PALs, GALs & CUPL

A lesson on Programmable Logic Devices and programming them using CUPL

What is Programmable Logic?

- Digital integrated circuits where the Boolean function can be determined by the user.
- PLDs can replace several specific purpose ICs in a digital design.
- A single PLD is functionally equivalent to a specific device containing from 5 to 10,000 gates.
- Typically PLDs implement Boolean functions using Sum Of Minterms (SOM) or Sum of Products (SOP) form. SOM and SOP use a AND-OR gate structure.

Basic Architecture of PLDs

- PLDs contain a pre-defined general architecture for a user to program a design into the device.
  Generally consists of one or more arrays of AND and OR terms
- Also contain flip-flops and latches to be used as storage elements for inputs and outputs
- Consist of programmable AND terms feeding fixed OR terms.
  All inputs can be AND-ed together but specific AND terms are dedicated to specific OR terms.
  Pins can be used as Input, Output, I/O with tri-state enables, also may have output registers.

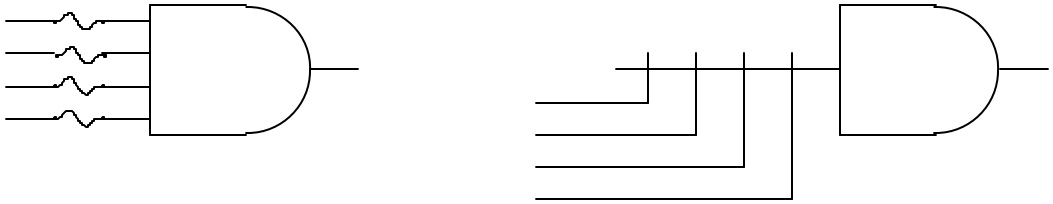
PLD Programming

PLDs are manufactured in a "blank" or "erased" form.

Programming is performed in concept blowing out fuses between inputs, AND gates, and OR gates in the generic AND-OR structure. An erased PLD has all fuses intact. Actual "fuses" may be implemented as:

Type	Function	Advantages
fuses	one-time programmable	low cost
EPROM	uv light erasable	Reprogrammable
CMOS EEPROM	electrically erasable	fast, easy reprogramming

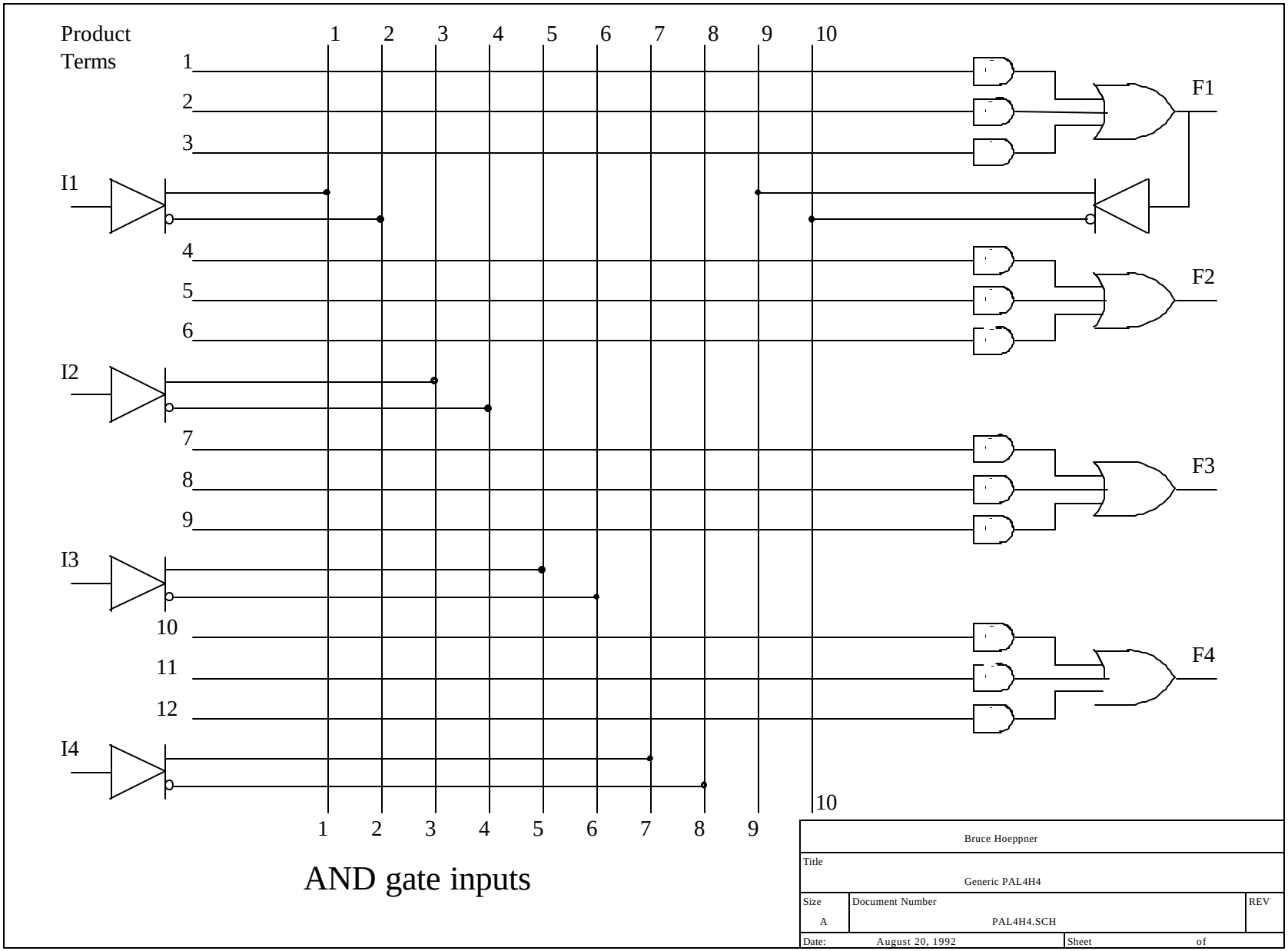
Fuses and Logic Implementation



A blown fuse acts like the input does not exist (or a logic 1 at the input).

PLD Advantages:

- reduce IC package count
- board space
- power
- shorten design time
- allow for future changes (maintainability)
- improve reliability (fewer packages)
- generally faster
- smaller inventory



PAL Example: Given functions w, x, y, and z. Implement with one PAL4H4.

Given: Sum of Minterms

$w(A,B,C,D) = \Sigma(2,12,13)$

$x(A,B,C,D) = \Sigma(7,8,9,10,11,12,13,14,15)$

$y(A,B,C,D) = \Sigma(0,2,3,4,5,6,7,8,10,11,15)$

$z(A,B,C,D) = \Sigma(1,2,8,12,13)$

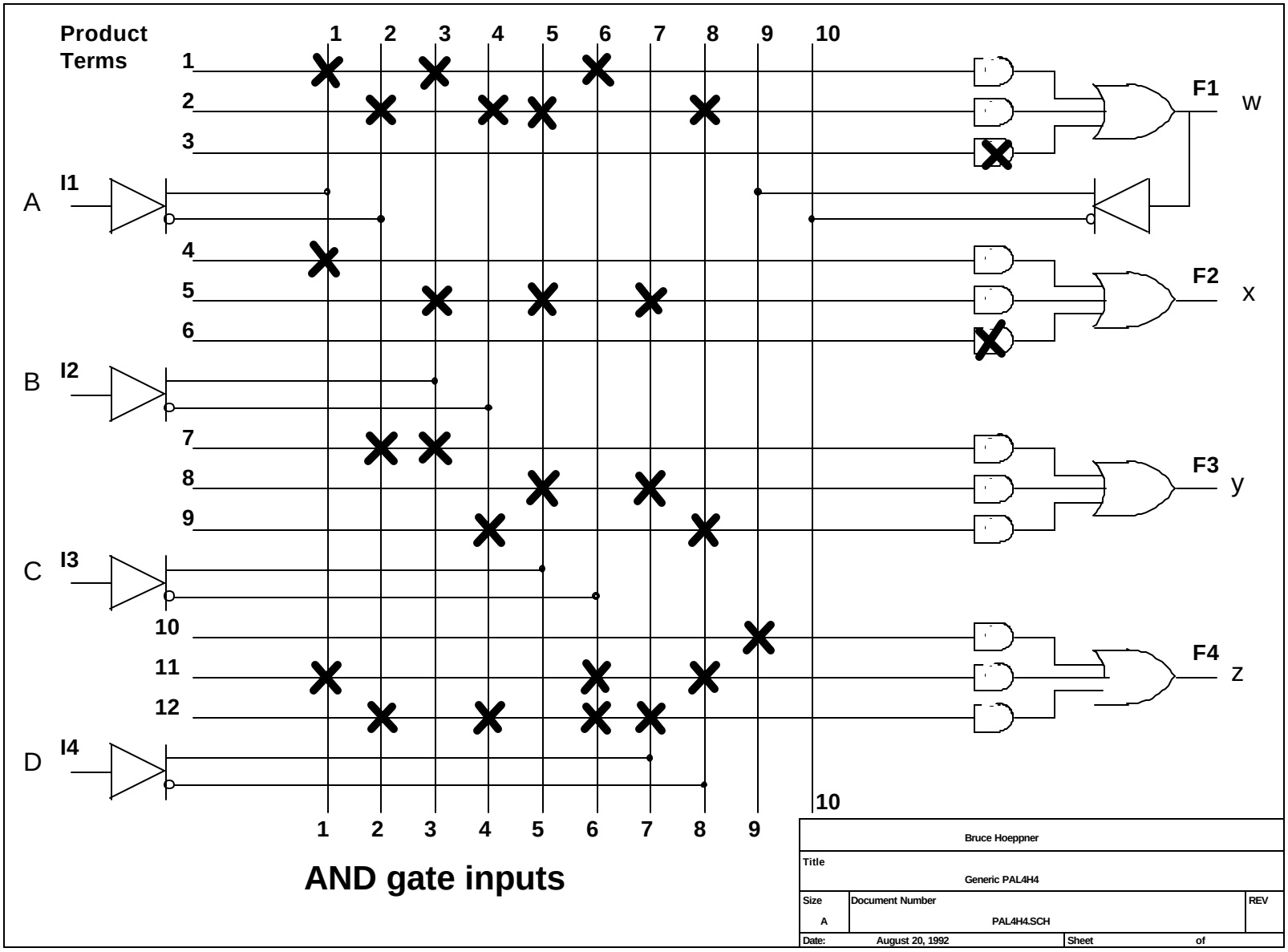
... After Simplification

w = ABC' + A'B'CD'

x = A + BCD

y = A'B + CD + B'D'

z = w + AC'D' + A'B'C'D



Programmable Array Logic

PALXXYZ

XX : Number of possible inputs

Y : L = Combinational R = Registers

Z : Number of possible outputs

Examples:

PAL16R4 : 16 inputs with 4 registered outputs

PAL16L8 : 16 inputs with 8 outputs

Generic Array Logic

Designed to emulate many common PALs

Reduces number of different devices

Electronically erasable

Such as GAL16V8, GAL22V10

GAL16V8

- 8 output Macrocells
- Pin 1: Clock
- Pin 11: Active-low Output Enable
  - Must be tied low to use Registered outputs
  - Must be declared in .PLD file
- Can emulate various PAL devices
- 8 Product Terms per Output

GAL22V10

- 10 output Macrocells
- 24 Pin Package
- Pin 1: Clock
- Pin 13: Active-low Output Enable
- Can emulate various PAL devices
- Pin 14,23 = 8 Product Terms
- Pins 15,22 = 10 Product Terms
- Pins 16,21 = 12 Product Terms
- Pins 17,20 = 14 Product Terms
- Pins 18,19 = 16 Product Terms

WinCUPL Programming Language

WinCUPL is a software package that runs on an PC. It performs most of the work in translating a PLD design into a programming file. The programming file can be used to program an IC to implement the desired logic functions. Marquette University (College of Engineering) has a site license for CUPL.

