PLDASM

Programmable logic devices

Programmable Logic Devices (PLDs) (also known as PALs) are popular devices for implementing digital designs. These devices can be used where earlier systems used TTL or CMOS logic ICs. The PLDASM is a tool that allows Boolean equations to be programmed into a PLD in order to perform a user-defined logic function. Boolean equations make it possible to describe a function in an efficient manner, and this assures that the designer achieves the most compact solution with the fastest propagation delays. Furthermore, with Boolean equations the PLD can function as an address decoder, state machine or counter, and perform any number of other tasks ranging from the simple to the complex. While initially PLDs provided a savings in the amount of space used on a PC board, recent high speed PLDs are often significantly faster than the equivalent circuit implemented in TTL logic. Another recent development in PLDs is the complexity of the macrocells used for I/O. PLDASM automatically configures these macrocells, according to a set of simple rules which apply to all the PLDs supported by PLDASM. This allows substitution of one device for another, and reduces the amount of time required to 'learn' a new PLD.

A PLDs internal structure is built as an AND/OR matrix. A programmable input AND array can generate any AND function of all device inputs (with or without inversion). These AND functions are called 'Product Terms'. Product terms feed a multiple input OR gate. Since the AND/OR matrix can express any Boolean transfer function, the flexibility and functionality of a PLD is limited only by the number of terms available in the AND - OR arrays. PLD devices are available in different sizes, some with over 40 inputs, and some with up to 19 Product Terms per output. The outputs range from simple tri-state drivers to complex registered macrocells with programmable inverters.

Boolean functions

In an unprogrammed PLD, all fuses are intact. In other words, every input line is 'ANDed' with all other input lines (including any feedback terms available in the device) The output of these AND functions is fed into an OR gate and is then either fed onto more complex functions or presented directly on the output pins of the device.
For example, let us assume that we have a simple PLD with two input terms (A and B) and two output terms (X and Y). Internally, the device also makes the inverse of the input terms available (/A and /B). In the unprogrammed state, the logical function of the device can be represented by the following Boolean equations.

X = A*B + /A*/B + A*/B + /A*B

Y = A*B + /A*/B + A*/B + /A*B

In this state clearly the device has little use, X and Y are always equal to 1, regardless of the inputs A and B. However, when some of the terms in each of the AND functions are removed, the power of the device becomes obvious. For example, let us assume that the following fuses are 'blown':

from X, /A*/B, A*/B

from Y, A*B, /A*/B, /A*B

In the example given, the fuses were 'blown' so that no connection remained. The equations that remain after programming of the device are shown below.

X = A*B + /A*B

Y = A*/B

As can be seen, very quickly it becomes possible to provide complicated logic functions in a single package. The other main advantage of PLDs is that their precise function can be adapted by the individual designer to meet the application needs, even if the design specification changes after PC boards have been built, (or if bugs are found during system testing and production).
The above equations are usually entered into a disk file using an editor such as Wordstar or Sidekick. Be careful to avoid printer control codes which are created by programs such as Word or Wordperfect. The disk file is passed through the PLDASM to create a JEDEC file. The JEDEC file can be easily loaded into the PAL program for programming a device. If desired, the equations can be viewed using the PAL Editor function. Please note that a blown fuse is represented by a '1' in the JEDEC file, or as a '-' in the XPLOT or Editor. An intact fuse is represented by a '0' or a 'X'.
Boolean to jedec translation

Obviously, the PLD itself is not able to understand Boolean equations in the form given above. It is therefore necessary to translate the information from simple Boolean equations into a form that may be used to program the PLD.

In order to program a PLD, it is necessary to address each fuse in the device individually and to either blow it (create an open circuit) or to do nothing and leave it intact. The SPRINT PLD programming utility uses a fuse map to determine which fuse to blow and which fuse to leave intact. The SPRINT PLD Macro-Assembler creates a fuse map from an ASCII text file which contains Boolean equations. The fuse map that is created is called the JEDEC file.

For each input signal, there are two input line numbers, one for the actual input signal and one for its inverse. So, for this device there will be four input line numbers (1 = A, 2 = /A, 3 = B, 4 = /B). Additionally, there will be eight product line numbers as there were eight OR combinations in the unprogrammed device (4 for each output term). Therefore, for this device, the fuse map needed by the programming utility to create the Boolean functions described is shown below.

	Input Line Number				
Product Line Number	1	2	3	4	
1	X	-	X	-	
2	-	X	X	-	
3	-	-	-	-	X
4	-	-	-	-	
5	X	-	-	X	
6	-	-	-	-	
7	-	-	-	-	Y
8	-	-	-	-	

The fuse map shown here is stored in a JEDEC file where each fuse location represented by an 'X' is stored as a '0' (zero) and will be unaffected by the programming utility. Each location represented by a '-' is stored as a '1' and will be blown by the programming utility.

The program that produces this fuse map is the SPRINT PLD Assembler.
SPRINT pld Macro-Assembler

The SPRINT PLD Macro-Assembler is designed to convert an input file, created by any editor, into a standard JEDEC file for use by the SPRINT EPLD/EPROM programmer. The JEDEC file can also be used with other types of programmers.

