PAL
PROGRAMMABLE ARRAY LOGIC
Handbook

Monolithic Memories
Bipolar is our business.

$5.00

PAL
PROGRAMMABLE ARRAY LOGIC
Handbook

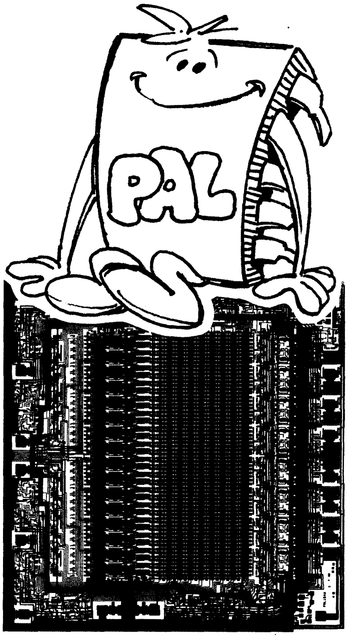
Patent Allowed

First Edition

author John Birkner

contributors, { *Bill Black*
Paul Franklin
Shlomo Waser

Monolithic Memories
Bipolar is our business.



The PAL Concept

Monolithic Memories' family of PAL devices gives designers a powerful tool with unique capabilities for use in new and existing logic designs. The PAL saves time and money by solving many of the system partitioning and interface problems brought about by increases in semiconductor device technology.

Rapid advances in large scale integration technology have led to larger and larger standard logic functions; single I.C.s now perform functions that formerly required complete circuit cards. While LSI offers many advantages, advances have been made at the expense of device flexibility. Most LSI devices still require large numbers of SSI/MSI devices for interfacing with user systems. Designers are still forced to turn to random logic for many applications.

The "complexity gap" between TTL and LSI devices has led to ineffective use of both. The TTL provides the speed and flexibility, but it is ineffective in terms of package count, power consumption, and P.C. board space. LSI offers high functional density and low power consumption, but it is slow and rigidly partitioned. Even the microprocessor, widely acclaimed for its flexibility, is slow and expensive when the costs of programming and support interfaces are considered.

The designer is confronted with another problem when a low to medium complexity product is designed. Often the function is well defined and could derive significant benefits from fabrication as an integrated circuit. However, the design cycle for a custom circuit is long and the costs can be very high. This makes the risk significant enough to deter most users. The technology to support maximum flexibility combined with fast turn around on custom logic has simply not been available.

Attempts to solve these problems have led to increasing interest in fuse programmable logic devices. These devices can all be user configured to provide custom functions. PROMs, FPLAs, FPGAs, and PMUXs have all been used in this way. These approaches have met with some success, but all are deficient in one or more areas. PROMs require careful design to avoid undesirable data transitions; FPLAs are expensive, difficult to program and complex to understand; FPGAs and PMUXs are not widely available and lack flexibility. All of these devices still require extensive interface logic for use in systems.

The PAL family offers a fresh approach to using fuse programmable logic. PALs are a conceptually unified group of devices which combine programmable flexibility with high speed and an extensive selection of interface options. PALs can lower inventory, cut design cycles and provide high complexity with maximum flexibility. These features, combined with lower package count and high reliability, truly make the PAL a circuit designer's best friend.

The PAL—Teaching Old PROMs New Tricks

MMI developed the modern PROM and introduced many of the architectures and techniques now regarded as industry standards. As the world's largest PROM manufacturer, MMI has the proven technology and high volume production capability required to manufacture and support the PAL.

The PAL is an extension of the fusible link technology pioneered by Monolithic Memories for use in bi-polar PROMs. The fusible link PROM first gave the digital systems designer the power to "write on silicon." In a few seconds he was able to transform a blank PROM from a general purpose device into one containing a custom algorithm, microprogram, or Boolean transfer function. This opened up new horizons for the use of PROMs in computer control stores, character generators, data storage tables and many other applications. The wide acceptance of this technology is clearly demonstrated by today's multi-million dollar PROM market.

The key to the PROM's success is that it allows the designer to quickly and easily customize the chip to fit his unique requirements. The PAL extends this programmable flexibility by utilizing proven fusible link technology to implement logic functions. Using PALs the designer can quickly and effectively implement custom logic varying in complexity from random gates to complex arithmetic functions.

ANDs and ORs

The PAL implements the familiar sum of products logic by using a programmable AND array whose output terms feed a fixed OR array. Since the sum of products form can express any Boolean transfer function, the PAL's uses are only limited by the number of terms available in the AND - OR arrays. PAL's come in different sizes to allow for effective logic optimization.

Figure 1 shows the basic PAL structure for a two input, one output logic segment. The general logic equation for this segment is

$$\text{Output} = (I_1f_1+\overline{f_1})(\overline{I_1}f_2+\overline{f_2})(I_2f_3+\overline{f_3})(\overline{I_2}f_4+\overline{f_4}) + (I_1f_5+\overline{f_5})(\overline{I_1}f_6+\overline{f_6})(I_2f_7+\overline{f_7})\ (\overline{I_2}f_8+\overline{f_8})$$

where the "f" terms represent the state of the fusible links in the PAL's AND array. An unblown link represents a logic 1. Thus,

fuse blown, f = 0

fuse intact, f = 1

An unprogrammed PAL has all fuses intact.

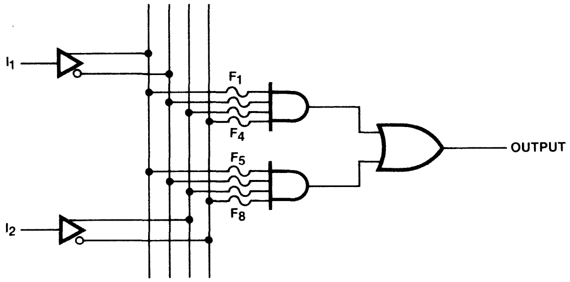


Figure 1

New Device, New Notation

Logic equations, while convenient for small functions, rapidly become cumbersome in large systems. To reduce possible confusion, complex logic networks are generally defined by logic diagrams and truth tables. Figure 2 shows the logic convention adopted to keep PAL logic easy to understand and use. In the figure, an "x" represents an intact fuse used to perform the logic AND function. (Note: the input terms on the common line with the x's are not connected together.) The logic symbology shown in Figure 2 has been informally adopted by integrated circuit manufacturers because it clearly establishes a one-to-one correspondence between the chip layout and the logic diagram. It also allows the logic diagram and truth table to be combined into a compact and easy to read form, thereby serving as a convenient shorthand for PALs. The two input - one output example from Figure 1 redrawn using the new logic convention is shown in Figure 3.

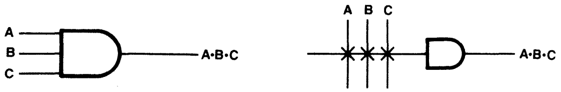


Figure 2

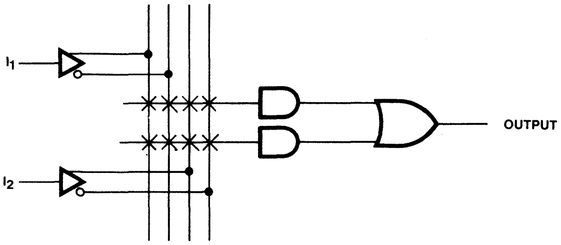


Figure 3

As a simple PAL example, consider the implementation of the transfer function:

$$\text{Output} = I_1\overline{I_2} + \overline{I_1}I_2$$

The normal combinatorial logic diagram for this function is shown in figure 4, with the PAL logic equivalent shown in figure 5.

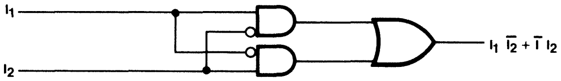


Figure 4

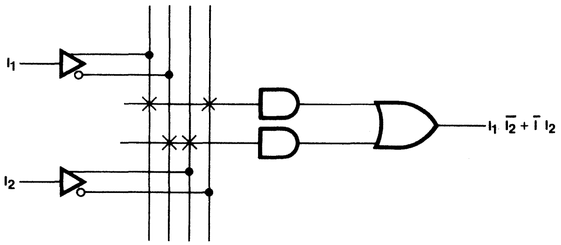


Figure 5

Using this logic convention it is now possible to compare the PAL structure to the structure of the more familiar PROM and PLA. The basic logic structure of a PROM consists of a fixed AND array whose outputs feed a programmable OR array (figure 6). PROMs are low-cost, easy to program, and available in a variety of sizes and organizations. They are most commonly used to store computer programs and data. In these applications the fixed input is a computer memory address; the output is the contents of that memory location.

PROM
16 Words X4 Bits

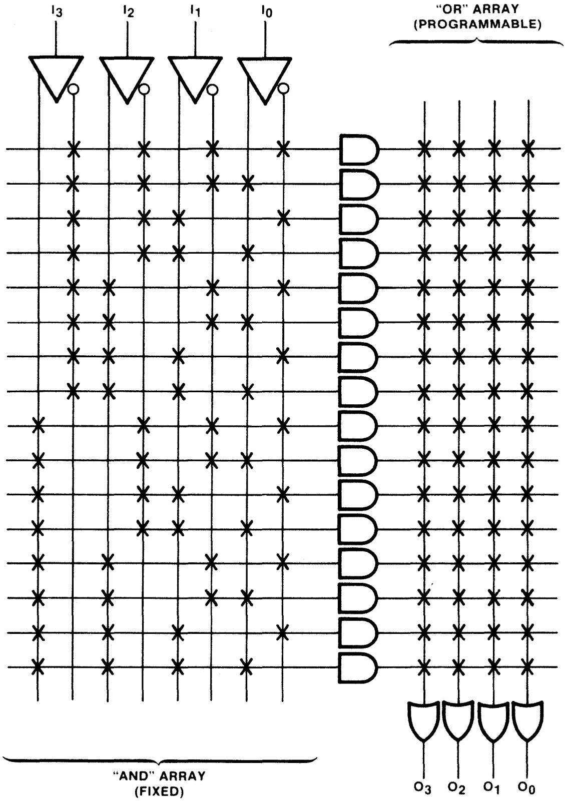


Figure 6

The basic logic structure of the PLA consists of a programmable AND array whose outputs feed a programmable OR array (Figure 7). Since the designer has complete control over all inputs and outputs, the PLA provides the ultimate flexibility for implementing logic functions. They are used in a wide variety of applications. However, this generality makes PLA's expensive, quite formidable to understand, and are costly to program (they require special programmers).

The basic logic structure of the PAL, as mentioned earlier, consists of a programmable AND array whose outputs feed a fixed OR array (Figure 8). The PAL combines much of the flexibility of the PLA with the low cost and easy programmability of the PROM. Table 1 summarizes the characteristics of the PROM, PLA, and PAL logic families.

FPLA
4 In•4 Out•16 Products

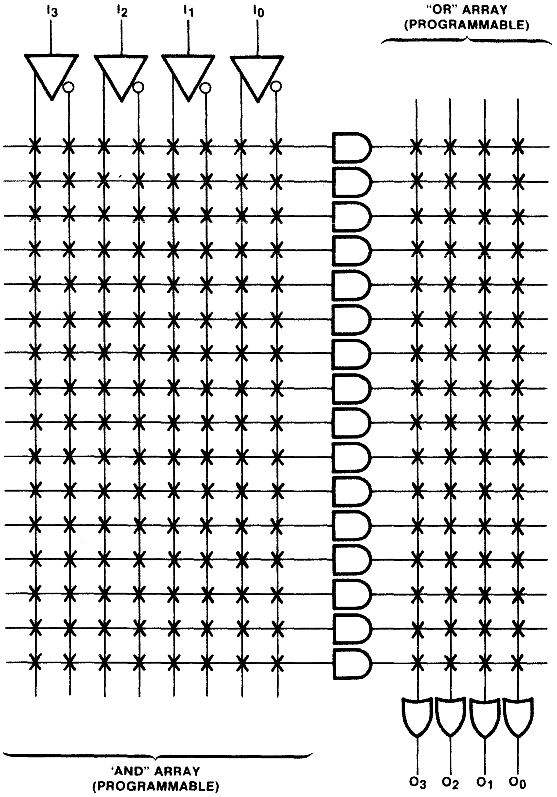


Figure 7

PAL
4 In•4 Out•16 Products

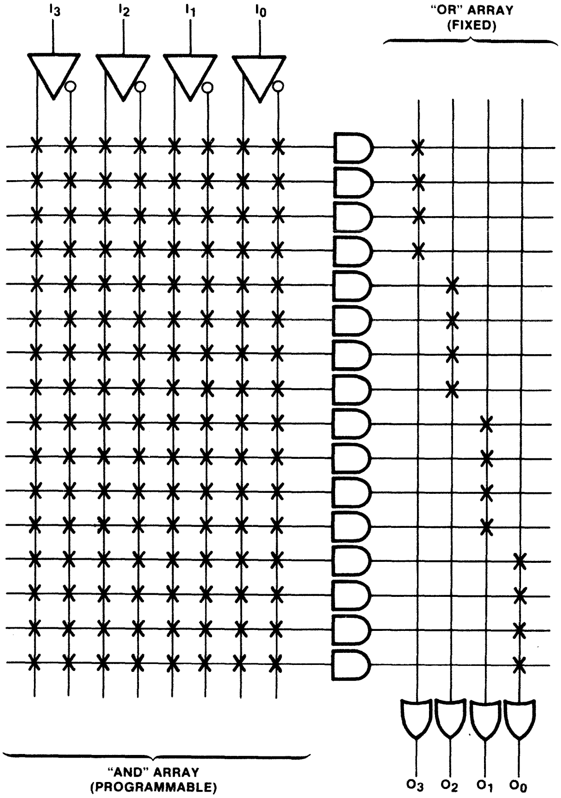


Figure 8

	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

Table 1

PALs For Every Task

The members of the PAL family and their characteristics are summarized in Table 2. They are designed to cover the spectrum of logic functions at reduced cost and lower package count. This allows the designer to select the PAL that best fits his application. PALs come in the following basic configurations:

Gate Arrays

PAL gate arrays are available in sizes from 10 x 8 (10 input terms, 8 output terms) to 16 x 2, with both active high and active low output configurations available. This wide variety of input/output formats allows the PAL to replace many different sized blocks of combinatorial logic with single packages.

Registered Outputs with Feedback

High end members of the PAL family feature registered data outputs with register feedback. Each product term is stored into a D-type output flip-flop on the rising edge of the system clock (Figure 9). The Q output of the flip-flop can then be gated to the output pin by enabling the active low three-state buffer.

In addition to being available for transmission, the Q output is fed back into the PAL array as an input term. This feedback

PAL input/output/function chart

PART NUMBER	INPUT	OUTPUT	PROGRAMMABLE I/O's	FEEDBACK register	OUTPUT polarity	FUNCTIONS	T_{PD} ns, TYP	I_{OL} mA	I_{CC} mA, TYP
PAL10H8	10	8			AND-OR	AND-OR GATE ARRAY	25	8	55
PAL12H6	12	6			AND-OR	AND-OR GATE ARRAY	25	8	55
PAL14H4	14	4			AND-OR	AND-OR GATE ARRAY	25	8	55
PAL16H2	16	2			AND-OR	AND-OR GATE ARRAY	25	8	55
PAL10L8	10	8			AND-NOR	AND-OR INVERT GATE ARRAY	25	8	55
PAL12L6	12	6			AND-NOR	AND-OR INVERT GATE ARRAY	25	8	55
PAL14L4	14	4			AND-NOR	AND-OR INVERT GATE ARRAY	25	8	55
PAL16L2	16	2			AND-NOR	AND-OR INVERT GATE ARRAY	25	8	55
PAL16C1	16	2			BOTH[1]	AND-OR GATE ARRAY	25	8	55
PAL16L8	10	8	6		AND-NOR	AND-OR INVERT GATE ARRAY	25	24	140
PAL16R8	8	8		8	AND-NOR	AND-OR INVERT ARRAY W/REG'S	25	24	150
PAL16R6	8	8	2	6	AND-NOR	AND-OR INVERT ARRAY W/REG'S	25	24	150
PAL16R4	8	8	4	4	AND-NOR	AND-OR INVERT ARRAY W/REG'S	25	24	150
PAL16X4	8	8	4	4	AND-NOR	AND-OR-XOR INVERT W/REG'S	25	24	160
PAL16A4	8	8	4	4	AND-NOR	AND-CARRY-OR-XOR INVERT W/REG'S	25	24	160

[1] Simultaneous AND-OR and AND-NOR outputs

Table 2

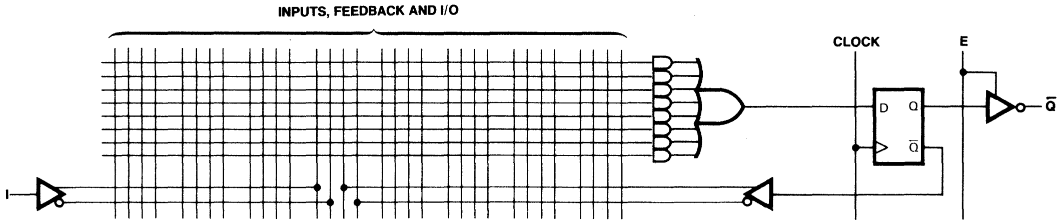


Figure 9

allows the PAL to "remember" the previous state, and it can alter its function based upon that state. This allows the designer to configure the PAL as a state sequencer which can be programmed to execute such elementary functions as count up, count down, skip, shift, and branch. These functions can be executed by the registered PAL at rates of up to 20 mHz.

Programmable I/O

Another feature of the high-end members of the PAL family is programmable input/output. This allows the product terms to directly control the outputs of the PAL (Figure 10). One product term is used to enable the three-state buffer, which in turn gates the summation term to the output pin. The output is also fed back into the PAL array as an input. Thus the PAL drives the I/O pin when the three-state gate is enabled; the I/O pin is an input to the PAL array when the three-state gate is disabled. This feature can be used to allocate available pins for I/O functions or to provide bi-directional output pins for operations such as shifting and rotating serial data.

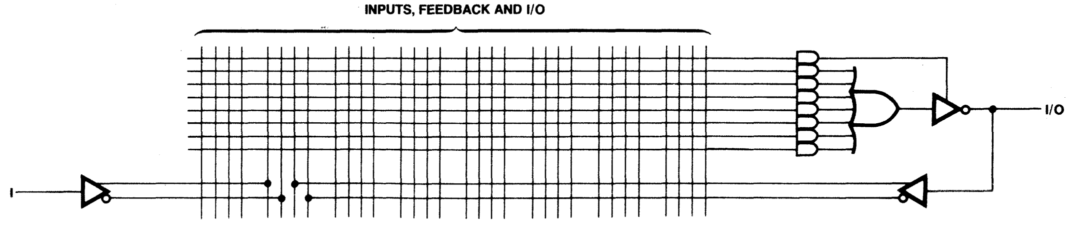


Figure 10

Arithmetic Functions

The arithmetic functions add, subtract, greater than, and less than are implemented by two additional features of the registered PAL (Figure 11). First, the sum of products is segmented into two sums exclusive OR ed (XOR) at the input of the D-type flip-flop. This allows carrys from previous operations to be XOR ed with the two variable sums generated by the PAL array. Second, the flip-flop's Q output is combined with input terms to form $I+Q$, $I+\overline{Q}$, $\overline{I}+Q$, and $\overline{I}+\overline{Q}$ terms which are then fed back into the PAL array as inputs. This option provides for versatile operations on two variables and facilitates the parallel generation of carrys necessary for fast arithmetic operations.

Figure 12 shows how the PAL array can be programmed to combine the available terms to form sixteen logical products in an ALU or controller application.

It should now be clear that the PAL family can replace most Small-Scale Integrated Logic (SSI) logic in use today, thereby lowering product cost and giving the designer even greater flexibility in implementing logic functions.

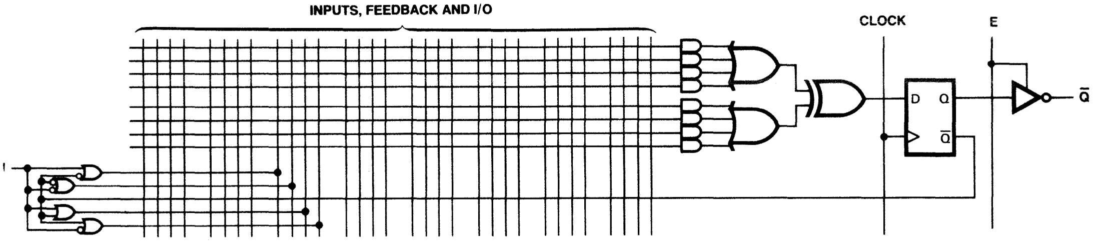


Figure 11

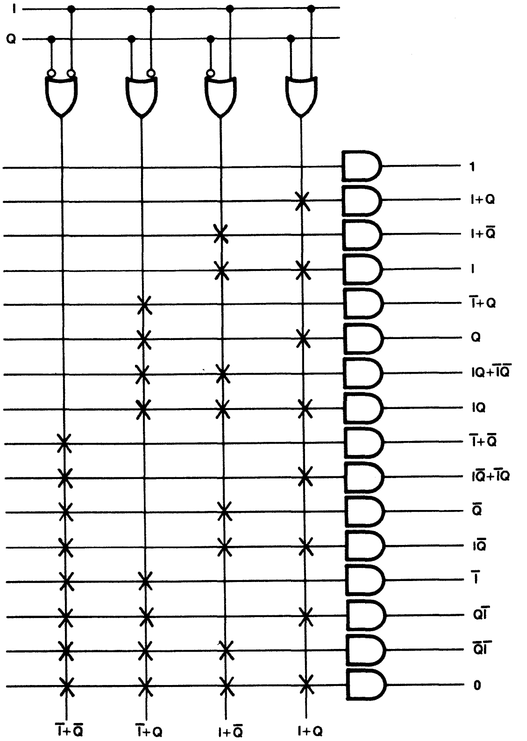


Figure 12

PAL Technology

PALs are manufactured using the proven TTL Schottky bipolar process used to make fusible-link PROMs. An NPN emitter follower array forms the programmable AND array. PNP inputs provide high-impedance inputs (.25 mA max.) to the array. All outputs are standard TTL drivers with internal active pull-up transistors. Typical PAL propagation delay time is 25 ns, and all PALs are packaged in space saving 20-pin "skinny-dips".

PAL Programming

PALs can be programmed in any standard PROM programmer with the addition of a PAL personality card. The PAL appears to the programmer as a 512 x 4 PROM. During programming four of the PAL outputs are selected for programming while the other four outputs and the eight inputs are used for addressing. The outputs are then switched to program the other locations. Verification uses the same procedure with the programming lines held in a low state.

PAL Data Security

The circuitry used for programming and logic verification can be used at any time to determine the logic pattern stored in the PAL array. For security, the PAL has a "last fuse" which can be blown to disable the verification logic. This provides a significant deterrent to potential copiers, and it can be used to effectively protect proprietary designs.

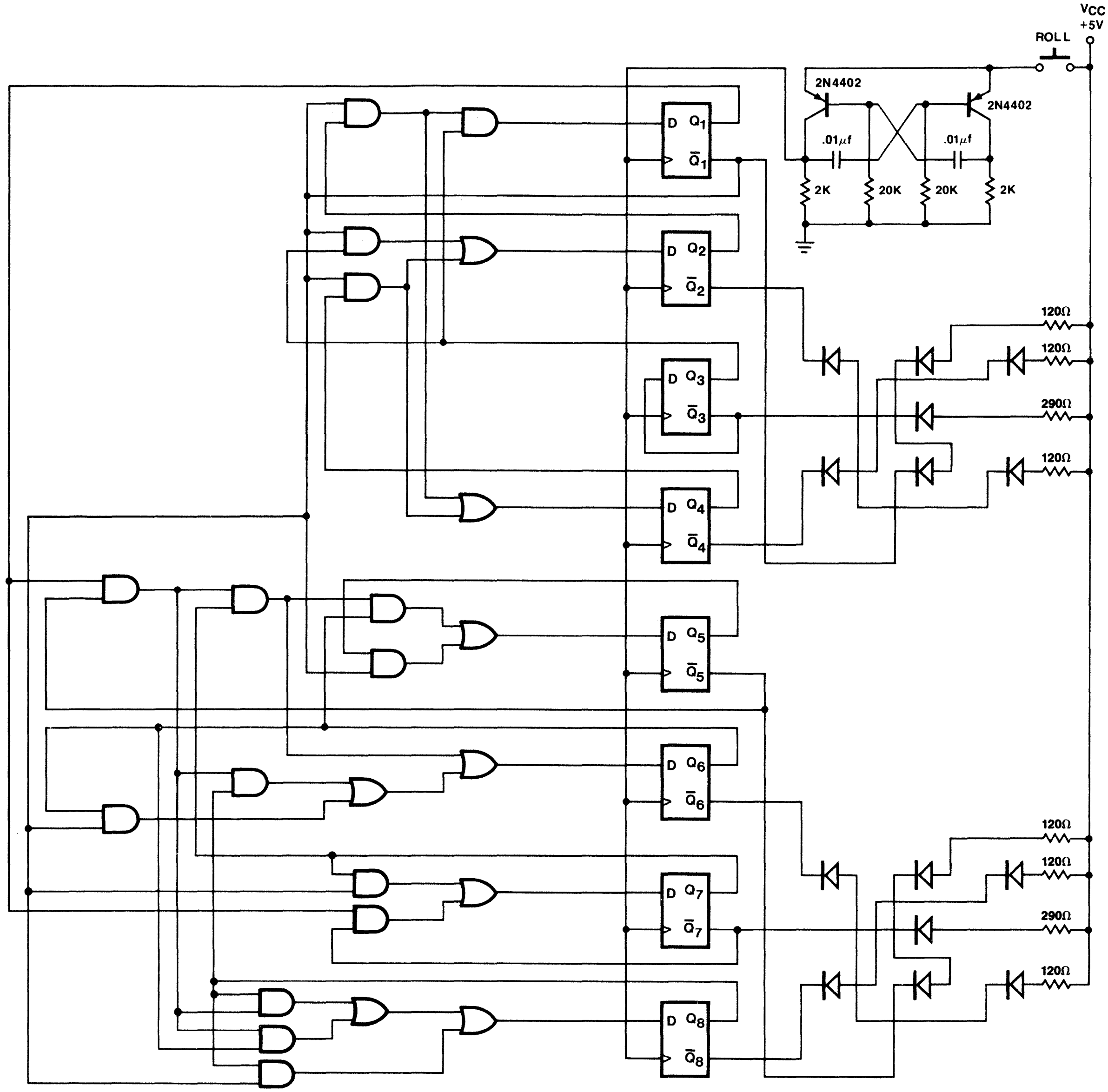

VCC
+5V
ROLL
2N4402
2N4402
.01μf
.01μf
2K
20K
20K
2K
D Q1
Q̄1
D Q2
Q̄2
D Q3
Q̄3
D Q4
Q̄4
D Q5
Q̄5
D Q6
Q̄6
D Q7
Q̄7
D Q8
Q̄8
120Ω
120Ω
290Ω
120Ω
120Ω
120Ω
290Ω
120Ω

Figure 13

PAL Part Numbers

The PAL part number is unique in that the part number code also defines the part's logic operation. The PAL parts code system is shown in Figure 14. For example, a PAL14L4CN would be a 14 input term, 4 output term, active-low PAL with a commercial temperature range packaged in a 20-pin plastic dip.

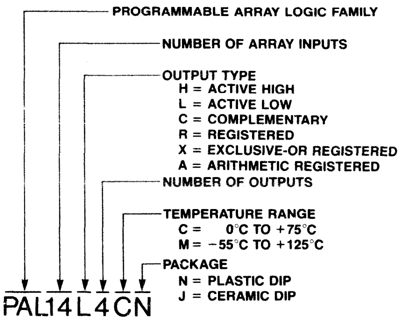


Figure 14

PAL Logic Symbols

The logic symbols for each of the individual PAL devices gives a concise functional description of that PAL's logic function. This symbol makes a convenient reference when selecting the PAL that best fits a specific application. Figure 15 shows the logic symbol for a PAL10H8 gate array.

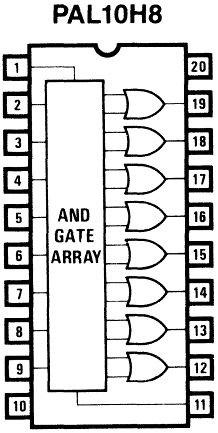


Figure 15

A PAL Example

As an example of how the PAL enables the designer to reduce costs and simplify logic design, consider the design of a simple, high-volume consumer product: an electronic dice game. This type of product will be produced in extremely high volume, so it is essential that every possible production cost be minimized.

The electronic dice game is simply constructed using a free running oscillator whose output is used to drive two asynchronous modulo six counters. When the user "rolls" the dice (presses a button), the current state of the counters is decoded and latched into a display resembling the pattern seen on an ordinary pair of dice.

A conventional logic diagram for the dice game is shown in Figure 13. (A detailed logic derivation is shown in the PAL applications section of this manual.) It is implemented using standard TTL, SSI and MSI parts, with a total I.C. count of eight: six quad gate packages and two quad D-latches. Looks like a nice, clean logic design, right? Wrong!!

PAL Goes to the Casino

A brief examination of Figure 13 reveals two basic facts: first, the circuit contains mostly simple, combinatorial logic, and second, it uses a clocked state transition sequence. Remembering that the PAL family contains ample provision for these features, the PAL catalog is consulted. The PAL16R8 has all the required functions, and the entire logic content of the circuit can be programmed into a single PAL shown in Figure 16.

In this example, the PAL effected an eight to one package count reduction and a significant cost savings. This is typical of the power and cost-effective performance that the PAL family brings to logic design.

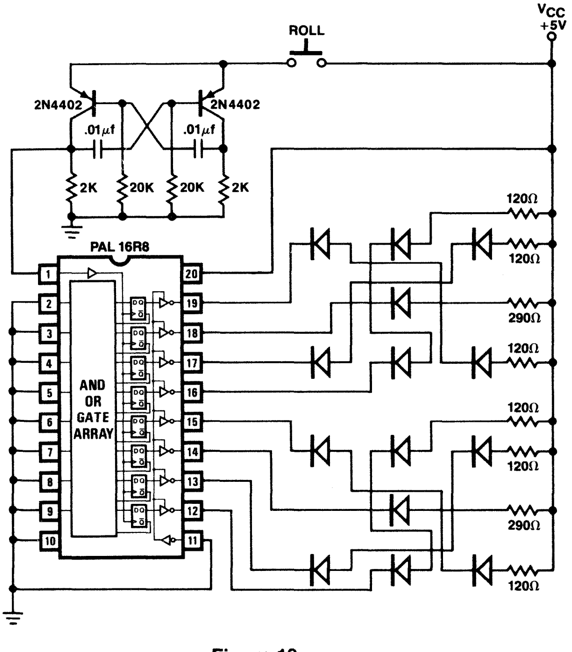
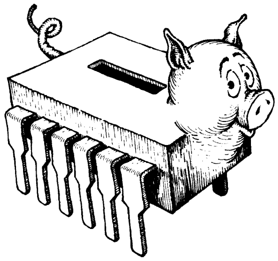


Figure 16

Advantages of Using PALs



The PAL has a unique place in the world of logic design. Not only does it offer many advantages over conventional logic, it also provides many features not found anywhere else. The PAL family:

- Programmable replacement for conventional TTL logic.
- Reduces IC inventories substantially and simplifies their control.
- Reduces chip count by 4 to 1.
- Expedites and simplifies prototyping and board layout.
- Saves space with 20-pin Skinny DIP packages.
- High speed: 25ns typical propagation delay.
- Programmed on standard PROM programmers.
- Programmable three-state outputs.
- Special feature eliminates possibility of copying by competitors.

All of these features combine together to lower product development costs and increase product cost effectiveness. The bottom line is that PALs save money.

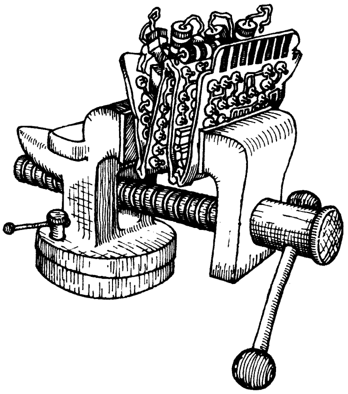
Direct Logic Replacement

In both new and existing designs the PAL can be used to replace various logic functions. This allows the designer to optimize a circuit in many ways never before possible. The PAL is particularly effective when used to provide interfaces required by many LSI functions. PAL flexibility combined with LSI function density makes a powerful team.

Design Flexibility

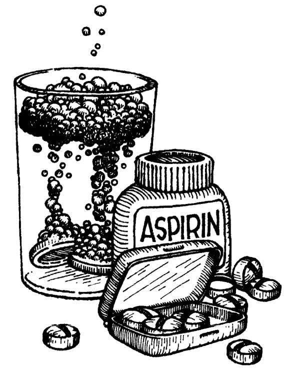
The PAL offers the systems logic designer a whole new world of options. Until now, the decision on logic system implementation was usually between SSI/MSI logic functions on one hand and microprocessors on the other. In many cases the function required is too awkward to implement the first way and too simple to justify the second. Now the PAL offers the designer high functional density, high speed, and low cost. Even better, PALs come in a variety of sizes and functions, thereby further increasing the designer's options.

Space Efficiency



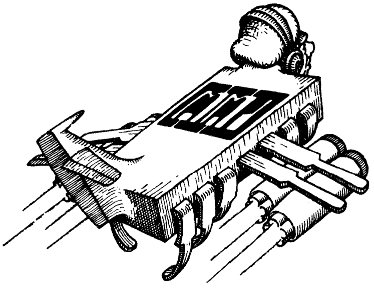
By allowing designers to replace many simple logic functions with single packages, the PAL allows more compact P.C. board layouts. The PAL's space saving 20-pin "skinny dip" helps to further reduce board area while simplifying board layout and fabrication. This means that many multi-card systems can now be reduced to one or two cards, and that can make the difference between a profitable success or an expensive disaster.

Smaller Inventory



The PAL family can be used to replace up to 90% of the conventional TTL family with just 15 parts. This considerably lowers both shelving and inventory cataloging requirements. Even better, small custom modifications to the standard functions are easy for PAL users, not so easy for standard TTL users.

High Speed

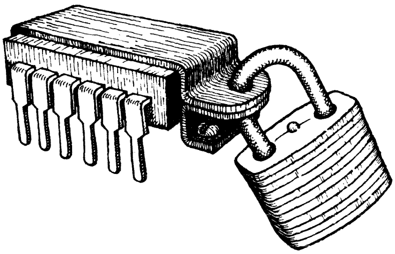


The PAL family runs faster or equal to the best of bipolar logic circuits. This makes the PAL the ideal choice for most logical operations or control sequence which requires a medium complexity and high speed. Also, in many microcomputer systems, the PAL can be used to handle high speed data interfaces that are not feasible for the microprocessor alone. This can be used to significantly extend the capabilities of the low-cost, low-speed NMOS microprocessor into areas formerly requiring high-cost bipolar microprocessors.

Easy Programming

The members of the PAL family can be quickly and easily programmed using standard PROM programmers. This allows designers to use PALs with a minimum investment in special equipment. Many types of programmable logic, such as the FPLA, require an expensive, dedicated programmer.

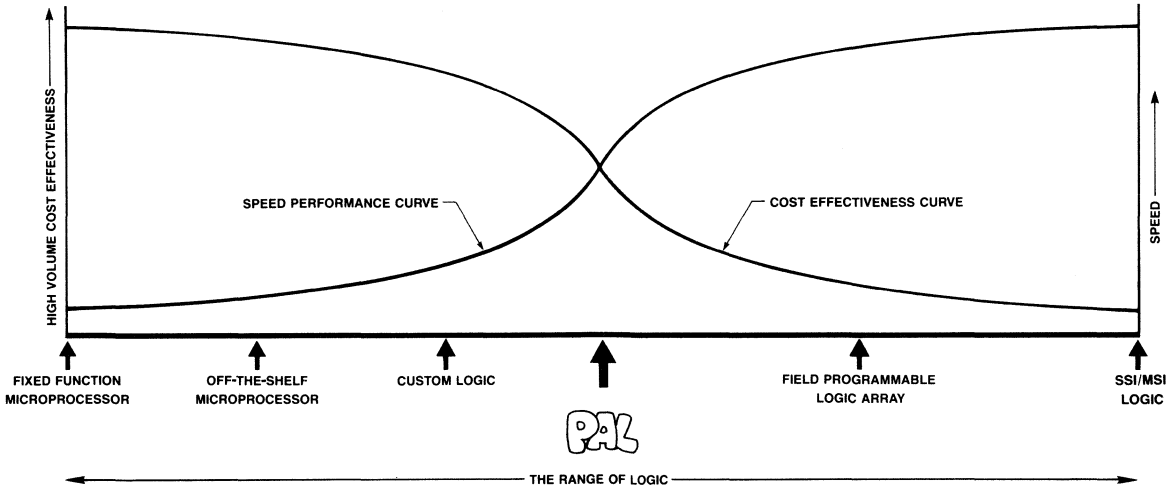
Secure Data

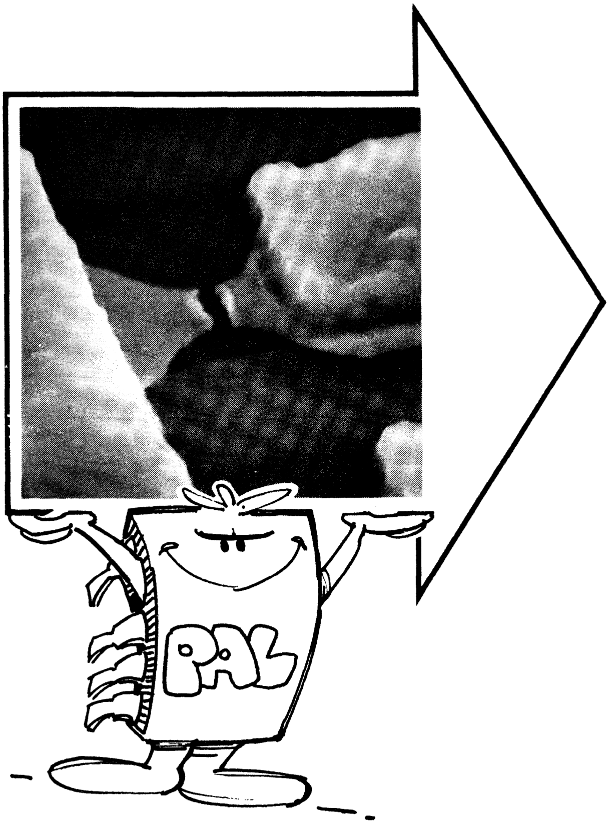


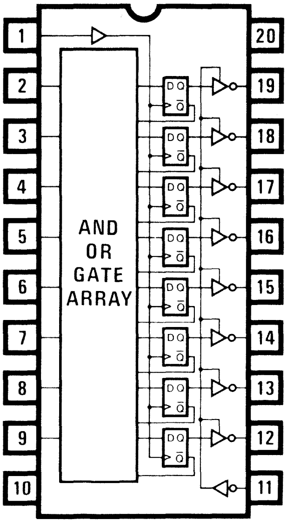
The PAL verification logic can be completely disabled by blowing out a special "last link." This prevents the unauthorized copying of valuable data, and makes the PAL perfect for use in any application where data integrity must be carefully guarded.

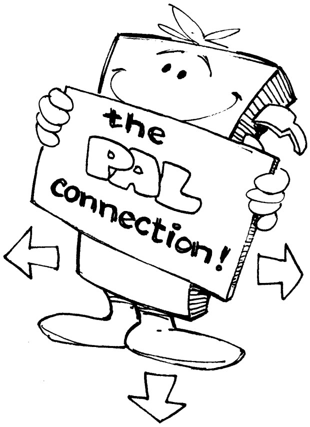
Summary

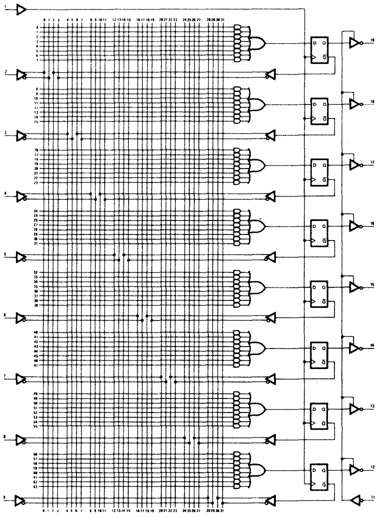
The 15 member PAL family of logic devices offers logic designers new options in the implementation of sequential and combinatorial logic designs. The family is fast, compact, flexible, and easy to use in both new and existing designs. It promises to reduce costs in most areas of design and production with a corresponding increase in product profitability.




PAL

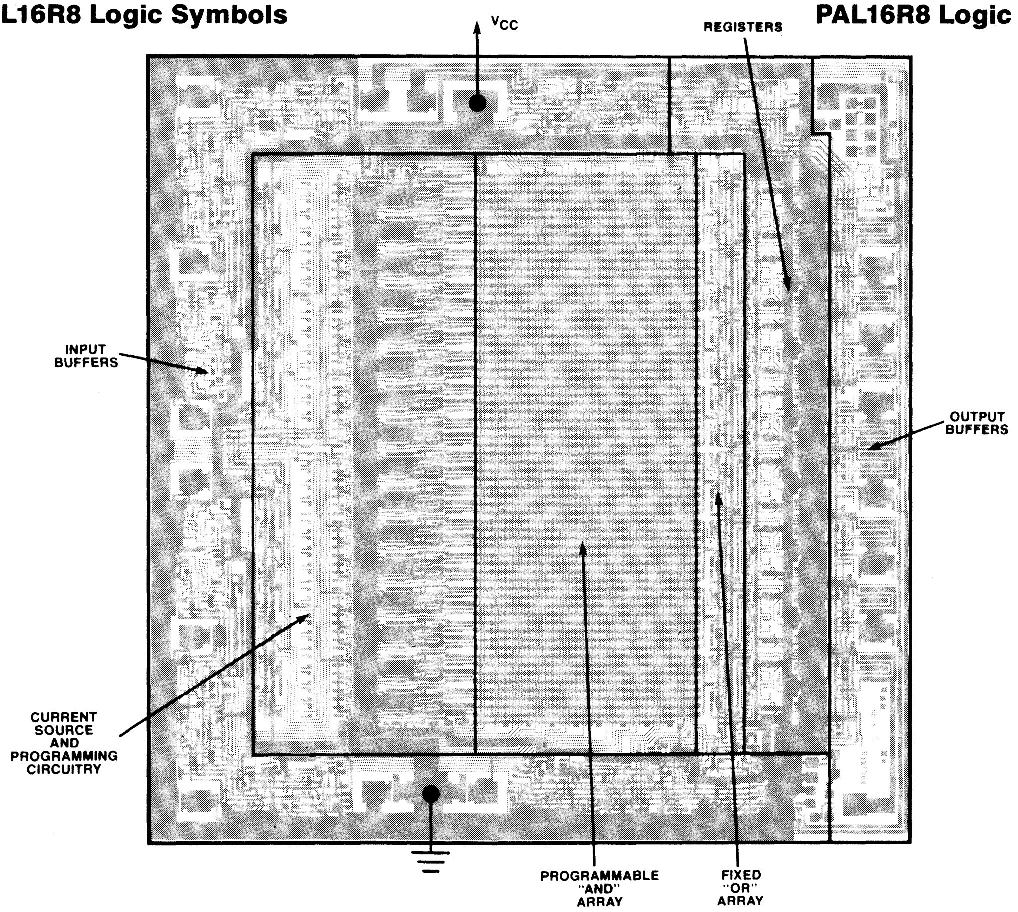






PAL16R8 Logic Symbols

PAL16R8 Logic Diagram



PAL16R8 Metalization

Figure 1

Reliability

Large-scale integrated circuits have always appealed to the system designer for the obvious reasons of lower cost and improved reliability. With a host of PROMs, RAMs and Microprocessors available today, you might expect most systems to reflect simple, small printed circuit boards populated mainly with LSI circuits. However, that is not the case. We still find a myriad of SSI and MSI packages surrounding a relatively few LSI circuits because the latter must be supported by a large number of gates, i.e., random logic, to efficiently meet a given hardware design goal. In the past many approaches have evolved from the R & D laboratory to circumvent the problem. Discretionary wiring techniques using metal mask options to delineate the interconnect pattern were the most actively pursued. Texas Instruments used this process and employed two or more levels of metalization interconnecting many die on a whole wafer to yield a custom device. Other notable attempts have Fairchild's "Micro-matrix" and most recently Hughes' "SCAT", Schottky cell array technology. All suffer from the same problems: 1. Complex two-or three-level metalization processing, 2. Custom design of new circuits and masks, 3. Small production runs with little meaningful accumulation of process and design history to optimize yield and reliability.

The latter point is the underlying factor for the failure of most custom LSI devices to live up to the reliability improvement factor expected of them over the equivalent SSI circuitry.

PALs present an equitable LSI solution to this random logic problem. Instead of metal mask options and small production runs we have a family of fifteen devices using conventional Schottky TTL processing and the fusible link technology that has made bipolar PROMs so widely accepted for over six years. On the average, the PAL accomplishes a fourfold reduction in overall pin/package count. The immediate benefits to system reliability are clear:

1. A reduction in board level complexity translates to fewer solder joints and plated through holes which are significant board-level failure points. In fact, multilayer PCB designs may be able to be implemented using the more reliable and less costly two-sided configuration.

2. Less devices on a board simplifies diagnostics resulting in a reduction of the overall handling, probing, etc. of the components. This reduces the probability of overstress through inadvertent shorting of traces or pins.

3. A reduction of packages means less wire bonds to fail. In fact, most semiconductor failures are related to packaging, i.e., hermeticity, die attach, etc. The less packages per system, the higher the system reliability.

PALs are simply programmable AND arrays feeding a fixed OR structure. See Figure 1. At the die level, the PAL circuit complexity does not exceed the 512-bit PROM on the smaller die and the 2048-bit PROM on the two larger die. Reference Table 1. In fact, in real applications, parts of the circuit are inactive or passive. On closer inspection one finds that as a maximum, only 70% of the circuitry is actually optioned at one time on a specific PAL, since the same die is used to produce PAL pinout options. The programming circuitry is only active during the fusing process. After programming it ceases to operate. Thus, the total active circuitry after programming is not more than 50% of what is on the die.

Another important aspect of the PAL that positively impacts reliability is the very low READ current in an unblown fuse, less than 0.5 mA as compared to a typical programming current of 50 mA. Since power dissipation in the fuse is a function of I^2, there are more than four orders of magnitude safety margin in actual operation. The programmable "AND" array is implemented using an emitter follower circuit with each NPN transistor in series with a fusible link. See Figure 2. The fusible link, with a typical resistance of 40Ω in the unprogrammed state, Figure 3,

DEVICE NUMBER	AND ARRAY ORGANIZATION				NUMBER OF TEST FUSES	DIE SIZE	POWER DISSIPATION (TYP)
	INPUT LINES X	T/C[1] X	PRODUCT LINES =	NUMBER OF FUSES			
PAL10H8	10	2	16	320	42	13K Mil^2	275 mW
PAL10L8	10	2	16	320	42	13K Mil^2	275 mW
PAL12H6	12	2	16	384	44	13K Mil^2	275 mW
PAL12L6	12	2	16	384	44	13K Mil^2	275 mW
PAL14H4	14	2	16	448	46	13K Mil^2	275 mW
PAL14L4	14	2	16	448	46	13K Mil^2	275 mW
PAL16H2	16	2	16	512	48	13K Mil^2	275 mW
PAL16L2	16	2	16	512	48	13K Mil^2	275 mW
PAL16C1	16	2	16	512	48	13K Mil^2	275 mW
PAL16L8	16	2	64	2048	98	18K Mil^2	700 mW
PAL16R8	16	2	64	2048	98	18K Mil^2	750 mW
PAL16R6	16	2	64	2048	98	18K Mil^2	750 mW
PAL16R4	16	2	64	2048	98	18K Mil^2	750 mW
PAL16X4	16	2	64	2048	98	19K Mil^2	800 mW
PAL16A4	16	2	74[2]	2048	98	19K Mil^2	800 mW

Note 1: True and Complement
Note 2: 10 Product Lines are Fixed

Table 1

Two AND Array Cells

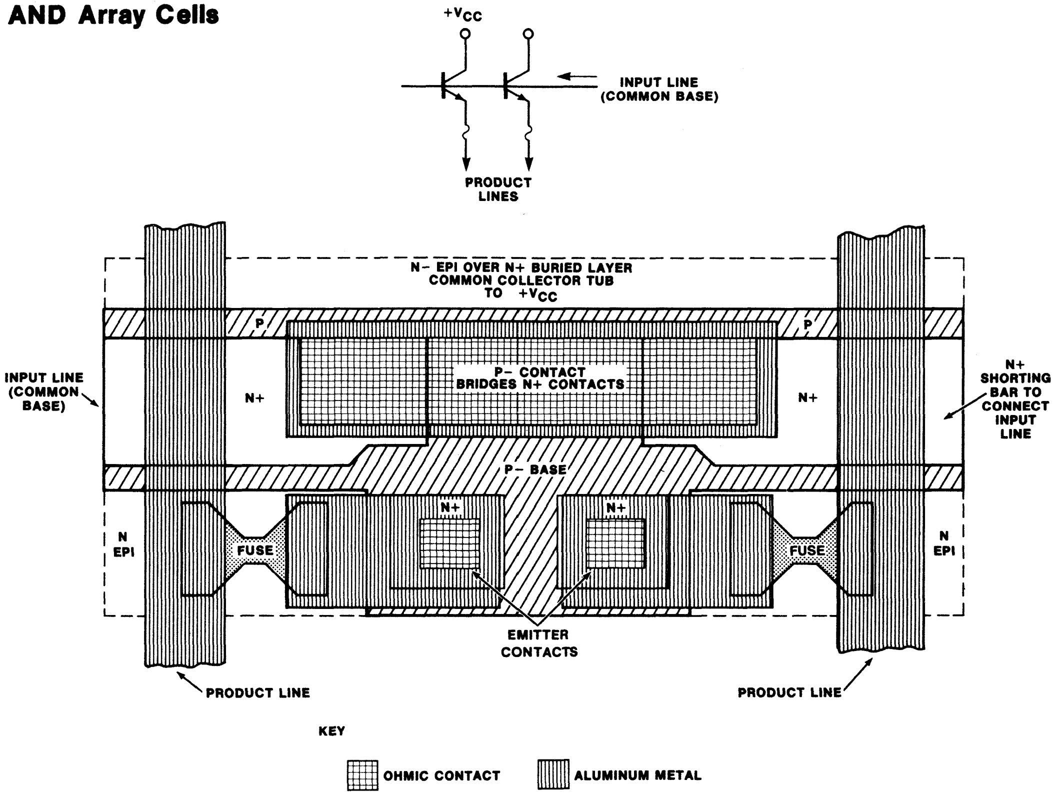


Figure 2

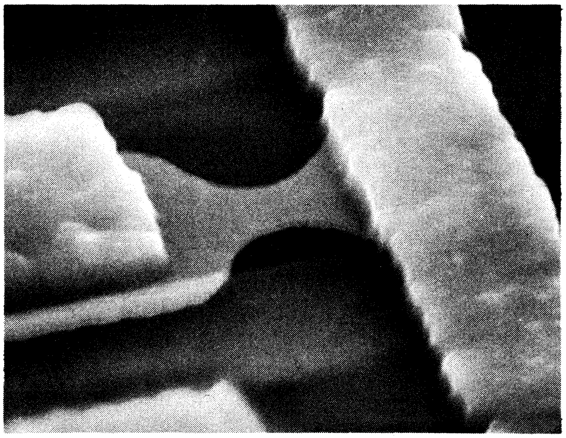


Figure 3, Unprogrammed Fuse

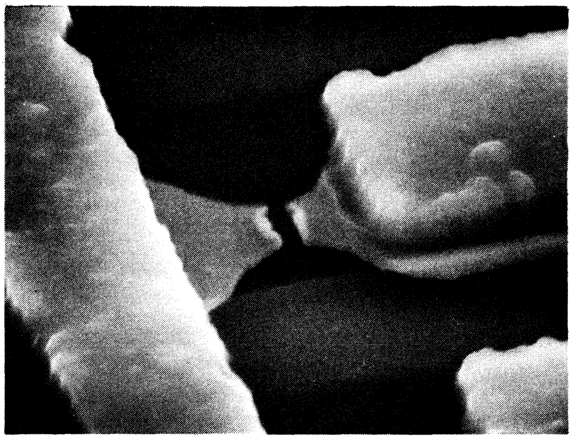
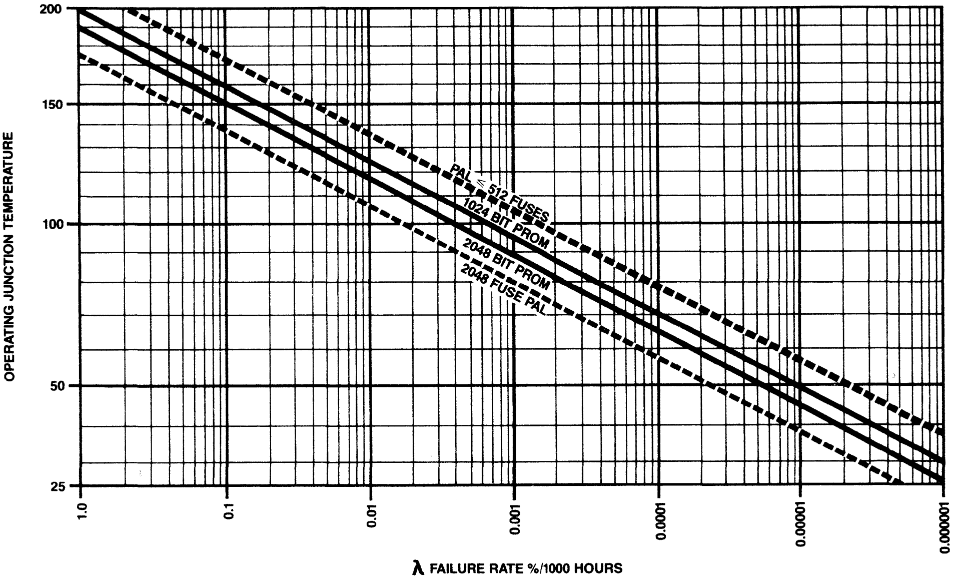


Figure 4, Programmed Fuse

becomes an open circuit after programming, Figure 4. To assure programmability, test fuses are provided along both the input line and the product line axis in the AND array (see Table 1). These are factory programmed to provide assurance that the devices will easily program in the field. Additionally, the test lines serve to check the decoding circuitry and provide a mechanism for AC testing the unprogrammed PAL.

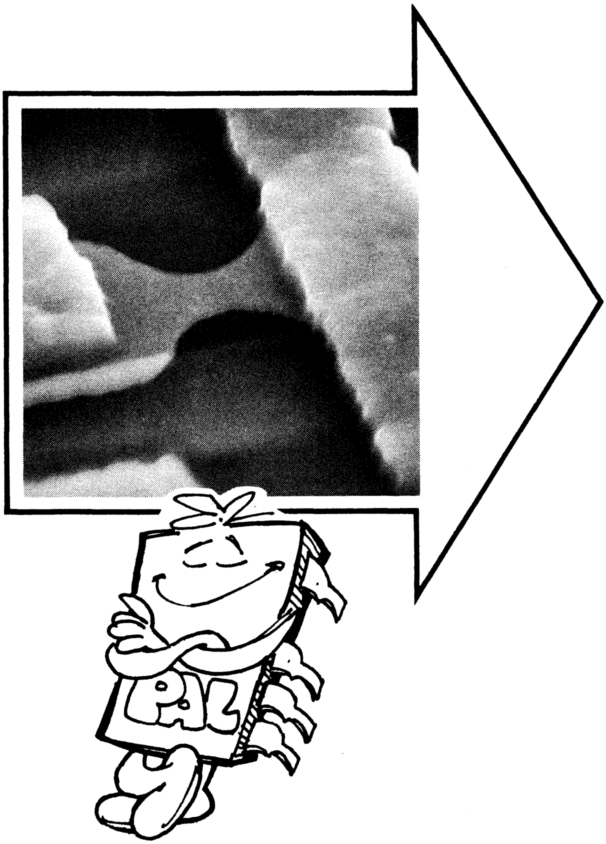
MMI produces these devices using the same reliable Schottky TTL technology and design rules as used in our PROM/ROM product line. Assembly is accomplished either in a standard 20-pin hermetic cerdip using a low temperature vitreous seal or a new plastic epoxy dip with thermal dissipation comparable to the ceramic, i.e., Θ_{ja} = 60°C/Watt. The cerdip package is specially processed to guarantee low residual moisture levels, < 500 PPM. Ultrasonic bonding using aluminum wire for cerdip and ultrasonic gold bonding for plastic both use a thicker, 1.25 mil diameter wire for increased reliability. Screening and testing follow MIL-STD-883.

Empirical data being collected on 1024 and 2048 bit PROMs that were manufactured two or more years ago, and representing actual field use in rugged operating environments support a .01% per thousand hour failure rate. Assuming progress on the design/process learning curve, it is fair to say that some improvements can be expected on present generation PROMs. Since PALs actually are smaller than these PROMs, we expect even better results. Extensive life testing of PALs is an integral part of Monolithic Memories' continuing program of product reliability evaluation. All MMI's circuits, including PALs, are manufactured in the same plant that has received DESC facility certification to MIL-M-38510. Projected failure rate data for PALs are described in Figure 5.



1. Curves based on 1eV activation energy
2. Prom data based on 1.2 million Device hours @ 125°C T_A (T_J = 160°) and 5.62 million device hours field data @ 115°C T_C (T_J = 145°C)
3. PAL curves estimated based on circuit and fuse count equivalancy; refer to Table 1.

Figure 5. PAL Projected Failure Rate Data


PAL

Programmable Array Logic Family

PAL Series 20 Data Sheet

Patent Allowed

Features/Benefits

- **Programmable replacement for conventional TTL logic.**
- **Reduces IC inventories substantially and simplifies their control.**
- **Reduces chip count by 4 to 1.**
- **Expedites and simplifies prototyping and board layout.**
- **Saves space with 20-pin Skinny DIP packages.**
- **High speed: 25ns typical propagation delay.**
- **Programmed on standard PROM programmers.**
- **Programmable three-state outputs.**
- **Special feature reduces possibility of copying by competitors.**

Description

The PAL family utilizes an advanced Schottky TTL process and the Bipolar PROM fusible link technology to provide user programmable logic for replacing conventional SSI/MSI gates and flip-flops at reduced chip count.

The family lets the systems engineer "design his own chip" by blowing fusible links to configure AND and OR gates to perform his desired logic function. Complex interconnections which previously required time-consuming layout are thus "lifted" from PC board etch and placed on silicon where they can be easily modified during prototype check-out or production.

The PAL transfer function is the familiar sum of products. Like the PROM, the PAL has a single array of fusible links. Unlike the PROM, the PAL is a programmable AND array driving a fixed OR array (the PROM is a fixed AND array driving a programmable OR array). In addition the PAL provides these options:

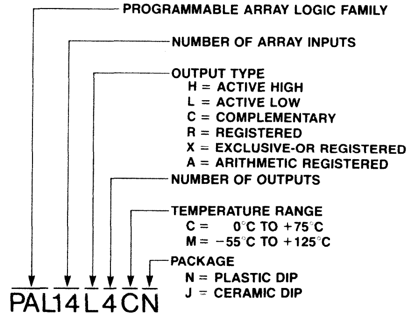
- Variable input/output pin ratio
- Programmable three-state outputs
- Registers with feedback
- Arithmetic capability

Unused inputs are tied directly to V_{CC} or GND. Product terms with all fuses blown assume the logical high state, and product terms connected to both true and complement of any single input assume the logical low state. Registers consist of D type flip-flops which are loaded on the low to high transition of the clock. All registers are designed to power up to logical high state at the output pin. PAL Logic Diagrams are shown with all fuses blown, enabling the designer use of the diagrams as coding sheets. 8½ x 11 Logic Diagrams are available on request.

The entire PAL family is programmed on inexpensive conventional PROM programmers with appropriate personality and socket adapter cards. Once the PAL is programmed and verified, two additional fuses may be blown to defeat verification. This feature gives the user a proprietary circuit which is very difficult to copy.

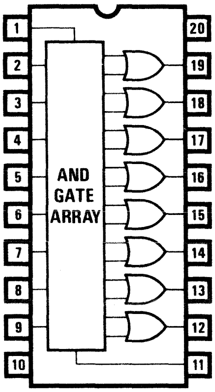
PART NUMBER	DESCRIPTION
PAL10H8	OCTAL 10 INPUT AND-OR GATE ARRAY
PAL12H6	HEX 12 INPUT AND-OR GATE ARRAY
PAL14H4	QUAD 14 INPUT AND-OR GATE ARRAY
PAL16H2	DUAL 16 INPUT AND-OR GATE ARRAY
PAL16C1	16 INPUT AND-OR/AND-OR-INVERT GATE ARRAY
PAL10L8	OCTAL 10 INPUT AND-OR-INVERT GATE ARRAY
PAL12L6	HEX 12 INPUT AND-OR-INVERT GATE ARRAY
PAL14L4	QUAD 14 INPUT AND-OR-INVERT GATE ARRAY
PAL16L2	DUAL 16 INPUT AND-OR-INVERT GATE ARRAY
PAL16L8	OCTAL 16 INPUT AND-OR-INVERT GATE ARRAY
PAL16R8	OCTAL 16 INPUT REGISTERED AND-OR GATE ARRAY
PAL16R6	HEX 16 INPUT REGISTERED AND-OR GATE ARRAY
PAL16R4	QUAD 16 INPUT REGISTERED AND-OR GATE ARRAY
PAL16X4	QUAD 16 INPUT REGISTERED AND-OR-XOR GATE ARRAY
PAL16A4	QUAD 16 INPUT REGISTERED AND-CARRY-OR-XOR GATE ARRAY

Ordering Information

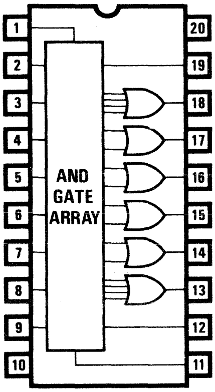


PAL Logic Symbols

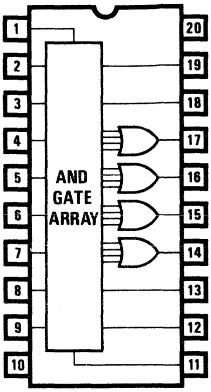
PAL10H8



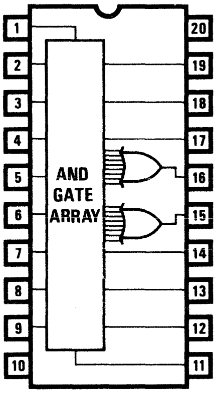
PAL12H6



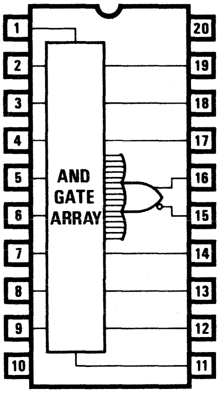
PAL14H4



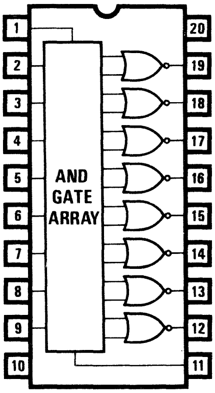
PAL16H2



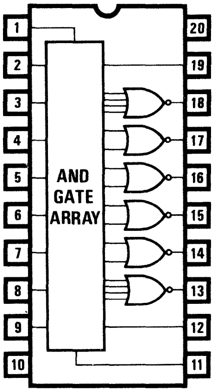
PAL16C1



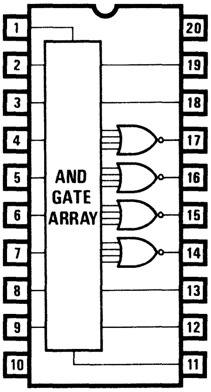
PAL10L8



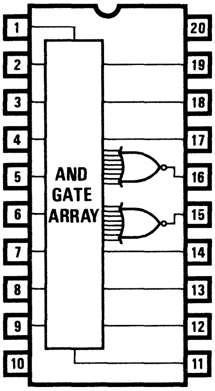
PAL12L6



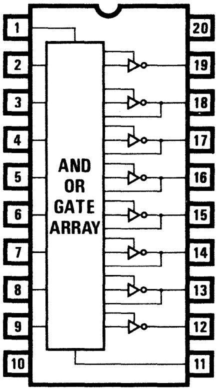
PAL14L4



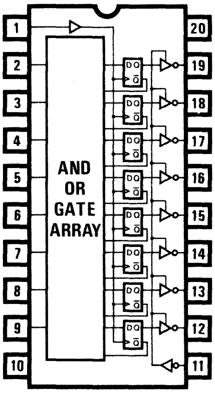
PAL16L2



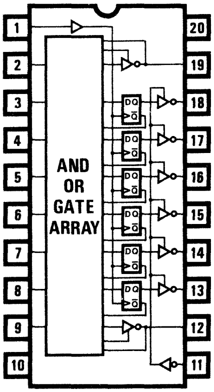
PAL16L8



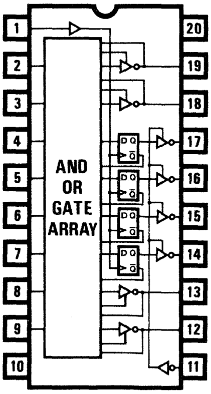
PAL16R8



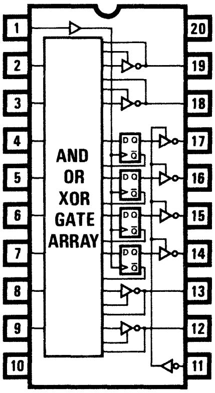
PAL16R6



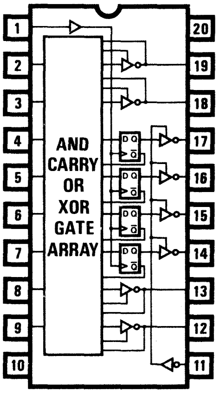
PAL16R4



PAL16X4



PAL16A4



Absolute Maximum Ratings

	Operating	Programming
Supply voltage V_{CC}	7V	12V
Input voltage	5.5V	12V
Off-state output voltage	5.5V	12V
Storage temperature range	−65°C to 150°C	

10H8, 12H6, 14H4, 16H2, 16C1, 10L8, 12L6, 14L4, 16L2

Recommended Operating Conditions

SYMBOL	PARAMETER	MILITARY			COMMERCIAL			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
V_{CC}	Supply voltage	4.5	5.0	5.5	4.75	5.00	5.25	V
I_{OH}	High-level output current			−2.0			−3.2	mA
I_{OL}	Low-level output current			8			8	mA
T_A	Operating free air temperature	−55		125*	0		75	°C

Electrical Characteristics

Over Recommended Operating Temperature Range

SYMBOL	PARAMETER	TEST CONDITIONS	MIN	TYP	MAX	UNIT
V_{IH}	High-level input voltage		2			V
V_{IL}	Low-level input voltage				0.8	V
V_{IC}	Input clamp voltage	V_{CC} = MIN, I_I = −18 mA			−1.5	V
V_{OH}	High-level output voltage	V_{CC} = MIN, V_{IH} = 2V, V_{IL} = 0.8V, I_{OH} = MAX	2.4			V
V_{OL}	Low-level output voltage	V_{CC} = MIN, V_{IH} = 2V, V_{IL} = 0.8V, I_{OL} = MAX			0.5	V
I_I	Input current at maximum input voltage	V_{CC} = MAX, V_I = 5.5V			1.0	mA
I_{IH}	High-level input current	V_{CC} = MAX, V_I = 2.4V			25	μA
I_{IL}	Low-level input current	V_{CC} = MAX, V_I = 0.4V			−250	μA
I_{OS}	Short-circuit output current	V_{CC} = MAX	−30		−130	mA
I_{CC}	Supply current	V_{CC} = MAX		55		mA

Switching Characteristics

Over Recommended Ranges of Temperature and V_{CC}

SYMBOL	PARAMETER	TEST CONDITIONS†† R_L = 2.0 K Ω	MILITARY* T_A = −55° to +125°C V_{CC} = 5.0V ± 10%			COMMERCIAL T_A = 0° to 75°C V_{CC} = 5.0V ± 5%			UNIT
			MIN	TYP	MAX	MIN	TYP	MAX	
t_{PD}	From any input to any output	C_L = 15pF		25			25		ns

* Operating Case Temperature only. T_C = 125°C

†† See Standard Test Load and Definition of Waveforms, page 3-24

16L8, 16R8, 16R6, 16R4, 16X4▼, 16A4▼

Recommended Operating Conditions

SYMBOL	PARAMETER	MILITARY			COMMERCIAL			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
V_{CC}	Supply voltage	4.5	5.0	5.5	4.75	5.00	5.25	V
I_{OH}	High-level output current			−2.0			−3.2	mA
I_{OL}	Low-level output current			12			24	mA
T_A	Operating free air temperature	−55		125*	0		75	°C

Electrical Characteristics

Over Recommended Operating Temperature Range

SYMBOL	PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
V_{IH}	High-level input voltage			2			V
V_{IL}	Low-level input voltage					0.8	V
V_{IC}	Input clamp voltage		V_{CC} = MIN, $1_I = -18mA$			−1.5	V
V_{OH}	High-level output voltage		V_{CC} = MIN, V_{IH} = 2V, V_{IL} = 0.8V, I_{OH} = MAX	2.4			V
V_{OL}	Low-level output voltage		V_{CC} = MIN, V_{IH}=2V, V_{IL} = 0.8V, I_{OL}=MAX			0.5	V
I_{OZH}	Off-state output current high-level voltage applied		V_{CC} = MAX, V_{IH} = 2V, V_O = 2.4V, VIL = 0.8V			100	μA
I_{OZL}	Off-state output current low-level voltage applied		V_{CC} = MAX, V_{IH} = 2V, V_O = 0.4V, VIL = 0.8V			–100	μA
I_I	Input current at maximum input voltage		V_{CC} = MAX, V_I = 5.5V			1.0	mA
I_{IH}	High-level input current		V_{CC} = MAX, V_I = 2.4V			25	μA
I_{IL}	Low-level input current		V_{CC} = MAX, V_I = 0.4V			−250	μA
I_{OS}	Short-circuit output current		V_{CC} = MAX,	−30		−130	mA
I_{CC}	Supply Current	16L8	V_{CC} = MAX		140	210†	mA
		16R4,16R6,16R8			150	225†	
		16X4,16A4			160		

Switching Characteristics

Over Recommended Ranges of Temperature and V_{CC}

SYMBOL	PARAMETER		TEST CONDITIONS†† R_L = 667 Ω	MILITARY T_A = −55° to +125°C* V_{CC} = 5.0V ± 10%			COMMERCIAL T_A = −0° to +75°C V_{CC} = 5.0V ± 5%			UNIT
				MIN	TYP	MAX	MIN	TYP	MAX	
t_{PD}	Input to output		C_L = 45pF		25	45		25	40	ns
t_{PD}	Clock to output				15	25		15	25	ns
t_{PZX}	Pin 11 to output enable				15	25		15	25	ns
t_{PXZ}	Pin 11 to output disable		CL = 5pF		15	25		15	25	ns
t_{PZX}	Input to output enable		C_L = 45pF		25	45		25	40	ns
t_{PXZ}	Input to output disable		C_L = 5pF		25	45		25	40	ns
t_w	Width of clock	High		25			25			ns
		Low		25			25			
t_{su}	Setup time	16R8,16R6,16R4		45			40			ns
		16X4, 16A4								
t_h	Hold time			0	−15		0	−15		ns

*Operating Case Temperature only, T_C = 125°C

†I_{CC} = MAX at minimum temperature

†† See Standard Test Load and Definition of Waveforms, page 3-24

▼PRELIMINARY

Programming

PAL fuses are programmed using a low-voltage linear-select procedure which is common to all 15 PAL types. The array is divided into two groups, products 0 thru 31 and products 32 thru 63, for which pin identifications are shown in Figure 1. To program a particular fuse, both an input line and a product line are selected according to the following procedure:

Step 1 Raise Output Disable, OD, to V_{IHH}

Step 2 Select an input line by specifying I_0, I_1, I_2, I_3, I_4, I_5, I_6, I_7 and L/R as shown in Table 1

Step 3 Select a product line by specifying A_0, A_1 and A_2 one-of-eight select as shown in Table 2

Step 4 Raise V_{CC} (pin 20) to V_{IHH}

Step 5 Program the fuse by pulsing the output pins, O, of the selected product group to V_{IHH} as shown in Table 2.

