Ryerson Polytechnic University
Department of Electrical and Computer Engineering
ELE 328 – Digital Systems and Microprocessors

Lab 1
CAD Tools **10 marks** (2 Weeks)
Fall 2002 Due Date: Week 4

Part A – PALASM "Tutorial Introduction"

A.1 Objectives:

- The main objective of this lab is to know how to use PALASM software. PALASM is used for Synthesis of digital logic design using PALs (programmable devices which can replace conventional logic circuits).

A.2 Installing Your Own Copy of PALASM for Home Use – Done later

- System Requirements
  - IBM PC/XT using DOS version 3.0 or later
  - 1.44 MB/3.5 in. floppy drive for software installation
  - Available memory for PALASM > 512 KB
  - Available Hard disk space > 10 MB

- License: Advance Micro Devices (AMD) has authorized PALASM software for unrestricted student use. The software is available to the student on the network in system directory /pub/DOS/palasm
- PALASM Files
  - On the network, cd /pab/DOS/palasm
  - Floppy disk No.1 in drive "A"
    cd /disk1
    mcopy * a:
  - Floppy disk No.2 in drive "A"
    cd /disk2
    mcopy * a:
  - Floppy disk No.3 in drive "A"
    cd /disk3
    mcopy * a:
  - Optional disk
    Floppy disk No.4 in drive "A"
    cd /disk4
    mcopy * a:
- Installation
  - Place installation disk 1 into drive "A".
  - Type "install" and [enter].
    The AMD copyright screen appears. Press any key.
  - Proceed with the installation

A.3 Procedure – Preparing the Program File

A suitable PAL programming file must be created before we can proceed with the compilation, simulation, and ultimately, the actual device programming processes. For this project #1 Part A, we prefer to use "lab1.pds" for the name of such a file (created in Lab 0 from page 7 lab1).
Any editor program may be used to prepare the file.
If you use a word processing program, be sure to create the file in such a way that the file contains no special formatting characters. In some word processors (e.g. WordStar), use the *non-document* mode. In others, it is *saved ASCII* or *print to disk* to create the file.
After the file has been created, it is a good idea to use the DOS **TYPE** command to examine the file. It should contain only printable characters.
Note on VIRUSES: Almost all students' disks are contaminated with viruses. A virus on your disk will eventually corrupt program and/or data files.
Use the **ALERT.EXE** program to check your disk and remove viruses. Instructions are contained in the document file (*.doc).
We use the JOVE editor from any workstation[1] to prepare the required file.
The file should be edited according to the following format:

- Edit the **TITLE, PATTERN, REVISION, AUTHOR, COMPANY** and **DATE.**
- Select the PAL configuration for **CHIP.** We are using PAL16V8, to be proceeded by the label lab1
- Assign the 20 pins of the PAL to the input and the output signals like A,B,C,D,… and F1,F2,… Assign pin#10 to GND, and pin#20 for VCC. Use NC (no connection) for pins with no connection
- Write the logic **Equations** for the output signals as a function of input signals like F1=A*B*C + /A*B*/C +…
- Write test vectors for **SIMULATION**. This is important to test the function for given inputs. Use "**TRACE-ON**" to set the probes on some (or all) input and output signals that we want to test. Set signal values by using "SETF". For example, SETF /A B /C D /E means to test the value of F1 when A=0, B=1, C=0, D=1 and E=0. We would simulate every input condition like the truth table. Use "**TRACE-OFF**" to end the simulation.
- Store[2] labl.pds file on a 3.5-inch floppy disk.

A.4 Procedure – Compiling and Simulation

To compile and simulate labl.pds at the PAL station, we do the following sequence of steps:

- Insert your floppy disk (with the *labl.pds* file) into drive A: of the PAL station.
- Type **palasm**, hit < **cr** > key, then press any key to start the PAL session.
- Change the directory to A: from the file menu.
- Type **r** to *retrieve existing design.* Hit < **cursor-down** > once, then press < **cr** > to list the *.pds files from the A: drive.
- Move cursor down to highlight *labl.pds,* and then hit < **cr** > to make the choice.
- Press < **F10** > key to accept the file selection.
- Move cursor right to open the *RUN* menu, and type **b** to select *BOTH.* Make sure the log file name is lab1.log. (If not, type it in.)
- Press < **F10** > key *twice* to proceed to the compilation and simulation. This process would last a minute or two. If all goes well, there should be a zero count of error (or warning).

[1] It is possible to create this file directly from the PAL station.
[2] Done in lab 0.

- Press < **Esc** > key *once* to return to the menu.
- Select the View option and then the select Waveform display and then the History option. Save it by using the F2 key and choosing file as the printer. Enter file name e.g. wave1, and then use right arrow to select run. Return to view menu.
- Then select Pin out and create the file labl.pin. Use F2 to save it as labl.pin and then exit to DOS by hitting the escape key twice and the y to confirm.
- Now print the above two files by typing copy wavel lptl and copy labl.pin lptl.

Show copies of your printouts identified with your name to your instructor.

A.5 Programming the PAL - Optional (See following pages for ChipLab, Data I/O ChipWriter and Palasm examples)

You will need these notes for later labs.

