More Programming with CUPL

Compiling PLD File

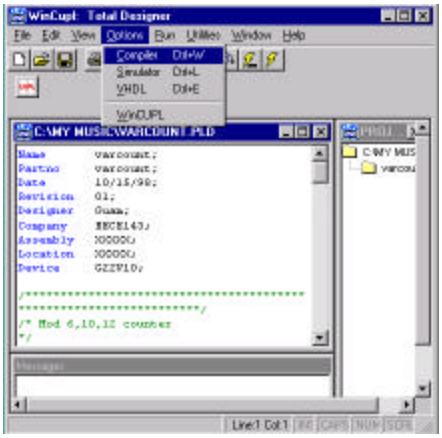
- Use WinCUPL or use another text editor to create your file
- Save your file with the extension .PLD
- Compile using WinCUPL make sure you have no errors.
- Save your .JED file on diskette

Using WinCUPL

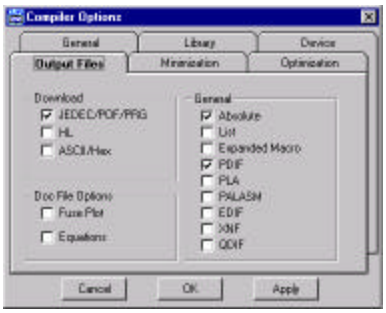
Located in Engineering Applications\ EECE\ Wincupl folder

After creating and editing your PLD file:

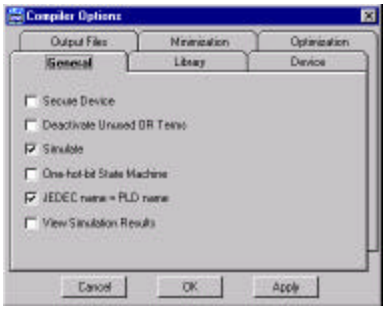
Open Options|Compiler menu item or type Ctrl+W



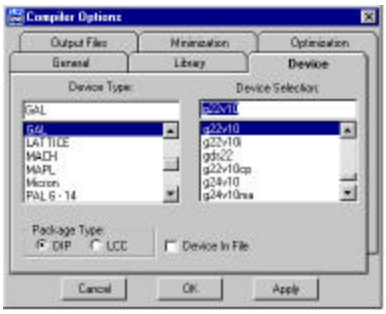
Programmable Logic Devices II EECE143 Lecture 5



Select Download JEDEC/PDF/PRG



Select JEDEC name = PLD name



Select Device:

Either g16v8 for GAL16V8

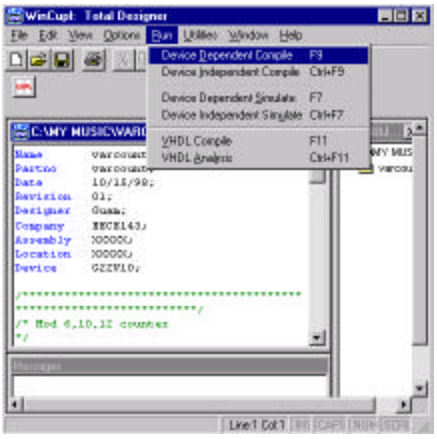
Or g22v10 for GAL22V10

DIP Package

Programmable Logic Devices II EECE143 Lecture 5

To compile your program, Select Run|Device Dependent Compile menu item

If you get errors, check the Messages window for error locations and make corrections to your code.



Programming PLD File

- Use Allpro programmer in EN365 or the ChipMaster 88 Programmers in the digital lab or the EECE Open Lab
- Be sure to follow instructions for programming for each programmer.