Step 6 Lower V_{CC} (pin 20) to 6.0 V

Step 7 Pulse the CLOCK pin and verify the output pin, O, to be Low for active Low PAL types or High for active High PAL types.

Step 8 Lower V_{CC} (pin 20) to 4.2 V and repeat step 7

Step 9 Should the output not verify, repeat steps 1 thru 8 up to five (5) times.

This procedure is repeated for all fuses to be blown (see programming waveforms).

To prevent further verification, two last fuses may be blown by raising pin I and pin II to V_P. V_{CC} is not required during this operation.

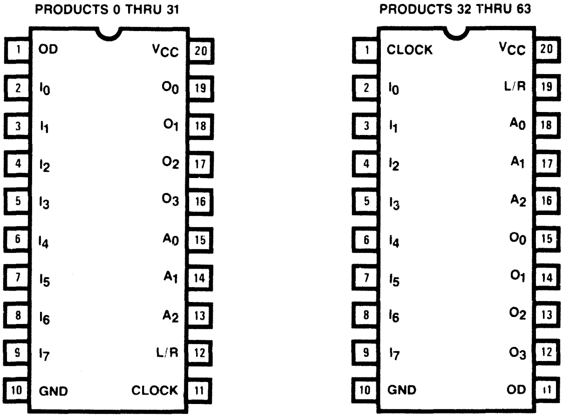
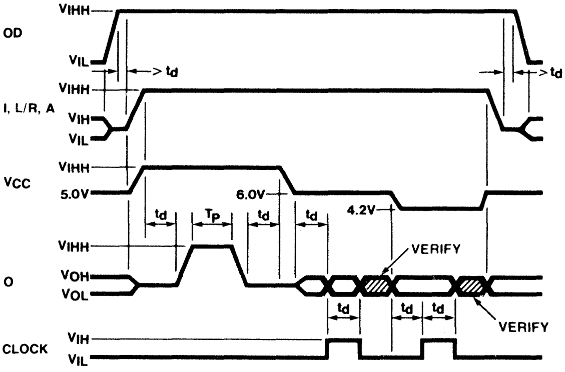


Figure 1 Pin Identification

Programming Waveforms



Programming Parameters

T_A = 25°C

SYMBOL	PARAMETER		LIMITS			UNIT
			MIN	TYP	MAX	
V_{IHH}	Program-level input voltage		10.5	11	11.5	V
I_{IHH}	Program-level input current	Output Program Pulse			50	mA
		Output Disable, OD			25	
		All Other Inputs			5	
I_{CCH}	Program Supply Current				400	mA
T_P	Program Pulse Width		10		50	µs
t_d	Delay time		100			ns
	Program Pulse duty cycle				25	%
V_P	Program/Verify-Protect-input voltage			20		V
I_P	Program/Verify-Protect-input current				400	mA

Voltage Legend

L = Low-level input voltage, V_{IL}
H = High-level input voltage, V_{IH}
HH = High-level program voltage, V_{IHH}

INPUT LINE NUMBER	PIN IDENTIFICATION								
	I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0	L/R
0	HH	HH	HH	HH	HH	HH	HH	L	L
1	HH	HH	HH	HH	HH	HH	HH	H	L
2	HH	HH	HH	HH	HH	HH	HH	L	HH
3	HH	HH	HH	HH	HH	HH	HH	H	HH
4	HH	HH	HH	HH	HH	HH	L	HH	L
5	HH	HH	HH	HH	HH	HH	H	HH	L
6	HH	HH	HH	HH	HH	HH	L	HH	HH
7	HH	HH	HH	HH	HH	HH	H	HH	HH
8	HH	HH	HH	HH	HH	L	HH	HH	L
9	HH	HH	HH	HH	HH	H	HH	HH	L
10	HH	HH	HH	HH	HH	L	HH	HH	HH
11	HH	HH	HH	HH	HH	H	HH	HH	HH
12	HH	HH	HH	HH	L	HH	HH	HH	L
13	HH	HH	HH	HH	H	HH	HH	HH	L
14	HH	HH	HH	HH	L	HH	HH	HH	HH
15	HH	HH	HH	HH	H	HH	HH	HH	HH
16	HH	HH	HH	L	HH	HH	HH	HH	L
17	HH	HH	HH	H	HH	HH	HH	HH	L
18	HH	HH	HH	L	HH	HH	HH	HH	HH
19	HH	HH	HH	H	HH	HH	HH	HH	HH
20	HH	HH	L	HH	HH	HH	HH	HH	L
21	HH	HH	H	HH	HH	HH	HH	HH	L
22	HH	HH	L	HH	HH	HH	HH	HH	HH
23	HH	HH	H	HH	HH	HH	HH	HH	HH
24	HH	L	HH	HH	HH	HH	HH	HH	L
25	HH	H	HH	HH	HH	HH	HH	HH	L
26	HH	L	HH	HH	HH	HH	HH	HH	HH
27	HH	H	HH	HH	HH	HH	HH	HH	HH
28	L	HH	HH	HH	HH	HH	HH	HH	L
29	H	HH	HH	HH	HH	HH	HH	HH	L
30	L	HH	HH	HH	HH	HH	HH	HH	HH
31	H	HH	HH	HH	HH	HH	HH	HH	HH

Table 1 Input Line Select

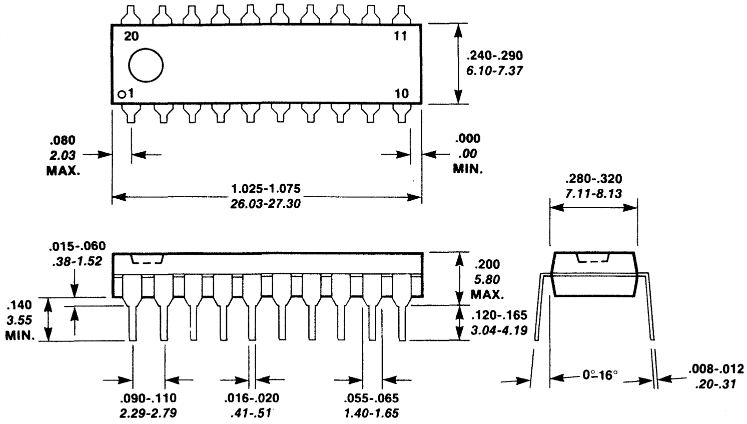
PRODUCT LINE NUMBER	PIN IDENTIFICATION						
	O_3	O_2	O_1	O_0	A_2	A_1	A_0
0, 32	L	L	L	HH	L	L	L
1, 33	L	L	L	HH	L	L	HH
2, 34	L	L	L	HH	L	HH	L
3, 35	L	L	L	HH	L	HH	HH
4, 36	L	L	L	HH	HH	L	L
5, 37	L	L	L	HH	HH	L	HH
6, 38	L	L	L	HH	HH	HH	L
7, 39	L	L	L	HH	HH	HH	HH
8, 40	L	L	HH	L	L	L	L
9, 41	L	L	HH	L	L	L	HH
10, 42	L	L	HH	L	L	HH	L
11, 43	L	L	HH	L	L	HH	HH
12, 44	L	L	HH	L	HH	L	L
13, 45	L	L	HH	L	HH	L	HH
14, 46	L	L	HH	L	HH	HH	L
15, 47	L	L	HH	L	HH	HH	HH
16, 48	L	HH	L	L	L	L	L
17, 49	L	HH	L	L	L	L	HH
18, 50	L	HH	L	L	L	HH	L
19, 51	L	HH	L	L	L	HH	HH
20, 52	L	HH	L	L	HH	L	L
21, 53	L	HH	L	L	HH	L	HH
22, 54	L	HH	L	L	HH	HH	L
23, 55	L	HH	L	L	HH	HH	HH
24, 56	HH	L	L	L	L	L	L
25, 57	HH	L	L	L	L	L	HH
26, 58	HH	L	L	L	L	HH	L
27, 59	HH	L	L	L	L	HH	HH
28, 60	HH	L	L	L	HH	L	L
29, 61	HH	L	L	L	HH	L	HH
30, 62	HH	L	L	L	HH	HH	L
31, 63	HH	L	L	L	HH	HH	HH

Table 2 Product Line Select

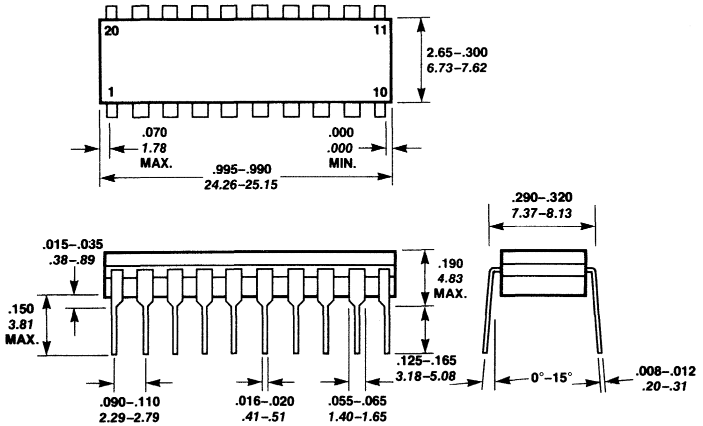
Package Drawings

UNLESS OTHERWISE SPECIFIED:
ALL DIMENSIONS MIN.-MAX. IN INCHES.
ALL DIMENSIONS MIN.-MAX. IN MILLIMETERS.

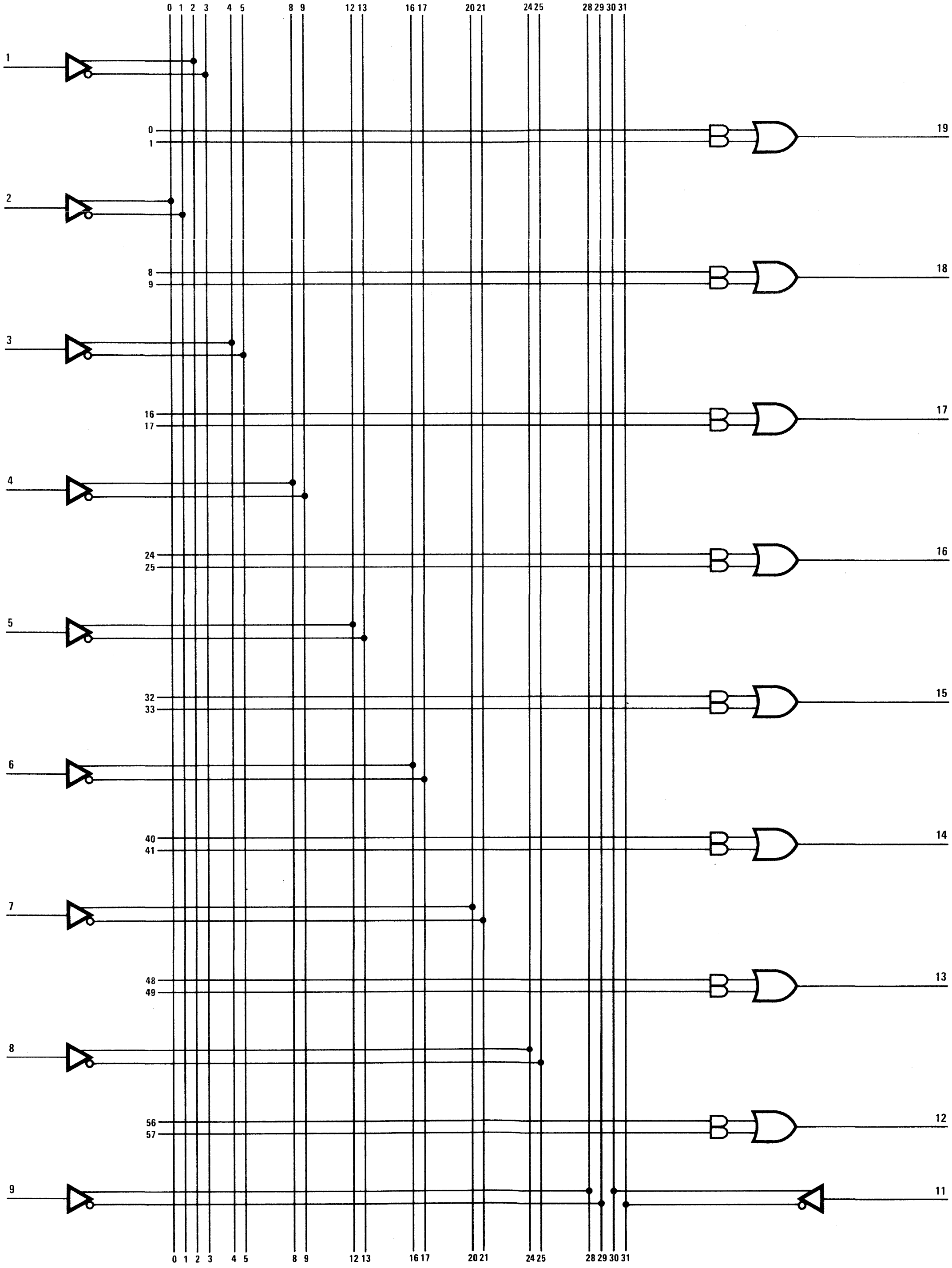
N20 Plastic Dip



J20 Ceramic Dip

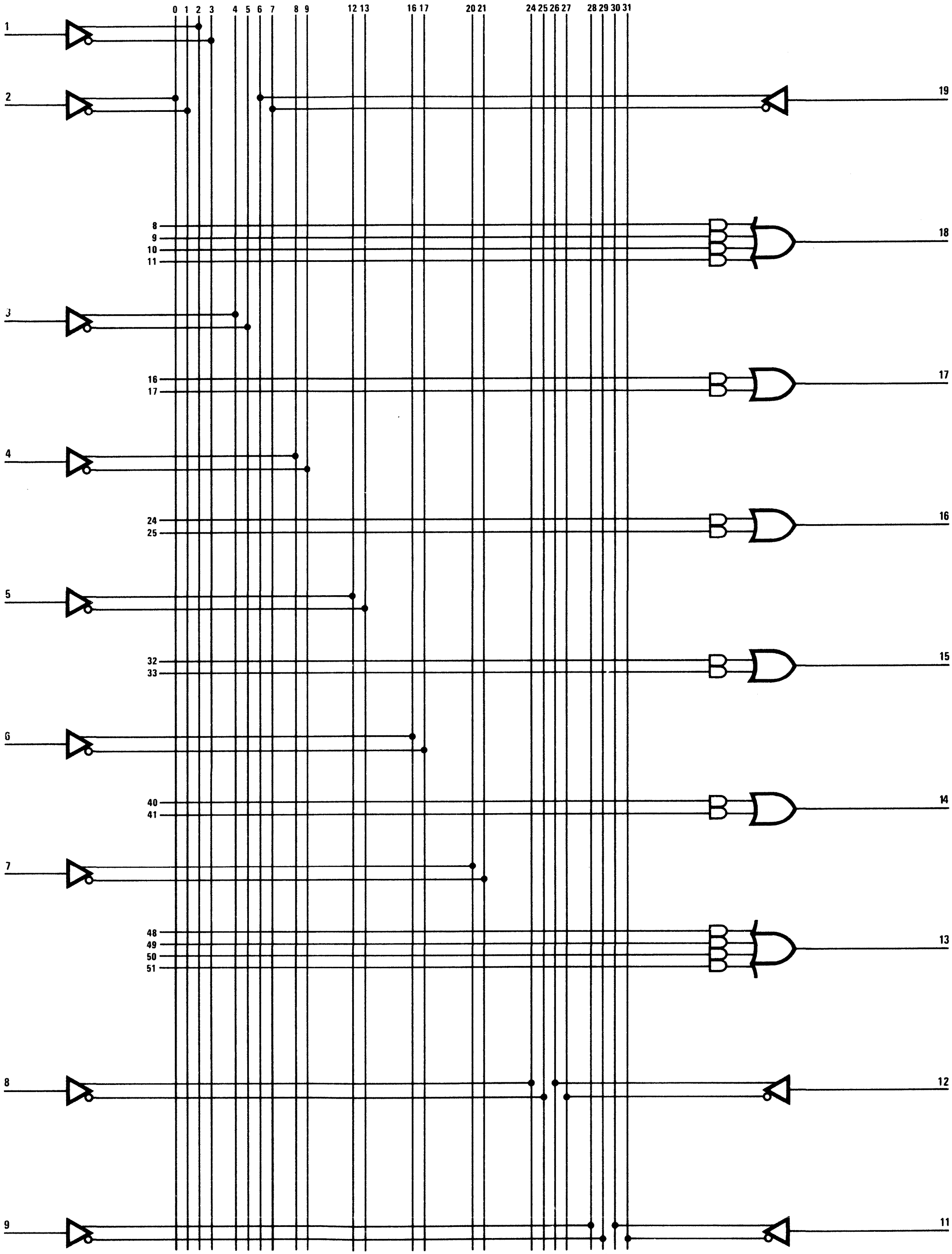


Logic Diagram PAL10H8

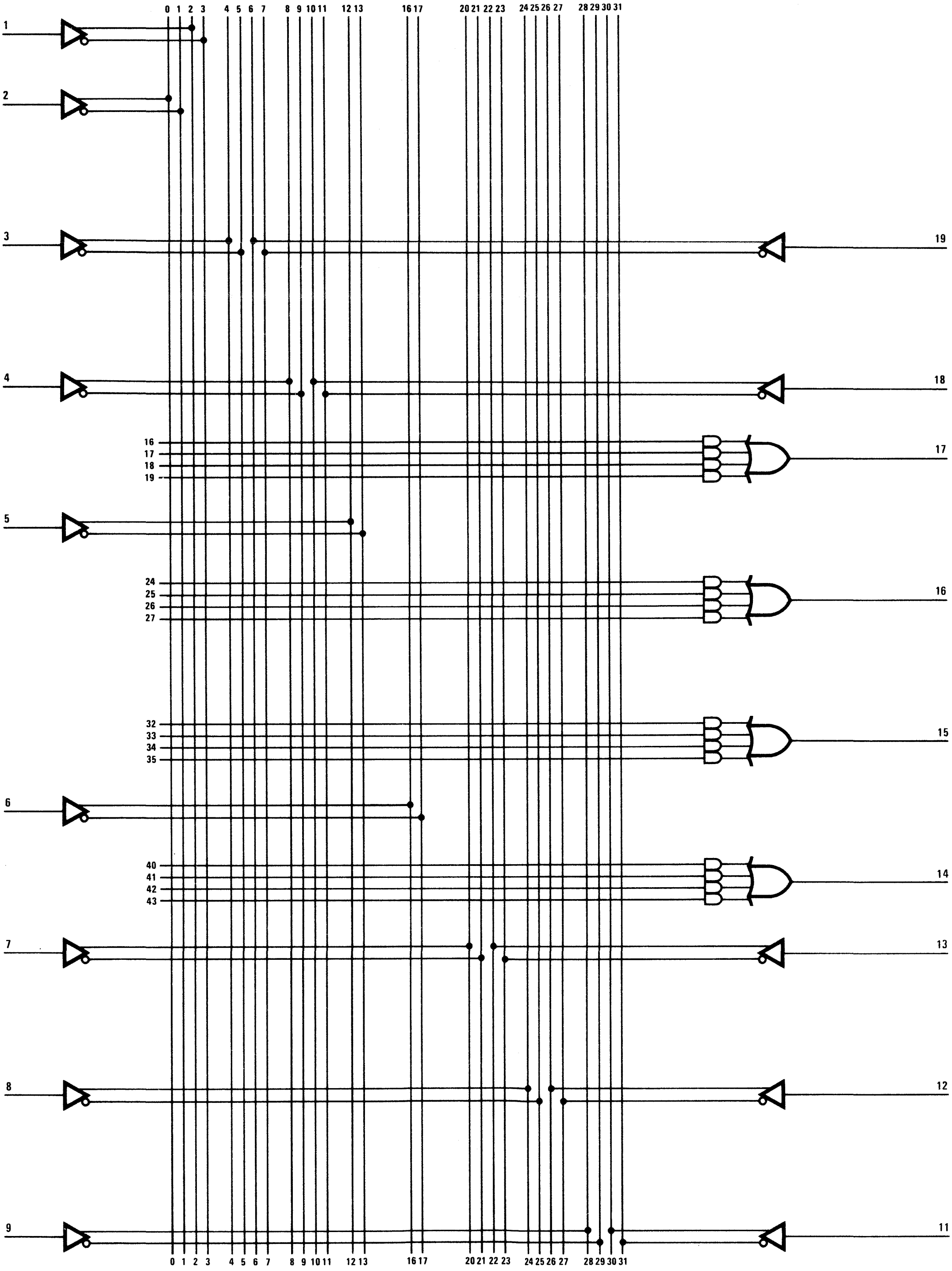


3

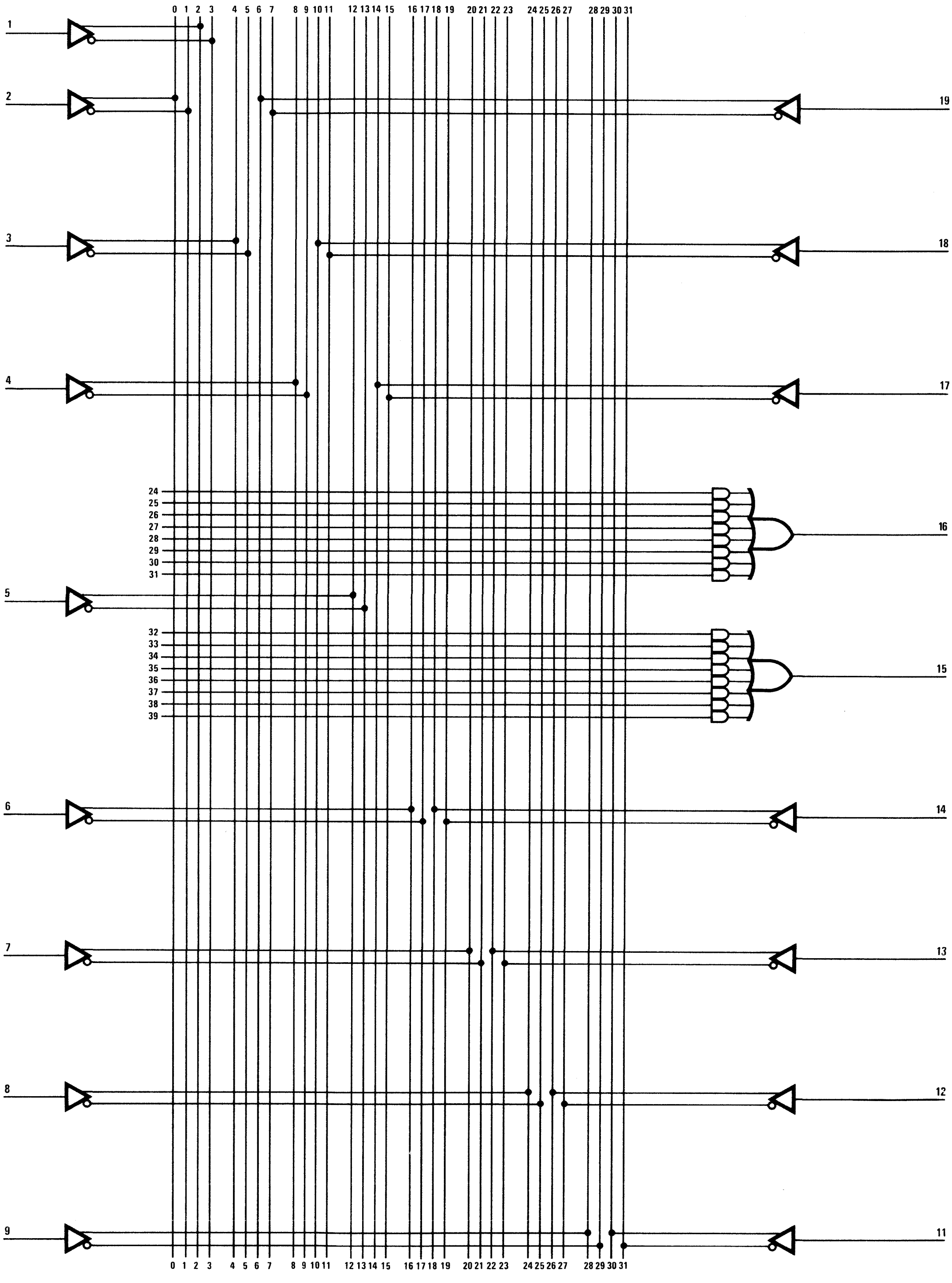
Logic Diagram PAL12H6



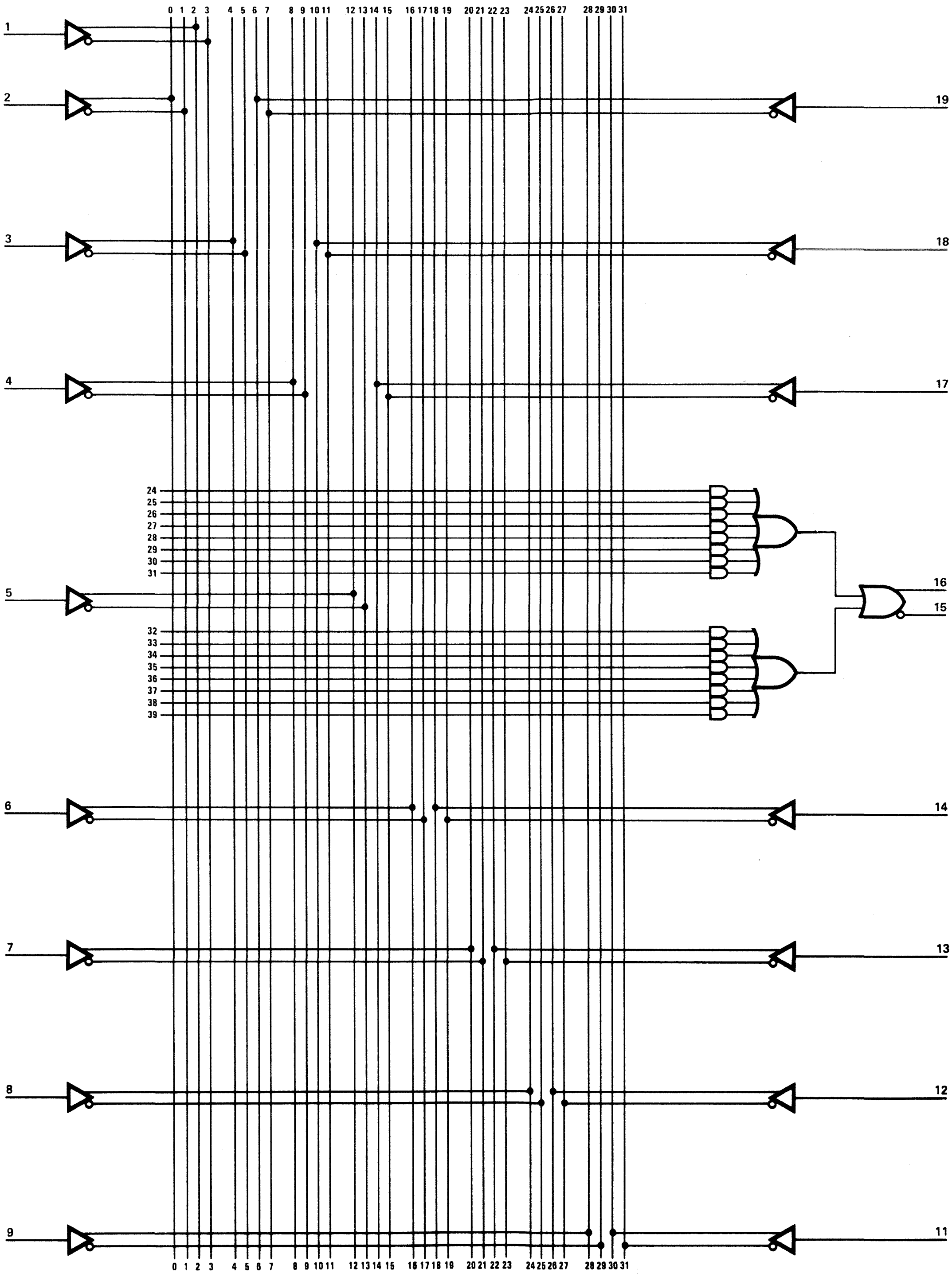
Logic Diagram PAL14H4



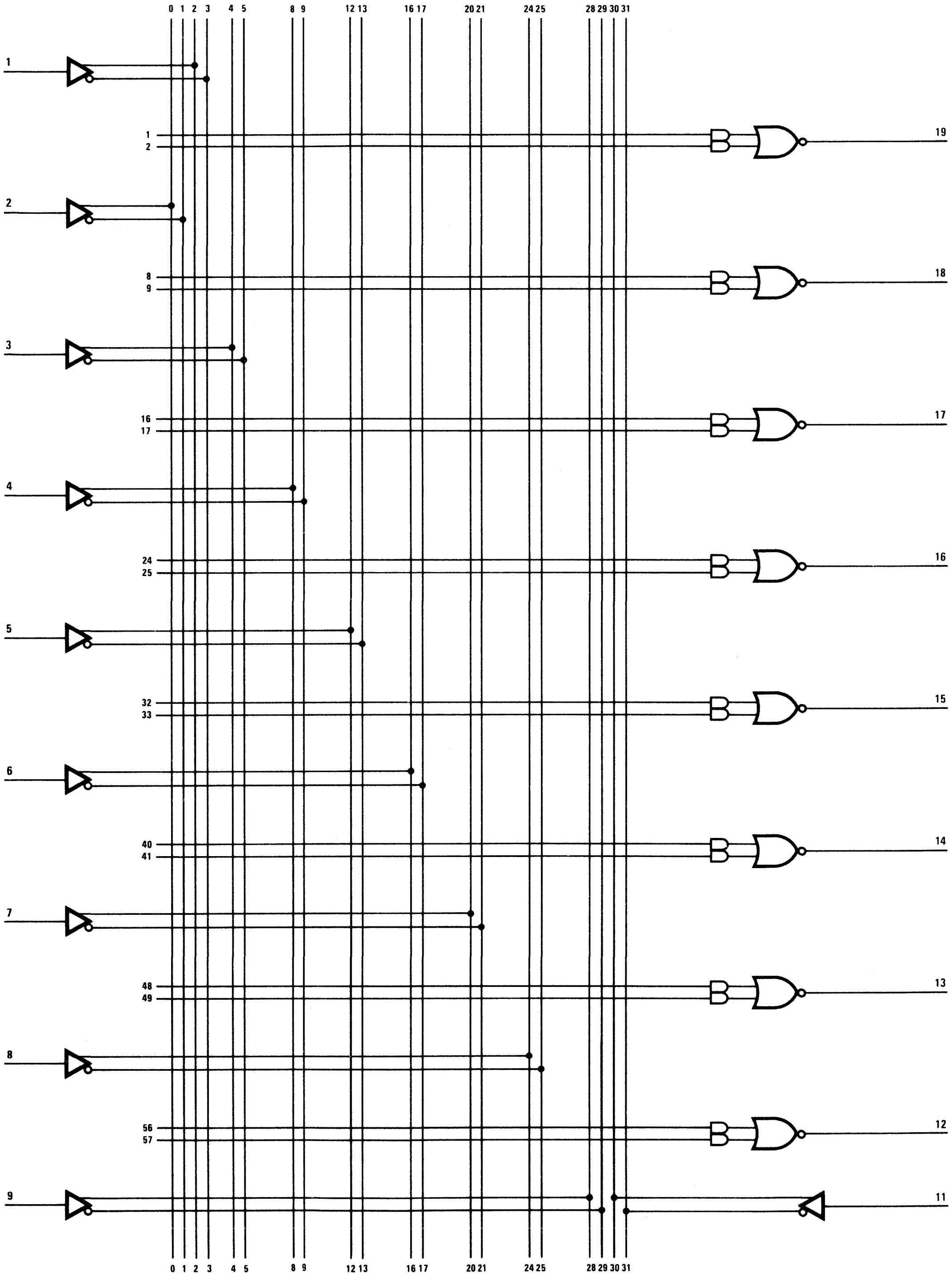
Logic Diagram PAL16H2



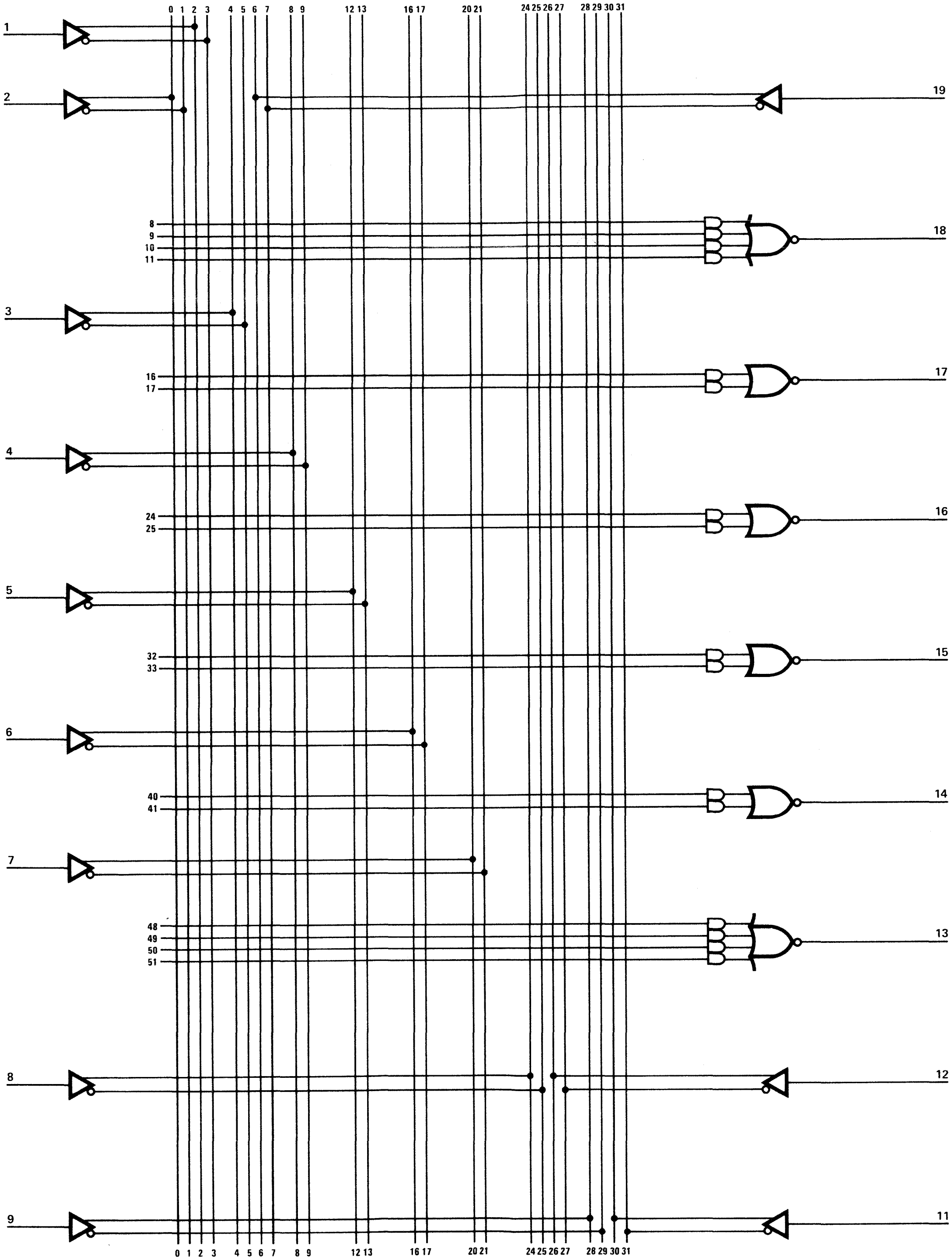
Logic Diagram PAL16C1



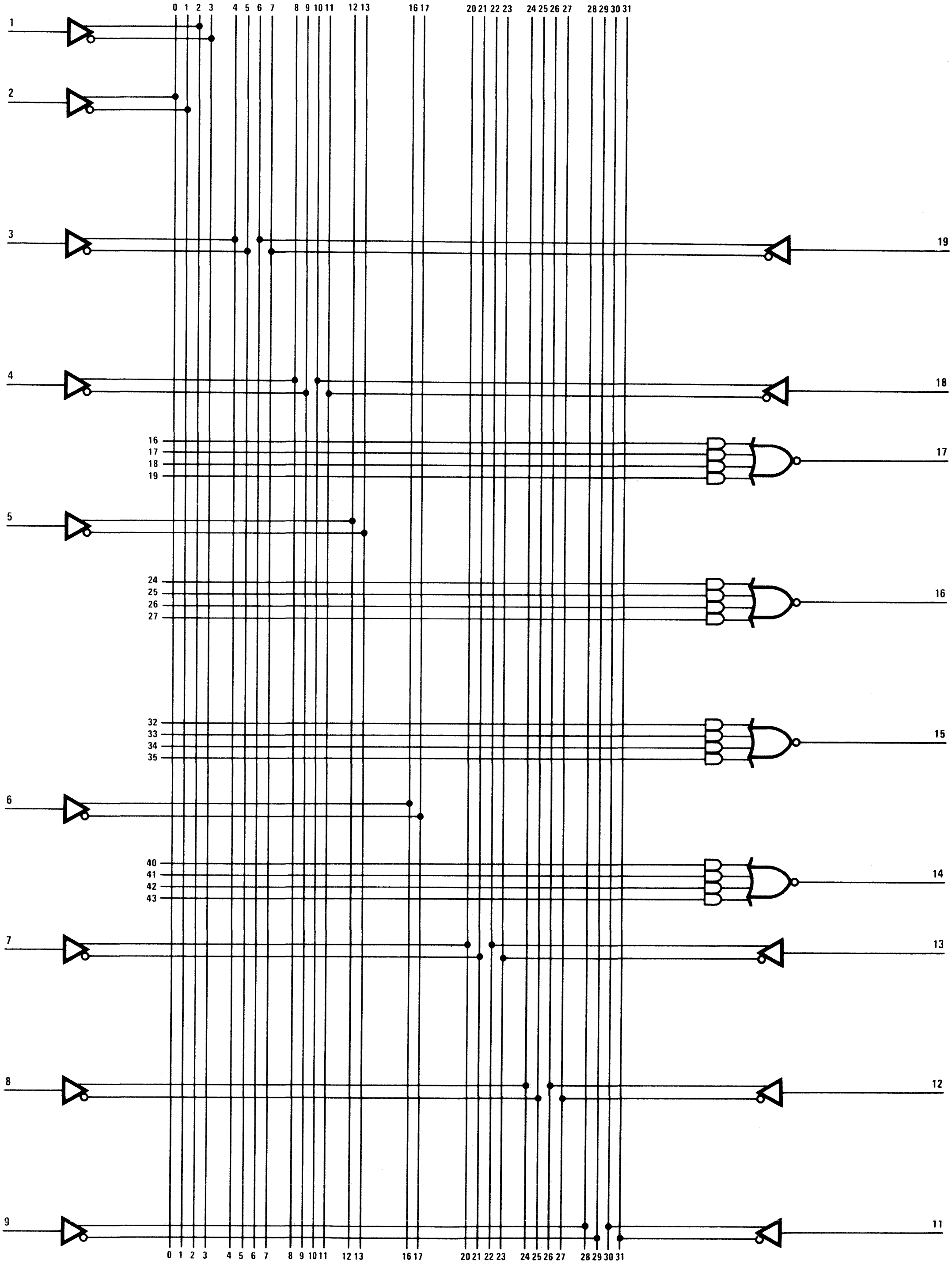
Logic Diagram PAL10L8



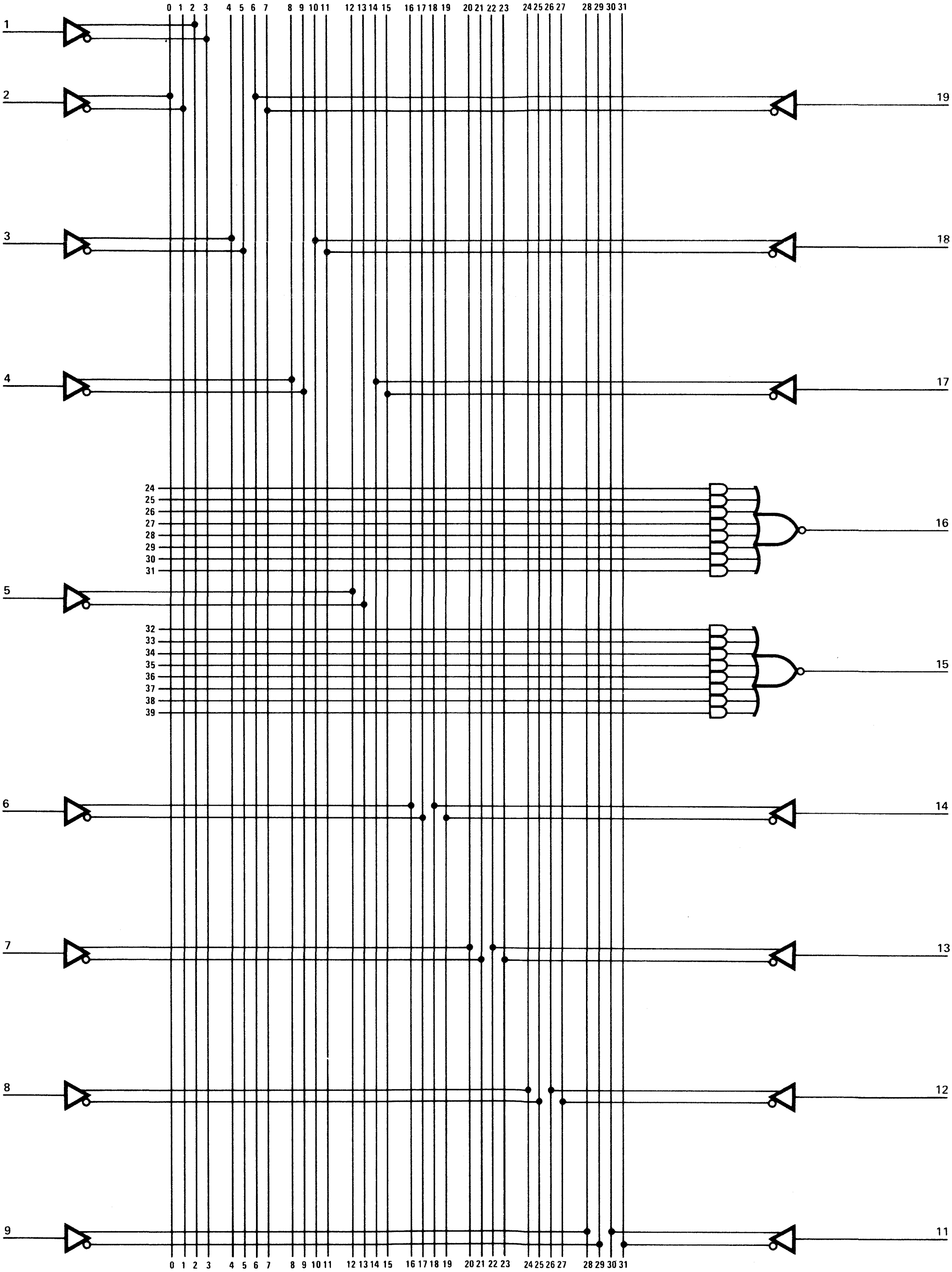
Logic Diagram PAL12L6



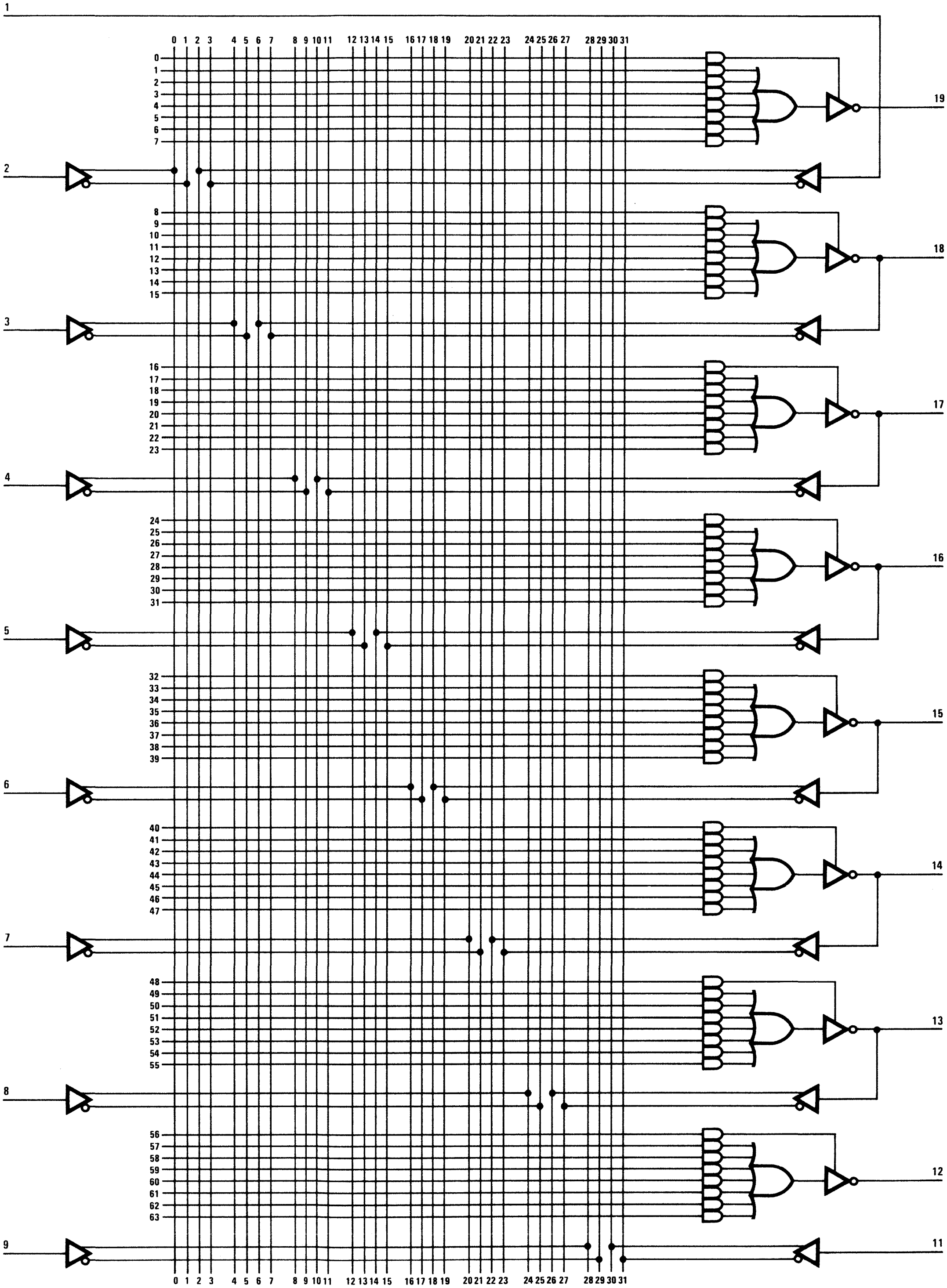
Logic Diagram PAL14L4



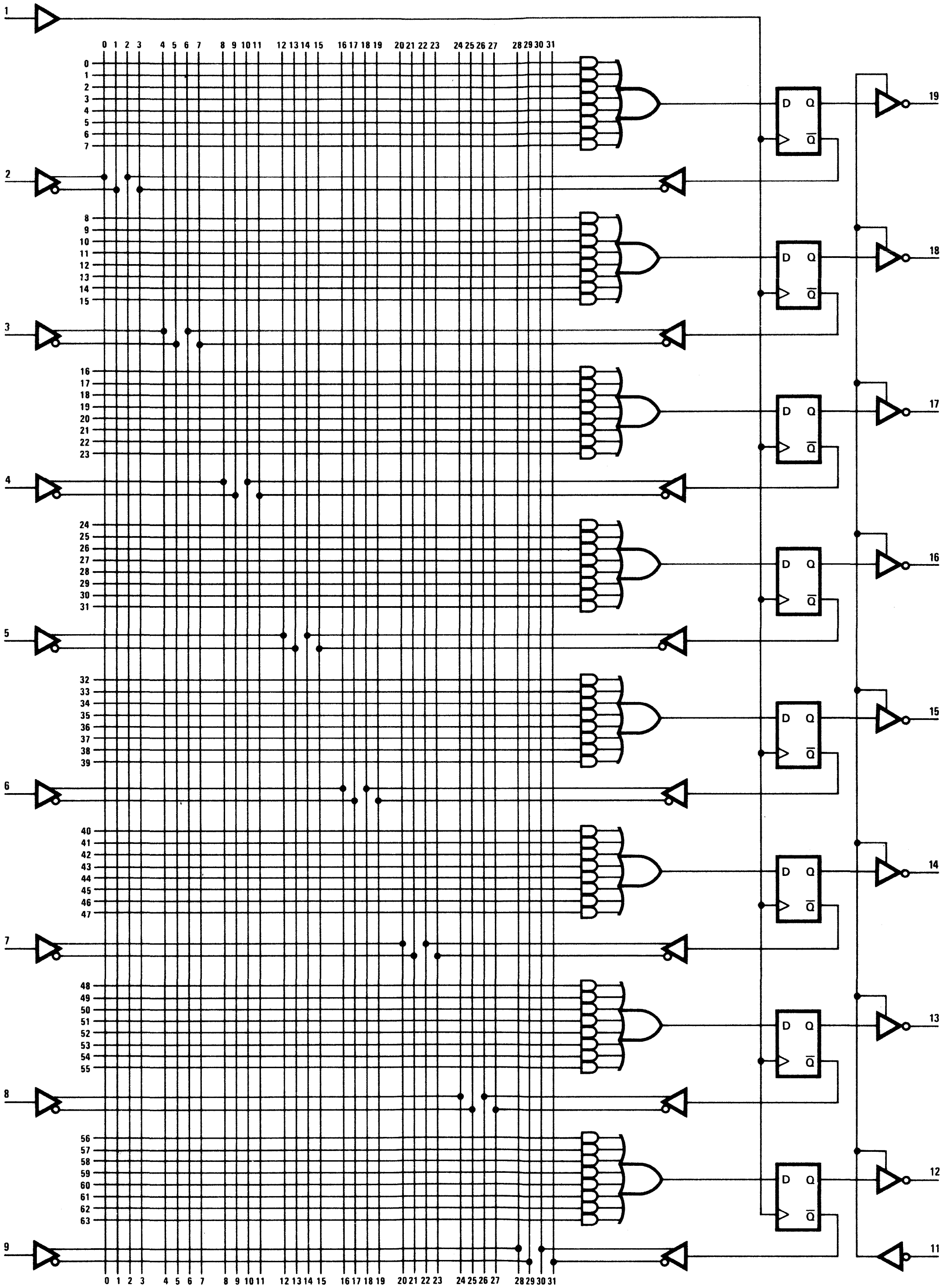
Logic Diagram PAL16L2



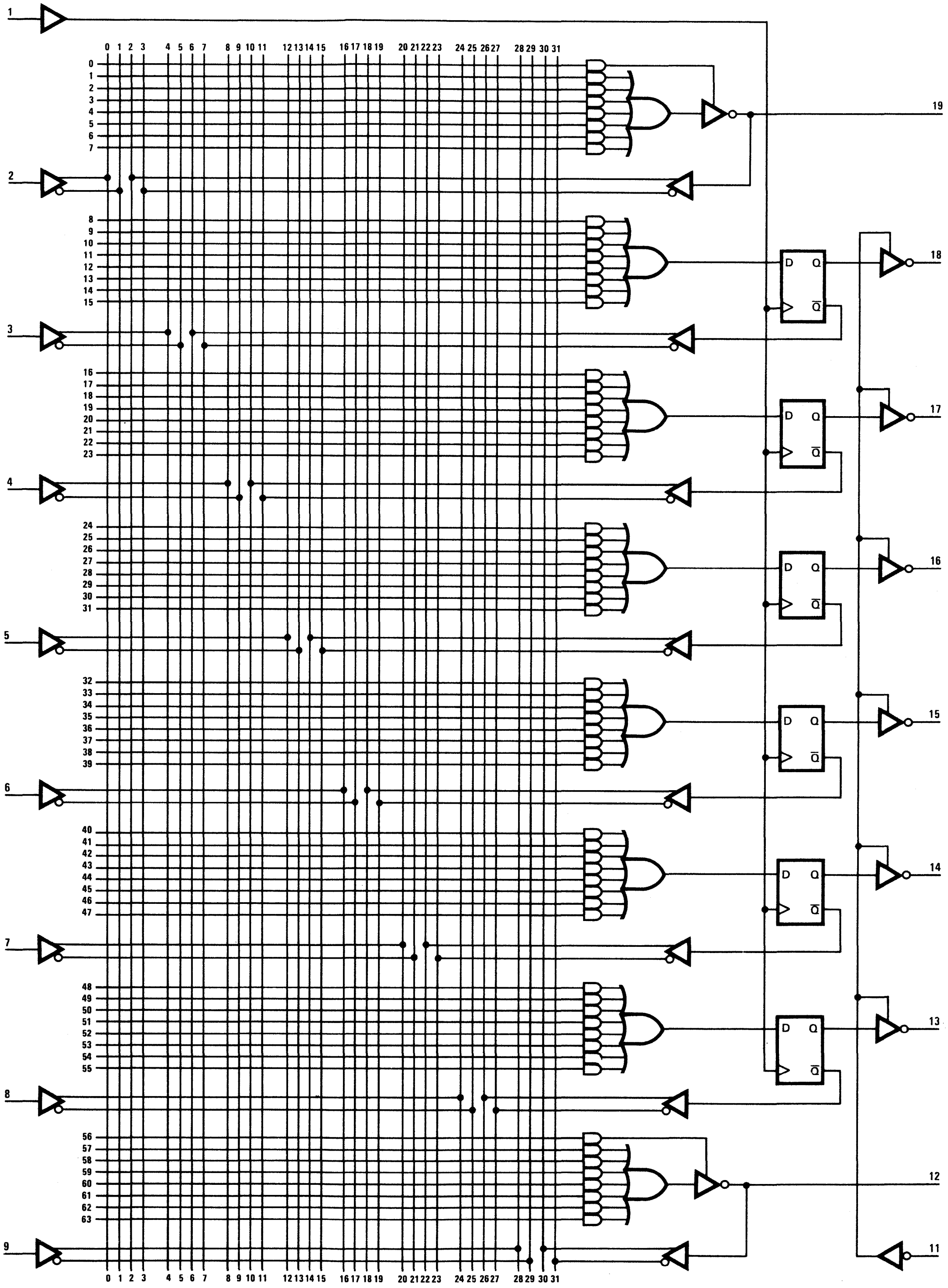
Logic Diagram PAL16L8



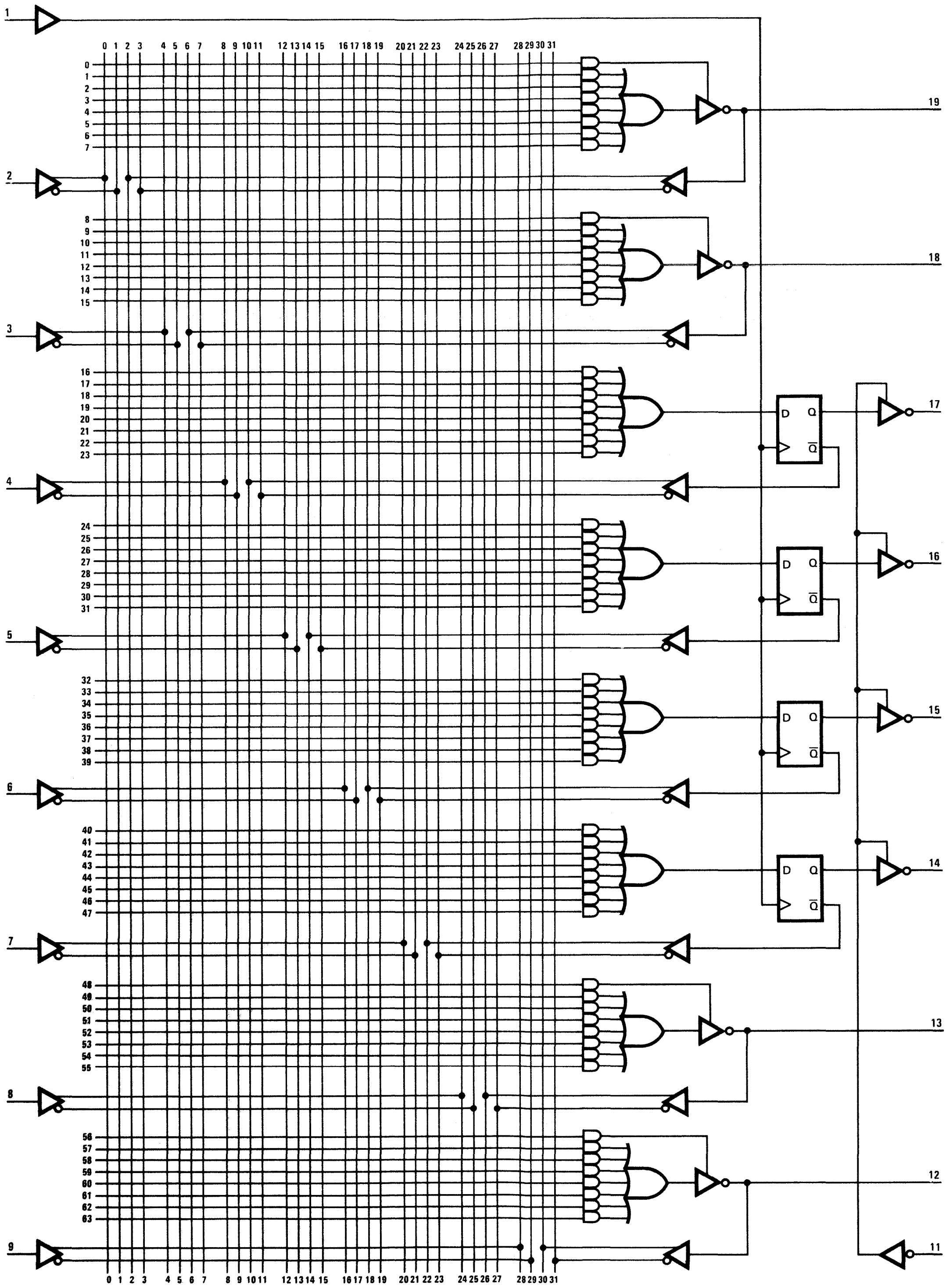
Logic Diagram PAL16R8



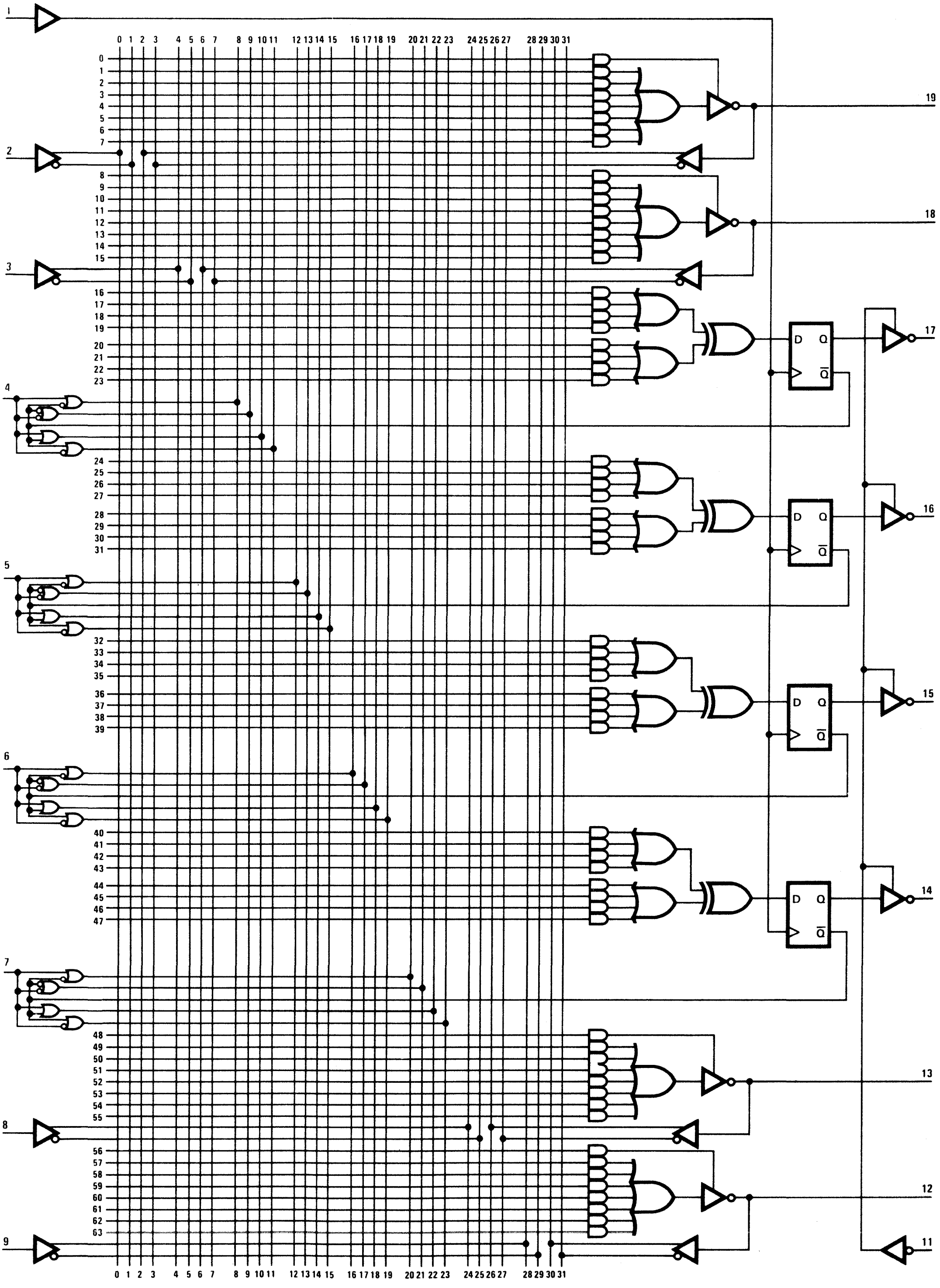
Logic Diagram PAL16R6



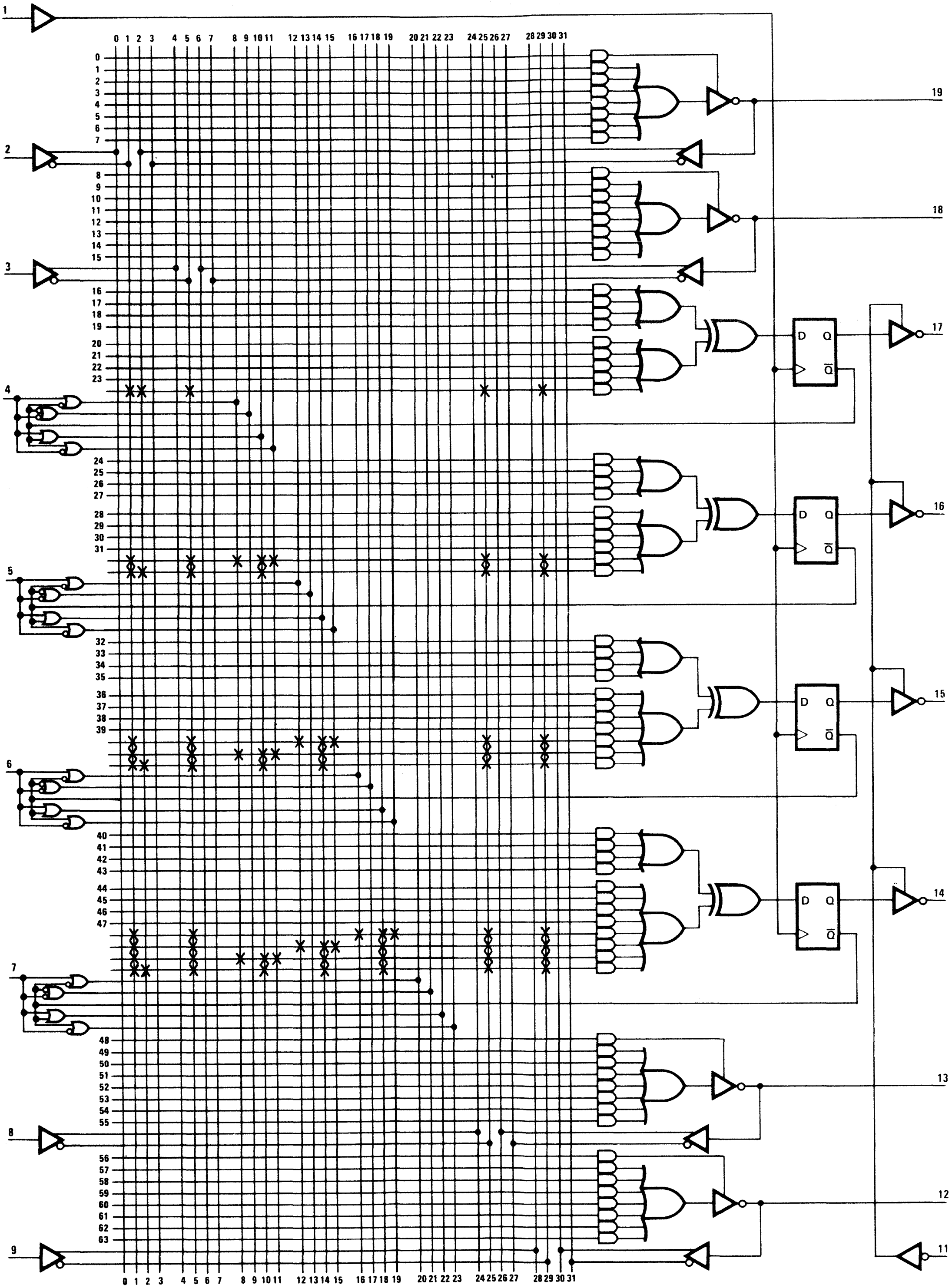
Logic Diagram PAL16R4



Logic Diagram PAL16X4

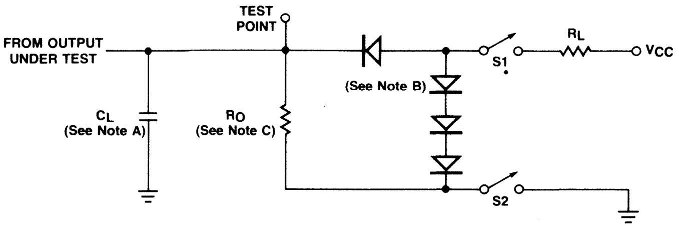


Logic Diagram PAL16A4

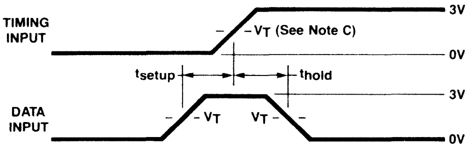


Standard Test Load and Definitions of Waveforms

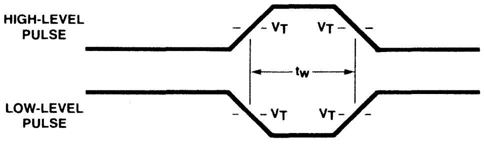
Standard Test Load



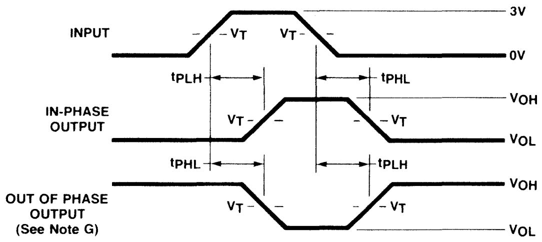
Test Waveforms



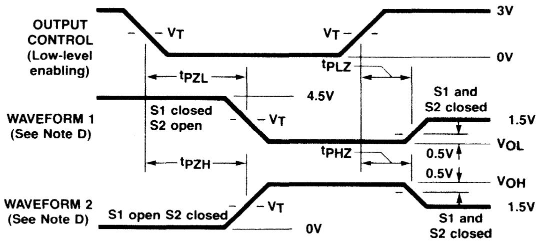
Setup and Hold



Pulse Width



Propagation Delay



Enable and Disable

NOTES: A. C_L includes probe and jig capacitance.

