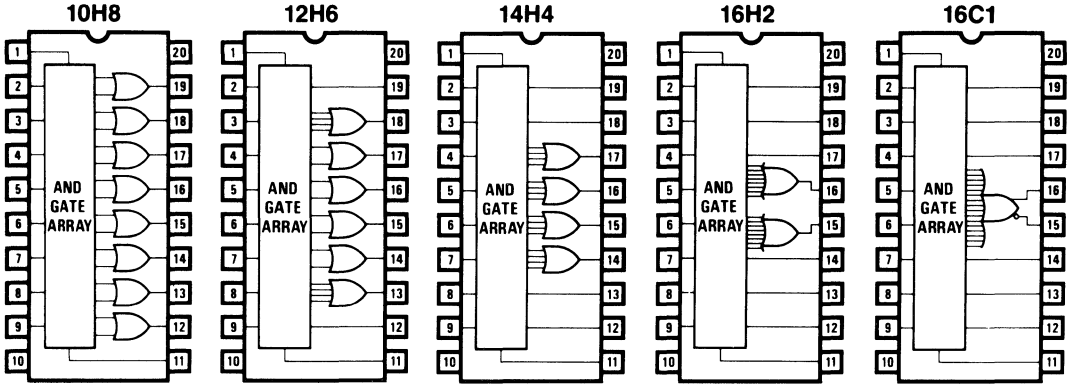
$14.95

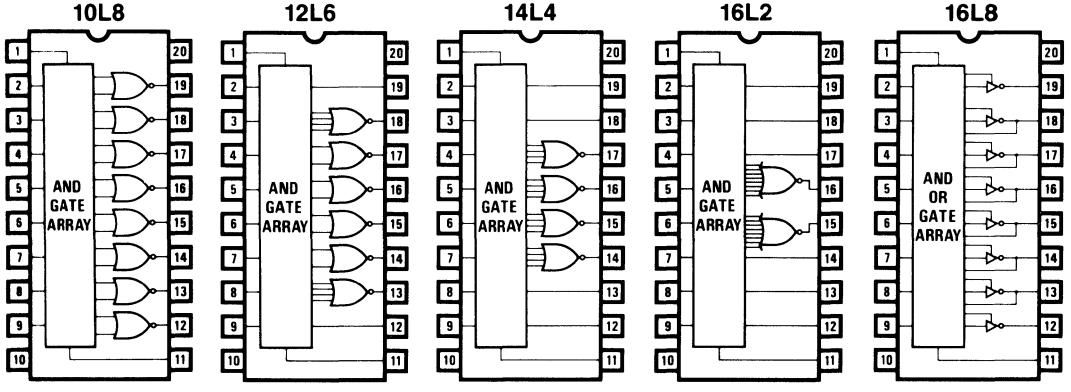
PAL® PROGRAMMABLE ARRAY LOGIC Handbook

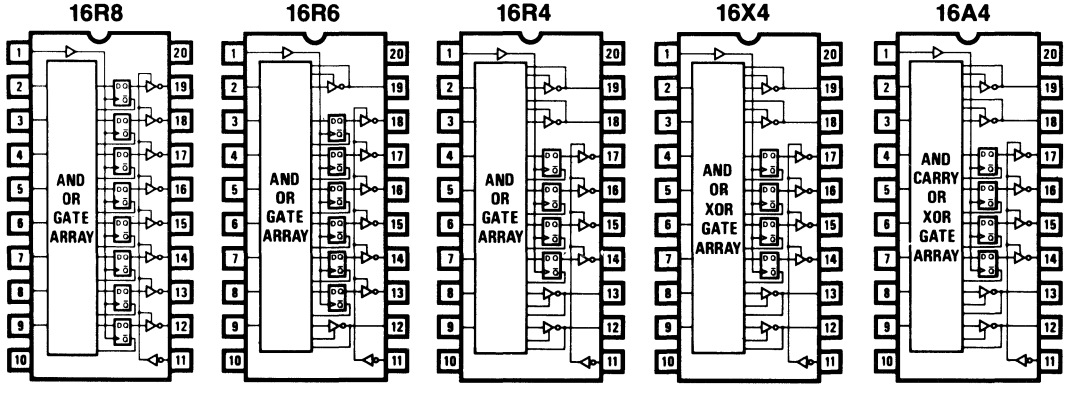
THIRD EDITION

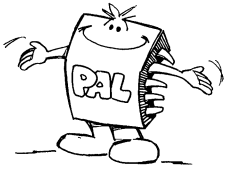
Monolithic MMI Memories

20-Pin PAL Logic Symbols









PAL®
PROGRAMMABLE ARRAY LOGIC
Handbook

THIRD EDITION

Authors: John M. Birkner & Vincent J. Coli
Video Controller by Ehud Gordon

Monolithic Memories MMI

Acknowledgements

The following employees of Monolithic Memories made contributions to this 3rd edition of the PAL HANDBOOK. Specific technical contributions are sited on the second line of the PAL DESIGN SPECIFICATION.

Product Planning and Applications

Imtiyaz Bengali
Ehud Gordon
Earle Jennings
Frank Lee
Nadia Sachs
Danesh Tavana
Mano Vafai
Shlomo Waser

Sales

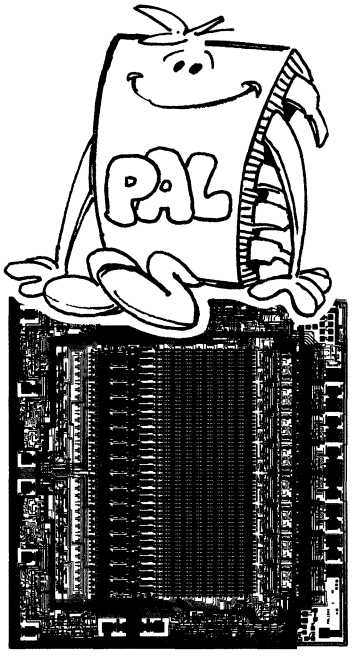
Bernard Brafman
Harry Hughes
Dan Jesswein
Dick Jones
Brad Mitchell
Willy Voldan
Mike Volpigno

To these individuals we extend our gratitude,

The Authors
John M. Birkner
Vincent J. Coli

Table of Contents

Table of Contents (cont.)



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 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. Monolithic Memories offers the programmable solution.

The PAL family offers a fresh approach to using fuse programmable logic. PAL circuits are a conceptually unified group of devices which combine programmable flexibility with high speed and an extensive selection of interface options. PAL devices 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 PAL circuits 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 circuit uses are only limited by the number of terms available in the AND - OR arrays. PAL devices 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

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

where the "f" terms represent the state of the fusible links in the PAL 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.

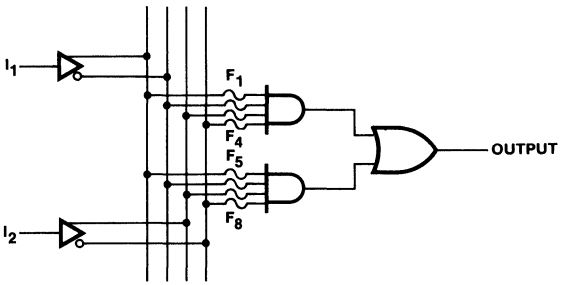


Figure 1

PAL 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 PAL circuits. The two input - one output example from Figure 1 redrawn using the new logic convention is shown in Figure 3.

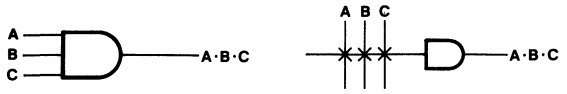


Figure 2

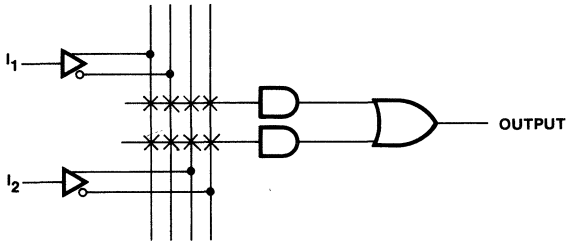


Figure 3

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.

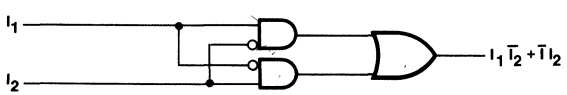


Figure 4

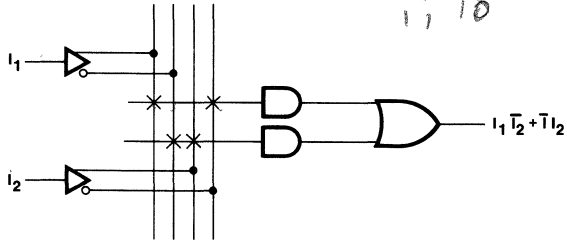


Figure 5

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.

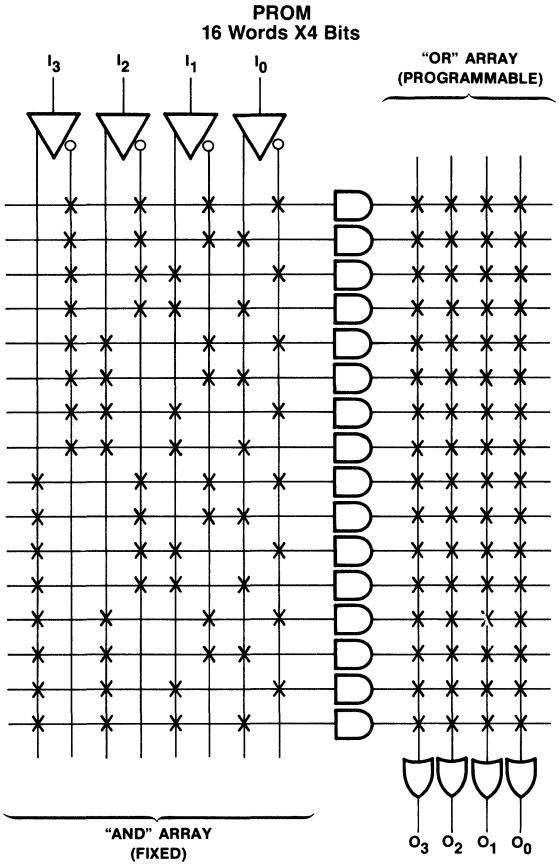


Figure 6

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 PLAs expensive, quite formidable to understand, and 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.

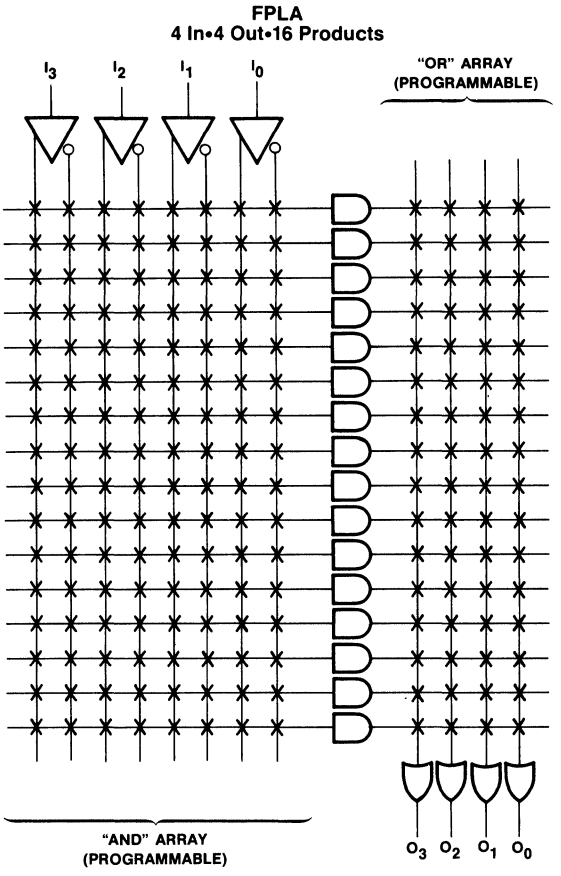


Figure 7

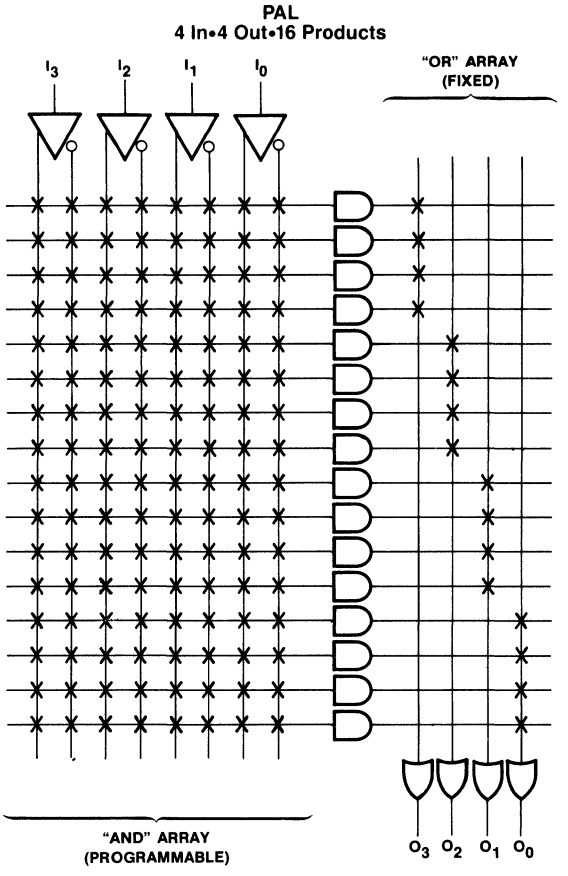


Figure 8

	AND	OR	OUTPUT OPTIONS
PROM	Fixed	Prog	TS, OC
FPLA	Prog	Prog	TS, OC, Fusible Polarity
FPGA	Prog	None	TS, OC, Fusible Polarity
FPLS	Prog	Prog	TS, Registered Feedback, I/O
PAL	Prog	Fixed	TS, Registered Feedback, I/O

Table 1

PAL Input/Output/Function/Performance Chart

PART NO.	INPUT	OUTPUT	PROGRAMMABLE I/O'S	FEEDBACK REGISTER	OUTPUT POLARITY	FUNCTIONS	PERFORMANCE			
							STD	A	-2	-4
10H8	10	8			AND-OR	AND-OR Gate Array	X		X	
12H6	12	6			AND-OR	AND-OR Gate Array	X		X	
14H4	14	4			AND-OR	AND-OR Gate Array	X		X	
16H2	16	2			AND-OR	AND-OR Gate Array	X		X	
16C1	16	2			BOTH[1]	AND-OR Gate Array	X		X	
20C1	20	2			BOTH[1]	AND-OR Gate Array	X			
10L8	10	8			AND-NOR	AND-OR Invert Gate Array	X		X	
12L6	12	6			AND-NOR	AND-OR Invert Gate Array	X		X	
14L4	14	4			AND-NOR	AND-OR Invert Gate Array	X		X	
16L2	16	2			AND-NOR	AND-OR Invert Gate Array	X		X	
12L10	12	10			AND-NOR	AND-OR Invert Gate Array	X			
14L8	14	8			AND-NOR	AND-OR Invert Gate Array	X			
16L6	16	6			AND-NOR	AND-OR Invert Gate Array	X			
18L4	18	4			AND-NOR	AND-OR Invert Gate Array	X			
20L2	20	2			AND-NOR	AND-OR Invert Gate Array	X			
16L8	10	2	6		AND-NOR	AND-OR Invert Gate Array	X	X	X	X
20L8	14	2	6		AND-NOR	AND-OR Invert Gate Array		X		
20L10	12	2	8		AND-NOR	AND-OR Invert Gate Array	X			
16R8	8	8		8	AND-NOR	AND-OR Invert Gate Array w/Reg's	X	X	X	X
16R6	8	6	2	6	AND-NOR	AND-OR Invert Array w/Reg's	X	X	X	X
16R4	8	4	4	4	AND-NOR	AND-OR Invert Array w/Reg's	X	X	X	X
20R8	12	8		8	AND-NOR	AND-OR Invert w/Reg's		X		
20R6	12	6	2	6	AND-NOR	AND-OR Invert w/Reg's		X		
20R4	12	4	4	4	AND-NOR	AND-OR Invert w/Reg's		X		
20X10	10	10		10	AND-NOR	AND-OR-XOR Invert w/Reg's	X			
20X8	10	8	2	8	AND-NOR	AND-OR-XOR Invert w/Reg's	X			
20X4	10	4	6	4	AND-NOR	AND-OR-XOR Invert w/Reg's	X			
16X4	8	4	4	4	AND-NOR	AND-OR-XOR Invert w/Reg's	X			
16A4	8	4	4	4	AND-NOR	AND-CARRY-OR-XOR Invert w/Reg's	X			

Table 2

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

PAL Circuits 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. PAL units come in the following basic configurations:

Gate Arrays

PAL gate arrays are available in sizes from 12x10 (12 input terms, 10 output terms) to 20x2, with both active high and active low output configurations available (figure 9). This wide variety of input/output formats allows the PAL to replace many different sized blocks of combinatorial logic with single packages.

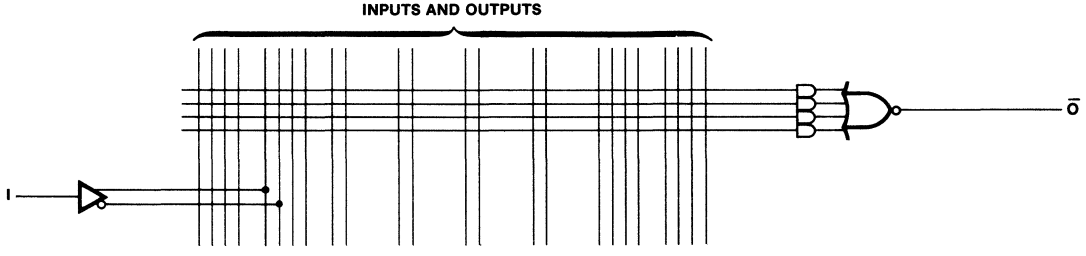


Figure 9

Programmable I/O

A 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.

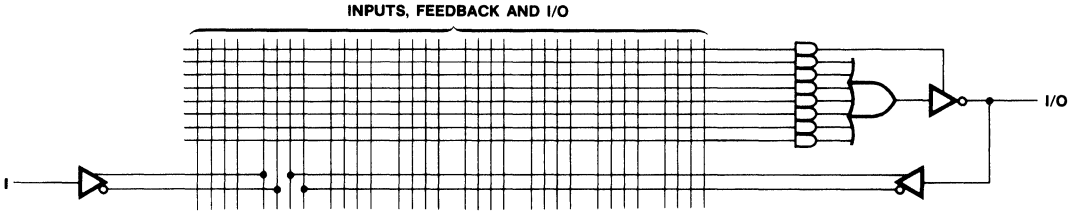


Figure 10

Registered Outputs with Feedback

Another feature of the high end members of the PAL family is registered data outputs with registered feedback. Each product term is stored into a D-type output flip-flop on the rising edge of the system clock (Figure 11). 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 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 25 MHz.

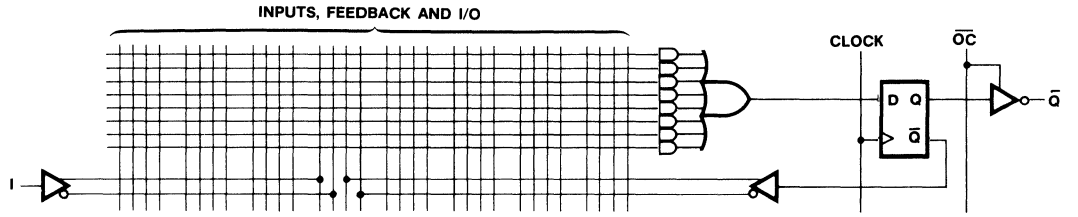


Figure 11

XOR PALs

These PAL devices feature an exclusive OR function. The sum of products is segmented into two sums which are then exclusive ORed (XOR) at the input of the D-type flip-flop (Figure 12). All of the features of the Registered PALs are included in the XOR PAL unit. The XOR function provides an easy implementation of the HOLD operation used in counters and other state sequencers.

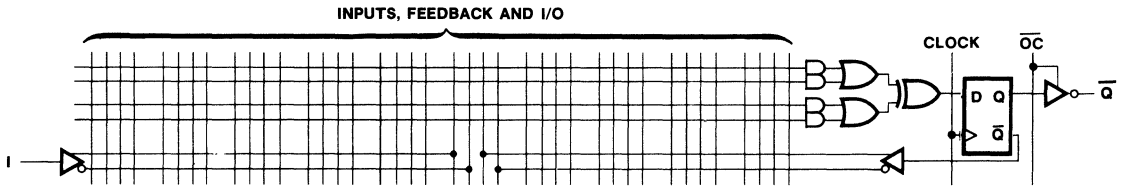


Figure 12

Arithmetic Gated Feedback

The arithmetic functions (add, subtract, greater than, and less than) are implemented by addition of gated feedback to the features of the XOR PAL device. The XOR at the input of the D-type flip-flop allows carrys from previous operations to be XORed with two variable sums generated by the PAL array. The flip-flop Q output is fed back to be gated with input terms A (Figure 13). This gated feedback provides any one of the 16 possible Boolean combinations which are mapped in the Karnaugh map (Figure 15). Figure 14 shows how the PAL array can be programmed to perform these 16 operations. These features provide for versatile operations on two variables and facilitate the parallel generation of carrys necessary for fast arithmetic operations.

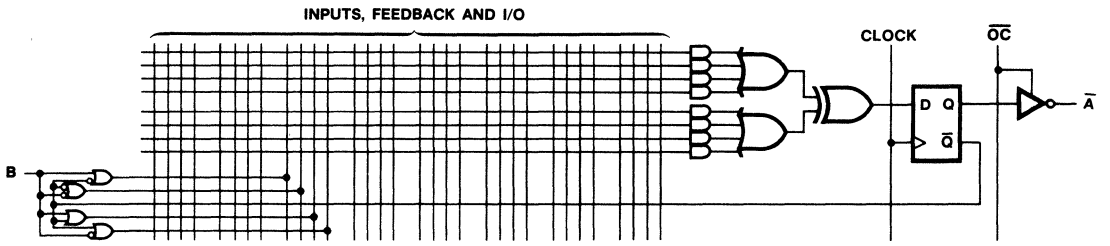


Figure 13

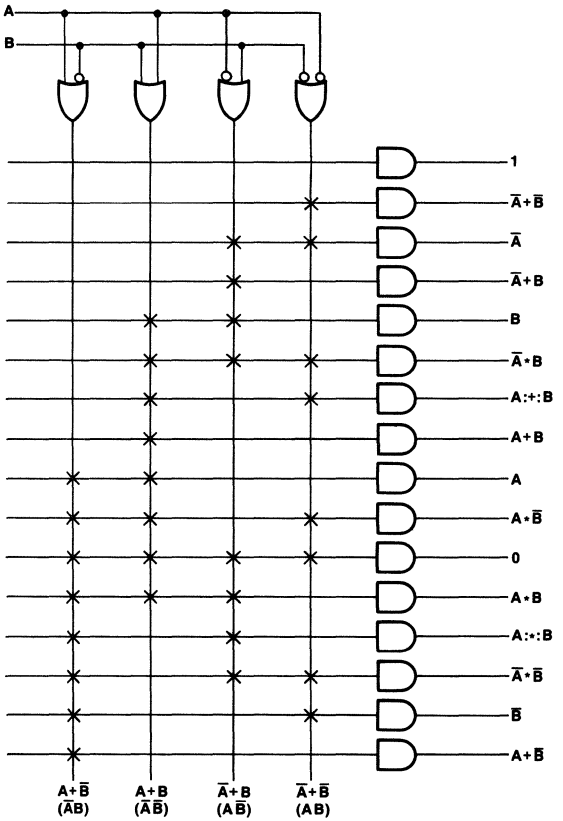


Figure 14

$(\overline{A}+B),(\overline{A}+\overline{B})$ / $(A+\overline{B}),(A+B)$	--	-x	xx	x-
--	1	$\overline{A}+\overline{B}$	$\overline{A}$	$\overline{A}+B$
-x	A+B	A:+:B	$\overline{A}*B$	B
xx	A	$A*\overline{B}$	0	A*B
x-	$A+\overline{B}$	$\overline{B}$	$\overline{A}*\overline{B}$	A:*:B

Figure 15

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.

PAL Programming

PAL devices can be programmed in most standard PROM programmers with the addition of a PAL personality card. The PAL appears to the programmer as a PROM. During programming half of the PAL outputs are selected for programming while the other outputs and the 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.

PALASM (PAL Assembler)

PALASM is the software used to define, simulate, build, and test PAL units. PALASM accepts the PAL Design Specification as an input file. It verifies the design against an optional function table and generates the fuse plot which is used to program the PAL devices. PALASM is available upon request for many computers and is documented in the PAL Design Concepts section.

HAL (Hard Array Logic)

The HAL family is the mask programmed version of a PAL. The HAL is to a PAL just as a ROM is to a PROM. A standard wafer is fabricated to the 6th mask. Then a custom metal mask is used to fabricate Aluminum links for a HAL instead of the programmable Ti-W fuse array used in a PAL.

The HAL is a cost-effective solution for large quantities and is unique in that it is a gate array with a programmable prototype.

HMSI (HAL Medium Scale Integration)

The HMSI family is derived from the PAL using HAL technology. These devices perform predetermined functions which are not available in the existing TTL family. Because they are produced in volume, the user receives the benefit of volume pricing. HMSI PAL designs are given in the Applications section with their industry standard 74LS part number in line 2 of the PAL Design Specification.

PMSI (PAL Medium Scale Integration)

The PMSI family is derived in a similar fashion to HMSI except this product is produced entirely from a PAL circuit. A HAL circuit mask is not generated and an industry standard 74LS part number is not assigned unless sales warrant it.

PAL Technology

PAL circuits are manufactured using the proven TTL Schottky bipolar Ti-W fuse process to make fusible-link PROMs. An NPN emitter follower array forms the programmable AND array. PNP inputs provide high-impedance inputs (0.25 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 and 24-pin SKINNYDIP® packages.

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.

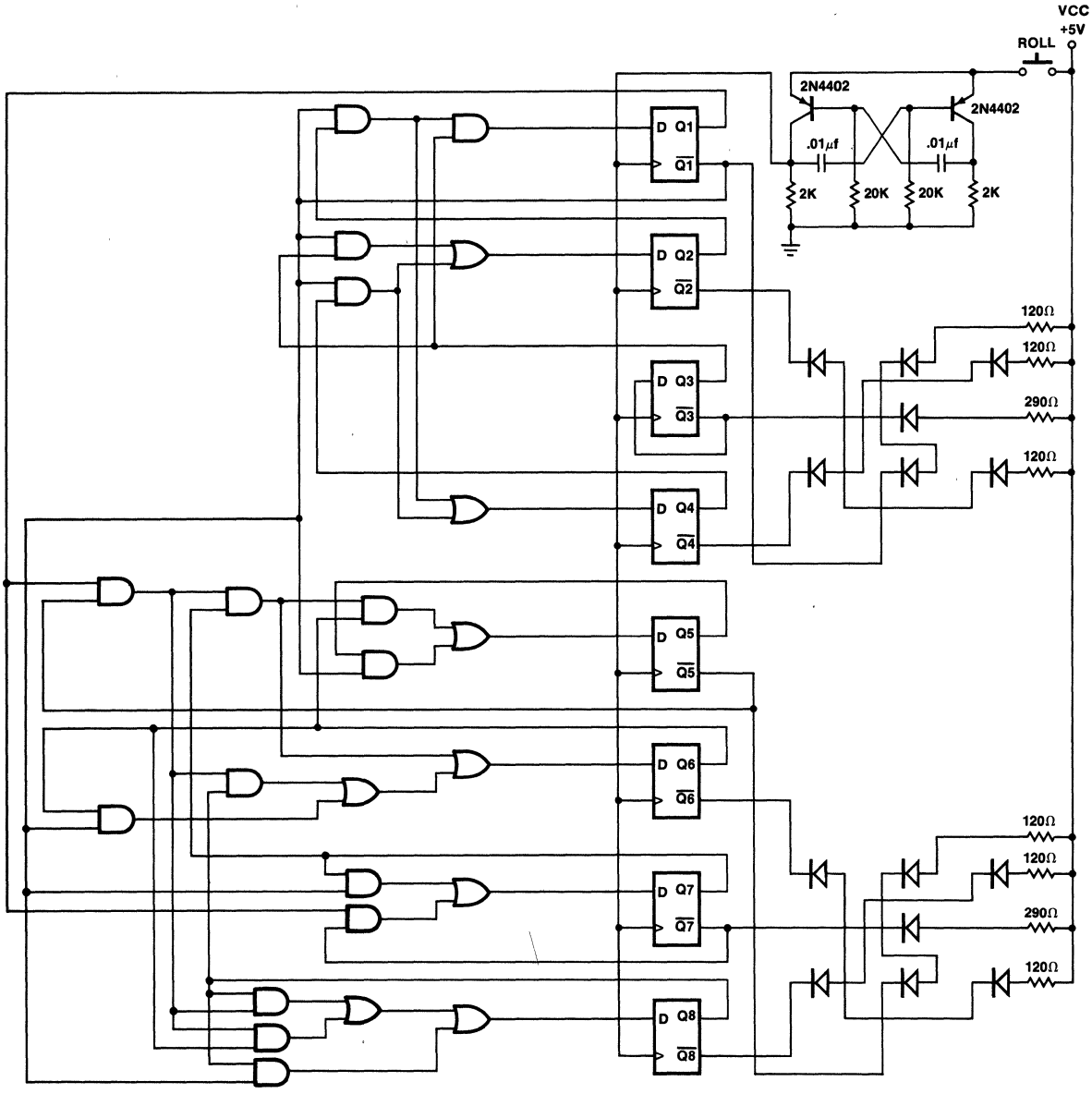
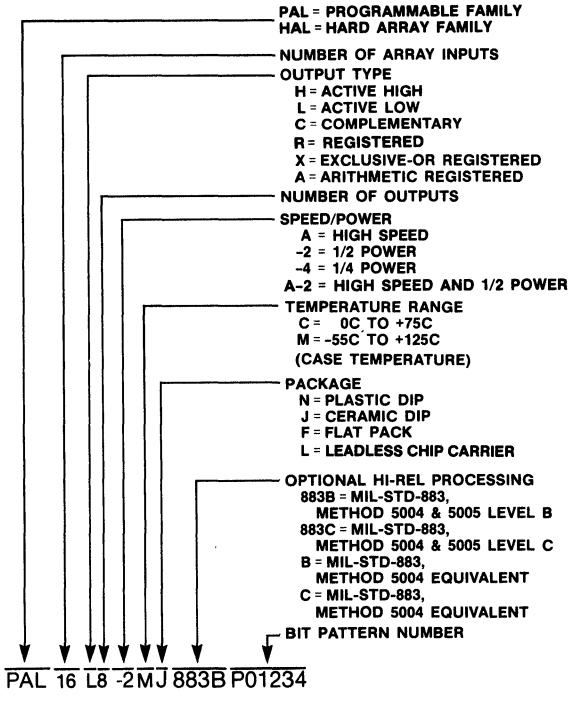

VCC
+5V
ROLL
2N4402
2N4402
.01μf
.01μf
2K
20K
20K
2K
D Q1
Q1
D Q2
Q2
D Q3
Q3
D Q4
Q4
D Q5
Q5
D Q6
Q6
D Q7
Q7
D Q8
Q8
120Ω
120Ω
290Ω
120Ω
120Ω
120Ω
290Ω
120Ω

Figure 16

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 17. 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.



PAL Logic Symbols

The logic symbols for each of the individual PAL devices gives a concise functional description of the PAL logic function. This symbol makes a convenient reference when selecting the PAL that best fits a specific application. Figure 18 shows the logic symbol for a PAL10H8 gate array.

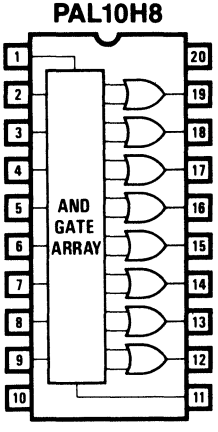


Figure 18

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 16. (A detailed logic derivation is shown in the PAL applications section of this handbook). 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 16 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 19.

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.

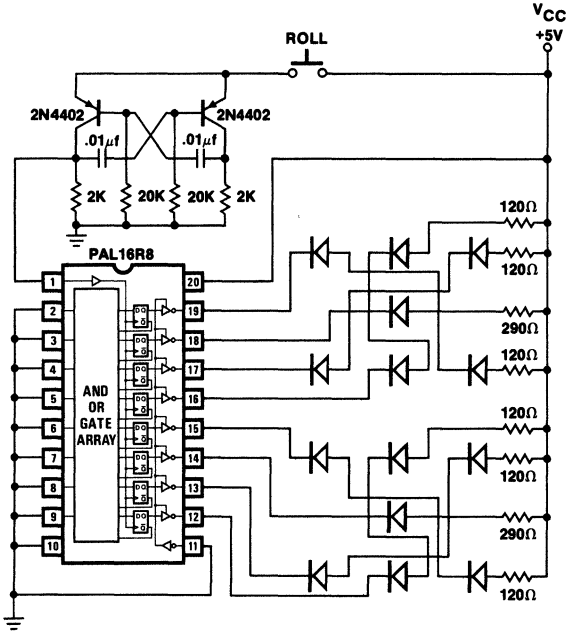
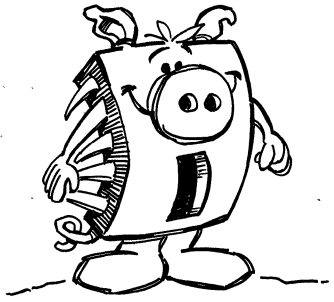


Figure 19

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 at least 4 to 1.
- Expedites and simplifies prototyping and board layout.
- Saves space with 20-pin and 24-pin Skinny DIP packages.
- High speed: 15ns 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 PAL units 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

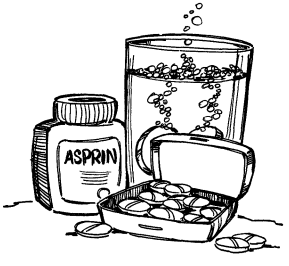
The PAL offers the systems logic designer a whole new world of options. Until now, the decision on logic system implementation was usually between SSI/MSI logic functions on one hand and microprocessors on the other. In many cases the function required is too awkward to implement the first way and too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

Space Efficiency

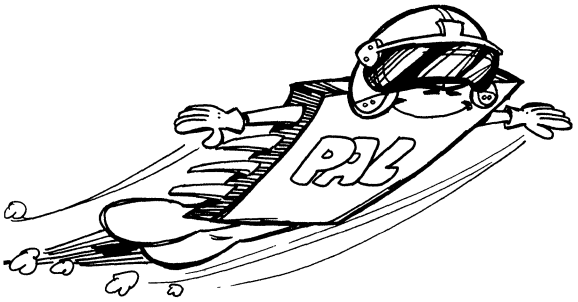
By allowing designers to replace many simple logic functions with single packages, the PAL allows more compact P.C. board layouts. The PAL space saving 20-pin and 24-pin "SKINNYDIP" helps to further reduce board area while simplifying board layout and fabrication. This means that many multi-card systems 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 29 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

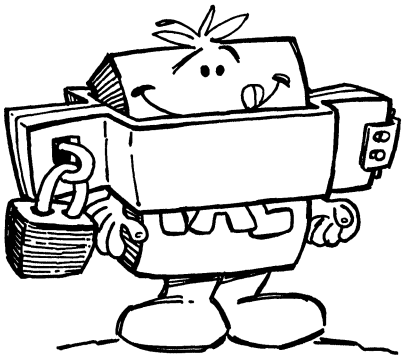


The PAL family runs faster or equal to the best of bipolar logic circuits. This makes the PAL the ideal choice for most logical operations or control sequence which requires a medium complexity and high speed. Also, in many microcomputer systems, the PAL can be used to handle high speed data interfaces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

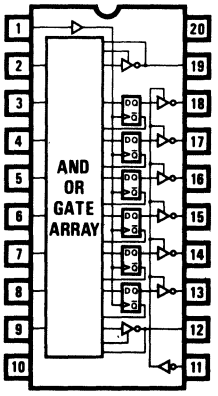


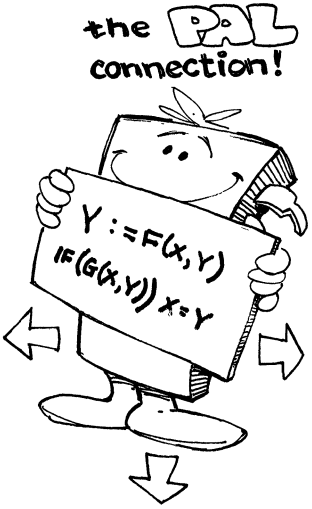
The PAL verification logic can be completely disabled by blowing out a special "last link." This prevents the unauthorized copying of valuable data, and makes the PAL perfect for use in any application where data integrity must be carefully guarded.

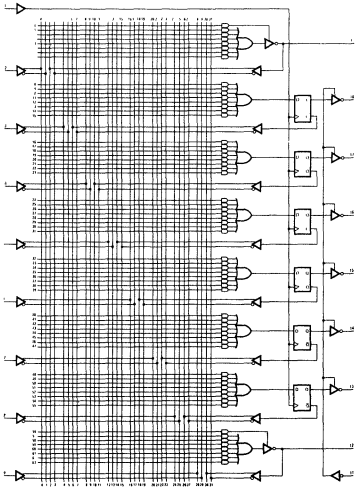
Summary

The 29 member PAL family of logic devices offer 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.

A Great Performer!

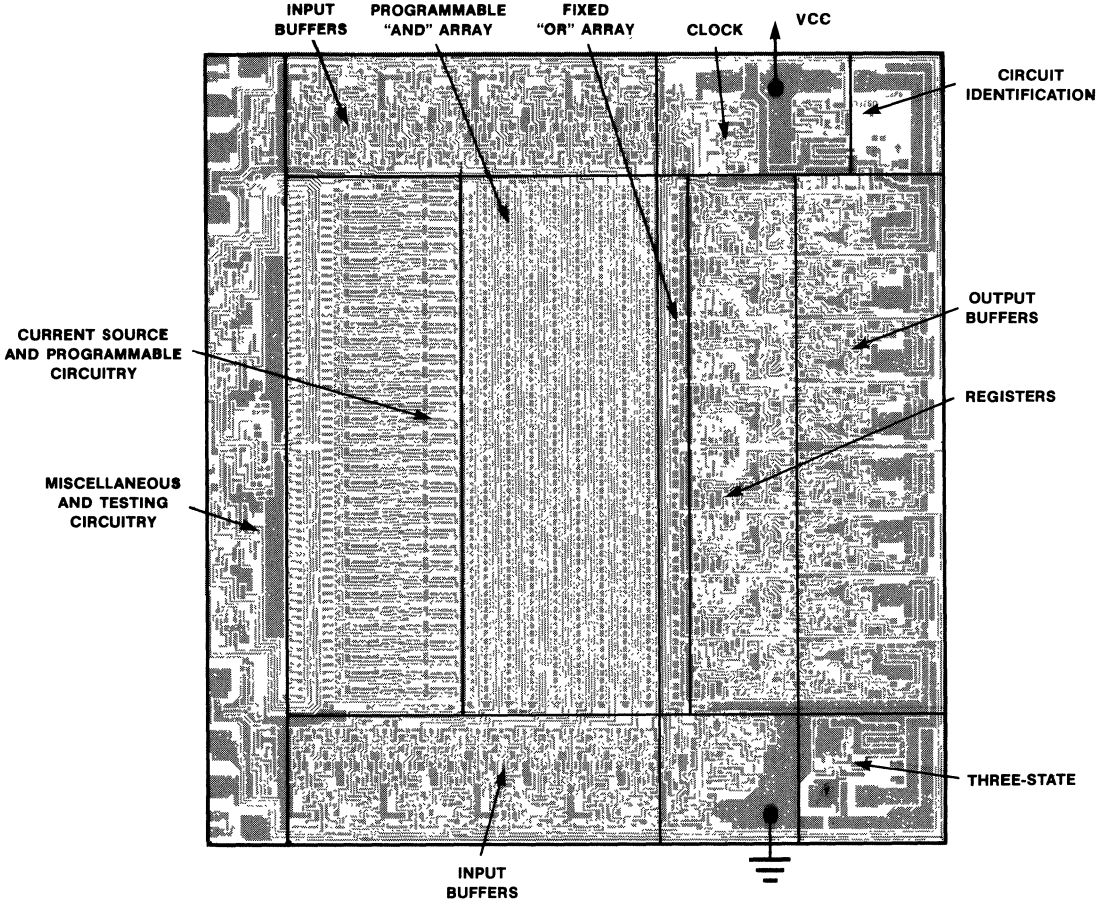






PAL16R6 Logic Symbols

PAL16R6 Logic Diagram



PAL16R6 Metalization

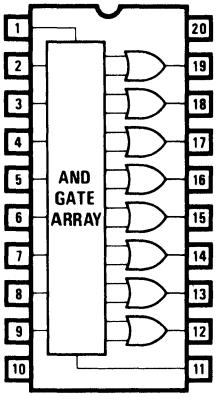
PAL

FAMILY PORTRAIT

of 29 - count 'em!

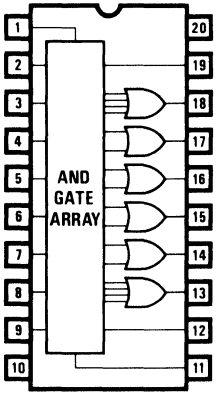
Logic Symbols

10H8



OCTAL 10 INPUT
AND-OR GATE ARRAY

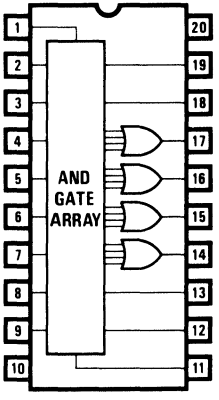
12H6



HEX 12 INPUT
AND-OR GATE ARRAY

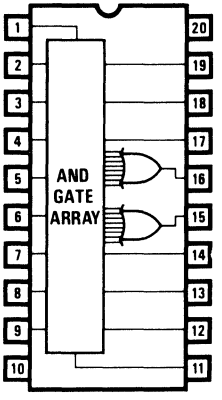
Active High

14H4

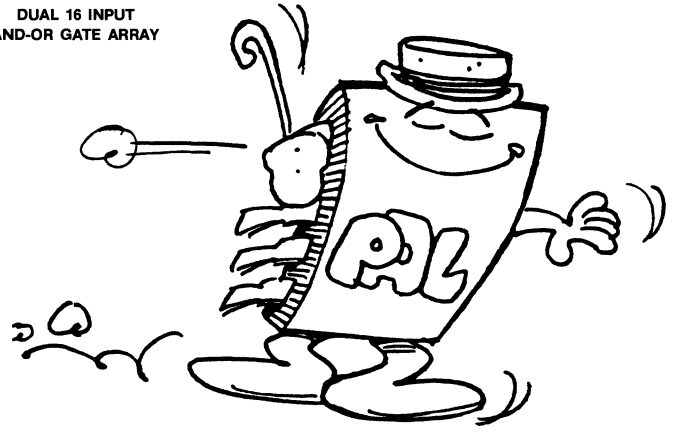


QUAD 14 INPUT
AND-OR GATE ARRAY

16H2

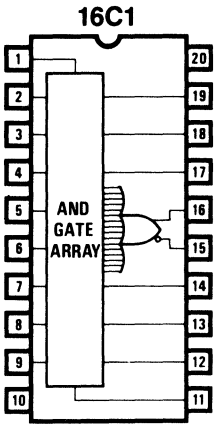


DUAL 16 INPUT
AND-OR GATE ARRAY

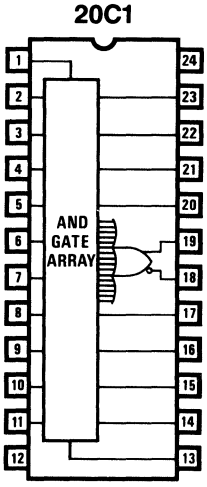


Logic Symbols

With Complementary Output

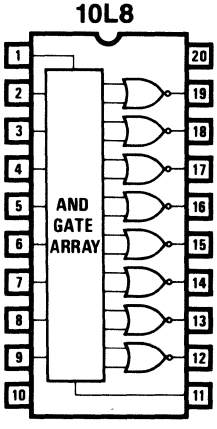


16 INPUT
AND-OR AND-OR-INVERT GATE ARRAY

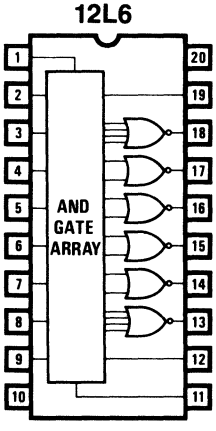


20 INPUT
AND-OR AND-OR-INVERT GATE ARRAY

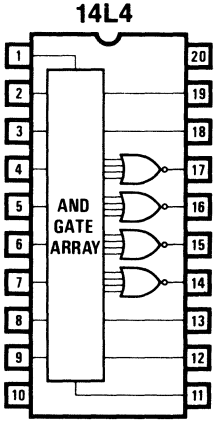
Active Low



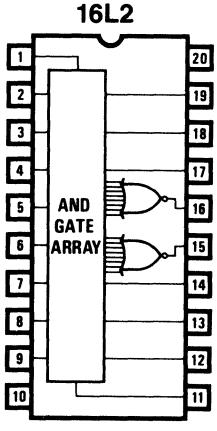
OCTAL 10 INPUT
AND-OR-INVERT GATE ARRAY



HEX 12 INPUT
AND-OR-INVERT GATE ARRAY



QUAD 14 INPUT
AND-OR-INVERT GATE ARRAY

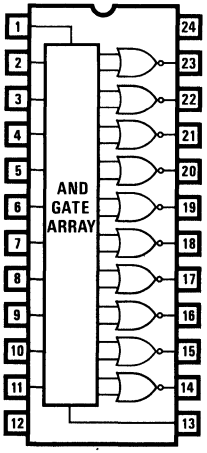


DUAL 16-INPUT
AND-OR-INVERT GATE ARRAY

Logic Symbols

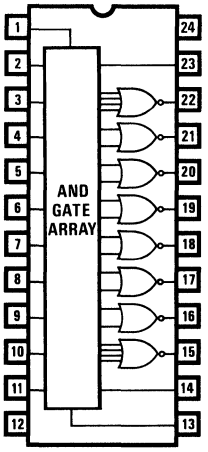
More Active Low PAL Arrays

12L10



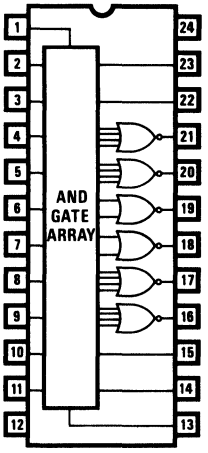
DECA 12 INPUT
AND-OR-INVERT GATE ARRAY

14L8



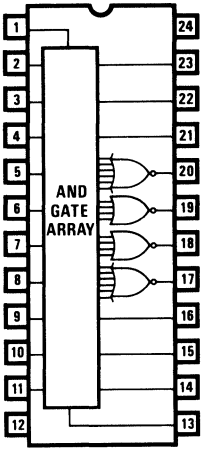
OCTAL 14 INPUT
AND-OR-INVERT GATE ARRAY

16L6



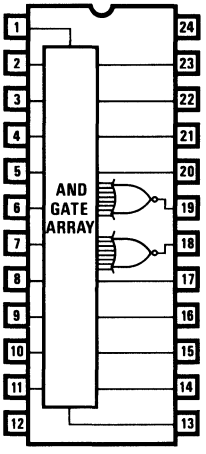
HEX 16 INPUT
AND-OR-INVERT GATE ARRAY

18L4

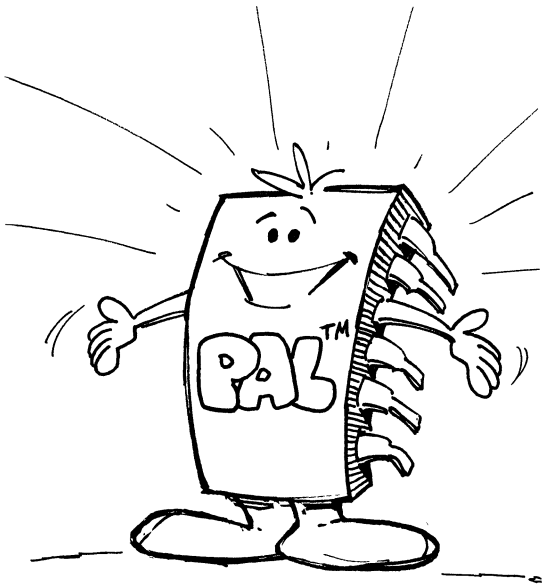


QUAD 18 INPUT
AND-OR-INVERT GATE ARRAY

20L2



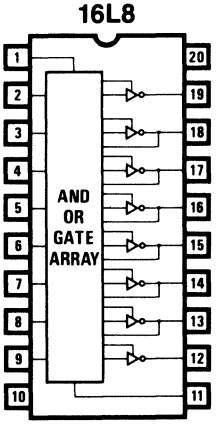
DUAL 20 INPUT
AND-OR-INVERT GATE ARRAY



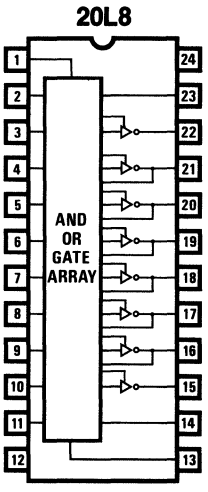
2

Logic Symbols

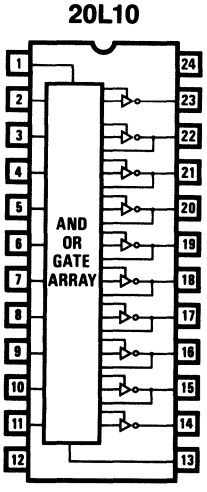
With Feedback



OCTAL 16 INPUT
AND-OR-INVERT GATE ARRAY



OCTAL 20 INPUT
AND-OR-INVERT GATE ARRAY

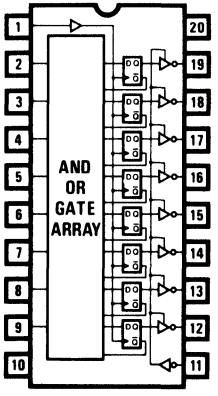


DECA 20 INPUT
AND-OR-INVERT GATE ARRAY

Logic Symbols

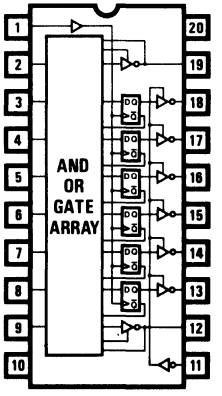
With Registered Outputs

16R8



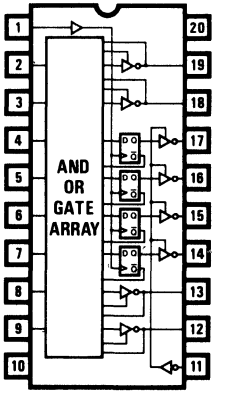
OCTAL 16 INPUT REGISTERED
AND-OR GATE ARRAY

16R6



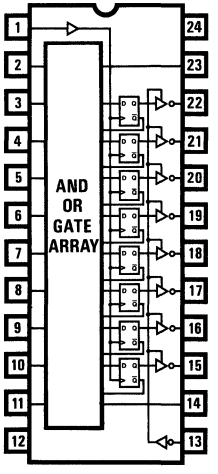
HEX 16 INPUT REGISTERED
AND-OR GATE ARRAY

16R4



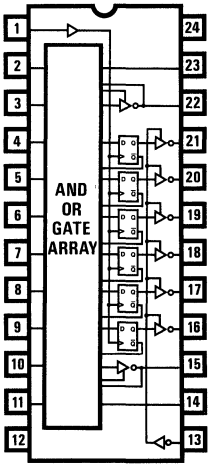
QUAD 16 INPUT REGISTERED
AND-OR ARRAY

20R8



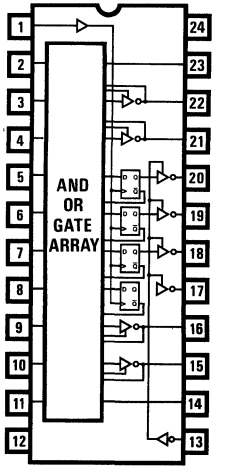
OCTAL 20 INPUT REGISTERED
AND-OR ARRAY

20R6



HEX 20 INPUT REGISTERED
AND-OR ARRAY

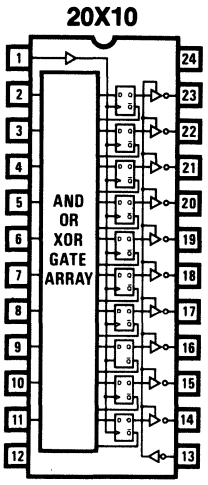
20R4



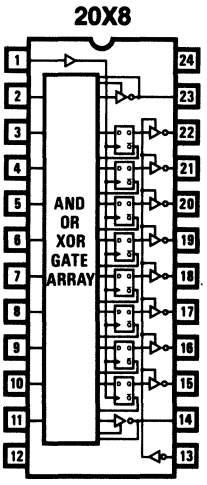
QUAD 20 INPUT REGISTERED
AND-OR ARRAY

Logic Symbols

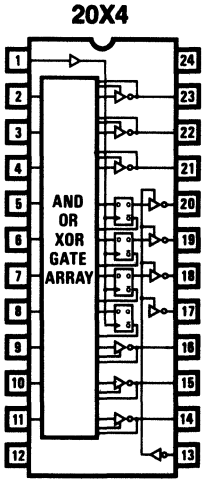
With Exclusive OR



DECA 20 INPUT REGISTERED
AND-OR-XOR GATE ARRAY

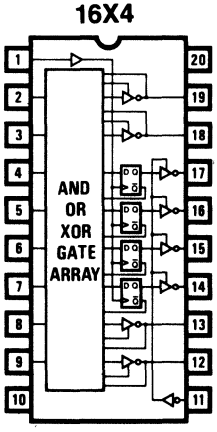


OCTAL 20 INPUT REGISTERED
AND-OR-XOR GATE ARRAY

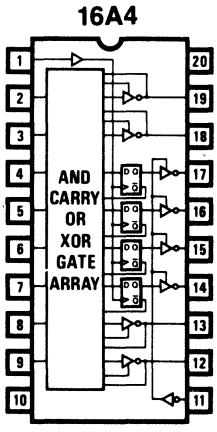


QUAD 20 INPUT REGISTERED
AND-OR-XOR GATE ARRAY

With Arithmetic Gated Feedback



QUAD 16 INPUT REGISTERED
AND-OR-XOR GATE ARRAY



QUAD 16 INPUT REGISTERED
AND-CARRY-OR-XOR GATE ARRAY

PAL®-Programmable Array Logic
HAL®-Hard Array Logic

Features/Benefits

- **Reduces SSI/MSI chip count greater than 5 to 1**
- **Saves space with SKINNYDIP® packages**
- **Reduces IC inventories substantially**
- **Expedites and simplifies prototyping and board layout**
- **PALASM™ silicon compiler provides auto routing and test vectors**
- **Security fuse 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 HAL family utilizes standard Low-Power Schottky TTL process and automated mask pattern generation directly from logic equations to provide a semi-custom gate array for replacing conventional SSI/MSI gates and flip-flops at reduced chip count.

There are four different speed/power families offered. Choose from either the standard, high speed, half power, or quarter power family to maximize design performance.

The PAL/Hal 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).

The HAL transfer function is the familiar sum of products. Like the ROM, the HAL has a single array of selectable gates. Unlike the ROM, the HAL is a selectable AND array driving a fixed OR array (the ROM is a fixed AND array driving a selectable OR array).

In addition the PAL/HAL provides these options:

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

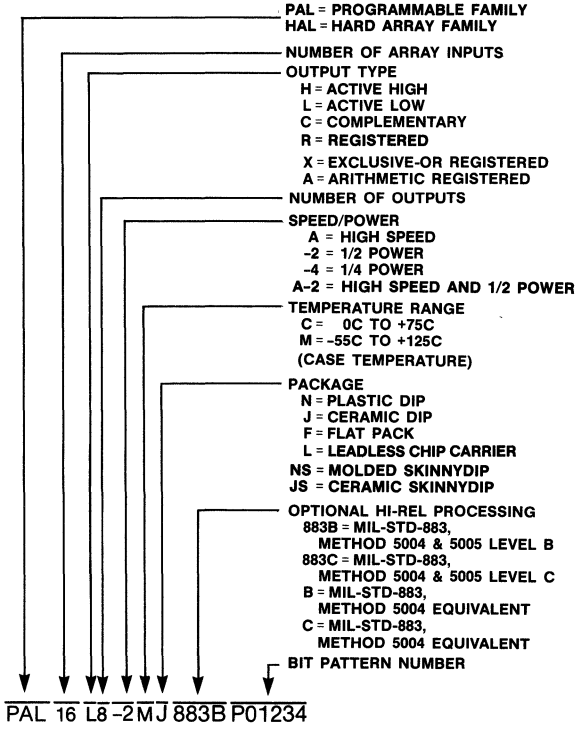
Unused inputs are tied directly to V_{CC} or GND. Product terms with all fuses blown assume the logical high state, and product terms connected to both true and complement of any single input assume the logical low state. Registers consist of D type flip-flops which are loaded on the low-to-high transition of the clock. PAL/HAL Logic Diagrams are shown with all fuses blown, enabling the designer to use the diagrams as coding sheets.

The entire PAL family is programmed using 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.

To design a HAL, the user first programs and debugs a PAL using PALASM and the "PAL DESIGN SPECIFICATION" standard format. This specification is submitted to Monolithic Memories where it is computer processed and assigned a bit pattern number, e.g., P01234.

Monolithic Memories will provide a PAL sample for customer qualification. The user then submits a purchase order for a HAL of the specified bit pattern number, e.g., HAL18L4 P01234. See Ordering Information below.

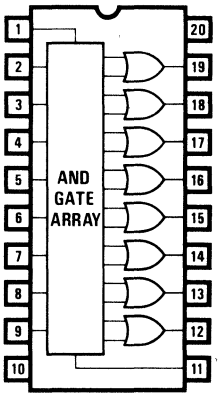
Ordering Information



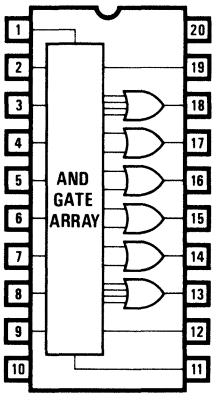
Monolithic Memories

GENERIC LOGIC	PINS	PACKAGE	DESCRIPTION	PART NUMBER			
				STANDARD	HIGH SPEED	1/2 POWER	1/4 POWER
10H8	20	N, J, F, L	Octal 10 Input And-Or Gate Array	PAL10H8 HAL10H8		PAL10H8-2 HAL10H8-2	
12H6	20	N, J, F, L	Hex 12 Input And-Or Gate Array	PAL12H6 HAL12H6		PAL12H6-2 HAL12H6-2	
14H4	20	N, J, F, L	Quad 14 Input And-Or Gate Array	PAL14H4 HAL14H4		PAL14H4-2 HAL14H4-2	
16H2	20	N, J, F, L	Dual 16 Input And-Or Gate Array	PAL16H2 HAL16H2		PAL16H2-2 HAL16H2-2	
16C1	20	N, J, F, L	16 Input And-Or/And-Or-Invert Gate Array	PAL16C1 HAL16C1		PAL16C1-2 PAL16C1-2	
10L8	20	N, J, F, L	Octal 10 Input And-Or-Invert Gate Array	PAL10L8 HAL10L8		PAL10L8-2 HAL10L8-2	
12L6	20	N, J, F, L	Hex 12 Input And-Or-Invert Gate Array	PAL12L6 HAL12L6		PAL12L6-2 HAL12L6-2	
14L4	20	N, J, F, L	Quad 14 Input And-Or-Invert Gate Array	PAL14L4 HAL14L4		PAL14L4-2 HAL14L4-2	
16L2	20	N, J, F, L	Dual 16 Input And-Or-Invert Gate Array	PAL16L2 HAL16L2		PAL16L2-2 HAL16L2-2	
16L8	20	N, J, F, L	Octal 16 Input And-Or-Invert Gate Array	PAL16L8 HAL16L8	PAL16L8A HAL16L8A	PAL16L8-2/A-2 HAL16L8-2/A-2	PAL16L8-4/A-4 HAL16L8-4/A-4
16R8	20	N, J, F, L	Octal 16 Input Registered And-Or Gate Array	PAL16R8 HAL16R8	PAL16R8A HAL16R8A	PAL16R8-2/A-2 HAL16R8-2/A-2	PAL16R8-4/A-4 HAL16R8-4/A-4
16R6	20	N, J, F, L	Hex 16 Input Registered And-Or Gate Array	PAL16R6 HAL16R6	PAL16R6A HAL16R6A	PAL16R6-2/A-2 HAL16R6-2/A-2	PAL16R6-6/A-4 HAL16R6-6/A-4
16R4	20	N, J, F, L	Quad 16 Input Registered And-Or Gate Array	PAL16R4 HAL16R4	PAL16R4A HAL16R4A	PAL16R4-2/A-2 HAL16R4-2/A-2	PAL16R4-4/A-4 HAL16R4-4/A-4
16X4	20	N, J	Quad 16 Input Registered And-Or-Xor Gate Array	PAL16X4 HAL16X4			
16A4	20	N, J	Quad 16 Input Registered And-Carry-Or-Xor Gate Array	PAL16A4 HAL16A4			
12L10	24	NS, JS, F, L	Deca 12 Input And-Or-Invert Gate Array	PAL12L10 HAL12L10			
14L8	24	NS, JS, F, L	Octal 14 Input And-Or-Invert Gate Array	PAL14L8 HAL14L8			
16L6	24	NS, JS, F, L	Hex 16 Input And-Or-Invert Gate Array	PAL16L6 HAL16L6			
18L4	24	NS, JS, F, L	Quad 18 Input And-Or-Invert Gate Array	PAL18L4 HAL18L4			
20L2	24	NS, JS, F, L	Dual 20 Input And-Or-Invert Gate Array	PAL20L2 HAL20L2			
20C1	24	NS, JS, F, L	20 Input And-Or/And-Or Invert Gate Array	PAL20C1 HAL20C1			
20L10	24	NS, JS, F, L	Deca 20 Input And-Or-Invert Gate Array	PAL20L10 HAL20L10			
20X10	24	NS, JS, F, L	Deca 20 Input Registered And-Or-Xor Gate Array	PAL20X10 HAL20X10			
20X8	24	NS, JS, F, L	Octal 20 Input Registered And-Or-Xor Gate Array	PAL20X8 HAL20X8			
20X4	24	NS, JS, F, L	Quad 20 Input Registered And-Or-Xor Gate Array	PAL20X4 HAL20X4			
20L8	24	NS, JS, F, L	Octal 20 Input And-Or-Invert Gate Array		PAL20L8A HAL20L8A		
20R8	24	NS, JS, F, L	Octal 20 Input Registered And-Or Gate Array		PAL20R8A HAL20R8A		
20R6	24	NS, JS, F, L	Hex 20 Input Registered And-Or Gate Array		PAL20R6A HAL20R6A		
20R4	24	NS, JS, F, L	Quad 20 Input Registered And-Or Gate Array		PAL20R4A HAL20R4A		

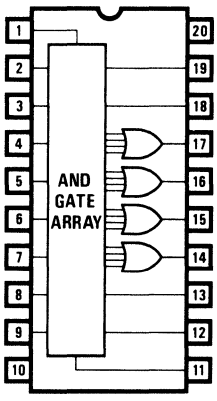
10H8



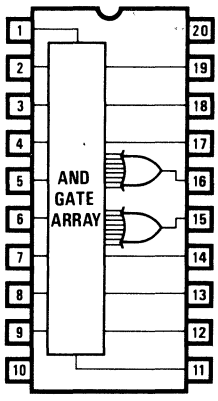
12H6



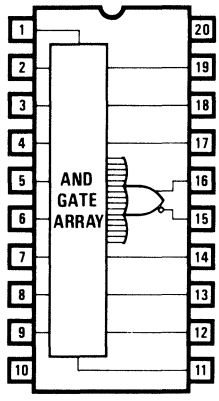
14H4



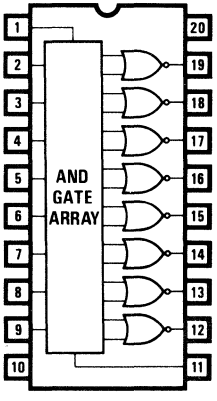
16H2



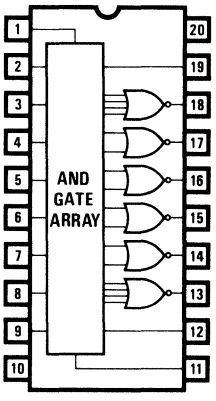
16C1



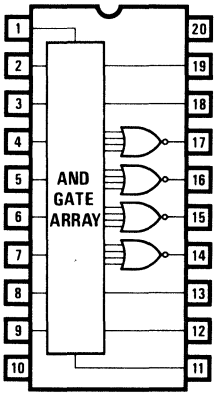
10L8



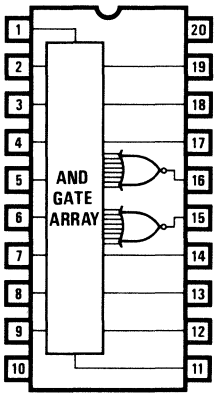
12L6



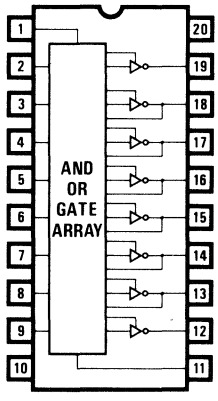
14L4



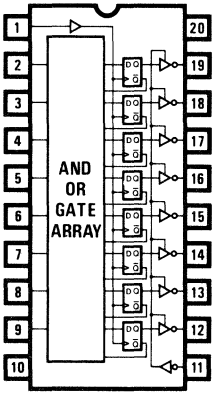
16L2



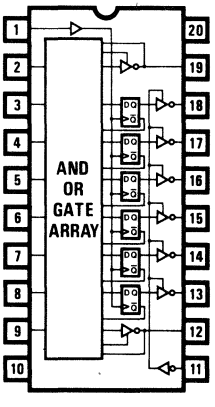
16L8



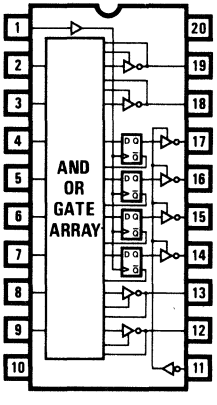
16R8



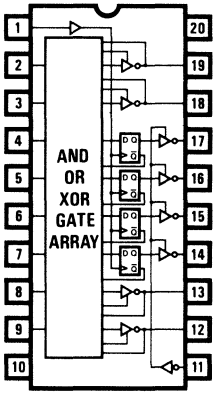
16R6



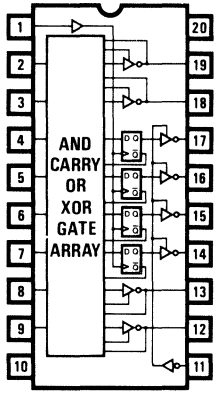
16R4



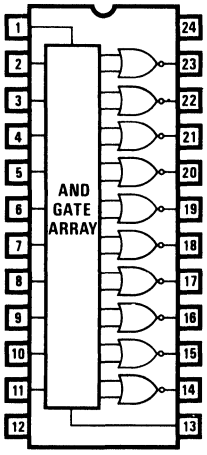
16X4



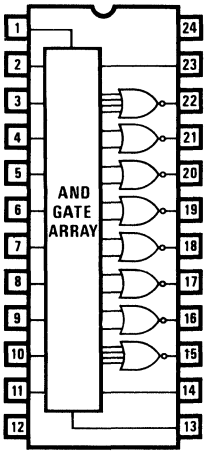
16A4



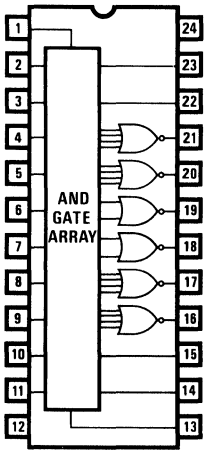
12L10



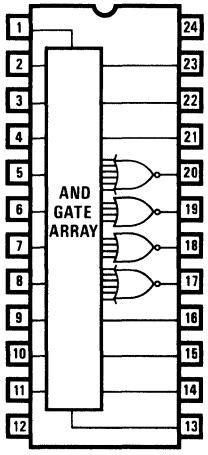
14L8



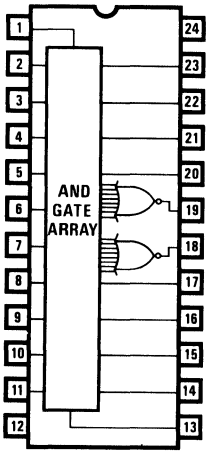
16L6



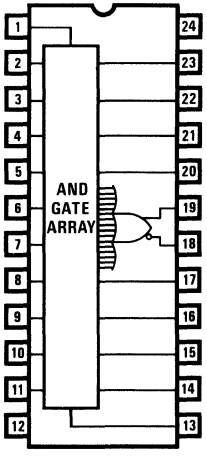
18L4



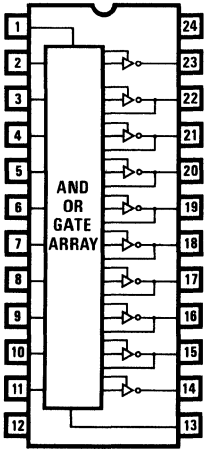
20L2



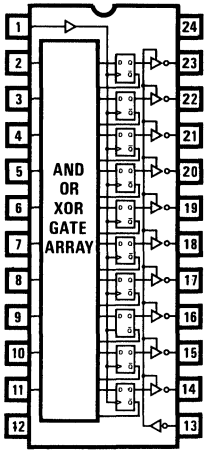
20C1



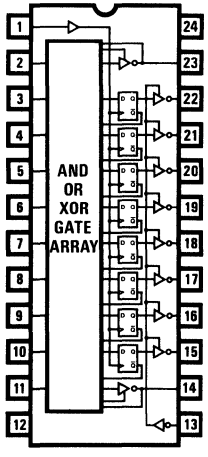
20L10



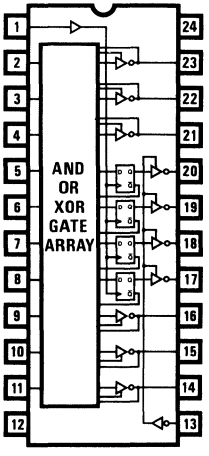
20X10



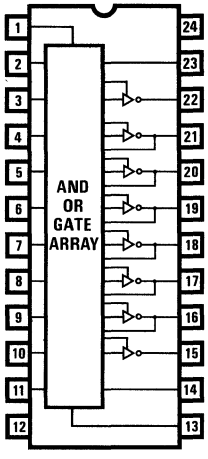
20X8



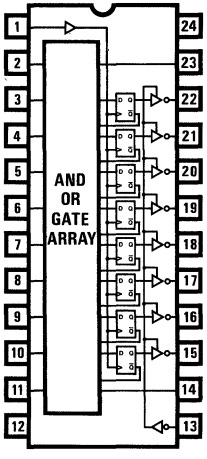
20X4



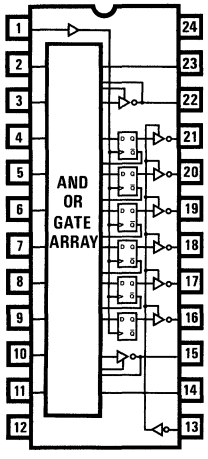
20L8



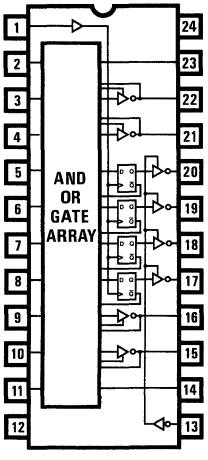
20R8



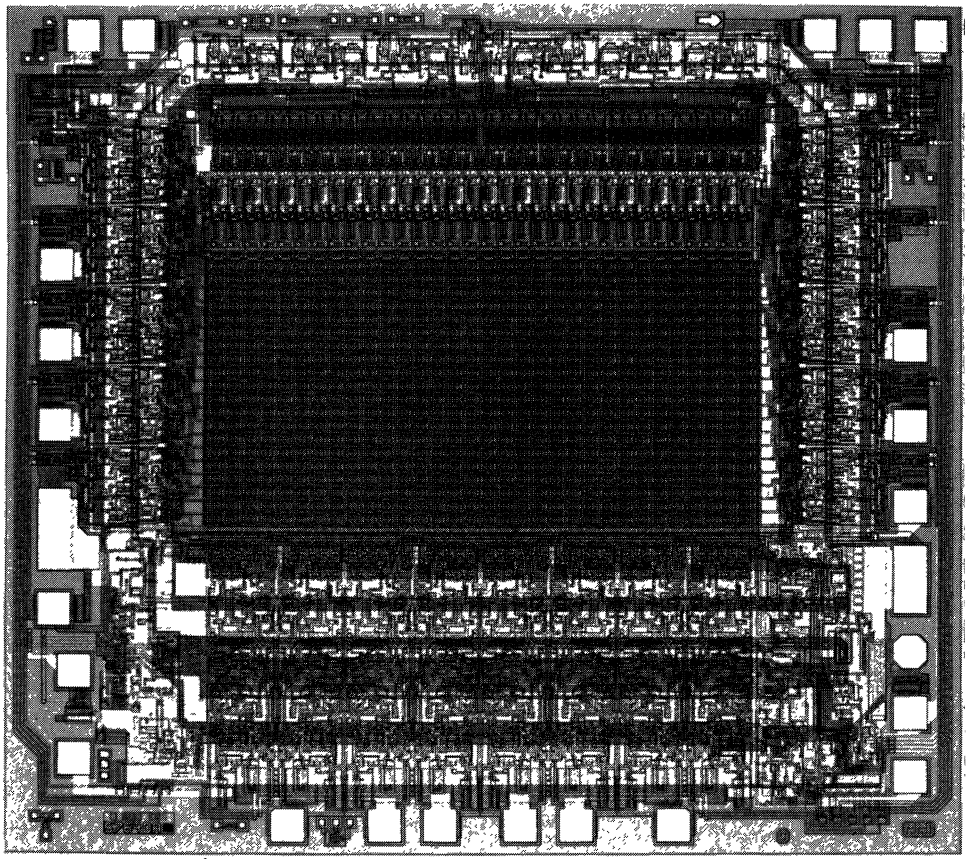
20R6



20R4



3

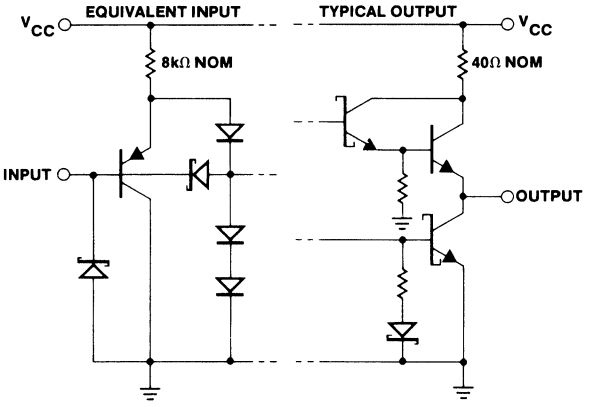


Absolute Maximum Ratings

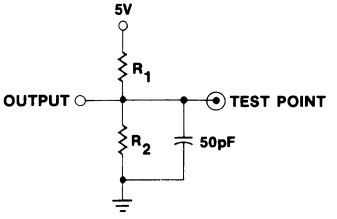
	Operating	Programming
Supply Voltage, V_{CC}	-0.5 to 7.0V	-0.5 to 12.0V
Input Voltage	-1.5 to 5.5V	-1.0 to 22V ⊕
Off-state output Voltage	5.5V	12.0V
Storage temperature		-65° to +150°C

⊕ Pins 1 and 11 may be raised to 20V

Schematic of Inputs and Outputs



Test Load



Other loads may be used.