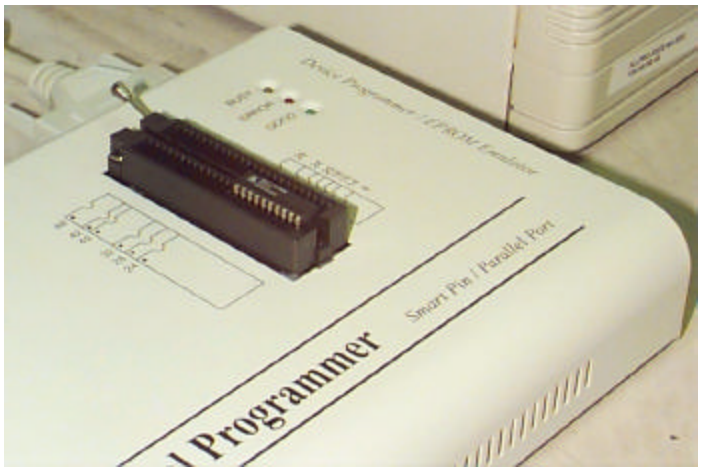
ChipMaster 6000 Intelligent Universal Programmer

Using the CM6000 Universal Programmer

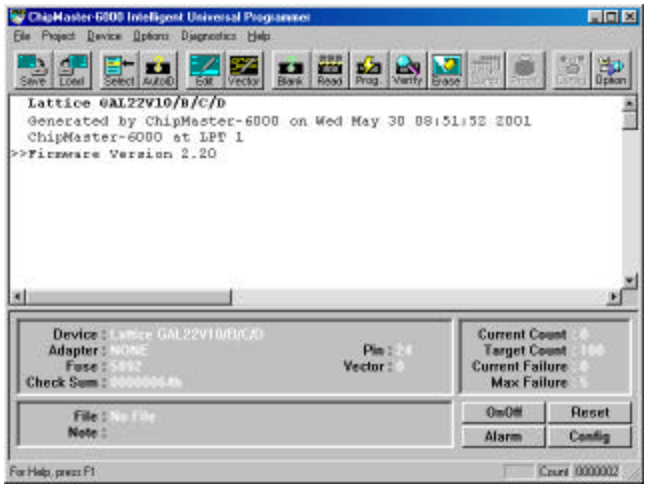
1. Load the device
2. Select a device
3. Perform Blank Check
4. Erase device
5. Read jedec file
6. Program device
7. Verify

Insert Device into the socket

1. Make sure the busy light is off.
2. Lift the lever of the ZIF socket up.
3. Insert the device. Align the device with the bottom of the socket with pin 1 facing the top of the socket.
4. Press the lever of the ZIF socket down.