- Translates PLD Design into programming file.
- CUPL expressions written in Sum of Products form. *Limit of 7 product terms*

! Invert

& AND

OR

$ XOR

CUPL Programming

- Combinational
  - Use equations or Truth table
  - Examples1.PLD, Example2.PLD
- Sequential
  - Use Equations or State Machine
  - Example3.PLD, Example4.PLD

Combinational Logic Implementation

Problem: CUPL expressions must be written for the function. Sum of Products (SOP) is the form CUPL uses. There is a limit of 7 product terms.

Given: A sum of products equation.

Find: A complement form of the equation also in sum of products.

Example: A = x'y + xy'z

In CUPL this becomes: **A = !x&y # x&!y&z;**

Example: B = x'y'z' + xy' + yz

In CUPL this becomes: **B = !x&!y&!z # x&!y # y&z;**

Example1.pld

```
Name        Example1;
Partno      none;
Date        06/01/98;
Revision    01;
Designer    Bruce Hoeppner;
Company     EECE143;
Assembly    none;
Location    EN365;
Device      G16V8;
/*************************************************/
/*  Example program to implement the following equations */
/*  in CUPL Programming language                 */
/*                                               */
/*     A = x'y + xy'z                            */
/*     B = x'y'z' + xy' + yz                     */
/*                                               */
/*************************************************/
```

Example1.pld

```
/**  Inputs  **/
Pin 2 = x;
Pin 3 = y;
Pin 4 = z;
/**  Outputs  **/
Pin 19 = A;
Pin 18 = B;
/** Declarations and Intermediate Variable Definitions **/
/**  Logic Equations  **/
A = !x&y # x&!y&z;
B = !x&!y&!z # x&!y # y&z;
/** End of example1.pld source code **/
```

Example2.pld

```
Name        Example2;
Partno      example2.pld;
Date        06/01/98;
Revision    01;
Designer    Bruce Hoeppner;
Company     EECE143;
Assembly    PLD Source Code;
Location    EN365;
Device      G16V8;

/*************************************************/
/*  Example program to implement the following equations*/
/*  in CUPL programming language using Truth Tables     */
/*                                                      */
/*    A = x'y + xy'z                                    */
/*    B = x'y'z' + xy' + yz                             */
/*                                                      */
/*************************************************/
/*  Allowable Target Device Types:                      */
/*************************************************/
```

Example2.pld

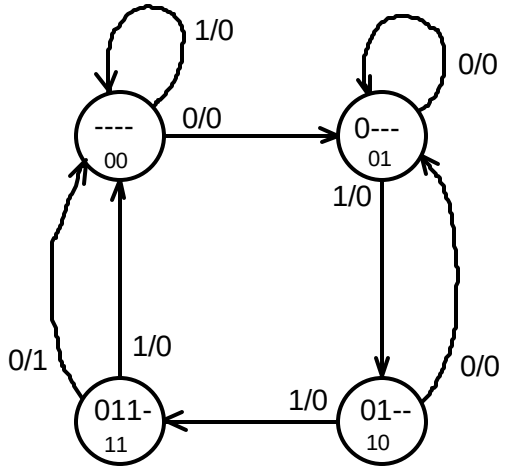
```
/**  Inputs  **/
Pin 2 = x;
Pin 3 = y;
Pin 4 = z;
/**  Outputs  **/
Pin 19 = A;
Pin 18 = B;
/** Declarations and Intermediate Variable Definitions **/
FIELD INPUT = [x,y,z];
FIELD OUTPUT = [A,B];
/**  Truth Table  **/
TABLE INPUT => OUTPUT {
0=>'b'01;
1=>'b'00;
2=>'b'10;
3=>'b'11;
4=>'b'01;
5=>'b'11;
6=>'b'00;
7=>'b'01;
}
/** End of example2.pld source code **/
```

Sequential Circuit Implementation of Programmable Logic

Given: The state diagram for a sequence recognizer (0110)

Problem: Implement using one PAL16R4 or GAL16V8

State	Input	Next	Output
B A	s	B A	found
0 0	0	0 1	0
0 0	1	0 0	0
0 1	0	0 1	0
0 1	1	1 0	0
1 0	0	0 1	0
1 0	1	1 1	0
1 1	0	0 0	1
1 1	1	0 0	0



Insert into K-Maps

B

B As \	00	01	11	10
0				
1		1		1

DB = s(B'A) + s(BA')

A

B As \	00	01	11	10
0	1	1		1
1				1

DA = s'B' + BA'

Let: pin 1 = clock, pin 2 = s, pin 19 = found, pin 17 = B, pin 16 = A, and pin 11 = !enable

Written in CUPL form:

B.d = s&!B&A # s&B&!A;

A.d = !s&!B # B&!A;

The output equation is:

B As \	00	01	11	10
0			1	
1				

found = !s&B&A;

Example3.pld

```
Name        Example3;
Partno      example3.pld;
Date        06/01/98;
Revision    01;
Designer    Bruce Hoeppner;
Company     EECE143;
Assembly    PLD Source Code;
Location    EN365;
Device      G16V8;

/*************************************************/
/*  Example program to implement the following equations*/
/*  in CUPL programming language using Equations        */
/*                                                      */
/*    DA = s'B' + BA'                                   */
/*    DB = sB'A + sBA'                                  */
/*    found = s'BA                                      */
*/
/*************************************************/
/*  Allowable Target Device Types:                     */
/*************************************************/
```

```
/**  Inputs  **/
Pin 1 = clock;
Pin 2 = s;                    /* combinational data input */
Pin 11 = !enable;

/**  Outputs  **/

Pin 19 = found;
Pin 17 = B;
Pin 16 = A;

/** Declarations and Intermediate Variable Definitions **/

/**  Equations  **/
B.d = s&!B&A # s&B&!A;
A.d = !s&!B # B&!A;
found = !s&B&A;
/** End of example3.pld source code **/
```

Example4.pld

```
Name        Example4;
Partno      example4.pld;
Date        06/01/98;
Revision    01;
Designer    Bruce Hoeppner;
Company     EECE143;
Assembly    PLD Source Code;
Location    EN365;
Device      G16V8;

/*************************************************/
/*  Example program to implement the sequential circuit*/
/*  in CUPL programming language using State Machines  */
/*                                                     */
/*                                                     */
/*************************************************/
/*  Allowable Target Device Types:                     */
/*************************************************/
```

```
/**  Inputs  **/
Pin 1 = clock;
Pin 2 = s;                    /* combinational data input */
Pin 11 = !enable;
/**  Outputs  **/
Pin [14..15] =[Q1..0];
Pin 19 = found;                       /* combinational output
*/
/** Declarations and Intermediate Variable Definitions **/
field state = [Q1..0];
$define S0 'b'00
$define S1 'b'01
$define S2 'b'10
$define S3 'b'11
```

```
/**  State Machine  **/
SEQUENCE state {
present S0    if s   next S0;
              default      next S1;
present S1    if s   next S2;
              default      next S1;
present S2    if s   next S3;
              default next S1;
present S3    if s   next S0      out found;
              default      next S0;
}
/** End of example4.pld source code **/
```

Experiment #5: Programmable Logic Devices 1

Programmable Logic Devices as replacements/improvements to standard logic ICs.

Goals:

Learn to use programmable logic devices as MSI TTL & CMOS replacements.

Prelab:

1. Create PAL source code that implements equations D, E &F from Experiment #1.
2. Create PAL source code that implements the sequential circuit for the circuit of Figure 4.1 (state diagram) of Experiment #4.
3. 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.

Compiling PLD File

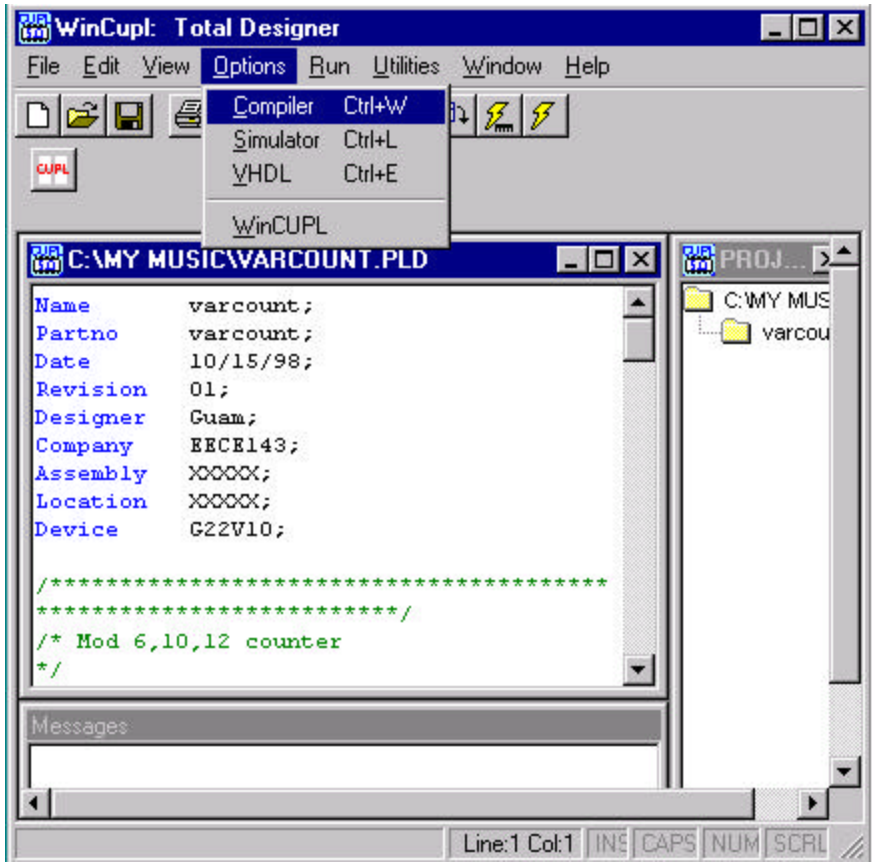
- Use WinCUPL or use another text editor to create your file
- Save your file with the extension .PLD
- Compile using WinCUPL make sure you have no errors.
- Save your .JED file on diskette