3

Typical notes for all the following specifications (pages 3-8 — 3-18)

Notes: Apply to electrical and switching characteristics

† I/O pin leakage is the worst case of I_{OZX} or I_{IX} e.g., I_{IL} and I_{OZH}.

* These are absolute voltages with respect to the ground pin on the device and includes all overshoots due to system and/or tester noise. Do not attempt to test these values without suitable equipment.

* * Only one output shorted at a time.

Operating Conditions

SYMBOL	PARAMETER	MILITARY			COMMERCIAL			UNIT
		MIN	TYP	MAX	MIN	TYP	MAX	
V_{CC}	Supply voltage	4.5	5	5.5	4.75	5	5.25	V
T_A	Operating free-air temperature	−55			0		75	°C
T_C	Operating case temperature			125				°C

Electrical Characteristics Over Operating Conditions

SYMBOL	PARAMETER	TEST CONDITIONS			MIN	TYP	MAX	UNIT
V_{IL}*	Low-level input voltage						0.8	V
V_{IH}*	High-level input voltage				2			V
V_{IC}	Input clamp voltage	V_{CC} = MIN	I_I = −18mA			−0.8	−1.5	V
I_{IL}	Low-level input current †	V_{CC} = MAX	V_I = 0.4V			−0.02	−0.25	mA
I_{IH}	High-level input current †	V_{CC} = MAX	V_I = 2.4V				25	μA
I_I	Maximum input current	V_{CC} = MAX	V_I = 5.5V				1	mA
V_{OL}	Low-level output voltage	V_{CC} = MIN	MIL	I_{OL} = 8mA		0.3	0.5	V
			COM	I_{OL} = 8mA				
V_{OH}	High-level output voltage	V_{CC} = MIN	MIL	I_{OH} = −2mA	2.4	2.8		V
			COM	I_{OH} = −3.2mA				
I_{OS}	Output short-circuit current**	V_{CC} = 5V		V_O = 0V	−30	−70	−130	mA
I_{CC}	Supply current	V_{CC} = MAX				55	90	mA

Switching Characteristics Over Operating Conditions

SYMBOL	PARAMETER		TEST CONDITIONS	MILITARY			COMMERCIAL			UNIT
				MIN	TYP	MAX	MIN	TYP	MAX	
t_{PD}	Input or feedback to output	Except 16C1	R1 = 560Ω, R2 = 1.1kΩ		25	45		25	35	ns
		16C1			25	45		25	40	

Operating Conditions

SYMBOL	PARAMETER	MILITARY			COMMERCIAL			UNIT
		MIN	TYP	MAX	MIN	TYP	MAX	
V_{CC}	Supply voltage	4.5	5	5.5	4.75	5	5.25	V
T_A	Operating free-air temperature	−55			0		75	°C
T_C	Operating case temperature			125				°C

Electrical Characteristics Over Operating Conditions

SYMBOL	PARAMETER	TEST CONDITIONS			MIN	TYP	MAX	UNIT
V_{IL}*	Low-level input voltage						0.8	V
V_{IH}*	High-level input voltage				2			V
V_{IC}	Input clamp voltage	V_{CC} = MIN		I_I = −18mA		−0.8	−1.5	V
I_{IL}	Low-level input current †	V_{CC} = MAX		V_I = 0.4V		−0.02	−0.25	mA
I_{IH}	High-level input current †	V_{CC} = MAX		V_I = 2.4V			25	μA
I_I	Maximum input current	V_{CC} = MAX		V_I = 5.5V			1	mA
V_{OL}	Low-level output voltage	V_{CC} = MIN	MIL	I_{OL} = 8mA		0.3	0.5	V
			COM	I_{OL} = 8mA				
V_{OH}	High-level output voltage	V_{CC} = MIN	MIL	I_{OH} = −2mA	2.4	2.8		V
			COM	I_{OH} = −3.2mA				
I_{OS}	Output short-circuit current**	V_{CC} = 5V		V_O = 0V	−30	−70	−130	mA
I_{CC}	Supply current	V_{CC} = MAX				60	100	mA

Switching Characteristics Over Operating Conditions

SYMBOL	PARAMETER	TEST CONDITIONS	MILITARY			COMMERCIAL			UNIT
			MIN	TYP	MAX	MIN	TYP	MAX	
t_{PD}	Input or feedback to output	R1 = 560Ω R2 = 1.1kΩ		25	45		25	40	ns

Operating Conditions

SYMBOL	PARAMETER		MILITARY MIN	MILITARY TYP	MILITARY MAX	COMMERCIAL MIN	COMMERCIAL TYP	COMMERCIAL MAX	UNIT
V_{CC}	Supply voltage		4.5	5	5.5	4.75	5	5.25	V
t_w	Width of clock	Low	25	10		25	10		ns
		High	25	10		25	10		
t_{su}	Set up time from input or feedback to clock	16R8 16R6 16R4	45	25		35	25		ns
		16X4 16A4	55	30		45	30		
t_h	Hold time		0	−15		0	−15		ns
T_A	Operating free-air temperature		−55			0		75	°C
T_C	Operating case temperature				125				°C

Electrical Characteristics Over Operating Conditions

SYMBOL	PARAMETER	TEST CONDITIONS			MIN	TYP	MAX	UNIT
V_{IL}*	Low-level input voltage						0 8	V
V_{IH}*	High-level input voltage				2			V
V_{IC}	Input clamp voltage	V_{CC} = MIN	I_I = −18mA			−0.8	−1.5	V
I_{IL}	Low-level input current †	V_{CC} = MAX	V_I = 0 4V			−0.02	−0.25	mA
I_{IH}	High-level input current †	V_{CC} = MAX	V_I = 2 4V				25	μA
I_I	Maximum input current	V_{CC} = MAX	V_I = 5.5V				1	mA
V_{OL}	Low-level output voltage	V_{CC} = MIN	MIL	I_{OL} = 12mA		0.3	0.5	V
			COM	I_{OL} = 24mA				
V_{OH}	High-level output voltage	V_{CC} = MIN	MIL	I_{OH} = −2mA	2.4	2.8		V
			COM	I_{OH} = −3.2mA				
I_{OZL}	Off-state output current †	V_{CC} = MAX		V_O = 0.4V			−100	μA
I_{OZH}				V_O = 2.4V			100	μA
I_{OS}	Output short-circuit current**	V_{CC} = 5V		V_O = 0V	−30	−70	−130	mA
I_{CC}	Supply current	V_{CC} = MAX	16R4 16R6 16R8 16L8			120	180	mA
			16X4			160	225	
			16A4			170	240	

Switching Characteristics Over Operating Conditions

SYMBOL	PARAMETER		TEST CONDITIONS	MILITARY MIN	MILITARY TYP	MILITARY MAX	COMMERCIAL MIN	COMMERCIAL TYP	COMMERCIAL MAX	UNIT
t_{PD}	Input or feedback to output	16R6 16R4 16L8	R_1 = 200Ω, R_2 = 390Ω		25	45		25	35	ns
		16X4 16R4			30	45		30	40	ns
t_{CLK}	Clock to output or feedback				15	25		15	25	ns
t_{PZX}	Pin 11 to output enable except 16L8				15	25		15	25	ns
t_{PXZ}	Pin 11 to output disable except 16L8				15	25		15	25	ns
t_{PZX}	Input to output enable	16R6 16R4 16L8			25	45		25	35	ns
		16X4 16A4			30	45		30	40	ns
t_{PXZ}	Input to output disable	16R6 16R4 16L8			25	45		25	35	ns
		16X4 16A4			30	45		30	40	ns
f_{MAX}	Maximum frequency	16R8 16R6 16R4		14	25		16	25		MHz
		16X4 16A4		12	22		14	22		

Operating Conditions

SYMBOL	PARAMETER		MILITARY MIN	MILITARY TYP	MILITARY MAX	COMMERCIAL MIN	COMMERCIAL TYP	COMMERCIAL MAX	UNIT
V_{CC}	Supply voltage		4.5	5	5.5	4.75	5	5.25	V
t_w	Width of clock	Low	40	20		35	20		ns
		High	30	10		25	10		
t_{su}	Set up time from input or feedback to clock		60	38		50	38		ns
t_h	Hold time		0	−15		0	−15		ns
T_A	Operating free-air temperature		−55			0		75	°C
T_C	Operating case temperature				125				°C

Electrical Characteristics Over Operating Conditions

SYMBOL	PARAMETER	TEST CONDITIONS			MIN	TYP	MAX	UNIT
V_{IL}*	Low-level input voltage						0.8	V
V_{IH}*	High-level input voltage				2			V
V_{IC}	Input clamp voltage	V_{CC} = MIN	I_I = −18mA			−0.8	−1.5	V
I_{IL}	Low-level input current †	V_{CC} = MAX	V_I = 0.4V			−0.02	−0.25	mA
I_{IH}	High-level input current †	V_{CC} = MAX	V_I = 2.4V				25	μA
I_I	Maximum input current	V_{CC} = MAX	V_I = 5.5V				1	mA
V_{OL}	Low-level output voltage	V_{CC} = MIN	MIL	I_{OL} = 12mA		0.3	0.5	V
			COM	I_{OL} = 24mA				
V_{OH}	High-level output voltage	V_{CC} = MIN	MIL	I_{OH} = −2mA	2.4	2.8		V
			COM	I_{OH} = −3.2mA				
I_{OZL}	Off-state output current †	V_{CC} = MAX		V_O = 0.4V			−100	μA
I_{OZH}				V_O = 2.4V			100	μA
I_{OS}	Output short-circuit current**	V_{CC} = 5V		V_O = 0V	−30	−70	−130	mA
I_{CC}	Supply current	V_{CC} = MAX	20X10	20X8 20X4		120	180	mA
I_{CC}	Supply current	V_{CC} = MAX	20L10			90	165	mA

Switching Characteristics Over Operating Conditions

SYMBOL	PARAMETER	TEST CONDITIONS	MILITARY MIN	MILITARY TYP	MILITARY MAX	COMMERCIAL MIN	COMMERCIAL TYP	COMMERCIAL MAX	UNIT
t_{PD}	Input or feedback to output	R_1 = 200Ω, R_2 = 390Ω		35	60		35	50	ns
t_{CLK}	Clock to output or feedback			20	35		20	30	ns
$t_{PXZ/ZX}$	Pin 13 to output disable/enable except 20L10			20	45		20	35	ns
t_{PZX}	Input to output enable except 20X10			35	55		35	45	ns
t_{PXZ}	Input to output disable except 20X10			35	55		35	45	ns
f_{MAX}	Maximum frequency		10.5	16		12.5	16		MHz

3

Operating Conditions

SYMBOL	PARAMETER		MILITARY MIN	MILITARY TYP	MILITARY MAX	COMMERCIAL MIN	COMMERCIAL TYP	COMMERCIAL MAX	UNIT
V_{CC}	Supply voltage		4.5	5	5.5	4.75	5	5.25	V
t_w	Width of clock	Low	20	10		15	10		ns
		High	20	10		15	10		
t_{su}	Set up time from input or feedback to clock	16R8A 16R6A 16R4A	30	15		25†	15		ns
t_h	Hold time		0	-10		0	-10		ns
T_A	Operating free-air temperature		-55			0		75	°C
T_C	Operating case temperature				125				°C

†Can select 20ns upon customer request.

Electrical Characteristics Over Operating Conditions

SYMBOL	PARAMETER	TEST CONDITIONS			MIN	TYP	MAX	UNIT
V_{IL}*	Low-level input voltage						0.8	V
V_{IH}*	High-level input voltage				2			V
V_{IC}	Input clamp voltage	V_{CC} = MIN		I_I = -18mA		-0.8	-1.5	V
I_{IL}	Low-level input current †	V_{CC} = MAX		V_I = 0.4V		-0.02	-0.25	mA
I_{IH}	High-level input current †	V_{CC} = MAX		V_I = 2.4V			25	μA
I_I	Maximum input current	V_{CC} = MAX		V_I = 5.5V			1	mA
V_{OL}	Low-level output voltage	V_{CC} = MIN	MIL	I_{OL} = 12mA		0.3	0.5	V
			COM	I_{OL} = 24mA				
V_{OH}	High-level output voltage	V_{CC} = MIN	MIL	I_{OH} = -2mA	2.4	2.8		V
			COM	I_{OH} = -3.2mA				
I_{OZL}	Off-state output current †	V_{CC} = MAX		V_O = 0.4V			-100	μA
I_{OZH}				V_O = 2.4V			100	μA
I_{OS}	Output short-circuit current**	V_{CC} = 5V		V_O = 0V	-30	-70	-130	mA
I_{CC}	Supply current	V_{CC} = MAX				120	180	mA

Switching Characteristics Over Operating Conditions

SYMBOL	PARAMETER		TEST CONDITIONS	MILITARY MIN	MILITARY TYP	MILITARY MAX	COMMERCIAL MIN	COMMERCIAL TYP	COMMERCIAL MAX	UNIT
t_{PD}	Input or feedback to output	16R6A 16R4A 16L8A	R_1 = 200Ω, R_2 = 390Ω		15	30		15	25	ns
t_{CLK}	Clock to output or feedback				10	20		10	15	ns
t_{PZX}	Pin 11 to output enable except 16L8A				10	25		10	20	ns
t_{PXZ}	Pin 11 to output disable except 16L8A				11	25		11	20	ns
t_{PZX}	Input to output enable	16R6A 16R4A 16L8A			10	30		10	25	ns
t_{PXZ}	Input to output disable	16R6A 16R4A 16L8A			13	30		13	25	ns
f_{MAX}	Maximum frequency	16R8A 16R6A 16R4A		20	40		28.5	40		MHz

Operating Conditions

SYMBOL	PARAMETER		MILITARY MIN	MILITARY TYP	MILITARY MAX	COMMERCIAL MIN	COMMERCIAL TYP	COMMERCIAL MAX	UNIT
V_{CC}	Supply voltage		4.5	5	5.5	4.75	5	5.25	V
t_w	Width of clock	Low	20	7		15	7		ns
		High	20	7		15	7		
t_{su}	Set up time from input or feedback to clock	20R8A 20R6A 20R4A	30	15		25	15		ns
t_h	Hold time		0	−10		0	−10		ns
T_A	Operating free-air temperature		−55			0		75	°C
T_C	Operating case temperature				125				°C

Electrical Characteristics Over Operating Conditions

SYMBOL	PARAMETER	TEST CONDITIONS			MIN	TYP	MAX	UNIT
V_{IL}*	Low-level input voltage						0.8	V
V_{IH}*	High-level input voltage				2			V
V_{IC}	Input clamp voltage	V_{CC} = MIN	I_I = −18mA			−0.8	−1.5	V
I_{IL}	Low-level input current †	V_{CC} = MAX	V_I = 0.4V			−0.02	−0.25	mA
I_{IH}	High-level input current †	V_{CC} = MAX	V_I = 2.4V				25	μA
I_I	Maximum input current	V_{CC} = MAX	V_I = 5.5V				1	mA
V_{OL}	Low-level output voltage	V_{CC} = MIN	MIL	I_{OL} = 12mA		0.3	0.5	V
			COM	I_{OL} = 24mA				
V_{OH}	High-level output voltage	V_{CC} = MIN	MIL	I_{OH} = −2mA	2.4	2.8		V
			COM	I_{OH} = −3.2mA				
I_{OZL}	Off-state output current †	V_{CC} = MAX		V_O = 0.4V			−100	μA
I_{OZH}				V_O = 2.4V			100	μA
I_{OS}	Output short-circuit current**	V_{CC} = 5V		V_O = 0V	−30	−90	−130	mA
I_{CC}	Supply current	V_{CC} = MAX				160	210	mA

Switching Characteristics Over Operating Conditions

SYMBOL	PARAMETER		TEST CONDITIONS	MILITARY MIN	MILITARY TYP	MILITARY MAX	COMMERCIAL MIN	COMMERCIAL TYP	COMMERCIAL MAX	UNIT
t_{PD}	Input or feedback to output	20R6A 20R4A 20L8A	R_1 = 200Ω, R_2 = 390Ω		15	30		15	25	ns
t_{CLK}	Clock to output or feedback				10	20		10	15	ns
t_{PZX}	Pin 13 to output enable except 20L8A				10	25		10	20	ns
t_{PXZ}	Pin 13 to output disable except 20L8A				11	25		11	20	ns
t_{PZX}	Input to output enable	20R6A 20R4A 20L8A			10	30		10	25	ns
t_{PXZ}	Input to output disable	20R6A 20R4A 20L8A			13	30		13	25	ns
f_{MAX}	Maximum frequency	20R8A 20R6A 20R4A		20	40		28.5	40		MHz

Operating Conditions

SYMBOL	PARAMETER	MILITARY			COMMERCIAL			UNIT
		MIN	TYP	MAX	MIN	TYP	MAX	
V_{CC}	Supply voltage	4.5	5	5.5	4.75	5	5.25	V
T_A	Operating free-air temperature	–55		125	0		75	°C

Electrical Characteristics Over Operating Conditions

SYMBOL	PARAMETER	TEST CONDITIONS			MIN	TYP	MAX	UNIT
V_{IL}*	Low-level input voltage						0.8	V
V_{IH}*	High-level input voltage				2			V
V_{IC}	Input clamp voltage	V_{CC} = MIN	I_I = –18mA			–0.8	–1 5	V
I_{IL}	Low-level input current †	V_{CC} = MAX	V_I = 0.4V			-0.02	–0.25	mA
I_{IH}	High-level input current †	V_{CC} = MAX	V_I = 2 4V				25	μA
I_I	Maximum input current	V_{CC} = MAX	V_I = 5.5V				1	mA
V_{OL}	Low-level output voltage	V_{CC} = MIN	MIL	I_{OL} = 4mA		0.3	0 5	V
			COM	I_{OL} = 4mA				
V_{OH}	High-level output voltage	V_{CC} = MIN	MIL	I_{OH} = –1mA	2.4	2.8		V
			COM	I_{OH} = –1mA				
I_{OS}	Output short-circuit current**	V_{CC} = 5V		V_O = 0V	–30	–70	–130	mA
I_{CC}	Supply current	V_{CC} = MAX				30	45	mA

Switching Characteristics Over Operating Conditions

SYMBOL	PARAMETER	TEST	MILITARY			COMMERCIAL			UNIT
			MIN	TYP	MAX	MIN	TYP	MAX	
t_{PD}	Input or feedback to output	R1 = 1.12kΩ R2 = 2.2kΩ		45	80		45	60	ns

Operating Conditions

SYMBOL	PARAMETER		MILITARY MIN	MILITARY TYP	MILITARY MAX	COMMERCIAL MIN	COMMERCIAL TYP	COMMERCIAL MAX	UNIT
V_{CC}	Supply voltage		4.5	5	5.5	4.75	5	5.25	V
t_w	Width of clock	Low	50	25		40	25		ns
		High	50	25		40	25		
t_{su}	Set up time from input or feedback to clock	16R8-2, 16R6-2 16R4-2	70	40		55	40		ns
t_h	Hold time		0	−15		0	−15		ns
T_A	Operating free-air temperature		−55		125	0		75	°C

Electrical Characteristics Over Operating Conditions

SYMBOL	PARAMETER	TEST CONDITIONS			MIN	TYP	MAX	UNIT
V_{IL}*	Low-level input voltage						0.8	V
V_{IH}*	High-level input voltage				2			V
V_{IC}	Input clamp voltage	V_{CC} = MIN	I_I = −18mA			−0.8	−1.5	V
I_{IL}	Low-level input current †	V_{CC} = MAX	V_I = 0.4V			−0.02	−0.25	mA
I_{IH}	High-level input current †	V_{CC} = MAX	V_I = 2.4V				25	μA
I_I	Maximum input current	V_{CC} = MAX	V_I = 5.5V				1	mA
V_{OL}	Low-level output voltage	V_{CC} = MIN	MIL	I_{OL} = 4mA		0.3	0.5	V
			COM	I_{OL} = 8mA				
V_{OH}	High-level output voltage	V_{CC} = MIN	MIL	I_{OH} = −1mA	2.4	2.8		V
			COM	I_{OH} = −1mA				
I_{OZL}	Off-state output current †	V_{CC} = MAX		V_O = 0.4V			−100	μA
I_{OZH}				V_O = 2.4V			100	μA
I_{OS}	Output short-circuit current**	V_{CC} = 5V		V_O = 0V	−30	−70	−130	mA
I_{CC}	Supply current	V_{CC} = MAX				60	90	mA

Switching Characteristics Over Operating Conditions

SYMBOL	PARAMETER		TEST CONDITIONS	MILITARY MIN	MILITARY TYP	MILITARY MAX	COMMERCIAL MIN	COMMERCIAL TYP	COMMERCIAL MAX	UNIT
t_{PD}	Input or feedback to output	16R6-2 16R4-2 16L8-2	R_1 = 400Ω, R_2 = 780Ω		30	65		30	50	ns
t_{CLK}	Clock to output or feedback				20	50		20	40	ns
$t_{PXZ/ZX}$	Pin 11 to output disable/enable				15	40		15	30	ns
t_{PZX}	Input to output enable	16R6-2 16R4-2 16L8-2			30	65		30	50	ns
t_{PXZ}	Input to output disable	16R6-2 16R4-2 16L8-2			30	65		30	50	ns
f_{MAX}	Maximum frequency	16R8-2 16R6-2 16R4-2		8	20		11	20		MHz

Operating Conditions

SYMBOL	PARAMETER		MILITARY MIN	MILITARY TYP	MILITARY MAX	COMMERCIAL MIN	COMMERCIAL TYP	COMMERCIAL MAX	UNIT
V_{CC}	Supply voltage		4.5	5	5.5	4.75	5	5.25	V
t_w	Width of clock	Low	70	25		50	25		ns
		High	70	25		50	25		
t_{su}	Set up time from input or feedback to clock	16R8-4 16R6-4 16R4-4	120	45		100	45		ns
t_h	Hold time		0	–15		0	–15		ns
T_A	Operating free-air temperature		–55		125	0		75	°C

Electrical Characteristics Over Operating Conditions

SYMBOL	PARAMETER	TEST CONDITIONS			MIN	TYP	MAX	UNIT
V_{IL}*	Low-level input voltage						0.8	V
V_{IH}*	High-level input voltage				2			V
V_{IC}	Input clamp voltage	V_{CC} = MIN	I_I = –18mA			–0.8	–1.5	V
I_{IL}	Low-level input current †	V_{CC} = MAX	V_I = 0.4V			–0.02	–0.25	mA
I_{IH}	High-level input current †	V_{CC} = MAX	V_I = 2.4V				25	μA
I_I	Maximum input current	V_{CC} = MAX	V_I = 5.5V				1	mA
V_{OL}	Low-level output voltage	V_{CC} = MIN	MIL	I_{OL} = 4mA		0.3	0.5	V
			COM	I_{OL} = 4mA				
V_{OH}	High-level output voltage	V_{CC} = MIN	MIL	I_{OH} = –400μA	2.4	2.8		V
			COM	I_{OH} = –400μA				
I_{OZL}	Off-state output current †	V_{CC} = MAX		V_O = 0.4V			–100	μA
I_{OZH}				V_O = 2.4V			100	μA
I_{OS}	Output short-circuit current**	V_{CC} = 5V		V_O = 0V	–30	–70	–130	mA
I_{CC}	Supply current	V_{CC} = MAX				30	50	mA

Switching Characteristics Over Operating Conditions

SYMBOL	PARAMETER		TEST	MILITARY MIN	MILITARY TYP	MILITARY MAX	COMMERCIAL MIN	COMMERCIAL TYP	COMMERCIAL MAX	UNIT
t_{PD}	Input or feedback to output	16R6-4 16R4-4 16L8-4	R_1 = 800Ω, R_2 = 1.56kΩ		55	100		55	80	ns
t_{CLK}	Clock to output or feedback				40	80		40	60	ns
$t_{PXZ/ZX}$	Pin 11 to output disable/enable except 16L8A				25	50		25	40	ns
t_{PZX}	Input to output enable	16R6-4 16R4-4 16L8-4			50	100		50	75	ns
t_{PXZ}	Input to output disable	16R6-4 16R4-4 16L8-4			50	100		50	75	ns
f_{MAX}	Maximum frequency	16R8-4 16R6-4 16R4-4		5	10		7	10		MHz

Operating Conditions

SYMBOL	PARAMETER		MILITARY MIN	MILITARY TYP	MILITARY MAX	COMMERCIAL MIN	COMMERCIAL TYP	COMMERCIAL MAX	UNIT
V_{CC}	Supply voltage		4.5	5	5.5	4.75	5	5.25	V
t_w	Width of clock	Low	25	10		25	10		ns
		High	25	10		25	10		
t_{su}	Set up time from input or feedback to clock	16R6A-2 16R4A-2 16R8A-2	50	25		35	25		ns
t_h	Hold time		0	-15		0	-15		ns
T_A	Operating free-air temperature		-55		125	0		75	°C

Electrical Characteristics Over Operating Conditions

SYMBOL	PARAMETER	TEST CONDITIONS			MIN	TYP	MAX	UNIT
V_{IL}*	Low-level input voltage						0.8	V
V_{IH}*	High-level input voltage				2			V
V_{IC}	Input clamp voltage	V_{CC} = MIN	I_I = -18mA			-0.8	-1.5	V
I_{IL}	Low-level input current †	V_{CC} = MAX	V_I = 0.4V			-0.02	-0 25	mA
I_{IH}	High-level input current †	V_{CC} = MAX	V_I = 2.4V				25	μA
I_I	Maximum input current	V_{CC} = MAX	V_I = 5.5V				1	mA
V_{OL}	Low-level output voltage	V_{CC} = MIN	MIL	I_{OL} = 12mA		0.3	0.5	V
			COM	I_{OL} = 24mA				
V_{OH}	High-level output voltage	V_{CC} = MIN	MIL	I_{OH} = -2mA	2 4	2.8		V
			COM	I_{OH} = -3.2mA				
I_{OZL}	Off-state output current †	V_{CC} = MAX		V_O = 0.4V			-100	μA
I_{OZH}				V_O = 2.4V			100	μA
I_{OS}	Output short-circuit current**	V_{CC} = 5V		V_O = 0V	-30	-70	-130	mA
I_{CC}	Supply current	V_{CC} = MAX				60	90	mA

Switching Characteristics Over Operating Conditions

SYMBOL	PARAMETER		TEST CONDITIONS	MILITARY MIN	MILITARY TYP	MILITARY MAX	COMMERCIAL MIN	COMMERCIAL TYP	COMMERCIAL MAX	UNIT
t_{PD}	Input or feedback to output	16L8A-2 16R6A-2 16R4A-2	R_1 = 200Ω, R_2 = 390Ω		25	50		25	35	ns
t_{CLK}	Clock to output or feedback				15	25		15	25	ns
$t_{PXZ/ZX}$	Pin 11 to output disable/enable except 16L8A-2				15	25		15	25	ns
t_{PZX}	Input to output enable	16L8A-2 16R6A-2 16R4A-2			25	45		25	35	ns
t_{PXZ}	Input to output disable	16R8A-2 16R6A-2 16R4A-2			25	45		25	35	ns
f_{MAX}	Maximum frequency	16R8A-2 16R6A-2 16R4A-2		14	25		16	25		MHz

Operating Conditions

SYMBOL	PARAMETER			MILITARY MIN	MILITARY TYP	MILITARY MAX	COMMERCIAL MIN	COMMERCIAL TYP	COMMERCIAL MAX	UNIT
V_{CC}	Supply voltage			4.5	5	5.5	4.75	5	5.25	V
t_w	Width of clock	16R8A-4 16R6A-4 16R4A-4	Low	40	20		30	20		ns
			High	40	20		30	20		
t_{su}	Set up time from input or feedback to clock	16R8A-4 16R6A-4 16R4A-4		90	45		60	45		ns
t_h	Hold time			0	−15		0	−15		ns
T_A	Operating free-air temperature			−55		125	0		75	°C

Electrical Characteristics Over Operating Conditions

SYMBOL	PARAMETER	TEST CONDITIONS			MIN	TYP	MAX	UNIT
V_{IL}*	Low-level input voltage						0 8	V
V_{IH}*	High-level input voltage				2			V
V_{IC}	Input clamp voltage	V_{CC} = MIN	I_I = −18mA			−0.8	−1.5	V
I_{IL}	Low-level input current †	V_{CC} = MAX	V_I = 0.4V			−0.02	−0 25	mA
I_{IH}	High-level input current †	V_{CC} = MAX	V_I = 2 4V				25	μA
I_I	Maximum input current	V_{CC} = MAX	V_I = 5.5V				1	mA
V_{OL}	Low-level output voltage	V_{CC} = MIN	MIL	I_{OL} = 4mA		0.3	0 5	V
			COM	I_{OL} = 8mA				
V_{OH}	High-level output voltage	V_{CC} = MIN	MIL	I_{OH} = −1mA	2.4	2.8		V
			COM	I_{OH} = −1 mA				
I_{OZL}	Output short-circuit current**	V_{CC} = MAX		V_O = 0.4V			−100	μA
I_{OZH}				V_O = 2.4V			100	μA
I_{OS}	Output short-circuit current	V_{CC} = 5V		V_O = 0V	−30	−70	−130	mA
I_{CC}	Supply current	V_{CC} = MAX	16R4A-4 16R6A-4 16R8A-4 16L8A-4			30	50	mA

Switching Characteristics Over Operating Conditions

SYMBOL	PARAMETER		TEST	MILITARY MIN	MILITARY TYP	MILITARY MAX	COMMERCIAL MIN	COMMERCIAL TYP	COMMERCIAL MAX	UNIT
t_{PD}	Input or feedback to output	16R6A-4 16R4A-4 16L8A-4	R_1 = 800Ω, R_2 = 1.56kΩ		35	75		35	55	ns
t_{CLK}	Clock to output or feedback				20	45		20	35	ns
$t_{PXZ/ZX}$	Pin 11 to output disable/enable—except 16L8A-4				15	40		15	30	ns
t_{PZX}	Input to output enable	16R6A-4 16R4A-4 16L8A-4			30	65		30	50	ns
t_{PXZ}	Input to output disable	16R6A-4 16R4A-4 16L8A-4			30	65		30	50	ns
f_{MAX}	Maximum frequency	16R8A-4 16R6A-4 16R4A-4		8	18		11	18		MHz

Programming/Verifying Procedure

NOTES: For programming purposes many PAL pins have double functions

For The PAL 20:

As long as Pin 1 is at HH, Pin 11 is at ground, and Pin 12 is either at HH or Z (as defined in Table 1) — Pins 16, 17, 18, and 19 are outputs. The other pin functions are: I0 (Pin 2) through I7 (Pin 9) plus Pin 12 address the proper row; A0 (Pin 15), A1 (Pin 14), and A2 (Pin 13) address the proper product lines.

When Pin 11 is at HH, Pin 1 is at ground and Pin 19 is either at HH or Z — Pins 12, 13, 14, and 15 are outputs. The other pin functions are: I0 (Pin 2) through I7 (Pin 9) plus Pin 19 address the proper row; A0 (now Pin 18), A1 (now Pin 17), and A2 (now Pin 16) address the proper product lines.

For The PAL 24:

As long as Pin 1 is at HH, Pin 13 is at ground and Pin 14 is either at HH or Z (as defined in Table 1) — Pins 19, 20, 21, 22, and 23 are outputs. The other pin functions are: I0 (Pin 2) through I9 (Pin 11) plus Pin 14 address the proper row; A0 (Pin 15), A1 (Pin 16), and A2 (Pin 17) address the proper product lines.

As long as Pin 13 is at HH, Pin 1 is at ground, and Pin 23 is either at HH or Z (as defined in Table 1) — Pins 14, 15, 16, 17, and 18 are outputs. The other pin functions are: I0 (Pin 2) through I9 (Pin 11) plus Pin 23 address the proper row; A0 (Pin 22), A1 (Pin 21), and A2 (Pin 20) address the proper product lines.

For The PAL 24A:

As long as Pin 1 is at HH, Pin 13 is at ground, and Pin 14 is either at HH or Z (as defined in Table 1) — Pins 19, 20, 21, and 22 are outputs. The other pin functions are: I0 (Pin 2) through I9 (Pin 11) plus Pin 14 address the proper row; A0 (Pin 15), A1 (Pin 16), and A2 (Pin 17) address the proper product lines.

As long as Pin 13 is at HH, Pin 1 is at ground, and Pin 23 is either at HH or Z (as defined in Table 1) — pins 15, 16, 17, and 18 are outputs. The other Pin functions are: I0 (Pin 2) through I9 (Pin 11) plus Pin 23 address the proper row; A0 (Pin 22), A1 (Pin 21), and A2 (Pin 20) address the proper product lines.

Pre-Verification

5.1.1 Raise V_{CC} to 5.0 volts.

5.1.2 Raise Output Disable pin, OD, to VIHH.

5.1.3 Select an input line by specifying Inputs and L/R as shown in Table 1 or Table 2.

5.1.4 Select a product line by specifying A0, A1, and A2 one-of-eight select as shown in Table 3, Table 4 or Table 5.

5.1.5 Pulse the CLOCK pin and verify (with CLOCK at VIL) that the output pin, O, is in the state corresponding to an unblown fuse.

— For verified unblown condition, continue procedure from 5.1.3 through 5.1.5.

— For verified blown condition, stop procedure and reject part.

Programming Algorithm

5.2.1 Raise Output Disable pin, OD, to VIHH

5.2.2 Programming pass. For all fuses to be blown:

5.2.2.1 Lower CLOCK pin to ground.

5.2.2.2 Select an input line by specifying Inputs and L/R as shown in Table 1 or Table 2.

5.2.2.3 Select a product line by specifying A0, A1, and A2 one-of-eight select as shown in Table 3, Table 4 or Table 5.

5.2.2.4 Raise V_{CC} to VIHH.

5.2.2.5 Program the fuse by pulsing the output pins of the selected product group -one at a time- to VIHH (as shown in the Programming Waveforms, Section 5.5).

5.2.2.6 Lower V_{CC} to 5.0 volts.

5.2.2.7 Repeat this procedure from 5.2.2.2 until pattern is complete.

5.2.3 First verification pass. For all fuse locations:

5.2.3.1 Select an input line by specifying Inputs and L/R as shown in Table 1 or Table 2.

5.2.3.2 Select a product line by specifying A0, A1, and A2 one-of-eight select as shown in Table 3, Table 4 or Table 5.

5.2.3.3 Pulse the CLOCK pin and verify (with CLOCK at VIL) that the output pin, O, is in the correct state.

— For verified output state, continue procedure

— For overblow condition, stop procedure and reject part.

— For underblow condition, reexecute steps 5.2.2.4 through 5.2.2.6 and 5.2.2.3. If successful, continue procedure. After three attempts to blow fuse without success, reject part but continue procedure.

5.2.3.4 Repeat this procedure from 5.2.3.1 until the entire array is exercised.

5.2.4 High Voltage Verify. For all fuse locations:

5.2.4.1 Raise V_{CC} to 5.5 volts.

5.2.4.2 Select an input line by specifying Inputs and L/R as shown in Table 1 or Table 2.

5.2.4.3 Select a product line by specifying A0, A1, and A2 one-of-eight select as shown in Table 3, Table 4 or Table 5.

5.2.4.4 Pulse the CLOCK pin and verify (with CLOCK at VIL) that the output pin, O, is in the correct state.

— For verified output state, continue procedure

— For invalid output state, stop procedure and reject part.

5.2.4.5 Repeat this procedure from 5.2.4.1 until the entire array is exercised.

5.2.5 Low Voltage Verify. For all fuse locations:

5.2.5.1 Lower V_{CC} to 4.5 volts.

5.2.5.2 Select an input line by specifying Inputs and L/R as shown in Table 1 or Table 2.

5.2.5.3 Select a product line by specifying A0, A1, and A2, one-of-eight select as shown in Table 3, Table 4 or Table 5.

5.2.5.4 Pulse the CLOCK pin and verify (with CLOCK at VIL) that the output pin, O, is in the correct state.

— For verified output state, continue procedure.

— For invalid output state, continue procedure and reject part.

Programming the Security Fuses

5.3.1 Verify per Section 5.2.4 and 5.2.5.

5.3.2 Raise V_{CC} to 6 volts.

5.3.3 For PAL 20:

— Program the first fuse by pulsing Pin 1 to VP. (From 1 to 5 pulses is acceptable.)

— Program the second fuse by pulsing Pin 11 to VP. (From 1 to 5 pulses is acceptable.)

5.3.4 For PAL 24 and PAL 24A:

— Program the first fuse by pulsing Pin 1 to VP. (From 1 to 5 pulses is acceptable.)

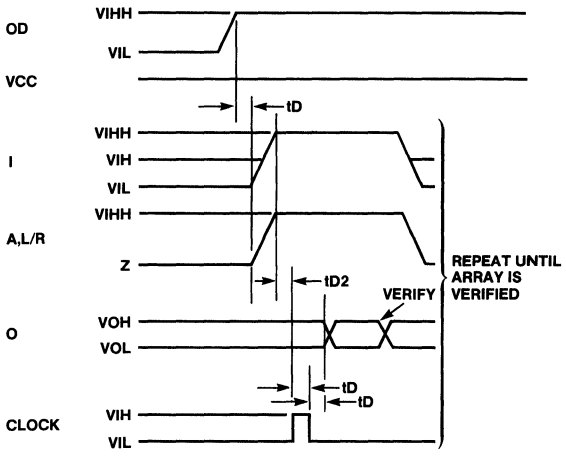
— Program the second fuse by pulsing Pin 13 to VP. (From 1 to 5 pulses is acceptable.)

5.3.5 Verify per Section 5.2.4 and 5.2.5:

— A device is "secure" if either half fails to verify.

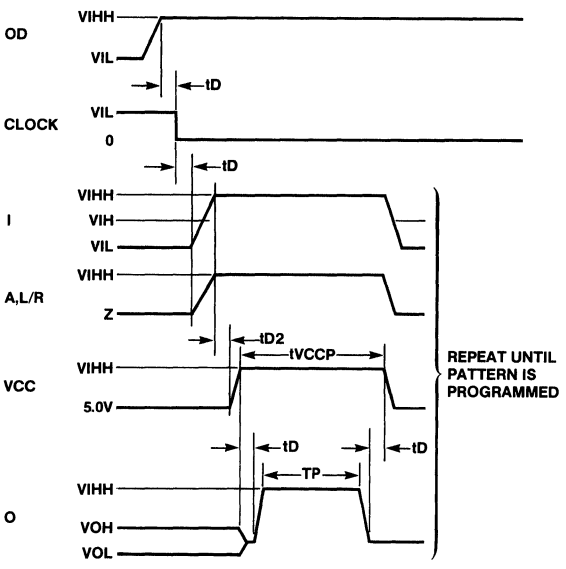
5.4 Programming Parameters

SYMBOL	PARAMETER		LIMITS MIN	TYP	MAX	UNIT
V_{IHH}	Program-level input voltage		11.5	11.75	12	V
I_{IHH}	Program-level input current	Output Program Pulse			50	mA
		OD, L/R			50	
		All other inputs			10	
I_{CCH}	Program Supply Current				900	mA
t_{VCCP}	Pulse Width of V_{CC} @ V_{IHH}				60	μS
T_P	Program Pulse Width		10	20	50	μS
t_D	Delay Time		100			ns
t_{D2}	Delay Time after L/R Pin		10			μS
	V_{CCP} Duty Cycle				20	%
V_P	Security Fuse Programming Voltage		18	18.5	19	V
I_P	Security Fuse Programming Supply Current				400	mA
T_{PP}	Security Fuse Programming Pulse Width		10	40	70	μS
	Security Fuse Programming Duty Cycle				50	%
t_{RP}	Rise time of output programming and address pulses		1	1.5	10	V/μS
t_{RP}	Rise Time of security fuse programming pulses		1	1.5	10	V/μS
V_{CCPP}	V_{CC} value during security fuse programming		5.75	6.0	6.25	V
	V_{CC}value for first verify		4.75	5.0	5.25	
	V_{CC} value for High V_{CC} verify		5.4	5.5	5.6	
	V_{CC} value for Low V_{CC} verify		4.4	4.5	4.6	

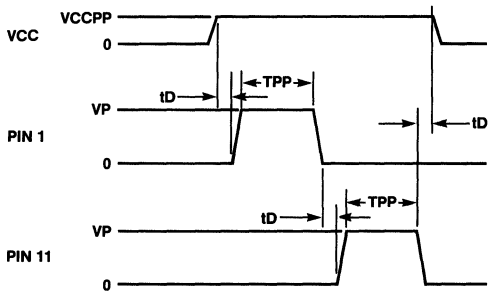


NOTE: VCC (Low Voltage Verify) = 4.5 volts
VCC (High Voltage Verify) = 5.5 volts
VCC (First Verify) = 5.0 volts
A Delay (tD2) must always precede the Positive Clock Transition. (e.g see section 5.2.3.3 for underblow condition)

5.5 ARRAY PROGRAMMING



5.6 SECURITY FUSE PROGRAMMING



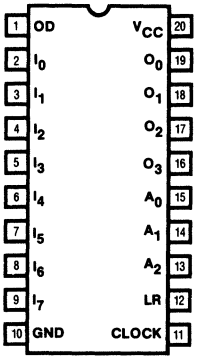
Programmer/Development Systems

VENDOR	PAL 20s (ALL)	PAL 24s (STD)	PAL 24s (FAST)
Data I/O	— LogicPak (Rev-010) — 1427 Card Set	LogicPak (Rev-010)	— LogicPak (Rev-010)
Structured Design	— SD 20/24 — PAL Burner*	— SD 20/24 — PAL Burner*	— SD 20/24 — PAL Burner*
STAG	— PM202 (Rev 3) — PM2200*	— PM202 (Rev 3) — PM2200*	— PM202 (Rev) — PM2200*
DIGELEC	— UP803 (FAM51) or (FAM52)	— UP803 (FAM51) or (FAM52)	— UP803 (FAM51) or (FAM52)
PROLOG	— M980 PM9068		
KONTRON	— MPP80S MOD 21		

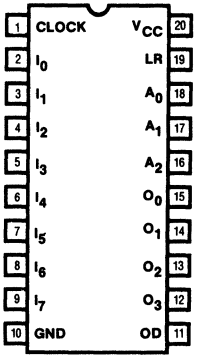
*Means that this version is being qualified.

Programming Pin Configurations

PRODUCTS 0 THRU 31



PRODUCTS 32 THRU 63



Voltage Legend

L = Low-level input voltage, V_{IL}
H = High-level input voltage, V_{IH}
HH = High-level program voltage, V_{IHH}
Z = High impedance (e.g., 10kΩ to 5.0V)

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	Z
1	HH	HH	HH	HH	HH	HH	HH	H	Z
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	Z
5	HH	HH	HH	HH	HH	HH	H	HH	Z
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	Z
9	HH	HH	HH	HH	HH	H	HH	HH	Z
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	Z
13	HH	HH	HH	HH	H	HH	HH	HH	Z
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	Z
17	HH	HH	HH	H	HH	HH	HH	HH	Z
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	Z
21	HH	HH	H	HH	HH	HH	HH	HH	Z
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	Z
25	HH	H	HH	HH	HH	HH	HH	HH	Z
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	Z
29	H	HH	HH	HH	HH	HH	HH	HH	Z
30	L	HH	HH	HH	HH	HH	HH	HH	HH
31	H	HH	HH	HH	HH	HH	HH	HH	HH

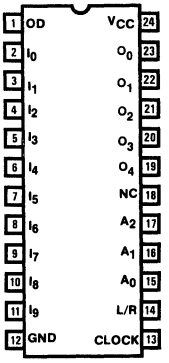
Table 1 Input Line Select

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

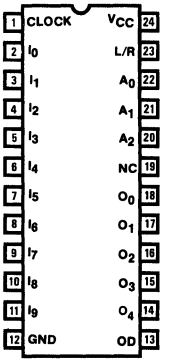
Table 2 Product Line Select

Programming Pin Configurations

PRODUCTS 0 THRU 39



PRODUCTS 40 THRU 79



Voltage Legend

L = Low-level input voltage, V_{IL}
H = High-level input voltage, V_{IH}
HH = High-level program voltage, V_{IHH}
Z = High impedance (e.g. 10K Ω to 5.0V)

INPUT LINE NUMBER	PIN IDENTIFICATION										
	I_9	I_8	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	HH	HH	L	Z
1	HH	HH	HH	HH	HH	HH	HH	HH	HH	H	Z
2	HH	HH	HH	HH	HH	HH	HH	HH	HH	L	HH
3	HH	HH	HH	HH	HH	HH	HH	HH	HH	H	HH
4	HH	HH	HH	HH	HH	HH	HH	HH	L	HH	Z
5	HH	HH	HH	HH	HH	HH	HH	HH	H	HH	Z
6	HH	HH	HH	HH	HH	HH	HH	HH	L	HH	HH
7	HH	HH	HH	HH	HH	HH	HH	HH	H	HH	HH
8	HH	HH	HH	HH	HH	HH	HH	L	HH	HH	Z
9	HH	HH	HH	HH	HH	HH	HH	H	HH	HH	Z
10	HH	HH	HH	HH	HH	HH	HH	L	HH	HH	HH
11	HH	HH	HH	HH	HH	HH	HH	H	HH	HH	HH
12	HH	HH	HH	HH	HH	HH	L	HH	HH	HH	Z
13	HH	HH	HH	HH	HH	HH	H	HH	HH	HH	Z
14	HH	HH	HH	HH	HH	HH	L	HH	HH	HH	HH
15	HH	HH	HH	HH	HH	HH	H	HH	HH	HH	HH
16	HH	HH	HH	HH	HH	L	HH	HH	HH	HH	Z
17	HH	HH	HH	HH	HH	H	HH	HH	HH	HH	Z
18	HH	HH	HH	HH	HH	L	HH	HH	HH	HH	HH
19	HH	HH	HH	HH	HH	H	HH	HH	HH	HH	HH
20	HH	HH	HH	HH	L	HH	HH	HH	HH	HH	Z
21	HH	HH	HH	HH	H	HH	HH	HH	HH	HH	Z
22	HH	HH	HH	HH	L	HH	HH	HH	HH	HH	HH
23	HH	HH	HH	HH	H	HH	HH	HH	HH	HH	HH
24	HH	HH	HH	L	HH	HH	HH	HH	HH	HH	Z
25	HH	HH	HH	H	HH	HH	HH	HH	HH	HH	Z
26	HH	HH	HH	L	HH	HH	HH	HH	HH	HH	HH
27	HH	HH	HH	H	HH	HH	HH	HH	HH	HH	HH
28	HH	HH	L	HH	HH	HH	HH	HH	HH	HH	Z
29	HH	HH	H	HH	HH	HH	HH	HH	HH	HH	Z
30	HH	HH	L	HH	HH	HH	HH	HH	HH	HH	HH
31	HH	HH	H	HH	HH	HH	HH	HH	HH	HH	HH
32	HH	L	HH	HH	HH	HH	HH	HH	HH	HH	Z
33	HH	H	HH	HH	HH	HH	HH	HH	HH	HH	Z
34	HH	L	HH	HH	HH	HH	HH	HH	HH	HH	HH
35	HH	H	HH	HH	HH	HH	HH	HH	HH	HH	HH
36	L	HH	HH	HH	HH	HH	HH	HH	HH	HH	Z
37	H	HH	HH	HH	HH	HH	HH	HH	HH	HH	Z
38	L	HH	HH	HH	HH	HH	HH	HH	HH	HH	HH
39	H	HH	HH	HH	HH	HH	HH	HH	HH	HH	HH

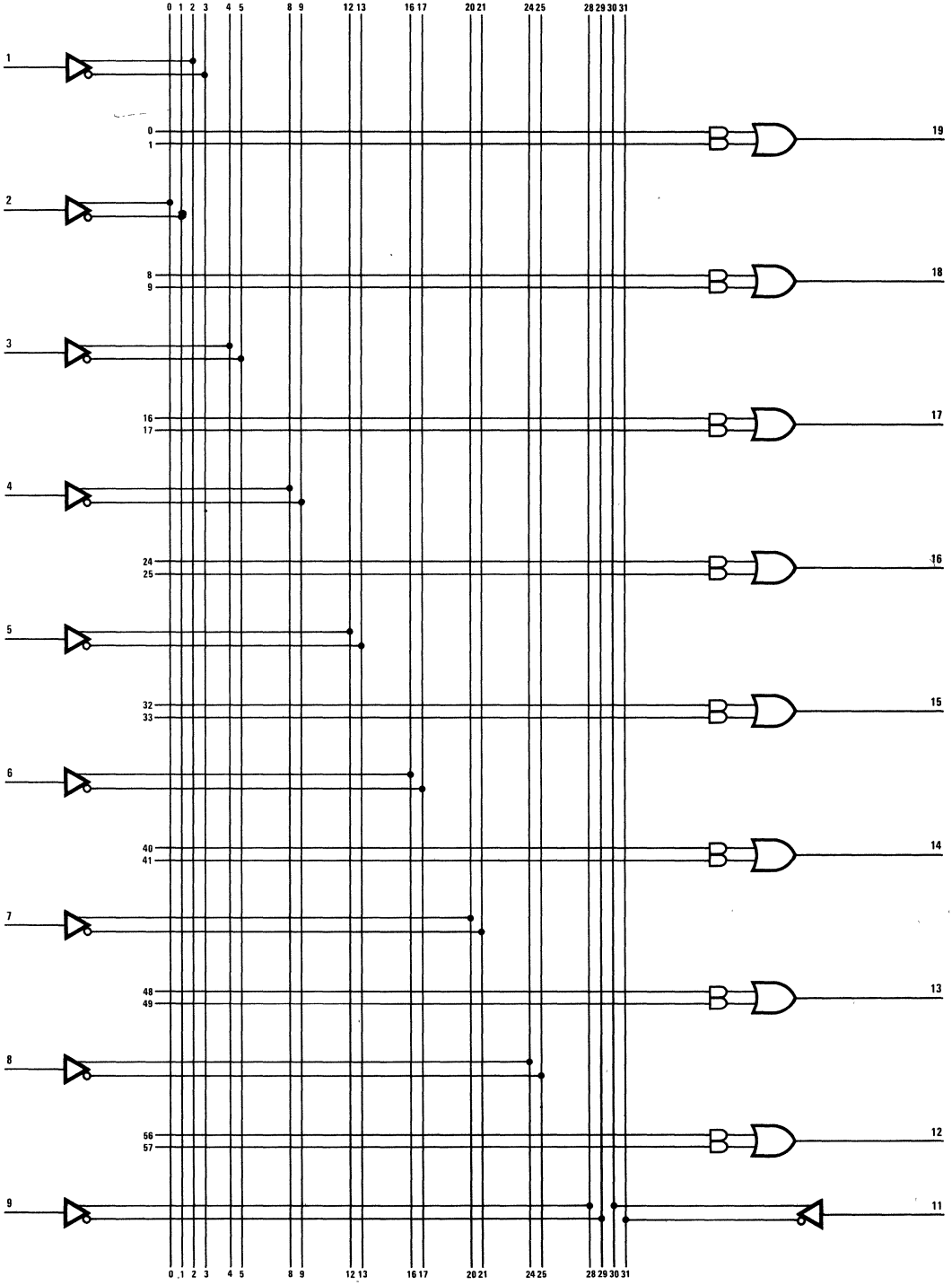
Table 1 Input Line Select

PRODUCT LINE NUMBER	PIN IDENTIFICATION							
	O_4	O_3	O_2	O_1	O_0	A_2	A_1	A_0
0, 40	Z	Z	Z	Z	HH	Z	Z	Z
1, 41	Z	Z	Z	Z	HH	Z	Z	HH
2, 42	Z	Z	Z	Z	HH	Z	HH	Z
3, 43	Z	Z	Z	Z	HH	Z	HH	HH
4, 44	Z	Z	Z	Z	HH	HH	Z	Z
5, 45	Z	Z	Z	Z	HH	HH	Z	HH
6, 46	Z	Z	Z	Z	HH	HH	HH	Z
7, 47	Z	Z	Z	Z	HH	HH	HH	HH
8, 48	Z	Z	Z	HH	Z	Z	Z	Z
9, 49	Z	Z	Z	HH	Z	Z	Z	HH
10, 50	Z	Z	Z	HH	Z	Z	HH	Z
11, 51	Z	Z	Z	HH	Z	Z	HH	HH
12, 52	Z	Z	Z	HH	Z	HH	Z	Z
13, 53	Z	Z	Z	HH	Z	HH	Z	HH
14, 54	Z	Z	Z	HH	Z	HH	HH	Z
15, 55	Z	Z	Z	HH	Z	HH	HH	HH
16, 56	Z	Z	HH	Z	Z	Z	Z	Z
17, 57	Z	Z	HH	Z	Z	Z	Z	HH
18, 58	Z	Z	HH	Z	Z	Z	HH	Z
19, 59	Z	Z	HH	Z	Z	Z	HH	HH
20, 60	Z	Z	HH	Z	Z	HH	Z	Z
21, 61	Z	Z	HH	Z	Z	HH	Z	HH
22, 62	Z	Z	HH	Z	Z	HH	HH	Z
23, 63	Z	Z	HH	Z	Z	HH	HH	HH
24, 64	Z	HH	Z	Z	Z	Z	Z	Z
25, 65	Z	HH	Z	Z	Z	Z	Z	HH
26, 66	Z	HH	Z	Z	Z	Z	HH	Z
27, 67	Z	HH	Z	Z	Z	Z	HH	HH
28, 68	Z	HH	Z	Z	Z	HH	Z	Z
29, 69	Z	HH	Z	Z	Z	HH	Z	HH
30, 70	Z	HH	Z	Z	Z	HH	HH	Z
31, 71	Z	HH	Z	Z	Z	HH	HH	HH
32, 72	HH	Z	Z	Z	Z	Z	Z	Z
33, 73	HH	Z	Z	Z	Z	Z	Z	HH
34, 74	HH	Z	Z	Z	Z	Z	HH	Z
35, 75	HH	Z	Z	Z	Z	Z	HH	HH
36, 76	HH	Z	Z	Z	Z	HH	Z	Z
37, 77	HH	Z	Z	Z	Z	HH	Z	HH
38, 78	HH	Z	Z	Z	Z	HH	HH	Z
39, 79	HH	Z	Z	Z	Z	HH	HH	HH

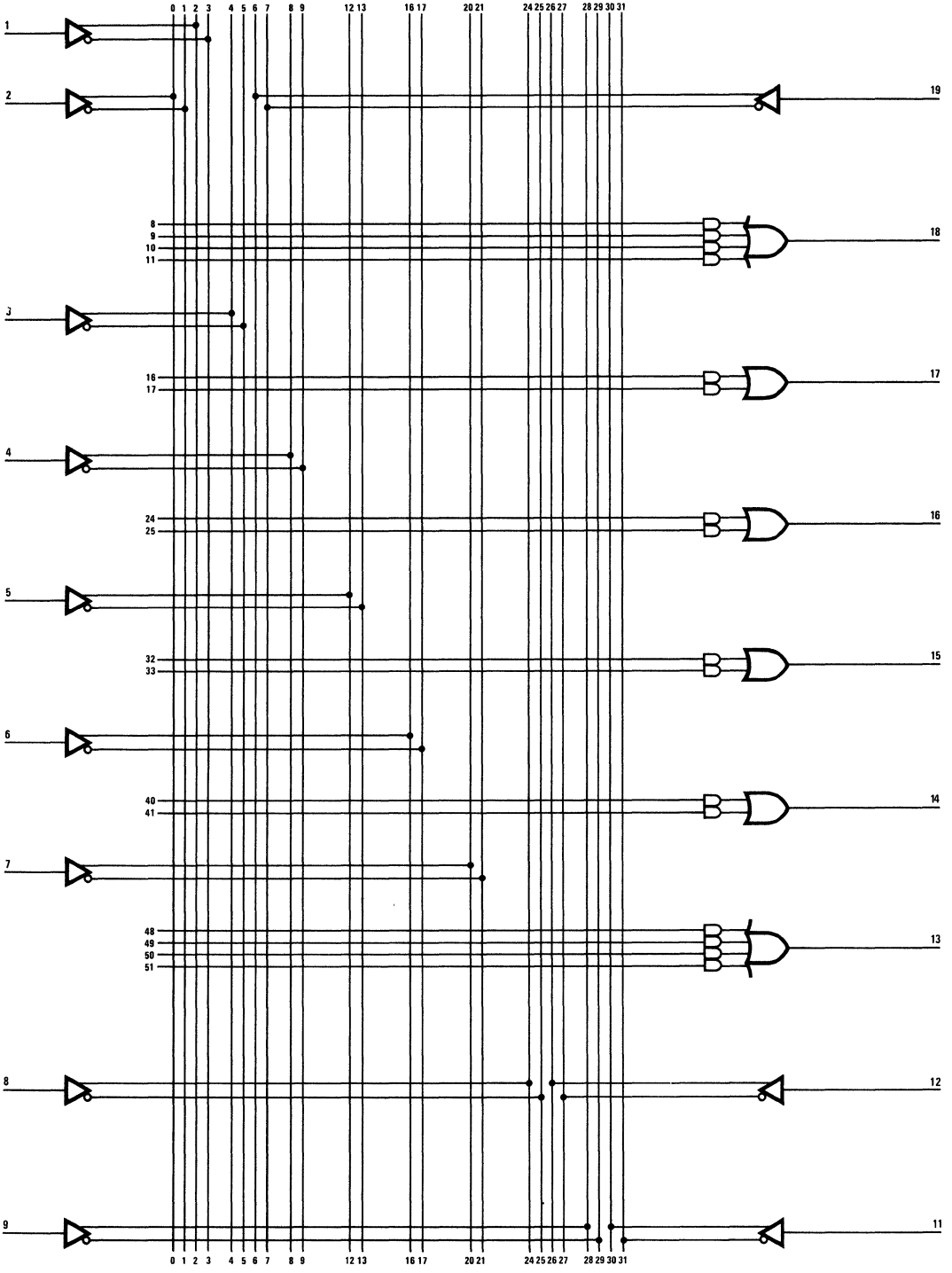
Table 2 Product Line Select

3

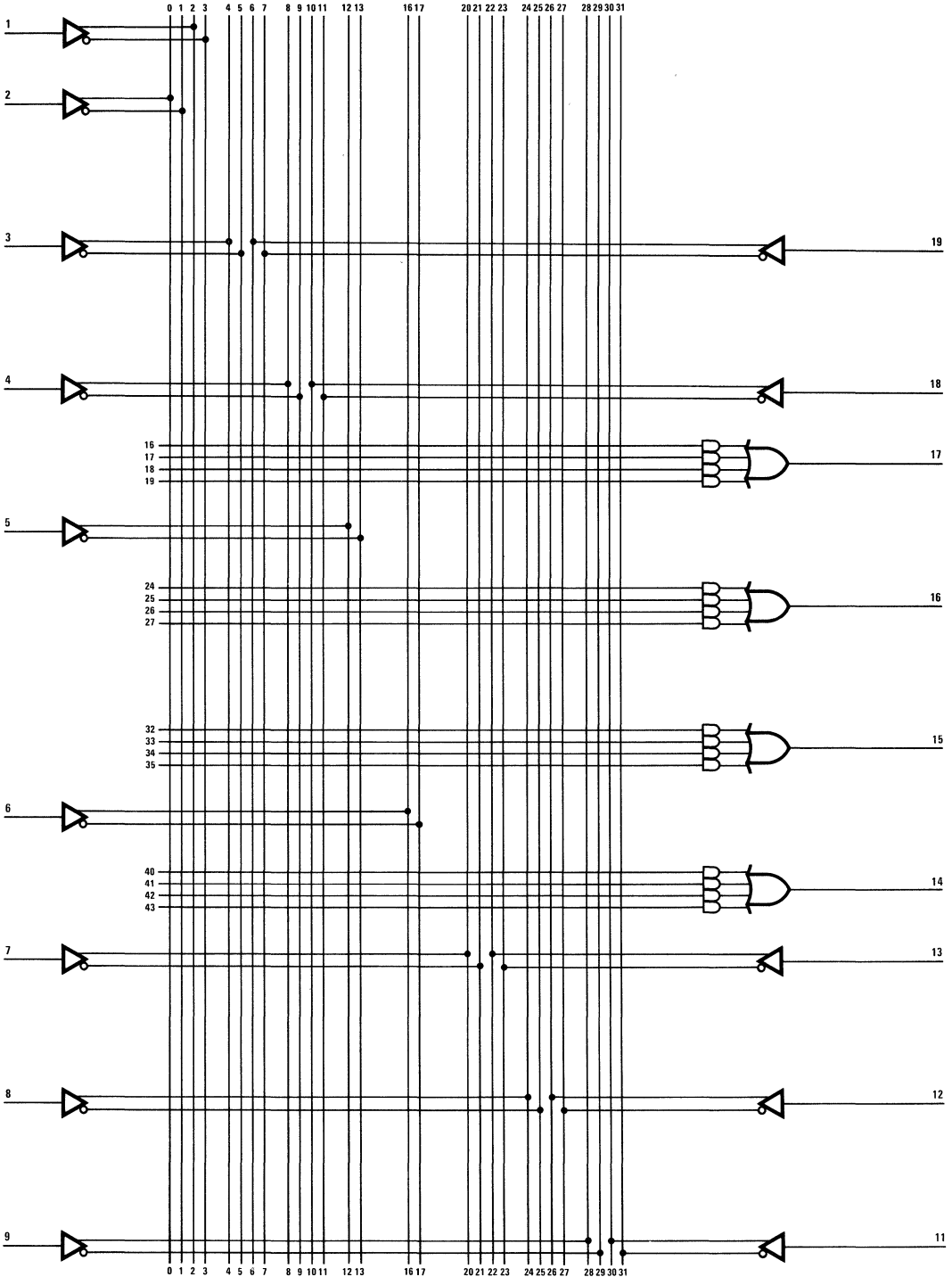
10H8


0 1 2 3 4 5 8 9 12 13 16 17 20 21 24 25 28 29 30 31
1
0
1
19
2
8
9
18
3
16
17
17
4
24
25
16
5
32
33
15
6
40
41
14
7
48
49
13
8
56
57
12
9
11
0 1 2 3 4 5 8 9 12 13 16 17 20 21 24 25 28 29 30 31