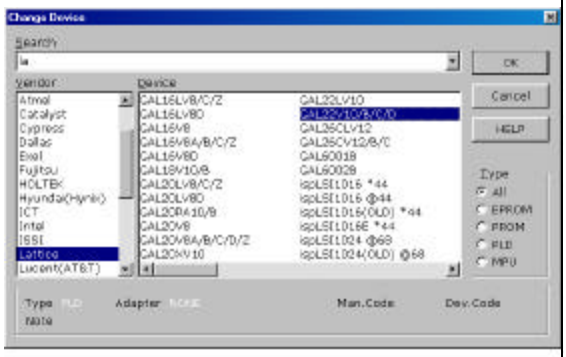


Startup



Select Device

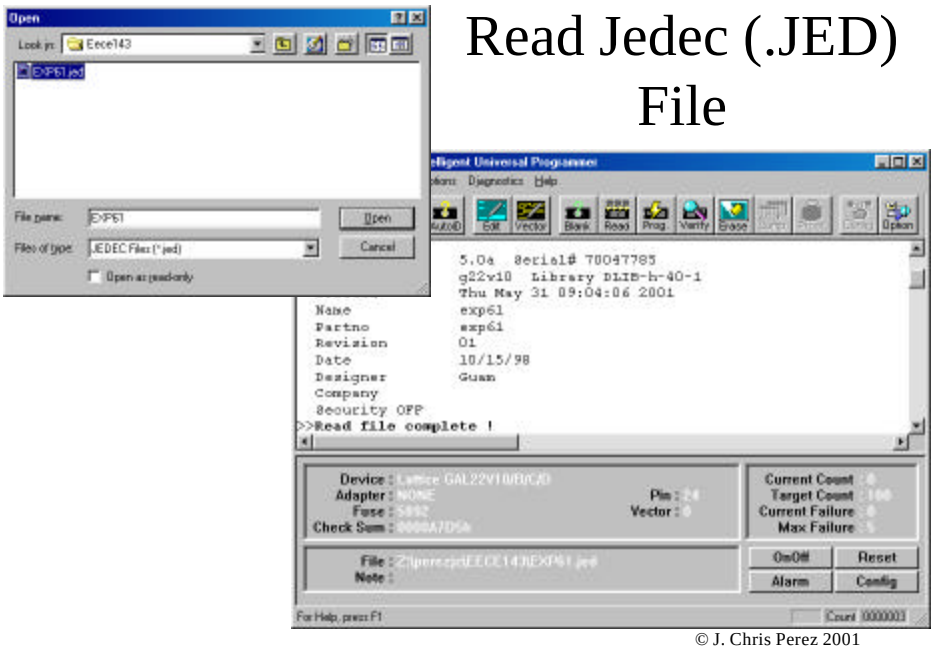
1. Type ALT-C or Select the Device|Change Device menu item.
2. Select the device required:
   Use the mouse: Click on 'All', EPROM', 'PROM', 'PLD', or 'MPU'
   Use the keyboard: Press TAB until the cursor is flashing in the 'Type' box. Use the up and down arrows to go to the appropriate type. Press the space bar to select the type.
3. Enter the part number in the 'Search' box.
   Use mouse: Click on 'Search'
   Use keyboard: Press TAB until the cursor flashes in the 'Search' box. Type in the part number.
4. Use the mouse to select a vendor.
5. Select a device and click 'OK. Use the TAB key to skip between various screens, use the arrow keys to move around each screen. Press <ENTER> to select the vendor/device.



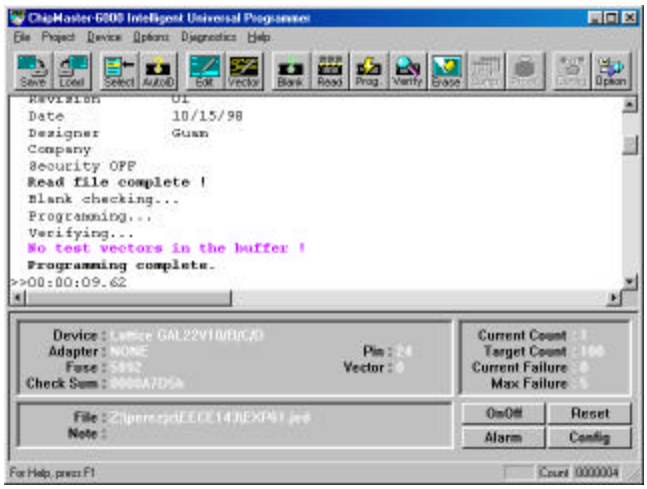
Load File into Memory

1. Press ALT-L for Load File or select the File|Load File menu item.
2. Press <TAB> to move the cursor to the list of files in the default directory
3. Select a file with the up and down arrow keys.
4. Press <ENTER>.
5. Select a file type with the up and down arrow keys. You want to select a jedec (.JED) file
6. Press <ENTER> to load the file.

Read Jedec (.JED) File



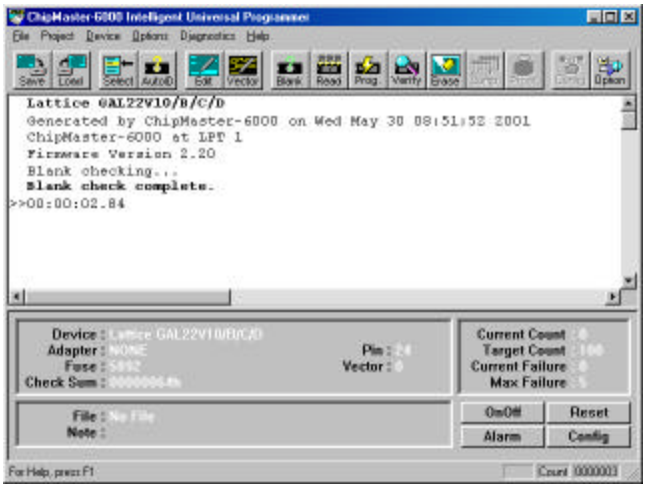
Program Device



Blank check the device

Insert the device in the socket.

