PALASM™/PLEASM™

User's Guide
. . . the Key Tools for

IdeaLogic™

Monolithic MMI Memories

Monolithic Memories MMI

2175 Mission College Boulevard
Santa Clara, California 95050
408-970-9700
TWX 910 338 2374
TWX 910 338 2376

USER NOTES ON PALASM/PLEASM - 8/1/84

This memo is designed to clarify some issues relating to new releases of PALASM programs. Version 1.7D is the latest release of PALASM designed to support all small & medium PAL devices including the programable polarity series (16P8). It has been extensively modified from its parent V1.6C program and offers many benefits. It is 30% smaller in size and 25% faster running on benchmark circuits. The 1.7D version also corrects several bugs which have been noted:

1. Unconnected inputs & Don't care outputs were (1.7B) assigned a logic zero in the JEDEC output. V1.7C & V1.7D provide a X state & allow the programmers to ignore those pins.

2. Product terms in the 16P8 PAL device for the enable pin had all fuses blown on that pin. This has been corrected.

Recent improvements - V1.7D:

1. Printing of periods during parts of the program where little or no activity is normally visible. (This allerts the user the computer is busy thinking and not lost.) The periods are generated when the user is sending output to a disk file.

2. Acceptance of TAB (CTRL-I) and FORM-FEED (CTRL-L) characters in PALASM design files. These characters are treated as spaces & ignored in processing design file. The user may find it more convenient to use them for creating presentable listing files.

3. Acceptance of UPPER & lower case letters in menu commands in PALASM. Commands will have the same effect regardless of the caps lock key position or shift key use. (Exception - File names must be entered in the correct case for their name.)

4. CPM versions have been modified to ignore & not read comments from input design files. This makes maximum use of the limited RAM space for design file storage. Echo of design file text will produce a listing with multiple spaces squeezed down to one & comments removed. This should not affect other processing functions.

PALASM V1.8C is the latest release of the software designed to support the MegaPALs (32R16 & 64R32). It is designed to provide programming support only - no simulation & test generation. It also has the same 4 improvements listed for V1.7D. Currently it is only available for VAX & IBM-PC computers. There are no plans to make it available for Intel or CPM machines, due to memory constraints. Simulation & test generation will only be provided on PALASM2 when it becomes available. All the usual commands of the menu driven PALASM are present. No special documentation is necessary to run the V1.8C software - follow directions from the V1.7 manual.

PALASM V1.9A is an early release of 2ØRA1Ø & 2ØRP1Ø programming support. No simulation facilities are provided. The program is written in Pascal & is quite different than the earlier Fortran programs. It is written in the format of PALASM2 - a compiled language. The first thing users will notice is its speed of action. The speed comes at the expense of error messages pin-pointing syntax problem in the design file. This will be corrected in future releases. V1.9A is only available for VAX & IBM-PC computers. Future versions should be available for 8-bit machines. A short user's manual has been written for each of the programs.

Check-sum calculation functions have been removed from all versions of PALASM. This was done for two particular reasons: first - it makes the program smaller & easier to implement without overlays on 8 bit machines, second - problems were noted in how the checksum worked on some machines & files - it needed major rework. This may cause the programmers to show error messages when these files are down-loaded. (New revision firmware on DATA-IO & other programmers will accept the NULL-ØØØØ check-sum used & properly continue.) Xmit checksum (the only one previously used) is not an accurate gauge of proper file transfer & can lead to problems on some operating systems.

JED.COM is a program that specifically calculates the proper checksums (fuse & xmit varieties) & places them in the JEDEC file. Use of both checksums insure a higher confidence in the accuracy of the processing. It is used by first creating a JEDEC output file by assigning the output to the disk & then executing the JED program. It will prompt for the names of the input & output files to use. Versions of JED.COM are available for all machines we distribute PALASM for.

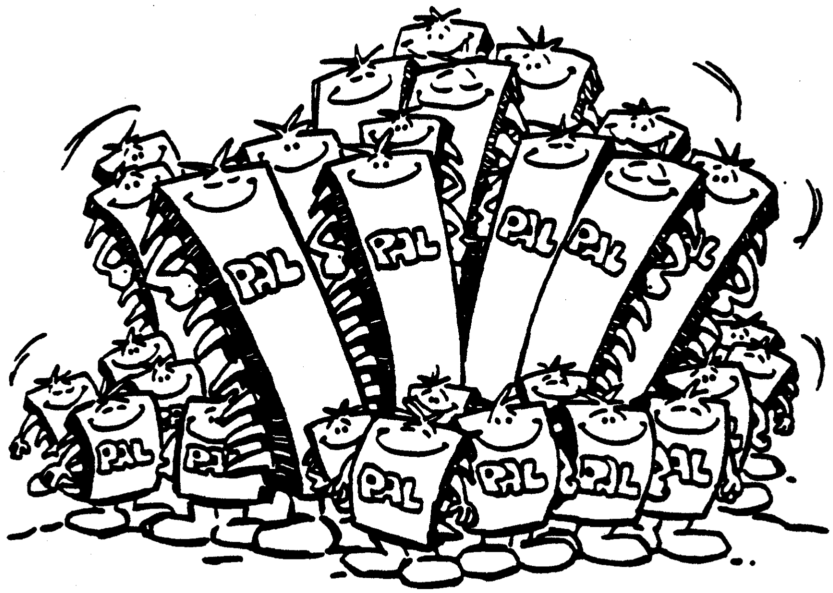
THE PALASM MANUAL FOR SOFTWARE RELEASE 1.7

Date March 15, 1984

Prepared by the IdeaLogic Exchange

Product Planning and Applications Department

Monolithic Memories, Inc.



PAL

FAMILY PORTRAIT

of 29 - count 'em!

COPYRIGHT

TRADEMARKS

The following are registered trademarks of MMI: PAL, HAL, PLE. MMI also has the trademarks: PALASM, PLEASM.

The following are registered trademarks of Digital Equipment Corporation: VAX, VMS, PDP, RSX.

IBM Corporation has the trademark: IBM PC, PC DOS.

The Osborne PC is a trademark of the Osborne Computer Corporation.

UNIX is a trademark of AT & T.

Intel/MDS is a registered trademark of Intel Corporation.

CP/M is a trademark of Digital Resarch, Inc.

MS-DOS is a trademark of Microsoft.

SSFORTRAN is a trademark of Supersoft Inc.

DISCLAIMER

Monolithic Memories Inc. makes no representations or warranties with respect to the contents within and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. Further, Monolithic Memories Inc. reserves the right to revise this publication and the product it describes and to otherwise make changes to the product without obligation of Monolithic Memories Inc. to notify any person or organization of such revision or changes.

Table of Contents

Section: Title:

1 Introduction

2 Minimum Equipment Requirements

3 The Syntax of PALASM

4 Running PALASM: Command Descriptions

APPENDIX TITLE

A PALASM Implementation Notes
- 1 VAX11/VMS
- 2 VAX11/UNIX
- 3 IBM PC
- 4 CP/M Operating System
- 5 Intel/MDS

B PAL Programmer Information

C PALASM Error Messages

D PAL Design File Library
- 1 Open Tooled Bit Patterns
- 2 Bit Pattern Cross-Reference

E Update Notes

F A PAL Primer

G Submitting a HAL Design to MMI

H Typical Questions and Answers

I Where to get help

J References

What is PALASM?

PALASM is a key tool used in automating the process of "designing your own chip". PALASM assembles and simulates PAL Design Specifications; it generates PAL fuse patterns in formats compatible with a variety of Monolithic Memories approved programmers.

PALASM provides designers a vital design tool which minimizes the "time to market" for new products by allowing designers to quickly and easily modify PAL designs.

Key Features :

- Assembles PAL Design Specifications.
- Simulates and verifies the Function Table.
- Tests each product term for stuck-at-zero (SA0) and stuck-at-one (SA1) faults.
- Generates PAL fuse patterns in various programmer formats.
- Reports assembly and Function Table errors.

Appendix A lists the computers upon which PALASM has been implemented, while Appendix B lists all MMI approved programmers.

PALASM reads from an external data file which contains the PAL Design Specification. The syntax for this file is given in Section 3.

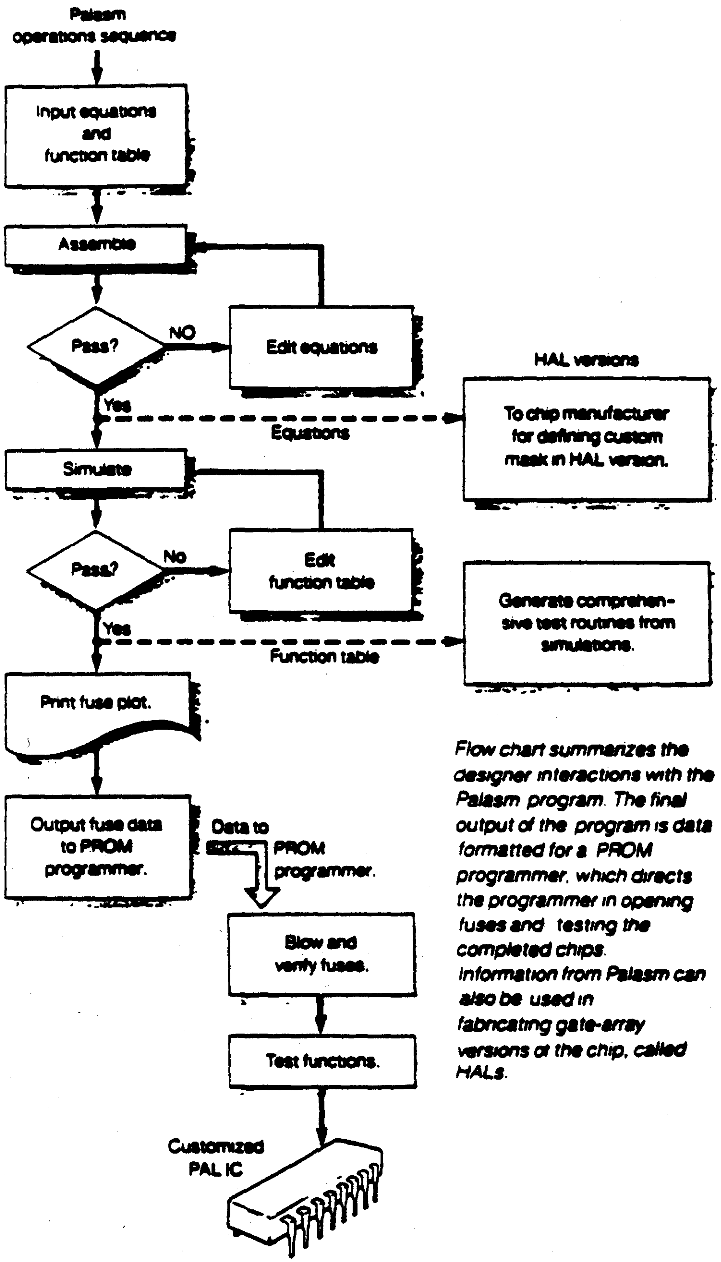
Designing with PALs and PALASM requires the following steps:

1. Specifying the design using Boolean equations in sum of products form.
2. Assembling the Boolean equations.
3. Simulating the design.
4. Generating programmer compatible outputs.

PALASM assembles the design specification and reports any syntax errors to the user. It provides the user with a list of menu driven options including Simulation, Fault Grading, generation of fuse plot patterns in multiple formats, etc.

Figure 1.1: The PALASM Flowchart.

From: J. Birkner, V. Coli, D. Sackett, Programming Logic Chips on Personal Computers, Machine Design, July 1983, p.81


Palasm operations sequence
Input equations and function table
Assemble
Pass?
NO
Edit equations
Yes
Equations
HAL versions
To chip manufacturer for defining custom mask in HAL version.
Simulate
Pass?
No
Edit function table
Yes
Function table
Generate comprehensive test routines from simulations.
Print fuse plot.
Output fuse data to PROM programmer.
Data to PROM programmer.
Blow and verify fuses.
Test functions.
Customized PAL IC

Flow chart summarizes the designer interactions with the Palasm program. The final output of the program is data formatted for a PROM programmer, which directs the programmer in opening fuses and testing the completed chips. Information from Palasm can also be used in fabricating gate-array versions of the chip, called HALs.

PALASM should run with minimal modifications on the following CPU´s provided the minimum system requirements mentioned later in this section are satisfied.

Mainframe Computers
VAXVMS, VAXUNIX, or IBM

Minicomputers
PDP11/RSX or PDP11/RT11

Microcomputers
IBMPC/MSDOS or the CPM system on the Radio Shack, Apple, Kaypro, Osborne, or Zenith computers

Other requirements are:

Removable Media: 5.25" or 8" disks, or tape.

Memory : 64K bytes minimum.

One EIA 232 port.

PAL programmers suggested: DATA I/O Model 19 Programmer with LogicPak
DATA I/O Model 29A with LogicPak
STRUCTURED DESIGN 20/24
STRUCTURED DESIGN PAL Burner
STAG PM202
STAG PM2200
DIGELEC(FAM51 or FAM52)
PROLOG M980
PROLOG 9068
KONTRON MPP80S
KONTRON MOD21
VARIX

For software development and user customization of the program, a FORTRAN compiler/linker and another disk drive are necessary to create the executable version of the program. The recommended compiler is the Supersoft FORTRAN compiler which was used to develop and test the programs on microcomputers at Monolithic Memories Inc.

PALASM is compatible with both FORTRAN IV LEVEL G and FORTRAN 77 standards. Only standard FORTRAN constructs are used to ensure portability to many computer systems.

When you receive your package, to protect your PALASM from accidental clobbering, please BACK IT UP on a second disk. The importance of this cannot be overemphasized.

To verify that the package works on your system, run each of the executable files to see if you get the correct starting prompts. Then run a few example design files to verify basic assembly.

If you are unable to execute the program or if the design examples do not assemble, the following are possible explanations for this:

1. Faulty interfacing to your operating system. The most probable cause is a non-standard implementation of the operating system on your machine.

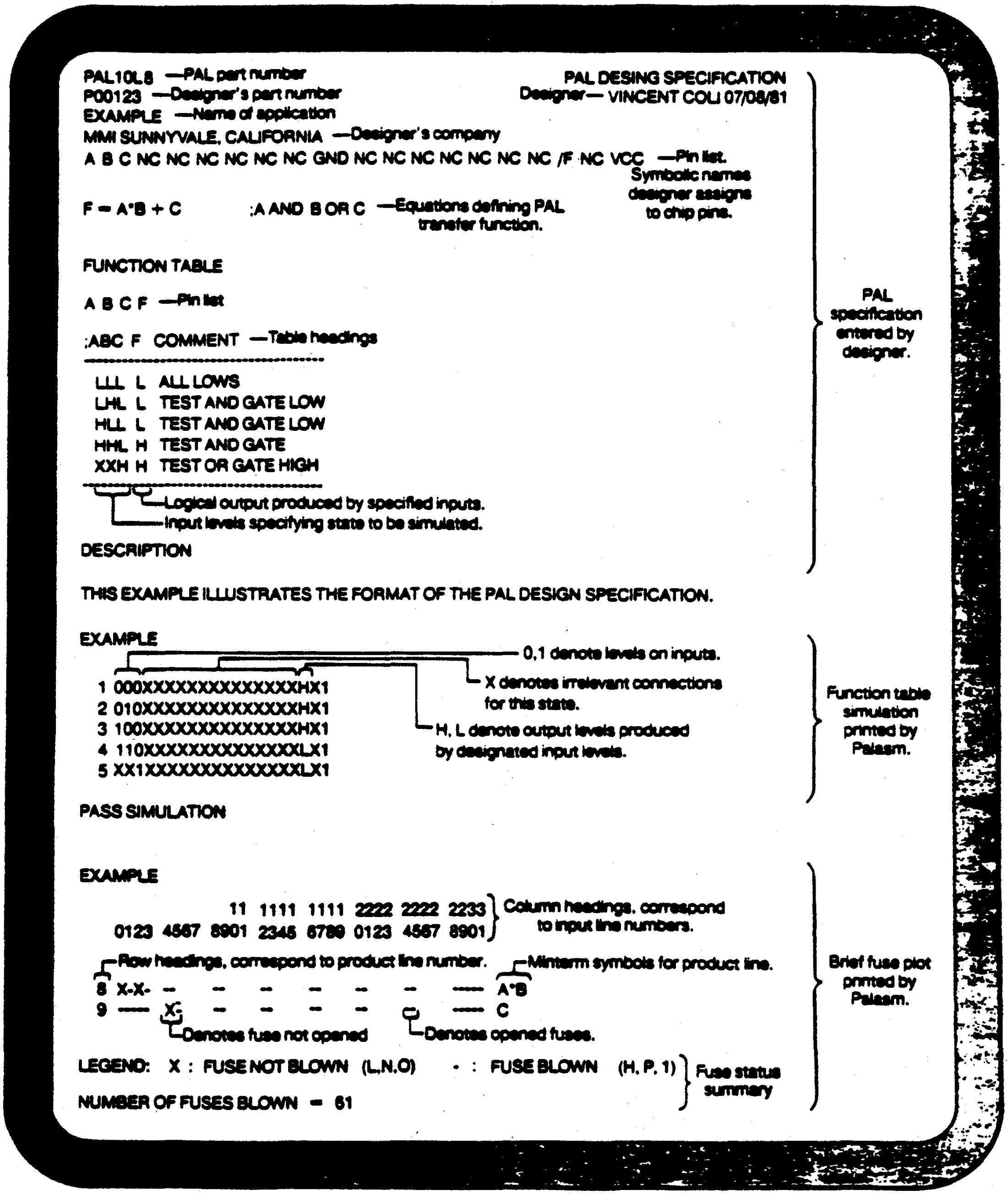
2. Inadequate memory capacity in your system. For instance, having only 64K on an IBM PC, when you need at least 128K.

3. The medium upon which you received the PALASM system has become corrupted.

4. The files may be in the wrong place, so that the operating system cannot find them.

Figure 2.1: The PALASM Syntax

From: J. Birkner, V. Coli, D. Sackett, Programming Logic Chips on Personal Computers, Machine Design, July 1983, p.84



```
PAL10L8  —PAL part number                                   PAL DESING SPECIFICATION
P00123  —Designer's part number                        Designer— VINCENT COLI 07/08/81
EXAMPLE  —Name of application
MMI SUNNYVALE, CALIFORNIA  —Designer's company
A B C NC NC NC NC NC NC GND NC NC NC NC NC NC NC /F NC VCC  —Pin list.
                                                              Symbolic names
                                                              designer assigns
F = A*B + C        ;A AND B OR C  —Equations defining PAL     to chip pins.
                                      transfer function.

FUNCTION TABLE

A B C F  —Pin list

;ABC F  COMMENT  —Table headings
------------------------------------
 LLL  L  ALL LOWS
 LHL  L  TEST AND GATE LOW
 HLL  L  TEST AND GATE LOW
 HHL  H  TEST AND GATE
 XXH  H  TEST OR GATE HIGH
------------------------------------
              —Logical output produced by specified inputs.
              —Input levels specifying state to be simulated.
DESCRIPTION
                                                                   PAL
                                                                   specification
                                                                   entered by
                                                                   designer.

THIS EXAMPLE ILLUSTRATES THE FORMAT OF THE PAL DESIGN SPECIFICATION.

EXAMPLE
                              — 0,1 denote levels on inputs.
                              — X denotes irrelevant connections
 1 000XXXXXXXXXXXXXXHX1           for this state.
 2 010XXXXXXXXXXXXXXHX1
 3 100XXXXXXXXXXXXXXHX1     — H, L denote output levels produced   Function table
 4 110XXXXXXXXXXXXXXLX1       by designated input levels.          simulation
 5 XX1XXXXXXXXXXXXXXLX1                                            printed by
                                                                   Palasm.
PASS SIMULATION


EXAMPLE

             11  1111 1111 2222 2222 2233  Column headings, correspond
  0123 4567 8901 2345 6789 0123 4567 8901    to input line numbers.
 —Row headings, correspond to product line number.  —Minterm symbols for product line.
 8 X-X- --   --   --   --   --   --   ----  A*B                 Brief fuse plot
 9 ----  X-  --   --   --   --   --   ----  C                   printed by
        —Denotes fuse not opened   —Denotes opened fuses.        Palasm.
LEGEND:  X : FUSE NOT BLOWN  (L,N,O)    - : FUSE BLOWN  (H, P, 1)  Fuse status
NUMBER OF FUSES BLOWN = 61                                          summary
```

3.0 The Syntax of PALASM

Overview

PALASM requires an input file to generate a PAL programming pattern. This section describes the syntax for a PAL design specification file.

Notational Conventions:

{ ... } indicates that whatever is inside the { } brackets is optional.

< ... > indicates that the name enclosed in the < > brackets will be discussed later.

(... denotes a comment. Comments begin with the (mark and continue till the end of the line.

One limitation that varies from one system to another is the acceptable maximum length of a PALASM Specification file.

PALASM Syntax

```
<pal type line>
<pal pattern line>
<application description line>
<company address line>
<pin list>
<equation list>
{       ( the following is optional:
FUNCTION TABLE          ( this must start in column 1 and have 1 space betwee
                        ( FUNCTION and TABLE
<function table pin list>
<function table list>
}
{       ( the following is optional if following FUNCTION TABLE.
DESCRIPTION             ( this must start in column 1
Descriptive commentary lines follow till end of file.
}
```

Note: a PALASM specification must have either a FUNCTION TABLE line or a DESCRIPTION line to terminate the PALASM <equation list>. PALASM will fail otherwise.

A Closer Look at the PALASM Syntax

<pal type line>:
Starting in line 1, column 1, this specifies the PAL type. Any text following is treated as a comment.

<pal pattern line>:
Documents the specified PAL pattern for future reference.

<application description line> and <company address line>:
These lines are without formal rules, they are provided for the convenience of our customers.

<pin list>:
For PALASM to work successfully, all device pins must be named in this list. These names do not have to start in column 1 however the list must be in numerical pin order and MUST begin on line 5. Pin names may be up to eight characters long and must be separated by at least one blank. PALs are available with either active high or active low outputs. To specify an active low output. To use active low PALs, use opposite polarities for that pin name in the pin list and in the left hand side of the PAL output equations.

Comments may be included in the lines of the pin list, equations and function table as follows:

<comments>:
A comment begins with a semicolon ";" and continues to the end of the current line. This same convention is used for commentary in the following parts of a PALASM specification:

```
                <pin list>
                <equation list>
                <function table pin list>
                <function table list>
```

<equation list>:
This is a sequence of <pal output equations>. The <pal output equation> will have one of the following forms:

```
                <SYMBOL>=<EXPRESSION>
                IF(<PRODUCT>) <SYMBOL>=<EXPRESSION>
                <SYMBOL>:=<EXPRESSION>
```

NOTE: Equations are not limited to just one line!!! To continue an equation, go to the next line, just as you would in PASCAL and similar languages. AVOID writing in column 80, as PALASM continues to scan in the next line in column 1, and this can get you into trouble.

<SYMBOL>:
This is a pin name. If preceeded by a slash "/", the state of the pin name is logically complemented.

IF( ):

This tells PALASM that the output pin within the parentheses is to be an output only when the PRODUCT is true. Otherwise, the output will be put in high impedance mode (high-Z).

<PRODUCT>:

This is a sequence of SYMBOLs separated by the AND operator "*".

<EXPRESSION>:

A sequence of SYMBOLS separated by the following:

*	AND	(product)
+	OR	(logical sum)
:+:	XOR	(exclusive OR)
:*:	XNOR	(exclusive NOR)

Discussion of <pal output equations>:

Each of the three equation forms have different functions.

The first form indicates that the expression directly determines the state of the output.

The second says that when the product specified within the () is true the pin will output the expression.

The third form says that the pin changes state to equal the expression only on the rising edge of the clock.

<function table list>:

The function table pin list specifies the sequence of entries in the function table. The function table itself begins with a dashed line and terminates with a dashed line. The length of the dashed lines is not significant. Each line or vector in the function table records the input and output pins of the function table pin list. There must be as many states per vector as there are pin names in the function table pin list. Any line in the function table may consist of a vector or a comment or both.

An input may have one of 4 states:

L Force a low on the input pin.
H Force a high on the input pin.
X Although specified as a don't care, PALASM defaults to a low signal onthispin.
C This input pin is being used as a clock, drive the logic signal from low to high state.

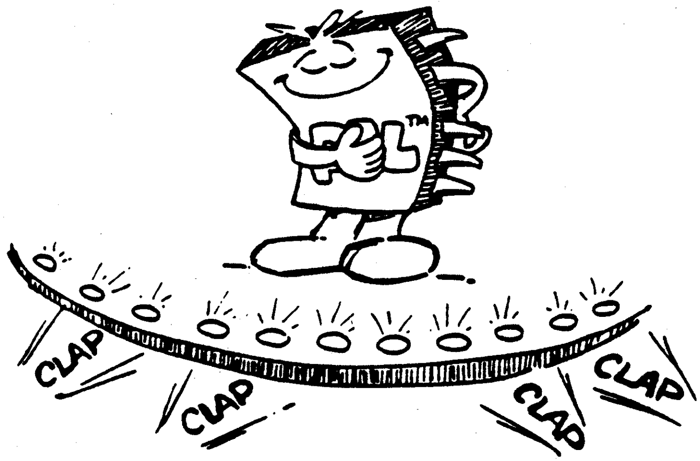
NOTE: PALASM will report inputs as 1 for H and 0 for L in its generated test vector list.

An output may have one of 4 states:

L Expect a logic 0 on this pin.
H Expect a logic 1 on this pin.
X Do not test this output.
Z Test for high impedance on this pin.

A brief example of the PAL input spec follows:

```
PAL12H6                                               PAL DESIGN SPECIFICATION
P7000                                                  JANE ENGINEER 03/12/82
SIMPLE EXAMPLE
MMI SANTA CLARA, CALIFORNIA
A B C NC NC NC NC NC NC GND NC NC F NC NC NC NC NC NC VCC


F = A + B*C           ;SIMPLE TRANSFER FUNCTION


FUNCTION TABLE

  A     B     C        F
;                  |
;     INPUTS       |  OUTPUT
;#1    #2    #3    |
-----------------------
  L     L     L        L
  L     L     H        L
  L     H     L        L
  L     H     H        H
  H     L     L        H
  H     L     H        H
  H     H     L        H
  H     H     H        H
-----------------------

DESCRIPTION

THIS EXAMPLE ILLUSTRATES THE USE OF FUSIBLE LOGIC TO IMPLEMENT A
SIMPLE BOOLEAN TRANSFER FUNCTION.

PALASM EXERCISES THE FUNCTION TABLE TO SIMULATE THE TRANSFER
FUNCTION.
```



Introducing: Programmable Polarity PALs

1. The architecture for these PALs are shown on the pages that follow. With the polarity fuses intact, these PALs are active low; when blown, the outputs are active high.

For example, if output 1, OUT1, is specified as uncomplemented in the pin list, and in the equations, it is also specified in the uncomplemented form, the polarity fuse WILL be blown because the output is specified as active high.

The converse is also true. If OUT1 is complemented in the pin list and also complemented in the equation, the fuse will also be blown.

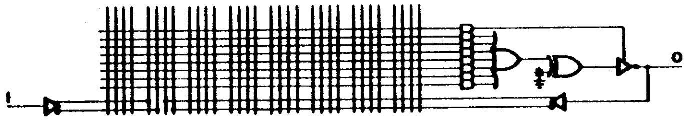
2. To specify a register preload during simulation, enter a "P" for the preload pin in the function table. The registers will then be preloaded with the pin information supplied with that P-vector.

AN ENHANCED VERSION OF 20 AND 24 PIN PAL CIRCUITS THAT PROVIDE "P" FEATURES

- PROGRAMMABLE POLARITY
- REGISTER PRELOAD
- COMPLETELY TESTABLE

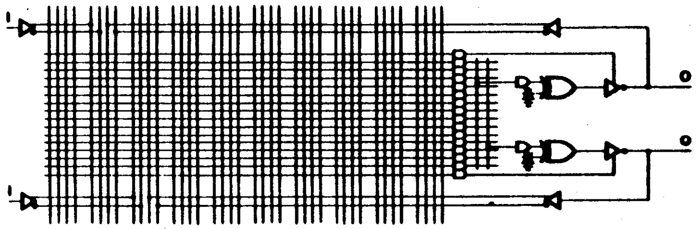
Monolithic Memories

16P8

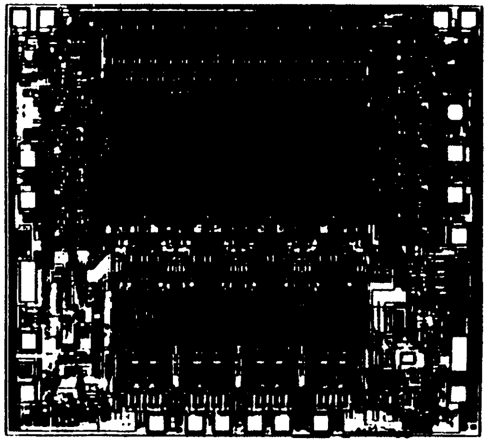


Monolithic Memories

20P10



Monolithic Memories



Monolithic Memories

4.0 Running PALASM

To start, check Appendix A to verify that you have all the necessary files to run PALASM on your particular computer and operating system. To gain some insight into the nature of PALASM, we suggest trying to run one of the several examples that are included in your package. P7000.DAT is the classic basic gates example we suggest you try first. Once the basic capabilities of the program are understood and once you have established a certain degree of comfort, work with some of the other examples or create your own designs.

Type in your system's execute command to run PALASM, you should see the following response on the screen, if PAL20 was chosen:

```
MONOLITHIC MEMORIES 20-PIN PALASM (tm) VERSION 1.7A
 (C) COPYRIGHT 1983, 1984 MONOLITHIC MEMORIES

WHAT IS THE SOURCE FILENAME (d:filename.ext) ?: P7030.DAT
```

Enter the name of the input file containing the PAL design Specification in response to the above prompt. In the above example, the P7030.DAT example has been chosen.

```
OUTPUT FILENAME - PRESS <ENTER> FOR NO OUTPUT FILE ?: <CR>
```

If <CR> is pressed, the output from this PALASM run will be sent to the console. Otherwise, if a filename is specified, all output will be channelled to that file.

After reading the input specification file, PALASM returns with a menu of the following form:

```
E=ECHO INPUT    O=PINOUT  S=SIMULATE   P=PLOT
B=BRIEF         H=HEX     I=INTEL HEX  D=DOCUMENT
J=JEDEC FORMAT  F=FAULT   TESTING
C=CATALOG       Q=QUIT

ENTER OPERATION CODE: _
```

The _ mark designates the cursor. Any selection you enter must be in UPPER CASE!!! A brief discussion and sample ouput from each of the options follows:

****

```
ENTER OPERATION CODE: D
```

Documents the entire PAL design by executing the echo, pinout and plot options on the menu. Sample output from running each of these options follow.

****

ENTER OPERATION CODE: E

Echos the PAL Design Specification sending it to the display device and helping verify that the input file has been read properly.

```
PAL16C1
P7030 (PMSI003)                                   BIRKNER/COLI 07/21/81
OCTAL COMPARATOR
MMI SUNNYVALE, CALIFORNIA
A7 A0 B0 A1 B1 A2 B2 A3 B3 GND
A4 B4 A5 B5 EQ NE A6 B6 B7 VCC


NE =  A0*/B0  +  /A0* B0     ;A0 :+: B0
   +  A1*/B1  +  /A1* B1     ;A1 :+: B1
   +  A2*/B2  +  /A2* B2     ;A2 :+: B2
   +  A3*/B3  +  /A3* B3     ;A3 :+: B3
   +  A4*/B4  +  /A4* B4     ;A4 :+: B4
   +  A5*/B5  +  /A5* B5     ;A5 :+: B5
   +  A6*/B6  +  /A6* B6     ;A6 :+: B6
   +  A7*/B7  +  /A7* B7     ;A7 :+: B7

FUNCTION TABLE

A7 A6 A5 A4 A3 A2 A1 A0 B7 B6 B5 B4 B3 B2 B1 B0 NE EQ

;INPUT  A  INPUT  B  OUTPUTS
;76543210  76543210   NE EQ    COMMENTS
------------------------------------------------------------
 HLLLLLLL  LLLLLLLL   H   L    A7=H, B7=L
 LHLLLLLL  LLLLLLLL   H   L    A6=H, B6=L
 LLHLLLLL  LLLLLLLL   H   L    A5=H, B5=L
    .
    .
    .
 LHLHLHLH  LHLHLHLH   L   H    TEST ODD  CHECKERBOARD
 HHLLHHLL  HHLLHHLL   L   H    TEST EVEN DOUBLE CHECKERBOARD
 LLHHLLHH  LLHHLLHH   L   H    TEST ODD  DOUBLE CHECKERBOARD
------------------------------------------------------------


DESCRIPTION

THE OCTAL COMPARATOR ESTABLISHES WHEN TWO 8-BIT DATA STRINGS
(A7-A0 AND B7-B0) ARE EQUIVALENT (EQ=H) OR NOT EQUIVALENT (NE=H).

*******************************************************************************
```

ENTER OPERATION CODE: O

Upon commanding a pinout, PALASM displays the location and names of all pins on the specified PAL.

OCTAL COMPARATOR

```
                  ***************  ***************
                  *             * *              *
               ****                              ****
          A7   * 1*            P A L             *20*  VCC
               ****                              ****
                  *           1 6 C 1            *
               ****                              ****
          A0   * 2*                              *19*  B7
               ****                              ****
                  *                              *
               ****                              ****
          B0   * 3*                              *18*  B6
               ****                              ****
                  *                              *
               ****                              ****
          A1   * 4*                              *17*  A6
               ****                              ****
                  *                              *
               ****                              ****
          B1   * 5*                              *16*  NE
               ****                              ****
                  *                              *
               ****                              ****
          A2   * 6*                              *15*  EQ
               ****                              ****
                  *                              *
               ****                              ****
          B2   * 7*                              *14*  B5
               ****                              ****
                  *                              *
               ****                              ****
          A3   * 8*                              *13*  A5
               ****                              ****
                  *                              *
               ****                              ****
          B3   * 9*                              *12*  B4
               ****                              ****
                  *                              *
               ****                              ****
         GND   *10*                              *11*  A4
               ****                              ****
                  *                              *
                  ********************************

******************************************************************************
```

ENTER OPERATION CODE: S

In executing this command, PALASM examines the Function Table entries and determines if the PAL performs as the entries indicate. These entries are checked against the equations thereby verifying the design. Test engineering can then use the resulting simulation vectors as seed vectors in generating functional test vectors.

All inputs in the logic equations are given the values specified in the Function Table vector and the equations are exercised. The evaluated outputs and those specified in the test vectors are checked and any inconsistencies between the two values are flagged as errors.

In evaluating the logic equations, three types of inputs are present: primary inputs, feedback inputs from registered outputs, and feedback inputs from combinatorial outputs. The values for primary inputs and the combinatorial feedback outputs are taken from the present vector under consideration. The values for registered feedback inputs are taken from the registered outputs of the previous vector. It is therefore essential that the first vector in the Function Table put the PAL state machine in a valid starting state and not "wake" the machine in an undefined state.

The Function Table allows the following as valid input states.

H High level
L Low level
C Transition from low to high
X Irrelevant (defaults to logic 0 for input)
Z High impedance

The following test vector states are output:

H Test for high state
L Test for low state
1 Drive to high state
0 Drive to low state
X Irrelevant (do not test if output)
C Drive input from low to high state

An important point to note is that all "don't care" conditions are treated as LOW signals. This option can be invoked only when a function table exists in the input specifications.

```
OCTAL COMPARATOR

    1 100000000X0000LH0001
    2 000000000X0000LH1001
    3 000000000X0010LH0001
        .
        .
        .
   20 011001100X1100HL1101
   21 100001111X0000HL1111
   22 011110000X1111HL0001

PASS SIMULATION

********************************************************************************

ENTER OPERATION CODE: P
```

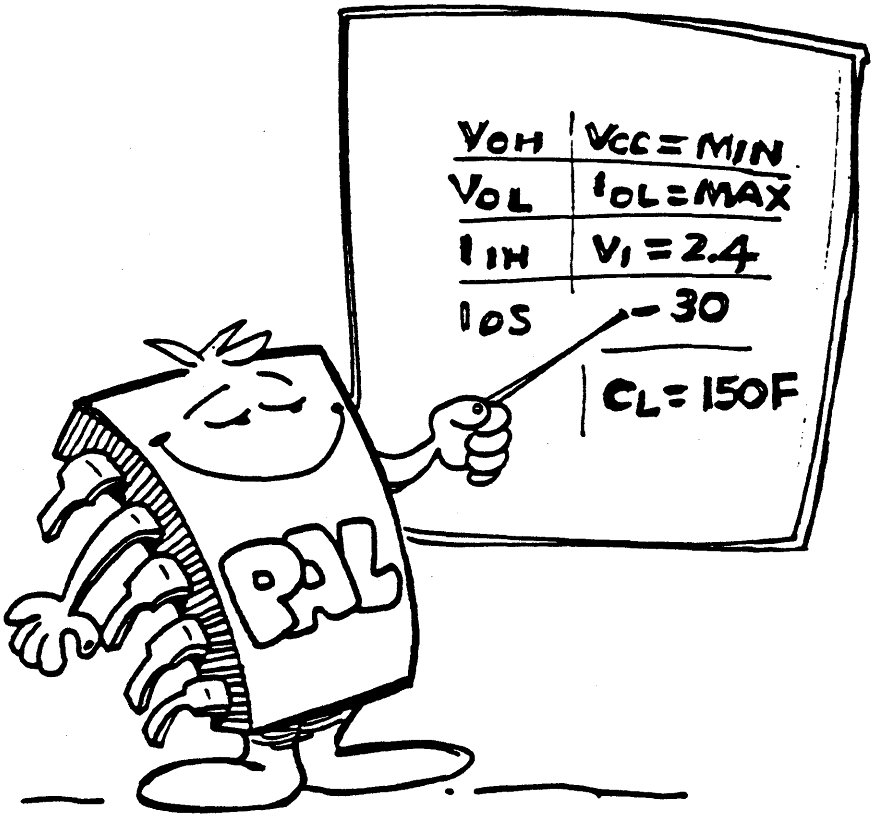
Displays complete fuse plot.

```
OCTAL COMPARATOR

                11 1111 1111 2222 2222 2233
   0123 4567 8901 2345 6789 0123 4567 8901

 0 0000 0000 0000 0000 0000 0000 0000 0000
 1 0000 0000 0000 0000 0000 0000 0000 0000
 2 0000 0000 0000 0000 0000 0000 0000 0000
      .
      .
      .
21 0000 0000 0000 0000 0000 0000 0000 0000
22 0000 0000 0000 0000 0000 0000 0000 0000
23 0000 0000 0000 0000 0000 0000 0000 0000

24 X--- -X-- ---- ---- ---- ---- ---- ---- A0*/B0
25 -X-- X--- ---- ---- ---- ---- ---- ---- /A0*B0
26 ---- ---- X--- -X-- ---- ---- ---- ---- A1*/B1
27 ---- ---- -X-- X--- ---- ---- ---- ---- /A1*B1
28 ---- ---- ---- ---- X--- -X-- ---- ---- A2*/B2
29 ---- ---- ---- ---- -X-- X--- ---- ---- /A2*B2
30 ---- ---- ---- ---- ---- ---- X--- -X-- A3*/B3
31 ---- ---- ---- ---- ---- ---- -X-- X--- /A3*B3

32 ---- ---- ---- ---- ---- ---- ---X --X- A4*/B4
33 ---- ---- ---- ---- ---- ---- --X- ---X /A4*B4
34 ---- ---- ---- ---- ---X --X- ---- ---- A5*/B5
35 ---- ---- ---- ---- --X- ---X ---- ---- /A5*B5
36 ---- ---- ---X --X- ---- ---- ---- ---- A6*/B6
37 ---- ---- --X- ---X ---- ---- ---- ---- /A6*B6
38 --X- ---X ---- ---- ---- ---- ---- ---- A7*/B7
39 ---X --X- ---- ---- ---- ---- ---- ---- /A7*B7
```

```
40 0000 0000 0000 0000 0000 0000 0000 0000
41 0000 0000 0000 0000 0000 0000 0000 0000
42 0000 0000 0000 0000 0000 0000 0000 0000
     .
     .
     .
61 0000 0000 0000 0000 0000 0000 0000 0000
62 0000 0000 0000 0000 0000 0000 0000 0000
63 0000 0000 0000 0000 0000 0000 0000 0000

LEGEND:  X : FUSE NOT BLOWN (L,N,0)    - : FUSE BLOWN    (H,P,1)
         0 : PHANTOM FUSE   (L,N,0)    O : PHANTOM FUSE  (H,P,1)

NUMBER OF FUSES BLOWN =  480

***************************************************************************************

ENTER OPERATION CODE: B
```

Displays only product terms containing blown fuses.

```
OCTAL COMPARATOR

               11 1111 1111 2222 2222 2233
   0123 4567 8901 2345 6789 0123 4567 8901


24 X--- -X-- ---- ---- ---- ---- ---- ---- A0*/B0
25 -X-- X--- ---- ---- ---- ---- ---- ---- /A0*B0
26 ---- ---- X--- -X-- ---- ---- ---- ---- A1*/B1
27 ---- ---- -X-- X--- ---- ---- ---- ---- /A1*B1
28 ---- ---- ---- ---- X--- -X-- ---- ---- A2*/B2
29 ---- ---- ---- ---- -X-- X--- ---- ---- /A2*B2
30 ---- ---- ---- ---- ---- ---- X--- -X-- A3*/B3
31 ---- ---- ---- ---- ---- ---- -X-- X--- /A3*B3

32 ---- ---- ---- ---- ---- ---- ---X --X- A4*/B4
33 ---- ---- ---- ---- ---- ---- --X- ---X /A4*B4
34 ---- ---- ---- ---- ---X --X- ---- ---- A5*/B5
35 ---- ---- ---- ---- --X- ---X ---- ---- /A5*B5
36 ---- ---- ---X --X- ---- ---- ---- ---- A6*/B6
37 ---- ---- --X- ---X ---- ---- ---- ---- /A6*B6
38 --X- ---X ---- ---- ---- ---- ---- ---- A7*/B7
39 ---X --X- ---- ---- ---- ---- ---- ---- /A7*B7

LEGEND:  X : FUSE NOT BLOWN (L,N,0)    - : FUSE BLOWN    (H,P,1)

NUMBER OF FUSES BLOWN =  480

SECURITY FUSE = XX

***********************************************************************************
```


VOH VCC = MIN
VOL IOL = MAX
IIH VI = 2.4
IOS -30
CL = 150F
PAL

ENTER OPERATION CODE: H

This format which is directly compatible with the DATA I/O 909-1427 20-pin Generic card set, is shown below.

```
A
BB BB BB BB BB BB BB BB BB
C

A   --> Record start character (STX)
BB  --> Data byte
c   --> End of text character (ETX)
```

Data is sent in streams of 16 8-bit bytes separated by spaces. An execute character, the ASCII period ".", is sent at the end of each stream of data. 4-bit data is padded up with zeros in the most significant 4 places. In addition, a hex checksum is passed at the end of the transmission.

```
7 F F F F 7 F F F F F F F F F F F F F F F F F F F F F F F F F F .
F 7 F F 7 F F F F F F F F F F F F F F F F F F F F F F F F F F F .
F F F F F F F F 7 F F F F 7 F F F F F F F F F F F F F F F F F F .
    .
    .
    .
1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 .
1 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 .
1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 .
```

HEX CHECK SUM = F70

****

ENTER OPERATION CODE: I

Generates the Intel Hex format for both 4- and 8-bit data downloading; the format is as follows:

```
:AABBBB00CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCDD

:    --> Starting colon marker
AA   --> Record length in hex
BBBB --> Record starting address in hex
C..C --> Data
DD   --> Hex checksum
```

All data is sent in streams of 16 8-bit bytes starting at 000H. 4-bit data is padded with zeros in the most significant 4 bits. The checksum is the negative of the sum of all 8-bit bytes starting at "AA" to "DD", modulo 256. Transmission is terminated by the bit stream ":00000001FF".

```
:20000000070F0F0F070F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F10
:200020000F070F0F070F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0FF0
:200040000F0F0F0F0F0F0F0F070F0F0F0F070F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0FD0
      .
      .
      .
:2001A0000101010101010100010101010001010101010101010101010101010101010121
:2001C0000101000101010100010101010101010101010101010101010101010101010101
:2001E00001010100010100010101010101010101010101010101010101010101010101E1
:00000001FF

*******************************************************************************

ENTER OPERATION CODE: J

    Provides JEDEC formatted fuse plot.

*D2221*F0*
L0000 01111011111111111111111111111111*
L0032 10110111111111111111111111111111*
L0064 11111111011110111111111111111111*
     .
     .
     .
L0416 11111111110111101111111111111111*
L0448 11011110111111111111111111111111*
L0480 11101101111111111111111111111111*
V0001 100000000X0000LH0001 *
V0002 000000000X0000LH1001 *
V0003 000000000X0010LH0001 *
     .
     .
     .
V0020 011001100X1100HL1101 *
V0021 100001111X0000HL1111 *
V0022 011110000X1111HL0001 *
F998

*******************************************************************************
```

```
ENTER OPERATION CODE: F
```

Assuming product lines can fail by being stuck-at-1 or stuck-at-0, the fault testing procedure determines those failures that the (function table entries can detect and identifies each product term (that cannot be detected.

Product term coverage is computed as follows:

- Total number of detected SA1 faults
- Total number of detected SA0 faults
- Product Term Coverage

$$\text{PTC} = \frac{\text{SA1 faults} + \text{SA0 faults}}{2 * \text{Total number of Product Terms}} * 100(\%)$$

PALASM notifies the user which product terms and what type of faults were not detected. For example,

" PRODUCT 2 OF EQUATION 1 UNTESTED (SA0) FAULT "

This means that the product term number 2 of equation 1 cannot be tested for (SA0) fault.

```
OCTAL COMPARATOR

   1 100000000X0000LH0001
   2 000000000X0000LH1001
   3 000000000X0010LH0001
     .
     .
     .
  20 011001100X1100HL1101
  21 100001111X0000HL1111
  22 011110000X1111HL0001

PASS SIMULATION

NUMBER OF STUCK AT ONE (SA1)  FAULTS ARE = 16

NUMBER OF STUCK AT ZERO (SA0) FAULTS ARE = 16

PRODUCT  TERM   COVERAGE                 =100%

*************************************************************************************
```

```
ENTER OPERATION CODE: C
```

Catalog of available options.

```
 MONOLITHIC MEMORIES 20-PIN PALASM VERSION 1.7
(C) COPYRIGHT 1983 MONOLITHIC MEMORIES

   DOCUMENT(D)    - PRINTS THE PAL DESIGN DOCUMENTATION
   ECHO (E)       - PRINTS THE PAL DESIGN SPECIFICATION
   PINOUT (O)     - PRINTS THE PINOUT OF THE PAL
   SIMULATE (S)   - EXERCISES THE FUNCTION TABLE VECTORS IN THE LOGIC
                    EQUATIONS AND GENERATES TEST VECTORS
   PLOT (P)       - PRINTS THE ENTIRE FUSE PLOT
   BRIEF (B)      - PRINTS ONLY THE USED PRODUCT LINES OF THE FUSE PLOT
                    PHANTOM FUSES ARE OMITTED
   HEX (H)        - GENERATES HEX PROGRAMMING FORMAT
   INTEL (I)      - INTEL HEX PROGRAMMING FORMAT
   JEDEC (J)      - JEDEC FORMAT FOR DATA I/O PROGRAMMER
   FAULT (F)      - FAULT TESTING
   CATALOG (C)    - PRINTS THE PALASM CATALOG
   QUIT (Q)       - EXIT PALASM
*******************************************************************************
```

Direct Logic Replacement

The following files should be on the tape if you ordered the load/go version.

PAL20.EXE and PAL24.EXE - Executable file for 20- and 24-pin PALs respectively.

P7000.DAT through P7099.DAT - Example PAL design specification

files. These are useful for demonstrating how the input file should be formatted or for verifying that the assembler is indeed working correctly.

BITPN.TXT - A directory of the example design specification files. This cross-reference is repeated in Appendix D.

IOINIT.FOR - User customization package for array dimensions and I/O.

If you ordered the development version, you will have the following additional files on your tape.

PAL20.FOR and PAL24.FOR - FORTRAN source files for the 20- and 24-pin modules. The programs were developed using FORTRAN77 however you should be able to compile under FORTRAN IV with minor changes.

IOLIB.FOR - This provides the I/O features that make this version of PALASM compatible with that on the IBM PC. This module must be linked in with the main program.

Unloading Your Mag Tape under VAX/VMS

Helpful Hints:

- The volume is labelled PALASM and is recorded in Files-11 format, 1600 BPI and 9-track mag tape.
- For the neophytes who wish to learn everything there is to learn about mag tapes and more, Digital Equipment Corporation has published The Magnetic Tape Users´ Guide (Order No. AA-M539A-TE).
- Your tape drive goes by many different names. For example, MTA0: or MSA0:. To list all devices on your installation, type SH DEV M<cr> when you see the $ prompt.

After loading your tape on the drive, when you see the $ prompt, type the following:

```
$ ALL MTA0: TAPE
```

Allocates space for TAPE on device MTA0:

```
$MOUNT/OVER=IDENT TAPE
```

This mounts the tape and overrides any tape labels. Operator privileges are required to do this.

```
$CREATE/DIR [.PALASM]
$SET/DEFAULT [.PALASM]
```

Sets up the appropriate default directory.

```
$COPY TAPE:*.*;* []*
```

This copies the tape files to your directory.

```
$DISMOUNT TAPE
```

Dismounts tape and wraps things up.

