

ADLIB

Version 1.0

September 1987

P25-02255-00

Changes are made periodically to the information contained in this manual. These changes will be incorporated into subsequent editions.

Altera Corporation
3525 Monroe Street
Santa Clara, CA 95051
(408) 984-2800
TELEX: 888496

Copyright © 1987 Altera Corporation. All rights reserved.

No part of this manual may be copied or reproduced in any form or by any means without prior written permission of Altera Corporation.

A+PLUS, SAM+PLUS, LogicMap, Turbo-Bit, MacroMuncher, SAM, BUSTER, EP310, EP320, EP600, EP610, EP900, EP910, EP1210, EP1800, EPB1400, EPS444, and EPS448 are trademarks of Altera Corporation. LogiCaps is a registered trademark of Altera Corporation. MS-DOS is a trademark of Microsoft Corporation. FutureNet DASH is a trademark of FutureNet Corporation. IBM Personal Computer is a registered trademark of International Business Machines Corporation.

Read This First...

Your Altera Design Librarian (ADLIB) documentation consists of two main parts:

Getting Started provides a functional description, instructions on invoking ADLIB, general design guidelines, and information on how the Flattener module of A+PLUS finds a custom-made MacroFunction.

ADLIB Tutorial describes, with the help of an example, how to create your own MacroFunction. It also includes a Worksheet designed to help you create your own MacroFunction.

In addition, the manual contains a section with *Error Messages*, a *Glossary*, and an *Index*.

At the back of the manual, you find a *Customer Comment Form* and a *Problem Report Card*.

Manual Updates

Altera documentation is updated with Change Pages, Section Reprints, and a **README** file.


Change Pages are issued for minor changes to the manual. New information is identified with vertical change bars in the margins next to the changed text. In addition, the date of issue is printed at the bottom of each page.

Section Reprints are issued if a section requires a substantial number of changes. The date of issue is indicated at the bottom of each page.

A **README File** is provided on the LogiCaps **INSTALL** diskette. This file contains information about recent changes to the software that are not yet reflected in the manual.

Printing Conventions

The following notational conventions are used throughout this manual:

- | | |
|---|--|
| Times Bold | — all A+PLUS commands, prompts, and messages |
| | — all user input, including keyboard keys |
| Times Light | — most file output as displayed on screen |
| <i>Helvetica Italics Bold</i> | — all references to Altera manual titles |
| <i>Helvetica Italics Light</i> | — all references to sections within Altera manuals |
|  | — information that requires special attention |

Contents

Section 1: Getting Started

Functional Description	1-2
Required Software Tools.....	1-3
Where to Begin?.....	1-4
Invoking ADLIB.....	1-5
Design Guidelines	1-7
MacroFunction Symbol	1-7
MacroFunction Stubs.....	1-8
Active Low Input and Output	1-9
Defaults	1-11
How Does A+PLUS Find a Custom-Made MacroFunction?	1-12
Invoking the DOS SET Command.....	1-12
What If You Get the Message, "Out of environment space"?	1-14

Section 2: ADLIB Tutorial

Ten Steps to Create Your Own MacroFunction	2-2
Have It Your Way!.....	2-12
ADLIB Messages.....	Messages-1
Glossary.....	Glossary-1
Index	Index-1

Illustrations

Figure		Page
1-1	Generic MacroFunction Symbol	1-8
2-1	MacroFunction Logic Title Block	2-2
2-2	MacroFunction Logic Schematic	2-3
2-3	ENCODER.LEF	2-6
2-4	ENCODER.SDA	2-8
2-5	ENCODER Test Design	2-9
	ADLIB MacroFunction Worksheet	2-15

SECTION 1

Getting Started

This section provides the following information:

- A functional description of ADLIB
- A list of the software tools needed to create a MacroFunction
- A list of steps recommended for creating and testing your MacroFunction
- Instructions on how to invoke ADLIB
- A list of design guidelines

Functional Description

The Altera Design Librarian (ADLIB) enables you to create your own MacroFunctions, thus greatly increasing your range of design possibilities. ADLIB takes the data of a Logic Equation File (<filename>.LEF) generated by the Altera Design Processor (ADP), creates a library entry of this MacroFunction, and converts the data into a MacroFunction Symbol Drawing Area file (<filename>.SDA). These SDA files are stored as area files that may be used in logic designs created with the LogiCaps schematic capture program.

MacroFunction logic designs and symbols created with ADLIB may incorporate any number of existing designs and symbols, as well as TTL MacroFunctions from the Altera-supplied library. ADLIB, therefore, is a powerful tool for creating hierarchical designs with many nested levels.

Required Software Tools

To generate your own customized MacroFunctions, you need the following software packages:

- A+PLUS version 5.0
- LogiCaps version 1.6
- TTL MacroFunction Library (PLSLIB–TTL) version 1.2
- Altera Design Librarian (ADLIB) version 1.0

In addition, to be able to verify the validity of your design, you should have the following software package:

- Functional Simulator (FSIM) version 2.5

Where to Begin?

To create a valid, customized MacroFunction and incorporate it into the MacroFunction library, follow these steps:

1. Create the desired logic design with LogiCaps. The title block for the design must specify **MACRO** in the EPLD field to ensure correct design processing.
2. Compile the design with A+PLUS. When you submit the design to the Altera Design Processor (ADP), **LEF Analysis** will be invoked automatically. The LEF Analyzer converts the Minimizer output into human-readable format similar to that of the Altera Design File (ADF).
3. Create a library entry for your new MacroFunction with the ADLIB program.
4. Create a small test design that includes the new MacroFunction.
5. With the Functional Simulator program, simulate the design. *This step is important for verifying the design's accuracy.*
6. Document your customized MacroFunction in a format similar to that of the Altera-supplied MacroFunctions (i.e., symbol, function table, schematic).

Refer also to *ADLIB Tutorial*.

Invoking ADLIB

To invoke ADLIB, type at the DOS prompt:

```
ADLIB <library_name> -option {<macrofunction_name>}
```

where

<library_name> may be any user-assigned filename *except* **MACRO**. The file will automatically be assigned the extension **.LIB** (default). If you enter a filename with another extension, ADLIB will override the extension and interpret the file as a **.LIB** file.

<macrofunction_name> may be one or more files with the extension **.LEF**. The file will automatically be assigned the extension **.LEF** (default). If you enter a filename with another extension, ADLIB will override the extension and interpret the file as a **.LEF** file.

option is one of the following:

-a Add one or more MacroFunctions to your existing library of customized MacroFunctions. In the example below, the files **MF1.LEF** and **MF9.LEF** will be added to the library **LIBRARY.LIB**:

```
ADLIB LIBRARY -a MF1 MF9
```

-c Create a library of customized MacroFunctions. In the example below, the library **LIBRARY.LIB** will be created:

```
ADLIB LIBRARY -c
```

-d Display the contents of a specified library. In the following example, the contents of **LIBRARY.LIB** will be displayed:

```
ADLIB LIBRARY -d
```

- e Erase one or more MacroFunctions from an existing library. In the following example, the MacroFunctions **MF5** and **MF8** will be removed from the file **LIBRARY.LIB**:

```
ADLIB LIBRARY -e MF5 MF8
```

- h Change the header of an existing library. **ADLIB** displays a header text with each library file. This option allows you to change the header text without changing the name of the library. If you invoke the **-h** option without a filename, it will default to the header **USER LIBRARY**. In the following example, the current header of the file **LIBRARY.LIB** will be removed and replaced with the header **MACROS**:

```
ADLIB LIBRARY -h MACROS
```

- r Redefine one or more existing MacroFunctions. You use this function if you wish to edit the logic design of a MacroFunction while retaining its name. In the following example, changes made to MacroFunctions **MF3** and **MF712** are saved:

```
ADLIB LIBRARY -r MF3 MF712
```


Design Guidelines

When designing your own MacroFunction, you should keep in mind the following design rules to ensure its correct operation. Refer to Figure 1-1 for a generic MacroFunction symbol.