1. Select the device algorithm.
2. Press the ALT -B.



Erase the device

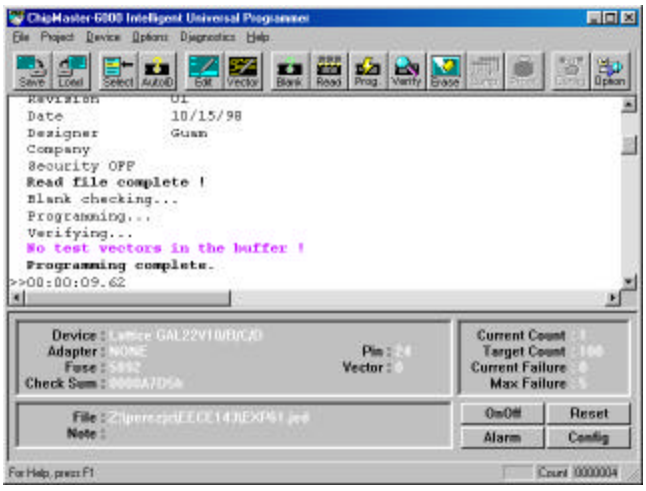
This option is only available for devices that support chip erase.

1. Insert the device in the socket.
2. Select the device algorithm.
3. Press CTRL-F1
4. Click OK at the Are you sure you want to erase the device?

Program the device

Before you program the device you have to load the file into memory.

1. Insert device into the socket.
2. Select the device algorithm.
3. Press ALT-P. or Click on the Prog. Button.



Verify the device

1. If the device is not inserted, insert it into the socket.
2. Select the device algorithm.

   The CM6000 verifies the device with the contents of the memory buffer. If the correct file is not loaded, load it now.
3. Press ALT-V.

After programming, the CM6000 automatically verifies.

You are done programming your chip.

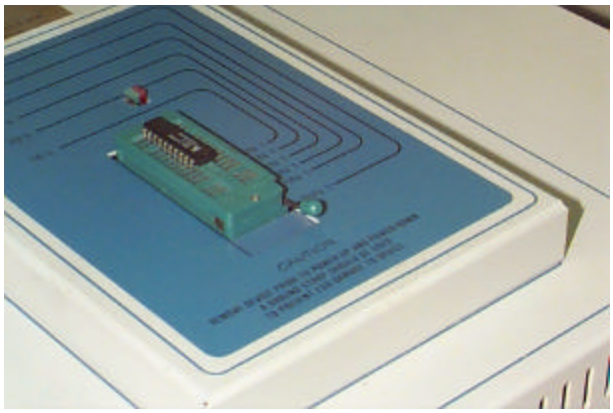
AllPro 88 Programmer

Using the AllPro 88 Programmer

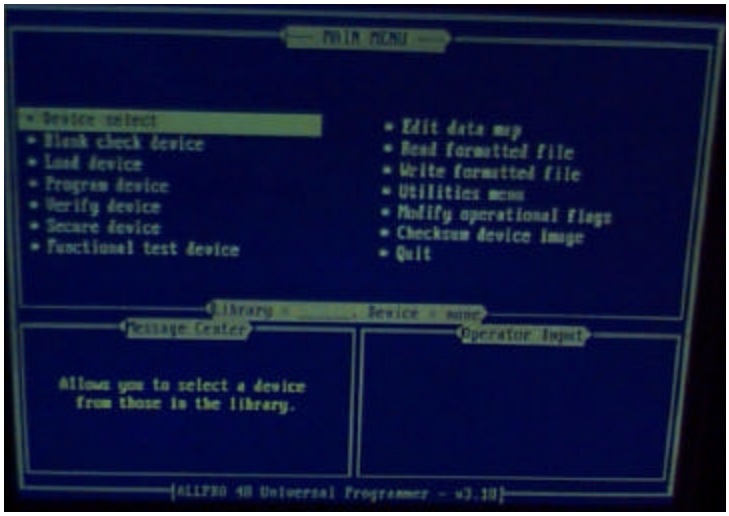
- Load Device into programmer
- Choose Library and Device from library
- Blank Check Device (optional)
- Read Jed file
- Program Device

Load Device in Programmer

Place chip so Pin 1 faces front of programmer.