DISREGARD THE SECTION BELOW IF YOU HAVE THE LOAD/GO VERSION
****

Compile and link the source program using the following sequence of commands to create the executable version. You may substitute PAL24 for PAL20 as appropriate.

```
$ FORTRAN PAL20,IOLIB
$ LINK PAL20,IOLIB
```

****

After creating your PAL design Specification files using one of your system editors, type the following to run PALASM:

```
$ RUN PAL20
```

At this point, refer to section 4 for step-by-step instructions on how to use the program.

Dumping a File From the VAX to the Data I/O:

Cable Connections:

The RS-232C cable that connects the VAX-11 to the Data I/O has lines 2 and 3 reversed. The only other pins that must be connected are pins 1 and 7.

Operating Procedures:

1. Turn Data I/O power off.
2. Connect the Data I/O programmer to the modem and VT100 terminal as shown in Fig. A.1-1.
3. Turn the Data I/O programmer on.
4. Press the "SELECT" (Data I/O).
5. Enter "EB"(Data I/O)
6. Press the "START"(Data I/O).
7. Type "TY FILENAME.DAT" (VT100). The file should be in the JEDEC format.
8. Press RETURN on the VT100.
9. Disconnect VT-100 terminal from the modem and Data I/O.
10. Reconnect Data I/O to VT-100 as shown in Fig. A.1-2.
11. Press the "SELECT" (Data I/O).
12. Press the "SELECT" (Data I/O).
13. Enter "E1" (Data I/O).
14. Press "START" (Data I/O).
15. Use VT-100 keyboard in order to communicate with the Data I/O.

```
                   PROGRAMMER            MODEM              VT100

                   !-------!           !-------!           !-------!
PROTECTIVE GND.    ! 1   0 !-----------! 0 1 0 !-----------! 0   1 ! PROTECTIVE GND.
                   !-------!           !-------!           !-------!
SEND DATA          ! 2   0 !-----------! 0 2 0 !-----------! 0   2 ! SEND DATA
                   !-------!           !-------!           !-------!
RECEIVE DATA       ! 3   0 !-----------! 0 3 0 !-----------! 0   3 ! RECEIVE DATA
                   !-------!           !-------!           !-------!
RTS                ! 4   0 !-----------! 0 4 0 !-----------! 0   4 !
                   !-------!           !-------!           !-------!
CTS                ! 5   0 !-----------! 0 5 0 !-----------! 0   5 !
                   !-------!           !-------!           !-------!
DSR                ! 6   0 !-----------! 0 6 0 !-----------! 0   6 !
                   !-------!           !-------!           !-------!
SIGNAL GND.        ! 7   0 !-----------! 0 7 0 !-----------! 0   7 ! SIGNAL GND.
                   !-------!           !-------!           !-------!
```

Fig. A.1-1: Downloading from host (VAX-11) to Programmer.
(VAX talking to Programmer and VT100).

```
                    PROGRAMMER              VT100
                    !-------!            !-------!
PROTECTIVE GND.     ! 1   0 !------------! 0   1 ! PROTECTIVE GND.
                    !-------!            !-------!
SEND DATA           ! 2   0 !------------! 0   2 ! SEND DATA
                    !-------!            !-------!
RECEIVE DATA        ! 3   0 !------------! 0   3 ! RECEIVE DATA
                    !-------!            !-------!
RTS                 ! 4   0 !------------! 0   4 !
                    !-------!            !-------!
CTS                 ! 5   0 !------------! 0   5 !
                    !-------!            !-------!
DSR                 ! 6   0 !------------! 0   6 !
                    !-------!            !-------!
SIGNAL GND.         ! 7   0 !------------! 0   7 ! SIGNAL GND.
                    !-------!            !-------!
```

Fig. A.1-2: Using Programmer as host.

If you ordered the load/go version, the following files should be on your TAR tape.

pals.mmi - A preprocessor that identifies the input file as a 20- or 24-pin PAL and invokes pal20 or pal24 as is necessary.

pal20 and pal24 - Executable file for 20- and 24-pin PALs respectively.

p7000.dat through p7099.dat - Example PAL design specification files. These are useful for demonstrating how the input file should be formatted or for verifying that the assembler is indeed working correctly.

bitpn.txt - A directory of the example design specification files. This cross-reference is repeated in Appendix D.

ioinit - User customization package for array dimensions and I/O.

If you ordered the development version, you will have the following additional files on your tape.

pal20.f and pal24.f - FORTRAN source files for the 20- and 24-pin modules. The programs were developed using FORTRAN77 .

iolib.f - This provides the I/O features that make this version of PALASM compatible with that on the IBM PC. This module must be linked in with the main program.

DISREGARD THE SECTION BELOW IF YOU HAVE THE LOAD/GO VERSION
****

Compile and link the source program using the following sequence of commands to create the executable version. You may substitute pal24 for pal20 as appropriate.

```
% f77 -w pal20.f -o pal20
% f77 -w iolib.f -o iolib
% link pal20,iolib
```

****

Create your PAL design Specification files using one of your system editors, then type the following to run PALASM:

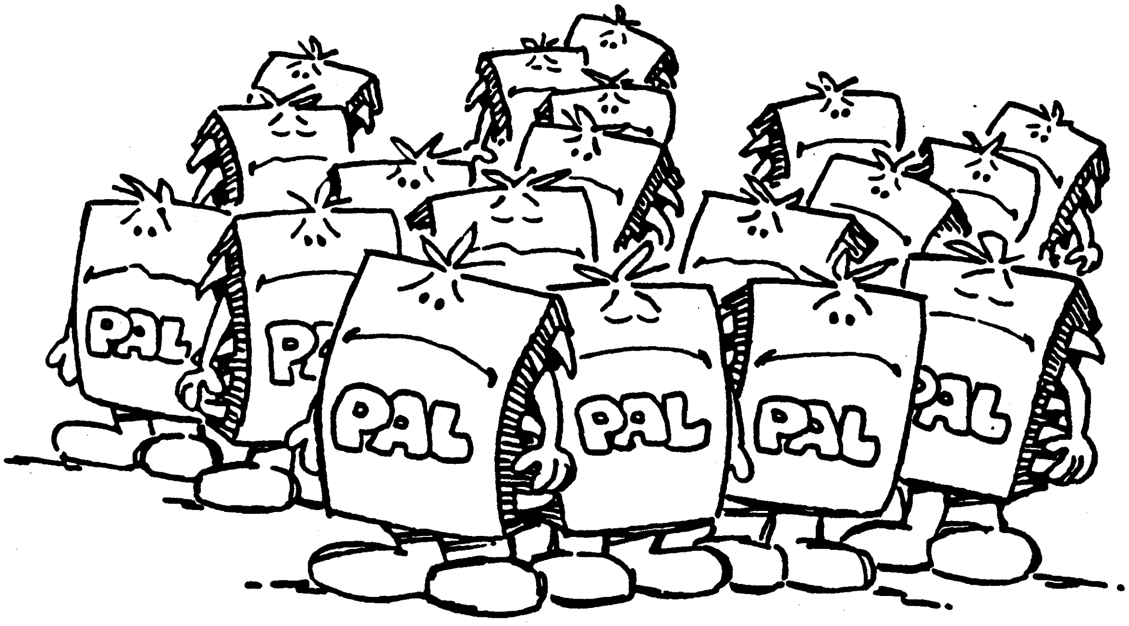
% pals.mmi

At this point, refer to section 4 for step-by-step instructions on how to use the program. The program pals.mmi

invokes pal20 or pal24 depending on which pal type is specified in line 1 of the input file.

An important point to note is that the input file should be available in the data file "pal.in". The appropriate data file should therefore be copied into "pal.in" using a command similar to the one below:

```
$ cp p7xxx.dat pal.in
```



Please Find Us a Home!

For the IBM PC, system requirements are as follows:

- 8088 based microprocessor system
- 128K bytes minimum of memory
- MS-DOS (PC-DOS) operating system
- Optional text printer
- 1 disk drive.

The IBM PC version comes with three diskettes which contain the following files:

```
Disk #1:PAL20.EXE, PAL24.EXE,  PALSETUP.EXE, PALCOMM.EXE,
        PALCOMM.DAT
Disk #2: P7000.DAT - P7049.DAT
Disk #3: P7050.DAT - P7099.DAT
```

- PAL20.EXE processes 20-pin PALs while PAL24.EXE handles 24-pin PALs. Both programs are written in FORTRAN77, however, each program is independent of the other during execution.
- PALSETUP.EXE allows the user to set up the baud rate, parity, data bits and stop bits for communication. This information is passed to the communications program, PALCOMM.EXE, through PALCOMM.DAT.
- Files P7000.DAT through P7099.DAT are example PAL Design Specifications taken from designs in the PAL Handbook.

To use PALASM on the IBM PC, two steps are necessary:

1. Create and edit the PAL Specification File.
2. Run PALASM.

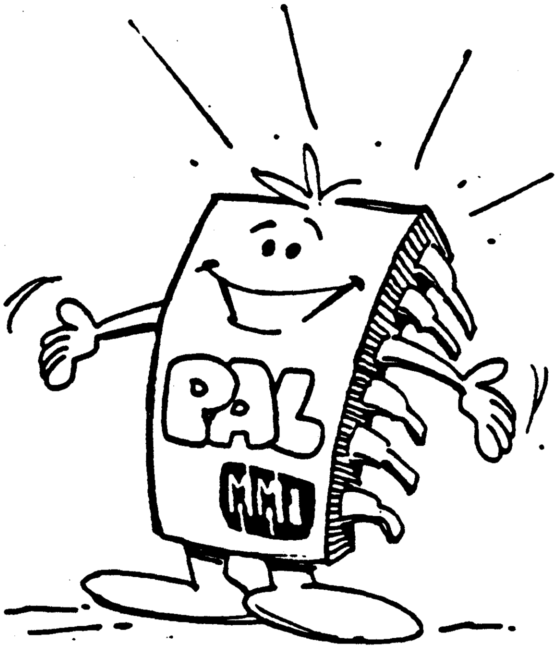
Assuming that you are working with a 20-pin PAL, to start PALASM, with Disk #1 in drive A, type the following when you see the A> prompt:

```
A>PAL20
```

Respond by typing the name of your source file and the drive on which it resides. If your input design spec cannot be found, the following error prompt is screened:

```
DISK I/O ERROR - MAYBE WRONG FILENAME ???

RESTART PALASM (Y/N)?
```

If the input file can be located, the user can then specify the output filename. Otherwise, pressing the carriage return key sends the output to the screen.

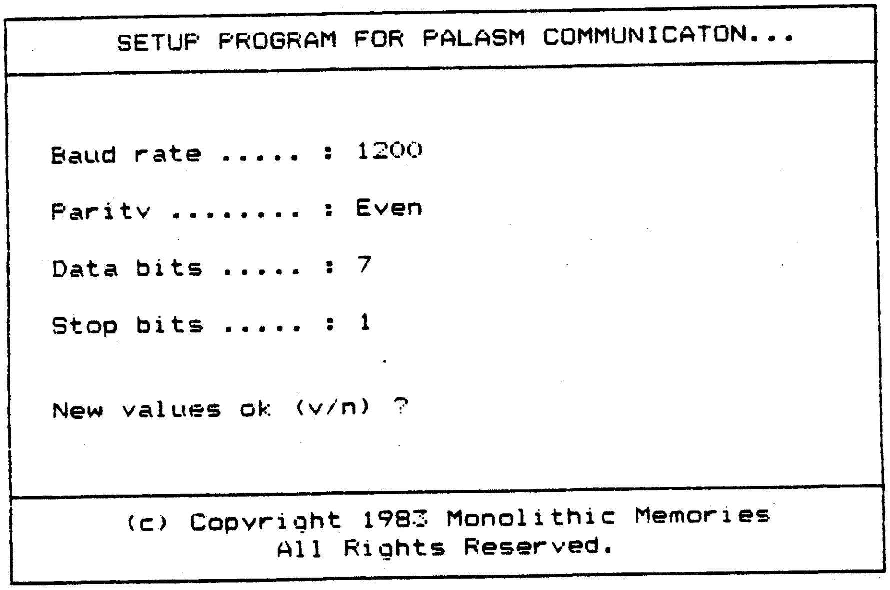


Programmer Communications

Before PALCOMM can be used, run PALSETUP to set up the baud rate, parity data bits and stop bits. This information is stored in PALCOMM.DAT and used by PALCOMM. To run the set up program, type:

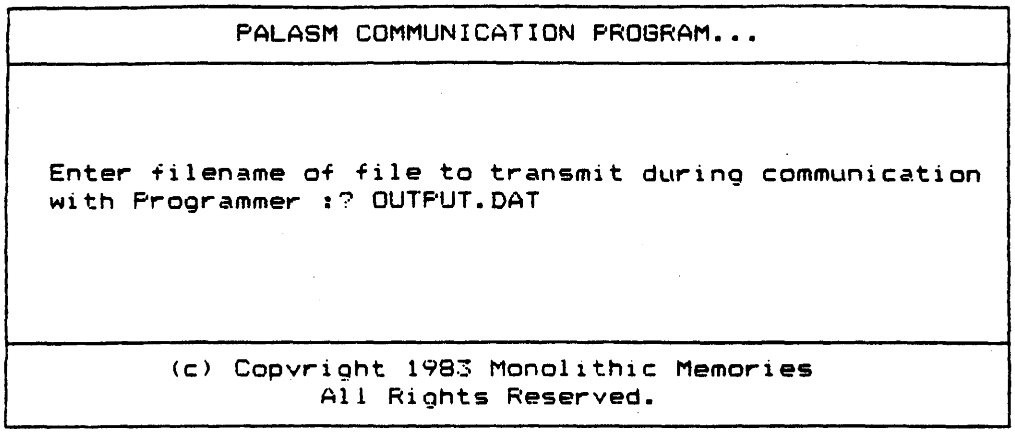
```
A>PALSETUP
```

The current default data is displayed. On the screen, the current parameters (baud rate, etc.) are highlighted, and the user can select an input value by pressing any key on the keyboard. Press the return key to select another parameter. When the last parameter value (stop bits) has been selected, the user is asked to confirm the information. If confirmed, the data is saved in PALCOMM.DAT.



```
SETUP PROGRAM FOR PALASM COMMUNICATON...

Baud rate ..... : 1200

Parity ........ : Even

Data bits ..... : 7

Stop bits ..... : 1

New values ok (y/n) ?

(c) Copyright 1983 Monolithic Memories
All Rights Reserved.
```

PALCOMM.EXE uses the data in PALCOMM.DAT to establish communications with the programmers. It then displays the following screen:



```
                PALASM COMMUNICATION PROGRAM...

  Enter filename of file to transmit during communication
  with Programmer :? OUTPUT.DAT

           (c) Copyright 1983 Monolithic Memories
                    All Rights Reserved.
```

Once the name of the file to be transmitted has been keyed in, the program responds with "Proceed...". The message shown below will be on line 25 of the screen. WARNING: All data is transmitted to the serial device. If no such device is available, the programs goes into time-out.

Press <F1> to send file ... Press <F10> to exit communications

Space Efficiency

Caveats, Restrictions and Other Gotchas

General

1. Do NOT terminate the program abnormally by pressing CTRL-C, BREAK, etc. when you are sending the output to a file rather than the screen. Use instead the QUIT option to terminate your session. If the session is terminated using CTRL-C, this may result in lost files on your disk. This is because your output file will not have been properly closed.

Supposingyour output file on B: drive is called OUTPUT.PRN, then if you abort your program using CTRL-C, when you look at the directory, it will indicate that your output file contains zero blocks:

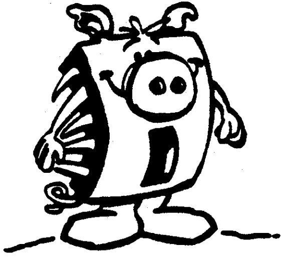
```
A>DIR

   OUTPUT.PRN   0        09-15-83    1:47p
```

Your disk however will contain lost files. To recover any lost files or clusters, run CHKDSK with the /F switch using the IBM 2.0 boot disk. Type:

```
A>CHKDSK B:/F

   1 lost clusters in 1 chain
   Convert lost chains to files (Y/N)?Y

   362496     bytes total disk space
        0     bytes in 1 hidden files
   216064     bytes in 2 user files
     1024     bytes in 1 recovered files
   145408     bytes available on disk

   327680     bytes total memory
   303104     bytes free
```

MS-DOS names the recovered file, FILE000.CHK. You can then delete OUTPUT.PRN and rename FILE000.CHK as you please.

Advantages of Using PALs



For CP/M-80 systems, the minimum configuration is:

- Z80 based microprocessor system
- 64K bytes minimum of memory
- CP/M-80 operating system
- Optional text printer
- 1 disk drive.

The three diskettes you receive will contain the following files:

Disk #1:PAL20.EXE, PAL24.EXE, PALSETUP.EXE, PALCOMM.EXE,
PALCOMM.DAT
Disk #2: P7000.DAT - P7049.DAT
Disk #3: P7050.DAT - P7099.DAT

- PAL20.EXE processes 20-pin PALs while PAL24.EXE handles 24-pin PALs. Both programs are written in FORTRAN77. Each, however, is independent of the other during execution.
- PALSETUP.EXE sets up the baud rate, parity, data bits and stop bits for communication. This information is passed to the communications program, PALCOMM.EXE, through PALCOMM.DAT.
- Files P7000.DAT through P7099.DAT are example PAL Specifications taken from designs inthe PALHandbook.

To use PALASM on the IBM PC, two steps are necessary:

1. Create and edit the PAL Specification File.
2. Run PALASM.

Assuming that you are working with a 20-pin PAL, to start PALASM, with Disk #1 in drive A, type the following when you see the A> prompt:

```
A>PAL20
```

Type the name of your source file and the drive on which it resides. If the input file does not exist, the following error prompt is screened:

```
DISK I/O ERROR - MAYBE WRONG FILENAME ???

RESTART PALASM (Y/N)?
```

If the input file is located, the user can then specify the output filename. Otherwise, pressing carriage return sends the output to the screen.

The PAL—Teaching Old PROMs New Tricks

The Intel/MDS version of PALASM is written in FORTRAN IV and is divided into four program segments:

- PAL20 assembles and generates fuse plots for 20-pin PALs
- PAL24 assembles and generates fuse plots for 24-pin PALs
- PAL20S simulates Function Table of 20-pin PALs
- PAL24S simulates Function Table of 24-pin PALs

ISIS-II Preliminaries

Under ISIS-II, the general format for naming a file is:

```
                :DEVICE:FILENAME.EXTENSION
                    ^       ^         ^
                    I       I         I
            device name     I         I
               1-6 alphanumeric chars I
                  1-3 alphanumeric chars
```

The following device names can be substituted for :DEVICE: in the line above:

:F0: thru :F5:	Disk Drives #0 thru #5
:TI:	Teletypewriter keyboard
:TO:	Teletypewriter printer
:TP:	Teletypewriter punch
:TR:	Teletypewriter reader
:VI:	Video terminal keyboard
:VO:	Video terminal screen
:HP:	High speed paper tape punch
:HR:	High speed paper tape reader
:LP:	Line printer

Since the console terminal may either be teletype or CRT, in order to input or output from the terminal, the following are also supported:

:CO:	Console output, and
:CI:	Console input

In addition, ISIS-II supports a number of user definable I/O devices (see ISIS-II User Guide, page 2-5 of manual order number: 9800306D).

Running the PALASM Assembler under ISIS-II

A typical session at a terminal is shown below. The PALASM disk is assumed to be in drive #0 and the Design Specification filename is P7000.DAT.

```
ISIS-II
-:F0:PAL20
```

The ISIS-II system prompt is a dash (-). Typing PAL20 invokes the PAL assembler.

```
PAL20 ASSEMBLER V2.4                          MONOLITHIC MEMORIES
ENTER PAL PROGRAM >>>
:F0:P7000.DAT
```

Our Design Specification is in drive #0

```
OPERATION CODES
P=PLOT B=BRIEF H=HEX D=JEDEC-FORMAT E=ECHO Q=QUIT
ENTER OPERATION CODE: P
```

Each operation code is described below:

P - Generates complete PAL fuse plot, where
  "X" - fuse not blown
  "-" - fuse blown
  "0" - low phantom fuse
  "O" - high phantom fuse

B - Generates brief fuse plot displaying only those fuse lines which are essential to describe the function.

H - Generates PAL fuse pattern in hexadecimal (HEX) programming format.

D - Generates JEDEC programming format. PALASM will ask for the interface to which the JEDEC format is to be downloaded. The Intel/MDS specifies every interface with a label:
  :CO: Console output
  :LP: Line printer, etc.

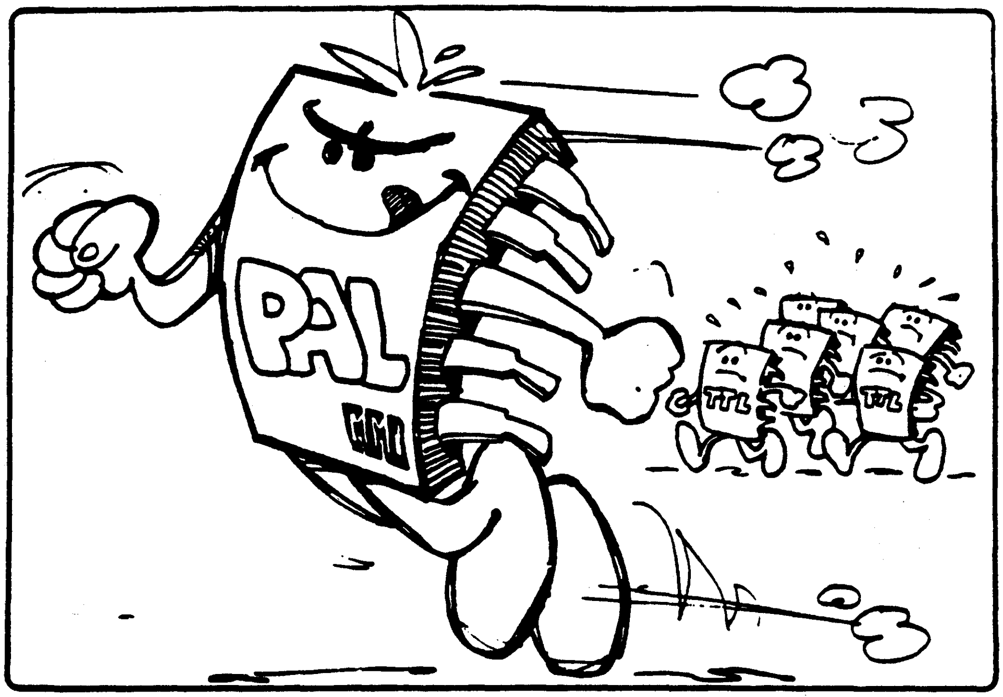
Although PALASM handles most interface labels, nevertheless, different series of the MDS often have different label names for the same interface. Check the Intel/MDS users´ manual for the interface to be driven.

Before down-loading data, put the Data I/O in receive mode by pressing <SELECT><E><B>.

E - Echos the PAL Design Specifications input file to the selected logical device number.

Q - Exits PALASM assembler.

PAL® Training



Monolithic MMI Memories

Caveats, Restrictions and Other Gotchas

General

1. PAL Design Specifications may be written using the ISIS-II Edit or Credit, but special invisible control characters like TAB, TOP OF FORM, etc. are not supported by PALASM. They can lead to unidentifiable errors.

2. PALASM is not foolproof. If you mistype the user data filename, PALASM complains with a FORTRAN I/O error (e.g. 156); or, if you try to open an already existing file during the PALASM output procedure, another FORTRAN I/O error is flagged (e.g. 155).

Assembler

1. PAL20 and PAL24 can only read up to 80 lines. Any file longer than 80 lines is truncated. Therefore, if your equations are longer than 80 lines, reformat them and reduce the number of lines in your input file. For example, instead of writing:

```
                        OUTPUT = A
                               + B
                               + C
    change to
                        OUTPUT = A + B + C
```

2. Function Table values for GND and VCC should not be specified; nor should GND and VCC be ANDed (*) with other inputs.

3. /GND and /VCC are recognized as complements.

4. GND allows the user to skip product lines and VCC provides for upper implementation of open collector functions in a PAL16L8 or PAL 20L10.

5. The assembler automatically skips to the third product line if an exclusive or operator :+: is encountered after the first minterm of an equation. This eliminates possible confusion when the PAL20X10, PAL20X8 and PAL20X4 are designed.

6. Many users write their equations with a feedback connection specified for pins 12 and 19 for PAL16L8 and pins 14 and 23 for PAL20L10. Since these pins have no internal feedback, the assembler flags this as an error.

7. The user should be aware that "active low" PALS will invert at the output even if the inversion is not specified in the equation. PALASM will flag an error.

8. The JEDEC format generated by this version of PALASM can be used with the Data I/O Model 19. In the next release, the JEDEC format will be modified to be compatible with the Model 29A.

Simulator

1. PAL20S and PAL24S, can read up to 140 lines, including logic equations and Function Table entries.

2. GND can be written in the product lines. If GND is specified in the logic equation, this product line is simulated as being always low. When VCC is specified, PALASM will blow all fuses for that product line. This product line is simulated as being always high.

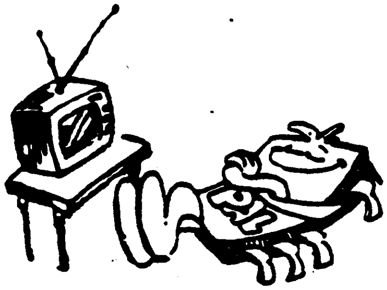
3. Note that IF(VCC) is optional (default) for conditional three-state outputs. The assembler automatically blows all fuses in this product line and simulates the output as being always enabled if no three-state line is specified.

4. The simulator assumes that registered outputs are enabled if the output control pin is excluded from the Function Table pin list. Many users prefer this when the output control pin is tied to GND.

5. Simulation of asynchronous outputs provides the ability to write a Function Table vector for a registered PAL without applying a clock pulse. Users are thus able to simulate asynchronous occurences in the PAL between clock pulses. Registered outputs can be tested or used as feedback with no change of state occuring.

6. The simulator bypasses the fuse plot. This enables the user to write and simulate more general Boolean equations which may not fit into the PAL architecture. Thus it is possible to simulate non-minimized Boolean equations.

7. The Intel/MDS version of PALASM is available in either single or double density 8-inch disks.



Following is a list of MMI approved programmer vendors. Although the list below is mostly for San Francisco Bay Area offices, you should be able to obtain information about their regional sales offices by calling the telephone numbers listed.

1. Data I/O
   473 Sapena Court/Suite 4
   Santa Clara, CA 95050 (408)727-0641

2. Structured Design
   1700 Wyatt Drive/Suite 3
   Santa Clara, CA 95054 (408)988-0725

3. Kontron
   630 Price Avenue
   Redwood City, CA 94063 (415)361-1012

4. Stag
   528-5 Weddell
   Sunnyvale, CA 94029 (408)745-1991

5. Digelec
   7335 E. Acoma Drive/Suite 103
   Scottsdale, AZ 85260 (602)991-7268

6. Storey Systems
   3213 N. Hwy 67/Suite 103
   Mesquite, TX 75150 (214)270-4135

7. Varix
   122 Spanish Village #608
   Dallas, TX 75248 (214)620-0925

8. Citel
   3060 Raymond Street
   Santa Clara, CA 95050 (408)727-6562

PAL Programmer Information

DATA I/O Model 29A PLDS Programmer:

Key Features:

- Uses JEDEC PAL programming format together with a "family & pin code" which uniquely identifies each PAL type.
- Capable of programming security fuse.
- Uses an RS-232C communications interface.

A Helpful Hint

- The latest DATA I/O E-PROM revision changes MMI`s manufacturer code from "95" to "22" in the JEDEC "family code". Please check that the configuration code for your E-PROM is EC8C.

Using the DATA I/O Model 29A

Turn on the DATA I/O PLDS programmer. After it finishes its self testing procedure, hit the following key sequence on the DATA I/O keyboard:

```
<SELECT>
<E>
<B>
<START>
<START>
```

If you are programming a PAL, rather than just running tests, you will see a sequence of lines starting with L.

If test vectors are to be sent to the DATA I/O, a sequence of lines beginning with V now appears.

The check sum data is displayed as it is transmitted to the DATA I/O.

The DATA I/O should now display a hexadecimal number on the left side of the display window and the LED on the proper socket should be on. If these conditions have not been met, something is wrong, probably the RS-232 interface.

SD20/24 PAL Programmer

Helpful Hints:

- Has 4K bytes of RAM.

- Uses an RS232 full-duplex serial interface.

- All units are shipped with 9600 BAUD rate.

- Data format is asynchronous bit serial with no parity, 1 start, 8 data and 1 stop bit.

- The SD20/24 prompts with "+" character.

Using the SD20/24 Programmer

1. To program a PAL, the equations must be downloaded from your computer to the SD20/24. Set the computer to the BAUD rate you desire. Make certain that the BAUD rate setting on the SD matches that of the computer!! To set the BAUD rate on the SD, remove the bottom metal cover, and select the desired BAUD rate using the jumper connector provided.

If you are using the PALCOMM program to communicate with the SD, put the SD into receive mode first by typing an L when you see the "+" prompt. Then press the "F1" key to begin transmission and Wwhen you see the end transmission prompt, press control-z to end it.

2. Insert the PAL device to be programmed in the socket and press the PROG key. If the RED LED flashes, this indicates that the PAL failed to program correctly or that the PAL failed functional testing.

3. The GREEN PASS LED will flash if programming is successful.

4. To verify the PAL against the fuse pattern in the RAM, insert the PAL in the socket and press the VERIFY switch. The SD20/24 performs a "smart verify"; it ignores all phantom fuse locations and unused products thereby decreasing program/verify times and increasing programming yields.

DIGELEC: sells a programmer for 20- and 24-pin PALs. It also uses an RS-232 interface. They have a unique programming format that is not presently supported by MMI. They are planning to produce a JEDEC format machine soon.

PROLOG: sells a 20-pin PAL programmer, on the RS-232 HEX format interface.

STAG: this RS-232 device uses the HEX format and can program 20- and 24-pin PALs.

High Speed

This section begins with a simple list of all error messages that each version of PALASM can generate on any system. This is followed by examples showing how these errors can be encountered.

1. A non-existent file has been specified as the source file or invalid filename specified as the output file.

DISK I/O ERROR - MAYBE WRONG FILENAME ???

****

2. The number of characters in the input file exceeds the maximum allowed. This should result in a run time error message. The message will vary depending on your particular operating system. The CP/M version allows up to 8000 characters in the input file. For the other versions, the maximum is 12000 characters.

TOO MANY CHARACTERS IN THE INPUT FILE

****

3. An invalid PAL part type has been specified in line 1 of the input file.

PAL PART TYPE $$$$$$$ IS INCORRECT

****

4. More than 20 pin names have been specified for a 20-pin part. The following message flags this error for 24-pin parts.

LESS THAN 20 PIN NAMES IN PIN LIST

LESS THAN 24 PIN NAMES IN PIN LIST

****

5. A pin name specified in the function table does not match any of the names specified in the device pin list.

ERROR SYMBOL = $$$$$$$$ IN LINE NUMBER $$$

****

6. An active low PAL has been selected therefore the output pin name in the Boolean equations should be the complement of the corresponding pin name in the device pin list.

OUTPUT MUST BE INVERTED SINCE PAL$$$$ IS AN ACTIVE LOW DEVICE

****

7. For an active high PAL, the output pin name should be the same as the pin name in the pin list.

```
OUTPUT CANNOT BE INVERTED SINCE PAL$$$$ IS AN ACTIVE HIGH DEVICE
```

****

8. Invalid output pin specified, please refer to the PAL handbook for device pinouts.

```
THIS PIN NUMBER $$ IS AN INVALID OUTPUT PIN FOR PAL$$$$
```

****

9. Problem minterm in the flagged output equation.

```
OUTPUT PIN NAME = $$$$$$$$     OUTPUT PIN NUMBER = $$
MINTERM IN LINE NUMBER $$$
```

****

10. This product line is non-existent for this PAL type.

```
THIS PRODUCT LINE NUMBER $$ IS NOT VALID FOR PAL$$$$
```

****

11. This PAL type has a maximum of 8 product lines; too many product terms have been specified.

```
MAXIMUM OF 8 PRODUCT LINES ARE VALID FOR PAL$$$$
TOO MANY MINTERMS ARE SPECIFIED
```

****

12. Invoking the SIMULATE option requires that the function table be present in the input specification.

```
FUNCTION TABLE MUST BE SUPPLIED IN ORDER TO PERFORM SIMULATION
```

****

13. The function table pin list does not correspond with the device pin list specified earlier.

```
FUNCTION TABLE PIN LIST ERROR AT $$$$$$$$
```

****

14. The symbol flagged is not a valid entry in the function table.

$ IS NOT AN ALLOWED FUNCTION TABLE ENTRY AT VECTOR $$$

****

15. The expected value for this output in the function table does not agree with that computed from the Boolean equations. The various forms of this error message are shown below:

FUNCTION TABLE ERROR AT VECTOR $$$ PIN = $$$$$$$$
EXPECT = H ACTUAL = L

FUNCTION TABLE ERROR AT VECTOR $$$ PIN = $$$$$$$$
EXPECT = L ACTUAL = H

FUNCTION TABLE ERROR AT VECTOR $$$ PIN = $$$$$$$$
EXPECT = OUTPUT ENABLE ACTUAL = Z

FUNCTION TABLE ERROR AT VECTOR $$$ PIN = $$$$$$$$
EXPECT = Z ACTUAL = $

The offending line in the function table is then printed out with a question mark in place of the incorrect entry.

****

16. Provides a count of errors detected in the function table.

NUMBER OF FUNCTION TABLE ERRORS = $$$$

****

17. A symbol other than H(high), L(low), or X(don´t care) has been entered in the function table.

ERROR SYMBOL = $$$$$$$$ IN LINE NUMBER $$$$

****

18. The number of entries in the vector does not match the number of pins defined in the function table pin list.

PIN NAME IS NOT DEFINED IN THE FUNCTION TABLE PIN LIST

****

1. Open Tooled Bit Patterns
by Vincent Coli

PAT #	ALT ID	TITLE	PAL TYPE
P7000	PH-4-3	Basic Gates	PAL12H6
P7001	PH-4-9	Basic Clocked Flip-Flops	PAL16R8
P7002	PH-4-17	Memory Mapped I/O	PAL16L2
P7003	PH-4-23	Memory Interface Logic for 6800 Microprocessor Bus	PAL10L8
P7004	PH-4-31	Video Logic	PAL20L10
P7005	PH-4-37	Binary to BCD Converter	PAL16R8
P7006	PH-4-47	Deglitcher	PAL16R4
P7007	PH-4-53	Electronic Dice Game	PAL16R8
P7008	PH-4-63 PMS1405	9-Bit Register	PAL20X10
P7009	PH-4-69 DB-8-12	Multifunction Octal Register (SN54/74LS380)	PAL20X8
P7010	PH-4-77	8-Bit Priority Interrupt Encoder with Register	PAL16R4
P7011	PH-4-83	BCD/HEX Counter	PAL20X8
P7012	PH-4-91	64K Dynamic RAM Refresh/Controller	PAL20X4
P7013	PH-4-99 AN-114	SN54/74S516 (16-Bit Multiplier/Divider/ Accumulator) State Counter	PAL16R8
P7014	PH-4-107	ALU/Accumulator	PAL16A4
P7015	PH-4-113	Stepper Motor Controller	PAL16R4
P7016	PH-4-123	Shaft Encoder	PAL16R4
P7017	PH-4-129 DB-8-28	Quad 4:1 Mux (SN54/74LS453)	PAL18L4
P7018	PH-4-137 DB-8-24	Dual 8:1 Mux (SN54/74LS451)	PAL20L2

P7019	PH-4-145 DB-8-20	16:1 Mux (SN54/74LS450)	PAL20C1
P7020	PH-4-153 DB-8-8	Octal Shift Register (SN54/74LS498)	PAL20X8
P7021	PH-4-161	4-Bit Shift Register/Comparator	PAL16R4
P7022	PH-4-169	4-Bit Counter with 2 Input Mux	PAL16R4
P7023	PH-4-177 DB-8-4	Octal Counter (SN54/74LS461)	PAL20X8
P7024	PH-4-185 PMS1401	Octal Up/Down Counter (SN54/74LS490)	PAL20X8
P7025	PH-4-193 DB-8-16	10-Bit Counter (SN54/74LS491)	PAL20X10
P7026	PH-4-201	4-Bit Up/Down Counter with Shift Register and Comparator	PAL16X4
P7027	PH-4-209	4-Bit Flash Gray A/D Converter	PAL20X4
P7028	PH-4-217	4-Bit Gray D/A Converter	PAL16R4
P7029	PH-4-225	8-Bit D/A Converter	PAL16L8
P7030	PH-4-233 PMS1003	Octal Comparator	PAL16C1
P7031	PH-4-241	Between Limits Comparator/Register	PAL16X4
P7032	PH-4-246	Between Limits Comparator/Logic	PAL16C1
P7033	PH-4-253	Memory Mapped Printer Decoder	PAL16L2
P7034	PH-4-256	Printer Data Register/Mux	PAL20X8
P7035	PH-4-263	8-State Machine and Win-Lose Decoder	PAL16R4
P7036	PH-4-268	36-State Machine	PAL16R6
P7037	PH-4-274	Dice Decoder	PAL16L8
P7038	PH-4-280	Store Point /2, /7, and /11 Decode	PAL16R4
P7039	PH-4-290	Traffic Signal Controller No. 1	PAL16R4
P7040	PH-4-296	Traffic Signal Controller No. 2	PAL16R8
P7041	PH-4-305	32-Bit CRC (Cyclical Redundancy Checking) Error Detection, Chip 1	PAL20X8

P7042	PH-4-310	32-Bit CRC (Cyclical Redundancy Checking) Error Detection, Chip 2	PAL20X8
P7043	PH-4-316	32-Bit CRC (Cyclical Redundancy Checking) Error Detection, Chip 3	PAL20X8
P7044	PH-4-322	32-Bit CRC (Cyclical Redundancy Checking) Error Detection, Chip 4	PAL20X8
P7045	PH-4-332	Check Bit Generator	PAL16X4
P7046	PH-4-336	Syndrome Bit Gnerator	PAL16X4
P7047	PH-4-342	Error Detection Unit No. 1	PAL16L8
P7048	PH-4-346	Error Detection Unit No. 2	PAL16L8
P7049	PH-5-17	Dot Generator	PAL20X8
P7050	PH-5-25	CHAR/CURS Generator	PAL20X10
P7051	PH-5-33	SCAN/LINE Generator	PAL20X10
P7052	PH-5-43	LINES/SCROL Generator	PAL20X10
P7053	PH-5-51	Composite Video/Baud Rate Generator	PAL20X10
P7054	PH-5-59	UART Shift Register and Control Key Detect	PAL20X8
P7055	PH-5-67	UART Control	PAL20X10
P7056	PH-5-77	RAM Control	PAL20X10
P7057	NONE	4-Bit Serial Switch	PAL14L4
P7058	NONE	4-Bit Shifter	PAL14L4
P7059	PMSI402	Octal Down Counter	PAL20X8
P7060	NONE	4-Bit Counter with Transparent Latch	PAL20X4
P7061	PMSI406	10-Bit Register	PAL20X10
P7062	PMSI001	3-to-8 Demultiplexer with Control Storage	PAL16R8
P7063	PMSI404	9-Bit Counter	PAL20X10
P7064	NONE	10-Bit Comparator (SN54/74LS460)	PAL20C1
P7065	NONE	6-Bit Shift Register	PAL16R6
P7066	NONE	6-Bit Counter	PAL16R6
P7067	NONE	6-Bit Multifunction Register	PAL16R6

P7068	NONE	Multifunction 9-Bit Buffer	PAL20L10
P7069	NONE	10-Bit Buffer	PAL20L10
P7070	NONE	10-Bit Open Collector Buffer	PAL20L10
P7071	NONE	10-Bit Open Collector Inverting Buffer	PAL20L10
P7072	PMSI407	10-Bit Addressable Register	PAL20X10
P7073	NONE	MC6800 Microprocessor Interface	PAL20L10
P7074	NONE	Quad 3:1 Multiplexer	PAL14L4
P7075	NONE	4-Bit Counter with Registers	PAL20X8
P7076	NONE	9-Bit Down Counter	PAL20X10
P7077	NONE	Refresh Clock Generator	PAL20X10
P7078	NONE	Octal Addressable Register	PAL16R8
P7079	NONE	Octal Addressable Register with Demux/Enables	PAL16R8
P7080	PMSI002	Octal Addressable Register with Demux/Clear	PAL16R8
P7081	NONE	Rounding-Control Logic	PAL16C1
P7082	NONE	4-Bit Counter with Terminal Count Lock	PAL16R6
P7083	NONE	Memory Mapped I/O	PAL18L4
P7084	NONE	9-Bit Counter with Terminal Count Lock	PAL20X10
P7085	NONE	'S508 Memory Map Interface with the Intel 8085	PAL16R4
P7086	NONE	'S508 I/O Device Interface with the Intel 8085	PAL16R4
P7087	PMSI403	2-Digit BCD Counter	PAL20X8
P7088	NONE	9-Bit Counter with Shift Register	PAL20X10
P7089	PMSI408	Interface Controller for 68000 to Zilog 8500	PAL20X10
P7090	NONE	16 Input Priority Encoder	PAL20R4
P7091	NONE	16 Input Priority Encoder Interrupt Flag	PAL16C1
P7092	NONE	15 Input Registered Priority Encoder	PAL20R4
P7093	NONE	8 Input Priority Encoder with Interupt Flag	PAL16R4
P7094	NONE	Dual Stepper Motor Controller	PAL16R8

P7095	NONE	Octal Registered Barrel Shifter	PAL20R8
P7096	NONE	Clean Octal Latch	PAL20L10
P7097	NONE	Shaft Encoder No. 1	PAL16R4
P7098	NONE	Shaft Encoder No. 2	PAL16R8
P7099	NONE	Shaft Encoder No. 3 (with Internal 4-Bit Up/Down Counter)	PAL20X10

LEGEND (Note: current MMI publications only)

PH-m-n = PAL Handbook Second Edition, section m, page n

DB-m-n = 1982 LSI Data Book, section m, page n (also referenced appendices)

PMSIxxx = PMSI - PAL Medium Scale Integration Book (xxx is the part number)

2. Bit Pattern Cross-Referance

PAL10L8	: P7003
PAL12H6	: P7000
PAL14L4	: P7057, P7058, P7074
PAL16A4	: P7014
PAL16C1	: P7030, P7032, P7081, P7091
PAL16H2	: P7033
PAL16L2	: P7002
PAL16L8	: P7029, P7037, P7047, P7048
PAL16R4	: P7006, P7010, P7015, P7016, P7021, P7022, P7028, P7035, P7038, P7039, P7085, P7086, P7093, P7097
PAL16R6	: P7036, P7065, P7066, P7067, P7082
PAL16R8	: P7001, P7005, P7007, P7013, P7040, P7062, P7078, P7079, P7080, P7094, P7098
PAL16X4	: P7026, P7031, P7045, P7046
PAL18L4	: P7017, P7083
PAL20C1	: P7019, P7064
PAL20L2	: P7018
PAL20L10	: P7004, P7068, P7069, P7070, P7071, P7073, P7096
PAL20R4	: P7090, P7092
PAL20R8	: P7095
PAL20X4	: P7012, P7027, P7060
PAL20X8	: P7009, P7011, P7020, P7023, P7024, P7034, P7041, P7042, P7043, P7044, P7049, P7054, P7059, P7075, P7087
PAL20X10	: P7008, P7025, P7050, P7051, P7052, P7053, P7055, P7056, P7061, P7063, P7072, P7076, P7077, P7084, P7088, P7089, P7099

3. Bit Pattern Descriptions

***

P7000

THIS EXAMPLE ILLUSTRATES THE USE OF FUSIBLE LOGIC TO IMPLEMENT THE BASIC GATES; INVERTER, AND GATE, OR GATE, NAND GATE, NOR GATE, AND EXCLUSIVE OR GATE.

THE FUNCTION TABLE EXERCISES ALL INPUTS AND TESTS BASIC FUNCTION PERFORMANCE. PALASM EXERCISES THE FUNCTION TABLE TO SIMULATE THE BASIC GATES.

***

P7001

THIS EXAMPLE ILLUSTRATES THE USE OF FUSIBLE LOGIC TO IMPLEMENT THE BASIC FLIP-FLOPS: J-K FLIP-FLOP, T FLIP-FLOP, D FLIP-FLOP, AND S-R FLIP-FLOP.

NEXT STATE TABLE FOR THE BASIC FLIP-FLOPS:

TYPE OF FLIP-FLOP	INPUT	Q = L		Q = H	
		Q+ = L	Q+ = H	Q+ = L	Q+ = H
J-K	J	L	H	X	X
	K	X	X	H	L
T	T	L	H	H	L
D	D	L	H	L	H
SET-RESET	S	L	H	L	X
	R	X	L	H	L

NOTE THAT A PAL16L8 MAY BE SUBSTITUTED FOR THIS DESIGN. THEN THE CLOCK INPUT(CLK) WOULD BE GATED WITH THE DATA INPUTS TO IMPLEMENT THE BASIC FLIP-FLOPS.

THE FUNCTION TABLE EXERCISES ALL INPUTS AND TESTS BASIC FUNCTION PERFORMANCE. PALASM EXERCISES THE FUNCTION TABLE TO SIMULATE THE BASIC CLOCKED FLIP-FLOPS.

***

P7002

THIS PAL PROVIDES A SINGLE CHIP DECODER FOR USE IN MEMORY MAPPED I/O OPERATIONS. EQUATION TERMS CAN BE CHANGED TO ACCOMMODATE ANY 16-BIT ADDRESS.

THE PAL WILL MONITOR THE SYSTEM MEMORY ADDRESS BUS AND DECODE THE SPECIFIED MEMORY ADDRESS WORD (1F78,1F79) TO PRODUCE A PORT ENABLE PIN FOR PORT0 AND PORT1.

***

P7003

THIS DEVICE PROVIDES THE INTERFACE LOGIC BETWEEN A 6800 MICROPRESSOR BUS AND FOUR STATIC 4k MEMORY CHIPS. ADDRESS BUS (A), READ/WRITE (RW), PHASE 2 CLOCK (PHASE2), AND VALID MEMORY ADDRESS (VPHASE2) ARE DECODED TO PRODUCE THE PROPER WRITE ENABLE (WEOE), CHIP ENABLE (CE), AND OUTPUT DISABLE (CSOD) SIGNALS FOR MEMORY DATA TRANSFERS.

NOTE THAT /CE0 AND /CE1 ARE THE COMPLEMENTS OF CSOD0 AND CSOD1, RESPECTIVELY. THESE FUNCTIONS ARE IMPLEMENTED BY THE EXTERNAL CONNECTIONS CSOD0 TO PIN 9 AND CSOD1 TO PIN 11.

***

P7004

THIS PAL REPLACES ALL OF THE TTL LOGIC USED ON A VIDEO DRIVER BOARD (5 ICs) TOGETHER WITH 4 PULL-UP RESISTORS.

***

P7005

THE FUNCTION OF THIS PAL IS TO CONVERT A SERIAL STREAM OF BINARY DATA INTO A PARALLEL BCD REPRESENTATION. AFTER EACH CLOCK PULSE, THE BCD OUTPUT CONTAINS THE CORRECT BCD REPRESENTATION FOR THE RELATIVE BINARY DATA SHIFTED SO FAR.

THE INPUT BINARY DATA IS SHIFTED LEFT (STARTING WITH THE MSB) INTO THE BCD REGISTER. THIS TECHNIQUE IS KNOWN AS COULEUR'S TECHNIQUE (BIDEC). THE COMBITORIAL NETWORK IS DESIGNED FROM THE FOLLOWING NEXT-STATE TABLE:

PRESENT STATE B3-B0	NEXT STATE B3-B0	COUT
0-4	0-8	0
5-9	0-8	1
10-15	X	X

***

P7006

THE SYSTEM DETECTS THE BEGINNING AND THE END OF A SIGNAL WHICH IS DISTURBED BY GLITCHES.

***

P7007

THE DUAL MODULO-SIX COUNTER INCREMENTS ON THE RISING EDGE OF THE

CLOCK (CK). THE THREE-STATE OUTPUTS ARE HIGH-Z WHEN THE OUTPUT CONTROL LINE (/OC) IS HIGH AND ENABLED WHEN THE OUTPUT CONTROL LINE (/OC) IS LOW.

THE "INIT" LINE IS NEEDED TO INITIALIZE THE COUNTER SO THAT THE FUNCTION TABLE SIMULATION COULD BE PERFORMED AND THE PART COULD BE TESTED AT THE TIME OF FABRICATION. THIS LINE AS WELL AS ALL OTHER UNUSED INPUTS SHOULD BE TIED TO GND.

THERE ARE 36 DIFFERENT STATES TO THE COUNT SEQUENCE. EACH STATE CORRESPONDS TO ONE OF THE NUMBER COMBINATIONS TO BE DISPLAYED ON THE DICE.

NOTE THAT THE PINOUT IS CHOSEN TO CONVENIENCE PC BOARD LAYOUT.

****

P7008

THIS 9-BIT REGISTER LOADS THE DATA (D8-D0) ON THE RISING EDGE OF THE CLOCK (CLK) IF THE LOAD LINE (/LD) IS ASSERTED (LOW ON PIN 11) AND OTHERWISE HOLDS THE ORIGINAL VALUE.