B. All diodes are 1N916 or 1N3064.

C. R_O = 1K, V_T = 1.5V

D. Waveform 1 is for an output with internal conditions such that the output is low except when disabled by the output control. Waveform 2 is for an output with internal conditions such that the output is high except when disabled by the output control.

E. In the examples above, the phase relationships between inputs and outputs have been chosen arbitrarily.

F. All input pulses are supplied by generators having the following characteristics: PRR $\leq$ 1 MHz, Z_{out} = 50Ω and: $t_r \leq$ 15 ns $t_f \leq$ 6 ns

G. When measuring propagation delay times of 3-state outputs, switches S1 and S2 are closed.

Clock Frequency

Maximum clock frequency, f_{max}

The highest rate at which the clock input of a bistable circuit can be driven through its required sequence while maintaining stable transitions of logic level at the output with input conditions established that should cause changes of output logic level in accordance with the specification.

Current

High-level input current, I_{IH}

The current into * an input when a high-level voltage is applied to that input.

High-level output current, I_{OH}

The current into * an output with input conditions applied that according to the product specification will establish a high level at the output.

Low-level input current, I_{IL}

The current into * an input when a low-level voltage is applied to that input.

Low-level output current, I_{OL}

The current into * an output with input conditions applied that according to the product specification will establish a low level at the output.

Off-state (high-impedance-state) output current (of a three-state output), I_{OZ}

The current into * an output having three-state capability with input conditions applied that according to the product specification will establish the high-impedance state at the output.

Short-circuit output current, I_{OS}

The current into * an output when that output is short-circuited to ground (or other specified potential) with input conditions applied to establish the output logic level farthest from ground potential (or other specified potential).

Supply current, I_{CC}

The current into * the V_{CC} supply terminal of an integrated circuit.

*Current out of a terminal is given as a negative value.

Hold Time

Hold time, t_h

The interval during which a signal is retained at a specified input terminal after an active transition occurs at another specified input terminal.

NOTES: 1. The hold time is the actual time between two events and may be insufficient to accomplish the intended result. A minimum value is specified that is the shortest interval for which correct operation of the logic element is guaranteed.

2. The hold time may have a negative value in which case the minimum limit defines the longest interval (between the release of data and the active transition) for which correct operation of the logic element is guaranteed.

Output Enable and Disable Time

Output enable time (of a three-state output) to high level, t_{PZH} (or low level, t_{PZL})

The propagation delay time between the specified reference points on the input and output voltage waveforms with the three-state output changing from a high-impedance (off) state to the defined high (or low) level.

Output enable time (of a three-state output) to high or low level, t_{PZX}

The propagation delay time between the specified reference points on the input and output voltage waveforms with the three-state output changing from a high-impedance (off) state to either of the defined active levels (high or low).

Output disable time (of a three-state output) from high level, t_{PHZ} (or low level, t_{PLZ})

The propagation delay time between the specified reference points on the input and output voltage waveforms with the three-state output changing from the defined high (or low) level to a high-impedance (off) state.

Output disable time (of a three-state output) from high or low level, t_{PXZ}

The propagation delay time between the specified reference points on the input and output voltage waveforms with the three-state output changing from either of the defined active levels (high or low) to a high-impedance (off) state.

Propagation Time

Propagation delay time, t_{PD}

The time between the specified reference points on the input and output voltage waveforms with the output changing from one defined level (high or low) to the other defined level.

Propagation delay time, low-to-high-level output, t_{PLH}

The time between the specified reference points on the input and output voltage waveforms with the output changing from the defined low level to the defined high level.

Propagation delay time, high-to-low-level output, t_{PHL}

The time between the specified reference points on the input and output voltage waveforms with the output changing from the defined high level to the defined low level.

Pulse Width

Pulse width, t_w

The time interval between specified reference points on the leading and trailing edges of the pulse waveform.

Setup Time

Setup time, t_{su}

The time interval between the application of a signal that is maintained at a specified input terminal and a consecutive active transition at another specified input terminal.

NOTES: 1. The setup time is the actual time between two events and may be insufficient to accomplish the setup. A minimum value is specified that is the shortest interval for which correct operation of the logic element is guaranteed.

2. The setup time may have a negative value in which case the minimum limit defines the longest interval (between the active transition and the application of the other signal) for which correct operation of the logic element is guaranteed.

Voltage

High-level input voltage, V_{IH}

An input voltage within the more positive (less negative) of the two ranges of values used to represent the binary variables.

NOTE: A minimum is specified that is the least positive value of high-level voltage for which operation of the logic element within specification limits is guaranteed.

High-level output voltage, V_{OH}

The voltage at an output terminal with input conditions applied that according to the product specification will establish a high level at the output.

Input clamp voltage, V_{IC}

An input voltage in a region of relatively low differential resistance that serves to limit the input voltage swing.

Low-level input voltage, V_{IL}

An input voltage level within the less positive (more negative) of the two ranges of values used to represent the binary variables.

NOTE: A maximum is specified that is the most positive value of low-level input voltage for which operation of the logic element within specification limits is guaranteed.

Low-level output voltage, V_{OL}

The voltage at an output terminal with input conditions applied that according to the product specification will establish a low level at the output.

Negative-going threshold voltage, V_T

The voltage level at a transition-operated input that causes operation of the logic element according to specification as the input voltage falls from a level above the positive-going threshold voltage, V_{T+}.

Positive-going threshold voltage, V_{T+}

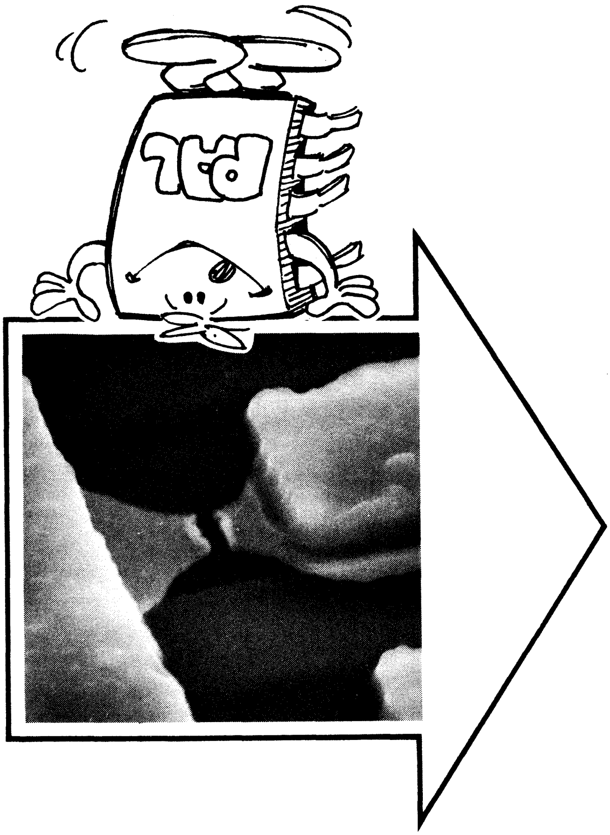
The voltage level at a transition-operated input that causes operation of the logic element according to specification as the input voltage rises from a level below the negative-going threshold voltage, V_{T-}.

Truth Table Explanations

H	=	high level (steady-state)
L	=	low level (steady-state)
↑	=	transition from low to high level
↓	=	transition from high to low level
X	=	irrelevant (any input, including transitions)
Z	=	off (high-impedance) state of a 3-state output
a..h	=	the level of steady-state inputs at inputs A through H respectively
Q_0	=	level of Q before the indicated steady-state input conditions were established
$\overline{Q}_0$	=	complement of Q_0 or level of $\overline{Q}$ before the indicated steady-state input conditions were established
Q_n	=	level of Q before the most recent active transition indicated by ↓ or ↑

If, in the input columns, a row contains only the symbols H, L, and/or X, this means the indicated output is valid whenever the input configuration is achieved and regardless of the sequence in which it is achieved. The output persists so long as the input configuration is maintained.

If, in the input columns, a row contains H, L, and/or X together with ↑ and/or ↓, this means the output is valid whenever the input configuration is achieved but the transition(s) must occur following the achievement of the steady-state levels. If the output is shown as a level (H, L, Q_0, or $\overline{Q}_0$), it persists so long as the steady-state input levels and the levels that terminate indicated transitions are maintained. Unless otherwise indicated, input transitions in the opposite direction to those shown have no effect at the output.



PAL

Selecting the Right PAL

The 15 PAL part types offer a wide range of complexity to choose from. Starting with the PAL10H8 (10 inputs, 8 active high outputs), the first 9 PAL types can replace random SSI gate functions at about a 4 to 1 chip count reduction. With a variety of input/output pin ratios and Active High or Active Low outputs, this group, described as combinatorial, is designed to provide the Low Power Schottky (LS) fan-out and fan-in characteristics of 8 mA output sink (I_{OL}) for totem-pole outputs and 250 μA input loading (I_{IL}).

The next 6 PALs provide the additional features of three-state outputs, state sequencing, arithmetic, and programmable input/output pin ratios. The three-state outputs drive the standard LS output sink of 24 mA (I_{OL}), providing bus driving capability. These sequential PALs are ideal for replacing existing MSI and/or defining new LSI functions not presently available.

Unused inputs should be tied to either V_{CC} or GND. The series resistor required for unused inputs on standard TTL is NOT required for PALs, thus using less parts.

Defining the Pinout

The first step in designing a PAL is selecting the Pinout. The example shown below shows a method for circling a section of conventionally drawn logic to define a boundary for a PAL function. This boundary will dictate a specific number of input pins and output pins. For the example, 8 inputs and 6 outputs are required, well within the capability of the PAL10L8. Assignment of inputs and outputs to specific pins can be done using the PAL Logic Symbol as shown below.

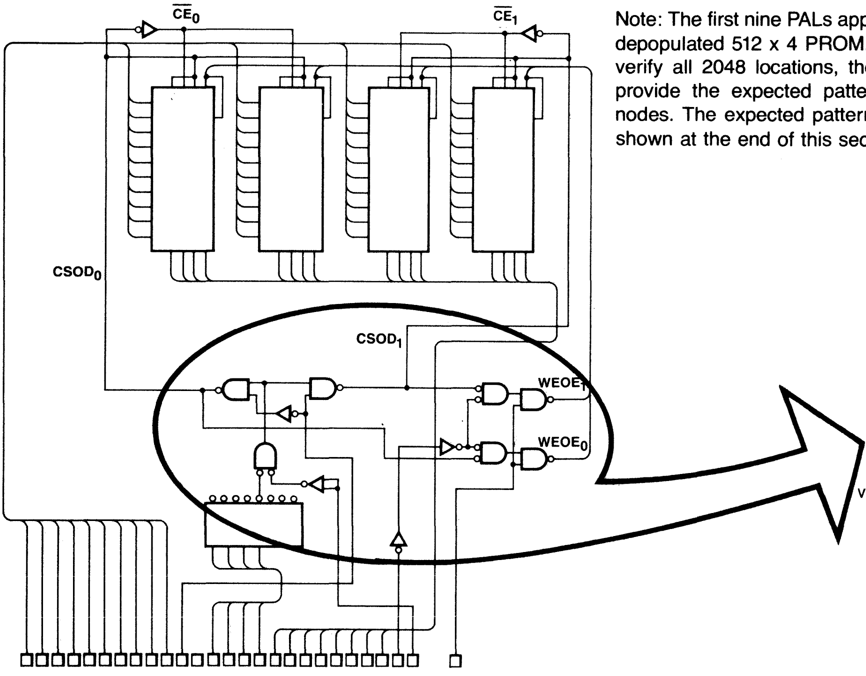
Note: This pinout selection can later be changed to suit printed circuit board layout as will be shown in the Dice Game example.

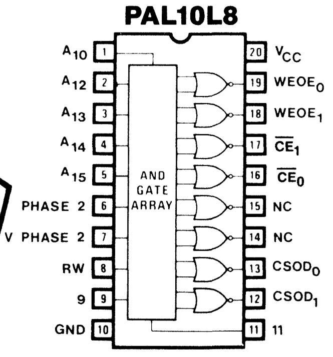
Specifying the Design

Once a pinout is selected, the fuse pattern is specified by one of several methods. The Applications section of this handbook shows a variety of examples and design techniques. The basic method is simply to mark the PAL Logic Diagram with appropriate fuse interconnections for the desired logic transfer function. This can be accomplished by translating conventional logic diagrams to PAL Logic Diagrams. Next, the PAL Logic Diagram is translated to the PAL Programming Format, which is compatible with standard PROM programmer format for 512 x 4 (2048-bit) PROMs. The most common PROM programmer input medium is paper tape, with BHLF, BPNF or Hexadecimal Formats.

Fuses left intact are indicated on the logic diagram by an "X" at the intersection of the input line and the AND gate product line. A blown fuse is not marked. The PAL Logic Diagrams are provided with no fuses marked, allowing the designer to use the diagram as a coding format. Actually, the unprogrammed PAL is shipped with all X's (all fuses) intact. Each fuse node is identified by a Product Line Number and an Input Line Number which are used to locate the corresponding square in the PAL Programming Format. An "X" on the logic diagram corresponds to an "L" or an "N" in the programming format. A blank on the logic diagram corresponds to an "H" or a "P" on the programming format.

Note: The first nine PALs appear to the PROM programmer as a depopulated 512 x 4 PROM. As the programmer will expect to verify all 2048 locations, the PAL Programming Format must provide the expected pattern for verifying non-existent fuse nodes. The expected patterns for non-existent fuse nodes are shown at the end of this section.





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

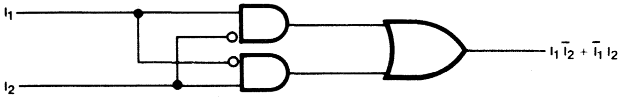
=	Equal, .EQ.
:=	Replaced by Following ↑
/	Complement
◆	AND, Product
+	OR, Sum
:+:	XOR, .NE.
( )	Conditional Three State, Arithmetic

Equations

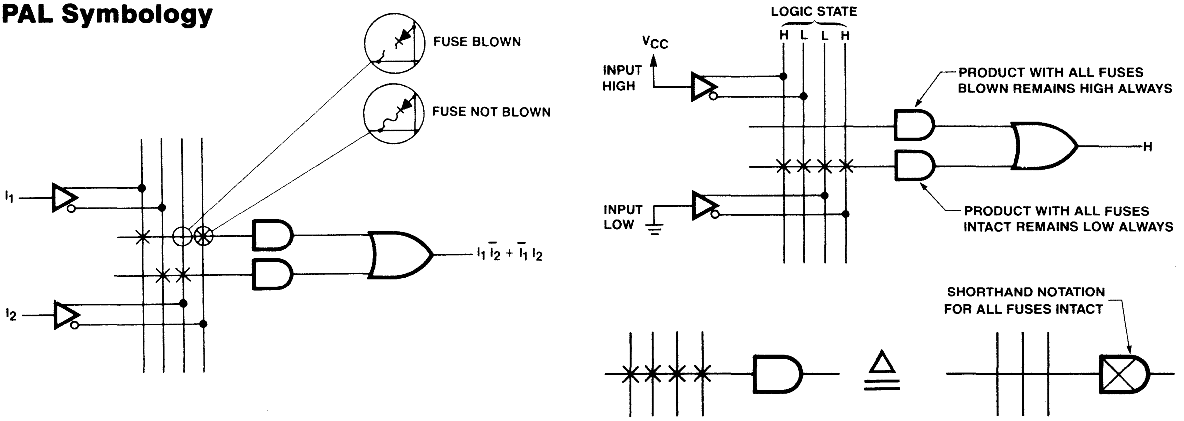
Standard $Q_1 = I_1 \bar{I}_2 + \bar{I}_1 I_2$

PALASM O1 = I1◆/I2 + /I1◆I2

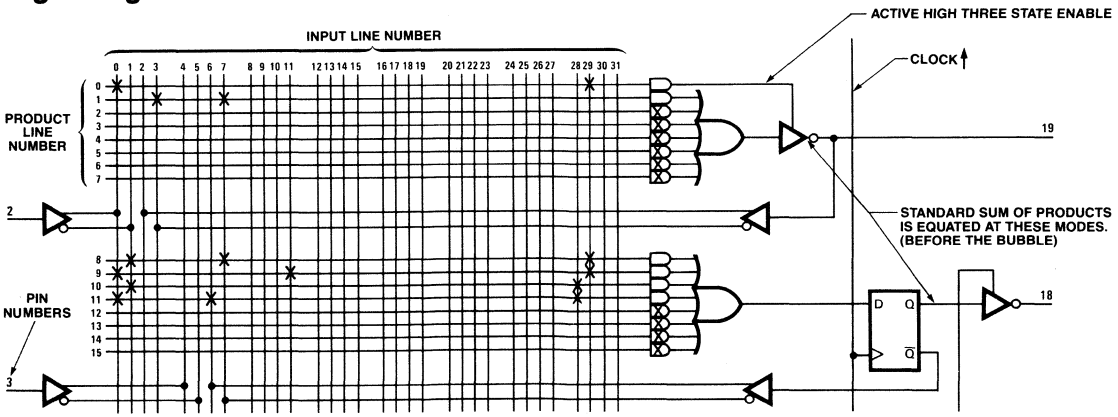
Conventional Symbology



PAL Symbology


FUSE BLOWN
FUSE NOT BLOWN
I_1
I_2
$I_1 \bar{I}_2 + \bar{I}_1 I_2$
LOGIC STATE
H L L H
V_{CC}
INPUT HIGH
INPUT LOW
PRODUCT WITH ALL FUSES BLOWN REMAINS HIGH ALWAYS
PRODUCT WITH ALL FUSES INTACT REMAINS LOW ALWAYS
H
SHORTHAND NOTATION FOR ALL FUSES INTACT
≜

PAL Logic Diagram



PAL Programming Format

Pal ____________________

Pattern ____________________

Name ____________________

For Products 0 thru 31

INPUTS 0 TO 31	WORD 0 TO 255	O_4	O_3	O_2	O_1																																			
		24	16	8	0		25	17	9	1		26	18	10	2		27	19	11	3		28	20	12	4		29	21	13	5		30	22	14	6		31	23	15	7
0	0					32					64					96					128					160					192					224				
1	1					33					65					97					129					161					193					225				
2	2					34					66					98					130					162					194					226				
3	3					35					67					99					131					163					195					227				
4	4					36					68					100					132					164					196					228				
5	5					37					69					101					133					165					197					229				
6	6					38					70					102					134					166					198					230				
7	7					39					71					103					135					167					199					231				
8	8					40					72					104					136					168					200					232				
9	9					41					73					105					137					169					201					233				
10	10					42					74					106					138					170					202					234				
11	11					43					75					107					139					171					203					235				
12	12					44					76					108					140					172					204					236				
13	13					45					77					109					141					173					205					237				
14	14					46					78					110					142					174					206					238				
15	15					47					79					111					143					175					207					239				
16	16					48					80					112					144					176					208					240				
17	17					49					81					113					145					177					209					241				
18	18					50					82					114					146					178					210					242				
19	19					51					83					115					147					179					211					243				
20	20					52					84					116					148					180					212					244				
21	21					53					85					117					149					181					213					245				
22	22					54					86					118					150					182					214					246				
23	23					55					87					119					151					183					215					247				
24	24					56					88					120					152					184					216					248				
25	25					57					89					121					153					185					217					249				
26	26					58					90					122					154					186					218					250				
27	27					59					91					123					155					187					219					251				
28	28					60					92					124					156					188					220					252				
29	29					61					93					125					157					189					221					253				
30	30					62					94					126					158					190					222					254				
31	31					63					95					127					159					191					223					255				

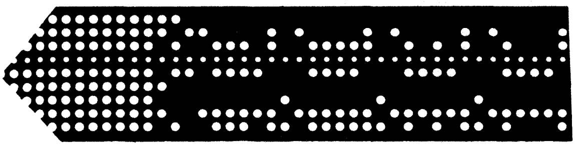
For Products 32 thru 63

INPUTS 0 TO 31	WORDS 256 TO 511	56	48	40	32		57	49	41	33		58	50	42	34		59	51	43	35		60	52	44	36		61	53	45	37		62	54	46	38		63	55	47	39
0	256					288					320					352					384					416					448					480				
1	257					289					321					353					385					417					449					481				
2	258					290					322					354					386					418					450					482				
3	259					291					323					355					387					419					451					483				
4	260					292					324					356					388					420					452					484				
5	261					293					325					357					389					421					453					485				
6	262					294					326					358					390					422					454					486				
7	263					295					327					359					391					423					455					487				
8	264					296					328					360					392					424					456					488				
9	265					297					329					361					393					425					457					489				
10	266					298					330					362					394					426					458					490				
11	267					299					331					363					395					427					459					491				
12	268					300					332					364					396					428					460					492				
13	269					301					333					365					397					429					461					493				
14	270					302					334					366					398					430					462					494				
15	271					303					335					367					399					431					463					495				
16	272					304					336					368					400					432					464					496				
17	273					305					337					369					401					433					465					497				
18	274					306					338					370					402					434					466					498				
19	275					307					339					371					403					435					467					499				
20	276					308					340					372					404					436					468					500				
21	277					309					341					373					405					437					469					501				
22	278					310					342					374					406					438					470					502				
23	279					311					343					375					407					439					471					503				
24	280					312					344					376					408					440					472					504				
25	281					313					345					377					409					441					473					505				
26	282					314					346					378					410					442					474					506				
27	283					315					347					379					411					443					475					507				
28	284					316					348					380					412					444					476					508				
29	285					317					349					381					413					445					477					509				
30	286					318					350					382					414					446					478					510				
31	287					319					351					383					415					447					479					511				

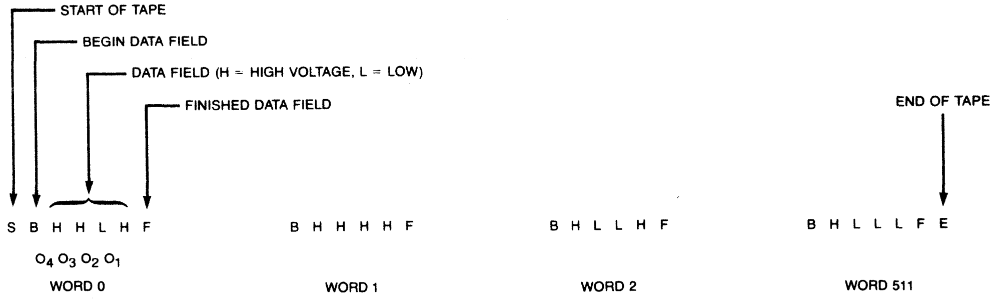
Paper Tape Inputs

Truth tables can be sent to MMI in an ASCII tape format. Information can be sent by mail or TWX. (MMI's TWX number is 910-339-9229.) Although MMI can program PALs with the tape in any format, the following formats have been the most popular.

8 Level TWX



BHLF Format



The required heading information at the beginning of the tape is as follows:

CUSTOMER'S NAME AND PHONE ____________________

CUSTOMER'S TWX NUMBER (IF ANY) ____________________

PURCHASE ORDER NUMBER ____________________

MMI PART NUMBER ____________________

CUSTOMER SYMBOLIZED PART NUMBER ____________________

TRUTH TABLE NUMBER (IF ANY) ____________________

TYPE OF FORMAT (IF ASCII, HEX, BHLF, ETC.) ____________________

25 BELL OR RUBOUT CHARACTERS ____________________

An example is shown below:

```
BLARNEY ELECTRONICS 408-735-8104
TWX911-338-9225
PO142
PAL16R8
0431
PAT0001
BHLF
```

(25 Bell or Rubout Characters)

```
S
BLLLHF BLLLLF BLHLHF BLHHHF BLLHHF BHHHHF BLLLHF BLHLHF
   .      .      .      .      .      .      .      .
   .      .      .      .      .      .      .      .
   .      .      .      .      .      .      .      .
   .      .      .      .      .      .      .      .
   .      .      .      .      .      .      .      .
BLLLLF BLHLHF BLHHHF BLLHHF BHHHHF BLLLHF BLHLHF BLLLLF
E
```

BPNF Format

This format is identical to the BHLF format with the exception that a "P" designates a positive bit, and hence a "high" level, and an "N" represents a negative bit, and hence a "low" level.

Hexadecimal Format

In this format the heading required is identical to the BHLF format but the data is different. Instead of an "S," the hexadecimal data begins with the SOH or STX character (control A). The data is then represented by the hexadecimal character (0-9 and A-F) which represents the output data of address' 0, followed by a space.

Next comes the output data of address 1 followed by a space, etc. The character ETX (control C) is used to end the data. Carriage return and line feed may be included to format the data when the tape is printed.

Documenting the Design

Along with the opportunity to "design your own chip" comes the responsibility of documenting the device in a way that others can understand it. The reader who has been puzzled by logic diagrams showing PROMs to implement random logic can identify with this problem. The documentation for that PROM was a truth table containing no more than ones and zeros, hardly enough information to quickly figure out the function being performed. Using PALs without proper documentation can cause similar confusion.

Consider the technician on the production floor who is tracing logic faults when he observes a PAL16R8, PAT2719 on his logic schematic. "I wonder what this part does," he says to himself. Adjacent to the 16R8 is a 74LS240 which the technician recalls is a 20 pin octal buffer. He knows this because he looked it up in his TTL Databook, where the *data sheet* told all about the function of an octal buffer. "How can I figure out what a PAT2719 does?" he says. At this point the technician would like to reach for a PAL Databook containing a *data sheet* on the PAL16R8, PAT2719.

PAL Design Specification

The PAL Design Specification is a recommended *data sheet* format for describing the function of a PAL once it has acquired the unique personality of a particular fuse pattern. A sample PAL Design Specification is shown on the next page. It contains the essential features of a data sheet including 1) Device Name, 2) Pin List, 3) Description, 4) Function Table and 5) Logic Diagram (on additional page). Two additional features not found in conventional data sheets are also included. First, is the author's name and the date. This information is standard on most engineering documents. Second, are the equations which define the PAL transfer function. The equations specify the precise operation of the PAL and, accordingly, are useful in understanding the function. Of higher importance, however, the equations are the key to automating the process of "designing your own chip."

PALASM

The equations in the PAL Design Specification, along with the PIN List and part number, contain the exact information necessary to generate PAL fuse patterns. Any friendly computer can transform the spec into programming instructions in the form of paper tape or, preferably, RS232C voltage waveforms for no-nonsense direct input to PROM programmers. With a little help from your computer, your PALs can be designed, documented and programmed within minutes.

PALASM, for PAL Assembler, is a Fortran IV program which translates a PAL Design Specification into a PAL Fuse Pattern and PAL Programming Format (BHLF or HEX). PALASM source code is available to users on the following pages. Other source code media is available on request.

Examples of PALASM output are shown in the Applications section. The flow chart outlines the PALASM operation.

Sample PAL Design Specification

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

PATTERN NO.

NAME OF DEVICE (MUST START ON LINE 3)

```
PALXXXX                                           PAL DESIGN SPECIFICATION
PATXXXX                                               AUTHOR'S NAME, DATE
NAME OF DEVICE (EG. CLOCK GENERATOR, PORT ADDRESS DECODER, ETC)

PIN1 PIN2 3 /4     5 6 7 8 9 GND 11 12 13 /14 15 16 /17
 18 19 VCC
```

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

```
19  = PIN1*4 + /PIN2

18  = 5 + 6 + 7 + /8 + 9*11

/17  := 8* 9

16 =    9 *8

IF(PIN1*PIN2) 15 = 3 + 6

/14        = 3+6

 IF( VCC ) 13   = 8*7+PIN2
```

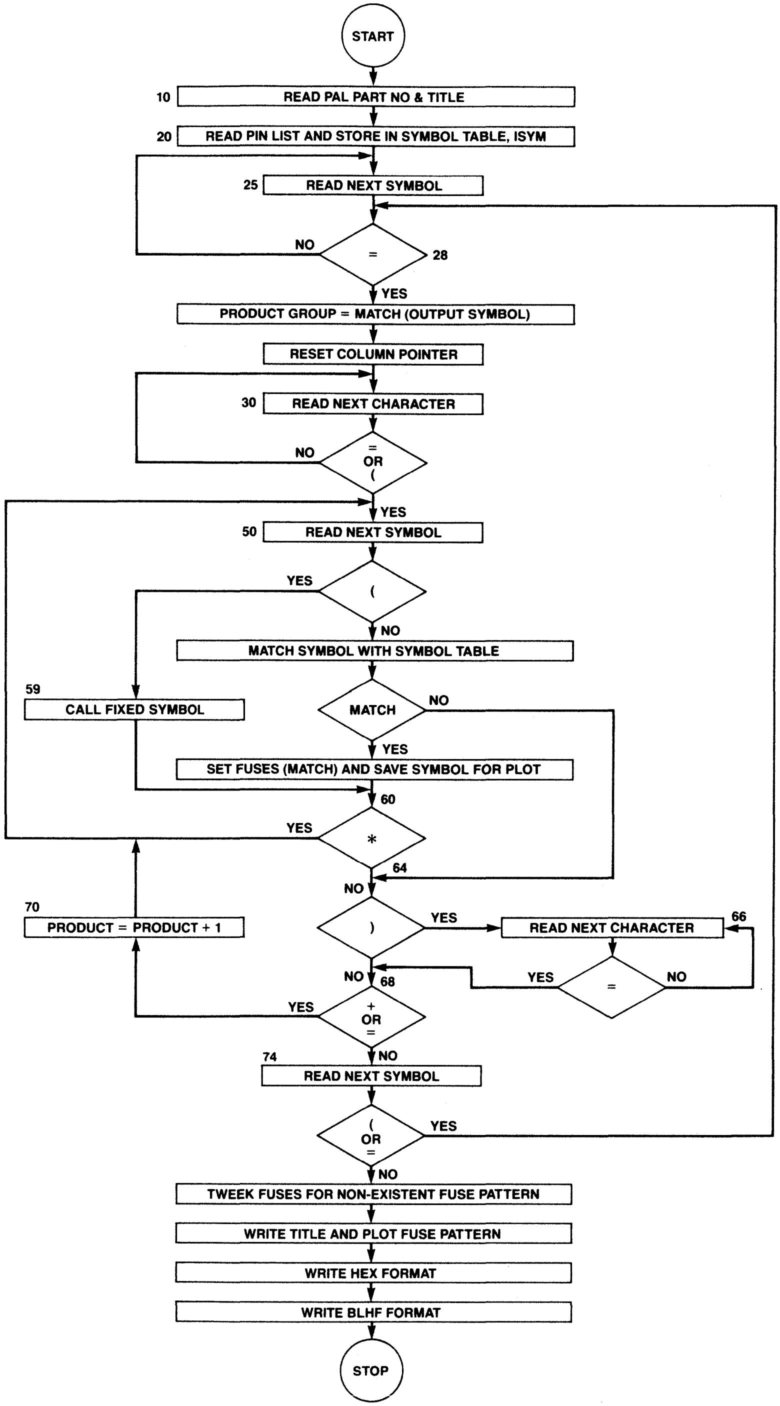
EQUATIONS

```
DESCRIPTION:

THIS PARAGRAPH DESCRIBES THE OPERATION OF THE DEVICE.  APPLICATIONS
INFORMATION MAY ALSO BE PROVIDED.



FUNCTION TABLE

---------------------------------------------------------------------
!     INPUTS        !          OUTPUTS             !                 !
-----------------------------------------------------!   OPERATION     !
! PIN1 PIN2 ! 7 8 9 ! 12 13 14 15 16 17 18 19 !                 !
---------------------------------------------------------------------
!   H    L  ! X L L !  H  H  L  H  L  H  H  L !  CLEAR          !
!   .    .  ! . . . !  .  .  .  .  .  .  .  . !    .            !
!   .    .  ! . . . !  .  .  .  .  .  .  .  . !    .            !
!   .    .  ! . . . !  .  .  .  .  .  .  .  . !    .            !
!   .    .  ! . . . !  .  .  .  .  .  .  .  . !    .            !
!   .    .  ! . . . !  .  .  .  .  .  .  .  . !    .            !
---------------------------------------------------------------------
```

SEE DEFINITION OF TERMS FOR FUNCTION TABLE DEFINITIONS

4

PALASM Flow Chart



PALASM Source Code

```
C     P A L A S M   -   TRANSLATES SYMBOLIC EQUATIONS INTO PAL OBJECT
C                       CODE  FORMATTED FOR DIRECT INPUT TO STANDARD
C                       PROM PROGRAMMERS.
C
C                       INPUT:          PAL DESIGN SPECIFICATION ASSIGNED
C                                       TO DATA SET REFERENCE NUMBER 1
C                                       AND TERMINAL INPUT ASSIGNED TO
C                                       DATA SET REFERENCE NUMBER 5
C
C                       OUTPUT:         FUSE PATTERN, HEX FORMAT, BHLF
C                                       FORMAT, OR BPNF FORMAT ON DATA SET
C                                       REFERENCE NUMBER 6
C
C                       PART NUMBER:    THE PAL PART NUMBER MUST
C                                       APPEAR IN COLUMN ONE OF LINE ONE
C
C                       PIN LIST:       20 SYMBOLIC PIN NAMES MUST APPEAR
C                                       STARTING ON LINE 5
C
C                       EQUATIONS:      STARTING FIRST LINE AFTER THE
C                                       PIN LIST IN THE FOLLOWING FORMS:
C
C                                          A = B*C + D
C
C                                          A := B*C + D
C
C                                          IF( A*B )  C = D + E
C
C                                          A2 := (A1.EQ.B1) + /C
C
C                                       BLANKS AND COLONS ARE IGNORED
C
C                       OPERATORS:      =      EQUALITY
C                                       :=     REPLACED BY (AFTER CLOCK)
C                                       /      COMPLEMENT
C                                       *      AND, PRODUCT
C                                       +      OR,  SUM
C                                       :+:    IMPLIED EXCLUSIVE OR
C                                       ()     CONDITIONAL THREE-STATE
C                                              OR FIXED SYMBOL
C
C                       FIXED SYMBOLS
C                       FOR PAL16X4
C                       AND PAL16A4
C                       ONLY:           (AN+/BN)        WHERE N = 0,1,2,3
C                                       (AN+BN)         FOR OUTPUT PINS
C                                       (AN)            17,16,15,14, RESP
C                                       (/AN+/BN)       A IS OUTPUT
C                                       (/BN)           B IS INPUT
C                                       (AN.NE.BN)
C                                       (AN*/BN)
C                                       (/AN+BN)
C                                       (AN.EQ.BN)
C                                       (BN)
C                                       (AN*BN)
C                                       (/AN)
C                                       (/AN*/BN)
C                                       (/AN*BN)
C
C                       SUBROUTINES:    INITLZ,GETSYM,INCR,MATCH,FIXSYM,
C                                       TWEEK,PLOT,HEX,BHLF
C
C                       FUNCTIONS:      IXLATE
C
C                       REV LEVEL:      D 5/7/78  JB
C
C                       NOTES:          THE SOURCE CODE AS PRINTED HERE
C                                       PRODUCED THE OBJECT CODE OF THE
C                                       EXAMPLES IN THE APPLICATIONS
C                                       SECTION ON A NATIONAL CSS IBM
C                                       SYSTEM/370 FORTRAN IV(G).
```

4

PALASM Source Code

```
C     MAIN PROGRAM
C
      COMMON  LBLANK,LLEFT,LAND,LOR,LSLASH,LEQUAL,LRIGHT
      LOGICAL LBLANK,LLEFT,LAND,LOR,LSLASH,LEQUAL,LRIGHT,LFIRST,LFIX
      LOGICAL LFUSES(32,64),LPHASE(20),LBUF(20),LMATCH
      INTEGER TITLE(80),ILINE(80),ICOLUM,ISYM(8,20),IBUF(8,20),L,H,N,P
      DATA LFUSES/2048*.FALSE./,ICOLUM/0/,L/'L'/,H/'H'/,N/'N'/,P/'P'/
C
      READ(1,10) INDAI,IOT,INOO,TITLE,ILINE
   10 FORMAT(3X,I2,A1,I1,//,80A1,//,80A1)
      CALL INITLZ(INDAI,IOT,INOO,ITYPE,LFUSES,ILINE,ICOLUM)
      DO 20 J=1,20
   20     CALL GETSYM(LPHASE,ISYM,J,ILINE,ICOLUM,LFIX)
   25 CALL GETSYM(LBUF,IBUF,1,ILINE,ICOLUM,LFIX)
   28     IF(.NOT.LEQUAL) GO TO 25
          CALL MATCH(IMATCH,IBUF,ISYM)
          IF( (IMATCH.LT.12) .OR. (IMATCH.GT.19) ) GO TO 100
          I88PRO=(19-IMATCH)*8 + 1
          ICOLUM=0
   30         CALL INCR(ILINE,ICOLUM)
              IF( .NOT. (  LEQUAL.OR.LLEFT  ) ) GO TO 30
          DO 70 I8PRO=1,16
              IPROD = I88PRO + I8PRO - 1
              LFIRST=.TRUE.
   50           CALL GETSYM(LBUF,IBUF,1,ILINE,ICOLUM,LFIX)
                IF(LFIX) GO TO 59
                CALL MATCH(IMATCH,IBUF,ISYM)
                IF(  (IMATCH.EQ.0) .OR. (IMATCH.EQ.10) )  GO TO 64
                IF(.NOT.LFIRST) GO TO 58
                    LFIRST=.FALSE.
                    DO 56 I=1,32
   56                   LFUSES(I,IPROD)=.TRUE.
   58           IBUBL=0
                IF(((     LPHASE(IMATCH)).AND.(.NOT.LBUF(1))).OR.
     C             ((.NOT.LPHASE(IMATCH)).AND.(     LBUF(1)))) IBUBL=1
                IINPUT=IXLATE(IMATCH,ITYPE)+IBUBL
                IF(IINPUT.LE.0) GO TO 60
                LFUSES(IINPUT,IPROD)=.FALSE.
                CALL PLOT(LBUF,IBUF,LFUSES,IPROD,TITLE,.FALSE.)
                GO TO 60
   59           CALL FIXSYM(LBUF,IBUF,ILINE,ICOLUM,LFIRST,LFUSES,IPROD)
   60           IF(LAND) GO TO 50
   64         IF( .NOT.LRIGHT ) GO TO 68
   66           CALL INCR(ILINE,ICOLUM)
                IF( .NOT.LEQUAL )  GO TO 66
   68         IF(.NOT. (LOR.OR.LEQUAL) ) GO TO 74
   70         CONTINUE
   74     CALL GETSYM(LBUF,IBUF,1,ILINE,ICOLUM,LFIX)
          IF(LLEFT.OR.LEQUAL) GO TO 28
  100 IF(ITYPE.LE.4) CALL TWEEK(ITYPE,IOT,LFUSES)
  105 WRITE(6,110)
  110 FORMAT(' OUTPUT ?   PLOT=P HEX=H BHLF=L BPNF=N QUIT=Q')
      READ(5,120) I
  120 FORMAT(1A1)
      IF(I.EQ.P) CALL PLOT(LBUF,IBUF,LFUSES,IPROD,TITLE,.TRUE.)
      IF(I.EQ.H) CALL HEX(LFUSES)
      IF(I.EQ.L) CALL BHLF(LFUSES,H,L)
      IF(I.EQ.N) CALL BHLF(LFUSES,P,N)
      IF( (I.EQ.P).OR.(I.EQ.H).OR.(I.EQ.L).OR.(I.EQ.N) ) GO TO 105
      STOP
      END
C
C***********************************************************************
C
      SUBROUTINE INITLZ(INDAI,IOT,INOO,ITYPE,LFUSES,ILINE,ICOLUM)
      INTEGER L,R,X,A
      DATA L/'L'/,R/'R'/,X/'X'/,A/'A'/
      IF( INDAI .LT. 16 )                        ITYPE = (INDAI/2) - 4
      IF( (INDAI .EQ. 16) .AND. (INOO .EQ. 2) )  ITYPE = 4
      IF( (INDAI .EQ. 16) .AND. (INOO .EQ. 1) )  ITYPE = 4
      IF( (INDAI .EQ. 16) .AND. (IOT .EQ. L) )   ITYPE = 5
      IF( (IOT .EQ. R) .OR. (IOT .EQ. A) .OR. (IOT .EQ. X) ) ITYPE =6
      CALL INCR(ILINE,ICOLUM)
      RETURN
      END
```

PALASM Source Code

```
      SUBROUTINE GETSYM(LPHASE,ISYM,J,ILINE,ICOLUM,LFIX)
      COMMON  LBLANK,LLEFT,LAND,LOR,LSLASH,LEQUAL,LRIGHT
      LOGICAL LBLANK,LLEFT,LAND,LOR,LSLASH,LEQUAL,LRIGHT,LPHASE(20),LFIX
      INTEGER ILINE(80),ISYM(8,20),IBLANK
      DATA IBLANK/' '/
      LFIX=.FALSE.
      IF( .NOT.(LLEFT.OR.LAND.OR.LOR.OR.LEQUAL.OR.LRIGHT) )  GO TO 10
      CALL INCR(ILINE,ICOLUM)
      IF(LLEFT) GO TO 60
   10 LPHASE(J)=( .NOT. LSLASH  )
      IF(LPHASE(J)) GO TO 15
      CALL INCR(ILINE,ICOLUM)
   15 DO 20 I=1,8
   20     ISYM(I,J)=IBLANK
   25 DO 30 I=1,7
   30     ISYM(I,J)=ISYM(I+1,J)
      ISYM(8,J)=ILINE(ICOLUM)
      CALL INCR(ILINE,ICOLUM)
      IF( LLEFT.OR.LBLANK.OR.LAND.OR.LOR.OR.LRIGHT.OR.LEQUAL ) GO TO 40
      GO TO 25
   40 CONTINUE
C     WRITE(6,50)  (ISYM(I,J), I=1,8)
C  50 FORMAT(' ',8A1)
      RETURN
   60 LFIX=.TRUE.
      RETURN
      END
C
C***********************************************************************
C
      SUBROUTINE INCR(ILINE,ICOLUM)
      COMMON  LBLANK,LLEFT,LAND,LOR,LSLASH,LEQUAL,LRIGHT
      LOGICAL LBLANK,LLEFT,LAND,LOR,LSLASH,LEQUAL,LRIGHT
      INTEGER ILINE(80),IBLANK,ILEFT,IAND,IOR,ISLASH,IEQUAL,IRIGHT,
     C        ICOLON
      DATA IBLANK/' '/,ILEFT/'('/,IAND/'*'/,IOR/'+'/,
     C        ISLASH/'/'/,IEQUAL/'='/,IRIGHT/')'/,ICOLON/':'/
      LBLANK=.FALSE.
   10 ICOLUM=ICOLUM+1
      IF(ICOLUM.LE.79) GO TO 30
      READ(1,20,ERR=60,END=60) ILINE
      ICOLUM=1
   20 FORMAT(80A1)
   30 IF(  ILINE(ICOLUM) .EQ. IBLANK   )  LBLANK=.TRUE.
      IF( ( ILINE(ICOLUM).EQ.IBLANK ) .OR. ( ILINE(ICOLUM).EQ.ICOLON ) )
     C  GO TO 10
      LLEFT =(  ILINE(ICOLUM) .EQ.   ILEFT )
      LAND  =(  ILINE(ICOLUM) .EQ.   IAND  )
      LOR   =(  ILINE(ICOLUM) .EQ.    IOR  )
      LSLASH=(  ILINE(ICOLUM) .EQ.   ISLASH)
      LEQUAL=(  ILINE(ICOLUM) .EQ.   IEQUAL)
      LRIGHT=(  ILINE(ICOLUM) .EQ.   IRIGHT)
C     WRITE(6,50) ILINE(ICOLUM)
C  50 FORMAT(' .',1A1)
   60 RETURN
      END
C
C***********************************************************************
C
      SUBROUTINE MATCH(IMATCH,IBUF,ISYM)
      INTEGER IBUF(8,20),ISYM(8,20)
      LOGICAL LMATCH
      IMATCH=0
      DO 20 J=1,20
          LMATCH=.TRUE.
          DO 10 I=1,8
   10         LMATCH=LMATCH.AND.(IBUF(I,1).EQ.ISYM(I,J))
          IF(LMATCH) IMATCH=J
   20     CONTINUE
      RETURN
      END
```

PALASM Source Code

```
      FUNCTION IXLATE(IMATCH,ITYPE)
      INTEGER ITABLE(20,6)
      DATA    ITABLE/
     C      3, 1, 5, 9,13,17,21,25,29,-1,31,-1,-1,-1,-1,-1,-1,-1,-1,-1,
     C      3, 1, 5, 9,13,17,21,25,29,-1,31,27,-1,-1,-1,-1,-1,-1, 7,-1,
     C      3, 1, 5, 9,13,17,21,25,29,-1,31,27,23,-1,-1,-1,-1,11, 7,-1,
     C      3, 1, 5, 9,13,17,21,25,29,-1,31,27,23,19,-1,-1,15,11, 7,-1,
     C      3, 1, 5, 9,13,17,21,25,29,-1,31,-1,27,23,19,15,11, 7,-1,-1,
     C     -1, 1, 5, 9,13,17,21,25,29,-1,-1,31,27,23,19,15,11, 7, 3,-1/
      IXLATE=ITABLE(IMATCH,ITYPE)
      RETURN
      END
C
C◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
C
      SUBROUTINE FIXSYM(LBUF,IBUF,ILINE,ICOLUM,LFIRST,LFUSES,IPROD)
      LOGICAL LBUF(20),LFUSES(32,64),LFIRST,LMATCH
      INTEGER IBUF(8,20),ILINE(80),FIXBUF(8),A,B,ISLASH,IOR,IAND,N,Q,
     C        N0,N1,N2,N3,IBLANK,IRIGHT,TABLE(5,14)
      DATA A/'A'/,B/'B'/,ISLASH/'/'/,IOR/'+'/,IBLANK/' '/,IRIGHT/')'/,
     C        IAND/'◆'/,N/'N'/,Q/'Q'/,N0/'0'/,N1/'1'/,N2/'2'/,N3/'3'/,
     C        TABLE      /      ' ','A','+','/','B',' ',' ','A','+','B',
     C      ' ',' ',' ',' ','A','/','A','+','/','B',' ',' ',' ','/','B',
     C      ' ',' ','A','N','B',' ','A','◆','/','B',' ','/','A','+','B',
     C      ' ',' ','A','Q','B',' ',' ',' ',' ','B',' ',' ','A','◆','B',
     C      ' ',' ',' ','/','A','/','A','◆','/','B',' ','/','A','◆','B'/
      IINPUT=0
      DO 20 I=1,8
          IBUF(I,1)=IBLANK
   20     FIXBUF(I)=IBLANK
   21 CALL INCR(ILINE,ICOLUM)
      I=ILINE(ICOLUM)
      IF(I.EQ.IRIGHT) GO TO 40
      IF(I.EQ.N0) IINPUT=8
      IF(I.EQ.N1) IINPUT=12
      IF(I.EQ.N2) IINPUT=16
      IF(I.EQ.N3) IINPUT=20
      DO 24 J=1,7
   24     IBUF(J,1)=IBUF(J+1,1)
      IBUF(8,1)=I
      IF(.NOT. ( (I.EQ.A).OR.(I.EQ.B).OR.(I.EQ.ISLASH).OR.(I.EQ.IOR)
     C       .OR.(I.EQ.IAND).OR.(I.EQ.N).OR.(I.EQ.Q) ) )  GO TO 21
      DO 30 I=1,4
   30     FIXBUF(I)=FIXBUF(I+1)
      FIXBUF(5)=ILINE(ICOLUM)
      GO TO 21
   40 IMATCH=0
      DO 60 J=1,14
          LMATCH=.TRUE.
          DO 50 I=1,5
   50         LMATCH=LMATCH .AND. ( FIXBUF(I).EQ.TABLE(I,J) )
   60     IF(LMATCH) IMATCH=J
      IF(IMATCH.EQ.0) GO TO 100
      IF(.NOT.LFIRST) GO TO 85
          LFIRST=.FALSE.
          DO 80 I=1,32
   80         LFUSES(I,IPROD)=.TRUE.
   85 DO 90 I=1,4
      IF( (IMATCH-7).GT.0 ) LFUSES(IINPUT+I,IPROD)=.FALSE.
      IF( (IMATCH-7).GT.0 ) IMATCH=IMATCH-8
   90 IMATCH=IMATCH+IMATCH
      LBUF(1)=.TRUE.
      CALL PLOT(LBUF,IBUF,LFUSES,IPROD,TITLE,.FALSE.)
  100 CALL INCR(ILINE,ICOLUM)
      RETURN
      END
```

PALASM Source Code

```
      SUBROUTINE PLOT(LBUF,IBUF,LFUSES,IPROD,TITLE,LDUMP)
      INTEGER IBUF(8,20),IOUT(64),IBLANK,IAND,IOR,ISLASH,IDASH,X,
     C        ISAVE(64,32),TITLE(80)
      DATA IBLANK/' '/,IAND/'*'/,ISAVE/2048*' '/,
     C        IOR/'+'/,ISLASH/'/'/,X/'X'/,IDASH/'-'/
      LOGICAL LBUF(20),LFUSES(32,64),LDUMP
      IF(LDUMP) GO TO 60
      IF(ISAVE(IPROD,1).NE.IBLANK) RETURN
      IF(  LBUF(1)   )  GO TO 5
      DO 30 J=1,31
   30     ISAVE(IPROD,J)=ISAVE(IPROD,J+1)
      ISAVE(IPROD,32)=ISLASH
    5 DO 20 I=1,8
         IF(  ISAVE(IPROD,1).NE.IBLANK   )  RETURN
          IF(  IBUF(I,1)  .EQ.  IBLANK   )  GO TO 20
          DO 10 J=1,31
   10         ISAVE(IPROD,J)=ISAVE(IPROD,J+1)
          ISAVE(IPROD,32)=IBUF(I,1)
   20     CONTINUE
      IF(ISAVE(IPROD,1).NE.IBLANK) RETURN
   40 DO 50 J=1,31
   50     ISAVE(IPROD,J)=ISAVE(IPROD,J+1)
      ISAVE(IPROD,32)=IAND
      RETURN
   60 WRITE(6,62) TITLE
   62 FORMAT(////,80A1,//)
      DO 100 I88PRO=1,57,8
          DO 94 I8PRO=1,8
              IPROD=I88PRO+I8PRO-1
              ISAVE(IPROD,32)=IBLANK
              DO 70 I=1,32
                  IF(  ISAVE(IPROD,1)  .NE.  IBLANK   )   GO TO 70
                  DO 65 J=1,31
   65                 ISAVE(IPROD,J)=ISAVE(IPROD,J+1)
                  ISAVE(IPROD,32)=IBLANK
   70             CONTINUE
              DO 80 I=1,32
                  IOUT(I)=X
                  IF(  LFUSES(I,IPROD)  )   IOUT(I)=IDASH
                  IOUT(I+32)=ISAVE(IPROD,I)
   80         CONTINUE
              WRITE(6,90) IOUT
   90         FORMAT(8(' ',4A1),' ',32A1)
   94         CONTINUE
          WRITE(6,96)
   96     FORMAT(1X)
  100     CONTINUE
      RETURN
      END
C
C***********************************************************************
C
      SUBROUTINE HEX(LFUSES)
      LOGICAL LFUSES(32,64)
      INTEGER ITEMP(32)
      DATA BYP/Z24000000/,STX/Z40000000/,BELL/ZE0E0E0E0/
      WRITE(6,10)
   10 FORMAT(//,'                                         .',//)
C***** NOTE: SOME PROM PROGRAMMERS NEED A START CHARACTER.
C*****       THIS PROGRAM OUTPUTS AN STX FOR THE DATA I/O MODEL 9
C*****       VIA A BYPASS CODE (BYP).  (USE SOH FOR MODEL 5)
      WRITE(6,5) BYP,BELL,BELL,BELL,BELL,BELL,BELL,BELL,BELL,BELL,STX
    5 FORMAT(11A1)
      DO 40 I=1,33,32
      INC=I-1
        DO 40 IPROD=1,8
            DO 20 IINPUT=1,32
            IHEX=0
            IF(LFUSES(IINPUT,IPROD+ 0+INC)) IHEX=IHEX+1
            IF(LFUSES(IINPUT,IPROD+ 8+INC)) IHEX=IHEX+2
            IF(LFUSES(IINPUT,IPROD+16+INC)) IHEX=IHEX+4
            IF(LFUSES(IINPUT,IPROD+24+INC)) IHEX=IHEX+8
   20       ITEMP(IINPUT)=IHEX
   40   WRITE(6,60) ITEMP
   60   FORMAT(' ',32(Z1,' '),'.')
      WRITE(6,10)
      RETURN
      END
```

PALASM Source Code

```
      SUBROUTINE TWEEK(ITYPE,IOT,LFUSES)
      INTEGER L,C
      LOGICAL LFUSES(32,64)
      DATA L/'L'/,C/'C'/
      IF(ITYPE.GE.4) GO TO 20
      DO 10 IPROD=1,64
          LFUSES(15,IPROD)=.TRUE.
          LFUSES(16,IPROD)=.TRUE.
          LFUSES(19,IPROD)=.TRUE.
          LFUSES(20,IPROD)=.TRUE.
          IF(ITYPE.GE.3) GO TO 10
          LFUSES(11,IPROD)=.TRUE.
          LFUSES(12,IPROD)=.TRUE.
          LFUSES(23,IPROD)=.TRUE.
          LFUSES(24,IPROD)=.TRUE.
          IF(ITYPE.GE.2) GO TO 10
          LFUSES( 7,IPROD)=.TRUE.
          LFUSES( 8,IPROD)=.TRUE.
          LFUSES(27,IPROD)=.TRUE.
          LFUSES(28,IPROD)=.TRUE.
   10     CONTINUE
   20 IF( ITYPE.EQ.1 ) GO TO 100
      DO 99 IINPUT=1,32
          DO 30 IPROD=1,8
              LFUSES(IINPUT,IPROD+ 0)= (IOT.NE.L)
   30         IF(IOT.NE.C) LFUSES(IINPUT,IPROD+56)= (IOT.NE.L)
          IF(ITYPE.LE.2) GO TO 99
          DO 40 IPROD=1,8
              LFUSES(IINPUT,IPROD+ 8)= (IOT.NE.L)
   40         IF(IOT.NE.C) LFUSES(IINPUT,IPROD+48)= (IOT.NE.L)
          IF(ITYPE.LE.3) GO TO 99
          DO 50 IPROD=1,8
              LFUSES(IINPUT,IPROD+16)= (IOT.NE.L)
   50         IF(IOT.NE.C) LFUSES(IINPUT,IPROD+40)= (IOT.NE.L)
   99     CONTINUE
  100 RETURN
      END
C
C***********************************************************************
C
      SUBROUTINE BHLF(LFUSES,H,L)
      LOGICAL LFUSES(32,64)
      INTEGER ITEMP(4,8),L,H
      WRITE(6,10)
   10 FORMAT(//,'                                          .',//)
      DO 20 I=1,33,32
      INC=I-1
        DO 20 IPROD=1,8
            DO 20 J=1,25,8
              DO 15 K=1,8
                IINPUT=J+K-1
                ITEMP(1,K)=L
                ITEMP(2,K)=L
                ITEMP(3,K)=L
                ITEMP(4,K)=L
                IF(LFUSES(IINPUT,IPROD+ 0+INC))  ITEMP(4,K)=H
                IF(LFUSES(IINPUT,IPROD+ 8+INC))  ITEMP(3,K)=H
                IF(LFUSES(IINPUT,IPROD+16+INC))  ITEMP(2,K)=H
                IF(LFUSES(IINPUT,IPROD+24+INC))  ITEMP(1,K)=H
   15           CONTINUE
   20         WRITE(6,30) ITEMP
   30         FORMAT(' ',8('B',4A1,'F '))
      WRITE(6,10)
      RETURN
      END
```

PROM Programmer

Fuse Pattern for Virgin PAL10H8,10L8

```
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX

XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX

XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX

XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX

XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX

XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX

XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX

XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
```

PROM Programmer

Fuse Pattern for Virgin PAL12H6

```
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----

XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX

XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX

XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX

XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX

XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX

XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX

---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
```

PROM Programmer

Fuse Pattern for Virgin PAL14H4

```
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----

---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----

XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX

XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX

XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX

XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX

---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----

---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
```

PROM Programmer

Fuse Pattern for Virgin PAL16H2

```
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----

---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----

---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----

---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----

---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
```

PROM Programmer

Fuse Pattern for Virgin PAL16C1

```
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----

---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----

---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
```

PROM Programmer

Fuse Pattern for Virgin PAL12L6

```
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX

XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX

XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX

XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX

XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX

XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
```

PROM Programmer

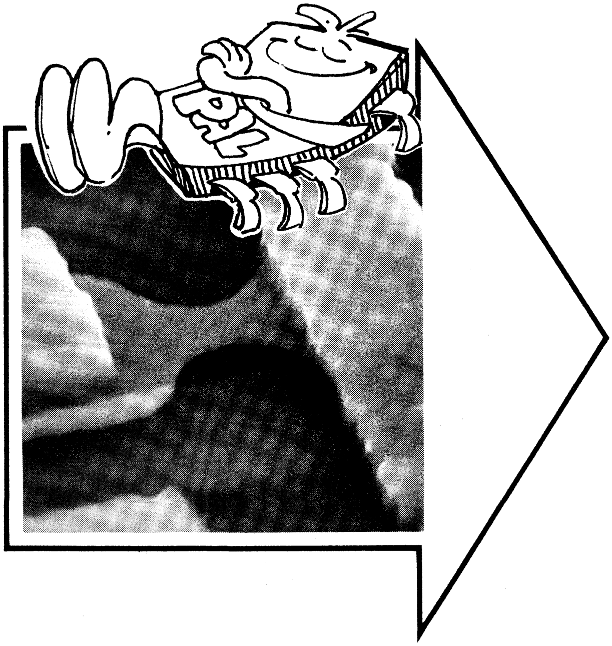
Fuse Pattern for Virgin PAL14L4

```
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX

XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX

XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX

XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
```

PROM Programmer

Fuse Pattern for Virgin PAL16L2, 16L8, 16R8, 16R6, 16R4, 16X4, 16A4

```
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
```


PAL

Applications Contents

Introduction to PAL Applications

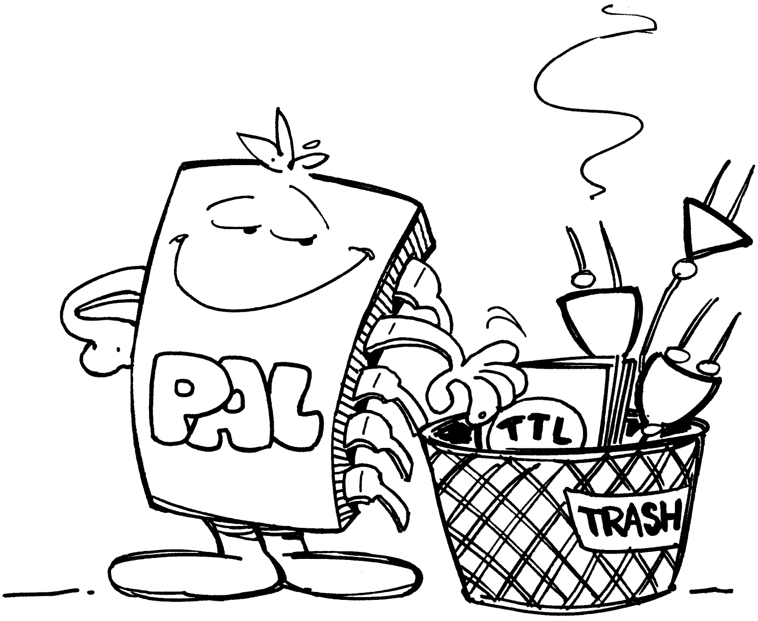
The PAL family brings a unique flexibility to the field of logic design. Using PALs, designers can both replace conventional logic in existing products and optimize the design of new products. Previous sections discussed the PAL concept and provided information on the advantages gained and the techniques used when designing with PALs. This section shows PALs at work in applications ranging from simple logic gate replacement to complex control sequencers.

Each example is presented as a complete PAL design. The required logic function is described, the PAL that best solves the problem is selected, and the actual PAL logic implementation is shown. The PAL logic is shown as both the PAL design specification and the actual PALASM output for the PAL programmer. This makes the examples complete enough to serve as guides for designers using PALs in their own systems.

The PAL is a versatile device whose applications are practically unlimited. These applications examples, combined with the PAL design information contained in the rest of this book, will help designers to get the feel of PAL design procedures. With a little practice and study, PAL design will become a natural extension of the normal logic design process.

Applications

Basic Gate Examples

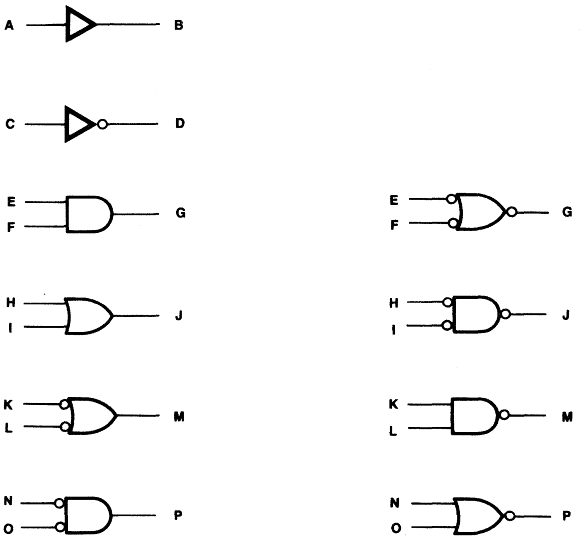


The members of the PAL family are ideal for direct replacement of much combinatorial logic in many conventional designs. The ease with which PAL inputs and outputs can be programmed makes them a natural for use in applications where a small number of SSI/MSI logic functions are required. This section presents simple gate-for-gate logic function replacements using PALs. Later sections show how entire logic functions can be replaced using single PALs.