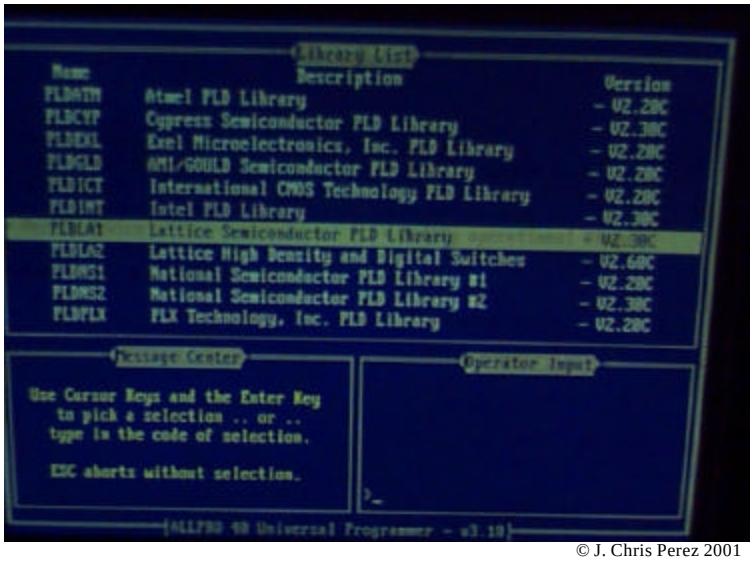


Choose Library and Device from library

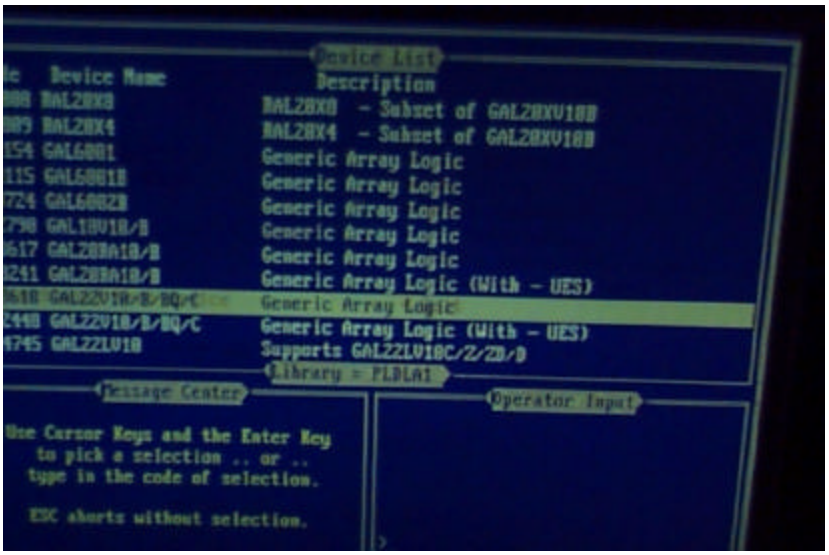


Choose Library and Device from library

Choose Library …



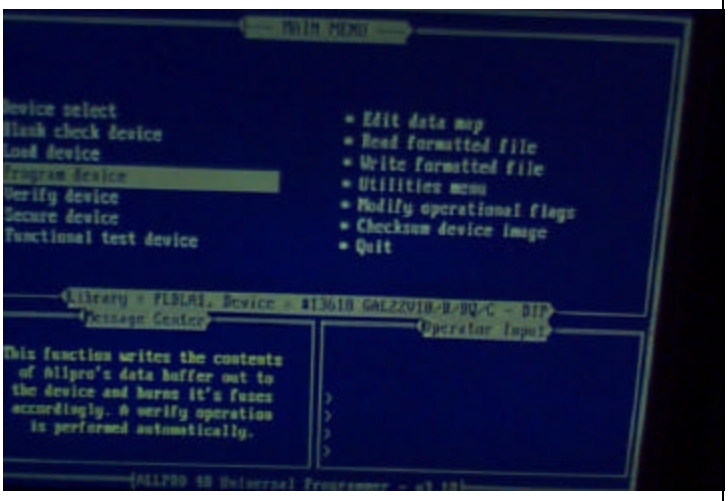
Choose Device …



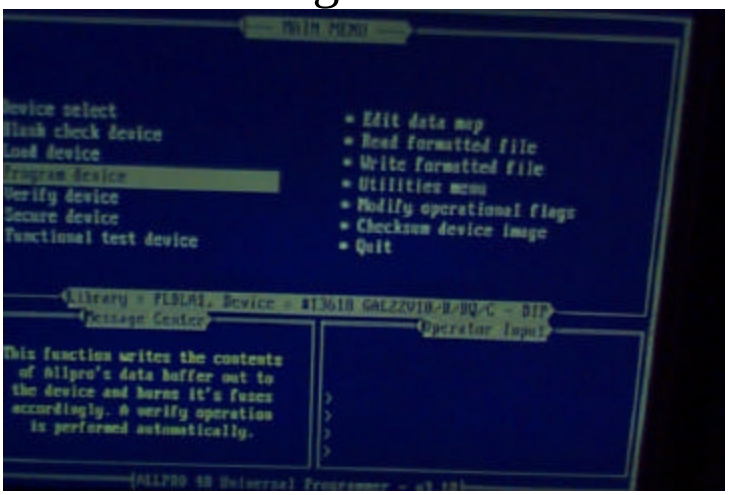
Blank Check Device (optional)

Read formatted File …

Select your .jed file from your floppy diskette



Program Device …



… and you are done.

HEX7SEG.PLD

```
Name      hex7seg;
Partno    XXXXX;
Date      03/15/01;
Revision  01;
Designer  J. Chris Perez;
Company   EECE143;
Assembly  XXXXX;
Location  XXXXX;
Device    GAL16V8;

/*************************************************************/
/* This program takes a 4-bit HEX number input and outputs   */
/* the necessary signals to drive a 7-segment display        */
/*                                                           */
/*************************************************************/
/*  Allowable Target Device Types:                           */
/*************************************************************/
```

Input	Output
0000	0
0001	1
0010	2
0011	3
…	
1010	A
1011	b
1100	c
1101	d
1110	e
1111	F

```