THE 9-BIT ARCHITECTURE MAKES THIS REGISTER IDEAL FOR PARITY BUS INTERFACING IN MICROPROGRAMMED SYSTEMS.

THESE OPERATIONS ARE EXERCISED IN THE FUNCTION TABLE AND SUMMARIZED IN OPERATIONS TABLE:

/OC	CLK	/LD	D8-D0	Q8-Q0	OPERATION
H	X	X	X	Z	HI-Z
L	C	H	X	Q	HOLD
L	C	L	D	D	LOAD

****

P7009

THIS IS AN 8-BIT SYNCHRONOUS REGISTER WITH PARALLEL LOAD, LOAD COMPLEMENT, PRESET, CLEAR, AND HOLD CAPABILITIES. FOUR CONTROL INPUTS (/LD,POL,/CLR,/PR) PROVIDE ONE OF FOUR OPERATIONS WHICH OCCUR SYNCHRONOUSLY WITH THE CLOCK (CLK).

THE LOAD OPERATION LOADS THE INPUTS (D7-D0) INTO THE OUTPUT REGISTER (Q7-Q0), WHEN POL=H OR LOADS THE COMPLEMENT OF THE INPUTS WHEN POL=L. THE CLEAR (/CLR) OPERATION RESETS THE OUTPUT REGISTERS TO ALL LOWS. THE PRESET (/PR) OPERATION PRESETS THE OUTPUT REGISTERS TO ALL HIGHS. THE HOLD OPERATION HOLDS THE PREVIOUS VALUE REGARDLESS OF CLOCK TRANSITIONS.

CLEAR OVERRIDES PRESET, PRESET OVERRIDES LOAD, AND LOAD OVERRIDES HOLD.

THESE OPERATIONS ARE EXERCISED IN THE FUNCTION TABLE AND SUMMARIZED IN THE

OPERATIONS TABLE:

/OC	CLK	/CLR	/PR	/LD	POL	D7-D0	Q7-Q0	OPERATION
H	X	X	X	X	X	X	Z	HI-Z
L	C	L	X	X	X	X	L	CLEAR
L	C	H	L	X	X	X	H	PRESET
L	C	H	H	H	X	X	Q	HOLD
L	C	H	H	L	H	D	D	LOAD TRUE
L	C	H	H	L	L	D	/D	LOAD COMP

****

P7010

THE I/O PRIORITY INTERRUPT ENCODER PRIORITIZES 8 I/O LINES (I1 THRU I8) PRODUCING 111 (Q3, Q2, AND Q1 RESPECTIVELY) FOR THE HIGHEST PRIORITY I/O DEVICE (I1) AND 000 FOR AN INTERRUPT FROM THE LOWEST PRIORITY I/O DEVICE (I8).

OUTPUT Q4 SERVES AS THE INTERRUPT FLAG AND GOES LOW WHEN ANY OF THE 8 I/O INPUTS GO HIGH.

THE PRIORITY INTERRUPT ENCODER REGISTERS ARE UPDATED ON THE RISING EDGE OF THE INTERRUPT CLOCK INPUT (CLK). THE 3-STATE OUTPUTS ARE HIGH-Z WHEN THE OUTPUT CONTROL LINE (/OC) IS LOW.

****

P7011

FOUR IDENTICALLY PROGRAMMED PAL DEVICES ARE USED TO DRIVE EIGHT OF HP'S NUMERIC AND HEX INDICATORS (5082-7340). EACH PAL CONSISTS OF TWO FOUR BIT COUNTERS. STAGE 1 IS THE LSB AND STAGE 2 IS THE MSB. CARRYOUT OF STAGE 1 IS CALLED INTERNAL CARRY (COUT1) AND IS FED EXTERNALLY TO STAGE 2. COUT2 IS FED INTO THE NEXT PAL. CARRYOUT AND INTERNAL CARRYS FROM THE LOWER PAL ARE CONNECTED TO ALL OF THE HIGHER PAL DEVICES TO PERFORM THE CARRY LOOK AHEAD OPERATION.

THE BCD/HEX COUNTER HAS BUILT IN TESTABILITY. COUT1 IS CONNECTED TO CIN EXTERNALLY AND CAN FORCE COUT1 TO GO HIGH, THUS STAGE 2 MAY START COUNTING AT THE SAME TIME AS STAGE 1 WHICH REDUCES THE NUMBER OF TEST VECTORS IN THE FUNCTION TABLE.

THIS COUNTER OPERATES AT 10 MHz AND CAN PERFORM THE FOLLOWING OPERATIONS:

HEX	CLR	OPERATION
X	H	CLEAR
L	L	COUNT BCD
H	L	COUNT HEX

****

P7012

TWO IDENTICALLY PROGRAMMED PAL20X4 CAN PERFORM THE 64K DYNAMIC RAM REFRESH CONTROL FUNCTION.

EITHER COLUMN OR ROW ADDRESSES TO THE RAM ARE SELECTED DEPENDING ON ROW ENABLE (ROWEN).

AN ADDRESS COUNTER (C3-C0) IS SELECTED DURING REFRESH WHEN ROW ENABLE (ROWEN) IS HIGH.

THESE OPERATIONS ARE EXERCISED IN THE FUNCTION TABLE AND SUMMERIZED IN THE OPERATIONS TABLE:

/OC	COUNT	REFEN	ROWEN	03-00	OPERATION
H	X	X	X	Z	HI-Z
L	C	L	L	A3-A0	SELECT LOW ADDR BITS
L	C	L	H	A11-A8	SELECT UPPER ADDR BITS
L	C	H	H	C3-C0	SELECT REFRESH ADDR BITS
L	C	H	L	H	SET REFRESH COUNTER

****

P7013

THIS PAL TRACKS ALL 14 OF THE VALID STATE TRANSITIONS FOR THE SN54/74S516, 16-BIT SEQUENTIAL MULTIPLIER/DIVIDER/ACCUMULATOR. THE PAL MONITORS THE 3-BIT INSTRUCTION LINE (I2-I0) AND CHIP ACTIVATION INPUT (/GO) TO THE 'S516 IN ORDER TO PROVIDE THE 4-BIT STATE OF THE MACHINE (D,C,B,A) SYNCHRONOUS WITH THE '516 CLOCK (CK).

THE PAL IS INITIALIZED TO STATE 0 BY DRIVING INIT HIGH (INIT=H) FOLLOWED BY A CLOCK PULSE. HOWEVER IN A SYSTEM WITH THE 'S516, THE PAL CAN BE INITIALIZED TO STATE 0 ALONG WITH THE 'S516 BY RECEIVING 23 CLOCK PULSES WITH INSTRUCTION CODE 7.

IN ADDITION TO THE INITIALIZATION PIN (INIT), TWO OTHER PINS ARE INCLUDED TO MAKE IT EASIER TO FUNCTIONAL TEST THE DEVICE:

SET8 - WHEN SET8=HIGH, THE MACHINE WILL BE SET TO STATE 8
SET10 - WHEN SET10=HIGH, THE MACHINE WILL BE SET TO STATE 10

NOTICE THAT INIT, SET8, AND SET10 CAN BE CONVENIENTLY TIED TO GROUND FOR NORMAL CIRCUIT OPERATION. ALSO SET10 OVERRIDES SET8.

OPERATIONS TABLE FOR THE 'S516 STATE COUNTER:

/OE	CK	INIT	/MODE	/GO	I2-I0	/D-/A	OPERATION
H	X	X	X	X	X	Z	HI-Z
L	C	H	X	X	X	H	INITIALIZATION
L	C	L	L	X	X	X	FRACTIONAL ARITHMETIC MODE *
L	C	L	H	X	X	X	INTEGER ARITHMETIC MODE *
L	C	L	X	H	X	S	HOLD STATE 0, 1, 3, 8, 10, OR
L	C	L	X	L	I	S+	STATE TRANSITION

* THIS PAL TRACKS THE 'S516 EXACTLY UNDER ALL CONDITIONS EXCEPT WHEN CHAINED INTEGER DIVSION IS TO BE PERFORMED. IN THIS CASE, THE /MODE PIN IS CONNECTED TO AN EXTERNAL REGISTER WHICH IS SET WHEN THE 'S516 PASSES THROUGH STATE 1 WITH INSTRUCTION CODE 6 AND CLEARED WHEN THE 'S516 PASSES THROUGH STATE 1 WITH INSTRUCTION CODE 5. IF YOU DO NOT REQUIRE CHAINED INTEGER DIVISION, THEN SIMPLY CONNECT THE /MODE PIN TO GROUND AND THE PAL STATE COUNTER WILL AUTOMATICALLY KEEP TRACK OF THE REQUIRED NUMBER OF DIVISION LOOPS FOR THE INTEGER AND FRACTIONAL MODES. CONSULT THE SN54/74S516 DATA SHEET FOR MORE INFORMATION ON THE INTEGER AND FRACTIONAL ARITHMETIC MODES.

****

P7014

THE ALU ACCUMULATOR LOADS THE A-REGISTER WITH ONE OF EIGHT OPERANDS ON THE RISING EDGE OF THE CLOCK. G AND P OUTPUT GENERATE AND PROPAGATE ON THE ADD INSTRUCTION. P OUTPUTS OP = ZERO ON INSTRUCTIONS 1,2,3,5,6,7.

OPERATIONS TABLE:

/OC	CLK	I3	I2	I1	I0	LIO	CIN	A3-A0	OPERATION	
H	X	X	X	X	X	X	X	Z	HI-Z	A =Z
L	C	L	L	L	L	X	L	A PLUS B	ADD	A:=A PLUS B
L	C	L	L	L	L	X	H	A PL B PL 1	ADD	A:=A PLUS B PLUS
L	C	L	L	L	H	X	X	A	HOLD	A:=A
L	C	L	L	H	L	X	X	B	LOAD	A:=B
L	C	L	L	H	H	X	X	A AND B	AND	A:=A*B
L	C	L	H	L	L	X	X	/B	LOAD COMP	A:=/B
L	C	L	H	L	H	X	X	A OR B	OR	A:=A+B
L	C	L	H	H	L	X	LI	SL(A)	SHIFT LEFT	
L	C	L	H	H	H	RI	X	SR(A)	SHIFT RIGHT	
L	C	H	X	X	X	X	X	A	HOLD	A:=A

****

P7015

THIS PAL16R4 PROVIDES THE LOGIC LEVELS REQUIRED TO DRIVE TWO STEPPER MOTORS IN THE FULL STEP MODE.

THE FOLLOWING OPERATIONS MAY BE PERFORMED FOR EACH STEPPER MOTOR CONTROLLER INDIVIDUALLY:

CLK	/E1	/E2	S	D	OPERATION
X	H	X	X	X	HOLD MOTOR IN CURRENT POSITION
X	X	H	X	X	HOLD MOTOR IN CURRENT POSITION

C	L	L	H	X	SET OUTPUTS TO STEP 1 LEVELS
C	L	L	L	L	STEP MOTOR CLOCKWISE
C	L	L	L	H	STEP MOTOR COUNTER-CLOCKWISE

****

P7016

****

P7017

THIS IS AN EXAMPLE OF A QUAD 4-TO-1 MULTIPLEXER USING A PAL18L4. SELECT LINES A,B ARE ENCODED IN BINARY, WITH A REPRESENTING THE LSB.

OPERATIONS TABLE:

INPUT SELECT		OUTPUTS
B	A	Y
L	L	C0
L	H	C1
H	L	C2
H	H	C3

****

P7018

THIS IS AN EXAMPLE OF A DUAL 8-TO-1 MULTIPLEXER USING A PAL20L2. A STROBE LINE (S) IS PROVIDED TO GATE THE OUTPUTS OFF (Y=H) WHEN THE STROBE INPUT IS HIGH. THE SELECT LINES A,B,C ARE ENCODED IN BINARY, WITH A REPRESENTING THE LSB.

OPERATIONS TABLE:

-----INPUTS-----				OUTPUTS
SELECT			STROBE	
C	B	A	S	Y
X	X	X	H	H
L	L	L	L	D0
L	L	H	L	D1
L	H	L	L	D2
L	H	H	L	D3
H	L	L	L	D4
H	L	H	L	D5
H	H	L	L	D6
H	H	H	L	D7

****

P7019

THIS IS AN EXAMPLE OF A 16-TO-1 MULTIPLEXER USING A PAL20C1. BOTH TRUE (Y) AND COMPLEMENT (W) OUTPUTS ARE PROVIDED. THE SELECT LINES A,B,C,D ARE ENCODED IN BINARY, WITH A REPRESENTING THE LSB AND D REPRESENTING THE MSB.

OPERATIONS TABLE:

INPUTS SELECT LINES				OUTPUTS	
D	C	B	A	W	Y
L	L	L	L	/E0	E0
L	L	L	H	/E1	E1
L	L	H	L	/E2	E2
L	L	H	H	/E3	E3
L	H	L	L	/E4	E4
L	H	L	H	/E5	E5
L	H	H	L	/E6	E6
L	H	H	H	/E7	E7
H	L	L	L	/E8	E8
H	L	L	H	/E9	E9
H	L	H	L	/E10	E10
H	L	H	H	/E11	E11
H	H	L	L	/E12	E12
H	H	L	H	/E13	E13
H	H	H	L	/E14	E14
H	H	H	H	/E15	E15

****

P7020

THIS PAL IS AN 8-BIT SHIFT REGISTER WITH PARALLEL LOAD AND HOLD CAPABILITY. TWO FUNCTION SELECT INPUTS (I0,I1) PROVIDE ONE OF FOUR OPERATIONS WHICH OCCUR SYNCHRONOUSLY ON THE RISING EDGE OF THE CLOCK (CLK). THESE OPERATIONS ARE:

/OC	CLK	I1	I0	D7-D0	Q7-Q0	OPERATION
H	X	X	X	X	Z	HI-Z
L	C	L	L	X	L	HOLD
L	C	L	H	X	SR(Q)	SHIFT RIGHT
L	C	H	L	X	SL(Q)	SHIFT LEFT
L	C	H	H	D	D	LOAD

TWO OR MORE OCTAL SHIFT REGISTERS MAY BE CASCADED TO PROVIDE LARGER SHIFT REGISTERS. RILO AND LIRO ARE LOCATED ON PINS 14 AND 23 RESPECTIVELY, WHICH PROVIDES FOR CONVENIENT INTERCONNECTIONS WHEN TWO OR MORE OCTAL SHIFT REGISTERS ARE CASCADED TO IMPLEMENT LARGER SHIFT REGISTERS.

****

P7021

THIS 4-BIT SHIFT REGISTER/COMPARATOR ACCEPTS POSITIVE NRZ INPUT DATA ON THE RISING EDGE OF THE CLOCK (CK) AND PRODUCES A PARALLEL OUTPUT (B) WITH A 'COMPARE TRUE PULSE' ON THE NEGATIVE EDGE OF THE CLOCK (CLK).

THE THREE-STATE OUTPUTS (B) ARE HIGH-Z WHEN THE OUTPUT CONTROL LINE (/OC) IS HIGH AND ENABLED WHEN THE OUTPUT CONTROL LINE (/OC) IS LOW.

****

P7022

THE 4-BIT COUNTER LOADS A OR B FROM THE MUX, INCREMENTS, OR HOLDS ON THE RISING EDGE OF THE CLOCK.

/OC	CLK	I1	I0	CI	A3-A0	B3-B0	Q3-Q0	OPERATION
H	X	X	X	X	X	X	Z	HI-Z
L	C	L	B	X	X	X	A	BOAD A
L	C	H	L	X	X	B	B	LOAD B
L	C	H	H	L	X	X	Q	HOLD
L	C	H	H	H	X	X	Q PLUS 1	INCREMENT

****

P7023

THIS IS AN 8-BIT SYNCHRONOUS COUNTER WITH PARALLEL LOAD, CLEAR, AND HOLD CAPABILITY. THE LOAD OPERATION LOADS THE INPUTS (D7-D0) INTO THE OUTPUT REGISTER (Q7-Q0). THE CLEAR OPERATION RESETS THE OUTPUT REGISTER TO ALL LOWS. THE HOLD OPERATION HOLDS THE PREVIOUS VALUE REGARDLESS OF CLOCK TRANSITIONS. THE INCREMENT OPERATION ADDS ONE TO THE OUTPUT REGISTER WHEN THE CARRY-IN IS TRUE (/CI=L), OTHERWISE THE OPERATION IS A HOLD. THE CARRY-OUT (/CO) IS TRUE (/CO=L) WHEN THE OUTPUT REGISTER (Q7-Q0) IS ALL HIGHS, OTHERWISE FALSE (/CO=H).

THESE OPERATIONS ARE EXERCISED IN THE FUNCTION TABLE AND SUMMARIZED IN THE OPERATIONS TABLE:

/OC	CLK	I1	I0	/CI	D7-D0	Q7-Q0	OPERATION
H	X	X	X	X	X	Z	HI-Z
L	C	L	L	X	X	L	CLEAR
L	C	L	H	X	X	Q	HOLD
L	C	H	L	X	D	D	LOAD
L	C	H	H	H	X	Q	HOLD
L	C	H	H	L	X	Q PLUS 1	INCREMENT

TWO OR MORE OCTAL COUNTERS MAY BE CASCADED TO PROVIDE LARGER COUNTERS. THE OPERATION CODES WERE CHOSEN SUCH THAT WHEN I1 IS HIGH, I0 MAY BE USED TO SELECT BETWEEN LOAD AND INCREMENT AS IN A PROGRAM COUNTER (JUMP/INCREMENT). ALSO, CARRY-OUT (/CO) AND

CARRY-IN (/CI) ARE LOCATED ON PINS 14 AND 23 RESPECTIVELY, WHICH PROVIDES FOR CONVENIENT INTERCONNECTIONS WHEN TWO OR MORE OCTAL COUNTERS ARE CASCADED TO IMPLEMENT LARGER COUNTERS.

****

P7024

THIS PAL IS AN 8-BIT SYNCHRONOUS UP/DOWN COUNTER WITH PARALLEL LOAD AND HOLD CAPABILITY. THREE FUNCTION SELECT INPUTS (/LD,/UD,/CBI) PROVIDE ONE OF FOUR OPERATIONS WHICH OCCUR SYNCHRONOUSLY ON THE RISING EDGE OF THE CLOCK (CLK). THE LOAD OPERATION LOADS THE INPUTS (D7-D0) INTO THE OUTPUT REGISTERS (Q7-Q0). THE HOLD OPERATION HOLDS THE PREVIOUS VALUE REGARDLESS OF CLOCK TRANSITIONS. THE INCREMENT OPERATION ADDS ONE TO THE OUTPUT REGISTER WHEN CARRY-IN INPUT IS TRUE (/CBI=L), OTHERWISE THE OPERATION IS A HOLD. THE CARRY-OUT (/CBO) IS TRUE (/CBO=L) WHEN THE OUTPUT REGISTER (Q7-Q0) IS ALL HIGHS, OTHERWISE FALSE (/CBO=H). THE DECREMENT OPERATION SUBTRACTS ONE FROM THE OUTPUT REGISTER WHEN THE BORROW-IN INPUT IS TRUE (/CBI=L), OTHERWISE THE OPERATION IS A HOLD. THE BORROW-OUT (/CBO) IS TRUE (/CBO=L) WHEN THE OUTPUT REGISTER (Q7-Q0) IS ALL LOWS, OTHERWISE FALSE (/CBO=H).

THESE OPERATIONS ARE EXERCISED IN THE FUNCTION TABLE AND SUMMARIZED IN THE OPERATIONS TABLE:

/OC	CLK	/LD	/UD	/CBI	D7-D0	Q7-Q0	OPERATION
H	X	X	X	X	X	Z	HI-Z
L	C	L	X	X	D	L	LOAD
L	C	H	L	H	X	Q	HOLD
L	C	H	L	L	X	Q PLUS 1	INCREMENT
L	C	H	H	H	X	Q	HOLD
L	C	H	H	L	X	Q MINUS 1	DECREMENT

THIS OCTAL COUNTER IS IMPLEMENTED WITH A SINGLE PAL20X8. CARRY/BORROW-OUT (/CBO) AND CARRY/BORROW-IN (/CBI) ARE LOCATED ON PINS 14 AND 23 RESPECTIVELY, WHICH PROVIDES FOR CONVENIENT INTERCONNECTIONS WHEN TWO OR MORE OCTAL UP/DOWN COUNTERS ARE CASCADED TO IMPLEMENT LARGER COUNTERS.

****

P7025

THE 10-BIT COUNTER CAN COUNT UP, COUNT DOWN, SET, AND LOAD 2 LSB'S (D0,D1), 2 MSB'S (D8,D9) AND 6 MIDDLE BITS (D2-7) HIGH OR LOW AS A GROUP.

SET OVERRIDES LOAD (/LD), COUNT (/CNT), AND HOLD. LOAD OVERRIDES COUNT. COUNT IS CONDITIONAL ON CARRY IN (/CIN), OTHERWISE IT HOLDS.

THESE OPERATIONS ARE EXERCISED IN THE FUNCTION TABLE AND SUMMARIZED IN THE OPERATIONS TABLE:

/OC	CLK	SET	/LD	/CNT	/CIN	/UP	D9-D0	Q9-Q0	OPERATION
H	X	X	X	X	X	X	X	Z	HI-Z
L	C	H	X	X	X	X	X	H	SET ALL HIGH
L	C	L	L	X	X	X	D	D	LOAD D
L	C	L	H	H	X	X	X	Q	HOLD (/CNT=H)
L	C	L	H	L	H	X	X	Q	HOLD (/CIN=H)
L	C	L	H	L	L	L	X	Q PLUS 1	INCREMENT
L	C	L	H	L	L	H	X	Q MINUS 1	DECREMENT

****
P7026

THE 4-BIT UP/DOWN COUNTER WITH SHIFT REGISTER AND COMPARATOR WILL COUNT UP, COUNT DOWN, SHIFT RIGHT, SHIFT LEFT, COMPARE GREATER THAN OR EQUAL TO, COMPARE LESS THAN OR EQUAL TO, CLEAR, SET, LOAD, OR HOLD AS SPECIFIED BY THE INSTRUCTION LINES (I2,I1,I0) AND CLEAR (CLR). ALL OPERATIONS OCCUR SYNCHRONOUSLY ON THE RISING EDGE OF THE CLOCK (CLK) EXCEPT FOR THE COMPARISION OPERATIONS WHICH ARE PERFORMED ASYNCHRONOUSLY AND WITH NO INSTRUCTON LINES REQUIRED.

THE COMPARISION OPERATIONS (/GE AND /LE) WILL COMPARE THE INPUT DATA (B) WITH THE DATA IN THE REGISTERS (A) AND SUPPLY THE FOLLOWING OUTPUTS:

COMPARISION	/GE	/LE
B IS GREATER THAN A	L	H
B IS EQUAL TO A	L	L
B IS NOT EQUAL TO A	H	H
B IS LESS THAT A	H	L

NOTE THAT BORROW, CARRY, AND SHIFT LEFT AND RIGHT INPUT/OUTPUTS SHARE THE SAME I/O LINES (/LIO AND /RIO) AND THESE LINES ARE INVERTED (ACTIVE LOW).

THESE OPERATIONS ARE EXERCISED IN THE FUNCTION TABLE AND SUMMARIZED IN THE OPERATIONS TABLE BELOW:

/OC	CLK	CLR	I2	I1	I0	B3-B0	/GE	/LE	/LIO	/RIO	A3-A0	OPERATION
H	X	X	X	X	X	X	STATUS		X	X	Z	HI-Z
L	C	H	X	X	X	X	X	X	X	X	L	CLEAR
L	C	L	L	L	L	X	STATUS		X	X	A	HOLD
L	C	L	L	L	H	B	X	X	X	X	B	LOAD B
L	C	L	L	H	L	X	STATUS		RI	A0	SR(RIO)	SHIFT RIGHT
L	C	L	L	H	H	X	X	X	Z	Z	H	SET
L	C	L	H	L	L	X	STATUS		A3	LI	SL(LIO)	SHIFT LEFT
L	C	L	H	L	H	X	X	X	H	Z	H	SET
L	C	L	H	H	L	X	STATUS		COUT	CIN	A PLUS 1	INCREMENT IF CIN
L	C	L	H	H	H	X	STATUS		BOUT	BIN	A MINUS 1	DECREMENT IF BIN

****
P7027

THE 4-BIT FLASH ANALOG-TO-DIGITAL CONVERTER CONVERTS AN ANALOG SIGNAL INTO A 4-BIT GRAY CODE. GRAY CODE IS CHOSEN TO ELIMINATE GLITCHES AT BINARY ROLL OVER POINTS.

THE MAXIUM SAMPLE RATE IS EQUAL TO 1/tpd OF THE PAL20X4. NOTE THAT NO FEEDBACK PROPAGATION DELAY IS INTRODUCED.

****
P7028

THE 4-BIT FLASH DIGITAL-TO-ANALOG CONVERTER CONVERTS A 4-BIT GRAY CODE (G) INTO A 4-BIT BINARY CODE (B), WHICH IS THEN CONVERTED INTO ANALOG OUTPUTS (A).

ANALOG OUTPUTS (A) ARE EITHER LOW OR HI-Z WHICH ALLOWS THE CONDITIONAL THREE-STATE OUTPUTS TO PERFORM THE OPEN COLLECTOR FUNCTION THAT IS NEEDED TO DRIVE THE RESISTOR NETWORK.

****
P7029

THIS PAL PERFORMS THE LOGIC NEEDED TO CONVERT AN 8-BIT DIGITAL SIGNAL INTO A 256 INCREMENT ANALOG SIGNAL.

OUTPUTS ARE EITHER LOW OR HI-Z WHICH ALLOWS THE CONDITIONAL THREE-STATE OUTPUTS TO PERFORM THE OPEN COLLECTOR FUNCTION THAT IS NEEDED TO DRIVE THE RESISTOR NETWORK.

****
P7030

THE OCTAL COMPARATOR ESTABLISHES WHEN TWO 8-BIT DATA STRINGS (A7-A0 AND B7-B0) ARE EQUIVALENT (EQ=H) OR NOT EQUIVALENT (NE=H).

****
P7031

THE DEVICE CONTINUOUSLY COMPARES THE VALUE OF BUS (B3-B0) WITH THE VALUE OF THE REGISTER (A3-A0) AND REPORTS THE STATUS ON OUTPUTS LT, EQ, NE, AND GT:

* LT INDICATES THAT B IS LESS THAN A
* EQ INDICATES THAT B IS EQUAL TO A
* NE INDICATES THAT B IS NOT EQUAL TO A
* GT INDICATES THAT B IS GREATFR THAN A

					STATUS				BUS	REG	
/OC1	/OC2	CLK	LOAD	CLEAR	LT	EQ	NE	GT	B3-B0	A3-A0	OPERATION

```
H       X     X     X       X     X  X  X  X     X       Z      REG HI-Z
X       H     X     X       X     Z  Z  Z  Z     X       X      STATUS HI-Z
L       X     X     L       L     X  X  X  X     X       A      READ REG
X       X     C     H       L     X  X  X  X     B       B      LOAD REG
X       X     C     L       L     X  X  X  X     X       A      HOLD
X       X     C     X       H     X  X  X  X     X       L      CLEAR REG
X       L     X     L       L       STATUS       B       A      COMPARE
--------------------------------------------------------------------------
```

****

P7032

THE BETWEEN LIMITS LOGIC CHECKS THE LT, EQ, AND GT STATUS FROM THE COMPARATOR REGISTERS TO DETERMINE IF THE DATA IS BETWEEN THE UPPER AND LOWER LIMITS LOADED IN THE COMPARATOR REGISTERS. BOTH BETWEEN LIMITS (BTWL) AND OUT OF LIMITS (OUTL) OUTPUTS ARE PROVIDED.

****

P7033

THIS IS A MEMORY DECODER FOR A PRINTER. THE PRINTER IS MAPPED TO HEX ADDRESS 37E8. THE WRITE OR READ LINES GO HIGH WHEN ADDRESS LINES ARE CORRECT AND A WRITE OR READ STROBE RESPECTIVELY IS PRESENT.

THIS IS THE FIRST IC OF A 2-PAL PRINTER INTERFACE FOR THE STANDARD CENTRONICS-TYPE PRINTER.

****

P7034

REGISTERED DATA FROM THE SYSTEM BUSS IS PRESENTED TO THE PRINTER WHEN A WRITE STROBE FROM THE DECODER IS PRESENT (DJPR1).

PRINTER STATUS DATA (PRINTER BUSY AND OUT OF PAPER) IS TRANSFERED TO THE SYSTEM DATA BUSS WHEN A READ STROBE FROM THE DECODER IS PRESENT.

THIS IS THE SECOND IC OF THE 2-PAL PRINTER INTERFACE FOR THE STANDARD CENTRONICS-TYPE PRINTER.

****

P7035

THIS PAL SERVES AS THE MAIN LOGIC UNIT. IT IS THE 8 STATE MACHINE WHICH CONTROLS WHERE WE ARE IN THE GAME. IT ALSO DETECTS THE WIN AND LOSE STATES. PIN /T IS USED TO INITIATE THE GAME TO STATE 000 FOR I.C. TEST EQUIPMENT.

****

P7036

THIS PAL IS A 36 STATE MACHINE USED TO SIMULATE THE EXACT ROLL OF THE DICE. THE STATE MACHINE IS SIMILAR TO THE "ELECTRONIC DICE

GAME".

****

P7037

THIS PAL DECODES THE 36 STATE MACHINE INTO A BINARY REPRESENTATION OF THE ROLL OF THE DICE (2 THRU 12). THERE ARE 36 POSSIBLE COMBINATIONS FOR THE TWO DICE. THE NUMBERS ARE DECODED SO THAT THE PROPER ODDS ARE MAINTAINED FOR THE ROLL OF EACH NUMBER.

****

P7038

THIS PAL IS USED TO STORE THE POINT IN THE GAME. IT ALSO DECODES THE NUMBERS 2, 7, AND 11 TO BE USED IN DETERMINING WIN OR LOST.

****

P7039

THIS PAL IMPLEMENTS A SIMPLE 2 CHANNEL TRAFFIC SIGNAL CONTROLLER. IT IS INTENDED AS AN EXAMPLE IN STATE MACHINE SYNTHESIS.

****

P7040

THIS PAL IS THE SECOND PASS AT THE TRAFFIC SIGNAL CONTROLLER IMPLEMENTATION. THE FOLLOWING SUBSTITUTIONS ARE MADE FOR STATE VARIABLES Q7-Q0:

```
   Q7 = REDA = /REDB        Q5 = GRNA        Q3 = GRNB
   Q6 = YELA                Q4 = YELB        Q2,Q1,Q0 = COUNTER
```

****

P7041

FIRST 8-BIT SHIFT REGISTER AND CHECK.

****

P7042

SECOND 8-BIT SHIFT REGISTER AND CHECK.

****

P7043

THIRD 8-BIT SHIFT REGISTER AND CHECK.

****

P7044

FOURTH 8-BIT SHIFT REGISTER, CHECK, AND SERIAL OUT.

****

P7045

THIS PAL GENERATES THE 4 CHECK BITS IN A 12 BIT HAMMING CODE WORD TO PROVIDE ERROR DETECTION AND CORRECTION ON AN 8 BIT DATA WORD.

****
P7046

THIS PAL GENERATES THE SYNDROME BITS FOR A 12 BIT HAMMING CODE WORD AS A FUNCTION OF THE 8 DATA BITS AND THE 4 CHECK BITS TO POINT TO ANY SINGLE BIT IN ERROR.

****
P7047

THIS PAL PERFORMS ERROR CORRECTION ON BITS B2-B7 BASED ON THE 4-BIT ERROR SYNDROME S0-S3.

****
P7048

THIS PAL PERFORMS ERROR CORRECTION ON BITS B0-B1 AND CHECKS BITS C0-C3 BASED ON THE 4 BIT ERROR SYNDROME S0-S3.

****
P7049

THE DOT GENERATOR PROVIDES THE OSCILLATOR/CLOCK DRIVER, THE DOT SHIFT REGISTER AND THE 3-BIT DOT COUNTER. IT IS LOADED WITH THE 5 DOTS GENERATED BY THE CHARACTER GENERATOR. THESE DOTS ARE SHIFTED OUT THROUGH A SHIFT REGISTER, ONE DOT AT A TIME AND DISPLAYED ON THE SCREEN. "DTCNT" COUNTS UNTIL 8: 5 COUNTS FOR THE CHARACTER AND 3 COUNTS FOR SPACE BETWEEN CHARACTERS.

THE 20X8 IS A REGISTERED PAL. DATA SHOULD BE VALID AND STABLE ON THE INPUT PINS ONE CYCLE BEFORE IT APPEARS ON THE OUTPUT PINS. THE REGISTER IS TRIGGERED ON THE RISING EDGE OF THE CLOCK AND THE DATA IS AVAILABLE ON THE OUTPUT PINS DURING THE NEXT CLOCK CYCLE.

"DTCNT" CAN BE INITIALIZED, CAN COUNT AND CAN HOLD.

"HBLANK" IS A SIGNAL FOR THE END OF ONE DOT LINE (48 CHARACTERS).

"LINE4" IS SET WHEN 16 LINES ARE DISPLAYED ON THE SCREEN.

"SCAN3" IS SET WHEN 7 DOT LINES ARE DISPLAYED.

"DOT" IS A SHIFT REGISTER. IT CAN BE LOADED OR CAN SHIFT LEFT.

WHEN "LINE4" AND/OR "SCAN3" ARE SET, "DOT" SENDS BLANK DOTS TO THE SCREEN FOR GENERATING SPACES BETWEEN CHARACTER LINES AND FOR THE MARGINS OF THE SCREEN. THE OUTPUT FROM THE REGISTER TO THE SCREEN IS THROUGH "DOT4".

****
P7050

"CHAR" AND "CURS" COUNT THE NUMBER OF CHARACTERS PER LINE. THEY ARE ALSO USED AS POINTERS TO THE RAM. "CURS" IS USED WHEN A CHARACTER IS WRITTEN INTO THE RAM AND "CHAR" WHEN A CHARACTER IS READ FROM THE RAM. "CURS" IS ALWAYS POINTED TO THE NEXT AVAILABLE LOCATION IN THE RAM WHERE A NEW CHARACTER CAN BE STORED.

"CHAR" IS INCREMENTED AT THE END OF 8 PIXELS MEANING IT COUNTS AFTER EACH CHARACTER. IT COUNTS FROM 0 TO 63 ALTHOUGH ONLY 48 CHARACTERS ARE VISIBLE. THE HORIZONTAL SYNC PULSE IS GIVEN BETWEEN CHARACTERS 56 AND 59. DURING THE COUNTS OF CHAR FROM 48 TO 63, BLACK SIGNALS ARE TRANSMITTED TO THE SCREEN. "CURS" IS INCREMENTED FOR ANY OPERATION ON THE KEY BOARD. THE TWO POINTERS USE THE RAM IN INTERLIVED FASHION. READ IS DONE EVERY CYCLE, BUT WRITE IS DONE ONLY WHEN A WRITE SIGNAL IS GIVEN. THE WRITE SIGNAL IS SET WHEN A NEW CHARACTER ENTERS THE SYSTEM THROUGH THE RS232 PORT.

THE FUNCTION TABLE ABOVE DESCRIBES OPERATIONS OF READ AND WRITE FOR A CERTAIN LINE. WHEN THIS LINE WAS PRINTED, 64 CHARACTERS WERE READ AND 15 CHARACTERS WERE WRITTEN.

SIGNALS CHAR0, CURS0, SWAPC, WRITE AND INCSCR ARE DERIVED IN PAL VP8.

****
P7051

EACH CHARACTER ON THE SCREEN CONSISTS OF 12 DOT LINES: 7 LINES FOR THE CHARACTER AND 5 LINES FOR SPACE BETWEEN CHARACTERS. THE FOLLOWING FIGURE SHOWS THE LETTER "U" AND THE SPACE WITH ALL THE PIXELS AROUND IT AS IT IS DISPLAYED BY THE VIDEO CONTROLLER.

"SCAN" IS A MODULE 12 COUNTER THAT COUNTS THE NUMBER OF THE DOT LINES FOR EACH CHARACTER.

"LINE" COUNTS THE NUMBER OF THE CHARACTER LINES. EACH CHARACTER LINE IS 12 SCAN LINES. THE COUNTER COUNTS UNTIL 21 ALTHOUGH ONLY 16 LINES ARE VISIBLE ON THE SCREEN.

THE HORIZONTAL SYNC PULSES ARE GIVEN IN EVERY SCANNED LINE BETWEEN CHAR 52 AND 55. THE VERTICAL SYNC PULSE IS GIVEN WHEN THE LINE COUNT IS 19, SCAN IS BETWEEN 0 AND 3, AND CHAR IS BETWEEN 48 AND 51.

THE NEXT FIGURE SHOWS THE SCREEN WITH THE CORRESPONDING LINE AND CHAR COUNTERS, AND THE SYNC PULSES.

****
P7052

"SCROL" AND "LINES" ARE COUNTERS AND POINTERS TO THE RAM. "LINES" IS A POINTER TO THE LINE THAT IS READ FROM THE RAM. "SCROL" IS A POINTER TO THE LOCATION IN THE RAM WHERE A NEW LINE CAN BE STORED. BOTH OF THEM COUNT UP TO A MAXIMUM OF 16 LINES.

THE BIT "SWAP" ENABLES THE TWO COUNTERS TO TALK TO THE SAME ADDRESS LINES OF THE RAM.

THE NEXT FIGURE SHOWS ALL THE POINTERS THAT HAVE BEEN DESCRIBED.

"W" MEANS WRITING INTO THE RAM, "R" MEANS READING FROM THE RAM AND "C" IS A TEMPORARY POINTER.

****

P7053

THIS PAL GENERATES THE BAUD RATE, THE VIDEO AND THE SYNC SIGNALS WHICH ARE COMBINED AT THE OUTPUTS TO FORM THE COMPOSITE VIDEO SIGNAL, AND THE "UEN" SIGNAL WHICH ENABLE THE "UART".

EVERY CHARACTER CONSISTS OF 10 BITS: 1 START BIT, 7 ASCII CODE BITS, 1 PARITY BIT, AND 1 STOP BIT. THE CHARACTER RATE IS 9600 Hz. EACH BIT IS DIVIDED INTO 8 SMALL BITS SO THE NUMBER OF BITS PER SECOND REQUIRED FOR OUR SYSTEM IS 9600*8 = 76800 OR 76800 Hz.

T = 1/76800 = 13 MICROSECOND
THE CLOCK FREQUENCY IS 8000000 Hz.
WE NEED TO DIVIDE THE CLOCK FREQUENCY BY 104 TO GENERATE A FREQUENCY OF 76800 Hz.

8000000/76800 = 104 = 13*8

"DTCNT" COUNTS 8, AND "BAUD" COUNTS 13. TO GET 104 COUNTS WE NEED TO COUNT FROM 0 TO 103. BECAUSE THERE IS ONE CLOCK CYCLE DELAY UNTIL THE DATA IS AVAILABLE ON THE OUTPUT PINS (REGISTERED PAL), MODULE 104 IS DETECTED BY COUNT 102 WHICH IS EQUAL TO 102/8 = 12 6/8.

****

P7054

THE "UART" SHIFT REGISTER IS A SEVEN BIT REGISTER FOR THE SEVEN BIT ASCII CODE. THE INFORMATION ENTERS THE SHIFT REGISTER IN D6, ONE BIT AT A TIME. IT COMES THROUGH RXD PIN WHICH IS THE TRANSMIT OR THE RECEIVE LINE OF THE RS232. THE OUTPUTS ARE TRANSFERED IN PARALLEL TO THE RAM. "UEN" ENABLES TREE STATE FOR THESE OUTPUTS. WHEN BITS D6 AND D5 TOGETHER IN THE ASCII CODE ARE ZEROES OR WHEN THE "CLRLIN" BIT IS SET, A "SPACE" CODE IS SHIFTED INTO THE "UART" REGISTER. THE SPACE CODE PRINTS A BLANK SPACE ON THE SCREEN. "SPACE" IN ASCII CODE IS 0100000 = 20 HEX.

****

P7055

"BC" IS A COUNTER FOR THE ASCII CODE BITS. A SAMPLE FROM EACH ASCII BIT IS TAKEN WHEN "DET" = 3.

THE "START" SIGNAL IS SET WHEN A START BIT IS DETECTED ON THE RXD LINE AND REMAINS SET UNTIL THE LAST ASCII BIT OF THE CHARACTER

THAT IS SAMPLED. AFTER ALL THE 7 BITS OF THE ASCII CODE WERE SAMPLED, A WRITE SIGNAL IS SENT TO THE RAM AND THE 7 BITS (A CODE FOR A CHARACTER) ARE WRITTEN IN PARALLEL INTO THE RAM.

WHEN "BC" COUNTS 8 THE "READY" SIGNAL GOES HIGH, "START" GOES LOW AND READY TO BE SET AGAIN IF A NEW START BIT IS DETECTED. A NEW CODE FOR A NEW CHARACTER WILL START TO BE SAMPLED.

BUT ASSUME THAT A NOISE OCCURS ON THE RXD LINE WHICH SET THE "START" SIGNAL.

TO DETERMINE IF THIS IS A TRUE OR FALSE START, WE CHECK THE RXD LINE. IF THE RXD LINE IS NOT SET, WE KNOW THAT NO CHARACTER WAS SENT ON THE RS232 LINE; THEREFORE, A NOISE/FALSE SIGNAL WAS DETECTED.

ERROR ANALYSIS FOR SAMPLING
==========================

SINCE BOTH A TRANSMITTER AND A RECEIVER ARE USED IN OBTAINING INFORMATION, AN ERROR CAN OCCUR THAT INVOLVES BOTH OF THESE COMPONENTS. ASSUME: ERROR IN TRANSMITTING FREQUENCY = EX
ERROR IN RECEIVING FREQUENCY = ER
THEN THE TOTAL ERROR FOR ONE BIT OF INFORMATION IS (EX + ER) AND THE TOTAL ERROR FOR THE N'TH BIT OF INFORMATION IS N*(EX + ER).

WHEN COUNTER "DET" IS EQUAL 3, WE HAVE THE IDEAL BIT FOR SAMPLING. THE MAXIMUM ERROR THAT IS ALLOWED DUE TO THE TOLERANCES IN THE FREQUENCIES IN BOTH TRANSMITTER AND RECEIVER, WILL BE WHEN THE SAMPLE IS TAKEN AT "DET" = 0 OR AT "DET" = 6, AS SHOWN HERE:

THE FREQUENCY FOR EACH BIT OF INFORMATION IS 1/9600 Hz (104 MICROSECOND). EACH BIT OF INFORMATION IS DIVIDED INTO EIGHT TIME SLOTS.

THE TOTAL ERROR ALLOWED FOR THE 7'TH BIT OF INFORMATION IS:

7 * (EX + ER) = (1/9600) * (3/8)

THE TOTAL ERROR ALLOWED FOR ONE BIT OF INFORMATION IS:

(EX + ER) = 104 * (3/8) * (1/7) = 5.5 MICROSECOND

THE FREQUENCY RATE BETWEEN RS232 AND THE RXD LINE FOR EVERY BIT OF INFORMATION SHOULD BE BETWEEN 10150 Hz AND 9130 Hz.

****

P7056

THIS PAL CONTROLS BOTH THE TIMING FOR WRITING A CHARACTER INTO THE RAM AND THE TIMING FOR READING A CHARACTER FROM THE RAM.

"WRITE" IS A SIGNAL THAT DATA IS AVAILABLE AND CAN BE WRITTEN INTO THE RAM WHEN "DTCNT" IS BETWEEN 0 AND 5.

"SWAP" AND "SWAPC" SWAP POINTERS TO ENABLE WRITING AND READING BY ADDRESSING THE RAM THROUGH THE SAME ADDRESS LINES.

"UEN" ENABLES THE UART.

"WE" WRITES TO THE RAM THE DATA THAT IS IN THE UART REGISTER.

"INCSCR" IS A SIGNAL THAT DETECT LINEFEED AND 48 CHARACTERS PER LINE.

"CLRLIN" ERASES EARLIER INFORMATION AND ALLOWS TO WRITE TO THAT LOCATION. WHEN THE "LF" KEY IS PUSHED, THE CLRLIN SIGNAL IS SET. IT REMAINS SET AS LONG AS A WRITE IS DONE TO THIS LOCATION.

"H255" DETECTS END OF LINE.

"HBLANK" BLANK OUT THE SCREEN WHEN "CHAR" IS BETWEEN 48 AND 63.

****

P7057

THIS IS A 4-BIT SERIAL SWITCH WHICH CONNECTS ANY INPUT (D) TO ANY OR ALL OUTPUTS (Y) IN ANY COMBINATION. A STROBE LINE (S) IS PROVIDED TO GATE THE OUTPUTS OFF (Y=H) WHEN THE STROBE INPUT IS HIGH. THE SWITCH LINES (S3-0) ARE ENCODED IN BINARY WITH S-0 REPRESENTING THE LSB.

****

P7058

THE 4-BIT SHIFTER ACCEPTS A 4-BIT DATA WORD (D) AND SHIFTS THE WORD 0, 1, 2, OR 3 PLACES TO OUTPUTS (Y). THE NUMBER OF PLACES TO BE SHIFTED IS DETERMINED BY TWO CONTROL LINES (I1,I0) WHICH ARE ENCODED IN BINARY WITH I0 REPRESENTING THE LSB. A STROBE LINE (S) IS PROVIDED TO GATE THE OUTPUTS OFF (Y=H) WHEN THE STROBE INPUT IS HIGH.

OPERATIONS TABLE:

S	I1	I0	D3-D-3	Y3-Y0	OPERATION
H	X	X	X	H	STROBE HIGH
L	L	L	D	D	NO SHIFT
L	L	H	D	S(D)1	SHIFT 1 PLACE
L	H	L	D	S(D)2	SHIFT 2 PLACES
L	H	H	D	S(D)3	SHIFT 3 PLACES

TWO OR MORE 4-BIT SHIFTERS MAY BE CONNECTED TO IMPLEMENT LARGER SHIFTERS. SHIFTING CAN BE LOGICAL, WITH ZEROES PULLED IN AT EITHER OR BOTH ENDS OF THE SHIFTING FIELD; ARITHMETIC, WHERE THE SIGN BIT IS REPEATED DURING A SHIFT DOWN; OR END AROUND, WHERE THE DATA WORD FORMS A CONTINUOUS LOOP.

****

P7059

THIS IS AN 8-BIT SYNCHRONOUS DOWN COUNTER WITH PARALLEL LOAD, PRESET, AND HOLD CAPABILITY. THE LOAD OPERATION LOADS THE INPUTS (D7-D0) INTO THE OUTPUT REGISTER (Q7-Q0). THE PRESET OPERATION SETS THE OUTPUT REGISTER TO ALL HIGHS. THE HOLD OPERATION HOLDS THE PREVIOUS VALUE REGARDLESS OF CLOCK TRANSITIONS. THE DECREMENT OPERATION SUBTRACTS ONE FROM THE OUTPUT REGISTER WHEN THE BORROW IN IS TRUE (/BI=L), OTHERWISE THE OPERATION IS A HOLD. THE BORROW-OUT (/BO) IS TRUE (/BO=L) WHEN THE OUTPUT REGISTER (Q7-Q0) IS ALL LOWS, OTHERWISE FALSE (/BO=H).

THESE OPERATIONS ARE EXERCISED IN THE FUNCTION TABLE AND SUMMARIZED IN THE OPERATIONS TABLE:

/OC	CLK	I1	I0	/BI	D7-D0	Q7-Q0	OPERATION
H	X	X	X	X	X	Z	HI-Z
L	C	L	L	X	X	H	PRESET
L	C	L	H	X	X	Q	HOLD
L	C	H	L	X	D	D	LOAD
L	C	H	H	H	X	Q	HOLD
L	C	H	H	L	X	Q MINUS 1	DECREMENT

TWO OR MORE OCTAL DOWN COUNTERS MAY BE CASCADED TO PROVIDE LARGER COUNTERS. THE OPERATION CODES WERE CHOSEN SUCH THAT WHEN I1 IS HIGH, I0 MAY BE USED TO SELECT BETWEEN LOAD AND DECREMENT AS IN A BUS ADDRESSER (JUMP/DECREMENT). ALSO BORROW-OUT (/BO) AND BORROW-IN (/BI) ARE LOCATED ON PINS 14 AND 23 RESPECTIVELY WHICH PROVIDES FOR CONVENIENT INTERCONNECTIONS WHEN TWO OR MORE OCTAL DOWN COUNTERS ARE CASCADED TO IMPLEMENT LARGER DOWN COUNTERS.

****

P7060

THIS PAL DESIGN SPECIFICATION DESCRIBES A 4-BIT COUNTER WITH A 4-BIT TRANS-PARENT LATCH. THE LATCH OUTPUTS (Y3-YC) WILL FOLLOW EITHER THE DATA INPUTS (D3-D0) OR THE COUNTER STATES (Q3-Q0), DEPENDING ON THE SELECT LINE (SEL), PROVIDING THE LATCH ENABLE IS TRUE (/LEN=L). THE OUTPUTS WILL BE LATCHED AT THE LEVEL OF THE INPUTS WHICH WERE SELECTED WHEN THE LATCH ENABLE IS FALSE (/LEN=H).

THE COUNTER IS FULLY SYNCHRONOUS WITH THE RISING EDGE OF THE CLOCK (CLK) AND CAN BE SYNCHRONOUSLY LOADED FROM THE LATCH OUTPUTS. A SYNCHRONOUS OVERRIDING CLEAR (/CLR) IS ALSO PROVIDED.

