1402: Both -j and -k options specified. "-X" option ignored.--
You can only have one of the two options.

W1403: UES ues_string on command line is ignored for device XXX.--
The device XXX does not have a User Electronic Signature. The UES string specified with the -u option on the command line is ignored.

E1404: Unable to do automatic pin assignment for device XXX.--
The automatic pin assignment feature does not work for the device XXX. You will have to assign the pins manually or use a special fitter program (such as FITMAPL) to do that.

E1405: P field must be defined before any V fields in vector file XXX.--
Since the P field indicates the order of the test conditions in the vectors, it must come before any V fields which defines the vectors.

E1406: Missing pin numbers in P Field of vector file XXX.--
The P field indicates the order of the test conditions in the vectors. The P field begins with a P followed by a list of space separated numbers equal to the number of pins for the device and ends with a *. The list of numbers in the P field must match the number of pins for the device.

E1407: Invalid character ? in P Field of vector file XXX.--
The P field indicates the order of the test conditions in the vectors. The P field begins with a P followed by a list of space separated numbers equal to the number of pins for the device and ends with a *. Only numbers are expected in the P Field.

E1408: Missing * to end the P Field in vector file XXX.--
The P field indicates the order of the test conditions in the vectors. The P

field begins with a P followed by a list of space separated numbers equal to the number of pins for the device and ends with a *. The * is required as a delimiter to indicate the end of the P field.

E1409: Missing test conditions for vector ## in file XXX.--
The V field begins with a V followed by a vector number. Then comes a list of the test conditions equal to the number of pins for the device. Finally a * is used as a delimiter to end the V field. The number of test conditions for vector ## must not match the number of pins for the device.

E1410: Missing * to end vector ## of vector
file XXX.--
The V field begins with a V followed by a vector number. Then comes a list of the test conditions equal to the number of pins for the device. Finally a * is used as a delimiter to end the V field. The * is missing for vector ##.

W1411: vec_file: vector ##: Invalid character X for pin_type pin ##.--
The test condition X in vector ## is illegal for a pin_type pin. pin_type could be input, output, feedback, unused, power, clock or output enable.

E1413: Pin ## in gal_name is not an I/O pin in Medium-PAL mode.--
In the Medium-PAL mode, pins 12 and 19 in the GAL16V8 (or 15 and 22 for the GAL20V8) are not I/O pins.

I1414: Using Registered-PAL mode with no registered outputs.--
In the registered-PAL mode, pin 1 is the clock pin for the buried registers and pin 11 in the GAL16V8 (or pin 13 in the GAL20V8) is the dedicated OE pin for the registered outputs.

W1415: Clock term ignored for COM output (pin ##).--
A clock equation is defined for the combinatorial output at pin ##. The equation is ignored. This message is for 16RA8 or 20RA10 only.

E1416: XXX term must be high to bypass register for output pin ##.--
Both the preset and reset terms must be high to bypass the register for a combinatorial output.

E1417: Label XXX (pin ##) specified to be both a JK and DE macrocell.--
This is usually caused by mismatched extensions for the label XXX.

E1418: Label XXX (pin ##) cannot be a combinatorial macrocell.--
The macrocell at pin ## cannot be configured as a combinatorial output. This message is for the MAPL128/144.

E1419: Both clock and CE function defined for label XXX (pin ##).--
A sum term clock and sum term CE cannot be declared for the same label XXX in the GAL6001.

E1420: label: Declared as both latched and registered input.--
label is declared to be a latched as well as a registered input.

E1421: Pins M-N must be configured to the same input type.--
For GAL6001 only. The pins M through N must be configured to be the same input type.

E1423: Pin ## is an output/buried node and cannot be an input.--
Pin ## is an output with buried feedback. When the output is disabled, there is no external input into the AND array.

E1424: Too many product terms for output XXX.--
There are not enough product terms in the device at output XXX to fit the given equations. Try specifying the -m flags in OPL2PLA and FITMAPL. In addition, attempt to minimize the design using Espresso.

E1425: Multiple dedicated output enables declared.--
More than one dedicated output enables declared. The device has only one dedicated OE pin.