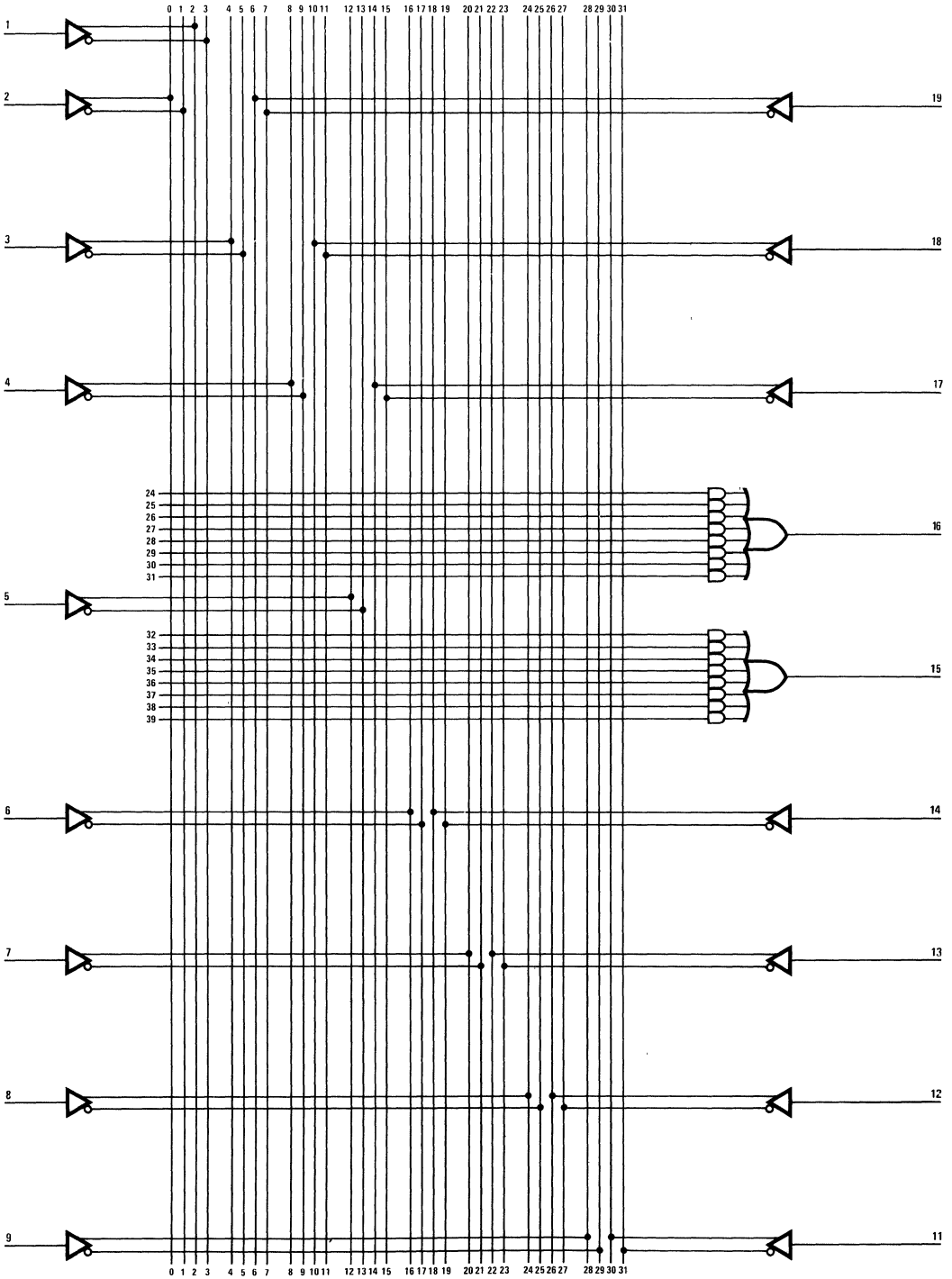
12H6


0 1 2 3
4 5 6 7
8 9
12 13
16 17
20 21
24 25 26 27
28 29 30 31
1
2
19
8
9
10
11
18
3
16
17
17
4
24
25
16
5
32
33
15
6
40
41
14
7
48
49
50
51
13
8
12
9
11

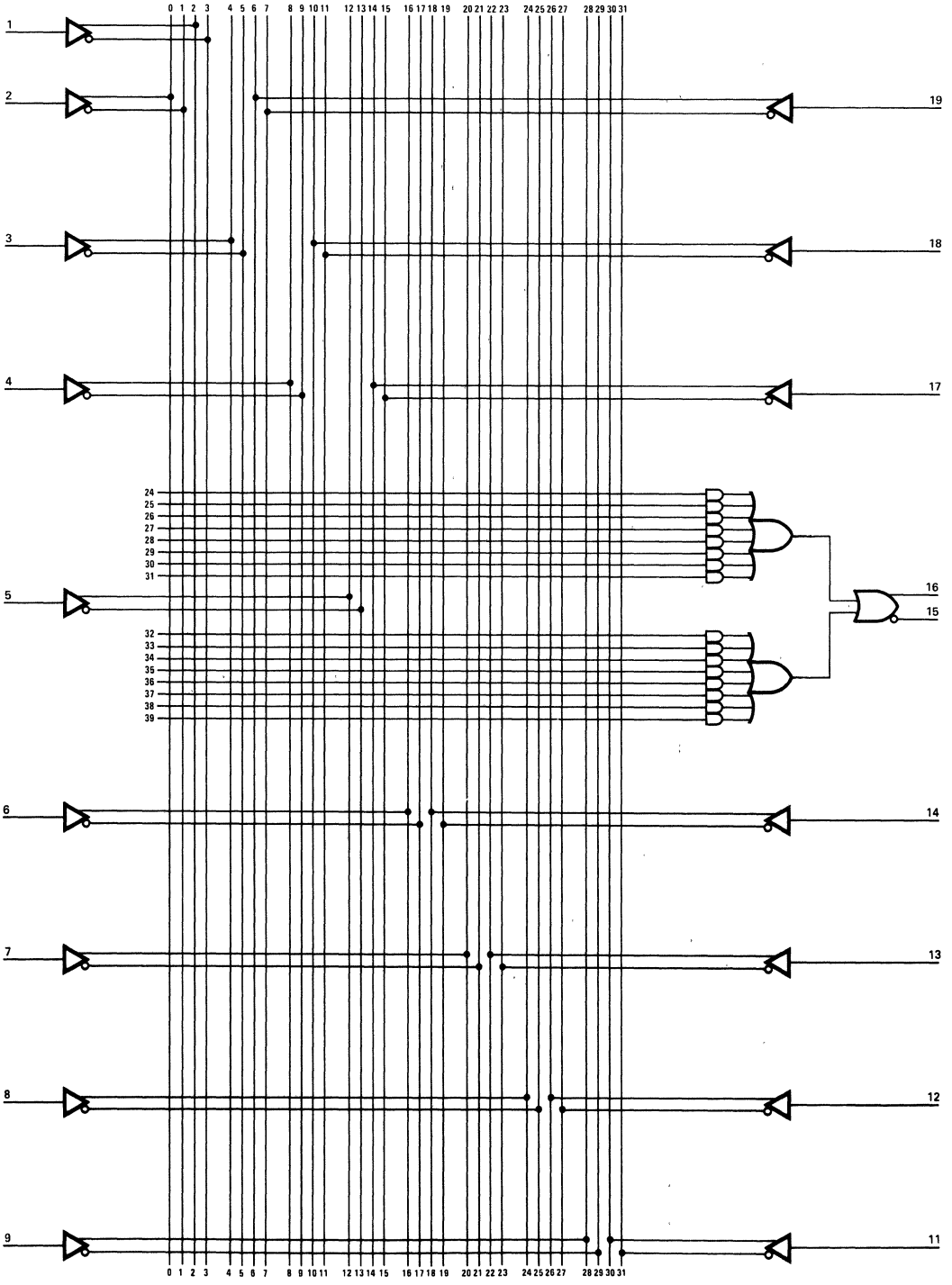
14H4


0 1 2 3 4 5 6 7 8 9 10 11 12 13 16 17 20 21 22 23 24 25 26 27 28 29 30 31
1
2
3
4
16
17
18
19
5
24
25
26
27
32
33
34
35
6
40
41
42
43
7
8
9
19
18
17
16
15
14
13
12
11

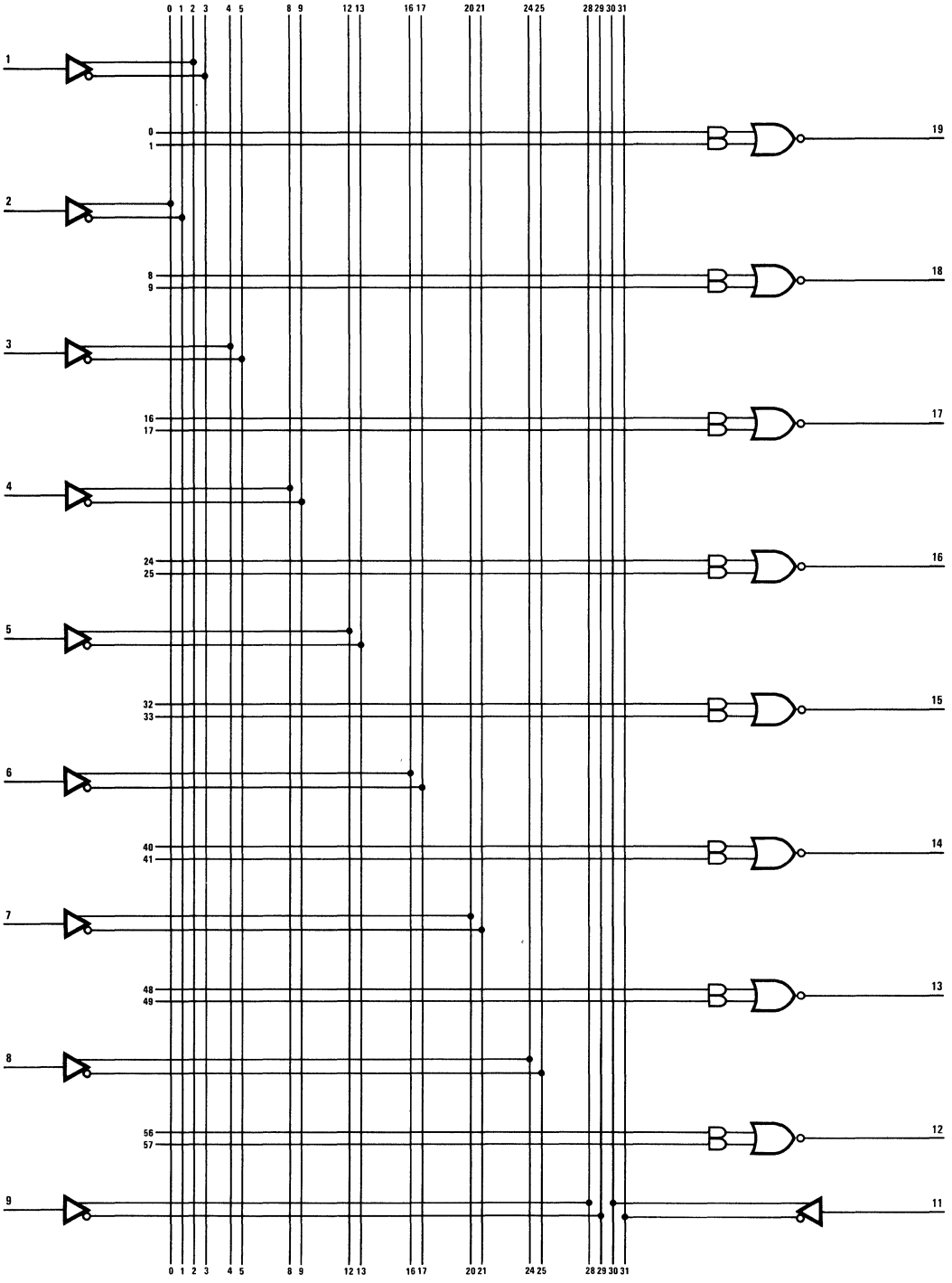
16H2


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
1
2
3
4
5
6
7
8
9
19
18
17
16
15
14
13
12
11
24 25 26 27 28 29 30 31
32 33 34 35 36 37 38 39

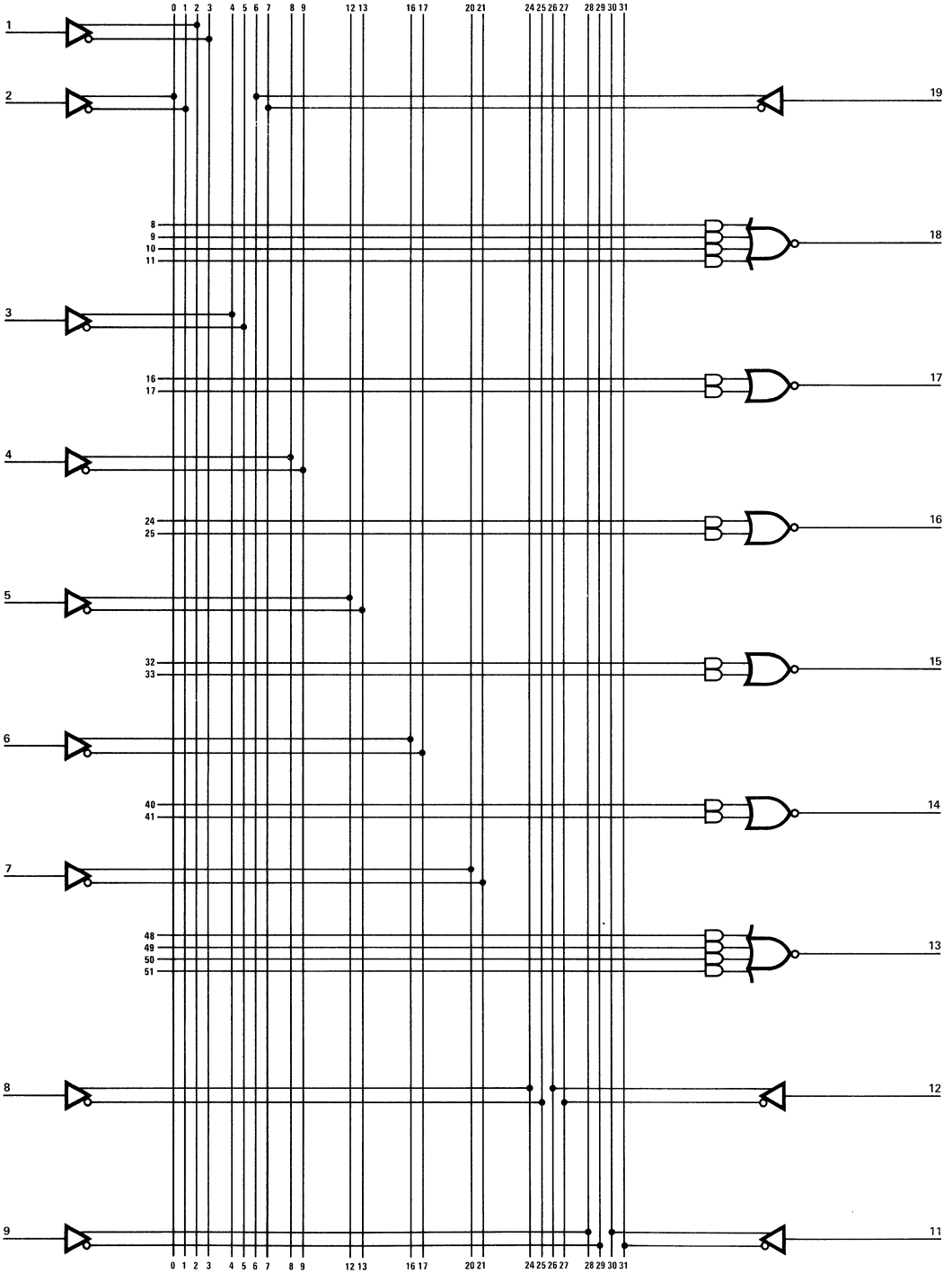
16C1


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
1
2
3
4
5
6
7
8
9
19
18
17
16
15
14
13
12
11
24 25 26 27 28 29 30 31
32 33 34 35 36 37 38 39

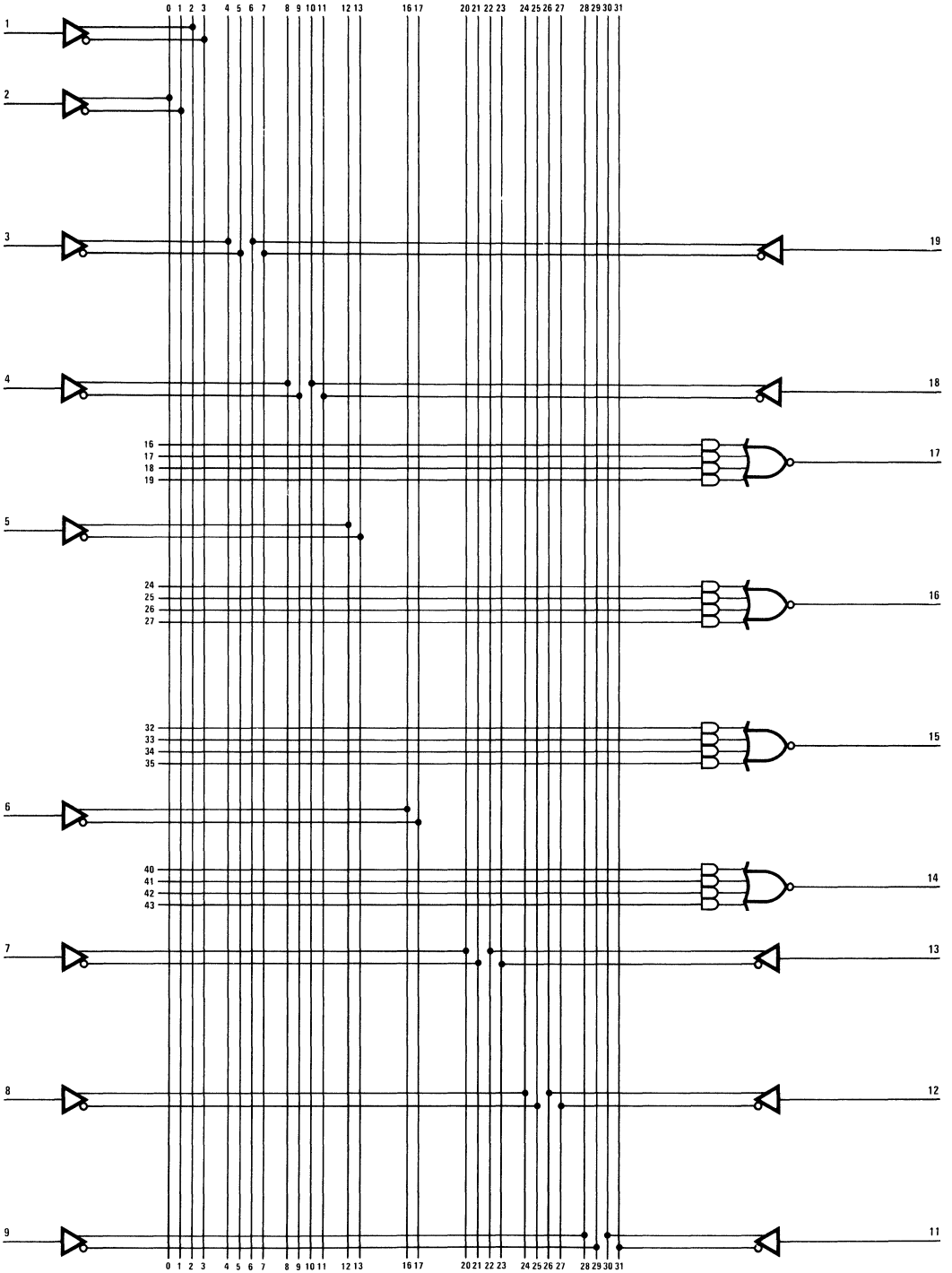
10L8


0 1 2 3 4 5 8 9 12 13 16 17 20 21 24 25 28 29 30 31
1
0
1
19
2
8
9
18
3
16
17
17
4
24
25
16
5
32
33
15
6
40
41
14
7
48
49
13
8
56
57
12
9
11

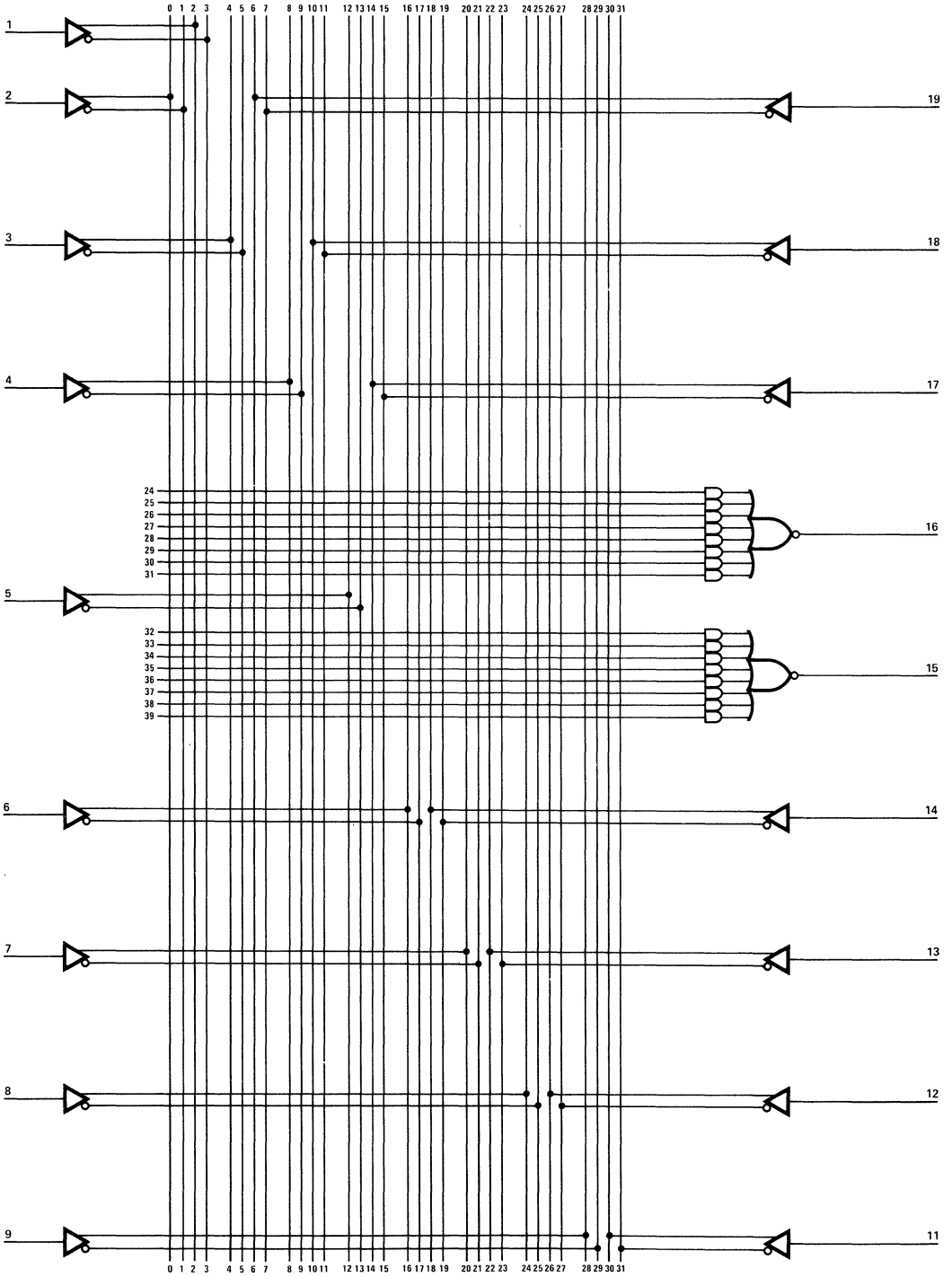
12L6


0 1 2 3 4 5 6 7 8 9 12 13 16 17 20 21 24 25 26 27 28 29 30 31
1
2
19
8
9
10
11
18
3
16
17
17
4
24
25
16
5
32
33
15
6
40
41
14
7
48
49
50
51
13
8
12
9
11
0 1 2 3 4 5 6 7 8 9 12 13 16 17 20 21 24 25 26 27 28 29 30 31

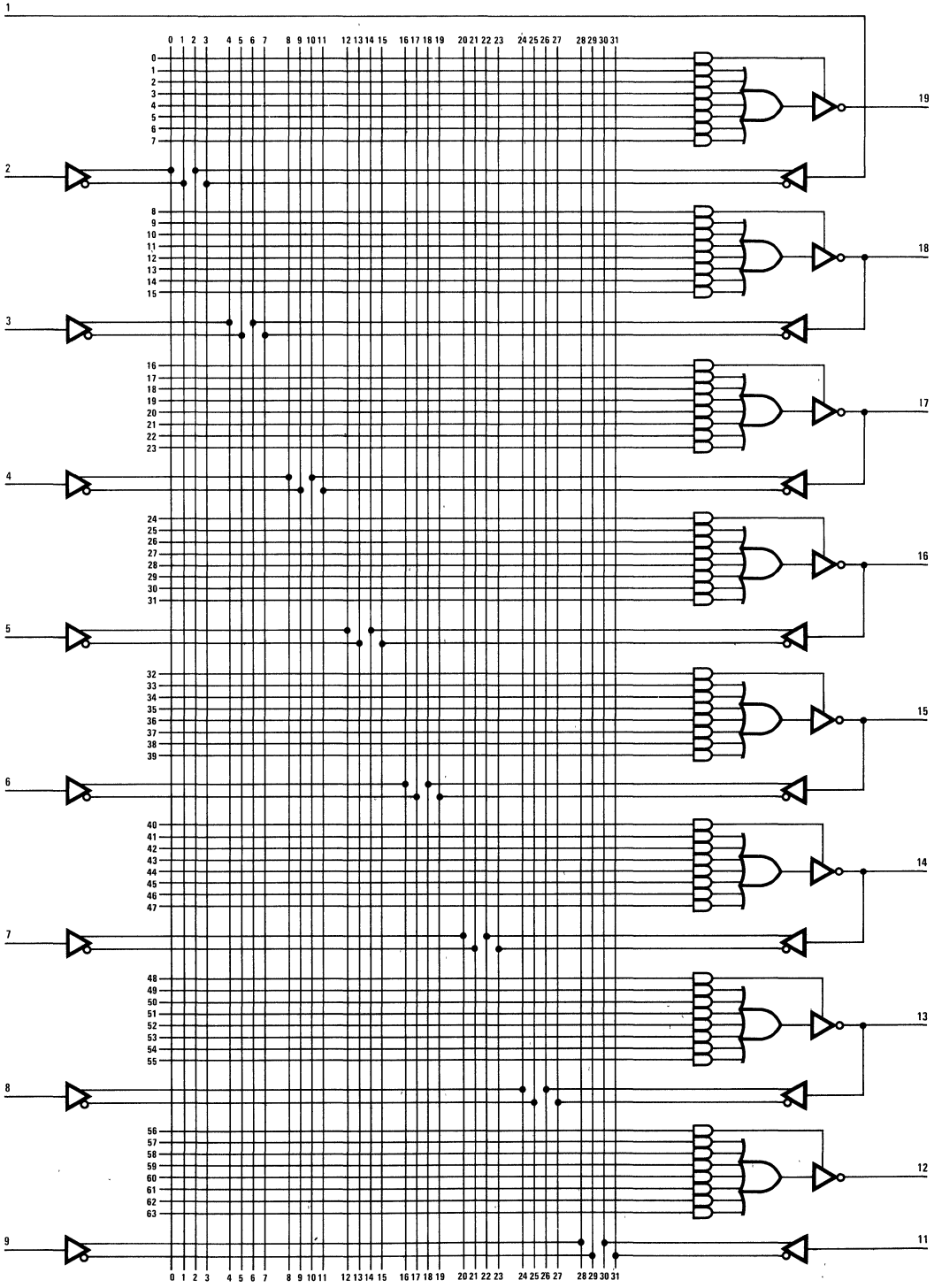
14L4


0 1 2 3 4 5 6 7 8 9 10 11 12 13 16 17 20 21 22 23 24 25 26 27 28 29 30 31
1
2
3
4
5
6
7
8
9
19
18
17
16
15
14
13
12
11
16 17 18 19
24 25 26 27
32 33 34 35
40 41 42 43

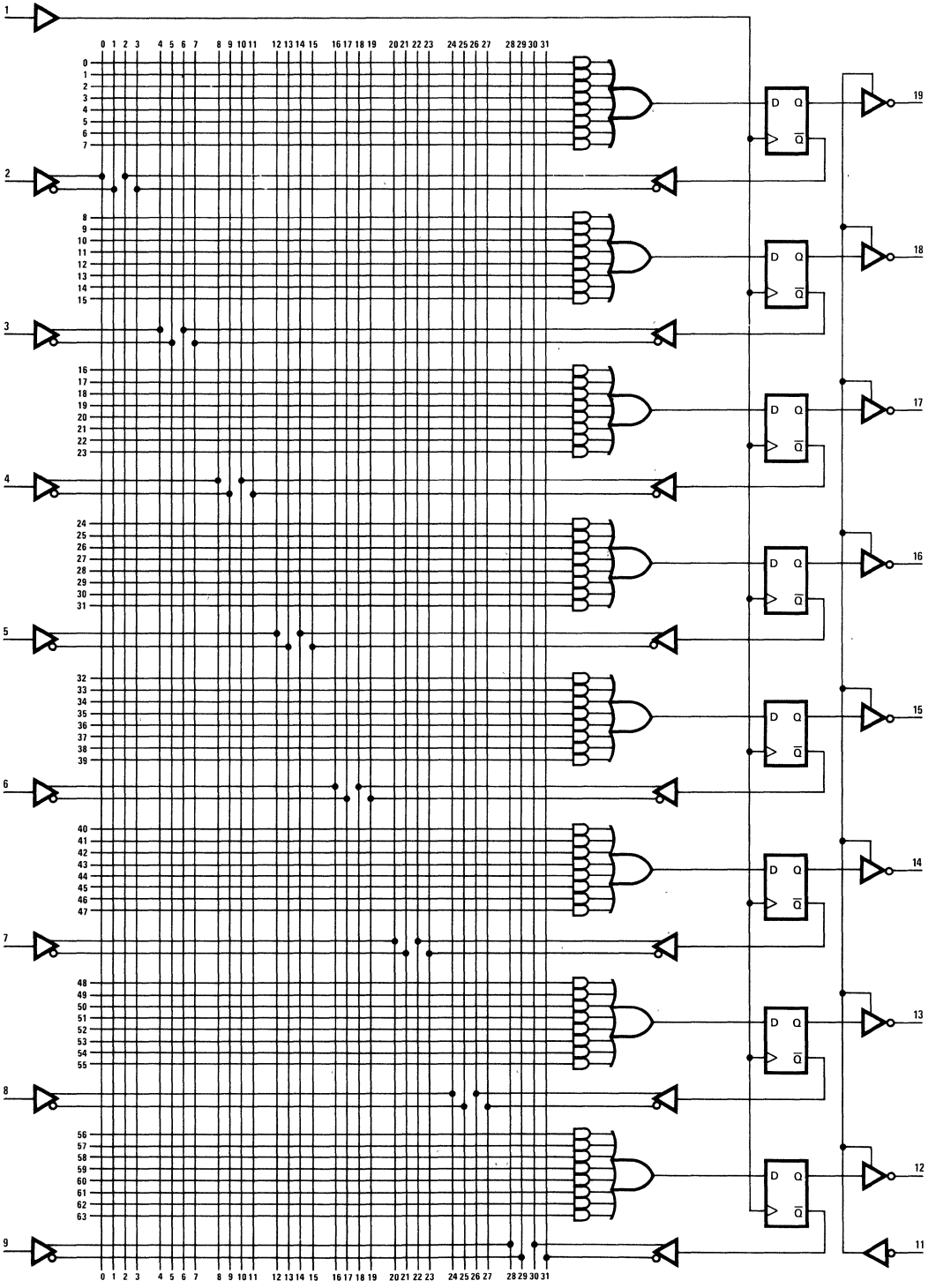
16L2


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
1
2
3
4
5
6
7
8
9
19
18
17
16
15
14
13
12
11
24 25 26 27 28 29 30 31
32 33 34 35 36 37 38 39

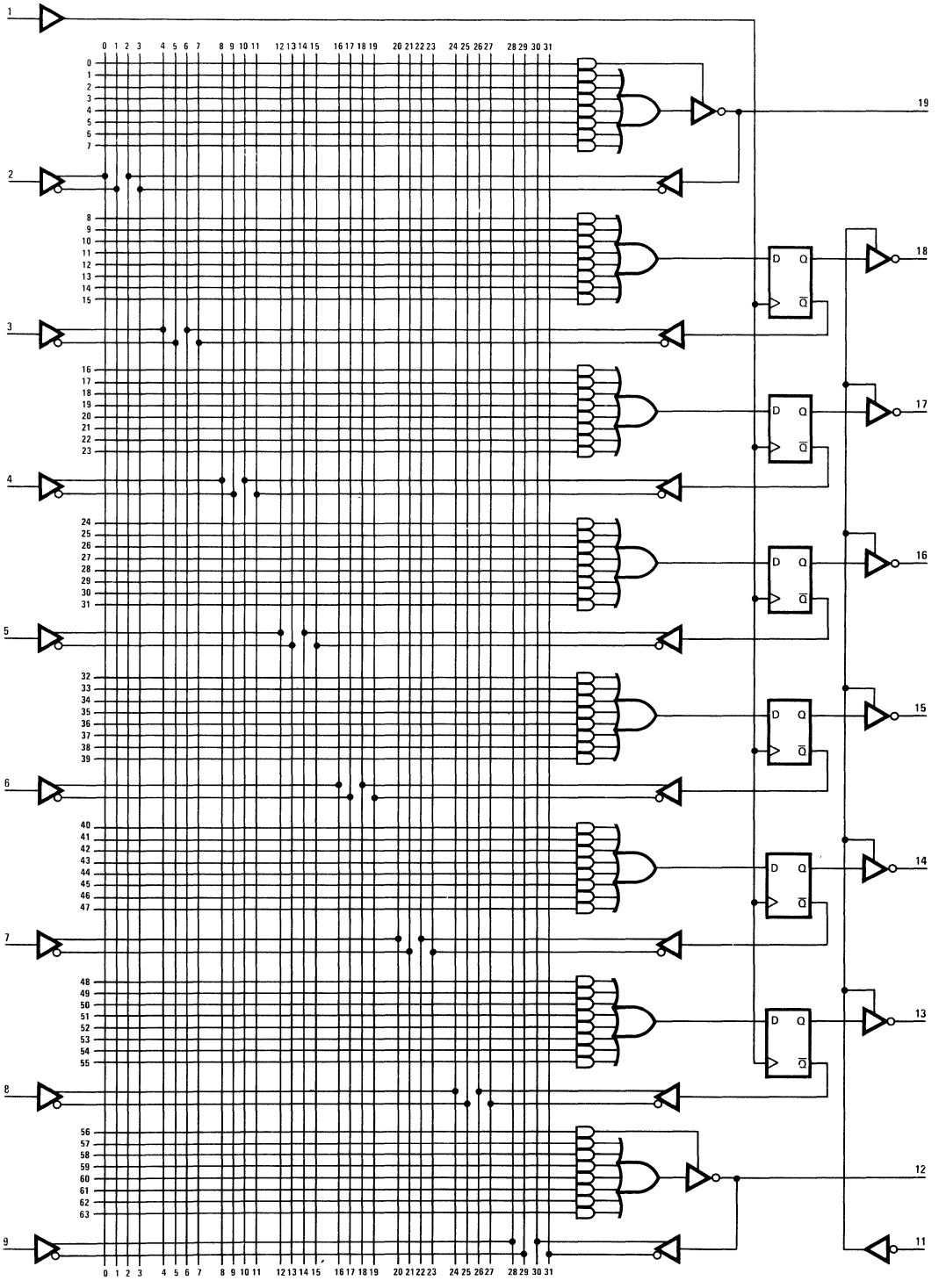
16L8


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

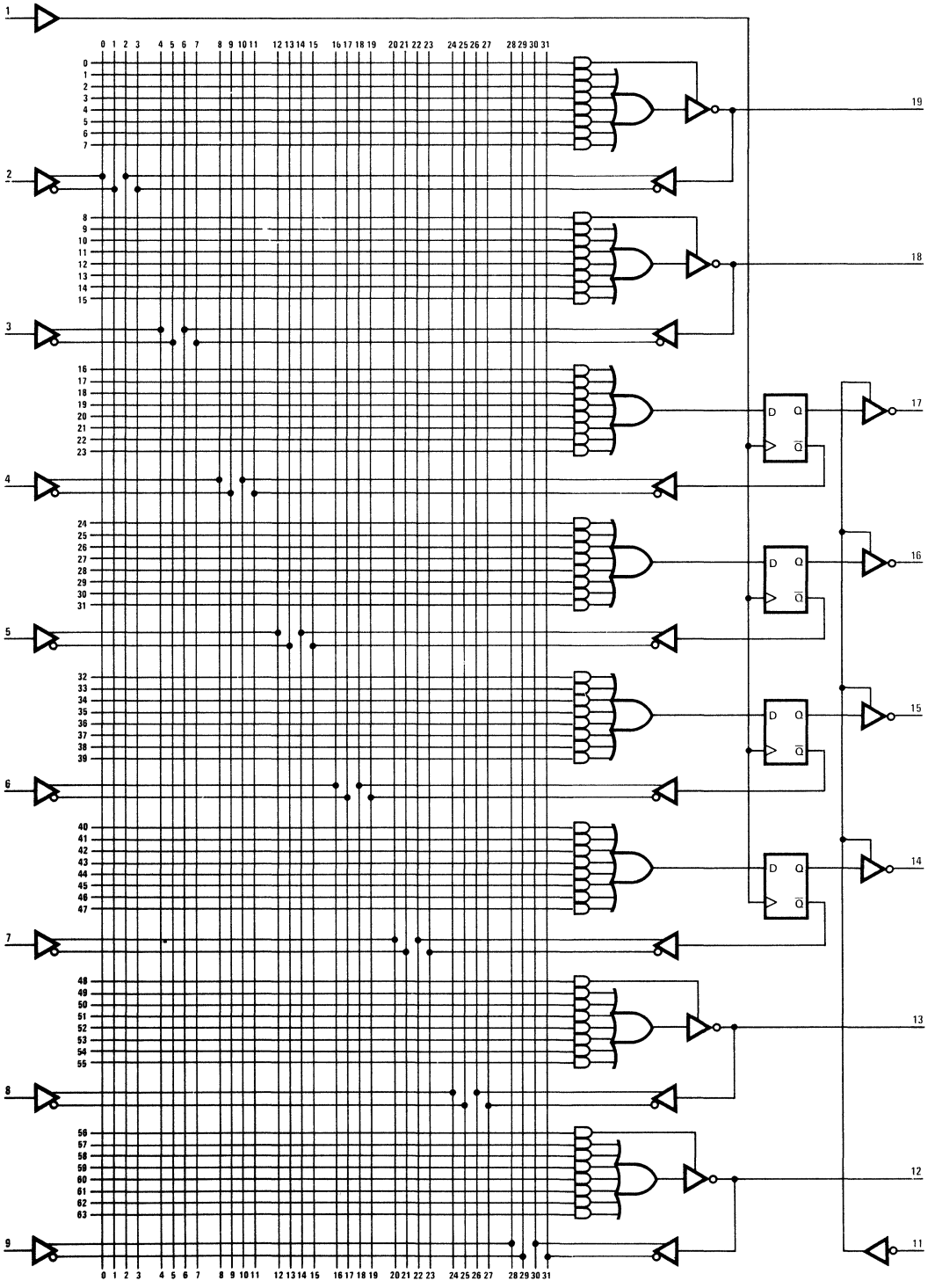
16R8


1
2
3
4
5
6
7
8
9
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
0 1 2 3 4 5 6 7
8 9 10 11 12 13 14 15
16 17 18 19 20 21 22 23
24 25 26 27 28 29 30 31
32 33 34 35 36 37 38 39
40 41 42 43 44 45 46 47
48 49 50 51 52 53 54 55
56 57 58 59 60 61 62 63
D Q
Q
19
18
17
16
15
14
13
12
11

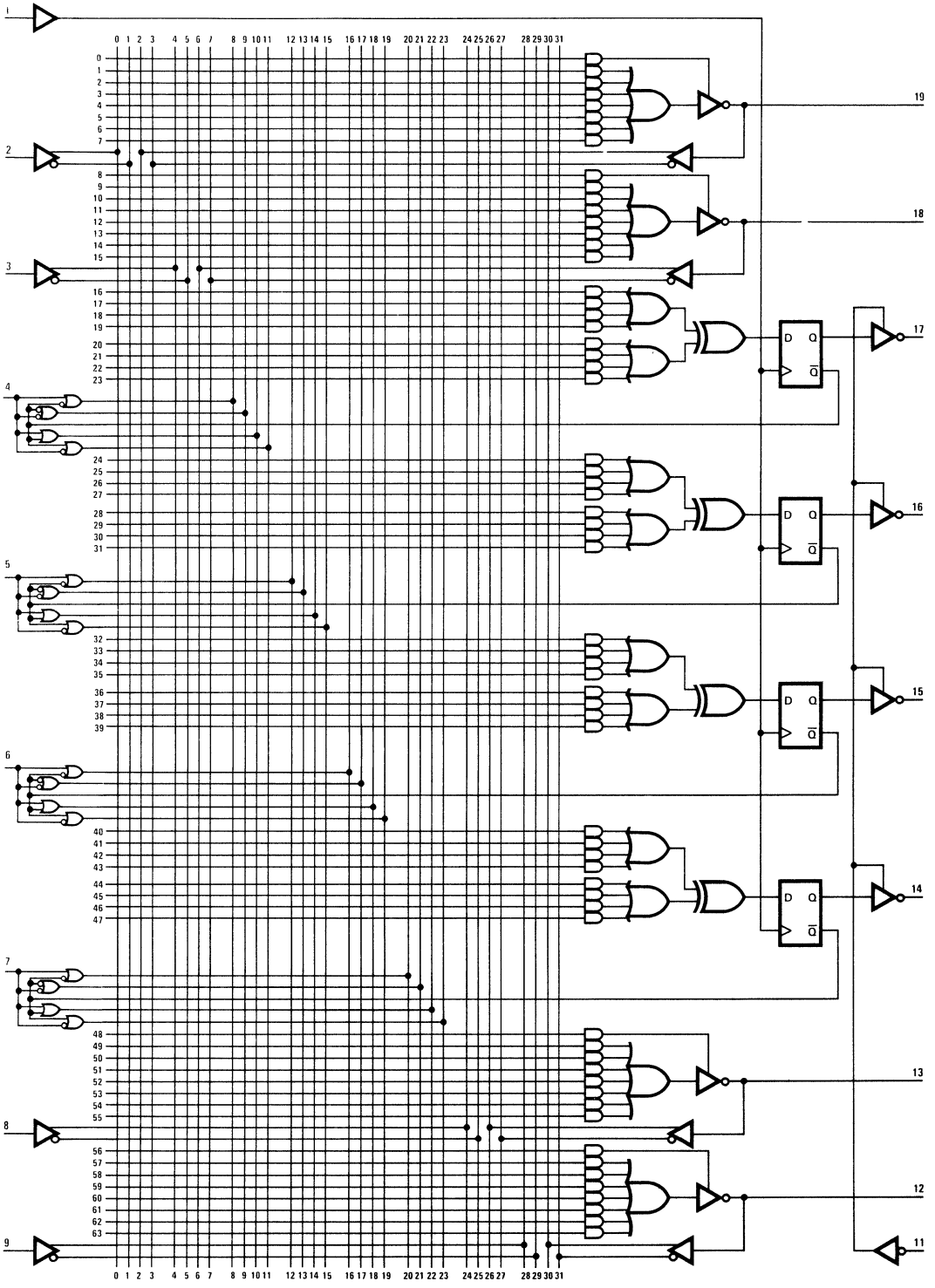
16R6


1
2
3
4
5
6
7
8
9
19
18
17
16
15
14
13
12
11
D
Q
Q̄
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
0 1 2 3 4 5 6 7
8 9 10 11 12 13 14 15
16 17 18 19 20 21 22 23
24 25 26 27 28 29 30 31
32 33 34 35 36 37 38 39
40 41 42 43 44 45 46 47
48 49 50 51 52 53 54 55
56 57 58 59 60 61 62 63

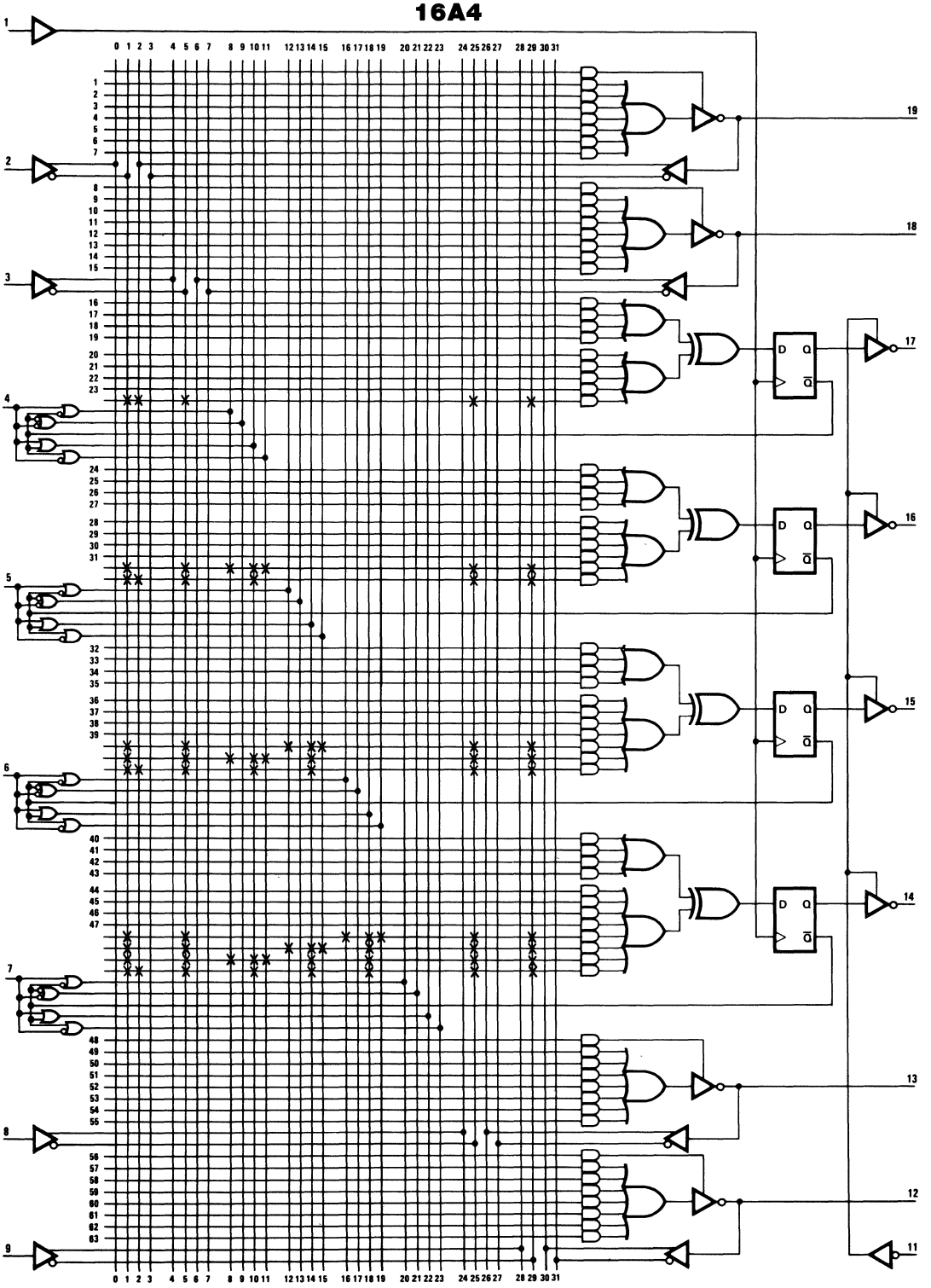
16R4


1
2
3
4
5
6
7
8
9
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
0 1 2 3 4 5 6 7
8 9 10 11 12 13 14 15
16 17 18 19 20 21 22 23
24 25 26 27 28 29 30 31
32 33 34 35 36 37 38 39
40 41 42 43 44 45 46 47
48 49 50 51 52 53 54 55
56 57 58 59 60 61 62 63
19
18
17
16
15
14
13
12
11
D
Q
Q̄

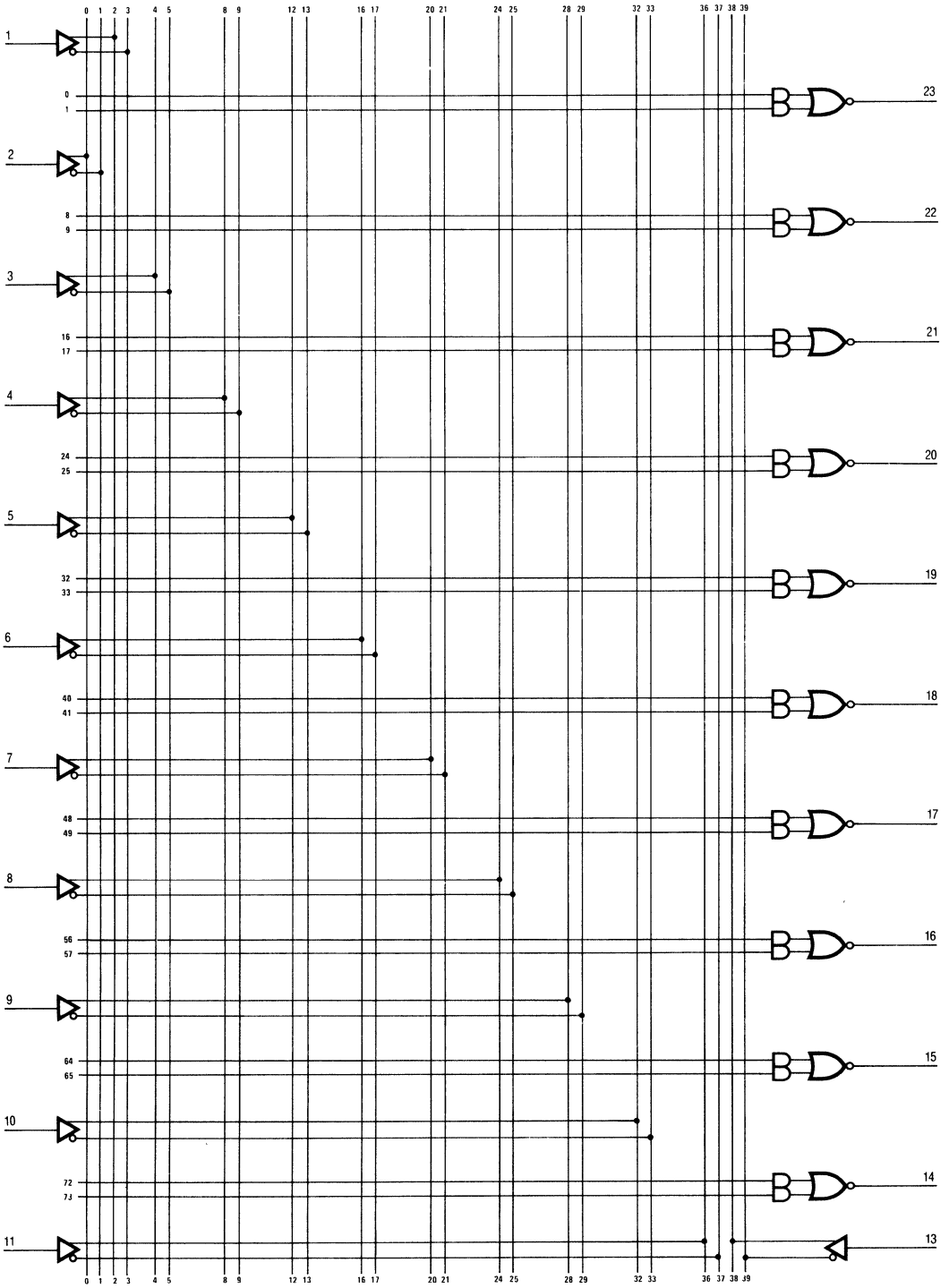
16X4


1
2
3
4
5
6
7
8
9
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
0 1 2 3 4 5 6 7
8 9 10 11 12 13 14 15
16 17 18 19 20 21 22 23
24 25 26 27 28 29 30 31
32 33 34 35 36 37 38 39
40 41 42 43 44 45 46 47
48 49 50 51 52 53 54 55
56 57 58 59 60 61 62 63
19
18
17
16
15
14
13
12
11
D Q
Q

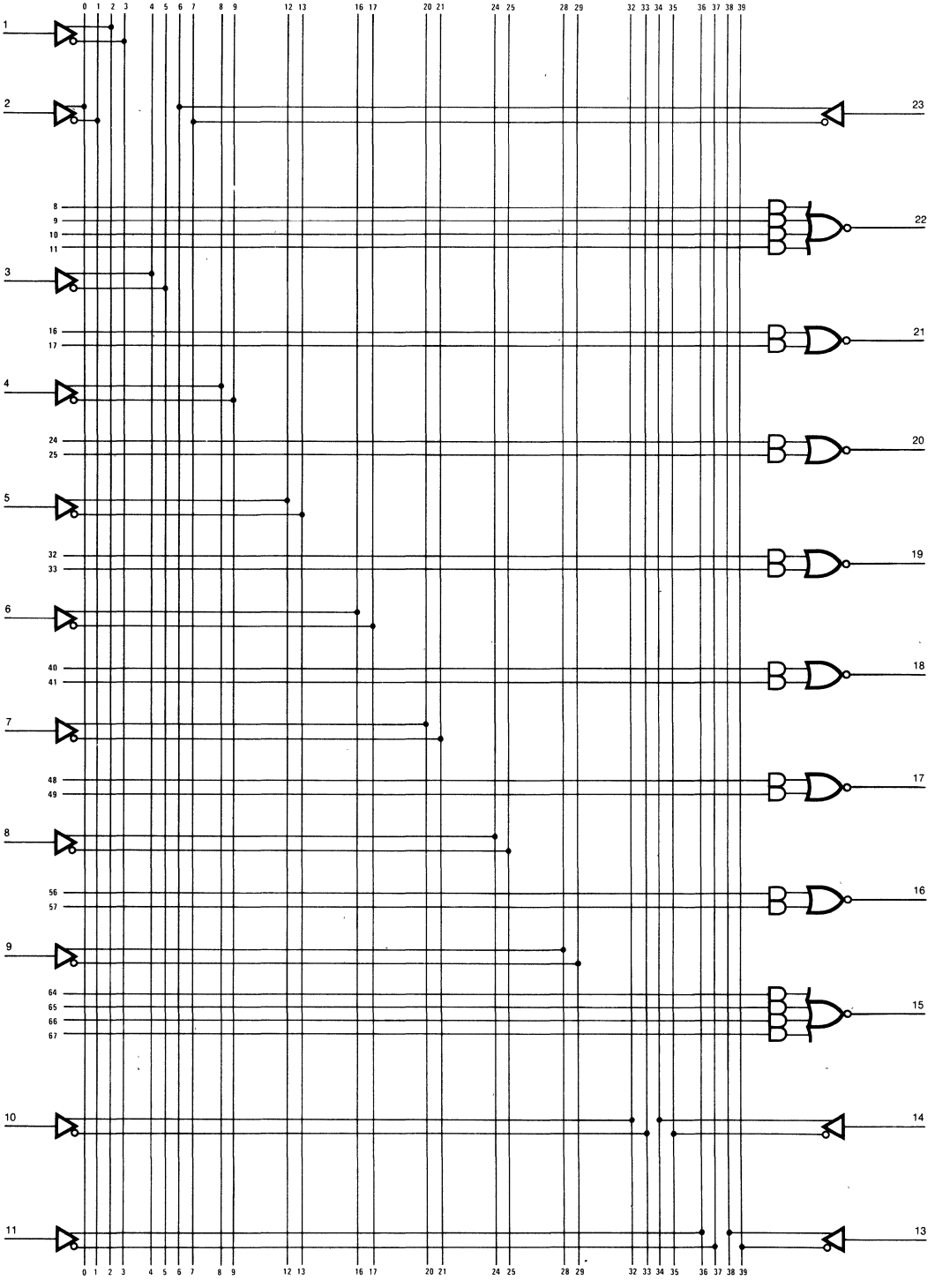
16A4


1
2
3
4
5
6
7
8
9
19
18
17
16
15
14
13
12
11
D
Q
Q̄
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

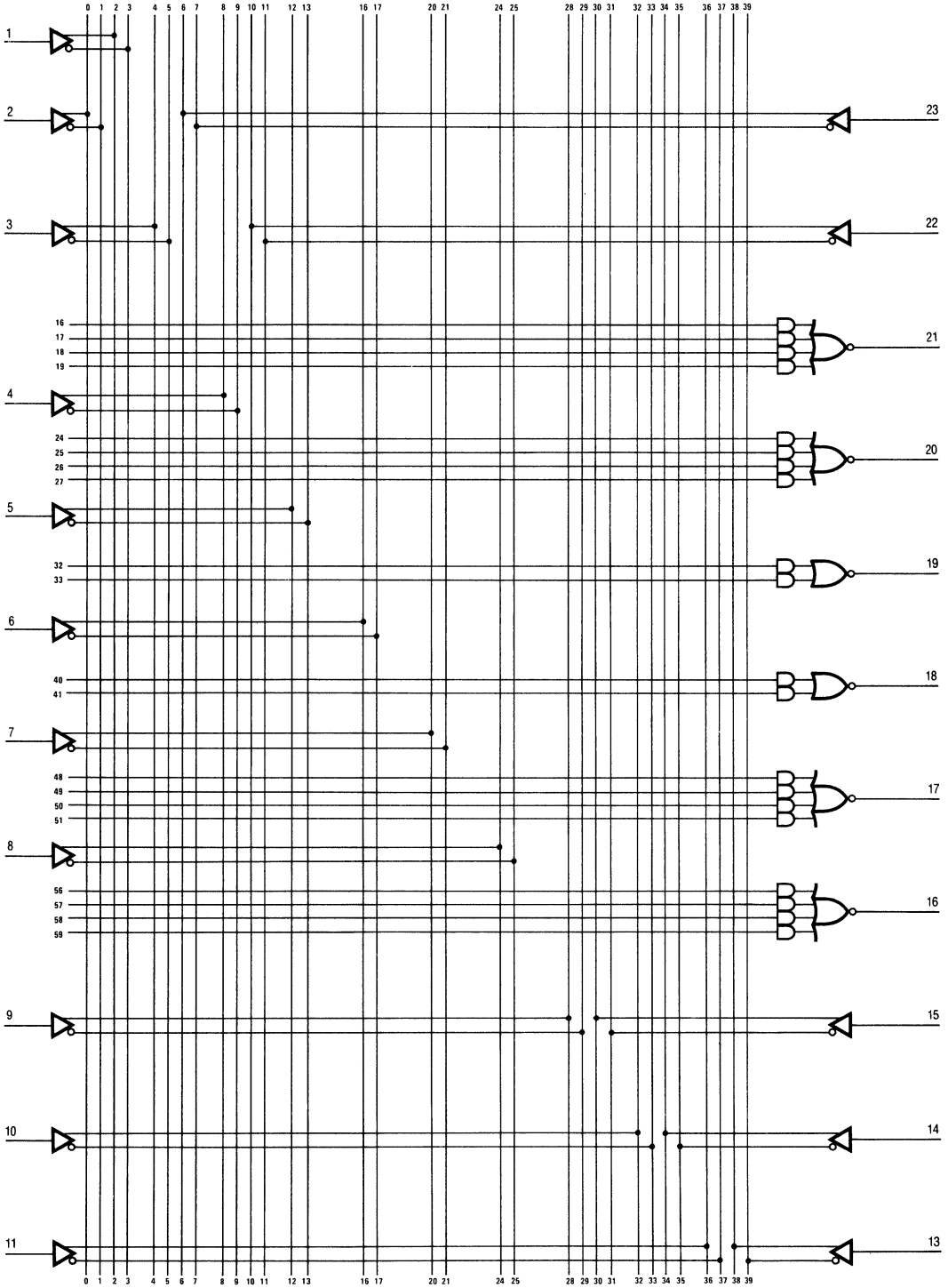
12L10


0 1 2 3 4 5 8 9 12 13 16 17 20 21 24 25 28 29 32 33 36 37 38 39
1
2
3
4
5
6
7
8
9
10
11
0
1
8
9
16
17
24
25
32
33
40
41
48
49
56
57
64
65
72
73
23
22
21
20
19
18
17
16
15
14
13

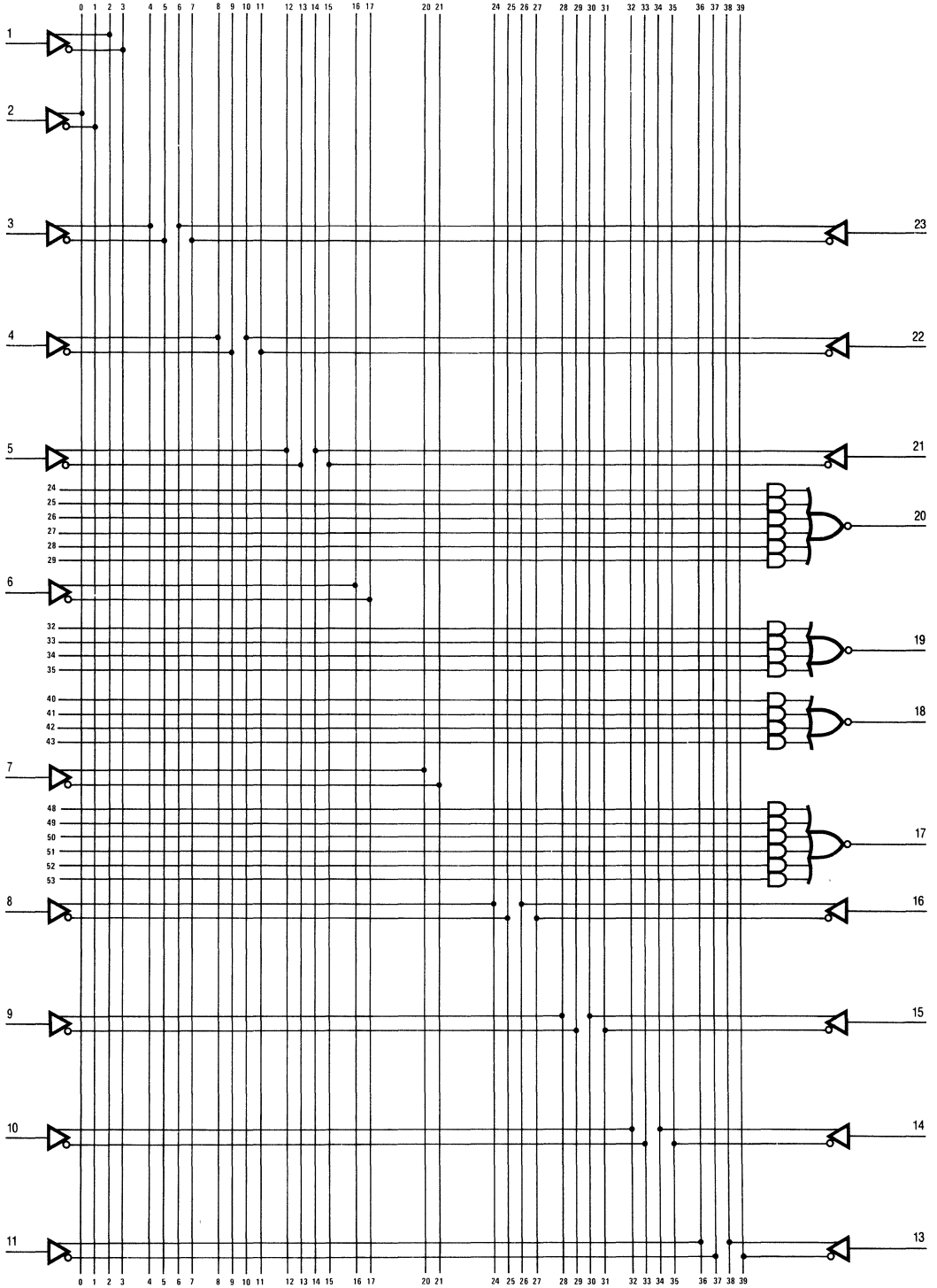
14L8


0 1 2 3
4 5 6 7
8 9
12 13
16 17
20 21
24 25
28 29
32 33 34 35
36 37 38 39
1
2
3
4
5
6
7
8
9
10
11
8
9
10
11
16
17
24
25
32
33
40
41
48
49
56
57
64
65
66
67
23
22
21
20
19
18
17
16
15
14
13