/** Inputs **/

Pin 1 = W;    /* Msb of HEX input   */
Pin 2 = X;    /*                    */
Pin 3 = Y;    /*                    */
Pin 4 = Z;    /* Lsb of HEX input   */

/** Outputs **/

Pin 19 = a;   /* Output Segments for 7-segment Display */
Pin 18 = b;   /*                    */
Pin 17 = c;   /*                    */
Pin 16 = d;   /*                    */
Pin 15 = e;   /*                    */
Pin 14 = f;   /*                    */
Pin 13 = g;   /*                    */
```



```
/** Logic Equations **/
FIELD INPUT = [W,X,Y,Z];  /* Defines input array   */

FIELD OUTPUT = [a,b,c,d,e,f,g]; /* Defines output array */

TABLE INPUT => OUTPUT {
'b'0000 => 'b'0000001;
'b'0001 => 'b'1001111;
'b'0010 => 'b'0010010;
'b'0011 => 'b'0000110;
'b'0100 => 'b'1001100;
'b'0101 => 'b'0100100;
'b'0110 => 'b'0100000;
'b'0111 => 'b'0001111;
'b'1000 => 'b'0000000;
'b'1001 => 'b'0001100;
'b'1010 => 'b'0001000;
'b'1011 => 'b'1100000;
'b'1100 => 'b'0110001;
'b'1101 => 'b'1000010;
'b'1110 => 'b'0110000;
'b'1111 => 'b'0111000;
}
```

This is called a “Bit Field” It is a means of grouping several variables into one entity.

The default base for numbers is HEXADECIMAL.