OPERATIONS TABLE:

	-COUNTER	CONTROL-			--LATCH	CONTROL--			DATA	CNTR	LATCH	
CLK	/OC	/CLR	/LD	/CNT	/LOC	/LEN	SEL	/CI	D3-D0	Q3-Q0	Y3-Y0	OPERATION
X	H	X	X	X	X	X	X	X	X	Z	X	HI-Z (COUNTER
C	L	L	X	X	X	X	X	X	X	L	X	CLEAR

C	L	H	L	X	L	H	X	X	X	Y	Y	LOAD FROM LATCH
C	L	H	L	X	L	L	L	X	D	D	D	LOAD FROM DATA
C	L	H	H	H	X	X	X	X	X	Q	X	HOLD
C	L	L	L	X	D	D	D	LOAD FROM DATA				
C	L	H	H	H	X	X	X	X	X	Q	X	HOLD
C	L	H	H	L	X	X	X	L	X	Q PLUS 1	X	INCREMENT
C	L	H	H	L	X	X	X	H	X	Q	X	HOLD
X	X	X	X	X	H	X	X	X	X	X	Z	HI-Z (LATCH)
X	X	X	X	X	L	H	X	X	X	X	Y	LATCH OUTPUT
X	X	X	X	X	L	L	L	X	D	X	D	SELECT DATA
X	X	X	X	X	L	L	H	X	X	Q	Q	SELECT COUNTER

****

P7061

THIS 10-BIT REGISTER LOADS THE DATA (D9-D0) ON THE RISING EDGE OF THE CLOCK (CLK) INTO THE REGISTER (Q9-Q0). THE DATA IS HELD IN THE REGISTER UNTIL THE NEXT POSITIVE EDGE OF THE CLOCK.

OPERATIONS TABLE:

/OC	CLK	D9-D0	Q9-Q0	OPERATION
H	X	X	Z	HI-Z
L	C	D	D	LOAD
L	L	X	Q	HOLD

****

P7062

THE 3-TO-8 DEMULTIPLEXER WITH CONTROL STORAGE PROVIDES A CONVENTIONAL 8-BIT DEMUX FUNCTION COMBINED WITH CONTROL STORAGE FUNCTIONS: LOAD TRUE, LOAD COMPLEMENT, HOLD, TOGGLE POLARITY, CLEAR, AND PRESET. FIVE INPUTS (/LD, /CLR, /PR, POL, TOG) SELECT ONE OF SIX OPERATIONS WHICH OCCUR SYNCHRONOUSLY WITH THE RISING EDGE OF THE CLOCK (CLK).

THE LOAD TRUE OPERATION LOADS THE DECODED BINARY INPUTS (A,B,C) INTO THE OUTPUT REGISTER (Q7-Q0) WHEN POLARITY IS TRUE (POL=H). THE COMPLEMENT OF THE BINARY INPUTS IS DECODED AND LOADED INTO THE OUTPUT REGISTER WHEN POLARITY IS FALSE (POL=L).

THE HOLD OPERATION HOLDS THE PREVIOUS VALUE IN THE REGISTER WHEN TOGGLE IS FALSE (TOG=L) REGARDLESS OF CLOCK TRANSITIONS. THE TOGGLE POLARITY OPERATION TOGGLES THE POLARITY OF THE DATA IN THE OUTPUT REGISTER WHEN TOGGLE IS TRUE (TOG=H).

THE CLEAR (/CLR) OPERATION RESETS THE OUTPUT REGISTER TO ALL LOWS. THE PRESET (/PR) OPERATION PRESETS THE OUTPUT REGISTER TO ALL HIGHS. NOTE THAT CLEAR OVERRIDES PRESET, PRESET OVERRIDES LOAD, AND LOAD OVERRIDES HOLD.

THE POLARITY OF "POL" MAY BE CHANGED IN THE LOGIC EQUATIONS TO SUIT SPECIFIC APPLICATIONS SO THAT CERTAIN OUTPUT POLARITIES ARE ASSERTIVE HIGH WHILE OTHERS ARE ASSERTIVE LOW.

THESE OPERATIONS ARE EXERCISED IN THE FUNCTION TABLE AND SUMMARIZED IN THE OPERATIONS TABLE:

CONTROL		FUNCTIONS			POLARITY		INPUTS	OUTPUTS	
/OC	CLK	/CLR	/PR	/LD	POL	TOG	ABC	Q7-Q0	OPERATION
H	X	X	X	X	X	X	X	Z	HI-Z
L	C	L	X	X	X	X	X	L	CLEAR
L	C	H	L	X	X	X	X	H	PRESET
L	C	H	H	L	H	X	I	MUX	LOAD TRUE
L	C	H	H	L	L	X	I	/MUX	LOAD COMP
L	C	H	H	H	X	L	X	Q	HOLD
L	C	H	H	H	X	H	X	/Q	TOGGLE POLARITY

****

P7063

THE 9-BIT SYNCHRONOUS COUNTER HAS PARALLEL LOAD, INCREMENT, AND HOLD CAPABILITIES. DATA (D8-D0) IS LOADED INTO THE OUTPUT REGISTER (Q8-Q0) WHEN THE LOAD INPUT IS TRUE (/LD=L) AND A POSITIVE EDGE PULSE IS RECEIVED ON THE CLOCK PIN (CLK). THE COUNTER WILL INCREMENT IF A CLOCK PULSE IS RECEIVED WITH THE LOAD INPUT BEING FALSE (/LD=H). THE OPERATION IS A HOLD IF NO CLOCK PULSE IS RECEIVED REGARDLESS OF ANY OTHER INPUTS.

THE CARRY OUT PIN (/CO) SHOWS HOW TO IMPLEMENT A CARRY OUT USING A REGISTER BY ANTICIPATED ONE COUNT BEFORE THE TERMINAL COUNT IF COUNTING AND THE TERMINAL COUNT IF LOADING.

OPERATIONS TABLE:

/OC	CLK	/LD	D8-D0	Q8-Q0	OPERATION
H	X	X	X	Z	HI-Z
L	L	X	X	Q	HOLD
L	C	L	D	D	LOAD
L	C	H	X	Q PLUS 1	INCREMENT

****

P7064

THE 10-BIT COMPARATOR ESTABLISHES WHEN TWO 10-BIT DATA STRINGS (A9-B0 AND B9-B0) ARE EQUIVALENT (EQ=H) OR NOT EQUIVALENT (NE=H).

****

P7065

THIS PAL IS A 6-BIT SHIFT REGISTER WITH PARALLEL LOAD AND HOLD CAPABILITY. TWO FUNCTION SELECT INPUT (I0,I1) PROVIDE ONE OF FOUR OPERATIONS WHICH OCCUR SYNCHRONOUSLY ON THE RISING EDGE OF THE

CLOCK (CLK). THESE OPERATIONS ARE:

/OC	CLK	I1	I0	D5-D0	Q5-Q0	OPERATION
H	X	X	X	X	Z	HI-Z
L	C	L	L	X	L	HOLD
L	C	L	H	X	SR(Q)	SHIFT RIGHT
L	C	H	L	D	SL(Q)	SHIFT LEFT
L	C	H	H	D	D	LOAD

TWO OR MORE 6-BIT SHIFT REGISTERS MAY BE CASCADED TO PROVIDE LARGER SHIFT REGISTERS. RILO AND LIRO ARE LOCATED ON PINS 12 AND 19 RESPECTIVELY, WHICH PROVIDES FOR CONVENIENT INTERCONNECTIONS WHEN TWO OR MORE 6-BIT SHIFT REGISTERS ARE CASCADED TO IMPLEMENT LARGER SHIFT REGISTERS.

****

P7066

THIS IS AN 6-BIT SYNCHRONOUS COUNTER WITH PARALLEL LOAD, PRESET, AND HOLD CAPABILITY. THE LOAD OPERATION LOADS THE INPUTS (D5-D0) INTO THE OUTPUT REGISTER (Q5-Q0). THE PRESET OPERATION SETS THE OUTPUT REGISTER TO ALL HIGHS. THE HOLD OPERATION HOLDS THE PREVIOUS VALUE REGARDLESS OF CLOCK TRANSITIONS. THE INCREMENT OPERATION ADDS ONE TO THE OUTPUT REGISTER WHEN THE CARRY-IN IS TRUE (/CI=L), OTHERWISE THE OPERATION IS A HOLD. THE CARRY-OUT (/CO) IS TRUE (/CO=L) WHEN THE OUTPUT REGISTER (Q5-Q0) IS ALL HIGHS, OTHERWISE FALSE (/CO=H).

THESE OPERATIONS ARE EXERCISED IN THE FUNCTION TABLE AND SUMMARIZED IN THE OPERATIONS TABLE:

/OC	CLK	I1	I0	/CI	D5-D0	Q5-Q0	OPERATION
H	X	X	X	X	X	Z	HI-Z
L	C	L	L	X	X	H	PRESET
L	C	L	H	X	X	Q	HOLD
L	C	H	L	X	D	D	LOAD
L	C	H	H	H	X	Q	HOLD
L	C	H	H	L	X	Q PLUS 1	INCREMENT

TWO OR MORE 6-BIT COUNTERS MAY BE CASCADED TO PROVIDE LARGER COUNTERS. THE OPERATION CODES WERE CHOSEN SUCH THAT WHEN I1 IS HIGH, I0 MAY BE USED TO SELECT BETWEEN LOAD AND INCREMENT AS IN A PROGRAM COUNTER (JUMP/INCREMENT). ALSO, CARRY-OUT (/CO) AND CARRY-IN (/CI) ARE LOCATED ON PINS 12 AND 19 RESPECTIVELY, WHICH PROVIDES FOR CONVENIENT INTERCONNECTIONS WHEN TWO OR MORE 6-BIT COUNTERS ARE CASCADED TO IMPLEMENT LARGER COUNTERS.

****

P7067

THIS IS AN 6-BIT SYNCHRONOUS REGISTER WITH PARALLEL LOAD, LOAD

COMPLEMENT, PRESET, CLEAR, AND HOLD CAPABILITIES. FOUR CONTROL INPUTS (/LD,POL,/CLR,/PR) PROVIDE ONE OF FOUR OPERATIONS WHICH OCCUR SYNCHRONOUSLY WITH THE CLOCK (CLK). THE LOAD OPERATION LOADS THE INPUTS (D5-D0) INTO THE OUTPUT REGISTER (Q5-Q0), WHEN POL=H OR LOADS THE COMPLEMENT OF THE INPUTS WHEN POL=L. THE CLEAR (/CLR) OPERATION RESETS THE OUTPUT REGISTERS TO ALL LOWS. THE PRESET (/PR) OPERATION PRESETS THE OUTPUT REGISTERS TO ALL HIGHS. THE HOLD OPERATION HOLDS THE PREVIOUS VALUE REGARDLESS OF CLOCK TRANSITIONS.

CLEAR OVERRIDES PRESET, PRESET OVERRIDES LOAD, AND LOAD OVERRIDES HOLD.

THESE OPERATIONS ARE EXERCISED IN THE FUNCTION TABLE AND SUMMARIZED IN THE OPERATIONS TABLE:

/OC	CLK	/CLR	/PR	/LD	POL	D5-D0	Q5-Q0	OPERATION
H	X	X	X	X	X	X	Z	HI-Z
L	C	L	X	X	X	X	L	CLEAR
L	C	H	L	L	X	X	H	PRESET
L	C	H	H	H	X	X	Q	HOLD
L	C	H	H	L	H	D	D	LOAD TRUE
L	C	H	H	L	L	D	/D	LOAD COMP

****

P7068

THIS IS A 9-BIT MULTIFUNCTION BUFFER WITH PARALLEL TRUE OUTPUT, COMPLEMENTARY OUTPUT, CLEAR, AND PRESET CAPABILITIES. THREE CONTROL INPUTS (POL,/CLR,/PR) PROVIDE ONE OF THREE OPERATIONS.

THE TRUE DATA WILL BE AVAILABLE AT THE OUTPUTS (Y=D) WHEN THE POLARITY LINE IS HIGH (POL=H). THE INPUT DATA WILL BE INVERTED (Y=/D) WHEN THE POLARITY LINE IS LOW (POL=L).

THE OUTPUTS WILL BE LOW (Y=L) WHEN THE CLEAR LINE IS LOW (/CLR=L) AND HIGH (Y=H) WHEN THE PRESET LINE IS LOW (/PR=L). NOTE THAT THESE OPERATIONS OVERRIDE DATA INPUTS. ALSO CLEAR OVERRIDES PRESET.

THE NINE OUTPUTS WILL BE HI-Z (Y=Z) IF THE OUTPUT CONTROL LINE IS HIGH (/OC=H) REGARDLESS OF OTHER CONTROL LINES AND INPUTS.

THESE OPERATIONS ARE EXERCISED IN THE FUNCTION TABLE AND SUMMARIZED IN THE OPERATIONS TABLE:

/OC	/CLR	/PR	POL	D8-D0	Y8-Y0	OPERATION
H	X	X	X	X	Z	HI-Z
L	L	X	X	X	L	CLEAR
L	H	L	X	X	H	PRESET
L	H	H	H	D	D	OUTPUT TRUE
L	H	H	L	D	/D	OUTPUT COMP

***

P7069

THE 10-BIT BUFFER CAN OUTPUT TRUE DATA, COMPLEMENTARY DATA, OR BE HIGH INPEDANCE. THE TRUE DATA WILL BE AVAILABLE AT THE OUTPUTS (Y=D) WHEN THE POLARITY LINE IS HIGH (POL=H). THE INPUT DATA WILL BE INVERTED (Y=/D) WHEN THE POLARITY LINE IS LOW (POL=L).

THE TEN OUTPUTS WILL BE HI-Z (Y=Z) IF THE OUTPUT CONTROL LINE IS HIGH (/OC=H) REGARDLESS OF OTHER INPUTS.

THESE OPERATIONS ARE EXERCISED IN THE FUNCTION TABLE AND SUMMARIZED IN THE OPERATIONS TABLE:

/OC	POL	D9-D0	Y9-Y0	OPERATION
H	X	X	Z	HI-Z
L	H	D	D	OUTPUT TRUE
L	L	D	/D	OUTPUT COMP

***

P7070

THE 10-BIT OPEN COLLECTOR BUFFER WILL OUTPUT THE INPUT DATA (D). THE OUTPUTS (Y) WILL BE EITHER L OR HI-Z.

CERTAIN OUTPUTS WILL BE HIGH-Z (Y=Z) IF EITHER OUTPUT CONTROL LINE IS HIGH (/OC=H) REGARDLESS OF OTHER INPUTS. NOTE THAT OC2 CONTROLS OUTPUTS Y9-Y5 AND OC1 CONTROLS OUTPUTS Y4-Y0. OC2 AND OC1 CONTROL INDEPENDENTLY.

OPERATIONS TABLE:

/OC2	/OC1	D9-D0	Y9-Y5	Y4-Y0	OPERATION
H	X	X	Z	X	HI-Z FOR UPPER 5 BITS
X	H	X	X	Z	HI-Z FOR LOWER 5 BITS
L	L	D	D	D	OUTPUT TRUE (L or HI-Z)

***

P7071

THE 10-BIT INVERTING OPEN COLLECTOR BUFFER WILL OUTPUT THE COMPLEMENT OF THE INPUT DATA (D). THE OUTPUTS (Y) WILL BE EITHER L OR HI-Z.

CERTAIN OUTPUTS WILL BE HIGH-Z (Y=Z) IF EITHER OUTPUT CONTROL LINE IS HIGH (/OC=H) REGARDLESS OF OTHER INPUTS. NOTE THAT OC2 CONTROLS OUTPUTS Y9-Y5 AND OC1 CONTROLS OUTPUTS Y4-Y0. OC2 AND OC1 CONTROL INDEPENDENTLY.

OPERATIONS TABLE:

/OC2	/OC1	D9-D0	Y9-Y5	Y4-Y0	OPERATION
H	X	X	Z	X	HI-Z FOR UPPER 5 BITS
X	H	X	X	Z	HI-Z FOR LOWER 5 BITS
L	L	D	/D	/D	OUTPUT COMP (L or HI-Z)

***

P7072

THE 10-BIT ADDRESSABLE REGISTER IS A SYNCHRONOUS GENERAL PURPOSE ADDRESSABLE REGISTER WITH CLEAR, PRESET, AND ENABLE. THE OUTPUT REGISTER (Q) IS SELECTED BY THE INPUT ADDRESS PINS (A,B,C,D). THE DATA (DIN) IS LOADED INTO THE SELECTED OUTPUT REGISTER ON THE RISING EDGE OF THE CLOCK (CLK) IF THE CHIP IS ENABLED (E1=HIGH,E2=HIGH,/E3=LOW). ALL OTHER OUTPUTS HOLD THEIR PREVIOUS STATES DURING THE LOAD OPERATION. ANY OTHER COMBINATION OF THE ENABLE PINS (E1,E2,/E3) WILL DISABLE THE REGISTER AND ALL OUTPUTS WILL HOLD THEIR PREVIOUS STATES. CLEAR (/CLR) AND PRESET (/PR) ARE ACTIVE LOW PINS WHICH SET THE REGISTERS TO ALL HIGH OR LOW RESPECTIVELY.

CLEAR OVERRIDES PRESET AND ENABLE, PRESET OVERRIDES ENABLE.

THESE FUNCTIONS ARE EXERCISED IN THE FUNCTION TABLE AND SUMMARIZED IN THE OPERATIONS TABLE:

/OC	CLK	/CLR	/PR	/E3	E2	E1	D	C	B	A	DIN	Q9-Q0	OPERATION
H	X	X	X	X	X	X	X	X	X	X	X	Z	HI-Z
L	C	L	L	X	X	X	X	X	X	X	X	L	CLEAR
L	C	H	L	X	X	X	X	X	X	X	X	H	PRESET
L	C	H	H	L	L	L	X	X	X	X	X	Q	HOLD PREVIOUS STATES
L	C	H	H	L	L	H	X	X	X	X	X	Q	HOLD PREVIOUS STATES
L	C	H	H	L	H	L	X	X	X	X	X	Q	HOLD PREVIOUS STATES
L	C	H	H	L	H	H	D	C	B	A	DIN	DIN	LOAD DIN TO ADDRESSED OUTPUT
L	C	H	H	H	L	L	X	X	X	X	X	Q	HOLD PREVIOUS STATES
L	C	H	H	H	L	H	X	X	X	X	X	Q	HOLD PREVIOUS STATES
L	C	H	H	H	H	L	X	X	X	X	X	Q	HOLD PREVIOUS STATES
L	C	H	H	H	H	H	X	X	X	X	X	Q	HOLD PREVIOUS STATES

OUTPUT SELECT TABLE

D	C	B	A	DIN	Q9	Q8	Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0
L	L	L	L	DIN	Q9	Q8	Q7	Q6	Q5	Q4	Q3	Q2	Q1	DIN
L	L	L	H	DIN	Q9	Q8	Q7	Q6	Q5	Q4	Q3	Q2	DIN	Q0
L	L	H	L	DIN	Q9	Q8	Q7	Q6	Q5	Q4	Q3	DIN	Q1	Q0
L	L	H	H	DIN	Q9	Q8	Q7	Q6	Q5	Q4	DIN	Q2	Q1	Q0
L	H	L	L	DIN	Q9	Q8	Q7	Q6	Q5	DIN	Q3	Q2	Q1	Q0
L	H	L	H	DIN	Q9	Q8	Q7	Q6	DIN	Q4	Q3	Q2	Q1	Q0

L	H	H	L	DIN	Q9	Q8	Q7	DIN	Q5	Q4	Q3	Q2	Q1	Q0
L	H	H	H	DIN	Q9	Q8	DIN	Q6	Q5	Q4	Q3	Q2	Q1	Q0
H	L	L	L	DIN	Q9	DIN	Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0
H	L	L	H	DIN	DIN	Q8	Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0
H	L	H	L	DIN	Q9	Q8	Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0
H	L	H	H	DIN	Q9	Q8	Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0
H	H	X	X	DIN	Q9	Q8	Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0

****

P7073

THIS PAL20L10 INTERFACES BETWEEN THE MOTOROLA MC6800 MICROPROCESSOR AND ITS SYSTEM COMPONENTS ON A SINGLE BOARD COMPUTER. THE FUNCTIONS IT PERFORMS, PREVIOUSLY DONE WITH RANDOM LOGIC ARE: ADDRESS DECODING, MEMORY AND I/O SELECT, RESET SIGNAL GENERATION, AND CONTROL OF THE BUFFER WHICH INTERFACES THE DATA BUS TO OTHER BOARDS IN THE SYSTEM.

****

P7074

THIS IS AN EXAMPLE OF A QUAD 3-TO-1 MULTIPLEXER USING A PAL18L4. SELECT LINES A,B ARE ENCODED IN BINARY, WITH A REPRESENTING THE LSB. THE OUTPUTS (Y) ARE ALL HIGH IF THE SELECT LINES ARE BOTH HIGH (B,A=H).

OPERATIONS TABLE:

INPUT SELECT		OUTPUTS
B	A	Y
L	L	C0
L	H	C1
H	L	C2
H	H	H

****

P7075

THIS PAL DESIGN SPECIFICATION DESCRIBES A 4-BIT SYNCHRONOUS COUNTER WITH 4-BIT REGISTER. DATA CAN BE LOADED TO THE COUNTER FROM THE REGISTER. IT CAN ALSO BE SYNCHRONOUSLY CLEARED. THE REGISTER CAN BE LOADED FROM EITHER THE COUNTER OR THE DATA INPUTS UNDER CONTROL OF THE SEL INPUT. THE REGISTER CAN ALSO BE SYNCHRONOUSLY CLEARED. THE COUNTER AND REGISTER HAVE A COMMON CLOCK FOR SYNCHRONOUS OPERATION.

****

P7076

THE 9-BIT SYNCHRONOUS COUNTER HAS PARALLEL LOAD, DECREMENT, AND HOLD CAPABILITIES. DATA (D8-D0) IS LOADED INTO THE OUTPUT

REGISTER (Q8-Q0) WHEN THE LOAD INPUT IS TRUE (/LD=L) AND A POSITIVE EDGE PULSE IS RECEIVED ON THE CLOCK PIN (CLK). THE COUNTER WILL DECREMENT IF A CLOCK PULSE IS RECEIVED WITH THE LOAD INPUT BEING FALSE (/LD=H). THE OPERATION IS A HOLD IF NO CLOCK PULSE IS RECEIVED REGARDLESS OF ANY OTHER INPUTS.

THE BORROW OUT PIN (/BO) SHOWS HOW TO IMPLEMENT A BORROW OUT USING A REGISTER BY ANTICIPATED ONE COUNT BEFORE THE TERMINAL COUNT IF COUNTING AND THE TERMINAL COUNT IF LOADING.

OPERATIONS TABLE:

/OC	CLK	/LD	D8-D0	Q8-Q0	OPERATION
H	X	X	X	Z	HI-Z
L	L	X	X	/Q	HOLD
L	C	L	D	D	LOAD
L	C	H	X	Q MINUS 1	DECREMENT

****

P7077

THE REFRESH CLOCK GENERATOR CAN GENERATE REFRESH CLOCK (RFCK) FOR THE DYNAMIC RAM CONTROLLERS. THE PERIOD OF RFCK DEPENDS ON F3-F0 WHILE THE DURATION OF RFCK BEING LOW DEPENDS ON M3-M0.

FFFF 3210	RFCK PERIOD (CYCLES)
0000	12
0001	28
0010	44
0011	60
0100	76
0101	92
0110	108
0111	124
1000	140
1001	156
1010	172
1011	188
1100	204
1101	220
1110	236
1111	252

MMMM 3210	RFCK LOW DURATION (CYCLES)
0000	0
0001	4
0010	8
0011	12
0100	16
0101	20*
0110	24
0111	28
1000	32
1001	36*
1010	40*
1011	44*
1100	48
1101	52*
1110	56
1111	60

*NOT ALLOWED DUE TO BAD WAVEFORMS

****

P7078

THE 8-BIT ADDRESSABLE REGISTER LOADS THE DATA (DIN) INTO THE APPROPRIATE ADDRESS LINE REGISTER (Q) ON THE RISING EDGE OF THE

L	H	H	L	DIN	Q9	Q8	Q7	DIN	Q5	Q4	Q3	Q2	Q1	Q0
L	H	H	H	DIN	Q9	Q8	DIN	Q6	Q5	Q4	Q3	Q2	Q1	Q0
H	L	L	L	DIN	Q9	DIN	Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0
H	L	L	H	DIN	DIN	Q8	Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0
H	L	H	L	DIN	Q9	Q8	Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0
H	L	H	H	DIN	Q9	Q8	Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0
H	H	X	X	DIN	Q9	Q8	Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0

****

P7073

THIS PAL20L10 INTERFACES BETWEEN THE MOTOROLA MC6800 MICROPROCESSOR AND ITS SYSTEM COMPONENTS ON A SINGLE BOARD COMPUTER. THE FUNCTIONS IT PERFORMS, PREVIOUSLY DONE WITH RANDOM LOGIC ARE: ADDRESS DECODING, MEMORY AND I/O SELECT, RESET SIGNAL GENERATION, AND CONTROL OF THE BUFFER WHICH INTERFACES THE DATA BUS TO OTHER BOARDS IN THE SYSTEM.

****

P7074

THIS IS AN EXAMPLE OF A QUAD 3-TO-1 MULTIPLEXER USING A PAL18L4. SELECT LINES A,B ARE ENCODED IN BINARY, WITH A REPRESENTING THE LSB. THE OUTPUTS (Y) ARE ALL HIGH IF THE SELECT LINES ARE BOTH HIGH (B,A=H).

OPERATIONS TABLE:

INPUT SELECT		OUTPUTS
B	A	Y
L	L	C0
L	H	C1
H	L	C2
H	H	H

****

P7075

THIS PAL DESIGN SPECIFICATION DESCRIBES A 4-BIT SYNCHRONOUS COUNTER WITH 4-BIT REGISTER. DATA CAN BE LOADED TO THE COUNTER FROM THE REGISTER. IT CAN ALSO BE SYNCHRONOUSLY CLEARED. THE REGISTER CAN BE LOADED FROM EITHER THE COUNTER OR THE DATA INPUTS UNDER CONTROL OF THE SEL INPUT. THE REGISTER CAN ALSO BE SYNCHRONOUSLY CLEARED. THE COUNTER AND REGISTER HAVE A COMMON CLOCK FOR SYNCHRONOUS OPERATION.

****

P7076

THE 9-BIT SYNCHRONOUS COUNTER HAS PARALLEL LOAD, DECREMENT, AND HOLD CAPABILITIES. DATA (D8-D0) IS LOADED INTO THE OUTPUT REGISTER (Q8-Q0) WHEN THE LOAD INPUT IS TRUE (/LD=L) AND A POSITIVE EDGE PULSE IS RECEIVED ON THE CLOCK PIN (CLK). THE COUNTER WILL DECREMENT IF A CLOCK PULSE IS RECEIVED WITH THE LOAD INPUT BEING FALSE (/LD=H). THE OPERATION IS A HOLD IF NO CLOCK PULSE IS RECEIVED REGARDLESS OF ANY OTHER INPUTS.

THE BORROW OUT PIN (/BO) SHOWS HOW TO IMPLEMENT A BORROW OUT USING A REGISTER BY ANTICIPATED ONE COUNT BEFORE THE TERMINAL COUNT IF COUNTING AND THE TERMINAL COUNT IF LOADING.

OPERATIONS TABLE:

/OC	CLK	/LD	D8-D0	Q8-Q0	OPERATION
H	X	X	X	Z	HI-Z
L	L	X	X	/Q	HOLD
L	C	L	D	D	LOAD
L	C	H	X	Q MINUS 1	DECREMENT

****

P7077

THE REFRESH CLOCK GENERATOR CAN GENERATE REFRESH CLOCK (RFCK) FOR THE DYNAMIC RAM CONTROLLERS. THE PERIOD OF RFCK DEPENDS ON F3-F0 WHILE THE DURATION OF RFCK BEING LOW DEPENDS ON M3-M0.

FFFF 3210	RFCK PERIOD (CYCLES)
0000	12
0001	28
0010	44
0011	60
0100	76
0101	92
0110	108
0111	124
1000	140
1001	156
1010	172
1011	188
1100	204
1101	220
1110	236
1111	252

MMMM 3210	RFCK LOW DURATION (CYCLES)
0000	0
0001	4
0010	8
0011	12
0100	16
0101	20*
0110	24
0111	28
1000	32
1001	36*
1010	40*
1011	44*
1100	48
1101	52*
1110	56
1111	60

*NOT ALLOWED DUE TO BAD WAVEFORMS

****

P7078

THE 8-BIT ADDRESSABLE REGISTER LOADS THE DATA (DIN) INTO THE APPROPRIATE ADDRESS LINE REGISTER (Q) ON THE RISING EDGE OF THE

****

P7082

THE 4-BIT COUNTER LOADS DATA, INCREMENTS, OR HOLDS ON THE RISING EDGE OF THE CLOCK (CLK).

THE COUNTER WILL HOLD WHEN THE TERMINAL COUNT IS REACHED UNTIL NEW DATA IS LOADED OR THE REGISTERS ARE CLEARED.

THESE OPERATIONS ARE EXERCISED IN THE FUNCTION TABLE AND SUMMARIZED IN THE IN THE OPERATIONS TABLE:

/OC	CLK	CLEAR	/LOAD	/EP	D3-D0	Q3-Q0	OPERATION
H	X	X	X	X	X	Z	HI-Z
L	C	H	X	X	L	L	CLEAR
L	C	L	L	X	A	A	LOAD
L	C	L	H	H	X	Q	HOLD
L	C	L	H	L	X	Q PLUS 1	INCREMENT

****

P7083

THIS PAL PROVIDES A SINGLE CHIP DECODER FOR USE IN MEMORY MAPPED I/O OPERATIONS REQUIRING FOUR ACTIVE LOW PORT ENABLES AND FULL 16-BIT DECODE WITH TWO TWO STROBE LINES (/RD AND /WR). EQUATION TERMS CAN BE CHANGED TO ACCOMMODATE ANY 16-BIT ADDRESS.

THE PAL WILL MONITOR THE SYSTEM MEMORY ADDRESS BUS (A15-A0) AND DECODE THE SPECIFIED MEMORY ADDRESS WORD (1F76,1F77,1F78,1F79) TO PRODUCE A PORT ENABLE FOR READ, WRITE, AUXILARY UNIT 1 (AUX1), AUXILARY UNIT 2 (AUX2) AS SHOWN IN THE FOLLOWING TABLE:

ADDRESS INPUTS	STROBES /RD	 /WR	PORT ENABLE
1F76	L	X	READ
1F77	X	L	WRITE
1F78	X	X	AUX1
1F79	X	X	AUX2

****

P7084

THE 9-BIT COUNTER HAS INCREMENT, LOAD, AND HOLD CAPABILITIES. DATA (D) IS LOADED INTO THE OUTPUT REGISTER (Q) WHEN THE LOAD LINE IS TRUE (/LD=L) AND A POSITIVE EDGE PULSE IS RECEIVED ON THE CLOCK PIN (CLK). THE COUNTER WILL INCREMENT IF A CLOCK PULSE IS RECEIVED WITH THE LOAD LINE BEING FALSE (/LD=H). THE COUNTER WILL LOCK AT THE TERMINAL COUNT REGARDLESS OF CLOCK PULSES UNTIL DATA

IS LOADED INTO THE REGISTER. THE OPERATION IS A HOLD IF NO CLOCK PULSE IS RECEIVED REGARDLESS OF ANY OTHER INPUTS.

THE CARRY OUT PIN (/CO) SHOWS HOW TO IMPLEMENT A CARRY OUT USING A REGISTER BY ANTICIPATED ONE COUNT BEFORE THE TERMINAL COUNT IF COUNTING AND THE TERMINAL COUNT IF LOADING. ALSO THE REGISTERED CARRY OUT WILL REMEMBER THE TERMINAL COUNT STATE IN ORDER TO DISABLE THE COUNTER AT THE TERMINAL COUNT.

OPERATIONS TABLE:

/OC	CLK	/LD	D8-D0	Q8-Q0	OPERATION
H	X	X	X	Z	HI-Z
L	L	X	X	Q	HOLD
L	C	L	D	D	LOAD D
L	C	H	X	Q PLUS 1	COUNT UP

****

P7085

THIS PAL PROVIDES THE DECODE LOGIC FOR INTERFACING THE MMI SN54/74S508, 8-BIT SEQUENTIAL MULTIPLIER/DIVIDER, WITH THE INTEL 8085 MICROPROCESSOR. THE PAL16R4 MONITORS THE LOWER 8 BIT ADDRESS/DATA BUS (AD0-AD7), THE ADDRESS LATCH ENABLE (ALE), AND THREE OF THE UPPER 8 BIT ADDRESS BUS (A8-A15) WHICH IS LABELED/E1,E2,E3 FROM THE 8085 IN ORDER TO DECODE AN ACTIVE LOW CHIP-ACTIVATION SIGNAL (/GO) FOR THE MULTIPLIER/DIVIDER. THE INSTRUCTION LINES AND CHIP-ACTIVATION SIGNAL ARE REGISTERED IN ORDER TO INSURE THAT INSTRUCTION INPUTS WILL NOT CHANGE WHEN THE CLOCK IS LOW (CLK=L).

FOR THIS PARTICULAR DESIGN, THE THREE INSTRUCTION INPUTS TO THE 74S508 (I0-I2) ARE ASSIGNED TO THE THREE LSB ADDRESS/DATA BUS (AD0-AD2), WHILE THE REMAINING ADDRESS BITS (AD3-AD7) ARE DECODED BY THE PAL TO DETERMINE IF THE 74S508 IS SELECTED. ALSO, ADDRESS 100 TO 107 IS RESERVED FOR THE 74S508. THE ADDRESS SPACE CAN BE CHANGE BY EDITING THE LOGIC EQUATIONS.

THIS PAL DESIGN CORRESPONDS TO FIGURE 6 ON PAGE 7 OF THE FOLLOWING APPLICATION NOTE:

AN-103 "A DEDICATED MULTIPLIER/DIVIDER SPEEDS UP MULTIPLICATION AND DIVISIO
FOR 8-BIT MICROPROCESSORS" by Ehud Gordon

****

P7086

THIS PAL PROVIDES THE DECODE LOGIC FOR INTERFACING THE MMI SN54/74S508, 8-BIT SEQUENTIAL MULTIPLIER/DIVIDER, WITH THE INTEL 8085 MICROPROCESSOR. THE PAL16R4 MONITORS THE UPPER 8 BIT ADDRESS BUS (A8-A15), THE ADDRESS LATCH ENABLE (ALE), AND THE THREE MACHINE CYCLE STATUS LINES (IOM,S1,S0) FROM THE 8085 IN ORDER TO DECODE AN ACTIVE LOW CHIP-ACTIVATION SIGNAL (/GO) FOR

THE 74S508. THE INSTRUCTION LINES AND CHIP-ACTIVATION SIGNAL ARE REGISTERED IN ORDER TO INSURE THAT INSTRUCTION INPUTS WILL NOT CHANGE WHEN THE CLOCK IS LOW (CLK=L). BY MONITORING THE MACHINE STATUS CYCLE, THE 74S508 CAN BE ADDRESS MAPPED BY THE 8085 AS IF IT WERE AN I/O DEVICE, THUS NOT USING ANY MEMORY MAP ADDRESS SPACE. THE MACHINE CYCLE STATUS FROM THE 8085 IS AVAILABLE ON THE FALLING EDGE OF ALE.

FOR THIS PARTICULAR DESIGN, THE THREE INSTRUCTION INPUTS TO THE 74S508 (I0-I2) ARE ASSIGNED TO THE THREE LSB BITS OF ADDRESS BUS (A8-A10), WHILE THE REMAINING ADDRESS BITS (A11-A15) ARE DECODED BY THE PAL TO DETERMINE IF THE 74S508 IS SELECTED. ALSO, ADDRESS 100 TO 107 IS RESERVED FOR THE 74S508. THE ADDRESS SPACE CAN BE CHANGED BY SIMPLY EDITING THE LOGIC EQUATIONS.

****

P7087

THE 2-DIGIT BCD (BINARY CODED DECIMAL) SYNCHRONOUS COUNTER WITH COMPLEMENTARY COUNT ENABLES, PARALLEL LOAD, AND CARRY OUT IS IMPLEMENTED IN A PAL20X8. THREE CONTROL INPUTS (/LD,/CE1,CE2) PROVIDE ONE OF THREE OPERATIONS WHICH OCCUR SYNCHRONOUSLY ON THE RISING EDGE OF THE CLOCK (CLK).

THE COUNTER WILL INCREMENT IN A BINARY-CODED-DECIMAL SEQUENCE WHEN BOTH COUNT ENABLES ARE TRUE (/CE1=L AND CE2=H) AND LOAD IS FALSE (/LD=H). THE COUNTER WILL HOLD IF ONE COUNT ENABLE IS FALSE (/CE1=H OR CE2=L).

THE LOAD OPERATION LOADS THE INPUTS (D1- AND D2-) INTO THE OUTPUT REGISTER (Q1- AND Q2-) WHEN LOAD IS TRUE (/LD=L). NOTE THAT LOAD OVERRIDES INCREMENT.

TWO OR MORE BCD COUNTERS CAN BE CASCADED TO IMPLEMENT LARGER BCD COUNTERS BY CONNECTING CARRY OUT (/CO) OF THE FIRST STAGE TO COUNT ENABLE (/CE1) OF THE SECOND STAGE.

THIS DESIGN IS IDEAL IN AN INDUSTRIAL CONTROL APPLICATION WHERE AN EVENT COUNTER IS NEEDED TO DRIVE NUMERIC DISPLAYS. THE PAL CAN RECEIVE ONE COUNT ENABLE IN THE FORM OF STROBES FROM A MOTOR OR OTHER DEVICE. THE SECOND COUNT ENABLE CAN RECEIVE THE PERIOD SIGNAL. THE PAL WILL PROVIDE TWO ACTIVE HIGH BCD OUTPUTS (Q1- AND Q2-) TO DRIVE TWO NUMERIC INDICATORS, SUCH AS THE HEWLETT PACKARD 5082-7300 WHICH FEATURES AN ON-BOARD DECODER/DRIVER AND LATCH ENABLE.

PARALLEL LOADING IS PROVIDED FOR PAL AND NUMERIC INDICATOR TESTIBILITY, HOWEVER, /LD CAN BE CHANGED TO A CLEAR FUNCTION (/CLR) BY TYING THE BCD PARALLEL INPUTS (D1- AND D2-) TO GROUND (GND).

THESE OPERATIONS ARE EXERCISED IN THE FUNCTION TABLE AND SUMMARIZED IN THE OPERATIONS TABLE:

/OC	CLK	/LD	/CE1	CE2	BCD-IN	BCD-OUT	OPERATION
H	X	X	X	X	X	Z	HI-Z
L	C	L	X	X	D	D	LOAD
L	C	H	H	X	X	Q	HOLD (/CE1=H)
L	C	H	X	L	X	Q	HOLD (CE2=L)
L	C	H	L	H	X	Q PLUS 1	INCREMENT

****

P7088

THE 9-BIT COUNTER WITH SHIFT REGISTER HAS LOAD, SHIFT, INCPEMENT, AND HOLD CAPABILITIES. DATA (D) IS LOADED INTO THE OUTPUT REGISTER (Q) WHEN THE LOAD IS TRUE (/LD=L) AND A POSITIVE EDGE PULSE IS RECEIVED ON THE CLOCK PIN (CLK). THE COUNTER WILL SHIFT DATA TOWARD THE MSB IF SHIFT IS TRUE (/SH=L) AND A CLOCK PULSE IS RECEIVED WITH LOAD BEING FALSE (/LD=H). THE COUNTER WILL INCREMENT IF COUNT IS TRUE (/CNT=L) AND A CLOCK PULSE IS RECEIVED WITH LOAD AND SHIFT BEING FALSE (/LD,/SH=H). THE OPERATION IS A HOLD IF LOAD, SHIFT, AND COUNT ARE FALSE (/LD,/SH,/CNT=H) REGARDLESS OF CLOCK TRANSITIONS.

LOAD OVERRIDES SHIFT AND INCREMENT, SHIFT OVERRIDES INCREMENT.

THE FOLLOWING OPERATIONS ARE SIMULATED USING THE FUNCTION TABLE AND SUMMARIZED IN THE OPERATIONS TABLE:

/OC	CLK	/LD	/SH	/CNT	D8-D0	Q8-Q0	OPERATION
H	X	X	X	X	X	Z	HI-Z
L	C	L	X	X	D	D	LOAD D
L	C	H	L	X	X	SHIFT Q	SHIFT
L	C	H	H	L	X	Q PLUS 1	COUNT UP
L	C	H	H	H	X	Q	HOLD

****

P7089

THE PAL PROVIDES A SINGLE CHIP INTERFACE CONTROLLER FOR INTERFACING MOTOROLA'S 8MHz 6800 MICROPROCESSOF TO ZILOG'S 4MHz Z85XX PERIPHERAL. THE INTERFACE CONTROLLER GENERATES ALL THE REQUIRED CONTROL SIGNALS, OF WHICH FOUR [/RD, /WR, PCLK(=/C0), AND /ACK] GO TO THE Z85XX AND /DTK GOES TO THE 68000.

IN ADDITION TO GENERATING CONTROL SIGNALS, THE INTERFACE CONTROLLER HAS THE CAPABILITY OF CONTROLLING THREE TYPES OF BUS CYCLES; READ, WRITE, AND INTERRUPT ACKNOWLEDGE.

TIMING FOR PROCESSING OF DATA TRANSFER IS CONTROLLED WITH AN INTERNAL DOWN COUNTER. A COUNTER DECODER CONTROLS TIMING OF /DTK(DATA TRANSFER ACKNOWLEDGE) TO THE 68000 TO INFORM A COMPLETE DATA TRANSFER CYCLE.

THESE OPERATIONS ARE EXERCISED IN THE FUNCTION TABLE AND

THESE OPERATIONS ARE EXERCISED IN THE FUNCTION TABLE AND SUMMARIZED IN THEOPERATION TABLE:

											/	/	/	
/	C		/	/	/	F	F	F	/	/	D	C	A	
O	L	/	A	C	R	C	C	C	W	R	T	Y	C	
C	K	R	S	S	W	2	1	0	R	D	K	C	K	OPERATION
H	X	X	X	X	X	X	X	X	Z	Z	Z	Z	Z	HI-Z
L	C	L	X	X	X	X	X	X	L	L	H	H	H	INITIALIZATION
L	C	H	L	L	H	L	L	L	H	L	H	L	H	READ
L	C	H	L	L	L	L	L	L	L	H	H	L	H	WRITE
L	C	H	L	H	H	H	H	H	H	L	H	L	L	READ DURING INTERRUPT

FUNCTIONAL DESCRIPTION

THE PAL GENERATES FIVE CONTROL SIGNALS AT WHICH [/RD, /WR, PCLK(=/C0), AND /ACK] TO THE Z85XX AND /DTK GOES TO THE 68000. /RD, /WR, AND /ACK ARE THREE TYPES OF Z85XX CYCLES, WHICH ARE CONTROLLED BY THE PAL:

/RD - READ CYCLE
/WR - WRITE CYCLE
/ACK - INTERRUPT ACKNOWLEDGE CYCLE

/C0, /C1, /C2, AND /C3 ARE PART OF A 4-BIT SYNCHRONOUS DOWN COUNTER WHERE /C3 IS THE MSB AND /C0 IS THE LSB. /C0 IS SIMPLY A DIVIDE BY 2 OF THE 68000 CLOCK USED BY THE PERIPHERAL FOR PCLK. THE OTHER THREE SIGNALS, C1, C2, C3 WITH /CYC ARE USED INTERNAL TO THE PAL FOR GENERATING THE ABOVE CONTROL SIGNALS.

THE COUNTER TOGGLES BETWEEN 14 AND 15 UNTIL /CYC(=CYCLE) GOES ACTIVE LOW AND THEN WILL BEGIN COUNTING DOWN. THE COUNTER WILL CONTINUE COUNTING DOWN UNTIL /AS GOES INACTIVE HIGH, SIGNALING THE END OF CYCLE. AT THIS TIME THE COUNTER WILL GO BACK TO TOGGLING BETWEEN COUNT 14 AND 15. REFER TO THE FUNCTION TABLE.

THE CONTROL SIGNALS, /RD, /WR, /ACK, AND /DTK ARE GENEREATED THROUGH THE PAL AND QUALIFIED WITH THE COUNTER TO GENERATE EDGES THAT MEET TIME RESTRAITS.

****

P7090

THE 16 INPUT REGISTERED PRIORITY ENCODER ACCEPTS SIXTEEN ACTIVE-LOW INPUTS (I0-I15) TO LOAD THE BINARY WEIGHTED CODE OF THE PRIORITY ORDER INTO THE OUTPUT REGISTER (Q3-Q0) ON THE RISING EDGE OF THE CLOCK (CLK). A PRIORITY IS ASSIGNED TO EACH INPUT SO THAT WHEN TWO INPUTS ARE SIMULTANEOUSLY ACTIVE, THE INPUT WITH THE HIGHEST PRIORITY IS LOADED INTO THE OUTPUT REGISTER. THERFORE THE HIGHEST PRIORITY INPUT (I0=H) PRODUCES HHHH IN THE OUTPUT REGISTER AND THE LOWEST PRIORITY INPUT (I15=H) PRODUCES

LLLL IN THE OUTPUT REGISTER.

OPERATIONS TABLE

/OC	CLK	I15-I0	Q3-Q0	OPERATION
H	X	X	Z	HI-Z
L	C	I0 =H	15	I0 INTERRUPT (HIGHEST PRICRITY INPUT)
L	C	I1 =H	14	I1 INTERRUPT
L	C	I1 =H	13	I2 INTERRUPT
L	C	I1 =H	12	I3 INTERRUPT
L	C	I1 =H	11	I4 INTERRUPT
L	C	I1 =H	10	I5 INTERRUPT
L	C	I1 =H	9	I6 INTERRUPT
L	C	I1 =H	8	I7 INTERRUPT
L	C	I1 =H	7	I8 INTERRUPT
L	C	I1 =H	6	I9 INTERRUPT
L	C	I1 =H	5	I10 INTERRUPT
L	C	I1 =H	4	I11 INTERRUPT
L	C	I1 =H	3	I12 INTERRUPT
L	C	I1 =H	2	I13 INTERRUPT
L	C	I1 =H	1	I14 INTERRUPT
L	C	I1 =H	0	I15 INTERRUPT (LOWEST PRIORITY INPUT)

****

P7091

THE 16 INPUT PRIORITY ENCODER INTERRUPT FLAG ACCEPTS SIXTEEN ACTIVE-LOW INPUTS (I0-I15) IN ORDER TO GENERATE AN ACTIVE LOW (NEG_INT) AND ACTIVE HIGH (POS_INT) INTERRUPT FLAG.

UP TO FOUR INTERRUPT ENABLE PINS OF ANY POLARITY CAN BE ADDED IF A PAL20C1 IS SUBSTITUTED.

****

P7092

THE 15 INPUT REGISTERED PRIORITY ENCODER ACCEPTS FIFTEEN ACTIVE-LOW INPUTS (I0-I14) TO LOAD THE BINARY WEIGHTED CODE OF THE PRIORITY ORDER INTO THE OUTPUT REGISTER (Q3-Q0) ON THE RISING EDGE OF THE CLOCK (CLK). A PRIORITY IS ASSIGNED TO EACH INPUT SO THAT WHEN TWO INPUTS ARE SIMULTANEOUSLY ACTIVE, THE INPUT WITH THE HIGHEST PRIORITY IS LOADED INTO THE OUTPUT REGISTER. THERFORE THE HIGHEST PRIORITY INPUT (I0=H) PRODUCES HHHH IN THE OUTPUT REGISTER AND THE LOWEST PRIORITY INPUT (I14=H) PRODUCES LLLH IN THE OUTPUT REGISTER. THE NON-REGISTERED INTERRUPT FLAG (FLAG) GOES HIGH WHEN AN INTERRUPT IS PRESENT AND REMAINS LOW WHEN THERE IS NO INTERRUPT PRESENT.

OPERATIONS TABLE

/OC	CLK	I14-I0	Q3-Q0	FLAG	OPERATION
H	X	X	Z	X	HI-Z

L	C	L	0	L	NO INTERRUPT
L	C	I0 =H	15	H	I0 INTERRUPT (HIGHEST PRIORITY INPUT)
L	C	I1 =H	14	H	I1 INTERRUPT
L	C	I2 =H	13	H	I2 INTERRUPT
L	C	I3 =H	12	H	I3 INTERRUPT
L	C	I4 =H	11	H	I4 INTERRUPT
L	C	I5 =H	10	H	I5 INTERRUPT
L	C	I6 =H	9	H	I6 INTERRUPT
L	C	I7 =H	8	H	I7 INTERRUPT
L	C	I8 =H	7	H	I8 INTERRUPT
L	C	I9 =H	6	H	I9 INTERRUPT
L	C	I10=H	5	H	I10 INTERRUPT
L	C	I11=H	4	H	I11 INTERRUPT
L	C	I12=H	3	H	I12 INTERRUPT
L	C	I13=H	2	H	I13 INTERRUPT
L	C	I14=H	1	H	I14 INTERRUPT (LOWEST PRIORITY INPUT)

P7093

THE 8 INPUT REGISTERED PRIORITY ENCODER ACCEPTS SIXTEEN ACTIVE-LOW INPUTS (I0-I7) TO LOAD THE BINARY WEIGHTED CODE OF THE PRIORITY ORDER INTO THE OUTPUT REGISTER (Q2-Q0) ON THE RISING EDGE OF THE CLOCK (CLK) PROVIDING THE FOUR ENABLE INPUTS ARE TRUE (E1=H,E2=H,/E3=L,/E4=L). A PRIORITY IS ASSIGNED TO EACH INPUT SO THAT WHEN TWO INPUTS ARE SIMULTANEOUSLY ACTIVE, THE INPUT WITH THE HIGHEST PRIORITY IS LOADED INTO THE OUTPUT REGISTER. THERE FORE THE HIGHEST PRIORITY INPUT (I0=H) PRODUCES HHH IN THE OUTPUT REGISTER AND THE LOWEST PRIORITY INPUT (I7=H) PRODUCES LLL IN THE OUTPUT REGISTER.