MacroFunction Symbol

The logic design for your MacroFunction may contain any Altera primitive *except*: LINP, CORF, ROCF, ROLF, BUSX, LBUSO, LBUSI, RBUSI, LIMP8, RIMP8.

The logic design for your MacroFunction may contain up to 1000 nodes.

All customized MacroFunctions may be stored as schematic areas for LogiCaps. If you wish to use a specific MacroFunction, load it with the **AL** (Area Load) command. You should place all your MacroFunctions into a separate subdirectory.

No I/O pins are included in a MacroFunction; the logic is always buried.

The appearance of your customized MacroFunction symbol is basically determined by two entries in the title block of the LogiCaps schematic:

1. *EPLD field*: it must contain the specification **MACRO**.
2. *Number field*: it may indicate the number of stubs that your symbol will contain. This entry controls the size of the symbol generated. ADLIB will automatically determine the size of symbol required for your logic design and make the stub assignments. However, you may override ADLIB's decisions (see *Have It Your Way!*).

Figure 1-1 shows a generic MacroFunction symbol:

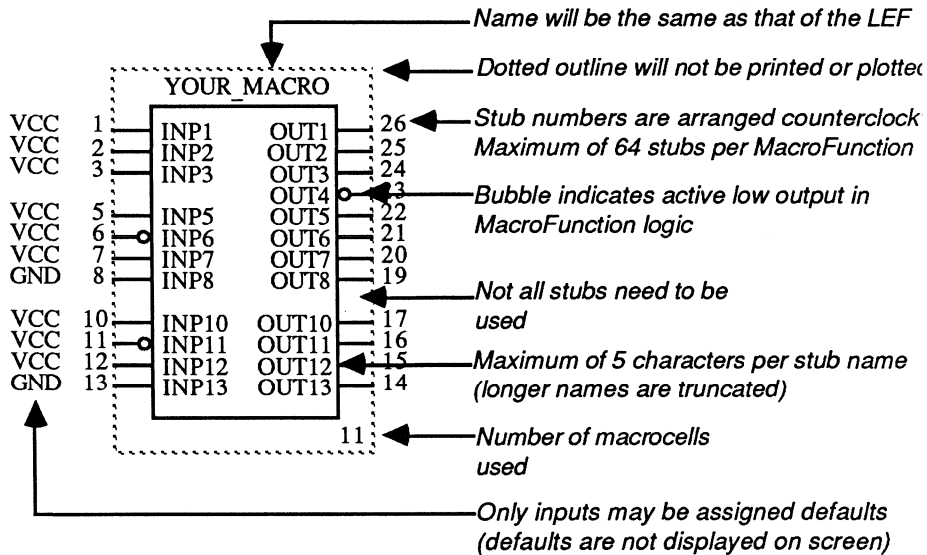


Figure 1-1. Generic MacroFunction Symbol

MacroFunction Stubs

Names assigned to MacroFunction stubs may consist of alphanumeric characters and the underscore character (`_`). At most five characters may be used (excluding `@`, `/`, `#`, and pin number). If a pin name is longer, ADLIB will truncate it in the MacroFunction symbol.



You may enter stub names in upper- or lowercase characters, but they will appear in uppercase in the MacroFunction symbol.

A leading slash (`/`), i.e., slash *before* a stub name, puts a bubble on the MacroFunction symbol stub associated with that stub name, indicating

an active low signal in the logic design. Note, however, that your logic design *must also* contain the active low signal. The bubble on the stub alone does not implement inversion.

The pound character (#) after a stub name defines the input default value. #VCC and #GND are the only valid default values.

The at-symbol (@) following a stub name assigns a specific stub location number to a stub name.

Each MacroFunction symbol may contain up to 64 stubs.

Examples: /CLEAR#VCC
LOAD#GND@5 (the default must precede the number)
/CLOCK@20

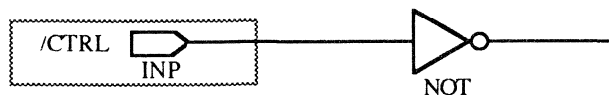
Unless you specifically request a certain stub arrangement on the MacroFunction symbol (see *Have It Your Way!*), ADLIB will arrange the stubs for you. Inputs are automatically placed on the left, outputs on the right. ADLIB will also choose an appropriate symbol size to accommodate the number of inputs and outputs.

Active Low Input and Output

The name and visual appearance of a stub do not change its logical behavior. For example, if you add a bubble to a stub, you are not changing the signal to active low; the bubble will serve only as a reference for you. The original logic design you create with LogiCaps must contain the active low signal to affect the logical behavior of the stub. See the illustration below:

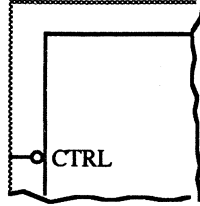
Example 1

MacroFunction logic for active low input:



The NOT gate creates active low input

MacroFunction symbol representing the active low input logic:

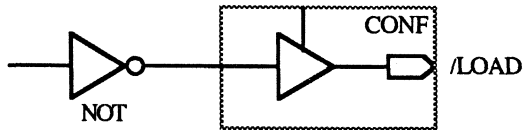


Example 2

MacroFunction logic for active low output.

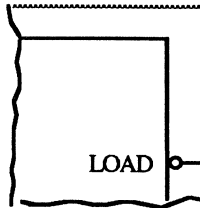


To create active low output, you must insert the inverter in the original design.



The NOT gate creates active low output

MacroFunction symbol representing the active low output logic:



Defaults

Default values may be assigned to inputs only. If you assign a default value to an output, you will get a warning message and ADLIB will ignore the default (see *ADLIB Messages*).

Default values are assigned with the pound (#) symbol: #VCC and #GND are the only valid default values.



VCC and GND *must* be in uppercase letters.

An input stub without default assignment *must* be wired to a signal. MacroFunctions containing stubs that are not wired to a signal or that lack a default assignment cause the Altera Design Processor (ADP) to generate the error message **Node missing source**.

MacroFunctions created with ADLIB contain buried logic definitions *only*. There are no Output Enable controls to tri-state buffers. An Output Enable *must* be connected to VCC or *must* default to VCC, otherwise ADLIB issues a warning message.

How Does A+PLUS Find a Custom-Made MacroFunction?

The Flattener module checks an Altera Design File (ADF) submitted to the Altera Design Processor (ADP) for all MacroFunction statements, whether they be generated from Altera-provided MacroFunctions or from custom-made ones. It then expands these statements and replaces them with the appropriate primitive statements.