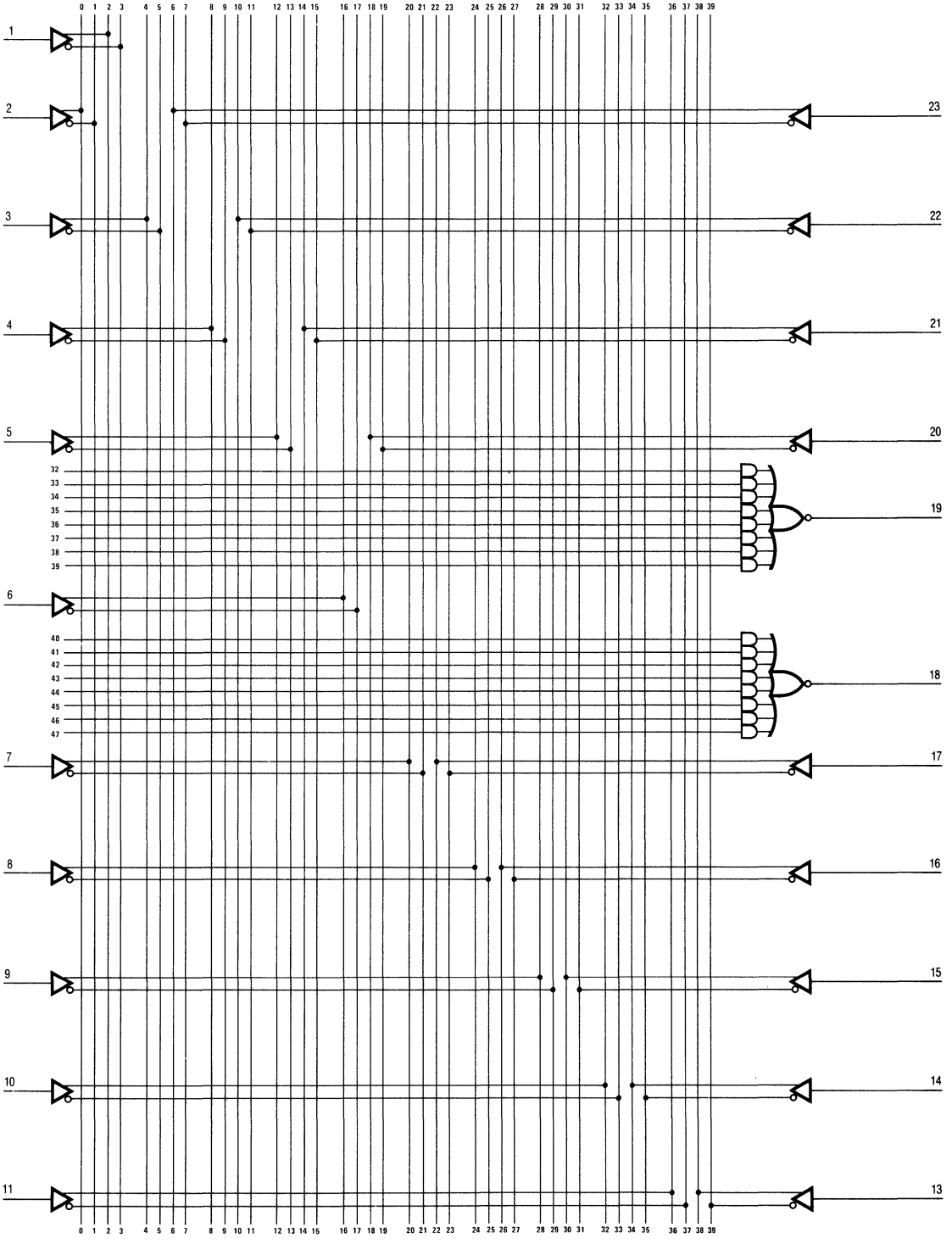
16L6


0 1 2 3 4 5 6 7 8 9 10 11 12 13 16 17 20 21 24 25 28 29 30 31 32 33 34 35 36 37 38 39
1
2
3
4
5
6
7
8
9
10
11
23
22
21
20
19
18
17
16
15
14
13
16 17 18 19
24 25 26 27
32 33
40 41
48 49 50 51
56 57 58 59

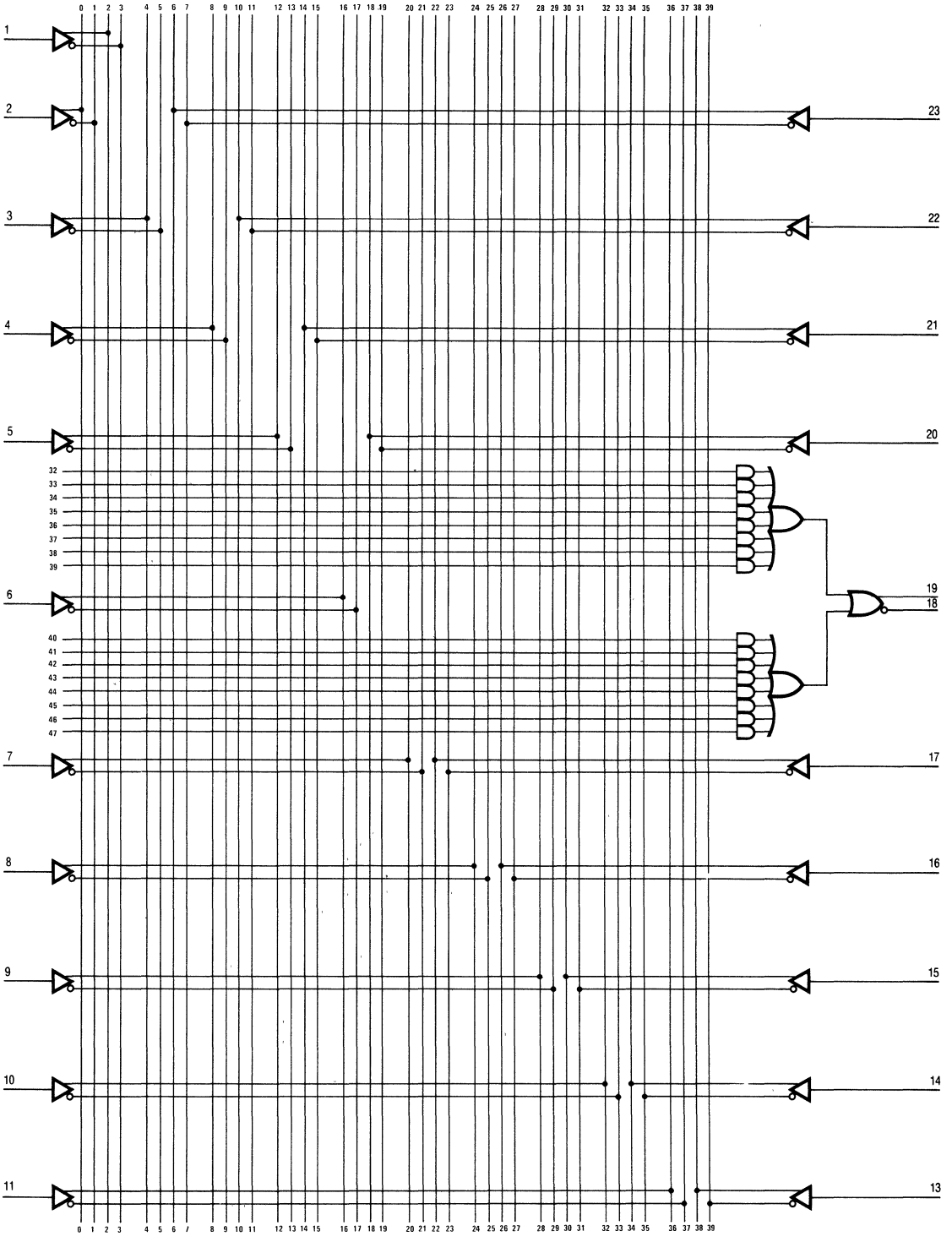
18L4


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 20 21 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
1
2
3
4
5
6
7
8
9
10
11
23
22
21
20
19
18
17
16
15
14
13
24 25 26 27 28 29
32 33 34 35
40 41 42 43
48 49 50 51 52 53

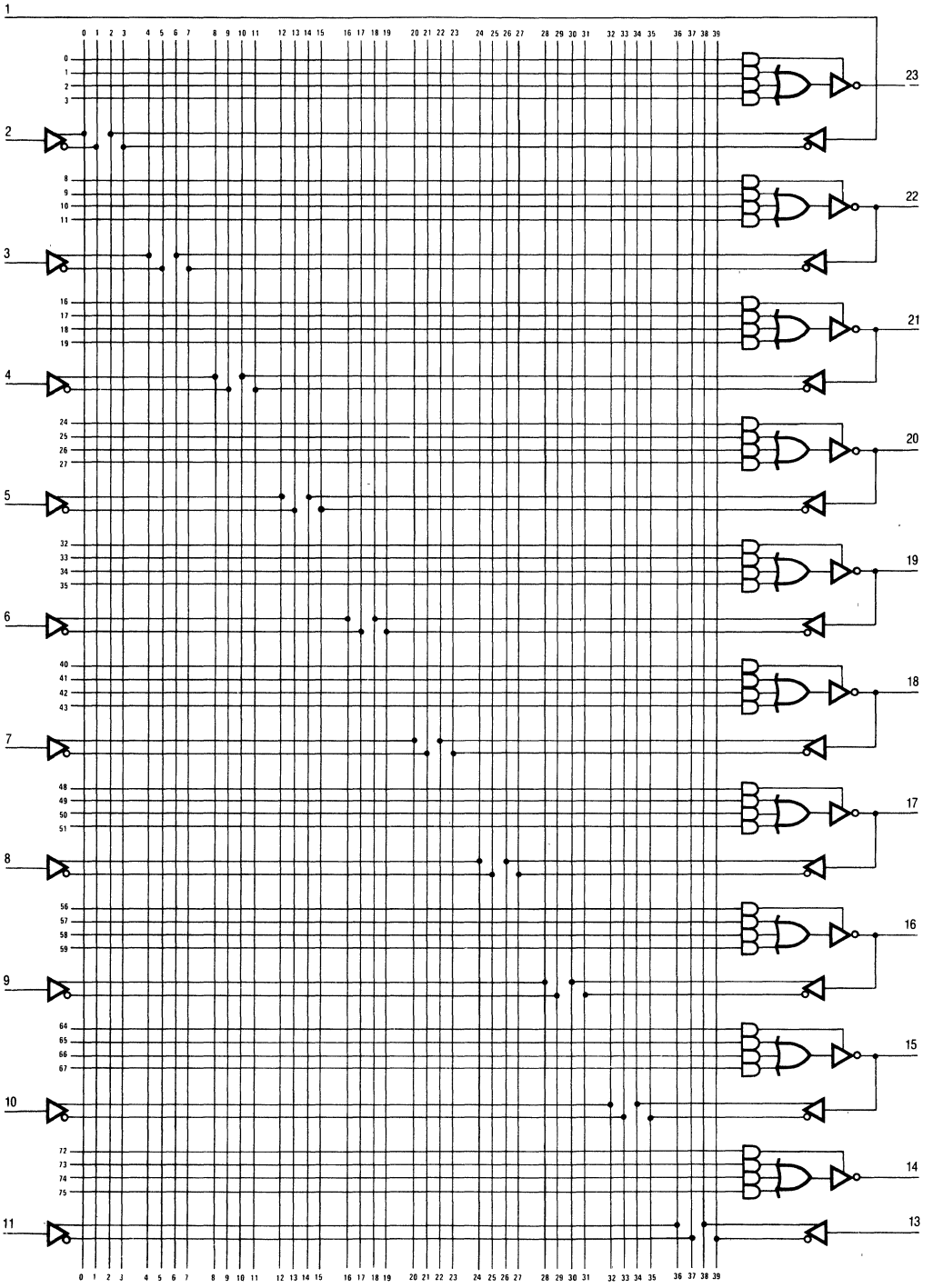
20L2


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
1
2
3
4
5
6
7
8
9
10
11
23
22
21
20
19
18
17
16
15
14
13
32 33 34 35 36 37 38 39
40 41 42 43 44 45 46 47

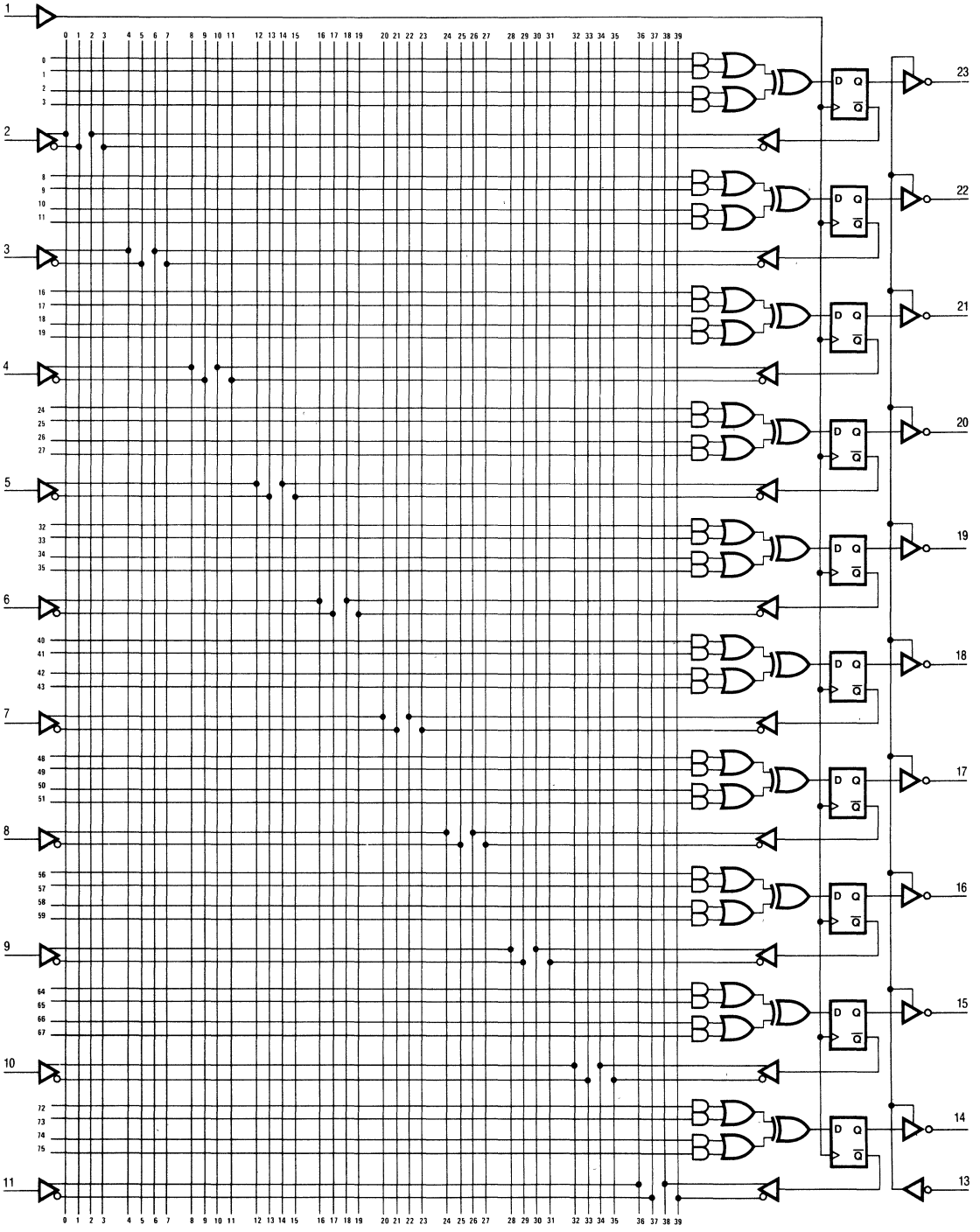
20C1


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
1
2
3
4
5
6
7
8
9
10
11
23
22
21
20
19
18
17
16
15
14
13
32 33 34 35 36 37 38 39
40 41 42 43 44 45 46 47

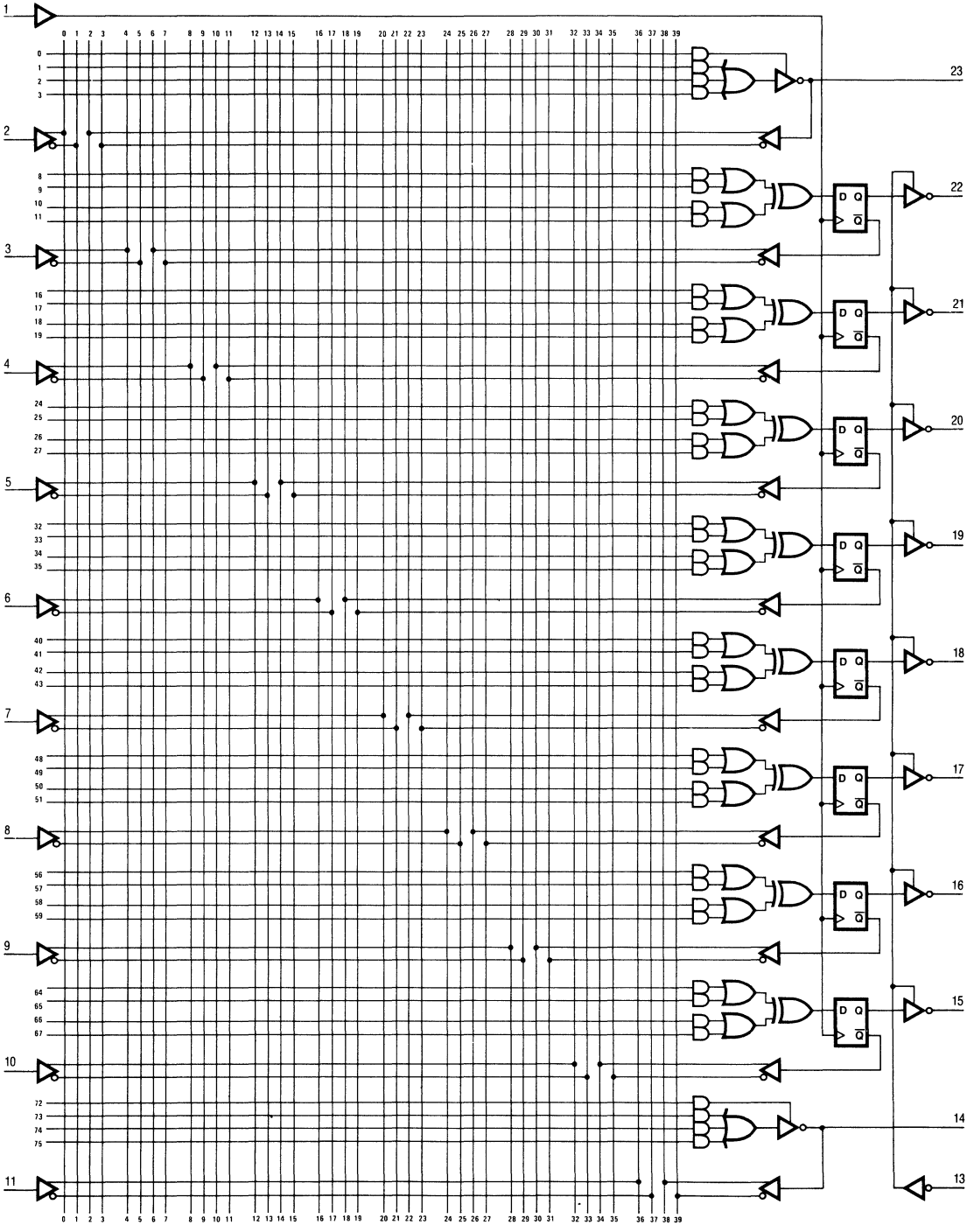
20L10


1
2
3
4
5
6
7
8
9
10
11
23
22
21
20
19
18
17
16
15
14
13
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39

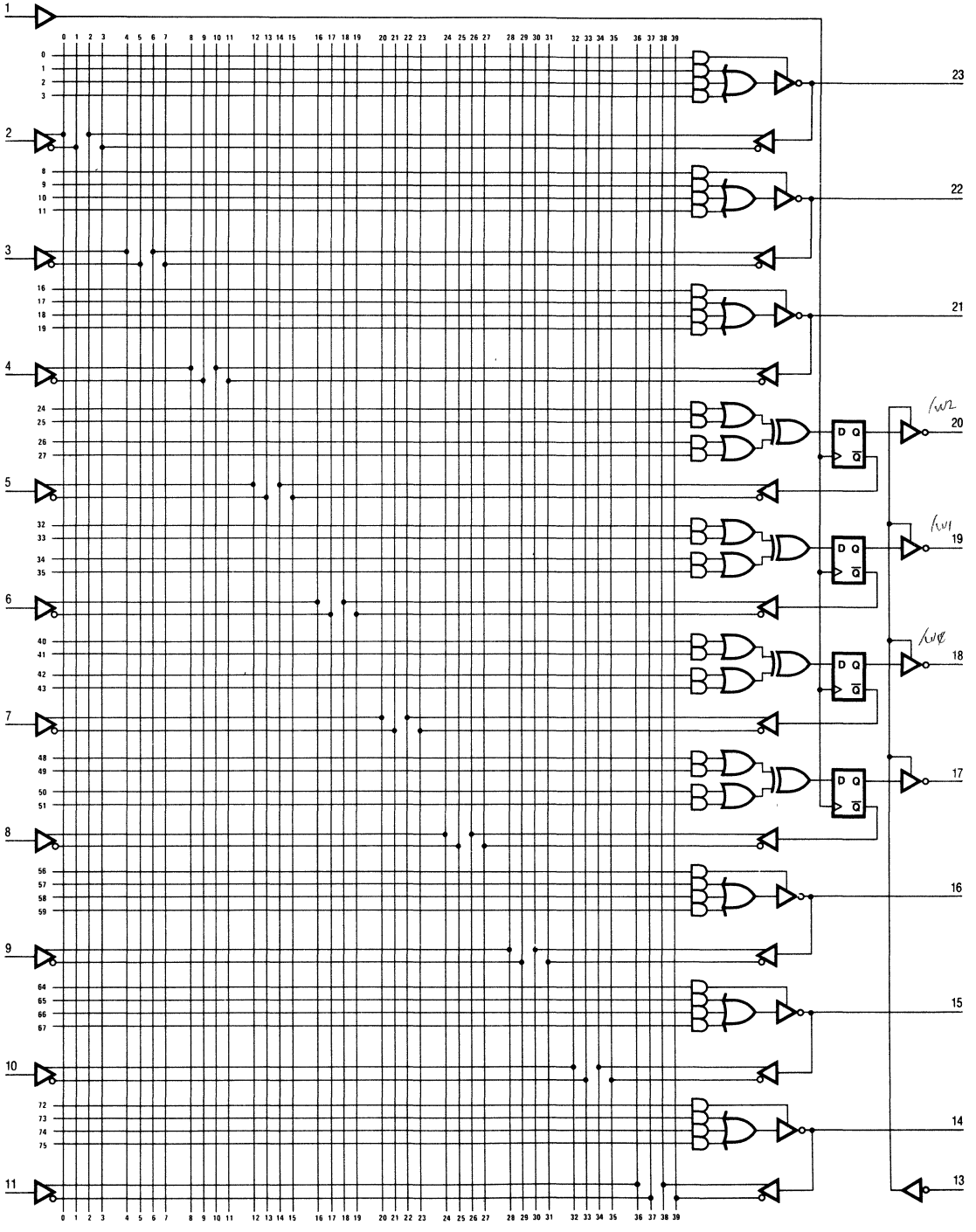
20X10


1
2
3
4
5
6
7
8
9
10
11
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
0
1
2
3
8
9
10
11
16
17
18
19
24
25
26
27
32
33
34
35
40
41
42
43
48
49
50
51
56
57
58
59
64
65
66
67
72
73
74
75
D Q
CI
23
22
21
20
19
18
17
16
15
14
13

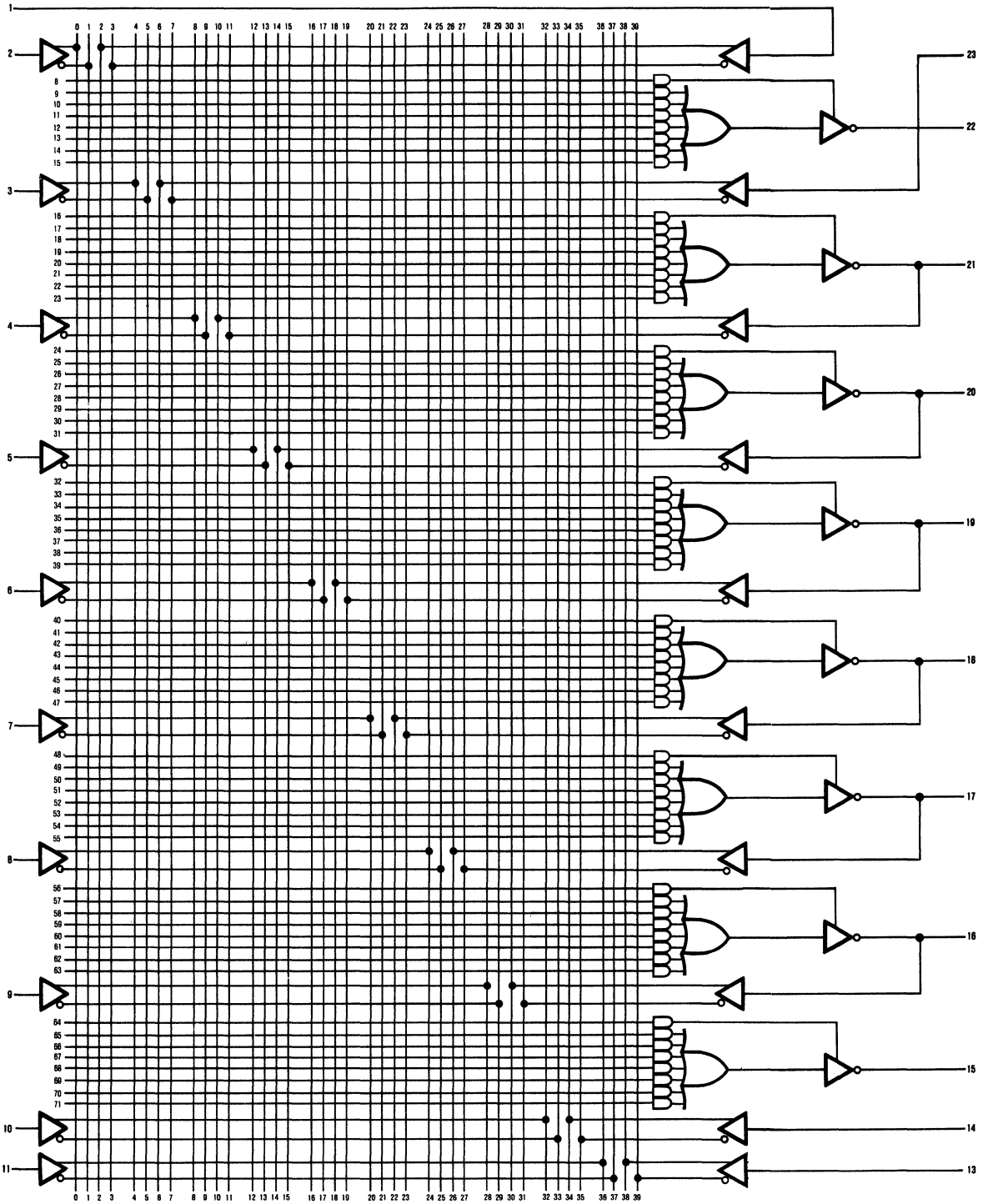
20X8



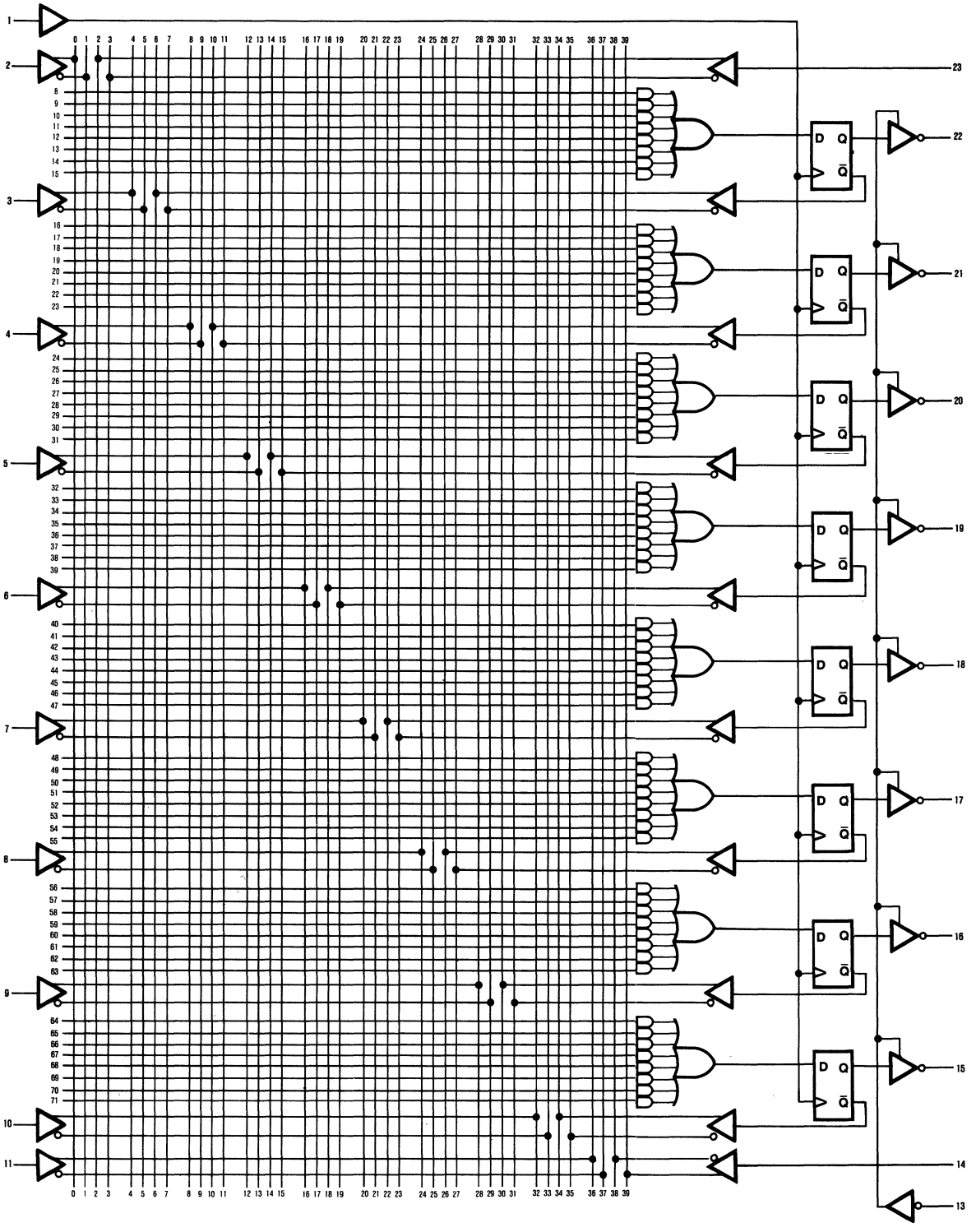
20X4



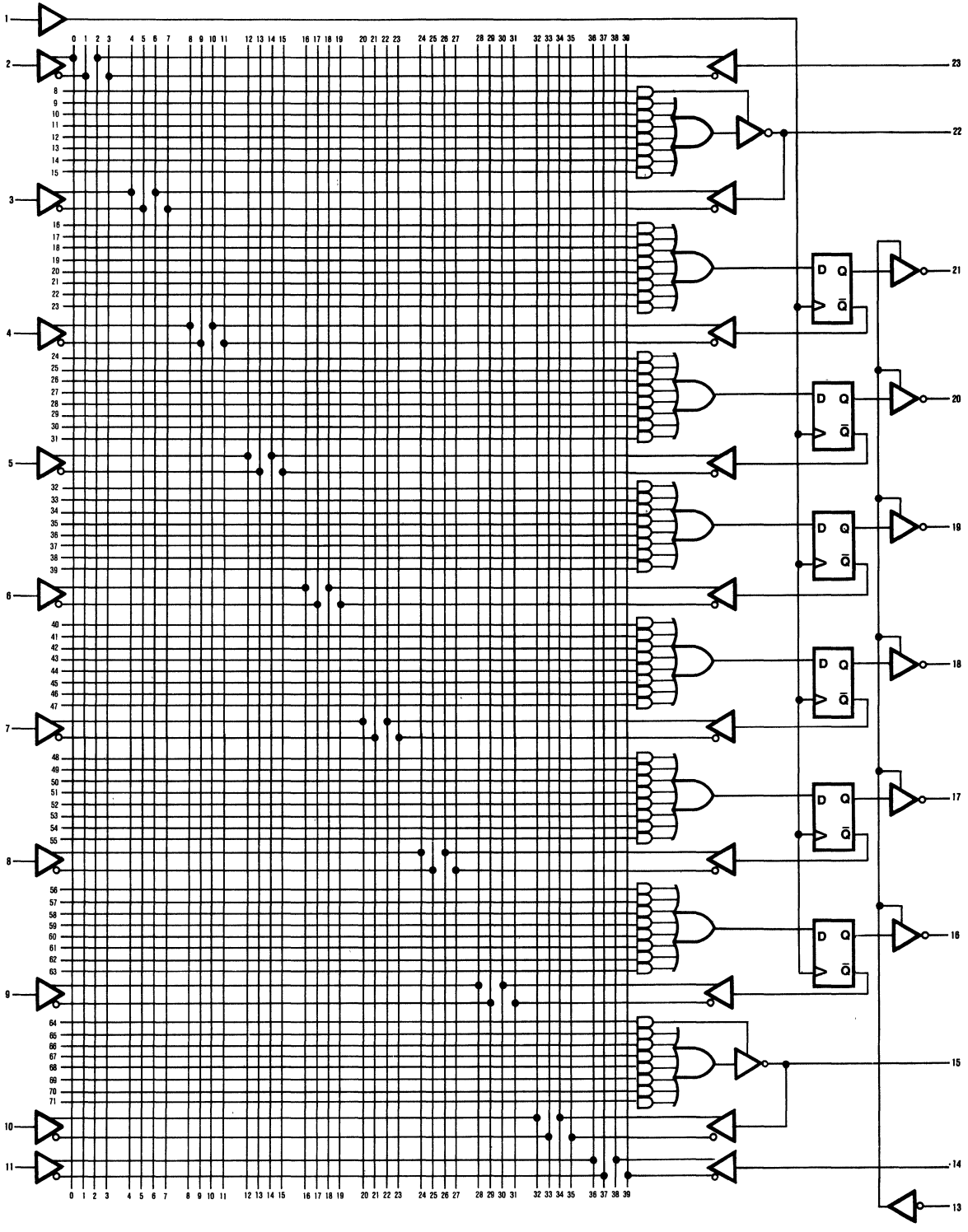
20L8


1
2
3
4
5
6
7
8
9
10
11
23
22
21
20
19
18
17
16
15
14
13
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
8 9 10 11 12 13 14 15
16 17 18 19 20 21 22 23
24 25 26 27 28 29 30 31
32 33 34 35 36 37 38 39
40 41 42 43 44 45 46 47
48 49 50 51 52 53 54 55
56 57 58 59 60 61 62 63
64 65 66 67 68 69 70 71

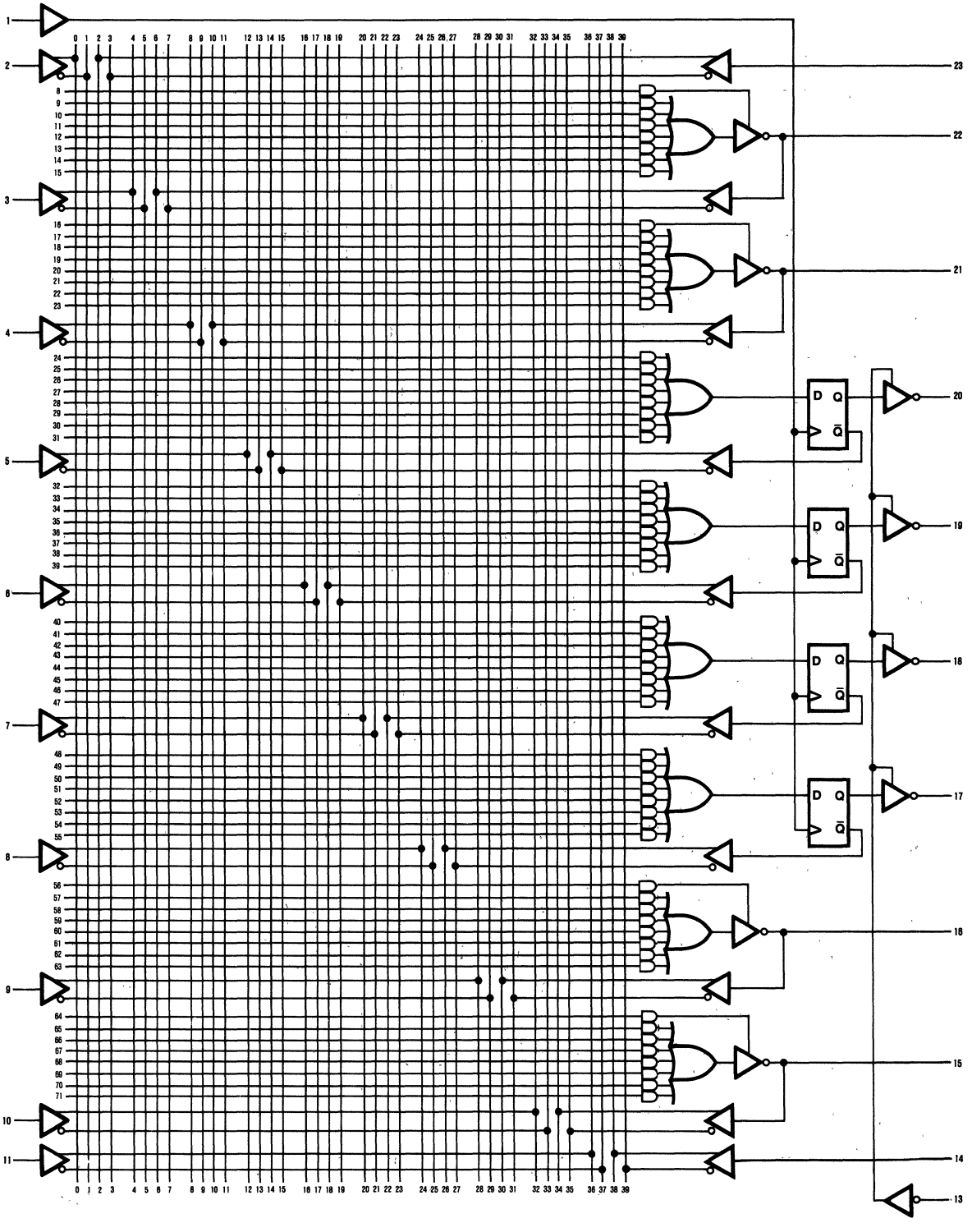
20R8


1
2
3
4
5
6
7
8
9
10
11
23
22
21
20
19
18
17
16
15
14
13
D
Q
Q̄
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39

20R6

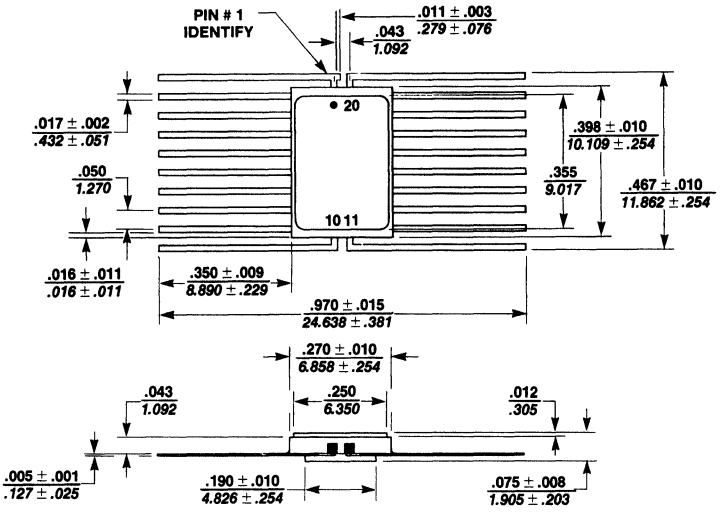

1
2
3
4
5
6
7
8
9
10
11
23
22
21
20
19
18
17
16
15
14
13
D Q
Q̄
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39

20R4



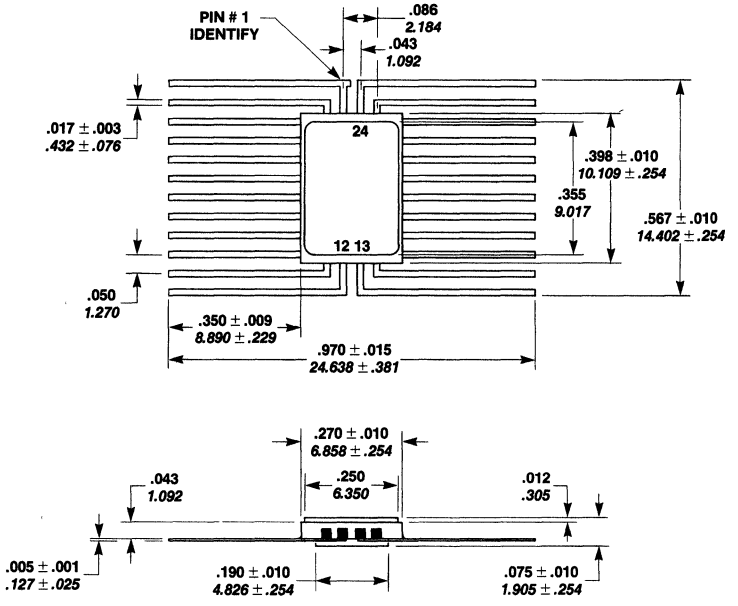
F20 Flatpack

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



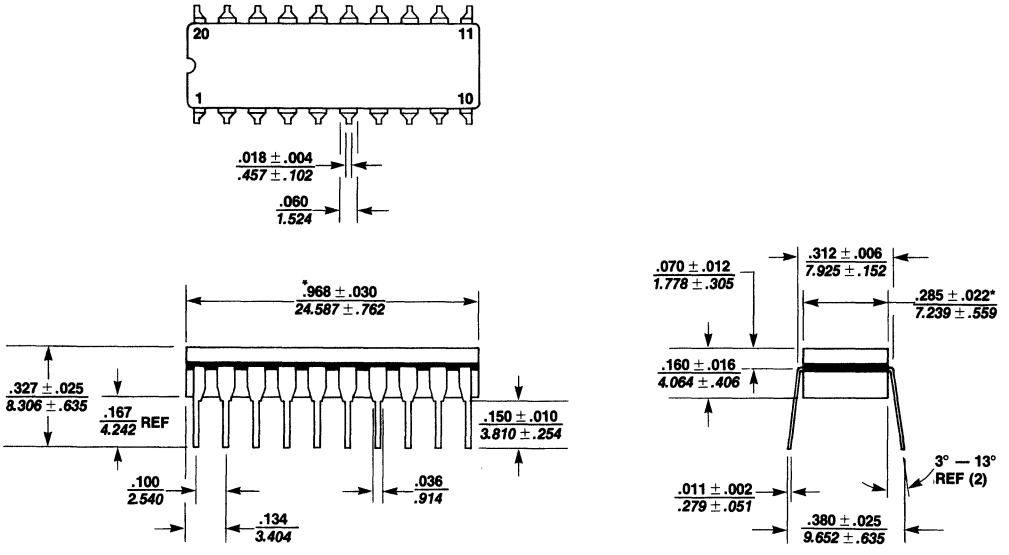
3

F24 Flatpack

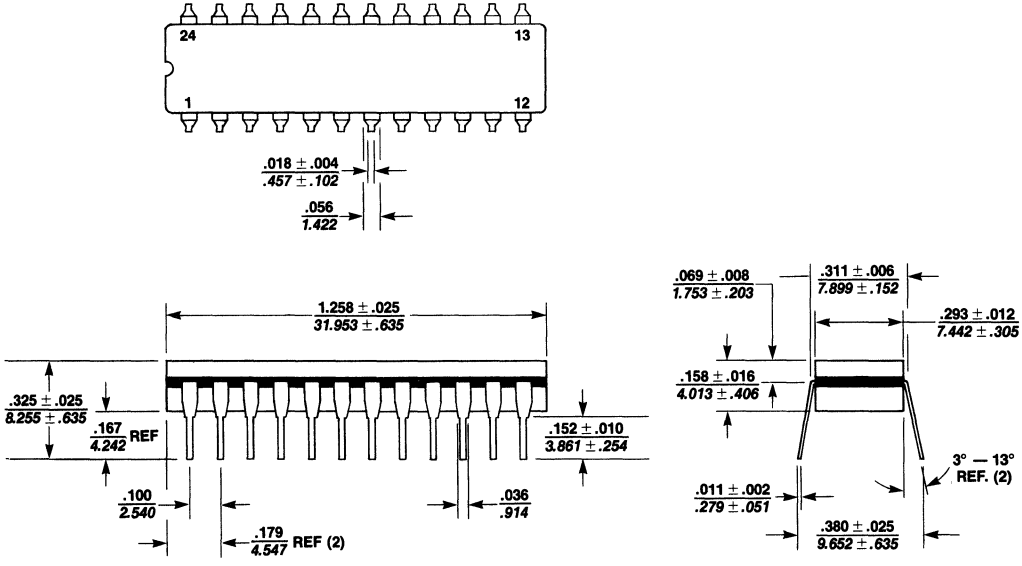


J20 Cerdip

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

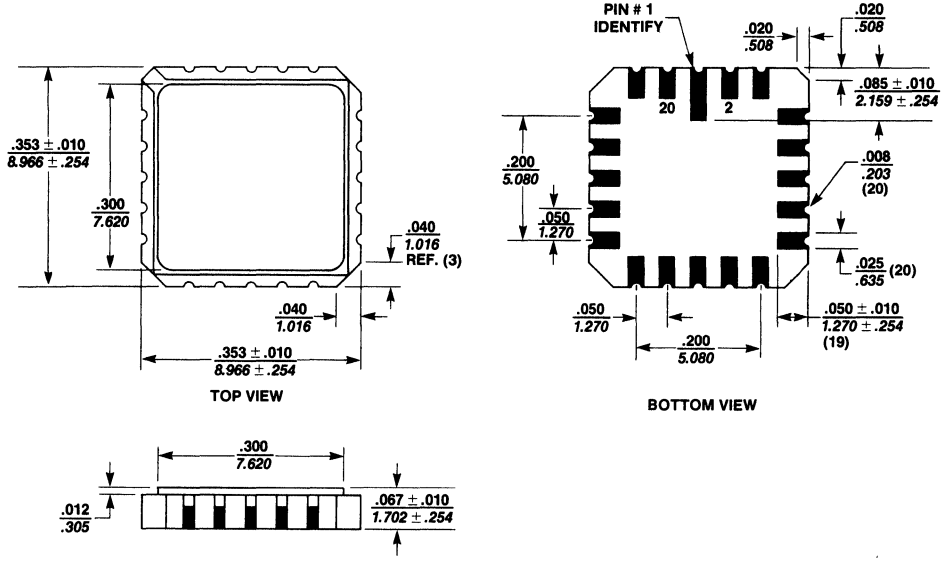


J24S Cerdip SKINNYDIP™

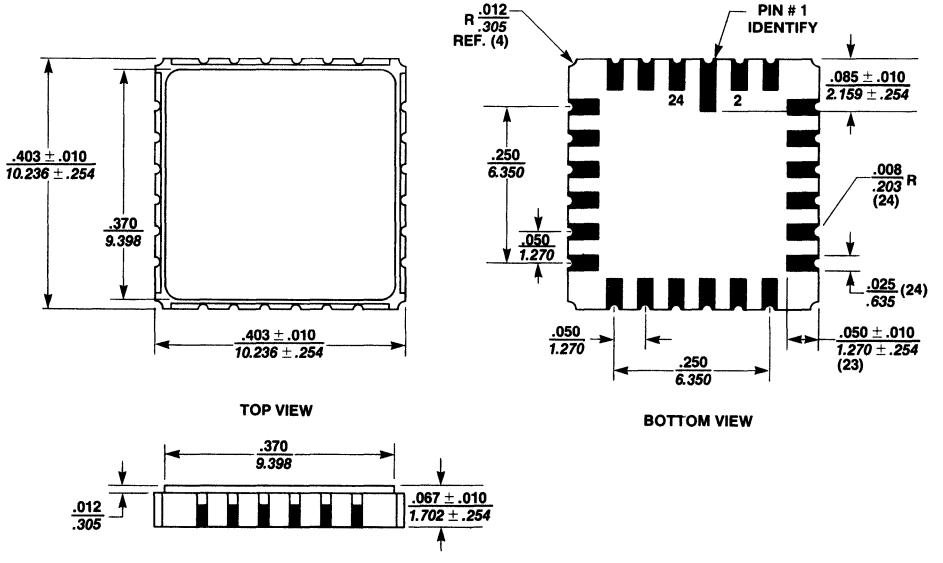


L20 Leadpack

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

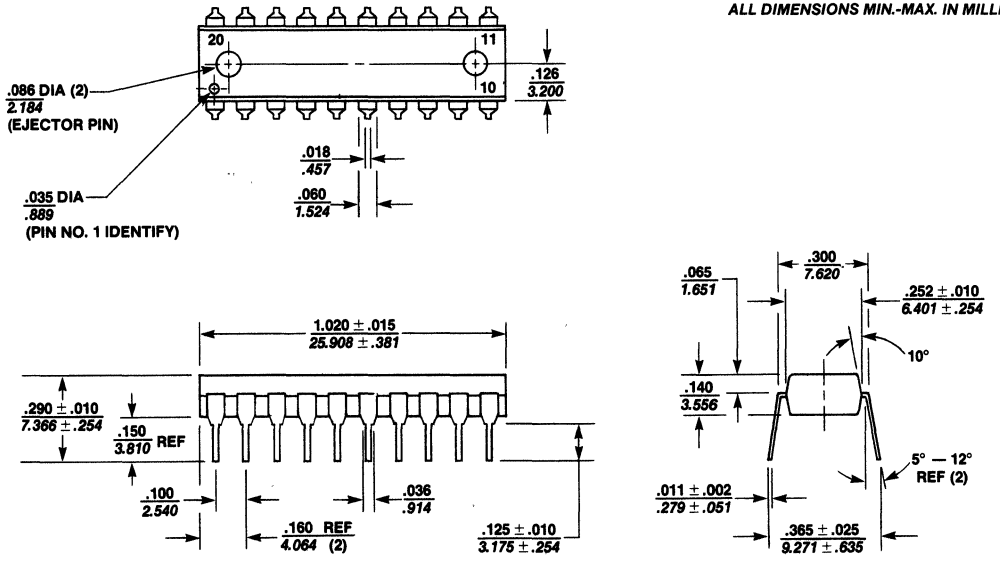


L24 Leadless

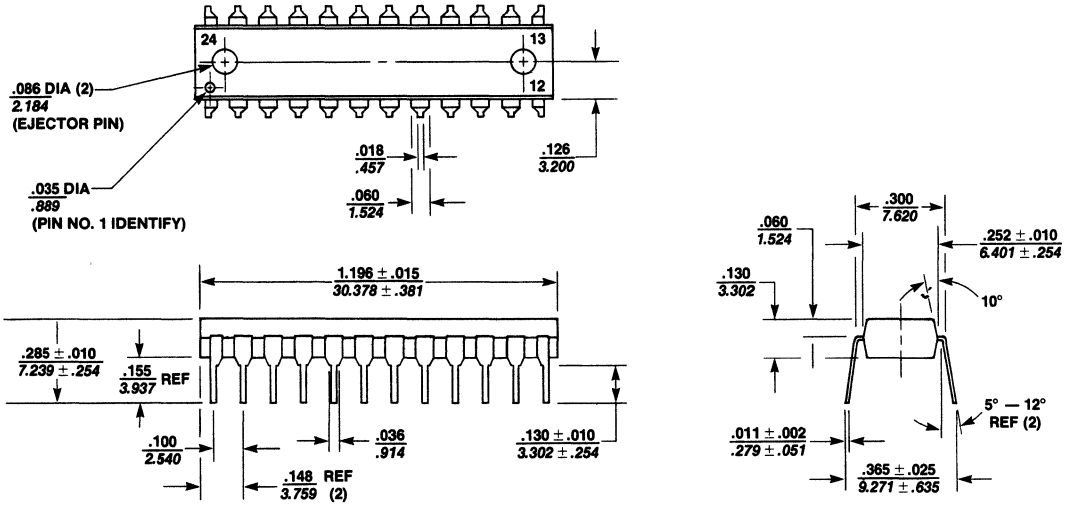


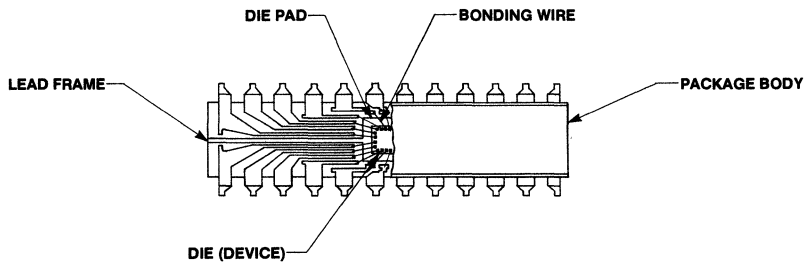
N20 Plastic

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



N24S Plastic SKINNYDIP





Selecting the Right PAL

The 29-member PAL Family offers a wide range of complexity to choose from. Starting with the PAL10H8 (10 inputs, 8 active high outputs), the first 15 PAL circuits can replace random SSI gate functions by at least 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 0.25 mA input loading (I_{IL}).

The rest of the PAL family provides the additional features of three-state outputs, programmable I/O pins, registered outputs with feedback, Exclusive OR operator, and arithmetic gated feedback.

The three-state outputs drive the standard LS output sink of 24 mA (I_{OL}) providing bus-driving capabilities.

The programmable I/O pins, allow individual product terms to directly control output data flow. The control logic can be used to either gate product terms out of the three-state buffer or to route data into the product term matrix. This bi-directional capability can be used to implement shift and rotate functions as well as programmable allocation of input and output pins.

The registered outputs allow individual product sums to be clocked into the internal D flip-flops on the rising edge of the clock. The stored data can then be gated to the output buffer by enabling the three-state buffer. The true or complement of the data can also be fed back into the array. This facilitates the construction of state machines in a single chip.

The Exclusive OR operator allows for easy implementation of the HOLD function needed in counters and other state sequencers.

The arithmetic gated feedback allows for any of the 16 possible Boolean operations to be performed between the feedback and input pin. This provides arithmetic and logic operations. A provision for carry propagation required for fast arithmetic operations is also available.

In all PAL devices 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 PAL units, thus using less parts.

Defining the Pinout

The first step in designing a PAL is selecting the pinout. The example shown below (Figure 1) shows a method for circling a section of conveniently drawn logic to define a boundary for a PAL function. This boundary will dictate a specific number of input pins and output pins. For this example, 8 inputs and 6 outputs are required, well within the capability of the PAL10L8. Assignment of inputs and outputs to specific pins can now be done using the PAL Logic Symbol as shown below.

NOTE This pinout can later be changed to suit printed circuit board

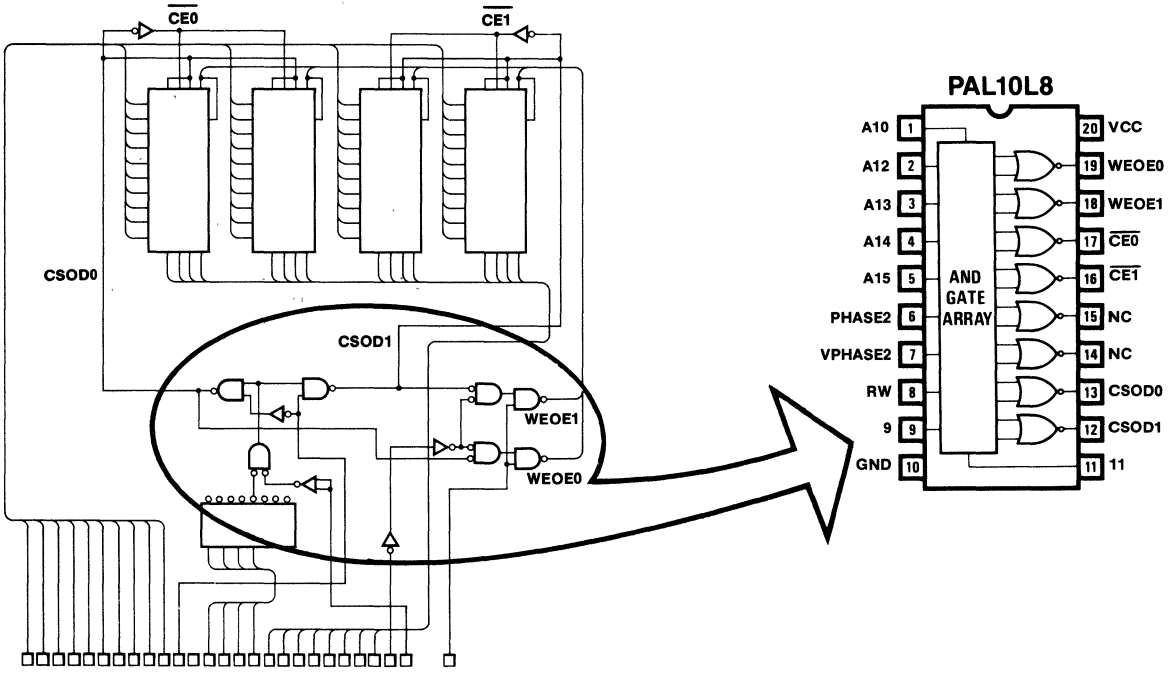


Figure 1. Using PAL to replace random logic in 6800 Microprocessor Bus

Specifying the Design

The next step is to write the Boolean logic equations (in sum of products form) which will transform the inputs into the desired outputs. These explicit logic equations specify the design precisely. They are easily simulated and edited.

Since PAL devices have predominately active low outputs, it may be necessary to apply DeMorgan's Law to change the output polarity of an equation when implementing it in a PAL. DeMorgan's Law states that the inverse or complement of any Boolean expression can be found by successively applying the following theorems:

$$\overline{(X + Y)} = \overline{X} * \overline{Y} \quad \text{and} \quad \overline{(X * Y)} = \overline{X} + \overline{Y}$$

Examples of equations written with active high and active low outputs are shown in the following table:

ACTIVE HIGH OUTPUT	ACTIVE LOW OUTPUT
$O = \overline{I1}$	$\overline{O} = I1$
$O = \overline{I1} \cdot \overline{I2}$	$\overline{O} = I1 + I2$
$O = I1 + \overline{I2}$	$\overline{O} = \overline{I1} \cdot I2$
$O = I1 + \overline{I2} \cdot I3$	$\overline{O} = \overline{I1} \cdot I2 + \overline{I1} \cdot \overline{I3}$
$O = I1 :+: I2$	$\overline{O} = I1 :\cdot: I2$

Some designers may prefer to consult the PAL Logic Diagram before writing the equations. The basic method is simply to choose the appropriate logic diagram, label the corresponding pin names, and mark the fuse interconnections needed to implement the desired logic transform function.

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 form. Actually, the unprogrammed PAL is shipped with all Xs (all fuses) intact.

The Boolean logic equations can then be read directly from the logic diagram.

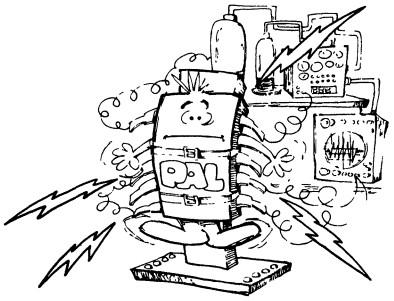
Design Verification and Documentation

An optional FUNCTION TABLE and DESCRIPTION may be included. The FUNCTION TABLE serves the dual purpose of documenting and verifying the design (using the PALASM simulator). The device operation and application are described in the DESCRIPTION.

Phantom Fuses

Phantom fuse locations are those locations where a fuse does not exist. Unprogrammed PAL with phantom fuse locations appear to a PROM programmer as being partially programmed. Since the PROM programmer will expect to verify all possible fuse locations, the PAL programming format must provide the expected pattern to verify non-existent fuse nodes. PALASM and PAL programmers tweek the fuse plot for the phantom fuses automatically so these nodes will be treated properly when verified. The following PAL circuits have phantom fuses:

PAL10H8	PAL10L8	PAL12L10
PAL12H6	PAL12L6	PAL14L8
PAL14H4	PAL14L4	PAL16L6
PAL16H2	PAL16L2	PAL18L4
	PAL16C1	PAL20L2
		PAL20C1



PALASM

PALASM is the key tool used in automating the process of "designing your own chip." PALASM is a FORTRAN IV program which assembles the PAL Design Specification translating the logic equation to a PAL fuse pattern. The fuse pattern may be generated in a format compatible with either a PAL or a PROM programmer.

PALASM also contains a simulator which exercises the Function Table vectors in the logic equations. Inconsistencies between the vectors and the equations are reported as errors. The simulator also translates the Function Table vectors to a set of universal test vectors which may be used for functional testing after the device is fabricated.

I. Boolean Algebra Review

Commutative Property: $A*B = B*A$; $A + B = B + A$

Associative Property: $A*(B*C) = (A*B)*C$; $A + (B + C) = (A + B) + C$

Distributive Property: $A*(B + C) = A*B + A*C$; $A + B*C = (A + B)*(A + C)$

Postulates:

$0 * 0 = 0$	$1 * 0 = 0$	$0 + 0 = 0$
$1 + 1 = 1$	$0 + 1 = 1$	$1 * 1 = 1$
$\overline{0} = 1$	$\overline{1} = 0$	

Theorems:

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

$\overline{(A + B + C + ...)} = \overline{A} * \overline{B} * \overline{C} * ...$ — DeMorgan

$\overline{A} + \overline{B} + \overline{C} + ... = \overline{(A * B * C * ...)}$

$\overline{(A * B * C * ...)} = \overline{A} + \overline{B} + \overline{C} + ...$ — DeMorgan

$\overline{A} * \overline{B} * \overline{C} * ... = \overline{(A + B + C + ...)}$

$A*(A + B) = A + A*B = A$

$A + A*B = A*(A + B) = A$

II. Karnaugh Map Matrix Labels

Top to bottom or left to right:

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

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

Note: * = AND + = OR / = NOT

2-Bit Counter Example

A = LSB B = MSB

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

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

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

thus:

$\overline{A} := A$; Toggle

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

$\overline{B} := \overline{B} * \overline{A} * UP$; Hold on Up
$+ \overline{B} * A * \overline{UP}$; Hold on Down
$+ B * A * UP$; Toggle Up
$+ B * \overline{A} * \overline{UP}$; Toggle Down

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

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

$\overline{B} := \overline{B} * \overline{A} * UP * \overline{PRS}$; Hold on Up
$+ \overline{B} * A * \overline{UP} * \overline{PRS}$; Hold on Down
$+ B * A * UP * \overline{PRS}$; Toggle Up
$+ B * \overline{A} * \overline{UP} * \overline{PRS}$; Toggle Down
$+ CLR$; Clear

To add a load function merely requires another product line.

$\overline{B} := D1 * * \overline{LD}$; Load
$+ \overline{B} * \overline{A} * UP * \overline{PRS} * \overline{LD}$; Hold on Up
$+ \overline{B} * A * \overline{UP} * \overline{PRS} * \overline{LD}$; Hold on Down
$+ B * A * UP * \overline{PRS} * \overline{LD}$; Toggle Up
$+ B * \overline{A} * \overline{UP} * \overline{PRS} * \overline{LD}$; Toggle Down
$+ CLR$; Clear

Verbal Logic Language

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

```
/Q3 := /D3 * LD                    ; Load
    + Q3 * /LD                     ; Hold
  :+: UP * /LD * Q2 * Q1 * Q0      ; Toggle Up
    + /UP * /LD * /Q2 * /Q1 * /Q0  ; Toggle Down
```

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

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

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

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

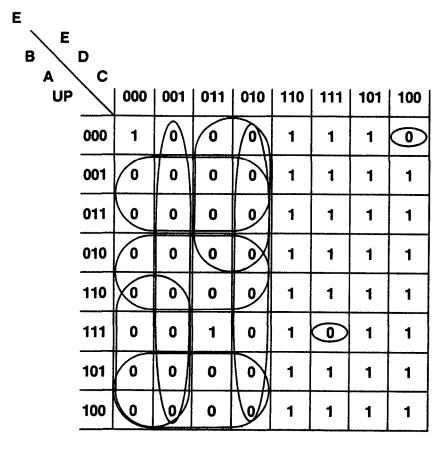
This same function could be written without the aid of the XOR gate. However, it would require more product terms. More on this when we get to the power of the XOR gate.

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

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

The Power of the Exclusive-OR

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



EDCBA

```
00000
00001
00010
00011
00100
00101
00110
00111
01000
01001
01010
01011
01100
01101
01110
01111
10000
10001
10010
10011
10100
10101
10110
10111
11000
11001
11010
11011
11100
11101
11110
11111
```

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

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

The equation for this output is as follows:

```
/E := /E * /B * UP                 ; 1 (7 Holds)
    + /E * A * /UP                 ; 2
    + /E * B * /A                  ; 3
    + /E * D * /B                  ; 4
    + /E * /D * B                  ; 5
    + /E * /D * C                  ; 6
    + /E * D * /C                  ; 7
    + E * D * C * B * A * UP       ; 8 (Toggle Up)
    + E * /D * /C * /B * /A * /UP  ; 9 (Toggle Down)
```

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

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

```
/E := /E                           ; Hold State
```

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

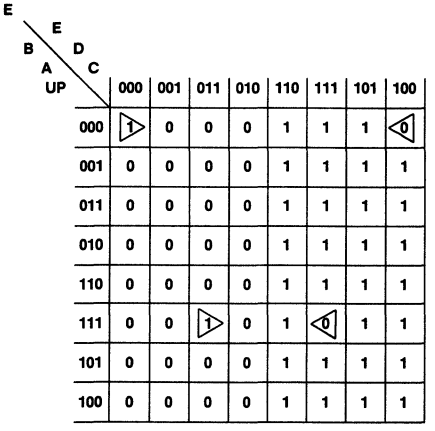
Closer examination reveals that the four cells actually reduce to two pairs of cells. These pairs are:

```
D * C * B * A * UP        AND
/D * /C * /B * /A * /UP
```

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

```
/E := /E                           ; Hold
  :+: D * C * B * A * UP           ; Toggle Up
    + /D * /C * /B * /A * /UP      ; Toggle Down
```

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



PAL Design Specification

The PAL Design Specification is the input file used with PALASM. It is also the recommended data sheet format for describing the function of a PAL once it has acquired the unique personality of a particular fuse pattern. The format for the PAL Design Specification as shown on the opposite page is:

Line 1 PAL part number left justified followed by PAL DESIGN SPECIFICATION

Line 2 User's part number followed by originator's name and the date

Line 3 Device application name

Line 4 User's company name, city, state

Line 5 **Pin List.**
The pin list is a sequence of symbolic names separated by one or more spaces on one or more lines in order of the device pin numbers. Each symbolic name is unique (except for unused pins which may have the same name.) All pins including power and ground must be named. Names may use any printable character except the operator: "=:*+/()". The prefix "/", may be used to logically complement the name.

Line m **Equations.**
The transfer function of the device is expressed in the following three forms:

1. SYMBOL = EXPRESSION
2. IF (PRODUCT) SYMBOL = EXPRESSION
3. SYMBOL: = EXPRESSION

The following terms are used to construct the equations:

SYMBOL	Pin name with optional prefix, "/".
PRODUCT	A sequence of SYMBOLS separated by the AND operator, "*".
IF	Conditional equality, when the PRODUCT is logically true. Otherwise, high impedance (high-Z).
EXPRESSION	A sequence of SYMBOLS separated by operators.

OPERATORS (in hierarchy of evaluation)

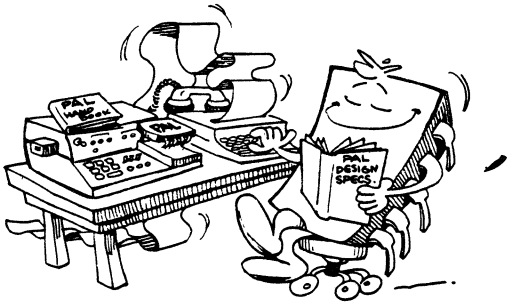
;	Comment follows
/	Complement, prefix to a pin name.
*	AND (product)
+	OR (sum)
:+:	XOR (exclusive OR)
:*:	XNOR (exclusive NOR)
( )	Conditional three-state (IF STATEMENT) or fixed symbol
=	Equality
:=	Replaced by after the low to high transition of the clock.

Line n **Function Table.** (optional)
The function table begins with the keyword, "FUNCTION TABLE." It is followed by a pin list which may be in a different order and polarity from the pin list in Line 5. VCC and GND cannot be listed. The pin list is followed by a dashed line; e.g., ---------- (length optional), which in turn is followed by a list of vectors, one vector per line. One state must be specified for each pin name and optionally separated by spaces. A vector is a sequence of states listed in the same order as the pin list and followed by an optional comment. The vector list is followed by another dashed line.