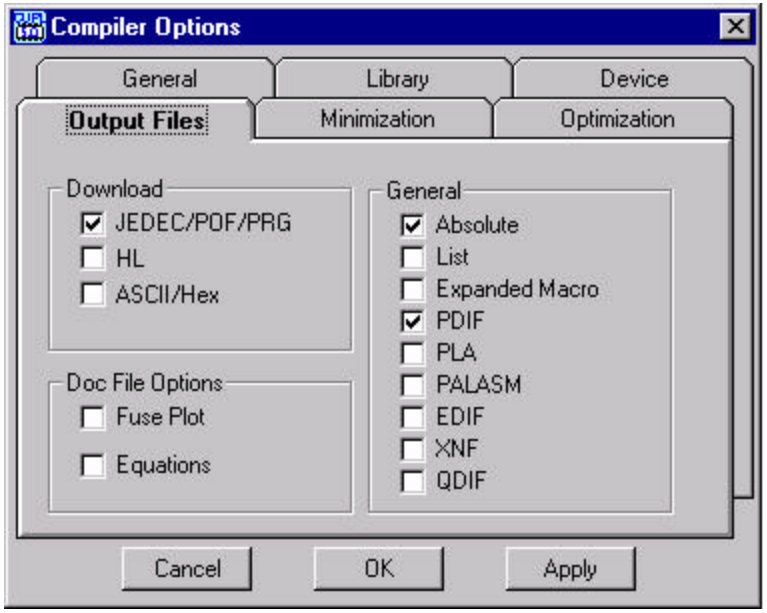
Using WinCUPL

Located in Engineering Applications\ EECE\ Wincupl folder

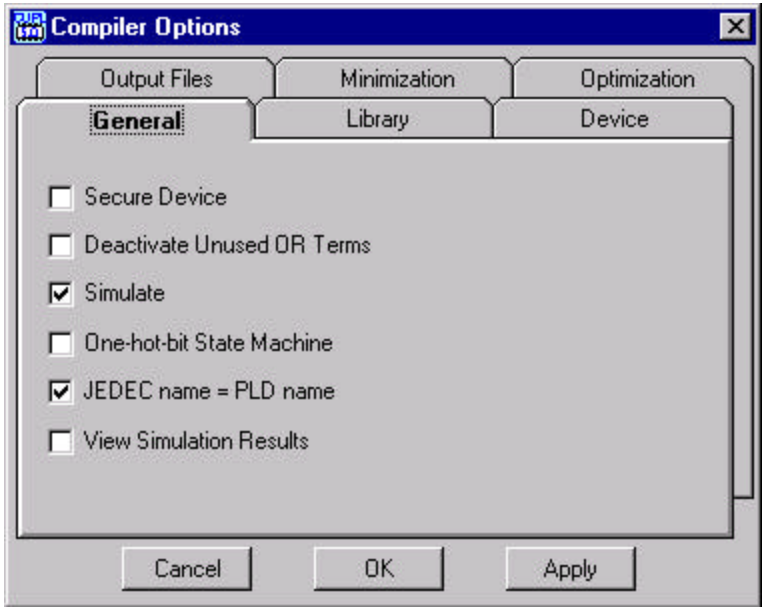
After creating and editing your PLD file:

Open Options|Compiler menu item or type Ctrl+W

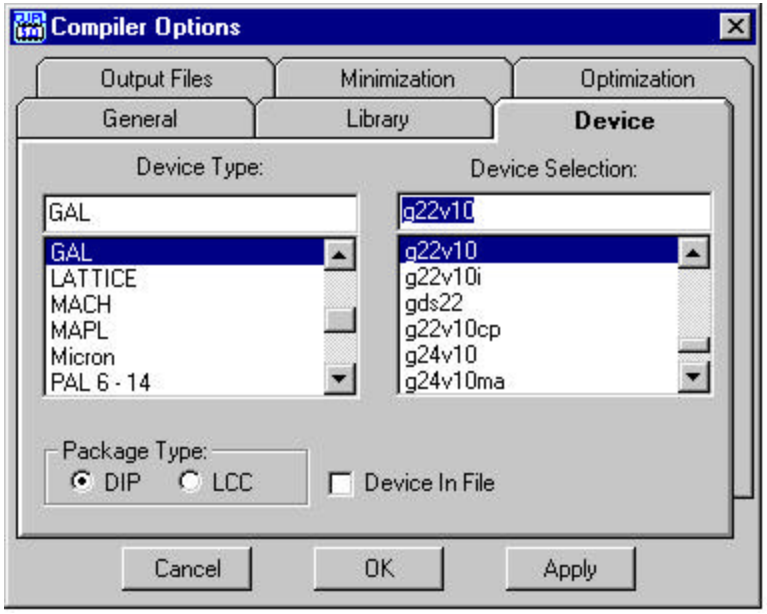




Select Download JEDEC/PDF/PRG



Select JEDEC name = PLD name



Select Device:

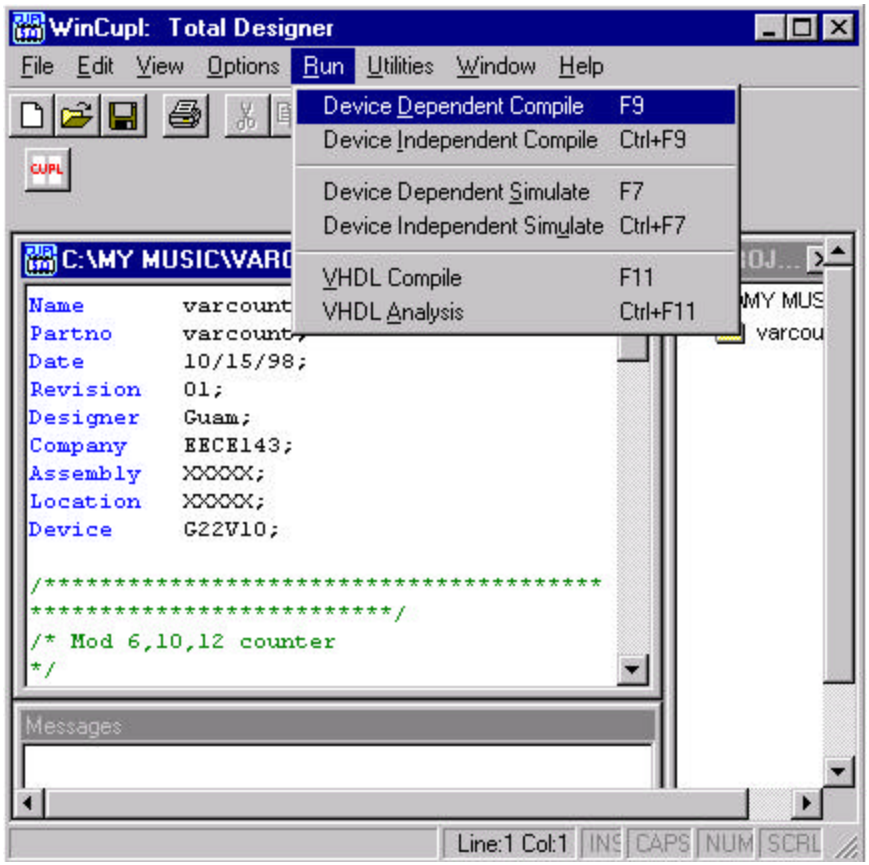
Either g16v8 for GAL16V8

Or g22v10 for GAL22V10

DIP Package

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.

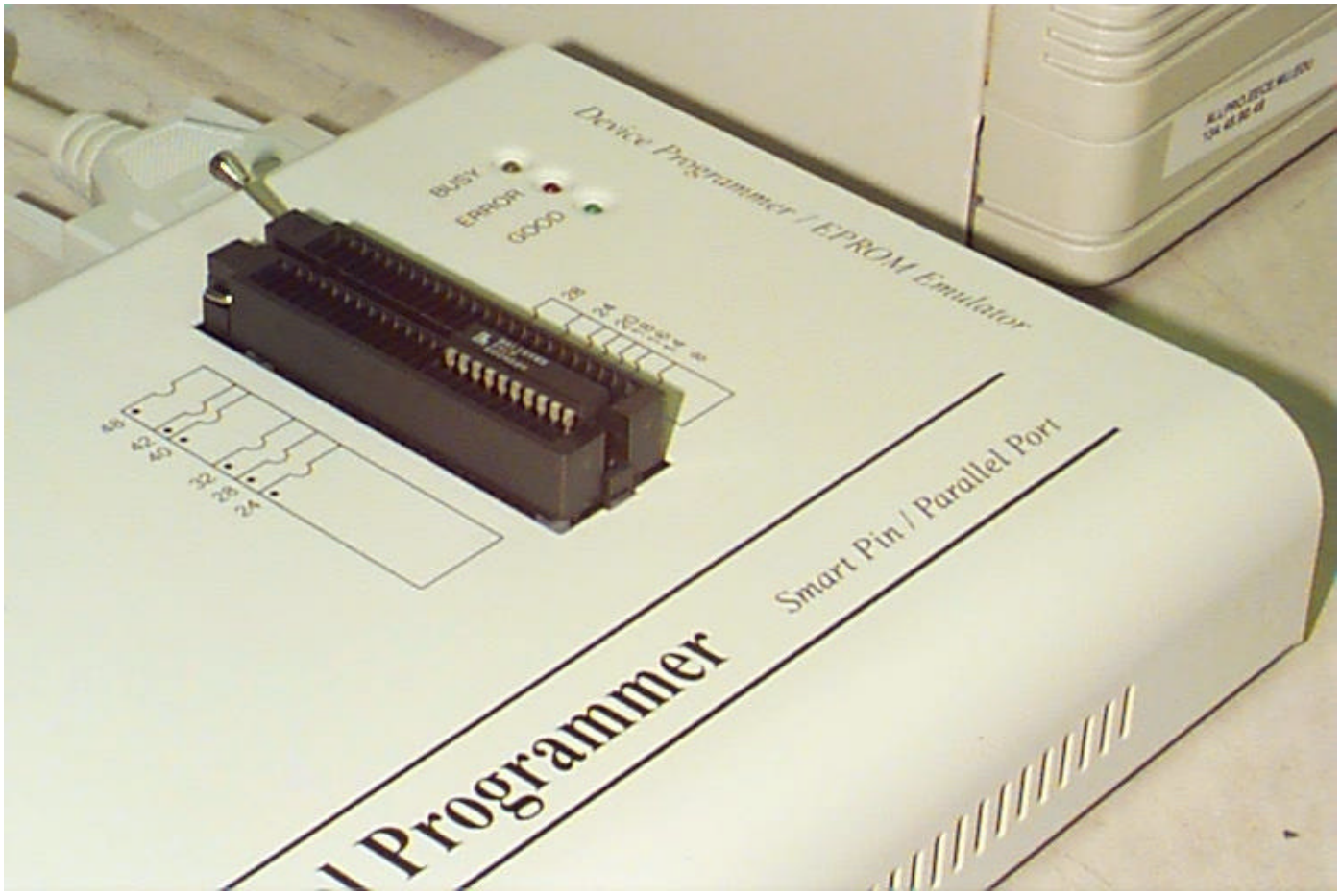
ChipMaster 6000
Intelligent Universal Programmer