All Altera-provided MacroFunctions are located in **MACRO.LIB**, while your custom-made MacroFunctions may be located in any one of your MacroFunction libraries *except* **MACRO.LIB**. In order to find all MacroFunctions, the Flattener goes through a systematic search in the following sequence:

1. First, the Flattener checks in the current directory for **MACRO.LIB**.
2. Next, the Flattener searches the directories in the **PATH** for **MACRO.LIB**.
3. Then, the Flattener looks for any other libraries with the extension **.LIB** that you have specified in the environment string **ADLIB**. Refer to **Invoking the DOS SET Command**.

Invoking the DOS SET Command

Since you may have a large number of custom-made MacroFunctions in various libraries and directories, and the Flattener must be able to locate all of them, you must first set the environment string **ADLIB** with the **DOS SET** command *before* invoking **A+PLUS**. This command makes the string **ADLIB** available to all commands and applications, enabling the Flattener to find specified custom-made MacroFunction no matter where they are.

To invoke the **SET** command, type at the DOS prompt:

```
SET ADLIB=[parameter]
```

where

ADLIB is the name of the variable added to the environment;

parameter is a series of strings specifying drives, directories, and MacroFunction libraries.

For example, the parameters for ADLIB might be as follows:

```
SET ADLIB=C:\DIR1\LIBRARY1.LIB;A:\DIR2\DIR3\
DIR4\LIBRARY2.LIB <Enter>
```

where:

C:	is a drive name
DIRECTORY1	is a directory name
LIBRARY1.LIB	is a MacroFunction library name
A:	is another drive name
DIRECTORY2	is a directory name
DIRECTORY3	is a directory name
DIRECTORY4	is a directory name
LIBRARY2.LIB	is a MacroFunction library name



The SET command may *not* span lines!

For complete details on the SET command, refer to your DOS manual.



Since the Flattener always looks for custom-made MacroFunctions in a set sequence, each MacroFunction should be given a unique name. Otherwise, if you happen to have two MacroFunctions with the same names in different libraries, the Flattener will simply take the one it encounters first—which may be the wrong one.

What If You Get the Message, “Out of environment space”?

When you are setting the environment string ADLIB and get the DOS message **Out of environment space**, you should note the following:

If you have DOS version 3.0 or greater...

...increase the environment space by adding the following line to the CONFIG.SYS file:

```
SHELL=COMMAND.COM/P/E:n
```

where **n** is the number of bytes to which to set the environment so that the environmental space will be increased. (The number must be in the range of 160 to 32768; 1024 would be a typical number.) Then reboot your system.



You may add the **SET ADLIB=drive:path\file.lib** command to your AUTOEXEC.BAT file so it will be present each time you start your computer.

If you have a DOS version smaller than 3.0...

...you don't have the **SHELL** command available to you and cannot increase the environment space. However, you may remove some directories from your current path to make more space available.



Refer to your DOS manual for complete details on the **SET** and **SHELL** commands.

After you have set ADLIB, invoke **A+PLUS**.

SECTION 2

ADLIB Tutorial

This section provides the following information:

- A tutorial showing you how to create a MacroFunction
- Additional suggestions for customizing your MacroFunctions
- A worksheet for designing your custom-made MacroFunction

Ten Steps to Create a MacroFunction

In this tutorial, you will create a MacroFunction named ENCODER.

Step 1: Invoke LogiCaps and Enter the Title Block

At the DOS prompt, type:

LOGICAPS <Enter>

Load the title block by typing:

SE TITLE <Enter>

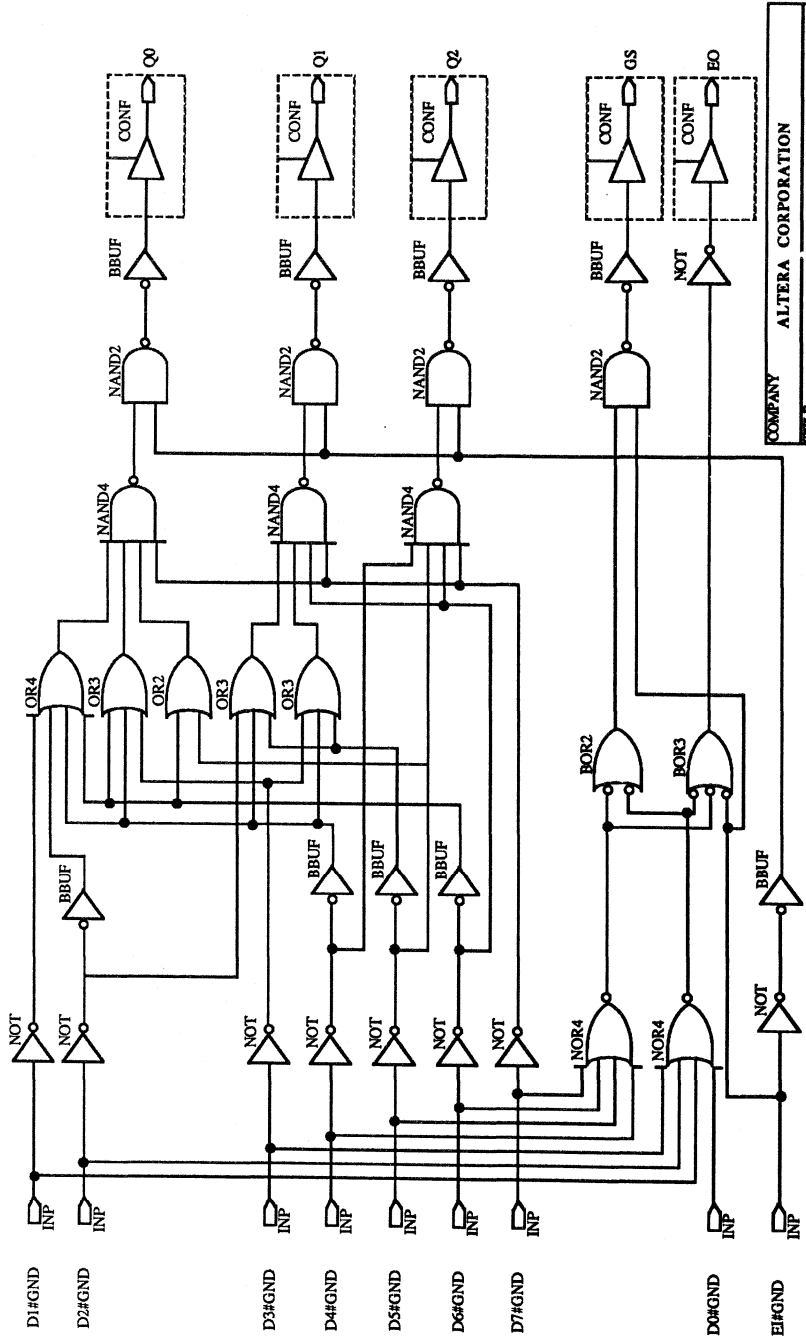
Next, fill in the fields as shown in Figure 2-1. Note that the NUMBER field contains the default number 1.00; it does not specify the number of stubs to be created. ADLIB will automatically create a symbol that best fits the number of stubs needed.