THE PRIORITY INTERRUPT ENCODER REGISTERS (Q3-Q0) ARE UPDATED ON THE RISING EDGE OF THE CLOCK (CLK) PROVIDING THE FOUR ENABLE INPUTS ARE TRUE (E1=H,E2=H,/E3=L,/E4=L). THE PREVIOUS DATA IS HELD IN THE PRIORITY ENCODER REGISTERS IF ANY OF THE ENABLE INPUTS ARE FALSE (E1=L,E2=L,/E3=H,/E4=H) REGARDLESS OF CLOCK TRANSITIONS. NOTE THAT THE POLARITY OF THE ENABLES CAN BE CHANGED BY MERELY EDITING THE LOGIC EQUATIONS.

OUTPUT Q4 SERVES AS THE INTERRUPT FLAG AND IS TRUE (Q4=L) WHEN ANY OF THE 8 INPUTS ARE ACTIVE (I=H) ON THE RISING EDGE OF THE CLOCK (CLK) PROVIDING THE FOUR ENABLE INPUTS ARE TRUE (E1=H,E2=H,/E3=L,/E4=L). THE INTERRUPT FLAG IS FALSE (Q4=H) WHEN ALL INPUTS ARE INACTIVE (I=L) OR WHEN ANY ONE OF THE FOUR ENABLE INPUTS ARE FALSE (E1=L,E2=L,/E3=H,/E4=H).

OPERATIONS TABLE

/OC	CLK	E1	E2	/E3	/E4	I7-I0	Q4	Q3-Q0	OPERATION
H	X	X	X	X	X	X	Z	Z	HI-Z
L	C	L	X	X	X	X	H	Q	NOT ENABLED (E1=L)
L	C	X	L	X	X	X	H	Q	NOT ENABLED (E2=L)
L	C	X	X	H	X	X	H	Q	NOT ENABLED (/E3=H)

L	C	X	X	X	H	X	H	Q	NOT ENABLED (/E4=H)
L	C	H	H	L	L	L	H	X	NO INTERRUPT FLAG
L	C	H	H	L	L	I0=H	L	7	I0 INTERRUPT (HIGHEST PRIORITY
L	C	H	H	L	L	I1=H	L	6	I1 INTERRUPT
L	C	H	H	L	L	I1=H	L	5	I2 INTERRUPT
L	C	H	H	L	L	I1=H	L	4	I3 INTERRUPT
L	C	H	H	L	L	I1=H	L	3	I4 INTERRUPT
L	C	H	H	L	L	I1=H	L	2	I5 INTERRUPT
L	C	H	H	L	L	I1=H	L	1	I6 INTERRUPT
L	C	H	H	L	L	I1=H	L	0	I7 INTERRUPT (LOWEST PRIORITY)

P7094

CLK	/OC	/EN	SET	ROT	MODE	SW1-SW4	COMMENTS
X	H	X	X	X	X	Z	HI-Z
C	L	H	X	X	X	HOLD	HOLD MOTOR POSITION
C	L	L	H	X	X	1	SET MOTOR POSITION TO STEP 1
C	L	L	L	H	H	SW PLUS 1	HALF-STEP MOTOR CLOCKWISE
C	L	L	L	H	L	SW PLUS 2	FULL-STEP MOTOR CLOCKWISE
C	L	L	L	L	H	SW MINUS 1	HALF-STEP MOTOR COUNTERCLOCKWISE
C	L	L	L	L	L	SW MINUS 2	FULL-STEP MOTOR COUNTERCLOCKWISE

P7095

THE OCTAL REGISTERED BARREL SHIFTER WILL SHIFT EIGHT BITS OF DATA (D7-D0) A NUMBER OF LOCATIONS INTO THE OUTPUT REGISTER (Q7-Q0) AS SPECIFIED BY THE BINARY ENCODED INPUT (I2-I0) SYNCHRONOUS WITH THE CLOCK INPUT (CLK) AND PROVIDING THE ENABLE PIN IS TRUE (/E=LOW). THE OUTPUT REGISTER WILL BE PRESET TO ALL HIGHS WHEN ENABLE IS FALSE (/E=HIGH). THE THREE-STATE OUTPUTS ARE HIGH-Z WHEN THE OUTPUT CONTROL LINE (/OC) IS LOW.

P7096

THIS PAL16L8 IMPLEMENTS AN 8-BIT LATCH FUNCTION WITH THREE-STATE OUTPUTS. THE LATCH PASSES EIGHT BITS OF DATA (D7-D0) TO THE EIGHT OUTPUTS (Q7-Q0) WHEN LATCH ENABLE IS TRUE (G=HIGH). THE DATA IS LATCHED WHEN LATCH ENABLE IS FALSE (G=LOW). THE OUTPUTS WILL BE DISABLED (HI-Z) WHEN OUTPUT ENABLE IS TRUE (/OC=TRUE) REGARDLESS OF ANY OTHER INPUTS.

/OC	G	D7-D0	Q7-Q0	COMMENTS
H	X	X	Z	HI-Z
L	L	X	Q	LATCH OUTPUT
L	H	D	D	LOAD LATCH

THIS DESIGN SHOWS HOW TO IMPLEMENT A "CLEAN" LATCH SINCE THERE ARE NO OUTPUT GLITCHES AS THE DEVICE CHANGES STATE. THE KARNAUGH MAP BELOW FOR O+ ILLUSTRATES THIS. THE TWO HORIZONTAL CIRCLES REPRESENT THE "LOAD LATCH" AND "LATCH OUTPUT" PRODUCT LINES. THE VERTICAL CIRCLE LINKS TOGETHER THE OTHER CIRCLES IN ORDER TO COVER A POTENTIAL SWITCHING HAZARD WHICH WOULD OCCUR WHEN THE OUTPUTS ARE ALWAYS HIGH.

```
G |_
    |_
      |- 00    01    11    10
        |----|----|----|----|
     0  |  0 |  1 |  1 |  0 |
        |-------------------|
     1  |  0 |  1 |  1 |  1 |
        |----|----|----|----|
```

****

P7097

THIS PAL16R4 IMPLEMENTS A TWO CHANNEL SHAFT ENCODER OF THE TYPE USED IN SPEED CONTROLLERS AND OPTICAL DEVICES.

BOTH THE "UP" AND "DOWN" OUTPUTS OF THE PAL ARE NORMALLY HIGH.

WHEN THE SIGNAL AT THE "PHI0" INPUT LEADS THE SIGNAL AT THE "PHI90" INPUT, THE "DOWN" OUTPUT ALTERNATES BETWEEN HIGH AND LOW LEVELS AT HALF THE "CLK" FREQUENCY RATE. ALSO, WHEN THE SIGNAL AT THE "PHI0" INPUT LAGS THE SIGNAL AT THE "PHI90" INPUT, THE "UP" OUTPUT ALTERNATES BETWEEN HIGH AND LOW LEVELS AT HALF THE "CLK" FREQUENCY RATE.

THE SHAFT ENCODER FEATURES THE CONFIGURATION AND OUTPUT POLARITY TO DRIVE AN 74S193 TYPE UP/DOWN COUNTER.

THIS DESIGN WITH GLITCHFREE OUTPUTS WILL BE EXTREMELY USEFUL IN ELECTRICALLY NOISY ENVIRONMENTS. THE PINNING IS GIVEN AS A FIRST PROPOSAL AND CAN BE CHANGED ACCORDING TO THE PC-BOARD LAYOUT.

****

P7098

THIS PAL16R8 IMPLEMENTS A TWO CHANNEL SHAFT ENCODER OF THE TYPE USED IN SPEED CONTROLLERS AND OPTICAL DEVICES.

THE "COUNT" OUTPUT OF THE PAL IS NORMALLY HIGH. DURING SHAFT ENCODING THIS OUTPUT ALTERNATES BETWEEN HIGH AND LOW.

INPUT "X4" SELECTS BETWEEN HALF (X4=H) OR QUARTER (X4=L) CLK FREQUENCY OF THE "COUNTER" OUTPUT.

OUTPUT "UD" DETERMINES WHETHER SIGNAL PHI0 LEADS (UD=H) OR LAGS (UD=L) SIGNAL PHI90.

THE SHAFT ENCODER FEATURES THE CONFIGURATION AND OUTPUT POLARITY

TO DRIVE AN 74S697 TYPE UP/DOWN COUNTER.

THIS DESIGN WITH GLITCHFREE OUTPUTS WILL BE EXTREMELY USEFUL IN ELECTRICALLY NOISY ENVIRONMENTS. THE PINNING IS GIVEN AS A FIRST PROPOSAL AND CAN BE CHANGED ACCORDING TO THE PC-BOARD LAYOUT.

****

P7099

THIS PAL20X10 IMPLEMENTS A TWO CHANNEL SHAFT ENCODER WITH AN INTERNAL 4-BIT UP/DOWN COUNTER.

BOTH THE "UP" AND "DOWN" OUTPUTS OF THE PAL ARE NORMALLY AT HIGH.

WHEN THE SIGNAL AT THE "PHI0" INPUT LEADS THE SIGNAL AT THE "PHI90" INPUT, THE "DOWN" OUTPUT ALTERNATES BETWEEN HIGH AND LOW LEVELS AND THE COUNTER WILL COUNT DOWN. WHEN THE SIGNAL AT THE "PHI0" INPUT LEADS THE SIGNAL AT THE "PHI90" INPUT, THE "UP" OUTPUT ALTERNATES BETWEEN HIGH AND LOW LEVELS AND THE COUNTER WILL COUNT UP.

INPUT "X4" SELECTS BETWEEN HALF (X4=H) OR QUARTER (X4=L) CLK FREQUENCY OF THE COUNTER OUTPUTS.

THE INTERNAL 4-BIT SYNCHRONOUS COUNTER HAS COUNT UP, COUNT DOWN CAPABILITIES. ALSO, THE COUNTER CAN PARALLEL LOAD AND HOLD DATA INDEPENDENTLY OF THE SHAFT ENCODER SECTION. THE REGISTERS ARE SYNCHRONOUSLY INITIALIZED WHEN /SET IS HELD LOW.

THE CONTROL INPUTS PROVIDE THESE OPERATIONS WHICH OCCUR SYNCHRONOUSLY AT THE RISING EDGE OF THE CLOCK.

IBM PC

1. Version 1.6C released on September 15, 1983. This version is compatible with the VAX11 version.

2. Version 1.7A released March 15, 1984 to handle the 20AP PAL family.

INTEL/MDS

1. Version 2.40 released September 15, 1983.

CP/M

1. Version 1.00 released September 15, 1983.

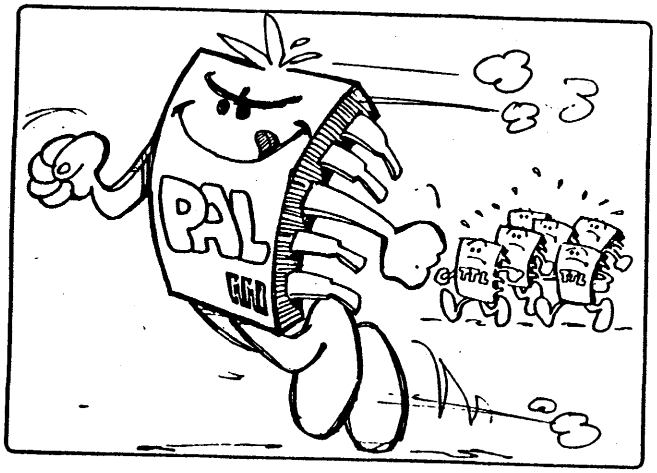
2. Version 1.7A released March 15, 1984 to handle the 20AP PAL family.

VAX11/IBM MAINFRAMES

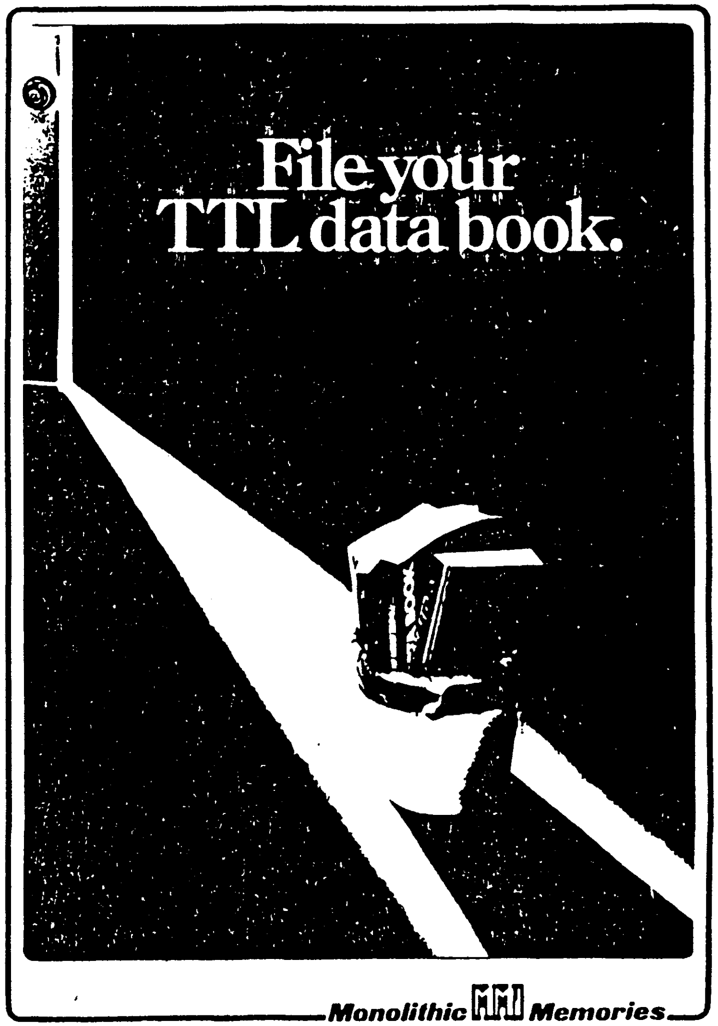
1. Version 1.6C released September 15, 1983.

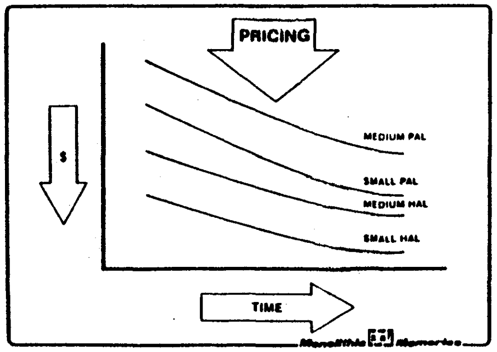
2. Version 1.7A released March 15, 1984 to handle the 20AP PAL family.

PAL® Training

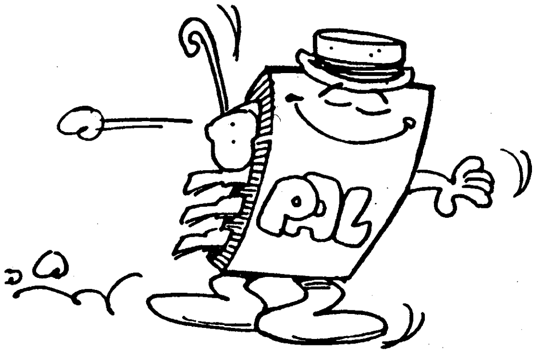


Monolithic MMI Memories





PAL BENEFITS AND FEATURES



Monolithic Memories

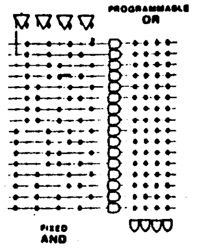
PAL — BENEFITS AND FEATURES

I.

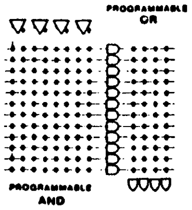
- Reduce chip count 12 to 1 or greater
- 25 ns typical delay
- Low cost with hard mask options (HAL)
- 15 parts replace 90% of conventional TTL Logic
- Second source available

Monolithic Memories

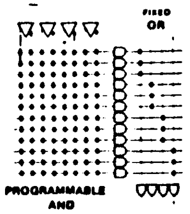
PROM



FPLA



PAL



GATE ARRAY

Advantages:

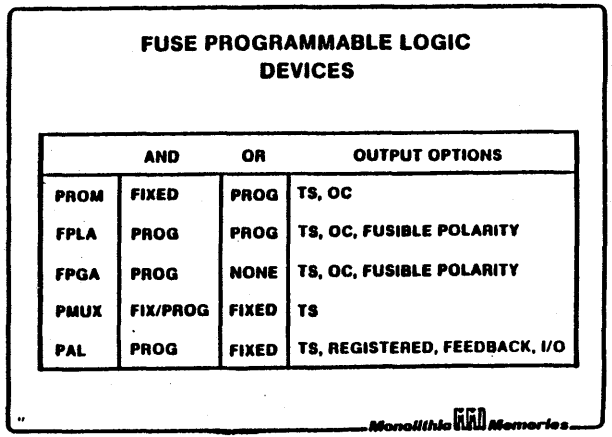
- Higher number of gates available, good for VLSI function
- Available in different technologies i.e., CMOS, NMOS, CML, I^2L, Bipolar TTL

Disadvantages:

- Takes three (3) months before one sees the sample prototype
- Initial cost very high ($5,000 to $10,000)
- Higher number of pinouts (40-pin typical)
- Chip layout — inefficient
- Loose design tolerances
- No second source

PAL VS GATE ARRAY

	PAL	GATE ARRAY
PROTOTYPES	INSTANT	12-16 WEEKS
DEVELOPMENT COST	NONE	30-60 K$
SECOND SOURCE	THREE	NONE
DENSITY	250 GATES	UP TO 1000 GATES



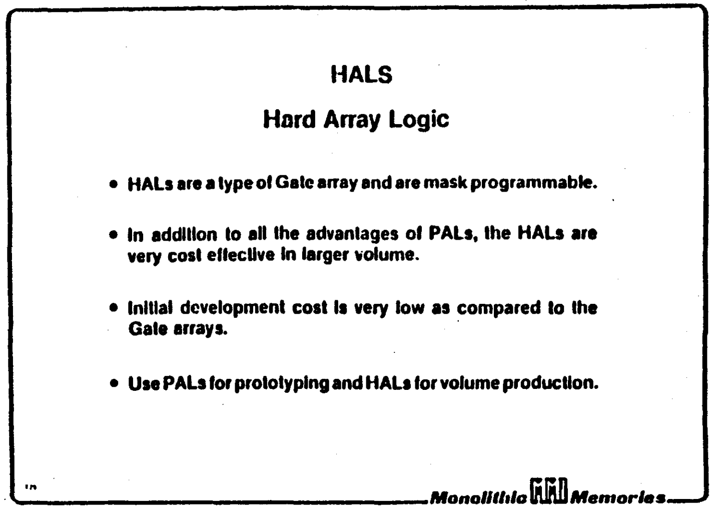
FUSE PROGRAMMABLE LOGIC DEVICES

	AND	OR	OUTPUT OPTIONS
PROM	FIXED	PROG	TS, OC
FPLA	PROG	PROG	TS, OC, FUSIBLE POLARITY
FPGA	PROG	NONE	TS, OC, FUSIBLE POLARITY
PMUX	FIX/PROG	FIXED	TS
PAL	PROG	FIXED	TS, REGISTERED, FEEDBACK, I/O

Monolithic Memories

THE PROGRAMMABLE SOLUTION

- FILLING THE GAP BETWEEN SSI/MSI AND CUSTOM
- SSI/MSI HAS LIMITED INTEGRATION CAPABILITY
- CUSTOM REQUIRES LARGE VOLUME AND LONG DESIGN CYCLE



HALS

Hard Array Logic

- HALs are a type of Gate array and are mask programmable.
- In addition to all the advantages of PALs, the HALs are very cost effective in larger volume.
- Initial development cost is very low as compared to the Gate arrays.
- Use PALs for prototyping and HALs for volume production.

Monolithic Memories

PROGRAMMABILITY MASK vs FUSE?

	MASK	FUSE
SOFTWARE	EXTENSIVE	SIMPLE EQUATIONS TO FUSE PATTERN
FLEXIBILITY	FIXED-CHANGE IS DIFFICULT	EASILY CHANGED
DESIGN CYCLE	4-6 months	INSTANT
PROTOTYPING	BREADBOARD AND SOFTWARE REQUIRE AN ELABORATE DEVELOPMENT SYSTEM. TRANSFERRED TO MASK AT VENDOR	SIMPLE SOFTWARE AND LOW COST PROGRAMMER AT CUSTOMER OR DISTRIBUTOR
COST	LONG TERM COST EFFECTIVE	IMPROVEMENT OVER SSI.MSI TTL
CUSTOMER FIT	MEDIUM TO LARGE COMPANIES	ANYONE
2ND SOURCE	GENERALLY NONE	LICENSED 2ND SOURCES

PAL Training Seminar Supplement

I. Boolean Algebra Review

Commutative Property	$A \cdot B = B \cdot A$ $A + B = B + A$
Associative Property	$A \cdot (B \cdot C) = (A \cdot B) \cdot C$ $A + (B + C) = (A + B) + C$
Distributive Property	$A \cdot (B + C) = A \cdot B + A \cdot C$ $A + B \cdot C = (A + B) \cdot (A + C)$

Postulates

$0 \cdot 0 = 0$	$1 \cdot 0 = 0$	$0 + 0 = 0$
$1 + 1 = 1$	$0 + 1 = 1$	$1 \cdot 1 = 1$
$\bar{0} = 1$	$\bar{1} = 0$	

Theorems.

$A + 0 = A$	$A \cdot A = A$	$A + A = A$
$A \cdot 1 = A$	$\overline{(A)} = \bar{A}$	$A + \bar{A} = 1$
$A + 1 = 1$	$\overline{(\bar{A})} = A$	
$A \cdot 0 = 0$	$A \cdot \bar{A} = 0$	

$$\overline{(A + B + C + \ldots)} = \bar{A} \cdot \bar{B} \cdot \bar{C} \cdot \ldots$$
$$A + B + C + \ldots = \overline{(\bar{A} \cdot \bar{B} \cdot \bar{C} \cdot \ldots)} \qquad \text{— De Morgan}$$
$$\overline{(A \cdot B \cdot C \cdot \ldots)} = \bar{A} + \bar{B} + \bar{C} + \ldots$$
$$A \cdot B \cdot D \cdot \ldots = \overline{(\bar{A} + \bar{B} + \bar{C} + \ldots)} \qquad \text{— De Morgan}$$

$$A \cdot (A + B) = A + A \cdot B = A$$
$$A + A \cdot B = A \cdot (A + B) = A$$

II. Karnaugh Map Matrix Lables

Top to bottom or left to right:

2-Variable	3-Variable	4-Variable
00	000	Add a '0' MSB and
01	001	use the 3-variable
11	011	chart for the first
10	010	half For the second
	110	half, add a '1' MSB
	111	and repeat the same
	101	chart in reverse order.
	100	

This same technique may be used for any number of variables that may be desired per axis For any axis greater than one variable, the second-half is a mirror image of the first-half with the MSB equal to a one (1) This can be seen above when comparing the 3-variable list to the 2-variable list.

Note	$\cdot$ = AND	+ = OR	/ = NOT

2-Bit Counter Example

A = LSB B = MSB

For Up Count	For Down Count
B A	B A
0 0	1 1
0 1	1 0
1 0	0 1
1 1	0 0
0 0	1 1

Using the verbal logic language, equations can actually be written directly from the function tables

The equation for A is fairly obvious. A always toggles. thus.

$$\bar{A} := A \qquad \text{; Toggle}$$

The equation for B has only two general functions They are hold ($\bar{B} := \bar{B}$) and toggle ($\bar{B} := B$). We also have two modes of count (Up and Down). By inspection of the function tables the following equation is written for B.

$$\begin{aligned} \bar{B} := {} & \bar{B} \cdot \bar{A} \cdot UP && \text{; Hold on Up} \\ + {} & \bar{B} \cdot A \cdot \overline{UP} && \text{; Hold on Down} \\ + {} & B \cdot A \cdot UP && \text{; Toggle Up} \\ + {} & B \cdot \bar{A} \cdot \overline{UP} && \text{; Toggle Down} \end{aligned}$$

Please note that the output pins are assumed to be called A and B. Thus, the equations must be written for $\bar{A}$ and $\bar{B}$. If you use A Karnaugh Map to solve the above problem, merely write your equations for the zero's. This will result in exactly the same solution as shown above

To demonstrate the flexibility of PAL we will now 'factor in' a preset and clear (both active high).

$$\begin{aligned} \bar{B} := {} & \bar{B} \cdot \bar{A} \cdot UP \cdot \overline{PRS} && \text{; Hold on Up} \\ + {} & \bar{B} \cdot A \cdot \overline{UP} \cdot \overline{PRS} && \text{; Hold on Down} \\ + {} & B \cdot A \cdot UP \cdot \overline{PRS} && \text{; Toggle Up} \\ + {} & B \cdot \bar{A} \cdot \overline{UP} \cdot \overline{PRS} && \text{; Toggle Down} \\ + {} & CLR && \text{; Clear} \end{aligned}$$

To add a load function merely requires another product line.

$$\begin{aligned} \bar{B} := {} & D1 \cdot \qquad\qquad \cdot LD && \text{; Load} \\ + {} & \bar{B} \cdot \bar{A} \cdot UP \cdot \overline{PRS} \cdot \overline{LD} && \text{; Hold on Up} \\ + {} & \bar{B} \cdot A \cdot \overline{UP} \cdot \overline{PRS} \cdot \overline{LD} && \text{; Hold on Down} \\ + {} & B \cdot A \cdot UP \cdot \overline{PRS} \cdot \overline{LD} && \text{. Toggle Up} \\ + {} & B \cdot \bar{A} \cdot \overline{UP} \cdot \overline{PRS} \cdot \overline{LD} && \text{. Toggle Down} \\ + {} & CLR && \text{; Clear} \end{aligned}$$

Verbal Logic Language

Symbol	Operator	Pronounced
:=	Equality	Replaced by
=	Equality	Equals
	Not	Not
+	Or	Or
$\cdot$	And	First — If Others — And
+	XOR	Unless — XOR PALS Only
$\cdot$	And	After Unless — And

$$\begin{aligned} \overline{Q3} := {} & \overline{D3} \cdot LD && \text{. Load} \\ + {} & \overline{Q3} \cdot \overline{LD} && \text{. Hold} \\ + {} & UP \cdot \overline{LD} \cdot Q2 \cdot Q1 \cdot Q0 && \text{. Toggle Up} \\ + {} & \overline{Up} \cdot \overline{LD} \cdot \overline{Q2} \cdot \overline{Q1} \cdot \overline{Q0} && \text{. Toggle Down} \end{aligned}$$

The above equation can be read easier by most people using a verbal approach as opposed to a Boolean approach

This happens to be the PAL equation for the three-bit of a 10-bit up/down counter with load

Q3 is replaced by D3 (load function) if load (LD) is asserted Or Q3 is replaced by Q3 (hold function) if load is not asserted unless all lesser significant bits are high and we are counting up or all lesser significant bits are low and we are counting down

This equation was written for a PAL 20X10 The XOR actually transforms the D-register into a toggle F F

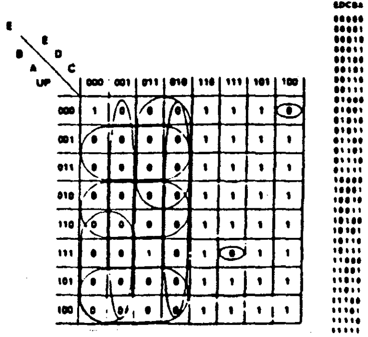
This same function could be written without the aid of the XOR gate However, it would require more product terms More on this when we get to the power of the XOR gate.

To include a preset function in this counter merely 'and each term with $\overline{PRS}$ for an active high preset, or with PRS for an active low preset

In general a preset costs an input line and a clear costs a product line

The Power of the Exclusive-OR

Shown below is the Karnaugh Map of the four-bit of a ten-bit up/down counter



On the right are the outputs of the five low order bits These have been mapped as shown above

To solve this not using the XOR gate requires nine product terms This is due to the exceptions scattered throughout the map

The equation for this output is as follows

$$\begin{aligned} \bar{E} = {} & \bar{E} \cdot \bar{B} \cdot UP && \text{. 1 (7 Holds)} \\ + {} & \bar{E} \cdot A \cdot \overline{UP} && \text{. 2} \\ + {} & \bar{E} \cdot B \cdot \bar{A} && \text{3} \\ + {} & \bar{E} \cdot D \cdot \bar{B} && \text{. 4} \\ + {} & \bar{E} \cdot \bar{D} \cdot B && \text{. 5} \\ + {} & \bar{E} \cdot \bar{D} \cdot C && \text{. 6} \\ + {} & \bar{E} \cdot D \cdot \bar{C} && \text{. 7} \\ + {} & E \cdot D \cdot C \cdot B \cdot A \cdot UP && \text{. 8 (Toggle Up)} \\ + {} & E \cdot \bar{D} \cdot \bar{C} \cdot \bar{B} \cdot \bar{A} \cdot \overline{UP} && \text{. 9 (Toggle Down)} \end{aligned}$$

Now let's look at the same Karnaugh Map in a different light.
Note that the entire left half is zeros with two exceptions
And that the right half is all ones with two exceptions

Also note that this is the value of E Thus this entire Map — with four exceptions — reduces to

$$\bar{E} = \bar{E} \qquad \text{. Hold State}$$

The exclusive or operator is the "exception" operator Using our verbal logic language, we could state that we want the value of E to hold unless we are in one of the four exception cells

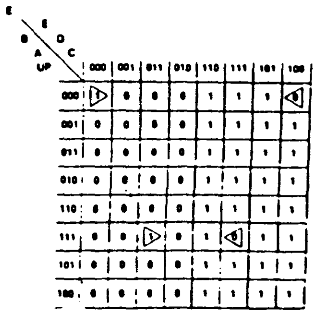
Closer examination reveals that the four cells actually reduce to two pairs of cells These pairs are

$$D \cdot C \cdot B \cdot A \cdot UP \qquad \text{AND}$$
$$\bar{D} \cdot \bar{C} \cdot \bar{B} \cdot \bar{A} \cdot \overline{UP}$$

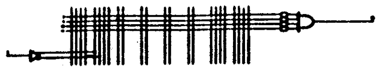
Hence our entire equation that was nine product terms is now reduced to three product terms as follows

$$\begin{aligned} \bar{E} = {} & \bar{E} && \text{. Hold} \\ + {} & D \cdot C \cdot B \cdot A \cdot UP && \text{. Toggle Up} \\ + {} & \bar{D} \cdot \bar{C} \cdot \bar{B} \cdot \bar{A} \cdot \overline{UP} && \text{. Toggle Down} \end{aligned}$$

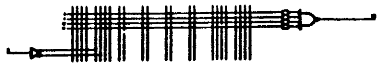
This can be read as follows
Not E is replaced by Not E unless all lesser significant bits are high and we are counting up or all lesser significant bits are low and we are counting down



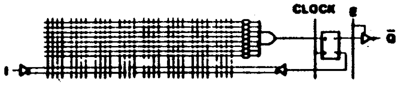
AND-OR ARRAY



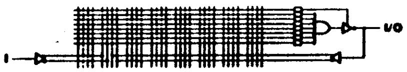
AND-NOR ARRAY



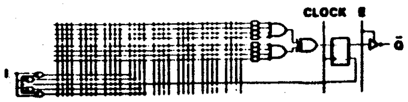
REGISTER WITH FEEDBACK CELL



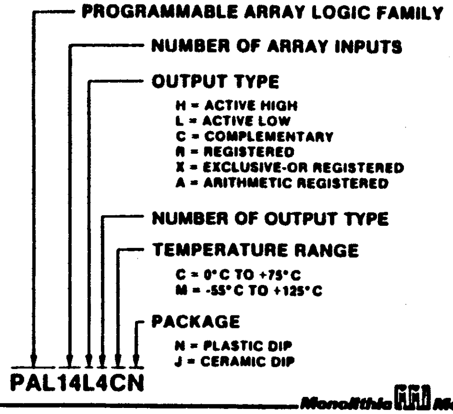
PROGRAMMABLE I/O



REGISTERED ARITHMETIC CELL



ORDERING INFORMATION



PROGRAMMABLE ARRAY LOGIC FAMILY

NUMBER OF ARRAY INPUTS

OUTPUT TYPE

H = ACTIVE HIGH
L = ACTIVE LOW
C = COMPLEMENTARY
R = REGISTERED
X = EXCLUSIVE-OR REGISTERED
A = ARITHMETIC REGISTERED

NUMBER OF OUTPUT TYPE

TEMPERATURE RANGE

C = 0°C TO +75°C
M = -55°C TO +125°C

PACKAGE

N = PLASTIC DIP
J = CERAMIC DIP

PAL14L4CN

PAL TECHNOLOGY

BIPOLAR TTL SCHOTTKY PROCESS

PT - SI SCHOTTKY

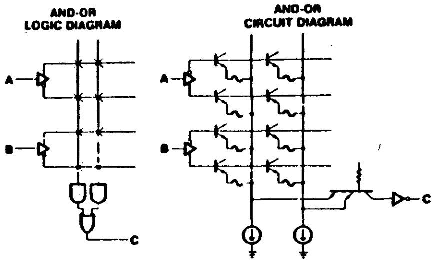
TI - W FUSE

SINGLE LAYER AND DUAL LAYER METAL

EMITTER FOLLOWER ARRAY

20 PIN SKINNY DIP (0.3" WIDE) PACKAGE

PAL ARRAY FORMAT



Monolithic Memories

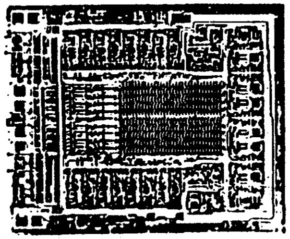
FUSIBLE LINKS

UNPROGRAMMED FUSE

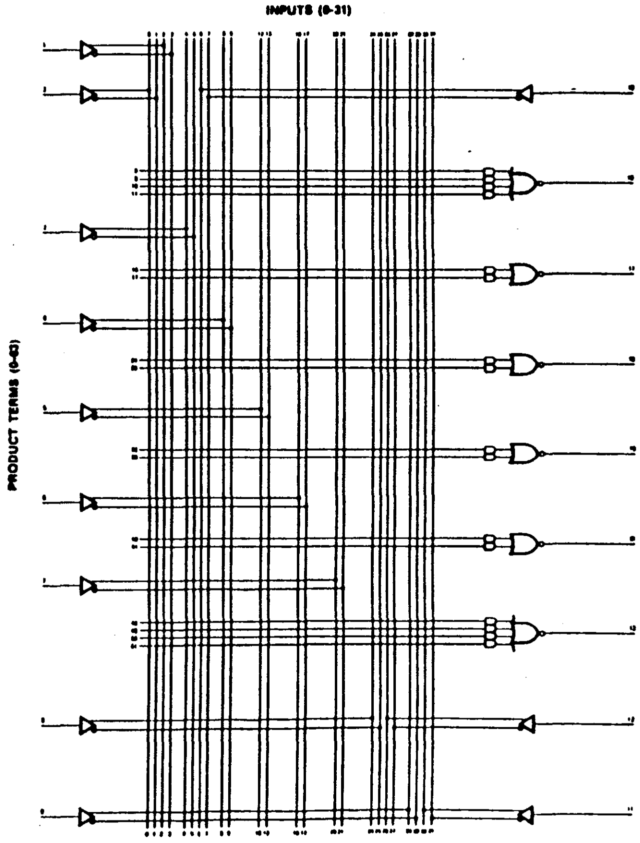
PROGRAMMED FUSE

Monolithic Memories

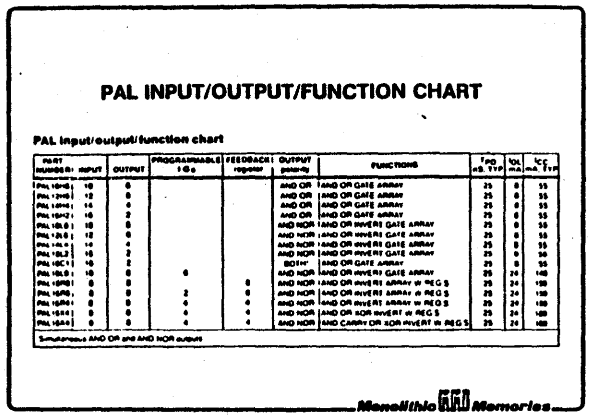
PAL


Monolithic Memories

LOGIC DIAGRAM PAL12L6



Monolithic Memories

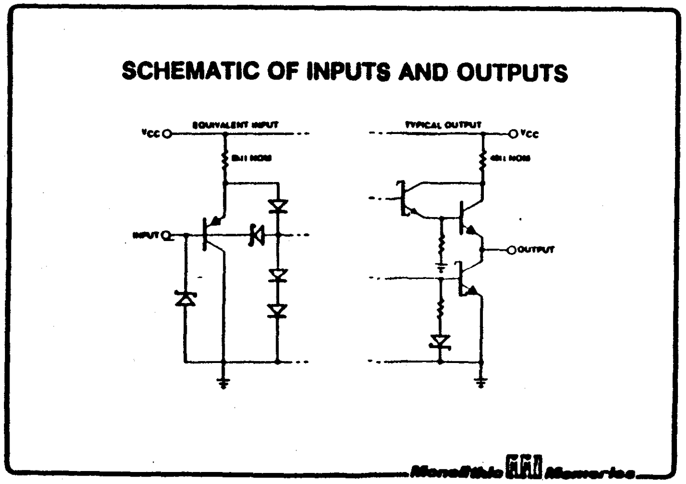


PAL INPUT/OUTPUT/FUNCTION CHART

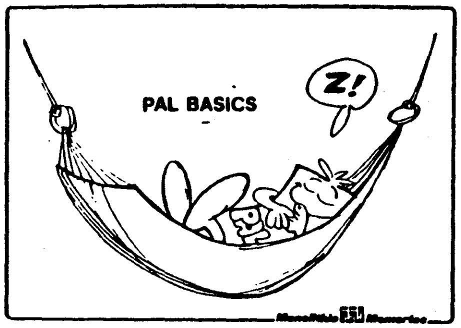
PAL input/output/function chart

PART NUMBER	INPUT	OUTPUT	PROGRAMMABLE I/Os	FEEDBACK register	OUTPUT polarity	FUNCTIONS	tPD nS. TYP	IOL mA	ICC mA. TYP
PAL 10H8	10	8			AND OR	AND OR GATE ARRAY	25	8	55
PAL 12H6	12	6			AND OR	AND OR GATE ARRAY	25	8	55
PAL 14H4	14	4			AND OR	AND OR GATE ARRAY	25	8	55
PAL 16H2	16	2			AND OR	AND OR GATE ARRAY	25	8	55
PAL 10L8	10	8			AND NOR	AND OR INVERT GATE ARRAY	25	8	55
PAL 12L6	12	6			AND NOR	AND OR INVERT GATE ARRAY	25	8	55
PAL 14L4	14	4			AND NOR	AND OR INVERT GATE ARRAY	25	8	55
PAL 16L2	16	2			AND NOR	AND OR INVERT GATE ARRAY	25	8	50
PAL 16C1	16	2			BOTH*	AND OR GATE ARRAY	25	8	55
PAL 16L8	10	8	6		AND NOR	AND OR INVERT GATE ARRAY	25	24	140
PAL 16R8	8	8		8	AND NOR	AND OR INVERT ARRAY w REG S	25	24	120
PAL 16R6	8	8	2	6	AND NOR	AND OR INVERT ARRAY w REG S	25	24	120
PAL 16R4	8	8	4	4	AND NOR	AND OR INVERT ARRAY w REG S	25	24	120
PAL 16X4	8	8	4	4	AND NOR	AND OR XOR INVERT w REG S	25	24	160
PAL 16A4	8	8	4	4	AND NOR	AND CARRY OR XOR INVERT w REG S	25	24	160

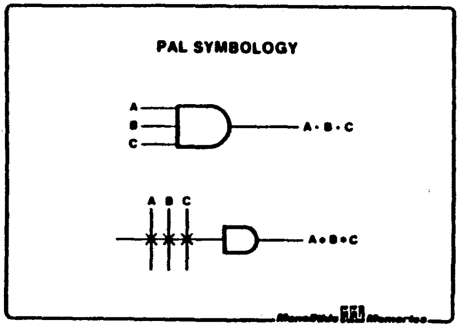
*Simultaneous AND OR and AND NOR outputs



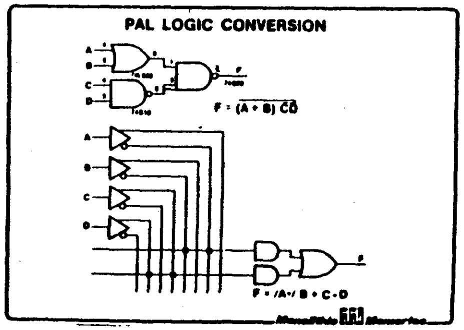
SCHEMATIC OF INPUTS AND OUTPUTS



PAL BASICS



PAL SYMBOLOGY



PAL LOGIC CONVERSION

PAL Concepts

PAL Legend

Constants

LOW (L)	NEGATIVE (N)	ZERO (0)	GND	FALSE	—+— FUSE NOT BLOWN
HIGH (H)	POSITIVE (P)	ONE (1)	V_{CC}	TRUE	—|— FUSE BLOWN

Operators (IN HIERARCHY OF EVALUATION)

;	COMMENT FOLLOWS
/	COMPLEMENT, PREFIX TO A PIN NAME
*	AND (PRODUCT)
+	OR (SUM)
:+:	XOR (EXCLUSIVE OR)
:*:	XNOR (EXCLUSIVE NOR)
( )	CONDITIONAL THREE-STATE (IF STATEMENT) OR FIXED SYMBOL
=	EQUALITY
:=	REPLACED BY AFTER THE LOW TO HIGH TRANSITION OF THE CLOCK

Equations Standard $O_1 = I_1 \overline{I_2} + \overline{I_1} I_2$ PALASM O1 = I1*/I2 + /I1*I2

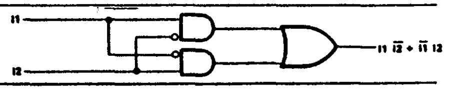
Function Table States

H = HIGH LEVEL
L = LOW LEVEL
X = IRRELEVANT
C = TRANSITION FROM LOW TO HIGH
Z = OFF (HIGH IMPEDANCE)

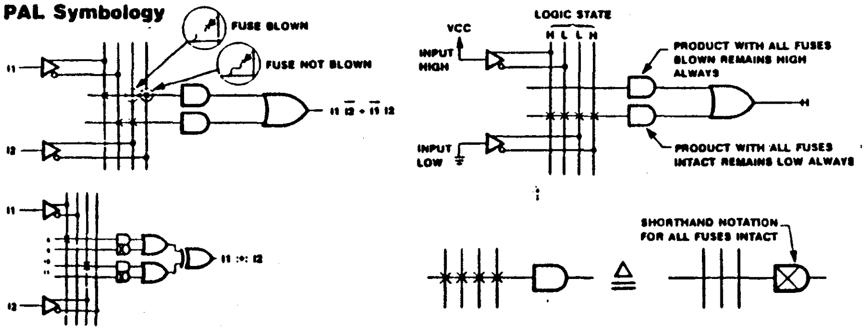
Test Conditions

H = TEST HIGH
L = TEST LOW
X = IRRELEVANT
1 = DRIVE HIGH
0 = DRIVE LOW
C = DRIVE INPUT FROM LOW TO HIGH
Z = TEST FOR HIGH IMPEDANCE

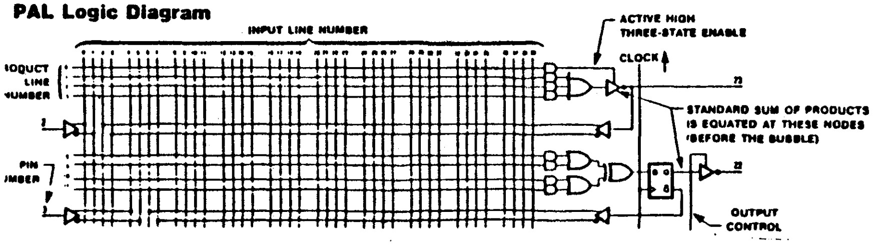
Conventional Symbology



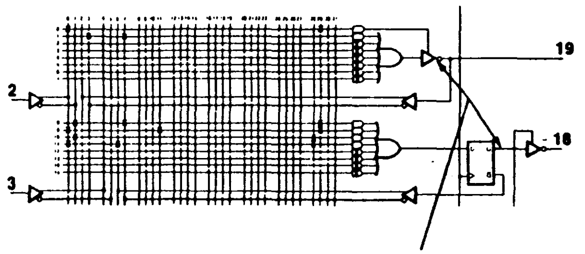
PAL Symbology


FUSE BLOWN
FUSE NOT BLOWN
I1
I2
I1 $\overline{I2}$ + $\overline{I1}$ I2
LOGIC STATE
H L L H
VCC
INPUT HIGH
INPUT LOW
PRODUCT WITH ALL FUSES BLOWN REMAINS HIGH ALWAYS
PRODUCT WITH ALL FUSES INTACT REMAINS LOW ALWAYS
H
I1 :*: I2
SHORTHAND NOTATION FOR ALL FUSES INTACT

PAL Logic Diagram


INPUT LINE NUMBER
PRODUCT LINE NUMBER
PIN NUMBER
1
2
ACTIVE HIGH THREE-STATE ENABLE
CLOCK
STANDARD SUM OF PRODUCTS IS EQUATED AT THESE NODES (BEFORE THE BUBBLE)
OUTPUT CONTROL
22
23

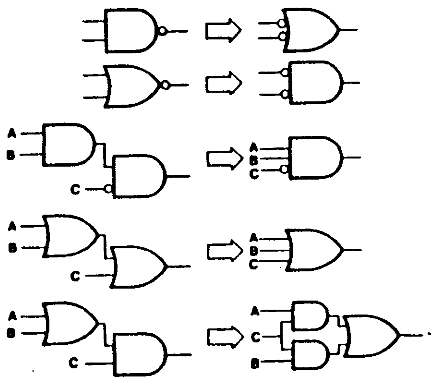
STANDARD PAL LOGIC EQUATION



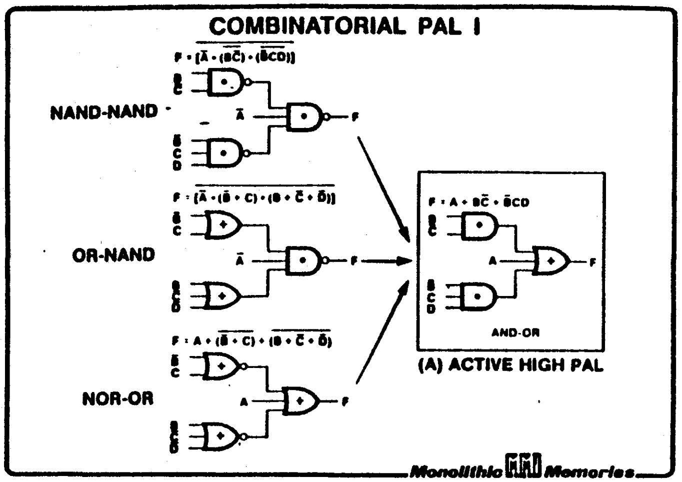
STANDARD SUM OF PRODUCTS IS EQUATED AT THESE NODES (BEFORE THE BUBBLE)

Monolithic Memories

BUBBLE THEORY



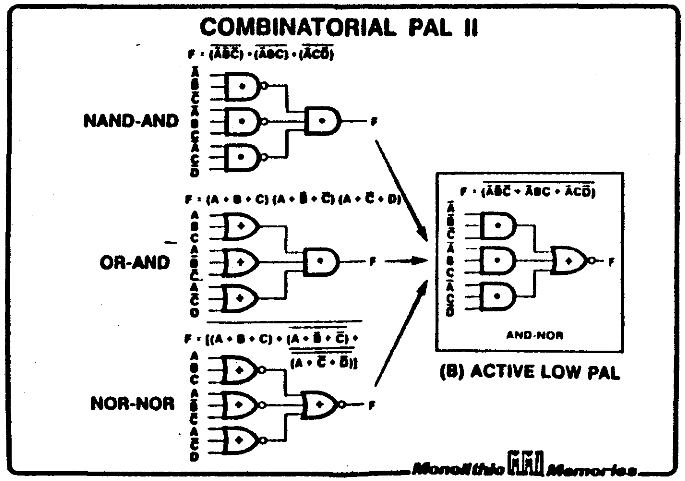
Monolithic Memories



COMBINATORIAL PAL I

$F = \overline{[\bar{A} \cdot (\overline{B\bar{C}}) \cdot (\overline{\bar{B}CD})]}$
NAND-NAND
B
C
$\bar{A}$
B
C
D
F
$F = \overline{[\bar{A} \cdot (\bar{B} + C) \cdot (B + \bar{C} + \bar{D})]}$
OR-NAND
B
C
$\bar{A}$
B
C
D
F
$F = A + \overline{(\bar{B} + C)} + \overline{(B + \bar{C} + \bar{D})}$
NOR-OR
B
C
A
B
C
D
F
$F = A + B\bar{C} + \bar{B}CD$
B
C
A
B
C
D
F
AND-OR

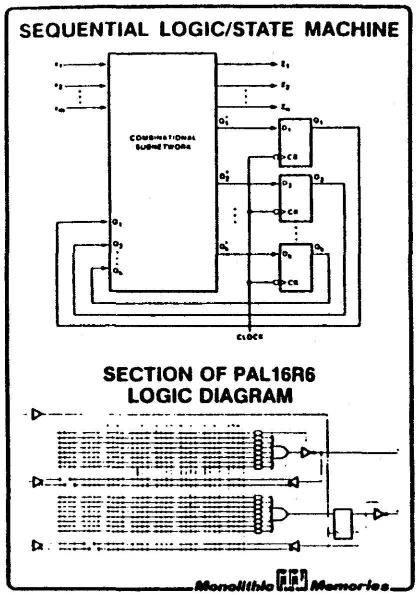
(A) ACTIVE HIGH PAL



COMBINATORIAL PAL II

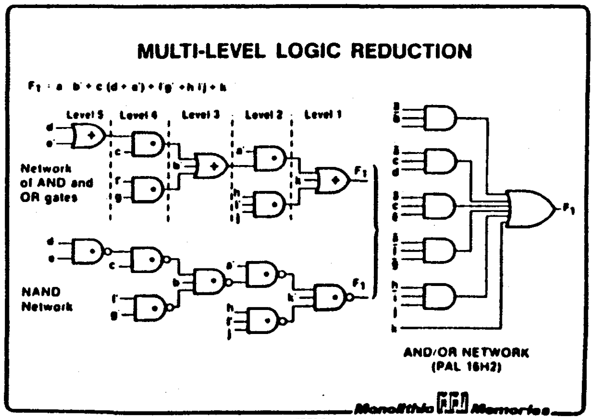
$F = \overline{(\bar{A}\bar{B}\bar{C})} \cdot \overline{(\bar{A}BC)} \cdot \overline{(\bar{A}C\bar{D})}$
NAND-AND
F
$F = (A + B + C)(A + \bar{B} + \bar{C})(A + \bar{C} + D)$
OR-AND
F
$F = \overline{[(A + B + C) + \overline{(A + \bar{B} + \bar{C})} + \overline{(A + \bar{C} + \bar{D})}]}$
NOR-NOR
F
$F = \overline{(\bar{A}\bar{B}\bar{C} + \bar{A}BC + \bar{A}C\bar{D})}$
F
AND-NOR

(B) ACTIVE LOW PAL



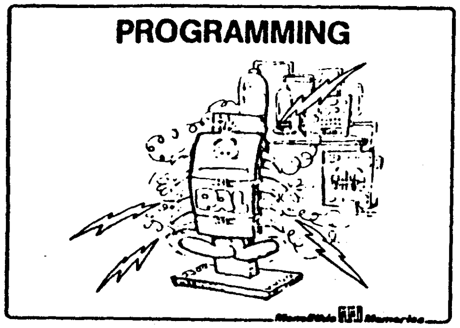
SEQUENTIAL LOGIC/STATE MACHINE

SECTION OF PAL16R6 LOGIC DIAGRAM

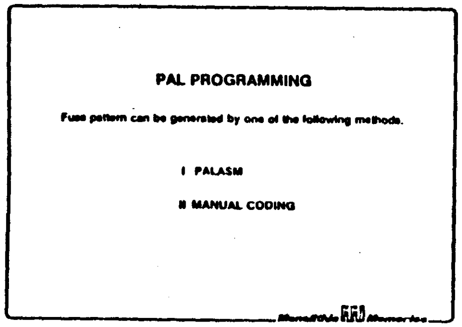


MULTI-LEVEL LOGIC REDUCTION

$F_1 : a \ b' + c(d + e') + fg' + hij + k$
Level 5
Level 4
Level 3
Level 2
Level 1
Network of AND and OR gates
NAND Network
F_1
F_1
F_1
AND/OR NETWORK (PAL 16H2)



PROGRAMMING

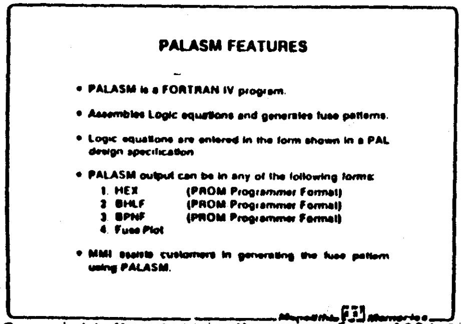


PAL PROGRAMMING

Fuse pattern can be generated by one of the following methods.