Definition of Function Table States:

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

Line o **Description.** (Optional if following Function Table).
This section begins with the keyword, "DESCRIPTION." The device operation and application are described here.



```
PAL10L8                                                PAL DESIGN SPECIFICATION 1
P00123                                                   VINCENT COLI 07/08/81 2
EXAMPLE                                                                        3
MMI SUNNYVALE, CALIFORNIA                                                      4
A B C NC NC NC NC NC NC GND NC NC NC NC NC NC NC /F NC VCC                      5


F = A*B + C       ;F EQUALS A AND B OR C                                       m


FUNCTION TABLE                                                                 n

A B C F

;ABC F  COMMENT
-------------------------
 LLL L  ALL LOWS
 LHL L  TEST AND GATE LOW
 HLL L  TEST AND GATE LOW
 HHL H  TEST AND GATE
 XXH H  TEST OR GATE HIGH
-------------------------


DESCRIPTION                                                                    o

THIS EXAMPLE ILLUSTRATES THE FORMAT OF THE PAL DESIGN SPECIFICATION.
```

PAL DESIGN SPECIFICATIO

4

```
EXAMPLE

 1 000XXXXXXXXXXXXXHXl
 2 010XXXXXXXXXXXXXHXl
 3 100XXXXXXXXXXXXXHXl
 4 110XXXXXXXXXXXXXLXl
 5 XXlXXXXXXXXXXXXXLXl

PASS SIMULATION
```

FUNCTION TABLE SIMULATION

```
EXAMPLE

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


 8 X-X- --   --   --   --   --   --   ---- A*B
 9 ---- X-   --   --   --   --   --   ---- C


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

NUMBER OF FUSES BLOWN =   61
```

BRIEF FUSE PLOT

```
PAL16R6 <---- PAL PART NUMBER (MUST START ON LINE 1, COLUMN 1)     PAL DESIGN SPECIFICATION
PART NO XXX <---- USER'S PART NUMBER (LINE 2)                          AUTHOR'S NAME  DATE
DEVICE APPLICATION NAME <---- NAME OF DEVICE (LINE 3)
COMPANY, CITY, STATE <---- USER'S COMPANY NAME, CITY, STATE (LINE 4)
CLK  I0  D0 D1 D2 D3 D4 D5  I1  GND  }  <-- PIN LIST (MUST START ON LINE 5)
/OC RILO Q5 Q4 Q3 Q2 Q1 Q0 LIRO VCC  }      CONSISTS OF 20 (24) SYMBOLIC NAMES
                                            WHICH ARE CONSECUTIVELY ASSIGNED
                                            TO PINS 1 THRU 20 (24)

/Q0  := /I1*/I0*/Q0 + /I1* I0*/Q1 + I1*/I0*/LIRO + I1* I0*/D0
/Q1  := /I1*/I0*/Q1 + /I1* I0*/Q2 + I1*/I0*/Q0 + I1* I0*/D1
/Q2  := /I1*/I0*/Q2 + /I1* I0*/Q3 + I1*/I0*/Q1 + I1* I0*/D2
/Q3  := /I1*/I0*/Q3 + /I1* I0*/Q4 + I1*/I0*/Q2 + I1* I0*/D3
/Q4  := /I1*/I0*/Q4 + /I1* I0*/Q5 + I1*/I0*/Q3 + I1* I0*/D4
/Q5  := /I1*/I0*/Q5 + /I1* I0*/RILO + I1*/I0*/Q4 + I1* I0*/D5
                                                          } EQUATIONS
IF(/I1*I0) /LIRO = /Q0            ;LEFT IN RIGHT OUT
IF(I1*/I0) /RILO = /Q5            ;RIGHT IN LEFT OUT
      <-- CONDITIONAL THREE-STATE
FUNCTION TABLE <---- KEYWORD (MUST START IN COLUMN 1)

I1 I0 D5 D4 D3 D2 D1 D0 CLK /OC RILO LIRO Q5 Q4 Q3 Q2 Q1 Q0 <---- FUNCTION TABLE PIN LIST

;        D IN                                    Q OUT                  }  OPTIONAL COMMENTS
;INST    D5--D0    CLK   /OC   RILO   LIRO   Q5--Q0    COMMENTS         }
-------------------------------------------------------------------------  <-- DASHED LINE
  HH     LLLLLL    C     L     Z      Z      LLLLLL    LOAD ZEROS             (LENGTH OPTIONAL)
  LL     XXXXXX    C     L     Z      Z      LLLLLL    HOLD
  HL     XXXXXX    C     L     L      H      LLLLLH    SHIFT LEFT IN A H   <-- FUNCTION TABLE VECTORS
  HL     XXXXXX    C     L     L      L      LLLLHL    SHIFT LEFT IN A L       ONE VECTOR PER
  HL     XXXXXX    C     L     L      L      LLLHLL    SHIFT LEFT IN A L       LINE FOLLOWED BY
  HL     XXXXXX    C     L     L      L      LLHLLL    SHIFT LEFT IN A L       OPTIONAL COMMENTS.
  HL     XXXXXX    C     L     L      L      LHLLLL    SHIFT LEFT IN A L       THE STATES (H, L, X, C, Z)
  HL     XXXXXX    C     L     H      L      HLLLLL    SHIFT LEFT IN A L       CORRESPOND TO THE
  HL     XXXXXX    C     L     L      L      LLLLLL    SHIFT LEFT IN A L       ORDER AND POLARITY
  HH     HHHHHH    C     L     Z      Z      HHHHHH    LOAD ONES               OF THE FUNCTION
  LL     XXXXXX    C     L     Z      Z      HHHHHH    HOLD                    TABLE PINLIST.
  LL     XXXXXX    X     H     Z      Z      ZZZZZZ    TEST HI-Z
-------------------------------------------------------------------------  <-- DASHED LINE
                                                                               (LENGTH OPTIONAL)

DESCRIPTION <---- KEYWORD, OPTIONAL IF FOLLOWING FUNCTION TABLE (MUST START COLUMN 1)
THIS PARAGRAPH DESCRIBES THE OPERATION AND APPLICATION OF THE DEVICE.
```

PAL Design Specification Example

PAL Legend

Constants

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

Operators (IN HIERARCHY OF EVALUATION)

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

Equations

Standard $\dot{O}_1 = I_1 \overline{I_2} + \overline{I_1} I_2$

PALASM `O1 = I1*/I2 + /I1*I2`

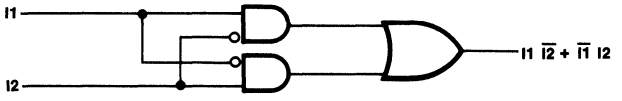
Function Table States

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

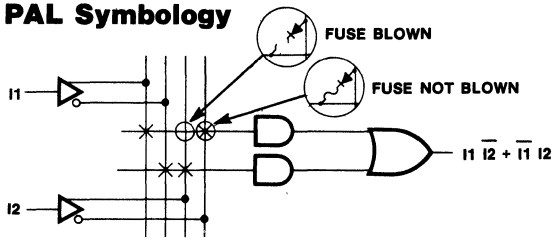
Test Conditions

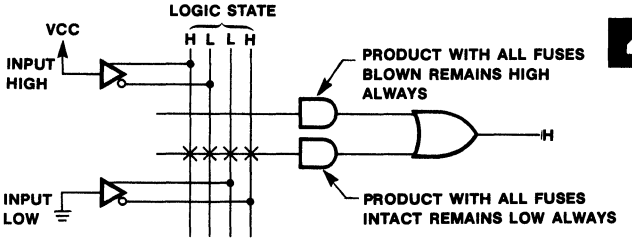
```
H = TEST HIGH     1 = DRIVE HIGH     C = DRIVE INPUT FROM LOW TO HIGH
L = TEST LOW      0 = DRIVE LOW      Z = TEST FOR HIGH IMPEDANCE
X = IRRELEVANT
```

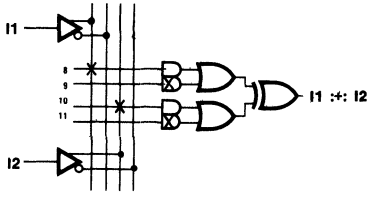
Conventional Symbology

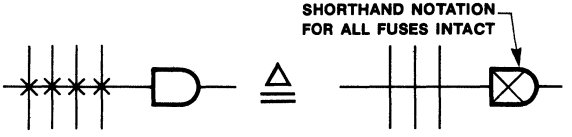


PAL Symbology

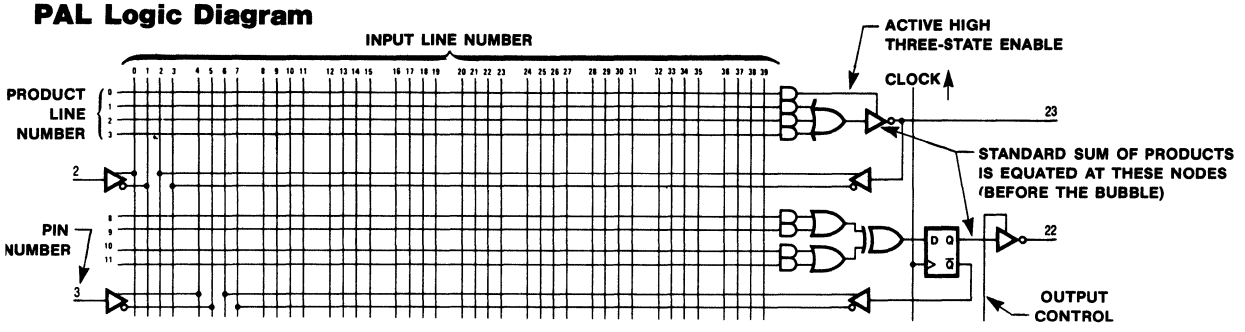








PAL Logic Diagram



Using PAL16L8 for Flip-Flops

There are two kinds which are normally discussed — the conventional 3-latch flip-flop and the master-slave 2-latch flip-flop. If PAL circuits are used for designing flip-flops, we need one output for every latch. A conventional 3-latch DFF is typically constructed as in Figure 1, whereas a master-slave DDF is shown in Figure 2.

The PAL equations for a design using 16L8 modified from Figure 1 are:

```
IF(VCC) A  = /PRE*/D* A
           + /PRE*/B* A
           + /CLK
           +  CLR

IF(VCC) B  = /A
           + /CLK
           + /CLR* D* B

IF(VCC) /Q =  A*/PRE*/Q
           + /B
           +  CLR
```

and that of the design as modified from Figure 2 are:

```
IF(VCC) Y  = /PRE*/CLK* D
           + /PRE* D* Y
           + /PRE* Y* CLK
           +  CLR

IF(VCC) /Q = /PRE* CLK*/Y
           + /PRE*/Y*/Q
           + /PRE*/Q*/CLK
           +  CLR
```

For JKFFF, we can replace D by J*Q + K*Q and D by J*Q + K*Q to get the corresponding equations:

3-latch solution:

```
IF(VCC) A  = /PRE* A*/J*/Q
           + /PRE* A* K* Q
           + /PRE* A*/B
           + /CLK
           +  CLR

IF(VCC) B  = /A
           +  J*/Q*/CLR* B
           + /K* Q*/CLR* B
           + /CLK

IF(VCC) /Q =  A*/PRE*/Q
           + /B
           +  CLR
```

2-latch solution:

```
IF(VCC) Y  = /PRE*/CLK* J*/Q
           + /PRE*/CLK*/K* Q
           + /PRE* CLK* Y
           + /PRE* J*/Q* Y
           + /PRE*/K* Q* Y
           +  CLR

IF(VCC) /Q = /PRE* CLK*/Y
           + /PRE*/Y*/Q
           + /PRE*/Q*/CLK
           +  CLR
```

For TFF, we can replace D by T*Q + T*Q and D by T*Q + T*Q to get the corresponding equations:

3-latch solution:

```
IF(VCC) A  = /PRE* A*/T*/Q
           + /PRE* A* T* Q
           + /PRE* A*/B
           + /CLK
           +  CLR

IF(VCC) B  = /A
           +  T*/Q*/CLR* B
           + /T* Q*/CLR* B
           + /CLK

IF(VCC) /Q =  A*/PRE*/Q
           + /B
           +  CLR
```

2-latch solution:

```
IF(VCC) Y  = /PRE*/CLK* T*/Q
           + /PRE*/CLK*/T* Q
           + /PRE* CLK* Y
           + /PRE* T*/Q* Y
           + /PRE*/T* Q* Y
           +  CLR

IF(VCC) /Q = /PRE* CLK*/Y
           + /PRE*/Y*/Q
           + /PRE*/Q*/CLK
           +  CLR
```

The corresponding diagrams of these equations are Figures 3-8.

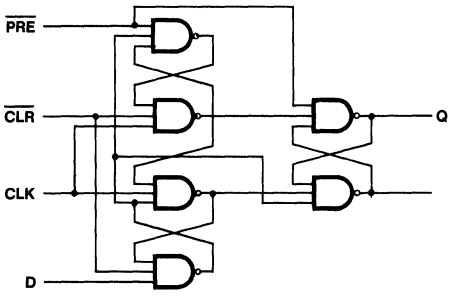


Figure 1

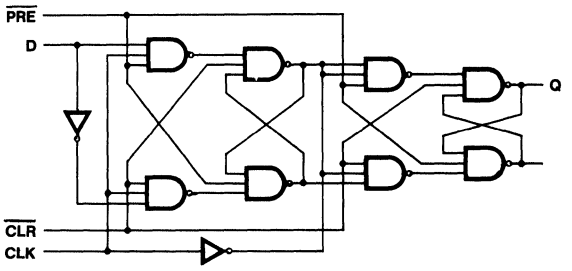


Figure 2

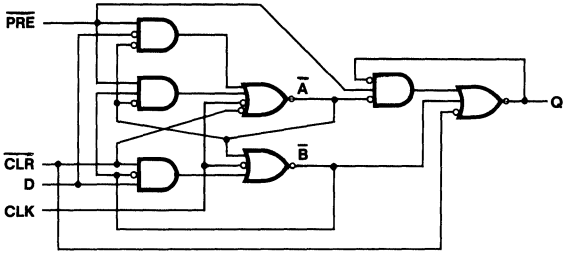


Figure 3

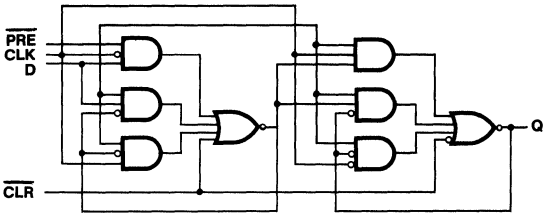


Figure 4

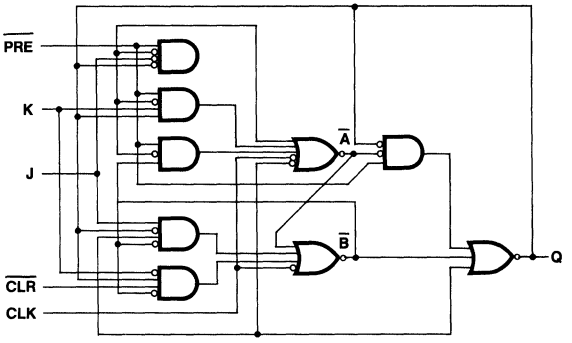


Figure 5

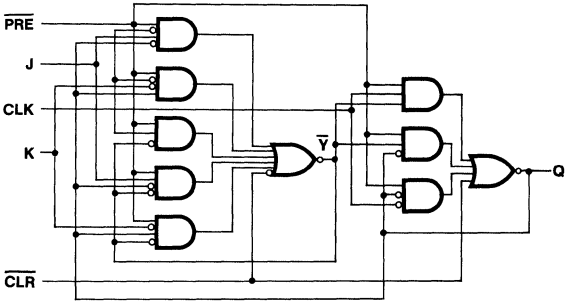


Figure 6

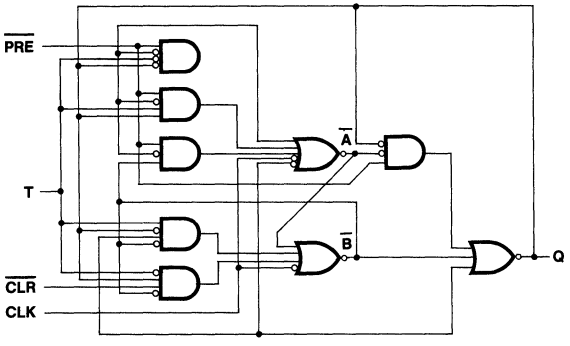


Figure 7

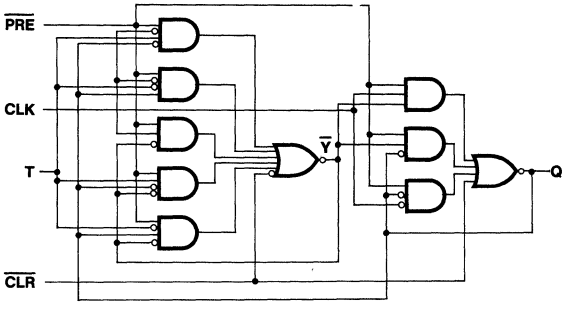
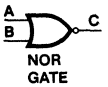


Figure 8

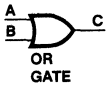
$B = \overline{A}$

A	B
H	L
L	H



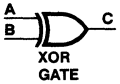
$\overline{C} = A + B$

A	B	C
L	L	H
H	L	L
L	H	L
H	H	L



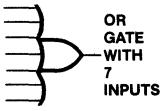
$C = A + B$

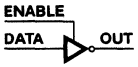
A	B	C
L	L	L
H	L	H
L	H	H
H	H	H



$C = A \div B$

A	B	C
L	L	L
H	L	H
L	H	H
H	H	L





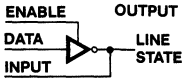
ACTIVE LOW
THREE-STATE
ENABLING
CIRCUIT

IF (ENABLE) OUT = $\overline{DATA}$

ENABLE	DATA	OUT
L	L	Z
H	L	H
L	H	Z
H	H	L

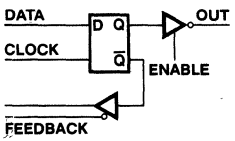
ENABLE	DATA	OUT
L	X	Z
H	L	H
H	H	L

OR EQUIVALENTLY



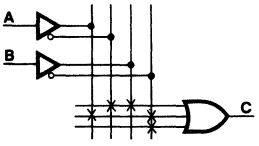
ACTIVE LOW THREE-STATE
ENABLING CIRCUIT WITH
FEEDBACK

ENABLE	DATA	OUTPUT	LINE STATE	INPUT
L	X	Z	L	L
L	X	Z	H	H
H	L	H	H	H
H	H	L	L	L



REGISTERED OUTPUT
CIRCUIT WITH FEEDBACK
AND THREE-STATE ENABLING

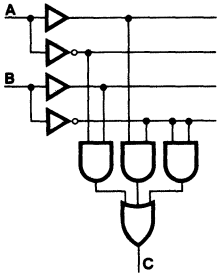
ENABLE	DATA	CLOCK	OUT	FEEDBACK	Q
L	X	L	Z	Q_0	Q_0
L	H	↑(C)	Z	H	H
L	L	↑(C)	Z	L	L
L	X	H	Z	Q_0	Q_0
L	X	↓	Z	Q_0	Q_0
H	X	L	NOT Q_0	Q_0	Q_0
H	H	↑(C)	L	H	H
H	L	↑(C)	H	L	L
H	X	H	NOT Q_0	Q_0	Q_0
H	X	↓	NOT Q_0	Q_0	Q_0



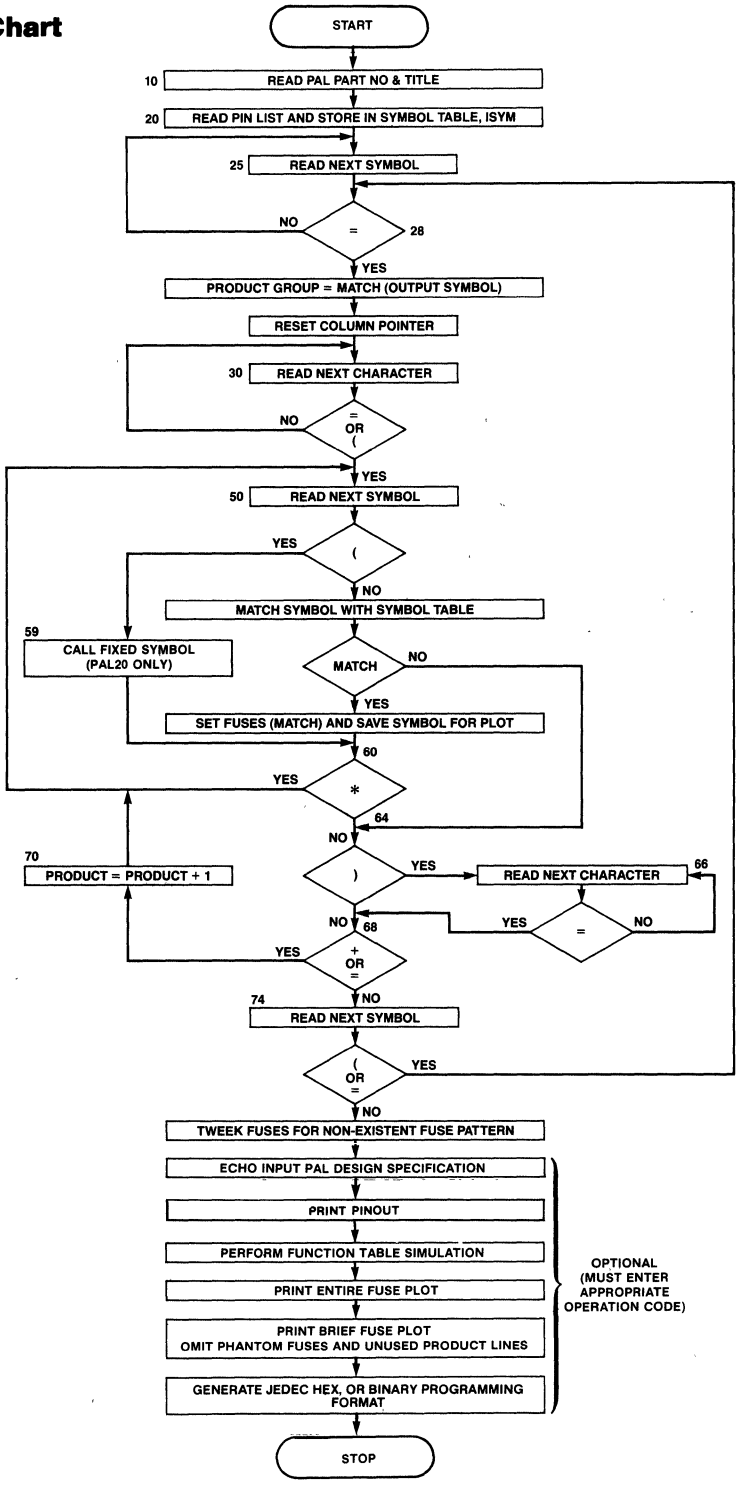
AND-OR PLANE
WITH TWO INPUTS
AND ONE OUTPUT
IN PAL NOTATION

EQUIVALENT CIRCUIT
IN TTL SYMBOLS

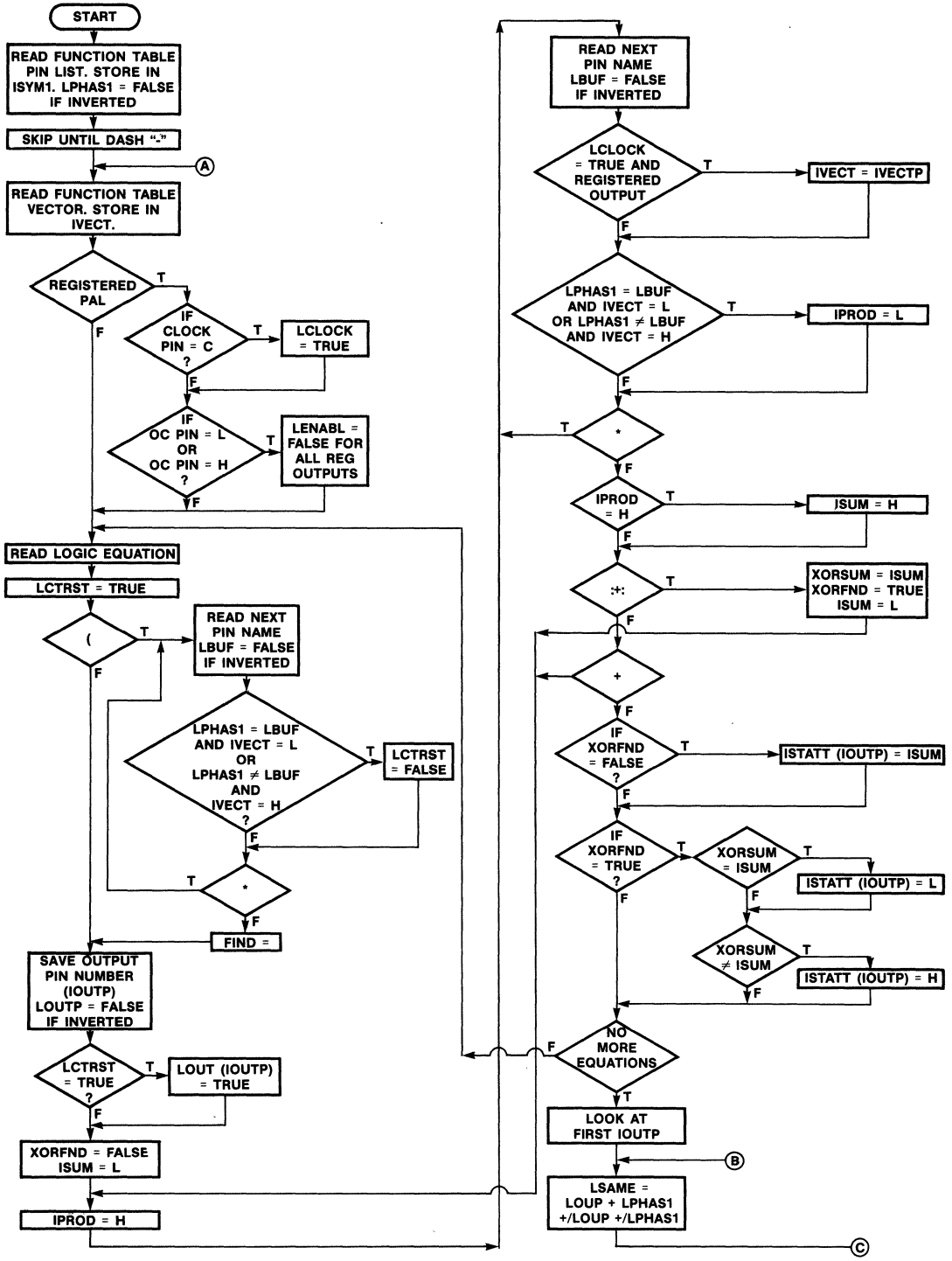
THE SYMBOLS
OF PAL DIAGRAMS

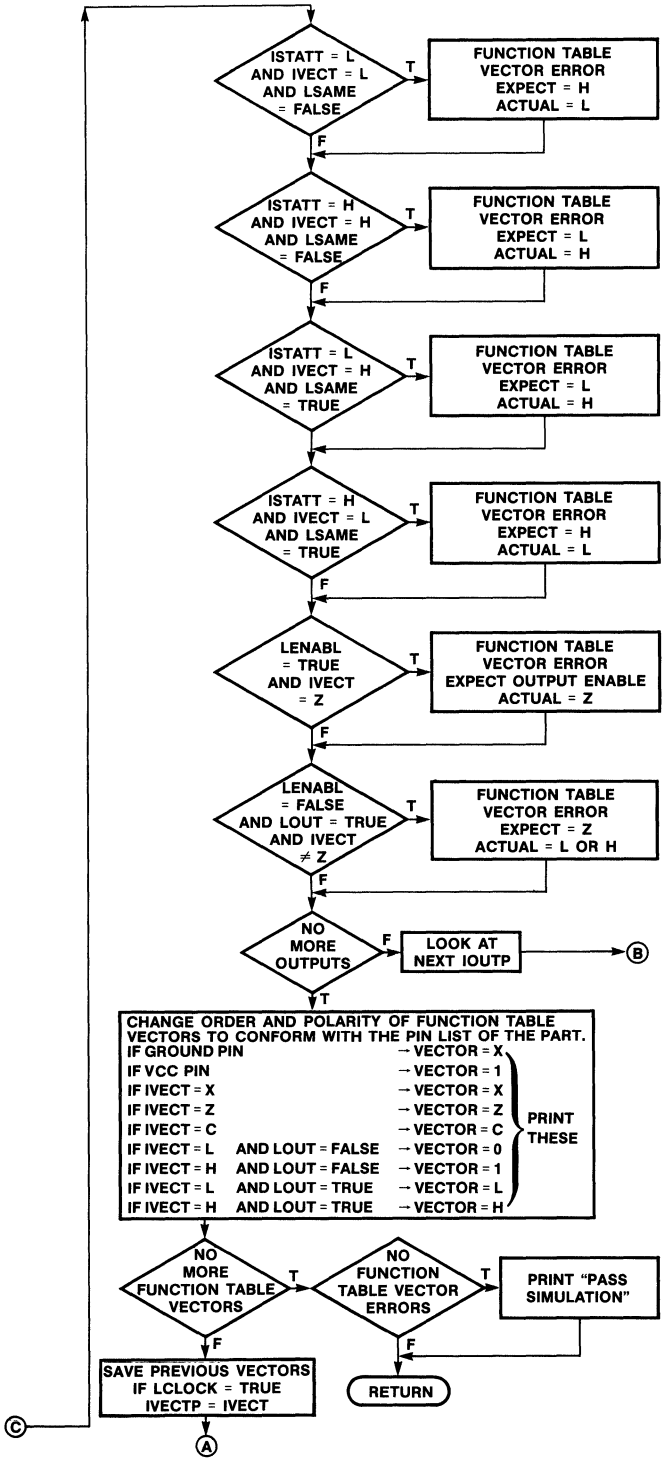


PALASM Flow Chart (Main Program)

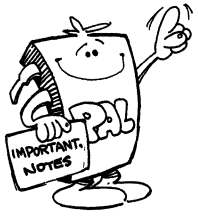


PALASM Flow Chart (Simulator)




ISTATT = L AND IVECT = L AND LSAME = FALSE
T
FUNCTION TABLE VECTOR ERROR EXPECT = H ACTUAL = L
F
ISTATT = H AND IVECT = H AND LSAME = FALSE
T
FUNCTION TABLE VECTOR ERROR EXPECT = L ACTUAL = H
F
ISTATT = L AND IVECT = H AND LSAME = TRUE
T
FUNCTION TABLE VECTOR ERROR EXPECT = L ACTUAL = H
ISTATT = H AND IVECT = L AND LSAME = TRUE
T
FUNCTION TABLE VECTOR ERROR EXPECT = H ACTUAL = L
F
LENABL = TRUE AND IVECT = Z
T
FUNCTION TABLE VECTOR ERROR EXPECT OUTPUT ENABLE ACTUAL = Z
F
LENABL = FALSE AND LOUT = TRUE AND IVECT ≠ Z
T
FUNCTION TABLE VECTOR ERROR EXPECT = Z ACTUAL = L OR H
F
NO MORE OUTPUTS
F
LOOK AT NEXT IOUTP
B
T
CHANGE ORDER AND POLARITY OF FUNCTION TABLE VECTORS TO CONFORM WITH THE PIN LIST OF THE PART.
IF GROUND PIN → VECTOR = X
IF VCC PIN → VECTOR = 1
IF IVECT = X → VECTOR = X
IF IVECT = Z → VECTOR = Z
IF IVECT = C → VECTOR = C
IF IVECT = L AND LOUT = FALSE → VECTOR = 0
IF IVECT = H AND LOUT = FALSE → VECTOR = 1
IF IVECT = L AND LOUT = TRUE → VECTOR = L
IF IVECT = H AND LOUT = TRUE → VECTOR = H
PRINT THESE
NO MORE FUNCTION TABLE VECTORS
T
NO FUNCTION TABLE VECTOR ERRORS
T
PRINT "PASS SIMULATION"
F
F
SAVE PREVIOUS VECTORS IF LCLOCK = TRUE IVECTP = IVECT
RETURN
C
A

Notes



PAL®
PROGRAMMABLE ARRAY LOGIC
Handbook

THIRD EDITION

Table of Contents

PMSI - PAL® Medium Scale Integration.

PAL devices have revolutionized logic design. No longer do logic designers have to wait on that chip they have been looking for. Now, any desired function, standard or non-standard, can be created by simply programming a PAL. Taking advantage of the **Instant Programmability** of PALs. Monolithic Memories introduces a new concept in logic design called PMSI.

Definition:

The PMSI family is a group of circuit designs derived by programming PALs. These devices perform predetermined logic functions which are not available in the TTL family, which Monolithic Memories believes, have the potential of becoming standard products.

Description:

Table I shows a list of PMSIs presently offered by Monolithic Memories. It, also, describes the function and the PAL, from which each PMSI has been derived. A PMSI datasheet contains: 1) Introduction and Ordering Information, 2) PMSI Logic Symbols, 3) PMSI Specifications, 4) PMSI Applications. A customer can use this information to program a standard unprogrammed PAL to obtain a PMSI. In addition, if a change in design has to be made to suit particular needs, it can simply be achieved by modifying the PAL design specification.

A system designer can take advantage of using a PMSI as a building block, because each PMSI design has been tested and characterized thoroughly.

PMSI NUMBER	FUNCTION	PART TYPE
PMSI001	3-to-8 Demultiplexer with Control Storage	16R6
PMSI002	Octal Addressable Register and Demultiplexer	16R8
PMSI003	Octal Comparator	16C1
PMSI401	Octal Up/Down Counter	20X8
PMSI402	Octal Down Counter	20X8
PMSI403	2-Digit BCD Counter	20X8
PMSI404	9-Bit Counter	20X10
PMSI405	9-Bit Register	20X10
PMSI406	10-Bit Register	20X10
PMSI407	10-Bit Addressable Register	20X10
PMSI408	Interface Controller for 68000 to Zilog 8500 Family	20X10

Table 1

Customer Benefits:

The PMSI family offers all the advantages and benefits of the PAL family. Listed below are some additional advantages:

- PMSI can be used as a logic building block.
- In high volume, PMSI costs less than PALs.
- PMSI inherently has PAL as a second source.
- Easy to modify the design to suit one's own needs.

Electricals:

This brochure does not contain the electrical specifications. Table I shows the PAL part number from which each PMSI has been derived. For electrical specification of each PMSI, refer to the corresponding PAL in Monolothic Memories' Bipolar LSI Databook or section 3 of the PAL Handbook.

Further Information:

PMSI can be obtained in the following different ways:

A) From unprogrammed PAL units. A customer who wishes to utilize a PMSI function can do so by programming standard off-th-shelf PAL units. Small quantities of PAL units can be purchased at any distribution location. PAL devices can be programmed:
   a) at customer location
   b) by local distributors
   c) by our field applications engineers

   The necessary inputs for programming PAL units are given in the PMSI datasheets. When ordering, use the corresponding PMSI number shown in Table 1.

B) Pre-programmed PMSI: Small quantities of pre-programmed PMSI can be obtained directly from Monolithic Memories, by special arrangements via your FAE or Reps. The pre-programmed PMSIs are not available through distribution.

   Once the volume of a particular PMSI gets high enough, it is converted into an HMSI (Hard Array Logic MSI). When this happens, it is offered as a standard product and is available through our distributors as off-the-shelf product. HMSI products include:

FUNCTION	PART NUMBER
Multifunction register	SN54/74LS380
16:1 Mux	SN54/74LS450
Dual 8:1 Mux	SN54/74LS451
Quad 4:1 Mux	SN54/74LS453
10-bit comparator	SN54/74LS460
Octal counter	SN54/74LS461
8-bit up/down counter	SN54/74LS469
10-bit counter	SN54/74LS491
Octal shift register	SN54/74LS498

Programming Support:

A) Software support: Monolithic Memories provides the necessary development software called PALASM, which converts logic equations as shown in the PMSI data sheets into a desired fuse pattern. This software is available from Monolithic Memories in several different formats. For more details, call our local representative or the factory.

B) Hardware support: Here is a list of few major programmer manufacturers:

1)	Data I/O	4)	Pro-Log
2)	Digilec	5)	Stag
3)	Kontron	6)	Structured Design

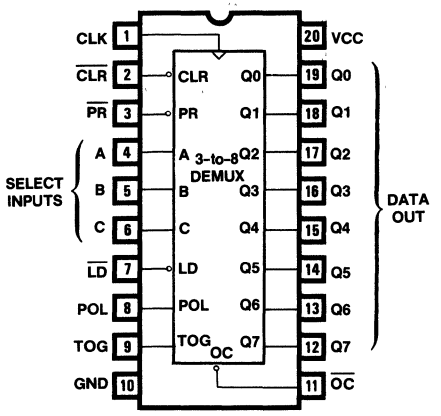
Some of these programmers support Monolithic Memories' PALASM software as well.

3-to-8 Demultiplexer with Control Storage

PMSI001

PAL16R8

see page 5-8

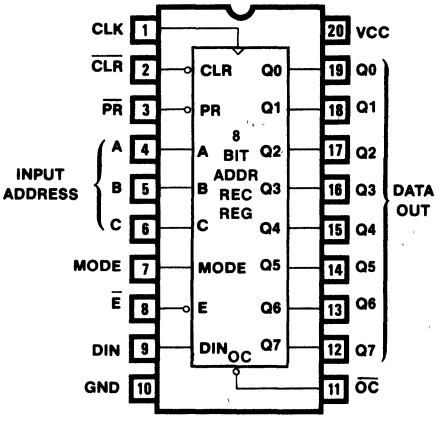


Octal Addressable Register and Demultiplexer

PMSI002

PAL16R8

see page 5-10

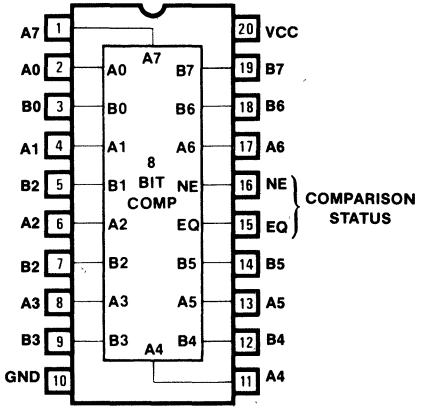


Octal Comparator

PMSI003

PAL16C1

see page 5-12

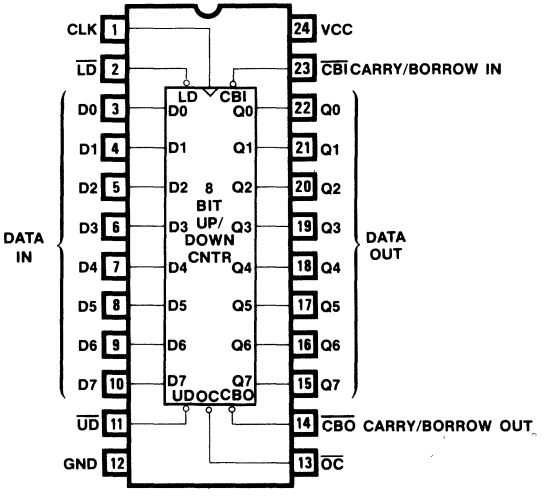


Octal Up/Down Counter

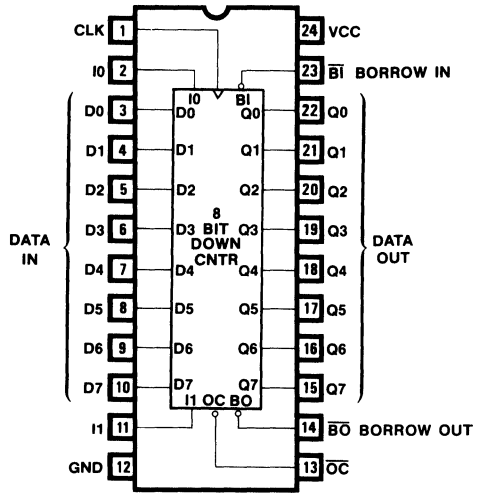
PMSI401

PAL20X8

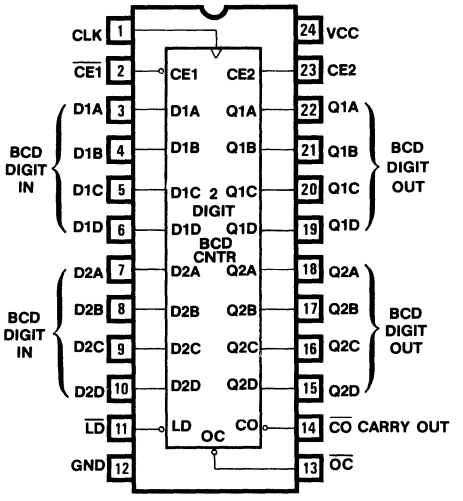
see page 5-14



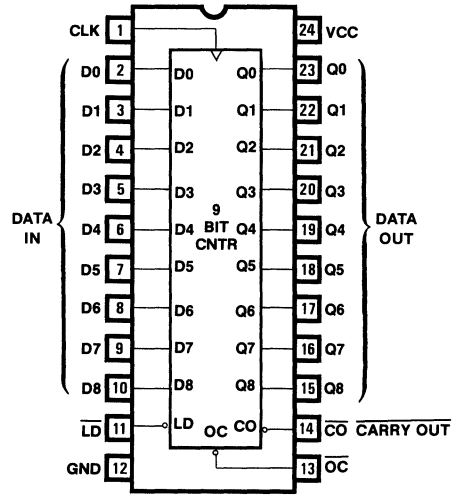
Octal Down Counter
PMSI402
PAL20X8
see page 5-16



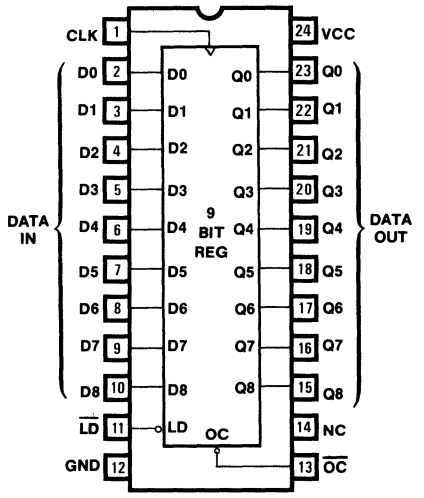
2-Digit BCD Counter
PMSI403
PAL20X8
see page 5-18



9-Bit Counter
PMSI404
PAL20X10
see page 5-20

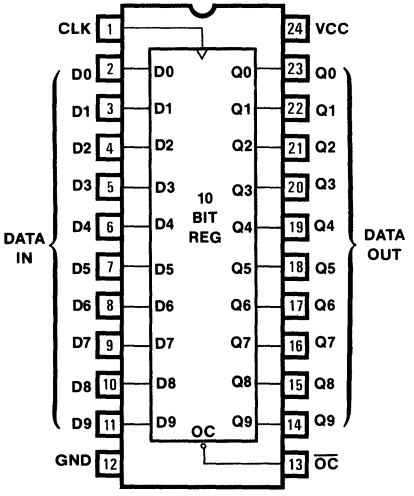


9-Bit Register
PMSI405
PAL20X10
see page 5-22

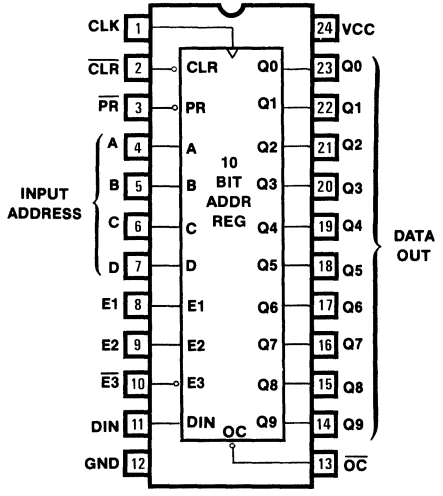


5

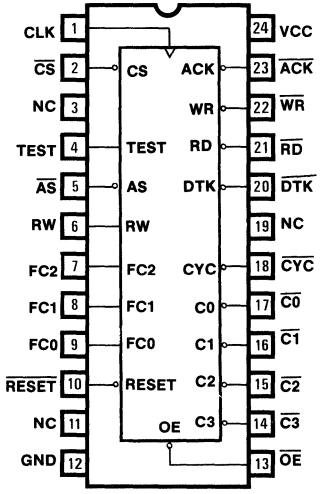
10-Bit Register
PMSI406
PAL20X10
see page 5-24



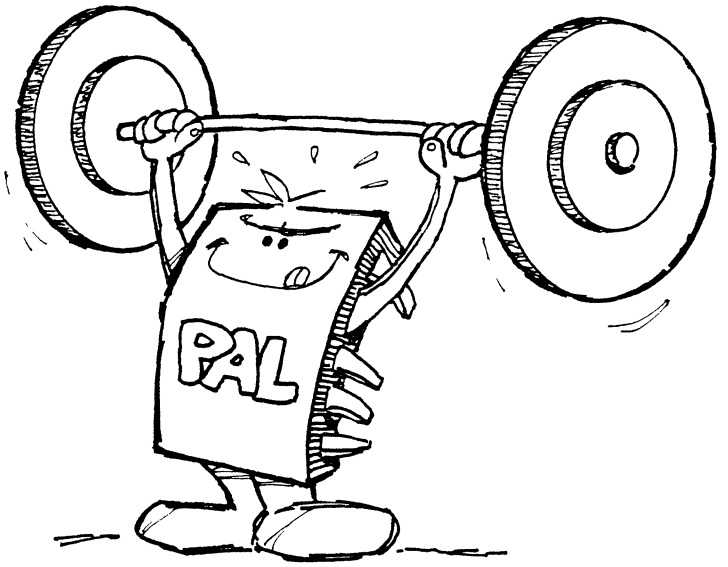
10-Bit Addressable Register
PMSI407
PAL20X10
see page 5-26



Interface Controller for 68000
to Zilog 8500 Family
PMSI408
PAL20X10
see page 5-28



PMSI SPECIFICATIONS



3 - to - 8 Demultiplexer with Control Storage

PAL16R8

PMSI001

Features/Benefits

- **Ideal for memory chip select decoding**
- **8 bits match byte boundaries**
- **Bus-structured pinout**
- **20 pin SKINNYDIP™ saves space**
- **3-state outputs drive bus lines**
- **Low current PNP iputs reduce loading**
- **Expandable in 8-bit increments**

Description

The PMSI001 is a 3-to-8 demultiplexer with control storage. Five control inputs ($\overline{LD}$, $\overline{CLR}$, $\overline{PR}$, POL, TOG) select one of six operations which occur synchronously on the rising edge of the clock (CLK).

The LOAD true operation loads the decoded binary inputs (A, B, C) into the output register (Q7-Q0) when polarity is TRUE (POL = HIGH). The complement of the binary inputs is decoded and loaded into the output register when polarity is FALSE (POL = LOW).

The HOLD operation holds the previous value in the register when toggle is FALSE (TOG=LOW) regardless of clock transitions. The TOGGLE POLARITY operation toggles the polarity of the data in the output register when toggle is TRUE (TOG=HIGH).

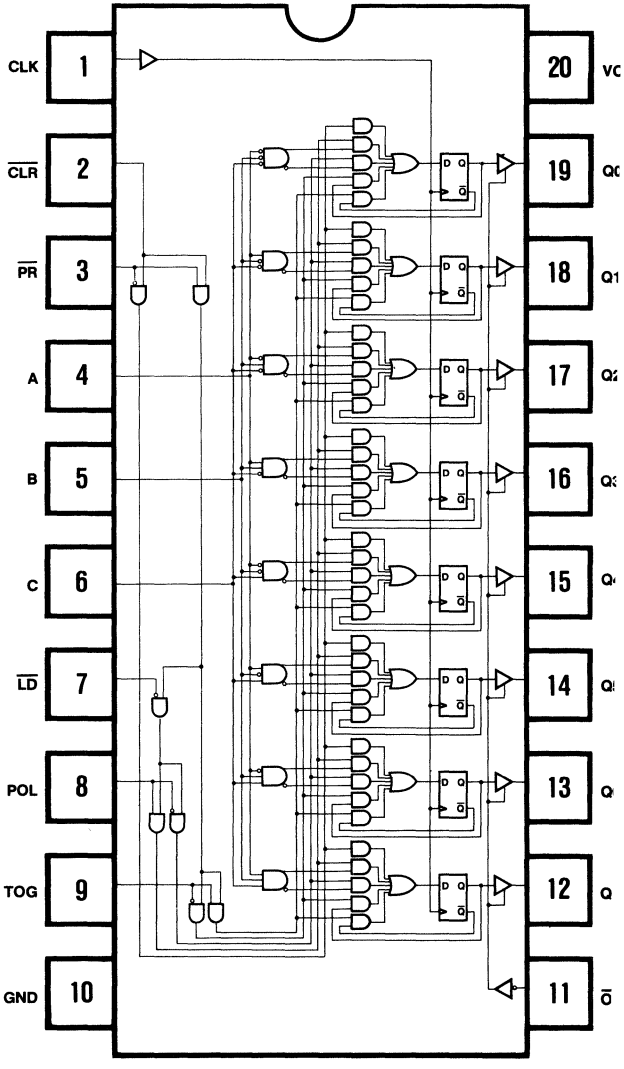
The CLEAR operation resets the output register to all LOWs. The PRESET operation presets the output register to all HIGHs. CLEAR overrides PRESET, PRESET overrides LOAD, and LOAD overrides HOLD.

The output register (Q_7-Q_0) is enabled when output control ($\overline{OC}$) is LOW, and disabled (HI-Z) when $\overline{OC}$ is HIGH. The output drivers will sink the 24 mA required for many bus interface standards.

Function Table

$\overline{OC}$	CLK	$\overline{CLR}$	$\overline{PR}$	$\overline{LD}$	POL	TOG	C-A	Q7-Q0	OPERATION
H	X	X	X	X	X	X	X	Z	HI-Z
L	↑	L	X	X	X	X	X	L	CLEAR
L	↑	H	L	X	X	X	X	X	PRESET
L	↑	H	H	L	H	X	I	MUX	LOAD TRUE
L	↑	H	H	L	L	X	I	MUX	LOAD COMP
L	↑	H	H	H	X	L	X	Q	HOLD
L	↑	H	H	H	X	H	X	Q	TOGGLE POLARITY

Logic Diagram



```
PAL16R8                                          PAL DESIGN SPECIFICATION
PMSI001                                            BIRKNER/COLI 04/07/82
3-TO-8 DEMULTIPLEXER WITH CONTROL STORAGE
MMI SUNNYVALE, CALIFORNIA
CLK /CLR /PR A  B  C  /LD POL TOG GND
/OC  Q7   Q6 Q5 Q4 Q3  Q2 Q1  Q0  VCC


/Q0 :=  CLR                          ;CLEAR Q0
     + /PR* LD*/POL*/C*/B*/A         ;LOAD COMPLEMENT (DECODE 000)
     + /PR* LD* POL*       A         ;LOAD TRUE
     + /PR* LD* POL*    B            ;LOAD TRUE
     + /PR* LD* POL* C               ;LOAD TRUE
     + /PR*/LD*/TOG*/Q0              ;HOLD
     + /PR*/LD* TOG* Q0              ;TOGGLE POLARITY

/Q1 :=  CLR                          ;CLEAR Q1
     + /PR* LD*/POL*/C*/B* A         ;LOAD COMPLEMENT (DECODE 001)
     + /PR* LD* POL*      /A         ;LOAD TRUE
     + /PR* LD* POL*    B            ;LOAD TRUE
     + /PR* LD* POL* C               ;LOAD TRUE
     + /PR*/LD*/TOG*/Q1              ;HOLD
     + /PR*/LD* TOG* Q1              ;TOGGLE POLARITY

/Q2 :=  CLR                          ;CLEAR Q2
     + /PR* LD*/POL*/C* B*/A         ;LOAD COMPLEMENT (DECODE 010)
     + /PR* LD* POL*       A         ;LOAD TRUE
     + /PR* LD* POL*   /B            ;LOAD TRUE
     + /PR* LD* POL* C               ;LOAD TRUE
     + /PR*/LD*/TOG*/Q2              ;HOLD
     + /PR*/LD* TOG* Q2              ;TOGGLE POLARITY

/Q3 :=  CLR                          ;CLEAR Q3
     + /PR* LD*/POL*/C* B* A         ;LOAD COMPLEMENT (DECODE 011)
     + /PR* LD* POL*      /A         ;LOAD TRUE
     + /PR* LD* POL*   /B            ;LOAD TRUE
     + /PR* LD* POL* C               ;LOAD TRUE
     + /PR*/LD*/TOG*/Q3              ;HOLD
     + /PR*/LD* TOG* Q3              ;TOGGLE POLARITY

/Q4 :=  CLR                          ;CLEAR Q4
     + /PR* LD*/POL* C*/B*/A         ;LOAD COMPLEMENT (DECODE 100)
     + /PR* LD* POL*       A         ;LOAD TRUE
     + /PR* LD* POL*    B            ;LOAD TRUE
     + /PR* LD* POL*/C               ;LOAD TRUE
     + /PR*/LD*/TOG*/Q4              ;HOLD
     + /PR*/LD* TOG* Q4              ;TOGGLE POLARITY

/Q5 :=  CLR                          ;CLEAR Q5
     + /PR* LD*/POL* C*/B* A         ;LOAD COMPLEMENT (DECODE 101)
     + /PR* LD* POL*      /A         ;LOAD TRUE
     + /PR* LD* POL*    B            ;LOAD TRUE
     + /PR* LD* POL*/C               ;LOAD TRUE
     + /PR*/LD*/TOG*/Q5              ;HOLD
     + /PR*/LD* TOG* Q5              ;TOGGLE POLARITY

/Q6 :=  CLR                          ;CLEAR Q6
     + /PR* LD*/POL* C* B*/A         ;LOAD COMPLEMENT (DECODE 110)
     + /PR* LD* POL*       A         ;LOAD TRUE
     + /PR* LD* POL*   /B            ;LOAD TRUE
     + /PR* LD* POL*/C               ;LOAD TRUE
     + /PR*/LD*/TOG*/Q6              ;HOLD
     + /PR*/LD* TOG* Q6              ;TOGGLE POLARITY

/Q7 :=  CLR                          ;CLEAR Q7
     + /PR* LD*/POL* C* B* A         ;LOAD COMPLEMENT (DECODE 111)
     + /PR* LD* POL*      /A         ;LOAD TRUE
     + /PR* LD* POL*   /B            ;LOAD TRUE
     + /PR* LD* POL*/C               ;LOAD TRUE
     + /PR*/LD*/TOG*/Q7              ;HOLD
     + /PR*/LD* TOG* Q7              ;TOGGLE POLARITY


FUNCTION TABLE

/OC CLK /CLR /PR /LD POL TOG C B A Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0

;CONTROL     FUNCTIONS     POLARITY  INPUT    OUTPUT
;/OC CLK   /CLR /PR /LD   POL  TOG   CBA    Q7----Q0    COMMENTS
-------------------------------------------------------------------------------
  L   C      L   L   L     H    L     XXX    LLLLLLLL    CLEAR  (OVERRIDES /PR)
  L   C      H   L   L     H    L     XXX    HHHHHHHH    PRESET (OVERRIDES /LD)
  L   C      H   H   L     H    X     LLL    LLLLLLLH    LOAD 0 (TRUE INPUT)
  L   C      H   H   L     H    X     LLH    LLLLLLHL    LOAD 1 (TRUE INPUT)
  L   C      H   H   L     H    X     LHL    LLLLLHLL    LOAD 2 (TRUE INPUT)
  L   C      H   H   L     H    X     LHH    LLLLHLLL    LOAD 3 (TRUE INPUT)
  L   C      H   H   L     H    X     HLL    LLLHLLLL    LOAD 4 (TRUE INPUT)
  L   C      H   H   L     H    X     HLH    LLHLLLLL    LOAD 5 (TRUE INPUT)
  L   C      H   H   L     H    X     HHL    LHLLLLLL    LOAD 6 (TRUE INPUT)
  L   C      H   H   L     H    X     HHH    HLLLLLLL    LOAD 7 (TRUE INPUT)
  L   C      H   H   H     X    L     XXX    HLLLLLLL    HOLD 7 (TRUE POLARITY)
  L   C      H   H   H     X    H     XXX    LHHHHHHH    HOLD (TOGGLE POLARITY)
  L   C      H   H   H     X    H     XXX    HLLLLLLL    HOLD (TOGGLE POLARITY)
  L   C      H   H   L     L    X     LLL    HHHHHHHL    LOAD 0 (COMP INPUT)
  L   C      H   H   L     L    X     LLH    HHHHHHLH    LOAD 1 (COMP INPUT)
  L   C      H   H   L     L    X     LHL    HHHHHLHH    LOAD 2 (COMP INPUT)
  L   C      H   H   L     L    X     LHH    HHHHLHHH    LOAD 3 (COMP INPUT)
  L   C      H   H   L     L    X     HLL    HHHLHHHH    LOAD 4 (COMP INPUT)
  L   C      H   H   L     L    X     HLH    HHLHHHHH    LOAD 5 (COMP INPUT)
  L   C      H   H   L     L    X     HHL    HLHHHHHH    LOAD 6 (COMP INPUT)
  L   C      H   H   L     L    X     HHH    LHHHHHHH    LOAD 7 (COMP INPUT)
  L   C      H   H   H     X    L     XXX    LHHHHHHH    HOLD 7 (TRUE POLARITY)
  L   C      H   H   H     X    H     XXX    HLLLLLLL    HOLD (TOGGLE POLARITY)
  L   C      H   H   H     X    H     XXX    LHHHHHHH    HOLD (TOGGLE POLARITY)
  H   X      X   X   X     X    X     XXX    ZZZZZZZZ    TEST HI-Z
-------------------------------------------------------------------------------
```

Octal Addressable Register and Demultiplexer

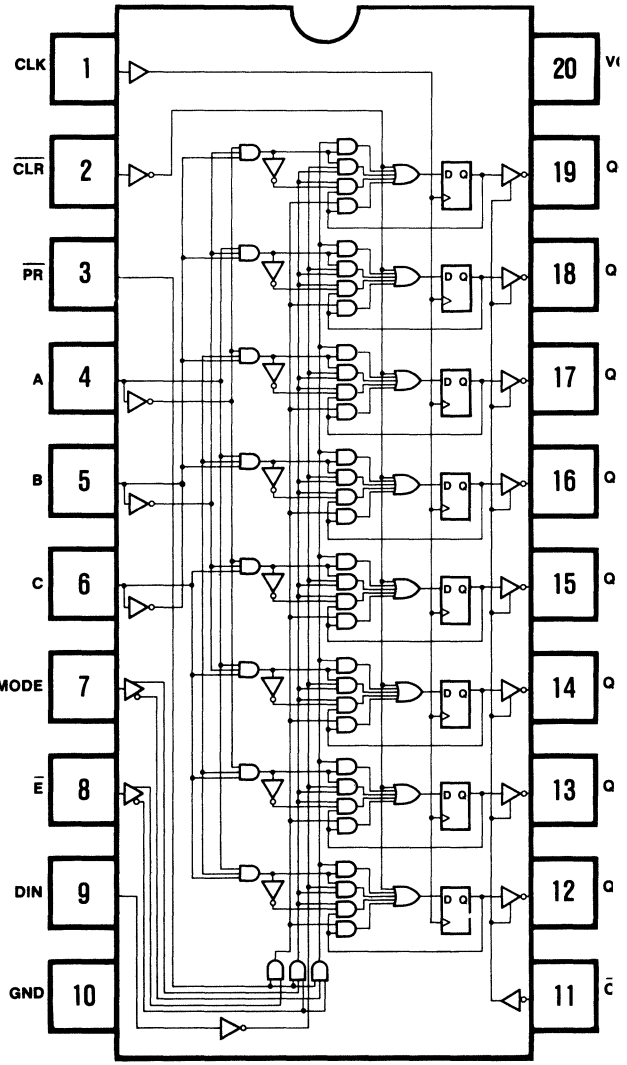
PAL16R8

PMSI002

Features/Benefits

- **8-bit addressable register**
- **3 to 8 demultiplexer**
- **Preset, clear, enable, and mode control gives added versatility**
- **8 bits match byte boundaries**
- **Bus structured pinout**
- **24-pin SKINNYDIP™ saves space**
- **3-state outputs drive bus lines**
- **Low current PNP inputs reduce loading**

Logic Diagram



Description

The octal addressable register and demultiplexer performs one of two MSI functions depending on the state of the mode select pin. (MODE = HIGH) Performs the function of an addressable register with 8 outputs (Q7-Q0) and one data input (DIN). The registered output is selected by three input address pins (A, B, C). (MODE = LOW) converts this chip into an active low demultiplexer, where the addressed output is low all others remain high.

CLEAR and PRESET are active low pins which set all outputs to low or high respectively. When ENABLE ($\overline{E}$) is high the chip is disabled and the outputs retain their previous states. CLEAR overrides PRESET and ENABLE. PRESET overrides ENABLE. The output register (Q7-Q0) is enabled when $\overline{OC}$ is LOW and disabled (Hi-Z) when $\overline{OC}$ is HIGH. The output drivers will sink the 24 mA required for many bus interfaces standards.