COMPANY				ALTERA CORPORATION			
TITLE				ENCODER			
DESIGNER				YOUR NAME			
SIZE	B	EPLD	MACRO	NUMBER	1.00	REV	A
DATE	SEPTEMBER 15, 1987			SHEET	1	OF	1
TURBO	ON			SECURITY	OFF		

Figure 2-1. MacroFunction Logic Title Block

Step 2: Enter the MacroFunction Logic Design

With the LogiCaps schematic capture program, create the logic design as shown in Figure 2-2.



COMPANY		ALTERA CORPORATION	
TITLE		ENCODER	
DESIGNER		YOUR NAME	
SIZE	EPID	NUMBER	REV
B		1-00	A
DATE	MACRO	SHEET	OF
SEPTEMBER 15, 1987		1	1
TURBO	ON	SECURITY	OFF

Figure 2-2. MacroFunction Logic Schematic

Step 3: Save the Drawing File

Type:

DW ENCODER <Enter>

The logic design will be saved under the name **ENCODER.SD**.

Step 4: Generate the .ADF File for the Logic Design

With the design displayed on screen, type:

DA <Enter>

LogiCaps generates the file **ENCODER.ADF**, displays it on screen, and writes it to disk.

Exit to DOS by typing:

Q <Enter>

Step 5: Invoke A+PLUS

To invoke A+PLUS and display the APLUS menu, type at the DOS prompt:

APLUS <Enter>

Step 6: Submit the .ADF File to the ADP

Press **<F4>** to open the ADP menu.

At the **Input Format** prompt, press **<Enter>**.

At the **File Name** prompt, type **ENCODER <Enter>**.

At the **Minimization** prompt, press **<Enter>**.

At the **Inversion Control** prompt, press **<Enter>**.

At the **LEF Analysis** prompt, type **YES** or **<Enter>**.

Press **Y** to execute the design.



A+PLUS will automatically invoke LEF Analysis if the EPLD field in the title block specifies **MACRO**. So, even if you forget to request LEF Analysis, A+PLUS will request it for you.

The ADP will generate the file **ENCODER.LEF**, similar to the one shown in Figure 2-3.

Type **N**, then **<F2>** to return to DOS.

YOUR NAME
ALTERA CORPORATION
SEPTEMBER 15, 1987
1.00
B
MACRO
ENCODER
LogiCaps Schematic Capture Version 1.6

Input Files: ENCODER.ADF
ADP Options: Minimization=Yes, Inversion Control=No, LEF Analysis=Yes

ANALYZER Version 5.0

OPTIONS: TURBO = ON, SECURITY = OFF

PART:

MACRO

INPUTS:

D1#GND, D2#GND, D3#GND, D4#GND, D5#GND, D6#GND, D7#GND,
D0#GND, EI#GND

OUTPUTS:

E0, GS, Q2, Q1, Q0

NETWORK:

.0021003 = INP(D1#GND)
.0021009 = INP(D2#GND)
.0021030 = INP(D3#GND)
.0021038 = INP(D4#GND)
.0021046 = INP(D5#GND)
.0021054 = INP(D6#GND)
.0021062 = INP(D7#GND)
.0021083 = INP(D0#GND)
.0021093 = INP(EI#GND)
EO = CONF(.0134083, VCC)
GS = CONF(.0134073, VCC)
Q2 = CONF(.0134052, VCC)
Q1 = CONF(.0134034, VCC)
Q0 = CONF(.0134016, VCC)

Figure 2-3. ENCODER.LEF (Part 1 of 2)

EQUATIONS:

```
.0046046 = .0031050 * .0031040' * .0031038' * .0031036' * .0031034'  
          * .0031048' * .0031046' * .0031044' * .0031042' ;  
.0046044' = .0031050'  
            + .0031048' * .0031046' * .0031044' * .0031042' *  
              .0031040' * .0031038' * .0031036' * .0031034' ;  
  
.0046042 = .0031050 * .0031048  
            + .0031050 * .0031046' * .0031044  
            + .0031050 * .0031046' * .0031042 * .0031040  
            + .0031050 * .0031046' * .0031042 * .0031036 * .0031038' ;  
  
.0046040 = .0031050 * .0031046  
            + .0031050 * .0031048  
            + .0031050 * .0031042' * .0031044' * .0031038  
            + .0031050 * .0031042' * .0031044' * .0031040 ;  
  
.0046038 = .0031050'  
            + .0031042' * .0031044' * .0031046' * .0031048 ;
```

END\$

Figure 2-3. ENCODER.LEF (Part 2 of 2)

Step 7: Invoke ADLIB

At the DOS prompt, type:

```
ADLIB LIBRARY -c <Enter>
```

This command performs the following functions:

- It invokes ADLIB.
- It creates a MacroFunction library named **LIBRARY.LIB**.

Then type:

```
ADLIB LIBRARY -a ENCODER <Enter>
```

This command performs the following functions:

- It generates the entry **ENCODER** in **LIBRARY.LIB** in your default directory.
- It generates the file **ENCODER.SDA** that contains the MacroFunction symbol shown in Figure 2-4.

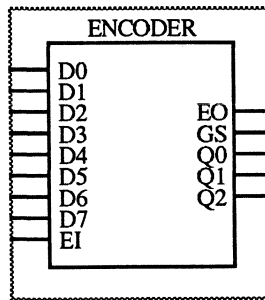


Figure 2-4. ENCODER.SDA

Step 8: Set the ADLIB Environment

To enable the Flattener to locate your custom-made MacroFunction, be sure to use the DOS **SET** command to create an ADLIB environment variable. Type at the DOS prompt:

```
SET ADLIB=[parameters]
```

where

[parameters] consists of the strings that specify drives, directories, and MacroFunction libraries. (Refer to *How Does A+PLUS Find a Custom-Made MacroFunction?*)

Step 9: Test Your MacroFunction

To test the validity of your new symbol, create a small design that incorporates the MacroFunction. Test the design with the Functional Simulator (see *Functional Simulator* in the **A+PLUS User Guide**).

- a. Invoke LogiCaps by typing at the DOS prompt:

LogiCaps <Enter>

- b. Load the MacroFunction with the AL (Area Load) command:

AL ENCODER <Enter>

- c. Create your test design and save it with the name **ENCODER1**. See Figure 2-5.