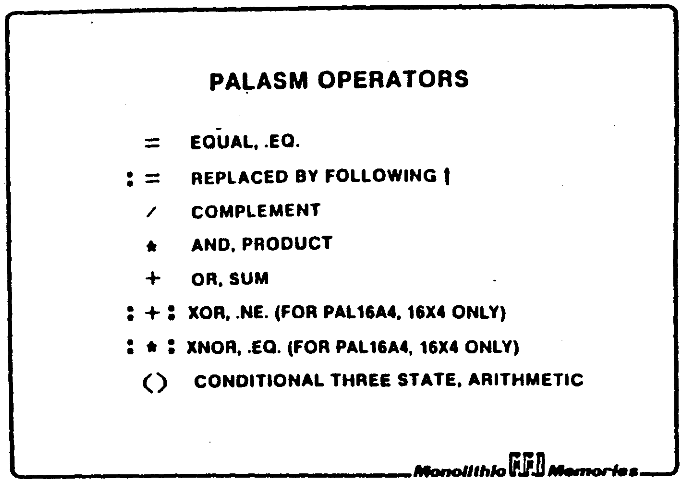
I PALASM

II MANUAL CODING



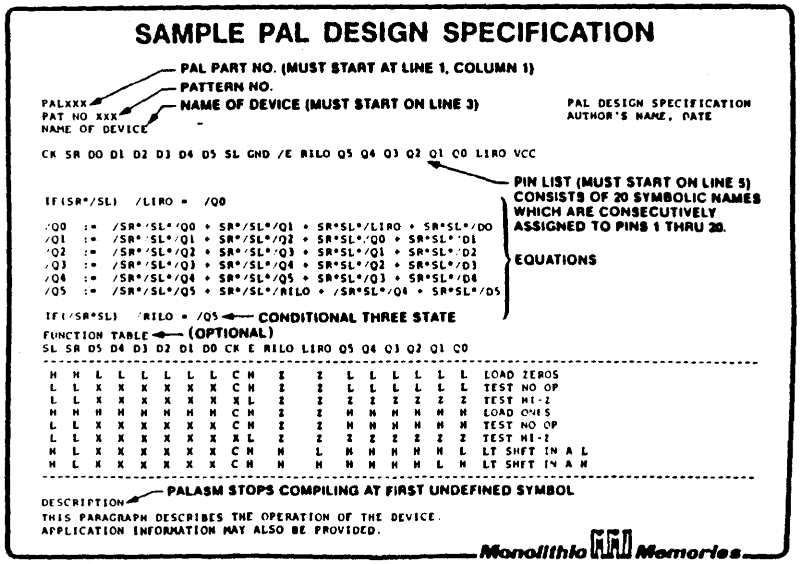
PALASM FEATURES

- PALASM is a FORTRAN IV program.
- Assembles Logic equations and generates fuse patterns.
- Logic equations are entered in the form shown in a PAL design specification
- PALASM output can be in any of the following forms:
  1. HEX (PROM Programmer Format)
  2. BHLF (PROM Programmer Format)
  3. BPNF (PROM Programmer Format)
  4. Fuse Plot
- MMI assists customers in generating the fuse pattern using PALASM.



PALASM OPERATORS

=	EQUAL, .EQ.
:=	REPLACED BY FOLLOWING !
/	COMPLEMENT
*	AND, PRODUCT
+	OR, SUM
:+:	XOR, .NE. (FOR PAL16A4, 16X4 ONLY)
:*:	XNOR, .EQ. (FOR PAL16A4, 16X4 ONLY)
( )	CONDITIONAL THREE STATE, ARITHMETIC



SAMPLE PAL DESIGN SPECIFICATION

```
PALXXX                                                    PAL DESIGN SPECIFICATION
PAT NO XXX                                                AUTHOR'S NAME, DATE
NAME OF DEVICE

CK SR D0 D1 D2 D3 D4 D5 SL GND /E RILO Q5 Q4 Q3 Q2 Q1 Q0 LIRO VCC

IF(SR*/SL)  /LIRO =  /Q0

/Q0   :=  /SR*/SL*/Q0 + SR*/SL*/Q1 + SR*SL*/LIRO + SR*SL*/D0
/Q1   :=  /SR*/SL*/Q1 + SR*/SL*/Q2 + SR*SL*/Q0 + SR*SL*/D1
/Q2   :=  /SR*/SL*/Q2 + SR*/SL*/Q3 + SR*SL*/Q1 + SR*SL*/D2
/Q3   :=  /SR*/SL*/Q3 + SR*/SL*/Q4 + SR*SL*/Q2 + SR*SL*/D3
/Q4   :=  /SR*/SL*/Q4 + SR*/SL*/Q5 + SR*SL*/Q3 + SR*SL*/D4
/Q5   :=  /SR*/SL*/Q5 + SR*/SL*/RILO + /SR*SL*/Q4 + SR*SL*/D5

IF(/SR*SL)  /RILO = /Q5
FUNCTION TABLE
SL SR D5 D4 D3 D2 D1 D0 CK E RILO LIRO Q5 Q4 Q3 Q2 Q1 Q0
--------------------------------------------------------------------------
 H  H  L  L  L  L  L  L  C H   Z    Z   L  L  L  L  L  L  LOAD ZEROS
 L  L  X  X  X  X  X  X  C H   Z    Z   L  L  L  L  L  L  TEST NO OP
 L  L  X  X  X  X  X  X  X L   Z    Z   Z  Z  Z  Z  Z  Z  TEST HI-Z
 H  H  H  H  H  H  H  H  C H   Z    Z   H  H  H  H  H  H  LOAD ONES
 L  L  X  X  X  X  X  X  C H   Z    Z   H  H  H  H  H  H  TEST NO OP
 L  L  X  X  X  X  X  X  X L   Z    Z   Z  Z  Z  Z  Z  Z  TEST HI-Z
 H  L  X  X  X  X  X  X  C H   H    L   H  H  H  H  H  L  LT SHFT IN A L
 H  L  X  X  X  X  X  X  C H   H    H   H  H  H  H  L  H  LT SHFT IN A H
--------------------------------------------------------------------------

DESCRIPTION
THIS PARAGRAPH DESCRIBES THE OPERATION OF THE DEVICE.
APPLICATION INFORMATION MAY ALSO BE PROVIDED.
```

PAL PART NO. (MUST START AT LINE 1, COLUMN 1)

PATTERN NO.

NAME OF DEVICE (MUST START ON LINE 3)

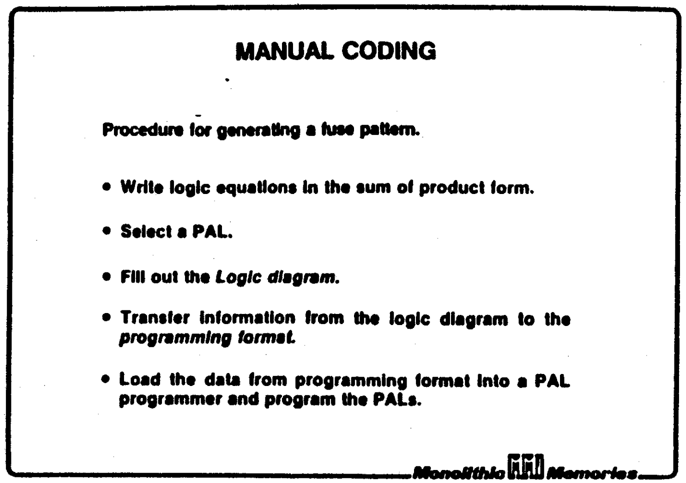
PIN LIST (MUST START ON LINE 5) CONSISTS OF 20 SYMBOLIC NAMES WHICH ARE CONSECUTIVELY ASSIGNED TO PINS 1 THRU 20.

EQUATIONS

CONDITIONAL THREE STATE

(OPTIONAL)

PALASM STOPS COMPILING AT FIRST UNDEFINED SYMBOL

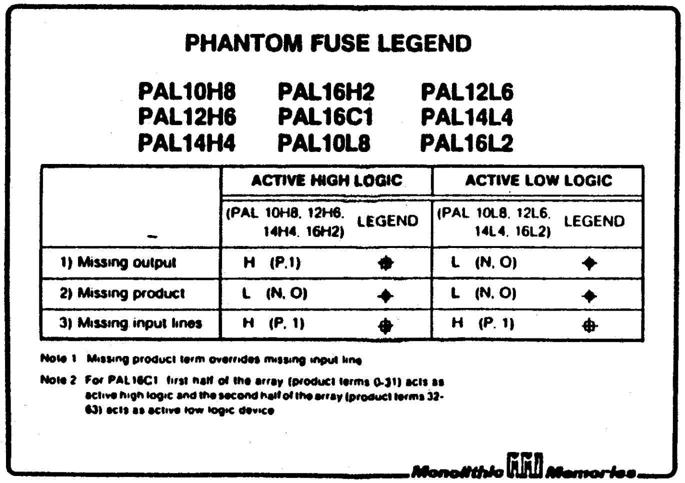


MANUAL CODING

Procedure for generating a fuse pattern.

- **Write logic equations in the sum of product form.**
- **Select a PAL.**
- **Fill out the *Logic diagram.***
- **Transfer information from the logic diagram to the *programming format.***
- **Load the data from programming format into a PAL programmer and program the PALs.**

Monolithic Memories



PHANTOM FUSE LEGEND

PAL10H8	PAL16H2	PAL12L6
PAL12H6	PAL16C1	PAL14L4
PAL14H4	PAL10L8	PAL16L2

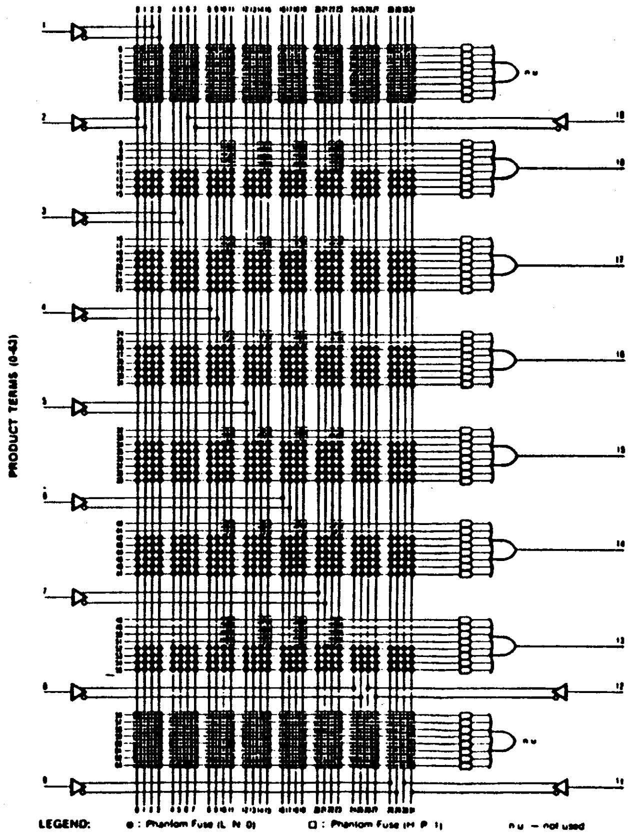
	ACTIVE HIGH LOGIC		ACTIVE LOW LOGIC	
-	(PAL 10H8, 12H6, 14H4, 16H2)	LEGEND	(PAL 10L8, 12L6, 14L4, 16L2)	LEGEND
1) Missing output	H (P,1)	⊕	L (N, O)	◆
2) Missing product	L (N, O)	◆	L (N, O)	◆
3) Missing input lines	H (P, 1)	⊕	H (P, 1)	⊕

Note 1 Missing product term overrides missing input line

Note 2 For PAL16C1 first half of the array (product terms 0-31) acts as active high logic and the second half of the array (product terms 32-63) acts as active low logic device

Monolithic Memories

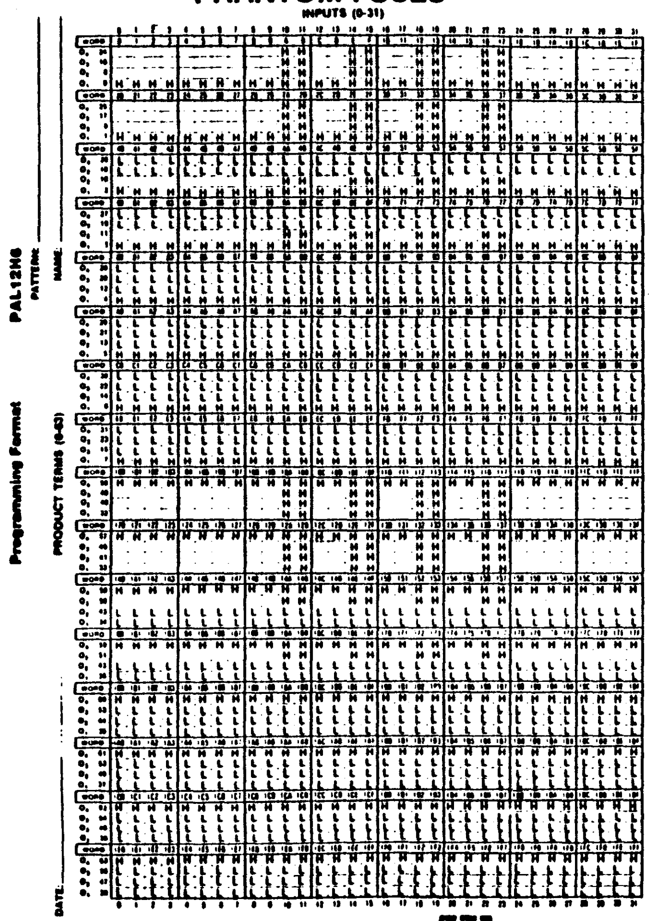
LOGIC DIAGRAM CONTAINING PHANTOM FUSES



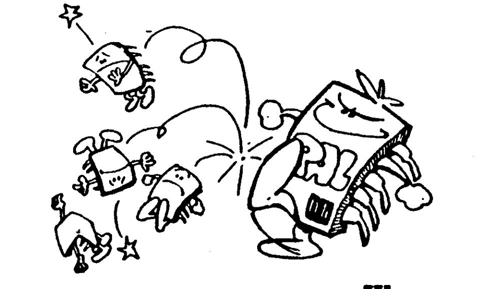
LEGEND: ● : Phantom Fuse (L, N, 0) □ : Phantom Fuse (H, P, 1) n.u. — not used

Monolithic Memories

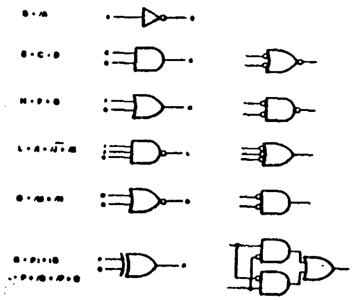
PROGRAMMING FORMAT CONTAINING PHANTOM FUSES


INPUTS (0-31)
PAL12H6
PATTERN
NAME
Programming Format
PRODUCT TERMS (0-63)
DATE

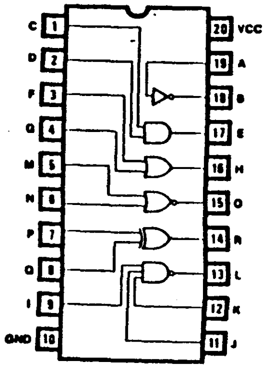
APPLICATIONS



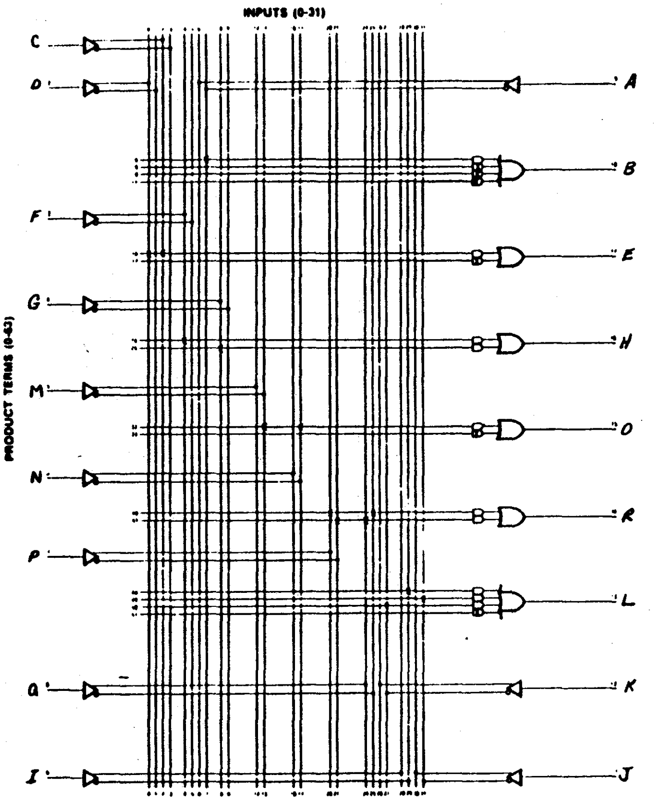
EXAMPLE — BASIC GATES



BASIC GATES PIN-OUT



LOGIC DIAGRAM-BASIC GATES



Monolithic Memories

DESIGN SPECIFICATION — PALASM OUTPUT: BASIC GATES

```
************************** P00211 ****************************
YOUR PAL DESIGN SPEC IS ACCEPTED AS MONOLITHIC MEMORIES BIT
PATTERN NUMBER P00211 . YOUR PASSWORD IS P0055A .
PLEASE CONFIRM THAT THESE EQUATIONS ARE CORRECT. MONOLITHIC
MEMORIES WILL NOT PROCESS A PRODUCTION ORDER WITHOUT PROPER
APPROVAL. USE BP NO. P00211 WHEN PLACING A PROGRAMMING ORDER.
**************************************************************

                     CUSTOMER CONFIRMATION
I HAVE REVIEWED THIS PAL DESIGN SPECIFICATION AND UNDERSTAND
THAT PALS ORDERED BY BIT PATTERN NUMBER P00211 WILL BE
PROGRAMMED AS SPECIFIED BELOW.
CUSTOMER COMPANY NAME  ___________   CITY.STATE ________
 AUTHORISED SIGNATURE  ___________         DATE ________
          NAME (PRINT) ___________        TITLE ________
RETURN TO MMI 1165 E. ARQUES SUNNYVALE, CA 94086 ATTN:_______
**************************************************************

PAL12H6                                PAL DESIGN SPECIFICATION
P0055A                                 KAZMI 11/21/79
BASIC GATES EXAMPLE
S.V. CA
C D F G M N  P Q I GND J K L R O H E B A VCC
 B=/A
 E=C*D
 H=F+G
 O=/M*/N
 R=P*/Q+/P*Q
 L=/I+/J+/K
DESCRIPTION: THIS EXAMPLE ILLUSTRATES THE USE OF FUSIBLE LOGIC TO
             IMPLEMENT THE BASIC GATES.
```

Monolithic Memories

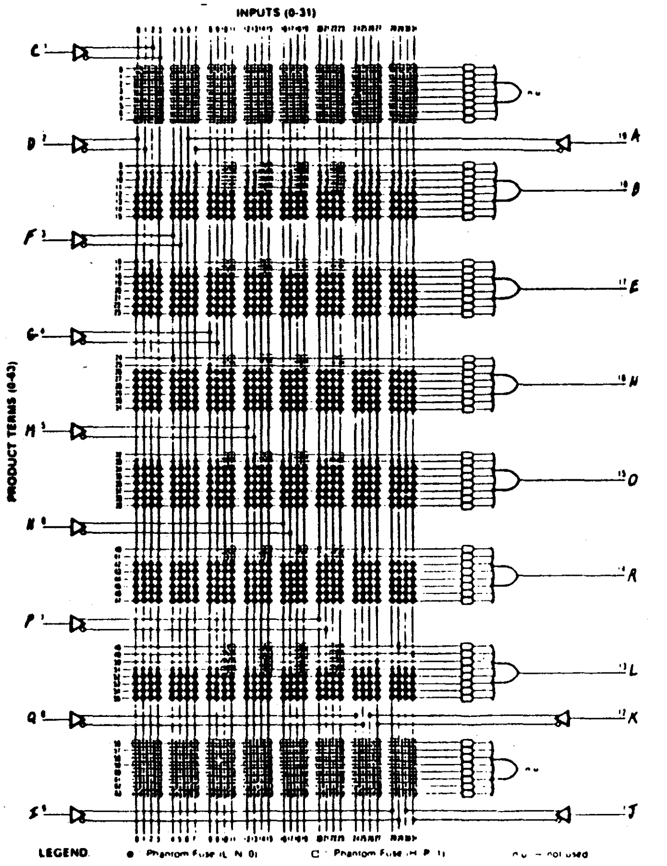
PALASM OUTPUT

HEX FORMAT **PAL12H6**

```
B F B F 7 F F D F F F F F F F F F F F F F F F F F F F F F F F F .
9 9 9 9 9 9 9 9 1 9 F F 9 9 F F 9 9 F F 9 9 F F 9 9 9 9 9 9 9 9 .
1 1 1 1 1 1 1 1 1 1 3 3 1 1 3 3 1 1 3 3 1 1 3 3 1 1 1 1 1 1 1 1 .
1 1 1 1 1 1 1 1 1 1 3 3 1 1 3 3 1 1 3 3 1 1 3 3 1 1 1 1 1 1 1 1 .
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 .
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 .
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 .
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 .
F F F F F F F F F F F F F E F F F E F F D F F F F D F F F B F F .
E E E E E E E E E E F F E E F F E E F F E C F F C E E E E E E A .
C C C C C C C C C C C C C C C C C C C C C C C C C C 0 C C C C C .
0 0 0 0 0 0 0 0 0 0 C C 0 0 C C 0 0 C C 0 0 C C 0 0 0 0 0 0 0 0 .
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 .
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 .
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 .
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 .
```

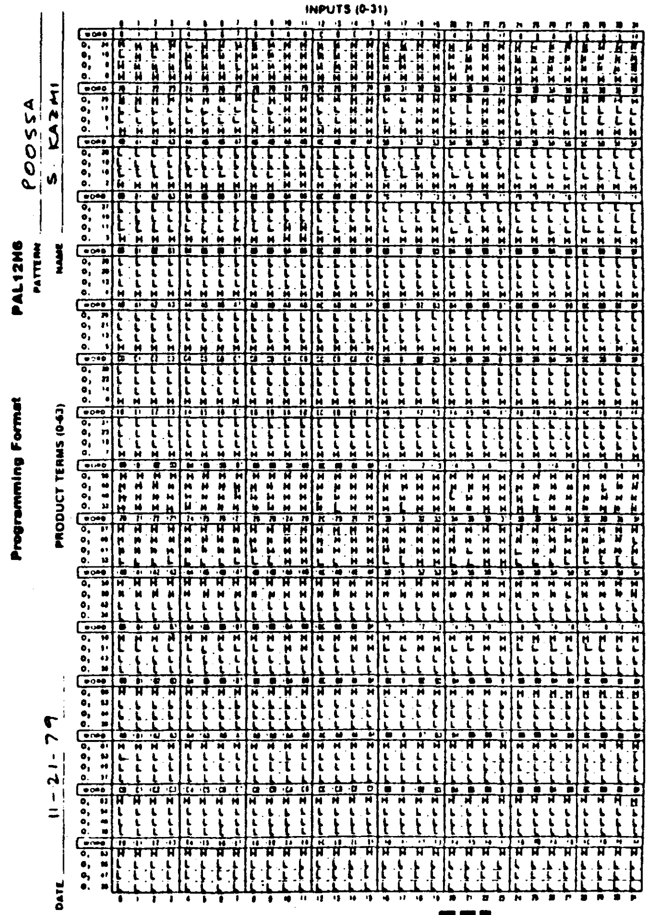
Monolithic Memories

LOGIC DIAGRAM — MANUAL CODING: BASIC GATES


INPUTS (0-31)
PRODUCT TERMS (0-63)
C
D
F
G
M
N
P
Q
S
A
B
E
H
O
R
L
K
J
n.u.
LEGEND ● Phantom Fuse (L, N, 0) ○ Phantom Fuse (H, P, 1) n.u. = not used

Monolithic MMI Memories

PROGRAMMING FORMAT — MANUAL CODING BASIC GATES


INPUTS (0-31)
PRODUCT TERMS (0-63)
WORD

Programming Format

PAL12H6

PATTERN P0055A

NAME S. KAZMI

DATE 11-21-79

Monolithic MMI Memories

PALASM OUTPUT: BASIC GATES

FUSE PLOT PAL12H6

```
                11 1111 1111 2222 2222 2233
     0123 4567 8901 2345 6789 0123 4567 8901

   0 OOOO OOOO OOOO OOOO OOOO OOOO OOOO OOOO
   1 OOOO OOOO OOOO OOOO OOOO OOOO OOOO OOOO
   2 OOOO OOOO OOOO OOOO OOOO OOOO OOOO OOOO
   3 OOOO OOOO OOOO OOOO OOOO OOOO OOOO OOOO
   4 OOOO OOOO OOOO OOOO OOOO OOOO OOOO OOOO
   5 OOOO OOOO OOOO OOOO OOOO OOOO OOOO OOOO
   6 OOOO OOOO OOOO OOOO OOOO OOOO OOOO OOOO
   7 OOOO OOOO OOOO OOOO OOOO OOOO OOOO OOOO

   8 ---- ---X --OO --OO --OO --OO ---- ---- /A
   9 XXXX XXXX XXOO XXOO XXOO XXOO XXXX XXXX
  10 XXXX XXXX XXOO XXOO XXOO XXOO XXXX XXXX
  11 XXXX XXXX XXOO XXOO XXOO XXOO XXXX XXXX
  12 0000 0000 0000 0000 0000 0000 0000 0000
  13 0000 0000 0000 0000 0000 0000 0000 0000
  14 0000 0000 0000 0000 0000 0000 0000 0000
  15 0000 0000 0000 0000 0000 0000 0000 0000

  16 X-X- ---- --OO --OO --OO --OO ---- ---- C*D
  17 XXXX XXXX XXOO XXOO XXOO XXOO XXXX XXXX
  18 0000 0000 0000 0000 0000 0000 0000 0000
  19 0000 0000 0000 0000 0000 0000 0000 0000
  20 0000 0000 0000 0000 0000 0000 0000 0000
  21 0000 0000 0000 0000 0000 0000 0000 0000
  22 0000 0000 0000 0000 0000 0000 0000 0000
  23 0000 0000 0000 0000 0000 0000 0000 0000

  24 ---- X--- --OO --OO --OO --OO ---- ---- F
  25 ---- ---- X-OO --OO --OO --OO ---- ---- G
  26 0000 0000 0000 0000 0000 0000 0000 0000
  27 0000 0000 0000 0000 0000 0000 0000 0000
  28 0000 0000 0000 0000 0000 0000 0000 0000
  29 0000 0000 0000 0000 0000 0000 0000 0000
  30 0000 0000 0000 0000 0000 0000 0000 0000
  31 0000 0000 0000 0000 0000 0000 0000 0000

  32 ---- ---- --OO -XOO -XOO --OO ---- ---- /M*/N
  33 XXXX XXXX XXOO XXOO XXOO XXOO XXXX XXXX
  34 0000 0000 0000 0000 0000 0000 0000 0000
  35 0000 0000 0000 0000 0000 0000 0000 0000
  36 0000 0000 0000 0000 0000 0000 0000 0000
  37 0000 0000 0000 0000 0000 0000 0000 0000
  38 0000 0000 0000 0000 0000 0000 0000 0000
  39 0000 0000 0000 0000 0000 0000 0000 0000

  40 ---- ---- --OO --OO --OO X-OO -X-- ---- P*/Q
  41 ---- ---- --OO --OO --OO -XOO X--- ---- /P*Q
  42 0000 0000 0000 0000 0000 0000 0000 0000
  43 0000 0000 0000 0000 0000 0000 0000 0000
  44 0000 0000 0000 0000 0000 0000 0000 0000
  45 0000 0000 0000 0000 0000 0000 0000 0000
  46 0000 0000 0000 0000 0000 0000 0000 0000
  47 0000 0000 0000 0000 0000 0000 0000 0000

  48 ---- ---- --OO --OO --OO --OO ---- -X-- /I
  49 ---- ---- --OO --OO --OO --OO ---- ---X /J
  50 ---- ---- --OO --OO --OO --OO ---X ---- /K
  51 XXXX XXXX XXOO XXOO XXOO XXOO XXXX XXXX
  52 0000 0000 0000 0000 0000 0000 0000 0000
  53 0000 0000 0000 0000 0000 0000 0000 0000
  54 0000 0000 0000 0000 0000 0000 0000 0000
  55 0000 0000 0000 0000 0000 0000 0000 0000

  56 OOOO OOOO OOOO OOOO OOOO OOOO OOOO OOOO
  57 OOOO OOOO OOOO OOOO OOOO OOOO OOOO OOOO
  58 OOOO OOOO OOOO OOOO OOOO OOOO OOOO OOOO
  59 OOOO OOOO OOOO OOOO OOOO OOOO OOOO OOOO
  60 OOOO OOOO OOOO OOOO OOOO OOOO OOOO OOOO
  61 OOOO OOOO OOOO OOOO OOOO OOOO OOOO OOOO
  62 OOOO OOOO OOOO OOOO OOOO OOOO OOOO OOOO
  63 OOOO OOOO OOOO OOOO OOOO OOOO OOOO OOOO

LEGEND:  X : FUSE NOT BLOWN (L,N,0)   - : FUSE BLOWN   (H,P,1)
         0 : PHANTOM FUSE   (L,N,0)   O : PHANTOM FUSE (H,P,1)
```

Monolithic Memories

MEMORY INTERFACE LOGIC FOR 6800 MICROPROCESSOR BUS

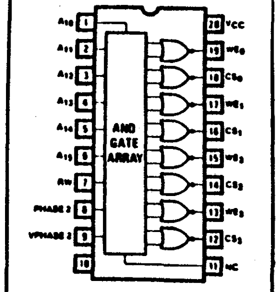
DESIGN SPECIFICATION PAL10L8

```
PAL10L8                                           PAL DESIGN SPECIFICATION
PAT NO 108                                        SAEED RAZMI        7-30-80
MEMORY INTERFACE LOGIC FOR 6800 MICROPROCESSOR BUS

A10 A11 A12 A13 A14 A15 RW PHASE2 VPHASE2 GND

NC /CS3 /WE3 /CS2 /WE2 /CS1 /WE1 /CS0 /WE0 VCC


WE0 = /A10*/A11*/A12*/A13*/A14*/A15*PHASE2*VPHASE2*/RW
CS0 = /A10*/A11*/A12*/A13*/A14*/A15*PHASE2
WE1 = A10*/A11*/A12*/A13*/A14*/A15*PHASE2*VPHASE2*/RW
CS1 = A10*/A11*/A12*/A13*/A14*/A15*PHASE2
WE2 =  A10*A11* A12* A11* A10* A15*PHASE2*VPHASE2*/RW
CS2 =  A10*A11* A12* A13* A14* A15*PHASE2
WE3 = A10*A11*/A12*/A13*/A14*/A15*PHASE2*VPHASE2*/RW
CS3 = A10*A11*/A12*/A13*/A14*/A15*PHASE2

DESCRIPTION:

THIS DEVICE PROVIDES THE INTERFACE LOGIC BETWEEN A 6800 MICROPROCESSOR
BUS AND EIGHT STATIC 1Kx4 MEMORY CHIPS.  ADDRESS BUS, PHASE2 AND
VPHASE2 ARE DECODED TO PRODUCE THE PROPER WRITE ENABLE AND CHIP SELECT
SIGNALS FOR MEMORY DATA TRANSFERS.
```

Monolithic Memories

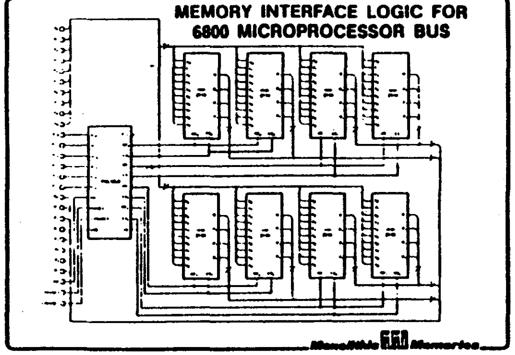
MEMORY INTERFACE LOGIC FOR 6800 MICROPROCESSOR BUS

PIN OUT

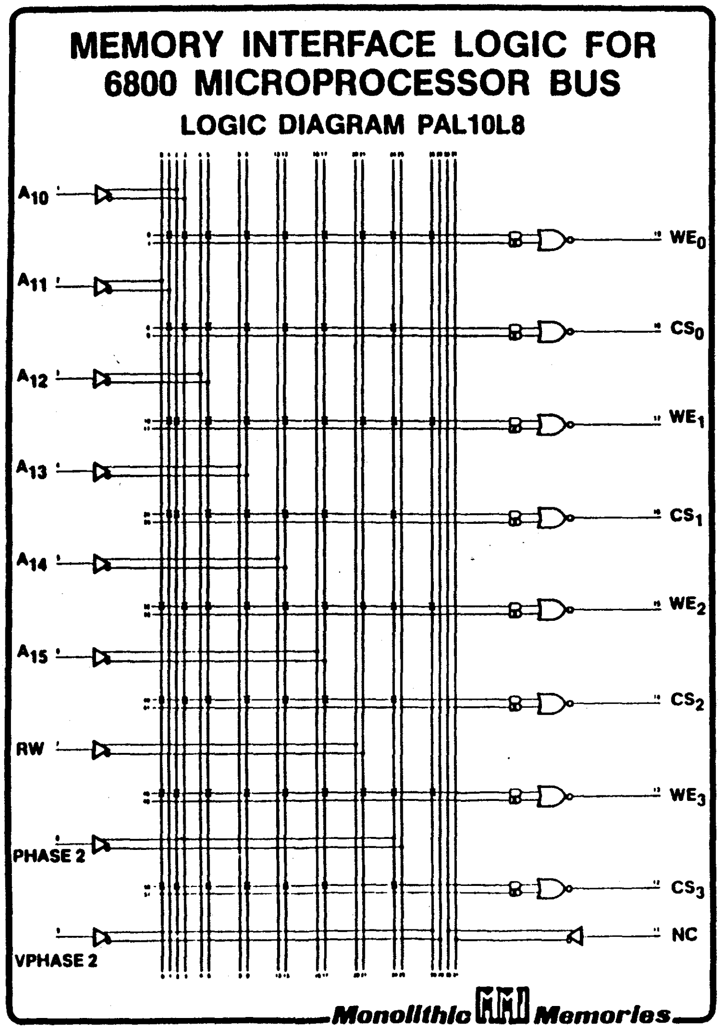


Monolithic Memories

MEMORY INTERFACE LOGIC FOR 6800 MICROPROCESSOR BUS

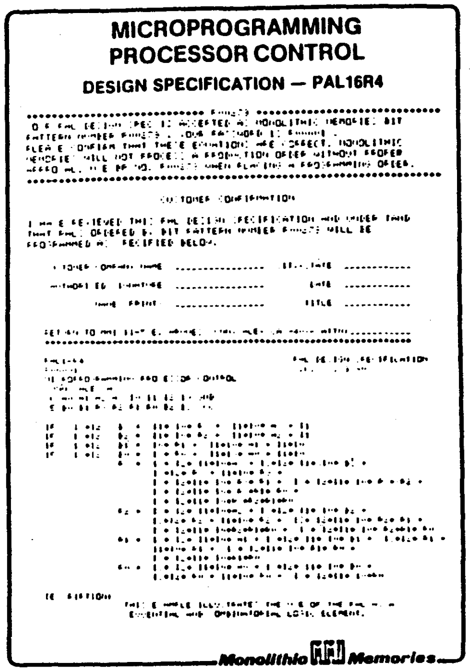


Monolithic Memories



MEMORY INTERFACE LOGIC FOR 6800 MICROPROCESSOR BUS

LOGIC DIAGRAM PAL10L8

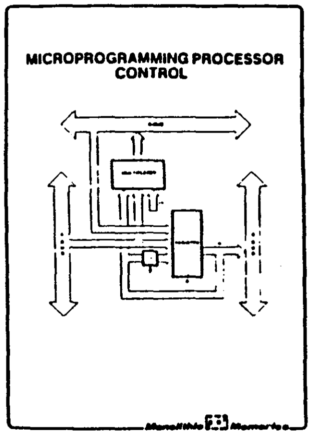


MICROPROGRAMMING PROCESSOR CONTROL

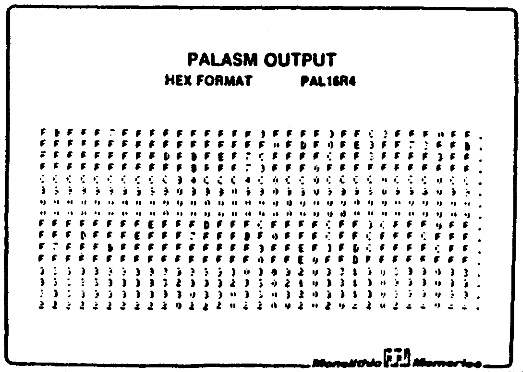
DESIGN SPECIFICATION — PAL16R4

[illegible]

Monolithic Memories



MICROPROGRAMMING PROCESSOR CONTROL



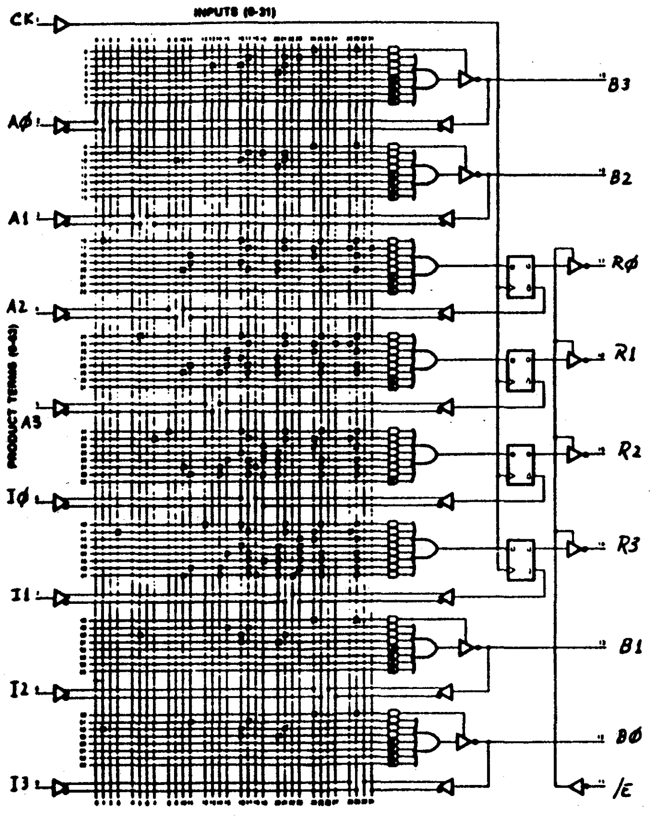
PALASM OUTPUT

HEX FORMAT PAL16R4

[illegible]

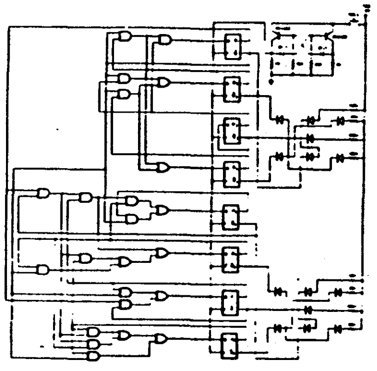
Monolithic Memories

LOGIC DIAGRAM — PAL16R4



Monolithic Memories

ELECTRONIC DICE GAME SSI/MSI IMPLEMENTATIONS



Monolithic Memories

PALASM OUTPUT

HEX FORMAT PAL16R4

[illegible]

Monolithic Memories

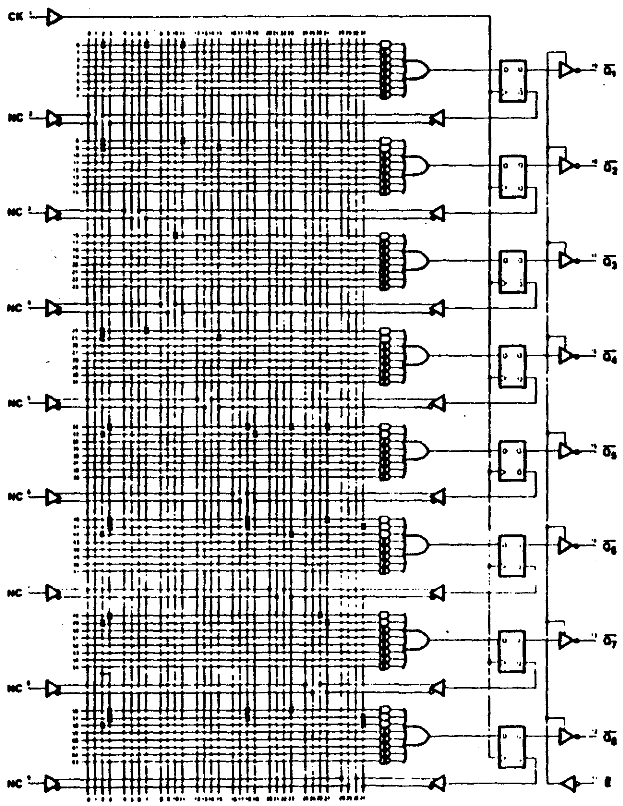
ELECTRONIC DICE GAME DESIGN SPECIFICATION PAL16R8

[illegible]

Monolithic Memories

ELECTRONIC DICE GAME

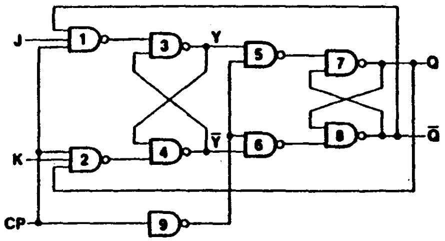
LOGIC DIAGRAM PAL16R8



Monolithic MMI Memories

CLOCKED MASTER SLAVE

JK FLIP FLOP



Monolithic MMI Memories

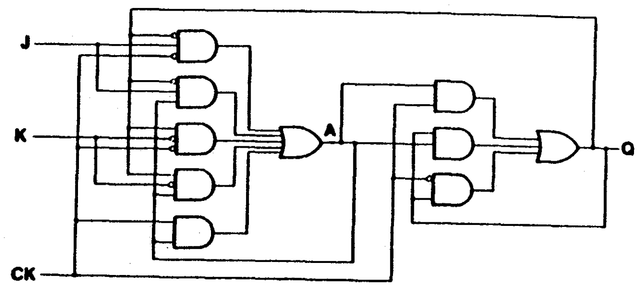
CLOCKED MASTER SLAVE JK FLIP FLOP

DESIGN SPECIFICATION PAL16L8

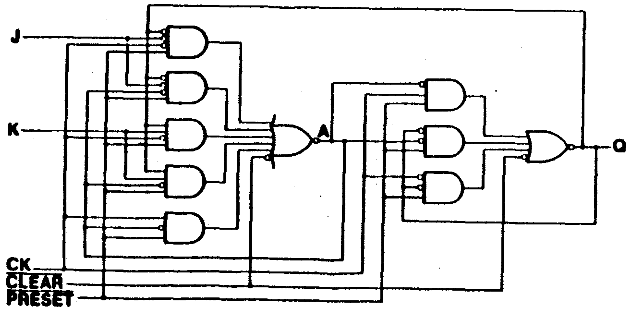
```
****************************** P00231 ***********************************
YOUR PAL DESIGN SPEC IS ACCEPTED AS MONOLITHIC MEMORIES BIT
PATTERN NUMBER P00231 . YOUR PASSWORD IS P0004A .
PLEASE CONFIRM THAT THESE EQUATIONS ARE CORRECT. MONOLITHIC
MEMORIES WILL NOT PROCESS A PRODUCTION ORDER WITHOUT PROPER
APPROVAL. USE SP NO. P00231 WHEN PLACING A PROGRAMMING ORDER. .
*************************************************************************
                         CUSTOMER CONFIRMATION

I HAVE REVIEWED THIS PAL DESIGN SPECIFICATION AND UNDERSTAND
THAT PALS ORDERED BY BIT PATTERN NUMBER P00231 WILL BE
PROGRAMMED AS SPECIFIED BELOW.

CUSTOMER COMPANY NAME  _______________  CITY, STATE  ___________

 AUTHORIZED SIGNATURE  _______________         DATE  ___________

         NAME (PRINT)  _______________        TITLE  ___________

RETURN TO MMI 1165 E. ARQUES SUNNYVALE, CA 94086 ATTN:__________
*************************************************************************
PAL16L8                                          PAL DESIGN SPECIFICATION
P0004A                                           [illegible] 3 21 84
CLOCKED MASTER SLAVE JK FLIP FLOP
SUNNYVALE CA
CK J K NC NC NC NC NC  PR GND  CLR NC NC NC NC NC Q A NC VCC

  A =  J*/Q*/CK*/PR +  J*/Q* A*/PR + K*Q*/CK*/PR +
       K*Q*/A*/PR   +   /A*CK*/PR    + CLR

  Q =  A*CK*/PR +  A*/Q*/PR +  /Q*/CK*/PR + CLR

DESCRIPTION:
             THIS IS AN EXAMPLE OF A CLOCKED MASTER SLAVE JK FLIP FLOP.
```