Using the CM6000 Universal Programmer

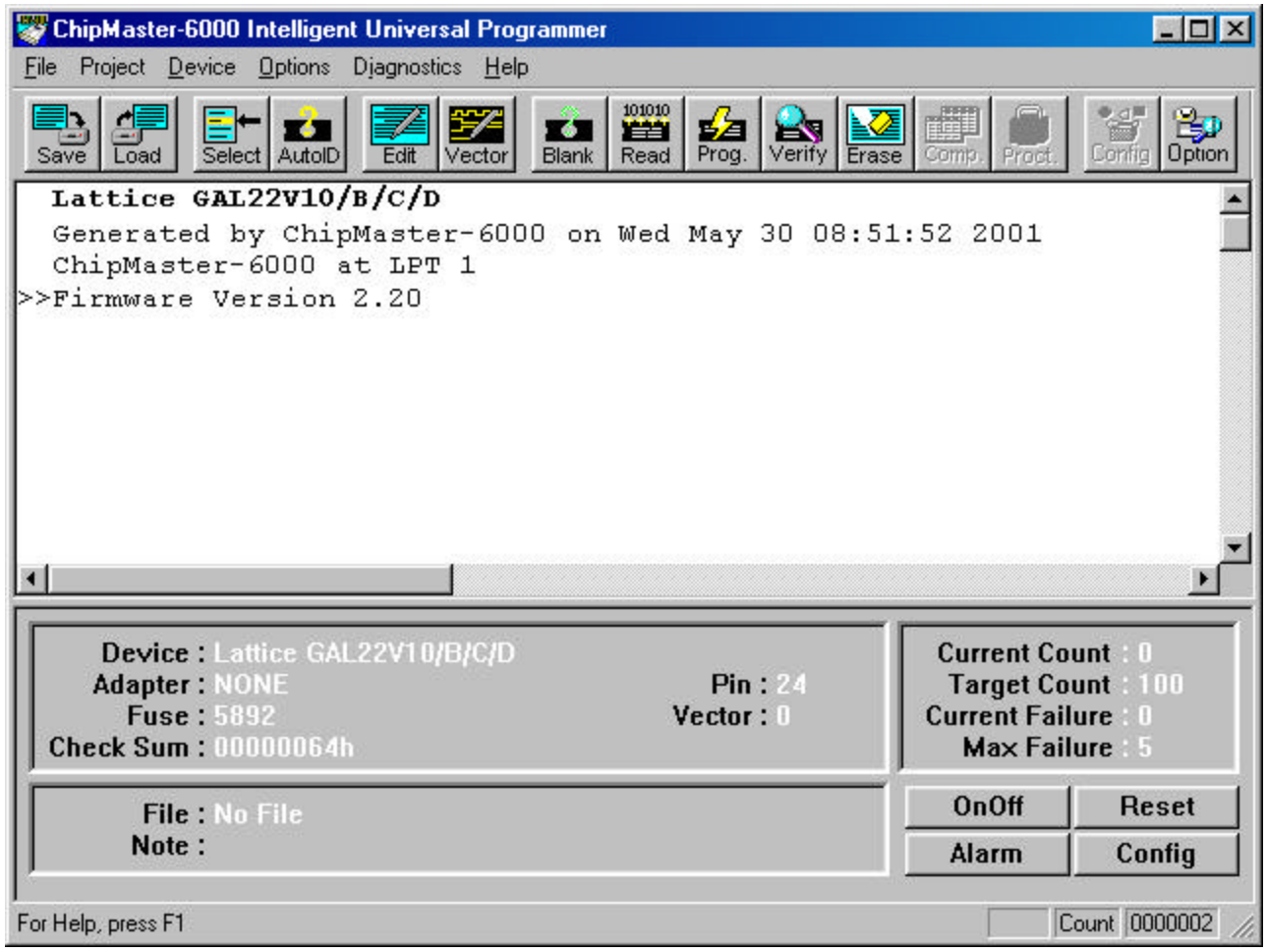
1. Load the device
2. Select a device
3. Perform Blank Check
4. Erase device
5. Read jedec file
6. Program device
7. Verify

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

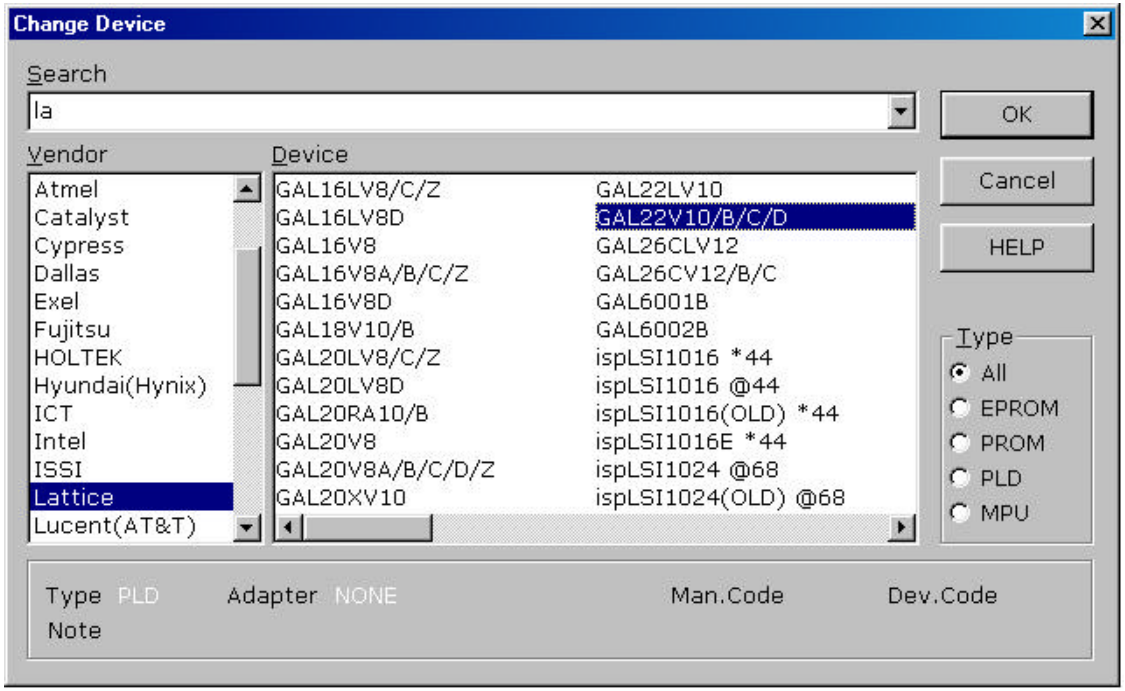
1. Type ALT-C or Select the Device|Change Device menu item.
2. Select the device required:

   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’

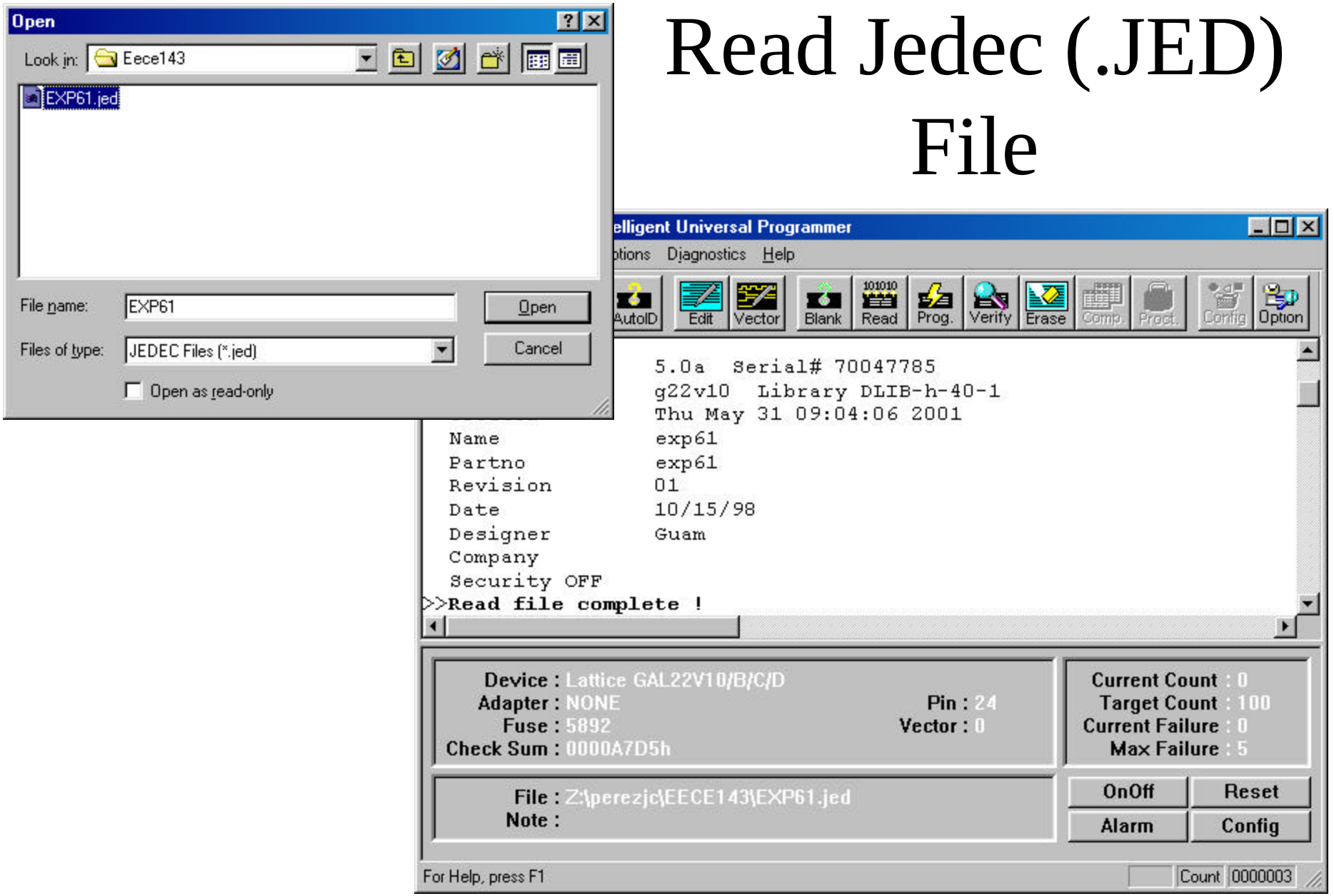
   Use keyboard: Press TAB until the cursor flashes in the ‘Search’ box. Type in the part number.
4. Use the mouse to select a vendor.
5. Select a device and click ‘OK. Use the TAB key to skip between various screens, use the arrow keys to move around each screen. Press <ENTER> to select the vendor/device.