To program our PAL[3] chip, we do the following step sequence:

- Make sure you are in A: directory.
- Type **9860** to start the programmer software package.
- Type **i** to select the *input file from disk* option from menu.
- Type **labl.jed** < **cr** >.
- Press < **Esc** > key to escape back to the main menu.
- Type **c** to select the *change device* option. (This opens up the *Manufacturer List).* Proceed to make the actual selection of *Make & Type* accordingly. Use the < **cr** > key to confirm each correct entry.
    - If our GAL chip is made by Lattice, select **16** from the manufacturer menu, and then select **00** (on page 1 of 4) from the device sub-menu (for GAL16V8/A/B/C/Z).
    - If our GAL chip is made by Advanced Micro Device (AMD), select **01** from the manufacturer menu, and then select **14** (on page 2 of 5) from the device sub-menu (for PALCE16V8H/4).
- Insert tine GAL chip into the programmer socket. Pin#10 & pin#11 must be placed at the bottom (near the front) of the socket, then close the socket.
- Type **p** (*Program device* option from the menu) to start the PAL programming. Type **y** to proceed.
- Type **v** *(Verify device* option from the menu) to be sure that the programming is successful.
- Remove the GAL chip from the socket, and type **q** to quit from the program.

[3] Actually, the chip we use is GAL

A.6 An Example of the PALASM File

```
TITLE          A VOTING MACHINE USING PALS
PATTERN        LAB1.PDS
REVISION       A
AUTHOR         YOUR NAME
COMPANY        RYERSON
DATE           MM/DD/01

CHIP lab1 PAL16V8 ; USE "VERSATILE" TYPE PLD.

NC A  B  C  D  NC  NC  NC  NC  GND
NC  NC  TIE  NO  YES  NC  NC  NC  NC  VCC

;The following illustrates a few basic principles in writing a PALASM program.
;Design a PAL circuit that will display the results of a vote among 4 persons A, B, C and D.
;INPUTS:     [ABCD]          yes = 1
;                            no = 0
;OUTPUTS:    YES ("yes" majority) = 1
;            NO ("no" majority) = 1
;            TIE (tie vote) = 1

Equations

YES = /A*B*C*D + A*/B*C*D + A*B*/C*D + A*B*C*/D + A*B*C*D
NO = /A*/B*/C*/D + /A*/B*/C*D + /A*/B*C*/D + /A*B*/C*/D + A*/B*/C*/D
TIE = /A*/B*C*D + /A*B*/C*D + /A*B*C*/D + A*/B*/C*D + A*/B*C*/D + A*B*/C*/D

SIMULATION

TRACE_ON  A  B  C  D  YES  NO  TIE  ;Determines signals to be displayed and their order.
SETF  /A  /B  /C  /D                 ;Default (TRACE-ON not used), as on pins.
SETF  D
SETF  C  /D
SETF  D
SETF  B  /C  /D
SETF  D
SETF  C  /D
SETF  D
SETF  A  /B  /C  /D
SETF  D
SETF  C  /D
SETF  D
SETF  B  /C  /D
SETF  D
SETF  C  /D
SETF  D
TRACE_OFF
```

A.7 An Introduction to PALASM

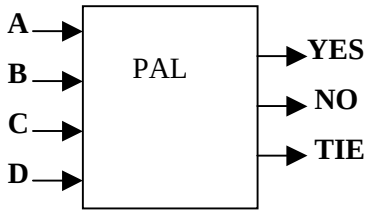
The following describes a relatively simple project to illustrate a few basic principles in writing a PALASM program.

A.7.1 Design Project

Design a PAL circuit that will display the results of a vote among four persons [ABCD]. A "sum-of-products" solution is derived from the truth table and implemented with a PAL device.

INPUTS: [**ABCD**] yes = 1
no = 0

OUTPUTS: **YES** = 1 ("yes" majority)
NO = 1 ("no" majority)
TIE = 1 (tie vote)



A	B	C	D	YES	NO	TIE
0	0	0	0	0	1	0
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	0	0	1
0	1	0	0	0	1	0
0	1	0	1	0	0	1
0	1	1	0	0	0	1
0	1	1	1	1	0	0
1	0	0	0	0	1	0
1	0	0	1	0	0	1
1	0	1	0	0	0	1
1	0	1	1	1	0	0
1	1	0	0	0	0	1
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	1	0	0

A.7.2 PAL Device

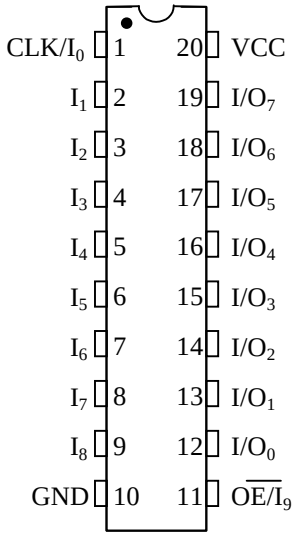
1. The PAL device used is the LATTICE GAL16V8A, or the equivalent AMD device, PALCE16V8.
2. In the **simple** mode (combinational) used in the design file "VOTE.pds" no memory or register is used.
3. In the **simple** mode, pins 15 & 16 do not have a feedback capability, and must always be configured as dedicated outputs.
4. In the **registered** mode (RVOTE.pds) pin 1 & pin 11 are permanently configured as clock and output enable.