The source file consists of definitions of all the pins in the device, including power and ground pins, followed by Boolean equations for the function to be performed by each output pin in the PLD. SPRINT PLDASM supports a flexible Macro feature which reduces the amount of data that has to be typed-in for the PLD equations. This Macro function also makes the input file easier to read and understand, for better documentation of the application.

The input file is entered using any editor on the IBM PC. If the editor inserts control characters for right justification or other purposes, the non-document mode (as in Wordstar) should be selected. This is because the SPRINT PLDASM Macro-Assembler will not be able to understand these control characters.

The completed source file is assembled into a JEDEC file using PLDASM. Errors in the source file are identified and reported to the user in clear text. The line number/character number where the error was detected is displayed on the screen for use in correcting the error.

The SPRINT PLDASM automatically performs two passes in the assembly process. The first pass sets the macrocell output stage in high-density PLDs, the second pass fills in the equations. Device macrocells are not to be confused with the Macro function of the SPRINT PLDASM.

Shown below is an example of the use of PLDASM which demonstrates the operation. From the main menu of the SPRINT, selecting PLDASM will load the PLDASM Macro-Assembler and display all files in the SPRINT default directory with the '.PLD' extension. The user can position the highlight to the filename desired and click the OK button, or a filename may be typed-in after the prompt as shown below:

PLDASM assumes a file type of '.PLD', so if the desired file has the file extension '.PLD'.
Select file : EXAMPLE

If the file EXAMPLE.PLD is on the default drive, it will be used by the SPRINT PLDASM as the source for the assembly, and a JEDEC file with the filename EXAMPLE.JED will be created on the same drive and in the same directory. The result file always has the file extension '.JED'.

Sprint PLDASM syntax

A source file for the Sprint PLDASM consists of the following parts:

Comments (Documentation)
Device selection
Pin definition
Macro definition
Equations
End

All text in the source file can be either upper or lower case. Sprint PLDASM ignores the case of the text. SPRINT PLDASM will ignore all blank lines included in the source file.

Comments

Comments may be entered on any line in the source file. Comment fields within the source file are started with the character '. All text after the ', until the end of the line are ignored. Comments can be placed anywhere in the input file. All text preceding the device selection denoted by the word 'DEVICE' is ignored and may be used as an extended comment field.

Label Definition

A label in the PLDASM consists of any string of up to 10 ASCII alphabetic or numeric characters. The first character of a label must not be a 1 or a 0. A '/' may precede the label to indicate active low functions. Some words are reserved, see the list below.
The labels '1' and '0' are reserved words and indicate fixed logical 1 and 0 conditions respectively. See the complete list of reserved words later on in this section.

Device definition

The word DEVICE (or device), in the source file, is the logical start of the data to be assembled by PLDASM. The word DEVICE must be in the text of the source file starting in column 1 of a new line to be recognized by the SPRINT PLDASM Macro-Assembler. The device type to be used by the assembler must follow the word DEVICE, there must be a space between DEVICE keyword and the type selected. Currently supported devices are listed on the screen by entering a <ESC> instead of the filename (see above). Note, device types entered are only generic device types, the manufacturer of the device to be programmed should not be included in the device definition line. It may be included as a comment field if desired by the user.

Example:

DEVICE 20L10
device 16c1

SECURIty fuse control

The security fuse in a PLD may be automatically set during programming by putting the keyword 'security' in the source file, between the device definition and the keyword 'start'. The output JEDEC file will include a command to the PLD programmer to set the security bit of the device after programming and verification. The default condition is to leave the security fuse intact. Setting the security fuse prohibits anyone from reading stored data from a PLD

Pin definition

After the device definition, and before the keyword 'START' (see below), the input pins, the output pins and Macros are defined. Typically the pins are defined first, but Macros and pin definitions may be mixed.

Only used pins need to be defined at this stage. The pins for VCC and ground may also be defined. During assembly, the SPRINT PLDASM assembler will verify that the pin definitions given match the device that was defined in the device definition. If the pin definition is incorrect or does not match the device definition, the SPRINT PLDASM assembler will produce an error and a JEDEC file will not be created.

The pin definition consists of a label followed by a pin number. There may be an optional equal ('=') sign between the label and the pin number. The pin number is the decimal physical pin number. One pin definition per line is allowed.

NOTE: Some macrocell functions are defined at pin definition time. See the application note section for details on the device being used.

MACRO definition

A MACRO definition consists of the word MACRO in column 1 of a new line. The MACRO keyword is followed by a label (the MACRO name) followed by a string of characters up to 180 characters long (end-of-line characters are ignored). The terminator is the character ';'. A common error when defining MACROs is to omit this terminating character. The ';' character is not included in the Macro. PLDASM allows up to 40 macros.

Later, in the equations, the Macro name can be inserted into the text where the string is to be substituted. In the equations, the Macro name is preceded by a '&'. An example is shown below:

```
....
MACRO  test1   /a13*/a12*/a11* a10* a9;
....
START
....
output = &test1* a1;
```

The AMD PALASM example of a Barrel Shifter is included on the distribution floppy diskette to show the use and advantage of Macros.

Macros cannot be inverted, so a '/' may not precede the macro name in the equations.