Example Gates No. 1

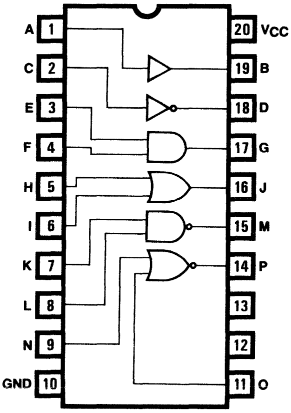
Design Specification PAL10H8

```
PAL10H8                                              PAL DESIGN SPECIFICATION
PAT0025                                                      JOHN DOE 12/10/77
EXAMPLE GATES     NO. 1

A C E F H I K L N GND O NOTUSED NOTUSED P M J G D B VCC



B = A

D = /C

G = E * F

J = H + I

M = /K + /L

P = /N * /O
```



```
DESCRIPTION:

THE EXAMPLE GATES DEMONSTRATE HOW FUSABLE LOGIC CAN IMPLEMENT THE BASIC
BUFFER, INVERTER, AND, OR, NAND, NOR, FUNCTIONS.  NOTE THE ONE FOR ONE
CORRESPONDENCE BETWEEN CONVENTIONAL LOGIC SYMBOLOGY AND PAL LOGIC SYMBOLOGY
```

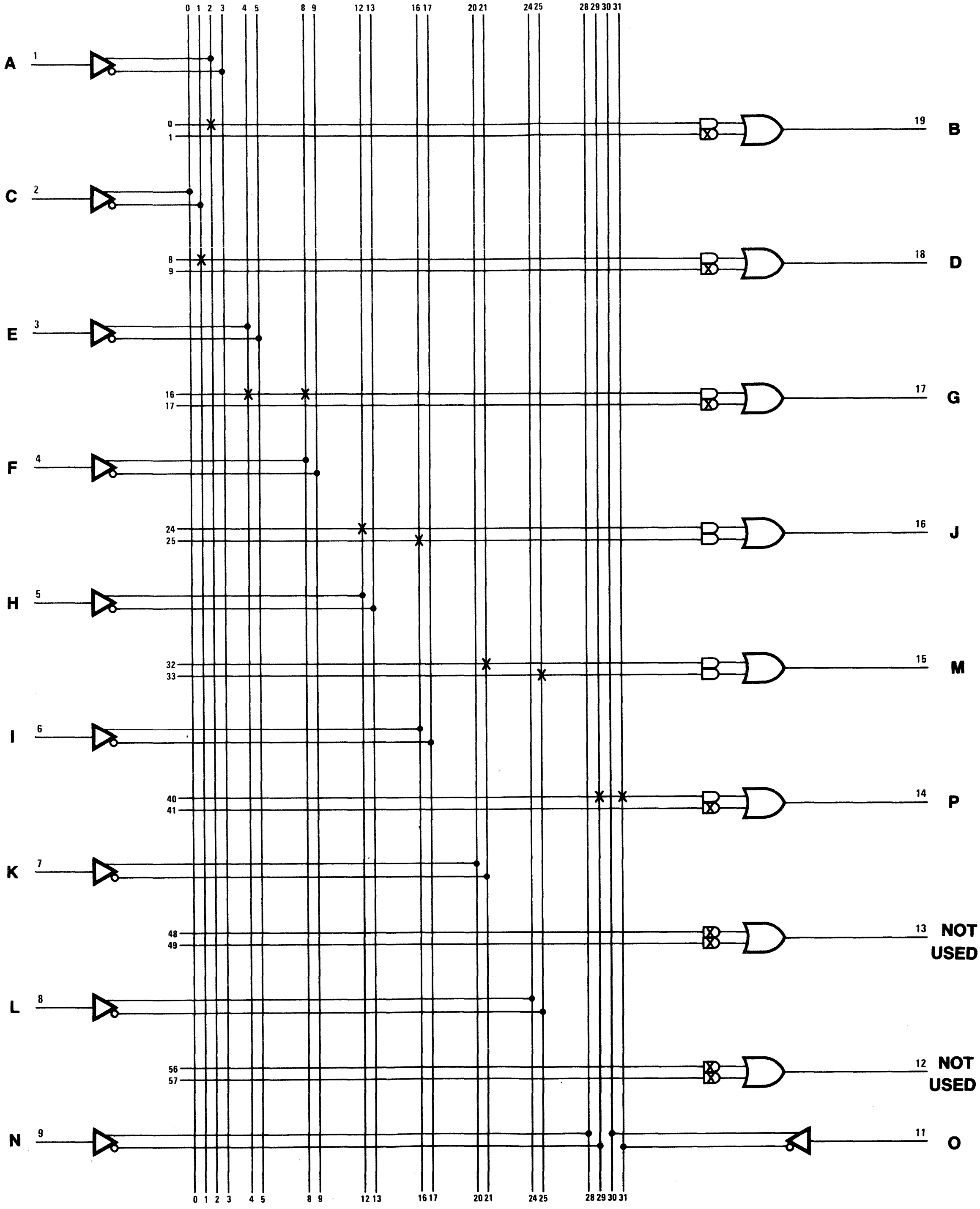
PAL10H8



Logic Symbol

Example Gates No. 1

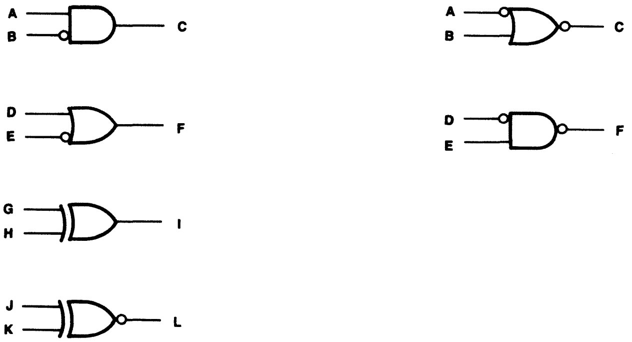
Logic Diagram PAL10H8



Example Gates No. 2

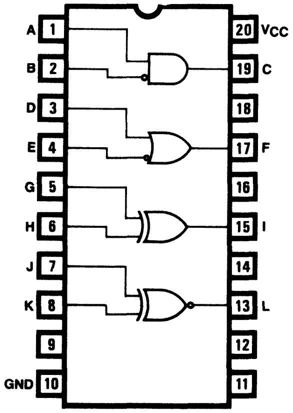
Design Specification PAL10H8

```
PAL10H8                                                PAL DESIGN SPECIFICATION
PAT0025                                                      JOHN DOE 12/10/77
EXAMPLE GATES     NO. 2

A B D E G H J K NC GND NC NC L NC I NC F NC C VCC


C = A * /B

F = D + /E

I = G*/H + /G*H

L = J*K + /J*/K
```



```
DESCRIPTION:

THE EXAMPLE GATES DEMONSTRATE HOW FUSABLE LOGIC CAN IMPLEMENT NON-STANDARD
GATE FUNCTIONS AND ALSO THE EXCLUSIVE OR/NOR FUNCTIONS.  NOTE THE ONE FOR
ONE CORRESPONDENCE BETWEEN CONVENTIONAL LOGIC SYMBOLOGY AND PAL LOGIC
SYMBOLOGY.
```

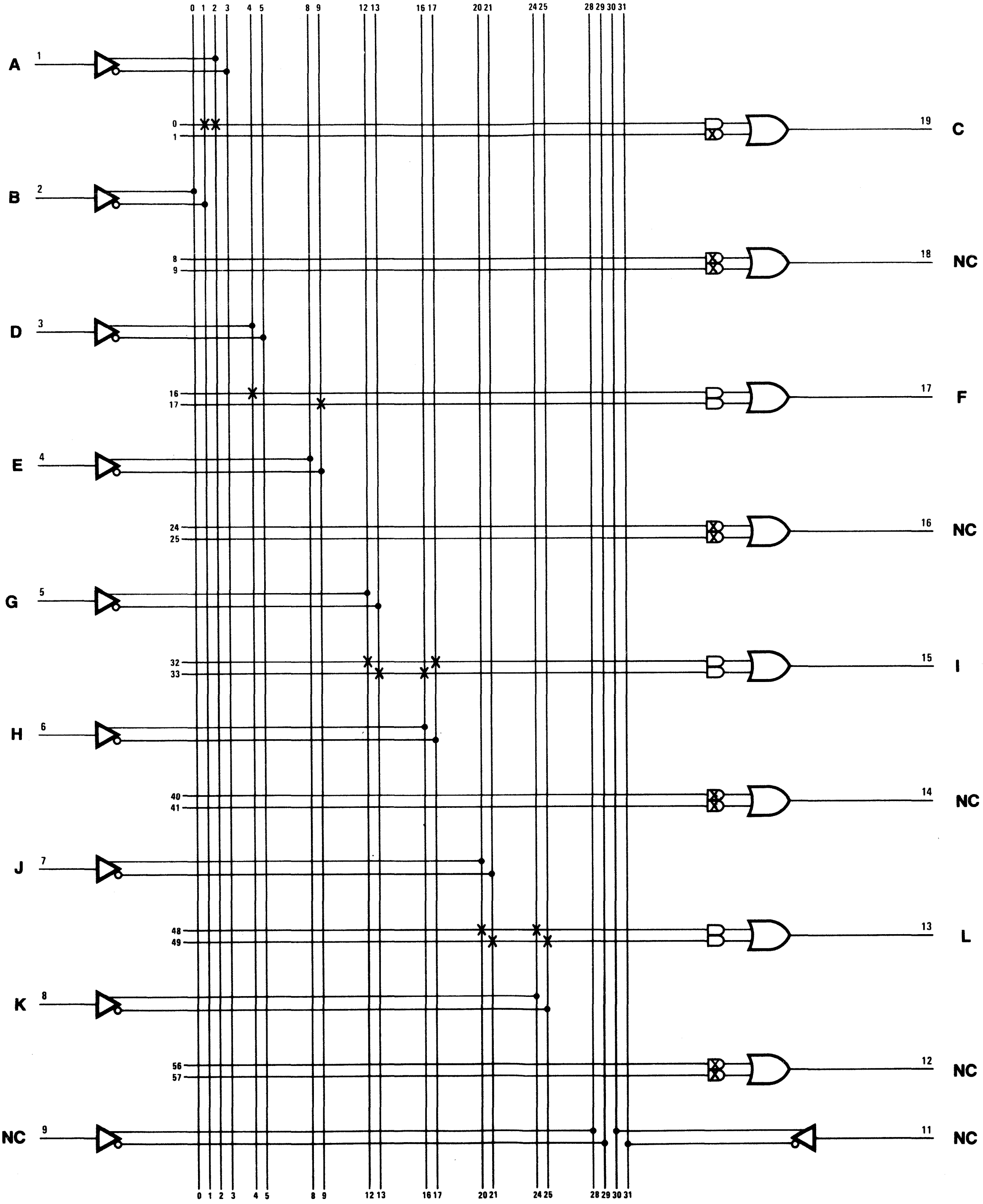
PAL10H8



Logic Symbol

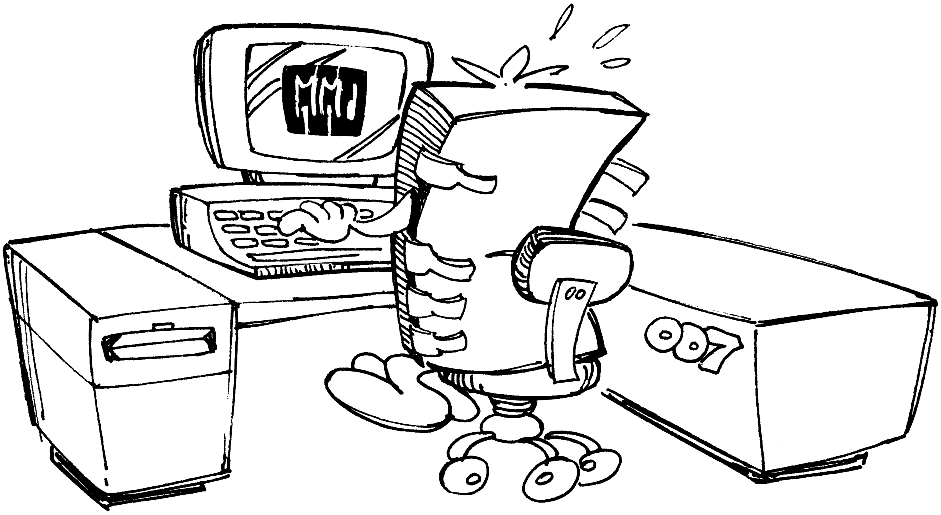
Example Gates No. 2

Logic Diagram PAL10H8



Applications

Control Store Sequencer



Solutions to Control Store Sequencing are as varied as the problems that are solved by micro-programmed hardware. Where an engineer goes to his "ALU Book" to choose the appropriate device for his design, he is much less likely to use his "Sequencer Book" to select that function. Rather, he builds the function from standard MSI and SSI devices. Devices from the "Sequencer Book" tend to require very horizontal control store structures, however they lack the speed which a designer is usually trying to achieve with a horizontal control store.

The sequencer described in this application is designed for use with a vertical control store structure. The vertical control store has narrow control fields and may share field functions to increase field use efficiency. Designs that are built around vertical control stores are useful in applications where events occur at intervals in the microsecond range.

Control Store Sequencer

Logic Schematic

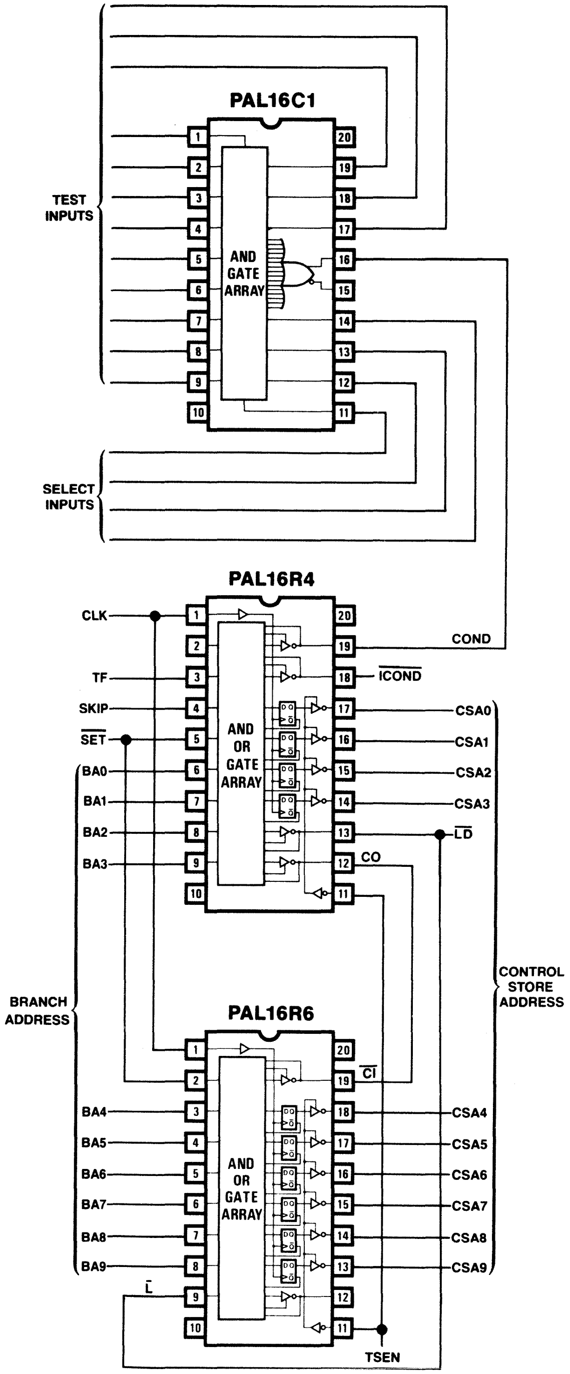


Figure 1

Functional Description

This Control Store Sequencer is designed to use a minimum of control bits while providing sufficient sequencing flexibility. Only three bits are required for the basic sequencer control. The three bits combine to perform the following operations:

CSA = CSA + 1	Increment the control store address;
CSA = CSA + 2	Skip the next control store address; this is a simple form of branch capability; or
CSA = BA	Load a branch address.

The three control bits are SKIP, COND, and TF. SKIP defines whether the sequencer will skip or load. COND is the condition which is tested to determine if the sequencer executes the operation defined by SKIP; and TF defines whether COND is tested true or false. Table 1 defines the sequencer operation.

SKIP	COND	TF	OPERATION
0	0	0	Load
0	0	1	Increment
0	1	0	Increment
0	1	1	Load
1	0	0	Skip
1	0	1	Increment
1	1	0	Increment
1	1	1	Skip

Table 1.

There are two additional control bits which are left to the user's discretion. They are $\overline{\text{SET}}$ and $\overline{\text{TSEN}}$. $\overline{\text{SET}}$ is a synchronous preset which is typically used as a power-on set; however it may also be used as a one-bit vector to the last addressable location during normal operation. $\overline{\text{TSEN}}$ is the enable for the three-state outputs. This has several possible uses, such as a method of testing the hardware. The sequencer outputs are disabled and a test address is supplied from an external source.

The ten-bit sequencer is divided into two parts. The least significant four bits is constructed from a PAL16R4 and is the heart of the skip operation. During the skip operation the state of the least significant bit is maintained and the next three bits function as a three-bit binary counter, while during the increment operation the least significant four bits function as a four-bit binary counter. Carry out ($\overline{\text{CO}}$) is generated during skip when CSA1 thru CSA3 equal 1, and during increment when CSA0 thru CSA3 equal 1. $\overline{\text{LD}}$ is also generated by the least significant part and is a function of SKIP, TF, and COND. If SKIP equal one, $\overline{\text{LD}}$ equal one; if SKIP equal zero, see Table 2.

The most significant six-bits is constructed from a PAL16R6 and is merely a six-bit binary counter with carry in (CI), synchronous load ($\overline{\text{LD}}$) and synchronous set ($\overline{\text{SET}}$). There is also an extra pin which may be used to generate carryout if it is desirable to expand beyond ten-bits.

TF	COND	$\overline{\text{LD}}$
0	0	0
0	1	1
1	0	1
1	1	0

Table 2.

System Integration

One of the first features that is desirable from a system standpoint is the expansion of the COND input, so that more than one condition is available for testing. This is accomplished nicely by using a PAL16C1 as a multiplexer. Four terms are used as input selects. This leaves twelve terms which are used for condition inputs. At first this may seem wasteful as the four select terms can decode sixteen inputs. This is only superficial as the PAL allows the designer a degree of flexibility not found in a standard mux. One COND output should be either TRUE or FALSE in order to generate unconditional increments, skips, and branches. TRUE or FALSE needs only to be a function of the four select terms and does not require a condition input to be grounded or pulled up to V_{CC}. Other functions which the PAL multiplexer performs nicely is the AND, OR, or EXCLUSIVE OR of the condition inputs; functions which must be done externally with a standard multiplexer.

A function which at first glance appears to be missing is subroutine capability. The figure shows how the sequencer can be integrated into a system to provide subroutine capability.

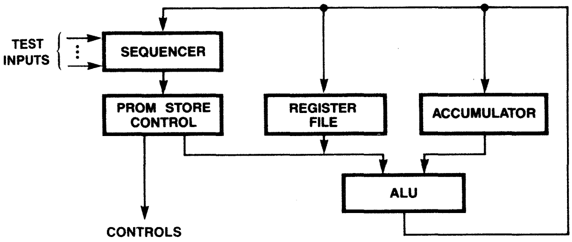


Figure 2

In Figure 2 the same control store field is used to generate literals and branch addresses. Subroutining is accomplished by loading the return address in the register file before the subroutine jump is taken and then reading the return address out of the file when the subroutine return is executed.

Conclusion

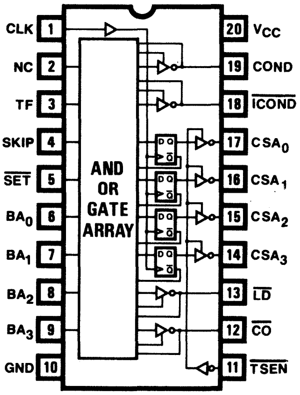
The PAL SEQUENCER is an example of the flexibility which previously was achieved only by the design of custom devices. In the design shown here, SKIP is just as easily defined as CSA = CSA + 4. The SKIP function can also be redefined to be a short branch; retain the state of the upper six bits and load the least significant four bits. The challenge of the PAL Family is not the integration of standard MSI and SSI functions but the direct implementation of system functions.

Control Store Sequencer, Least Significant Stage

Design Specification PAL16R4

```
PAL16R4                                    PAL DESIGN SPECIFICATION
PAT0028                                       BILL BLACK 12/14/77
CONTROL STORE SEQUENCER, LEAST SIGNIFICANT STAGE

CLK NC TF SKIP /SET BA0 BA1 BA2 BA3 GND /TSEN /CO /LD
CSA3 CSA2 CSA1 CSA0 /ICOND COND VCC



/CSA0 := /SET*/ICOND*CSA0 + /SET*ICOND*SKIP*/CSA0 + /SET*ICOND*/SKIP*/BA0


/CSA1 := /SET*CSA0*CSA1*/ICOND + /SET*/ICOND*/CSA0*/CSA1
       + /SET*ICOND*SKIP*CSA1 + /SET*ICOND*/SKIP*/BA2


/CSA2 := /SET*/ICOND*CSA0*CSA1*CSA2 + /SET*/ICOND*/CSA0*/CSA2
       + /SET*/LD*/CSA1*/CSA2 + /SET*/ICOND*SKIP*CSA1*CSA2
       + /SET*ICOND*/SKIP*/BA2


/CSA3 := /SET*/ICOND*CSA0*CSA1*CSA2*CSA3
       + /SET*/ICOND*/CSA0*/CSA3 + /SET*/LD*/CSA1*/CSA3
       + /SET*/LD*/CSA2*/CSA3 + /SET*ICOND*SKIP*CSA1*CSA2*CSA3
       + /SET*ICOND*/SKIP*/BA3

IF(VCC)  CO = CSA0*CSA1*CSA2*CSA3
            + TF*COND*SKIP*CSA1*CSA2*CSA3
            + /TF*/COND*SKIP*CSA1*CSA2*CSA3


IF(VCC)  ICOND = TF*COND + /TF*/COND


IF(VCC)  LD = TF*COND*/SKIP + /TF*/COND*/SKIP


DESCRIPTION:  SEE TEXT
```

PAL16R4



Logic Symbol

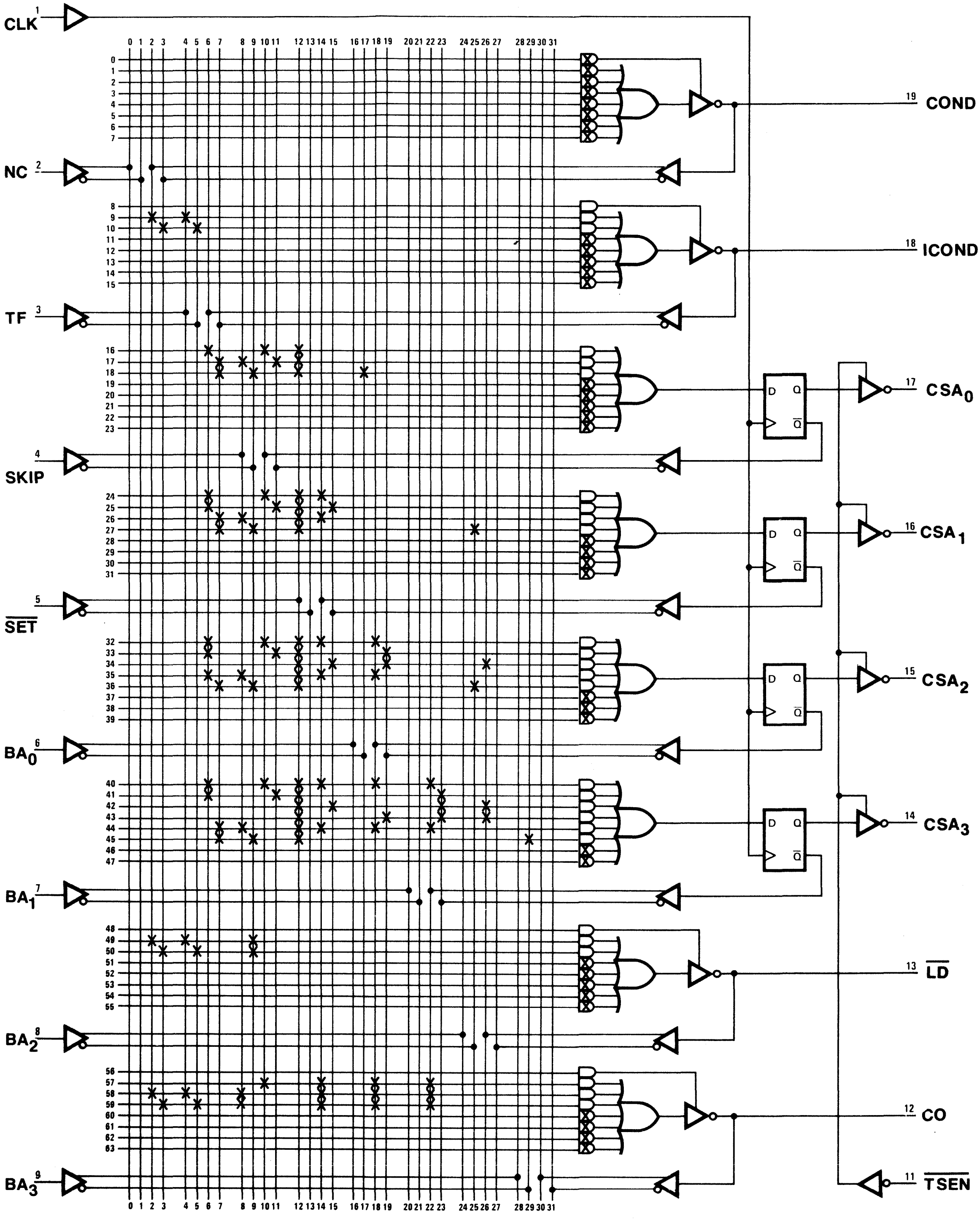
Control Store Sequencer, Least Significant Stage

Fuse Pattern PAL16R4

```
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---- ---- ---- ---- ---- ---- ---- ----
--X- X--- ---- ---- ---- ---- ---- ---- TF◆COND
---X -X-- ---- ---- ---- ---- ---- ---- /TF◆/COND
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---- --X- --X- X--- ---- ---- ---- ---- /SET◆/ICOND◆CSA0
---- ---X X--X X--- ---- ---- ---- ---- /SET◆ICOND◆SKIP◆/CSA0
---- ---X -X-- X--- -X-- ---- ---- ---- /SET◆ICOND◆/SKIP◆/BA0
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---- --X- --X- X-X- ---- ---- ---- ---- /SET◆CSA0◆CSA1◆/ICOND
---- --X- ---X X--X ---- ---- ---- ---- /SET◆/ICOND◆/CSA0◆/CSA1
---- ---X X--- X-X- ---- ---- ---- ---- /SET◆ICOND◆SKIP◆CSA1
---- ---X -X-- X--- ---- ---- -X-- ---- /SET◆ICOND◆/SKIP◆/BA2
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---- --X- --X- X-X- --X- ---- ---- ---- /SET◆/ICOND◆CSA0◆CSA1◆CSA2
---- --X- ---X X--- ---X ---- ---- ---- /SET◆/ICOND◆/CSA0◆/CSA2
---- ---- ---- X--X ---X ---- --X- ---- /SET◆/LD◆/CSA1◆/CSA2
---- --X- X--- X-X- --X- ---- ---- ---- /SET◆/ICOND◆SKIP◆CSA1◆CSA2
---- ---X -X-- X--- ---- ---- -X-- ---- /SET◆ICOND◆/SKIP◆/BA2
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---- --X- --X- X-X- --X- --X- ---- ---- /SET◆/ICOND◆CSA0◆CSA1◆CSA2◆CSA3
---- --X- ---X X--- ---- ---X ---- ---- /SET◆/ICOND◆/CSA0◆/CSA3
---- ---- ---- X--X ---- ---X --X- ---- /SET◆/LD◆/CSA1◆/CSA3
---- ---- ---- X--- ---X ---X --X- ---- /SET◆/LD◆/CSA2◆/CSA3
---- ---X X--- X-X- --X- --X- ---- ---- /SET◆ICOND◆SKIP◆CSA1◆CSA2◆CSA3
---- ---X -X-- X--- ---- ---- ---- -X-- /SET◆ICOND◆/SKIP◆/BA3
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---- ---- ---- ---- ---- ---- ---- ----
--X- X--- -X-- ---- ---- ---- ---- ---- TF◆COND◆/SKIP
---X -X-- -X-- ---- ---- ---- ---- ---- /TF◆/COND◆/SKIP
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---- ---- ---- ---- ---- ---- ---- ----
---- ---- --X- --X- --X- --X- ---- ---- CSA0◆CSA1◆CSA2◆CSA3
--X- X--- X--- --X- --X- --X- ---- ---- TF◆COND◆SKIP◆CSA1◆CSA2◆CSA3
---X -X-- X--- --X- --X- --X- ---- ---- /TF◆/COND◆SKIP◆CSA1◆CSA2◆CSA3
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
```

Control Store Sequencer, Least Significant Stage

Logic Diagram PAL16R4



Control Store Sequencer, Most Significant Stage

Design Specification PAL16R6

```
PAL16R6                                          PAL DESIGN SPECIFICATION
PAT0029                                              BILL BLACK 12/14/77
CONTROL STORE SEQUENCER, MOST SIGNIFICANT STAGE

CLK /SET BA4 BA5 BA6 BA7 BA8 BA9 /LD GND /TSEN NC CSA9 CSA8 CSA7 CSA6
CSA5 CSA4 /CI VCC


/CSA4 := /SET*CSA4*CI*/LD + /CSA4*/CI*/LD*/SET + /SET*LD*/BA4


/CSA5 := /SET*CSA4*CSA5*CI*/LD + /SET*/CSA5*/CI*/LD
         + /SET*/CSA4*/CSA5*/LD + /SET*LD*/BA5


/CSA6 := /SET*CSA4*CSA5*CSA6*CI*/LD + /SET*/CSA6*/CI*/LD
         + /CSA4*/CSA6*/LD*/SET + /SET*/CSA5*/CSA6*/LD
         + /SET*LD*/BA6


/CSA7 := /SET*CSA4*CSA5*CSA6*CSA7*CI*/LD
         + /SET*/CSA7*/CI*/LD + /SET*/CSA4*/CSA7*/LD
         + /SET*/CSA5*/CSA7*/LD + /SET*/CSA6*/CSA7*/LD
         + LD*/BA7*/SET


/CSA8 := /SET*CSA4*CSA5*CSA6*CSA7*CSA8*CI*/LD
         + /SET*/CSA8*/CI*/LD + /SET*/CSA4*/CSA8*/LD
         + /SET*/CSA5*/CSA8*/LD + /SET*/CSA6*/CSA8*/LD
         + /SET*/CSA7*/CSA8*/LD + /SET*LD*/BA8


/CSA9 := /SET*CSA4*CSA5*CSA6*CSA7*CSA8*CSA9*CI*/LD
         + /SET*/CSA9*/CI*/LD + /SET*/CSA4*/CSA9*/LD
         + /SET*/CSA5*/CSA9*/LD + /SET*/CSA6*/CSA9*/LD
         + /SET*/CSA7*/CSA9*/LD + /SET*/CSA8*/CSA9*/LD
         + /SET*LD*/BA9


DESCRIPTION:

THE 6-BIT COUNTER INCREMENTS WHEN THE /LD LINE IS HIGH
IF CARRY AND /SET.  THE OUTPUTS ARE ENABLED WHEN /TSEN IS LOW.


FUNCTION TABLE:

-------------------------------------------------------
! /SET ! CI !  /LD  ! CLK !    CSA     ! OPERATION   !
-------------------------------------------------------
!   L  !  X !   X   ! L-H ! ALL HIGH   !  SET        !
!   H  !  X !   L   ! L-H ! BA         !  LD         !
!   H  !  H !   H   ! L-H ! CSA        !  NOP        !
!   H  !  L !   H   ! L-H ! CSA PLUS 1!  INCR        !
-------------------------------------------------------
```

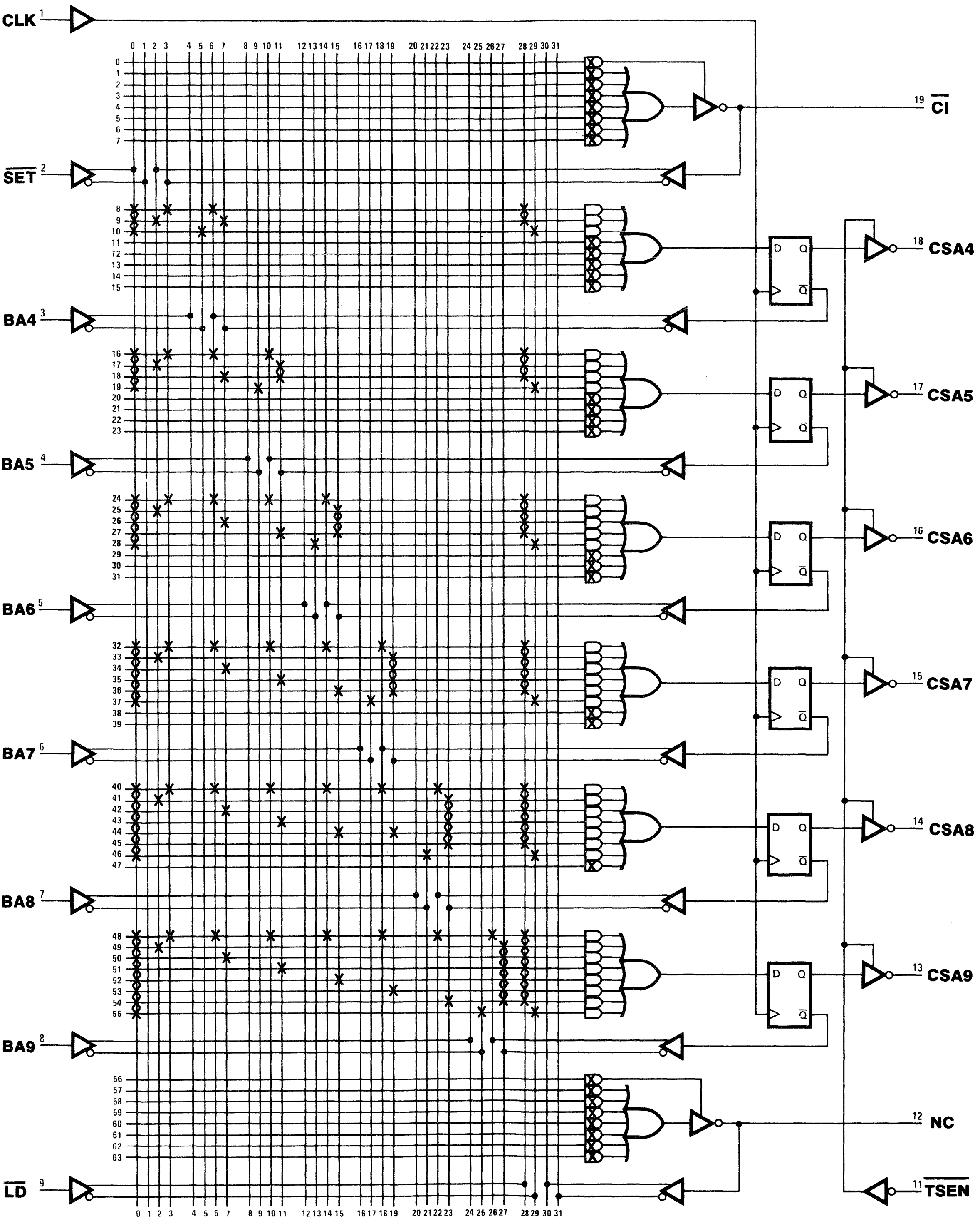
Control Store Sequencer, Most Significant Stage

Fuse Pattern PAL16R6

```
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

X--X --X- ---- ---- ---- ---- ---- X--- /SET◆CSA4◆CI◆/LD
X-X- ---X ---- ---- ---- ---- ---- X--- /CSA4◆/CI◆/LD◆/SET
X--- -X-- ---- ---- ---- ---- ---- -X-- /SET◆LD◆/BA4
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

X--X --X- --X- ---- ---- ---- ---- X--- /SET◆CSA4◆CSA5◆CI◆/LD
X-X- ---- ---X ---- ---- ---- ---- X--- /SET◆/CSA5◆/CI◆/LD
X--- ---X ---X ---- ---- ---- ---- X--- /SET◆/CSA4◆/CSA5◆/LD
X--- ---- -X-- ---- ---- ---- ---- -X-- /SET◆LD◆/BA5
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

X--X --X- --X- --X- ---- ---- ---- X--- /SET◆CSA4◆CSA5◆CSA6◆CI◆/LD
X-X- ---- ---- ---X ---- ---- ---- X--- /SET◆/CSA6◆/CI◆/LD
X--- ---X ---- ---X ---- ---- ---- X--- /CSA4◆/CSA6◆/LD◆/SET
X--- ---- ---X ---X ---- ---- ---- X--- /SET◆/CSA5◆/CSA6◆/LD
X--- ---- ---- -X-- ---- ---- ---- -X-- /SET◆LD◆/BA6
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

X--X --X- --X- --X- --X- ---- ---- X--- /SET◆CSA4◆CSA5◆CSA6◆CSA7◆CI◆/LD
X-X- ---- ---- ---- ---X ---- ---- X--- /SET◆/CSA7◆/CI◆/LD
X--- ---X ---- ---- ---X ---- ---- X--- /SET◆/CSA4◆/CSA7◆/LD
X--- ---- ---X ---- ---X ---- ---- X--- /SET◆/CSA5◆/CSA7◆/LD
X--- ---- ---- ---X ---X ---- ---- X--- /SET◆/CSA6◆/CSA7◆/LD
X--- ---- ---- ---- -X-- ---- ---- -X-- LD◆/BA7◆/SET
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

X--X --X- --X- --X- --X- --X- ---- X--- SET◆CSA4◆CSA5◆CSA6◆CSA7◆CSA8◆CI
X-X- ---- ---- ---- ---- ---X ---- X--- /SET◆/CSA8◆/CI◆/LD
X--- ---X ---- ---- ---- ---X ---- X--- /SET◆/CSA4◆/CSA8◆/LD
X--- ---- ---X ---- ---- ---X ---- X--- /SET◆/CSA5◆/CSA8◆/LD
X--- ---- ---- ---X ---- ---X ---- X--- /SET◆/CSA6◆/CSA8◆/LD
X--- ---- ---- ---- ---X ---X ---- X--- /SET◆/CSA7◆/CSA8◆/LD
X--- ---- ---- ---- ---- -X-- ---- -X-- /SET◆LD◆/BA8
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

X--X --X- --X- --X- --X- --X- --X- X--- /SET◆CSA4◆CSA5◆CSA6◆CSA7◆CSA8◆C
X-X- ---- ---- ---- ---- ---- ---X X--- /SET◆/CSA9◆/CI◆/LD
X--- ---X ---- ---- ---- ---- ---X X--- /SET◆/CSA4◆/CSA9◆/LD
X--- ---- ---X ---- ---- ---- ---X X--- /SET◆/CSA5◆/CSA9◆/LD
X--- ---- ---- ---X ---- ---- ---X X--- /SET◆/CSA6◆/CSA9◆/LD
X--- ---- ---- ---- ---X ---- ---X X--- /SET◆/CSA7◆/CSA9◆/LD
X--- ---- ---- ---- ---- ---X ---X X--- /SET◆/CSA8◆/CSA9◆/LD
X--- ---- ---- ---- ---- ---- -X-- -X-- /SET◆LD◆/BA9

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
```

Control Store Sequencer, Most Significant Stage

Logic Diagram PAL16R6



Applications

Memory Interface Logic for 6800 microprocessor Bus

Most microcomputer system designs require the use of read/write and read only memory. The logic required to decode the microprocessor control signals into those signals required by the memory can be easily generated using a single PAL. This example shows how this logic is implemented for a read/write memory for the M6800 microcomputer. With minor modifications this logic can be easily extended to most other available microprocessors.

M6800 Memory Interface

Functional Description

The M6800 microprocessor is interfaced to memories by decoding the system memory address bus and several system control lines to generate the required memory control signals. This function is normally performed by combinatorial logic. In many applications, however, the PAL provides a more effective solution.

Circuit Operation

The logic schematic shown in Figure 1 is typical of most M6800 memory interfaces. The circuit shown is a 2048 x 8 bit static memory organized as four 1k x 4 bit RAM chips. The inputs to the RAM are the 10 memory address lines to select the individual memory location, the read/write line to determine whether data is to be read from or written into the memory, and the chip enable line to allow the device to perform the requested data transfer. Data to be written into the memory must be stable on the system data bus when the write signal is given. Data read from the memory will be placed on the data bus within one memory access time after the address has been decoded and the read signal given.

The circled area of combinatorial logic in Figure 1 is used to decode the 6800 address and control signals. Address bits 0-9 are routed directly to all four memory chips. Bit 10 is used to select whether chips 0 and 1 or chips 2 and 3 are to be selected. Bits 12-15 are connected to the A inputs of a digital comparator whose B inputs are jumpered to select the memory page address. If the memory is to be located from 0-800H (the first 2k page in the memory), all four comparator B inputs would be grounded. Then, whenever an address with bits 12-15 low appeared on the bus, a match would occur and the memory would be selected. Changing the jumpers allows the memory to be used anywhere in the 6800's address space or allows the use of multiple cards to construct a larger memory.

The read/write control logic for the memory is generated by decoding the read/write (R/W), phase 2 clock (Phase2), and valid memory address (Vphase2) system control lines. A logic high on the R/W line indicates a memory read; a logic high indicates a memory write.

The valid memory address line (Vphase2) is used to enable the address decoder output. When this enable occurs, the state of the address select and read/write logic is established. Then, when the phase 2 clock goes high, the memory transfer is performed. The relationship between the signals for both read and write operations is shown in Figure 2. (Consult 6800 data book for detailed design information.)

PAL Implementation

All of the combinatorial logic in the circled area of figure 1 can be replaced by a single PAL. This will lower system cost by reducing the device package count and lowering P.C. board area. The logic section has eight input terms and six output terms. Referring to the PAL family table it is seen that the PAL 10L8 fits the task nicely. The only tricky part of the transition from combinatorial logic to a PAL is encountered in the address decoding section. In the original circuit the decoder is jumpered with a match address for use during the address comparison. With the PAL, the address is programmed directly into the gate array.

The general logic equations for the decoder are as follows:

$$\overline{\text{WEOE0}} = \text{VPHASE2} \cdot \overline{\text{A10}} \cdot \overline{\text{A12}} \cdot \overline{\text{A13}} \cdot \overline{\text{A14}} \cdot \overline{\text{A15}} \cdot \text{PHASE2} \cdot \overline{\text{RW}}$$

$$\overline{\text{WEOE1}} = \text{VPHASE2} \cdot \text{A10} \cdot \overline{\text{A12}} \cdot \overline{\text{A13}} \cdot \overline{\text{A14}} \cdot \overline{\text{A15}} \cdot \text{PHASE2} \cdot \overline{\text{RW}}$$

$$\text{CSOD0} = \overline{\text{A10}} \cdot \overline{\text{A12}} \cdot \overline{\text{A13}} \cdot \overline{\text{A14}} \cdot \overline{\text{A15}} \cdot \text{PHASE2}$$

$$\text{CSOD1} = \text{A10} \cdot \overline{\text{A12}} \cdot \overline{\text{A13}} \cdot \overline{\text{A14}} \cdot \overline{\text{A15}} \cdot \text{PHASE2}$$

$$\text{CE0} = \overline{\text{CSOD0}}$$

$$\text{CE1} = \overline{\text{CSOD1}}$$

The above equations show the decoder set for page 0 (0-800H), but this could be easily changed by modifying the address terms. Note that the CE0 and CE1 terms can either be derived directly or by feeding the CSOD0 and CSOD1 terms back into the PAL as inputs.

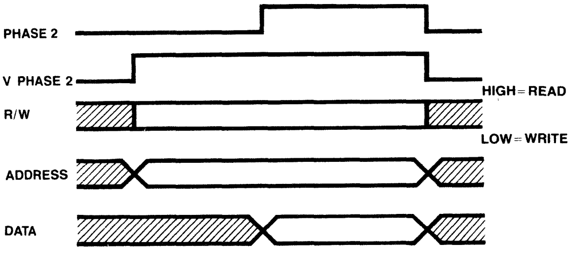


Figure 2

Conclusion

The PAL makes an effective direct logic replacement in many combinatorial logic applications. This can make both new and old designs more cost effective. In this example, the PAL both lowers package count and increases circuit reliability in a typical microcomputer memory application. These advantages can be easily extended to similar designs.

Memory Interface Logic for 6800 Microprocessor Bus

Logic Schematic

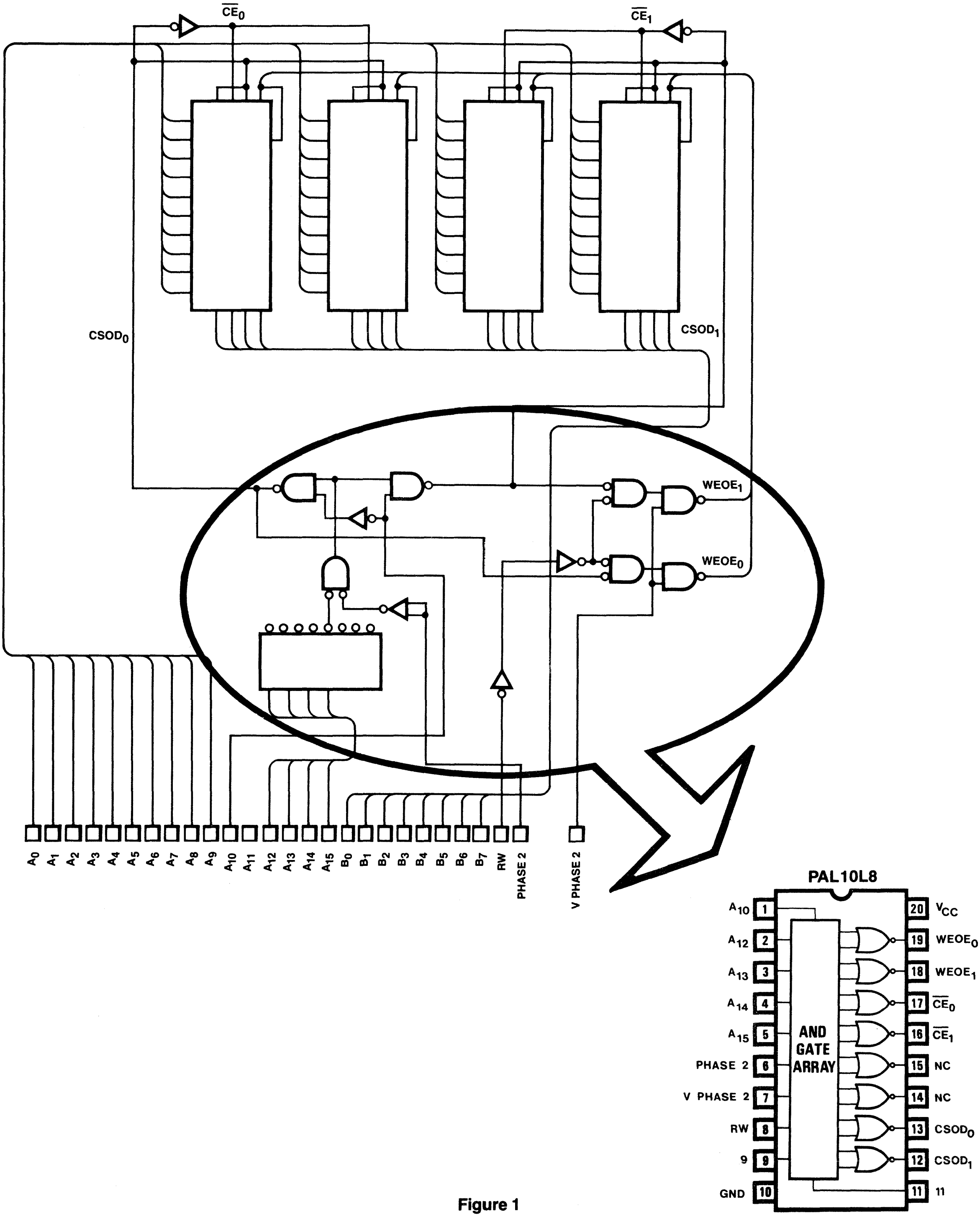
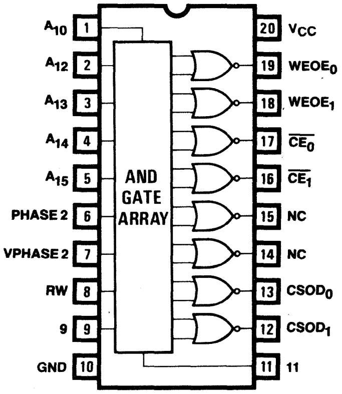


Figure 1

Memory Interface Logic for 6800 Microprocessor Bus   Design Specification PAL10L8

```
PAL10L8                                            PAL DESIGN SPECIFICATION
PAT0011                                                JOHN BIRKNER 12/7/77
MEMORY INTERFACE LOGIC FOR 6800 MICROPROCESSOR BUS

A10 A12 A13 A14 A15 PHASE2 VPHASE2 RW 9 GND
11 CSOD1 CSOD0 NC NC /CE1 /CE0 WEOE1 WEOE0 VCC




/WEOE0 = VPHASE2*/A10*/A12*/A13*A14*/A15*PHASE2*/RW

/WEOE1 = VPHASE2*A10*/A12*/A13*A14*A15*PHASE2*/RW

CE0    = 9

CE1    = 11


CSOD0  = /A10*/A12*/A13*/A14*/A15*PHASE2

CSOD1  =  A10*/A12*/A13*A14*/A15*PHASE2

DESCRIPTION:

THIS DEVICE PROVIDES THE INTERFACE LOGIC BETWEEN A 6800 MICROPROCESSOR
BUS AND FOUR STATIC 4K MEMORY CHIPS.  ADDRESS BUS, RW, PHASE2 AND
VPHASE2 ARE DECODED TO PRODUCE THE PROPER WRITE ENABLE, CHIP ENABLE
AND OUTPUT DISABLE SIGNALS FOR MEMORY DATA TRANSFERS.  /CE0 AND /CE1
ARE THE COMPLEMENTS OF CSOP0 AND CSOD1, RESPECTIVELY BY EXTERNAL
CONNECTIONS CSOD0 - PIN 9  AND  CSOD1 - PIN 11.
```

PAL10L8



Logic Symbol

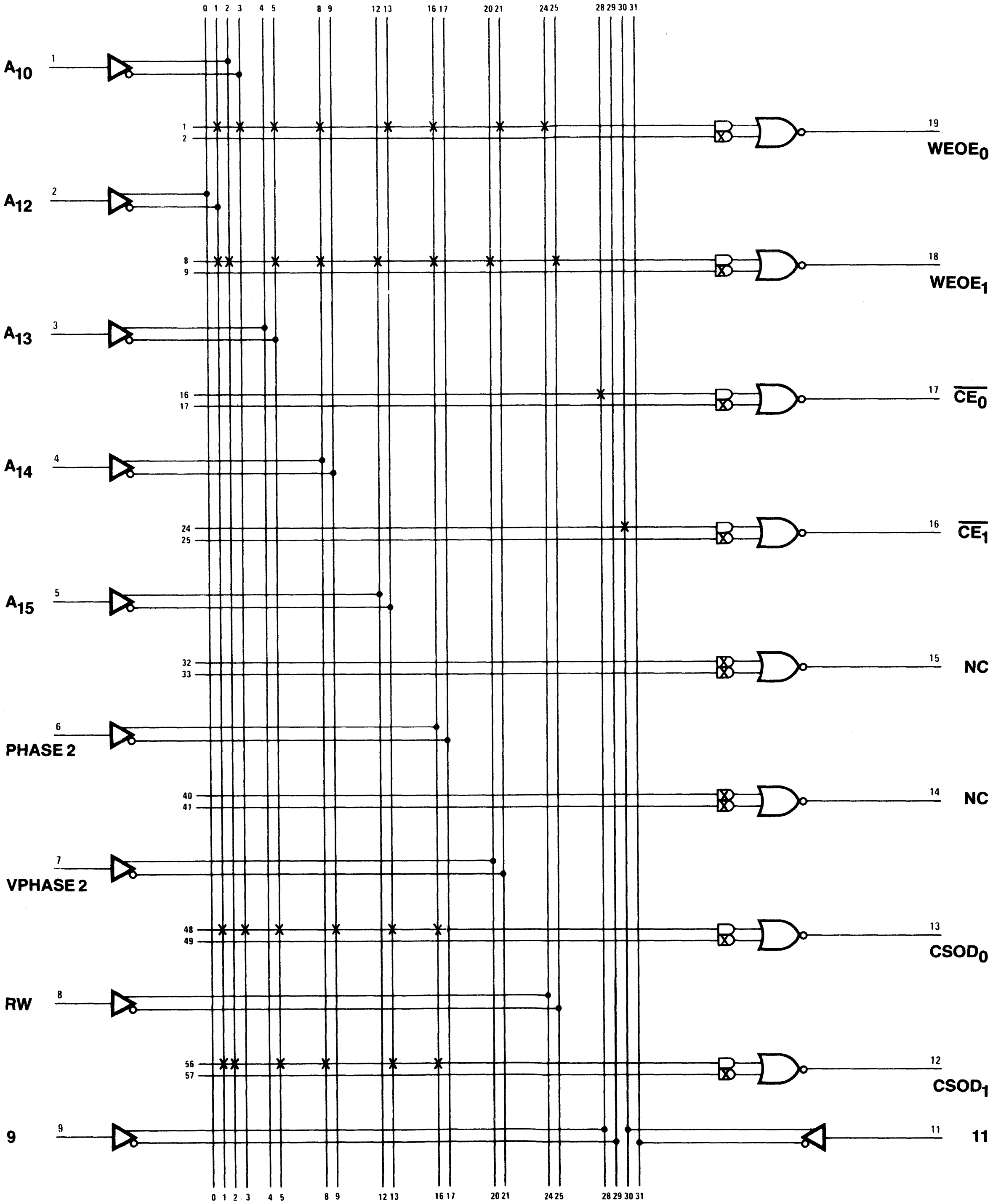
Memory Interface Logic for 6800 Microprocessor Bus

Fuse Pattern PAL10L8

```
MEMORY INTERFACE LOGIC FOR 6800 MICROPROCESSOR BUS

-X-X -X-- X--- -X-- X--- X--- -X-- ---- VPHASE2♦/A10♦/A12♦/A13♦A14♦/A15
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX

-XX- -X-- X--- X--- X--- X--- -X-- ---- VPHASE2♦A10♦/A12♦/A13♦A14♦A15♦P
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX

---- ---- ---- ---- ---- ---- ---- X--- 9
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX

---- ---- ---- ---- ---- ---- ---- --X- 11
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX

XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX

XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX

-X-X -X-- -X-- -X-- X--- ---- ---- ---- /A10♦/A12♦/A13♦/A14♦/A15♦PHASE2
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX

-XX- -X-- X--- -X-- X--- ---- ---- ---- A10♦/A12♦/A13♦A14♦/A15♦PHASE2
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
XXXX XX-- XX-- XX-- XX-- XX-- XX-- XXXX
```

Memory Interface Logic for 6800 Microprocessor Bus

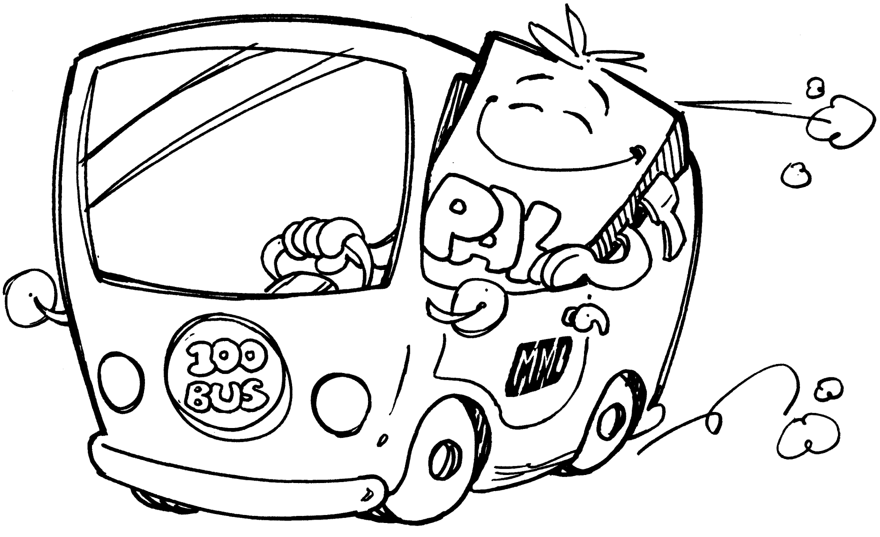
Logic Diagram PAL10L8



5

Applications

Dual I/O Port Decoder for S-100 Bus



The S-100 bus is an informal standard bus structure used in many hobbyist computers. It has, however, proved useful and cost effective enough to find its way into many commercial applications. S-100 bus computers are based in the 8080/Z-80 family microcomputers, and this example shows how the S-100 bus signals can be decoded and used to control two bi-directional I/O ports in one of these systems.

Dual I/O Port Decoder for the S100 Bus

Functional Description

The S-100 bus compatible family of computers are based on the 8080/Z-80 family of microcomputers. For I/O operations these computers place an 8 bit I/O address on the high and low order 8 bits of the 16 bit wide system memory address bus. This system allows up to 256 direct I/O devices in any system. The I/O addresses and the system control signals are decoded by I/O devices to initiate and control data transfer operations. The PAL provides an effective means of managing this type of data transfer.

Circuit Description

The circuit shown in Figure 1 is a complete bi-directional two input, two output port interface for an S-100 bus computer. The PAL is used to decode the I/O addresses and the I/O control signals.

For input operations data is gated from the selected input port (A or B) onto the system data bus. For output operations data is gated from the system data bus to the selected output port.

The data I/O circuitry consists of four octal registers and two three state octal buffers connected together by an internal eight bit bi-directional data bus. For input operations the data is latched into the octal register when the device strobes the latch clock. When a read option addresses that input port, the register three state enable is clocked and the data is gated from the register, through the octal buffer and onto the S-100 data bus. For output operations an output address match causes the data to be gated off the S-100 data bus, through the octal buffer and latched into the octal output register. It will remain latched and available for reading until new data is written to that address. I/O devices can read the data by enabling the octal register's three state enable line.

PAL Implementation

The PAL is used to monitor the bus I/O address and control signals. Address lines A0-A7 provide the address of the I/O port being accessed. The PDBIN signal is used to select the direction of data transfer: a logic high indicates an input operation, while a logic low indicates an output operation. The SINP signal indicates the bus is ready for an input data transfer; SOUT indicates that the data on the data bus is a valid output. The PWR signal indicates that the system bus is correctly powered up; PWR must be low for any data transfers to take place.

The control signals are decoded to produce six signals which operate the I/O logic. AIN and BIN are generated when the logic decode indicates an input from port A or B is to be performed. This will occur when the address programmed for A or B matches the address on the bus and PDBIN and SINP indicate an input operation. Similarly, AOUT and BOUT are generated when the address on the bus matches and PDBIN and SOUT indicate an output operation is being performed.

DIN is generated whenever an address match occurs and an input operation is to be performed. It is used to gate data from the selected input port through the octal buffer and onto the system data bus. DOUT is generated whenever an address match occurs and an output operation is being performed. It is used to gate data from the system data bus to the destination output register.

The logic equations for the control signals are as follows:

$$\overline{\mathrm{AIN}} = \overline{\mathrm{A0}} \cdot \mathrm{A1} \cdot \overline{\mathrm{A2}} \cdot \mathrm{A3} \cdot \overline{\mathrm{A4}} \cdot \mathrm{A5} \cdot \mathrm{A6} \cdot \overline{\mathrm{A7}} \cdot \overline{\mathrm{PWR}} \cdot \mathrm{SINP}$$
$$\overline{\mathrm{AOUT}} = \overline{\mathrm{A0}} \cdot \mathrm{A1} \cdot \overline{\mathrm{A2}} \cdot \mathrm{A3} \cdot \overline{\mathrm{A4}} \cdot \mathrm{A5} \cdot \mathrm{A6} \cdot \overline{\mathrm{A7}} \cdot \overline{\mathrm{PWR}} \cdot \mathrm{SOUT}$$
$$\overline{\mathrm{BIN}} = \mathrm{A0} \cdot \overline{\mathrm{A1}} \cdot \mathrm{A2} \cdot \overline{\mathrm{A3}} \cdot \mathrm{A4} \cdot \mathrm{A5} \cdot \overline{\mathrm{A6}} \cdot \mathrm{A7} \cdot \overline{\mathrm{PWR}} \cdot \mathrm{SINP}$$
$$\overline{\mathrm{BOUT}} = \mathrm{A0} \cdot \overline{\mathrm{A1}} \cdot \mathrm{A2} \cdot \overline{\mathrm{A3}} \cdot \mathrm{A4} \cdot \mathrm{A5} \cdot \overline{\mathrm{A6}} \cdot \mathrm{A7} \cdot \overline{\mathrm{PWR}} \cdot \mathrm{SOUT}$$
$$\overline{\mathrm{DIN}} = \overline{\mathrm{AIN}} + \overline{\mathrm{BIN}}$$
$$\overline{\mathrm{DOUT}} = \overline{\mathrm{AOUT}} + \overline{\mathrm{BOUT}}$$

The above equations are for an A port address of 6A hex and a B port address of B5 hex. By changing terms any desired port address can be obtained.

This application has a total of 12 inputs (A0-A7, PWR, PDBIN, SINP, and SOUT) and six outputs (AIN, AOUT, BIN, BOUT, DIN, and DOUT). The entire function can be performed using a single PAL 12L6.

One advantage of using a PAL in this type of application is that it allows the address of the A and B I/O port pairs to be any eight bit address. As mentioned, in this example the A address is 6A hex and the B address is B5 hex. In a normal combinatorial design of this type the addresses must usually be made contiguous (i.e. A = 80 hex, B = 81 hex) to save decoding logic. The PAL approach offers more flexibility and uses fewer packages.

Conclusion

The PAL combines together with latches and data buffers to make an efficient I/O port decoder and controller. This example has been for the S-100 bus, but the concept can be easily extended to other microcomputers and bus structures.