Function Table

$\overline{OC}$	CLK	$\overline{CLR}$	$\overline{PR}$	$\overline{E}$	MODE	C	B	A	DIN	Q7-Q0	OPERATION
H	X	X	X	X	X	X	X	X	X	Z	HI-Z
L	↑	L	X	X	X	X	X	X	X	L	CLEAR
L	↑	H	L	X	X	X	X	X	X	H	PRESET
L	↑	H	H	L	L	C	B	A	X	MUX	Addressed output = LOW all other outputs = HIGH
L	↑	H	H	L	H	C	B	A	DIN	REG	Addressed output = DIN, all other outputs hold state
L	↑	H	H	H	X		X		X	Q	HOLD

```
PAL16R8                                          PAL DESIGN SPECIFICATION
PMSI002                                            DANESH TAVANA 04/27/82
OCTAL ADDRESSABLE REGISTER AND DEMULTIPLEXER
MMI SUNNYVALE, CALIFORNIA
CLK /CLR /PR A  B  C  MODE /E DIN GND
/OC  Q7   Q6 Q5 Q4 Q3  Q2  Q1 Q0  VCC


/Q0 :=  CLR                          ;CLEAR (LSB)
     + /PR* E*/MODE*/C*/B*/A         ;DEMULTIPLEX OUTPUT  /Q0=H
     + /PR* E* MODE*/C*/B*/A*/DIN    ;REGISTER OUTPUT /Q0=/DIN
     + /PR* E* MODE*       A*/Q0     ;LOAD PREVIOUS STATE (/Q0) OR A LOW
     + /PR* E* MODE*    B   */Q0     ;LOAD PREVIOUS STATE (/Q0) OR A LOW
     + /PR* E* MODE* C      */Q0     ;LOAD PREVIOUS STATE (/Q0) OR A LOW
     + /PR*/E               */Q0     ;HOLD IF NOT LOADING (/E=H)

/Q1 :=  CLR                          ;CLEAR
     + /PR* E*/MODE*/C*/B* A         ;DEMULTIPLEX OUTPUT  /Q1=H
     + /PR* E* MODE*/C*/B* A*/DIN    ;REGISTER OUTPUT /Q1=/DIN
     + /PR* E* MODE*      /A*/Q1     ;LOAD PREVIOUS STATE (/Q1) OR A LOW
     + /PR* E* MODE*    B   */Q1     ;LOAD PREVIOUS STATE (/Q1) OR A LOW
     + /PR* E* MODE* C      */Q1     ;LOAD PREVIOUS STATE (/Q1) OR A LOW
     + /PR*/E               */Q1     ;HOLD IF NOT LOADING (/E=H)

/Q2 :=  CLR                          ;CLEAR
     + /PR* E*/MODE*/C* B*/A         ;DEMULTIPLEX OUTPUT  /Q2=H
     + /PR* E* MODE*/C* B*/A*/DIN    ;REGISTER OUTPUT /Q2=/DIN
     + /PR* E* MODE*       A*/Q2     ;LOAD PREVIOUS STATE (/Q2) OR A LOW
     + /PR* E* MODE*   /B   */Q2     ;LOAD PREVIOUS STATE (/Q2) OR A LOW
     + /PR* E* MODE* C      */Q2     ;LOAD PREVIOUS STATE (/Q2) OR A LOW
     + /PR*/E               */Q2     ;HOLD IF NOT LOADING (/E=H)

/Q3 :=  CLR                          ;CLEAR
     + /PR* E*/MODE*/C* B* A         ;DEMULTIPLEX OUTPUT  /Q3=H
     + /PR* E* MODE*/C* B* A*/DIN    ;REGISTER OUTPUT /Q3=/DIN
     + /PR* E* MODE*      /A*/Q3     ;LOAD PREVIOUS STATE (/Q3) OR A LOW
     + /PR* E* MODE*   /B   */Q3     ;LOAD PREVIOUS STATE (/Q3) OR A LOW
     + /PR* E* MODE* C      */Q3     ;LOAD PREVIOUS STATE (/Q3) OR A LOW
     + /PR*/E               */Q3     ;HOLD IF NOT LOADING (/E=H)

/Q4 :=  CLR                          ;CLEAR
     + /PR* E*/MODE* C*/B*/A         ;DEMULTIPLEX OUTPUT  /Q4=H
     + /PR* E* MODE* C*/B*/A*/DIN    ;REGISTER OUTPUT /Q4=/DIN
     + /PR* E* MODE*       A*/Q4     ;LOAD PREVIOUS STATE (/Q4) OR A LOW
     + /PR* E* MODE*    B   */Q4     ;LOAD PREVIOUS STATE (/Q4) OR A LOW
     + /PR* E* MODE*/C      */Q4     ;LOAD PREVIOUS STATE (/Q4) OR A LOW
     + /PR*/E               */Q4     ;HOLD IF NOT LOADING (/E=H)

/Q5 :=  CLR                          ;CLEAR
     + /PR* E*/MODE* C*/B* A         ;DEMULTIPLEX OUTPUT  /Q5=H
     + /PR* E* MODE* C*/B* A*/DIN    ;REGISTER OUTPUT /Q5=/DIN
     + /PR* E* MODE*      /A*/Q5     ;LOAD PREVIOUS STATE (/Q5) OR A LOW
     + /PR* E* MODE*    B   */Q5     ;LOAD PREVIOUS STATE (/Q5) OR A LOW
     + /PR* E* MODE*/C      */Q5     ;LOAD PREVIOUS STATE (/Q5) OR A LOW
     + /PR*/E               */Q5     ;HOLD IF NOT LOADING (/E=H)

/Q6 :=  CLR                          ;CLEAR
     + /PR* E*/MODE* C* B*/A         ;DEMULTIPLEX OUTPUT  /Q6=H
     + /PR* E* MODE* C* B*/A*/DIN    ;REGISTER OUTPUT /Q5=/DIN
     + /PR* E* MODE*       A*/Q6     ;LOAD PREVIOUS STATE (/Q6) OR A LOW
     + /PR* E* MODE*   /B   */Q6     ;LOAD PREVIOUS STATE (/Q6) OR A LOW
     + /PR* E* MODE*/C      */Q6     ;LOAD PREVIOUS STATE (/Q6) OR A LOW
     + /PR*/E               */Q6     ;HOLD IF NOT LOADING (/E=H)

/Q7 :=  CLR                          ;CLEAR (MSB)
     + /PR* E*/MODE* C* B* A         ;DEMULTIPLEX OUTPUT  /Q7=H
     + /PR* E* MODE* C* B* A*/DIN    ;REGISTER OUTPUT /Q7=/DIN
     + /PR* E* MODE*      /A*/Q7     ;LOAD PREVIOUS STATE (/Q7) OR A LOW
     + /PR* E* MODE*   /B   */Q7     ;LOAD PREVIOUS STATE (/Q7) OR A LOW
     + /PR* E* MODE*/C      */Q7     ;LOAD PREVIOUS STATE (/Q7) OR A LOW
     + /PR*/E               */Q7     ;HOLD IF NOT LOADING (/E=H)
```

FUNCTION TABLE

/OC CLK /CLR /PR /E MODE C B A DIN Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0

;CONTROL		---FUNCTIONS----				-INPUTS--				----OUTPUTS----								
;/OC	CLK	/CLR	/PR	/E	MODE	C	B	A	DIN	Q 7	Q 6	Q 5	Q 4	Q 3	Q 2	Q 1	Q 0	COMMENTS
L	C	L	L	X	X	X	X	X	X	L	L	L	L	L	L	L	L	CLEAR (OVERRD /PR)
L	C	H	L	X	X	X	X	X	X	H	H	H	H	H	H	H	H	PRESET (OVERRD ENABLES)
L	C	H	H	L	H	L	L	L	L	H	H	H	H	H	H	H	L	LOAD Q0 WITH DIN
L	C	H	H	L	H	L	L	H	L	H	H	H	H	H	H	L	L	LOAD Q1 WITH DIN
L	C	H	H	L	H	L	H	L	L	H	H	H	H	H	L	L	L	LOAD Q2 WITH DIN
L	C	H	H	L	H	L	H	H	L	H	H	H	H	L	L	L	L	LOAD Q3 WITH DIN
L	C	H	H	L	H	H	L	L	L	H	H	H	L	L	L	L	L	LOAD Q4 WITH DIN
L	C	H	H	L	H	H	L	H	L	H	H	L	L	L	L	L	L	LOAD Q5 WITH DIN
L	C	H	H	L	H	H	H	L	L	H	L	L	L	L	L	L	L	LOAD Q6 WITH DIN
L	C	H	H	L	H	H	H	H	L	L	L	L	L	L	L	L	L	LOAD Q7 WITH DIN
L	C	H	H	L	H	H	H	H	H	H	L	L	L	L	L	L	L	LOAD Q7 WITH DIN
L	C	H	H	L	H	H	H	L	H	H	H	L	L	L	L	L	L	LOAD Q6 WITH DIN
L	C	H	H	L	H	H	L	H	H	H	H	H	L	L	L	L	L	LOAD Q5 WITH DIN
L	C	H	H	L	H	H	L	L	H	H	H	H	H	L	L	L	L	LOAD Q4 WITH DIN
L	C	H	H	L	H	L	H	H	H	H	H	H	H	H	L	L	L	LOAD Q3 WITH DIN
L	C	H	H	L	H	L	H	L	H	H	H	H	H	H	H	L	L	LOAD Q2 WITH DIN
L	C	H	H	L	H	L	L	H	H	H	H	H	H	H	H	H	L	LOAD Q1 WITH DIN
L	C	H	H	L	H	L	L	L	H	H	H	H	H	H	H	H	H	LOAD Q0 WITH DIN
L	C	H	H	L	L	L	L	L	X	H	H	H	H	H	H	H	L	DECODE ADDRESS LINE 0
L	C	H	H	L	L	L	L	H	X	H	H	H	H	H	H	L	H	DECODE ADDRESS LINE 1
L	C	H	H	L	L	L	H	L	X	H	H	H	H	H	L	H	H	DECODE ADDRESS LINE 2
L	C	H	H	L	L	L	H	H	X	H	H	H	H	L	H	H	H	DECODE ADDRESS LINE 3
L	C	H	H	L	L	H	L	L	X	H	H	H	L	H	H	H	H	DECODE ADDRESS LINE 4
L	C	H	H	L	L	H	L	H	X	H	H	L	H	H	H	H	H	DECODE ADDRESS LINE 5
L	C	H	H	L	L	H	H	L	X	H	L	H	H	H	H	H	H	DECODE ADDRESS LINE 6
L	C	H	H	L	L	H	H	H	X	L	H	H	H	H	H	H	H	DECODE ADDRESS LINE 7
L	C	H	H	H	X	X	X	X	X	L	H	H	H	H	H	H	H	HOLD PREVIOUS STATE
H	X	X	X	X	X	X	X	X	X	Z	Z	Z	Z	Z	Z	Z	Z	TEST HI-Z

Octal Comparator
PAL16C1
PMSI003

Features/Benefits

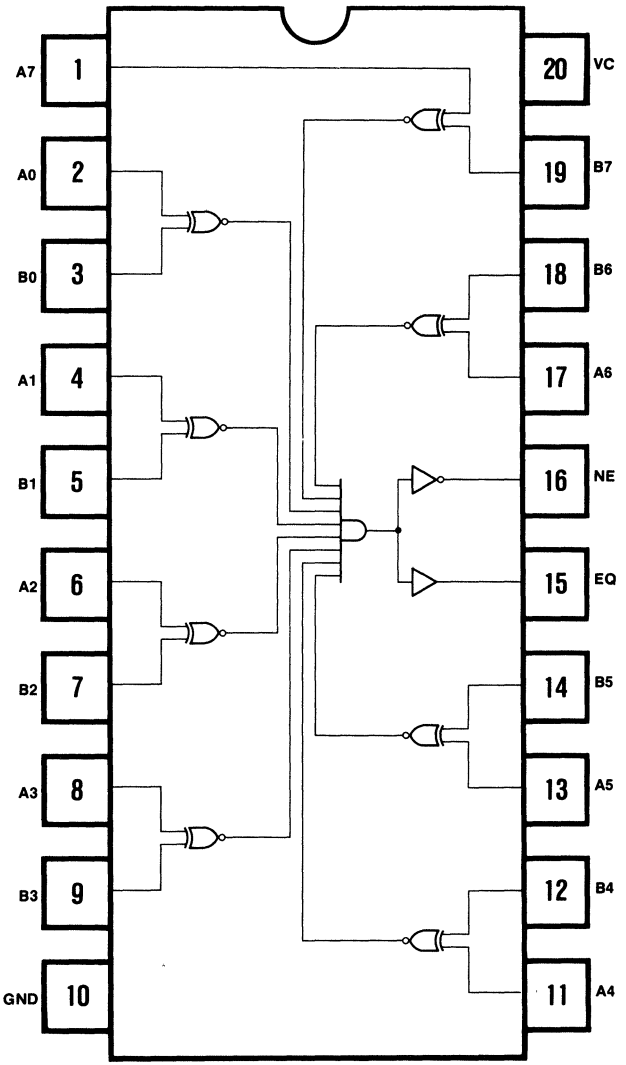
- **8 bits match byte boundaries**
- **True and complement comparison status outputs**
- **20-pin SKINNYDIP™ saves space**
- **Low current PNP inputs reduce loading**
- **Expandable in 8-bit increments**

Description

The PMSI003 is an 8-bit comparator with true and complement comparison status outputs. The device compares two 8-bit data strings (A_7-A_0 and B_7-B_0) to establish if this data is Equivalent (EQ = HIGH and NE = LOW) or Not Equivalent (EQ = LOW and NE = HIGH).

Outputs conform to the standard 8mA LS totem pole drive standard.

Logic Diagram



Function Table

A7-A0	B7-B0	EQ	NE	OPERATION
A	A	H	L	Equivalent (A = B)
B	B	H	L	
A	B	L	H	Not Equivalent (A ≠ B)

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


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

```
FUNCTION TABLE

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

;INPUT  A  INPUT  B  OUTPUTS
;76543210  76543210   NE EQ    COMMENTS
-------------------------------------------------------------------
 HLLLLLLL  LLLLLLLL   H   L    A7=H, B7=L
 LHLLLLLL  LLLLLLLL   H   L    A6=H, B6=L
 LLHLLLLL  LLLLLLLL   H   L    A5=H, B5=L
 LLLHLLLL  LLLLLLLL   H   L    A4=H, A5=L
 LLLLHLLL  LLLLLLLL   H   L    A3=H, B3=L
 LLLLLHLL  LLLLLLLL   H   L    A2=H, B2=L
 LLLLLLHL  LLLLLLLL   H   L    A1=H, B1=L
 LLLLLLLH  LLLLLLLL   H   L    A0=H, B0=L
 LLLLLLLL  HLLLLLLL   H   L    A7=L, B7=H
 LLLLLLLL  LHLLLLLL   H   L    A6=L, B6=H
 LLLLLLLL  LLHLLLLL   H   L    A5=L, B5=H
 LLLLLLLL  LLLHLLLL   H   L    A4=L, B4=H
 LLLLLLLL  LLLLHLLL   H   L    A3=L, B3=H
 LLLLLLLL  LLLLLHLL   H   L    A2=L, B2=H
 LLLLLLLL  LLLLLLHL   H   L    A1=L, B1=H
 LLLLLLLL  LLLLLLLH   H   L    A0=L, B0=H
 LLLLLLLL  LLLLLLLL   L   H    TEST ALL L'S
 HHHHHHHH  HHHHHHHH   L   H    TEST ALL H'S
 HLHLHLHL  HLHLHLHL   L   H    TEST EVEN CHECKERBOARD
 LHLHLHLH  LHLHLHLH   L   H    TEST ODD  CHECKERBOARD
 HHLLHHLL  HHLLHHLL   L   H    TEST EVEN DOUBLE CHECKERBOARD
 LLHHLLHH  LLHHLLHH   L   H    TEST ODD  DOUBLE CHECKERBOARD
-------------------------------------------------------------------
```

Octal Up/Down Counter
PAL20X8
PMSI401

Features/Benefits

- **Octal up/down counter for microprogram-counter, DMA controller and general purpose counting applications**
- **8 bits match byte boundaries**
- **Bus-structured pinout**
- **24-pin SKINNYDIP™ saves space**
- **3-state outputs drive bus lines**
- **Low current PNP inputs reduce loading**
- **Expandable in 8-bit increments**

Description

The PMSI401 is an 8-bit synchronous up/down counter with parallel load and hold capability Three function select inputs ($\overline{LD}$, $\overline{UD}$, $\overline{CBI}$) provide one of four operations which occur synchronously on the rising edge of the clock (CLK).

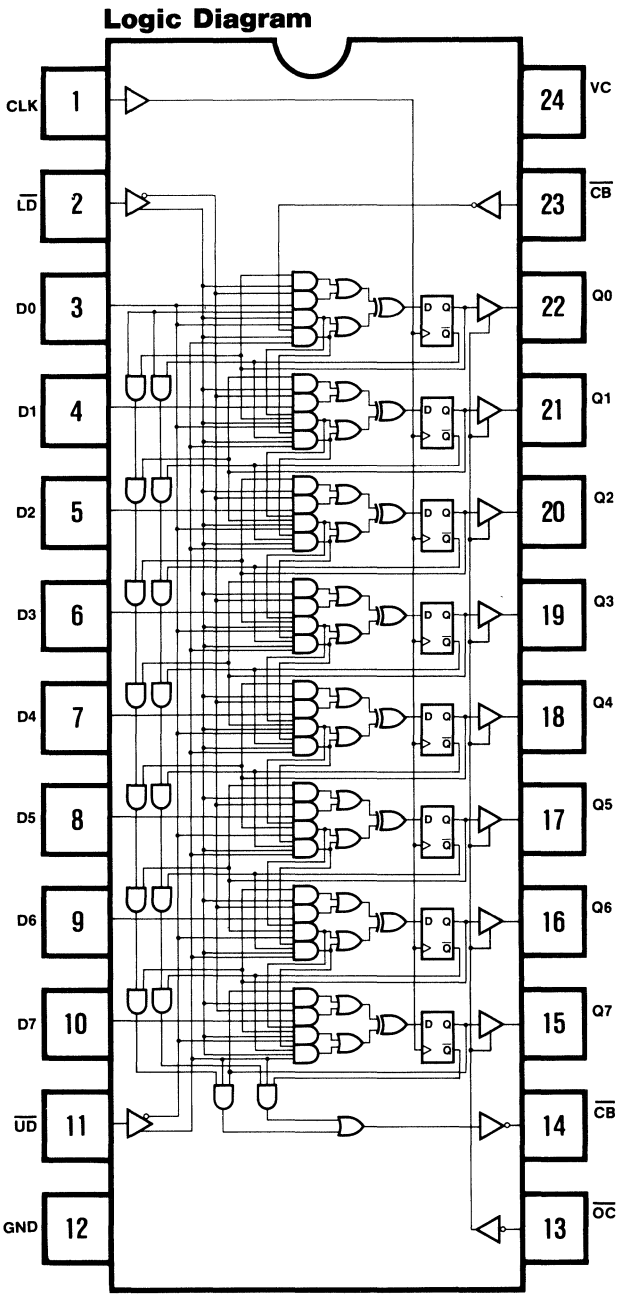
The LOAD operation loads the inputs (D_7-D_0) into the output register (Q_7-Q_0). The HOLD operation holds the previous value regardless of clock transitions. The INCREMENT operation adds one to the output register when the carry-in input is TRUE ($\overline{CBI}$ = LOW), otherwise the operation is a HOLD. The carry-out ($\overline{CBO}$) is TRUE ($\overline{CBO}$ = LOW) when the output register (Q_7-Q_0) is all HIGHs, otherwise FALSE ($\overline{CBO}$ = HIGH) The DECREMENT operation subtracts one from the output register when the borrow-in input is TRUE ($\overline{CBI}$ = LOW), otherwise the operation is a HOLD The borrow-out ($\overline{CBO}$) is TRUE ($\overline{CBO}$ = LOW) when the output register (Q_7-Q_0) is all LOWs, otherwise FALSE ($\overline{CBO}$ = HIGH)

The output register (Q_7-Q_0) is enabled when $\overline{OC}$ is LOW, and disabled (HI-Z) when $\overline{OC}$ is HIGH The output drivers will sink the 24 mA required for many bus interface standards Two or More PMSI401 octal up/down counters may be cascaded to provide larger counters.

Function Table

$\overline{OC}$	CLK	$\overline{LD}$	$\overline{UD}$	$\overline{CBI}$	D7-D0	Q7-Q0	OPERATION
H	X	X	X	X	X	Z	HI-Z
L	↑	L	X	X	D	D	LOAD
L	↑	H	L	H	X	Q	HOLD
L	↑	H	L	L	X	Q plus 1	INCREMENT
L	↑	H	H	H	X	Q	HOLD
L	↑	H	H	L	X	Q minus 1	DECREMENT

Logic Diagram



```
PAL20X8                                          PAL DESIGN SPECIFICATION
PMSI401                                            VINCENT COLI 07/24/81
OCTAL UP/DOWN COUNTER
MMI SUNNYVALE, CALIFORNIA
CLK /LD  D0 D1 D2 D3 D4 D5 D6 D7 /UD  GND
/OC /CBO Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0 /CBI VCC


/Q0 :=  /LD*/Q0                                          ;HOLD Q0
     +   LD*/D0                                          ;LOAD D0 (LSB)
    :+: /LD* UD* CBI                                     ;INCREMENT
     +  /LD*/UD* CBI                                     ;DECREMENT

/Q1 :=  /LD*/Q1                                          ;HOLD Q1
     +   LD*/D1                                          ;LOAD D1
    :+: /LD* UD* CBI* Q0                                 ;INCREMENT
     +  /LD*/UD* CBI*/Q0                                 ;DECREMENT

/Q2 :=  /LD*/Q2                                          ;HOLD Q2
     +   LD*/D2                                          ;LOAD D2
    :+: /LD* UD* CBI* Q0* Q1                             ;INCREMENT
     +  /LD*/UD* CBI*/Q0*/Q1                             ;DECREMENT

/Q3 :=  /LD*/Q3                                          ;HOLD Q3
     +   LD*/D3                                          ;LOAD D3
    :+: /LD* UD* CBI* Q0* Q1* Q2                         ;INCREMENT
     +  /LD*/UD* CBI*/Q0*/Q1*/Q2                         ;DECREMENT

/Q4 :=  /LD*/Q4                                          ;HOLD Q4
     +   LD*/D4                                          ;LOAD D4
    :+: /LD* UD* CBI* Q0* Q1* Q2* Q3                     ;INCREMENT
     +  /LD*/UD* CBI*/Q0*/Q1*/Q2*/Q3                     ;DECREMENT

/Q5 :=  /LD*/Q5                                          ;HOLD Q5
     +   LD*/D5                                          ;LOAD D5
    :+: /LD* UD* CBI* Q0* Q1* Q2* Q3* Q4                 ;INCREMENT
     +  /LD*/UD* CBI*/Q0*/Q1*/Q2*/Q3*/Q4                 ;DECREMENT

/Q6 :=  /LD*/Q6                                          ;HOLD Q6
     +   LD*/D6                                          ;LOAD D6
    :+: /LD* UD* CBI* Q0* Q1* Q2* Q3* Q4* Q5             ;INCREMENT
     +  /LD*/UD* CBI*/Q0*/Q1*/Q2*/Q3*/Q4*/Q5             ;DECREMENT

/Q7 :=  /LD*/Q7                                          ;HOLD Q7
     +   LD*/D7                                          ;LOAD D7 (MSB)
    :+: /LD* UD* CBI* Q0* Q1* Q2* Q3* Q4* Q5* Q6         ;INCREMENT
     +  /LD*/UD* CBI*/Q0*/Q1*/Q2*/Q3*/Q4*/Q5*/Q6         ;DECREMENT

IF(VCC) CBO =  UD* CBI* Q0* Q1* Q2* Q3* Q4* Q5* Q6* Q7 ;CARRY OUT
            + /UD* CBI*/Q0*/Q1*/Q2*/Q3*/Q4*/Q5*/Q6*/Q7 ;BORROW OUT
```

FUNCTION TABLE

CLK /OC /LD /UD D7 D6 D5 D4 D3 D2 D1 D0 /CBI /CBO Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0

; ;CLK	----CONTROL---- /OC	/LD	/UD	-INPUTS- DDDDDDDD 76543210	CARR/BORR /CBI	/CBO	OUTPUTS QQQQQQQQ 76543210	COMMENTS (HEX VALUES)
C	L	L	X	LLLLLLLH	X	X	LLLLLLLH	LOAD (01)
C	L	H	L	XXXXXXXX	L	H	LLLLLLHL	INCREMENT
C	L	H	H	XXXXXXXX	L	H	LLLLLLLH	DECREMENT
C	L	L	X	LLLLLLHH	X	X	LLLLLLHH	LOAD (03)
C	L	H	L	XXXXXXXX	L	H	LLLLLHLL	INCREMENT
C	L	H	H	XXXXXXXX	L	H	LLLLLLHH	DECREMENT
C	L	L	X	LLLLLHHH	X	X	LLLLLHHH	LOAD (07)
C	L	H	L	XXXXXXXX	L	H	LLLLHLLL	INCREMENT
C	L	H	H	XXXXXXXX	L	H	LLLLLHHH	DECREMENT
C	L	L	X	LLLLHHHH	X	X	LLLLHHHH	LOAD (0F)
C	L	H	L	XXXXXXXX	L	H	LLLHLLLL	INCREMENT
C	L	H	H	XXXXXXXX	L	H	LLLLHHHH	DECREMENT
C	L	L	X	LLLHHHHH	X	X	LLLHHHHH	LOAD (1F)
C	L	H	L	XXXXXXXX	L	H	LLHLLLLL	INCREMENT
C	L	H	H	XXXXXXXX	L	H	LLLHHHHH	DECREMENT
C	L	L	X	LLHHHHHH	X	X	LLHHHHHH	LOAD (3F)
C	L	H	L	XXXXXXXX	L	H	LHLLLLLL	INCREMENT
C	L	H	H	XXXXXXXX	L	H	LLHHHHHH	DECREMENT
C	L	L	X	LHHHHHHH	X	X	LHHHHHHH	LOAD (7F)
C	L	H	L	XXXXXXXX	L	H	HLLLLLLL	INCREMENT
C	L	H	H	XXXXXXXX	L	H	LHHHHHHH	DECREMENT
C	L	L	X	HHHHHHHH	X	X	HHHHHHHH	LOAD (FF)
C	L	L	X	LHHHHHHH	X	X	LHHHHHHH	LOAD (7F)
C	L	L	X	HLHHHHHH	X	X	HLHHHHHH	LOAD (BF)
C	L	L	X	HHLHHHHH	X	X	HHLHHHHH	LOAD (DF)
C	L	L	X	HHHLHHHH	X	X	HHHLHHHH	LOAD (EF)
C	L	L	X	HHHHLHHH	X	X	HHHHLHHH	LOAD (F7)
C	L	L	X	HHHHHLHH	X	X	HHHHHLHH	LOAD (FB)
C	L	L	X	HHHHHHLH	X	X	HHHHHHLH	LOAD (FD)
C	L	L	X	HHHHHHHL	X	X	HHHHHHHL	LOAD (FE)
C	L	H	L	XXXXXXXX	L	L	HHHHHHHH	INCREMENT TO (FF) (/CBO=L)
C	L	H	L	XXXXXXXX	L	H	LLLLLLLL	INCREMENT TO (00) (ROLL OVER)
C	L	H	L	XXXXXXXX	L	H	LLLLLLLH	INCREMENT TO (01)
C	L	H	L	XXXXXXXX	L	H	LLLLLLHL	INCREMENT TO (02)
C	L	H	L	XXXXXXXX	H	H	LLLLLLHL	HOLD
C	L	H	L	XXXXXXXX	L	H	LLLLLLHH	INCREMENT TO (03)
C	L	H	H	XXXXXXXX	L	H	LLLLLLHL	DECREMENT TO (02)
C	L	H	H	XXXXXXXX	L	H	LLLLLLLH	DECREMENT TO (01)
C	L	H	H	XXXXXXXX	L	L	LLLLLLLL	DECREMENT TO (00) (/CBO=L)
C	L	H	H	XXXXXXXX	L	H	HHHHHHHH	DECREMENT TO (FF)(ROLL UNDER)
C	L	H	H	XXXXXXXX	L	H	HHHHHHHL	DECREMENT TO (FE)
C	L	H	H	XXXXXXXX	L	H	HHHHHHLH	DECREMENT TO (FD)
C	L	H	H	XXXXXXXX	H	H	HHHHHHLH	HOLD
C	L	H	H	XXXXXXXX	L	H	HHHHHHLL	DECREMENT TO (FC)
X	H	X	X	XXXXXXXX	X	X	ZZZZZZZZ	TEST HI-Z

Octal Down Counter
PAL20X8
PMSI402

Features/Benefits

- **Octal counter for microprogram-counter, DMA controller and general purpose counting applications**
- **8 bits match byte boundaries**
- **Bus-structured pinout**
- **24-pin SKINNYDIP™ saves space**
- **3-state outputs drive bus lines**
- **Low current PNP inputs reduce loading**
- **Expandable in 8-bit increments**

Description

The PMSI402 is an 8-bit synchronous down counter with parallel load, preset, and hold capability. Two function select inputs (I_0, I_1) provide one of four operations which occur synchronously on the rising edge of the clock (CLK).

The LOAD operation loads the inputs (D_7-D_0) into the output register (Q_7-Q_0) The PRESET operation presets the output register to all Highs. The HOLD operation holds the previous value regardless of clock transitions. The DECREMENT operation subtracts one from the output register when the borrow-in input is TRUE ($\overline{BI}$ = LOW), otherwise the operation is a HOLD. The borrow-out ($\overline{BO}$) is TRUE ($\overline{BO}$ = LOW) when the output register (Q_7-Q_0) is all HIGHs, otherwise FALSE ($\overline{OC}$ = HIGH).

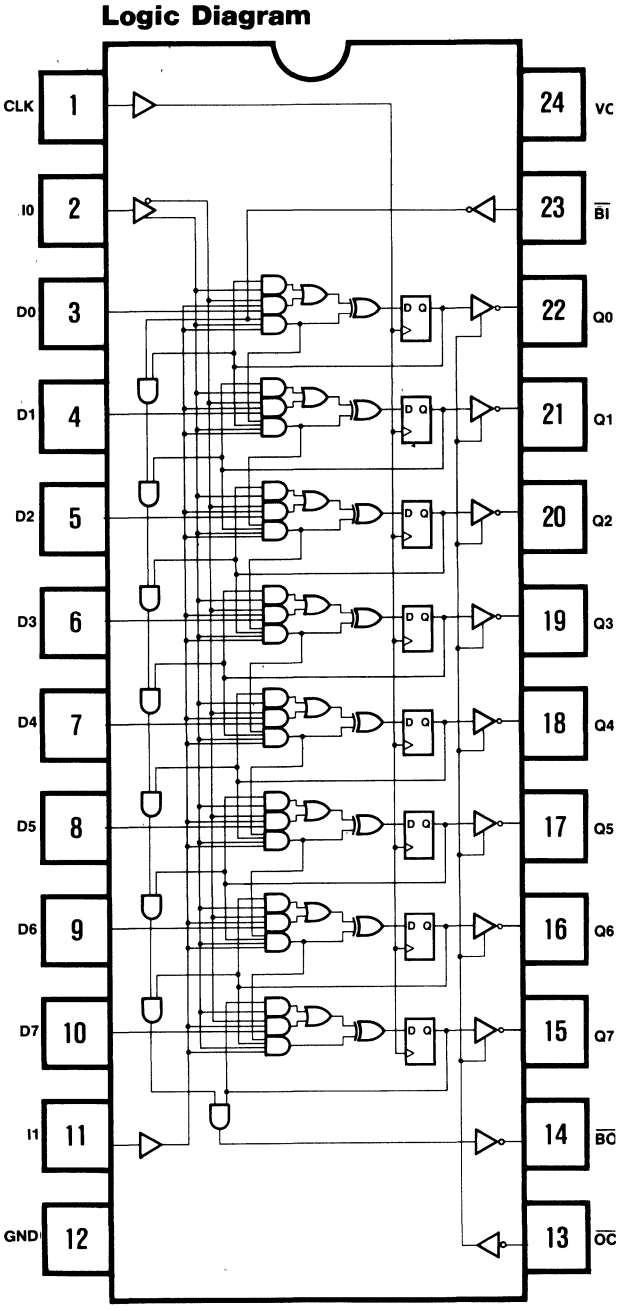
The output register (Q_7-Q_0) is enabled when $\overline{OC}$ is LOW, and disabled (HI-Z) when $\overline{OC}$ is HIGH. The output drivers will sink the 24 mA required for many bus interface standards.

Two or more PMSI402 octal down counters may be cascaded to provide larger counters. The operation codes were chosen such that when I_1 is HIGH. I_0 may be used to select between LOAD and DECREMENT.

Function Table

$\overline{OC}$	CLK	I1	I0	$\overline{BI}$	D7-D0	Q7-Q0	OPERATION
H	X	X	X	X	X	Z	HI-Z
L	↑	L	L	X	X	H	PRESET
L	↑	L	H	X	X	Q	HOLD
L	↑	H	L	X	D	D	LOAD
L	↑	H	H	H	X	Q	HOLD
L	↑	H	H	L	X	Q minus 1	DECREMENT

Logic Diagram



```
PAL20X8                                          PAL DESIGN SPECIFICATION
PMSI402                                            COLI/MITCHELL 09/09/82
OCTAL DOWN COUNTER
MMI SUNNYVALE, CALIFORNIA
CLK  I0 D0 D1 D2 D3 D4 D5 D6 D7  I1 GND
/OC /BO Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0 /BI VCC


/Q0 :=   I1*/I0*/D0                                  ;LOAD D0 (LSB)
     +   I1* I0* BI                                  ;COUNT DOWN
    :+:      I0*/Q0                                  ;COUNT DOWN/HOLD

/Q1 :=   I1*/I0*/D1                                  ;LOAD D1
     +   I1* I0* BI*/Q0                              ;COUNT DOWN
    :+:      I0*/Q1                                  ;COUNT DOWN/HOLD

/Q2 :=   I1*/I0*/D2                                  ;LOAD D2
     +   I1* I0* BI*/Q0*/Q1                          ;COUNT DOWN
    :+:      I0*/Q2                                  ;COUNT DOWN/HOLD

/Q3 :=   I1*/I0*/D3                                  ;LOAD D3
     +   I1* I0* BI*/Q0*/Q1*/Q2                      ;COUNT DOWN
    :+:      I0*/Q3                                  ;COUNT DOWN/HOLD

/Q4 :=   I1*/I0*/D4                                  ;LOAD D4
     +   I1* I0* BI*/Q0*/Q1*/Q2*/Q3                  ;COUNT DOWN
    :+:      I0*/Q4                                  ;COUNT DOWN/HOLD

/Q5 :=   I1*/I0*/D5                                  ;LOAD D5
     +   I1* I0* BI*/Q0*/Q1*/Q2*/Q3*/Q4              ;COUNT DOWN
    :+:      I0*/Q5                                  ;COUNT DOWN/HOLD

/Q6 :=   I1*/I0*/D6                                  ;LOAD D6
     +   I1* I0* BI*/Q0*/Q1*/Q2*/Q3*/Q4*/Q5          ;COUNT DOWN
    :+:      I0*/Q6                                  ;COUNT DOWN/HOLD

/Q7 :=   I1*/I0*/D7                                  ;LOAD D7 (MSB)
     +   I1* I0* BI*/Q0*/Q1*/Q2*/Q3*/Q4*/Q5*/Q6      ;COUNT DOWN
    :+:      I0*/Q7                                  ;COUNT DOWN/HOLD

IF (VCC)  BO = BI*/Q0*/Q1*/Q2*/Q3*/Q4*/Q5*/Q6*/Q7    ;BORROW OUT
```

FUNCTION TABLE

CLK /OC I1 I0 D7 D6 D5 D4 D3 D2 D1 D0 /BI /BO Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0

;CONTROL /CLK	/OC	INSTR I1	I0	-INPUT- DDDDDDDD 76543210	BORROW /BI	/BO	-OUTPUT- QQQQQQQQ 76543210	COMMENTS (HEX VALUES)
C	L	H	L	HHHHHHHL	X	H	HHHHHHHL	LOAD (FE)
C	L	H	H	XXXXXXXX	L	H	HHHHHHLH	DECREMENT
C	L	H	L	HHHHHHLL	X	H	HHHHHHLL	LOAD (FC)
C	L	H	H	XXXXXXXX	L	H	HHHHHLHH	DECREMENT
C	L	H	L	HHHHHLLL	X	H	HHHHHLLL	LOAD (F8)
C	L	H	H	XXXXXXXX	L	H	HHHHLHHH	DECREMENT
C	L	H	L	HHHHLLLL	X	H	HHHHLLLL	LOAD (F0)
C	L	H	H	XXXXXXXX	L	H	HHHLHHHH	DECREMENT
C	L	H	L	HHHLLLLL	X	H	HHHLLLLL	LOAD (E0)
C	L	H	H	XXXXXXXX	L	H	HHLHHHHH	DECREMENT
C	L	H	L	HHLLLLLL	X	H	HHLLLLLL	LOAD (C0)
C	L	H	H	XXXXXXXX	L	H	HLHHHHHH	DECREMENT
C	L	H	L	HLLLLLLL	X	H	HLLLLLLL	LOAD (80)
C	L	H	H	XXXXXXXX	L	H	LHHHHHHH	DECREMENT
C	L	H	L	LLLLLLLL	L	L	LLLLLLLL	LOAD (00)
C	L	H	H	XXXXXXXX	L	H	HHHHHHHH	DECREMENT (ROLL UNDER)
C	L	H	L	LLLLLLLL	L	L	LLLLLLLL	LOAD (00)
C	L	H	L	LLLLLLLH	X	H	LLLLLLLH	LOAD (01)
C	L	H	L	LLLLLLHL	X	H	LLLLLLHL	LOAD (02)
C	L	H	L	LLLLLHLL	X	H	LLLLLHLL	LOAD (04)
C	L	H	L	LLLLHLLL	X	H	LLLLHLLL	LOAD (08)
C	L	H	L	LLLHLLLL	X	H	LLLHLLLL	LOAD (10)
C	L	H	L	LLHLLLLL	X	H	LLHLLLLL	LOAD (20)
C	L	H	L	LHLLLLLL	X	H	LHLLLLLL	LOAD (40)
C	L	H	L	HLLLLLLL	X	H	HLLLLLLL	LOAD (80)
C	L	H	L	LLLLLLLL	L	L	LLLLLLLL	LOAD (00)
C	L	L	L	XXXXXXXX	X	H	HHHHHHHH	PRESET
C	L	H	H	XXXXXXXX	L	H	HHHHHHHL	DECREMENT TO (FE)
C	L	H	H	XXXXXXXX	L	H	HHHHHHLH	DECREMENT TO (FD)
C	L	H	H	XXXXXXXX	L	H	HHHHHHLL	DECREMENT TO (FC)
C	L	H	H	XXXXXXXX	L	H	HHHHHLHH	DECREMENT TO (FB)
C	L	H	H	XXXXXXXX	L	H	HHHHHLHL	DECREMENT TO (FA)
C	L	H	H	XXXXXXXX	L	H	HHHHHLLH	DECREMENT TO (F9)
C	L	H	H	XXXXXXXX	L	H	HHHHHLLL	DECREMENT TO (F8)
C	L	H	H	XXXXXXXX	L	H	HHHHLHHH	DECREMENT TO (F7)
C	L	H	H	XXXXXXXX	L	H	HHHHLHHL	DECREMENT TO (F6)
C	L	H	H	XXXXXXXX	L	H	HHHHLHLH	DECREMENT TO (F5)
C	L	H	H	XXXXXXXX	L	H	HHHHLHLL	DECREMENT TO (F4)
C	L	H	H	XXXXXXXX	L	H	HHHHLLHH	DECREMENT TO (F3)
C	L	H	H	XXXXXXXX	L	H	HHHHLLHL	DECREMENT TO (F2)
C	L	H	H	XXXXXXXX	L	H	HHHHLLLH	DECREMENT TO (F1)
C	L	H	H	XXXXXXXX	L	H	HHHHLLLL	DECREMENT TO (F0)
C	L	H	L	LLLLLLLH	X	H	LLLLLLLH	LOAD (01)
C	L	H	H	XXXXXXXX	L	L	LLLLLLLL	DECREMENT TO (00) /BO=L
C	L	H	H	XXXXXXXX	H	H	LLLLLLLL	BI INHIBITS COUNT AND BO
C	L	L	H	HHHHHHHH	L	L	LLLLLLLL	HOLD SEL INHIBITS COUNT ONLY
C	L	H	H	LLLLLLLL	L	H	HHHHHHHH	DECREMENT TO (FF)
X	H	X	X	XXXXXXXX	X	X	ZZZZZZZZ	TEST HI-Z

2-Digit BCD Counter
PAL20X8
PMSI403

Features/Benefits

- **Drive numeric displays**
- **Carry out provides expansion in 2-digit increments**
- **Bus structured pinout**
- **24-Pin SKINNYDIP™ saves space**
- **3-State outputs drive bus lines**
- **Low current PNP inputs reduce loading**

Description

The 2-digit BCD (Binary Coded Decimal) counter is a synchronous counter with complementary count enables, parallel load, and carry out. Three control inputs ($\overline{LD}$, $\overline{CE1}$, CE2) provide one of three operations which occur synchronously on the rising edge of the clock.

The LOAD operation loads the inputs (D1x, D2x) into the output register (Q1x, Q2x) when load is true ($\overline{LD}$ = L). Note that load overrides increment.

When LOAD is false ($\overline{LD}$ = H), the counter will increment in a BCD sequence if both count enables are asserted ($\overline{CE1}$ = L, CE2 = H), otherwise it holds. Also, two or more BCD counters can be cascaded to implement larger BCD counters by connecting carry out ($\overline{CO}$) of one stage to count enable ($\overline{CE1}$) of the next stage.

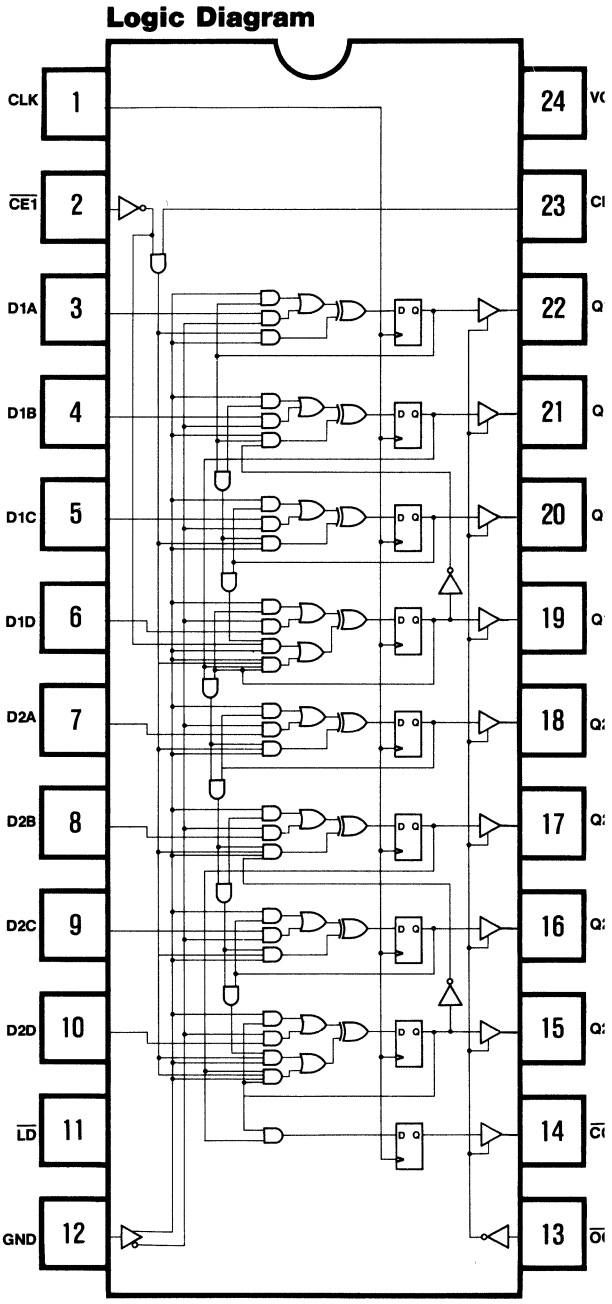
Parallel loading is provided for PAL and numeric indicator testibility. Load ($\overline{LD}$) can be changed to a clear function ($\overline{CLR}$) by tying the BCD parallel inputs (D1x, D2x) to ground.

This design is ideal in an industrial control application where an event counter is needed to drive numeric displays. The PAL can receive one count enable in the form of strobes from a motor or other device. The second count enable can receive the period signal. The PAL will provide two active high BCD outputs (Q1x, Q2x) to drive two numeric indicators, such as the Hewlett-Packard 5082-7300 which features on-board Decoder/Driver and Latch Enable.

Function Table

$\overline{OC}$	CLK	$\overline{LD}$	$\overline{CE1}$	CE2	BCD-IN	BCD-OUT	OPERATION
H	X	X	X	X	X	Z	HI-Z
L	↑	L	X	X	D	D	LOAD
L	↑	H	H	X	X	Q	HOLD ($\overline{CE1}$=H)
L	↑	H	X	L	X	Q	HOLD (CE2=L)
L	↑	H	L	H	X	Q PLUS 1	INCREMENT

Logic Diagram



```
PAL20X8                                                    PAL DESIGN SPECIFICATION
PMSI403                                                      VINCENT COLI 06/21/82
2-DIGIT BCD COUNTER
MMI SUNNYVALE, CALIFORNIA
CLK /CE1 D1A D1B D1C D1D D2A D2B D2C D2D /LD GND
/OC /CO  Q2D Q2C Q2B Q2A Q1D Q1C Q1B Q1A CE2 VCC


/Q1A :=  /LD*/Q1A                                  ;HOLD FIRST BIT OF 1 DECIMAL
      +   LD*/D1A                                  ;LOAD D1A
     :+: /LD* CE1* CE2                             ;COUNT

/Q1B :=  /LD*/Q1B                                  ;HOLD SECOND BIT OF 1 DECIMAL
      +   LD*/D1B                                  ;LOAD D1B
     :+: /LD* CE1* CE2* Q1A*/Q1D                   ;COUNT

/Q1C :=  /LD*/Q1C                                  ;HOLD THIRD BIT OF 1 DECIMAL
      +   LD*/D1C                                  ;LOAD D1C
     :+: /LD* CE1* CE2* Q1A* Q1B                   ;COUNT

/Q1D :=  /LD*/Q1D                                  ;HOLD FOURTH BIT OF 1 DECIMAL
      +   LD*/D1D                                  ;LOAD D1D
     :+: /LD* CE1* Q1A* Q1B* Q1C                   ;COUNT
      +  /LD* CE1* CE2* Q1A* Q1D                   ;ROLL OVER

/Q2A :=  /LD*/Q2A                                  ;HOLD FIRST BIT OF 10 DECIMAL
      +   LD*/D2A                                  ;LOAD D2A
     :+: /LD* CE1* CE2* Q1A* Q1D                   ;COUNT

/Q2B :=  /LD*/Q2B                                  ;HOLD SECOND BIT OF 10 DECIMAL
      +   LD*/D2B                                  ;LOAD D2B
     :+: /LD* CE1* CE2* Q1A* Q1D* Q2A*/Q2D         ;COUNT

/Q2C :=  /LD*/Q2C                                  ;HOLD THIRD BIT OF 10 DECIMAL
      +   LD*/D2C                                  ;LOAD D2C
     :+: /LD* CE1* CE2* Q1A* Q1D* Q2A* Q2B         ;COUNT

/Q2D :=  /LD*/Q2D                                  ;HOLD FOURTH BIT OF 10 DECIMAL
      +   LD*/D2D                                  ;LOAD D2D
     :+: /LD* CE1* CE2* Q1A* Q1D* Q2A* Q2B* Q2C    ;COUNT
      +  /LD* CE1* CE2* Q1A* Q1D* Q2A* Q2D         ;ROLL OVER

IF (VCC)  CO =  Q1A* Q1D* Q2A* Q2D                 ;CARRY OUT (10 DECIMAL)
```

FUNCTION TABLE

CLK /OC /LD /CE1 CE2 D2D D2C D2B D2A D1D D1C D1B D1A /CO
Q2D Q2C Q2B Q2A Q1D Q1C Q1B Q1A

; CHIP											
					--INPUTS---			--OUTPUTS--			
;CONTROL		INSTRUCTIONS			BCD 2	BCD 1		BCD 2	BCD 1		COMMENT
;CLK	/OC	/LD	/CE1	CE2	DCBA	DCBA	/CO	DCBA	DCBA		(DECIMAL VALUE)
C	L	L	L	H	LLLL	LLLL	H	LLLL	LLLL		LOAD (00) /LD=L
C	L	H	H	H	XXXX	XXXX	H	LLLL	LLLL		HOLD (00)/CE1=H
C	L	H	L	H	XXXX	XXXX	H	LLLL	LLLH		COUNT (01)
C	L	H	L	H	XXXX	XXXX	H	LLLL	LLHL		COUNT (02)
C	L	H	L	H	XXXX	XXXX	H	LLLL	LLHH		COUNT (03)
C	L	H	L	H	XXXX	XXXX	H	LLLL	LHLL		COUNT (04)
C	L	H	L	H	XXXX	XXXX	H	LLLL	LHLH		COUNT (05)
C	L	H	L	H	XXXX	XXXX	H	LLLL	LHHL		COUNT (06)
C	L	H	L	H	XXXX	XXXX	H	LLLL	LHHH		COUNT (07)
C	L	H	L	H	XXXX	XXXX	H	LLLL	HLLL		COUNT (08)
C	L	H	L	H	XXXX	XXXX	H	LLLL	HLLH		COUNT (09)
C	L	H	L	H	XXXX	XXXX	H	LLLH	LLLL		COUNT (10)
C	L	H	L	H	XXXX	XXXX	H	LLLH	LLLH		COUNT (11)
C	L	L	H	L	LLLH	HLLH	H	LLLH	HLLH		LOAD (19)
C	L	H	L	H	XXXX	XXXX	H	LLHL	LLLL		COUNT (20)
C	L	L	H	L	LLHL	HLLH	H	LLHL	HLLH		LOAD (29)
C	L	H	L	H	XXXX	XXXX	H	LLHH	LLLL		COUNT (30)
C	L	L	H	L	LLHH	HLLH	H	LLHH	HLLH		LOAD (39)
C	L	H	L	H	XXXX	XXXX	H	LHLL	LLLL		COUNT (40)
C	L	L	H	L	LHLL	HLLH	H	LHLL	HLLH		LOAD (49)
C	L	H	L	H	XXXX	XXXX	H	LHLH	LLLL		COUNT (50)
C	L	L	H	L	LHLH	HLLH	H	LHLH	HLLH		LOAD (59)
C	L	H	L	H	XXXX	XXXX	H	LHHL	LLLL		COUNT (60)
C	L	L	H	L	LHHL	HLLH	H	LHHL	HLLH		LOAD (69)
C	L	H	L	H	XXXX	XXXX	H	LHHH	LLLL		COUNT (70)
C	L	L	H	L	LHHH	HLLH	H	LHHH	HLLH		LOAD (79)
C	L	H	L	H	XXXX	XXXX	H	HLLL	LLLL		COUNT (80)
C	L	L	H	L	HLLL	HLLH	H	HLLL	HLLH		LOAD (89)
C	L	H	L	H	XXXX	XXXX	H	HLLH	LLLL		COUNT (90)
C	L	L	H	L	HLLH	HLLH	L	HLLH	HLLH		LOAD (99) /CO=L
C	L	H	L	H	XXXX	XXXX	H	LLLL	LLLL		COUNT (00)
C	L	H	L	L	XXXX	XXXX	H	LLLL	LLLL		HOLD (00) CE2=L
X	H	X	X	X	XXXX	XXXX	X	ZZZZ	ZZZZ		TEST HI-Z

9-Bit Counter
PAL 20X10
PMSI404

Features/Benefits

- **9 bits is ideal for parity in/out application**
- **Bus-structured pinout**
- **24-pin SKINNYDIP™ saves space**
- **3-state outputs drive bus lines**
- **Low current PNP inputs reduce loading**

Description

The PMSI404 is a 9-bit synchronous counter with parallel load, increment, and hold capability.

The LOAD operation loads the inputs (D8-D0) into the output register (Q8-Q0) when load is TRUE ($\overline{LD}$=LOW) and a positive edge pulse is received on the clock pin (CLK). The INCREMENT operation adds one to the output register when load is FALSE ($\overline{LD}$=HIGH) and a positive edge clock pulse is received. The HOLD operation holds the previous value in the output register if no clock pulse is received regardless of any other inputs.

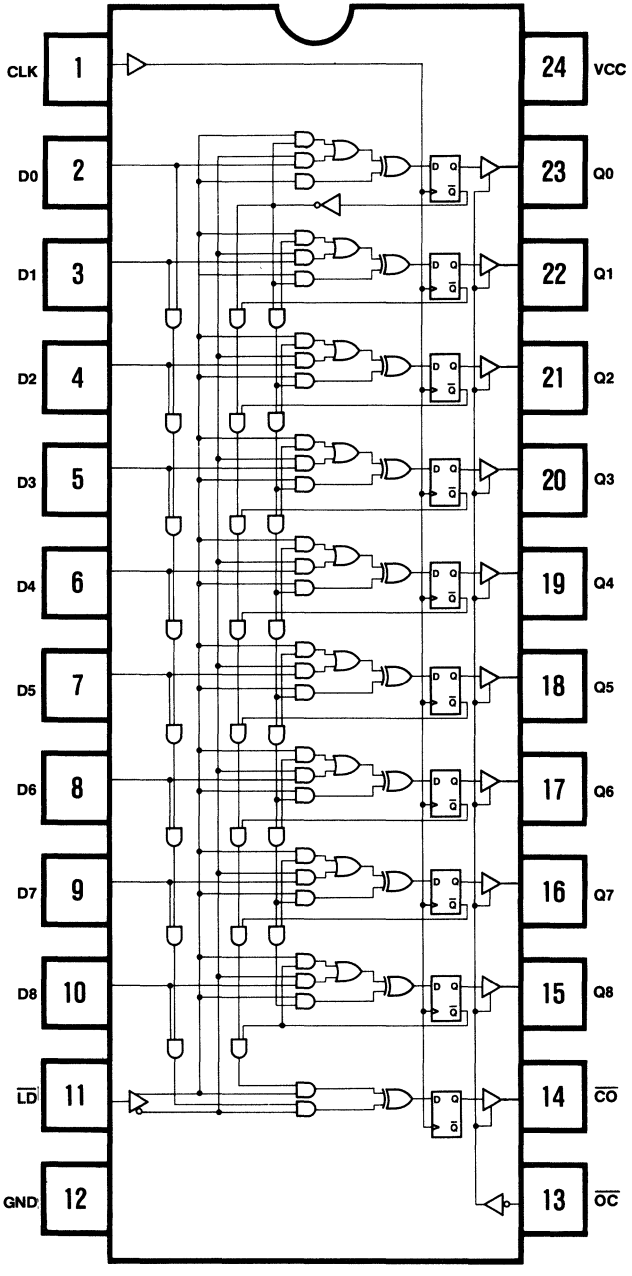
A registered active low carry out is provided. Carry out will be TRUE ($\overline{CO}$=LOW) when the output register (Q8-Q0) is all HIGHs, otherwise FALSE ($\overline{CO}$=HIGH).

The output register (Q8-Q0) and carry out ($\overline{CO}$) are enabled when $\overline{OC}$ is LOW, and disabled (HI-Z) when $\overline{OC}$ is HIGH. The output drivers will sink the 24 mA required for many bus interface standards.

Function Table

$\overline{OC}$	CLK	$\overline{LD}$	D8-D0	Q8-Q0	OPERATION
H	X	X	X	Z	HI-Z
L	L	X	X	Q	HOLD
L	↑	L	D	D	LOAD
L	↑	H	X	Q PLUS 1	INCREMENT

Logic Diagram



```
PAL20X10                                              PAL DESIGN SPECIFICATION
PMSI404                                                  BIRKNER/COLI 05/07/82
9-BIT COUNTER
MMI SUNNYVALE, CALIFORNIA
CLK  D0 D1 D2 D3 D4 D5 D6 D7 D8 /LD GND
/OC /CO Q8 Q7 Q6 Q5 Q4 Q3 Q2 Q1  Q0 VCC


/Q0 :=  /LD*/Q0                                       ;HOLD Q0
     +   LD*/D0                                       ;LOAD D0 (LSB)
    :+: /LD                                           ;COUNT

/Q1 :=  /LD*/Q1                                       ;HOLD Q1
     +   LD*/D1                                       ;LOAD D1
    :+: /LD* Q0                                       ;COUNT

/Q2 :=  /LD*/Q2                                       ;HOLD Q2
     +   LD*/D2                                       ;LOAD D2
    :+: /LD* Q0* Q1                                   ;COUNT

/Q3 :=  /LD*/Q3                                       ;HOLD Q3
     +   LD*/D3                                       ;LOAD D3
    :+: /LD* Q0* Q1* Q2                               ;COUNT

/Q4 :=  /LD*/Q4                                       ;HOLD Q4
     +   LD*/D4                                       ;LOAD D4
    :+: /LD* Q0* Q1* Q2* Q3                           ;COUNT

/Q5 :=  /LD*/Q5                                       ;HOLD Q5
     +   LD*/D5                                       ;LOAD D5
    :+: /LD* Q0* Q1* Q2* Q3* Q4                       ;COUNT

/Q6 :=  /LD*/Q6                                       ;HOLD Q6
     +   LD*/D6                                       ;LOAD D6
    :+: /LD* Q0* Q1* Q2* Q3* Q4* Q5                   ;COUNT

/Q7 :=  /LD*/Q7                                       ;HOLD Q7
     +   LD*/D7                                       ;LOAD D7
    :+: /LD* Q0* Q1* Q2* Q3* Q4* Q5* Q6               ;COUNT

/Q8 :=  /LD*/Q8                                       ;HOLD Q8
     +   LD*/D8                                       ;LOAD D8 (MSB)
    :+: /LD* Q0* Q1* Q2* Q3* Q4* Q5* Q6* Q7           ;COUNT

 CO :=  /LD*/Q0* Q1* Q2* Q3* Q4* Q5* Q6* Q7* Q8       ;CARRY OUT (ANTICIPATE COUNT)
     +   LD* D0* D1* D2* D3* D4* D5* D6* D7* D8       ;CARRY OUT (ANTICIPATE LOAD)
```

FUNCTION TABLE

```
CLK /OC /LD D8 D7 D6 D5 D4 D3 D2 D1 D0 /CO Q8 Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0

;                 DATA IN          DATA  OUT
;  CONTROL        DDDDDDDDD        QQQQQQQQQ
;CLK /OC /LD     876543210   /CO   876543210     COMMENT
-------------------------------------------------------------------------
  C   L   L     LLLLLLLLL    H     LLLLLLLLL     LOAD
  C   L   H     XXXXXXXXX    H     LLLLLLLLH     INCREMENT
  C   L   L     LLLLLLLLH    H     LLLLLLLLH     LOAD
  C   L   H     XXXXXXXXX    H     LLLLLLLHL     INCREMENT
  C   L   L     LLLLLLLHH    H     LLLLLLLHH     LOAD
  C   L   H     XXXXXXXXX    H     LLLLLLHLL     INCREMENT
  C   L   L     LLLLLLHHH    H     LLLLLLHHH     LOAD
  C   L   H     XXXXXXXXX    H     LLLLLHLLL     INCREMENT
  C   L   L     LLLLLHHHH    H     LLLLLHHHH     LOAD
  C   L   H     XXXXXXXXX    H     LLLLHLLLL     INCREMENT
  C   L   L     LLLLHHHHH    H     LLLLHHHHH     LOAD
  C   L   H     XXXXXXXXX    H     LLLHLLLLL     INCREMENT
  C   L   L     LLLHHHHHH    H     LLLHHHHHH     LOAD
  C   L   H     XXXXXXXXX    H     LLHLLLLLL     INCREMENT
  C   L   L     LLHHHHHHH    H     LLHHHHHHH     LOAD
  C   L   H     XXXXXXXXX    H     LHLLLLLLL     INCREMENT
  C   L   L     LHHHHHHHH    H     LHHHHHHHH     LOAD
  C   L   H     XXXXXXXXX    H     HLLLLLLLL     INCREMENT
  C   L   L     HHHHHHHHH    L     HHHHHHHHH     LOAD (CARRY OUT)
  C   L   H     XXXXXXXXX    H     LLLLLLLLL     INCREMENT (ROLL OVER)
  C   L   H     XXXXXXXXX    H     LLLLLLLLH     INCREMENT
  C   L   H     XXXXXXXXX    H     LLLLLLLHL     INCREMENT
  C   L   H     XXXXXXXXX    H     LLLLLLLHH     INCREMENT
  C   L   L     HHHHHHHLL    H     HHHHHHHLL     LOAD
  C   L   H     XXXXXXXXX    H     HHHHHHHLH     INCREMENT
  L   L   H     XXXXXXXXX    H     HHHHHHHLH     HOLD
  C   L   H     XXXXXXXXX    H     HHHHHHHHL     INCREMENT
  C   L   H     XXXXXXXXX    L     HHHHHHHHH     INCREMENT (CARRY OUT)
  C   L   H     XXXXXXXXX    H     LLLLLLLLL     INCREMENT (ROLL OVER)
  X   H   X     XXXXXXXXX    Z     ZZZZZZZZZ     TEST HI-Z
-------------------------------------------------------------------------
```

9-Bit Register
PAL 20X10
PMSI405

Features/Benefits

- **9 bits is ideal for parity bus interfacing**
- **Bus-structured pinout**
- **24-pin SKINNYDIP™ saves space**
- **3-state outputs drive bus lines**
- **Low current PNP inputs reduce loading**
- **Expandable in 9-bit increments**

Description

The PMSI405 is a 9-bit synchronous register with parallel load and hold capability.

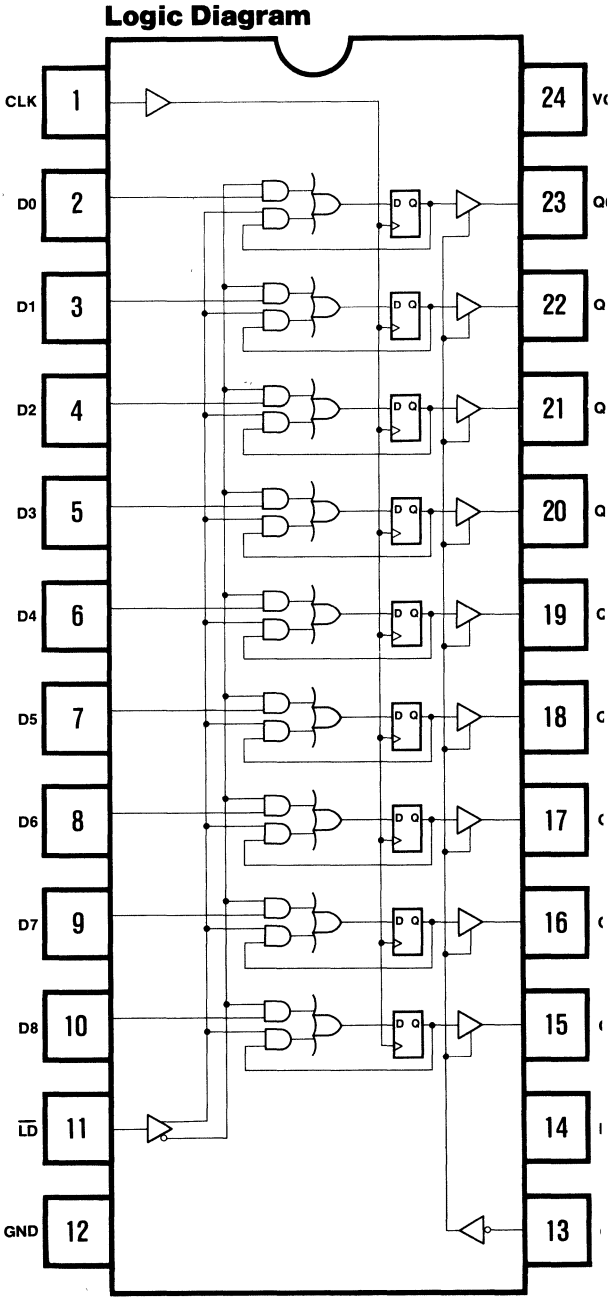
LOAD operation loads the inputs (D_8-D_0) into the output register (Q_8-Q_0). The HOLD operation holds the previous value regardless of clock transitions.

The output register (Q_8-Q_9) is enabled when $\overline{OC}$ is LOW, and disabled (HI-Z) when $\overline{OC}$ is HIGH. The output drivers will sink the 24 mA required for many bus interface standards.

Function Table

$\overline{OC}$	CLK	$\overline{LD}$	D8-D0	Q8-Q0	OPERATION
H	X	X	X	Z	HI-Z
L	↑	H	X	Q	HOLD
L	↑	L	D	D	LOAD

Logic Diagram



```
PAL20X10                                        PAL DESIGN SPECIFICATION
PMSI405                                            BIRKNER/COLI 07/19/81
9-BIT REGISTER
MMI SUNNYVALE, CALIFORNIA
CLK D0 D1 D2 D3 D4 D5 D6 D7 D8 /LD GND
/OC NC Q8 Q7 Q6 Q5 Q4 Q3 Q2 Q1  Q0 VCC


/Q0 := /Q0*/LD      ;HOLD Q0
     + /D0* LD      ;LOAD D0

/Q1 := /Q1*/LD      ;HOLD Q1
     + /D1* LD      ;LOAD D1

/Q2 := /Q2*/LD      ;HOLD Q2
     + /D2* LD      ;LOAD D2

/Q3 := /Q3*/LD      ;HOLD Q3
     + /D3* LD      ;LOAD D3

/Q4 := /Q4*/LD      ;HOLD Q4
     + /D4* LD      ;LOAD D4

/Q5 := /Q5*/LD      ;HOLD Q5
     + /D5* LD      ;LOAD D5

/Q6 := /Q6*/LD      ;HOLD Q6
     + /D6* LD      ;LOAD D6

/Q7 := /Q7*/LD      ;HOLD Q7
     + /D7* LD      ;LOAD D7

/Q8 := /Q8*/LD      ;HOLD Q8
     + /D8* LD      ;LOAD D8
```

FUNCTION TABLE

```
/OC CLK /LD D8 D7 D6 D5 D4 D3 D2 D1 D0 Q8 Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0

;/ C /    DATA IN     DATA OUT
;O L L   DDDDDDDDD   QQQQQQQQQ
;C K D   876543210   876543210     COMMENT
-------------------------------------------------------------
 L C L   LLLLLLLLL   LLLLLLLLL     LOAD ALL ZEROS
 L C H   XXXXXXXXX   LLLLLLLLL     HOLD ALL ZEROS
 L C L   HHHHHHHHH   HHHHHHHHH     LOAD ALL ONES
 L C H   XXXXXXXXX   HHHHHHHHH     HOLD ALL ONES
 L C L   LHLHLHLHL   LHLHLHLHL     LOAD EVEN CHECKERBOARD
 L C H   XXXXXXXXX   LHLHLHLHL     HOLD EVEN CHECKERBOARD
 L C L   HLHLHLHLH   HLHLHLHLH     LOAD ODD  CHECKERBOARD
 L C H   XXXXXXXXX   HLHLHLHLH     HOLD ODD  CHECKERBOARD
 H X X   XXXXXXXXX   ZZZZZZZZZ     TEST HI-Z
-------------------------------------------------------------
```

10-Bit Register
PAL 20X10
PMSI406

Features/Benefits

- **Bus-structured pinout**
- **24-pin SKINNYDIP™ saves space**
- **3-state outputs drive bus lines**
- **Low current PNP inputs reduce loading**
- **Expandable in 10-bit increments**

Description

The PMSI406 is a 10-bit synchronous register with parallel load and hold capability.

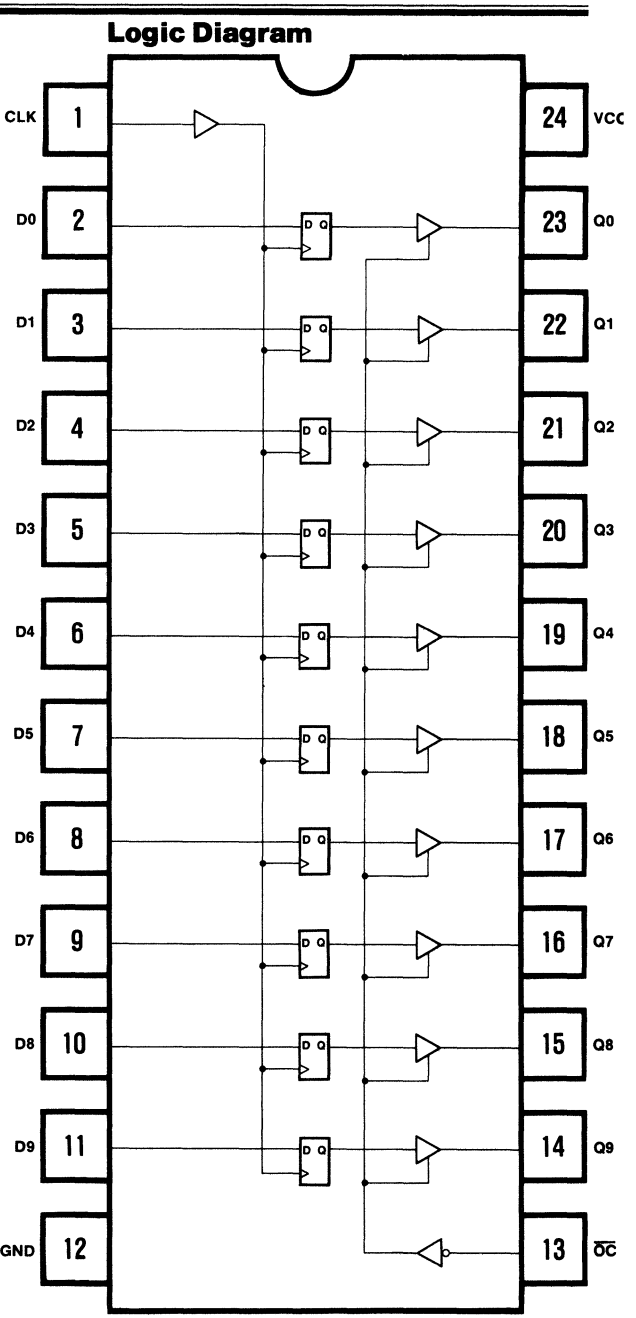
The LOAD operation loads the inputs (D_9-D_0) into the output register (Q_9-Q_0) synchronous with the positive edge of the clock. The HOLD operation holds the data until the next clock pulse.

The output register (Q_9-Q_0) is enabled when $\overline{OC}$ is LOW, and disabled (HI-Z) when $\overline{OC}$ is HIGH. The output drivers will sink the 24 mA required for many bus interface standards.

Function Table

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

Logic Diagram



```
PAL20X10                                        PAL DESIGN SPECIFICATION
PMSI406                                           VINCENT COLI 07/27/82
10-BIT REGISTER
MMI SUNNYVALE, CALIFORNIA
CLK D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 GND
/OC Q9 Q8 Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0 VCC

/Q0 := /D0      ;LOAD D0

/Q1 := /D1      ;LOAD D1

/Q2 := /D2      ;LOAD D2

/Q3 := /D3      ;LOAD D3

/Q4 := /D4      ;LOAD D4

/Q5 := /D5      ;LOAD D5

/Q6 := /D6      ;LOAD D6

/Q7 := /D7      ;LOAD D7

/Q8 := /D8      ;LOAD D8

/Q9 := /D9      ;LOAD D9
```

FUNCTION TABLE

/OC CLK D9 D8 D7 D6 D5 D4 D3 D2 D1 D0 Q9 Q8 Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0

;CONTROL		DATA IN	DATA OUT	
;/OC	CLK	9876543210	9876543210	COMMENT
L	C	LLLLLLLLLL	LLLLLLLLLL	LOAD ALL ZEROS
L	L	XXXXXXXXXX	LLLLLLLLLL	HOLD ALL ZEROS
L	C	HHHHHHHHHH	HHHHHHHHHH	LOAD ALL ONES
L	L	XXXXXXXXXX	HHHHHHHHHH	HOLD ALL ONES
L	C	HLHLHLHLHL	HLHLHLHLHL	LOAD EVEN CHECKERBOARD
L	L	XXXXXXXXXX	HLHLHLHLHL	HOLD EVEN CHECKERBOARD
L	C	LHLHLHLHLH	LHLHLHLHLH	LOAD ODD CHECKERBOARD
L	L	XXXXXXXXXX	LHLHLHLHLH	HOLD ODD CHECKERBOARD
H	X	XXXXXXXXXX	ZZZZZZZZZZ	TEST HI-Z

10-Bit Addressable Register
PAL20X10
PMSI407

Features/Benefits

- **10-bit addressable register with clear, preset and enable for general purpose applications**
- **10-bits exceed byte boundaries to give 2 additional bits**
- **24-pin SKINNYDIP™ saves space**
- **3-state outputs for bus line**
- **Low current PNP input reduces loading**

Description

The 10-bit addressable register is a synchronous general purpose addressable register with clear, preset and enable. The output is selected by the input address pins (A, B, C, D). The selected output loads the data (DIN) on the rising edge of the clock if the chip is enabled (E1 = HIGH, E2 = HIGH, $\overline{E3}$ = LOW); all other outputs hold their previous states. Any other combination of enable pins (E1, E2, $\overline{E3}$) will disable the register and all outputs will hold their previous states. Clear ($\overline{CLR}$) and preset ($\overline{PR}$) are active low pins which set all outputs to low or high respectively. Clear overrides preset and enable Preset overrides enable. $\overline{OC}$ = HIGH puts all outputs into a high impedance state.