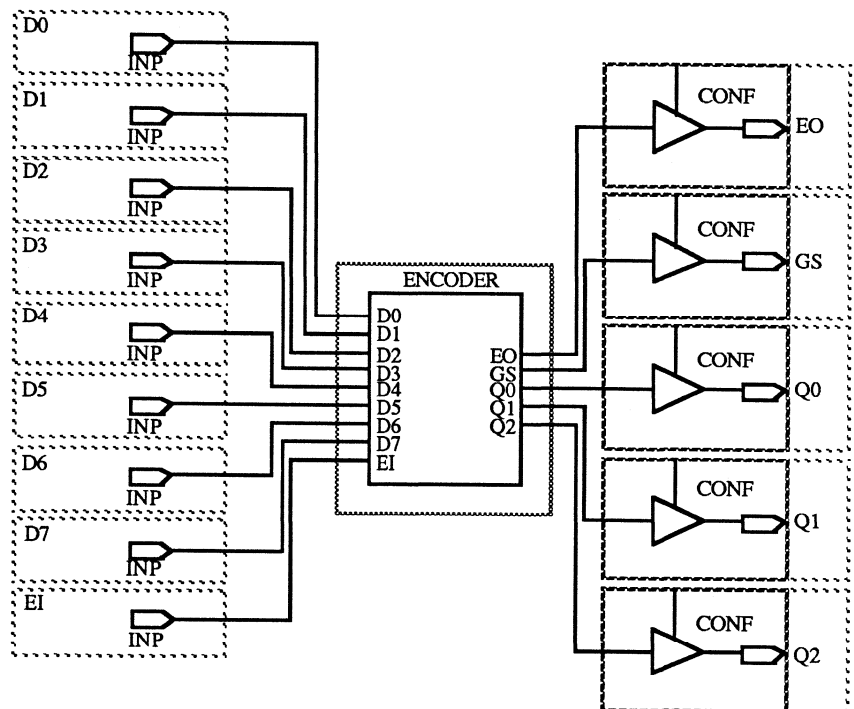


Figure 2-5. ENCODER Test Design

d. Create an ADF by typing:

DA <Enter>

Save the file by typing:

DW ENCODER1 <Enter>

Then quit LogiCaps (Q <Enter>) and submit the ADF to the ADP by typing:

APLUS <Enter>

Press <F4> to open the ADP menu.

At the **Input Format** prompt, press <Enter>.

At the **File Name** prompt, type **ENCODER1 <Enter>**.

At the **Minimization** prompt, press <Enter>.

At the **Inversion Control** prompt, press <Enter>.

At the **LEF Analysis** prompt, type **YES** or <Enter>.

Press **Y** to execute the design.

The ADP will generate the file **ENCODER1.JED**.

Type **N**, then <F2> to return to DOS.

e. With your text processor, create a simple vector file, **ENCODER1.VEC**, for the design:

TABLE: EI D0 D1 D2 D3 D4 D5 D6 D7;

0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	1	0
1	0	0	0	0	0	1	0	0	0
1	0	0	0	0	1	0	0	0	0
1	0	0	0	1	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0

- f. With your text processor, create a command file:

```
VEC @ENCODER1;  
WATCH GS Q0 Q1 Q2 EO @ENCODER1.TBL;  
SIM 10;  
DISP WATCH;
```

Save this file as **ENCODER1.CMD**.

- g. Open the APLUS Menu and select <F6> to invoke FSIM. When you are prompted for the file to simulate, type:

ENCODER1 <Enter>

FSIM will generate the following **ENCODER1.TBL** file:

```
C  
Y  
C  
L G Q Q Q E  
E S 0 1 2 O  
  
1 0 0 0 0 0  
2 0 0 0 0 1  
3 1 1 1 1 0  
4 1 0 1 1 0  
5 1 1 0 1 0  
6 1 0 0 1 0  
7 1 1 1 0 0  
8 1 0 1 0 0  
9 1 1 0 0 0  
10 1 0 0 0 0
```

Simulation cover: 100%

Step 10: Document Your MacroFunction

Document your MacroFunction. See the documentation format of the TTL MacroFunctions in ***TTL MacroFunctions***.



This MacroFunction is a standard CMOS 8-bit priority encoder known as CD4532, documented in the ***RCA DATABOOK (CMOS Integrated Circuits)***.

Have It Your Way!

If you would like additional control over the appearance of your customized MacroFunction symbol, you have the following options:

1. If you know how many input and output stubs your MacroFunction requires, you can enter that number into the NUMBER field in the design's title block. ADLIB will try to accommodate your wishes and build a symbol that comes close to your specifications.

Of course, if the number you enter is much too high, ADLIB will create a big symbol with few actual stubs inserted. On the other hand, if the specified number is much too low, ADLIB will ignore it and use its own best judgment as to the symbol configuration.

For example, if you enter 40 into the title block but your logic design only needs 4 stubs, the symbol created will be large enough for 20 stubs on either side of the symbol, but only four stubs will actually be present. On the other hand, if you specify too few stubs, ADLIB will automatically create a symbol large enough to accommodate all input and output stubs.

2. You can exercise additional control by specifying the exact stub number for a stub name. For example, **LOAD@15** will place the LOAD signal at stub 15. If you choose this option, you should assign *all* signals to stub numbers. If you assign only some signals, ADLIB will try to assign the remaining ones to *any* remaining free stub positions. (Note that the default specification must precede the stub number assignment, i.e., # comes before @.)
3. If you are not quite certain about stub number assignments, you may wish to first run the design through ADLIB without making specific assignments. Then, when you see the symbol created automatically by ADLIB, you can decide which changes you would like, make specific stub assignments, enter the number in the title block of the logic design, and run the design through ADLIB again.
4. If you don't like the appearance of a symbol, you can always adjust the number in the title block and then run the design through the ADP again. Alternatively, you might draw a sketch of

the symbol as you'd like it to appear and then, with the sketch as your reference, use the @ symbol to specify stub numbers for all stub names.

To create MacroFunction symbols most desirable for your designs, you may wish to make a copy of the following worksheet and use it to sketch your symbol.



Altera invites you to submit your original MacroFunction designs to the company. They will be collected into an Altera-customer MacroFunction library, which will be accessible via Altera's Electronic Design Support System to all registered Altera software maintenance agreement owners. Simply submit your complete MacroFunction design **together** with the completed *Altera Product Comment Form* that is at the back of this manual.

the symbol as you'd like it to appear and then, with the sketch as your reference, use the @ symbol to specify stub numbers for all stub names.

To create MacroFunction symbols most desirable for your designs, you may wish to make a copy of the following worksheet and use it to sketch your symbol.