E1426: Insufficient pin_type pins in device XXX to fit equations.--
The equations requires more pin_type pins than available in the device XXX. pin_type can be input, output or feedback. Either redefine some of your signals (possibly using external wiring for feedbacks), or try fitting to a different device.

E1427: Wrong pin assignment for label XXX (pin ##).--
The label XXX which is assigned to pin ## is assigned wrongly. For example, an active-high output is assigned to an active-low output. Usually this message occurs when the label XXX is assigned manually. However, if the label was assigned automatically using the -f option, there is a possible internal error in EQN2JED. If this is the case, it is advisable to assign the labels manually. If you suspect an internal error, please contact your representative listed at the front of the manual.

E1428: Unable to assign pin_type label XXX.--
The label XXX cannot be assigned to any pin. Either the device has insufficient pin_type pins or there are no matching pins for the label XXX. pin_type can be input, output or feedback.

E1429: Cannot set pin_type pin for label XXX (pin ##) to VCC or GND.--
pin_type is clock or output enable.

E1430: Label XXX (pin ##) is not an output.--
nternal error. The label XXX which is assigned to pin ## is not an output. Please contact your representative listed at the front of this manual.

E1431: Term ## is not a XXX term.--
Internal error (for MAPL parts only). XXX could be reset or output enable. Please contact your representative listed at the front of this manual.

E1432: No more OE terms to disable I/O pin ## used as input only.--
There are no more output enable terms to disable an I/O pin that is used as a dedicated input. This error occurs for the MAPL128 and MAPL144 only where we have 3 OE terms for 12 I/O pins. You could try different assignments for the pins or new equations for the output enables. If you did the pin assignment manually, try letting the OPAL software do it for you.

W1433: No more OE terms to disable unused I/O pin ##.--
In the MAPL128, unused I/O pins are disabled by default rather than left floating provided there are enough OE terms to do it.

E1434: Global XXX equation is invalid.--
The global equation of type XXX is invalid.

E1435: Label XXX (pin ##) is not an input to the AND array.--
Internal error. The label XXX which is assigned to pin ## is not an input to the AND array. Please contact your representative listed at the front of the manual.

E1436: Out of term_type terms for label XXX (pin ##).--
There are more terms of type term_type for label XXX in the design than are available at pin ##. Examine your design and the device which you are trying to fit your design into for requirements and limitations.

E1437: label: No type terms at pin ## in device XXX.--
There are no terms of function type at pin ## in the device XXX for the label. type is either a clock or output enable.

E1438: label: Pin ## is not a dedicated pin_type pin.--
Pin ## (assigned to label) is not a dedicated pin of type pin_type.

E1439: label: No active low function in device XXX.--
The active low dot extension equation for label is not available in device XXX. Either use a different dot extension or a different device.

E1440: Label XXX cannot be assigned to pin ## (pintype).--
The label XXX cannot be assigned to the pin ## of type pin_type. If the pin_type is not given, the pin ## has a configurable macrocell.

E1441: Missing assignment operator in equation for output XXX.--
The assignment operator := or = is missing in the equation for the output XXX.

E1442: Equation for label XXX has already been defined.--
There is more than one equation defined for the same label XXX.

E1443: Multiple clocks defined. Device XXX has one clock pin only.--
There is more than one clock defined in the equations. The device XXX has only one dedicated clock pin.

E1444: The dedicated OE pin ## in device XXX is active low.--
The dedicated output enable pin ## in the device XXX is active low.

W1445: Dedicated OE pin ## is not an input for the term in XXX.OE.--
Although there is a dedicated output enable pin, it cannot be used in the product term for the signal XXX.OE. If that is necessary for the design, try to use an input pin as an output enable and make all of your OE equations dependent upon that pin.

I1446: Output enable is an AND of a dedicated OE pin and an OE product term.--
The output enable for the outputs in a 16RA8 or a 20RA10 is the ANDing of a dedicated output enable pin with a product term. If you do not want to use the OE pin and this message appears, then you may consider connecting the OE pin to VCC in the final design.