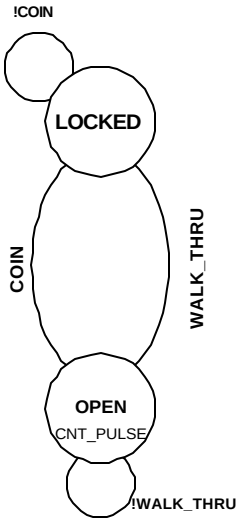
Use one of the following to denote which base you are using:

Binary	‘b’
Octal	‘o’
Decimal	‘d’
Hexadecimal	‘h’

A Subway Turnstile

The controller waits for a signal that a coin has been deposited. It then changes state from locked to open. In the open phase, it waits for someone to walk through the turnstile, then it changes from open to locked.

This two state design cycles between open and locked using a coin detector and a walk-through detector as inputs.




```
Name     Turnstil;
Partno   XXXXX;
Date     6/5/01;
Revision 01;
Designer Guam;
Company  EECE143;
Assembly XXXXX;
Location XXXXX;
Device   G16V8;

/*********************************/
/* Controls a Subway Turnstile   */
/*                               */
/*********************************/
/*  Allowable Target Device Types:  */
/*********************************/

/** Inputs **/
Pin 1 = clock;
Pin 2 = WALK_THRU;
Pin 3 = COIN;
Pin 11 = !enable;

/** Outputs **/

Pin 14 = CNT_PULSE;
Pin 15 = LOCK;

/** Declarations and Intermediate Variable Definitions
**/

/** Logic: SUbway Turnstile example expressed in CUPL
**/
$define LOCKED 'b'0
$define OPEN 'b'1

/** State Machine **/

SEQUENCE LOCK {
Present LOCKED
        if COIN Next OPEN;
        if !COIN Next Locked;

Present OPEN
        if WALK_THRU Next LOCKED;
        Default Next OPEN;
        Out CNT_PULSE;
}
```

One thing to note is that you can define your states so you can use the state variables as outputs. Your state sequence does not have to follow numeric order. That is it does not need to go 0000, 0001,0010,0011…

You can define S1 as 1011, S2 as 1111, S3 as 0110…

This will allow more flexibility for your designs.

Experiment #6: Programmable Logic Devices 2

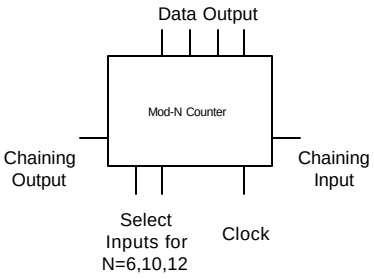
Prelab:

- Create PAL source code that implements a MOD-N counter where N can be 6,10 or 12 depending on 2 or 3 select inputs. ***Include inputs and outputs that allow synchronous chaining of counters.***

Output will count one of the following sequences depending on Select input:

0,1,2,3,4,5,0,…
0,1,2,3,…,8,9,0,…
1,2,3,…,11,12,1,…

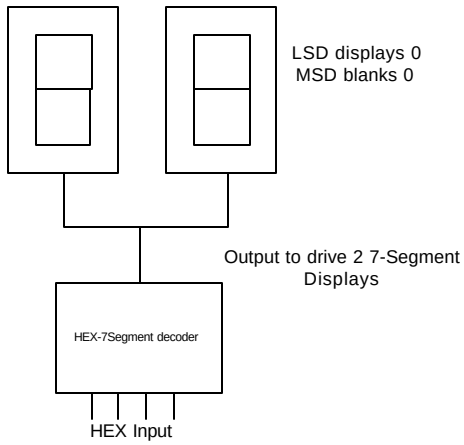
Counter will count when Chaining Input =1 and will not count when Chaining Input=0



Create PAL source code that implements a HEX to 7-segment code converter. The outputs of the code converter should be able to drive two 7-segment displays to display Hex inputs from 0 to F as decimal values from 0 to 15.

Inputs: 0000 - 1111 (0-F Hex)

Outputs: 0-15 decimal



Compile all PAL source codes. Bring source code listings (on paper) and floppy disk containing the files to lab. The files should be error free at the beginning of the lab period.

Remember to show your complete schematic diagrams and to include data tables.