Dual I/O Port Decoder for the S100 Bus

Logic Schematic PAL12L6

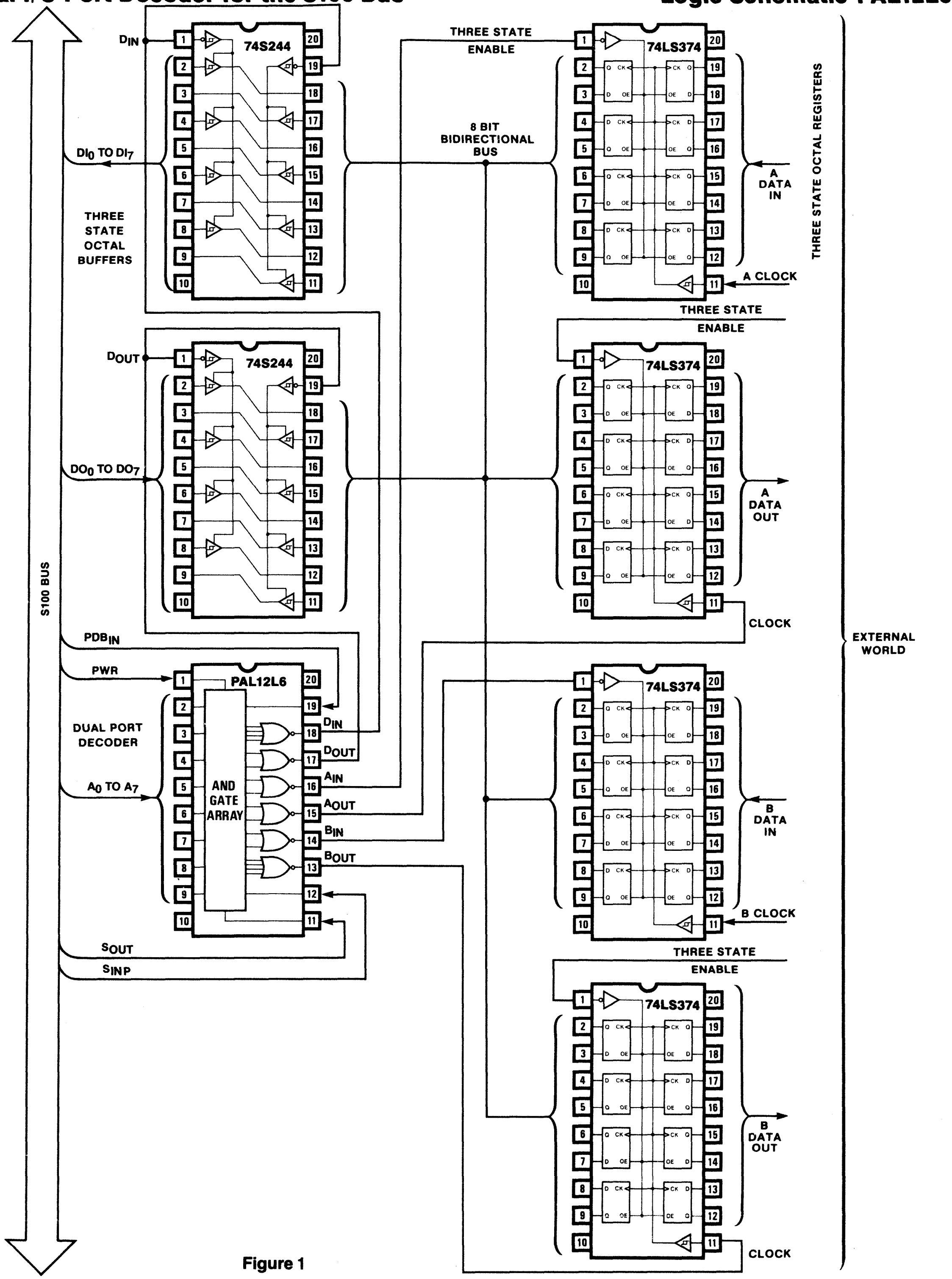
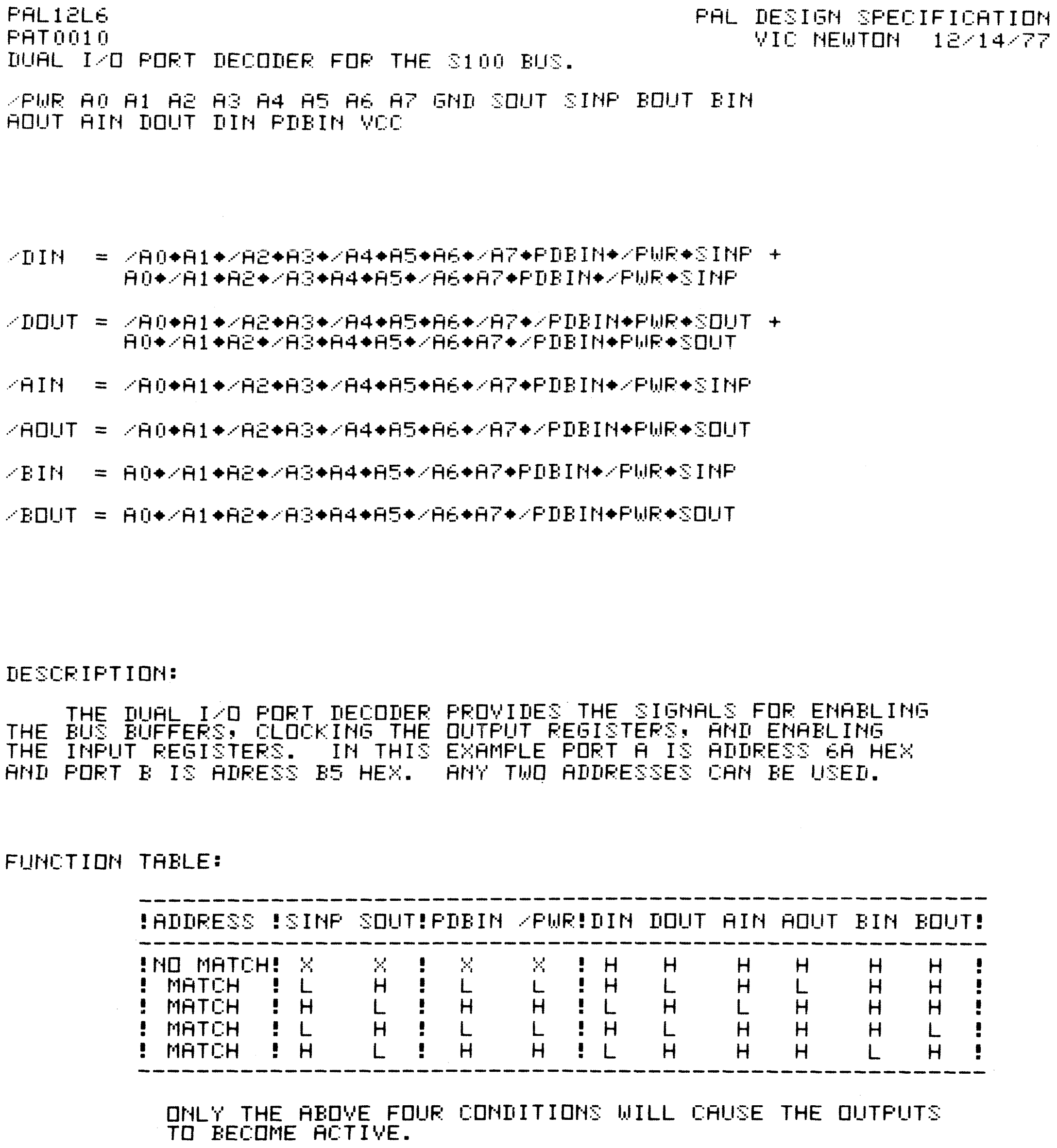


Figure 1

Dual I/O Port Decoder for the S100 Bus

Design Specification PAL12L6



```
PAL12L6                                         PAL DESIGN SPECIFICATION
PAT0010                                             VIC NEWTON  12/14/77
DUAL I/O PORT DECODER FOR THE S100 BUS.

/PWR A0 A1 A2 A3 A4 A5 A6 A7 GND SOUT SINP BOUT BIN
AOUT AIN DOUT DIN PDBIN VCC



/DIN  = /A0*A1*/A2*A3*/A4*A5*A6*/A7*PDBIN*/PWR*SINP +
        A0*/A1*A2*/A3*A4*A5*/A6*A7*PDBIN*/PWR*SINP

/DOUT = /A0*A1*/A2*A3*/A4*A5*A6*/A7*/PDBIN*PWR*SOUT +
        A0*/A1*A2*/A3*A4*A5*/A6*A7*/PDBIN*PWR*SOUT

/AIN  = /A0*A1*/A2*A3*/A4*A5*A6*/A7*PDBIN*/PWR*SINP

/AOUT = /A0*A1*/A2*A3*/A4*A5*A6*/A7*/PDBIN*PWR*SOUT

/BIN  = A0*/A1*A2*/A3*A4*A5*/A6*A7*PDBIN*/PWR*SINP

/BOUT = A0*/A1*A2*/A3*A4*A5*/A6*A7*/PDBIN*PWR*SOUT



DESCRIPTION:

    THE DUAL I/O PORT DECODER PROVIDES THE SIGNALS FOR ENABLING
THE BUS BUFFERS, CLOCKING THE OUTPUT REGISTERS, AND ENABLING
THE INPUT REGISTERS.  IN THIS EXAMPLE PORT A IS ADDRESS 6A HEX
AND PORT B IS ADRESS B5 HEX.  ANY TWO ADDRESSES CAN BE USED.


FUNCTION TABLE:

        ----------------------------------------------------------
        !ADDRESS !SINP SOUT!PDBIN /PWR!DIN DOUT AIN AOUT BIN BOUT!
        ----------------------------------------------------------
        !NO MATCH! X    X  !  X    X  ! H   H    H   H    H   H  !
        ! MATCH  ! L    H  !  L    L  ! H   L    H   L    H   H  !
        ! MATCH  ! H    L  !  H    H  ! L   H    L   H    H   H  !
        ! MATCH  ! L    H  !  L    L  ! H   L    H   H    H   L  !
        ! MATCH  ! H    L  !  H    H  ! L   H    H   H    L   H  !
        ----------------------------------------------------------

           ONLY THE ABOVE FOUR CONDITIONS WILL CAUSE THE OUTPUTS
           TO BECOME ACTIVE.
```

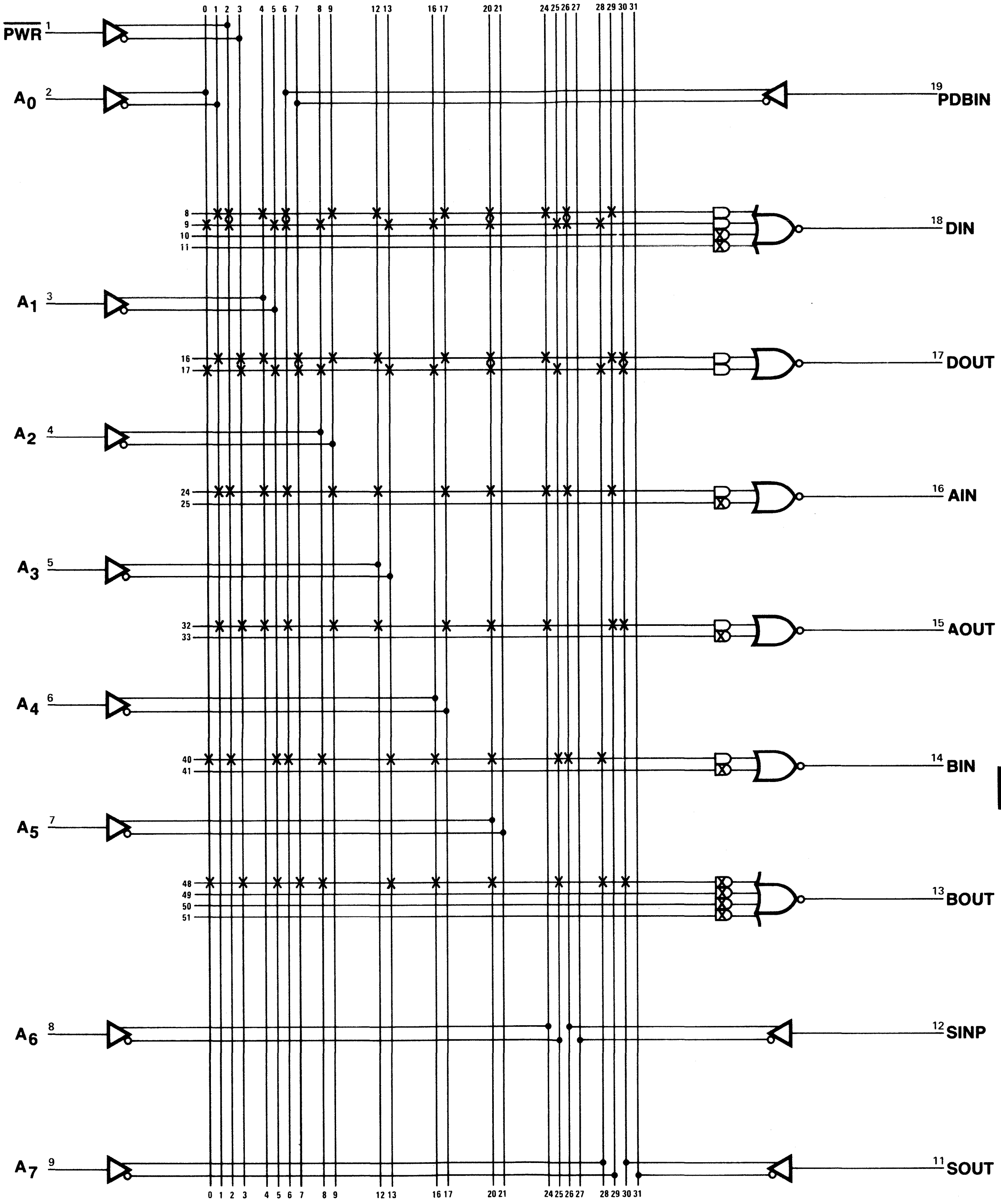
Dual I/O Port Decoder for the S100 Bus

Fuse Pattern PAL12L6

```
DUAL I/O PORT DECODER FOR THE S100 BUS.

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

-XX- X-X- -X-- X--- -X-- X--- X-X- -X-- /A0◆A1◆/A2◆A3◆/A4◆A5◆A6◆/A7◆PDB
X-X- -XX- X--- -X-- X--- X--- -XX- X--- A0◆/A1◆A2◆/A3◆A4◆A5◆/A6◆A7◆PDBI
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX

-X-X X--X -X-- X--- -X-- X--- X--- -XX- /A0◆A1◆/A2◆A3◆/A4◆A5◆A6◆/A7◆/PD
X--X -X-X X--- -X-- X--- X--- -X-- X-X- A0◆/A1◆A2◆/A3◆A4◆A5◆/A6◆A7◆/PDB
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX

-XX- X-X- -X-- X--- -X-- X--- X-X- -X-- /A0◆A1◆/A2◆A3◆/A4◆A5◆A6◆/A7◆PDB
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX

-X-X X--X -X-- X--- -X-- X--- X--- -XX- /A0◆A1◆/A2◆A3◆/A4◆A5◆A6◆/A7◆/PD
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX

X-X- -XX- X--- -X-- X--- X--- -XX- X--- A0◆/A1◆A2◆/A3◆A4◆A5◆/A6◆A7◆PDBI
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX

X--X -X-X X--- -X-- X--- X--- -X-- X-X- A0◆/A1◆A2◆/A3◆A4◆A5◆/A6◆A7◆/PDB
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
```

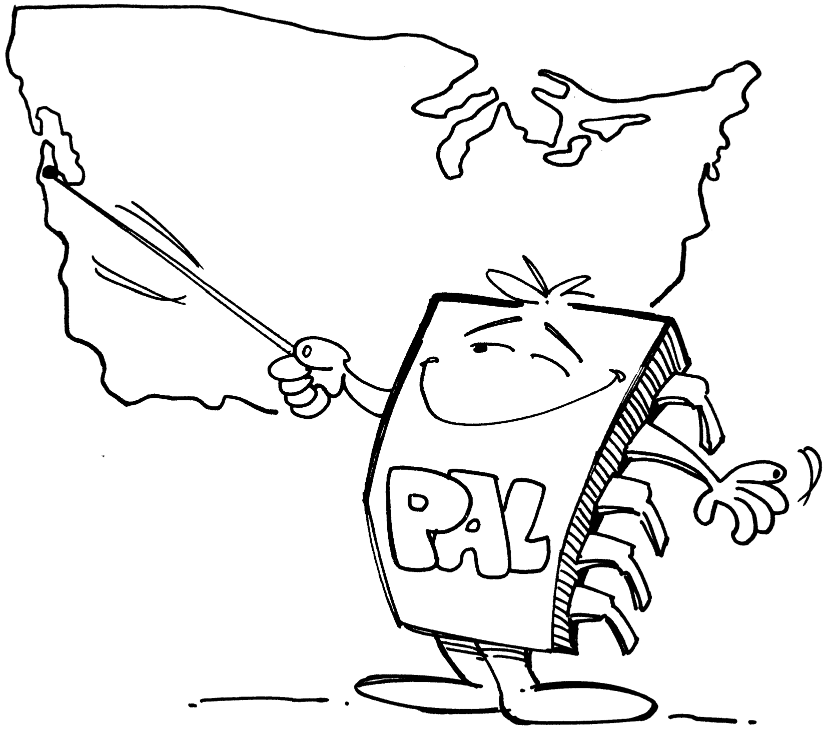
Dual I/O Port Decoder for the S100 Bus

Logic Diagram PAL12L6



Applications

Memory Mapped I/O



Memory mapped I/O is an interface technique which addresses I/O devices as a part of the computer's memory address space. Most computers (particularly microcomputers) provide many more instructions to manipulate memory contents than they have for direct I/O. Therefore, the use of memory mapping can make I/O control much more flexible. PALs can be used to make memory mapped I/O implementation easy and, if different memory addresses are required, the PAL can easily accommodate the changes.

Memory Mapped I/O

Functional Description

Memory mapped I/O interfaces I/O devices to a computer by treating the device's physical address as a memory address. This removes the requirement for special I/O decoding and enhances the flexibility of the I/O system. The PAL provides a simple and direct method for implementing memory mapped I/O in mini and micro computer systems.

Circuit Operation

The circuits shown in Figure 1 are typical of those found in memory mapped I/O applications. The inputs to the decode logic are the system memory address lines, A0-AF. The logic compares the address on the memory bus with the programmed comparison address. When an address on the bus matches, the I/O port enable signal is set. This enable signal can then be used in conjunction with other system control signals to transfer data to and from the system data bus. Other examples in this applications section cover this I/O control decoding in more detail.

PAL Design

The PAL is used to monitor the system memory address bus. Typical microcomputers use a 16 bit address, so fully decoding the I/O addresses for two ports can be accomplished using the PAL 16L2. Partial decoding for a larger number of ports can be performed by other members of the PAL family.

The logic equations for the memory mapped I/O logic are as follows:

$$\begin{aligned}\mathbf{PORT\ 0} = {} & \overline{\mathbf{AB0}} \cdot \overline{\mathbf{AB1}} \cdot \overline{\mathbf{AB2}} \cdot \mathbf{AB3} \cdot \mathbf{AB4} \cdot \mathbf{AB5} \cdot \mathbf{AB6} \cdot \overline{\mathbf{AB7}} \cdot \\ & \mathbf{AB8} \cdot \mathbf{AB9} \cdot \mathbf{ABA} \cdot \mathbf{ABB} \cdot \mathbf{ABC} \cdot \overline{\mathbf{ABD}} \cdot \mathbf{ABE} \cdot \overline{\mathbf{ABF}}\end{aligned}$$

$$\begin{aligned}\mathbf{PORT\ 1} = {} & \mathbf{AB0} \cdot \overline{\mathbf{AB1}} \cdot \overline{\mathbf{AB2}} \cdot \mathbf{AB3} \cdot \mathbf{AB4} \cdot \mathbf{AB5} \cdot \mathbf{AB6} \cdot \overline{\mathbf{AB7}} \cdot \\ & \mathbf{AB8} \cdot \mathbf{AB9} \cdot \mathbf{ABA} \cdot \mathbf{ABB} \cdot \mathbf{ABC} \cdot \overline{\mathbf{ABD}} \cdot \overline{\mathbf{ABE}} \cdot \overline{\mathbf{ABF}}\end{aligned}$$

The above example shows address decoding for memory locations 1f78 hex and 1f79 hex. Equation terms can be changed to accommodate any 16 bit address.

In operation, the PAL enable outputs will go high whenever one of the programmed addresses matches the address on the system memory address bus. Since the PAL fully decodes the address, any two I/O address may be used.

Conclusion

The PAL provides a single chip decoder for use in memory mapped I/O operations. This technique lowers interface parts counts and allows users an effective way to interface I/O devices to the microcomputer system.

Memory Mapped I/O

Design Specification PAL16L2

```
PAL16L2                                              PAL DESIGN SPECIFICATION
PAT0008                                                  JOHN BIRKNER 12/5/77
MEMORY MAPPED I/O

AB0 AB1 AB2 AB3 AB4 AB5 AB6 AB7 AB8 GND AB9 ABA ABB ABC
/PORT1 /PORT0 ABD ABE ABF VCC


PORT0  =   /AB0*/AB1*/AB2*AB3*AB4*AB5*AB6*/AB7*AB8*AB9*ABA*ABB*ABC*
           /ABD*/ABE*/ABF

PORT1  =    AB0*/AB1*/AB2*AB3*AB4*AB5*AB6*/AB7*AB8*AB9*ABA*ABB*ABC*
           /ABD*/ABE*/ABF


DESCRIPTION:

THE PAL DECODES THE SPECIFIED MEMORY ADDRESS WORD TO PRODUCE A PORT
ENABLE FOR PORT0 AND PORT1 AS FOLLOWS:
```

```
      PORT0  =  1F78                             PORT1  =  1F79
```

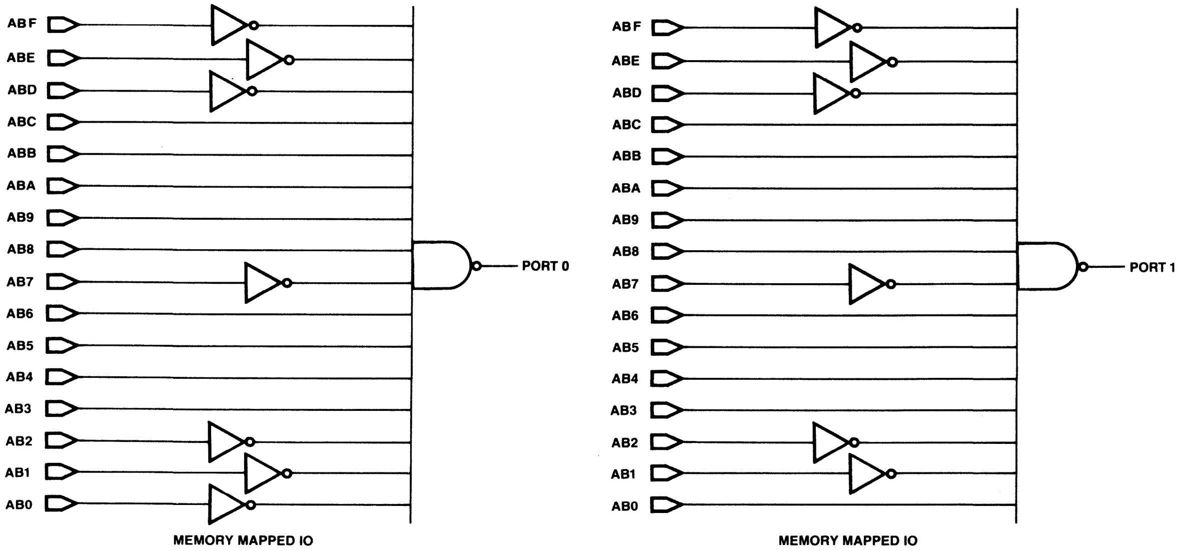


Figure 1

Memory Mapped I/O

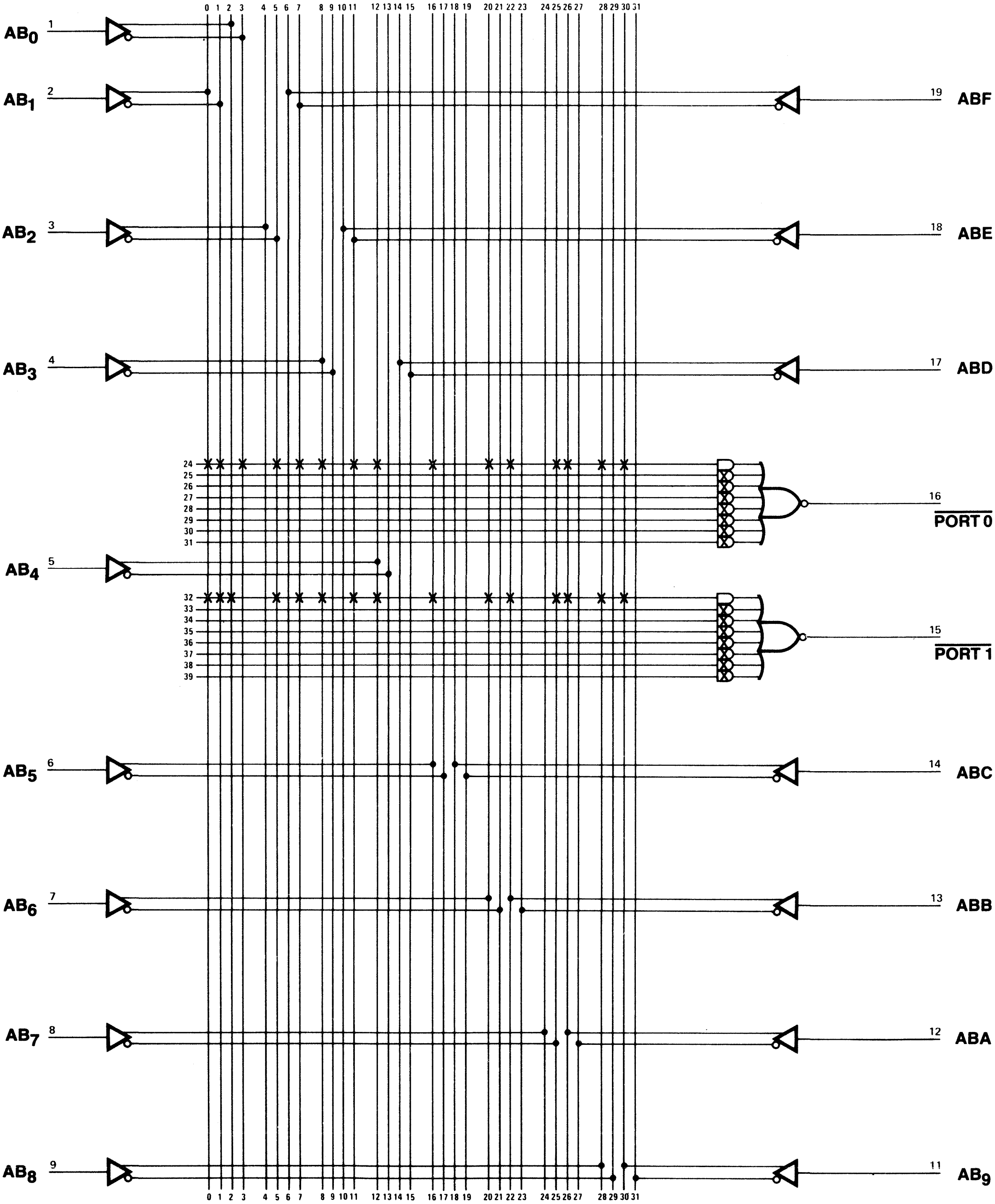
Fuse Pattern PAL16L2

```
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

XX-X -X-X X--X X--- X--- X-X- -XX- X-X- /AB0◆/AB1◆/AB2◆AB3◆AB4◆AB5◆AB6◆*
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

XXX- -X-X X--X X--- X--- X-X- -XX- X-X- AB0◆/AB1◆/AB2◆AB3◆AB4◆AB5◆AB6◆/*
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
```

*PALASM Truncates beyond 31 characters

Memory Mapped I/O

Logic Diagram PAL16L2



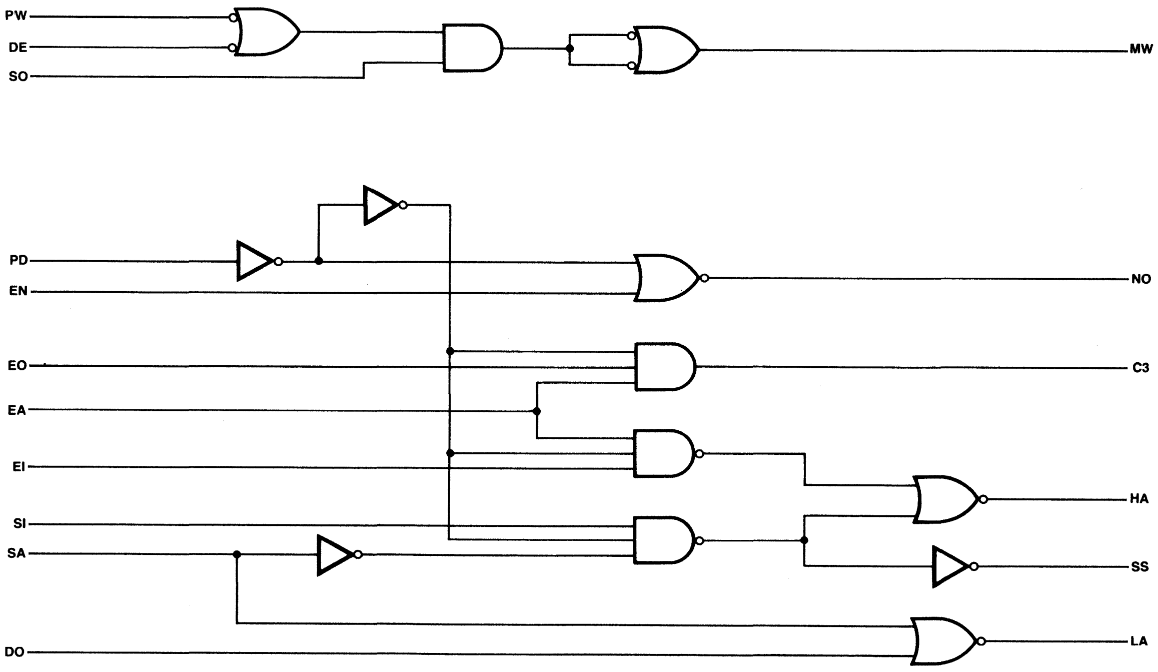
Applications

8080 Control Logic for CPU Board

The 8080 is one of the most widely used of all current microprocessor designs. However, using the 8080 in a system requires that the designer decode and supply a fairly complex set of control signals. With the rapid decline in 8080 prices, the logic required to perform this control decoding has become more expensive than the 8080 itself. This example shows how a PAL can be used to eliminate much of this costly support logic in an 8080 based system CPU card.

Portion of Random Control Logic for 8080 CPU Board   Design Specification PAL16L8

```
PAL16L8                                                  PAL DESIGN SPECIFICATION
PAT0012                                                      BOB BOSNYAK 12/16/77
PORTION OF RANDOM CONTROL LOGIC FOR 8080 CPU BOARD

PD EN EO EA S1 SA E1 DO DE GND SO NO C3 HA SS LA MW PW NC3 VCC



IF(VCC) /MW= SO◆/PW + SO◆/DE

IF(VCC) /LA= SA + DO

IF(VCC) /SS= /S1 + /PD + SA

IF(VCC) /HA= /S1 +/PD + SA + /EA + /E1

IF(VCC) /C3= /PD + /EO + /EA

IF(VCC) /NO= /PD + EN


DESCRIPTION:
  PORTION OF LOGIC FROM 8080 CPU BOARD
```



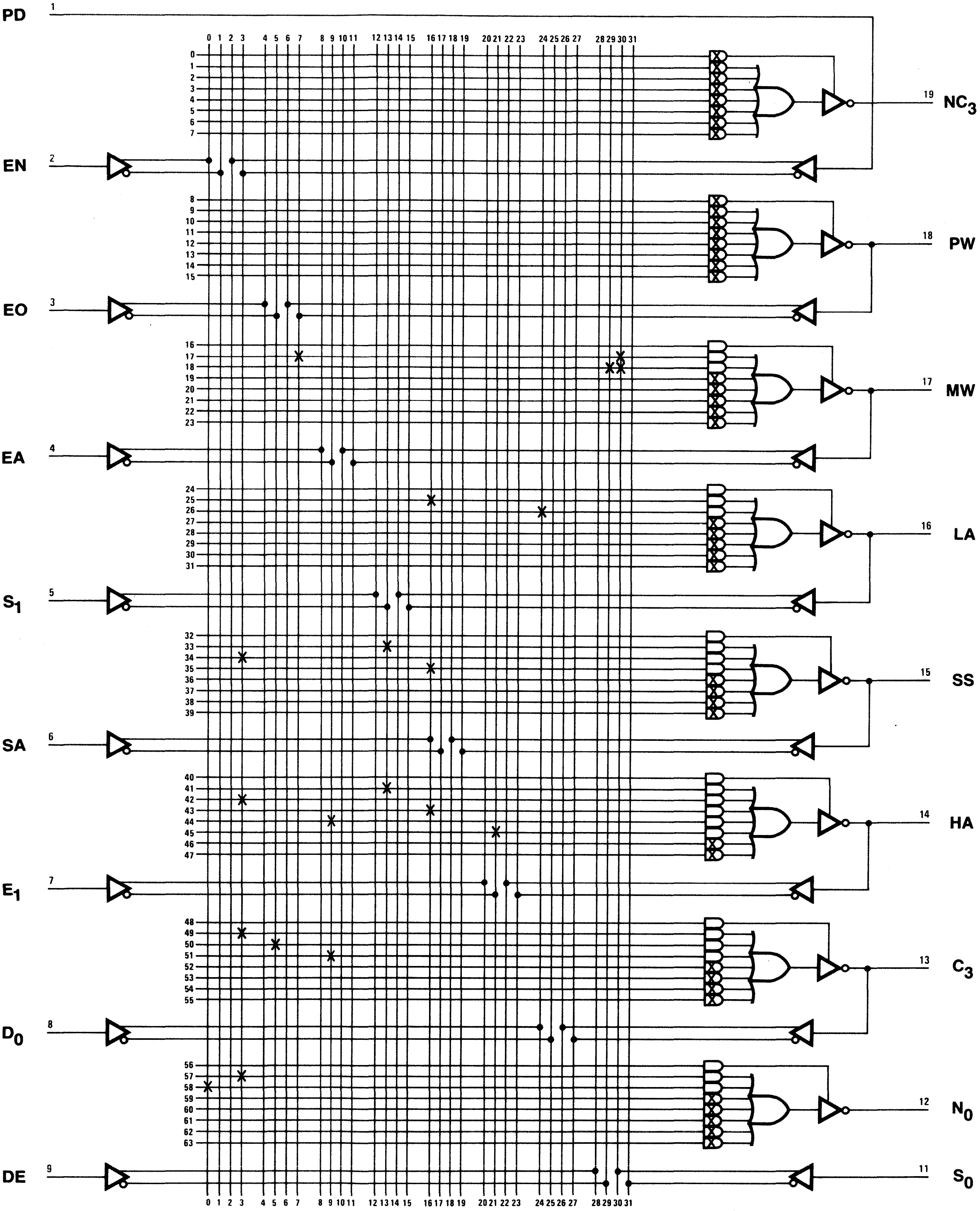
Portion of Random Control Logic for 8080 CPU Board

Fuse Pattern PAL16L8

```
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---- ---- ---- ---- ---- ---- ---- ----
---- ---X ---- ---- ---- ---- ---- --X- SO*/PW
---- ---- ---- ---- ---- ---- ---- -XX- SO*/DE
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- X--- ---- ---- ---- SA
---- ---- ---- ---- ---- ---- X--- ---- DO
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- -X-- ---- ---- ---- ---- /S1
---X ---- ---- ---- ---- ---- ---- ---- /PD
---- ---- ---- ---- X--- -- -- ---- ---- SA
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- -X-- ---- ---- ---- ---- /S1
---X ---- ---- ---- ---- ---- ---- ---- /PD
---- ---- ---- ---- X--- ---- ---- ---- SA
---- ---- -X-- ---- ---- ---- ---- ---- /EA
---- ---- ---- ---- ---- -X-- ---- ---- /E1
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---- ---- ---- ---- ---- ---- ---- ----
---X ---- ---- ---- ---- ---- ---- ---- /PD
---- -X-- ---- ---- ---- ---- ---- ---- /EO
---- ---- -X-- ---- ---- ---- ---- ---- /EA
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---- ---- ---- ---- ---- ---- ---- ----
---X ---- ---- ---- ---- ---- ---- ---- /PD
X--- ---- ---- ---- ---- ---- ---- ---- EN
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
```

Portion of Random Control Logic for 8080 CPU Board

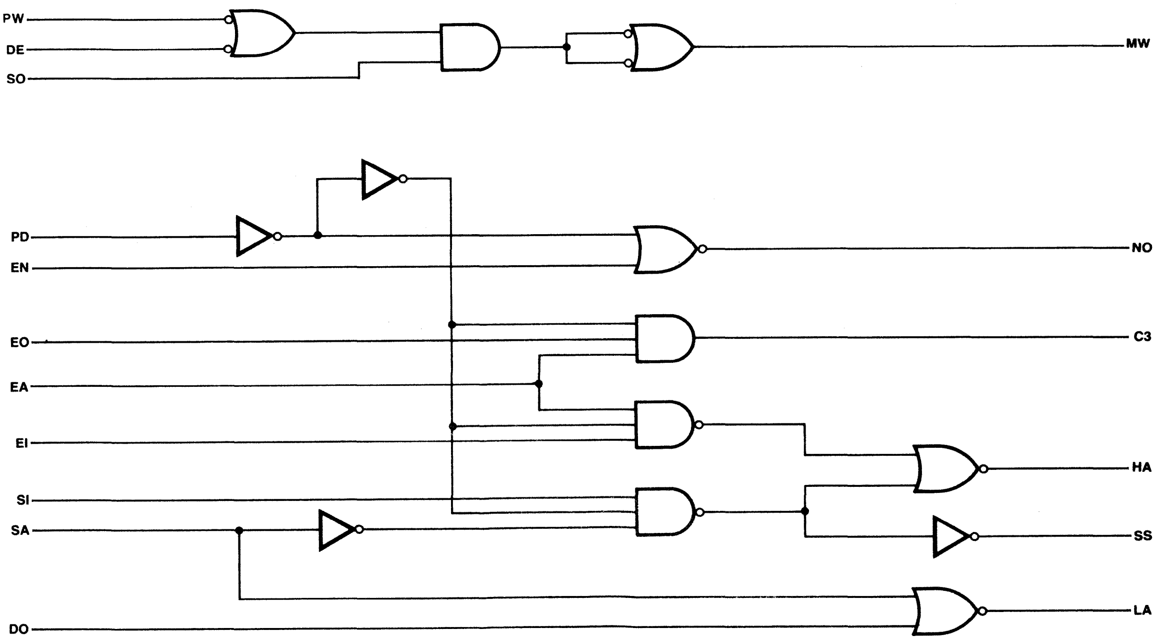
Logic Diagram PAL16L8



Portion of Random Control Logic for 8080 CPU Board (Improved Design)

Design Specification PAL12H6

```
PAL12H6                                              PAL DESIGN SPECIFICATION
PAT0013                                                 JOHN BIRKNER 12/17/77
PORTION OF RANDOM CONTROL LOGIC FOR 8080 CPU BOARD (IMPROVED DESIGN)

PD EN EO EA S1 SA E1 DO DE GND SO NC3 NO C3 HA SS LA MW PW VCC



MW = /SO + PW*DE

LA = /SA*/DO

SS = S1*PD*/SA

HA = S1*PD*/SA*EA*E1

C3 = PD*EO*EA

NO = PD*/EN



DESCRIPTION:
  PORTION OF LOGIC FROM 8080 CPU BOARD

NOTE:  THIS DESIGN IS IMPROVED OVER THE PREVIOUS EXAMPLE AS WE WERE
ABLE TO IMPLEMENT THE SAME EQUATIONS IN A SMALLER PAL.  THIS WAS
ACCOMPLISHED BY INVERTING THE EQUATIONS, THUS, REDUCING THE NUMBER OF
PRODUCTS PER OUTPUT TO A MAXIMUM OF TWO.
```



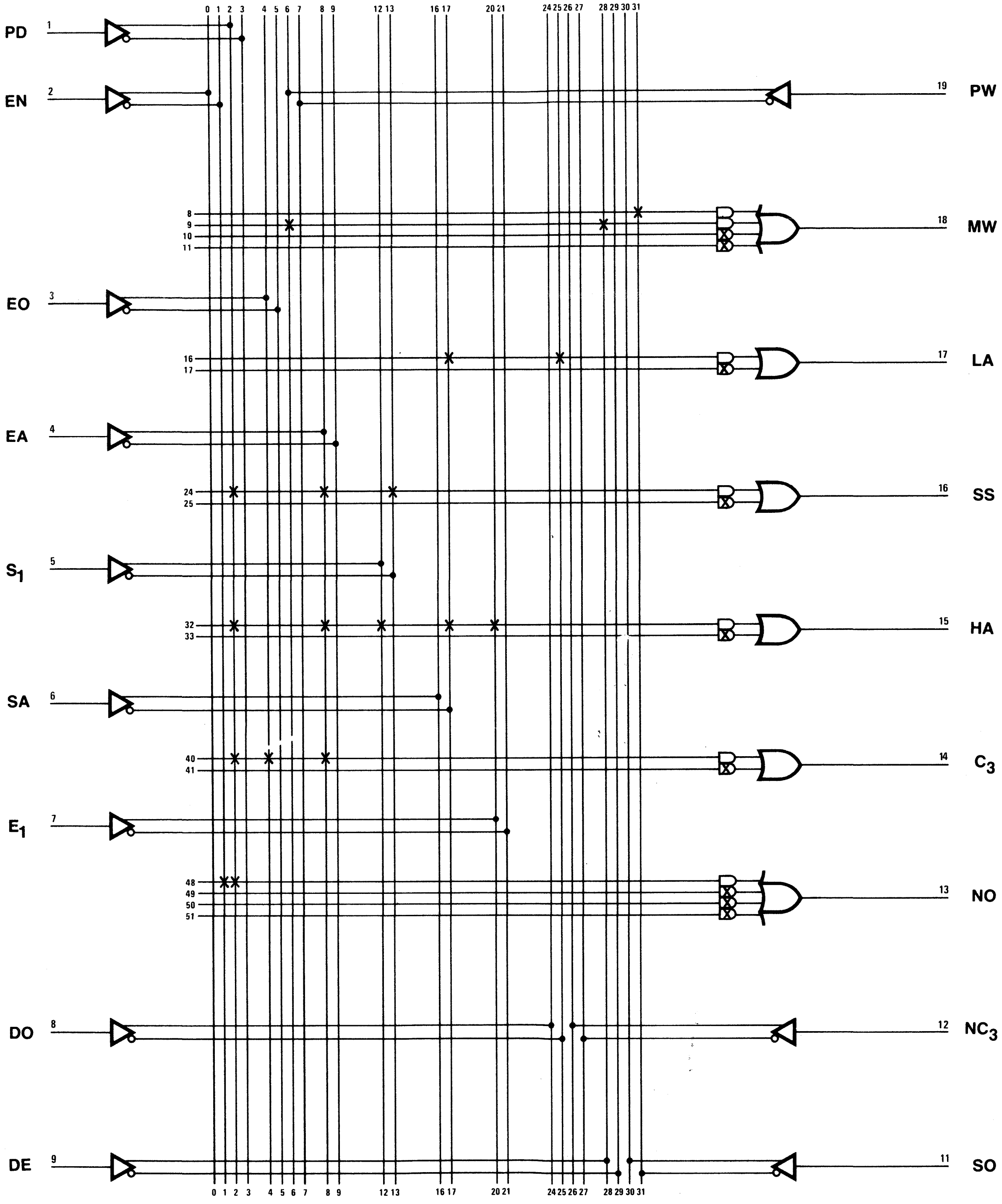
Portion of Random Control Logic for 8080 CPU Board (Improved Design)

Fuse Pattern PAL12H6

```
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----

---- ---- ---- ---- ---- ---- ---- ---X /SO
---- --X- ---- ---- ---- ---- ---- X--- PW◆DE
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX

---- ---- ---- ---- -X-- ---- -X-- ---- /SA◆/DO
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX

--X- ---- ---- X--- -X-- ---- ---- ---- S1◆PD◆/SA
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX

--X- ---- X--- X--- -X-- X--- ---- ---- S1◆PD◆/SA◆EA◆E1
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX

--X- X--- X--- ---- ---- ---- ---- ---- PD◆ED◆EA
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX

-XX- ---- ---- ---- ---- ---- ---- ---- PD◆/EM
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX
XXXX XXXX XX-- XX-- XX-- XX-- XXXX XXXX

---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
```

Portion of Random Control Logic for 8080 CPU Board (Improved Design)

Logic Diagram PAL12H6



Applications

Hexadecimal Decoder Lamp Driver

The increasing use of microcomputers has led to an increased need to display numbers in hexadecimal format (0 - 9, A - F). Standard drivers for this function are not available, so most applications are forced to use several packages to decode each digit of the display. Since six to twelve digits are often being displayed, this approach can become very expensive. This example demonstrates how the hexadecimal display format can be both decoded and the LED indicators driven using a single PAL for each digit of the display.

Hexadecimal Decoder Lamp Driver

Functional Description

A hexadecimal decoder/lamp driver accepts a four bit hexadecimal digit, converts it to its corresponding seven segment display code, and activates the appropriate segments on the display. These drivers can be used in both direct drive and multiplexed display applications. A single PAL can provide both the basic decode/drive functions and additional useful features.

Circuit Description

Figure 1 shows a three digit display system using three PALS to implement the complete decoding and display driving functions. The inputs to each section are a hexadecimal code on pins D0-D3, a ripple blanking signal, an intensity control signal, and a lamp test signal.

The hexadecimal codes will be decoded to form the seven segment code patterns shown in Figure 1. The input codes, digit represented, and output segments driven are as follows:

D3	D2	D1	D0	DIGIT	SEGMENTS
0	0	0	0	0	A,B,C,D,E,F
0	0	0	1	1	B,C
0	0	1	0	2	A,B,D,E,G
0	0	1	1	3	A,B,C,D,G
0	1	0	0	4	B,C,F,G
0	1	0	1	5	A,C,D,F
0	1	1	0	6	A,C,D,E,F
0	1	1	1	7	A,B,C
1	0	0	0	8	A,B,C,D,E,F,G
1	0	0	1	9	A,B,C,F,G
1	0	1	0	A	A,B,C,E,F,G
1	0	1	1	B	C,D,E,F,G
1	1	0	0	C	A,D,E,F
1	1	0	1	D	B,C,D,E,G
1	1	1	0	E	A,D,E,F,G
1	1	1	1	F	A,E,F,G

The ripple blanking input (RBI) is used to suppress leading zeroes in the display. The signal is propagated from the most significant digits down to the least significant digits. If the digit input is zero and the ripple blanking input is low (indicating that the previous digit is also a zero), all display segments are left blank and this digit position's ripple blanking output signal (RBO) is set low.

The intensity control signal (IC) is used to control the duty cycle of the display driver. When the signal is high, all segment outputs are turned off and the display is blank. Pulsing this pin with a duty cycled signal allows the adjustment of the display's perceived intensity.

The lamp test signal (LT) is used to verify that all segments in the display are working. When it is held high, all display segments are turned on.

PAL Implementation

The display decoder driver requires an output section that can directly drive a seven segment display. Each display digit has seven inputs (D0-D3, RBI, IC, and LT) and eight output signals (segments A-G, and RBO). The PAL 16L8 has both the required number of I/O pins and the required drive capability.

The logic equations for the driver are shown on the PALASM input. They can be easily derived from the input codes and required blanking and test functions. One PAL is used for each digit, and the number of digits can be as large as required. With minor changes this basic logic could also be used with multiplexor logic to allow a single PAL to decode and drive a multi-digit display.

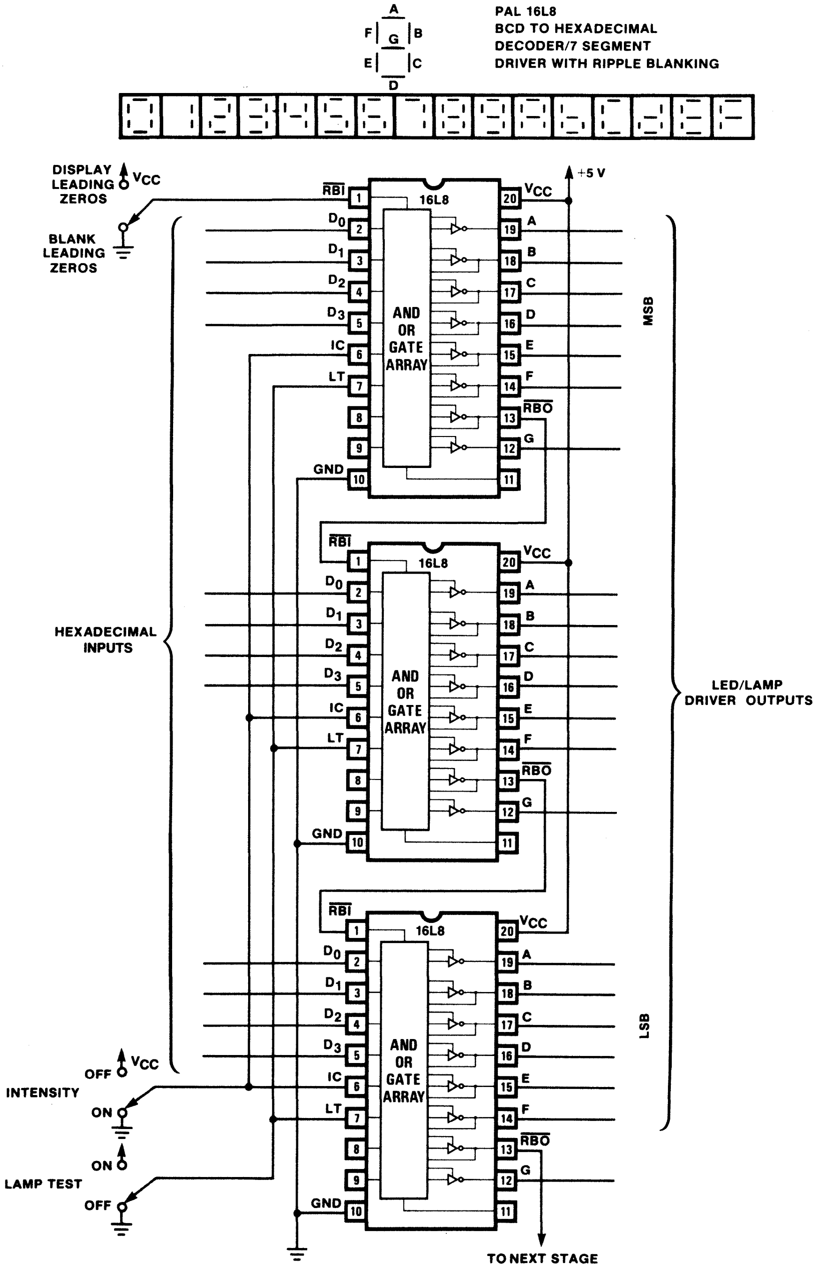
Summary

The PAL provides a low cost and functionally flexible direct decoder/driver for use in a variety of display applications. This example demonstrated hexadecimal to seven segment code conversions, but many other display formats and code conversions can be implemented using similar techniques.

Hex Decoder/7 Seg. Driver w/Ripple Blanking, Intensity Con., & Lamp Test

Logic Schematic

THREE STAGE HEXADECIMAL DECODER/DRIVER



Hex Decoder/7 Seg. Driver w/Ripple Blanking, Intensity Con., & Lamp Test

Design Specification PAL16L8

```
PAL16L8                                          PAL DESIGN SPECIFICATION
PAT0007                                            LES GARDINIER 12-14-77
HEX DECODER/7SEG. DRIVER W/RIPPLE BLANKING, INTENSITY CON., & LAMP TEST

/RBI D0 D1 D2 D3 IC LT NC NC GND NC G /RBO F E D C B A VCC



IF (/IC) /A = /RBO*/D0*/D2 + /RBO*/D0*D3 + /RBO*D1*D2 +
              /RBO*D1*/D2*/D3 + /RBO*D0*D2*/D3 + /RBO*/D1*/D2*D3 + LT

IF (/IC) /B = /RBO*/D2*/D3 + /RBO*/D0*/D2 + /RBO*/D0*/D1*/D3 +
              /RBO*D0*D1*/D3 + /RBO*D0*/D1*D3 + LT

IF (/IC) /C = /RBO*D0*/D1 + /RBO*D0*/D2 + /RBO*/D1*/D2 +
              /RBO*D2*/D3 + /RBO*/D2*D3 + LT

IF (/IC) /D = /RBO*/D1*D3 + /RBO*/D0*/D2*/D3 +
              /RBO*D0*D1*/D2 + /RBO*/D0*D1*D2 + /RBO*D0*/D1*D2 + LT

IF (/IC) /E = /RBO*/D0*/D2 + /RBO*D2*D3 + /RBO*/D0*D1 +
              /RBO*D1*D3 + LT

IF (/IC) /F = /RBO*/D0*/D1 + /RBO*/D2*D3 + /RBO*D1*D3 +
              /RBO*/D0*D2 + /RBO*/D1*D2*/D3 + LT

IF (VCC)  RBO = /D0*/D1*/D2*/D3*/RBI

IF (/IC) /G = /RBO*D1*/D2 + /RBO*D0*D3 + /RBO*/D2*D3 +
              /RBO*/D0*D1 + /RBO*/D1*D2*/D3 + LT



DESCRIPTION:
THE HEXFDECIMAL DECODER/7-SEGMENT DRIVER FEATURES ACTIVE LOW OUTPUTS
FOR DRIVING DISPLAY DIRECTLY.

IF DATA INPUT IS ZERO AND RIPPLE BLANKING INPUT (/RBI) IS LOW THAT
DIGIT WILL BE BLANKED AND RIPPLE BLANKING OUTPUT WILL BE LOW.

THE RIPPLE BLANKING OUTPUT (/RBO) PROVIDES BLANKING INFORMATION
FOR THE NEXT LEAST SIGNIFICANT STAGE. IT PROVIDES A LOW IF /RBI IS
LOW AND THE DATA IN IS ZERO.

WHEN HIGH THE INTENSITY CONTROL (IC) WILL TURN OFF THE ENTIRE DISPLAY.
IC MAY BE PULSED TO VARY THE INTENSITY OF THE DISPLAY.

WHEN HIGH THE LAMP TEST INPUT (LT) WILL TURN ON THE DISPLAY.


FUNCTION TABLE:
-------------------------------------------------
             INPUTS            !      OUTPUTS      !
-------------------------------!-------------------!
LT IC /RBI D0 D1 D2 D3!A B C D E F G /RBO  !
-------------------------------!-------------------!
 L  H   X   X  X  X  X !Z Z Z Z Z Z Z   X    !
 L  L   H   L  L  L  L !L L L L L L H   L    !
 L  L   L   L  L  L  L !H H H H H H H   L    !
 H  L   X   X  X  X  X !L L L L L L L   X    !
-------------------------------------------------
```

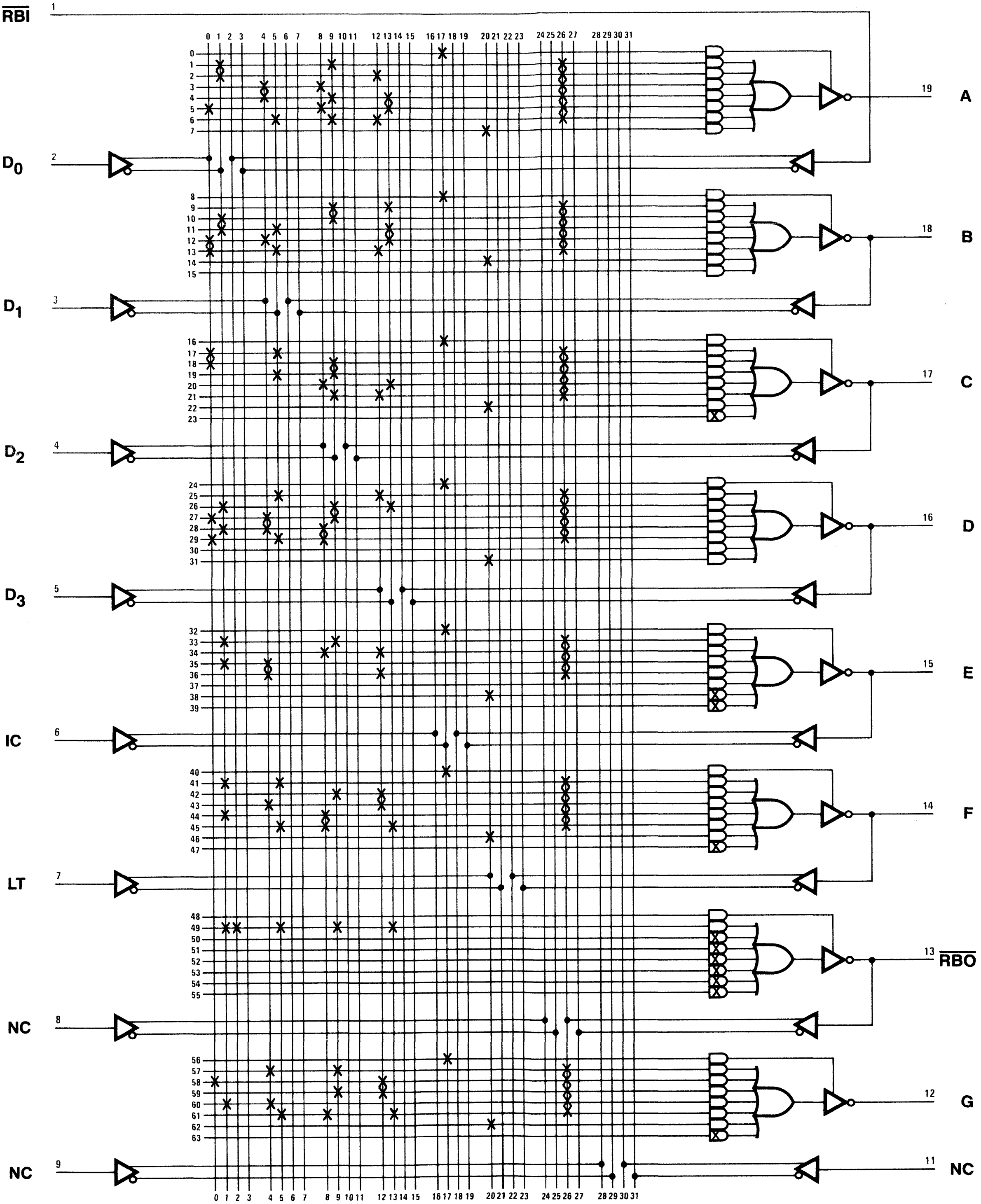
Hex Decoder/7 Seg. Driver w/Ripple Blanking, Intensity Con., & Lamp Test

Fuse Pattern PAL16L8

```
---- ---- ---- ---- -X-- ---- ---- ---- /IC
-X-- ---- -X-- ---- ---- ---- --X- ---- /RBO*/D0*/D2
-X-- ---- ---- X--- ---- ---- --X- ---- /RBO*/D0*D3
---- X--- X--- ---- ---- ---- --X- ---- /RBO*D1*D2
---- X--- -X-- -X-- ---- ---- --X- ---- /RBO*D1*/D2*/D3
X--- ---- X--- -X-- ---- ---- --X- ---- /RBO*D0*D2*/D3
---- -X-- -X-- X--- ---- ---- --X- ---- /RBO*/D1*/D2*D3
---- ---- ---- ---- ---- X--- ---- ---- LT

---- ---- ---- ---- -X-- ---- ---- ---- /IC
---- ---- -X-- -X-- ---- ---- --X- ---- /RBO*/D2*/D3
-X-- ---- -X-- ---- ---- ---- --X- ---- /RBO*/D0*/D2
-X-- -X-- ---- -X-- ---- ---- --X- ---- /RBO*/D0*/D1*/D3
X--- X--- ---- -X-- ---- ---- --X- ---- /RBO*D0*D1*/D3
X--- -X-- ---- X--- ---- ---- --X- ---- /RBO*D0*/D1*D3
---- ---- ---- ---- ---- X--- ---- ---- LT
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---- ---- ---- ---- -X-- ---- ---- ---- /IC
X--- -X-- ---- ---- ---- ---- --X- ---- /RBO*D0*/D1
X--- ---- -X-- ---- ---- ---- --X- ---- /RBO*D0*/D2
---- -X-- -X-- ---- ---- ---- --X- ---- /RBO*/D1*/D2
---- ---- X--- -X-- ---- ---- --X- ---- /RBO*D2*/D3
---- ---- -X-- X--- ---- ---- --X- ---- /RBO*/D2*D3
---- ---- ---- ---- ---- X--- ---- ---- LT
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---- ---- ---- ---- -X-- ---- ---- ---- /IC
---- -X-- ---- X--- ---- ---- --X- ---- /RBO*/D1*D3
-X-- ---- -X-- -X-- ---- ---- --X- ---- /RBO*/D0*/D2*/D3
X--- X--- -X-- ---- ---- ---- --X- ---- /RBO*D0*D1*/D2
-X-- X--- X--- ---- ---- ---- --X- ---- /RBO*/D0*D1*D2
X--- -X-- X--- ---- ---- ---- --X- ---- /RBO*D0*/D1*D2
---- ---- ---- ---- ---- X--- ---- ---- LT
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---- ---- ---- ---- -X-- ---- ---- ---- /IC
-X-- ---- -X-- ---- ---- ---- --X- ---- /RBO*/D0*/D2
---- ---- X--- X--- ---- ---- --X- ---- /RBO*D2*D3
-X-- X--- ---- ---- ---- ---- --X- ---- /RBO*/D0*D1
---- X--- ---- X--- ---- ---- --X- ---- /RBO*D1*/D3
---- ---- ---- ---- ---- X--- ---- ---- LT
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---- ---- ---- ---- -X-- ---- ---- ---- /IC
-X-- -X-- ---- ---- ---- ---- --X- ---- /RBO*/D0*/D1
---- ---- -X-- X--- ---- ---- --X- ---- /RBO*/D2*D3
---- X--- ---- X--- ---- ---- --X- ---- /RBO*D1*D3
-X-- ---- X--- ---- ---- ---- --X- ---- /RBO*/D0*D2
---- -X-- X--- -X-- ---- ---- --X- ---- /RBO*/D1*D2*/D3
---- ---- ---- ---- ---- X--- ---- ---- LT
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---- ---- ---- ---- ---- ---- ---- ----
-XX- -X-- -X-- -X-- ---- ---- ---- ---- /D0*/D1*/D2*/D3*/RBI
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---- ---- ---- ---- -X-- ---- ---- ---- /IC
---- X--- -X-- ---- ---- ---- --X- ---- /RBO*D1*/D2
X--- ---- ---- X--- ---- ---- --X- ---- /RBO*D0*D3
---- ---- -X-- X--- ---- ---- --X- ---- /RBO*/D2*D3
-X-- X--- ---- ---- ---- ---- --X- ---- /RBO*/D0*D1
---- -X-- X--- -X-- ---- ---- --X- ---- /RBO*/D1*D2*/D3
---- ---- ---- ---- ---- X--- ---- ---- LT
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
```

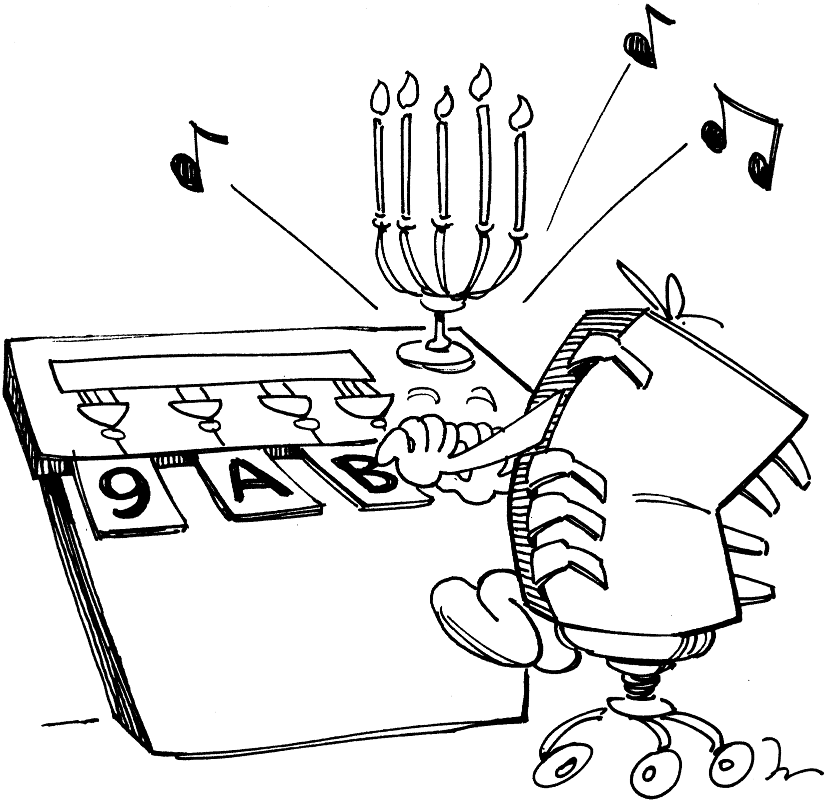
Hex Decoder/7 Seg. Driver w/Ripple Blanking, Intensity Con., & Lamp Test

Logic Diagram PAL16L8



Applications

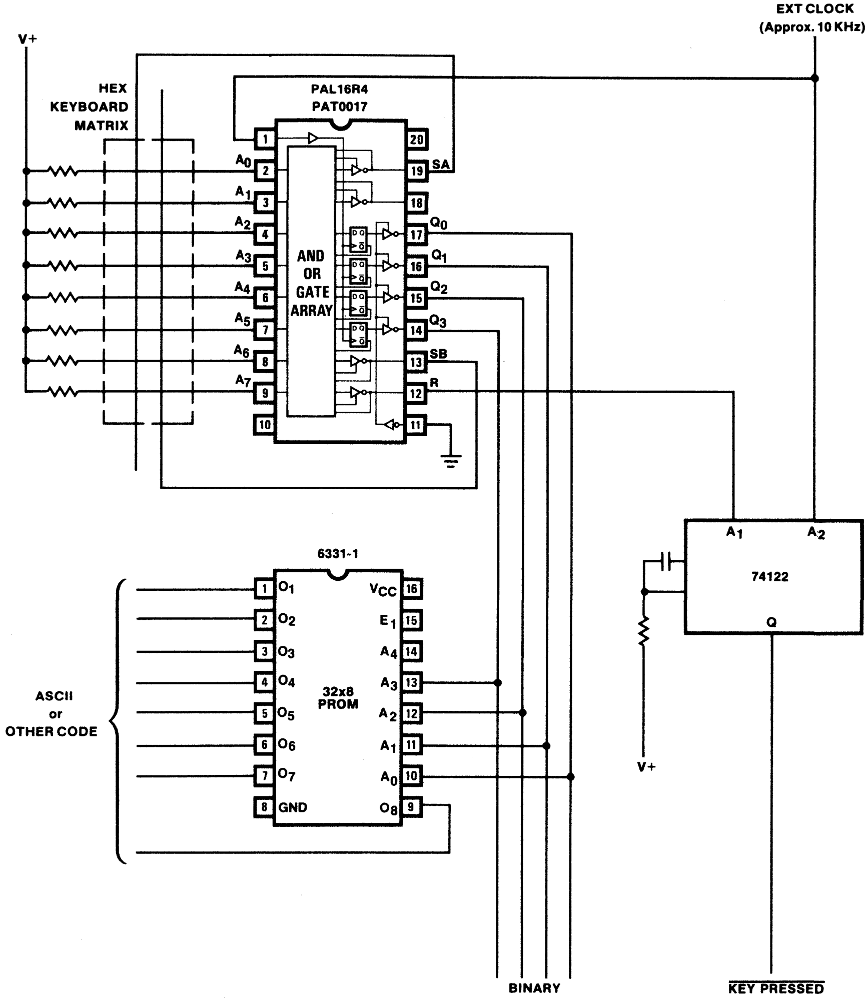
Hex Keyboard Scanner



A problem similar to driving the hexadecimal display in the previous example is encountered in system input design. The popularity of consumer calculators has made the small keypad a widely available and low cost system input device. The logic required to scan these small keyboards is generated by using either SSI/MSI logic or a computer generated software scan. The logic may be quite expensive, and, if the microcomputer is in a functionally busy system, the software scan may be inadequate. A single PAL and a few other parts can now be used to implement this system function.

Hex Keyboard Scanner

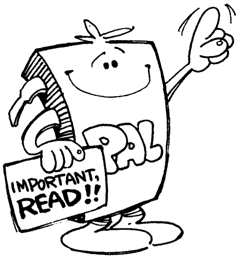
Logic Diagram PAL16L8



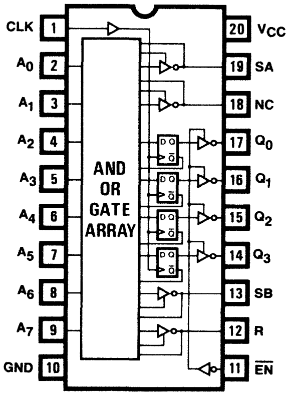
Hex Keyboard Scanner

Design Specification PAL16R4

```
PAL16R4                                                    PAL DESIGN SPECIFICATION
PAT0017                                                           VIC NEWTON   12/15/77
HEX KEYBOARD SCANNER

CLK A0 A1 A2 A3 A4 A5 A6 A7 GND /EN R SB Q3 Q2 Q1 Q0 NC SA VCC


IF ( VCC ) /SA = /Q3


IF ( VCC ) /NC   = /A0*/Q0*/Q1*/Q2 + /A1*Q0*/Q1*/Q2 +
                   /A2*/Q0*Q1*/Q2 + /A3*Q0*Q1*/Q2 +
                   /A4*/Q0*/Q1*Q2 + /A5*Q0*/Q1*Q2 +
                   /A6*/Q0*Q1*Q2

/Q0 := Q0*R + /Q0*/R

/Q1 := Q0*Q1*R + /Q0*/Q1*R + /Q1*/R

/Q2 := Q0*Q1*Q2*R+/Q0*/Q2*R + /Q1*/Q2*R + /Q2*/R

/Q3 := Q0*Q1*Q2*Q3*R + /Q0*/Q3*R + /Q1*/Q3*R + /Q2*/Q3*R + /Q3*/R

IF ( VCC ) /SB = Q3

IF ( VCC )  /R = /NC + /A7*Q0*Q1*Q2


DESCRIPTION:

    THE KEYBOARD SCANNER WILL SCAN A 16 KEY KEYBOARD ARRANGED IN A
2X8 MATRIX.  THE SCANNER WORKS BY SELECTING ONE ROW OF 8 SWITCHES
AND THEN SCANNING THE 8 INPUTS.  A LOW ON ANY INPUT WILL DISABLE THE CLOCK
GOING INTO THE ONE-SHOT.  THE ONE-SHOT IS USED AS A DELAY TO ALLOW THE
SWITCH BOUNCE TO SETTLE OUT.  AT THE END OF THE TIME DELAY (10MS), KEYPRESSED
WILL GO LOW.  THE OUTPUTS WILL THEN GIVE THE BINARY CODE FOR THE SWITCH
SELECTED.  WHEN THE SWITCH IS RELEASED, KEYPRESSED WILL GO HIGH, AND
SCANNING WILL CONTINUE.  WHEN THE END OF THE ROW IS REACHED, THE
SCANNER SWITCHES TO THE OTHER ROW AND CONTINUES SCANNING.
    THE EXTERNAL CLOCK SHOULD RUN IN THE RANGE OF 10 KHZ.
IF ASCII OR OTHER CODED CHARACTERS ARE DESIRED, THE BINARY CAN BE
 CONVERTED USING A PROM.
```



PAL16R4



Logic Symbol

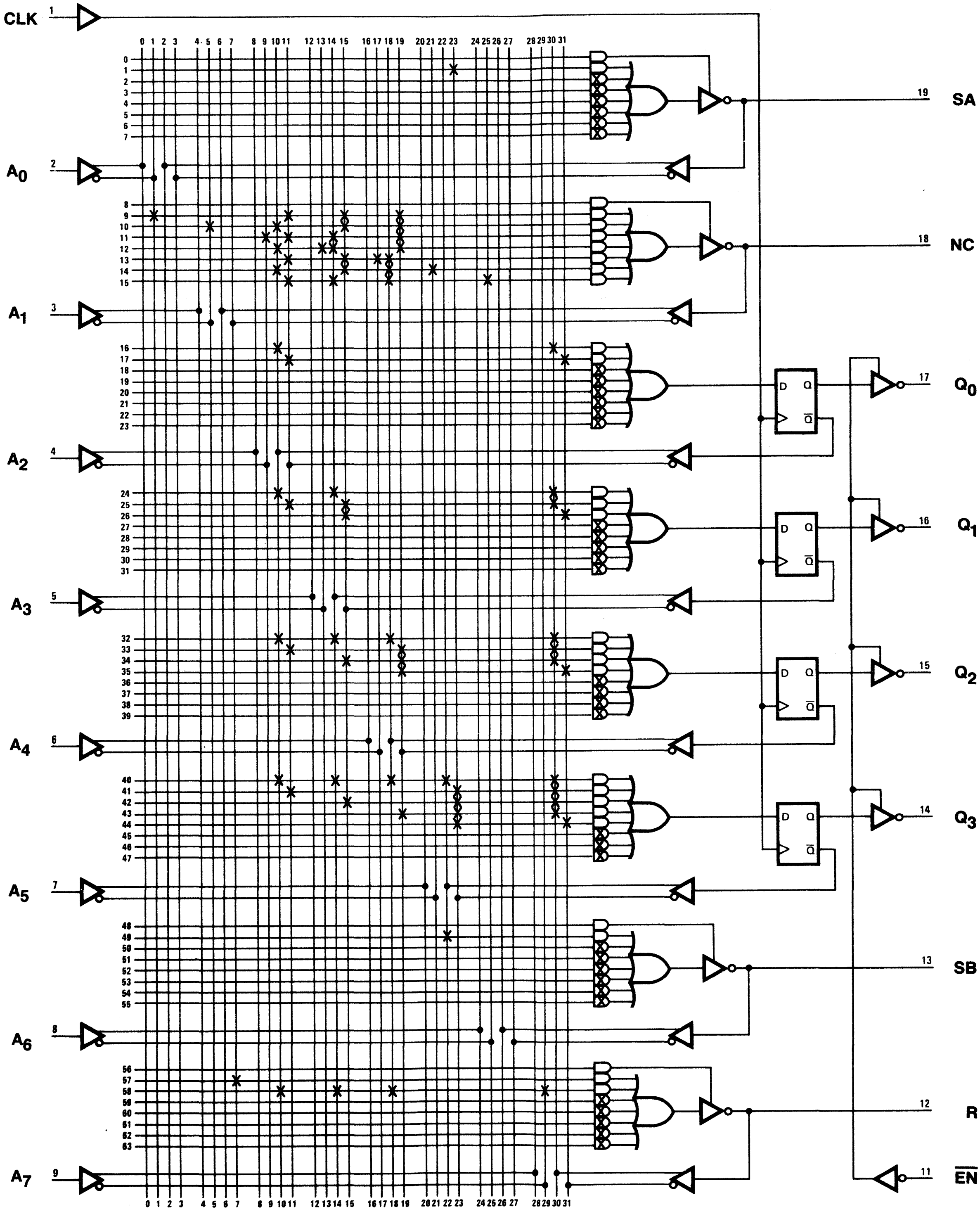
Hex Keyboard Scanner

Fuse Pattern PAL16R4

```
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---X ---- ---- /Q3
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---- ---- ---- ---- ---- ---- ---- ----
-X-- ---- ---X ---X ---X ---- ---- ---- /A0◆/Q0◆/Q1◆/Q2
---- -X-- --X- ---X ---X ---- ---- ---- /A1◆Q0◆/Q1◆/Q2
---- ---- -X-X --X- ---X ---- ---- ---- /A2◆/Q0◆Q1◆/Q2
---- ---- --X- -XX- ---X ---- ---- ---- /A3◆Q0◆Q1◆/Q2
---- ---- ---X ---X -XX- ---- ---- ---- /A4◆/Q0◆/Q1◆Q2
---- ---- --X- ---X --X- -X-- ---- ---- /A5◆Q0◆/Q1◆Q2
---- ---- ---X --X- --X- ---- -X-- ---- /A6◆/Q0◆Q1◆Q2

---- ---- --X- ---- ---- ---- ---- --X- Q0◆R
---- ---- ---X ---- ---- ---- ---- ---X /Q0◆/R
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---- ---- --X- --X- ---- ---- ---- --X- Q0◆Q1◆R
---- ---- ---X ---X ---- ---- ---- --X- /Q0◆/Q1◆R
---- ---- ---- ---X ---- ---- ---- ---X /Q1◆/R
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---- ---- --X- --X- --X- ---- ---- --X- Q0◆Q1◆Q2◆R
---- ---- ---X ---- ---X ---- ---- --X- /Q0◆/Q2◆R
---- ---- ---- ---X ---X ---- ---- --X- /Q1◆/Q2◆R
---- ---- ---- ---- ---X ---- ---- ---X /Q2◆/R
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---- ---- --X- --X- --X- --X- ---- --X- Q0◆Q1◆Q2◆Q3◆R
---- ---- ---X ---- ---- ---X ---- --X- /Q0◆/Q3◆R
---- ---- ---- ---X ---- ---X ---- --X- /Q1◆/Q3◆R
---- ---- ---- ---- ---X ---X ---- --X- /Q2◆/Q3◆R
---- ---- ---- ---- ---- ---X ---- ---X /Q3◆/R
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- --X- ---- ---- Q3
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---- ---- ---- ---- ---- ---- ---- ----
---- ---X ---- ---- ---- ---- ---- ---- /NC
---- ---- --X- --X- --X- ---- ---- -X-- /A7◆Q0◆Q1◆Q2
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
```

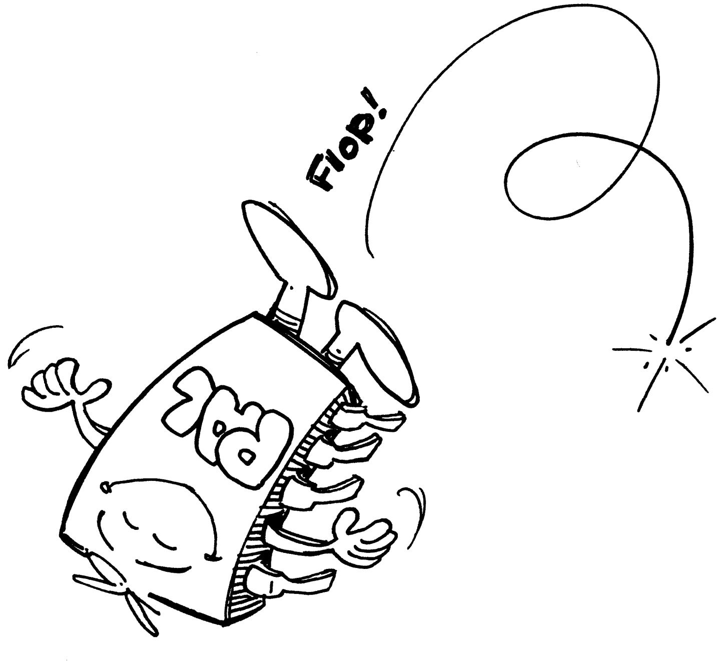
Hex Keyboard Scanner

Logic Diagram PAL16R4



Applications

Micro Floppy Control Logic

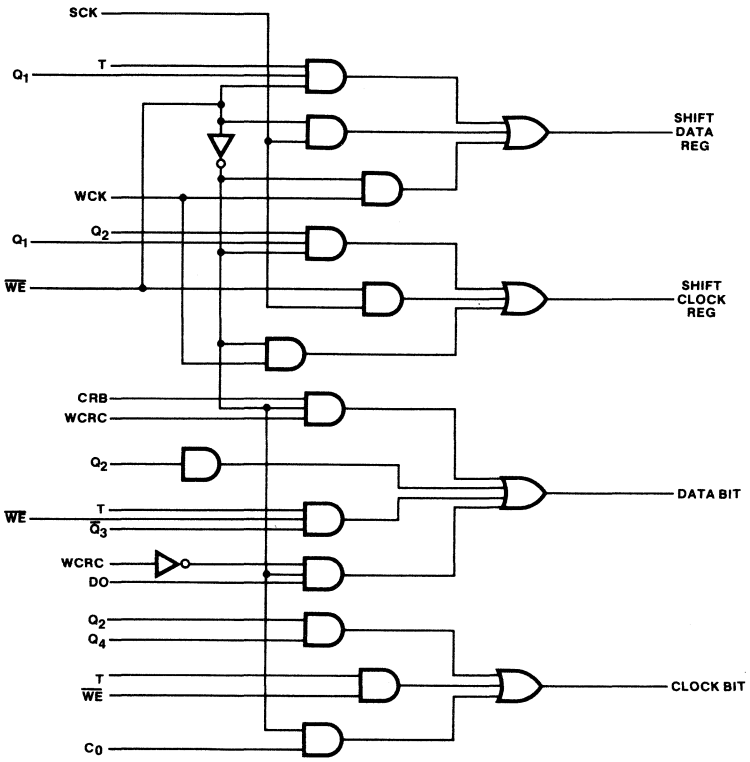


The floppy disk, and its smaller brother the micro-floppy, are becoming increasingly popular as mass storage devices on small systems. Most of these small systems are destined for high volume applications, so all possible production cost economies must be made. The disc controller is the most complicated (and expensive) portion of the disc sub-system, and this example shows how a PAL can be used to reduce the cost and size of the controller for a micro-floppy disc controller.