Note: Pin 1 is marked for orientation

PIN DESIGNATORS

CLK	- Clock	I/O	- Input/Output
GND	- Ground	$\overline{OE}$	- Output Enable
I	- Input	VCC	- Supply Voltage

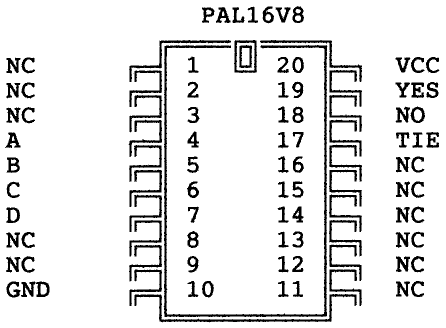
CONNECTION DIAGRAM
Top View



A.7.3 PALASM Design Files

1. Use an ASCII text editor to prepare a design ".pds" file.
2. The 2 example files are VOTE.pds (simple mode) and RVOTE.pds (registered mode).

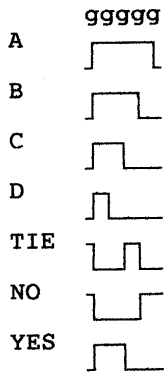
```
TITLE       PAL VOTING MACHINE
PATTERN     VOTE.PDS
REVISION    A
AUTHOR      Your name
COMPANY     RYERSON
DATE        MM/DD/01

CHIP        VOTE        PAL16V8     ; Use "versatile" type PAL.
NC NC NC A B C D NC NC GND          ; Simple mode
NC NC NC NC NC NC TIE NO YES VCC

EQUATIONS               ; The "=" operator indicates a combinational output.
YES = /A*B*C*D + A*/B*C*D + A*B*/C*D + A*B*C*/D + A*B*C*D
NO = /A*/B*/C*/D + /A*/B*/C*D + /A*/B*C*/D +/A*B*/C*/D + A*/B*/C*/D
TIE = /A*/B*C*D + /A*B*/C*D + /A*B*C*/D + A*/B*/C*D + A*/B*C*/D + A*B*/C*/D

SIMULATION              ; A new "SETF" statement uses the pre-
                        ; vious statement as an initial condition.
TRACE_ON    A B C D YES NO TIE
SETF        A B C D     ; 1 1 1 1     Sets A=B=C=D=1
SETF        /D          ; 1 1 1 0     Sets /D=1, i.e. D=0
SETF        /C          ; 1 1 0 0
SETF        /B          ; 1 0 0 0
SETF        /A          ; 0 0 0 0
TRACE_OFF
```



VOTE.pin

A.7.4 Simulation Results

1. The use of TRACE _ON is optional, and controls the selection, sequence, and polarity of signals displayed in simulation. The results for **every** pin as defined in the pin list are displayed using the .hst file.
2. Simulation results can be displayed as simulation data or as a waveform display.
3. In the waveform & data display, the letters **g** and **c** indicate the occurrence of SETF and CLOCKF commands, respectively.
4. The pin layout diagram can be saved to disk as a .pin file. All other files generated by PALASM are automatically transferred to the floppy disk containing the .pds file.



WAWEFORM DISPLAY

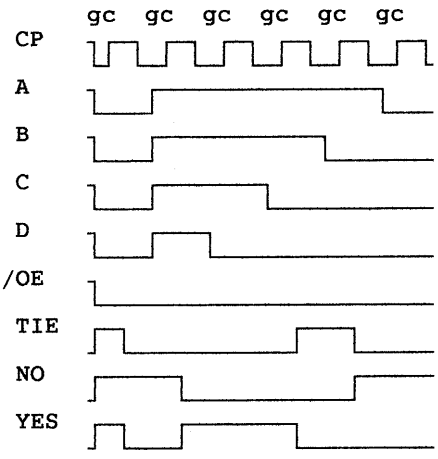
```
      ggggg
A     HHHHL
B     HHHLL
C     HHLLL
D     HLLLL
YES   HHLLL
NO    LLLHH
TIE   LLHLL
```