Equations

The keyword 'START' in column 1 signifies the end of the pin and MACRO definitions. The text following the 'START' keyword, until the 'END' keyword is found in column 1 of a new line, are the equations which define the functions of each output pin in the specified device. The equations are assembled into the JEDEC file ready for programming into the PLD device chosen. Illegal combinations, such as using a dedicated input pin for output, are identified by the SPRINT PLDASM assembler and will be reported with line and character number.

Each equation line consists of the following parts:

output pin label
enable (optional) label.ena
function (first line) macrocell function
input equations set of 'AND' equations
+ indicates additional OR'd Product Terms
^ indicates an exclusive OR'd Product Term
; end of equation for this pin

FUNCTIONS

All devices have outputs, some devices allow hidden 'nodes'. PLDASM treats both as macrocells. The Boolean result of the equations entered for a macrocell can be applied directly to the output, or can be synchronized by either D or T type registers. In addition, the equation can be inverted, allowing the application of the De Morgen theory to reduce the number of product terms required. These inversion and register options are defined with the syntax shown below:

macrocell functions (outputs)

=	combinatorial equation result
/=	active low combinatorial result
:=	D type registered result
/:=	active low D type registered result
^:=	T type registered result
/^:=	active low T type registered result

The data sheet of the device selected will indicate the types of outputs allowed. Incorrect selections will be reported as errors by PLDASM. If a LABEL is assigned to a physical device pin and the output enable is active, (by default or explicitly controlled with the '.ENA' modifier), the equations stored inside of the device will define the function of the output pin. For those devices with hidden functions, the LABEL could be an internal NODE, and the equation would define the function of that NODE.

The '/' character before a label signifies inversion. SPRINT PLDASM enforces the concept that the label in the pin definition and the output pin label in the equations, must have the same inversion status. Use a '/=' or a '/:=' in the FUNCTION field to indicate that the result of the equation is active low. When converting from a MMI/AMD PALASM file, simply move the '/' from before the output pin to the macrocell function pin, i.e. before the = or :=.

Example:

MMI/AMD equation.	/OP1	:=	IP1*IP2 + IP3
SPRINT PLDASM equation	OP1	/:=	IP1*IP2 + IP3

INTERnal NODES

Every output of an AND or OR equation in a device is a node. Most nodes are buried inside a device, and do not have to be given names. For example the eight AND functions in each output stage of the 16R8 go into a fixed OR gate, there is no need to reference these points, they are hidden and are automatically managed

Sometimes other internal functions need to be defined, for example the RESET and PRESET product terms available in many PALs.

PLDASM assigns these internal function with pseudo pin numbers called 'node numbers'. The ARESET and SPRESET and other built in nodes require no definition by the user. See the reserved word list below.

Many new devices have internal registered or combinatorial feedback logic that is either always hidden inside the device, or can be hidden by the user. PLDASM assigns these internal logic elements a pseudo 'pin' number that is higher than the number of pins in the physical package. To use a NODE, simply assign a LABEL to the number shown in the table, and write the output equations in the same way as for a physical output pin. See the section on Dual Feedbacks and Buried Registers for more information on the use of nodes.

actual boolean EQUATIONS

After the output (or node) has been specified, and the macrocell function is defined, the actual Boolean equations for each output stage are listed. The equations consist of groups of AND and OR terms.

AND terms are specified by the '*' symbol. For example, A 'AND' B is specified by the equation 'A * B'. OR terms are specified by the '+' symbol. For example, A 'OR' B is specified by the equation 'A + B' and A 'XOR' B is specified by the equation 'A ^ B'. Each equation for each output pin must be terminated with a ';'.

Each AND term consists of LABELs, and MACROs separated by '*' and terminated by either a ';' or '+' or '^'. The ';' terminator denotes the end of the equations for this pin. The '+' and '^' terminators separate AND terms, it indicates that these AND terms are OR'd or XOR'd together.

The LABEL is a standard PLDASM label, with an optional '/' in front to indicate inversion. MACROs can be used to replace any combination of LABEL, '/', '*', '+' or '^'. MACROs are invoked by an '&' in front of the MACRO name defined above. Macros cannot be inverted, so no '/' may precede the MACRO name.

The example in this chapter shows the use of MACROs within the input equations.

CLOCKS

Traditional PALs and EPLDs use a single clock for all flip flops (pin 1). In these devices the PLDASM assumes that the clock comes from pin 1. Some newer devices allow a clock Product Term to be defined for the each flip flop. These devices are usually called Asynchronous PALs. PLDASM can generate this clock Product Term via the '.CLK' modifier which allows an equation to be written for each of the clock Product Terms. In the EP600, EP900 and other devices with a selectable synchronous or asynchronous clock, the PLDASM will automatically default to synchronous unless a '.CLK' modified Product Term is defined.

OUTPUT ENABLE

Some devices have the ability of producing tri-state outputs on some pins and hence the ability to enable or disable the output driver on those pins. The determination of whether to enable or disable the output from these pins may be defined in a separate equation, thereby only enabling the output to be high or low under certain logical conditions. This extra programming is not mandatory. If no enable equation is included in the equations for any output pin supporting this function, SPRINT PLDASM will assume that the output is continually enabled.