Portion of Micro Floppy Control Logic

Design Specification PAL14H4

```
PAL14H4                                          PAL DESIGN SPECIFICATION
PAT0024                                              BOB BOSNYAK 12/16/77
PORTION OF MICRO FLOPPY CONTROL LOGIC

SCK T Q1 WCK Q2 /WE CRB WCRC /Q3 GND Q4 CO DO CKB DATABT SCREG SDREG N2 N1 VCC


SDREG= T◆Q1◆/WE + SCK◆/WE + WE◆WCK

SCREG= Q2◆Q1◆WE + SCK◆/WE + WE◆WCK

DATABT= CRB◆WE◆WCRC + Q2 + T◆/WE◆/Q3 + /WCRC◆WE◆DO

CKB= Q2◆Q4 + T◆/WE + WE◆CO


DESCRIPTION:

   PORTION OF FLOPPY DISC CONTROL LOGIC
```



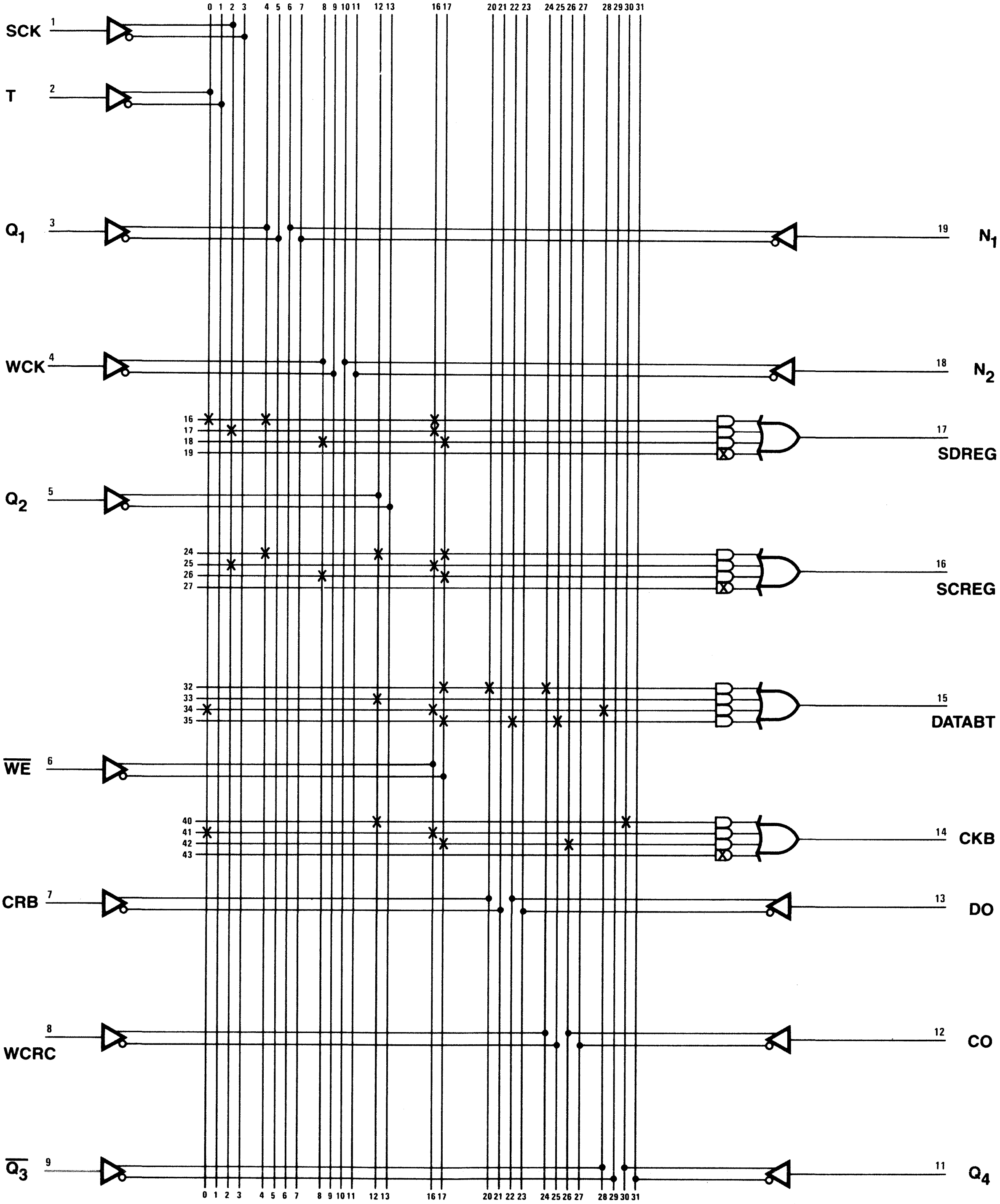
Portion of Micro Floppy Control Logic

Fuse Pattern PAL14H4

```
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----

---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----

X--- X--- ---- ---- X--- ---- ---- ---- T◆Q1◆/WE
--X- ---- ---- ---- X--- ---- ---- ---- SCK◆/WE
---- ---- X--- ---- -X-- ---- ---- ---- WE◆WCK
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX

---- X--- ---- X--- -X-- ---- ---- ---- Q2◆Q1◆WE
--X- ---- ---- ---- X--- ---- ---- ---- SCK◆/WE
---- ---- X--- ---- -X-- ---- ---- ---- WE◆WCK
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX

---- ---- ---- ---- -X-- X--- X--- ---- CRB◆WE◆WCRC
---- ---- ---- X--- ---- ---- ---- ---- Q2
X--- ---- ---- ---- X--- ---- ---- X--- T◆/WE◆/Q3
---- ---- ---- ---- -X-- --X- -X-- ---- /WCRC◆WE◆DO
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX

---- ---- ---- X--- ---- ---- ---- --X- Q2◆Q4
X--- ---- ---- ---- X--- ---- ---- ---- T◆/WE
---- ---- ---- ---- -X-- ---- --X- ---- WE◆CO
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX
XXXX XXXX XXXX XX-- XX-- XXXX XXXX XXXX

---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----

---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
```

Portion of Micro Floppy Control Logic

Logic Diagram PAL14H4



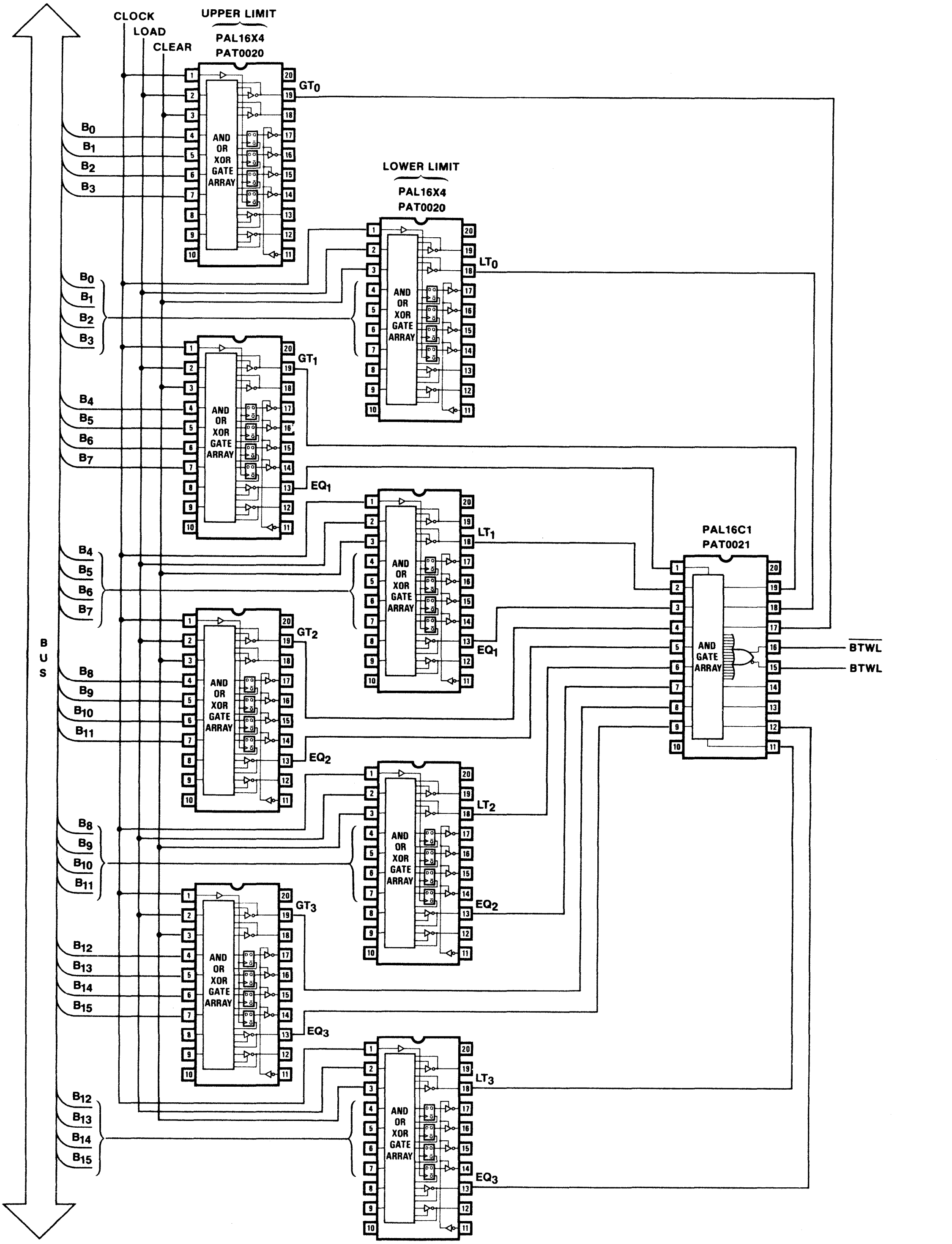
Applications

Between Limits Comparator

In many systems applications it is desirable to keep a running check on the data passing by certain points in the system. This can be done for systems security or simply as a part of system diagnosis and self-checking. In various cases the limit checker may search for specific values, missing values, values in a specific range, or values out of a specific range. This example shows a PAL in a microcomputer bus watching application typical of many system limit checking requirements.

Between Limits Comparator

Logic Schematic



Between Limits Comparator/Register

Design Specification PAL16X4

```
PAL16X4                                     PAL DESIGN SPECIFICATION
PAT0020                                        JOHN BIRKNER 12/18/77
BETWEEN LIMITS COMPARATOR / REGISTER

CK LOAD CLEAR B0 B1 B2 B3 NC NC GND /E NC /EQ A3 A2 A1 A0 /LT /GT VCC


IF(VCC) LT = (A3*/B3) +
             (A2*/B2) * (A3.EQ.B3) +
             (A1*/B1) * (A3.EQ.B3) * (A2.EQ.B2) +
             (A0*/B0) * (A3.EQ.B3) * (A2.EQ.B2) * (A1.EQ.B1)

IF(VCC) GT = (/A3*B3) +
             (/A2*B2) * (A3.EQ.B3) +
             (/A1*B1) * (A3.EQ.B3) * (A2.EQ.B2) +
             (/A0*B0) * (A3.EQ.B3) * (A2.EQ.B2) * (A1.EQ.B1)



    /A0 := (/A0)*/LOAD*/CLEAR + (/B0)*LOAD*/CLEAR

    /A1 := (/A1)*/LOAD*/CLEAR + (/B1)*LOAD*/CLEAR

    /A2 := (/A2)*/LOAD*/CLEAR + (/B2)*LOAD*/CLEAR

    /A3 := (/A3)*/LOAD*/CLEAR + (/B3)*LOAD*/CLEAR



IF(VCC) EQ = (A3.EQ.B3) * (A2.EQ.B2) * (A1.EQ.B1) * (A0.EQ.B0)



DESCRIPTION:

THE DEVICE CONTINUOUSLY COMPARES THE VALUE OF BUS, B, WITH THE VALUE OF
REGISTER A AND REPORTS THE STATUS ON OUTPUTS GT, LT, AND EQ.  GT INDICATES
THAT B IS GTEATER THAN A.  LT INDICATES THAT B IS LESS THAN A.  EQ
INDICATES THAT A IS EQUAL TO B.  THE VALUE OF REGISTER A
MAY READ BY LOWERING ENABLE LINE, /E.  REGISTER A IS LOADED
WITH THE VALUE ON BUS, B, WHEN THE LOAD LINE IS HIGH AND THE CLEAR LINE
IS LOW ON THE LOW TO HIGH TRANSITION OF THE CLOCK.


FUNCTION TABLE:

-----------------------------------------------------------
! LOAD ! CLEAR ! CLOCK ! REGISTER A ! OPERATION !
-----------------------------------------------------------
!   L  !   L   !  L-H  !     A      ! NOP       !
!   X  !   H   !  L-H  !  ALL HIGH  ! CLEAR     !
!   H  !   L   !  L-H  !     B      ! LOAD B    !
-----------------------------------------------------------
```

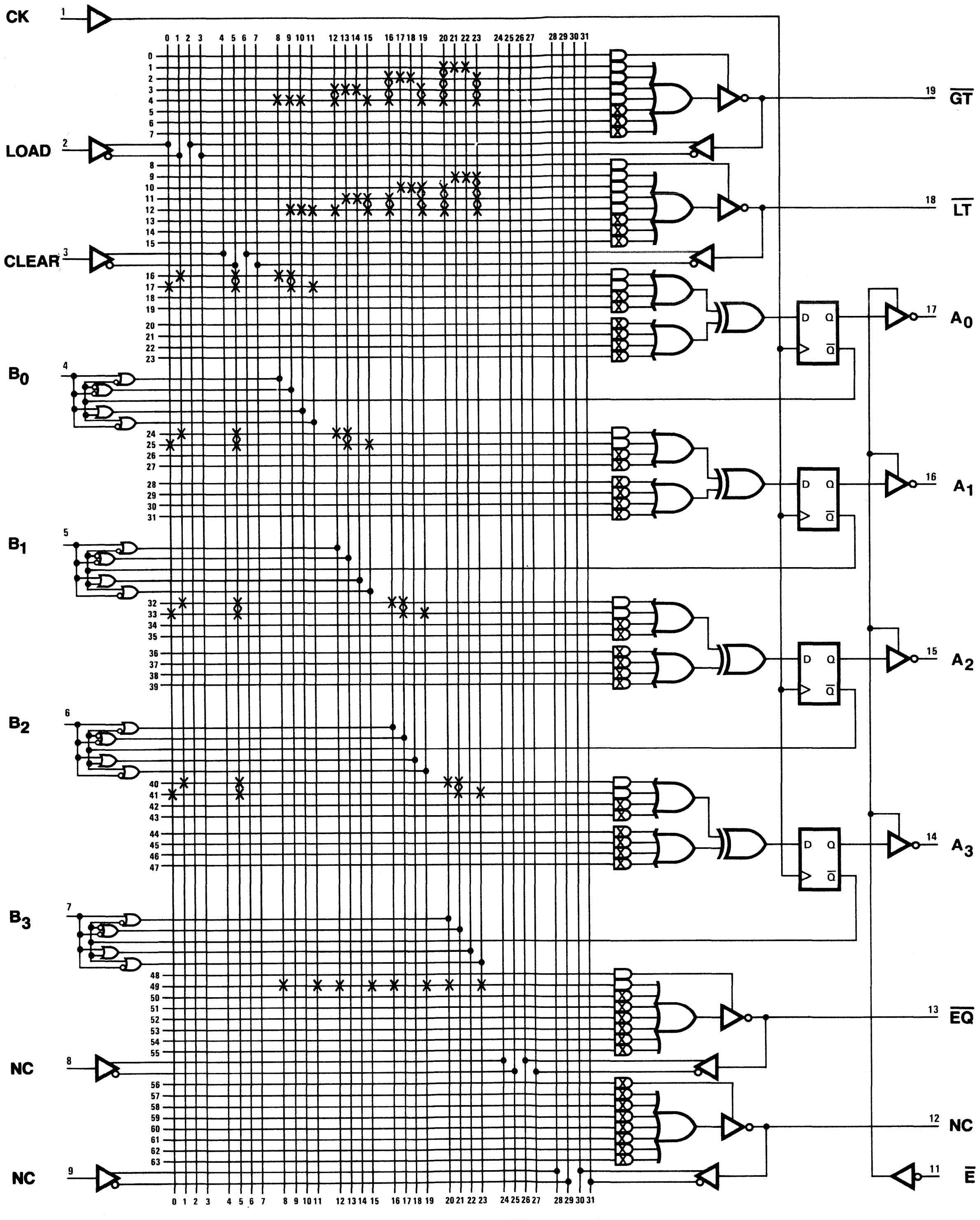
Between Limits Comparator/Register

Fuse Pattern PAL16X4

```
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- XXX- ---- ---- /A3◆B3
---- ---- ---- ---- XXX- X--X ---- ---- /A2◆B2◆A3.EQ.B3
---- ---- ---- XXX- X--X X--X ---- ---- /A1◆B1◆A3.EQ.B3◆A2.EQ.B2
---- ---- XXX- X--X X--X X--X ---- ---- /A0◆B0◆A3.EQ.B3◆A2.EQ.B2◆A1.EQ.
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- -XXX ---- ---- A3◆/B3
---- ---- ---- ---- -XXX X--X ---- ---- A2◆/B2◆A3.EQ.B3
---- ---- ---- -XXX X--X X--X ---- ---- A1◆/B1◆A3.EQ.B3◆A2.EQ.B2
---- ---- -XXX X--X X--X X--X ---- ---- A0◆/B0◆A3.EQ.B3◆A2.EQ.B2◆A1.EQ.
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

-X-- -X-- XX-- ---- ---- ---- ---- ---- /A0◆/LOAD◆/CLEAR
X--- -X-- -X-X ---- ---- ---- ---- ---- /B0◆LOAD◆/CLEAR
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

-X-- -X-- ---- XX-- ---- ---- ---- ---- /A1◆/LOAD◆/CLEAR
X--- -X-- ---- -X-X ---- ---- ---- ---- /B1◆LOAD◆/CLEAR
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

-X-- -X-- ---- ---- XX-- ---- ---- ---- /A2◆/LOAD◆/CLEAR
X--- -X-- ---- ---- -X-X ---- ---- ---- /B2◆LOAD◆/CLEAR
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

-X-- -X-- ---- ---- ---- XX-- ---- ---- /A3◆/LOAD◆/CLEAR
X--- -X-- ---- ---- ---- -X-X ---- ---- /B3◆LOAD◆/CLEAR
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---- ---- ---- ---- ---- ---- ---- ----
---- ---- X--X X--X X--X X--X ---- ---- A3.EQ.B3◆A2.EQ.B2◆A1.EQ.B1◆A0.E
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
```

Between Limits Comparator/Register

Logic Diagram PAL16X4

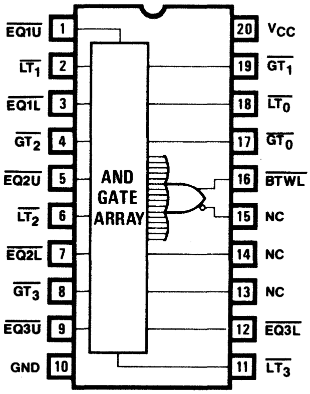


Between Limits Comparator/Logic

Design Specification PAL16C1

```
PAL16C1                                            PAL DESIGN SPECIFICATION
PAT0021                                               JOHN BIRKNER 12/18/77
BETWEEN LIMITS COMPARITOR / LOGIC

/EQ1U /LT1 /EQ1L /GT2 /EQ2U /LT2 /EQ2L /GT3 /EQ3U GND
/LT3 /EQ3L NC NC NC /BTWL /GT0 /LT0 /GT1 VCC


/BTWL = GT3 + GT2*EQ3U + GT1*EQ3U*EQ2U + GT0*EQ3U*EQ2U*EQ1U +

        LT3 + LT2*EQ3L + LT1*EQ3L*EQ2L + LT0*EQ3L*EQ2L*EQ1L



DESCRIPTION:

THE BETWEEN LIMITS LOGIC DETERMINES THE BTWL STATUS AS
A FUNCTION OF THE GT, LT AND EQ STATUS FROM THE COMPARITOR
REGISTERS.
```

PAL16C1



Logic Symbol

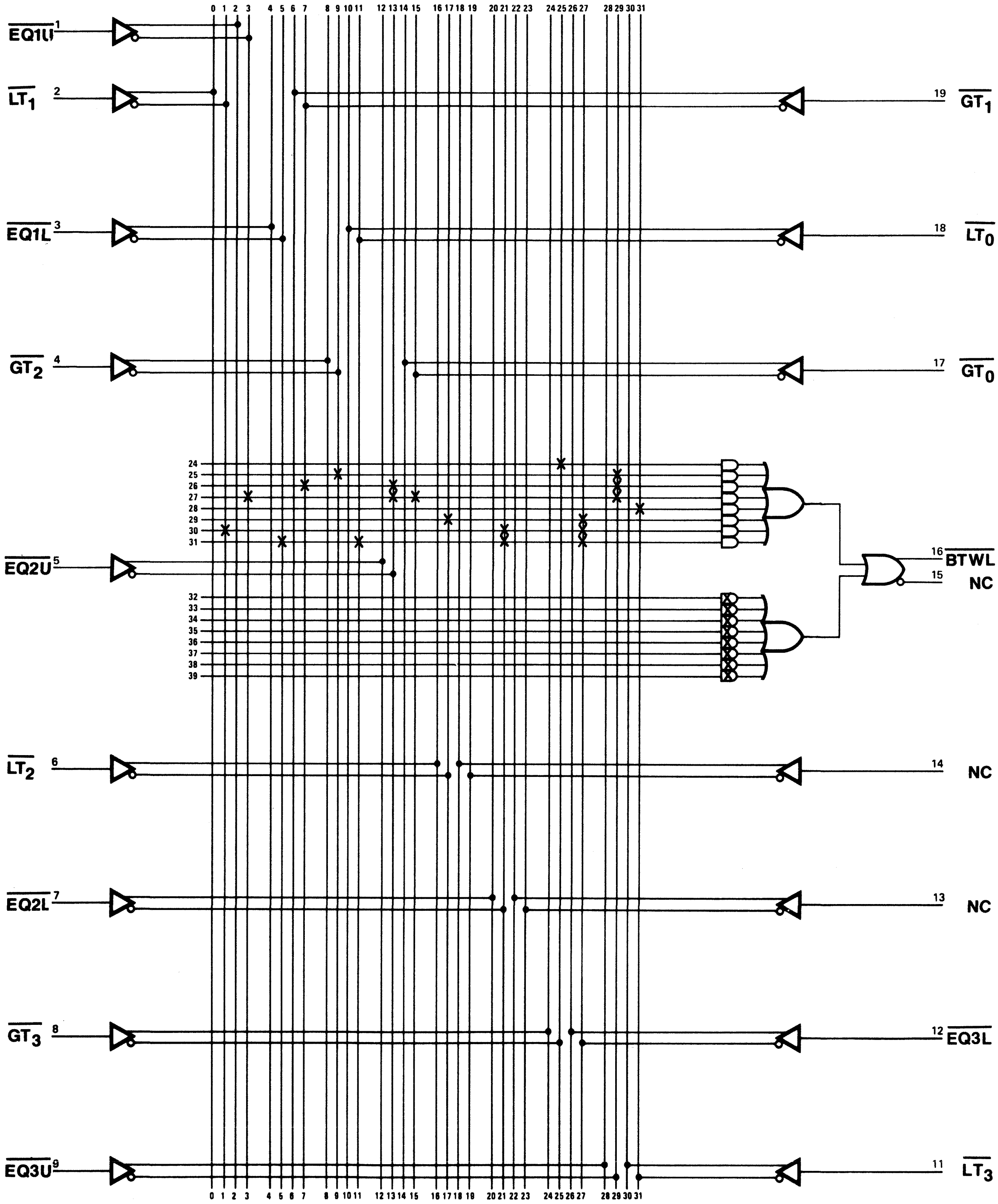
Between Limits/Comparator/Logic

Fuse Pattern PAL16C1

```
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----

---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----

---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ----

---- ---- ---- ---- ---- ---- -X-- ---- GT3
---- ---- -X-- ---- ---- ---- ---- -X-- GT2◆EQ3U
---- ---X ---- -X-- ---- ---- ---- -X-- GT1◆EQ3U◆EQ2U
---X ---- ---- -X-X ---- ---- ---- -X-- GT0◆EQ3U◆EQ2U◆EQ1U
---- ---- ---- ---- ---- ---- ---- ---X LT3
---- ---- ---- ---- -X-- ---- ---X ---- LT2◆EQ3L
-X-- ---- ---- ---- ---- -X-- ---X ---- LT1◆EQ3L◆EQ2L
---- -X-- ---X ---- ---- -X-- ---X ---- LT0◆EQ3L◆EQ2L◆EQ1L

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
```

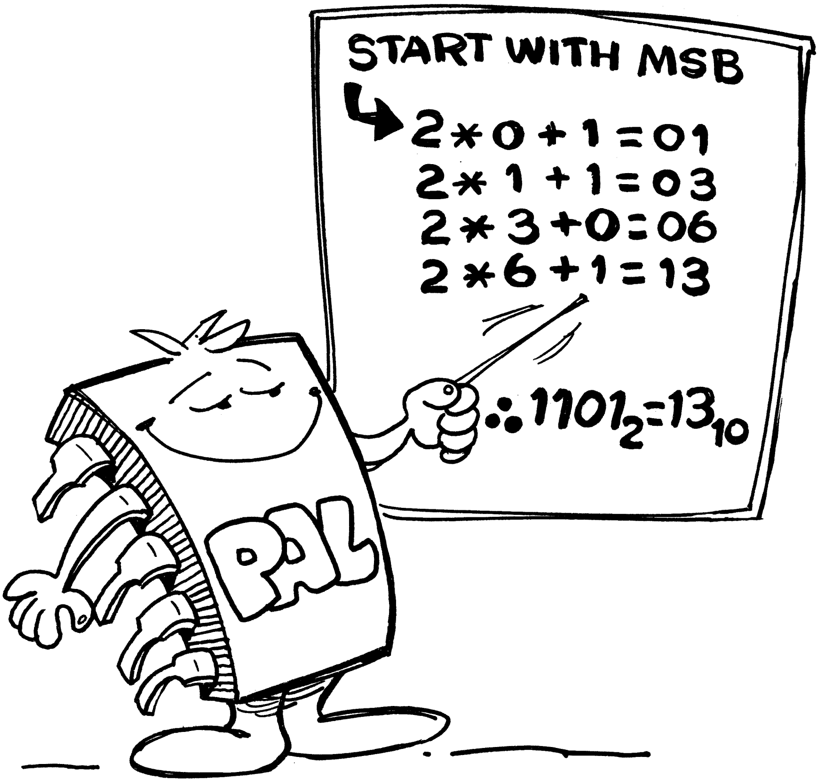
Between Limits/Comparator/Logic

Logic Diagram PAL16C1



Applications

Binary to BCD



One of the most common system requirements is the conversion between various code formats. In computer systems the normal codes used are binary, while most peripheral devices exchange data in codes based on BCD. (Both EBCDIC and ASCII codes can be considered in this class.) Whenever data is exchanged between these systems there is a high probability that some data will require code conversion. The PAL in this example performs binary to BCD conversion. In many applications this can be used to speed system operation by replacing slow software conversions with a low cost, fast hardware code converter.

cost, fast hardware code converter.

Serial Binary to Parallel BCD Conversion

The purpose of this circuit is to convert a serial stream of binary data into a parallel BCD representation as depicted by Figure 1.

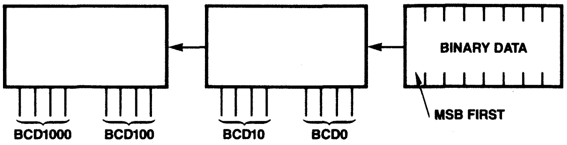


Figure 1. Conceptual Diagram of Binary to BCD Converter

Couleur's Technique (BIDEC)

In this conversion technique (Ref. 1), the input binary data is shifted left (starting with the MSB) into the BCD register. The beauty of this method is that after each clock pulse, the BCD output contains correct BCD representation for the "relative" binary data shifted so far. We illustrate the last statement in Figure 2.

Logic Design

The overall conversion problem can be segmented into four-bit binary blocks. Each block represents one BCD digit and is expandable by C_{IN} and C_{OUT}. The BCD building block, is shown in Figure 3 as a state machine.

The combinational network can be designed from a "next-state" truth table. The truth table can be constructed by observing that a left shift is a multiplication by 2, and a carryin adds 1 to the LSB.

In equation form:

BCD (next state): = 2∗ BCD (present state) + C_{IN}
Of course, if it were binary it will be simply,

$$Q_n: = Q_{n-1}$$

However, BCD requires some corrections as will be shown shortly. For simplicity we analyze the three relevent cases:

a) BCD (present state): = 0 – 4
then BCD (next state): = 0 – 8 which are representable in BCD format

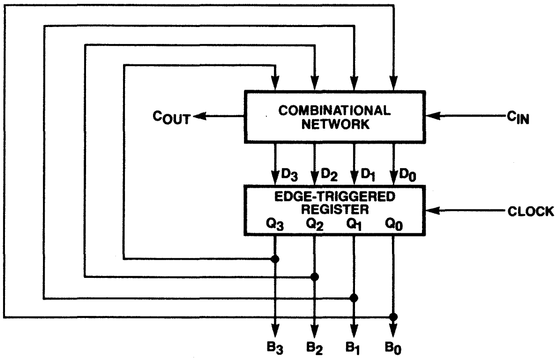


Figure 3, BCD Converter Building Block.

b) BCD (present state): = 5 – 9
then BCD (next state): = 10 – 18 which are not representable in one BCD digit, and a carry out is generated.
e.g. If BCD (present state): = 6
then BCD (next state): = 2 and C_{OUT} = 1

c) If C_{IN} = 1 then it is simply shifted into the LSB of the BCD digit (the old LSB was shifted left leaving a zero in its original position).
Thus, regardless of the BCD (present state) value, the following is true:

BO (next state): = C_{IN}

Truth Table

The preceding discussion is summarized by Tables 1 and 2.

PRESENT STATE B_3-B_0	NEXT STATE B_3-B_0	C_{OUT}
0-4	0-8	0
5-9	0-8	1
10-15	DON'T CARE	DON'T CARE

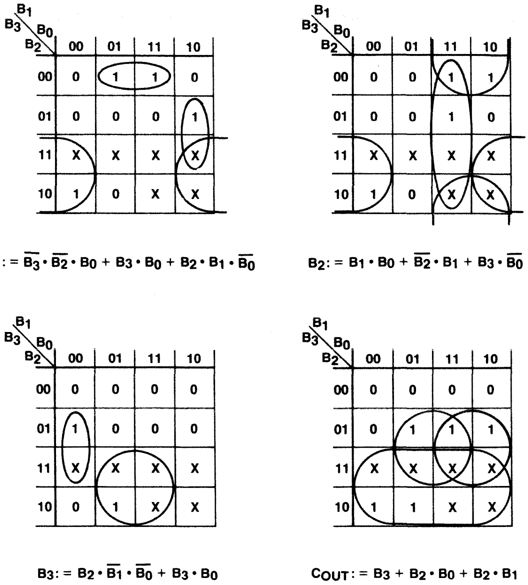
Table 1, General Truth Table.

n	PRESENT STATE				NEXT STATE				C_{OUT}
	B_3	B_2	B_1	B_0	B_3	B_2	B_1	B_0	
0	0	0	0	0	0	0	0	C_{IN}	
1	0	0	0	1	0	0	1	C_{IN}	0
2	0	0	1	0	0	1	0	C_{IN}	0
3	0	0	1	1	0	1	1	C_{IN}	0
4	0	1	0	0	1	0	0	C_{IN}	0
5	0	1	0	1	0	0	0	C_{IN}	1
6	0	1	1	0	0	0	1	C_{IN}	1
7	0	1	1	1	0	1	0	C_{IN}	1
8	1	0	0	0	0	1	1	C_{IN}	1
9	1	0	0	1	1	0	0	C_{IN}	1
10-15					X	X	X	X	X

Table 2: Detailed Truth Table.

												128	64	32	16	8	4	2	1	← RELATIVE WEIGHT BEFORE LAST SHIFT
												64	32	16	8	4	2	1		← RELATIVE WEIGHT BEFORE 7TH SHIFT
												32	16	8	4	2	1			← RELATIVE WEIGHT BEFORE 6TH SHIFT
												16	8	4	2	1				← RELATIVE WEIGHT BEFORE 5TH SHIFT
												8	4	2	1					← RELATIVE WEIGHT BEFORE 4TH SHIFT
												4	2	1						← RELATIVE WEIGHT BEFORE 3RD SHIFT
												2	1							← RELATIVE WEIGHT BEFORE 2ND SHIFT
CONVERSION REGISTER												1								← RELATIVE WEIGHT BEFORE 1ST SHIFT
10^2				10^1				10^0				1	0	0	1	0	0	0	0	← BINARY INPUT
											1	0	0	1	0	0	0	0		← 1 SHIFTED IN
										1	0	0	1	0	0	0	0			← 2 SHIFTED IN
									1	0	0	1	0	0	0	0				← 4 SHIFTED IN
								1	0	0	1	0	0	0	0					← 9 SHIFTED IN
							1	1	0	0	0	0	0	0						← 18 SHIFTED IN
						1	1	0	1	1	0	0	0							← 36 SHIFTED IN
					1	1	1	0	0	1	0	0								← 72 SHIFTED IN
			1	0	1	0	0	0	1	0	0									← 144 SHIFTED IN
1				4				4												

Figure 2: Tabular Representation of the Conversion Cycle.



$B_1 : = \overline{B_3} \cdot \overline{B_2} \cdot B_0 + B_3 \cdot B_0 + B_2 \cdot B_1 \cdot \overline{B_0}$

$B_2 : = B_1 \cdot B_0 + \overline{B_2} \cdot B_1 + B_3 \cdot \overline{B_0}$

$B_3 : = B_2 \cdot \overline{B_1} \cdot \overline{B_0} + B_3 \cdot B_0$

$C_{OUT} : = B_3 + B_2 \cdot B_0 + B_2 \cdot B_1$

Figure 4, Karnaugh Maps

Pal Implementation

The PAL16R8 implements two BCD digits. One of the pins is assigned to the clear (CLR) function. The BCD conversion register must be initialized to zero before shifting of the binary input data is started. The eight output registers are assigned to the two BCD digits.

At this point, it seems that we are short of one output pin for the C_{OUT} in expanding to more BCD digits. However, the basic equations indicate that C_{OUT} is a function of the four preceding BCD bits. Therefore, by inputting these four bits to the next stage, the C_{OUT} is derived internally by the latter stage. A similar trick is used in each chip to cascade internally.

This expansion solution implies that in the least significant BCD stage the equation is:

(1) BO: $= C_{IN}$

whereas in later stages the equation is:

(2) BO: $= C_{13} + C_{12} \cdot C_{10} + C_{12} \cdot C_{11}$

where the C terms are driven by the corresponding B terms of a preceding stage. However, in order to have a universal solution, we OR the two equations. If the PAL is used as the least significant stage C_{10}, C_{11}, C_{12} and C_{13} are grounded and equation (1) holds. If the PAL is used as an intermediate stage, C_{IN} is grounded and equation (2) holds.

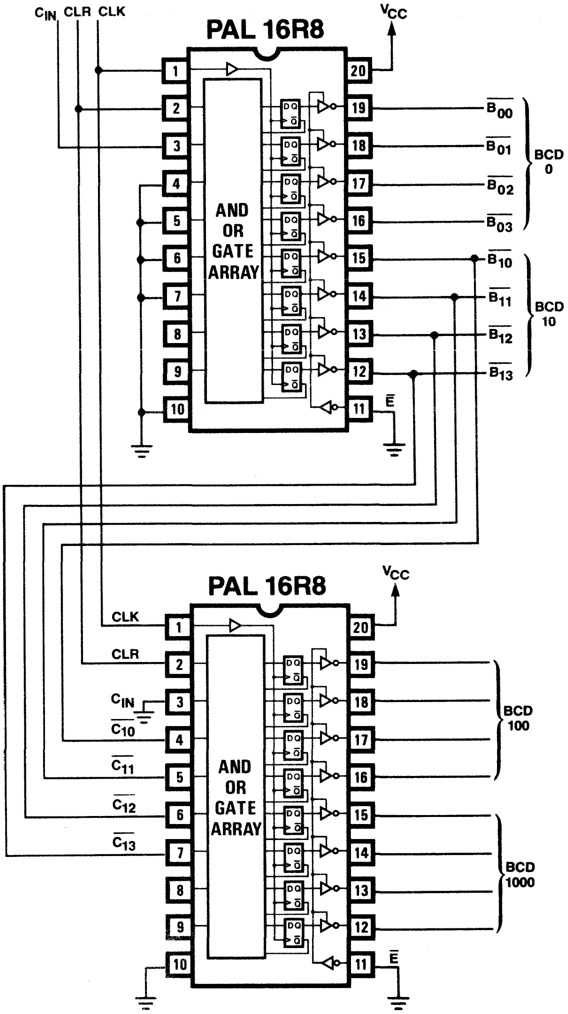


Figure 5, Logic Schematic

Summary

A similar algorithm was described in Ref. 2, where the two BCD digits were implemented with four ICs and could be clocked at 80 ns. Here we described one chip implementation that can be clocked at 60 ns.

References

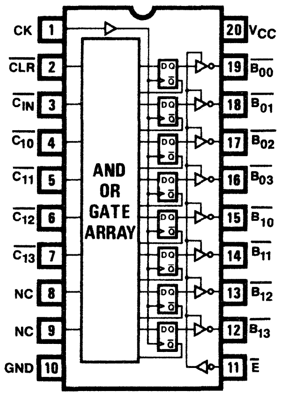
1. *"Binary to BCD Conversion Techniques"* by B. MacDonald, EDN Dec. 1, 1969.
2. *"Special PROM Mode Effects Binary to BCD Converter"*, by D.M. Brockman, Electronics.

Binary to BCD Converter

Design Specification PAL16R8

```
PAL16R8                                                  PAL DESIGN SPECIFICATION
PAT014                                                       SHLOMO WASER 1/6/78
BINARY TO BCD CONVERTER

CK /CLR CIN /C10 /C11 /C12 /C13  NC NC GND
/E /B13 /B12 /B11 /B10 /B03 /B02 /B01 /B00 VCC


B00 := /CLR*CIN + /CLR*C13 + /CLR*C12*C10 + /CLR*C12*C11

B01 := /CLR*/B03*/B02*B00 + /CLR*B03*/B00 + /CLR*B02*B01*/B00

B02 := /CLR*B01*B00 + /CLR*/B02*B01 + /CLR*B03*/B00

B03 := /CLR*B02*/B01*/B00 + /CLR*B03*B00

B10 := /CLR*B03 + /CLR*B02*B00 + /CLR*B02*B01

B11 := /CLR*/B13*/B12*B10 + /CLR*B13*/B10 + /CLR*B12*B11*/B10

B12 := /CLR*B11*B10 + /CLR*/B12*B11 + /CLR*B13*/B10

B13 := /CLR*B12*/B11*/B10 + /CLR*B13*B10



DESCRIPTION: SEE TEXT
```

PAL16R8



Logic Symbol

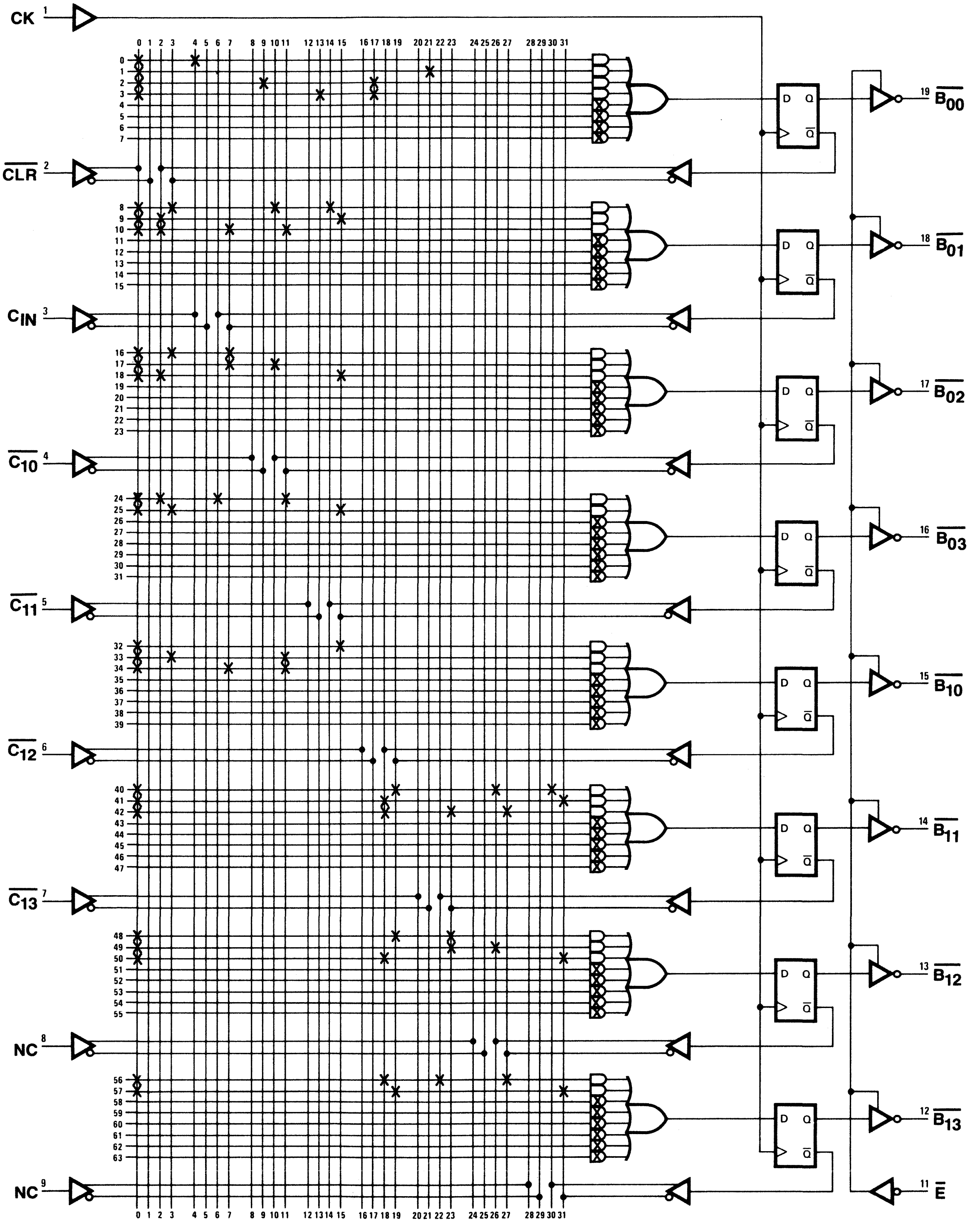
Binary to BCD Converter

Fuse Pattern PAL16R8

```
X--- X--- ---- ---- ---- ---- ---- ---- /CLR◆CIN
X--- ---- ---- ---- ---- -X-- ---- ---- /CLR◆C13
X--- ---- -X-- ---- -X-- ---- ---- ---- /CLR◆C12◆C10
X--- ---- ---- -X-- -X-- ---- ---- ---- /CLR◆C12◆C11
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

X--X ---- --X- --X- ---- ---- ---- ---- /CLR◆/B03◆/B02◆B00
X-X- ---- ---- ---X ---- ---- ---- ---- /CLR◆B03◆/B00
X-X- ---X ---X ---- ---- ---- ---- ---- /CLR◆B02◆B01◆/B00
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

X--X ---X ---- ---- ---- ---- ---- ---- /CLR◆B01◆B00
X--- ---X --X- ---- ---- ---- ---- ---- /CLR◆/B02◆B01
X-X- ---- ---- ---X ---- ---- ---- ---- /CLR◆B03◆/B00
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

X-X- --X- ---X ---- ---- ---- ---- ---- /CLR◆B02◆/B01◆/B00
X--X ---- ---- ---X ---- ---- ---- ---- /CLR◆B03◆B00
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

X--- ---- ---- ---X ---- ---- ---- ---- /CLR◆B03
X--X ---- ---X ---- ---- ---- ---- ---- /CLR◆B02◆B00
X--- ---X ---X ---- ---- ---- ---- ---- /CLR◆B02◆B01
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

X--- ---- ---- ---- ---X ---- --X- --X- /CLR◆/B13◆/B12◆B10
X--- ---- ---- ---- --X- ---- ---- ---X /CLR◆B13◆/B10
X--- ---- ---- ---- --X- ---X ---X ---- /CLR◆B12◆B11◆/B10
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

X--- ---- ---- ---- ---X ---X ---- ---- /CLR◆B11◆B10
X--- ---- ---- ---- ---- ---X --X- ---- /CLR◆/B12◆B11
X--- ---- ---- ---- --X- ---- ---- ---X /CLR◆B13◆/B10
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

X--- ---- ---- ---- --X- --X- ---X ---- /CLR◆B12◆/B11◆/B10
X--- ---- ---- ---- ---X ---- ---- ---X /CLR◆B13◆B10
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
```

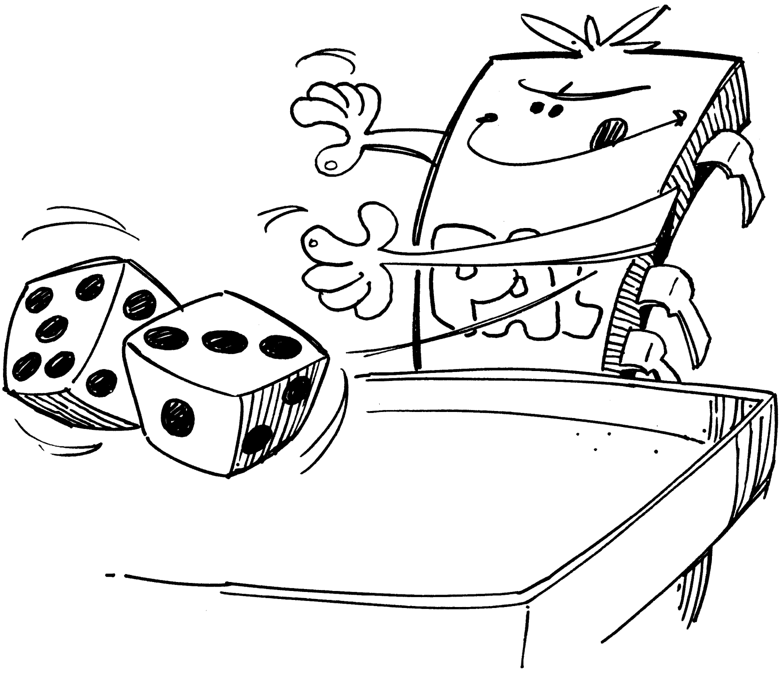
Binary to BCD Converter

Logic Diagram PAL16R8



Applications

Electronic Dice Game



A dice game can be implemented using a simple state sequencer and an oscillator. When the oscillator is running (ROLL), the dual modulo six counter increments through all possible states. When the free running oscillator is stopped, a random count will appear on the outputs.

In one die there are six states (1 through 6). In a set of dice there are six states on the one die for each of the six states of the other (36 states in all). Since the two dice are very similar it is possible to design one die and use the equations from it to design the second. Two designs will be shown. First, an eight chip SSI/MSI solution, then, a single chip PAL solution.

Logic Design Using Standard TTL

In one die, seven LEDs make up the display (Figure 1). Notice they can be connected such that only four lines are required to drive them. The LEDs are turned on when the appropriate line is driven low. Since there are four lines to be driven it is necessary to use four D-type flip-flops for each die. For reference the outputs of the flip-flops are labeled Q_1-Q_4 and the inputs are labeled D_1-D_4 (Figure 2). By using the inverting output of the Flip-Flop we can use positive logic in the design. That is, a logical "1" at a Q output represents an LED being turned on.

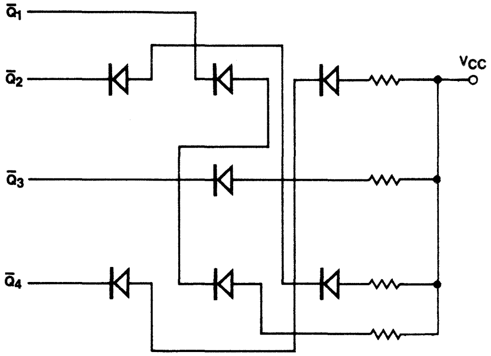


Figure 1

The present state of table 1 shows the preferred sequence in which the LEDs should turn on. The next state shows the conditions necessary to increment when clocked. From these two tables the Karnaugh Maps of Figure 3 were made. Using the Karnaugh Maps the following equations are obtained;

$$D_1 = \overline{Q_1}\, Q_2\, Q_3 \qquad D_2 = \overline{Q_1}\, Q_3 + \overline{Q_1}\, Q_4$$

$$D_3 = \overline{Q_3} \qquad D_4 = \overline{Q_1}\, Q_2 + \overline{Q_1}\, Q_4$$

These equations satisfy the requirements for one die. By substituting Q_5-Q_8 for Q_1-Q_4 and D_5-D_8 for D_1-D_4 we have the following equations;

$$D_5 = \overline{Q_5}\, Q_6\, Q_7 \qquad D_6 = \overline{Q_5}\, Q_7 + \overline{Q_5}\, Q_8$$

$$D_7 = \overline{Q_7} \qquad D_8 = \overline{Q_5}\, \overline{Q_7} + \overline{Q_5}\, Q_8$$

Looking at the Karnal Maps (Figure 3), it may be noticed that the simplest logic equations were not generated. This was to insure a path to a valid state from all invalid states.

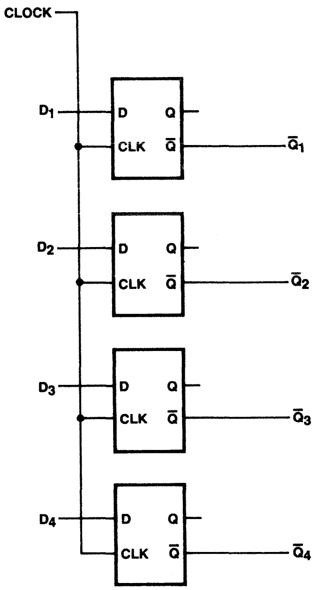
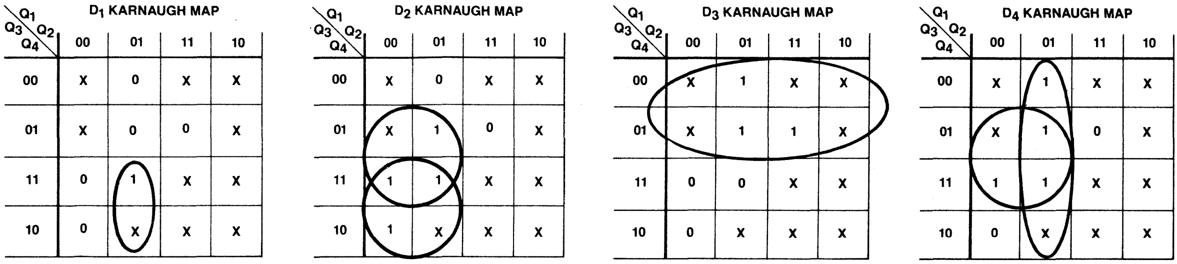


Figure 2

	PRESENT STATE				NEXT STATE			
STATE	Q_4	Q_3	Q_2	Q_1	D_4	D_3	D_2	D_1
1	0	1	0	0	0	0	1	0
2	0	0	1	0	1	1	0	0
3	1	1	0	0	1	0	1	0
4	1	0	1	0	1	1	1	0
5	1	1	1	0	1	0	1	1
6	1	0	1	1	0	1	0	0

Table 1



D₁ KARNAUGH MAP

Q_3Q_4 \ Q_1Q_2	00	01	11	10
00	X	0	X	X
01	X	0	0	X
11	0	1	X	X
10	0	X	X	X

D₂ KARNAUGH MAP

Q_3Q_4 \ Q_1Q_2	00	01	11	10
00	X	0	X	X
01	X	1	0	X
11	1	1	X	X
10	1	X	X	X

D₃ KARNAUGH MAP

Q_3Q_4 \ Q_1Q_2	00	01	11	10
00	X	1	X	X
01	X	1	1	X
11	0	0	X	X
10	0	X	X	X

D₄ KARNAUGH MAP

Q_3Q_4 \ Q_1Q_2	00	01	11	10
00	X	1	X	X
01	X	1	0	X
11	1	1	X	X
10	0	X	X	X

Note: X means Don't Care

Figure 3

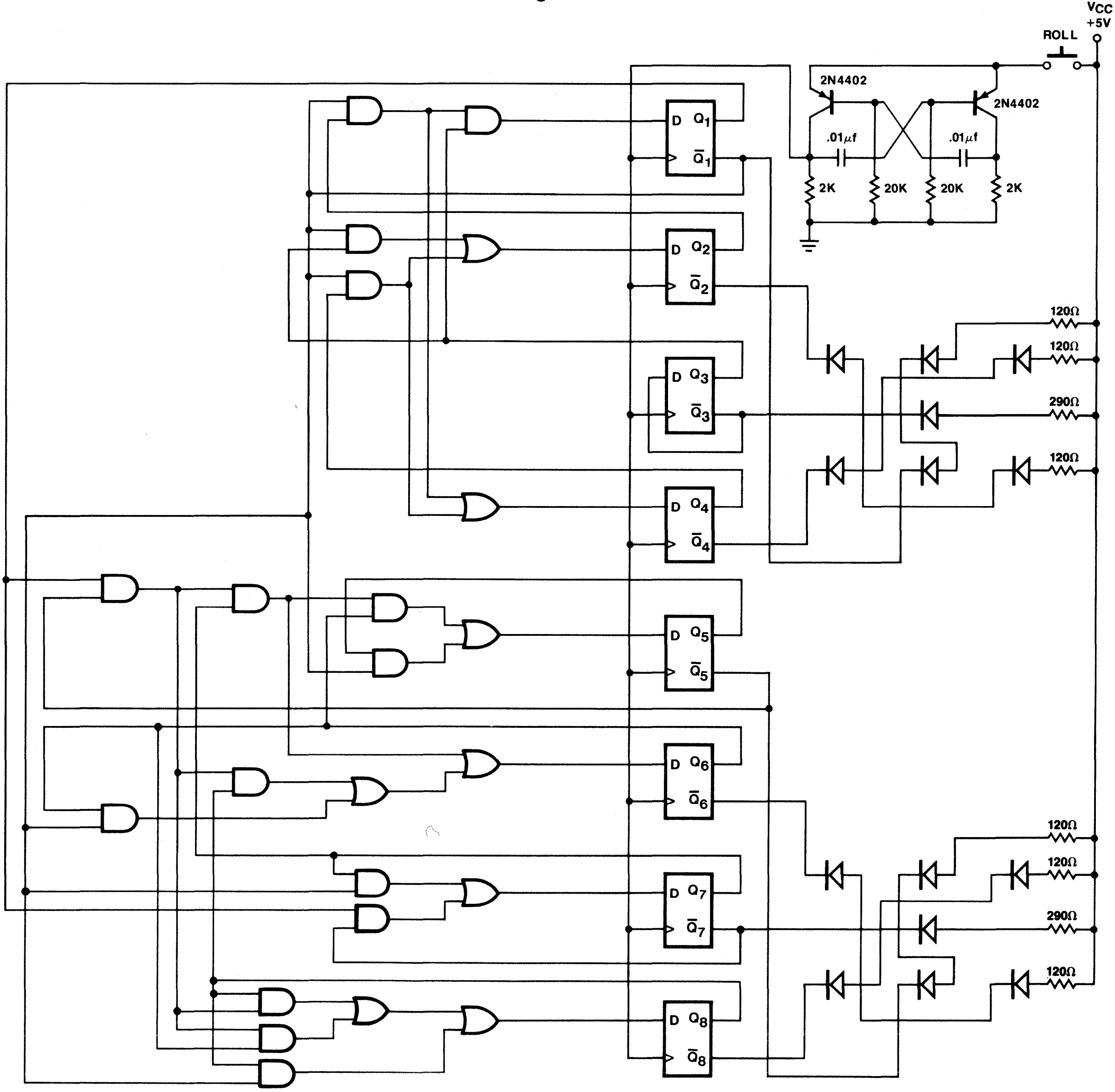


Figure 4

However, since this is a synchronous design the clocks of the two die are common. If the same equations are used for both die there will be only six different states. To get around this the first die is allowed to go through each of the six states incrementing with each clock. The second die is inhibited from incrementing except when the first die goes from the 6th state to the 1st state. At this time the second die is allowed to increment one time. Looking at the present state of table 1 it is noticed that whenever output Q_1 is high, the next clock should increment the second die. Whenever Q_1 is low the second die should remain the same. From this we now write all the equations.

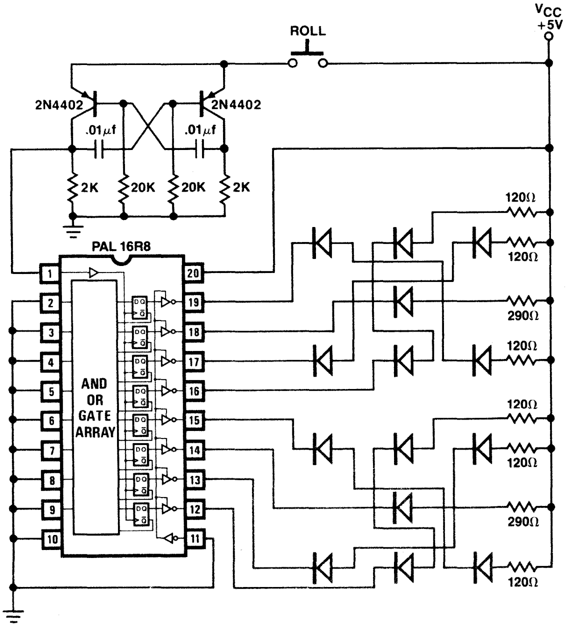
$$D_1 = \overline{Q_1}\, Q_2\, Q_3$$
$$D_2 = \overline{Q_1}\, Q_3 + \overline{Q_1}\, Q_4$$
$$D_3 = \overline{Q_3}$$
$$D_4 = \overline{Q_1}\, Q_2 + \overline{Q_1}\, Q_4$$
$$D_5 = \overline{Q_1}\, Q_5 + Q_1\, \overline{Q_5}\, Q_6\, Q_7$$
$$D_6 = \overline{Q_1}\, Q_6 + Q_1\, \overline{Q_5}\, Q_7 + Q_1\, \overline{Q_5}\, Q_8$$
$$D_7 = \overline{Q_1}\, Q_7 + Q_1\, \overline{Q_7}$$
$$D_8 = \overline{Q_1}\, Q_8 + Q_1\, \overline{Q_5}\, Q_6 + Q_1\, \overline{Q_5}\, Q_8$$

From these equations the logic diagram can be drawn. (Figure 4)

Logic Design Using PALs

The design requires 8 registered outputs. Looking at the PAL Data Sheet we determine that a PAL16R8 best suits this application. The equations developed above can be used here without change. The PAL Design Specification shows the implementation of these equations.

Using this arbitrary pinout, however, makes the PC board layout awkward. Thus, another PAL Design Specification is shown with the pinouts chosen to convenience the PC board layout. This design is chosen for the circuit diagram of figure 5.



Applications

Rules for CRAPS

The following is a set of rules that apply whether you are playing in Las Vegas with dice or at home with a PAL.