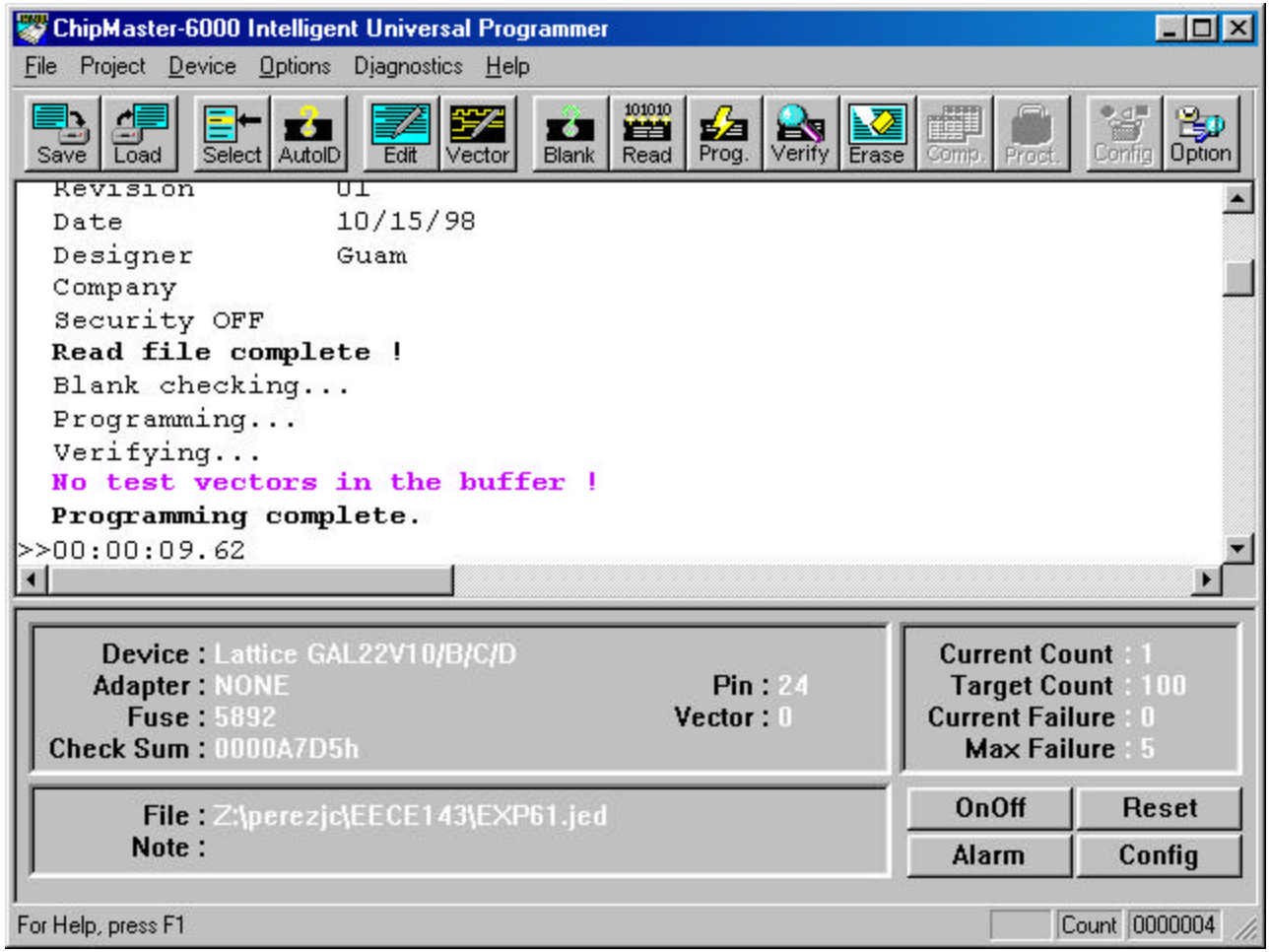
Load File into Memory

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



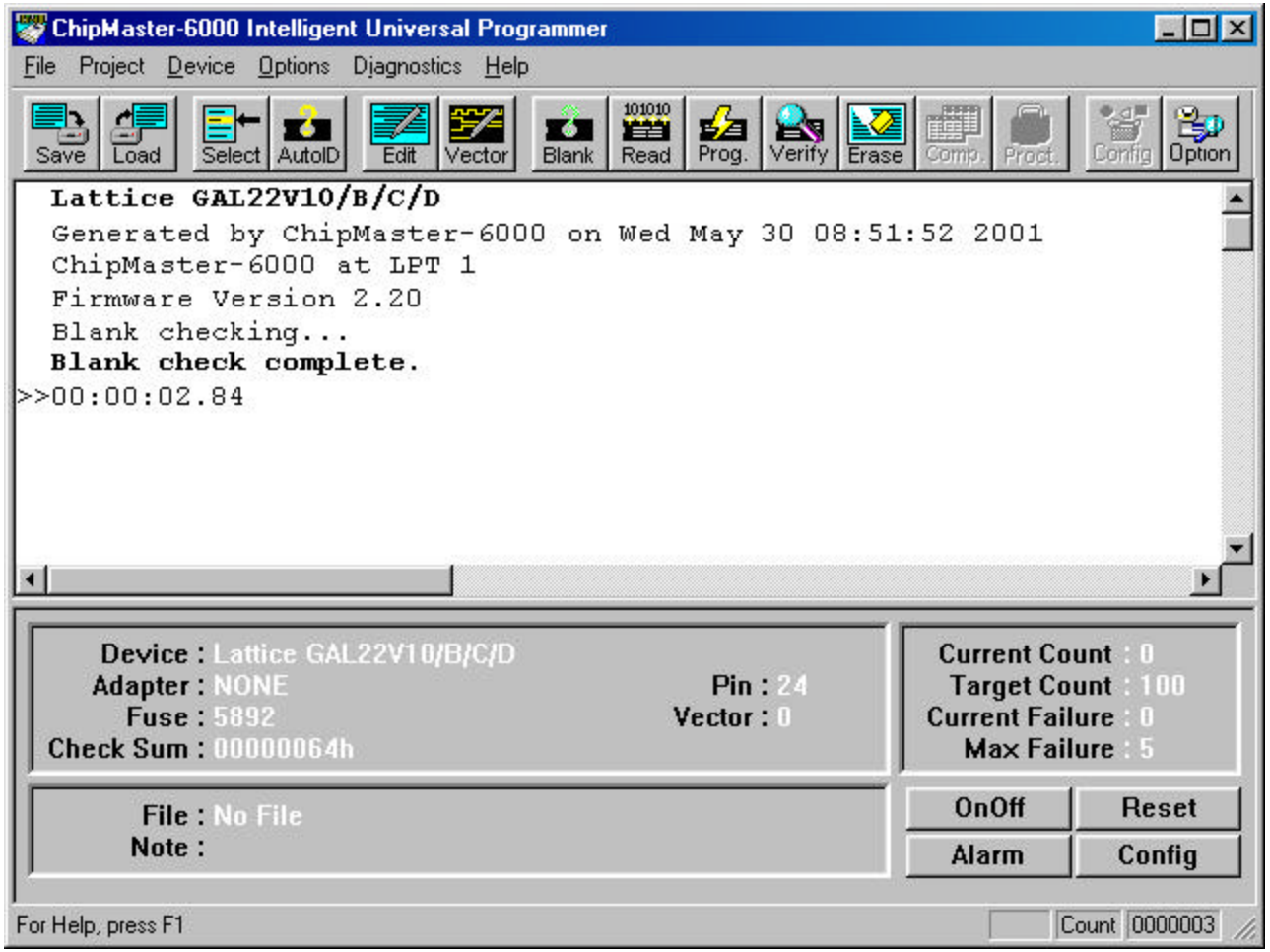
Program Device



Blank check the device

Insert the device in the socket.

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



Erase the device

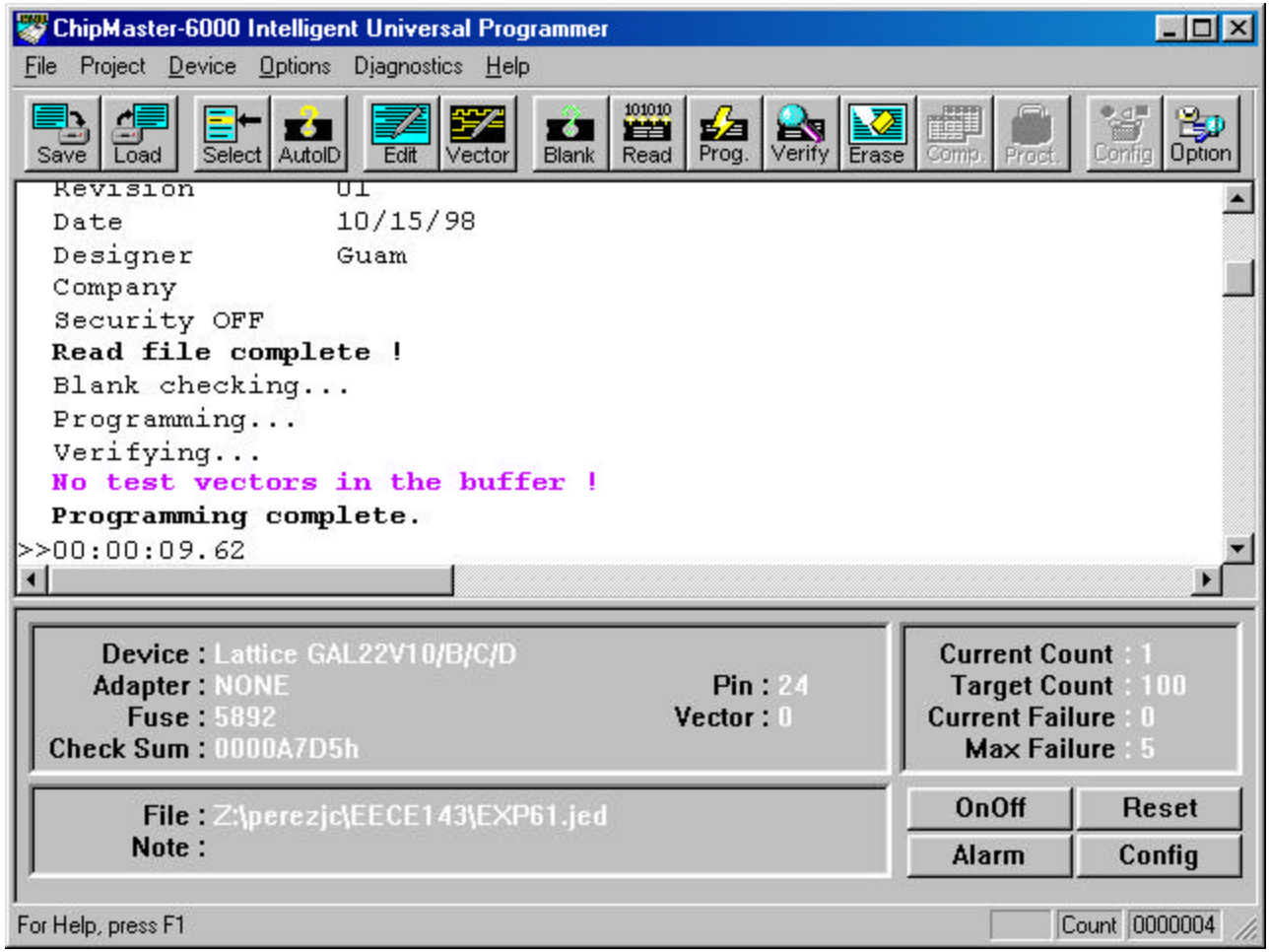
This option is only available for devices that support chip erase.