SIMULATION DATA

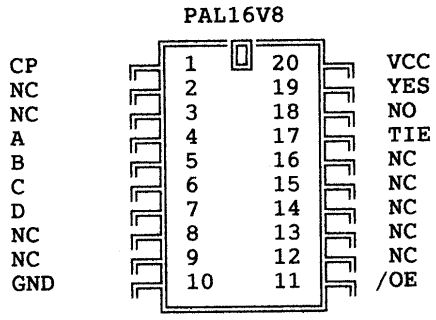
A.7.5 Registered Mode

1. The design file RVOTE.pds causes the software to select the "registered" mode of operation. The change to output signals due to input signal variations are not available until a clock pulse is applied.

```
TITLE       PAL VOTING MACHINE
PATTERN     RVOTE.PDS
REVISION    B
AUTHOR      Your name
COMPANY     RYERSON
DATE        MM/DD/01

CHIP        RVOTE       PAL16V8
CP  NC  NC  A  B  C  D  NC  NC  GND ; Registered mode
/OE NC  NC  NC NC NC TIE NO  YES VCC

EQUATIONS                   ; The ": =" operator indicates a registered output.
YES : = /A*B*C*D + A*/B*C*D + A*B*/C*D + A*B*C*/D + A*B*C*D
NO : = /A*/B*/C*/D + /A*/B*/C*D + /A*/B*C*/D +/A*B*/C*/D + A*/B*/C*/D
TIE : = /A*/B*C*D + /A*B*/C*D + /A*B*C*/D + A*/B*/C*D
        + A*/B*C*/D + A*B*/C*/D

SIMULATION
TRACE_ON    CP A B C D YES NO TIE
SETF        /CP OE          ; Initialize clock & enable outputs (/OE=0)
CLOCKF      CP              ; Applies 1 clock pulse to pin
CP
SETF        A B C D         ; 1 1 1 1
CLOCKF      CP
SETF              /D        ; 1 1 1 0
CLOCKF      CP
SETF            /C          ; 1 1 0 0
CLOCKF      CP
SETF          /B            ; 1 0 0 0
CLOCKF      CP
SETF        /A              ; 0 0 0 0
CLOCKF      CP
TRACE_OFF
```



WAVEFORM DISPLAY



RVOTE.pin

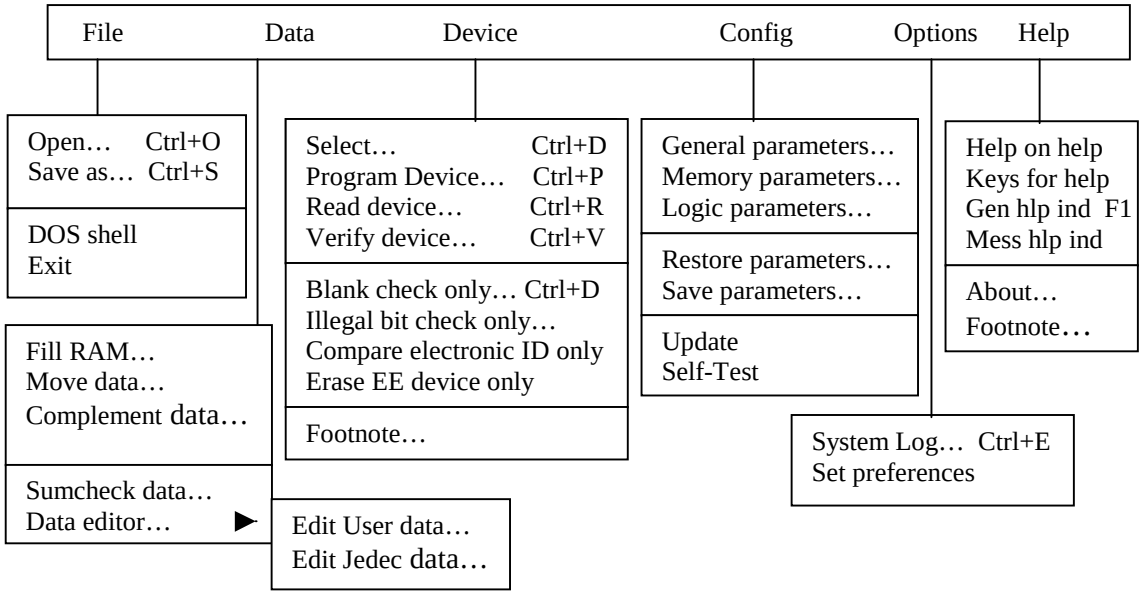
```
     gc  gc  gc  gc  gc  gc
CP   LHHLLHHLLHHLLHHLLHHLLHHL
A    LLLLHHHHHHHHHHHHHHHHLLLL
B    LLLLHHHHHHHHHHHHLLLLLLLL
C    LLLLHHHHHHHHLLLLLLLLLLLL
D    LLLLHHHHLLLLLLLLLLLLLLLL
YES  HHLLLLHHHHHHHHLLLLLLLLLL
NO   HHHHHHLLLLLLLLLLLLHHHHHH
TIE  HHLLLLLLLLLLLLHHHHLLLLLL
```

SIMULATION DATA

A.8 Programming PLDs & EPROMs (ChipLab)

1. All of the CHIPLAB functions are accessed from the main menu bar and the pull-down menus.
2. It is recommended that the "mouse" has been used (for ease of operation) to access and select the items from the pull-down menus.
3. Check the green light on the Chiplab programmer "pod" to confirm that power is being applied.

ChipLab's Main Screen
(showing pull-down menus)



A.8.1 Program PLDS

Example devices: 1. AMD - PALCE16VH-25/4
2. Lattice - GAL16V8A

1. PLACE FLOPPY DISK with prepared data file (PALASM - e.g. hex-7.jed) in disk drive (A or B). At the "DOS" prompt, type "Chiplab". Wait until the system initialization is complete.