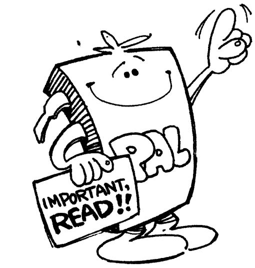
The first roll is called the come-out and you win on a 7 or 11 or you "crap-out" on a 2 (snake eyes), 3 (ace caught a deuce) or 12 (box cars). If none of the above happens you will have rolled a number between 4 and 10. Mark this number well, you will need to roll it again to win. At this point no "crap" can hurt you, but unless you've programmed your PAL right, a seven can. Normal probabilities in 36 throws:

7 will appear	6 times
6	5
8	5
5	4
9	4
4	3
10	3
3	2
11	2
2	1
12	1

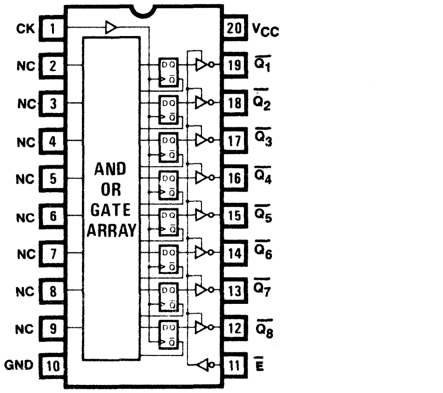
Electronic Dice Game

Design Specification PAL16R8

```
PAL16R8                                              PAL DESIGN SPECIFICATION
PAT0001                                                    ED VETTER    12/8/77
ELECTRONIC DICE GAME

CK NC NC NC NC NC NC NC NC GND /E /Q8 /Q7 /Q6 /Q5 /Q4 /Q3 /Q2 /Q1 VCC


Q1 := /Q1*Q2*Q3

Q2 := /Q1*Q3 + /Q1*Q4

Q3 := /Q3

Q4 := /Q1*Q2 + /Q1*Q4

Q5 := Q1*/Q5*Q6*Q7 + /Q1*Q5

Q6 := Q1*/Q5*Q7 + Q1*/Q5*Q8 + /Q1*Q6

Q7 := Q1*/Q7 + /Q1*Q7

Q8 := Q1*/Q5*Q6 + Q1*/Q5*Q8 +/Q1*Q8


DESCRIPTION:

THE DUAL MODULO-SIX COUNTER INCREMENTS ON THE RISING EDGE OF THE CLOCK INPUT
(CK). THE THREE-STATE OUTPUTS ARE HIGH-Z WHEN THE ENABLE LINE (/E) IS HIGH AND
ENABLED WHEN ENABLE LINE (/E) IS LOW. THERE ARE 36 DIFFERENT STATES TO THE
COUNT SEQUENCE, EACH STATE CORRESPONDS TO ONE OF THE NUMBER COMBINATIONS
TO BE DISPLAYED ON THE DICE.  (SEE NEXT PAGE)
```



PAL16R8



Electronic Dice Game

State Sequence Table PAL16R8

```
-----------------------------------------------------------------------
! Q    Q    Q    Q    Q    Q    Q    Q !  NUMBER  DISPLAYED          !
! 1    2    3    4    5    6    7    8 !  DIE 2   -    DIE 1         !
-----------------------------------------------------------------------
! H    H    L    H    H    H    L    H !    1     -      1            !
! H    L    H    H    H    H    L    H !    2     -      1            !
! H    H    L    L    H    H    L    H !    3     -      1            !
! H    L    H    L    H    H    L    H !    4     -      1            !
! H    L    L    L    H    H    L    H !    5     -      1            !
! L    L    H    L    H    H    L    H !    6     -      1            !
-----------------------------------------------------------------------
! H    H    L    H    H    L    H    H !    1     -      2            !
! H    L    H    H    H    L    H    H !    2     -      2            !
! H    H    L    L    H    L    H    H !    3     -      2            !
! H    L    H    L    H    L    H    H !    4     -      2            !
! H    L    L    L    H    L    H    H !    5     -      2            !
! L    L    H    L    H    L    H    H !    6     -      2            !
-----------------------------------------------------------------------
! H    H    L    H    H    H    L    L !    1     -      3            !
! H    L    H    H    H    H    L    L !    2     -      3            !
! H    H    L    L    H    H    L    L !    3     -      3            !
! H    L    H    L    H    H    L    L !    4     -      3            !
! H    L    L    L    H    H    L    L !    5     -      3            !
! L    L    H    L    H    H    L    L !    6     -      3            !
-----------------------------------------------------------------------
! H    H    L    H    H    L    H    L !    1     -      4            !
! H    L    H    H    H    L    H    L !    2     -      4            !
! H    H    L    L    H    L    H    L !    3     -      4            !
! H    L    H    L    H    L    H    L !    4     -      4            !
! H    L    L    L    H    L    H    L !    5     -      4            !
! L    L    H    L    H    L    H    L !    6     -      4            !
-----------------------------------------------------------------------
! H    H    L    H    H    L    L    L !    1     -      5            !
! H    L    H    H    H    L    L    L !    2     -      5            !
! H    H    L    L    H    L    L    L !    3     -      5            !
! H    L    H    L    H    L    L    L !    4     -      5            !
! H    L    L    L    H    L    L    L !    5     -      5            !
! L    L    H    L    H    L    L    L !    6     -      5            !
-----------------------------------------------------------------------
! H    H    L    H    L    L    H    L !    1     -      6            !
! H    L    H    H    L    L    H    L !    2     -      6            !
! H    H    L    L    L    L    H    L !    3     -      6            !
! H    L    H    L    L    L    H    L !    4     -      6            !
! H    L    L    L    L    L    H    L !    5     -      6            !
! L    L    H    L    L    L    H    L !    6     -      6            !
-----------------------------------------------------------------------
```

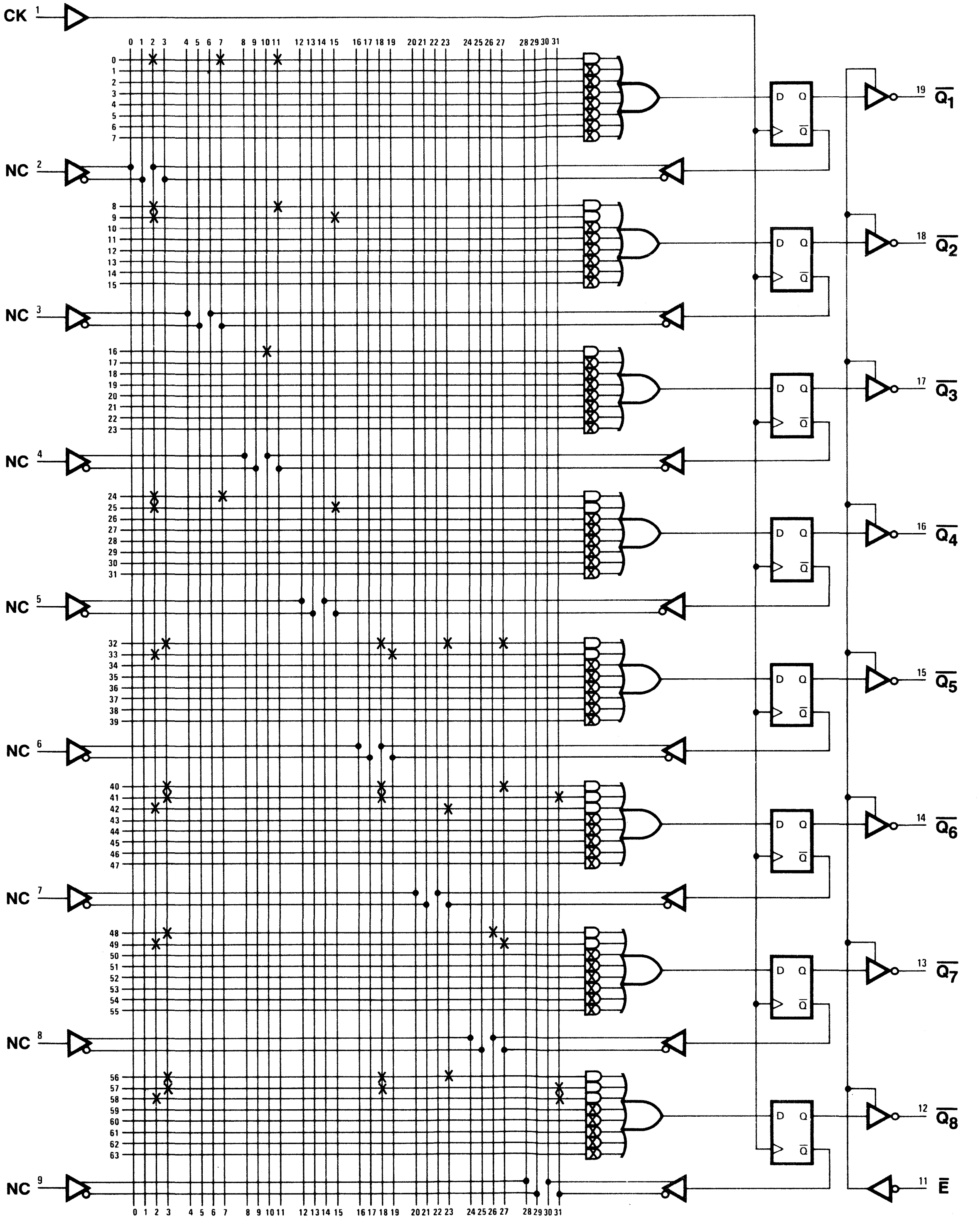
Electronic Dice Game

Fuse Pattern PAL16R8

```
--X- ---X ---X ---- ---- ---- ---- ---- /Q1♦Q2♦Q3
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

--X- ---- ---X ---- ---- ---- ---- ---- /Q1♦Q3
--X- ---- ---- ---X ---- ---- ---- ---- /Q1♦Q4
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---- ---- --X- ---- ---- ---- ---- ---- /Q3
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

--X- ---X ---- ---- ---- ---- ---- ---- /Q1♦Q2
--X- ---- ---- ---X ---- ---- ---- ---- /Q1♦Q4
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---X ---- ---- ---- --X- ---X ---X ---- Q1♦/Q5♦Q6♦Q7
--X- ---- ---- ---- ---X ---- ---- ---- /Q1♦Q5
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---X ---- ---- ---- --X- ---- ---X ---- Q1♦/Q5♦Q7
---X ---- ---- ---- --X- ---- ---- ---X Q1♦/Q5♦Q8
--X- ---- ---- ---- ---- ---X ---- ---- /Q1♦Q6
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---X ---- ---- ---- ---- ---- --X- ---- Q1♦/Q7
--X- ---- ---- ---- ---- ---- ---X ---- /Q1♦Q7
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---X ---- ---- ---- --X- ---X ---- ---- Q1♦/Q5♦Q6
---X ---- ---- ---- --X- ---- ---- ---X Q1♦/Q5♦Q8
--X- ---- ---- ---- ---- ---- ---- ---X /Q1♦Q8
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
```

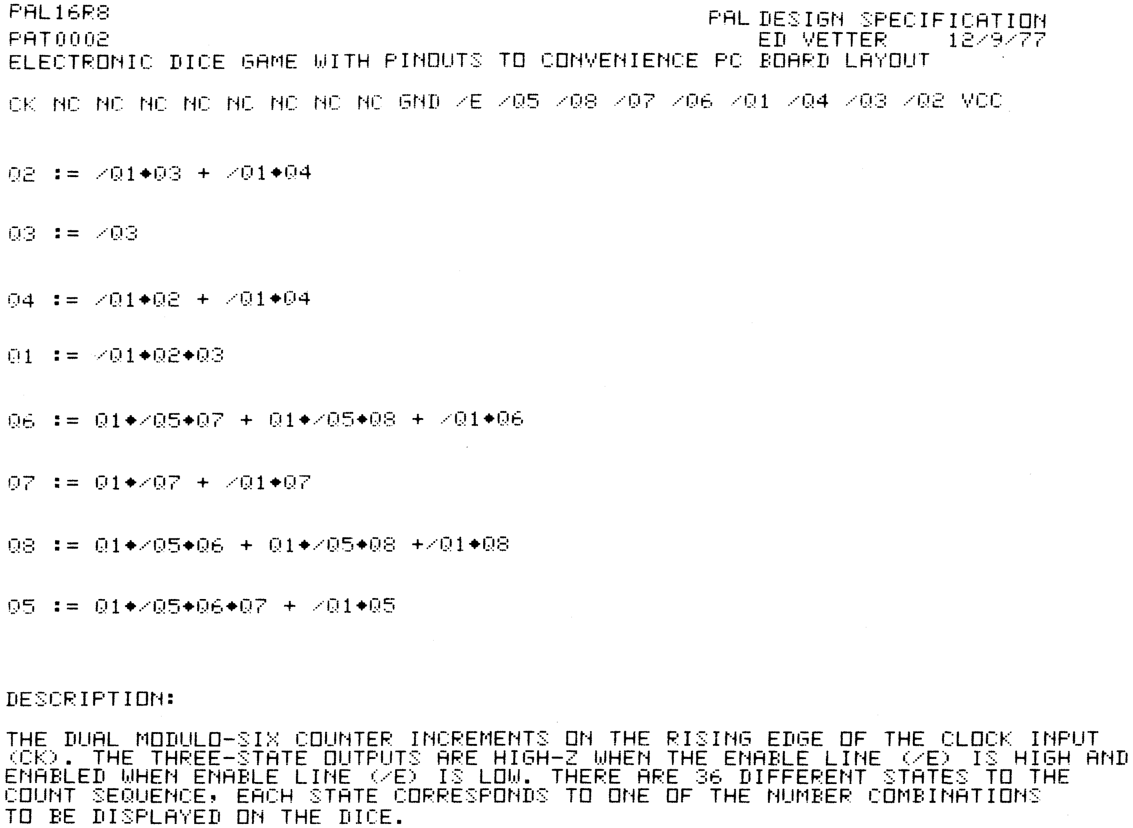
Electronic Dice Game

Logic Diagram PAL16R8



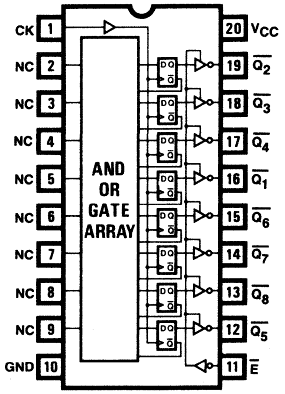
Electronic Dice Game with Pinouts to Convenience PC Board Layout

Design Specification PAL16R8



```
PAL16R8                                               PAL DESIGN SPECIFICATION
PAT0002                                                   ED VETTER      12/9/77
ELECTRONIC DICE GAME WITH PINOUTS TO CONVENIENCE PC BOARD LAYOUT

CK NC NC NC NC NC NC NC NC GND /E /Q5 /Q8 /Q7 /Q6 /Q1 /Q4 /Q3 /Q2 VCC


Q2 := /Q1*Q3 + /Q1*Q4

Q3 := /Q3

Q4 := /Q1*Q2 + /Q1*Q4

Q1 := /Q1*Q2*Q3

Q6 := Q1*/Q5*Q7 + Q1*/Q5*Q8 + /Q1*Q6

Q7 := Q1*/Q7 + /Q1*Q7

Q8 := Q1*/Q5*Q6 + Q1*/Q5*Q8 +/Q1*Q8

Q5 := Q1*/Q5*Q6*Q7 + /Q1*Q5


DESCRIPTION:

THE DUAL MODULO-SIX COUNTER INCREMENTS ON THE RISING EDGE OF THE CLOCK INPUT
(CK). THE THREE-STATE OUTPUTS ARE HIGH-Z WHEN THE ENABLE LINE (/E) IS HIGH AND
ENABLED WHEN ENABLE LINE (/E) IS LOW. THERE ARE 36 DIFFERENT STATES TO THE
COUNT SEQUENCE, EACH STATE CORRESPONDS TO ONE OF THE NUMBER COMBINATIONS
TO BE DISPLAYED ON THE DICE.
```

PAL16R8



Logic Symbol

Electronic Dice Game with Pinouts to Convenience PC Board Layout

State Sequence Table PAL16R8

Q	Q	Q	Q	Q	Q	Q	Q	NUMBER DISPLAYED		
2	3	4	1	6	7	8	5	DIE 2	-	DIE 1
H	L	H	H	H	L	H	H	1	-	1
L	H	H	H	H	L	H	H	2	-	1
H	L	L	H	H	L	H	H	3	-	1
L	H	L	H	H	L	H	H	4	-	1
L	L	L	H	H	L	H	H	5	-	1
L	H	L	L	H	L	H	H	6	-	1
H	L	H	H	L	H	H	H	1	-	2
L	H	H	H	L	H	H	H	2	-	2
H	L	L	H	L	H	H	H	3	-	2
L	H	L	H	L	H	H	H	4	-	2
L	L	L	H	L	H	H	H	5	-	2
L	H	L	L	L	H	H	H	6	-	2
H	L	H	H	H	L	L	H	1	-	3
L	H	H	H	H	L	L	H	2	-	3
H	L	L	H	H	L	L	H	3	-	3
L	H	L	H	H	L	L	H	4	-	3
L	L	L	H	H	L	L	H	5	-	3
L	H	L	L	H	L	L	H	6	-	3
H	L	H	H	L	H	L	H	1	-	4
L	H	H	H	L	H	L	H	2	-	4
H	L	L	H	L	H	L	H	3	-	4
L	H	L	H	L	H	L	H	4	-	4
L	L	L	H	L	H	L	H	5	-	4
L	H	L	L	L	H	L	H	6	-	4
H	L	H	H	L	L	L	H	1	-	5
L	H	H	H	L	L	L	H	2	-	5
H	L	L	H	L	L	L	H	3	-	5
L	H	L	H	L	L	L	H	4	-	5
L	L	L	H	L	L	L	H	5	-	5
L	H	L	L	L	L	L	H	6	-	5
H	L	H	H	L	H	L	L	1	-	6
L	H	H	H	L	H	L	L	2	-	6
H	L	L	H	L	H	L	L	3	-	6
L	H	L	H	L	H	L	L	4	-	6
L	L	L	H	L	H	L	L	5	-	6
L	H	L	L	L	H	L	L	6	-	6

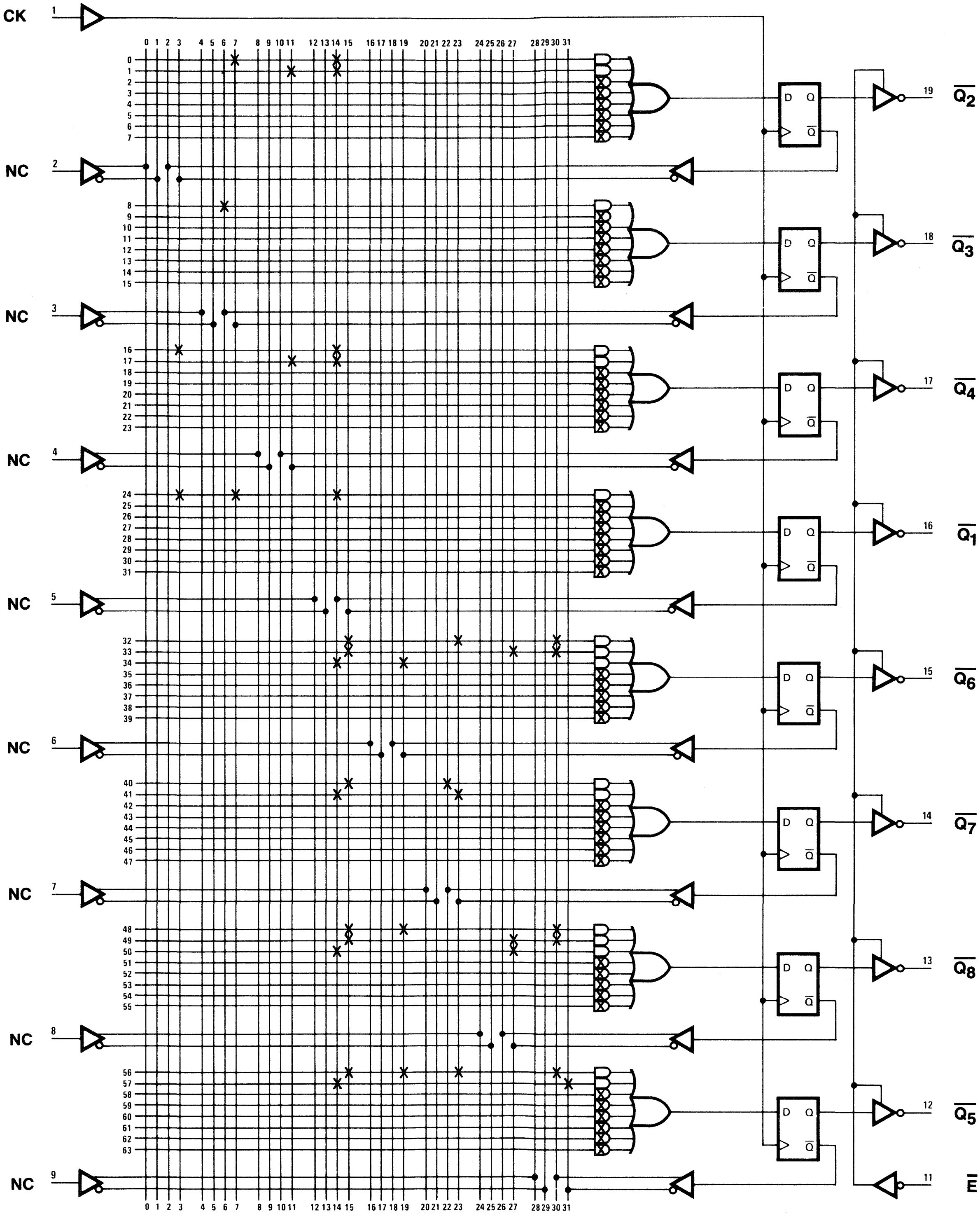
Electronic Dice Game with Pinouts to Convenience PC Board Layout

Fuse Pattern PAL16R8

```
---- ---X ---- --X- ---- ---- ---- ---- /Q1*Q3
---- ---- ---X --X- ---- ---- ---- ---- /Q1*Q4
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---- --X- ---- ---- ---- ---- ---- ---- /Q3
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---X ---- ---- --X- ---- ---- ---- ---- /Q1*Q2
---- ---- ---X --X- ---- ---- ---- ---- /Q1*Q4
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---X ---X ---- --X- ---- ---- ---- ---- /Q1*Q2*Q3
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---- ---- ---- ---X ---- ---X ---- --X- Q1*/Q5*Q7
---- ---- ---- ---X ---- ---- ---X --X- Q1*/Q5*Q8
---- ---- ---- --X- ---X ---- ---- ---- /Q1*Q6
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---- ---- ---- ---X ---- --X- ---- ---- Q1*/Q7
---- ---- ---- --X- ---- ---X ---- ---- /Q1*Q7
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---- ---- ---- ---X ---X ---- ---- --X- Q1*/Q5*Q6
---- ---- ---- ---X ---- ---- ---X --X- Q1*/Q5*Q8
---- ---- ---- --X- ---- ---- ---X ---- /Q1*Q8
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---- ---- ---- ---X ---X ---X ---- --X- Q1*/Q5*Q6*Q7
---- ---- ---- --X- ---- ---- ---- ---X /Q1*Q5
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
```

Electronic Dice Game with Pinouts to Convenience PC Board Layout

Logic Diagram PAL16R8



PALASM Outputs

```
ELECTRONIC DICE GAME WITH PINOUTS TO CONVENIENCE PC BOARD LAYOUT

---- ---X ---- --X- ---- ---- ---- ---- /Q1*Q3
---- ---- ---X --X- ---- ---- ---- ---- /Q1*Q4
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---- --X- ---- ---- ---- ---- ---- ---- /Q3
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---X ---- ---- --X- ---- ---- ---- ---- /Q1*Q2
---- ---- ---X --X- ---- ---- ---- ---- /Q1*Q4
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---X ---X ---- --X- ---- ---- ---- ---- /Q1*Q2*Q3
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---- ---- ---- ---X ---- ---X ---- --X- Q1*/Q5*Q7
---- ---- ---- ---X ---- ---- ---X --X- Q1*/Q5*Q8
---- ---- ---- --X- ---X ---- ---- ---- /Q1*Q6
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---- ---- ---- ---X ---- --X- ---- ---- Q1*/Q7
---- ---- ---- --X- ---- ---X ---- ---- /Q1*Q7
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---- ---- ---- ---X ---X ---- ---- --X- Q1*/Q5*Q6
---- ---- ---- ---X ---- ---- ---X --X- Q1*/Q5*Q8
---- ---- ---- --X- ---- ---- ---X ---- /Q1*Q8
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---- ---- ---- ---X ---X ---X ---- --X- Q1*/Q5*Q6*Q7
---- ---- ---- --X- ---- ---- ---- ---X /Q1*Q5
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
```

```
BHLF FORMAT

BHHHHF BHHHHF BHHHHF BLLHHF BHHHHF BHHHHF BHHLHF BLHHLF
BHHHHF BHHHHF BHHHHF BHHHHF BHHHHF BHHHHF BLLHLF BHHHHF
BHHHHF BHHHHF BHHHHF BHHHHF BHHHHF BHHHHF BHHHHF BHHHHF
BHHHHF BHHHHF BHHHHF BHHHHF BHHHHF BHHHHF BHHHHF BHHHHF
BLHLHF BLHLHF BLHLHF BLHLHF BLHLHF BLHLHF BLHLHF BLHLHF
BLHLHF BLHLHF BLHLHF BLLLLF BLHLHF BLHLHF BLLLLF BLHLHF
BLHLHF BLHLHF BLHLHF BLHLHF BLHLHF BLHLHF BLHLHF BLHLHF
BLHLHF BLHLHF BLHLHF BLHLHF BLHLHF BLHLHF BLHLHF BLHLHF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BHHHHF BHHHHF BHHHHF BHHHHF BHHHHF BHHHHF BHHHHF BHHHHF
BHHHHF BHHHHF BHHHHF BHHHHF BHHHHF BHHHHF BHHHHF BLLLLF
BHHHHF BHHHHF BHHHHF BLLHHF BHHHHF BHHHHF BHHLHF BLHHLF
BHHHHF BHHHHF BHHHHF BHHHHF BHHHHF BHHHHF BLLHLF BHHHHF
BHHHHF BHHHHF BHHHHF BHHHHF BHHHHF BHHHHF BHHHHF BHHHHF
BHHHHF BHHHHF BHHHHF BHHHHF BHHHHF BHHHHF BLHLHF BHLHLF
BHHHHF BHHHHF BHHHHF BHHHHF BHHHHF BHHHHF BHHHHF BHHLHF
BHHHHF BHHHHF BHHHHF BHLHLF BHHHHF BHHHHF BHLHLF BLHHHF
BLHLHF BLHLHF BLHLHF BLHLHF BLHLHF BLHLHF BLHLHF BLHLHF
BLHLHF BLHLHF BLHLHF BLHLHF BLHLHF BLHLHF BLLLLF BLHLHF
BLHLHF BLHLHF BLHLHF BLHLLF BLHLHF BLHLHF BLHLHF BLHLHF
BLHLHF BLHLHF BLHLHF BLLLHF BLHLHF BLHLHF BLHLHF BLHLHF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF BLLLLF
```

```
HEX FORMAT

F F F 3 F F D 6 F F F F F F 2 F F F F F F F F F F F F F F F F F .
5 5 5 5 5 5 5 5 5 5 5 0 5 5 0 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 .
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 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 .
F F F F F F F F F F F F F F F 0 F F F 3 F F D 6 F F F F F F 2 F .
F F F F F F F F F F F F F F 5 A F F F F F F F D F F F A F F A 7 .
5 5 5 5 5 5 5 5 5 5 5 5 5 5 0 5 5 5 5 4 5 5 5 5 5 5 5 1 5 5 5 5 .
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 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 .
```

PAL Programming Format

Pal 16R8

Pattern PAT0002

Name ELECTRONIC DICE GAME

For Products 0 thru 31

INPUTS 0 TO 31	WORD 0 TO 255	24	16	8	0		25	17	9	1		26	18	10	2		27	19	11	3		28	20	12	4		29	21	13	5		30	22	14	6		31	23	15	7
0	0	H	H	H	H	32	L	H	L	H	64	L	L	L	L	96	L	L	L	L	128	L	L	L	L	160	L	L	L	L	192	L	L	L	L	224	L	L	L	L
1	1	H	H	H	H	33	L	H	L	H	65	L	L	L	L	97	L	L	L	L	129	L	L	L	L	161	L	L	L	L	193	L	L	L	L	225	L	L	L	L
2	2	H	H	H	H	34	L	H	L	H	66	L	L	L	L	98	L	L	L	L	130	L	L	L	L	162	L	L	L	L	194	L	L	L	L	226	L	L	L	L
3	3	L	L	H	H	35	L	H	L	H	67	L	L	L	L	99	L	L	L	L	131	L	L	L	L	163	L	L	L	L	195	L	L	L	L	227	L	L	L	L
4	4	H	H	H	H	36	L	H	L	H	68	L	L	L	L	100	L	L	L	L	132	L	L	L	L	164	L	L	L	L	196	L	L	L	L	228	L	L	L	L
5	5	H	H	H	H	37	L	H	L	H	69	L	L	L	L	101	L	L	L	L	133	L	L	L	L	165	L	L	L	L	197	L	L	L	L	229	L	L	L	L
6	6	H	H	L	H	38	L	H	L	H	70	L	L	L	L	102	L	L	L	L	134	L	L	L	L	166	L	L	L	L	198	L	L	L	L	230	L	L	L	L
7	7	L	H	H	L	39	L	H	L	H	71	L	L	L	L	103	L	L	L	L	135	L	L	L	L	167	L	L	L	L	199	L	L	L	L	231	L	L	L	L
8	8	H	H	H	H	40	L	H	L	H	72	L	L	L	L	104	L	L	L	L	136	L	L	L	L	168	L	L	L	L	200	L	L	L	L	232	L	L	L	L
9	9	H	H	H	H	41	L	H	L	H	73	L	L	L	L	105	L	L	L	L	137	L	L	L	L	169	L	L	L	L	201	L	L	L	L	233	L	L	L	L
10	10	H	H	H	H	42	L	H	L	H	74	L	L	L	L	106	L	L	L	L	138	L	L	L	L	170	L	L	L	L	202	L	L	L	L	234	L	L	L	L
11	11	H	H	H	H	43	L	L	L	L	75	L	L	L	L	107	L	L	L	L	139	L	L	L	L	171	L	L	L	L	203	L	L	L	L	235	L	L	L	L
12	12	H	H	H	H	44	L	H	L	H	76	L	L	L	L	108	L	L	L	L	140	L	L	L	L	172	L	L	L	L	204	L	L	L	L	236	L	L	L	L
13	13	H	H	H	H	45	L	H	L	H	77	L	L	L	L	109	L	L	L	L	141	L	L	L	L	173	L	L	L	L	205	L	L	L	L	237	L	L	L	L
14	14	L	L	H	L	46	L	L	L	L	78	L	L	L	L	110	L	L	L	L	142	L	L	L	L	174	L	L	L	L	206	L	L	L	L	238	L	L	L	L
15	15	H	H	H	H	47	L	H	L	H	79	L	L	L	L	111	L	L	L	L	143	L	L	L	L	175	L	L	L	L	207	L	L	L	L	239	L	L	L	L
16	16	H	H	H	H	48	L	H	L	H	80	L	L	L	L	112	L	L	L	L	144	L	L	L	L	176	L	L	L	L	208	L	L	L	L	240	L	L	L	L
17	17	H	H	H	H	49	L	H	L	H	81	L	L	L	L	113	L	L	L	L	145	L	L	L	L	177	L	L	L	L	209	L	L	L	L	241	L	L	L	L
18	18	H	H	H	H	50	L	H	L	H	82	L	L	L	L	114	L	L	L	L	146	L	L	L	L	178	L	L	L	L	210	L	L	L	L	242	L	L	L	L
19	19	H	H	H	H	51	L	H	L	H	83	L	L	L	L	115	L	L	L	L	147	L	L	L	L	179	L	L	L	L	211	L	L	L	L	243	L	L	L	L
20	20	H	H	H	H	52	L	H	L	H	84	L	L	L	L	116	L	L	L	L	148	L	L	L	L	180	L	L	L	L	212	L	L	L	L	244	L	L	L	L
21	21	H	H	H	H	53	L	H	L	H	85	L	L	L	L	117	L	L	L	L	149	L	L	L	L	181	L	L	L	L	213	L	L	L	L	245	L	L	L	L
22	22	H	H	H	H	54	L	H	L	H	86	L	L	L	L	118	L	L	L	L	150	L	L	L	L	182	L	L	L	L	214	L	L	L	L	246	L	L	L	L
23	23	H	H	H	H	55	L	H	L	H	87	L	L	L	L	119	L	L	L	L	151	L	L	L	L	183	L	L	L	L	215	L	L	L	L	247	L	L	L	L
24	24	H	H	H	H	56	L	H	L	H	88	L	L	L	L	120	L	L	L	L	152	L	L	L	L	184	L	L	L	L	216	L	L	L	L	248	L	L	L	L
25	25	H	H	H	H	57	L	H	L	H	89	L	L	L	L	121	L	L	L	L	153	L	L	L	L	185	L	L	L	L	217	L	L	L	L	249	L	L	L	L
26	26	H	H	H	H	58	L	H	L	H	90	L	L	L	L	122	L	L	L	L	154	L	L	L	L	186	L	L	L	L	218	L	L	L	L	250	L	L	L	L
27	27	H	H	H	H	59	L	H	L	H	91	L	L	L	L	123	L	L	L	L	155	L	L	L	L	187	L	L	L	L	219	L	L	L	L	251	L	L	L	L
28	28	H	H	H	H	60	L	H	L	H	92	L	L	L	L	124	L	L	L	L	156	L	L	L	L	188	L	L	L	L	220	L	L	L	L	252	L	L	L	L
29	29	H	H	H	H	61	L	H	L	H	93	L	L	L	L	125	L	L	L	L	157	L	L	L	L	189	L	L	L	L	221	L	L	L	L	253	L	L	L	L
30	30	H	H	H	H	62	L	H	L	H	94	L	L	L	L	126	L	L	L	L	158	L	L	L	L	190	L	L	L	L	222	L	L	L	L	254	L	L	L	L
31	31	H	H	H	H	63	L	H	L	H	95	L	L	L	L	127	L	L	L	L	159	L	L	L	L	191	L	L	L	L	223	L	L	L	L	255	L	L	L	L

For Products 32 thru 63

INPUTS 0 TO 31	WORDS 256 TO 511	56	48	40	32		57	49	41	33		58	50	42	34		59	51	43	35		60	52	44	36		61	53	45	37		62	54	46	38		63	55	47	39
0	256	H	H	H	H	288	H	H	H	H	320	L	H	L	H	352	L	L	L	L	384	L	L	L	L	416	L	L	L	L	448	L	L	L	L	480	L	L	L	L
1	257	H	H	H	H	289	H	H	H	H	321	L	H	L	H	353	L	L	L	L	385	L	L	L	L	417	L	L	L	L	449	L	L	L	L	481	L	L	L	L
2	258	H	H	H	H	290	H	H	H	H	322	L	H	L	H	354	L	L	L	L	386	L	L	L	L	418	L	L	L	L	450	L	L	L	L	482	L	L	L	L
3	259	H	H	H	H	291	H	H	H	H	323	L	H	L	H	355	L	L	L	L	387	L	L	L	L	419	L	L	L	L	451	L	L	L	L	483	L	L	L	L
4	260	H	H	H	H	292	H	H	H	H	324	L	H	L	H	356	L	L	L	L	388	L	L	L	L	420	L	L	L	L	452	L	L	L	L	484	L	L	L	L
5	261	H	H	H	H	293	H	H	H	H	325	L	H	L	H	357	L	L	L	L	389	L	L	L	L	421	L	L	L	L	453	L	L	L	L	485	L	L	L	L
6	262	H	H	H	H	294	H	H	H	H	326	L	H	L	H	358	L	L	L	L	390	L	L	L	L	422	L	L	L	L	454	L	L	L	L	486	L	L	L	L
7	263	H	H	H	H	295	H	H	H	H	327	L	H	L	H	359	L	L	L	L	391	L	L	L	L	423	L	L	L	L	455	L	L	L	L	487	L	L	L	L
8	264	H	H	H	H	296	H	H	H	H	328	L	H	L	H	360	L	L	L	L	392	L	L	L	L	424	L	L	L	L	456	L	L	L	L	488	L	L	L	L
9	265	H	H	H	H	297	H	H	H	H	329	L	H	L	H	361	L	L	L	L	393	L	L	L	L	425	L	L	L	L	457	L	L	L	L	489	L	L	L	L
10	266	H	H	H	H	298	H	H	H	H	330	L	H	L	H	362	L	L	L	L	394	L	L	L	L	426	L	L	L	L	458	L	L	L	L	490	L	L	L	L
11	267	H	H	H	H	299	H	H	H	H	331	L	H	L	H	363	L	L	L	L	395	L	L	L	L	427	L	L	L	L	459	L	L	L	L	491	L	L	L	L
12	268	H	H	H	H	300	H	H	H	H	332	L	H	L	H	364	L	L	L	L	396	L	L	L	L	428	L	L	L	L	460	L	L	L	L	492	L	L	L	L
13	269	H	H	H	H	301	H	H	H	H	333	L	H	L	H	365	L	L	L	L	397	L	L	L	L	429	L	L	L	L	461	L	L	L	L	493	L	L	L	L
14	270	H	H	H	H	302	L	H	L	H	334	L	L	L	L	366	L	L	L	L	398	L	L	L	L	430	L	L	L	L	462	L	L	L	L	494	L	L	L	L
15	271	L	L	L	L	303	H	L	H	L	335	L	H	L	H	367	L	L	L	L	399	L	L	L	L	431	L	L	L	L	463	L	L	L	L	495	L	L	L	L
16	272	H	H	H	H	304	H	H	H	H	336	L	H	L	H	368	L	L	L	L	400	L	L	L	L	432	L	L	L	L	464	L	L	L	L	496	L	L	L	L
17	273	H	H	H	H	305	H	H	H	H	337	L	H	L	H	369	L	L	L	L	401	L	L	L	L	433	L	L	L	L	465	L	L	L	L	497	L	L	L	L
18	274	H	H	H	H	306	H	H	H	H	338	L	H	L	H	370	L	L	L	L	402	L	L	L	L	434	L	L	L	L	466	L	L	L	L	498	L	L	L	L
19	275	L	L	H	H	307	H	H	H	H	339	L	H	L	L	371	L	L	L	L	403	L	L	L	L	435	L	L	L	L	467	L	L	L	L	499	L	L	L	L
20	276	H	H	H	H	308	H	H	H	H	340	L	H	L	H	372	L	L	L	L	404	L	L	L	L	436	L	L	L	L	468	L	L	L	L	500	L	L	L	L
21	277	H	H	H	H	309	H	H	H	H	341	L	H	L	H	373	L	L	L	L	405	L	L	L	L	437	L	L	L	L	469	L	L	L	L	501	L	L	L	L
22	278	H	H	L	H	310	H	H	H	H	342	L	H	L	H	374	L	L	L	L	406	L	L	L	L	438	L	L	L	L	470	L	L	L	L	502	L	L	L	L
23	279	L	H	H	L	311	H	H	L	H	343	L	H	L	H	375	L	L	L	L	407	L	L	L	L	439	L	L	L	L	471	L	L	L	L	503	L	L	L	L
24	280	H	H	H	H	312	H	H	H	H	344	L	H	L	H	376	L	L	L	L	408	L	L	L	L	440	L	L	L	L	472	L	L	L	L	504	L	L	L	L
25	281	H	H	H	H	313	H	H	H	H	345	L	H	L	H	377	L	L	L	L	409	L	L	L	L	441	L	L	L	L	473	L	L	L	L	505	L	L	L	L
26	282	H	H	H	H	314	H	H	H	H	346	L	H	L	H	378	L	L	L	L	410	L	L	L	L	442	L	L	L	L	474	L	L	L	L	506	L	L	L	L
27	283	H	H	H	H	315	H	L	H	L	347	L	L	L	H	379	L	L	L	L	411	L	L	L	L	443	L	L	L	L	475	L	L	L	L	507	L	L	L	L
28	284	H	H	H	H	316	H	H	H	H	348	L	H	L	H	380	L	L	L	L	412	L	L	L	L	444	L	L	L	L	476	L	L	L	L	508	L	L	L	L
29	285	H	H	H	H	317	H	H	H	H	349	L	H	L	H	381	L	L	L	L	413	L	L	L	L	445	L	L	L	L	477	L	L	L	L	509	L	L	L	L
30	286	L	L	H	L	318	H	L	H	L	350	L	H	L	H	382	L	L	L	L	414	L	L	L	L	446	L	L	L	L	478	L	L	L	L	510	L	L	L	L
31	287	H	H	H	H	319	L	H	H	H	351	L	H	L	H	383	L	L	L	L	415	L	L	L	L	447	L	L	L	L	479	L	L	L	L	511	L	L	L	L

Applications

and ... HERE'S MORE!!

This section presents more PAL examples used in various systems. Some of the applications present PAL equivalents of standard logic functions. At first glance this may not seem useful. However, consider the case of the standard logic function that is "almost correct". If any special functions are required SSI/MSI logic must be added with an increase in cost. The PAL version of the same function can probably be modified internally to accommodate the change with no additional parts.

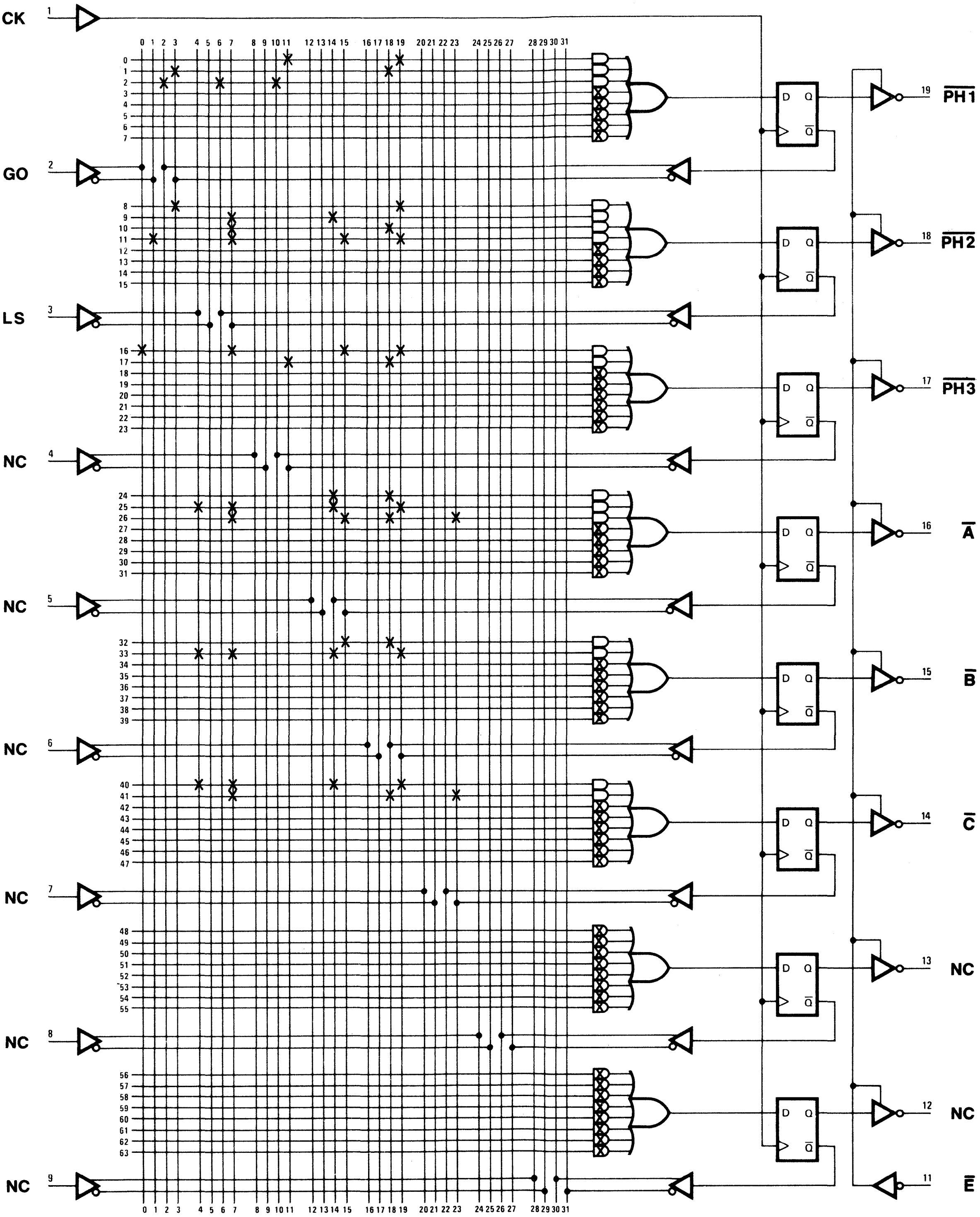
Three Phase Micro Engine

Design Specification PAL16R8

```
PAL16R8                                               PAL DESIGN SPECIFICATION
PAT0031                 MICHAEL D. GARVEY,  FRED W. GROTHMAN     1/6/78
THREE PHASE MICRO ENGINE

CK GO LS NC NC NC NC NC NC GND /E NC NC /C /B /A /PH3 /PH2 /PH1 VCC



  PH1  := PH3◆B + PH1◆/B + /PH1◆/PH2◆/PH3

  PH2  := PH1◆B + PH2◆/A + PH2◆/B + PH2◆A◆B◆/GO

  PH3  := PH2◆GO◆A◆B + PH3◆/B

  A    := /A◆/B + PH2◆/A◆B◆LS + PH2◆A◆/B◆C

  B    := A◆/B + PH2◆/A◆B◆LS

  C    := PH2◆/A◆B◆LS + PH2◆/B◆C



DESCRIPTION:

    THIS IS AN APPLICATION NOTE FOR A THREE PHASE MICRO-ENGINE.
THE DESIGN REQUIREMENTS ARE AS FOLLOWS:

   1.   THE THREE PHASES WILL BE CALLED PH1, PH2, AND PH3.
   2.   PH2 WILL BE OF VARIABLE LENGTH.
   3.   THE MICRO-ENGINE MUST BE ABLE TO STOP AT THE END OF PH2.


        ◆◆◆◆   STATE DIAGRAM OF MICRO-ENGINE   ◆◆◆◆

              (SHORT CYCLE)                          (LONG CYCLE)

    PH1    1 1 1 0 0 0 0 0 0 0              1 1 1 0 0 0 0 0 0 0 0 0
    PH2    0 0 0 1 1 1 1 0 0 0              0 0 0 1 1 1 1 1 1 0 0 0
    PH3    0 0 0 0 0 0 0 1 1 1              0 0 0 0 0 0 0 0 0 1 1 1
    A      0 1 0 0 1 0 1 0 1 0              0 1 0 0 1 0 1 1 1 0 1 0
    B      0 0 1 0 0 1 1 0 0 1              0 0 1 0 0 1 0 0 1 0 0 1
    C      0 0 0 0 0 0 0 0 0 0              0 0 0 0 0 0 1 1 1 0 0 0
    GO     X X X X X X ◆ X X X              X X X X X X X X ◆ X X X
    LS     X X X X X 0 X X X X              X X X X X 1 X X X X X X


    NOTES:

     1.   ◆ GO = 0 WILL STOP,   GO = 1 WILL CONTINUE
     2.   X IS "DON'T CARE"
     3.   LS = 0   SHORT CYCLE,   LS = 1   LONG CYCLE
     4.   A, B, AND C ARE STATE COUNTERS
     5.   GO AND LS ARE INPUTS
```

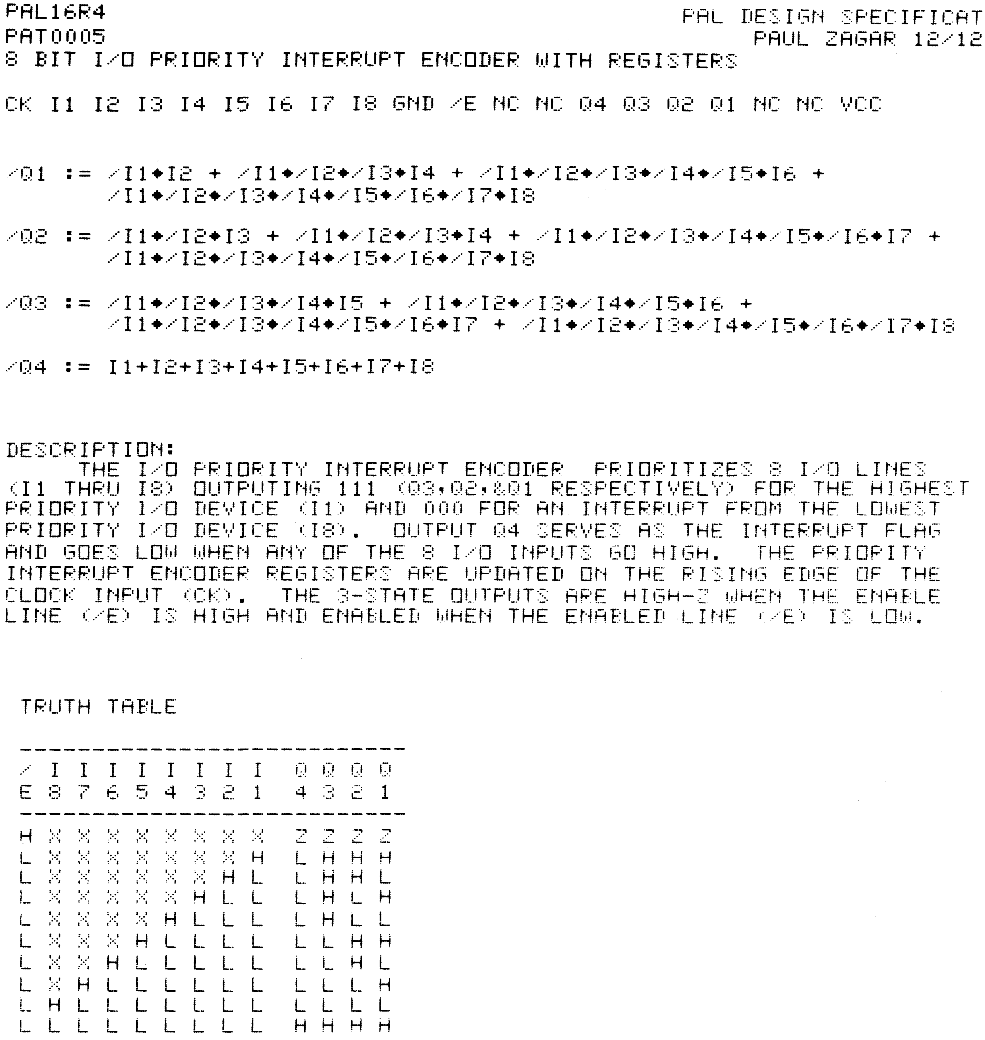
Three Phase Micro Engine

Logic Diagram PAL16R8



5

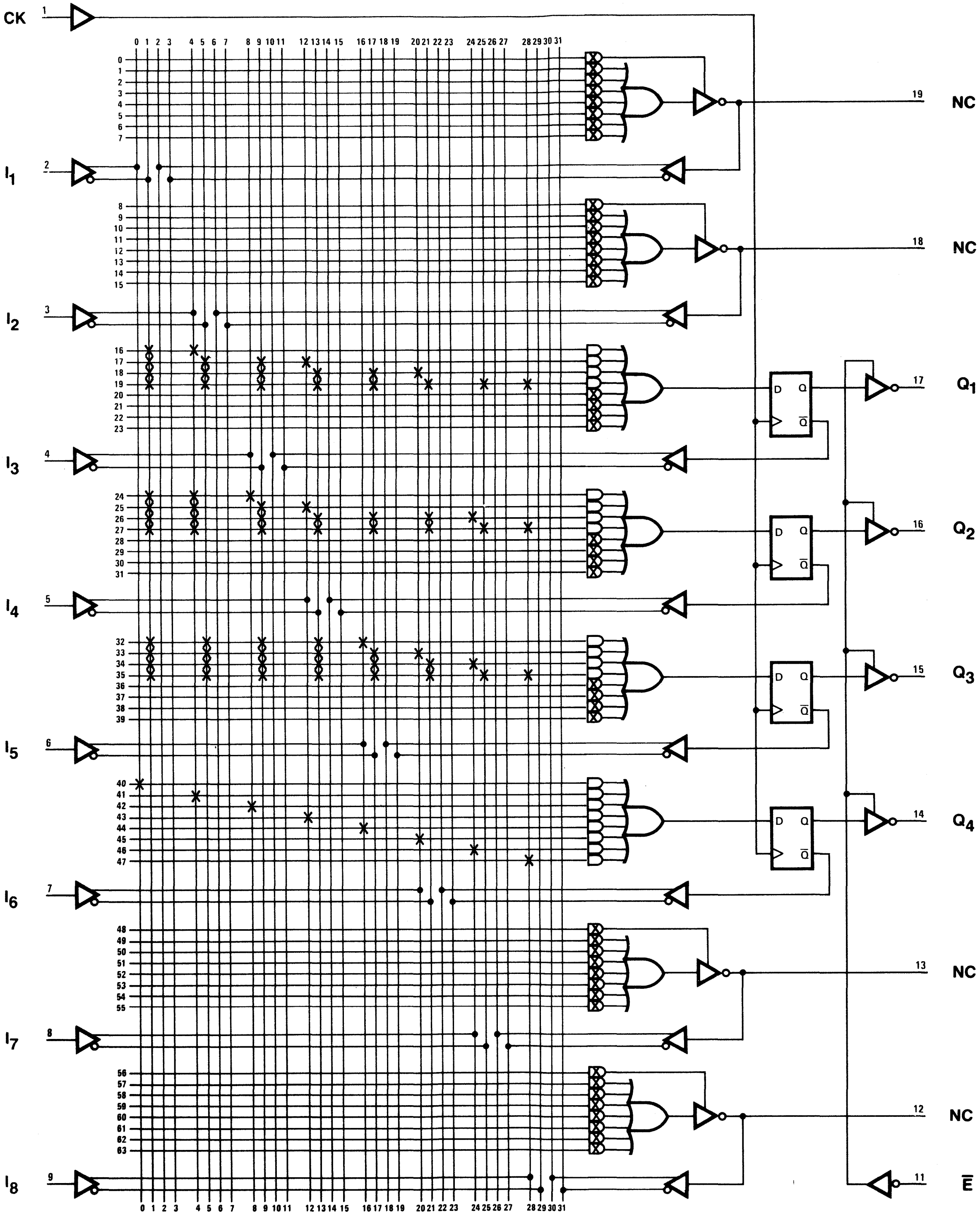
8 Bit I/O Priority Interrupt Encoder with Registers — Design Specification PAL16R4



```
PAL16R4                                          PAL DESIGN SPECIFICATION
PAT0005                                              PAUL ZAGAR 12/12/77
8 BIT I/O PRIORITY INTERRUPT ENCODER WITH REGISTERS

CK I1 I2 I3 I4 I5 I6 I7 I8 GND /E NC NC Q4 Q3 Q2 Q1 NC NC VCC


/Q1 := /I1*I2 + /I1*/I2*/I3*I4 + /I1*/I2*/I3*/I4*/I5*I6 +
       /I1*/I2*/I3*/I4*/I5*/I6*/I7*I8

/Q2 := /I1*/I2*I3 + /I1*/I2*/I3*I4 + /I1*/I2*/I3*/I4*/I5*/I6*I7 +
       /I1*/I2*/I3*/I4*/I5*/I6*/I7*I8

/Q3 := /I1*/I2*/I3*/I4*I5 + /I1*/I2*/I3*/I4*/I5*I6 +
       /I1*/I2*/I3*/I4*/I5*/I6*I7 + /I1*/I2*/I3*/I4*/I5*/I6*/I7*I8

/Q4 := I1+I2+I3+I4+I5+I6+I7+I8


DESCRIPTION:
     THE I/O PRIORITY INTERRUPT ENCODER  PRIORITIZES 8 I/O LINES
(I1 THRU I8) OUTPUTING 111 (Q3,Q2,&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
CLOCK INPUT (CK).  THE 3-STATE OUTPUTS ARE HIGH-Z WHEN THE ENABLE
LINE (/E) IS HIGH AND ENABLED WHEN THE ENABLED LINE (/E) IS LOW.


 TRUTH TABLE

 ---------------------------
 / I I I I I I I I  Q Q Q Q
 E 8 7 6 5 4 3 2 1  4 3 2 1
 ---------------------------
 H X X X X X X X X  Z Z Z Z
 L X X X X X X X H  L H H H
 L X X X X X X H L  L H H L
 L X X X X X H L L  L H L H
 L X X X X H L L L  L H L L
 L X X X H L L L L  L L H H
 L X X H L L L L L  L L H L
 L X H L L L L L L  L L L H
 L H L L L L L L L  L L L L
 L L L L L L L L L  H H H H
```

8 Bit I/O Priority Interrupt Encoder with Registers

Logic Diagram PAL16R4

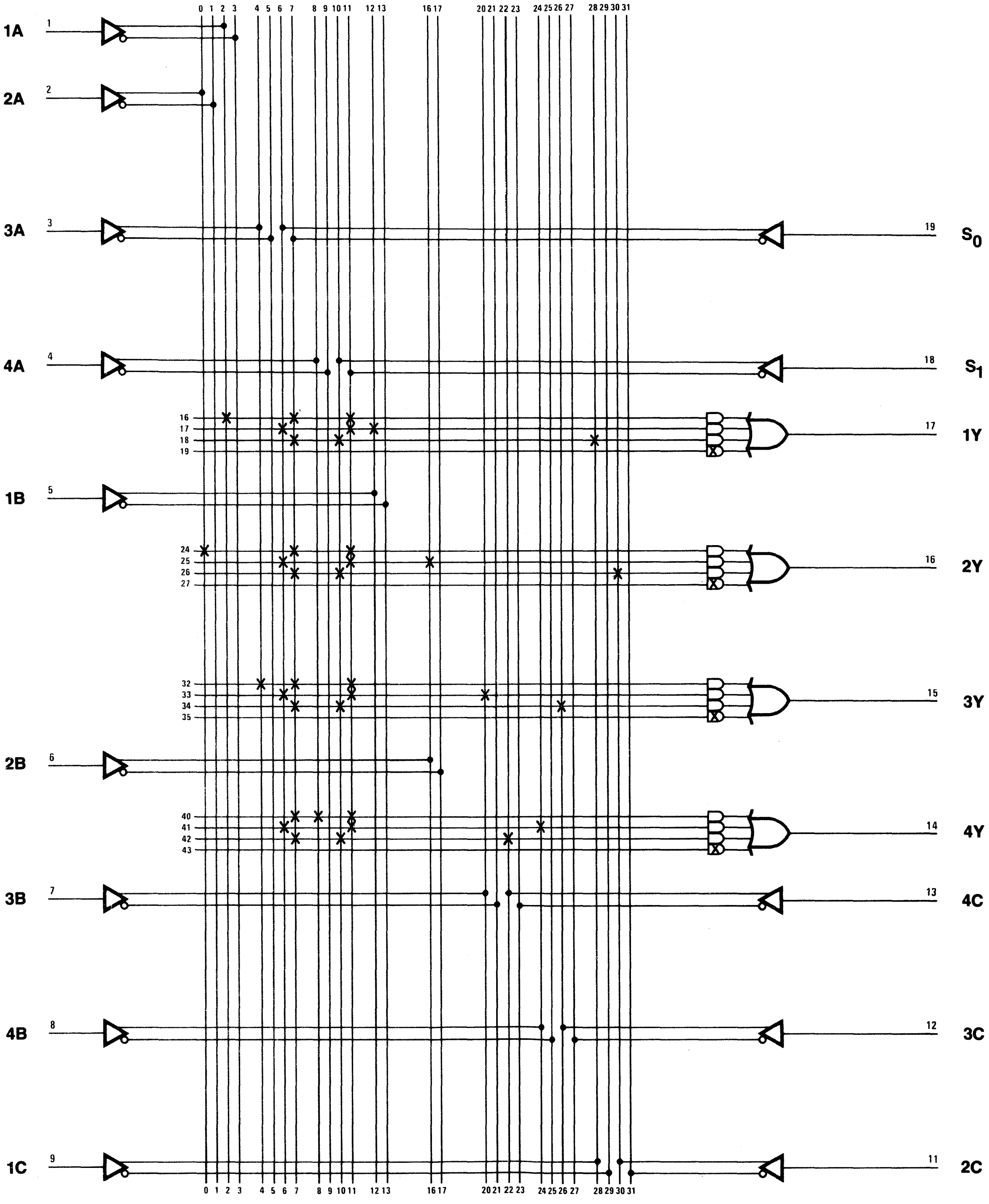


Quadruple 3-Line-to-1-Line Data Selector Multiplexer — Design Specification PAL14H4

```
PAL14H4                                                 PAL DESIGN SPECIFICATION
PAT0016                                                     VIC NEWTON  12/21/77
QUADRUPLE 3-LINE-TO-1-LINE DATA SELECTOR MULTIPLEXER

1A 2A 3A 4A 1B 2B 3B 4B 1C GND 2C 3C 4C 4Y 3Y 2Y 1Y S1 S0 VCC



 1Y = 1A*/S0*/S1 + 1B*S0*/S1 + 1C*/S0*S1

 2Y = 2A*/S0*/S1 + 2B*S0*/S1 + 2C*/S0*S1

 3Y = 3A*/S0*/S1 + 3B*S0*/S1 + 3C*/S0*S1

 4Y = 4A*/S0*/S1 + 4B*S0*/S1 + 4C*/S0*S1



DESCRIPTION:

    A 4-BIT WORD IS SELECTED FROM ONE OF THREE SOURCES AND IS ROUTED
TO THE FOUR OUTPUTS.  TRUE DATA IS PRESENTED AT THE OUTPUTS.  IF
INVERTED DATA IS DESIRED, USE THE SAME EQUATIONS WITH THE PAL14L4.

FUNCTION TABLE:

  -------------------                            ------
  !S0 !S1 ! OUTPUTS !            1A   1 +      + 20 VCC
  -------------------                   !      !
  ! L ! L ! A DATA  !            2A   2 +      + 19 S0
  ! H ! L ! B DATA  !                   !      !
  ! L ! H ! C DATA  !            3A   3 +      + 18 S1
  ! H ! H !  LOW    !                   !      !
  -------------------            4A   4 +      + 17 1Y
                                        !      !
                                 1B   5 +      + 16 2Y
                                        !      !
                                 2B   6 +      + 15 3Y
                                        !      !
                                 3B   7 +      + 14 4Y
                                        !      !
                                 4B   8 +      + 13 4C
                                        !      !
                                 1C   9 +      + 12 3C
                                        !      !
                                GND  10 +      + 11 2C
                                         ------

                                          PINOUT
```

Quadruple 3-Line-to-1-Line Data Selector Multiplexer

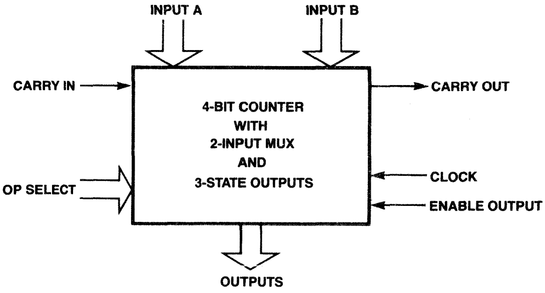
Logic Diagram PAL14H4



4-Bit Counter with 2 Input Mux

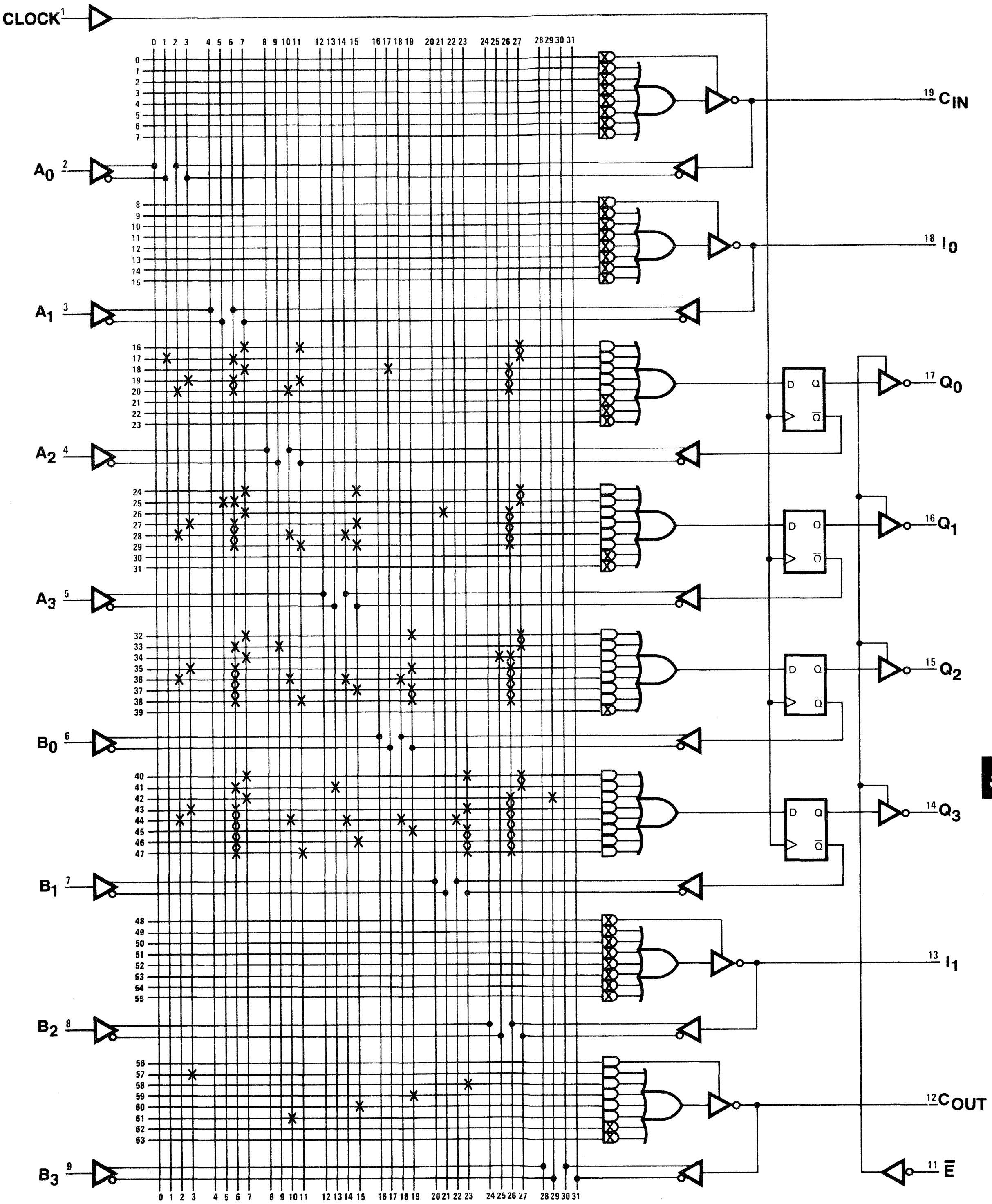
Design Specification PAL16R4

```
PAL16R4                                            PAL DESIGN SPECIFICATION
PAT0034                                               JOHN BIRKNER 12/23/77
4-BIT COUNTER WITH 2 INPUT MUX

CLOCK A0 A1 A2 A3 B0 B1 B2 B3 GND /E COUT I1 Q3 Q2 Q1 Q0 I0 CIN VCC


/Q0 := /I1*/I0*/Q0 + /I1*I0*/A0 + I1*/I0*/B0 +  I1*I0*/CIN*/Q0 +
       I1*I0*CIN*Q0


/Q1 := /I1*/I0*/Q1 + /I1*I0*/A1 + I1*/I0*/B1 +  I1*I0*/CIN*/Q1 +
       I1*I0*CIN*Q1*Q0 + I1*I0*/Q1*/Q0


/Q2 := /I1*/I0*/Q2 + /I1*I0*/A2 + I1*/I0*/B2 +  I1*I0*/CIN*/Q2 +
       I1*I0*CIN*Q2*Q1*Q0 + I1*I0*/Q2*/Q1 +I1*I0*/Q2*/Q0


/Q3 := /I1*/I0*/Q3 + /I1*I0*/A3 + I1*/I0*/B3 +  I1*I0*/CIN*/Q3 +
       I1*I0*CIN*Q3*Q2*Q1*Q0 + I1*I0*/Q3*/Q2 + I1*I0*/Q3*/Q1 + I1*I0*/Q3*/Q0


IF(VCC) /COUT = /CIN + /Q3 + /Q2 + /Q1 + /Q0


DESCRIPTION:

THE 4-BIT COUNTER LOADS A OR B FROM THE MUX, OR COUNTS UP.
THE THREE STATE OUTPUTS ARE ACTIVE WHEN /E IS LOW.


FUNCTION TABLE:

-----------------------------------------------------------------
! /E ! I1 I0 ! CIN ! CLOCK ! OUTPUT Q ! OPERATION !
-----------------------------------------------------------------
!  L !  L  L !  X  !  L-H  !    Q     ! NOP       !
!  L !  L  H !  X  !  L-H  !    A     ! LOAD A    !
!  L !  H  L !  X  !  L-H  !    B     ! LOAD B    !
!  L !  H  H !  L  !  L-H  !    Q     ! NOP       !
!  L !  H  H !  H  !  L-H  ! Q PLUS 1 ! INCREMENT !
-----------------------------------------------------------------
```