In order to specify that the desired output pin only be enabled under certain logical conditions, an equation needs to be defined for the conditions under which the output should be driven. The equation defining this is virtually the same as a standard output equation, except that the OUTPUT pin definition in the first column is suffixed with '.ENA'. This is called a pin modifier. Other modifiers are also possible - please see the next section for more information. If the '.ENA' enable is used, then two equations need to appear for that output pin, one to enable the output and one for the logic state of the output. The function that is allowed in an enable equation is '=' or '/=' depending on the device. The enable equation must be terminated with a ';'. An example of the enable function is included in this manual to demonstrate its use.

Standard PALs have the output enable function hard-wired for registered outputs to pin 11 or pin 13 (20 or 24 pin devices). If an equation is provided to connect the output enable to these hard-wired connections, PLDASM will check to verify that this hard-wired connection exists in the physical PAL. The feature is provided to allow upwards compatibility to devices which are socket compatible to the older PALs, but also offer a programmable output enable Product Term, for example the 18U8 and 20G10 type of devices. It is recommended to always include the hard-wired enable equation:

```
LABEL.ENA  =  /PIN11
```

when using older 16R4 or similar type PALs to allow for easy conversion to new, CMOS EPLDs. This approach is also used by the UNASM to allow conversion from one type to another.

If the selected device has a direct connect path to an I/O pin which can be used to bypass the output enable Product Term (for example the Cypress 20G10), then the PLDASM will automatically select the higher speed direct connect path if the output enable equation includes only one pin, and that pin matches the physical direct connect path.

CLocks, XOR and other special features

Many newer device support special features in the macrocells, for example XOR Product Terms, async clocks, resets etc. PLDASM uses the same concept as the .ENA for other function. Examples of these files are on your PAL diskette in the Examples sub-directory. A modifier is added to the pin name used in an output equation. An operator is used in the equation itself. A list of modifiers and operators is shown below:

modifier	function	typical devices supported
.if	force feedback from the I/O pin	See list
.rf	force feedback from the register	See list
.cf	force feedback from the PT OR gate	EP310
.ena	output enable	most
.clk	output register clock, asynchronous	7C331 EP600 20RA10 ATV750 and others
.rst	output register reset, one register only	7C331 20RA10 EP600 ATV750
.set	output register preset, one register only	7C331 20RA10
.xor	XOR Product Term	7C331 20X4/6/8 20XR4/6/8 etc
.iclk	input register clock Product Term	7C331
.irst	input register reset	7C331
.iset	input register set	7C331
.j	J input of JK flip flop	78C800
.k	K input of JK flip flop	78C800

Operators separate the inputs in the equations. Three kinds of operators are allowed:

operator	function	devices supported
*	AND	all
+	OR	all
^	XOR	20X4/6/8

FUSE

Some PAL or EPLD devices have fuse functions which cannot be effectively described in the PLDASM syntax. Presently, the only device that fits this limitation is the Ricoh 16LC8 family. To use these unusual functions, write the code as if the extra fuses were not set, then set each fuse with the statement 'FUSE xxxx;' where xxxx is the number of the fuse to be specially programmed. FUSE can be used in any device to force any condition. It is suggested that FUSE not be used without a detailed understanding of the device being programmed.

Example:

```
FUSE  1234
```

will set fuse number 1234 to a 1. Note, the default condition for all fuses is 0. Always put the 'FUSE' statement at the end of your files.

VECTORS

Test vectors may be inserted into the source file by using the label 'VECTOR' followed by the exact test vector sequence as used in the standard JEDEC file. The purpose of this feature is to allow the UNASM to store the test vectors found in a JEDEC file in a form that can be used to re-assemble the source into another device - without loss of the test vector information. See Chapter 2 for information on the test vector standard. We suggest that test vectors be generated using the PIN EXERCISER function of PAL.

End of file

The keyword 'END' indicates the end of the file. If the SPRINT PLDASM program terminates without the message 'Assembly Complete', then it is likely that the keyword 'END' has been omitted from the source file. Successful completion of the SPRINT PLDASM assembler will result in a JEDEC file being written to the disk and a message on the screen signifying a successful completion of the operation.

Reserved Words

Some words are reserved for use by the PLDASM, these must not be used as labels. The words are found in 3 groups: Assembler Directives, Device Special Features, and Device Built-In Nodes.

1) Assembler Directives are control words for the PLDASM itself. Note that the numbers '0' and '1' can be used to set a Product Term to all OFF(0) or ON (1):

DEVICE	LABEL	END	START	
MACRO	VECTOR	FUSE	0	1

2) Device Special Features control internal fuses in the device:

SECURITY	this enables automatic security fuse setting after programming
MISER	see device data sheet
TURBO	see device data sheet
ZERO	set zero power standby mode

3) Device Built-In Nodes:

label	note	device examples
ARESET	global	22V10, and others
SPRESET	global	22V10, and others
OBSERVE	global	23S8
$CLRA	78C800	
$CLRB	78C800	
$LEA	78C800	
$LEB	78C800	

Device Built-In Nodes are used in the same manner as any other LABEL in PLDASM. For example:

```
ARESET = label1 * label2;
```

SPRINT PLDASM EXAMPLE

The directory \EXAMPLES contains source code examples for many devices, especially ones with unusual functions. You can use these examples to understand the use of the PLDASM. One example, the 16R4 will be explained here in detail.

The attached example shows a file using the PLDASM and MACROs. The file is from the AMD data book on PALs.