Monolithic MMI Memories

LOGIC CONVERSION I

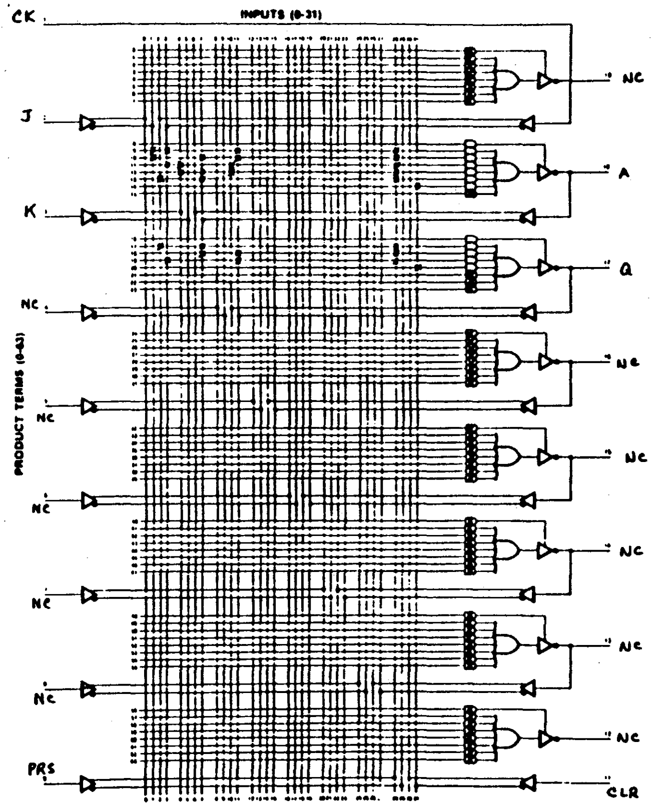


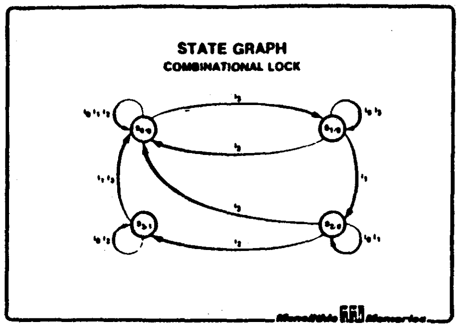
LOGIC CONVERSION II



CLOCKED MASTER SLAVE JK FLIP FLOP

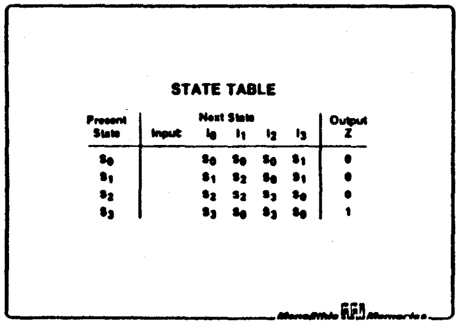
LOGIC DIAGRAM PAL16L8





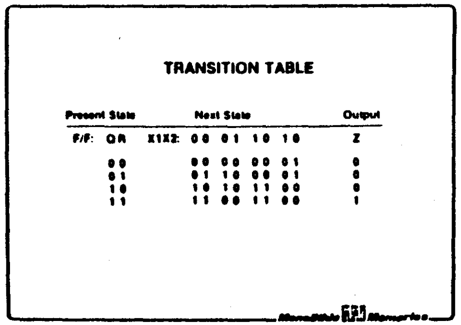
STATE GRAPH

COMBINATIONAL LOCK



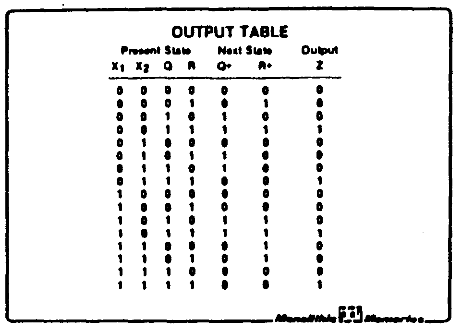
STATE TABLE

Present State	Input	Next State I_0	I_1	I_2	I_3	Output Z
S_0		S_0	S_0	S_0	S_1	0
S_1		S_1	S_2	S_0	S_1	0
S_2		S_2	S_2	S_3	S_0	0
S_3		S_3	S_0	S_3	S_0	1



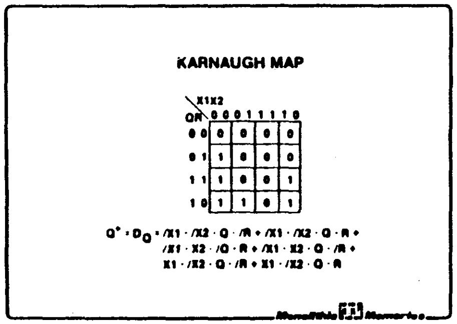
TRANSITION TABLE

Present State		Next State				Output
F/F: QR	X1X2:	0 0	0 1	1 0	1 0	Z
0 0		0 0	0 0	0 0	0 1	0
0 1		0 1	1 0	0 0	0 1	0
1 0		1 0	1 0	1 1	0 0	0
1 1		1 1	0 0	1 1	0 0	1



OUTPUT TABLE

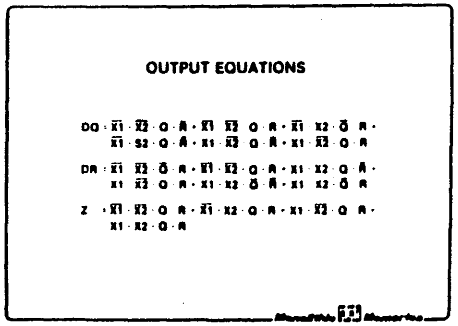
Present State				Next State		Output
X_1	X_2	Q	R	Q+	R+	Z
0	0	0	0	0	0	0
0	0	0	1	0	1	0
0	0	1	0	1	0	0
0	0	1	1	1	1	1
0	1	0	0	0	0	0
0	1	0	1	1	0	0
0	1	1	0	1	0	0
0	1	1	1	0	0	1
1	0	0	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	1	0
1	0	1	1	1	1	1
1	1	0	0	0	1	0
1	1	0	1	0	1	0
1	1	1	0	0	0	0
1	1	1	1	0	0	1



KARNAUGH MAP

QR \ X1X2	0 0	0 1	1 1	1 0
0 0	0	0	0	0
0 1	1	0	0	0
1 1	1	0	0	1
1 0	1	1	0	1

$$Q^{+} = D_Q = /X1 \cdot /X2 \cdot Q \cdot /R + /X1 \cdot /X2 \cdot Q \cdot R + /X1 \cdot X2 \cdot /Q \cdot R + /X1 \cdot X2 \cdot Q \cdot /R + X1 \cdot /X2 \cdot Q \cdot /R + X1 \cdot /X2 \cdot Q \cdot R$$



OUTPUT EQUATIONS

$$DQ = \overline{X1} \cdot \overline{X2} \cdot Q \cdot \overline{R} + \overline{X1}\ \overline{X2}\ Q \cdot R + \overline{X1}\ X2 \cdot \overline{Q}\ R +$$
$$\overline{X1} \cdot S2 \cdot Q \cdot \overline{R} + X1 \cdot \overline{X2}\ Q \cdot \overline{R} + X1 \cdot \overline{X2} \cdot Q \cdot R$$

$$DR = \overline{X1}\ \overline{X2} \cdot \overline{Q} \cdot R + \overline{X1} \cdot \overline{X2} \cdot Q \cdot \overline{R} + X1 \cdot X2 \cdot Q \cdot \overline{R} +$$
$$X1\ \overline{X2} \cdot Q \cdot R + X1 \cdot X2 \cdot \overline{Q} \cdot \overline{R} + X1\ X2 \cdot \overline{Q}\ R$$

$$Z = \overline{X1} \cdot \overline{X2} \cdot Q\ R + \overline{X1} \cdot X2 \cdot Q \cdot R + X1 \cdot \overline{X2} \cdot Q\ R +$$
$$X1 \cdot X2 \cdot Q \cdot R$$

COMBINATIONAL LOCK

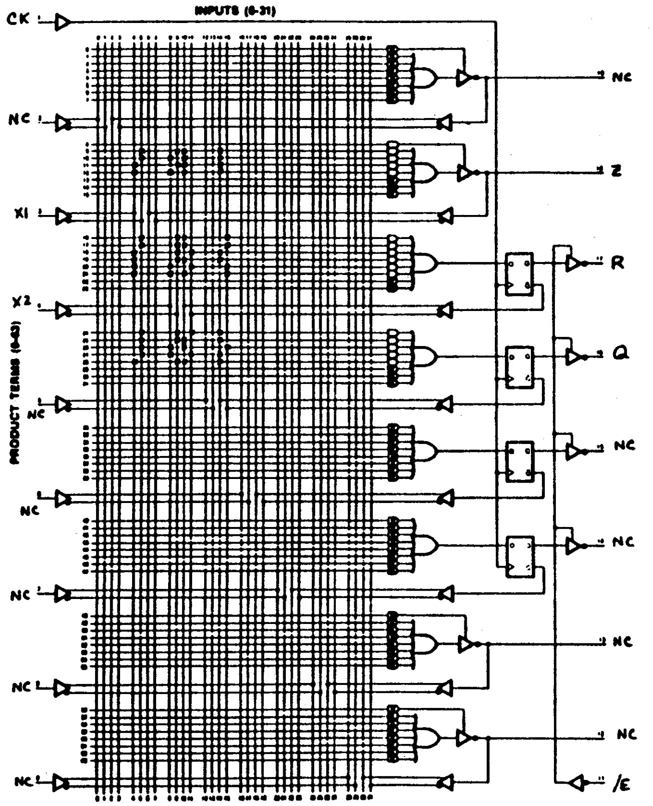
PAL DESIGN SPECIFICATIONS PAL16R4

```
**************************** P00279 ****************************
YOUR PAL DESIGN SPEC IS ACCEPTED AS MONOLITHIC MEMORIES BIT
PATTERN NUMBER P00279 . YOUR PASSWORD IS P0123A .
PLEASE CONFIRM THAT THESE EQUATIONS ARE CORRECT. MONOLITHIC
MEMORIES WILL NOT PROCESS A PRODUCTION ORDER WITHOUT PROPER
APPROVAL. USE BP NO. P00279 WHEN PLACING A PROGRAMMING ORDER.
****************************************************************

                    CUSTOMER CONFIRMATION

I HAVE REVIEWED THIS PAL DESIGN SPECIFICATION AND UNDERSTAND
THAT PALS ORDERED BY BIT PATTERN NUMBER P00279 WILL BE
PROGRAMMED AS SPECIFIED BELOW.

CUSTOMER COMPANY NAME  ----------------  CITY,STATE  -----------

 AUTHORIZED SIGNATURE  ----------------        DATE  -----------

         NAME (PRINT)  ----------------       TITLE  -----------

RETURN TO MMI 1165 E. ARQUES SUNNYVALE, CA 94086 ATTN:__________
****************************************************************

PAL16R4                                    PAL DESIGN SPECIFICATION
P0123A                                     SAEED KAZMI  10/26/79
DESIGN OF A COMBINATIONAL LOCK
SUNNYVALE CA
CK NC X1 X2 NC NC NC NC NC GND /E NC NC NC NC Q R Z NC VCC

  Z =  X1*/X2*Q*R + /X1*X2*Q*R + X1*/X2*Q*R + X1*X2*Q*R

  R =  X1*/X2*/Q*R + /X1*/X2*/Q*R + X1*/X2*Q*/R +
       /X1*/X2*Q*R  +  X1*X2*/Q*/R + X1*X2*/Q*R

  Q = /X1*/X2*Q*/R + /X1*/X2*Q*R +  X1*X2*/Q*R +
      /X1*X2*Q*/R  +  X1*/X2*Q*R

DESCRIPTION:
          THIS EXAMPLE ILLUSTRATES HOW TO DESIGN A SEQUENTIAL MACHINE
          WITH PALS. STARTING WITH A STATE GRAPH, CONVERTING IT INTO
          STATE TABLE AND DETERMINING THE NEXT STATES. D TYPE FLIP-
          FLOPS ARE BEING USED AND THE LOGIC EQUATIONS ARE USED TO
          GENERATE THE FUSE PATTERN.
```

Monolithic MMI Memories

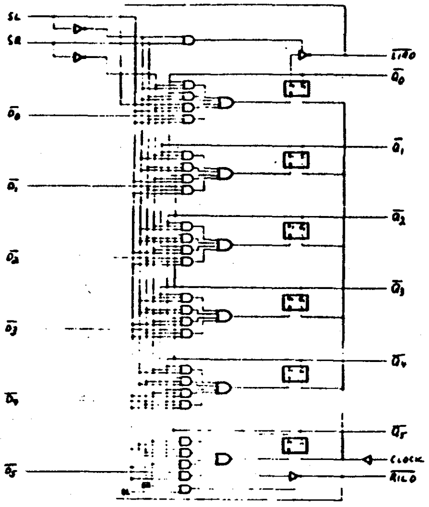
COMBINATIONAL LOCK

LOGIC DIAGRAM PAL16R4



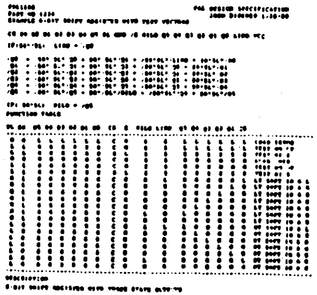
Monolithic MMI Memories

6-BIT SHIFT REGISTER WITH THREE-STATE OUTPUTS



Monolithic Memories

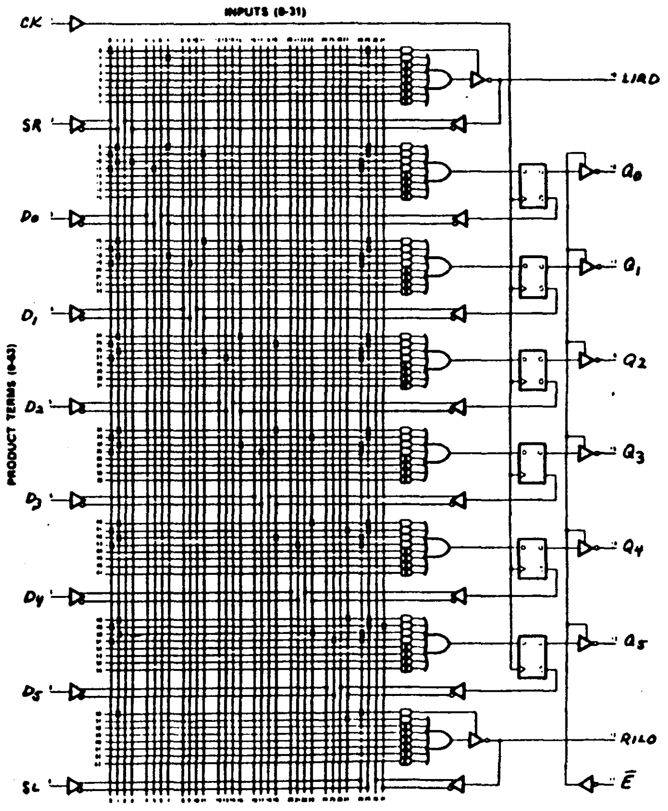
6-BIT SHIFT REGISTER
PAL DESIGN SPECIFICATION


[illegible]

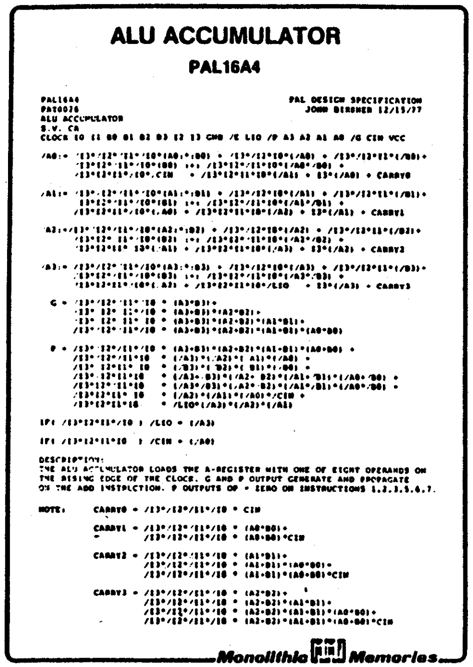
Monolithic Memories

6-BIT SHIFT REGISTER WITH THREE-STATE OUTPUTS

LOGIC DIAGRAM PAL16R6



Monolithic Memories

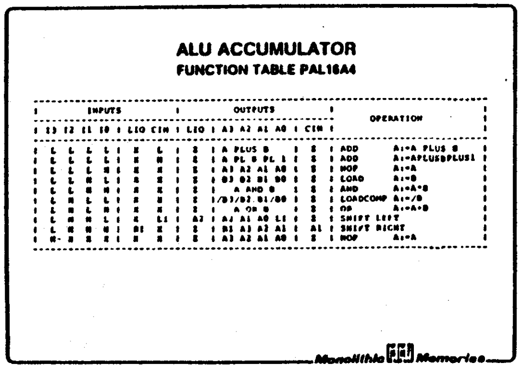


ALU ACCUMULATOR

PAL16A4

```
PAL16A4                                          PAL DESIGN SPECIFICATION
PAT0026                                            JOHN BIRKNER 12/15/77
ALU ACCUMULATOR
S.V. CA
CLOCK I0 I1 B0 B1 B2 B3 I2 I3 GND /E LIO /P A3 A2 A1 A0 /G CIN VCC

/A0:= /I3*/I2*/I1*/I0*(A0:+:B0) + /I3*/I2*I0*(/A0) + /I3*/I2*I1*(/B0)+
      /I3*I2*/I1*/I0*(B0) :+: /I3*I2*/I1*I0*(/A0*/B0) +
      /I3*I2*I1*/I0*/CIN  + /I3*I2*I1*I0*(/A1) + I3*(/A0) + CARRY0

/A1:= /I3*/I2*/I1*/I0*(A1:+:B1) + /I3*/I2*I0*(/A1) + /I3*/I2*I1*(/B1)+
      /I3*I2*/I1*/I0*(B1) :+: /I3*I2*/I1*I0*(/A1*/B1) +
      /I3*I2*I1*/I0*(/A0) + /I3*I2*I1*I0*(/A2) + I3*(/A1) + CARRY1

/A2:=/I3*/I2*/I1*/I0*(A2:+:B2) + /I3*/I2*I0*(/A2) + /I3*/I2*I1*(/B2)+
      /I3*I2*/I1*/I0*(B2) :+: /I3*I2*/I1*I0*(/A2*/B2) +
      /I3*I2*I1*/I0*(/A1) + /I3*I2*I1*I0*(/A3) + I3*(/A2) + CARRY2

/A3:= /I3*/I2*/I1*/I0*(A3:+:B3) + /I3*/I2*I0*(/A3) + /I3*/I2*I1*(/B3)+
      /I3*I2*/I1*/I0*(B3) :+: /I3*I2*/I1*I0*(/A3*/B3) +
      /I3*I2*I1*/I0*(/A2) + /I3*I2*I1*I0*/LIO   + I3*(/A3) + CARRY3

  G = /I3*/I2*/I1*/I0 * (A3*B3)+
      /I3*/I2*/I1*/I0 * (A3+B3)*(A2*B2)+
      /I3*/I2*/I1*/I0 * (A3+B3)*(A2+B2)*(A1*B1)+
      /I3*/I2*/I1*/I0 * (A3+B3)*(A2+B2)*(A1+B1)*(A0*B0)

  P = /I3*/I2*/I1*/I0 * (A3+B3)*(A2+B2)*(A1+B1)*(A0+B0) +
      /I3*/I2*/I1*I0  * (/A3)*(/A2)*(/A1)*(/A0) +
      /I3*/I2*I1*/I0  * (/B3)*(/B2)*(/B1)*(/B0) +
      /I3*/I2*I1*I0   * (/A3+/B3)*(/A2+/B2)*(/A1+/B1)*(/A0+/B0) +
      /I3*I2*/I1*I0   * (/A3*/B3)*(/A2*/B2)*(/A1*/B1)*(/A0*/B0) +
      /I3*I2*I1*/I0   * (/A2)*(/A1)*(/A0)*/CIN +
      /I3*I2*I1*I0    * /LIO*(/A3)*(/A2)*(/A1)

IF( /I3*I2*I1*/I0 ) /LIO = (/A3)

IF( /I3*I2*I1*I0  ) /CIN = (/A0)

DESCRIPTION:
THE ALU ACCUMULATOR LOADS THE A-REGISTER WITH ONE OF EIGHT OPERANDS ON
THE RISING EDGE OF THE CLOCK. G AND P OUTPUT GENERATE AND PROPAGATE
ON THE ADD INSTRUCTION. P OUTPUTS OP = ZERO ON INSTRUCTIONS 1,2,3,5,6,7.

NOTE:     CARRY0 = /I3*/I2*/I1*/I0 * CIN

          CARRY1 = /I3*/I2*/I1*/I0 * (A0*B0)+
          =        /I3*/I2*/I1*/I0 * (A0+B0)*CIN

          CARRY2 = /I3*/I2*/I1*/I0 * (A1*B1)+
                   /I3*/I2*/I1*/I0 * (A1+B1)*(A0*B0)+
                   /I3*/I2*/I1*/I0 * (A1+B1)*(A0+B0)*CIN

          CARRY3 = /I3*/I2*/I1*/I0 * (A2*B2)+
                   /I3*/I2*/I1*/I0 * (A2+B2)*(A1*B1)+
                   /I3*/I2*/I1*/I0 * (A2+B2)*(A1+B1)*(A0*B0)+
                   /I3*/I2*/I1*/I0 * (A2+B2)*(A1+B1)*(A0+B0)*CIN
```

Monolithic Memories

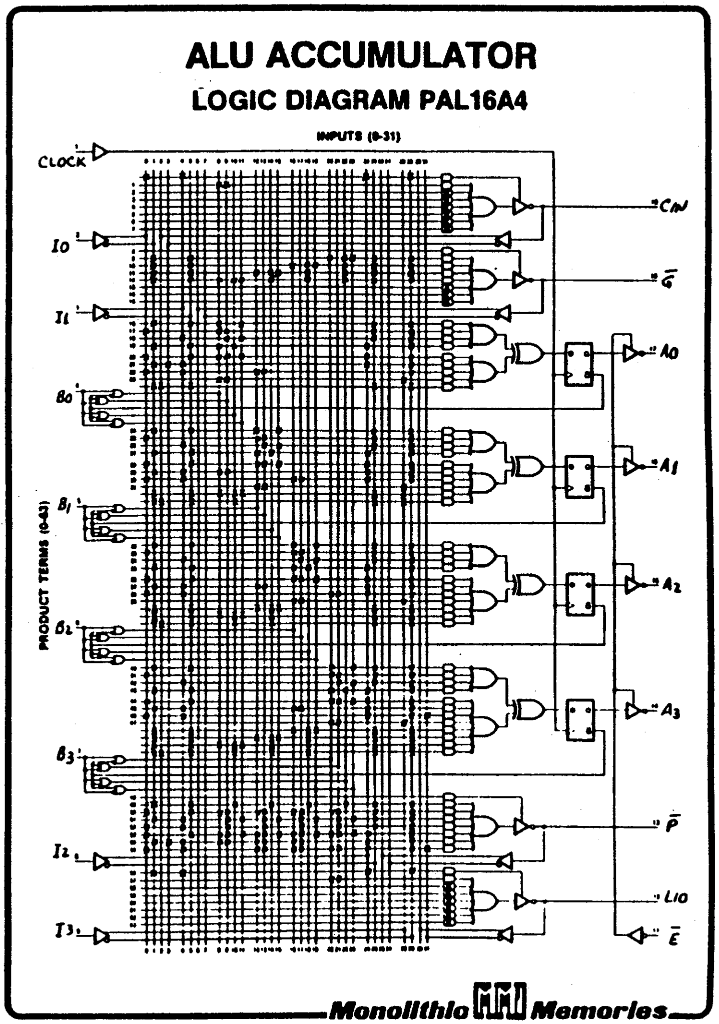


ALU ACCUMULATOR

FUNCTION TABLE PAL16A4

INPUTS						OUTPUTS			OPERATION	
I3	I2	I1	I0	LIO	CIN	LIO	A3 A2 A1 A0	CIN		
L	L	L	L	X	L	Z	A PLUS B	Z	ADD	A:=A PLUS B
L	L	L	L	X	H	Z	A PL B PL 1	Z	ADD	A:=APLUSBPLUS1
L	L	L	H	X	X	Z	A3 A2 A1 A0	Z	NOP	A:=A
L	L	H	L	X	X	Z	B3 B2 B1 B0	Z	LOAD	A:=B
L	L	H	H	X	X	Z	A AND B	Z	AND	A:=A*B
L	H	L	L	X	X	Z	/B3/B2 B1/B0	Z	LOADCOMP	A:=/B
L	H	L	H	X	X	Z	A OR B	Z	OR	A:=A+B
L	H	H	L	X	L1	A3	A2 A1 A0 L1	Z	SHIFT LEFT	
L	H	H	H	R1	X	Z	R1 A3 A2 A1	A1	SHIFT RIGHT	
H-	X	X	X	X	X	Z	A3 A2 A1 A0	Z	NOP	A:=A

Monolithic Memories

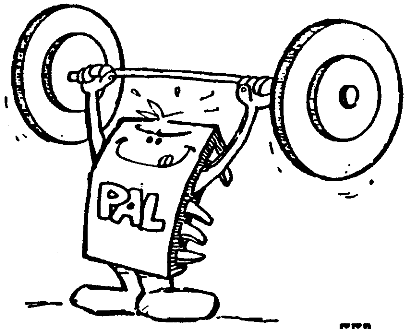


ALU ACCUMULATOR

LOGIC DIAGRAM PAL16A4

Monolithic Memories

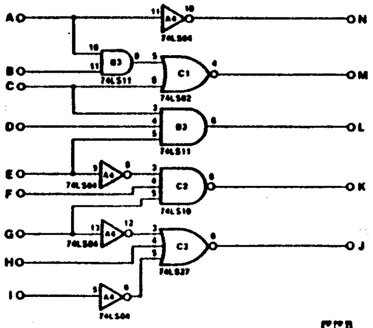
PAL EXERCISE



Monolithic Memories

EXERCISE 1:

IMPLEMENT AN ACTIVE LOW PAL

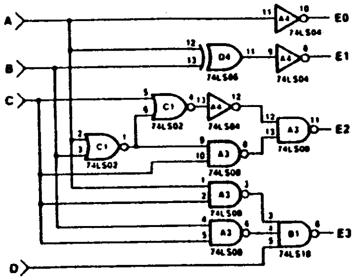


Monolithic Memories

EXERCISE 2:

IMPLEMENT AN ACTIVE HIGH PAL

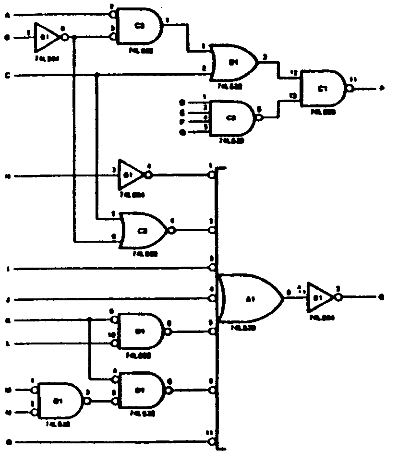
MULTI-LEVEL LOGIC REDUCTION



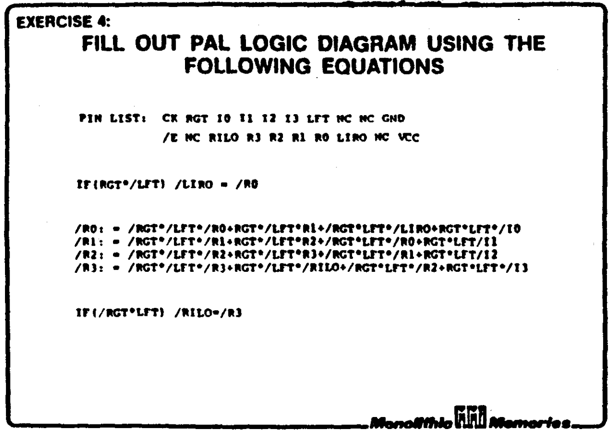
Monolithic Memories

EXERCISE 3:

IMPLEMENT THE FOLLOWING LOGIC USING
(a) ACTIVE HIGH PAL
(b) ACTIVE LOW PAL



Monolithic Memories

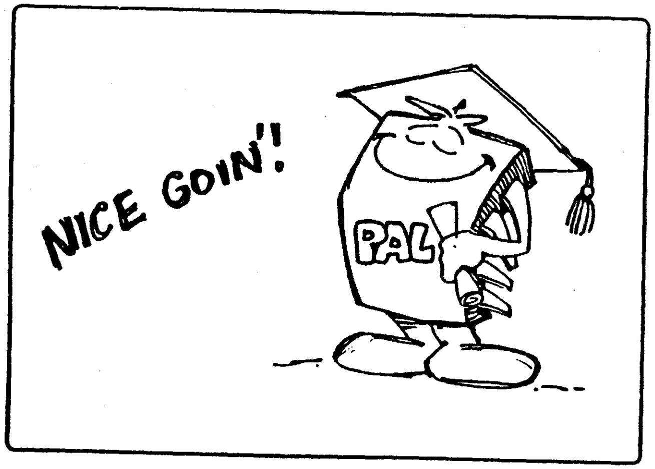


EXERCISE 4:

FILL OUT PAL LOGIC DIAGRAM USING THE FOLLOWING EQUATIONS

```
PIN LIST:  CK RGT I0 I1 I2 I3 LFT NC NC GND
           /E NC RILO R3 R2 R1 R0 LIRO NC VCC


IF(RGT*/LFT) /LIRO = /R0


/R0: = /RGT*/LFT*/R0+RGT*/LFT*R1+/RGT*LFT*/LIRO+RGT*LFT*/I0
/R1: = /RGT*/LFT*/R1+RGT*/LFT*R2+/RGT*LFT*/R0+RGT*LFT/I1
/R2: = /RGT*/LFT*/R2+RGT*/LFT*R3+/RGT*LFT*/R1+RGT*LFT/I2
/R3: = /RGT*/LFT*/R3+RGT*/LFT*/RILO+/RGT*LFT*/R2+RGT*LFT*/I3


IF(/RGT*LFT) /RILO=/R3
```

Monolithic Memories



Monolithic Memories

1. When submitting a HAL design to MMI, logic equations which conform to MMI's PAL Design Specification are required. These logic equations may be provided through the following means:

1.1 Direct input to the "MMI User" account on the VAX at MMI.

1.2Senda legible, signed and dated print-out of the equations of a PALASM run to the inside sales representative or customer sales representative.

1.3 Floppy disk or magnetic tape in formats acceptable by MMI's computer system.

2. Except for small PALs, a Function Table must be provided. This requirement can be waived for programmed PAL samples only.

3. This Function Table should describe the operation of the device and the number of vectors should be sufficient to guarantee the functionality of the device. Normally, the Function Table vectors should test a minimum of 50% of the "stuck at" faults as measured by the fault grading of TEGAS.

4. The Function Table vectors should initialize the device to a known state within a specified number of clocks.

5. Should the simulation by PALASM differ from the simulation by TEGAS, the TEGAS simulation will be used.

6. Any problems that arise shall be handled through the Product Marketing Engineer.

7. MMI will send the LOGIC DESIGN/SAMPLE APPROVAL form to the customer for signoff and confirmation. The customer will, at this time, have the option of using TGEN, MMI's test generation software, to generate additional test vectors. If the customer elects this option, his/her approval of this TGEN generated test pattern is solicited using the CUSTOMER TEST PATTERN APPROVAL form. The signoff loop is now complete.

LOGIC DESIGN/SAMPLE APPROVAL

Dear Customer:

Your PAL Design Specification has been accepted as Pattern No. ______________________ on (Date)_______________.

Please confirm these logic equations (attached) and refer to the pattern number when placing your order. Please be advised that any samples delivered were programmed but not functionally tested.

Confirmation:

I have reviewed the attached PAL/HAL Design Specification and understand that devices ordered with this pattern number will be programmed and functionally tested according to the information submitted. I also understand that the functional test sequence submitted to MMI will first be reviewed by MMI and may require further clarification and refinement (with respect to initialization and general testability) prior to actual implementation by MMI.

Customer______________________________
Company______________________________
Address______________________________
____

Authorized Signature______________________ Date ______________
Name (Please Print)______________________ Title______________

```
Initial  !------!
         !      !
         !------!
```

I am exercising the option of using the MMI expansion of my submitted functional test sequence as the basis for the test specification. I understand that this expansion will be done using TGEN, MMI's test generation software and that an additional formal approval, via the Customer Test Pattern Approval, is necessary once this expanded functional test sequence has been prepared. Until this final approval is accomplished, it is understood that all production delivery commitments are conditional upon a timely approval.

Customer Test Pattern Approval Date:_______________

The attached functional test sequence has been prepared for Pattern No.____________________ by MMI from your design input utilizing TGEN, the MMI test generation software.

This functional test sequence, combined with MMI's data sheet parameters of the ________________________ is the basis of the electrical test specification for the pattern when delivered as a ProPAL or a HAL.

Acknowledged by:

Name ____
Company ____
Title ____
Date ____

Q1. PALASM DOESN'T WORK!!!!

A1. Please don't call us with just this message but before you call us, try running one or two design examples you received in your package. Pick an example that uses the same part type. If this doesn't work, when you call, tell us what specifically was output from your computer. If it is a problem in interfacing with your programmer, verify that you have established communications with the programmer. More often than not, the problem lies in the RS232 connection.

Q2. What is the relationship between the polarity of the pin list, the output pin equations, and the polarity of the function table pin list?

A2. The function table list vector components are evaluated using the polarity specified in the function table pin list. When PALASM actually gets to simulation, it evaluates the output values in terms of the following table:

Pin List Polarity	Output Pin Polarity	Interpreted Output Polarity
H	H	H
H	L	L
L	H	L
L	L	H

By the same reasoning:

Function Pin	Vector Component	Internal Evaluation
List Polarity	Polarity	
H	H	H
H	L	L
L	H	L
L	L	H

Q3. How does PALASM deal with "X" for simulation purposes?

A3. An "X" entered as an input is treated as a LOW signal. Similarly, an "X" on an output is LOW, even as a feedback.

Q4. How do you specify always enabling a tri-state output pin?

A4. In the specification file, use the form:

```
IF (VCC) OUTPUT=WHATEVER.EXPRESSION
```

Q5. How do you disable the output of a pin with feedback for use as an input?

A5. In the specification file, do not write an output equation using that pin, and it will default to putting the output in high impedance, so the feedback part of the circuit will input directly from the pin.

Q6. How do you determine the next state for a registered PAL?

A6. The PALASM simulator initializes the state of the simulated component with the first vector of the function table. It remembers the states of the various output pins till the simulation of the next function table vector is calculated, at which point, it updates the pin states.

Q7. Can 100% fault testing be achieved for all PAL designs?

A7. Sorry, no. Not all logic circuits can be fully tested. PERIOD. However, PALs with the "P" features allow 100% fault testing because they can be pre-loaded. Earlier generation PALs do not have this pre-load feature therefore they may not be fully testable. There are techniques for obtaining greater fault coverage but none guarantee 100% coverage.

Q8. How does somebody know when they have built the best possible set of test vectors?

A8. No absolute method exists for determining that you have the best possible set of test vectors. You may be able to achieve 100% fault coverage for simple designs, but this is not always possible. If you are using redundant product term techniques, you will not be able to fully test each product term.

Should you encounter any problems with PALs or PALASM, write or call:

The IdeaLogic Exchange
Mail-Stop (08-26)
Monolithic Memories Inc.
4250 Burton Drive
Santa Clara, CA 95050

Tel: (408)970-9700 extension 8130

SOFTWARE

1. Birkner, John B., "Macros for Programmable Logic," Wescon Professional Conference Session Record, 1982.

2. Birkner, John B., "High Level Language for Programmable Array Logic,"

3. Birkner, John B., Coli, Vincent J., and Sackett, David M., "Design Your Own Chip With PALASM, Your Personal Computer, And A PROM Programmer," Machine Design, July 1983, p81-85.

4. Birkner, John B., "CAD Methodology Parallels Advances in Programmable Logic,".

ALS

1. Birkner, John B. and Coli, Vincent J., "Hard Array Logic Provides New TTL Standards,".

2. Birkner, John B., "CAD Methodology Parallels Advances in Programmable Logic," Electronica 82.

3. Miller, Warren, "The Philosophies of Fuse Programmable Logic," Wescon Conference Professional Session Record, 1982.

4. Miller, Warren, "New Developments in Programmable Logic," Wescon Conference Professional Session Record, 1982.

5. "Programmable Logic: A Basic Guide for the Designer," Data I/O Corporation

AL APPLICATIONS

1. Coli, Vincent J., "PAL Bumps Eight Chips from Microprocessor Interface," Electronic Design, November 25, 1982, p180-182.

2. Blasco, Richard W., "PALs Shrink Audio Spectrum Analyzer," Electronic Design, August 20, 1980.

3. Coli, Vincent J., "Using a PAL to Emulate the Internal State Counter of the MMI 'S516 LSI Multiplier/Divider," The Best of the Computer Faires, Volume VIII, 1983.

PALASM FOR 20RA10 PAL

Der_Haur Hsieh
Imtiyaz Bengali
Monolithic Memories

This program is written in Pascal for IBM PC/XT but it uses standard Pascal features to be portable on different computer systems. It enables the user to translate his set of equations into fusepattern for 20RA10 PAL. The equations have to be written in a specific format which will be described later in the section.

The program asks the user for the file in which he has his design specification. If he does not specify any file name, then the program takes the default file 'specs.dat'. The file name can be specified as follows :

```
[drive:] filename [.ext]
```

The [drive:] and [.ext] are optional. If no drive is mentioned than the default drive is assumed. The filename together with drive cannot be more than 8 characters.

```
default file :  specs.dat
```

The program generates XPLOT and JEDEC format fusepattern and stores them in two files:

```
Xplot is stored in  [drive:] filename .xpt
Jedec is stored in  [drive:] filename .jed
```

The filename and [drive:] are the same as specified by the user for his design specification file. If the default file is assumed then these two files will also be defaulted to

```
default files  specs.xpt and specs.jed.
```

For clarifiction, the program informs the user where the xplot and jedec format are stored.

The program detects most of errors and reports them to the user in a useful way. The errors reported are :

1) If there is any signal used in the equations and not specified in the pinlist declaration.

   ```
   <signal name>
    message explaining the error.
   ```

2) If there are excess number of product terms used for a particular function. For e.g., the data function can have maximum of four product terms. If there are more than four product terms, then an error will be reported as follows:

    <equation will be written>
    message expalining the error.

There are five of these types of errors for TRST, SET RESET, CLK and DATA functions.

3) If there is a contradiction in an application. For e.g. a particular output is used as a combinatorial output and the user specifies either a SET, RESET or a CLK function, then this is an error and will be reported as follows:

    <Output name >
    message explaining the error.

4) Pin numbers 1 and 13 are special pins and cannot used in the equations. If they are found in any equations then they will be reported as a an error as follows:

    <signal name>
    message expalining the error.

Total of nine type of errors are detected and reported to the user. The messages are self expanatory and will be illustrated through examples.

Design Specification Format

```
CHIP  <example_name> <pal type>    ; example_name and pal type are
                                   ; optional

<pin1>,<pin2>,...,<pin 11>,GND,    ; pin1 are signal names and can
<pin 13>,<pin 14>,..,<pin 23>,VCC  ; be true or complemented  e.g.
                                   ; (A or /A). GND, VCC are used
                                   ; for pin 12 and pin 24 resp.
                                   ; Any pin that is not used is
                                   ; given NC. The signal names
                                   ; cannot be more than 12 chars.

EQU                                ; Key word for start of
                                   ; equations.


output(i) := input (j) * /input(k) * output (l)  ; sum of product
           + input (m) * /~~output (n)~~                ; terms in terms
           + input (a)                           ; of inputs and
                                                 ; outputs . The
                                                 ; output(i) can
                                                 ; be true or
                                                 ; inverted

output(i).trst = <product term>     ; for tri-state control.
                                    ; defaults to high
                                    ;(always enabled)

output(i).set = <product term>      ; for set control.
                                    ; defaults to low

output(i).reset = <produc term>     ; for reset control.
                                    ; defaults to low

output(i).clk = <product term>      ; for clock control.
                                    ; defaults to low


; The following equation is for combinatorial output.
; The RESET and SET product terms are all HIGH i.e.
; all the fuses are blown.

output (b)  = <product term 1>      ; combinatorial output
            + <product term 2>
            + <product term 3>
            + <product term 4>

output(b).trst = <product term>
```

CHIP

This the key word for start of the design specification.

There are two optional fields :

i) Design name and
ii) PAL type.

Pin List

There are twenty four pins to this device and hence 24 signals can be named. All the signals should be ordered in the order of the pins. Pin 12 and Pin 24 have key words GND and VCC respectively. Each signal name should not be more than 12 characters. Each signal name can be true or complemented. The signal names are separated by commas. Unused pins are given NC as the pin name.

```
e.g. ABCD, Q1 , NC,NC,NC, /CD,RESET,/INT,D1,D2,D3,GND,
     /OE, Q2, Q3, CAS, /RAS, MEMWR, /MEMRD, NC ,NC, CNT, VCC
```

Equations

EQU indicates the start of equations. It is a key word which has to be specified before the equations are written.

```
  output := f (inputs,outputs)
```

Each output can be specified as sum of product terms. Each product term can be made up of input and output signals in true or complemented form. There has to be maximum of four product terms.

Associated with each output are four control product terms viz:

```
TRST  for tri state control
CLK   for clock control
RESET for reset of the register
SET   for set of the register.
```

Each of this control functions can be specified as a produc terms in the following manner:

```
output.TRST  = product term
output.SET   = product term
output.RESET = product term
output.CLK   = product term
```

If any of these conrtrol functions are not specified than they are defaulted to the following states:

```
TRST is defaulted to HIGH   RESET is defaulted to LOW
SET  is defaulted to LOW    CLK is defaulted to LOW
```

If the output is a combinatorial then it hasto be specified as

```
      output = <product terml>
             + <product term 2>
             + <product term 3>
             + <product term 4>
```

In case of combinatorial output, no SET , RESET or CLK product terms have to be specified. If they are specified then the program will report an error. The SET and RESET product terms are defaulted to HIGH and CLK is defaulted to LOW. One can specify TRST control.

The output can be specified as a function of GND or VCC as follows:

```
      output := VCC         ; the input to this register is hooked
or                          ; to VCC
      output := GND         ;the input to this register is hooked
                            ; to GND.
```

```
C>RAPAL

C>PARSER
 Enter input file[specs.dat] : mmi.exm

C>20RA10
The xplot is saved in file ==> mmi.xpt

The jedec is saved in file ==> mmi.jed

C>
C>
```

```
C>RAPAL

C>PARSER
 Enter input file[specs.dat] : mmi

C>20RA10
The xplot is saved in file ==> mmi.xpt

The jedec is saved in file ==> mmi.jed

C>
C>
```

```
C>RAPAL

C>PARSER
 Enter input file[specs.dat] : a:20ra10.pal
```

```
C>20RA10
The xplot is saved in file ==> a:20ral0.xpt

The jedec is saved in file ==> a:20ral0.jed

C>
C>
```

```
rename mmi specs.dat
```

```
C>RAPAL

C>PARSER
 Enter input file[specs.dat] :

C>20RA10
The xplot is saved in file ==> specs.xpt

The jedec is saved in file ==> specs.jed

C>
```

```
CHIP
/ALE, A20, A19, A18, INP0, INPHASE0A, NC, NC, NC, NC, NC, GND,
/OE, START,PHASE0, PHASE1, P0,P1,P2,/DTCK,/RAS,/CAS,PHASE0A,VCC

EQU

START := VCC
START.CLK =  A20 *A19 * A18
START.RESET = DTCK

PHASE0:=VCC
PHASE0.CLK = START * /P0
PHASE0.RESET=INPHASE0A

PHASE0A = PHASE0 + INPHASE0A       ; combinatorial output

P0 := PHASE1 + INP0

P1 := /P1
P1.CLK = /P0
P1.RESET = /START

P2 := /P2
P2.CLK = /P1
P2.RESET = /START

DTCK := VCC                        ; different polarity
/DTCK.CLK = P2 * P1 * P0
/DTCK.RESET = /START

/RAS := VCC
/RAS.CLK = /P2 * P1 * P0
/RAS.RESET= P2 * P1 * P0
/RAS.SET = DTCK

/CAS := VCC
/CAS.CLK = /P2 * P1 * P0
/CAS.RESET = P2 * P1 * /P0
/CAS.SET = DTCK
```

```
a20
             11 1111 1111 2222 2222 2233 3333 3333
  0123 4567 8901 2345 6789 0123 4567 8901 2345 6789

 0 ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
 1 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 2 ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
 3 ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
 4 ---- ---- ---- ---- ---- ---- ---- ---- --x- ----
 5 ---- ---- ---- ---- x--- ---- ---- ---- ---- ----
 6 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
 7 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

 8 ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
 9 ---- ---- ---- ---- ---x --x- --x- ---- ---- ----
10 ---- ---- ---- ---- --x- --x- ---x ---- ---- ----
11 ---- ---- ---- ---x ---- ---- ---- ---- ---- ----
12 ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
13 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
14 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
15 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

16 ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
17 ---- ---- ---- ---- ---x --x- --x- ---- ---- ----
18 ---- ---- ---- ---- --x- --x- --x- ---- ---- ----
19 ---- ---- ---- ---x ---- ---- ---- ---- ---- ----
20 ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
21 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
22 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
23 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

24 ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
25 ---- ---- ---- ---- --x- --x- --x- ---- ---- ----
26 ---- ---- ---- ---- ---- ---- ---- ---- ---- ---x
27 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
28 ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
29 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
30 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
31 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

32 ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
33 ---- ---- ---- ---- ---- ---x ---- ---- ---- ----
34 ---- ---- ---- ---- ---- ---- ---- ---- ---- ---x
35 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
36 ---- ---- ---- ---- ---x ---- ---- ---- ---- ----
37 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
38 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
39 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
```

```
40 ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
41 ---- ---- ---- ---- ---- ---- ---x ---- ---- ----
42 ---- ---- ---- ---- ---- ---- ---- ---- ---- ---x
43 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
44 ---- ---- ---- ---- ---- ---x ---- ---- ---- ----
45 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
46 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
47 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

48 ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
49 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
50 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
51 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
52 ---- ---- ---- ---- ---- ---- ---- --x- ---- ----
53 ---- ---- ---- x--- ---- ---- ---- ---- ---- ----
54 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
55 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

56 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
57 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
58 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
59 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
60 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
61 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
62 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
63 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

64 ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
65 ---- ---- ---- ---- ---- ---- ---x ---- ---- --x-
66 ---- ---- ---- ---- x--- ---- ---- ---- ---- ----
67 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
68 ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
69 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
70 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
71 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

72 ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
73 x--- x--- x--- ---- ---- ---- ---- ---- ---- ----
74 ---- ---- ---- ---x ---- ---- ---- ---- ---- ----
75 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
76 ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
77 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
78 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
79 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Polarity Fuses

Output pin: 1111112222
            4567890123

            --x---x---

Legend:  fuse blown: -   fuse intact: x

Fuses Blown : 1493
```

```
   *
QP24*Q3210*
L00 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
    1111 1111 1111 1111 1111 1111 1111 1111 1111 1111
    1111 1111 1111 1111 1111 1111 1111 1111 1111 1111
    1111 1111 1111 1111 1111 1111 1111 1111 1101 1111
    1111 1111 1111 1111 0111 1111 1111 1111 1111 1111
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000*
L08 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111
    1111 1111 1111 1111 1110 1101 1101 1111 1111 1111
    1111 1111 1111 1111 1101 1101 1110 1111 1111 1111
    1111 1111 1111 1110 1111 1111 1111 1111 1111 1111
    1111 1111 1111 1111 1111 1111 1111 1111 1111 1111
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000*
L16 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111
    1111 1111 1111 1111 1110 1101 1101 1111 1111 1111
    1111 1111 1111 1111 1101 1101 1101 1111 1111 1111
    1111 1111 1111 1110 1111 1111 1111 1111 1111 1111
    1111 1111 1111 1111 1111 1111 1111 1111 1111 1111
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000*
L24 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111
    1111 1111 1111 1111 1101 1101 1101 1111 1111 1111
    1111 1111 1111 1111 1111 1111 1111 1111 1111 1110
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
    1111 1111 1111 1111 1111 1111 1111 1111 1111 1111
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000*
L32 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111
    1111 1111 1111 1111 1111 1110 1111 1111 1111 1111
    1111 1111 1111 1111 1111 1111 1111 1111 1111 1110
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
    1111 1111 1111 1111 1110 1111 1111 1111 1111 1111
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000*
L40 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111
    1111 1111 1111 1111 1111 1111 1110 1111 1111 1111
    1111 1111 1111 1111 1111 1111 1111 1111 1111 1110
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
    1111 1111 1111 1111 1111 1110 1111 1111 1111 1111
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000*
L48 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
    1111 1111 1111 1111 1111 1111 1111 1101 1111 1111
    1111 1111 1111 0111 1111 1111 1111 1111 1111 1111
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000*
L56 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000*
L64 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111
    1111 1111 1111 1111 1111 1111 1110 1111 1111 1101
    1111 1111 1111 1111 0111 1111 1111 1111 1111 1111
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
    1111 1111 1111 1111 1111 1111 1111 1111 1111 1111
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000*
L72 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111
    0111 0111 0111 1111 1111 1111 1111 1111 1111 1111
    1111 1111 1111 1110 1111 1111 1111 1111 1111 1111
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
    1111 1111 1111 1111 1111 1111 1111 1111 1111 1111
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000*

L80 1110111011*402E
```

```
CHIP
/ALE, A20, A19, A18, INP0, INPHASE0A, NC, NC, NC, NC, NC, GND,
/OE, START,PHASE0, PHASE1, P0,P1,P2,/DTCK,/RAS,/CAS,PHASE0A,VCC

EQU

START := VCC
START.CLK =  A220 *A19 * A18       ; A220 is undeclared
START.RESET = DTCK
```

```
C>RAPAL

C>PARSER
 Enter input file[specs.dat] :

C>20RA10

****** ERROR !!! *******


a220                  UNDEFINED SIGNAL NAME


 C>
 C>
```

```
CHIP
/ALE, A20, A19, A18, INP0, INPHASE0A, NC, NC, NC, NC, NC, GND,
/OE, START,PHASE0, PHASE1, P0,P1,P2,/DTCK,/RAS,/CAS,PHASE0A,VCC

EQU

START := VCC
START.CLK =  A20 *A19 * A18
START.RESET = DTCK + A20 * A19
```

```
C>RAPAL

C>PARSER
 Enter input file[specs.dat] : specs1.dat

C>20RA10

****** ERROR !!! *******


start.reset  =dtck
                + a20 * a19


RESET HAS MORE THAN ONE PRODUCT TERM


C>
C>
```

```
CHIP
/ALE, A20, A19, A18, INP0, INPHASE0A, NC, NC, NC, NC, NC, GND,
/OE, START,PHASE0, PHASE1, P0,P1,P2,/DTCK,/RAS,/CAS,PHASE0A,VCC

EQU

START := VCC
START.CLK =  A20 *A19 * A18
START.RESET = DTCK * ALE          ; pin 1 used in a product term
```

```
C>RAPAL

C>PARSER
 Enter input file[specs.dat] : specs2.dat

C>20RA10

****** ERROR !!! *******



ale

THIS SIGNAL IS PIN 1 AND CANNOT BE USED AS
A SIGNAL IN AN EQUATION


C>
C>
```

```
CHIP
/ALE, A20, A19, A18, INP0, INPHASE0A, NC, NC, NC, NC, NC, GND,
/OE, START,PHASE0, PHASE1, P0,P1,P2,/DTCK,/RAS,/CAS,PHASE0A,VCC

EQU


PHASE0A = PHASE0 + INPHASE0A        ; combinatorial output
PHASE0A.RESET = /START              ; having a reset ??
```

```
C>RAPAL

C>PARSER
 Enter input file[specs.dat] : specs3.dat

C>20RA10

****** ERROR !!! *******



phase0a

THIS OUTPUT IS COMBINATORIAL AND CANNOT
HAVE SET/RESET/CLOCK PRODUCT TERM



C>
C>
```

```
CHIP
/ALE, A20, A19, A18, INP0, INPHASE0A, NC, NC, NC, NC, NC, GND,
/OE, START,PHASE0, PHASE1, P0,P1,P2,/DTCK,/RAS,/CAS,PHASE0A,VCC

EQU


P0 := PHASE1 + INP0                      ; more than four product terms
     + /P1 * P2
     +  P1 * /P2
     + /START
```

```
C>RAPAL

C>PARSER
 Enter input file[specs.dat] : specs4.dat

C>20RA10

****** ERROR !!! *******


p0 :=phase1
      + inp0
      +/p1 * p2
      +p1 * /p2
      +/start


OUTPUT HAS MORE THAN 4 PRODUCT TERMS


C>
C>
```

```
CHIP  EXAMPLE, 20RP10,
CL, A3, A2, A1, A0, D0, D1, I0, I1, DATA, P ,GND,
/OE, Q0, Q1, Q2, Q3, /Q4, Q5, Q6, Q7, Q8, Q9, VCC

EQU

Q0 := A3 * Q0 * I0 * CL
    + A2 * Q0 * I0 * I1
    + A1 * Q0 * I0 * I1
    + A0 * Q0 * I0 * i1
    +/A3 * /A2 * /A1 * /A0 * DATA */I0 * I1 * /P
    +/A3 * /A2 * /A1 * /A0 * /DATA * /I0 * I1 * P
    + /I0 * /I1 * P
    + /A3 * D0 * I0 * /I1
```

PALASM FOR 20RP10 PAL

Der_Haur Hsieh
Imtiyaz Bengali
Monolithic Memories

This program is written in Pascal for IBM PC/XT but it uses standard Pascal features to be portable on different computer systems. It enables the user to translate his set of equations into fusepattern for 20RP10 PAL. The equations have to be written in a specific format which will be described later in the section.