4-Bit Counter with 2 Input Mux

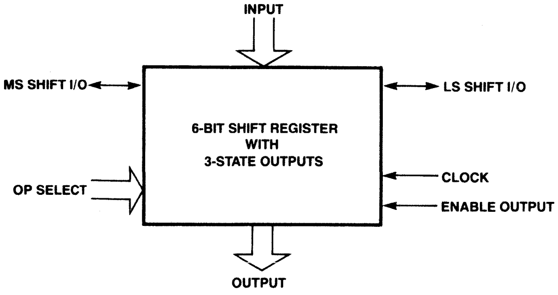
Logic Diagram PAL16R4



6-Bit Shift Register with Three-State Outputs

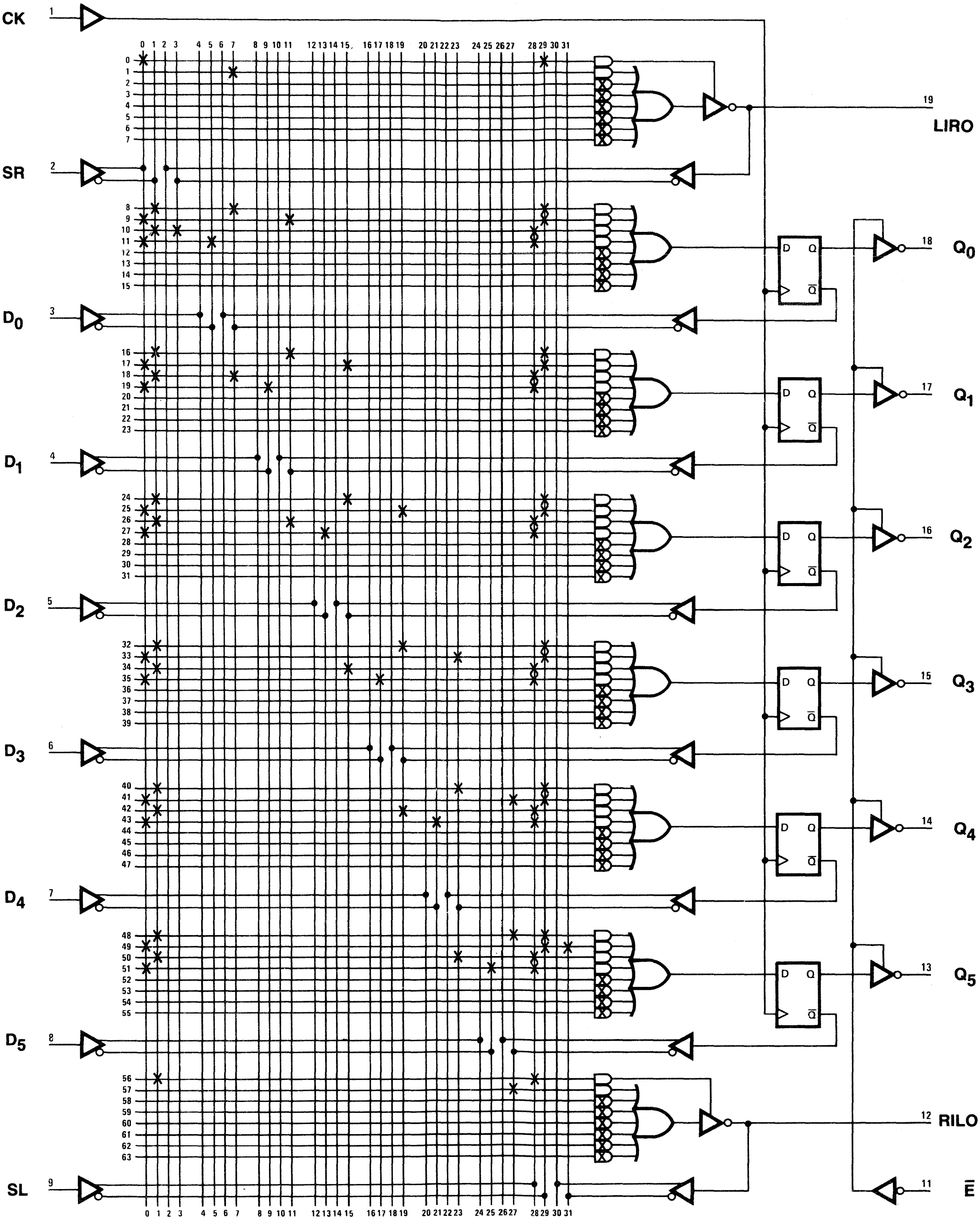
Design Specification PAL16R6

```
PAL16R6
                                                            PAL DESIGN SPECIFICATION
PAT0005                                                        JOHN BIRKNER 11/20/77
6-BIT SHIFT REGISTER WITH THREE-STATE OUTPUTS

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


DESCRIPTION:

THE 6-BIT SHIFT REGISTER WILL HOLD, SHIFT RIGHT, SHIFT LEFT, OR LOAD ON THE
RISING EDGE OF THE CLOCK (CK).
THE THREE-STATE OUTPUTS ARE HIGH-Z WHEN THE ENABLE LINE (/E) IS HIGH
AND ENABLED WHEN ENABLE LINE (/E) IS LOW.

FUNCTION TABLE

---------------------------------------------------------------------------
!               INPUTS                 !               OUTPUTS             !
---------------------------------------------------------------------------
! SL SR ! RILO ! LIRO ! CLOCK ! RILO ! Q5 Q4 Q3 Q2 Q1 Q0 ! LIRO ! OPERATION !
---------------------------------------------------------------------------
!  L  L !  X   !  X   !   L   !  Z   ! Q5 Q4 Q3 Q2 Q1 Q0 !  Z   ! NOP       !
!  H  H !  X   !  X   !   L   !  Z   ! Q5 Q4 Q3 Q2 Q1 Q0 !  Z   ! NOP       !
!  L  L !  X   !  X   !   H   !  Z   ! Q5 Q4 Q3 Q2 Q1 Q0 !  Z   ! NOP       !
!  H  H !  X   !  X   !   H   !  Z   ! Q5 Q4 Q3 Q2 Q1 Q0 !  Z   ! NOP       !
!  L  H !  X   !  X   !   L   !  Z   ! Q5 Q4 Q3 Q2 Q1 Q0 !  Q0  ! NOP       !
!  L  H !  X   !  X   !   H   !  Z   ! Q5 Q4 Q3 Q2 Q1 Q0 !  Q0  ! NOP       !
!  H  L !  X   !  X   !   L   !  Q5  ! Q5 Q4 Q3 Q2 Q1 Q0 !  Z   ! NOP       !
!  H  L !  X   !  X   !   H   !  Q5  ! Q5 Q4 Q3 Q2 Q1 Q0 !  Z   ! NOP       !
!  L  L !  X   !  X   !  L-H  !  Z   ! Q5 Q4 Q3 Q2 Q1 Q0 !  Z   ! HOLD      !
!  L  H !  RI  !  X   !  L-H  !  Z   ! RI Q5 Q4 Q3 Q2 Q1 !  Q1  ! RIGHT SHFT!
!  H  L !  X   !  LI  !  L-H  !  Q4  ! Q4 Q3 Q2 Q1 Q0 LI !  Z   ! LEFT  SHFT!
!  H  H !  X   !  X   !  L-H  !  Z   ! D5 D4 D3 D2 D1 D0 !  Z   ! LOAD D    !
---------------------------------------------------------------------------
```



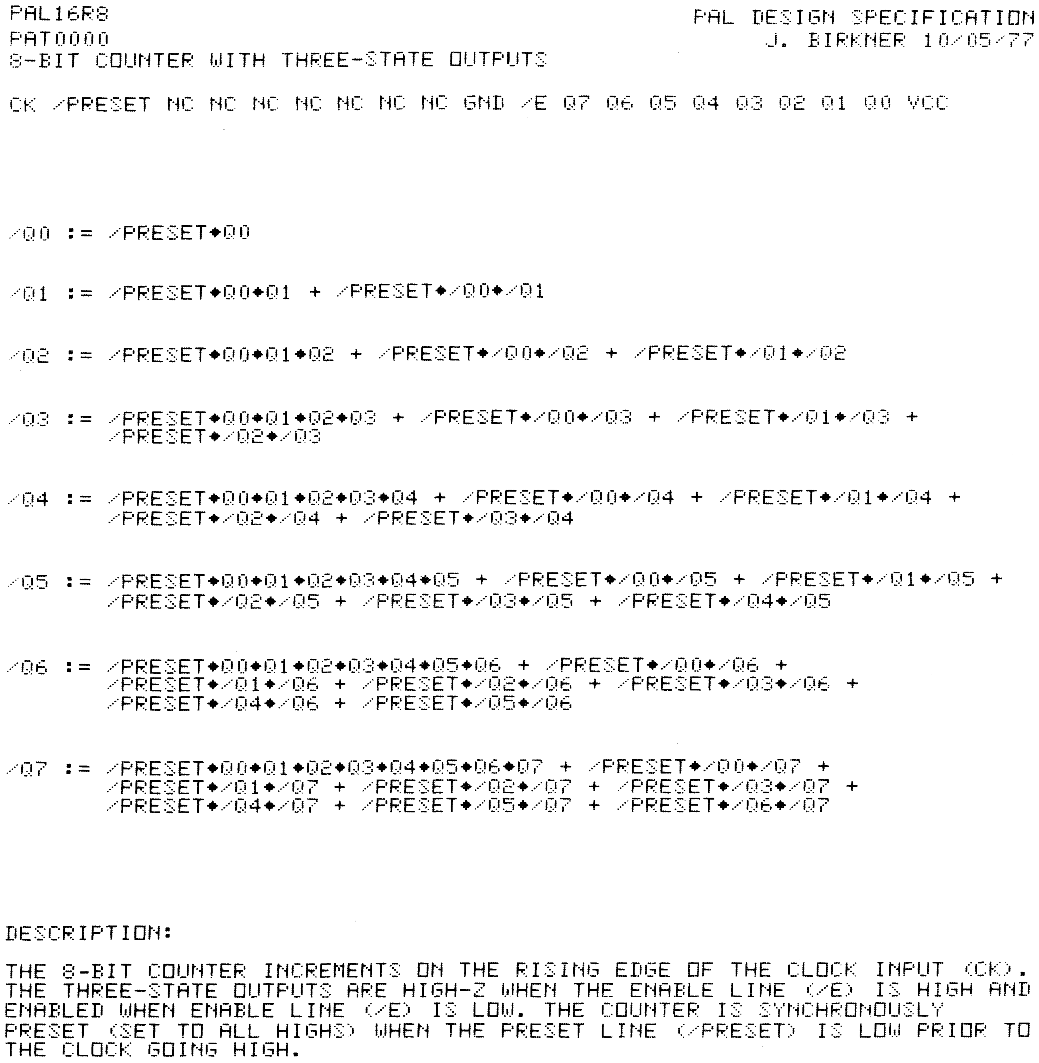
6-Bit Shift Register with Three-State Outputs

Logic Diagram PAL16R6



8-Bit Counter with Three-State Outputs

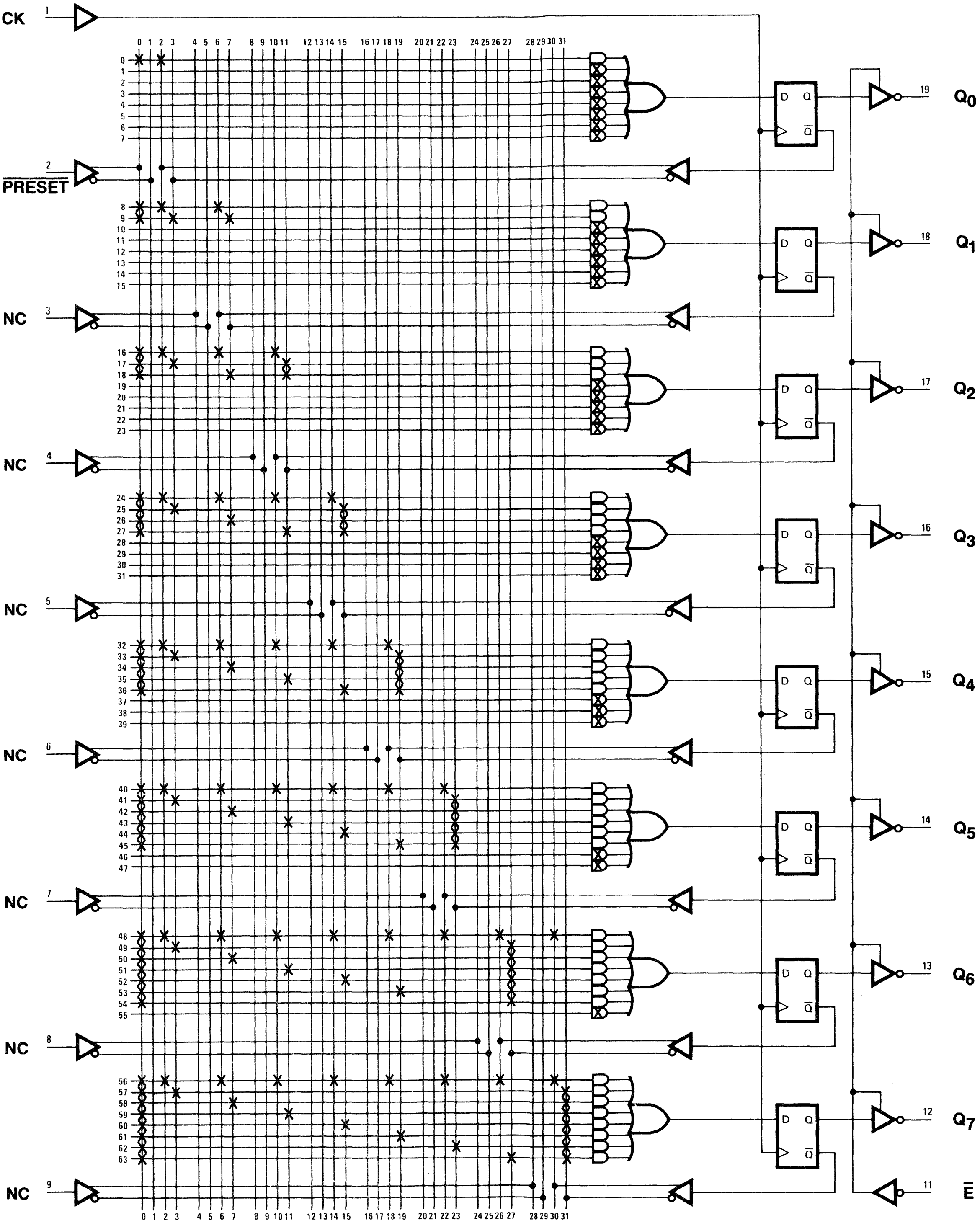
Design Specification PAL16R8



```
PAL16R8                                               PAL DESIGN SPECIFICATION
PAT0000                                                   J. BIRKNER 10/05/77
8-BIT COUNTER WITH THREE-STATE OUTPUTS

CK /PRESET NC NC NC NC NC NC NC GND /E Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0 VCC




/Q0 := /PRESET*Q0


/Q1 := /PRESET*Q0*Q1 + /PRESET*/Q0*/Q1


/Q2 := /PRESET*Q0*Q1*Q2 + /PRESET*/Q0*/Q2 + /PRESET*/Q1*/Q2


/Q3 := /PRESET*Q0*Q1*Q2*Q3 + /PRESET*/Q0*/Q3 + /PRESET*/Q1*/Q3 +
       /PRESET*/Q2*/Q3


/Q4 := /PRESET*Q0*Q1*Q2*Q3*Q4 + /PRESET*/Q0*/Q4 + /PRESET*/Q1*/Q4 +
       /PRESET*/Q2*/Q4 + /PRESET*/Q3*/Q4


/Q5 := /PRESET*Q0*Q1*Q2*Q3*Q4*Q5 + /PRESET*/Q0*/Q5 + /PRESET*/Q1*/Q5 +
       /PRESET*/Q2*/Q5 + /PRESET*/Q3*/Q5 + /PRESET*/Q4*/Q5


/Q6 := /PRESET*Q0*Q1*Q2*Q3*Q4*Q5*Q6 + /PRESET*/Q0*/Q6 +
       /PRESET*/Q1*/Q6 + /PRESET*/Q2*/Q6 + /PRESET*/Q3*/Q6 +
       /PRESET*/Q4*/Q6 + /PRESET*/Q5*/Q6


/Q7 := /PRESET*Q0*Q1*Q2*Q3*Q4*Q5*Q6*Q7 + /PRESET*/Q0*/Q7 +
       /PRESET*/Q1*/Q7 + /PRESET*/Q2*/Q7 + /PRESET*/Q3*/Q7 +
       /PRESET*/Q4*/Q7 + /PRESET*/Q5*/Q7 + /PRESET*/Q6*/Q7



DESCRIPTION:

THE 8-BIT COUNTER INCREMENTS ON THE RISING EDGE OF THE CLOCK INPUT (CK).
THE THREE-STATE OUTPUTS ARE HIGH-Z WHEN THE ENABLE LINE (/E) IS HIGH AND
ENABLED WHEN ENABLE LINE (/E) IS LOW. THE COUNTER IS SYNCHRONOUSLY
PRESET (SET TO ALL HIGHS) WHEN THE PRESET LINE (/PRESET) IS LOW PRIOR TO
THE CLOCK GOING HIGH.
```

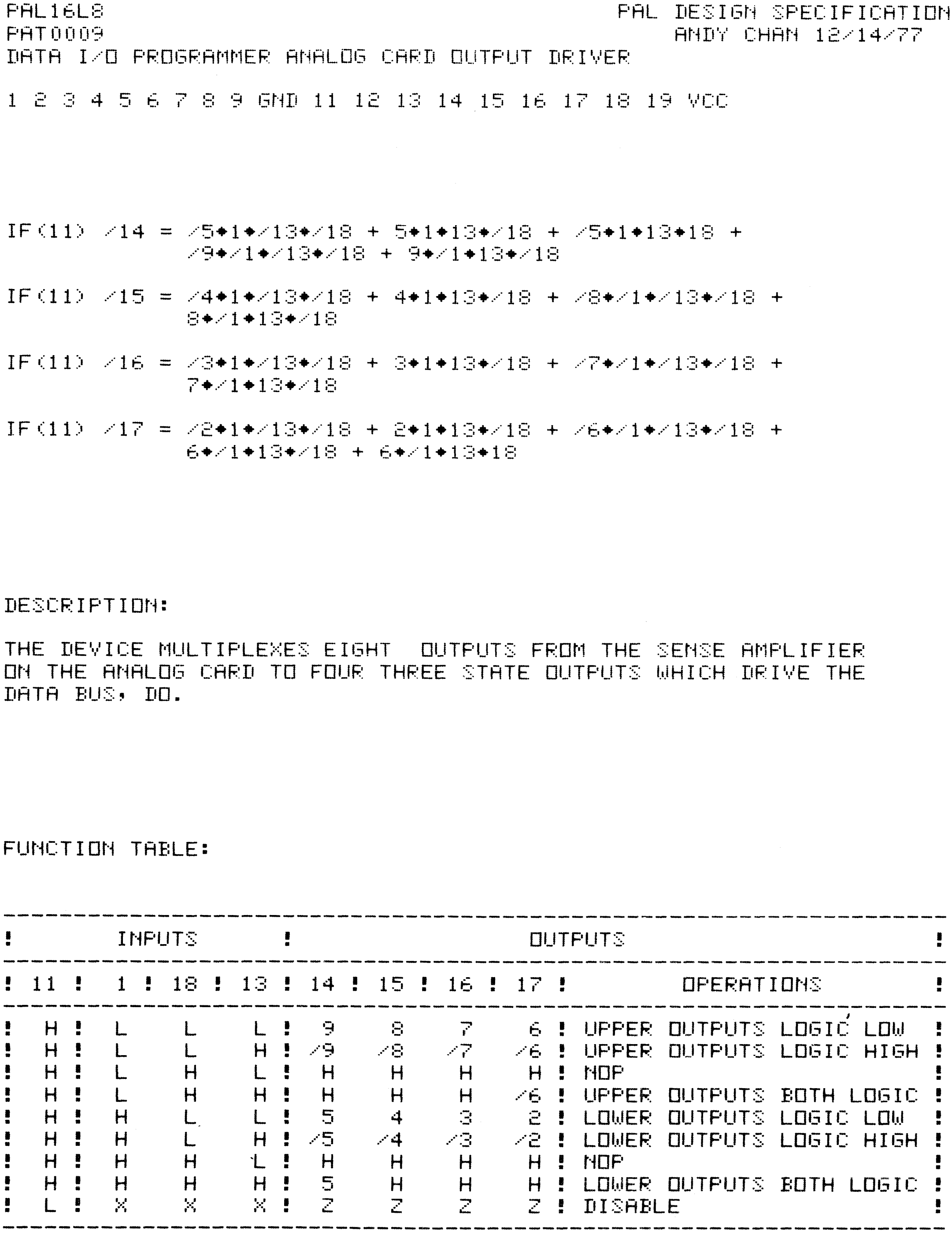
8-Bit Counter with Three-State Outputs

Logic Diagram PAL16R8



Data I/O Programmer Analog Card Output Driver

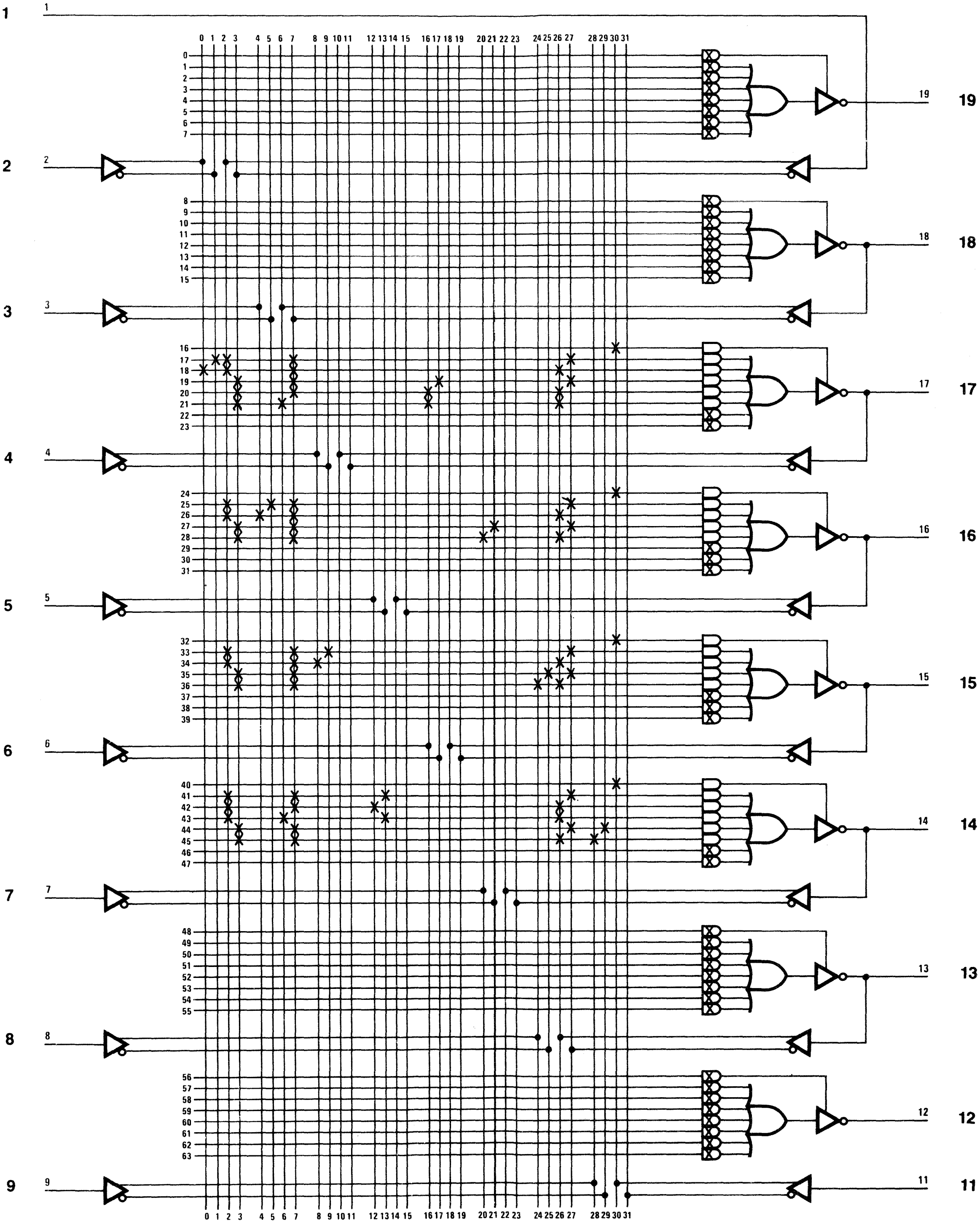
Design Specification PAL16L8



```
PAL16L8                                                 PAL DESIGN SPECIFICATION
PAT0009                                                     ANDY CHAN 12/14/77
DATA I/O PROGRAMMER ANALOG CARD OUTPUT DRIVER

1 2 3 4 5 6 7 8 9 GND 11 12 13 14 15 16 17 18 19 VCC



IF(11) /14 = /5*1*/13*/18 + 5*1*13*/18 + /5*1*13*18 +
             /9*/1*/13*/18 + 9*/1*13*/18

IF(11) /15 = /4*1*/13*/18 + 4*1*13*/18 + /8*/1*/13*/18 +
             8*/1*13*/18

IF(11) /16 = /3*1*/13*/18 + 3*1*13*/18 + /7*/1*/13*/18 +
             7*/1*13*/18

IF(11) /17 = /2*1*/13*/18 + 2*1*13*/18 + /6*/1*/13*/18 +
             6*/1*13*/18 + 6*/1*13*18



DESCRIPTION:

THE DEVICE MULTIPLEXES EIGHT  OUTPUTS FROM THE SENSE AMPLIFIER
ON THE ANALOG CARD TO FOUR THREE STATE OUTPUTS WHICH DRIVE THE
DATA BUS, DO.



FUNCTION TABLE:


-------------------------------------------------------------------
!        INPUTS        !                  OUTPUTS                 !
-------------------------------------------------------------------
! 11 !  1 ! 18 ! 13 ! 14 ! 15 ! 16 ! 17 !       OPERATIONS        !
-------------------------------------------------------------------
!  H !  L    L    L !  9    8    7    6 ! UPPER OUTPUTS LOGIC LOW !
!  H !  L    L    H ! /9   /8   /7   /6 ! UPPER OUTPUTS LOGIC HIGH !
!  H !  L    H    L !  H    H    H    H ! NOP                     !
!  H !  L    H    H !  H    H    H   /6 ! UPPER OUTPUTS BOTH LOGIC !
!  H !  H    L    L !  5    4    3    2 ! LOWER OUTPUTS LOGIC LOW !
!  H !  H    L    H ! /5   /4   /3   /2 ! LOWER OUTPUTS LOGIC HIGH !
!  H !  H    H    L !  H    H    H    H ! NOP                     !
!  H !  H    H    H !  5    H    H    H ! LOWER OUTPUTS BOTH LOGIC !
!  L !  X    X    X !  Z    Z    Z    Z ! DISABLE                 !
-------------------------------------------------------------------
```

Data I/O Programmer Analog Card Output Driver

Logic Diagram PAL16L8



24-Bit Fast Look-Ahead Carry Generator

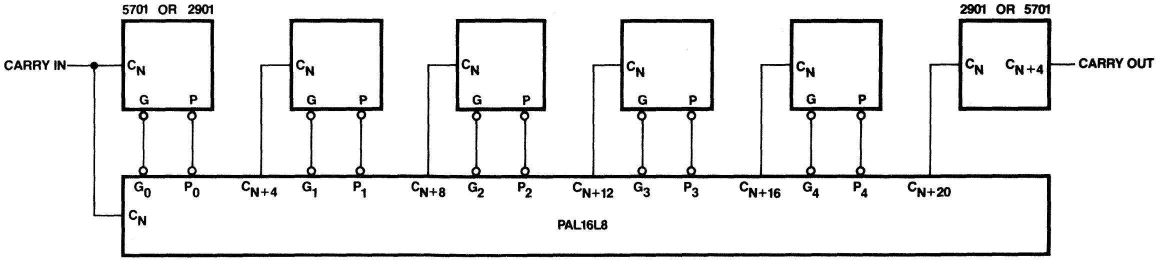
Design Specification PAL16L8

```
PAL16L8                                               PAL DESIGN SPECIFICATION
PAT0018                                                  VIC NEWTON  12/19/77
24 BIT FAST LOOK-AHEAD CARRY GENERATOR

CN /G0 /P0 /G1 /P1 /G2 /P2 /G3 /P3 GND /G4 NC1 /P4 NC2
CN20 CN16 CN12 CN8 CN4 VCC




IF (VCC) /CN4  = /G0*/P0 + /G0*/CN

IF (VCC) /CN8  = /G1*/P1 + /G1*/G0*/P0 + /G1*/G0*/CN

IF (VCC) /CN12 = /G2*/P2 + /G2*/G1*/P1 + /G2*/G1*/G0*/P0 +
                 /G2*/G1*/G0*/CN

IF (VCC) /CN16 = /G3*/P3 + /G3*/G2*/P2 + /G3*/G2*/G1*/P1 +
                 /G3*/G2*/G1*/G0*/P0 + /G3*/G2*/G1*/G0*/CN

IF (VCC) /CN20 = /G4*/P4 + /G4*/G3*/P3 + /G4*/G3*/G2*/P2 +
                 /G4*/G3*/G2*/G1*/P1 + /G4*/G3*/G2*/G1*/G0*/P0 +
                 /G4*/G3*/G2*/G1*/G0*/CN
```

DESCRIPTION:

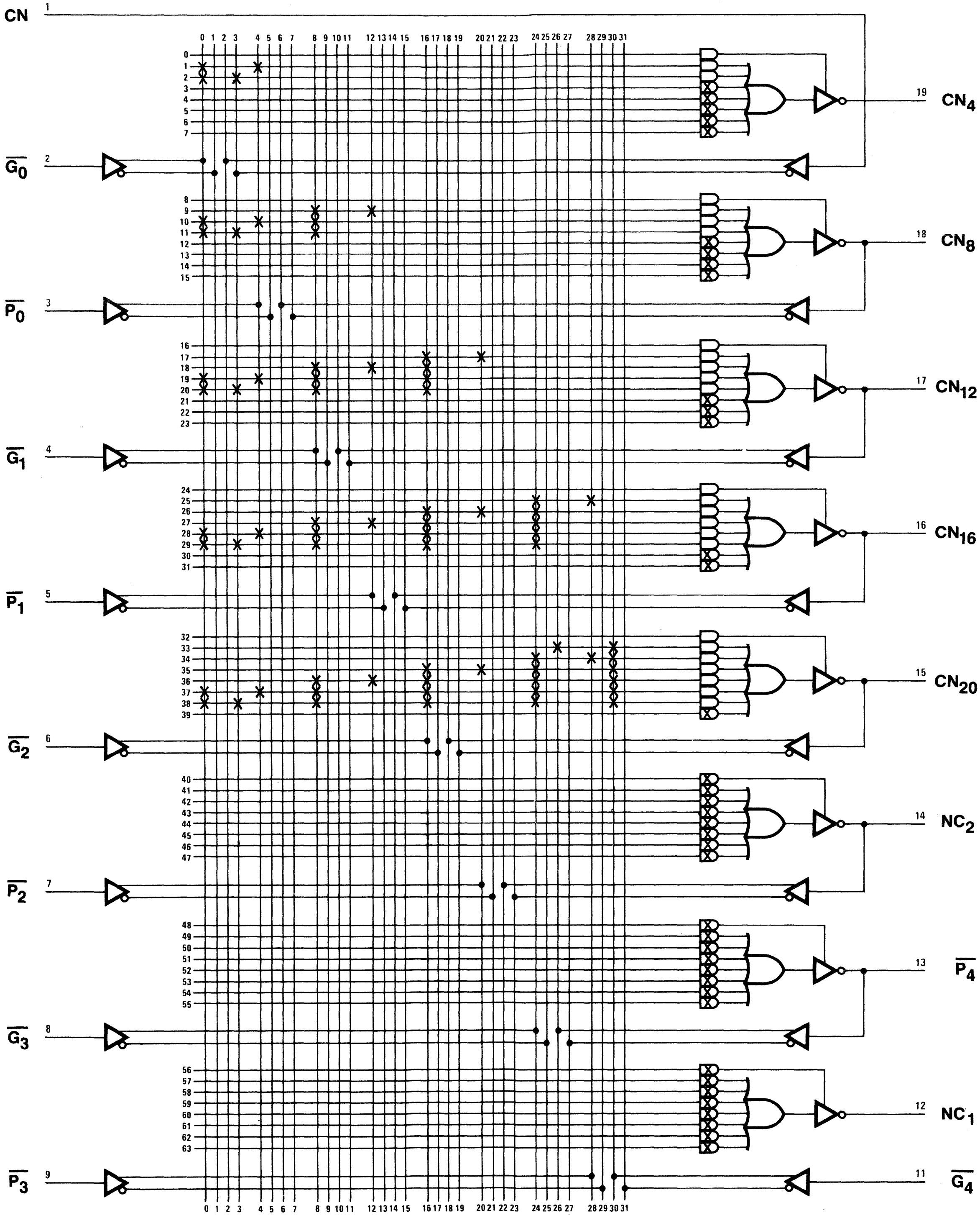
THE LOOK-AHEAD CARRY GENERATOR ACCEPTS A CARRY INPUT AND UP TO FIVE PAIRS OF CARRY PROPAGATE AND CARRY GENERATE SIGNALS. IT PROVIDES ANTICIPATED CARRIES ACROSS SIX GROUPS OF BINARY ALU'S. THE LOOK-AHEAD CARRY GENERATOR CAN BE USED WITH BINARY ALU'S IN AN ACTIVE LOW OR ACTIVE HIGH INPUT OPERAND MODE BY REINTERPERTING THE CARRY FUNCTIONS. THE CONNECTIONS ARE IDENTICAL IN BOTH CASES.

24 BIT ALU'S CAN BE OBTAINED BY USING 4 BIT SLICES SUCH AS THE 5701 OR 2901.



24-Bit Fast Look-Ahead Carry Generator

Logic Diagram PAL16L8

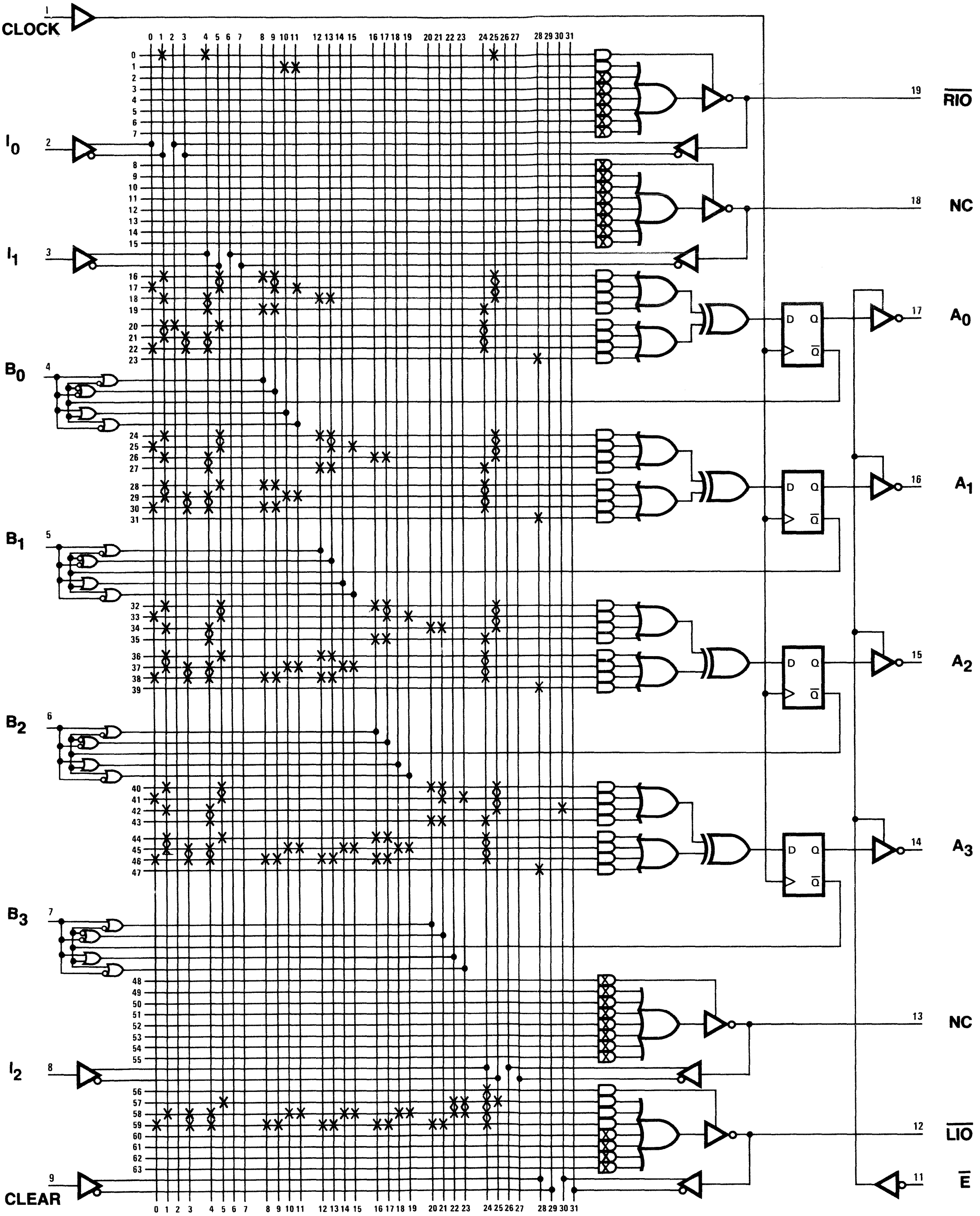


4-Bit Up/Down Counter with Shift and Three-State Outputs

Design Specification PAL16X4

```
PAL16X4                                                  PAL DESIGN SPECIFICATION
PAT0026                                                     JOHN BIRKNER 12/10/77
4 BIT UP/DOWN COUNTER WITH SHIFT AND THREE-STATE OUTPUTS

CLOCK I0 I1 B0 B1 B2 B3 I2 CLEAR GND /E /LIO NC A3 A2 A1 A0 NC /RIO VCC



IF( /I2*I1*/I0 ) RIO = (A0)


/A0 := (/A0)*/I2*/I1*/I0 + (/B0)*/I2*/I1*I0 + (/A1)*/I2*I1*/I0 + (/A0)*I2*I1
       :+: /RIO*I2*/I1*/I0 +
           RIO*I2*I1*/I0 +
           RIO*I2*I1* I0 + CLEAR


/A1 := (/A1)*/I2*/I1*/I0 + (/B1)*/I2*/I1*I0 + (/A2)*/I2*I1*/I0 + (/A1)*I2*I1
       :+: (/A0)*I2*/I1*/I0 +
           ( A0)*RIO*I2*I1*/I0 +
           (/A0)*RIO*I2*I1* I0 + CLEAR


/A2 := (/A2)*/I2*/I1*/I0 + (/B2)*/I2*/I1*I0 + (/A3)*/I2*I1*/I0 + (/A2)*I2*I1
       :+: (/A1)*I2*/I1*/I0 +
           ( A1)*( A0)*RIO*I2*I1*/I0 +
           (/A1)*(/A0)*RIO*I2*I1* I0 + CLEAR


/A3 := (/A3)*/I2*/I1*/I0 + (/B3)*/I2*/I1*I0 + /LIO*/I2*I1*/I0 + (/A3)*I2*I1
       :+: (/A2)*I2*/I1*/I0 +
           ( A2)*( A1)*( A0)*RIO*I2*I1*/I0 +
           (/A2)*(/A1)*(/A0)*RIO*I2*I1* I0 + CLEAR


IF( I2 )  LIO = (A3)*I2*/I1*/I2 + ( A3)*( A2)*( A1)*( A0)*RIO*I2*I1*/I0 +
                                  (/A3)*(/A2)*(/A1)*(/A0)*RIO*I2*I1* I0


DESCRIPTION:
THE UP/DOWN COUNTER WITH SHIFT WILL LOAD, SHIFT, COUNT UP, COUNT DOWN,
CLEAR OR NOP ON THE RISING EDGE OF THE CLOCK AS SPECIFIED BY THE
INSTRUCTION, I.  SHIFT I/O, CARRY AND BORROW SHARE THE SAME I/O LINES
(LIO AND RIO).  ACTIVE HIGH OUTPUTS, A, ARE ENABLED WHEN /E IS
LOW.  NOTE:  THE IMPLIED EXCLUSIVE OR, :+: , MUST BE PLACED BETWEEN
THE FOURTH AND FIFTH PRODUCT TERMS.


FUNCTION TABLE:

---------------------------------------------------------------------------------
!            INPUTS                            !               OUTPUTS          !
---------------------------------------------------------------------------------
! CLEAR ! I2 I1 I0 ! LIO RIO ! CLOCK ! LIO ! A3 A2 A1 A0 ! RIO ! OPERATION !
---------------------------------------------------------------------------------
!   L   !  L  L  L !  X   X  !  L-H  !  Z  ! A3 A2 A1 A0 !  Z  ! NOP       !
!   L   !  L  L  H !  X   X  !  L-H  !  Z  ! B3 B2 B1 B0 !  Z  ! LOAD B    !
!   L   !  L  H  L !  RI  Z  !  L-H  !  Z  ! RI A3 A2 A1 !  A1 ! SHIFT RT  !
!   L   !  L  H  H !  X   X  !  L-H  !  Z  !   ALL HIGH  !  Z  ! SET       !
!   L   !  H  L  L !  Z   LI !  L-H  !  A2 ! A2 A1 A0 LI !  Z  ! SHIFT LT  !
!   L   !  H  L  H !  Z   X  !  L-H  !  L  !   ALL HIGH  !  Z  ! SET       !
!   L   !  H  H  L !  Z  CIN !  L-H  ! COUT! A PLUS ONE  !  Z  ! INC IF CIN!
!   L   !  H  H  H !  Z  BIN !  L-H  ! BOUT! A MINUS ONE !  Z  ! DEC IF BIN!
!   H   !  X  X  X !  X   X  !  L-H  !  Z  !   ALL LOW   !  Z  ! CLEAR     !
---------------------------------------------------------------------------------
```

4-Bit Up/Down Counter with Shift and Three-State Outputs   Logic Diagram PAL16X4



ALU Accumulator

Design Specification PAL16A4

```
PAL16A4                                            PAL DESIGN SPECIFICATION
PAT0026                                               JOHN BIRKNER 12/15/77
ALU/ACCUMULATOR

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.EQ.B0) + /I3◆/I2◆/I1◆I0◆(/A0) + /I3◆/I2◆I1◆/I0◆(/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.EQ.B1) + /I3◆/I2◆/I1◆I0◆(/A1) + /I3◆/I2◆I1◆/I0◆(/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.EQ.B2) + /I3◆/I2◆/I1◆I0◆(/A2) + /I3◆/I2◆I1◆/I0◆(/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.EQ.B3) + /I3◆/I2◆/I1◆I0◆(/A3) + /I3◆/I2◆I1◆/I0◆(/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

IF(VCC) 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)

IF(VCC) 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)

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

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.

---------------------------------------------------------------------------------
!        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   LI !  A2 ! A2 A1 A0 LI !  Z  ! SHIFT LEFT                   !
!  L  H  H  H !  RI  X  !  Z  ! RI A3 A2 A1 !  A1 ! SHIFT RIGHT                  !
!  H  X  X  X !  X   X  !  Z  ! A3 A2 A1 A0 !  Z  ! NOP       A:=A               !
---------------------------------------------------------------------------------
```

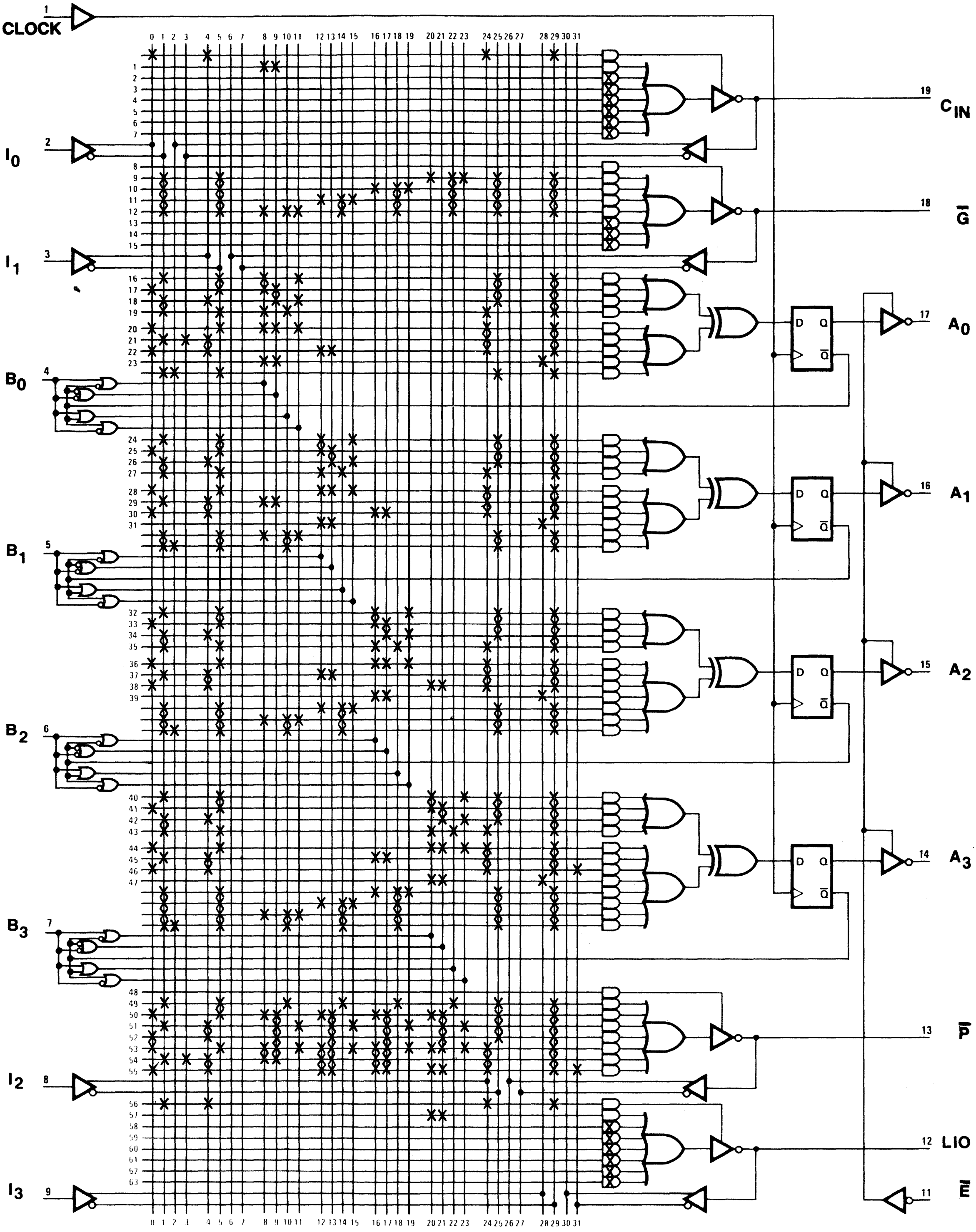
ALU Accumulator

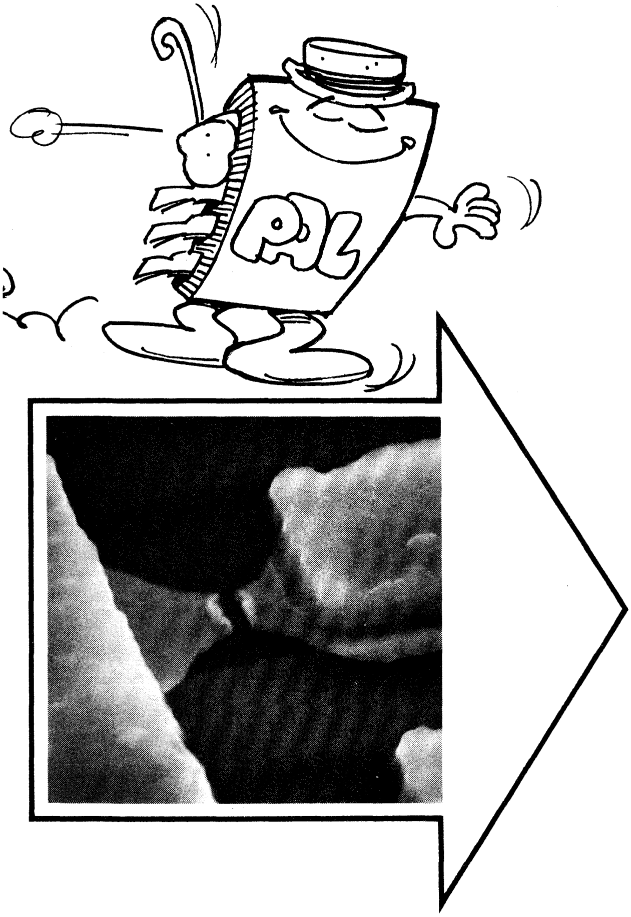
Fuse Pattern PAL16A4

```
X--- X--- ---- ---- ---- ---- X--- -X-- /I3◆I2◆I1◆I0
---- ---- XX-- ---- ---- ---- ---- ---- /A0
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

---- ---- ---- ---- ---- ---- ---- ----
-X-- -X-- ---- ---- ---- X-XX -X-- -X-- /I3◆/I2◆/I1◆/I0◆A3◆B3
-X-- -X-- ---- ---- X-XX --X- -X-- -X-- /I3◆/I2◆/I1◆/I0◆A3+B3◆A2◆B2
-X-- -X-- ---- X-XX --X- --X- -X-- -X-- /I3◆/I2◆/I1◆/I0◆A3+B3◆A2+B2◆A1◆
-X-- -X-- X-XX --X- --X- --X- -X-- -X-- /I3◆/I2◆/I1◆/I0◆A3+B3◆A2+B2◆A1+
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

-X-- -X-- X--X ---- ---- ---- -X-- -X-- /I3◆/I2◆/I1◆/I0◆A0.EQ.B0
X--- -X-- XX-- ---- ---- ---- -X-- -X-- /I3◆/I2◆/I1◆I0◆/A0
-X-- X--- -X-X ---- ---- ---- -X-- -X-- /I3◆/I2◆I1◆/I0◆/B0
-X-- -X-- X-X- ---- ---- ---- X--- -X-- /I3◆I2◆/I1◆/I0◆B0
X--- -X-- XX-X ---- ---- ---- X--- -X-- /I3◆I2◆/I1◆I0◆/A0◆/B0
-X-X X--- ---- ---- ---- ---- X--- -X-- /I3◆I2◆I1◆/I0◆/CIN
X--- X--- ---- XX-- ---- ---- X--- -X-- /I3◆I2◆I1◆I0◆/A1
---- ---- XX-- ---- ---- ---- ---- X--- I3◆/A0

-X-- -X-- ---- X--X ---- ---- -X-- -X-- /I3◆/I2◆/I1◆/I0◆A1.EQ.B1
X--- -X-- ---- XX-- ---- ---- -X-- -X-- /I3◆/I2◆/I1◆I0◆/A1
-X-- X--- ---- -X-X ---- ---- -X-- -X-- /I3◆/I2◆I1◆/I0◆/B1
-X-- -X-- ---- X-X- ---- ---- X--- -X-- /I3◆I2◆/I1◆/I0◆B1
X--- -X-- ---- XX-X ---- ---- X--- -X-- /I3◆I2◆/I1◆I0◆/A1◆/B1
-X-- X--- XX-- ---- ---- ---- X--- -X-- /I3◆I2◆I1◆/I0◆/A0
X--- X--- ---- ---- XX-- ---- X--- -X-- /I3◆I2◆I1◆I0◆/A2
---- ---- ---- XX-- ---- ---- ---- X--- I3◆/A1

-X-- -X-- ---- ---- X--X ---- -X-- -X-- /I3◆/I2◆/I1◆/I0◆A2.EQ.B2
X--- -X-- ---- ---- XX-- ---- -X-- -X-- /I3◆/I2◆/I1◆I0◆/A2
-X-- X--- ---- ---- -X-X ---- -X-- -X-- /I3◆/I2◆I1◆/I0◆/B2
-X-- -X-- ---- ---- X-X- ---- X--- -X-- /I3◆I2◆/I1◆/I0◆B2
X--- -X-- ---- ---- XX-X ---- X--- -X-- /I3◆I2◆/I1◆I0◆/A2◆/B2
-X-- X--- ---- XX-- ---- ---- X--- -X-- /I3◆I2◆I1◆/I0◆/A1
X--- X--- ---- ---- ---- XX-- X--- -X-- /I3◆I2◆I1◆I0◆/A3
---- ---- ---- ---- XX-- ---- ---- X--- I3◆/A2

-X-- -X-- ---- ---- ---- X--X -X-- -X-- /I3◆/I2◆/I1◆/I0◆A3.EQ.B3
X--- -X-- ---- ---- ---- XX-- -X-- -X-- /I3◆/I2◆/I1◆I0◆/A3
-X-- X--- ---- ---- ---- -X-X -X-- -X-- /I3◆/I2◆I1◆/I0◆/B3
-X-- -X-- ---- ---- ---- X-X- X--- -X-- /I3◆I2◆/I1◆/I0◆B3
X--- -X-- ---- ---- ---- XX-X X--- -X-- /I3◆I2◆/I1◆I0◆/A3◆/B3
-X-- X--- ---- ---- XX-- ---- X--- -X-- /I3◆I2◆I1◆/I0◆/A2
X--- X--- ---- ---- ---- ---- X--- -X-X /I3◆I2◆I1◆I0◆/LID
---- ---- ---- ---- ---- XX-- ---- X--- I3◆/A3

---- ---- ---- ---- ---- ---- ---- ----
-X-- -X-- --X- --X- --X- --X- -X-- -X-- /I3◆/I2◆/I1◆/I0◆A3+B3◆A2+B2◆A1+
X--- -X-- XX-- XX-- XX-- XX-- -X-- -X-- /I3◆/I2◆/I1◆I0◆/A3◆/A2◆/A1◆/A0
-X-- X--- -X-X -X-X -X-X -X-X -X-- -X-- /I3◆/I2◆I1◆/I0◆/B3◆/B2◆/B1◆/B0
X--- X--- -X-- -X-- -X-- -X-- -X-- -X-- /I3◆/I2◆I1◆I0◆/A3+/B3◆/A2+/B2◆/
X--- -X-- XX-X XX-X XX-X XX-X X--- -X-- /I3◆I2◆/I1◆I0◆/A3◆/B3◆/A2◆/B2◆/
-X-X X--- XX-- XX-- XX-- ---- X--- -X-- /I3◆I2◆I1◆/I0◆/A2◆/A1◆/A0◆/CIN
X--- X--- ---- XX-- XX-- XX-- X--- -X-X /I3◆I2◆I1◆I0◆/LID◆/A3◆/A2◆/A1

-X-- X--- ---- ---- ---- ---- X--- -X-- /I3◆I2◆I1◆/I0
---- ---- ---- ---- ---- XX-- ---- ---- /A3
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
```

ALU Accumulator

Logic Diagram PAL16A4




PAL

Representatives

U.S.A.

Alabama	
Huntsville	
REP, Inc.	(205) 881-9270
Arizona	
Scottsdale	
Summit Sales	(602) 994-4587
California	
Culver City	
Bestronics	(213) 870-9191
Irvine	
Bestronics	(714) 979-9910
Mountain View	
Thresum Associates	(415) 965-9180
San Diego	
Littlefield & Smith	(714) 455-0055
Colorado	
Wheatridge	
Waugaman Associates	(303) 423-1020
Connecticut	
North Haven	
Comp Rep Associates	(203) 239-9762
Florida	
Altamonte Springs	
Dyne-A-Mark	(305) 831-2097
Clearwater	
Dyne-A-Mark	(813) 441-4702
Fort Lauderdale	
Dyne-A-Mark	(305) 771-6501
Georgia	
Tucker	
REP, Inc.	(404) 938-4358
Illinois	
Rolling Meadows	
Sumer	(312) 394-4900
Indiana	
Indianapolis	
Electro Reps	(317) 255-4147
Iowa	
Cedar Rapids	
S & O Sales	(319) 393-1845
Kansas	
Olathe	
Rush and West	(913) 764-2700
Maryland	
Baltimore	
Monolithic Sales	(301) 296-2444
Rockville	
Monolithic Sales	(301) 340-2130
Massachusetts	
Needham Heights	
Comp Rep Associates	(617) 444-2484
Michigan	
Grosse Point	
Greiner Associates	(313) 499-0188
Minnesota	
Minneapolis	
Nortec Sales	(612) 835-7414
Missouri	
Ballwin	
Rush and West	(314) 394-7271
New Jersey	
Teaneck	
R.T. Reid Associates	(201) 692-0200
New York	
Rochester	
L-Mar Associates	(716) 328-5240
Syracuse	
L-Mar Associates	(315) 437-7779
North Carolina	
Raleigh	
REP, Inc.	(919) 851-3007
Ohio	
Cincinnati	
Makin Associates	(513) 871-2424
Mentor	
Makin Associates	(216) 464-4330
Oregon	
Portland	
N.W. Marketing	(503) 297-2581
Pennsylvania	
Oreland	
CMS Marketing	(215) 885-5106
Tennessee	
Jefferson City	
REP, Inc.	(615) 475-4105
Texas	
Dallas	
West and Associates	(214) 661-9400
Utah	
Salt Lake City	
Waugaman Associates	(801) 363-0275
Washington	
Bellevue	
Northwest Marketing	(206) 455-5846
Wisconsin	
Milwaukee	
Sumer	(414) 259-9060

CANADA

Ontario	
Milton	
Cantec	(416) 624-9696
Ottawa	
Cantec	(613) 225-0363
Quebec	
Pierre Ponds	
Cantec	(514) 620-3121

Distributors

U.S.A.

Alabama	**Huntsville**	Hall-Mark Electronics	(205) 837-8700
Arizona	**Phoenix**	Kierulff Electronics	(602) 243-4101
		Sterling Electronics	(602) 258-4531
California	**Los Angeles**	Kierulff Electronics	(213) 685-5511
	Palo Alto	Kierulff Electronics	(415) 968-6292
	San Diego	Intermark Electronics	(714) 279-5200
		Kierulff Electronics	(714) 278-2112
	Santa Ana	Intermark Electronics	(714) 540-1322
	Sunnyvale	Diplomat/Westland	(408) 734-1900
		Intermark Electronics	(408) 738-1111
	Tustin	Kierulff Electronics	(714) 731-5711
Colorado	**Denver**	Kierulff Electronics	(303) 371-6500
	Wheatridge	Century Electronics	(303) 424-1985
Connecticut	**Hamden**	Arrow Electronics	(203) 248-3801
	Wilton	Components for Industries	(203) 762-8691
Florida	**Clearwater**	Diplomat/Southland	(813) 443-4514
	Fort Lauderdale	Arrow Electronics	(305) 776-7790
		Hall-Mark Electronics	(305) 971-9280
	Orlando	Hall-Mark Electronics	(305) 855-4020
Illinois	**Elk Grove Village**	Hall-Mark Electronics	(312) 437-8800
		Kierulff Electronics	(312) 640-0200
Indiana	**Indianapolis**	Advent Electronics	(317) 297-4910
Kansas	**Shawnee Mission**	Hall-Mark Electronics	(913) 888-4747
Maryland	**Baltimore**	Arrow Electronics	(202) 737-1700
		Pyttronics/Savage	(301) 792-0780
	Gaithersburg	Pioneer Washington	(301) 948-0710
	Savage	Pyttronics Industries	(301) 792-0780
Massachusetts	**Billerica**	Kierulff Electronics	(617) 935-5134
	Burlington	Lionex	(617) 272-9400
	Woburn	Arrow Electronics	(617) 933-8130
Michigan	**Ann Arbor**	Arrow Electronics	(313) 971-8220
	Farmington	Diplomat/Northland	(313) 477-3200
Minnesota	**Bloomington**	Arrow Electronics	(612) 887-6400
		Hall-Mark Electronics	(612) 884-9056
Missouri	**Earth City**	Hall-Mark Electronics	(314) 291-5350
New Hampshire	**Manchester**	Arrow Electronics	(603) 668-6968
New Jersey	**Fairfield**	Kierulff Electronics	(201) 575-6750
	Totowa	Diplomat/IPC	(201) 785-1830
	Moorestown	Arrow Electronics	(609) 235-1900
	Rutherford	Kierulff Electronics	(201) 935-2120
	Saddlebrook	Arrow Electronics	(201) 797-5800
New Mexico	**Albuquerque**	Century Electronics	(505) 292-2700
New York	**Buffalo**	Summit Distributors	(716) 884-3450
	Farmingdale	Arrow Electronics	(516) 694-6800
	Rochester	Summit Distributors	(716) 334-8110
	Smithtown	Current Components	(516) 979-9030
	Woodbury	Diplomat Electronics	(516) 921-9373
North Carolina	**Raleigh**	Hall-Mark Electronics	(919) 832-4465
Ohio	**Cleveland**	Arrow Electronics	(216) 464-2000
	Dayton	Arrow Electronics	(513) 253-9176
Oklahoma	**Tulsa**	Hall-Mark Electronics	(918) 835-8458
Oregon	**Portland**	Almac/Stroum Electronics	(503) 292-3534
Pennsylvania	**Horsham**	Pioneer/Delaware Valley	(215) 674-4000
Texas	**Austin**	Hall-Mark Electronics	(512) 837-2814
		Quality Components	(512) 458-4181
	Dallas	Hall-Mark Electronics	(214) 234-7400
		Quality Components	(214) 387-4949
	Houston	Hall-Mark Electronics	(713) 781-6100
		Quality Components	(713) 772-7100
		Sterling Electronics	(713) 627-9800
Utah	**Salt Lake City**	Calron Electric Supply	(801) 487-7451
		Century Electronics	(810) 972-6969
Washington	**Seattle**	Almac/Stroum Electronics	(206) 763-2300
	Tukwila	Kierulff Electronics	(206) 575-4420
Wisconsin	**West Allis**	Hall-Mark Electronics	(414) 476-1270

CANADA

Ontario	**Ottawa**	Zentronics Limited	(613) 238-6411
	Toronto	Future Electronics	(416) 675-7820
		Zentronics Limited	(613) 238-6411
British Columbia	**Vancouver**	Bowtek Electric	(604) 736-1141
Manitoba	**Winnipeg**	Bowtek Electric	(204) 633-9523
Quebec	**Montreal**	Future Electronics	(514) 735-5775
		Zentronics Limited	(514) 735-5361
Alberta	**Edmonton**	Bowtek Electric	(403) 426-1072

Overseas Representatives and Distributors

EUROPE

Headquarters

Monolithic Memories, GMBH
Mauerkircherstr. 4
8000 München 80
West Germany
Phone: 89-982601
Telex: 524385
Fax: 89-983162

AUSTRIA
Ing. Ernst Steiner
Geylinggasse 16
1130 Wien
Phone: 222-822674
Telex: 74013

BELGIUM
Ritro Electronics N.V.
Plantin & Moretuslei 172
2000 Antwerpen
Phone: 31-353272
Telex: 33637

DENMARK
C-88
Uldvejen 10
DK 2970 Hørsholm
Phone: 2-570888
Telex: 37578

ENGLAND
Memory Devices Ltd.
Central Avenue
East Molesey
KT8 OSN
Phone: 1-9411066
Telex: 929962

Macro Marketing Ltd.
396 Bath Road
Slough, Berkshire
Phone: 6286-63011
Telex: 847083

FINLAND
Telercas O.Y.
P.O. Box 2
01511 Vantaa 51
Phone: 0-821655
Telex: 121111

FRANCE
Monolithic Memories France S.A.R.L.
Silic 463
94613 Rungis Cedex
Paris
Phone: 1-6874500
Telex: 202146F

ALFATRONIC
La Tour d'Asnieres 4
Avenue Laurent
Cely 92606
Asnieres / Frankreich
Phone: Asnieres 1-7914444
Telex: 612790

A2M
Assistance Microprocesseurs
Microprogrammation
40, Rue des Tilleuls
92100 Boulogne
Phone: 1-6036640
Telex: 200491

Radio Equipments-Antares S.A.
9, Rue Ernest Cognacq
92301 Levallois Perret
Phone: 1-7581111
Telex: 620630

GERMANY
Monolithic Memories, GMBH
Mauerkircherstr. 4
8000 München 80
West Germany
Phone: 89-982601
Telex: 524385
Fax: 89-983162

ASTRONIC GMBH
Winzererstrasse 47 D
8000 München 40
Phone: 89-304011
Telex: 5216187

ELECTRONIC 2000
Vertriebs-GMBH
Neumarkter Strasse 75
8000 München 80
Phone: 89-434061
Telex: 522561

PANEL Electronic Vertriebs GMBH
Hermann Oberth Str. 7
8011 Putzbrunn
Phone: 89-464024
Telex: 529238

POSITRON Bauelemente
Vertriebs GMBH
Benzstrasse 1
Postfach 1140
7016 Gerlingen
Phone: 7156-23051
Telex: 7245266

NETHERLANDS
Ritro Electronics B.V.
Gelreweg 22
Postbus 123
2930 Barneveld
Phone: 3420-5041
Telex: 40553

Famatra Benelux
P.O. Box 721
4803 AS Breda
Phone: 76-133457
Telex: 54521

ISRAEL
TELSYS Ltd.
54, Jabotinsky Rd.
Ramat-Gan
Phone: 3-739865
Telex: 032392

ITALY
Comprel S.R.L.
20092 Cinisello Balsamo/Milano
Viale Romagna 1
Phone: 2-9280809
Telex: 39484

NORWAY
Henaco A/S
P.O. Box 248
Okern Torgvei 13
Oslo 5
Phone: 2-157550
Telex: 16716

SPAIN
Comisa Ingenieros S.A.
Reina Mercedes, 20
Madrid 20
Spanien
Phone: 1-2542901/04
Telex: 237231

SWEDEN
NAXAB
Box 4115
17104 Solna
Phone: 8-985140
Telex: 17912

SWITZERLAND
Baerlocher A.G.
Förrlibuckstrasse 110
8005 Zürich
Phone: 1-429900
Telex: 53118

FAR EAST

Headquarters

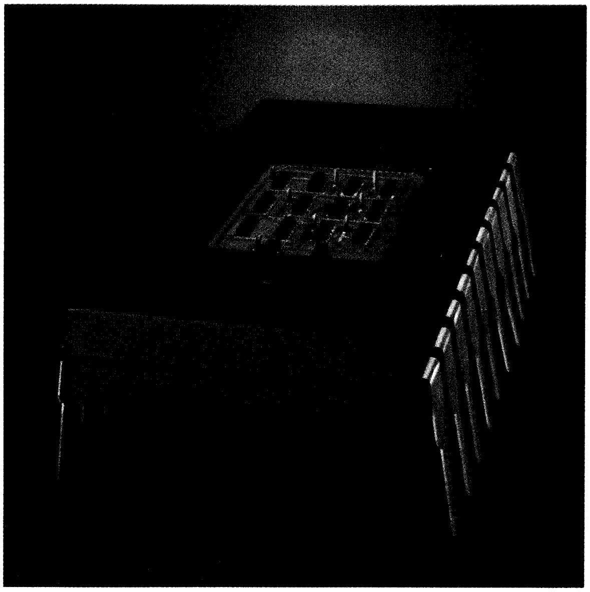
Monolithic Memories Japan KK
Parkside-Flat Bldg.
4-2-2, Sendagaya
Shibuya-Ku
Tokyo 151
Phone: 3-4039061
Telex: 781-26364

AUSTRALIA
R & D Electronics Pty Ltd.
23 Burwood Rd.
Burwood, Vic. 3125
Phone: 288-8232
Telex: AA33288

INDIA
Chawla Sales
3481, Netaji Subhash Marg.
Daryaganj, New Delhi — 110002
Phone: 277388

Zenith Electronics
541 Panchratna
Mama Parmanand Marg
Bombay 4
Phone: 384212
Telex: 0113152

PAL
PROGRAMMABLE ARRAY LOGIC
Handbook



Monolithic Memories
Bipolar is our business.

MONOLITHIC MEMORIES GMBH · EUROPEAN HEADQUARTERS
MAUERKIRCHERSTR. 4 · D-8000 MÜNCHEN 80
TELEFON (089) *98 26 01 · TELEX 5 24385 mono d
TELEKOPIERER INFOTEC 6000: 98 31 62

MMI reserves the right to make changes in order to improve circuitry and supply the best product possible.

Monolithic Memories cannot assume responsibility for the use of any circuitry described other than circuitry entirely embodied in their product. No other circuit patent licenses are implied.