The first section is the header text: all data up to the reserved word 'DEVICE' in column one is assumed to be comment and is ignored by PLDASM. In the example, this explains who wrote the file, and some background information as documentation for the reader only.

Then the DEVICE is defined, here it is a PAL16R4. The JEDEC file created for a device is the same for all manufacturers of the same device, so no vendor need be specified, only the generic device type.

Then the pins, and MACROs are defined. Any sequence may be used, but typically the numeric order is the easiest to read.

The keyword 'START' indicates the beginning of the equations. The output pin label starts in column 1. It has the same polarity (a / in front) as the definition above.

The output pin /ZERO has been added as an example of the use of the output enable. When the inputs S0, S1, and S2 are all 0, the output will be enabled. It will be high if any of the data bits (D0 to D7) are 1, otherwise it will be low. If the inputs S0, S1, S2 are not all 0, then the output will be high impedance.

The keyword 'END' indicates the end of the file. After the detection of this keyword, the SPRINT PLDASM will write the translated JEDEC file with the file extension '.JED' to the disk. The data can then be read by SPRINT by an input command with the same filename as the original '.PLD' file.

The resulting JEDEC file is shown after the '.PLD' file. See next page.

```
PAL DESIGN SPECIFICATION
PAT001 KEVIN M.  OW-WING 6-22-85
4-BIT SLICE FOR AN 8 BIT BARREL SHIFTER
ADVANCED MICRO DEVICES
Modified to show the output enable function.

device 16r4;
CK              1
D7              2
D6              3
D5              4
D4              5
D3              6
D2              7
D1              8
D0              9
GND             10              'This is a comment
/E              11
/EQU0           12
S0              13
Q0              14
Q1              15
Q2              16
Q3              17
S1              18
S2              19
VCC             20

macro   f0                      /S2*/S1*/S0;
macro   f1                      /S2*/S1* S0;
macro   f2                      /S2* S1*/S0;

macro   f3                      /S2* S1* S0;
macro   f4                       S2*/S1*/S0;
macro   f5                       S2*/S1* S0;
macro   f6                       S2* S1*/S0;
macro   f7                       S2* S1* S0;

start
Q3 /:=          &f0*/D3 +  &f1*/D2 + &f2*/D1 + &f3*/D0 +
                &f4*/D7 +  &f5*/D6 + &f6*/D5 + &f7*/D4;

Q2 /:=          &f0*/D2 +  &f1*/D1 + &f2*/D0 + &f3*/D7 +
                &f4*/D6 +  &f5*/D5 + &f6*/D4 + &f7*/D3;

Q1 /:=          &f0*/D1 +  &f1*/D0 + &f2*/D7 + &f3*/D6 +
                &f4*/D5 +  &f5*/D4 + &f6*/D3 + &f7*/D2;

Q0 /:=          &f0*/D0 +  &f1*/D7 + &f2*/D6 + &f3*/D5 +

                &f4*/D4 +  &f5*/D3 + &f6*/D2 + &f7*/D1;
' Now define the output enable for the "/ZERO" pin

/EQU0.ENA        = /S0*/S1*/S2;
/EQU0           /= /D0*/D1*/D2*/D3*/D4*/D5*/D6*/D7;
end
```

SPRINT PLDASM Operation

After starting PLDASM, the filenames with the type '.PLD' on the current default directory will be displayed. Use the mouse to highlight the desired file and click OK, or enter the filename and click OK.

After the file is selected, then the processing of PASS1, PASS2 and the writing of the JEDEC file to the disk will be reported on the screen.

If any errors are detected, they will be reported with the line number and character number of the position in the file where the error was detected. The type of error will be displayed with one of the messages described overleaf. The line containing the error, and the previous four lines will be displayed on the screen for easy identification. An arrow points to the location where PLDASM detected the error. In some cases this may be a few characters after the actual error location. Note that the character position counts tabs as one character. The error messages are designed to be self-explanatory, additional comments to the types are listed below: (next page)

SPRINT PLDASM ERROR messages

Error Message	Explanation
Device not supported	See list of currently supported devices.
Macro definition error	Macro too long, or too many Macros.
Reserved word	Some words are reserved for use by the PLDASM, see the reserved word list. In any case a reserved word was found in an unexpected location.
Too many/few pins	Attempt was made to define pins not on the physical device.
Missing terminator -> ;	A';' is required at end of the line.
Not enough product terms	The user attempted to use more OR terms than in the physical device
Label not defined	Check spelling, labels need to be listed in the definition portion of the file.
Improper use of pin	Attempt was made to use a pin in a manner not supported by the physical device. For example an active high output on a PAL16L8
Use /= or /:= for output inversion	See section on OUTPUT PIN
Output enable error	Attempt was made to use an output enable on a pin of a device which does not have it.
Name already defined, or missing START	A LABEL has been defined twice, or the directive START is missing

Error Message	Explanation
Pin cannot be used as input	The physical device does not allow this pin to be used as an input
XOR not allowed on this pin	XOR is not supported.
Feedback from this pin is not possible	The pin is output only. Occurs in GAL direct mode for example.
This node does not allow inversion	Node is active high only.
Missing keyword or ; character	Usually occurs when End Of File is reached before the directive END
Missing operator - either ^ or + needed	Occurs when a two labels are shown without an operand between them.