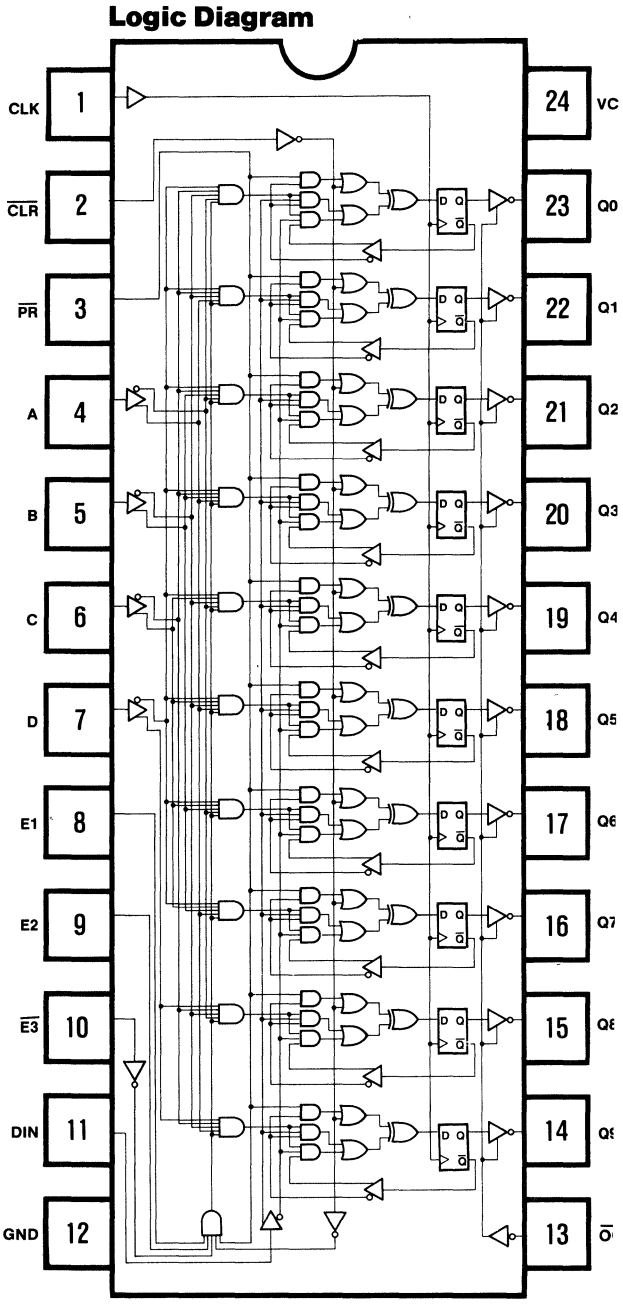
Output Select Table

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

Function Table

$\overline{OC}$	CLK	$\overline{CLR}$	$\overline{PR}$	E3	$\overline{E2}$	E1	D	C	B	A	DIN	Q9-Q0	OPERATION
H	X	X	X	X	X	X	X	X	X	X	X	Z	Hi-Z
L	C	L	L	X	X	X	X	X	X	X	X	L	Clear
L	C	H	L	X	X	X	X	X	X	X	X	H	Preset
L	C	H	H	L	L	L	X	X	X	X	X	Q	Hold previous states
L	C	H	H	L	L	H	X	X	X	X	X	Q	Hold previous states
L	C	H	H	L	H	L	X	X	X	X	X	Q	Hold previous states
L	C	H	H	L	H	H	D	C	B	A	DIN	DIN	Load DIN to addressed output
L	C	H	H	H	L	L	X	X	X	X	X	Q	Hold previous states
L	C	H	H	H	L	H	X	X	X	X	X	Q	Hold previous states
L	C	H	H	H	H	L	X	X	X	X	X	Q	Hold previous states
L	C	H	H	H	H	H	X	X	X	X	X	Q	Hold previous states

Logic Diagram



```
PAL20X10                                                    PAL DESIGN SPECIFICATION
PMSI407                                                       DANESH TAVANA 04/05/82
10-BIT ADDRESSABLE REGISTER
MMI SUNNYVALE, CALIFORNIA
CLK /CLR /PR A  B  C  D  E1 E2 /E3 DIN GND
/OC  Q9   Q8 Q7 Q6 Q5 Q4 Q3 Q2  Q1 Q0  VCC


/Q0 :=   CLR                                          ;CLEAR (LSB)
     +  /PR*/Q0                                       ;HOLD (/Q0)
    :+: /PR*/CLR* E1* E2* E3*/D*/C*/B*/A*/Q0*.DIN     ;LOAD ( DIN:+:/Q0) IF /Q0=H
     +  /PR*/CLR* E1* E2* E3*/D*/C*/B*/A* Q0*/DIN     ;LOAD ( DIN:+:/Q0) IF /Q0=L

/Q1 :=   CLR                                          ;CLEAR
     +  /PR*/Q1                                       ;HOLD (/Q1)
    :+: /PR*/CLR* E1* E2* E3*/D*/C*/B* A*/Q1* DIN     ;LOAD ( DIN:+:/Q1) IF /Q1=H
     +  /PR*/CLR* E1* E2* E3*/D*/C*/B* A* Q1*/DIN     ;LOAD (/DIN:+:/Q1) IF /Q1=L

/Q2 :=   CLR                                          ;CLEAR
     +  /PR*/Q2                                       ;HOLD (/Q2)
    :+: /PR*/CLR* E1* E2* E3*/D*/C* B*/A*/Q2* DIN     ;LOAD ( DIN:+:/Q2) IF /Q2=H
     +  /PR*/CLR* E1* E2* E3*/D*/C* B*/A* Q2*/DIN     ;LOAD (/DIN:+:/Q2) IF /Q2=L

/Q3 :=   CLR                                          ;CLEAR
     +  /PR*/Q3                                       ;HOLD (/Q3)
    :+: /PR*/CLR* E1* E2* E3*/D*/C* B* A*/Q3* DIN     ;LOAD ( DIN:+:/Q3) IF /Q3=H
     +  /PR*/CLR* E1* E2* E3*/D*/C* B* A* Q3*/DIN     ;LOAD (/DIN:+:/Q3) IF /Q3=L

/Q4 :=   CLR                                          ;CLEAR
     +  /PR*/Q4                                       ;HOLD (/Q4)
    :+: /PR*/CLR* E1* E2* E3*/D* C*/B*/A*/Q4* DIN     ;LOAD ( DIN:+:/Q4) IF /Q4=H
     +  /PR*/CLR* E1* E2* E3*/D* C*/B*/A* Q4*/DIN     ;LOAD (/DIN:+:/Q4) IF /Q4=L

/Q5 :=   CLR                                          ;CLEAR
     +  /PR*/Q5                                       ;HOLD (/Q5)
    :+: /PR*/CLR* E1* E2* E3*/D* C*/B* A*/Q5* DIN     ;LOAD ( DIN:+:/Q5) IF /Q5=H
     +  /PR*/CLR* E1* E2* E3*/D* C*/B* A* Q5*/DIN     ;LOAD (/DIN:+:/Q5) IF /Q5=L

/Q6 :=   CLR                                          ;CLEAR
     +  /PR*/Q6                                       ;HOLD (/Q6)
    :+: /PR*/CLR* E1* E2* E3*/D* C* B*/A*/Q6* DIN     ;LOAD ( DIN:+:/Q6) IF /Q6=H
     +  /PR*/CLR* E1* E2* E3*/D* C* B*/A* Q6*/DIN     ;LOAD (/DIN:+:/Q6) IF /Q6=L

/Q7 :=   CLR                                          ;CLEAR
     +  /PR*/Q7                                       ;HOLD (/Q7)
    :+: /PR*/CLR* E1* E2* E3*/D* C* B* A*/Q7* DIN     ;LOAD ( DIN:+:/Q7) IF /Q7=H
     +  /PR*/CLR* E1* E2* E3*/D* C* B* A* Q7*/DIN     ;LOAD (/DIN:+:/Q7) IF /Q7=L

/Q8 :=   CLR                                          ;CLEAR
     +  /PR*/Q8                                       ;HOLD (/Q8)
    :+: /PR*/CLR* E1* E2* E3* D*/C*/B*/A*/Q8* DIN     ;LOAD ( DIN:+:/Q8) IF /Q8=H
     +  /PR*/CLR* E1* E2* E3* D*/C*/B*/A* Q8*/DIN     ;LOAD (/DIN:+:/Q8) IF /Q8=L

/Q9 :=   CLR                                          ;CLEAR (MSB)
     +  /PR*/Q9                                       ;HOLD (/Q9)
    :+: /PR*/CLR* E1* E2* E3* D*/C*/B* A*/Q9* DIN     ;LOAD ( DIN:+:/Q9) IF /Q9=H
     +  /PR*/CLR* E1* E2* E3* D*/C*/B* A* Q9*/DIN     ;LOAD (/DIN:+:/Q9) IF /Q9=L
```

FUNCTION TABLE

/OC CLK /CLR /PR /E3 E2 E1 D C B A DIN Q9 Q8 Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0

;CONTROL		-----FUNCTIONS----					---INPUTS--					-----OUTPUTS-------										
;/OC	CLK	/CLR	/PR	/E3	E2	E1	D	C	B	A	DIN	Q9	Q8	Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0	COMMENTS
L	C	L	L	X	X	X	X	X	X	X	X	L	L	L	L	L	L	L	L	L	L	/CLR OVRRD /PR
L	C	H	L	X	X	X	X	X	X	X	X	H	H	H	H	H	H	H	H	H	H	/PR OVRRD ENABLE
L	C	H	H	L	H	H	L	L	L	L	L	H	H	H	H	H	H	H	H	H	L	LOAD Q0 WITH DIN
L	C	H	H	L	H	H	L	L	L	H	L	H	H	H	H	H	H	H	H	L	L	LOAD Q1 WITH DIN
L	C	H	H	L	H	H	L	L	H	L	L	H	H	H	H	H	H	H	L	L	L	LOAD Q2 WITH DIN
L	C	H	H	L	H	H	L	L	H	H	L	H	H	H	H	H	H	L	L	L	L	LOAD Q3 WITH DIN
L	C	H	H	L	H	H	L	H	L	L	L	H	H	H	H	H	L	L	L	L	L	LOAD Q4 WITH DIN
L	C	H	H	L	H	H	L	H	L	H	L	H	H	H	H	L	L	L	L	L	L	LOAD Q5 WITH DIN
L	C	H	H	L	H	H	L	H	H	L	L	H	H	H	L	L	L	L	L	L	L	LOAD Q6 WITH DIN
L	C	H	H	L	H	H	L	H	H	H	L	H	H	L	L	L	L	L	L	L	L	LOAD Q7 WITH DIN
L	C	H	H	L	H	H	H	L	L	L	L	H	L	L	L	L	L	L	L	L	L	LOAD Q8 WITH DIN
L	C	H	H	L	H	H	H	L	L	H	L	L	L	L	L	L	L	L	L	L	L	LOAD Q9 WITH DIN
L	C	H	H	L	H	H	H	L	L	H	H	H	L	L	L	L	L	L	L	L	L	LOAD Q9 WITH DIN
L	C	H	H	L	H	H	H	L	L	L	H	H	H	L	L	L	L	L	L	L	L	LOAD Q8 WITH DIN
L	C	H	H	L	H	H	L	H	H	H	H	H	H	H	L	L	L	L	L	L	L	LOAD Q7 WITH DIN
L	C	H	H	L	H	H	L	H	H	L	H	H	H	H	L	L	L	L	L	L	L	LOAD Q6 WITH DIN
L	C	H	H	L	H	H	L	H	L	H	H	H	H	H	H	L	L	L	L	L	L	LOAD Q5 WITH DIN
L	C	H	H	L	H	H	L	H	L	L	H	H	H	H	H	H	L	L	L	L	L	LOAD Q4 WITH DIN
L	C	H	H	L	H	H	L	L	H	H	H	H	H	H	H	H	H	H	L	L	L	LOAD Q3 WITH DIN
L	C	H	H	L	H	H	L	L	H	L	H	H	H	H	H	H	H	H	H	L	L	LOAD Q2 WITH DIN
L	C	H	H	L	H	H	L	L	L	H	H	H	H	H	H	H	H	H	H	H	L	LOAD Q1 WITH DIN
L	C	H	H	L	H	H	L	L	L	L	H	H	H	H	H	H	H	H	H	H	H	LOAD Q0 WITH DIN
L	C	H	H	L	H	L	X	X	X	X	X	H	H	H	H	H	H	H	H	H	H	HOLD STATE
L	C	H	H	L	H	H	L	L	L	L	L	H	H	H	H	H	H	H	H	H	L	LOAD Q0 WITH DIN
L	C	H	H	L	L	H	X	X	X	X	X	H	H	H	H	H	H	H	H	H	L	HOLD STATE
L	C	H	H	L	H	H	L	L	H	L	L	H	H	H	H	H	H	H	H	L	L	LOAD Q2 WITH DIN
L	C	H	H	L	H	L	X	X	X	X	X	H	H	H	H	H	H	H	H	L	H	HOLD STATE
L	C	H	H	L	H	H	L	H	L	L	L	H	H	H	H	H	L	H	L	H	L	LOAD Q4 WITH DIN
L	C	H	H	H	L	L	X	X	X	X	X	H	H	H	H	L	H	L	H	L	H	HOLD STATE
L	C	H	H	L	H	H	L	H	H	L	L	H	H	H	L	H	L	H	L	H	L	LOAD Q6 WITH DIN
L	C	H	H	H	L	H	X	X	X	X	X	H	H	H	L	H	L	H	L	H	L	HOLD STATE
L	C	H	H	L	L	H	H	L	L	L	L	H	L	H	L	H	L	H	L	H	L	LOAD Q8 WITH DIN
L	C	H	H	H	H	H	X	X	X	X	X	H	L	H	L	H	L	H	L	H	L	HOLD STATE
H	X	X	X	X	X	X	X	X	X	X	X	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	TEST HI-Z

Interface Controller for 68000 to Zilog 8500 Family

PAL 20X10

PMSI408

Features/Benefits

- **Flexible programmable control logic replacement for TTL MSI**
- **24-pin SKINNYDIP™ saves space**
- **Low current PNP inputs reduce loading**

Description

The interface controller is used to interface an 8 MHz microprocessor (68000) to Zilogs 4 MHz Z8500 peripherals with the capability of controlling read, write, and interrupt.

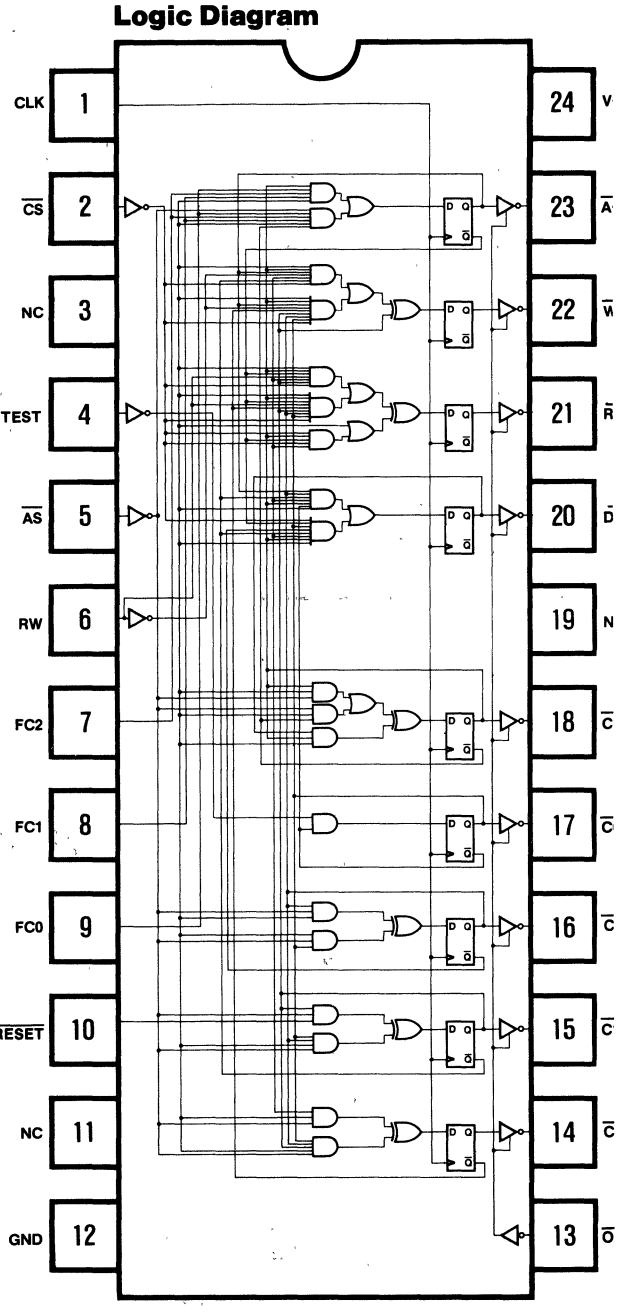
Timing for processing of data transfer is controlled with an internal down counter. A count decoder controls timing of $\overline{DTK}$ signal (Data Transfer Acknowledge) to the microprocessor so that valid operations are allowed between the microprocessor and the peripheral.

During a read cycle, the microprocessor reads data from the peripheral. During a write cycle, the microprocessor writes data to the peripheral. During an interrupt cycle, the microprocessor reads interrupt vectors from the peripheral. All cycles synchronous with the clock (CLK).

Function Table

$\overline{OE}$	CLK	$\overline{R}$	$\overline{AS}$	$\overline{CS}$	$\overline{RW}$	FC2	FC1	FC0	$\overline{WR}$	$\overline{RD}$	$\overline{DTK}$	$\overline{CYC}$	$\overline{ACK}$	OPERATION
H	X	X	X	X	X	X	X	X	Z	Z	Z	Z	Z	HI-Z
L	↑	L	X	X	X	X	X	X	L	L	H	H	H	INITIALIZATION
L	↑	H	L	L	H	L	L	L	H	L	H	L	H	READ
L	↑	H	L	L	L	L	L	L	L	H	H	L	H	WRITE
L	↑	H	L	H	H	H	H	H	H	L	H	L	L	READ DURING INTERRUPT

Logic Diagram



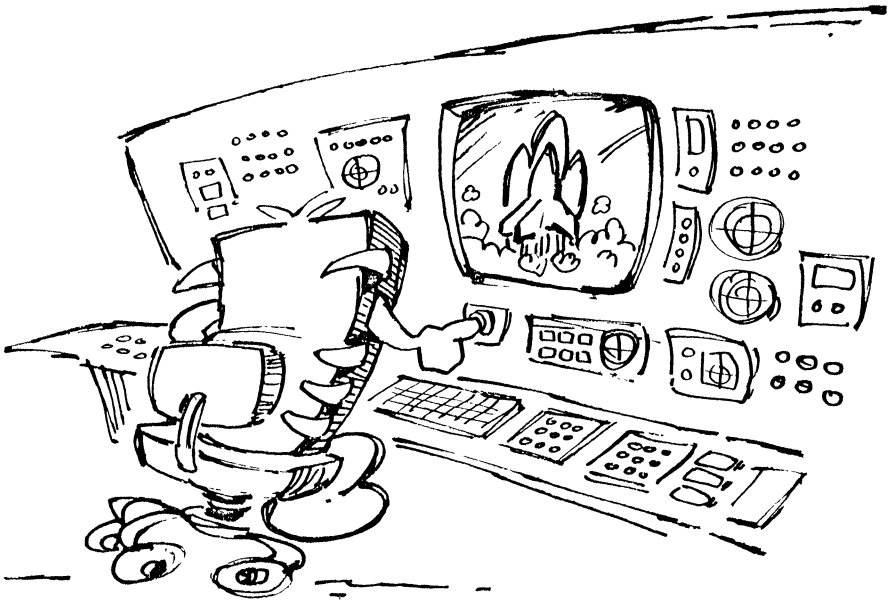
```
PAL20X10                                                PAL DESIGN SPECIFICATION
PMSI408                                          JANE LEE(MMI)/KEN THOMAS(ZILOG)
INTERFACE CONTROLLER FOR 68000 uP TO ZILOG 8500 FAMILY
MMI SUNNYVALE, CALIFORNIA
CLK /CS  NC TEST /AS  RW  FC2  FC1  FC0 /RESET  NC  GND
/OC /C3 /C2 /C1  /C0 /CYC NC  /DTK /RD  /WR    /ACK VCC


C0  :=  /C0*/TEST                                       ;COUNT/HOLD (LSB)

C1  :=  /RESET* AS* C1                                  ;HOLD
    :+: /RESET* AS* C0                                  ;DECREMENT

C2  :=  /RESET* AS* C2                                  ;HOLD
    :+: /RESET* AS* C0* C1                              ;DECREMENT

C3  :=  /RESET* AS* C3                                  ;HOLD
    :+: /RESET* AS* C0* C1* C2                          ;DECREMENT

DTK :=  /RESET*/ACK* CYC* C3*/C2*/C1* C0* CS            ;DTACK FOR RD/WR CYCLE
     +  /RESET* ACK* CYC* C3*/C2* C1*/C0                ;DTACK FOR INTERRUPT
                                                        ;CYCLE

CYC :=  /RESET* AS*/CYC* C0                             ;START NEW CYCLE
     +  /RESET* AS* CYC                                 ;HOLD PROCESS OF CYCLE
    :+: /RESET* CYC* DTK                                ;END OF CYCLE

RD  :=  /RESET* CYC*/ACK* RW* C3*/C2* CS                ;NORMAL READ OPERATION
     +  /RESET* CYC*/ACK* RW*/C3* C2* C1* C0* CS        ;NORMAL READ OPERATION
    :+: /RESET* CYC* ACK* RW* C3                        ;READ DURING INTERRUPT
     +   RESET                                          ;RESET

WR  :=  /RESET* CYC*/ACK*/RW* C3*/C2* CS                ;WRITE
     +  /RESET* CYC*/ACK*/RW*/C3* C2* C1* C0* CS        ;WRITE
    :+:  RESET                                          ;RESET

ACK :=  /RESET* FC0* FC1* FC2* AS* CYC*/C0              ;START OF INTERRUPT ACK
     +  /RESET* FC0* FC1* FC2* CYC                      ;HOLDS INTERRUPT
```

FUNCTION TABLE

CLK /OC /RESET /C3 /C2 /C1 /C0 TEST /CS FC2 FC1 FC0 RW /AS /CYC /RD /WR /DTK /ACK

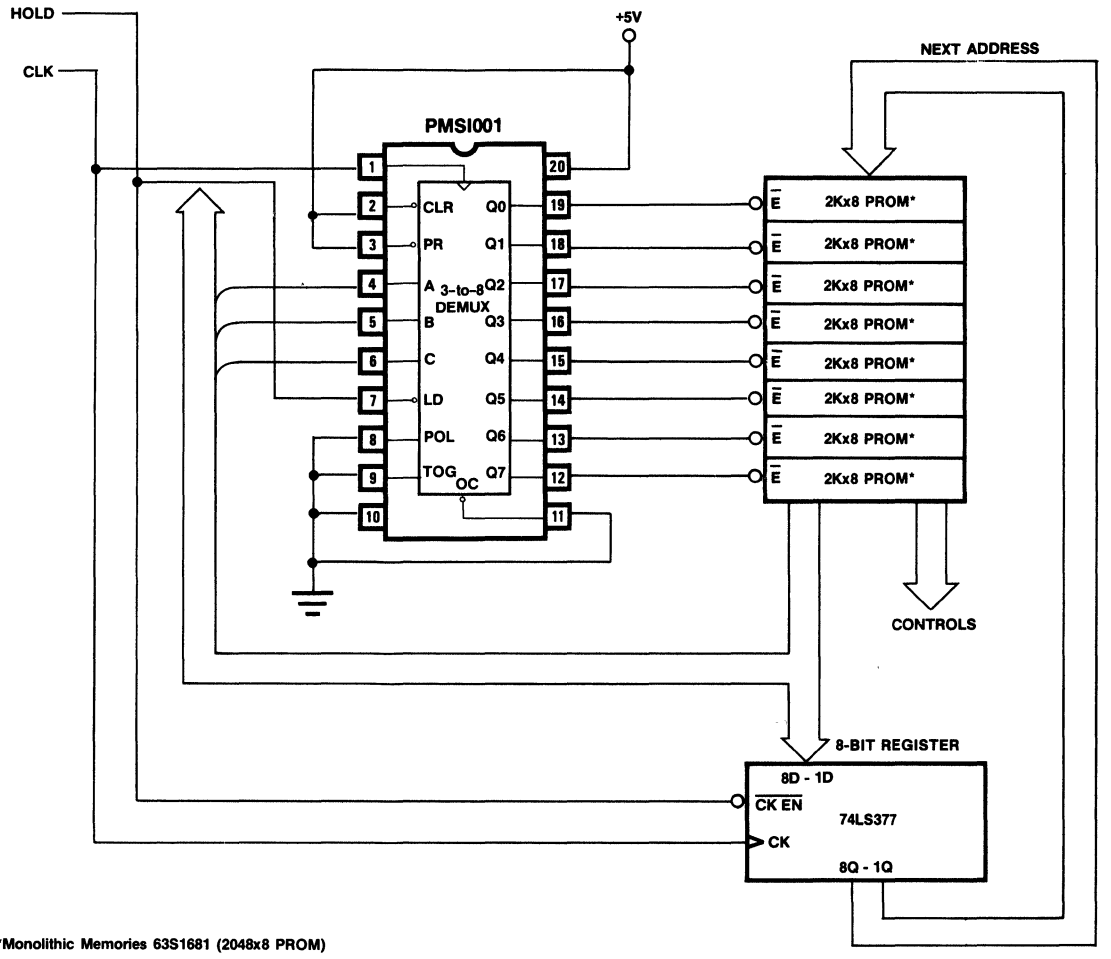
;K	ER	3210	SC	CCC	R/W	A/S	C//CDR	DA/TC/KK	CYCLE	COMMENTS
C	LL	HHHH	HH	LLL	H	H	HLL	HH	1	INITIALIZATION
C	LH	HHHL	LH	LLL	H	H	HHH	HH	2	
C	LH	HHHH	LH	LLL	H	H	HHH	HH	3	
C	LH	HHHL	LL	LLL	H	H	HHH	HH	4	
C	LH	HHLH	LL	LLL	H	L	LHH	HH	5	BEGINNING OF WAIT FOR READ CYCLE
C	LH	HHLL	LL	LLL	H	L	LHH	HH	6	
C	LH	HLHH	LL	LLL	H	L	LHH	HH	7	
C	LH	HLHL	LL	LLL	H	L	LHH	HH	8	
C	LH	HLLH	LL	LLL	H	L	LHH	HH	9	
C	LH	HLLL	LL	LLL	H	L	LHH	HH	10	
C	LH	LHHH	LL	LLL	H	L	LLH	HH	11	/RD GOES ACTIVE
C	LH	LHHL	LL	LLL	H	L	LLH	HH	12	
C	LH	LHLH	LL	LLL	H	L	LLH	LH	13	ACKNOWLEDGE END OF DATA TRANSFER
C	LH	LHLL	LL	LLL	H	L	HLH	HH	14	END OF READ CYCLE
C	LH	HHHH	LH	LLL	H	H	HHH	HH	15	INITIALIZATION
C	LH	HHHL	LH	LLL	H	H	HHH	HH	16	
C	LH	HHLH	LL	LLL	L	L	LHH	HH	17	BEGINNING OF WAIT FOR WRITE CYCLE
C	LH	HHLL	LL	LLL	L	L	LHH	HH	18	
C	LH	HLHH	LL	LLL	L	L	LHH	HH	19	
C	LH	HLHL	LL	LLL	L	L	LHH	HH	20	
C	LH	HLLH	LL	LLL	L	L	LHH	HH	21	
C	LH	HLLL	LL	LLL	L	L	LHH	HH	22	
C	LH	LHHH	LL	LLL	L	L	LHL	HH	23	/WR GOES ACTIVE
C	LH	LHHL	LL	LLL	L	L	LHL	HH	24	
C	LH	LHLH	LL	LLL	L	L	LHL	LH	25	ACKNOWLEDGE END OF DATA TRANSFER
C	LH	LHLL	LL	LLL	L	L	HHL	HH	26	END OF WRITE CYLE
C	LH	HHHH	LH	LLL	H	H	HHH	HH	27	INITIALIZATION
C	LH	HHHL	LH	LLL	H	H	HHH	HH	28	
C	LH	HHLH	LH	HHH	H	L	LHH	HH	29	INTERRUPT CYCLE (WAITS RD SIGNAL)
C	LH	HHLL	LH	HHH	H	L	LHH	HL	30	
C	LH	HLHH	LH	HHH	H	L	LHH	HL	31	
C	LH	HLHL	LH	HHH	H	L	LHH	HL	32	
C	LH	HLLH	LH	HHH	H	L	LHH	HL	33	
C	LH	HLLL	LH	HHH	H	L	LHH	HL	34	
C	LH	LHHH	LH	HHH	H	L	LHH	HL	35	
C	LH	LHHL	LH	HHH	H	L	LLH	HL	36	/RD GOES ACTIVE
C	LH	LHLH	LH	HHH	H	L	LLH	HL	37	
C	LH	LHLL	LH	HHH	H	L	LLH	LL	38	
C	LH	LLHH	LH	HHH	H	L	HLH	HL	39	
C	LH	HHHL	LH	LLL	H	H	HHH	HH	40	END OF CYCLE

PMSI APPLICATIONS



Application

Pipeline Demux for Microprogram State Machine

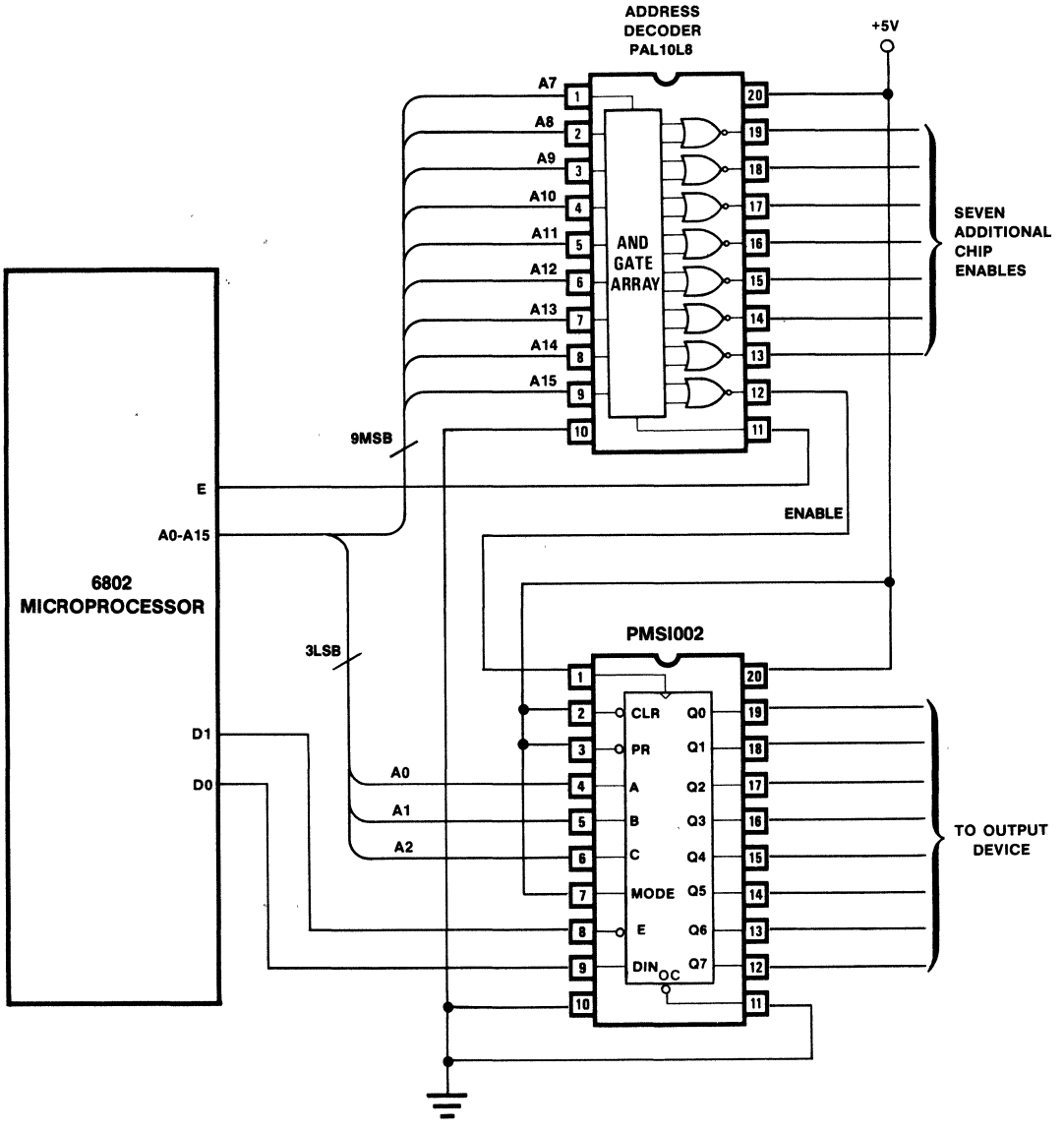


*Monolithic Memories 63S1681 (2048x8 PROM)

5

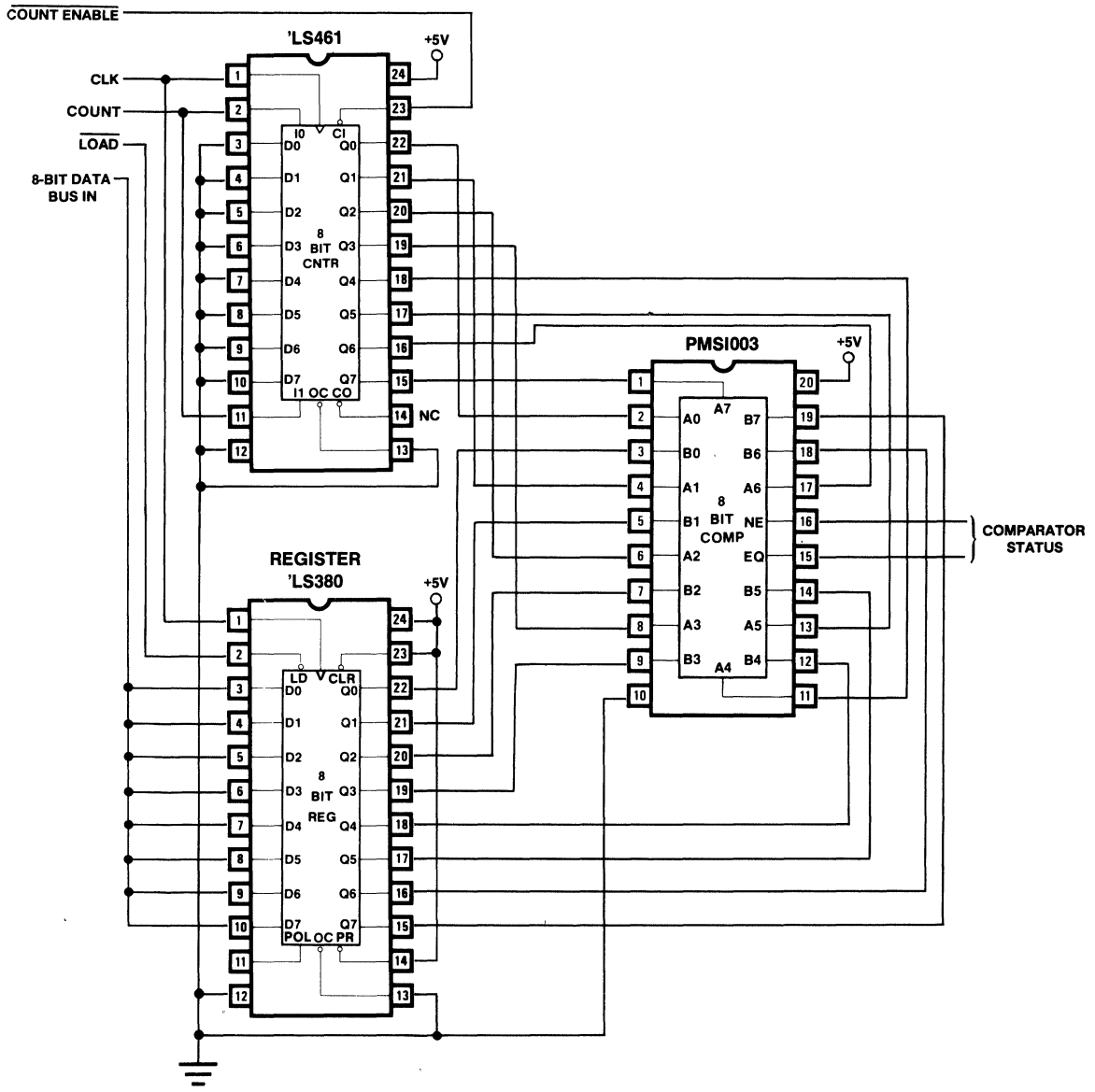
Application

Octal Addressable Register as a PIA



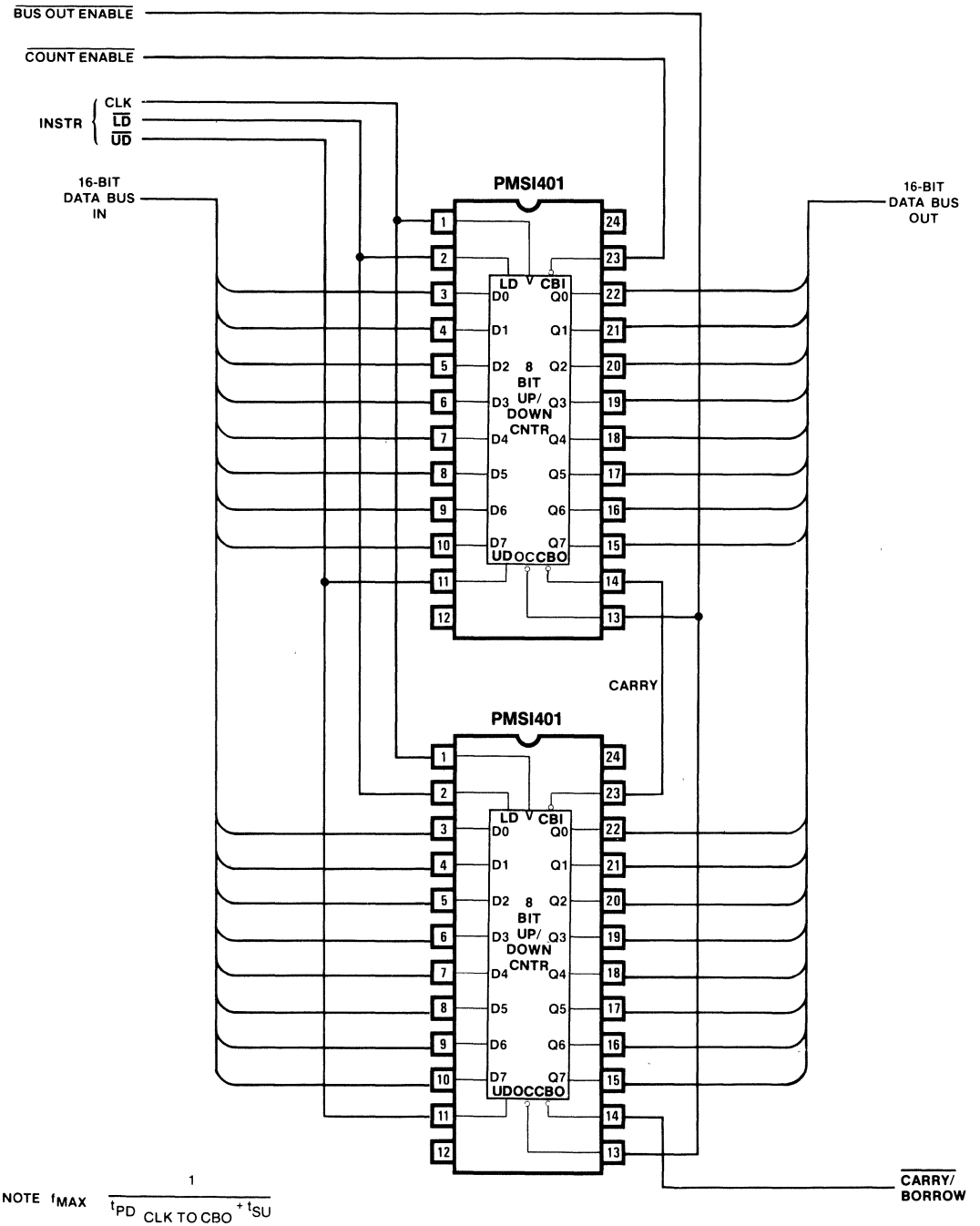
Application

Octal Counter/Register Comparator



Application

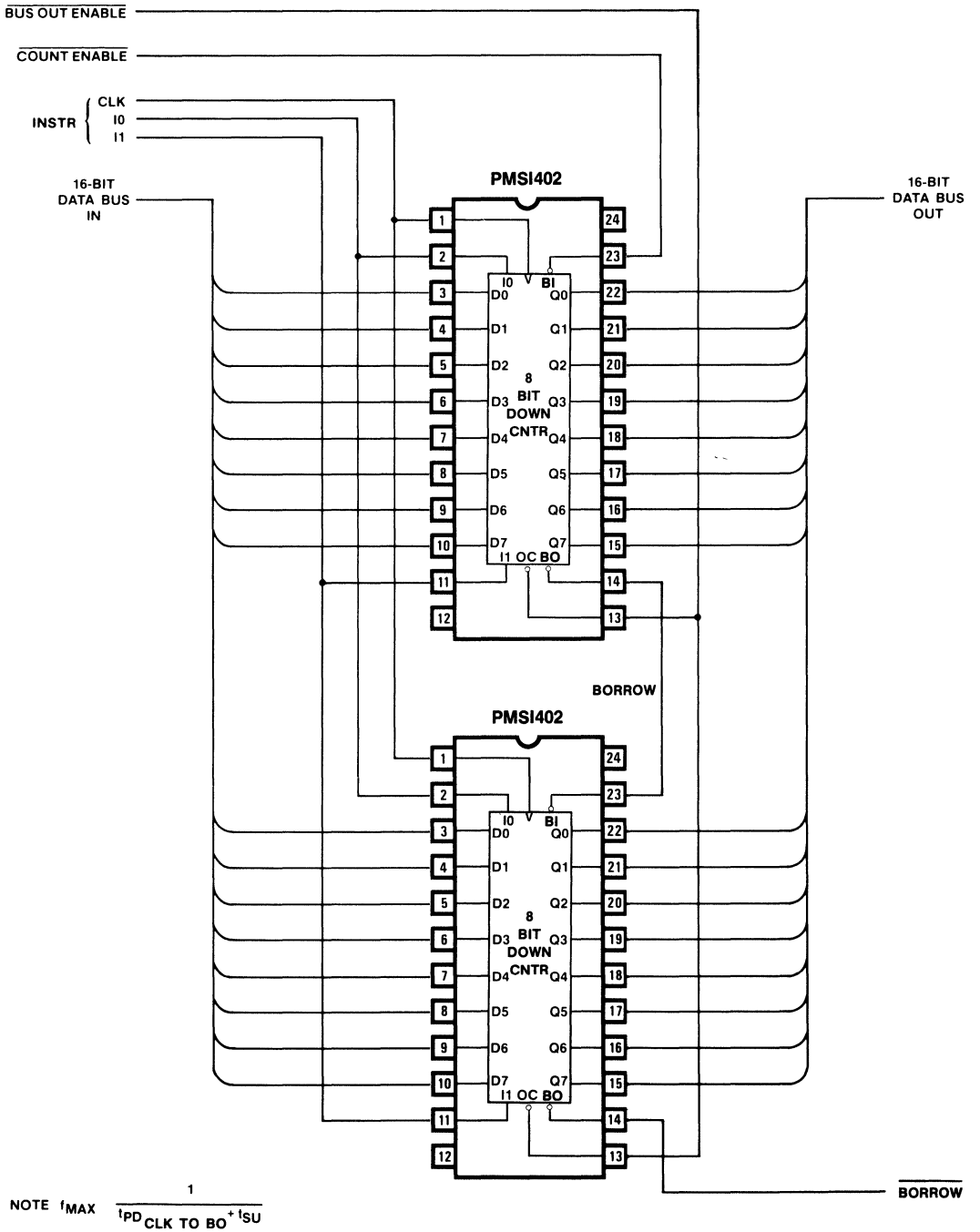
16-Bit Up/Down Counter



NOTE $f_{MAX} \frac{1}{t_{PD\ CLK\ TO\ CBO} + t_{SU}}$

Application

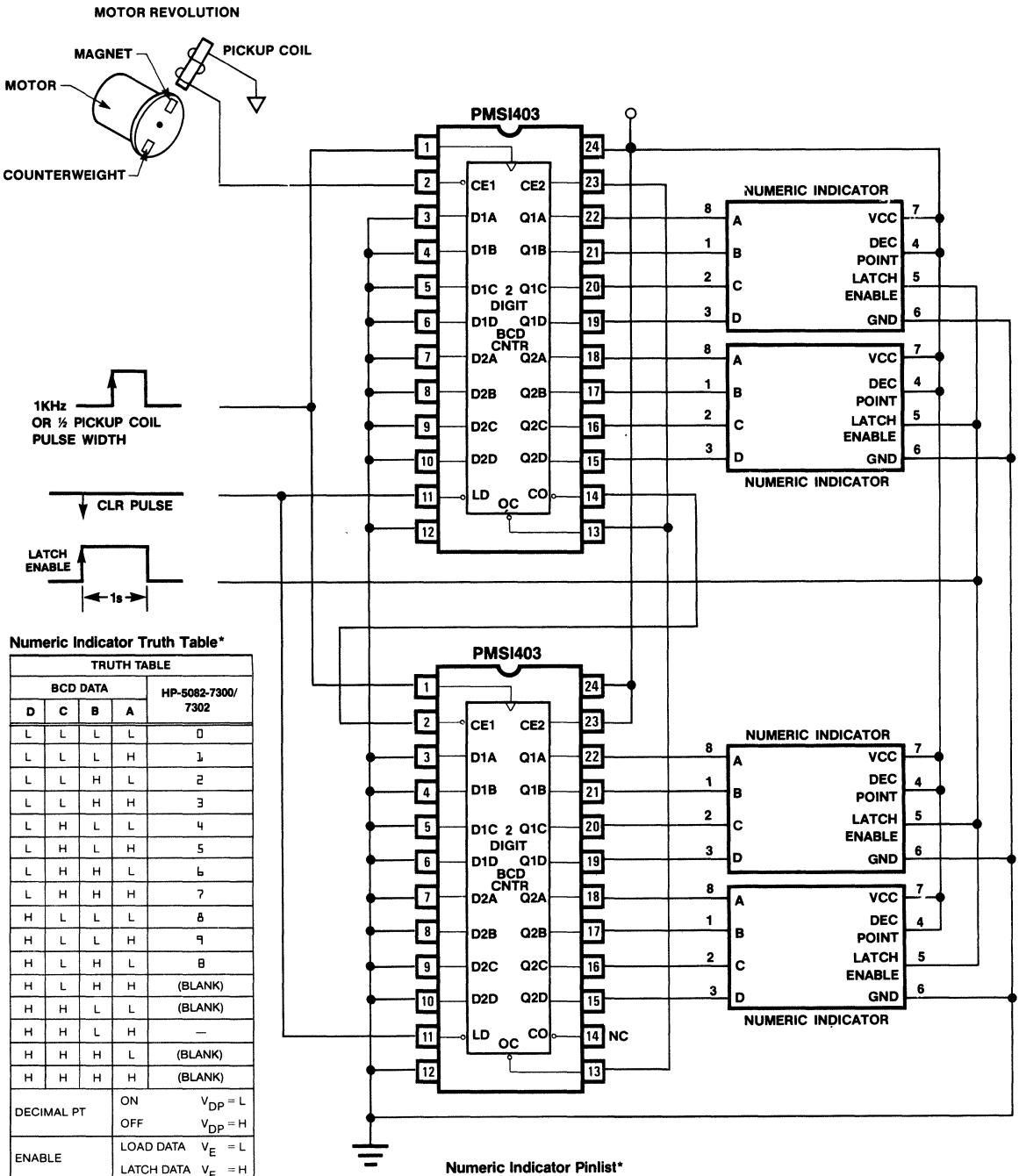
16-Bit Down Counter



NOTE $f_{MAX} \frac{1}{t_{PD_{CLK\ TO\ BO}} + t_{SU}}$

Application

Motor Revolution Per Second Indicator



Numeric Indicator Truth Table*

TRUTH TABLE				
BCD DATA				HP-5082-7300/ 7302
D	C	B	A	
L	L	L	L	0
L	L	L	H	1
L	L	H	L	2
L	L	H	H	3
L	H	L	L	4
L	H	L	H	5
L	H	H	L	6
L	H	H	H	7
H	L	L	L	8
H	L	L	H	9
H	L	H	L	8
H	L	H	H	(BLANK)
H	H	L	L	(BLANK)
H	H	L	H	—
H	H	H	L	(BLANK)
H	H	H	H	(BLANK)
DECIMAL PT			ON	V_{DP} = L
			OFF	V_{DP} = H
ENABLE			LOAD DATA	V_E = L
			LATCH DATA	V_E = H
BLANKING			DISPLAY-ON	V_B = L
			DISPLAY-OFF	V_B = H

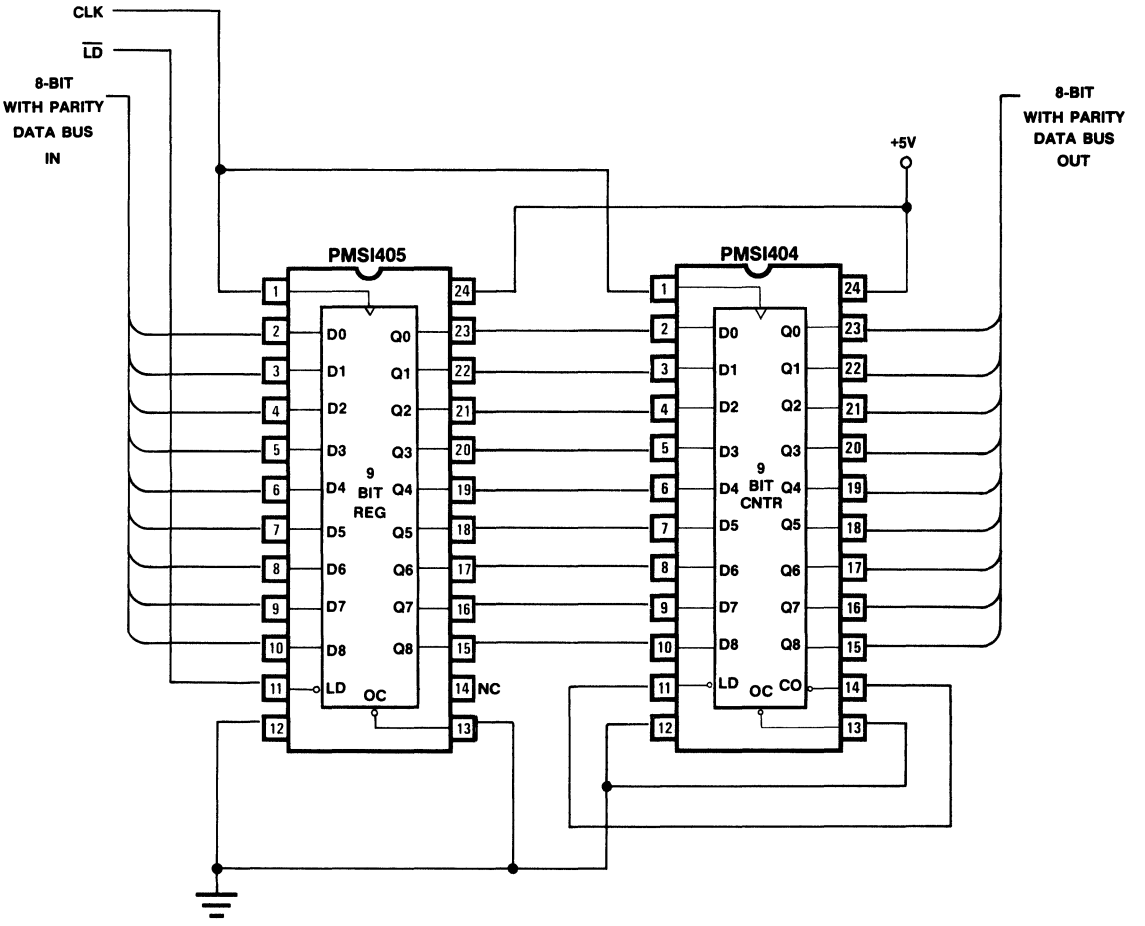
Numeric Indicator Pinlist*

PIN #	1	2	3	4	5	6	7	8
FUNCTION	B	C	D	Decimal Point	Latch Enable	GND	VCC	A

*Consult the Hewlett-Packard "Optoelectronics Designer's Catalog" for more details.

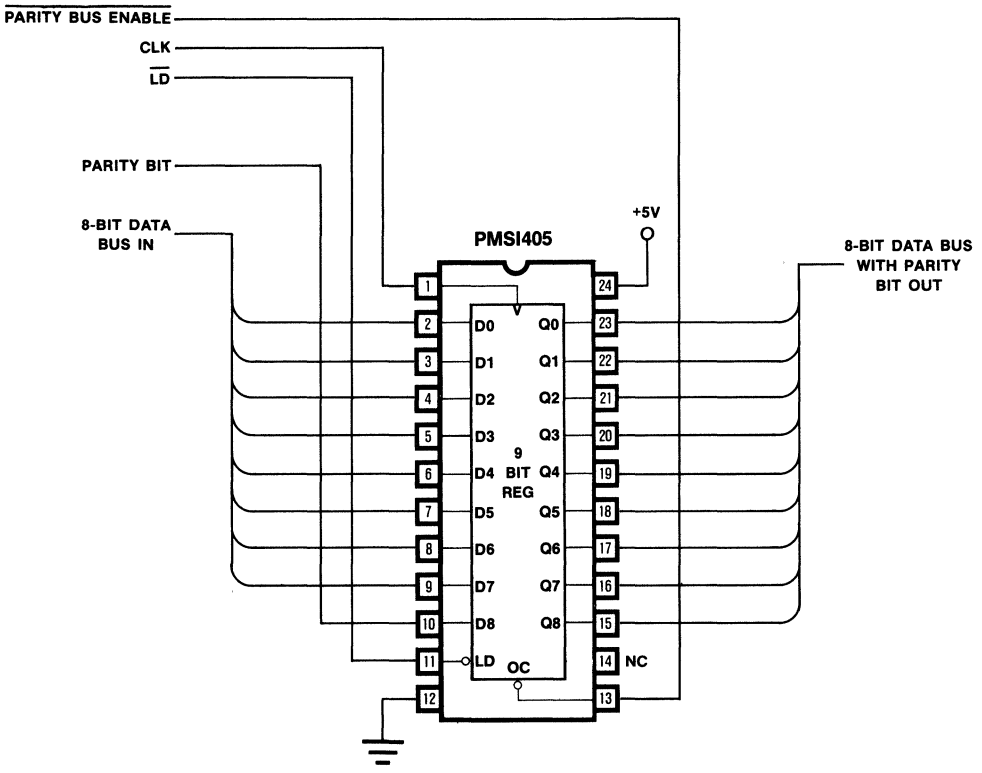
Application

9-Bit Counter with Registered Load



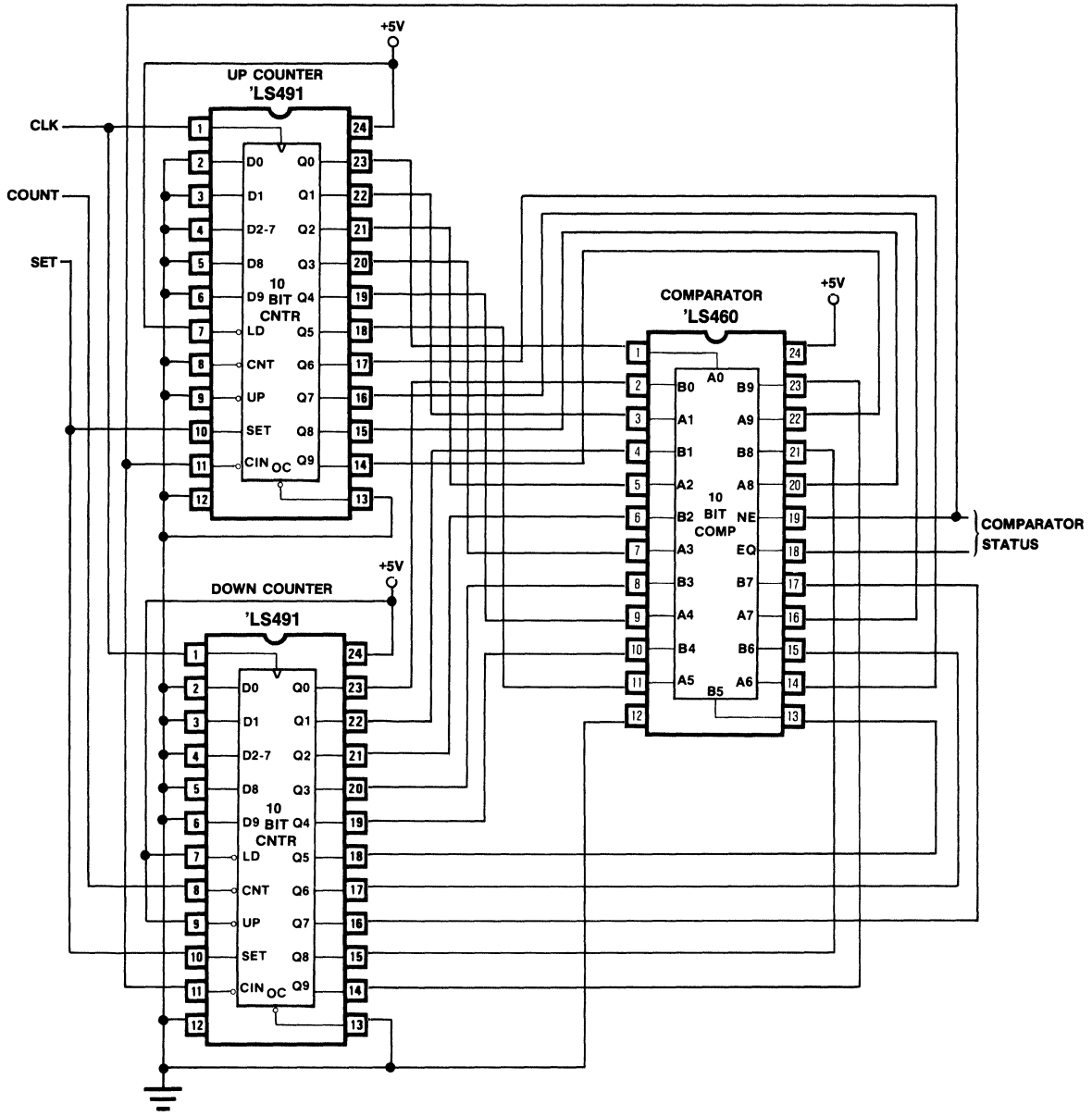
Application

Parity Bus Interface



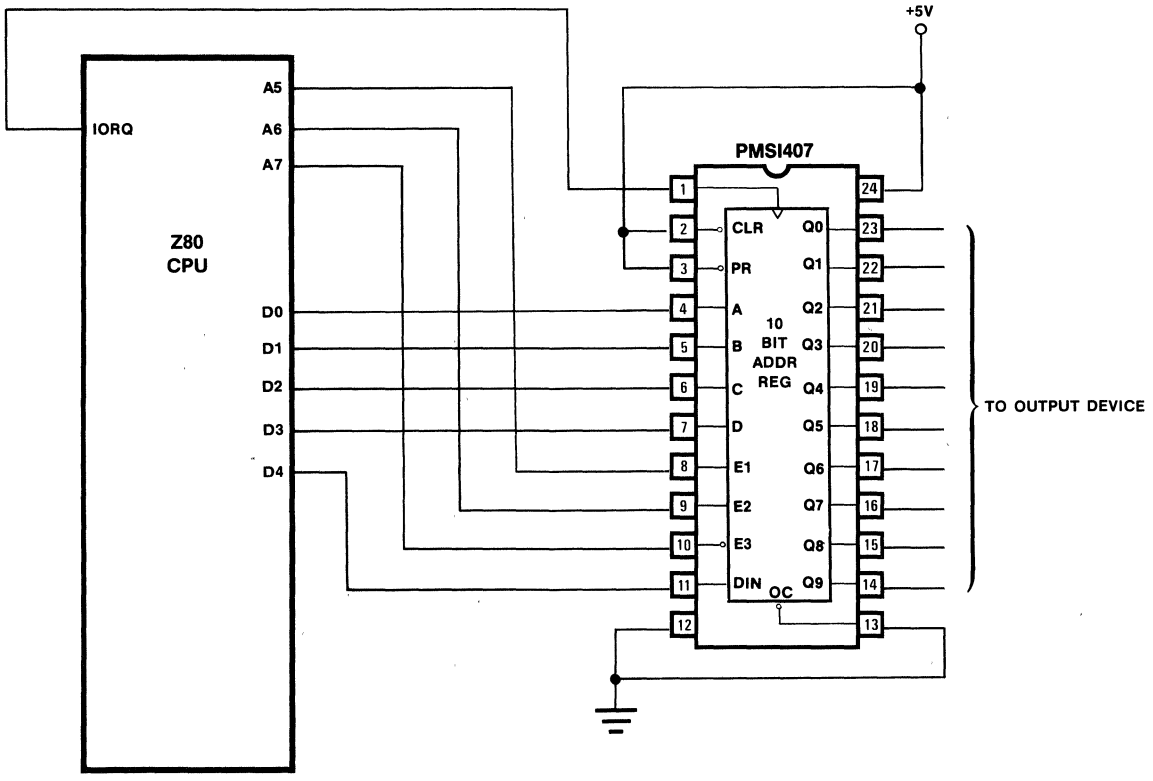
Application

10-Bit Up Counter/Down Counter Comparator



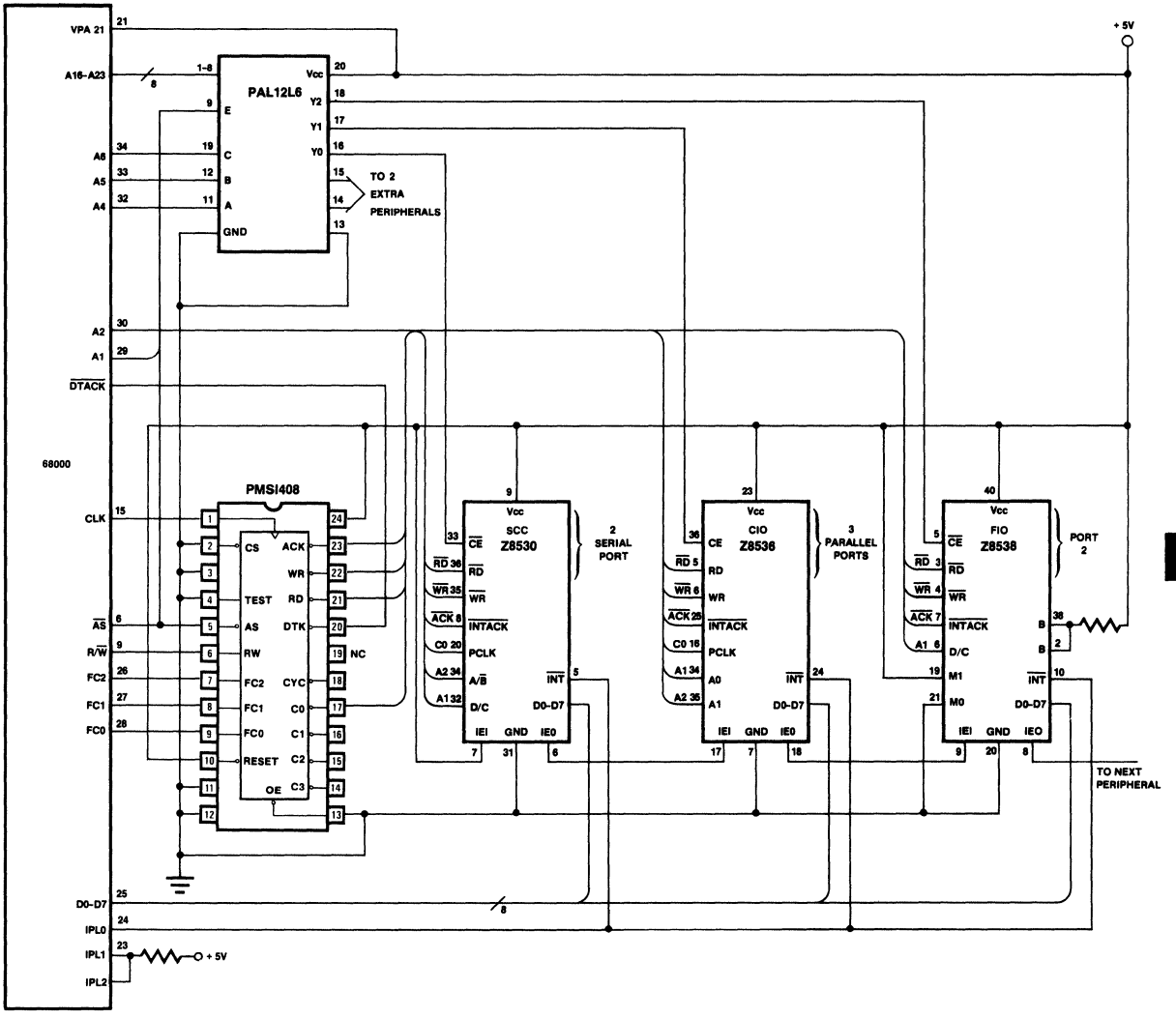
Application

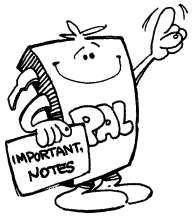
10-Bit Addressable Register



Application

Zilog 8500 Interface with the 68000 Microprocessor




IMPORTANT
NOTES

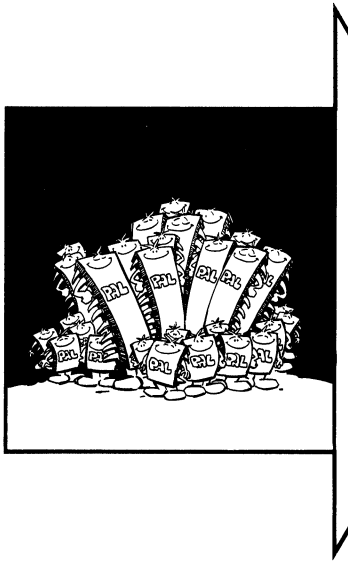


Table of Contents

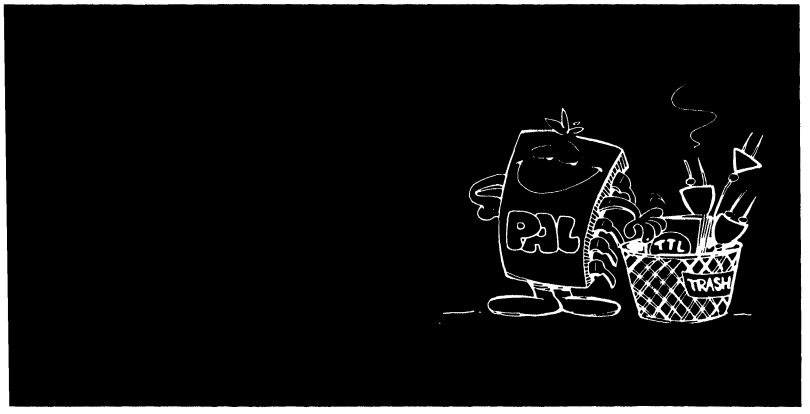
Introduction to PAL Applications

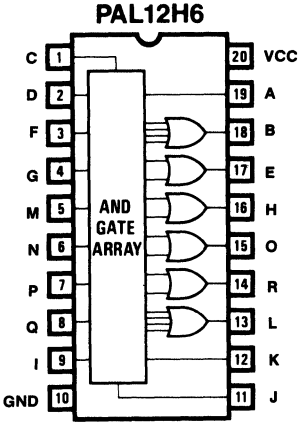
The PAL family brings a unique flexibility to the field of logic design. Using PAL circuits, designers can both replace conventional logic in existing products and optimize the design of new products. Previous sections discussed the PAL concept and provided information ont he advantages gained and the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

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.