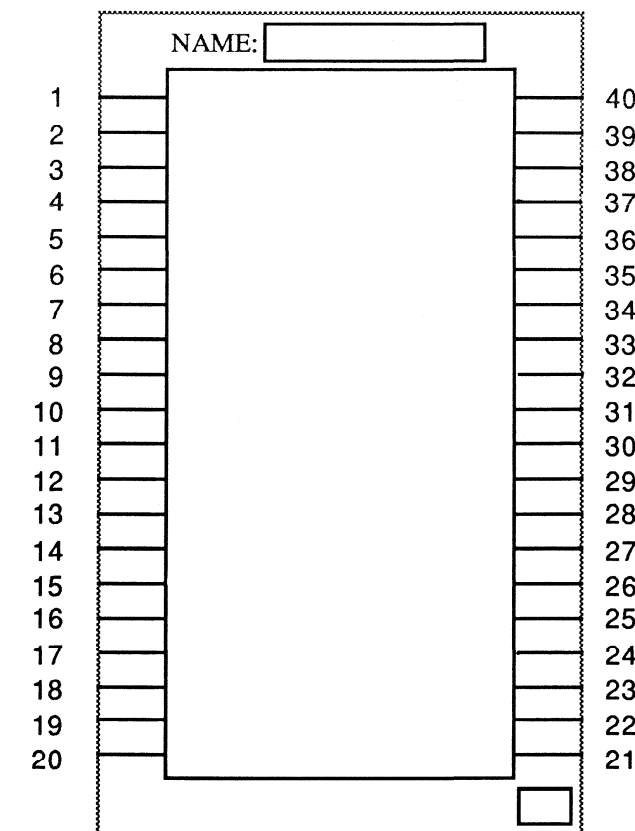
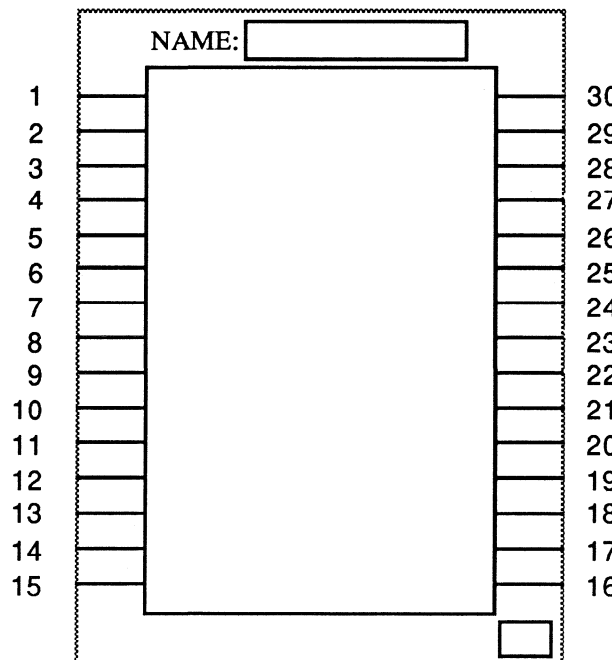
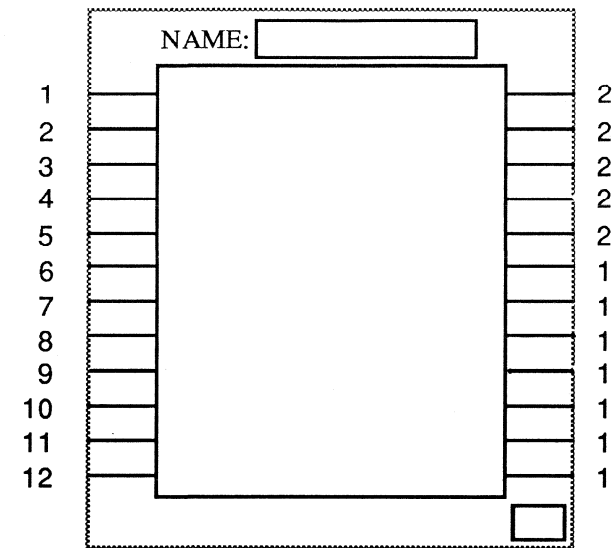
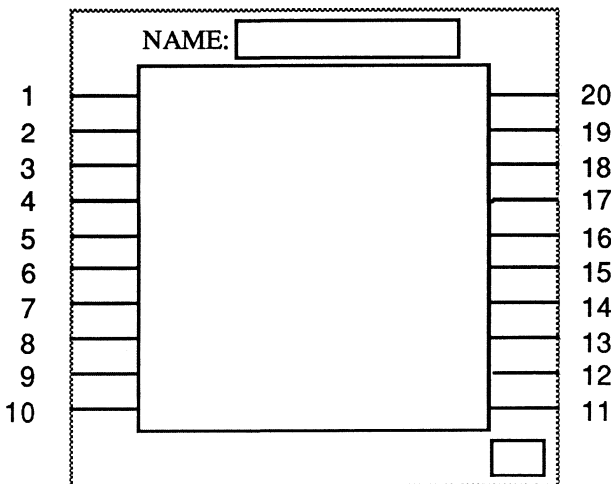
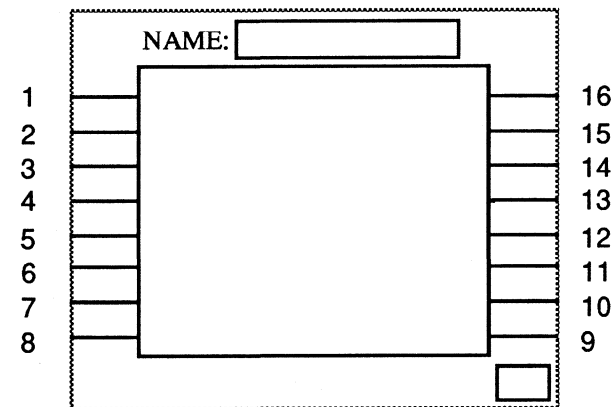
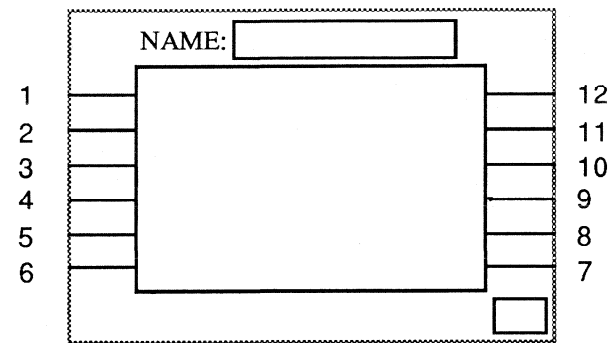
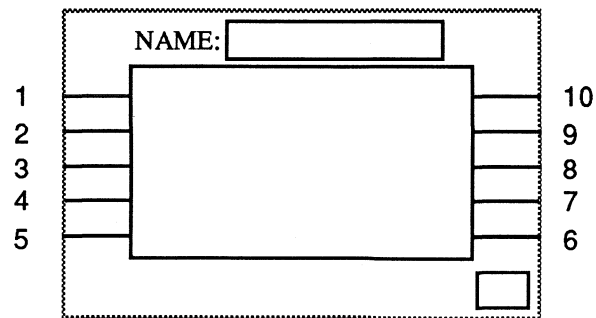
Altera invites you to submit your original MacroFunction designs to the company. They will be collected into an Altera-customer MacroFunction library, which will be accessible via Altera's Electronic Design Support System to all registered Altera software maintenance agreement owners. Simply submit your complete MacroFunction design **together** with the completed *Altera Product Comment Form* that is at the back of this manual.

ADLIB MacroFunction Worksheet

This worksheet shows seven generic MacroFunction symbols representing some of the most frequently used pin stub arrangements.

Note the following design restrictions:

1. Name is the same as that of the LEF.
2. Stub numbers are arranged counterclockwise.
3. Maximum of 64 stubs per MacroFunction.
4. Bubble indicates active low input or output in MacroFunction logic.
5. Not all stubs must be used.
6. Maximum of 5 characters per stub name (longer names are truncated).
7. Use the # symbol to define the input default value (VCC or GND).
Only inputs may be assigned defaults.
8. Use the @ symbol to assign a stub number.



ADLIB Messages

The Altera Design Librarian (ADLIB) generates error, warning, and information messages essential for correcting and improving your design. These messages are written to the screen.

ADLIB messages are usually identified with a reference text indicating where they originated, followed by **Error:**, **Warning:**, or **Info:**. Error messages indicate that ADLIB will stop the current process to allow you to decide whether you wish to continue. Some error messages are fatal, i.e., they will interrupt processing and return you to DOS. Information and warning messages do not require a response, but warning messages usually indicate a design feature that should be changed.