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.

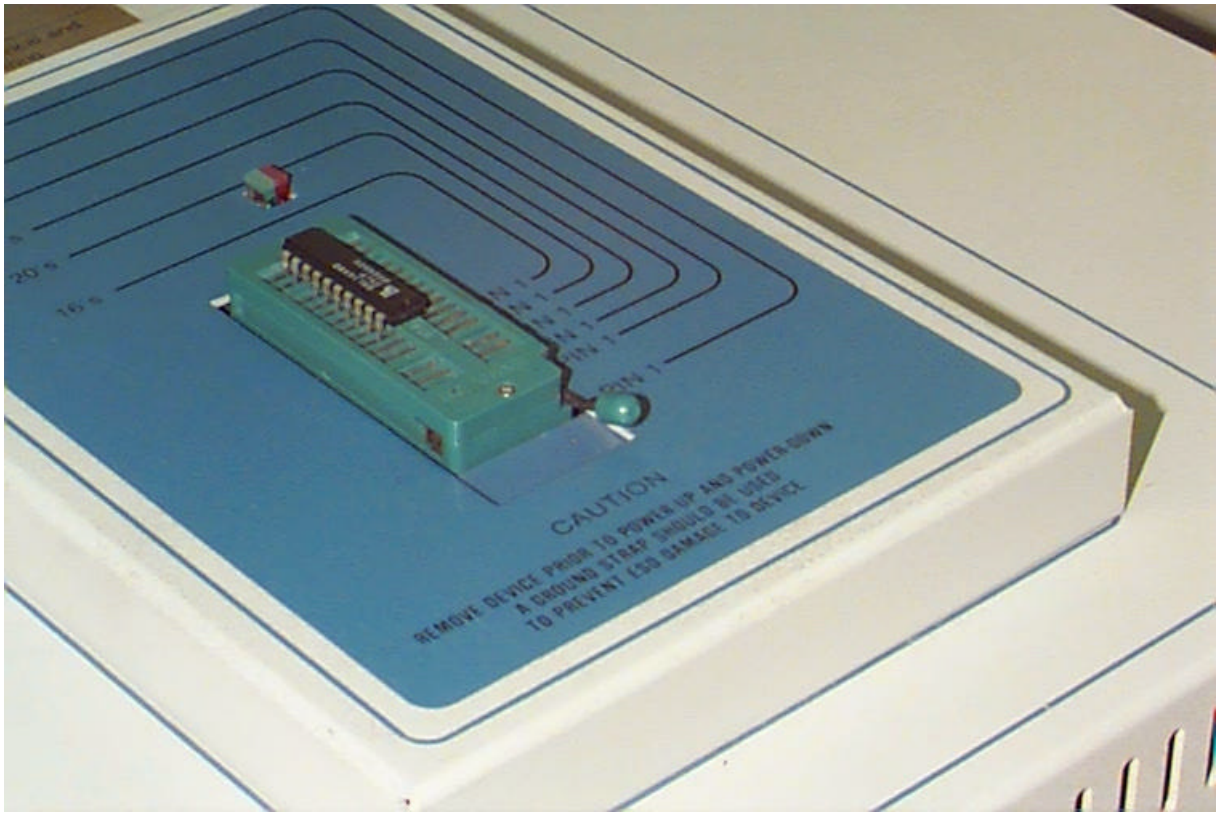
AllPro 88 Programmer

Using the AllPro 88 Programmer

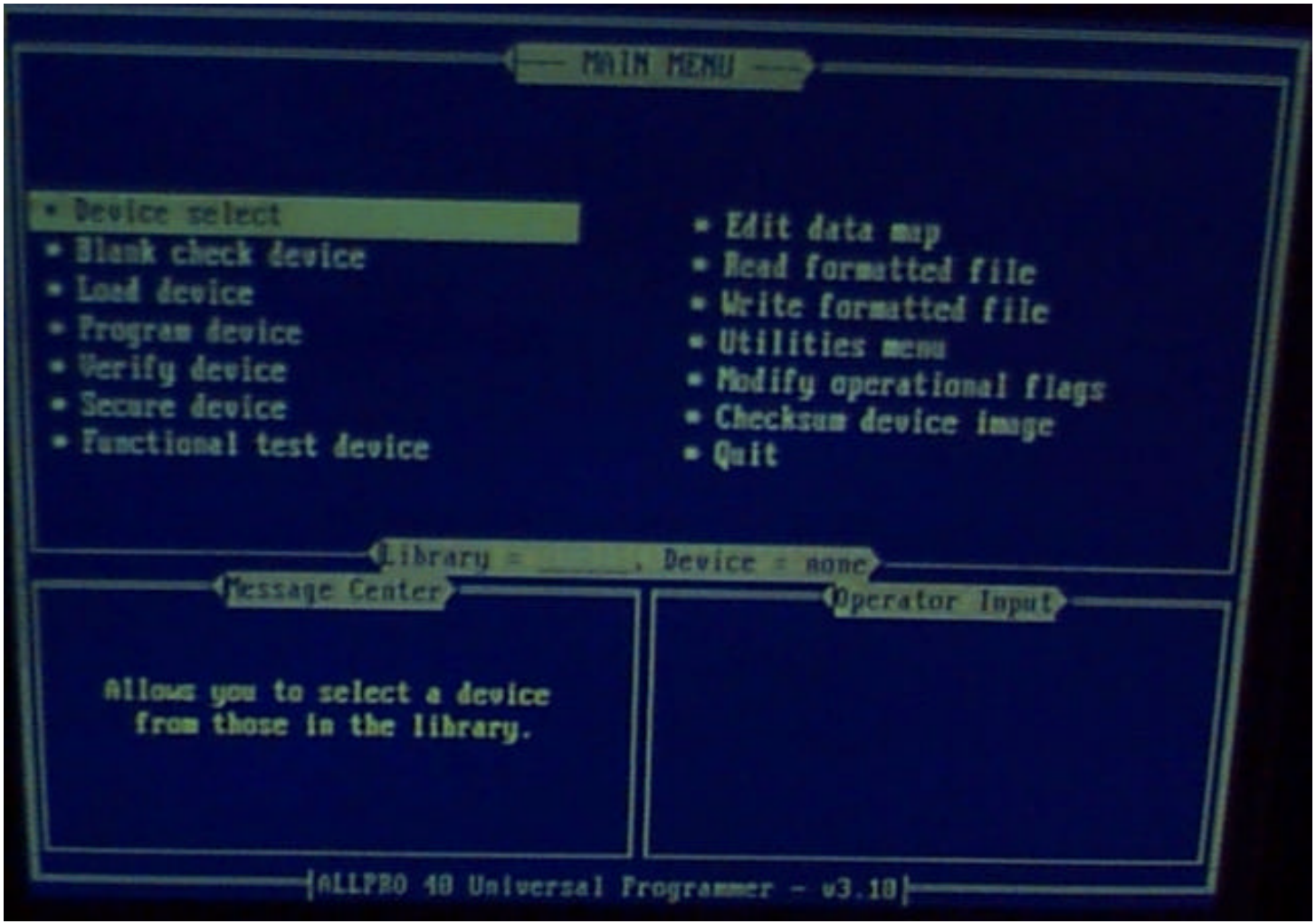
- Load Device into programmer
- Choose Library and Device from library
- Blank Check Device (optional)
- Read Jed file
- Program Device