2. SELECT DEVICE: Access the *"Device"* pull-down menu. Sequence (with the mouse):
   a) *Select*
   b) *Specify* one of the following: AMD, CE16V8H-25/4, LDip 20 logic; or Lattice, 16V8A, LDip 20 logic.
   c) *Select*

3. READ DATA FILE: Access the *"File"* pull-down menu. Sequence:
   a) *Open*
   b) *Source file* - e.g. "a:labl.jed"
   c) *File format* - JEDEC (Full), code 91
   d) *Read* ... At this point, "File a:labl.jed loaded" should appear at the bottom of the screen.

4. INSERT PLD: Insert the PAL (or GAL) device into the programmer "pod" socket. Ensure that the PLD device is bottom justified and the socket lever is locked. At this point, if required, *"Blank check only"* can be accessed from the *"Device"* pull-down menu.

5. PROGRAM DEVICE: Access the *"Device"* pull-down menu. Sequence:
   a) *Program device*
   b) *Program ...* The message "program operation successful" should appear. Press *ESC* to return to the main menu. The screen is now ready for the next user.

A.8.2 Read & Display PLD Contents

1. READ DEVICE: Access the *"Device"* pull-down menu. Select *Read device*. The programmed device data is read into user memory.

2. DISPLAY PLD CONTENTS: Access the *"Data"* pull-down menu. Sequence:
   a) *Data editor*
   b) *Edit Jedec data...* The "mapping" of the programmed PLD will appear on the screen. This display (i.e. fuse map) can be compared to the ".jed" or the ".xpt" file as generated by the PALASM program

A.8.3 Program EPROMS

NOTE: Similar to the PLD procedure described above, but with a few variations in file types and formats as shown below.

1. FLOPPY DISK: Prepare an *"Intel hex record"*. Use "mkintel <*.asc>*.hex" program which is available at all PC stations in room T230.

2. SELECT DEVICE: e.g. Texas Instruments (or equivalent) - DIP EPROM 27128

3. READ DATA FILE: a) *Source file* - e.g. "a:lab5.hex"
   b) *File format* - Intel Intellec 8/MDS, code 83.

A.8.4 Read & Display EPROM Contents

1. READ DEVICE: Access the *"Device"* pull-down menu. Select *Read device*.

2. DISPLAY EPROM CONTENTS:
   Access the *"Data"* pull-down menu. Sequence:
   a) *Data editor*
   b) *Edit User data ...* Wait until the operation is completed. This display (Hex and ASCII format) can be compared to the prepared ".hex" or ".asc" data files.

A.9 ChipWriter - Quick start tutorials

A.9.1 Tutorial 1: Programming an EPROM or Microcontroller from a master device

1. Type CWEPROM <ENTER>
2. Select the Device you are using as a master, using DEVICE/TYPE. Select the device type you are wishing to use and press <ENTER>. You will then be given a list of manufacturers who support the selected device type. Select the appropriate manufacturer and press <ENTER>. You will then be prompted for the device name. Choose the correct device name from the menu you are given and press <ENTER> to accept. The details of the device you have chosen will now be displayed at the bottom of the screen.
3. Ensure that the data memory buffer size is large enough to hold the data to be read from the master device. You can change the buffer size using DATA/BUFFER-SIZE.
4. Insert the master device in the programmer.
5. Select PROCESS/READ to read the contents of the chip into the data memory buffer. Press <F10> to accept the default addresses.
6. Select the device you are copying to, if it is different (see 2.).
7. Now place the blank copy device in the programmer socket.
8. Select PROCESS/PROGRAM. Press <F10> to accept the default addresses.
9. The message: *Device programmed and verified* will be displayed.

A.9.2 Tutorial 2: Programming an EPROM or Microcontroller from a file on disk

1. Type CWEPROM <ENTER>
2. Select the Device you are using as a master, using DEVICE/TYPE. Select the device type you are wishing to use and press <ENTER>. You will then be given a list of manufacturers who support the selected device type. Select the appropriate manufacturer and press <ENTER>. You will then be prompted for the device name. Choose the correct device name from the menu you are given and press <ENTER> to accept. The details of the device you have chosen will now be displayed at the bottom of the screen.
3. Ensure that the data memory buffer size is large enough to hold the data to be read from the master device. You can change the buffer size using DATA/BUFFER-SIZE.
4. Select FILE/OPEN
5. Specify the path and filename where your data is located.
6. Select the file format used (HEX auto-recognition will detect any of the available HEX formats)
7. Press <F10 > to accept the default settings and load the data. A checksum will be displayed on the screen.
8. Select the device you are copying to if it is different (see 2.)
9. Place the blank device in the programmer socket.
10. Select PROCESS/PROGRAM. Press <F10> to accept the default addresses.
11. The message: *Device programmed and verified* will be displayed.

A.9.3 Tutorial 3: Programming a logic device from a file on disk

1. Type CWPAL <ENTER>
2. Select the device you are going to program using DEVICE/ MANUFACTURER. Select manufacturer and press <ENTER>. Choose the device name from the list on the screen by highlighting the correct device and pressing <ENTER>.
3. Select FILE/OPEN.
4. Specify the path and filename where your JEDEC file is located.
5. Press <F10> to accept the defaults. A checksum will be displayed once the file has loaded.

6. Place the blank device in the programmer socket.
7. Select PROCESS/PROGRAM. The device will now be erased (where relevant), programmed, verified and any test vectors present in the JEDEC file will be applied.
12. The message: *Device programmed and verified* will be displayed.