All ADLIB messages are listed alphabetically below. Each message is accompanied by an explanation and suggestions for corrective action (if necessary).

Bad LEF file

- CAUSE:** You entered a file that is not a valid .LEF file. Maybe it has been corrupted, or your design did not produce a .LEF file valid for an entry in ADLIB.
- ACTION:** Check your logic design, submit it to the ADP again, and generate a new .LEF file.

Bad library file

- CAUSE:** The library file you specified is not recognized by ADLIB.
- ACTION:** Enter a valid library, created with ADLIB.

Cannot close file

- CAUSE:** Your system's memory and disk are full. The file could not be closed.
- ACTION:** Make enough memory available and process your design again.

Cannot create new library <filename>

- CAUSE:** You probably don't have enough available disk space to create a new library.
- ACTION:** Check your disk space and make enough memory available.

Cannot open file

- CAUSE:** You tried to open a file that does not exist or you ran out of disk space.
- ACTION:** Check to see if the file exists or verify that you have enough available memory.

Cannot open temporary file

- CAUSE:** You don't have enough available disk space.
- ACTION:** Try to free some disk space.

Cannot use MacroFunction file(s) with this option

- CAUSE:** You entered one or more filenames with an ADLIB command option that does not require a MacroFunction filename (-c, -d).
- ACTION:** Re-enter the command without the filename specification.

Cannot write file

- CAUSE: The file was not written because you don't have enough available disk space.
- ACTION: Make enough memory available and process your design again.

Could not make backup for <filename>

- CAUSE: The system was not able to make a backup of the specified file. It is probably low on disk space.
- ACTION: Try to free additional memory and process the file again.

Default value has been assigned to output <output name>

- CAUSE: You have assigned a default value to an output stub name. Outputs cannot accept defaults. ADLIB will ignore the default.
- ACTION: None—just a friendly warning from your neighborhood librarian!

Header truncated to <str>

- CAUSE: A library file header may be up to 59 characters long. If it is longer, ADLIB simply truncates it.
- ACTION: Just a warning, no action required.

Ignored output enable for stub <stub name>

- CAUSE: You assigned an output enable to a stub name. Output enable must be connected to VCC or default to VCC.
- ACTION: This is a warning only. ADLIB will ignore your assignment.

Internal Error: <text>

- CAUSE: You have encountered an internal error .
- ACTION: Please call Altera Applications and supply the module name and text associated with the internal error.

Invalid primitive <primitive name> in line <n>

- CAUSE: Your MacroFunction logic design contains one or more of the following primitives: LINP, CORF, ROCF, ROLF, BUSX, LBUSO, LBUSI, RBUSI, LINP8, RINP8. MacroFunctions may not contain any of these primitives.
- ACTION: Edit your design to eliminate the offending primitive(s) and resubmit it to the ADP to generate another .LEF file.

Library full

- CAUSE:** Your library file already contains the maximum of 256 MacroFunctions, or it may have been corrupted.
- ACTION:** Create a new library file.

Library name MACRO.LIB is reserved for Altera use

- CAUSE:** You tried to name your library **MACRO**. Or you tried to add a custom-made MacroFunction to **MACRO.LIB**. This name is reserved, and you may not add your own MacroFunctions to it.
- ACTION:** Give your library another name, or put your MacroFunction into a library other than **MACRO.LIB**.

MacroFunction already exists in library

- CAUSE:** You specified a MacroFunction name that already exists in your library.
- ACTION:** Rename your current MacroFunction. If you want to replace the old MacroFunction with the new one but retain the old name, you must use the **-r** option.

MacroFunction contains duplicate stub names <name>

- CAUSE:** You have probably specified the same stub name for two stubs, one with a leading slash to indicate inversion, one without. ADLIB interprets these as the same name.
- ACTION:** Edit your logic design so that you have two different names, generate a new .LEF file, and submit this file to ADLIB.

MacroFunction has too many symbol stubs <n>

- CAUSE:** You have specified more than 64 symbol stubs. A MacroFunction may have a maximum of 64 stubs.
- ACTION:** Edit your logic design, breaking it into more than one portion if necessary.

MacroFunction has too many nodes

- CAUSE:** The logic design you created for the MacroFunction contains more than 1000 nodes. ADLIB cannot create MacroFunction for this design.
- ACTION:** Break your design into smaller portions and generate more than one MacroFunction.

MacroFunction not found in library

- CAUSE:** You tried to refer to a MacroFunction that does not exist in your library file.
- ACTION:** Make sure that you enter the name correctly and that you are in the correct library.

Requested stub number is too large <name>@<n>

- CAUSE:** You assigned a stub number to a stub name that is greater than 64. MacroFunction symbol stubs are numbered from 1 to 64.
- ACTION:** Choose a number between 1 and 64.

Symbol has <n> Input(s), <n> Output(s), <n> MCells

- CAUSE:** ADLIB has created a MacroFunction symbol for your logic design and is summarizing the results for you.
- ACTION:** This is an information message, no action is required.

Stub name <name> has been truncated

- CAUSE:** You entered a symbol stub name that is longer than five characters. ADLIB truncates names exceeding five characters.
- ACTION:** This is a warning message. You need not do anything if you can live with a truncated na...

This option requires LEF file(s)

- CAUSE:** The ADLIB command option (-a, -e, or -r) you entered requires a .LEF filename specification. You did not enter a filename.
- ACTION:** Enter the command with the desired filename.

Unrecognized primitive <name> on line <n>

- CAUSE:** Your .LEF file contains a primitive that is not a valid Altera primitive .
- ACTION:** Check your logic design and edit it if necessary.

Glossary

- .ADF (Altera Design File)** The filename extension assigned automatically to an Altera Design File generated from a schematic. This file is entered into the Altera Design Processor (ADP) for further processing
- .CMD (Command file)** A file containing commands that guide the simulation process. It consists of a list of instructions created with a text editor, and is used only when the Functional Simulator is run in batch mode.
- .JED (JEDEC file)** An industry-wide standard for the transfer of information between a data preparation system and a logic device programmer. The ADP converts your input ADF file into a JEDEC file that describes your design.
- .LEF (Logic Equation File)** A data structure used by several Altera Design Processor modules. In the LEF, the Boolean portion of the design is separated from the non-Boolean portion to allow down-stream programs to process the design according to their own specifications. In the case of ADLIB, the .LEF file has the name of a custom-made MacroFunction.