To start the program type : RPPAL <CR>

The program asks the user for the file in which he has his design specification. If he does not specify any file name, then the program takes the default file 'specs.dat'. The file name can be specified as follows :

```
[drive:] filename [.ext]
```

The [drive:] and [.ext] are optional. If no drive is mentioned than the default drive is assumed. The filename together with drive cannot be more than 8 characters.

```
default file :  specs.dat
```

The program generates XPLOT and JEDEC format fusepattern and stores them in two files:

```
Xplot is stored in  [drive:] filename .xpt
Jedec is stored in  [drive:] filename .jed
```

The filename and [drive:] are the same as specified by the user for his design specification file. If the default file is assumed then these two files will also be defaulted to

```
default files  specs.xpt and specs.jed.
```

For clarifiction, the program informs the user where the xplot and jedec format are stored.

The program detects most of errors and reports them to the user in a useful way. The errors reported are :

1) If there is any signal used in the equations and not specified in the pinlist declaration.

```
<signal name>
 message explaining the error.
```

2) If there are excess number of product terms used for particular outputs. For e.g. pin 14 and pin 23 can

have maximum of eight product terms and output pairs 15-16, 17-18, 19-20, 21-22 can share 16 product terms exclusively. If any of these conditions are violated then an error is reported as follows:

```
<output (s) will be written>
message expalining the error.
```

3) Pin numbers 1 and 13 are special pins and cannot used in the equations. If they are found in any equations then they will be reported as a an error as follows:

```
<signal name>
message expalining the error.
```

Total of nine type of errors are detected and reported to the user. The messages are self expanatory and will be illustrated through examples.

Design Specification Format

```
CHIP  <example_name> <pal type>   ; example_name and pal type are
                                  ; optional

<pin1>,<pin2>,...,<pin 11>,GND,   ; pin1 are signal names and can
<pin 13>,<pin 14>,..,<pin 23>,VCC ; be true or complemented  e.g.
                                  ; (A or /A). GND, VCC are used
                                  ; for pin 12 and pin 24 resp.
                                  ; Any pin that is not used is
                                  ; given NC. The signal names
                                  ; cannot be more than 12 chars.

EQU                               ; Key word for start of
                                  ; equations.


output(i) := input (j) * /input(k) * output (1)  ; sum of product
           + input (m) * /output (n)             ; terms in terms
           + input (a)                           ; of inputs and
                                                 ; outputs . The
                                                 ; output(i) can
                                                 ; be true or
                                                 ; inverted
```

CHIP

This the key word for start of the design specification.

There are three optional fields :
  i) Design name and
 ii) PAL type.
iii) Pin list

Pin List

There are twenty four pins to this device and hence 24 signals can be named. All the signals should be ordered in the order of the pins. Pin 12 and Pin 24 have key words GND and VCC respectively. Each signal name should not be more than 12 characters. Each signal name can be true or complemented. The signal names are separated by commas. Unused pins are given NC as the pin name.

```
e.g. ABCD, Q1 , NC,NC,NC, /CD,RESET,/INT,D1,D2,D3,GND,
     /OE, Q2, Q3, CAS, /RAS, MEMWR, /MEMRD, NC ,NC, CNT, VCC
```

Equations

EQU indicates the start of equations. It is a key word which to be specified before the equations are written.

```
  output := f (inputs,outputs)
```

Each output can be specified as sum of product terms. Each product term can be made up of input and output signals in true or complemented form. There has to be maximum of four product terms.

The output can be specified as a function of GND or VCC as follows:

```
      output := VCC        ; the input to this register is hooked
or                         ; to VCC
      output := GND        ;the input to this register is hooked
                           ; to GND.
```

```
CHIP  EXAMPLE 20RP10
CL, A3, A2, A1, A0, D0, D1, I0, I1, DATA, P ,GND,
/OE, Q0, Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8, Q9, VCC

EQU

Q0 := A3 * Q0 * I0
    + A2 * Q0 * I0 * I1
    + A1 * Q0 * I0 * I1
    + A0 * Q0 * I0 * i1
    +/A3 * /A2 * /A1 * /A0 * DATA */I0 * I1 * /P
    +/A3 * /A2 * /A1 * /A0 * /DATA * /I0 * I1 * P
    + /I0 * /I1 * P
    + /A3 * D0 * I0 * /I1

Q1 := A3 * Q1 * I0
    + A2 * Q1 * I0 * I1
    + A1 * Q1 * I0 * I1
    + A0 * Q1 * I0 * I1
    +/A3 * /A2 * /A1 * /A0 * DATA * /I0 * I1 * /P
    +/A3 * /A2 * /A1 * /A0 * /DATA * /I0 * I1 * P
    +/I0 * /I1 * P
    +/A3 * D1 * I0 * /I1

 Q2 := A3 * Q2 * I0
    + A2 * Q2 * I0 * I1
    + A1 * Q2 * I0 * I1
    + A0 * Q2 * I0 * I1
    + /A3 * /A2 * /A1 * /A0 * DATA * /I0 * I1 * /P
    + /A3 * /A2 * /A1 * /A0 * /DATA * /I0 * I1 * P
    + /I0 * /I1 * P
    + /A3  * D0 * I0 * /I1

Q3 := A3 * Q3 * I0
    + A2 * Q3 * I0 *I1
    + A1 * Q3 * I0 *I1
    + A0 * Q3 * I0 *I1
    + /A3 * /A2 * /A1 * /A0 * DATA * /I0 * I1 * /P
    + /A3 * /A2 * /A1 * /A0 * /DATA * /I0 * I1 * P
    + /I0 * /I1 * P
    + /A3 * D1 *  I0 * /I1

Q4 := A3 * Q4 * I0
    + A2 * Q4 * I0 * I1
    + A1 * Q4 * I0 * I1
    + A0 * Q4 * I0 * I1
    +/A3 * /A2 * /A1 * /A0 * DATA * /I0 * I1 * /P
    +/A3 * /A2 * /A1 * /A0 * /DATA * /I0 * I1 * P
    +/I0 * /I1 * P
    +/A3 * D0 * I0 * /I1
```

```
Q5  := A3 * Q5 * I0
    +  A2 * Q5 * I0 * I1
    +  A1 * Q5 * I0 * I1
    +  A0 * Q5 * I0 * I1
    + /A3 * /A2 * /A1 * /A0 * DATA * /I0 * I1 * /P
    + /A3 * /A2 * /A1 * /A0 * /DATA * /I0 * I1 * P
    + /I0 * /I1 * P
    + /A3 * D1 * I0 * /I1

Q6  := A3 * Q6 * I0
    +  A2 * Q6 * I0 * I1
    +  A1 * Q6 * I0 * I1
    +  A0 * Q6 * I0 * I1
    + /A3 * /A2 * /A1 * /A0 * DATA * /I0 * I1 * /P
    + /A3 * /A2 * /A1 * /A0 * /DATA * /I0 * I1 * P
    + /I0 * /I1 * P
    + /A3  * D0 * I0 * /I1

Q7  := A3 * Q7 * I0
    +  A2 * Q7 * I0 *I1
    +  A1 * Q7 * I0 * I1
    +  A0 * Q7 * I0 *I1
    + /A3 * /A2 * /A1 * /A0 * DATA * /I0 * I1 * /P
    + /A3 * /A2 * /A1 * /A0 * /DATA * /I0 * I1 * P
    + /I0 * /I1 * P
    + /A3  * D1 * I0 * /I1

Q8  := A3 * Q8 * I0
    +  A2 * Q8 * I0 *I1
    +  A1 * Q8 * I0 * I1
    +  A0 * Q8 * I0 * I1
    +  /A3 * /A2 * /A1 * /A0 * DATA * /I0 * I1 * /P
    + /A3  * /A2 * /A1 * /A0 * /DATA * /I0 * I1 *P
    + /I0 * /I1 * P
    + /A3 * D0 * I0 * /I1

Q9  := A3 * Q9 * I0
     + A2 * Q9 * I0 * I1
     + A1 * Q9 * I0 * I1
     + A0 * Q9 * I0 * I1
     + /A3 * /A2 * /A1 * /A0 * DATA * /I0 * I1 * /P
     + /A3 * /A2 * /A1 * /A0 * /DATA * /I0 * I1 * P
     + /I0 * /I1 * P
     + /A3 * D1 * I0 *I1
```

```
example                rp10
               11 1111 1111 2222 2222 2233 3333 3333
   0123 4567 8901 2345 6789 0123 4567 8901 2345 6789  01

 0 x-x- ---- ---- ---- ---- ---- x--- ---- ---- ----  x
 1 --x- x--- ---- ---- ---- ---- x--- x--- ---- ----  x
 2 --x- ---- x--- ---- ---- ---- x--- x--- ---- ----  x
 3 --x- ---- ---- x--- ---- ---- x--- x--- ---- ----  x
 4 -x-- -x-- -x-- -x-- ---- ---- -x-- x--- x--- -x--  x
 5 -x-- -x-- -x-- -x-- ---- ---- -x-- x--- -x-- x---  x
 6 ---- ---- ---- ---- ---- ---- -x-- -x-- ---- x---  x
 7 -x-- ---- ---- ---- ---- x--- x--- x--- ---- ----  x

 8 x--- --x- ---- ---- ---- ---- x--- ---- ---- ----  x-
 9 ---- x-x- ---- ---- ---- ---- x--- x--- ---- ----  x-
10 ---- --x- x--- ---- ---- ---- x--- x--- ---- ----  x-
11 ---- --x- ---- x--- ---- ---- x--- x--- ---- ----  x-
12 -x-- -x-- -x-- -x-- ---- ---- -x-- x--- x--- -x--  x-
13 -x-- -x-- -x-- -x-- ---- ---- -x-- x--- -x-- x---  x-
14 ---- ---- ---- ---- ---- ---- -x-- -x-- ---- x---  x-
15 -x-- ---- ---- ---- x--- ---- x--- -x-- ---- ----  x-
16 -x-- ---- ---- ---- ---- x--- x--- -x-- ---- ----  -x
17 ---- ---- ---- ---- ---- ---- -x-- -x-- ---- x---  -x
18 -x-- -x-- -x-- -x-- ---- ---- -x-- x--- -x-- x---  -x
19 -x-- -x-- -x-- -x-- ---- ---- -x-- x--- x--- -x--  -x
20 ---- ---- --x- x--- ---- ---- x--- x--- ---- ----  -x
21 ---- ---- x-x- ---- ---- ---- x--- x--- ---- ----  -x
22 ---- x--- --x- ---- ---- ---- x--- x--- ---- ----  -x
23 x--- ---- --x- ---- ---- ---- x--- ---- ---- ----  -x

24 x--- ---- ---- --x- ---- ---- x--- ---- ---- ----  x-
25 ---- x--- ---- --x- ---- ---- x--- x--- ---- ----  x-
26 ---- ---- x--- --x- ---- ---- x--- x--- ---- ----  x-
27 ---- ---- ---- x-x- ---- ---- x--- x--- ---- ----  x-
28 -x-- -x-- -x-- -x-- ---- ---- -x-- x--- x--- -x--  x-
29 -x-- -x-- -x-- -x-- ---- ---- -x-- x--- -x-- x---  x-
30 ---- ---- ---- ---- ---- ---- -x-- -x-- ---- x---  x-
31 -x-- ---- ---- ---- x--- ---- x--- -x-- ---- ----  x-
32 -x-- ---- ---- ---- ---- x--- x--- -x-- ---- ----  -x
33 ---- ---- ---- ---- ---- ---- -x-- -x-- ---- x---  -x
34 -x-- -x-- -x-- -x-- ---- ---- -x-- x--- -x-- x---  -x
35 -x-- -x-- -x-- -x-- ---- ---- -x-- x--- x--- -x--  -x
36 ---- ---- ---- x--- --x- ---- x--- x--- ---- ----  -x
37 ---- ---- x--- ---- --x- ---- x--- x--- ---- ----  -x
38 ---- x--- ---- ---- --x- ---- x--- x--- ---- ----  -x
39 x--- ---- ---- ---- --x- ---- x--- ---- ---- ----  -x
```

```
40 x--- ---- ---- ---- ---- --x- x--- ---- ---- ----  x-
41 ---- x--- ---- ---- ---- --x- x--- x--- ---- ----  x-
42 ---- ---- x--- ---- ---- --x- x--- x--- ---- ----  x-
43 ---- ---- ---- x--- ---- --x- x--- x--- ---- ----  x-
44 -x-- -x-- -x-- -x-- ---- ---- -x-- x--- x--- -x--  x-
45 -x-- -x-- -x-- -x-- ---- ---- -x-- x--- -x-- x---  x-
46 ---- ---- ---- ---- ---- ---- -x-- -x-- ---- x---  x-
47 -x-- ---- ---- ---- x--- ---- x--- -x-- ---- ----  x-
48 -x-- ---- ---- ---- ---- x--- x--- -x-- ---- ----  -x
49 ---- ---- ---- ---- ---- ---- -x-- -x-- ---- x---  -x
50 -x-- -x-- -x-- -x-- ---- ---- -x-- x--- -x-- x---  -x
51 -x-- -x-- -x-- -x-- ---- ---- -x-- x--- x--- -x--  -x
52 ---- ---- ---- x--- ---- ---- x-x- x--- ---- ----  -x
53 ---- ---- x--- ---- ---- ---- x-x- x--- ---- ----  -x
54 ---- x--- ---- ---- ---- ---- x-x- x--- ---- ----  -x
55 x--- ---- ---- ---- ---- ---- x-x- ---- ---- ----  -x

56 x--- ---- ---- ---- ---- ---- x--- --x- ---- ----  x-
57 ---- x--- ---- ---- ---- ---- x--- x-x- ---- ----  x-
58 ---- ---- x--- ---- ---- ---- x--- x-x- ---- ----  x-
59 ---- ---- ---- x--- ---- ---- x--- x-x- ---- ----  x-
60 -x-- -x-- -x-- -x-- ---- ---- -x-- x--- x--- -x--  x-
61 -x-- -x-- -x-- -x-- ---- ---- -x-- x--- -x-- x---  x-
62 ---- ---- ---- ---- ---- ---- -x-- -x-- ---- x---  x-
63 -x-- ---- ---- ---- x--- ---- x--- -x-- ---- ----  x-
64 -x-- ---- ---- ---- ---- x--- x--- -x-- ---- ----  -x
65 ---- ---- ---- ---- ---- ---- -x-- -x-- ---- x---  -x
66 -x-- -x-- -x-- -x-- ---- ---- -x-- x--- -x-- x---  -x
67 -x-- -x-- -x-- -x-- ---- ---- -x-- x--- x--- -x--  -x
68 ---- ---- ---- x--- ---- ---- x--- x--- --x- ----  -x
69 ---- ---- x--- ---- ---- ---- x--- x--- --x- ----  -x
70 ---- x--- ---- ---- ---- ---- x--- x--- --x- ----  -x
71 x--- ---- ---- ---- ---- ---- x--- ---- --x- ----  -x

72 x--- ---- ---- ---- ---- ---- x--- ---- ---- --x-  x
73 ---- x--- ---- ---- ---- ---- x--- x--- ---- --x-  x
74 ---- ---- x--- ---- ---- ---- x--- x--- ---- --x-  x
75 ---- ---- ---- x--- ---- ---- x--- x--- ---- --x-  x
76 -x-- -x-- -x-- -x-- ---- ---- -x-- x--- x--- -x--  x
77 -x-- -x-- -x-- -x-- ---- ---- -x-- x--- -x-- x---  x
78 ---- ---- ---- ---- ---- ---- -x-- -x-- ---- x---  x
79 -x-- ---- ---- ---- x--- ---- x--- -x-- ---- ----  x

Polarity Fuses

Output pin: 1111112222
            4567890123

            ----------

Legend:  fuse blown: -   fuse intact: x

Fuses Blown : 2884
```

```
1111 1111 1111 0111 1111 1101 0111 0111 1111 1111 01
1011 1011 1011 1011 1111 1111 1011 0111 0111 1011 01
1011 1011 1011 1011 1111 1111 1011 0111 1011 0111 01
1111 1111 1111 1111 1111 1111 1011 1011 1111 0111 01
1011 1111 1111 1111 0111 1111 0111 1011 1111 1111 01
1011 1111 1111 1111 1111 0111 0111 1011 1111 1111 10
1111 1111 1111 1111 1111 1111 1011 1011 1111 0111 10
1011 1011 1011 1011 1111 1111 1011 0111 1011 0111 10
1011 1011 1011 1011 1111 1111 1011 0111 0111 1011 10
1111 1111 1111 0111 1111 1111 0101 0111 1111 1111 10
1111 1111 0111 1111 1111 1111 0101 0111 1111 1111 10
1111 0111 1111 1111 1111 1111 0101 0111 1111 1111 10
0111 1111 1111 1111 1111 1111 0101 1111 1111 1111 10
0111 1111 1111 1111 1111 1111 0111 1101 1111 1111 01
1111 0111 1111 1111 1111 1111 0111 0101 1111 1111 01
1111 1111 0111 1111 1111 1111 0111 0101 1111 1111 01
1111 1111 1111 0111 1111 1111 0111 0101 1111 1111 01
1011 1011 1011 1011 1111 1111 1011 0111 0111 1011 01
1011 1011 1011 1011 1111 1111 1011 0111 1011 0111 01
1111 1111 1111 1111 1111 1111 1011 1011 1111 0111 01
1011 1111 1111 1111 0111 1111 0111 1011 1111 1111 01
1011 1111 1111 1111 1111 0111 0111 1011 1111 1111 10
1111 1111 1111 1111 1111 1111 1011 1011 1111 0111 10
1011 1011 1011 1011 1111 1111 1011 0111 1011 0111 10
1011 1011 1011 1011 1111 1111 1011 0111 0111 1011 10
1111 1111 1111 0111 1111 1111 0111 0111 1101 1111 10
1111 1111 0111 1111 1111 1111 0111 0111 1101 1111 10
1111 0111 1111 1111 1111 1111 0111 0111 1101 1111 10
0111 1111 1111 1111 1111 1111 0111 1111 1101 1111 10
0111 1111 1111 1111 1111 1111 0111 1111 1111 1101 0
1111 0111 1111 1111 1111 1111 0111 0111 1111 1101 0
1111 1111 0111 1111 1111 1111 0111 0111 1111 1101 0
1111 1111 1111 0111 1111 1111 0111 0111 1111 1101 0
1011 1011 1011 1011 1111 1111 1011 0111 0111 1011 0
1011 1011 1011 1011 1111 1111 1011 0111 1011 0111 0
1111 1111 1111 1111 1111 1111 1011 1011 1111 0111 0
1011 1111 1111 1111 0111 1111 0111 1011 1111 1111 0

1111111111*8031
```

```
   example    rp10    *
QP24*Q3290*
L00 0101 1111 1111 1111 1111 1111 0111 1111 1111 1111 0
    1101 0111 1111 1111 1111 1111 0111 0111 1111 1111 0
    1101 1111 0111 1111 1111 1111 0111 0111 1111 1111 0
    1101 1111 1111 0111 1111 1111 0111 0111 1111 1111 0
    1011 1011 1011 1011 1111 1111 1011 0111 0111 1011 0
    1011 1011 1011 1011 1111 1111 1011 0111 1011 0111 0
    1111 1111 1111 1111 1111 1111 1011 1011 1111 0111 0
    1011 1111 1111 1111 1111 0111 0111 0111 1111 1111 0
    0111 1101 1111 1111 1111 1111 0111 1111 1111 1111 01
    1111 0101 1111 1111 1111 1111 0111 0111 1111 1111 01
    1111 1101 0111 1111 1111 1111 0111 0111 1111 1111 01
    1111 1101 1111 0111 1111 1111 0111 0111 1111 1111 01
    1011 1011 1011 1011 1111 1111 1011 0111 0111 1011 01
    1011 1011 1011 1011 1111 1111 1011 0111 1011 0111 01
    1111 1111 1111 1111 1111 1111 1011 1011 1111 0111 01
    1011 1111 1111 1111 0111 1111 0111 1011 1111 1111 01
    1011 1111 1111 1111 1111 0111 0111 1011 1111 1111 10
    1111 1111 1111 1111 1111 1111 1011 1011 1111 0111 10
    1011 1011 1011 1011 1111 1111 1011 0111 1011 0111 10
    1011 1011 1011 1011 1111 1111 1011 0111 0111 1011 10
    1111 1111 1101 0111 1111 1111 0111 0111 1111 1111 10
    1111 1111 0101 1111 1111 1111 0111 0111 1111 1111 10
    1111 0111 1101 1111 1111 1111 0111 0111 1111 1111 10
    0111 1111 1101 1111 1111 1111 0111 1111 1111 1111 10
    0111 1111 1111 1101 1111 1111 0111 1111 1111 1111 01
    1111 0111 1111 1101 1111 1111 0111 0111 1111 1111 01
    1111 1111 0111 1101 1111 1111 0111 0111 1111 1111 01
    1111 1111 1111 0101 1111 1111 0111 0111 1111 1111 01
    1011 1011 1011 1011 1111 1111 1011 0111 0111 1011 01
    1011 1011 1011 1011 1111 1111 1011 0111 1011 0111 01
    1111 1111 1111 1111 1111 1111 1011 1011 1111 0111 01
    1011 1111 1111 1111 0111 1111 0111 1011 1111 1111 01
    1011 1111 1111 1111 1111 0111 0111 1011 1111 1111 10
    1111 1111 1111 1111 1111 1111 1011 1011 1111 0111 10
    1011 1011 1011 1011 1111 1111 1011 0111 1011 0111 10
    1011 1011 1011 1011 1111 1111 1011 0111 0111 1011 10
    1111 1111 1111 0111 1101 1111 0111 0111 1111 1111 10
    1111 1111 0111 1111 1101 1111 0111 0111 1111 1111 10
    1111 0111 1111 1111 1101 1111 0111 0111 1111 1111 10
    0111 1111 1111 1111 1101 1111 0111 1111 1111 1111 10
    0111 1111 1111 1111 1111 1101 0111 1111 1111 1111 01
    1111 0111 1111 1111 1111 1101 0111 0111 1111 1111 01
```

```
CHIP  EXAMPLE 20RP10
CL, A3, A2, A1, A0, D0, D1, I0, I1, DATA, P ,GND,
/OE, Q0, Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8, Q9, VCC

EQU

Q0 := A3 * Q0 * I0
    + A2 * Q0 * I0 * I1
    + A1 * Q0 * I0 * I1
    + A0 * Q0 * I0 * i1
    +/A3 * /A2 * /A1 * /A0 * DATA */I0 * I1 * /P
    +/A3 * /A2 * /A1 * /A0 * /DATA * /I0 * I1 * P
    + /I0 * /I1 * P
    + /A3 * D0 * I0 * /I1

Q1 := A3 * Q1 * I0
    + A2 * Q1 * I0 * I1
    + A1 * Q1 * I0 * I1
    + A0 * Q1 * I0 * I1
    +/A3 * /A2 * /A1 * /A0 * DATA * /I0 * I1 * /P
    +/A3 * /A2 * /A1 * /A0 * /DATA * /I0 * I1 * P
    +/I0 * /I1 * P
    +/A3 * D1 * I0 * /I1

 Q2 := A3 * Q2 * I0
    + A2 * Q2 * I0 * I1
    + A1 * Q2 * I0 * I1
    + A0 * Q2 * I0 * I1
    + /A3 * /A2 * /A1 * /A0 * DATA * /I0 * I1 * /P
    + /A3 * /A2 * /A1 * /A0 * /DATA * /I0 * I1 * P
    + /I0 * /I1 * P
    + /A3  * D0 * I0 * /I1

Q3 := A3 * Q3 * I0
    + A2 * Q3 * I0 *I1
    + A1 * Q3 * I0 *I1
    + A0 * Q3 * I0 *I1
    + /A3 * /A2 * /A1 * /A0 * DATA * /I0 * I1 * /P
    + /A3 * /A2 * /A1 * /A0 * /DATA * /I0 * I1 * P
    + /I0 * /I1 * P

Q4 := A3 * Q4 * I0
    + A2 * Q4 * I0 * I1
    + A1 * Q4 * I0 * I1
    + A0 * Q4 * I0 * I1
    +/A3 * /A2 * /A1 * /A0 * DATA * /I0 * I1 * /P
    +/A3 * /A2 * /A1 * /A0 * /DATA * /I0 * I1 * P
```

```
Q5 := A3 * Q5 * I0
   +  A2 * Q5 * I0 * I1
   +  A1 * Q5 * I0 * I1
   +  A0 * Q5 * I0 * I1
   + /A3 * /A2 * /A1 * /A0 * DATA * /I0 * I1 * /P
   + /A3 * /A2 * /A1 * /A0 * /DATA * /I0 * I1 * P
   + /I0 * /I1 * P
   + /A3 * D1 * I0 * /I1

Q6 := A3 * Q6 * I0
   +  A2 * Q6 * I0 * I1
   +  A1 * Q6 * I0 * I1
   +  A0 * Q6 * I0 * I1
   + /A3 * /A2 * /A1 * /A0 * DATA * /I0 * I1 * /P
   + /A3 * /A2 * /A1 * /A0 * /DATA * /I0 * I1 * P
   + /I0 * /I1 * P
   + /A3  * D0 * I0 * /I1

Q7 := A3 * Q7 * I0
   +  A2 * Q7 * I0 *I1
   +  A1 * Q7 * I0 * I1
   +  A0 * Q7 * I0 *I1
   + /A3 * /A2 * /A1 * /A0 * DATA * /I0 * I1 * /P
   + /A3 * /A2 * /A1 * /A0 * /DATA * /I0 * I1 * P
   + /I0 * /I1 * P
   + /A3  * D1 * I0 * /I1

Q8 := A3 * Q8 * I0
   +  A2 * Q8 * I0 *I1
   +  A1 * Q8 * I0 * I1
   +  A0 * Q8 * I0 * I1
   +  /A3 * /A2 * /A1 * /A0 * DATA * /I0 * I1 * /P
   + /A3  * /A2 * /A1 * /A0 * /DATA * /I0 * I1 *P
   + /I0 * /I1 * P
   + /A3 * D0 * I0 * /I1

Q9 := A3 * Q9 * I0
    + A2 * Q9 * I0 * I1
    + A1 * Q9 * I0 * I1
    + A0 * Q9 * I0 * I1
    + /A3 * /A2 * /A1 * /A0 * DATA * /I0 * I1 * /P
    + /A3 * /A2 * /A1 * /A0 * /DATA * /I0 * I1 * P
    + /I0 * /I1 * P
    + /A3 * D1 * I0 *I1
```

```
example         rp10
               11 1111 1111 2222 2222 2233 3333 3333
   0123 4567 8901 2345 6789 0123 4567 8901 2345 6789  01

 0 x-x- ---- ---- ---- ---- ---- x--- ---- ---- ----  x
 1 --x- x--- ---- ---- ---- ---- x--- x--- ---- ----  x
 2 --x- ---- x--- ---- ---- ---- x--- x--- ---- ----  x
 3 --x- ---- ---- x--- ---- ---- x--- x--- ---- ----  x
 4 -x-- -x-- -x-- -x-- ---- ---- -x-- x--- x--- -x--  x
 5 -x-- -x-- -x-- -x-- ---- ---- -x-- x--- -x-- x---  x
 6 ---- ---- ---- ---- ---- ---- -x-- -x-- ---- x---  x
 7 -x-- ---- ---- ---- ---- x--- x--- x--- ---- ----  x

 8 x--- --x- ---- ---- ---- ---- x--- ---- ---- ----  x-
 9 ---- x-x- ---- ---- ---- ---- x--- x--- ---- ----  x-
10 ---- --x- x--- ---- ---- ---- x--- x--- ---- ----  x-
11 ---- --x- ---- x--- ---- ---- x--- x--- ---- ----  x-
12 -x-- -x-- -x-- -x-- ---- ---- -x-- x--- x--- -x--  x-
13 -x-- -x-- -x-- -x-- ---- ---- -x-- x--- -x-- x---  x-
14 ---- ---- ---- ---- ---- ---- -x-- -x-- ---- x---  x-
15 -x-- ---- ---- ---- x--- ---- x--- -x-- ---- ----  x-
16 -x-- ---- ---- ---- ---- x--- x--- -x-- ---- ----  -x
17 ---- ---- ---- ---- ---- ---- -x-- -x-- ---- x---  -x
18 -x-- -x-- -x-- -x-- ---- ---- -x-- x--- -x-- x---  -x
19 -x-- -x-- -x-- -x-- ---- ---- -x-- x--- x--- -x--  -x
20 ---- ---- --x- x--- ---- ---- x--- x--- ---- ----  -x
21 ---- ---- x-x- ---- ---- ---- x--- x--- ---- ----  -x
22 ---- x--- --x- ---- ---- ---- x--- x--- ---- ----  -x
23 x--- ---- --x- ---- ---- ---- x--- ---- ---- ----  -x

24 x--- ---- ---- --x- ---- ---- x--- ---- ---- ----  x-
25 ---- x--- ---- --x- ---- ---- x--- x--- ---- ----  x-
26 ---- ---- x--- --x- ---- ---- x--- x--- ---- ----  x-
27 ---- ---- ---- x-x- ---- ---- x--- x--- ---- ----  x-
28 -x-- -x-- -x-- -x-- ---- ---- -x-- x--- x--- -x--  x-
29 -x-- -x-- -x-- -x-- ---- ---- -x-- x--- -x-- x---  x-
30 ---- ---- ---- ---- ---- ---- -x-- -x-- ---- x---  x-
31 -x-- ---- ---- ---- x--- ---- x--- -x-- ---- ----  x-
32 -x-- ---- ---- ---- ---- x--- x--- -x-- ---- ----  -x
33 ---- ---- ---- ---- ---- ---- -x-- -x-- ---- x---  -x
34 -x-- -x-- -x-- -x-- ---- ---- -x-- x--- -x-- x---  -x
35 -x-- -x-- -x-- -x-- ---- ---- -x-- x--- x--- -x--  -x
36 ---- ---- ---- x--- --x- ---- x--- x--- ---- ----  -x
37 ---- ---- x--- ---- --x- ---- x--- x--- ---- ----  -x
38 ---- x--- ---- ---- --x- ---- x--- x--- ---- ----  -x
39 x--- ---- ---- ---- --x- ---- x--- ---- ---- ----  -x
```

```
40 x--- ---- ---- ---- ---- --x- x--- ---- ---- ----  x-
41 ---- x--- ---- ---- ---- --x- x--- x--- ---- ----  x-
42 ---- ---- x--- ---- ---- --x- x--- x--- ---- ----  x-
43 ---- ---- ---- x--- ---- --x- x--- x--- ---- ----  x-
44 -x-- -x-- -x-- -x-- ---- ---- -x-- x--- x--- -x--  x-
45 -x-- -x-- -x-- -x-- ---- ---- -x-- x--- -x-- x---  x-
46 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx  xx
47 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx  xx
48 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx  xx
49 ---- ---- ---- ---- ---- ---- -x-- -x-- ---- x---  -x
50 -x-- -x-- -x-- -x-- ---- ---- -x-- x--- -x-- x---  -x
51 -x-- -x-- -x-- -x-- ---- ---- -x-- x--- x--- -x--  -x
52 ---- ---- ---- x--- ---- ---- x-x- x--- ---- ----  -x
53 ---- ---- x--- ---- ---- ---- x-x- x--- ---- ----  -x
54 ---- x--- ---- ---- ---- ---- x-x- x--- ---- ----  -x
55 x--- ---- ---- ---- ---- ---- x-x- ---- ---- ----  -x

56 x--- ---- ---- ---- ---- ---- x--- --x- ---- ----  x-
57 ---- x--- ---- ---- ---- ---- x--- x-x- ---- ----  x-
58 ---- ---- x--- ---- ---- ---- x--- x-x- ---- ----  x-
59 ---- ---- ---- x--- ---- ---- x--- x-x- ---- ----  x-
60 -x-- -x-- -x-- -x-- ---- ---- -x-- x--- x--- -x--  x-
61 -x-- -x-- -x-- -x-- ---- ---- -x-- x--- -x-- x---  x-
62 ---- ---- ---- ---- ---- ---- -x-- -x-- ---- x---  x-
63 -x-- ---- ---- ---- x--- ---- x--- -x-- ---- ----  x-
64 -x-- ---- ---- ---- ---- x--- x--- -x-- ---- ----  -x
65 ---- ---- ---- ---- ---- ---- -x-- -x-- ---- x---  -x
66 -x-- -x-- -x-- -x-- ---- ---- -x-- x--- -x-- x---  -x
67 -x-- -x-- -x-- -x-- ---- ---- -x-- x--- x--- -x--  -x
68 ---- ---- ---- x--- ---- ---- x--- x--- --x- ----  -x
69 ---- ---- x--- ---- ---- ---- x--- x--- --x- ----  -x
70 ---- x--- ---- ---- ---- ---- x--- x--- --x- ----  -x
71 x--- ---- ---- ---- ---- ---- x--- ---- --x- ----  -x

72 x--- ---- ---- ---- ---- ---- x--- ---- ---- --x-  x
73 ---- x--- ---- ---- ---- ---- x--- x--- ---- --x-  x
74 ---- ---- x--- ---- ---- ---- x--- x--- ---- --x-  x
75 ---- ---- ---- x--- ---- ---- x--- x--- ---- --x-  x
76 -x-- -x-- -x-- -x-- ---- ---- -x-- x--- x--- -x--  x
77 -x-- -x-- -x-- -x-- ---- ---- -x-- x--- -x-- x---  x
78 ---- ---- ---- ---- ---- ---- -x-- -x-- ---- x---  x
79 -x-- ---- ---- ---- x--- ---- x--- -x-- ---- ----  x

Polarity Fuses

Output pin: 1111112222
            4567890123

            ----------

Legend:  fuse blown: -   fuse intact: x

Fuses Blown : 2772
```

```
C>RPPAL

C>PARSER
 Enter input file[specs.dat] : SPECS41.DAT

C>20RP10

****** ERROR !!! *******



q0

OUTPUT HAS MORE THAN 8 PRODUCT TERMS


C>

C>
```

```
C>RPPAL

C>PARSER
 Enter input file[specs.dat] : SPECS42.DAT

C>20RP10

****** ERROR !!! *******


q1               and q2

 HAS MORE THAN 16  PRODUCT  TERMS


C>

C>
```

```
C>RPPAL

C>PARSER
 Enter input file[specs.dat] : SPECS43.DAT

C>20RP10
```

```
q9

OUTPUT HAS MORE THAN 8 PRODUCT TERMS


C>

C>




C>RPPAL

C>PARSER
 Enter input file[specs.dat] : SPECS45.DAT

C>20RP10

****** ERROR !!! *******


a32                 UNDEFINED SIGNAL NAME


C>

C>






C>RPPAL

C>PARSER
 Enter input file[specs.dat] : SPECS44.DAT

C>20RP10

****** ERROR !!! *******


cl

THIS SIGNAL IS PIN 1 AND CANNOT BE USED AS
A SIGNAL IN AN EQUATION

C>
```

```
CHIP  EXAMPLE 20RP10
CL, A3, A2, Al, A0, D0, D1, I0, Il, DATA, P ,GND,
/OE, Q0, Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8, Q9, VCC

EQU

Q0 := A3 * Q0 * I0
    + A2 * Q0 * I0 * Il
    + Al * Q0 * I0 * Il
    + A0 * Q0 * I0 * il
    +/A3 * /A2 * /Al * /A0 * DATA */I0 * Il * /P
    +/A3 * /A2 * /Al * /A0 * /DATA * /I0 * Il * P
    + /I0 * /Il * P
    + /A3 * D0 * I0 * /Il
    + A3 * Q0 * /I0
```

```
CHIP  EXAMPLE 20RP10
CL, A3, A2, A1, A0, D0, D1, I0, I1, DATA, P ,GND,
/OE, Q0, Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8, Q9, VCC

EQU


Q1 := A3 * Q1 * I0
    + A2 * Q1 * I0 * I1
    + A1 * Q1 * I0 * I1
    + A0 * Q1 * I0 * I1
    +/A3 * /A2 * /A1 * /A0 * DATA * /I0 * I1 * /P
    +/A3 * /A2 * /A1 * /A0 * /DATA * /I0 * I1 * P
    +/I0 * /I1 * P
    +/A3 * D1 * I0 * /I1
    + A3 * /Q1 * /I0 * I1

 Q2 := A3 * Q2 * I0
    + A2 * Q2 * I0 * I1
    + A1 * Q2 * I0 * I1
    + A0 * Q2 * I0 * I1
    + /A3 * /A2 * /A1 * /A0 * DATA * /I0 * I1 * /P
    + /A3 * /A2 * /A1 * /A0 * /DATA * /I0 * I1 * P
    + /I0 * /I1 * P
    + /A3  * D0 * I0 * /I1
```

```
CHIP  EXAMPLE 20RP10
CL, A3, A2, A1, A0, D0, D1, I0, I1, DATA, P ,GND,
/OE, Q0, Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8, Q9, VCC

EQU


Q9 := A3 * Q9 * I0
    + A2 * Q9 * I0 * I1
    + A1 * Q9 * I0 * I1
    + A0 * Q9 * I0 * I1
    + /A3 * /A2 * /A1 * /A0 * DATA * /I0 * I1 * /P
    + /A3 * /A2 * /A1 * /A0 * /DATA * /I0 * I1 * P
    + /I0 * /I1 * P
    + /A3 * D1 * I0 *I1
    + A3 * /Q9 * /I0
```

To run CP/M PALASM, the following are minimum system requirements:

- 64K bytes of memory
- 2 disk drives
- CP/M operating system
- Optional text printer

Key Features

- Assembles PAL Design Specifications for both 20- and 24-pin PALS.
- PAL fuse patterns generated in various programmer formats.
- Available in industry standard 8-inch single-sided single-density CP/M format. Special requests are necessary for 5.25-inch disks. This is because of the various permutations of disk formats for such disks.
- This version of PALASM should run on all standard CP/M systems. These include: Altos, Heath with Magnolia, TRS-80 Model 2, Osborne 1, etc.

The CP/M version of PALASM is written in Microsoft BASIC. It has two program segments and a data table.

- PAL1 is the driver program which assembles the equations and generates the fuse plot and puts it into a temporary work file called PALTEMP.DAT. Unlike other versions of PALASM, this one module handles BOTH 20- and 24-pin PALs.
- PAL2 uses PALTEMP.DAT to generate the various programmer formats.
- PALTABLE.DAT is a data table which describes the structure of each PAL type.

The CP/M package you receive should have three 8-inch disks. Below is a directory of each disk.

Disk #1: PAL1.COM, PAL2.COM, PALTABLE.DAT and P7000.DAT-P7039.DAT
Disk #2: P7040.DAT-P7079.DAT
Disk #3: P7080.DAT-P7098.DAT

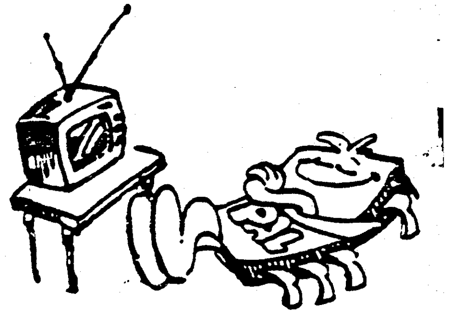
P7000.DAT-P7098.DAT are are design examples from the PAL handbook.

As with other versions of PALASM, two steps are necessary:

1. Generate the PALASM specification file using your favorite editor.
2. Run PALASM itself, which is what we will discuss here.

To start, when you see the A> prompt, type the following:

```
A>PAL1

    (c) Copyright 1983 Monolithic Memories Inc. All Rights Reserved"

                    PALASM 20/24 in Basic
                    Revision Level 2.0
                    07/15/81 D. Jones
                    06/22/83 U. Mueller & C.B. Lee

    Note: When using the 20X PALs in the series 24
    family, the XOR operator ':+:' should start a new
    line.  Thus, /Q1 := A*B + C*D :+: E*F + G*H
    is an error
    It should read:
           /Q1 := A*B + C*D    or      /Q1:= A*B
               :+: E*F + G*H                 + C*D
                                            :+: E*F
    The second format is recommended         + G*H
    for ease of reading and commenting.
    Note also a space is required before and after
    the '+' in the first format.

    Press a key to continue...
```



(Pressing any key clears the screen and provides the next prompt
(at the bottom of the screen. The exchange shown below assumes that
(the data file is called P7000.DAT and is located in drive A together
with PAL1.COM and PAL2.COM:.

```
    What is your input filename?P7000.DAT
```

(You will then see a series of prompts which indicate the type of
(processing being performed.

```
    ASSEMBLING...PLEASE WAIT !!!
    PART NUMBER ... OK !!!
    PIN LIST ...... OK !!!
```

(As each equation is assembled, PALASM prompts with a message
(for each output, the general form of which is shown below. Each
(output is identified by name and the corresponding product line is
(given. You should see a number incrementing at the bottom left hand
(corner of the screen. This is the current product line being processed.
(For 20-pin PALs, this number should count from 0 to 63 while for 24-pin
(PALs the count is from 0 to 79.

```
    ASSEMBLING OUTPUT:   B; PL = 8
    ASSEMBLING OUTPUT:   E; PL = 16
    ASSEMBLING OUTPUT:   H; PL = 24
    ASSEMBLING OUTPUT:   L; PL = 48
    ASSEMBLING OUTPUT:   O; PL = 32
    ASSEMBLING OUTPUT:   R; PL = 40
    _17
```

(When assembly is complete, you will see the following options
(menu on the screen. Either lower or upper case letters can be used.
(At this time, PAL2 is automatically loaded and executed. As PALTEMP.DAT
(is being read, the product line count is repeated.

```
    (c) Copyright 1983 Monolithic Memories Inc. All Rights Reserved

    Enter option at this time:
    X - Xplot
    B - Brief Xplot
    H - Hex (Ascii Hex programmer format)
    N - BPNF (Ascii programmer format)
    L - BHLF (Ascii programmer format)
    P - Program PAL (SD20/24 format)
    O - Pinout
    J - Jedec format
    E - Echo (Reprints PAL design spec.)
    R - Restart PAL-assembler
    Q - Quit (End program)

    OPTION (without Return) ?
```

(For each of these options, the user has the choice of directing
(the output to the screen or creating a new output file. The example
(below directs the output to the new file BASIC.GAT.

```
    Enter filename for output (return for none)
    B:BASIC.GAT
```

X - Generates complete PAL fuse plot, where
 "X" - unblown fuse
 "-" - fuse blown
 "o" - low phantom fuse
 "O" - high phantom fuse

B - Generates brief fuse plot displaying only those fuse lines which are essential to describe the function.

H - Generates PAL fuse pattern in hexadecimal (HEX) programming format.

N - Provides BPNF programming format.

L - Gives BHLF programming format.

P - Downloads the PAL Design Specification into the SD20/24 PAL programm
When this option is picked, PALASM responds with:

```
Is programmer 'ON' and 'RESET'
```

If the SD20/24 is now turned on, PALASM starts sending the PAL Design Specification file to the RS232 port. You will then see:

```
WAIT - LOADING ..."
```

When loading is completed, the next prompt is displayed:

```
On yellow light : Programmer is busy - Don't
                  Touch anything !!!

On green light  : Insert your PAL and press
                  <PROG> on the programmer

On red light    : Assembling Error - Your PAL
                  Design Spec. is wrong.
```

O - Provides a pinout of the input spec.

J - JEDEC format. This is the format used bv most DATA I/O programmers. You have the choice of sending the output to the serial port (DATA or to the output file you specified. The program prompts with:

```
Press 1 for DATA I/O
      2 for File or
<return> for none of both
```

After you respond, you will see the next screen:

```
Power on Data I/O.
For information about how to setup the Data I/O
to receive mode, look in the PALASM manual under
Appendix B:

Press <return> when ready...
```

After this press <RETURN> on the computer.

As the JEDEC file is being down-loaded, the next prompt is:

```
Please Wait !!!
You are now down-loading to the Data I/O.
```

R - Restarts the PAL assembler.

E - Echos the PAL Design Specification file.

Q - Exits the program and returns to the operating system.

Caveats, Restrictions and Other Gotchas

General

1. At present, the simulator and fault grading options have not yet been implemented on the CP/M version. When these become available, an update will be sent out to you.

2. For communications with your PAL programmer, make certain that the LST: device for your system is a serial port.