Application Note:

This section provides some special notes on the use of PLD devices with special macrocell features. The traditional MMI PAL type of structure as used in the 12L6 or 16R4 type of devices forms the basis for all devices in the PLDASM. Devices with higher pin counts and more Product Terms do not change the basic concept. Even those parts with macrocells are still compatible with the original concept, the macrocells only give the user the important freedom to re-configure certain pins to match the task better. All devices supported by PLDASM, except those listed below, conform fully to the original MMI concept. The parts listed here have some extra features that require device specific explanations.

PLDASM supports automatically all device features, the user need only focus on the application. Detailed error messages from PLDASM will inform the user when a device does not support the functions attempted.

Programmable feedback options

Some devices (marked * in the following list) support independently programmable feedback options. Normally a PAL or EPLD has one feedback per macrocell additional feedback options compared to all other PAL type devices. The PLDASM will default to treat these macrocells in the same manner as the standard MMI type PAL devices unless overridden by feedback options in the label definition. The feedback override option types are:

.CF	Combinational - output of or gate in macrocell
.RF	Registered - output of register in macrocell
.IF	I/O pin - feedback is from the device I/O pin

The default feedback conditions are:

.RF	for registered outputs (:= macrocells)
.IF	for combinational outputs (= macrocells)

The designer can force .IF, .CF or .RF by adding this name to the label in the label definition. For example:

```
DTACK.IF        15;
COUNT.RF        16;
```

This will cause the PLDASM to use the device pin 15 as feedback. The output pin can be registered or not, without affecting the selection of .IF . Likewise, the register in the macrocell of pin 16 will be used for feedback into the device, even if pin 16 is set for combinational mode.

Hidden (buried) registers

Some devices have the ability to have flip flops that are used inside the device without any direct access to the I/O pins. These are called buried or hidden registers. PLDASM assigns these flip flops pseudo pin numbers called 'internal nodes'. A list of devices with internal nodes is shown below. The node numbers are like pin numbers, except that pseudo pin number is higher than the actual number of pins in the device package.

Dual feedbacks

Many new devices now support dual feedback input terms. For many years this concept was known to offer the possibility to double the use of the I/O pins in the package, by 'hiding' the registers in the I/O macrocells. However the price for dual feedbacks was device speed, since the extra feedback lines increased the number of input terms by 50% - making a larger array - which is slower. Now some parts have even dual and also hidden registers. PLDASM supports dual feedbacks by providing pseudo pin numbers called internal nodes for the extra feedback. This allows the register in a I/O macrocell to have a different name than the I/O pin which is connected to that macrocell.

PLDASM uses the default concept for all dual feedback macrocells. If an output is registered, then the default feedback will be from the register. If an output is combinatorial, than the default will be from the I/O pin. If the second feedback is required, then the internal nodes (pseudo pins) must be used. This is best explained in the four possible cases:

CASE 1:registered output, pin feedback not used. This is coded like any PAL16R8.

CASE 2:combinatorial output, register not used. This is coded like any PAL16L8.

CASE 3:registered output, both register and pin feedback is used. The register section is coded like any 16R8, the pin feedback is the internal 'node' that is assigned to that pin. If the output enable is disabled, then the register is hidden.

CASE 4:combinatorial output, register is used. This example implies that the equation going to the 'D' of the register is available on the I/O pin. The combinatorial section is coded like any 16L8, the feedback from the register output can be referenced via the internal 'node' that is assigned to that pin.

In the next example, we see that pins 26, 27 and 28 do not exist in the physical device, but do exist as internal nodes. In bit0 of our presettable counter, the register is not hidden, and it can be read-out with the /OE control, the default feedback is from the register, but in order to do preload from a bi-directional bus, we need to access the I/O pin with a different name. In bit1, the register cannot be read-out, the pin is used as an input, and this input has a function not directly related to the register, the I/O is accessed via the internal node. In bit2, the pin is combinatorial, therefore the feedback has the same name as the output signal. If we wish to access the register's Q output, then we use the internal node for this pin. As we see, the internal node refers to the feedback path not selected as the default condition.

dual feedback Example:

```
device EPL204

' pins with registered outputs

reg0            19      'pin 19 registered output
dbus0           28      'pin 19 I/O pin

hidreg1         18      'pin 18 register
preload         27      'pin 18 I/O pin


' pins with combinatorial outputs

hidreg2d17      'D input of pin 17 register
                        '  also I/O pin name
hidreg2q26      'Q output of pin 17 reg

' regular input pins

OE              2
dbus1           3

start

reg0.ena = /OE;
reg0            := reg0         *       /preload                ' case 3
                 + dbus0 *      preload;

hidreg1.ena      = 0;                                                   ' hidden reg.
hidreg1         := hidreg1      *       /preload                ' case 3
                 + dbus1 *      preload;

hidreg2d = hidreg2q     *       /preload                ' case 4
                + hidreg2d      *       preload;


end
```

Lattice GAL16V8/20V8, AMD PALCE16V8/20V8.

These devices have three basic configuration modes. While the data sheet shows 5 different macrocell combinations, the 16V8 and 20V8 silicon does not allow all of these to be selected at the same time.