EPROM Tutorial

1 Introduction

This is a demonstration lab to show you how to prepare an MS-DOS floppy disk with a file that can be used to program an EPROM (Erasable Programmable Read Only Memory).

First, let us examine how an EPROM can be used to implement arbitrary combinational logic, how it is physically programmed, and how dedicated hardware (an EPROM burner) and software can be used to make the programming less arduous.

Consider a 8K byte EPROM (i.e. 8192 words of 8 bits each). The chip will have 13 address inputs (A_{12} to A_0) and 8 data outputs (D_7 to D_0). Suppose we wish to implement the logic functions:

$Z1 = ABCDEFGHIJ\overline{K}\,\overline{L}M + \overline{A}CG$

$Z2 = F_2(A, B, C, D, E, F, G, H, J, K, L, M)$

All of the logic functions can be represented in truth tables, converted to Karnaugh maps and minimized to obtain the optimal sum of products expressions. With 13 input variables, of course, this would take a lot of time!

Things are *conceptually* much simpler when EPROMS are used, however. We simply use input A as address line A_{12}, B as A_{11}, C as A_9, etc. Similarly, we use output D_0 as output Z1, D_1 as Z2, etc.

If the inputs ABCDEFGHIJKLM are 1111111111001, out first equation indicates that Z1 must be a 1. If we also assume that Z2 must be zero with these same inputs, we must program the contents of memory location 1111111111001 to contain the value xxxxxx01. We can program the EPROM so that this occurs by setting the address lines to 1111111111001, the data lines to xxxxxx01 and applying a programming pulse to the chip.

In principle, this same procedure can be applied for every address on the chip. Alas, this amounts to a large amount of tedious work. For every memory location 13 switches have to be set to specify the address, another 8 to specify the contents and the programming pulse applied.

The tedium can be reduced by using an EPROM burner. It can generate sequential addresses automatically and take care of the programming pulse. But then the question becomes: "How is the ERPOM burner to be instructed on the memory contents?".

Most EPROM burners receive their instructions over a serial data link connected to a computer. The information transferred over the serial link is usually in *Intel Hex Record* format. This information in turn is retrieved from a computer *file* on a disk. Other software, such as a word processor, editor, or something even more sophisticated is used to prepare this file.

1.1 Intel Hex Records

An Intel Hex Record contains a starting address of where to begin burning data and the actual data to be programmed at that and subsequent addresses. The amount of data specified in a single hex record can be as little as zero bytes and as much as 255 bytes. A file can contain several Hex records so that the contents of more than 255 locations can be specified. The starting addresses of each record do not have to follow any particular order. By convention, the last record indicates zero bytes at address 0.

The format of an Intel Hex Record is as follows:

```
:0520000001020304 05CC
```

Figure 3: Hex Record format (1 line)

The first two ascii characters give the 2 hex digit count of the number of memory locations defined by this record. In this case, the contents of 5 locations are defined. The next 4 characters give the starting address in hex of the locations defined. Here, the record indicates that 5 locations starting at address HEX2000 are defined. For our purposes, we assume the next 2 characters are always '0'. The following characters are the ascii-encoded hex values of the sequential memory locations being defined. Since the hex record above defines the contents of 5 memory locations, the next 5 character pairs (i.e. 10 characters in all) define the contents of locations HEX2000 through HEX2004 as 01, 02, 03, 04 and 05 respectively. For example, the contents of location HEX2001 is defined as HEX02. The final two characters represent the hex value of a *checksum*. The checksum has a value so that if it is added to all the other bytes in the hex record, the lower 8 bits of the result are exactly HEX00. In this case, we can verify that HEX05 + HEX20 + HEX01 + HEX02 + HEX03 + HEX04 + HEX05 + HEXCC = HEX100 with lower eight bits = HEX00.

A file containing hex records that set the locations starting at 100 (hex) to 00, 01, 02, 03, ... 17 is shown below:

```
:10010000010203040506070809 0A0B0C0D0E0F77
:08011000101112131415161 74B
:0000000000
```

As an exercise, you should verify that the checksums are correct for each of 3 hex records.

1.2 Preparing Hex Records

The Intel Hex Record is a standard format used to prepare data for programming an EPROM. Given the data, one can prepare the hex record by hand, by the proper use of an assembler (e.g. AS6809, & ASLINK), or by some other utility.

For those who are unwilling to do it by hand, or are not ready to tackle assembly language for the moment, here is a simple way of preparing the Intel hex record for an EPROM.

First, one must use a text editor (e.g. JOVE, or EDIT) to create the data file in ascii. For example, the following is the contents of a file called 'job1.asc':

```
0000 01 02 03 04 05 06 07 08      ; comments can be placed here.
0008 10 11 12 13 14 15 16 17      ;
0100 AB CD EF
0200 0A 1F
```

- Each 'line' must start with a 4-digit (hex) address, followed by any number of 2-digit (hex) data bytes.
- Leading zeros must be entered for both address and data.
- Spaces between data are for appearance only, and are ignored by the program.
- Comments can be accommodated for each line following the data. Any text following the ';' character at that line is ignored by the program.
- There **must not** be any blank lines.