E1447: Conflicting register types for label XXX.--
There are two or more equations for label XXX with dot extensions which conflict in the register types. For example, XXX.d and XXX.j conflict.

E1448: Wrong polarity for label XXX (Pin ##).--
The label XXX cannot be assigned to pin ## due to a polarity mismatch.

E1449: Invalid extension(s) ext for type output XXX.--
There are invalid extensions `ext' for the label XXX of type type.

E1450: Label XXX requires an OE, but pin ## does not have one.--
The label XXX needs an output enable due to the design, but the pin to which it has been assigned does not have one. Try assigning it to a different pin after you have looked at the device databook.

W1451: No test condition for label L. Character X used instead. --
No test vector conditions were supplied for the label, so it will be treated as unknown (for inputs) or dont cares (for outputs).

E1452: XXX is expected to be an active-low dedicated OE (pin ##).--
For the device specified, the dedicated output enable pin is active-low. You should define the label XXX to be active-low.

E1453: XXX: Latch input not available in device deviceName.--
Latched inputs were specified on a device that does not supporte them. Either select another device, or remove the lached input designation in the original

design file.

E1454: XXX: Register input not available in device deviceName.--
Registered inputs were specified on a device that does not support them.
Either select another device, or remove the registered input designation in
the original design file.

E1455: Unknown/Bad vector format in file XXX.--
The vector format in the file `XXX' does not conform to either the VEC file
format or the JEDEC test vector format. Please consult Appendicies D and E.

E1456: Equations defined for both XXX (pin ##1) and YYY (node ##2).--
In the GAL6001, equations cannot be defined for both the external output pin
and its corresponding buried (dual) node which shares the same register.

E1457: Same pin assigned to labels 'XXX' and 'YYY'.--

9.6 MESSAGES FROM JED2EQN (1800-1999)

E1801: Missing JEDEC filename on command line.--
The input JEDEC filename is missing on the command line. The default
extension is .jed.

E1802: Illegal input mode for GAL6001.--
The macrocell configuration bits LATCH = 0 and SYN = 1 were specified for an
input or input/output latch macrocell for the GAL6001 device. This is an
invalid input mode on the device.

E1803: Duplicate label XXX found.--
There is more than one instance of the label XXX in the NOTE PIN or NOTE NODE
statements in the JEDEC file.

E1804: Device name not found in file XXX.--
The device name could not be found in the JEDEC file XXX. For the device name
to be recognized automatically, it must begin with PAL, GAL, PLA, or MAPL, as
listed in Appendix G. In addition, it must be prior to the first * character
in the file. You can either specify the device name on the command line, or
edit the JEDEC file to conform to these requirements.

9.7 MESSAGES FROM PAL2GAL (2000-2199)

E2001: Device name not found in file XXX.--
The device name is not found in the JEDEC file XXX. You can either specify
the device name on the command line with the -d option or edit the JEDEC file
by placing the device name at the beginning of the file.

E2002: Unable to convert PAL device XXX.--
This module does not support device XXX. Please refer to Section 6.2 for a
list of PAL devices that can be converted to supported GALs.

E2003: Illegal character X in UES string XXX.--
The UES string specified with the "-u" option on the command line must
consists of alphanumeric characters only.

E2004: Missing PAL JEDEC filename on command line.--
The program expected the PAL JEDEC filename to be specified on the command
line. The default extension is .jed.

E2005: XXX: Invalid/unsupported GAL specified on command line.--
The GAL device name specified with the -g option on the command line is either
invalid or not supported. Please refer to Section 6.2 for a list of supported
devices.

E2006: Cannot convert PALxxxx to GALxxxx.--
The PAL device PALxxxx cannot be converted to the GAL device GALxxxx. Please refer to Section 6.2 for the correct GAL to convert to.

9.8 MESSAGES FROM JED2XPT (3400-3600)

E3401: Missing JEDEC filename on command line.--
The input JEDEC filename is missing on the command line. The default extension is .eqn.

E3402: Device name not found in file XXX.--
The device name was not found in the JEDEC file XXX. You can either specify the device name on the command line with the -d option or edit the JEDEC file by placing the device name at the beginning of the file.