1) The default mode is the 'dedicated' mode. In this mode all outputs are combinational, no output enable is permitted, but there are up to eight Product Terms per output stage. The outputs cannot be used as inputs (no feedback).

2) The combinational output mode is selected by PLDASM if any of the outputs in the equations have an output enable term (note that a 'LABEL.ENA = 1;' may be used for the always enabled condition). In the GAL16V8/20V8 combinational mode, the two outermost macrocells are output only (no input or feedback), the remaining six may be used as output or input or bi-directional. The equations may have up to seven Product Terms per output, plus one output enable.

3) The third mode is the registered mode, this mode is selected if any of the outputs have been set as a registered output. In this mode, registered outputs have eight Product Terms and a pin 11 (13) direct coupled output enable, while combinational outputs have seven Product Terms and a programmable output enable.

In these devices, an additional set of fuses is provided to turn-off unused Product Terms in order to reduce power consumption. These fuses are automatically set by PLDASM for the lowest possible power consumption.

The signature bytes described in the data sheet can be edited in ASCII with the 'O' command in 'PAL'. Any data entered with this editor can be stored in a JEDEC file with the 'W' command.

PLX Technology PLX448/464

The dual feedback node numbers for pins 13,15-23 are 32-39. These devices also have an open collector fuse option for pins 13,15,22,23. In order to select open collector outputs, use the FUSE statement to set the correct bits as shown below:

Fuse 5098	pin 23 open collector
Fuse 5101	pin 22 open collector
Fuse 5112	pin 15 open collector
Fuse 5115	pin 13 open collector

Cypress 20G10

This device supports a direct output enable path from pin 13. While the normal user will generate an output enable (if required) with a Boolean equation, the speed of the pin 13 direct path is about 5 ns faster. This direct path will automatically be selected by PLDASM if the equation for the output enable of any pin contains pin 13 only.

FPLAs

The FPLAs supported by PLDASM are of the folded array or expander concept. An example is the EXEL 78C800. The PLDASM will automatically solve the equations to match the device resources. AND/OR combinations will be assigned to internal Product Terms. A simple minimizer assures that expanders are only used when a NOR function is required. Note that the outputs of an expander can only be OR'd, not AND'd since the expanders only have one, active high, output. The expanders may be assigned node numbers in order to build Set/Reset flip flops, otherwise it is normally not necessary to assign them node names. For the 78C800, see the special file, 'EXEL.DOC' on the PAL disk for examples

.
INTERNAL NODE NUMBERING and other special functions

Cypress CY7C331

Pin	Node	(12 input flip flops on pins 28 to 15)
28	29	
27	30	
....	....	
16	39	
15	40	

Cypress CY7C332

This device uses the following definitions:

PINx.RF=		registered input latch
PINx.CF=		combinatorial input
PINx	=	latched input (default)

Clock 1 is the default, use the FUSE commands to select clock 2. Clocks are positive edge, use the FUSE command to select negative edge.

EXAMPLE:

FUSE 9612	sets pin 4 clock from pin 2
FUSE 9637	sets pin 11 clock to negative edge

AMD 23S8

Buried registers 0 to 5 are mapped to nodes 21 to 26

Atmel ATV750

Nodes 32-41 are used to reference the outputs of the Q1 registers in the macrocells on pins 14-23. Node 32 == 14 etc.

PLDASM automatically controls the macrocell control bit 'S1', if more than 1/2 the total product terms are used. When S1 is set, Q1 should not be used.

Optional uses of Q0 and the physical pin (X == not used)

Physical pin is:	node
input	14-23 standard input nodes
	42-51 feedback from Q0 (hidden use of Q0 : do not use :=, only =)
combinatorial	14-23 standard
	42-51 feedback from Q0 (comb delayed by clock)
registered output	14-23 feedback from Q0
	42-51 feedback from physical pin (another way to hide the register)

If a macrocell is used for registered output - then Q0 will be the output register, and the feedback path from Q0 is selected by the names assigned to pins 14-23. If Q0 is to be hidden, then nodes 42-51 would be used to indicate the physical I/O pin.

Ricoh EPL204

all output macrocells have dual feedbacks. Pins 12-19 have dual feedback nodes 21-28.

ICT 22CV10Z

The default mode is zero standby power. To disable this feature, use the TURBO keyword, this will make the part active at all time.

Lattice GAL6001

The device has the following special features :

Input registers:
PINx = direct (default)
PINx.LI = latched input
PINx.RI = registered
PINx.CI = direct

Note : All inputs in the same bank must have the same type (LI, RI or CI)

Q14 - Q23 = feedback from before the output inverter of the pins 14 to 23, is referenced via the labels of pin 14 to pin 23. the polarity of this signal is matched to the pin (inverted).
P14 - P23 = feedback from the physical pin is via NODEs 40 to 49 (pin14 = node40)

Q0 - Q7 = (burried) NODEs 32 to 39
Note : Can be registered or combinatorial

Pin modifiers:
.ENA for output enable (default is always enabled)
.CLK for async clocks (default is sync clock pin 13)
.CENA for clock enable (default is always enabled)

Use PIN.CENA /= for inverted clock enable
Use PIN.CLK /= for inverted async clock

ARESET is the reset function

DEvice names and features

PLDASM uses the name shown in the first column to select a device. Note that some vendors have used the same name for incompatible devices (Lattice and Signetics 16V8 for example). Thus two names are provided. All devices use the standard syntax, some have additional features as listed below:

Name	Vendor	Specials
153		PLS153 type
173		PLS173 type
473	Signetics	
6l16	MMI	
8l14	MMI	
10h8		
10l8		
12h6		
12l6		
12l10		
14h4		

14l4		
14l8		
16c1		
16h2		
16l2		
16l6		
16l8		
16lc8	Sprague	
16n8	TI	
16rc4	Sprague	
16rc6	Sprague	
16rc8		
16p8		
16r4		
16r6		
16r8		
16rp4		
16rp6		
16rp8		
16v8	Lattice	Automatic Mode selection
*18cv8	ICT	
18l4		
*18n8	TI	
18p8	AMD	
18u8	AMD/MMI	ARESET, SPRESET
18v10	Lattice	
20c1		
*20cg10	ICT	ARESET, SPRESET
20g10	Cypress	
20l2		
20l8		
20l10		
20r4		
20r6		
20r8		
20ra10	MMI	Async clocks

(continued on next page)

Name	Vendor	Specials
20rp4		
20rp6		
20rp8		
20x4		XOR
20x8		XOR
20x10		XOR
20xrp4	AMD	XOR
20xrp6	AMD	XOR
20xrp8	AMD	XOR
20xrp10	AMD	XOR
20v8	Lattice	Automatic Mode selection
.22cv10z	ICT	ARESET, SPRESET, TURBO
22p10	AMD	
22v10	CYP, AMD, TI and others	ARESET, SPRESET
22vf10	AMD	ARESET, SPRESET

*22vp10	TI,Cypress	ARESET, SPRESET
22xp10	AMD	
*23s8	AMD	ARESET, SPRESET, OBSERVE, buried
24r4	AMD	
24r8	AMD	
24r10	AMD	
24l10	AMD	
26cv12	Lattice	ARESET, SPRESET
26v12	AMD	ARESET, SPRESET
32vx10	MMI	ARESET, SPRESET, dual feedbacks
*5c031	Intel	ARESET, SPRESET
5c032	Intel	Turbo, Miser
*5c060	Intel	Async clocks, T type Registers, turbo
7c331	Cypress	Async clocks, T type registers
7c332	Cypress	Input registers, RF, CF options
78c800	Exel	FPLA structure, see EXEL.DOC
85c220	Intel	Turbo
85c508	Intel	
e16p8	National	ECL
e301	Cypress	ECL same as E16P8 function
*ep310	Altera	ARESET, SPRESET
ep320	Altera	Turbo, Miser
*ep600	Altera	Async clocks, T type, turbo
*ep610	Altera	Async clocks, T type, turbo
ep900	Altera	Async clocks, T type, turbo
ep910	Altera	Async clocks, T type, turbo
epl10p8	Ricoh	Extra features accessed via FUSE
epl12p6	Ricoh	statement in these RICOH parts
epl14p4	Ricoh	
epl16p2	Ricoh	
epl16p8	Ricoh	
epl16rp4	Ricoh	
epl16rp6	Ricoh	
epl16rp8	Ricoh	
epl204	Ricoh	dual feedbacks, nodes 21-29

(continued on next page)

Name	Vendor	Specials
plc16v8	Signetics	different JEDEC map than Lattice
plc18v8z	Signetics	ARESET, SPRESET
plc20v8	Signetics	different JEDEC map than Lattice
plx448	PLX Tech	dual feedbacks, open collector, ARESET, SPRESET
plx464	PLX Tech	dual feedbacks, open collector, ARESET, SPRESET
t9800/1	Toshiba	
v750n	Atmel	Async clocks, dual feedbacks, old
v750i	Atmel	Async clocks, dual feedbacks, new

Notes: Devices marked (*) have programmable feedback options. These default to the MMI original 16R6 style register feedback or I/O pin feedback. The default can be overridden using the .if, .rf, .cf label modifiers in the pin definition to force I/O feedback, Register feedback or Combinatorial feedback. Note that Combinatorial feedback (.cf) is not possible with all devices. Check the data book for details of the output macrocells.

EP600, EP900 JK and RS flip flops, Using T flip flops

In the data sheets for these devices, the output macrocells are shown with D, T, JK and RS flip flops. In the physical silicon, there is only a D and a T option. The user must create out of the T type, the JK and RS type of flip flops.

This note can be applied to any EPLD with T type flip flops, for example the 32VX10, 7C330, 7C331, 20Xxx etc.

The following general purpose equation will explain how to code the JK option. The RS option is similiar.

EXAMPLE:

```
Q.J     :=      A * B;                  ' this is the function required
Q.K     :=      C * D;
```

EP600 solution:

```
Q               ^:= A * B * /Q
                  + C * D *  Q;
```

NOTE: EP600 output inversion method:

```
/Q              ^:= A * B * /Q          ' active low ouput
                  + C * D *  Q;
```

(the use of ^:= causes Q to toggle when the equations are valid
the use of /^:= causes Q to toggle when the equations are not valid)

20X8 solution:

```
Q               /:= A * B * /Q          ' 'J' function
                  + C * D *  Q          ' 'K' function
                  ^ Q;                  ' make 20X8 'D' into 'T'
```