Load Device in Programmer

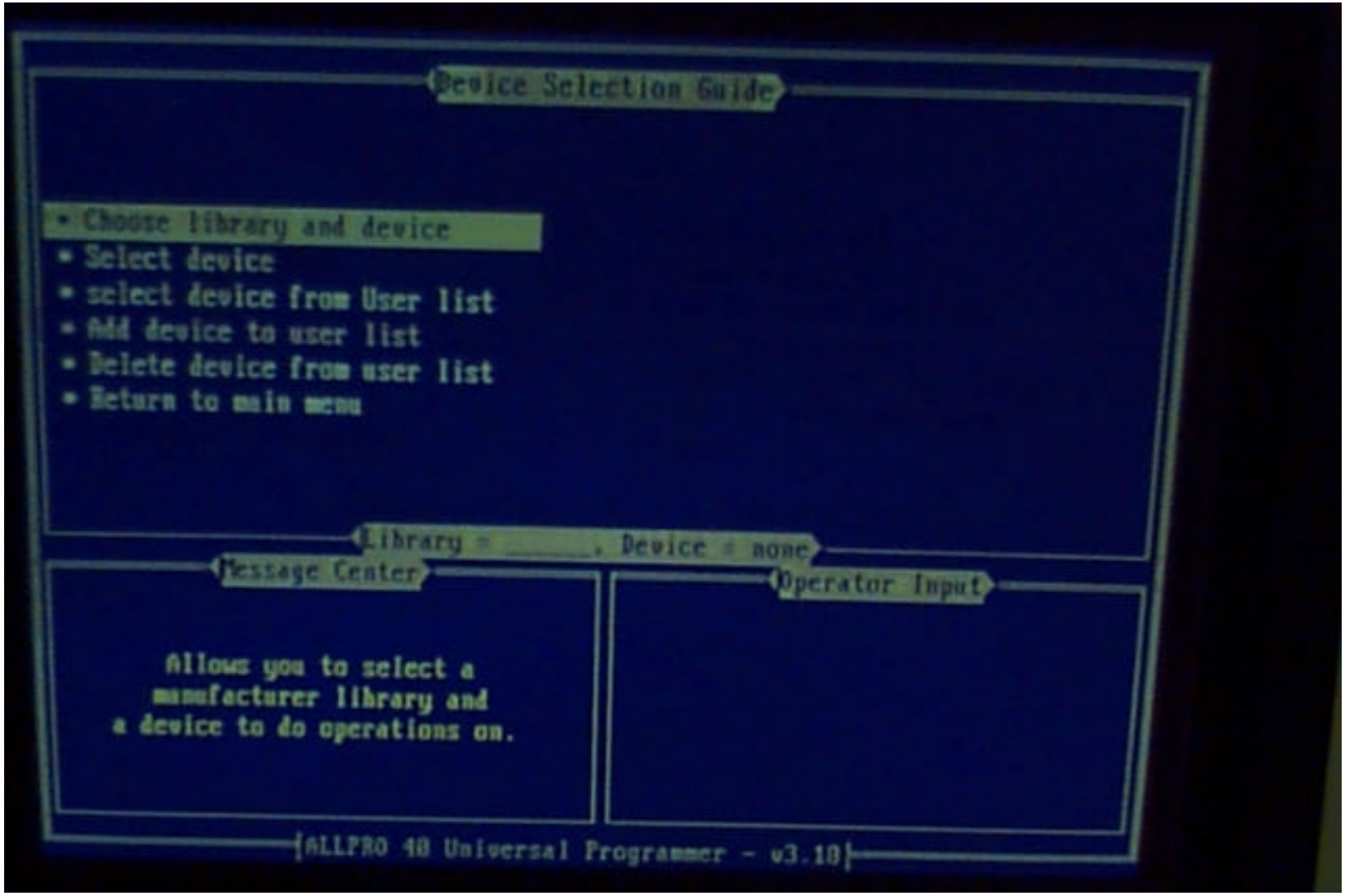
Place chip so Pin 1 faces front of programmer.