- .LIB (MacroFunction library file)** A file recognized by ADLIB as containing custom-made MacroFunctions. The file may not be named **MACRO**, since this name is reserved for Altera's MacroFunction library file **MACRO.LIB**.
- .SD (Symbol Drawing file)** A filename extension used in LogiCaps. It is assigned automatically to a schematic when it is saved.
- .SDA (Symbol Drawing Area file)** A filename extension used in LogiCaps. It is assigned automatically to an area (including MacroFunctions) when it is saved.
- .TBL** The vector table output file generated by the Functional Simulator (FSIM). It contains a tabular description of the nodes requested by the **WATCH** command and describes the state of the nodes in 1s and 0s.
- .VEC (Vector file)** A file describing the input waveforms on which the simulation is to be based. It contains the vectors that specify the levels for individual inputs and outputs within a design.
- ADLIB (Altera Design Librarian)** An Altera software program that enables you to create your own MacroFunctions. It takes the data of a Logic Equation File generated by the Altera Design Processor and converts these data into a MacroFunction Symbol Drawing Area file. These files may then be used in future logic designs created with LogiCaps.
- ADP (Altera Design Processor)** That portion of A+PLUS that processes the Altera Design File. It is the user-interface to the A+PLUS software. It control the individual modules and converts the ADF into a JEDEC file which is used to program an Altera EPLD.
- A+PLUS (Altera Programmable Logic User System)** A set of computer programs and hardware support devices that facilitate design and implementation of custom logic circuits with Altera programmable logic products.
- AUTOEXEC.BAT** A DOS system file that is modified during installation of A+PLUS software. (The original AUTOEXEC.BAT file is saved as AUTOEXEC.BAK.) This file is executed by DOS whenever the computer is booted.
- buried register** A register in an Altera EPLD that is not associated with any pin and can be used to implement internal logic. It also is any register that does not need an output at a chip pin.
- CONFIG.SYS** A DOS system file that is modified during installation of A+PLUS software. (The original CONFIG.SYS file is saved as CONFIG.BAK.) DOS uses this file to set up the DOS environment.

- EPLD** Erasable Programmable Logic Device, i.e., any member of the Altera family of parts.
- EPLD.SYS** The A+PLUS hardware installation file. You may access this file via the installation menu if you wish to change the address location of the programming card or disable the color display option of LogicMap II.
- Flattener** The Altera Design Processor module that checks every Altera Design File for MacroFunction statements. It expands these statements and replaces them with the appropriate primitive statements. A flattened file has the extension .SDF. The flattening process does not alter the electrical connectivity of the original design.
- Functional Simulator (FSIM)** An Altera software program that tests the logical operation of your A+PLUS design. FSIM uses specified design and part information to model the operation of the part before the design is actually committed to hardware. FSIM can be run in either interactive or batch mode. It outputs a graphical waveform description of the simulated design (.WAV file) and a log of FSIM commands used for execution (.LOG file).
- LEF Analyzer** The ADP module that converts a binary Logic Equation File output of the Minimizer module into a human-readable file.
- LogiCaps** Altera's logic schematic capture program used to create your MacroFunctions.
- LogicMap II** The interface between the JEDEC file and the programming unit. The LogicMap II program allows the user to program an EPLD. It performs timing and transfer functions for A+PLUS programming. Data are stored in standard JEDEC format.
- MACRO.LIB** The file containing all MacroFunctions supplied by Altera. You may not add your own MacroFunctions to this file or alter it in any other way.
- MacroFunction** A high-level building block used together with existing gate and flipflop primitives to provide a versatile design environment for EPLD logic development.
- MacroFunction library** A library file containing your custom-made MacroFunctions. It may not have the name **MACRO**.
- Minimizer** The ADP module that takes the Boolean expressions output by the Expander module and reduces them to determine whether the product term count is within the part-imposed limits.

READ.ME File A file on the A+PLUS INSTALL and the SAM+PLUS INSTALLS distribution diskettes. It contains information about changes made to the installation procedure or other programs since the release of the documentation, and should be read before A+PLUS and SAM+PLUS are installed.

Security bit A control bit that, when set to ON, prevents interrogation or inadvertent reprogramming of an Altera EPLD .

title block A schematic symbol required for all logic schematics created with LogiCaps. It is used to document a schematic.

tri-state buffer A buffer with an input, output, and controlling signal. If the controlling signal is high, the output is a defined function of the input. If the controlling signal is low, the output is not a defined function of the input.

Turbo-Bit A control bit that, when set to ON, allows you to choose the speed and power characteristics of an Altera EPLD.

Index

.ADF 2-4
.LEF 1-5
.LIB 1-5, 12
<filename>.LEF 1-2
<filename>.SDA 1-2
<library_name> 1-5
<macrofunction_name> 1-5
@ 1-9

A

A+PLUS 1-3
active low signal 1-9
active low input assignment 1-9
active low output assignment 1-9, 10
ADLIB definition 1-2
advanced features 2-12
AL (Area Load) command 1-7; 2-9
Altera Design File 1-4, 12; 2-4

Altera Design Librarian (ADLIB) 1-2
Altera Design Processor 1-2, 4, 12
assigning a stub location 1-9
assigning default values 1-11
AUTOEXEC.BAT 1-14

B

bubbles in a design 1-8
buried logic 1-11

C

changing the library header text 1-6
CONFIG.SYS 1-14
controlling the symbol size 2-12
creating a MacroFunction 1-4; 2-2

D

default library header 1-6
default library filename 1-5
default MacroFunction filename 1-5
default values 1-11
design guidelines 1-7; 2-12
DOS SET command 1-12, 14; 2-8

E

ENCODER.ADF 2-4
ENCODER.LEF 2-5, 6
ENCODER.SD 2-4
ENCODER.SDA 2-8
ENCODER1 2-9
ENCODER1.CMD 2-11
ENCODER1.JED 2-10
ENCODER1.TBL 2-11
ENCODER1.VEC 2-10
entering a bubble 1-8
entering the title block 2-2
environment string 1-12, 14; 2-8
EPLD field 2-5

F

Flattener module 1-12, 13
FSIM 1-3; 2-11
functional description 1-2

I

I/O pins 1-7
invoking A+PLUS 2-4
invoking ADLIB 1-5; 2-7
invoking LogiCaps 2-2
invoking the DOS SET command 1-12; 2-8

L

LEF Analysis 1-4; 2-5
LEF Analyzer 1-4
LIBRARY.LIB 2-7, 8
loading a MacroFunction 1-7; 2-9
locating a MacroFunction 1-12
Logic Equation File (LEF) 1-2, 5
LogiCaps 1-2, 3; 2-2

M

MACRO.LIB 1-12
MACRO in title block 1-4, 5, 7; 2-5
MacroFunction Symbol Drawing Area file 1-2
MacroFunction templates 2-15

N

naming a MacroFunction 1-13
"Node missing source" message 1-11
NUMBER field 2-2

O

options for invoking ADLIB 1-5
"Out of environment space" message 1-14
Output Enable controls 1-11

P

PLSLIB–TTL 1-3

R

redefining a MacroFunction 1-6

restrictions 1-7; 2-12

S

saving a drawing file 2-4

SDA file storage 1-2

SET ADLIB 1-12, 13; 2-8

SET command 1-12, 14

setting the ADLIB environment 2-8

setting the environment string 1-12; 2-8

SHELL command 1-14

size of the symbol 1-7; 2-12

software requirements 1-3

specifying the symbol size 2-12

storing MacroFunctions 1-7

stub assignments 1-7

stub name restrictions 1-8

T

template for MacroFunctions 2-15

testing the MacroFunction 2-9

title block 1-4, 7

TTL MacroFunctions 1-2, 3

tutorial 2-1

U

USER LIBRARY (default header) 1-6

V

valid default values 1-9

verifying a design 1-3

W

worksheet template 2-15