Now, type the following command line (in any one of the PC stations in T230):

mkintel < job1.asc > job1.hex and hit the Return key.

The required Intel hex record is now in *job1.hex*.

EPROM Programmer Tutorial

1 OBJECTIVE

To program an EPROM (eg. 2764) using the American Reliance Universal Programmer (Modal 9860) located in T230.

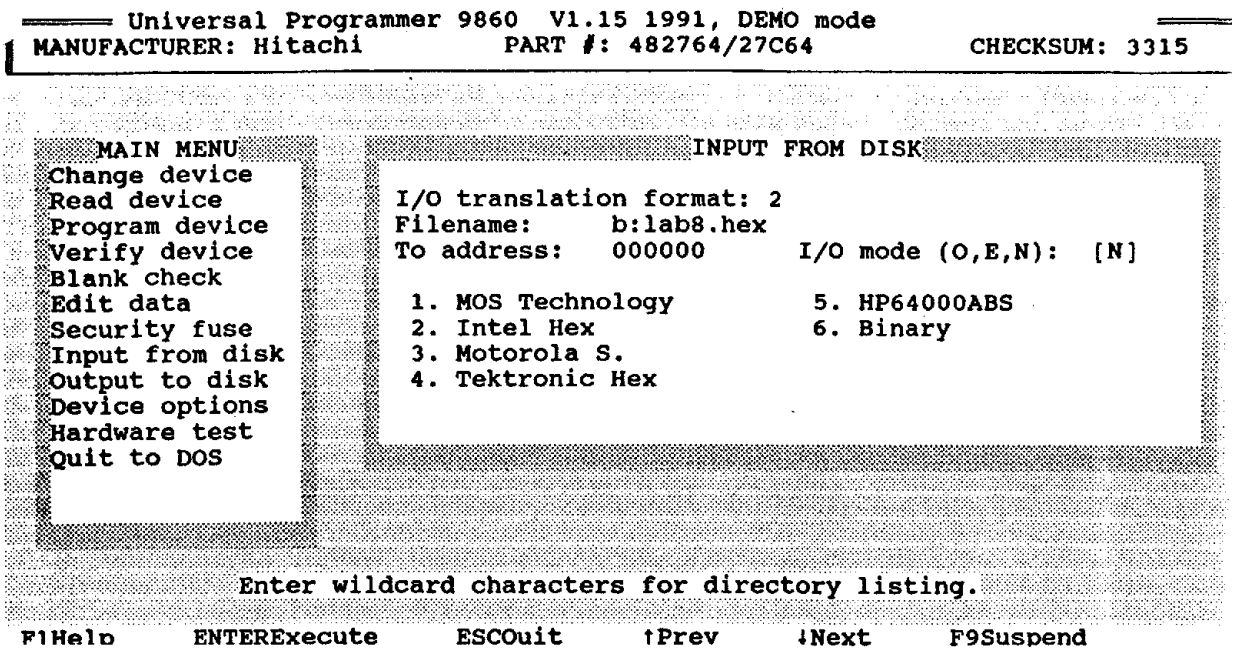
2 Intel HEX RECORD

The required Intel Hex record is prepared (5.25" diskette) according to "Making Intel Hex records".
Example: **mkintel < job1.asc > job1.hex** and hit Return key.

3 PROCEDURE

1. Place diskette containing required Intel hex record in drive A or B as required.
2. At the system prompt C>, type "9860". Do NOT insert EPROM at this time.
3. Main menu appears as shown on page 2.
4. Change Device Command: Select with cursor or type "c".
   (a) Select the manufacturer by entering the corresponding number and press ENTER (eg. 10 for Hitachi).
   (b) Select device by entering corresponding number and press ENTER (eg. 3 for 482764).

   Note: There may be more than one page on the menu.
5. EPROM Insertion: Clamping lever must be UP before inserting the chip. Fit EPROM chip into holder (notch at TOP). Press lever forward and down.
6. Blank Check Command: Select with cursor or type "b".
   (a) This command verifies that the socketed device has not been programmed.
   (b) Warning: Do NOT remove device until operation is complete for this may damage device.
7. Input From Disk: Select (type "i").
   (a) This command loads data from selected file (eg. b:lab8.hex) into a memory buffer.
   (b) Enter parameters -lab8.hex, address 0, mode N, Intel Hex (2)
   (c) Press ENTER to begin the read operation. If operation is successful, "Operation Complete" displayed.
8. Program Device: Select (type "p").
   (a) This command programs the socketed device with the contents of the memory buffer. If operation is successful, "Device programmed OK" displayed.
   (b) Warning: Do NOT remove device until operation is complete.
9. Verify Device: Select (type "v").
   (a) This command verifies that the contents of the socketed device are the same as the contents of the memory buffer.
   (b) If successful, "Device verified OK" will be displayed.
   (c) Warning: Do NOT remove device until operation is complete.
10. Precaution: Transfer programmed EPROM to a piece of conducting foam with the grounded chip extractor. Avoid touching the connecting pins. !