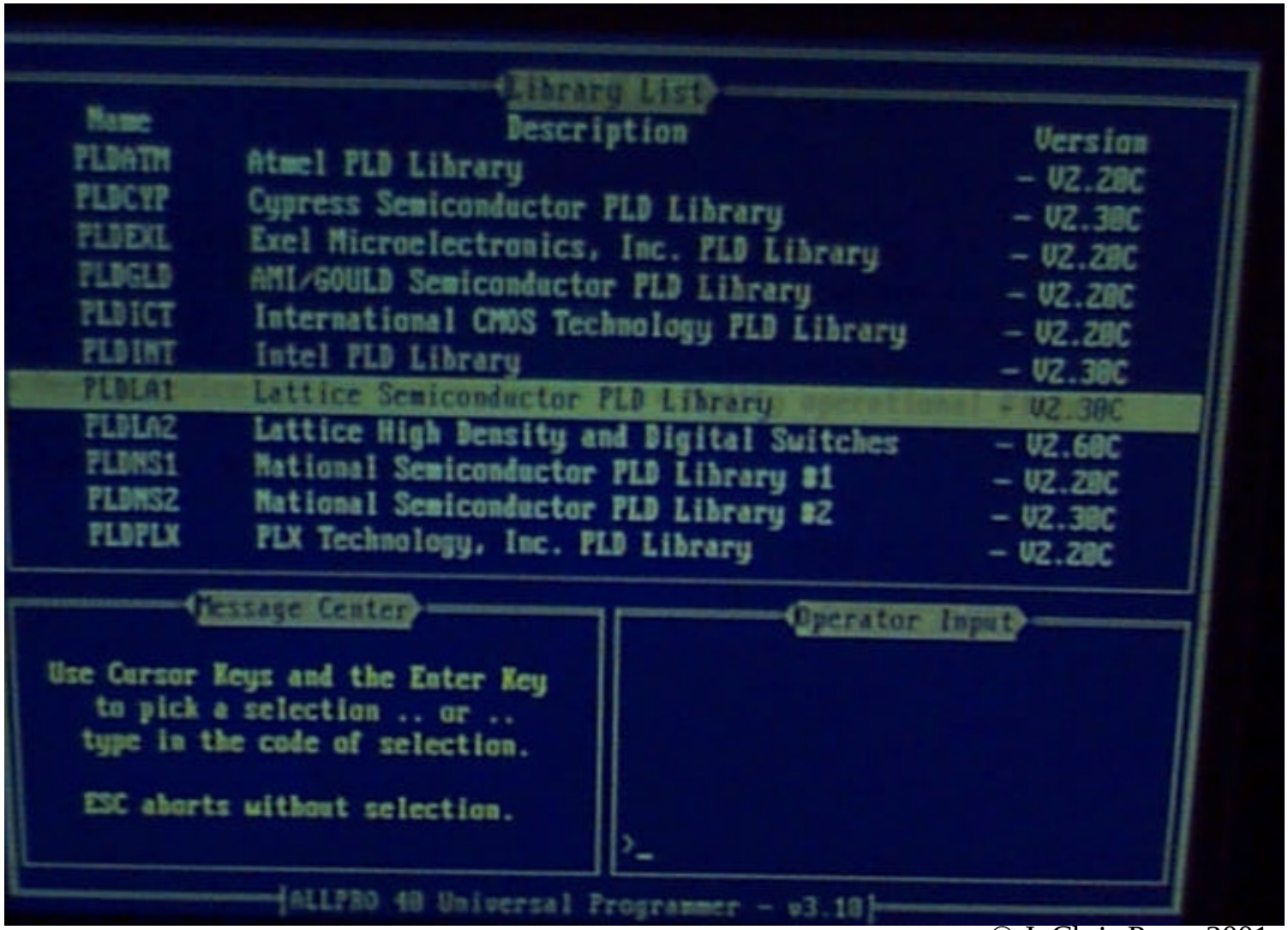
Choose Library and Device from library



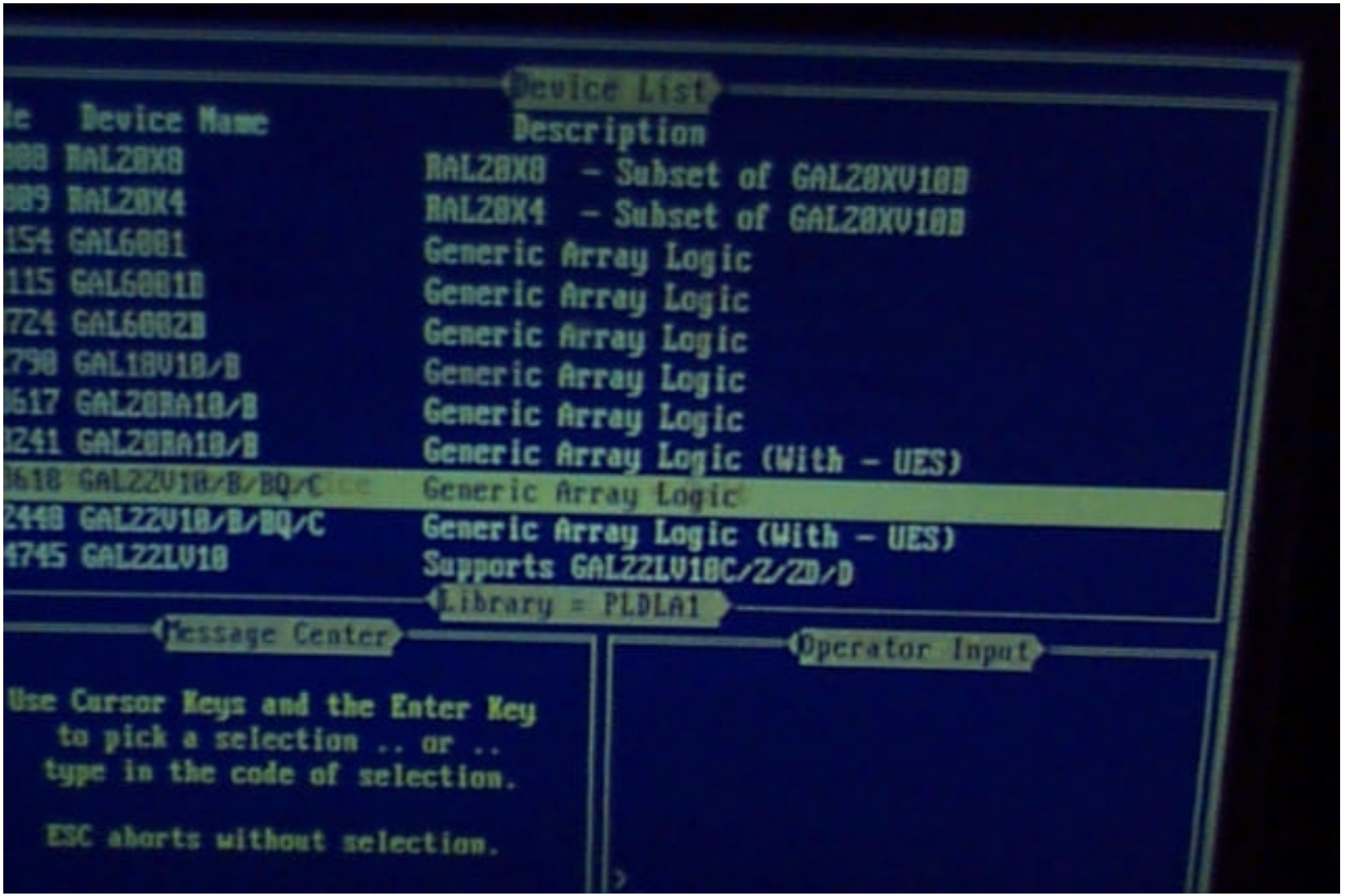
Choose Library and Device from library



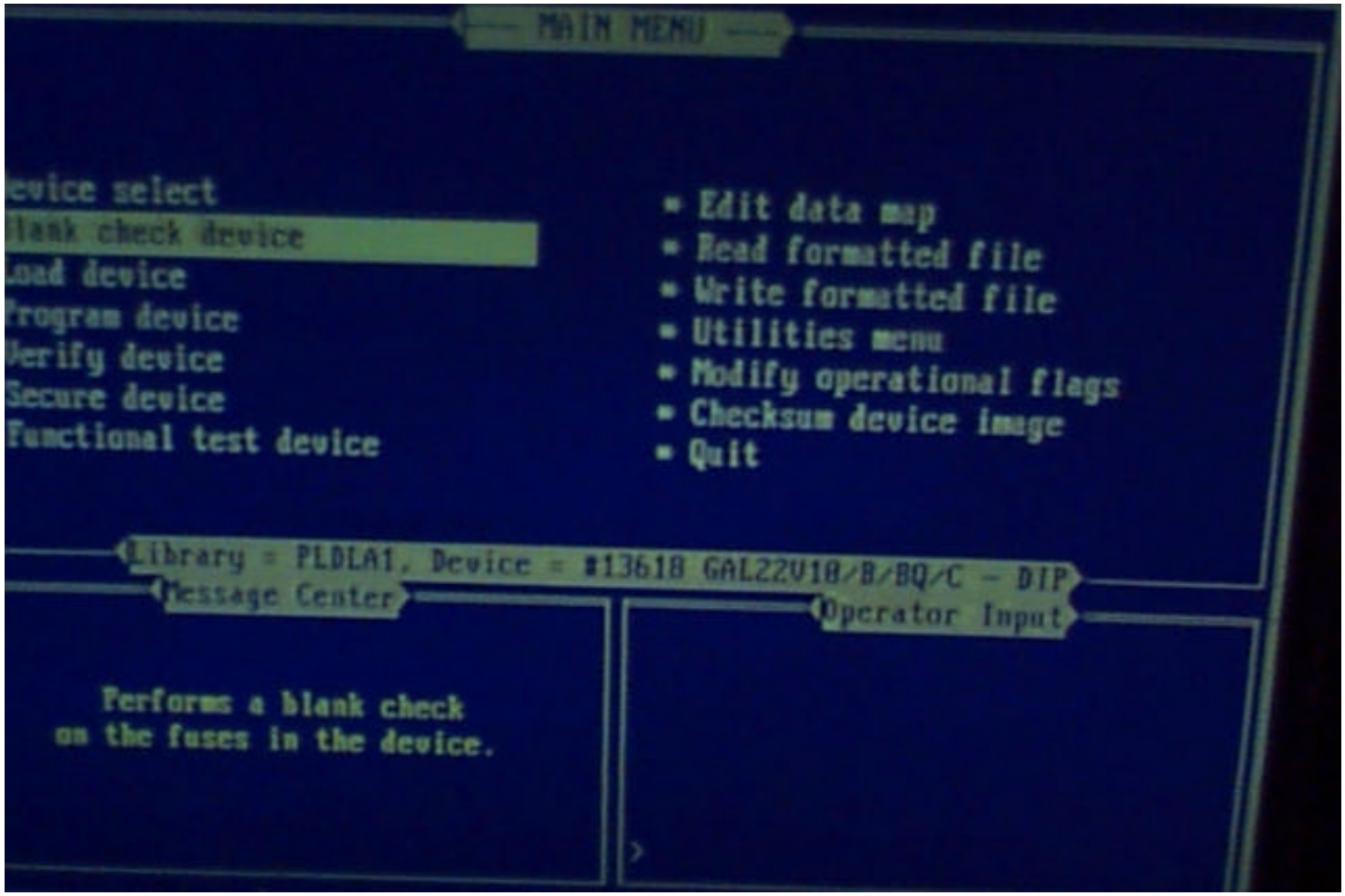
Choose Library …



Choose Device …

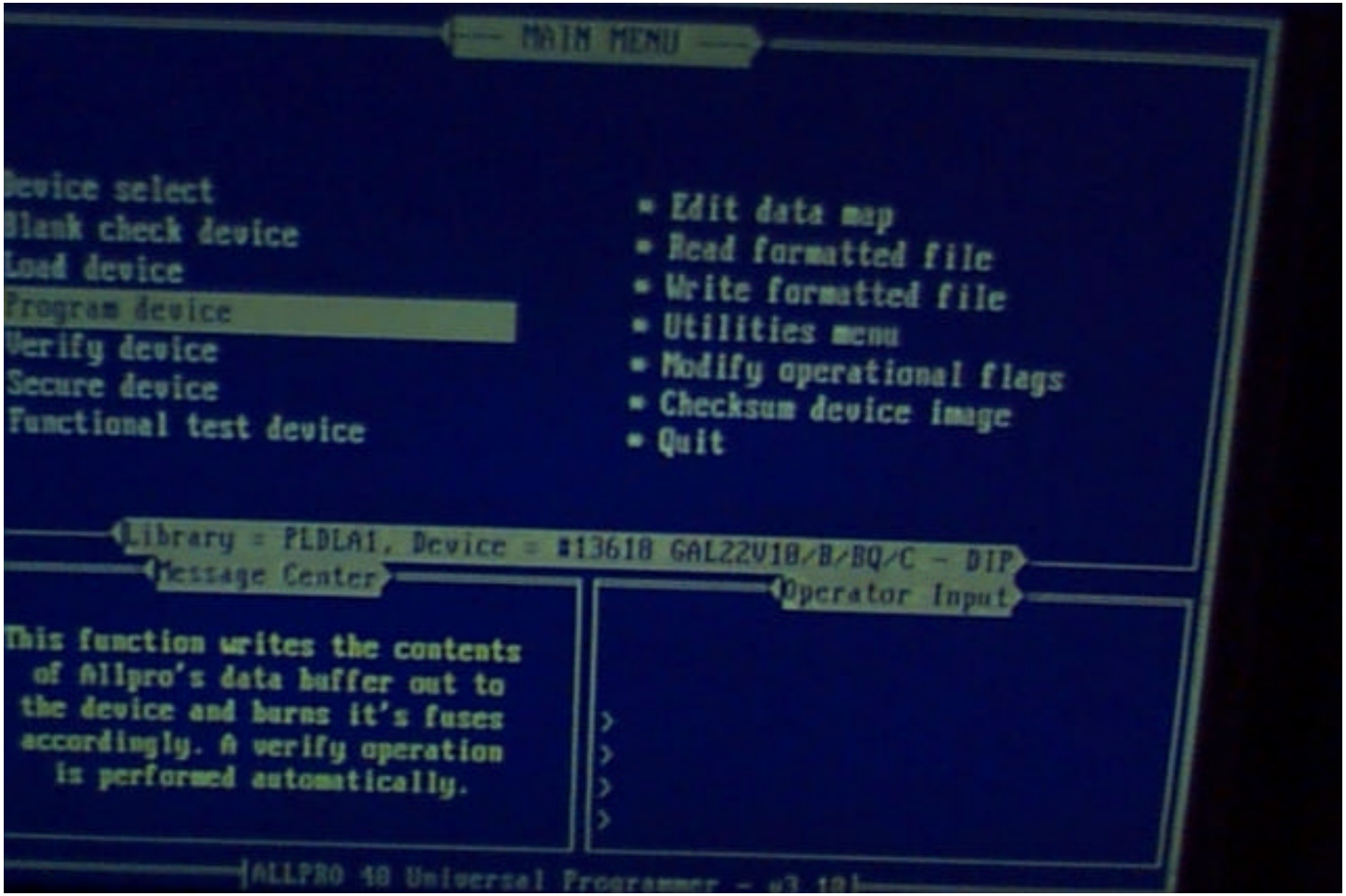


Blank Check Device (optional)

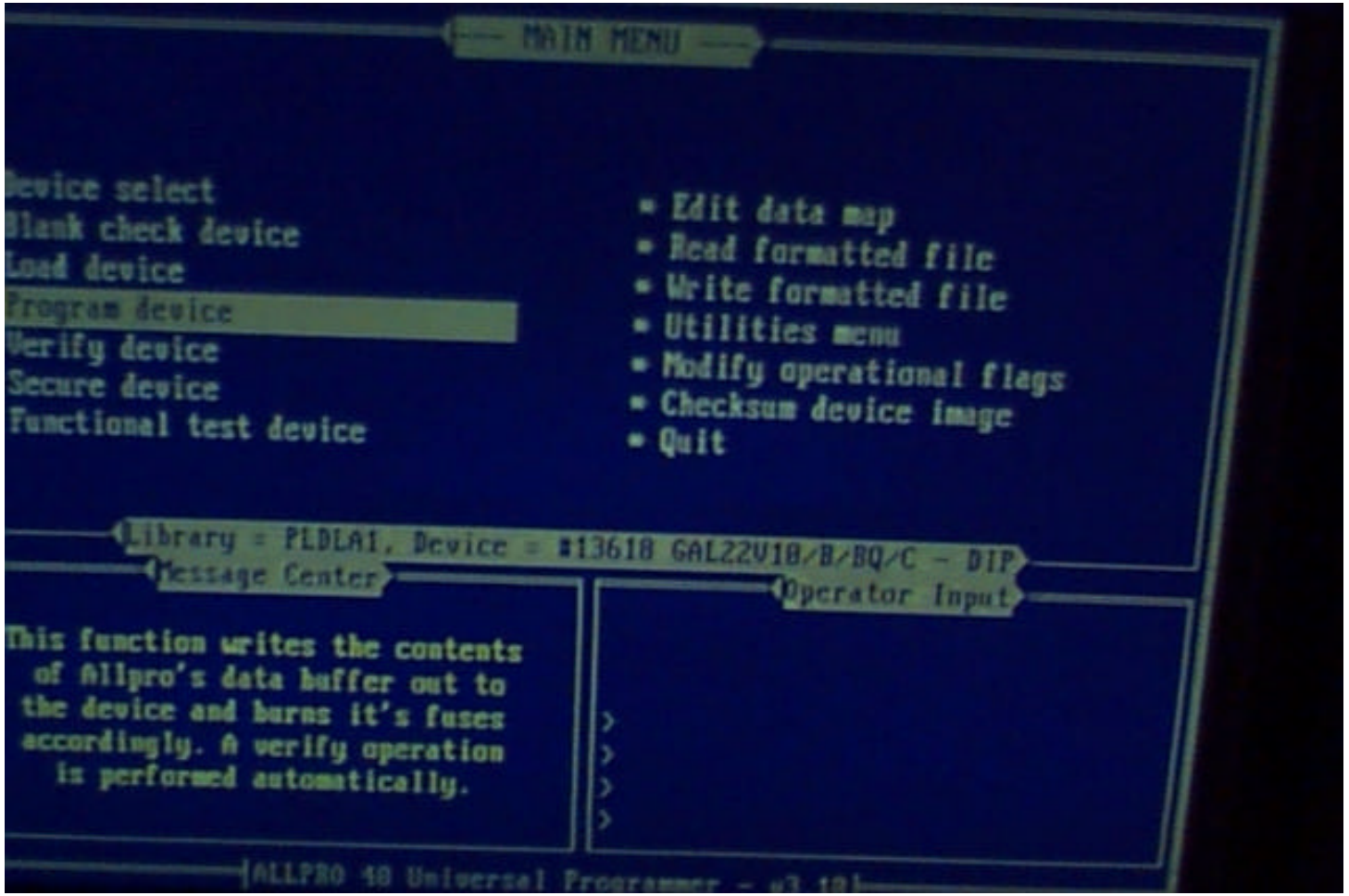


Read formatted File …

Select your .jed file from your floppy diskette



Program Device …



… and you are done.