Universal Programmer 9860 V1.15 1991, DEMO mode
MANUFACTURER: Hitachi
PART #: 482764/27C64
CHECKSUM: 3315
MAIN MENU
Change device
Read device
Program device
Verify device
Blank check
Edit data
Security fuse
Input from disk
Output to disk
Device options
Hardware test
Quit to DOS
INPUT FROM DISK
I/O translation format: 2
Filename: b:lab8.hex
To address: 000000
I/O mode (O,E,N): [N]
1. MOS Technology
2. Intel Hex
3. Motorola S.
4. Tektronic Hex
5. HP64000ABS
6. Binary
Enter wildcard characters for directory listing.
F1Help
ENTERExecute
ESCQuit
↑Prev
↓Next
F9Suspend

Part B – MAX+PLUS II "Tutorial Introduction"

B.1 Objectives:

- The main objective of this lab is to know how to use MAX+PLUS II software. MAX+PLUS II is used for Synthesis of digital logic design using Complex Programmable Logic Devices (CPLDs) and Field Programmable Gate Arrays (FPGAs).
  Note: See the Appendix B (course text), chapters B.1, B.2, and B.4

B.2 Installing Your Own Copy of MAX+PLUS II – Done later

- System Requirements
  - IBM PC using Windows 95, 98 or NT (3.51 or 4.0)
  - CD drive for software installation
  - Available memory > 48 MB
  - Available Hard disk space > 80 MB
- Installation
  - Run the command <CD-ROM drive>:\pc\setup.exe <Enter>. The setup program will guide you through the installation process. After installation you have to obtain the License File for "MAX+plus II 9.23 Baseline" from Altera (www.altera.com/maxplus2-student). When obtaining the License File you will be asked to provide the serial number of your hard disk. To obtain this number start MAX+plus II and click on the *Options* menu, *License Setup* and *System Info*. The hard disk serial number is then displayed.
  - The first time you start MAX+plus II it will ask you whether you want instructions to obtain a free license. Answer "No", and go to the web site to obtain your license file.
  - After you have provided to Altera the required information, the License File will be e-mailed to you along with instructions how to enable the software.

B.3 Procedure – Preparing Design Entry

Design entry can be performed by three different methods: using schematic capture, truth table, and writing VHDL code. In this lab, we will use schematic capture and the truth table method to enter, compile, and simulate simple designs.

B.4 Pre-lab preparation

1. To do the pre-lab you will need to logon to a department workstation in lab T230
2. To save your files, create the subdirectory *max2work*, [mkdir max2work]. Under the directory *max2work* create another directory called *tutorial1*. This will be working directory for the design created in this tutorial.
3. Start up Max+plus II by typing max2win at the prompt. This will open Max+plus II window, which gives you access to an integrated suite of eleven application tools.
   Note: When Max+plus II starts for the first time, a file called maxplus2.ini will be created in your home directory. This is an initialization file required by Max+plus II and it takes a little time to create. Do not delete it! A directory called windows will also be created in your home directory. Do not delete it!

B.5 Creating a Schematic Design

Schematic designs are created with the **Graphic Schematic Editor** tool of Mux+plus II.

- To start Graphic Design Editor, select Max+plus II then Graphic Editor
- Refer to Appendix B.2 of the reference text and enter the schematic represented in Figure B.4 of Appendix B.2
- Save and check the design by selecting File then Project and then Save & Check. This will open the compiler. If there are no errors or warnings, proceed to the simulation.

B.6 Compiling and Simulating a Design

A design is simulated by compiling, creating input waveforms (test vectors), simulating and viewing the results.

- To open compiler select Max+plus II then Compiler
- Select Processing then Functional SNF Extractor and click Start in the compiler window. The output of the compilation is the *simulator netlist file (.snf).* The simulator also requires a *simulator channel file (.scf),* which contains information about logic levels for input, output and hidden signals in the circuit. Each input and output is often referred to as a *channel*
- To create a simulator channel file, select Max+plus II then Waveform Editor. Refer to Appendix B.2.4 of reference text and select the channels for simulation
- Simulate the circuit by selecting Max+plus II then Simulator. Click start. Obtain printouts of the simulation and hand them in to your instructor.

B.7 Design Entry Using Truth Tables

In this section we will design a logic circuit using truth table. We will implement the truth table shown in Figure B.25 of Appendix B.4 (see reference text). It will be entered into the CAD system by drawing a timing diagram, which is equivalent to truth table.

- Open the Waveform Editor
- Specify the type of the file to be created by selecting File then Save As. In the box labeled File Name, type the name *timing1.wdf.*
- Specify the input and output signals (refer to Appendix B4.2 of the text book)
- Specify a waveform for the output of the circuit that corresponds to its truth table
- Select File then Save. Obtain printouts of the simulation and hand them in to your instructor

B.8 Mixed Design-Entry Methods

Here we will create a schematic that includes the circuit designed using the truth table in the previous section. Use the same directory as for the previous projects (see Appendix B5 of the reference text)

- Open the Graphic Editor
- Specify the type of the file to be created by selecting File then Save As. In the box labeled File Name, type the name *mixed1.gdf.*
- Import the *timing1* circuit into the Graphic Editor.
- Follow the procedure of Appendix B.5 of the reference text and draw the schematic of figure B.30. Save the schematic.
- Obtain printouts of the simulation and hand them in to your instructor