Basic Gates





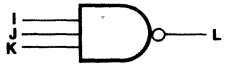
6

```
PAL12H6                                    PAL DESIGN SPECIFICATION
P7000                                        VINCENT COLI 03/12/82
BASIC GATES
MMI SUNNYVALE, CALIFORNIA
C D F G M N P Q I GND J K L R O H E B A VCC

B = /A              ;INVERTER
```


```
E = C*D             ;AND GATE
```

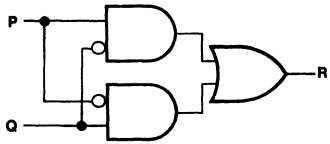

```
H = F + G           ;OR GATE
```


```
L = /I + /J + /K    ;NAND GATE
```

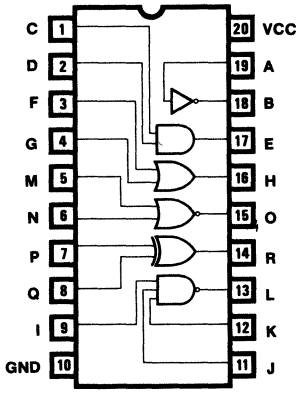



```
O = /M*/N           ;NOR GATE
```


```
R = P*/Q + /P*Q     ;EXCLUSIVE OR GATE
```



PAL12H6



FUNCTION TABLE

A B C D E F G H I J K L M N O P Q R

;AB	CDE	FGH	IJKL	MNO	PQR	COMMENTS
LH	XXX	XXX	XXXX	XXX	XXX	TEST INVERTER
HL	XXX	XXX	XXXX	XXX	XXX	TEST INVERTER
XX	LLL	XXX	XXXX	XXX	XXX	TEST AND GATE
XX	LHL	XXX	XXXX	XXX	XXX	TEST AND GATE
XX	HLL	XXX	XXXX	XXX	XXX	TEST AND GATE
XX	HHH	XXX	XXXX	XXX	XXX	TEST AND GATE
XX	XXX	LLL	XXXX	XXX	XXX	TEST OR GATE
XX	XXX	LHH	XXXX	XXX	XXX	TEST OR GATE
XX	XXX	HLH	XXXX	XXX	XXX	TEST OR GATE
XX	XXX	HHH	XXXX	XXX	XXX	TEST OR GATE
XX	XXX	XXX	LLLH	XXX	XXX	TEST NAND GATE
XX	XXX	XXX	HHHL	XXX	XXX	TEST NAND GATE
XX	XXX	XXX	LHHH	XXX	XXX	TEST NAND GATE
XX	XXX	XXX	HLHH	XXX	XXX	TEST NAND GATE
XX	XXX	XXX	HHLH	XXX	XXX	TEST NAND GATE
XX	XXX	XXX	XXXX	LLH	XXX	TEST NOR GATE
XX	XXX	XXX	XXXX	LHL	XXX	TEST NOR GATE
XX	XXX	XXX	XXXX	HLL	XXX	TEST NOR GATE
XX	XXX	XXX	XXXX	HHL	XXX	TEST NOR GATE
XX	XXX	XXX	XXXX	XXX	LLL	TEST EXCLUSIVE OR GATE
XX	XXX	XXX	XXXX	XXX	LHH	TEST EXCLUSIVE OR GATE
XX	XXX	XXX	XXXX	XXX	HLH	TEST EXCLUSIVE OR GATE
XX	XXX	XXX	XXXX	XXX	HHL	TEST EXCLUSIVE OR GATE

DESCRIPTION

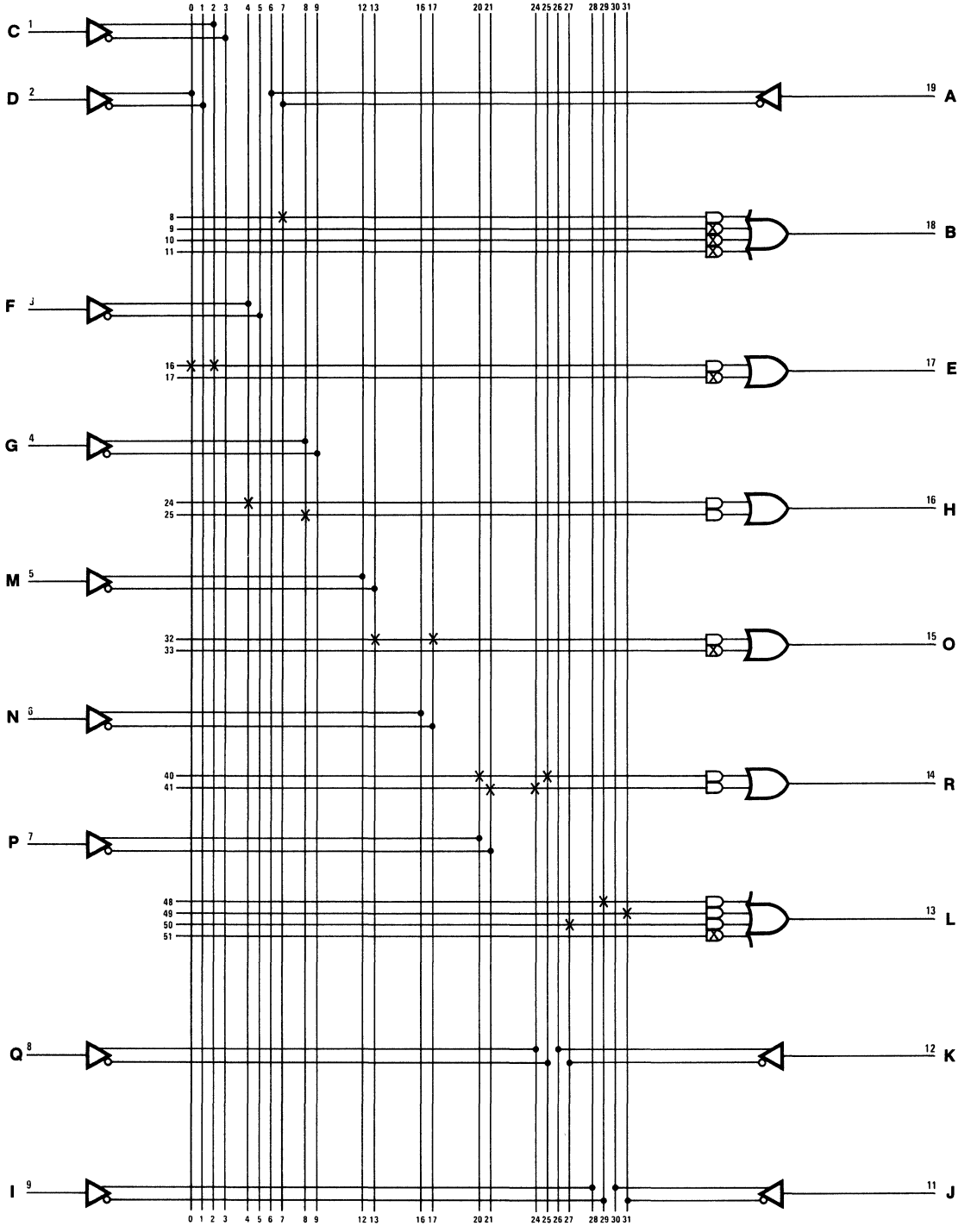
THIS EXAMPLE ILLUSTRATES THE USE OF FUSIBLE LOGIC TO IMPLEMENT THE BASIC GATES; INVERTER, AND GATE, OR GATE, NAND GATE, NOR GATE, AND EXCLUSIVE OR GATE.

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

```
BASIC GATES

   1 XXXXXXXXXXXXXXXXXH01
   2 XXXXXXXXXXXXXXXXXL11
   3 00XXXXXXXXXXXXXXLXX1
   4 01XXXXXXXXXXXXXXLXX1
   5 10XXXXXXXXXXXXXXLXX1
   6 11XXXXXXXXXXXXXXHXX1
   7 XX00XXXXXXXXXXXLXXX1
   8 XX01XXXXXXXXXXXHXXX1
   9 XX10XXXXXXXXXXXHXXX1
  10 XX11XXXXXXXXXXXHXXX1
  11 XXXXXXXX0X00HXXXXXX1
  12 XXXXXXXX1X11LXXXXXX1
  13 XXXXXXXX0X11HXXXXXX1
  14 XXXXXXXX1X01HXXXXXX1
  15 XXXXXXXX1X10HXXXXXX1
  16 XXXX00XXXXXXXXHXXXX1
  17 XXXX01XXXXXXXXLXXXX1
  18 XXXX10XXXXXXXXLXXXX1
  19 XXXX11XXXXXXXXLXXXX1
  20 XXXXXX00XXXXXLXXXXX1
  21 XXXXXX01XXXXXHXXXXX1
  22 XXXXXX10XXXXXHXXXXX1
  23 XXXXXX11XXXXXLXXXXX1

PASS SIMULATION

BASIC GATES

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


 8 ---- ---X --   --   --   --   ---- ---- /A

16 X-X- ---- --   --   --   --   ---- ---- C*D

24 ---- X--- --   --   --   --   ---- ---- F
25 ---- ---- X-   --   --   --   ---- ---- G

32 ---- ---- --   -X   -X   --   ---- ---- /M*/N

40 ---- ---- --   --   --   X-   -X-- ---- P*/Q
41 ---- ---- --   --   --   -X   X--- ---- /P*Q

48 ---- ---- --   --   --   --   ---- -X-- /I
49 ---- ---- --   --   --   --   ---- ---X /J
50 ---- ---- --   --   --   --   ---X ---- /K


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

NUMBER OF FUSES BLOWN =  306
```

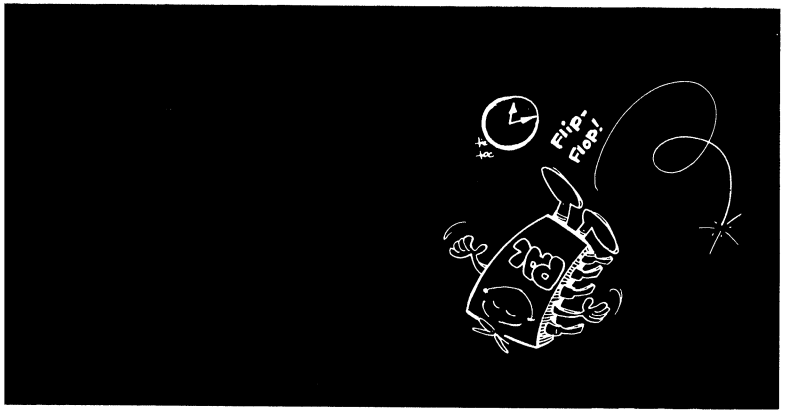
Basic Gates

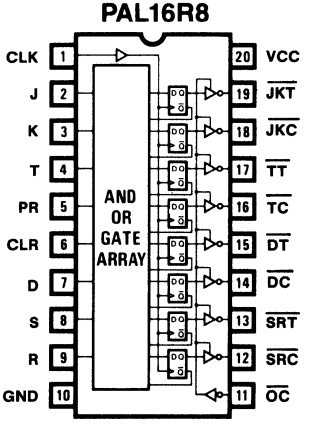
Logic Diagram PAL12H6




IMPORTANT
NOTES

Basic Clocked Flip-Flops





```
PAL16R8                                          PAL DESIGN SPECIFICATION
BFLIP                                              VINCENT COLI 07/19/81
BASIC CLOCKED FLIP-FLOPS
MMI SUNNYVALE, CALIFORNIA
CLK J K T PR CLR D S R GND
/OC /SRC /SRT /DC /DT /TC /TT /JKC /JKT VCC


JKT :=  J*/JKT*/CLR          ;J-K FLIP-FLOP (TRUE)
     + /K* JKT*/CLR          ; (JKT = Q)
     +  PR                   ;PRESET Q

JKC := /J* K  */PR           ;J-K FLIP-FLOP (COMPLEMENT)
     + /J*/JKT*/PR           ; (JKC = /Q)
     +  K* JKT*/PR           ;
     +  CLR                  ;CLEAR /Q

TT  :=  T*/TT*/CLR           ;T FLIP-FLOP (TRUE)
     + /T* TT*/CLR           ; (TT = Q)
     +  PR                   ;PRESET Q

TC  := /T*/TT*/PR            ;T FLIP-FLOP (COMPLEMENT)
     +  T* TT*/PR            ; (TC = /Q)
     +  CLR                  ;CLEAR /Q

DT  :=  D*/CLR               ;D FLIP-FLOP (TRUE) (DT = Q)
     +  PR                   ;PRESET Q

DC  := /D*/PR                ;D FLIP-FLOP (COMPLEMENT) (DC = /Q)
     +  CLR                  ;CLEAR /Q

SRT :=  S*     /CLR          ;SET-RESET FLIP-FLOP (TRUE)
     + /R* SRT*/CLR          ; (SRT = Q)
     +  PR                   ;PRESET Q

SRC := /S* R  */PR           ;SET-RESET FLIP-FLOP (COMPLEMENT)
     + /S*/SRT*/PR           ; (SRC = /Q)
     +  CLR                  ;CLEAR /Q
```

```
FUNCTION TABLE

CLK /OC PR CLR J K JKT JKC T TT TC D DT DC S R SRT SRC

;   CONTROL     J-K FLIP-FLOP   T FLIP-FLOP   D FLIP-FLOP   S-R FLIP-FLOP
; C  /            IN    OUT     IN    OUT     IN    OUT     IN    OUT
;LK OC PR CLR   J  K  Q  /Q    T  Q  /Q    D  Q  /Q    S  R  Q  /Q    COMMENT
-------------------------------------------------------------------------------
  X  H  X  X    X  X  Z  Z    X  Z  Z    X  Z  Z    X  X  Z  Z    HI-Z
;
;   TEST J-K FLIP-FLOP
;
  C  L  L  H    X  X  L  H    X  X  X    X  X  X    X  X  X  X    CLEAR
  C  L  L  L    L  L  L  H    X  X  X    X  X  X    X  X  X  X
  C  L  L  L    L  H  L  H    X  X  X    X  X  X    X  X  X  X
  C  L  L  L    H  H  H  L    X  X  X    X  X  X    X  X  X  X    TOGGLE
  C  L  L  L    H  L  H  L    X  X  X    X  X  X    X  X  X  X
  C  L  L  L    L  L  H  L    X  X  X    X  X  X    X  X  X  X
  C  L  L  L    L  H  L  H    X  X  X    X  X  X    X  X  X  X
  C  L  H  L    X  X  H  L    X  X  X    X  X  X    X  X  X  X    PRESET
  C  L  L  L    H  H  L  H    X  X  X    X  X  X    X  X  X  X    TOGGLE
  C  L  L  L    H  L  H  L    X  X  X    X  X  X    X  X  X  X
;
;   TEST T FLIP-FLOP
;
  C  L  L  H    X  X  X  X    X  L  H    X  X  X    X  X  X  X    CLEAR
  C  L  L  L    X  X  X  X    L  L  H    X  X  X    X  X  X  X
  C  L  L  L    X  X  X  X    H  H  L    X  X  X    X  X  X  X    TOGGLE
  C  L  L  L    X  X  X  X    H  L  H    X  X  X    X  X  X  X    TOGGLE
  C  L  H  L    X  X  X  X    X  H  L    X  X  X    X  X  X  X    PRESET
;
;   TEST D FLIP-FLOP
;
  C  L  L  H    X  X  X  X    X  X  X    X  L  H    X  X  X  X    CLEAR
  C  L  L  L    X  X  X  X    X  X  X    L  L  H    X  X  X  X
  C  L  L  L    X  X  X  X    X  X  X    H  H  L    X  X  X  X
  C  L  L  L    X  X  X  X    X  X  X    L  L  H    X  X  X  X
  C  L  H  L    X  X  X  X    X  X  X    X  H  L    X  X  X  X    PRESET
;
;   TEST S-R FLIP-FLOP
;
  C  L  L  H    X  X  X  X    X  X  X    X  X  X    X  X  L  H    CLEAR
  C  L  L  L    X  X  X  X    X  X  X    X  X  X    L  L  L  H
  C  L  L  L    X  X  X  X    X  X  X    X  X  X    H  L  H  L    SET
  C  L  L  L    X  X  X  X    X  X  X    X  X  X    L  H  L  H    RESET
  C  L  L  L    X  X  X  X    X  X  X    X  X  X    L  H  L  H    HOLD
  C  L  H  L    X  X  X  X    X  X  X    X  X  X    X  X  H  L    PRESET
  C  L  L  L    X  X  X  X    X  X  X    X  X  X    L  L  H  L
  C  L  L  L    X  X  X  X    X  X  X    X  X  X    H  L  H  L
-------------------------------------------------------------------------------
```

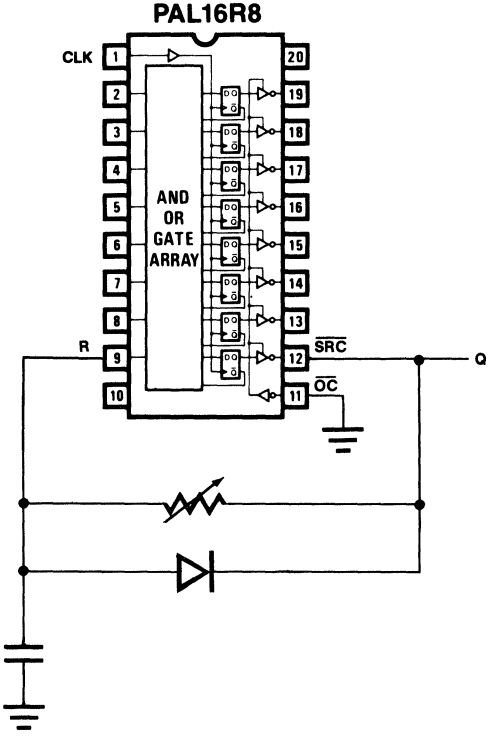
DESCRIPTION
THIS EXAMPLE ILLUSTRATES THE USE OF FUSIBLE LOGIC TO IMPLEMENT THE BASIC FLIP-FLOPS: J-K FLIP-FLOP, T FLIP-FLOP, D FLIP-FLOP, AND S-R FLIP-FLOP.

NEXT STATE TABLE FOR THE BASIC FLIP-FLOPS:

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

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

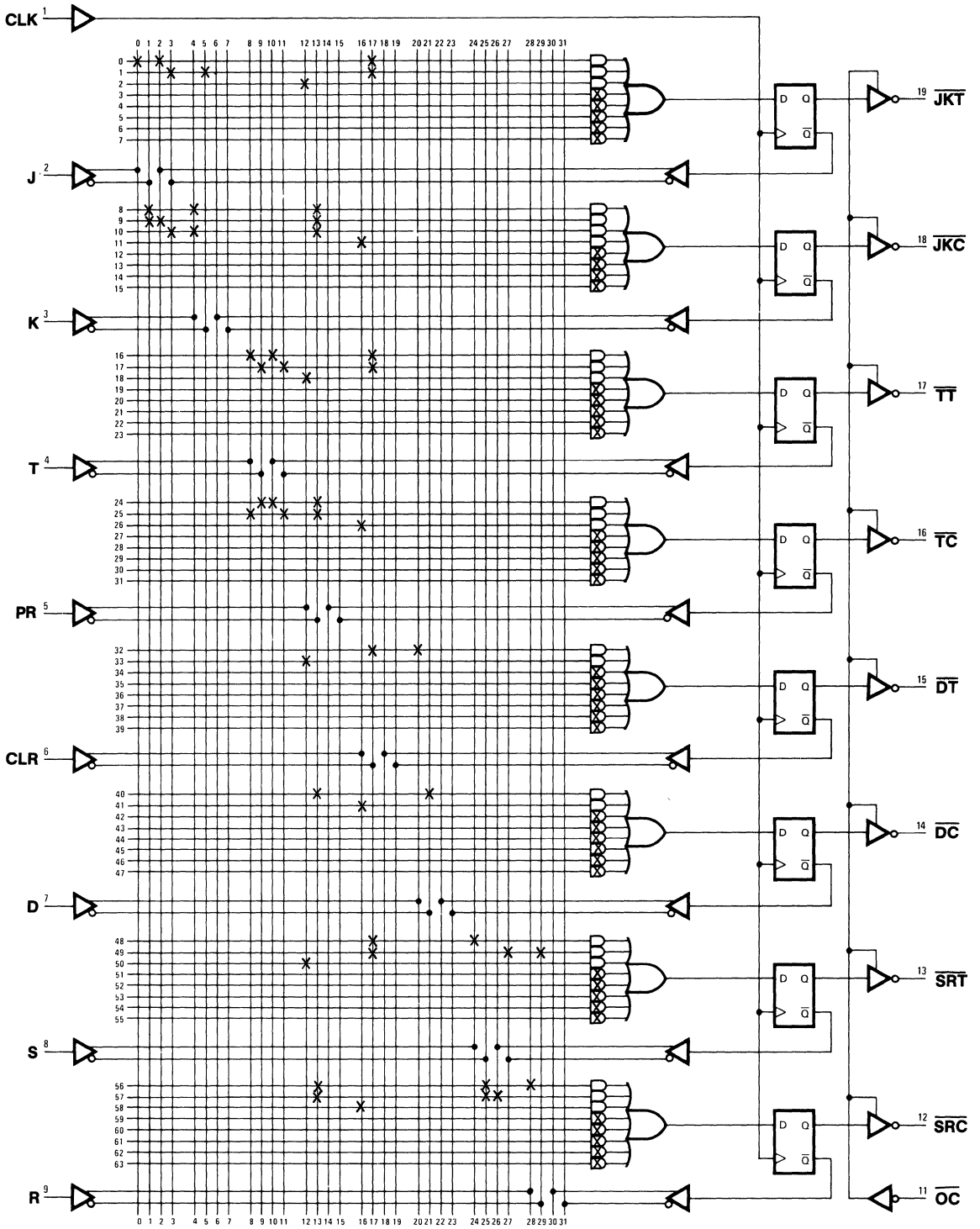
THE FUNCTION TABLE EXERCISES ALL INPUTS AND TESTS BASIC FUNCTION PERFORMANCE. PALASM EXERCISES THE FUNCTION TABLE TO SIMULATE THE BASIC CLOCKED FLIP-FLOPS.



IMPLEMENTATION OF THE S-R FLIP-FLOP AS A DIVIDE-BY-N COUNTER

Basic Clocked Flip-Flops

Logic Diagram PAL16R8

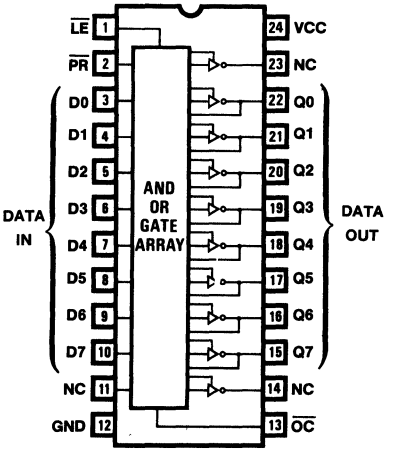



IMPORTANT
NOTES

Clean Octal Latch

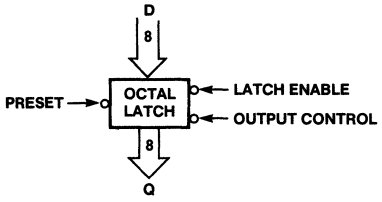
[redacted]

PAL20L10



6

```
PAL20L10                                              PAL DESIGN SPECIFICATION
P7096                                                    VINCENT COLI 10/05/82
CLEAN OCTAL LATCH
MMI SUNNYVALE, CALIFORNIA
/LE /PR D0 D1 D2 D3 D4 D5 D6 D7 NC GND
/OC  NC Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0 NC VCC


IF(OC)  /Q0 = /PR*/LE*/D0        ;LOAD LATCH (Q0)
            + /PR* LE*/Q0        ;LATCH OUTPUT
            + /PR*/D0*/Q0        ;COVER ALWAYS HIGH HAZARD

IF(OC)  /Q1 = /PR*/LE*/D1        ;LOAD LATCH (Q1)
            + /PR* LE*/Q1        ;LATCH OUTPUT
            + /PR*/D1*/Q1        ;COVER ALWAYS HIGH HAZARD

IF(OC)  /Q2 = /PR*/LE*/D2        ;LOAD LATCH (Q2)
            + /PR* LE*/Q2        ;LATCH OUTPUT
            + /PR*/D2*/Q2        ;COVER ALWAYS HIGH HAZARD

IF(OC)  /Q3 = /PR*/LE*/D3        ;LOAD LATCH (Q3)
            + /PR* LE*/Q3        ;LATCH OUTPUT
            + /PR*/D3*/Q3        ;COVER ALWAYS HIGH HAZARD

IF(OC)  /Q4 = /PR*/LE*/D4        ;LOAD LATCH (Q4)
            + /PR* LE*/Q4        ;LATCH OUTPUT
            + /PR*/D4*/Q4        ;COVER ALWAYS HIGH HAZARD

IF(OC)  /Q5 = /PR*/LE*/D5        ;LOAD LATCH (Q5)
            + /PR* LE*/Q5        ;LATCH OUTPUT
            + /PR*/D5*/Q5        ;COVER ALWAYS HIGH HAZARD

IF(OC)  /Q6 = /PR*/LE*/D6        ;LOAD LATCH (Q6)
            + /PR* LE*/Q6        ;LATCH OUTPUT
            + /PR*/D6*/Q6        ;COVER ALWAYS HIGH HAZARD

IF(OC)  /Q7 = /PR*/LE*/D7        ;LOAD LATCH (Q7)
            + /PR* LE*/Q7        ;LATCH OUTPUT
            + /PR*/D7*/Q7        ;COVER ALWAYS HIGH HAZARD
```



FUNCTION TABLE

/OC /PR /LE D7 D6 D5 D4 D3 D2 D1 D0 Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0

;CHIP CONTROLS			DDDDDDDD	QQQQQQQQ	
;/OC	/PR	/LE	76543210	76543210	COMMENTS
L	L	X	XXXXXXXX	HHHHHHHH	PRESET
L	H	H	LLLLLLLL	LLLLLLLL	LOAD ALL ZEROS
L	H	L	XXXXXXXX	LLLLLLLL	LATCH ALL ZEROS
L	H	H	HHHHHHHH	HHHHHHHH	LOAD ALL ONES
L	H	L	XXXXXXXX	HHHHHHHH	LATCH ALL ONES
L	H	H	HLHLHLHL	HLHLHLHL	LOAD EVEN CHECKERBOARD
L	H	L	XXXXXXXX	HLHLHLHL	LATCH EVEN CHECKERBOARD
L	H	H	LHLHLHLH	LHLHLHLH	LOAD ODD CHECKERBOARD
L	H	L	XXXXXXXX	LHLHLHLH	LATCH ODD CHECKERBOARD
H	X	X	XXXXXXXX	ZZZZZZZZ	TEST HI-Z

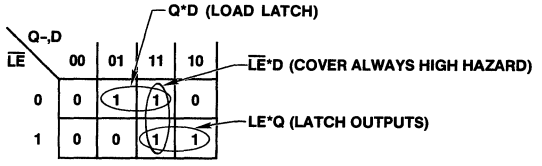
DESCRIPTION

THIS PAL20L10 IMPLEMENTS AN OCTAL LATCH WITH PRESET AND THREE-STATE OUTPUTS. THE LATCH PASSES EIGHT BITS OF DATA (D7-D0) TO THE OUTPUTS (Q7-Q0) WHEN LATCH ENABLE IS FALSE (/LE=HIGH) AND PRESET IS FALSE (/PR=HIGH). THE DATA IS LATCHED WHEN LATCH ENABLE IS TRUE (/LE=LOW) AND PRESET IS FALSE (/PR=HIGH). THE OUTPUTS WILL BE ALL HIGH WHEN PRESET IS TRUE (/PR=LOW) REGARDLESS OF LATCH ENABLE. THE OUTPUTS WILL BE DISABLED (HI-Z) WHEN OUTPUT CONTROL IS TRUE (/OC=HIGH) REGARDLESS OF ANY OTHER INPUTS.

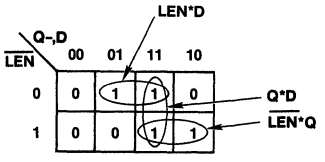
THESE OPERATIONS ARE EXERCISED IN THE FUNCTION TABLE AND SUMMARIZED IN THE OPERATIONS TABLE:

/OE	/PR	/LE	D7-D0	Q7-Q0	COMMENTS
H	X	X	X	Z	HI-Z
L	L	X	X	H	PRESET
L	H	H	D	D	LOAD LATCH
L	H	L	X	Q	LATCH OUTPUT

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



How to do a "Clean" Latch in a PAL Conditional Three-State Output:



LEN	Q	D
L	D	D
H	Q	X

ACTIVE HIGH LOGIC

```
Q = LEN * D          ; LOAD LATCH
  + LEN * Q          ; LATCH OUTPUT
  + D * Q            ; COVER HAZARD
```

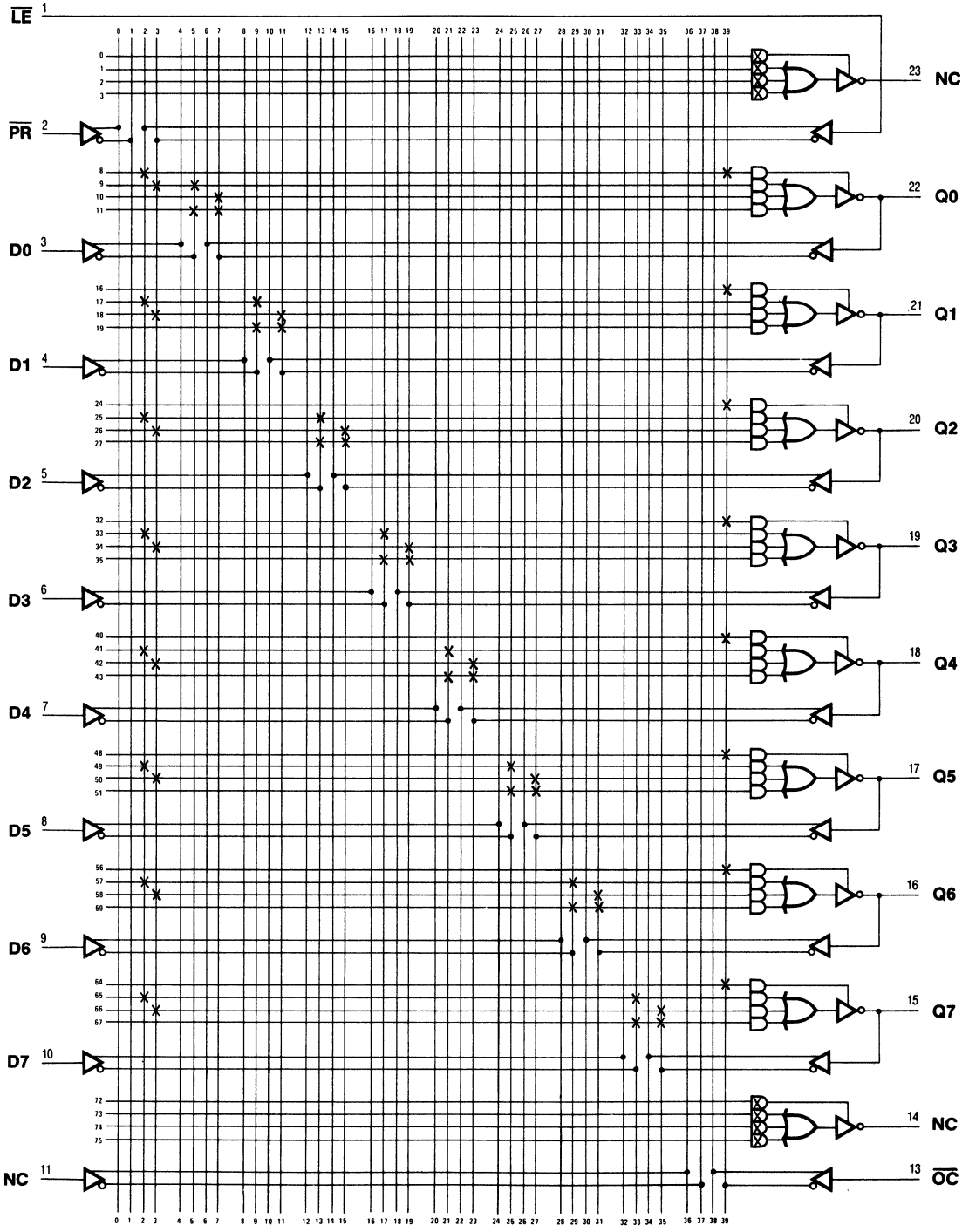
ACTIVE LOW LOGIC

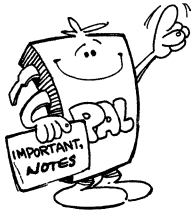
```
Q= LEN * D           ; LOAD LATCH
 + LEN * Q           ; LATCH OUTPUT
 + D * Q             ; COVER HAZARD
```

See page 8-80 for Latch Testability Information.

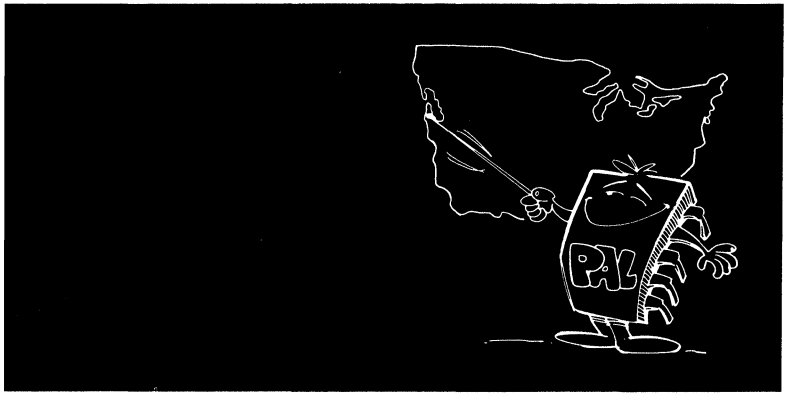
Clean Octal Latch

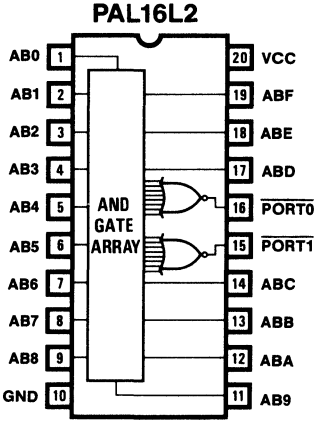
Logic Diagram PAL20L10




IMPORTANT
NOTES

Memory Mapped I/O





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:

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

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

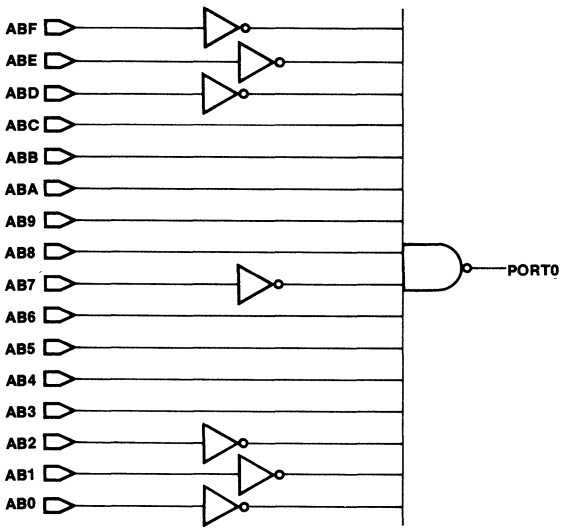
The above example shows address decoding for memory locations 1F78 hex and 1F79 hex. Equation terms can be changed to accommodate any 16 bit address.

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

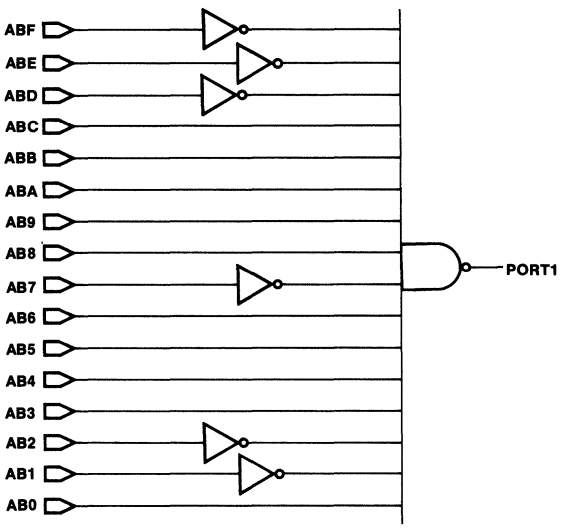
The PAL provides a single chip decoder for use in memory mapped I/O operations. This technique lowers interface parts counts and allows users an effective way to interface I/O devices to the microcomputer system.

PORT0 = 1F78



MEMORY MAPPED I/O

PORT1 = 1F79



MEMORY MAPPED I/O

```
PAL16L2                                          PAL DESIGN SPECIFICATION
MMAP                                             BIRKNER/COLI 06/29/81
MEMORY MAPPED I/O
MMI SUNNYVALE, CALIFORNIA
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  ;SELECT PORT0
      *  AB8* AB9* ABA *ABB* ABC*/ABD*/ABE*/ABF  ;   (1F78)

PORT1 =  AB0*/AB1*/AB2* AB3* AB4* AB5* AB6*/AB7  ;SELECT PORT1
      *  AB8* AB9* ABA* ABB* ABC*/ABD*/ABE*/ABF  ;   (1F79)


FUNCTION TABLE

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

;---INPUTS AB----      ---OUTPUTS---
;FEDCBA9876543210     /PORT0 /PORT1     COMMENTS
-------------------------------------------------------------------------
 LLLLHHHHLHHHHLLL        H      H       TEST 0F78
 LLLHHHHLLHHHHLLL        H      H       TEST 1E78
 LLLHHHHHHHHHHLLL        H      H       TEST 1FF8
 LLLHHHHHLHHHLLLL        H      H       TEST 1F70
 LLLHHHHHLHHHHLLL        L      H       TEST 1F78
 LLLHHHHHLHHHHLLH        H      L       TEST 1F79
 LLLHHHHHLHHHHLHL        H      H       TEST 1F7A
 LLLHHHHHHHHHHLLH        H      H       TEST 1FF9
 LLLHHHHLLHHHHLLH        H      H       TEST 1E79
 LLHHHHHHLHHHHLLH        H      H       TEST 3F79
 LLLLLLLLLLLLLLLL        H      H       TEST ALL L'S
 HHHHHHHHHHHHHHHH        H      H       TEST ALL H'S
 LHLHLHLHLHLHLHLH        H      H       TEST ODD  CHECKERBOARD
 HLHLHLHLHLHLHLHL        H      H       TEST EVEN CHECKERBOARD
 LLHHLLHHLLHHLLHH        H      H       TEST ODD  DOUBLE CHECKERBOARD
 HHLLHHLLHHLLHHLL        H      H       TEST EVEN DOUBLE CHECKERBOARD
-------------------------------------------------------------------------
```

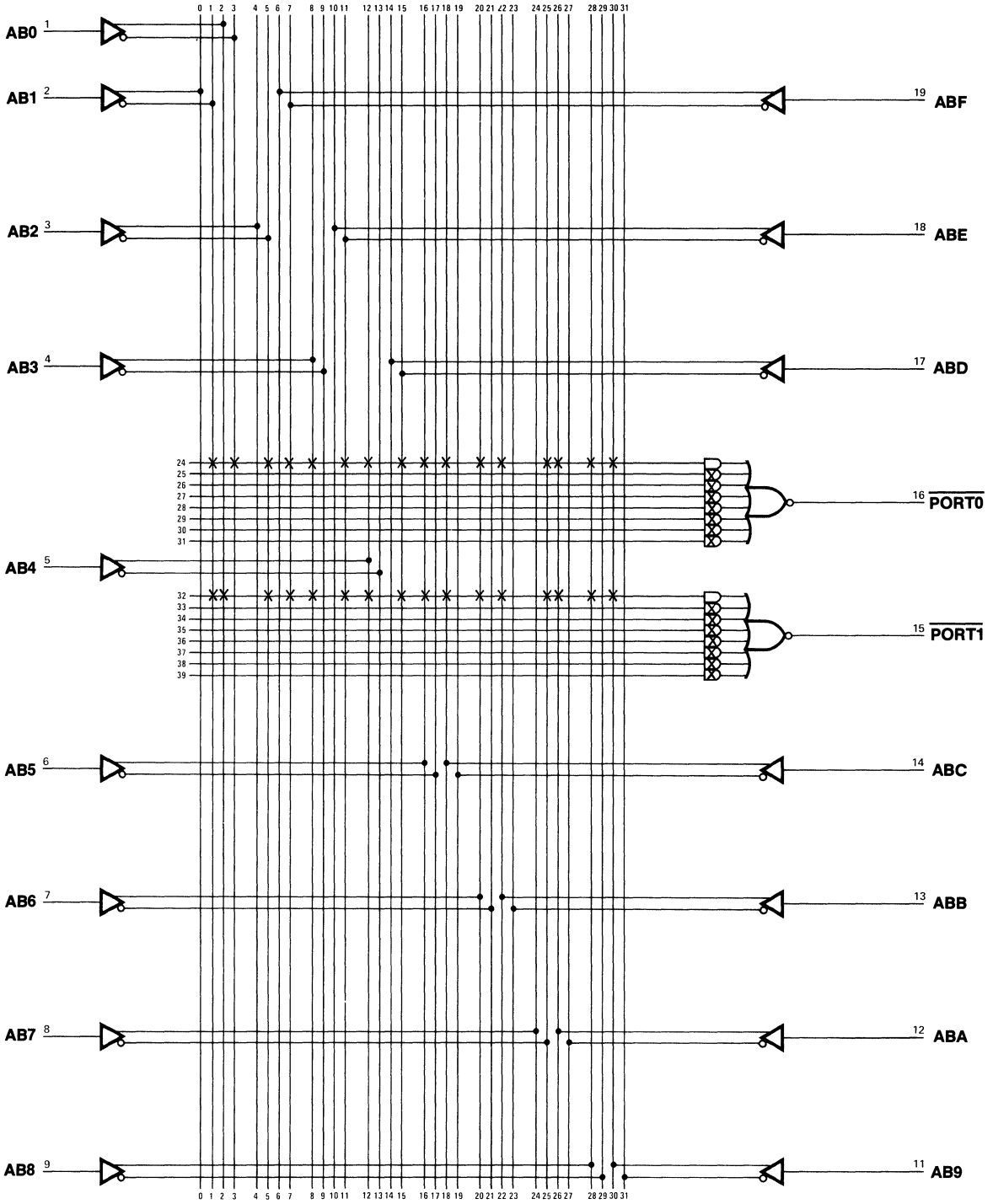
DESCRIPTION

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

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

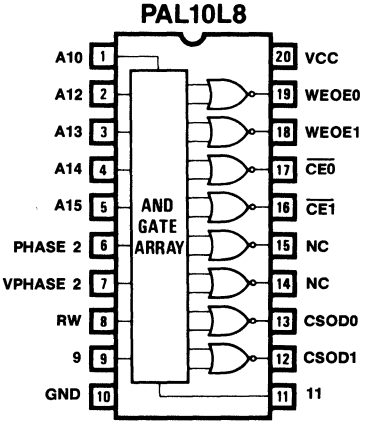
Memory Mapped I/O

Logic Diagram PAL16L2




IMPORTANT
NOTES

Memory Interface Logic for 6800 Microprocessor Bus



Memory Interface Logic for 6800 Microprocessor Bus — Logic Schematic

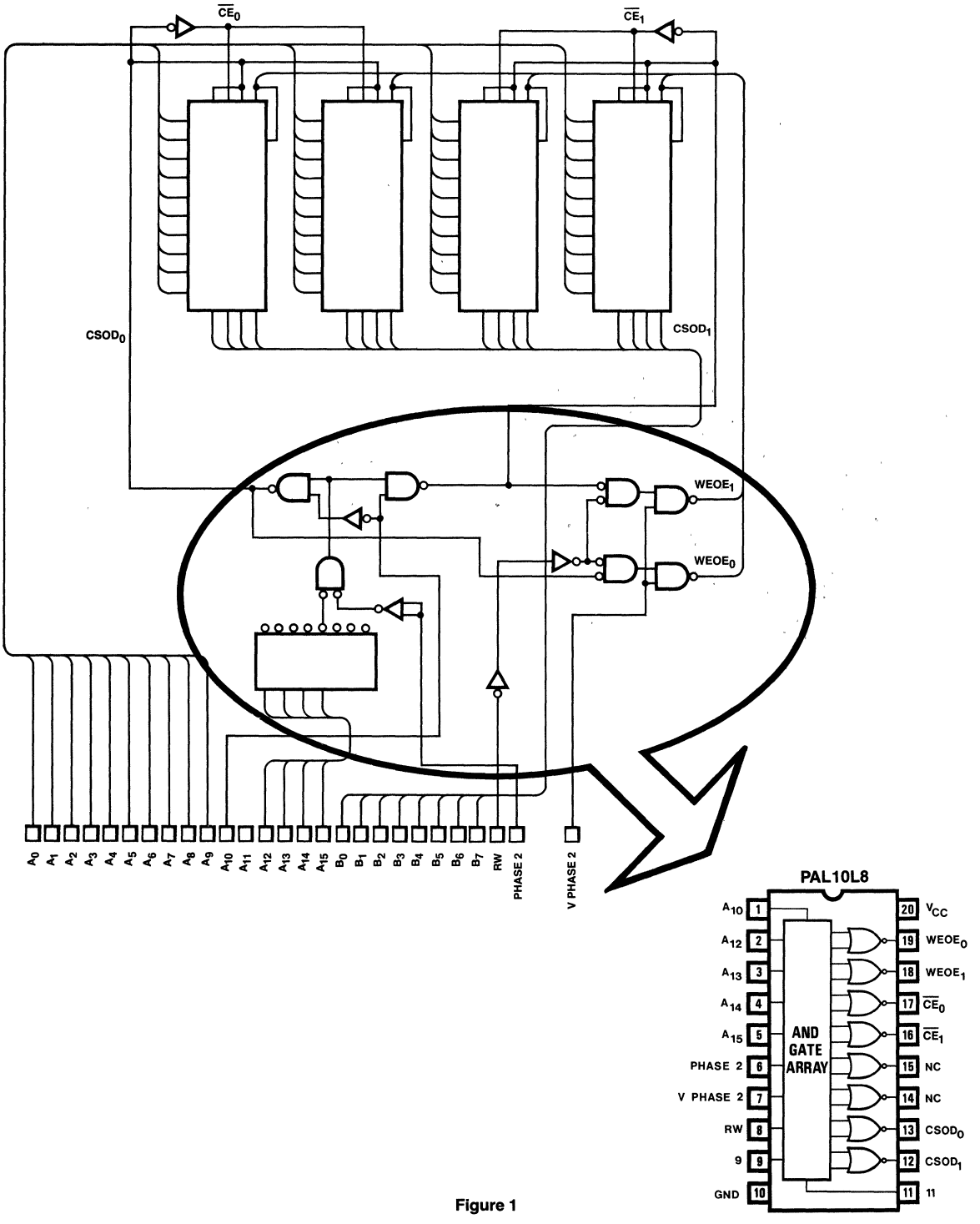


Figure 1

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:

$$\overline{WEOE0} = \overline{A10} \cdot \overline{A12} \cdot \overline{A13} \cdot A14 \cdot \overline{A15} \cdot PHASE2 \cdot VPHASE2 \cdot \overline{RW}$$
$$\overline{WEOE1} = A10 \cdot \overline{A12} \cdot \overline{A13} \cdot A14 \cdot A15 \cdot PHASE2 \cdot VPHASE2 \cdot \overline{RW}$$
$$CSOD0 = \overline{A10} \cdot \overline{A12} \cdot \overline{A13} \cdot \overline{A14} \cdot \overline{A15} \cdot PHASE2$$
$$CSOD1 = A10 \cdot \overline{A12} \cdot \overline{A13} \cdot A14 \cdot \overline{A15} \cdot PHASE2$$
$$CE0 = \overline{CSOD0}$$
$$CE1 = \overline{CSOD1}$$

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.

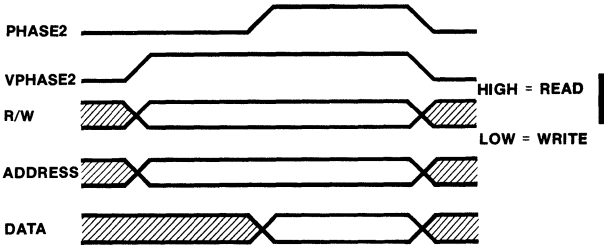


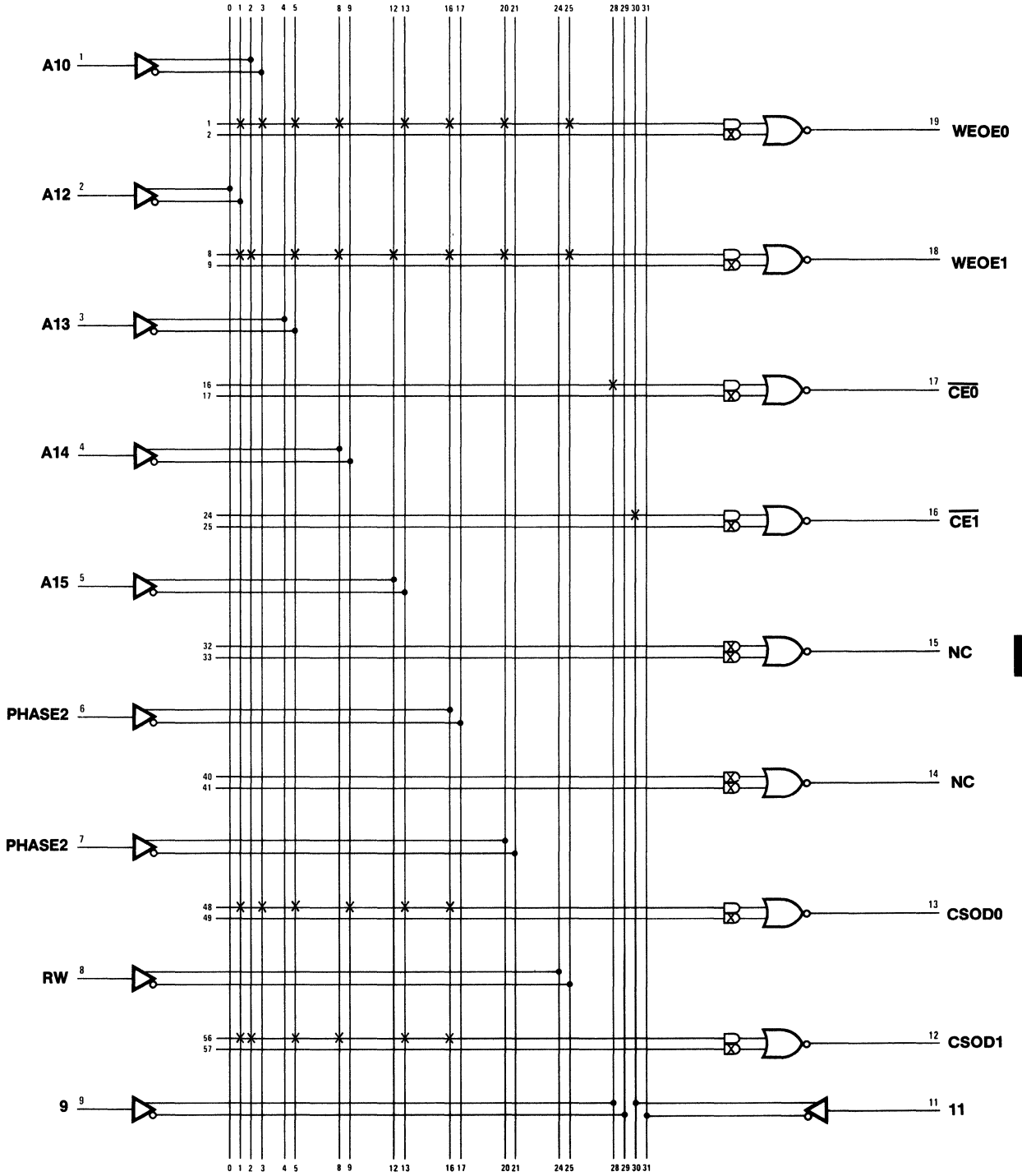
Figure 2

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.

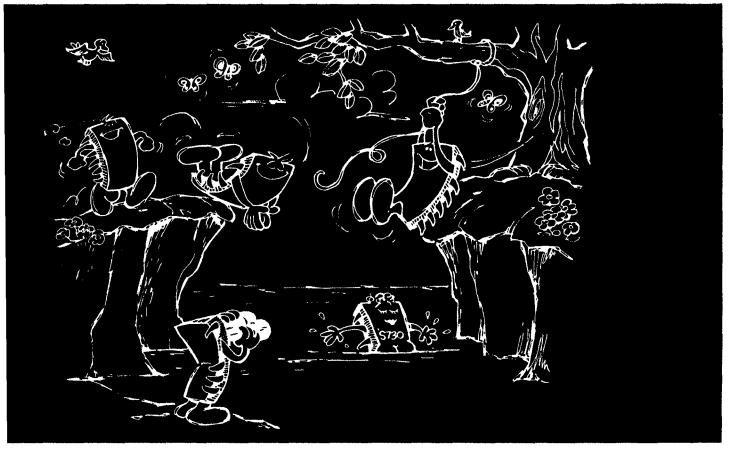
```
PAL10L8                                                    PAL DESIGN SPECIFICATION
MIL                                                          BIRKNER/COLI 07/21/81
MEMORY INTERFACE LOGIC FOR 6800 MICROPROCESSOR BUS
MMI SUNNYVALE, CALIFORNIA
A10 A12 A13 A14 A15 PHASE2 VPHASE2 RW 9 GND
11 CSOD1 CSOD0 NC NC /CE1 /CE0 WEOE1 WEOE0 VCC


/WEOE0 = /A10*/A12*/A13* A14*/A15* PHASE2* VPHASE2*/RW ;DEC WRITE ENABLE CHIP 0

/WEOE1 =  A10*/A12*/A13* A14* A15* PHASE2* VPHASE2*/RW ;DEC WRITE ENABLE CHIP 1

/CSOD0 = /A10*/A12*/A13*/A14*/A15* PHASE2          ;DECODE OUTPUT DISABLE CHIP 0

/CSOD1 =  A10*/A12*/A13* A14*/A15* PHASE2          ;DECODE OUTPUT DISABLE CHIP 1

   CE0 = 9   ;ENABLE CHIP 0 (COMPLEMENT OF CSOD0)

   CE1 = 11  ;ENABLE CHIP 1 (COMPLEMENT OF CSOD1)


FUNCTION TABLE

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

;ADD BUS
; 11111   PHASE R PINS  WRITE-ENABLE  CHIP-ENABLE  OUTPUT-DISABLE
; 02345   2  V2 W 9 11     0    1        0   1        0      1     COMMENTS
-------------------------------------------------------------------------------
  LLLHL   H  H  L H H      L    H        L   L        H      H     WRITE EN 0
  HLLHH   H  H  L H H      H    L        L   L        H      H     WRITE EN 1
  LLLLL   H  X  X L H      H    H        H   L        L      H     OUTPUT EN 0
  HLLHL   H  X  X H L      H    H        L   H        H      L     OUTPUT EN 1
  LLLHL   H  H  H H H      H    H        L   L        H      H     RW=H
  HLLHH   H  H  H H H      H    H        L   L        H      H     RW=H
  XXXXX   L  X  X H H      H    H        L   L        H      H     PHASE2=L
  LLLHL   H  L  L H H      H    H        L   L        H      H     VPHASE2=L
  HLLHH   H  L  L H H      H    H        L   L        H      H     VPHASE2=L
-------------------------------------------------------------------------------


DESCRIPTION

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

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

Memory Interface Logic for 6800 Microprocessor Bus — Logic Diagram PAL10L8

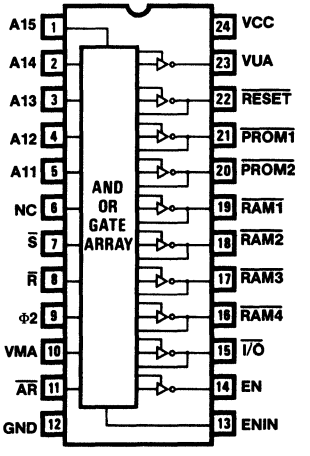



IMPORTANT
NOTES

MC6800 Microprocessor Interface



PAL20L10



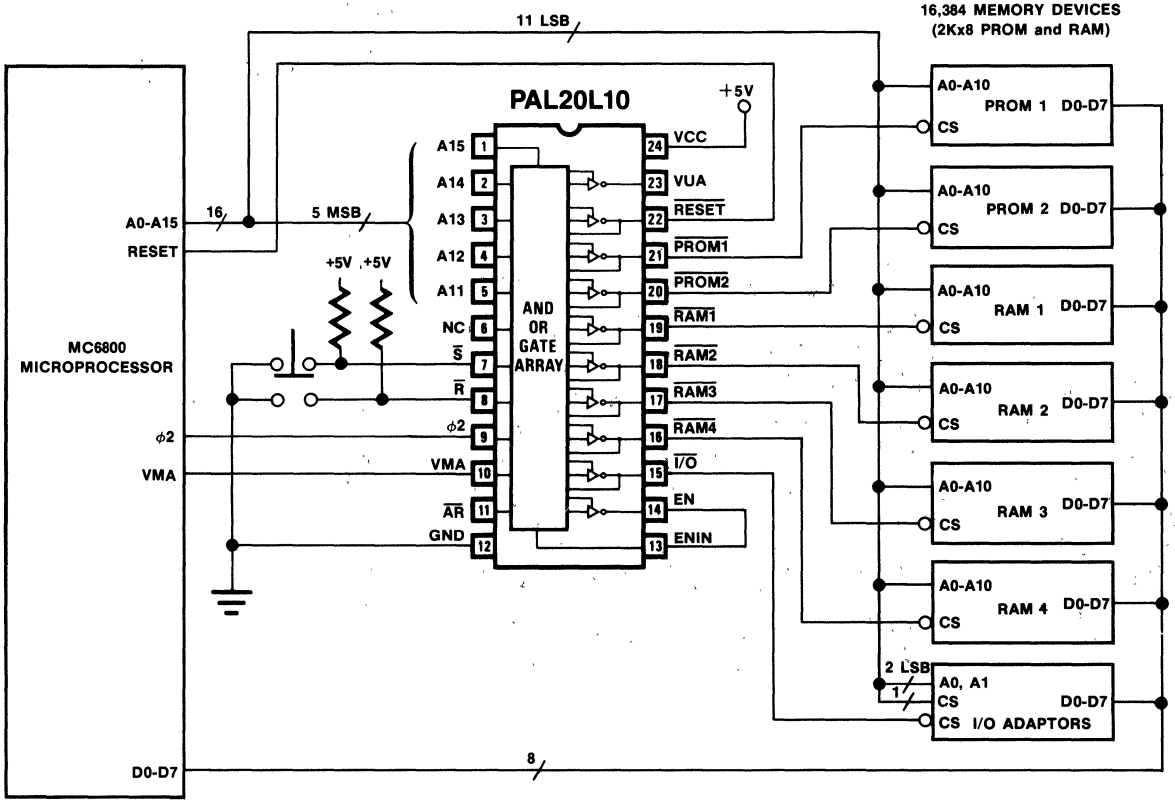


Figure 1. PAL Implementation

Functional Description

This application describes the use of a PAL20L10 as the interface element between an MC6800 microprocessor and it's system components on a single board computer. The functions it performs, previously done with random logic, are address decoding and memory/I-O select, reset signal generation and control of the buffer which interfaces the data bus to other boards in the system.

Circuit Operation

In addition to an MC6800 microprocessor, the Micromodule™ contains the following system components:

2 EPROM — either 2716 (2Kx8) or 2732 (4Kx8)
4 Static RAM — either 4108 (1Kx8) or 2016 (2Kx8)
2 PIA — MC6821 type 20 parallel I/O port adaptor
2 ACIA — MC6850 type 1 serial I/O port adaptor
1 PTM — MC6840 type 3 programmable timer modules

™ Micromodule is a trademark of Motorola

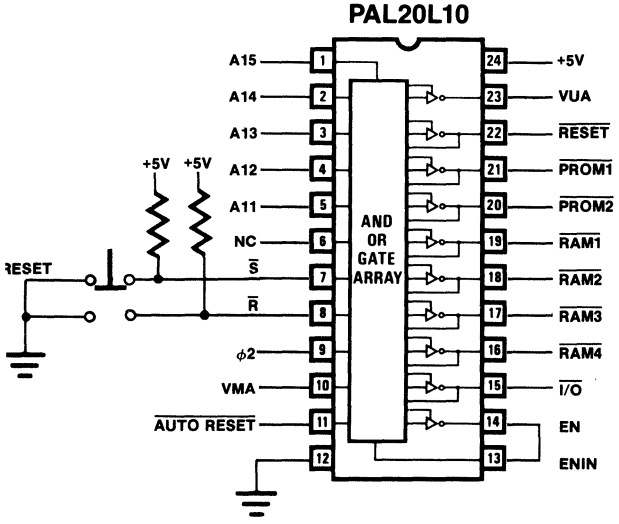
- Interface to bottom edge connector, interfaces this board to other boards in the system.
- Auto-reset function
- Pushbutton controlled reset function

An implementation of this system using TTL is shown in Figure 2.

Part counts for the TTL implementation:

1	74LS00	Quad 2-input NAND
4	74LS30	Single 8-input NAND
1	74LS04	Hex inverter
1	74LS133	Single 13-input NAND
1	74LS138	3-to-8 decoder
8	chips total	

All functions could be accomplished by the use of a single PAL20L10:



RESET SIGNAL GENERATION CIRCUIT

The logic equations for all interface signals can be expressed as follows:

$\overline{PROM1} = A15 \cdot A14 \cdot A13 \cdot A12 \cdot VMA \cdot \phi 2 \cdot \overline{RESET}$

$\overline{PROM2} = A15 \cdot A14 \cdot A13 \cdot \overline{A12} \cdot VMA \cdot \phi 2 \cdot \overline{RESET}$

$\overline{RAM1} = \overline{A15} \cdot \overline{A14} \cdot \overline{A13} \cdot \overline{A12} \cdot \overline{A11} \cdot VMA \cdot \phi 2 \cdot \overline{RESET}$

$\overline{RAM2} = \overline{A15} \cdot \overline{A14} \cdot \overline{A13} \cdot \overline{A12} \cdot A11 \cdot VMA \cdot \phi 2 \cdot \overline{RESET}$

$\overline{RAM3} = \overline{A15} \cdot \overline{A14} \cdot \overline{A13} \cdot A12 \cdot \overline{A11} \cdot VMA \cdot \phi 2 \cdot \overline{RESET}$

$\overline{RAM4} = \overline{A15} \cdot \overline{A14} \cdot \overline{A13} \cdot A12 \cdot A11 \cdot VMA \cdot \phi 2 \cdot \overline{RESET}$

$\overline{I/O} = A15 \cdot A14 \cdot \overline{A13} \cdot A12 \cdot A11 \cdot VMA \cdot \phi 2 \cdot \overline{RESET}$

$\overline{EN} = \overline{PROM1} \cdot \overline{PROM2} \cdot \overline{RAM1} \cdot \overline{RAM2} \cdot \overline{RAM3} \cdot \overline{RAM4} \cdot \overline{I/O} \cdot VMA \cdot \overline{RESET}$

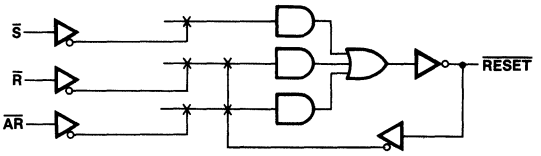
$VUA = EN$

This corresponds to the desired memory map:

DEVICE	HEX ADDRESS
PROM1	F000-FFFF
PROM2	E000-EFFF
RAM1	0000-07FF
RAM2	0800-0FFF
RAM3	1000-17FF
RAM4	1800-1FFF
I/O*	D800-DFFF

*Decoding between the five I/O devices is provided by routing the appropriate address signal to the extra CS input on the I/O device

The logic circuit for the reset signal generation is as follows:



Conclusion

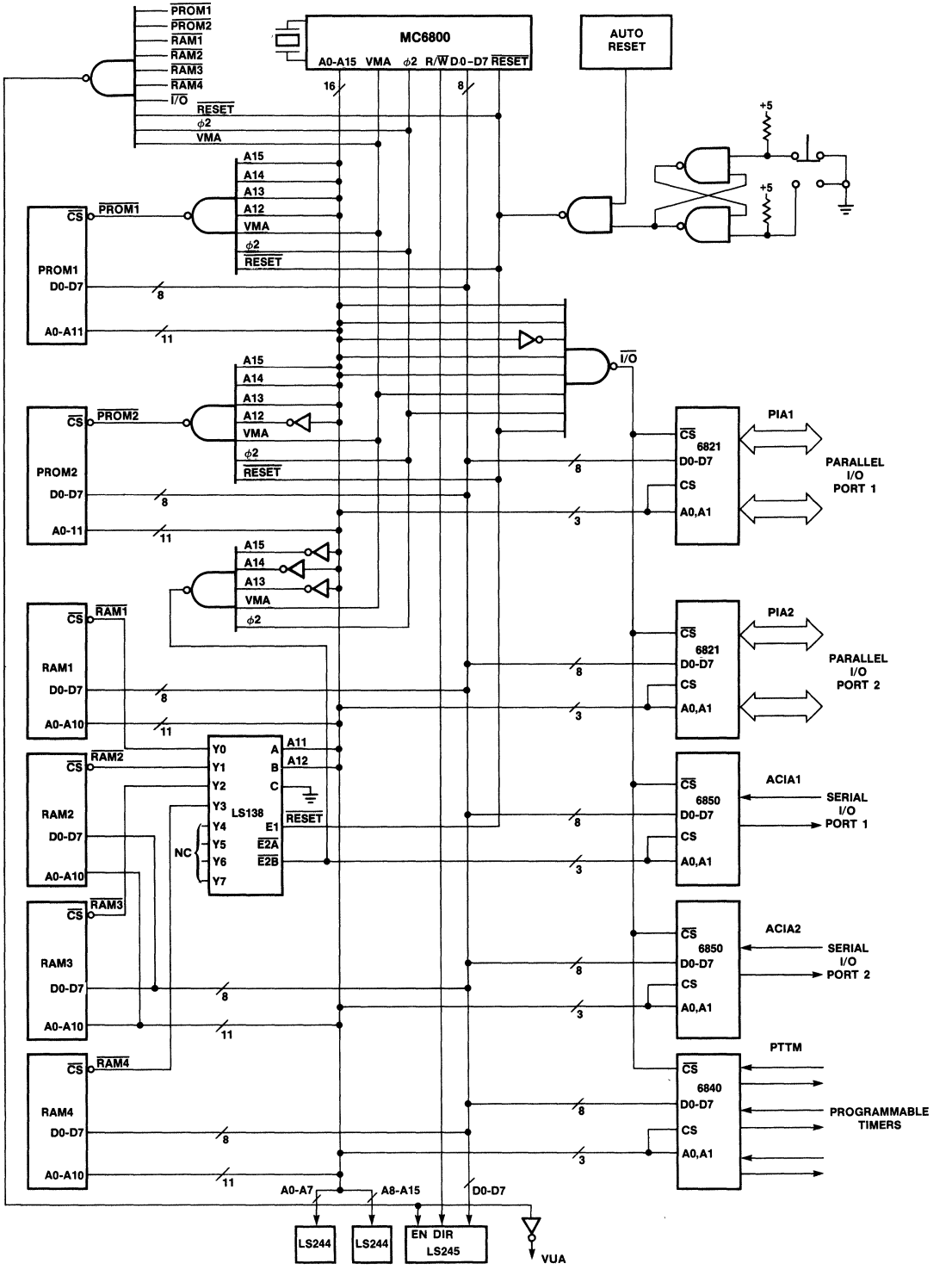
The PAL implementation as shown in Figure 1 has the following advantages over the TTL version:

- Lowers chip count by a factor of 8 to 1
- Allows movement of system components through the memory space with only a program change. The TTL version could require a board change.†
- Allows the freeing up of address space should all the listed memory and/or I/O not be required (or desired) on the microcomputer board. This space would then be available for use by other boards in the system. Again, to do this on the TTL version could require a board change. With the PAL, only a program change is required.
- Unused input pin (pin #8) can be used for additional address decoding in an upgraded system.

†For additional information on this topic consult "Programmable Array Logic Leads to Flexible Applications of 8-Bit Wide Memories" by Bernard Brafman, Monolithic Memories Article Reprint AR-114 or page 8-40 in the PAL Handbook.

6

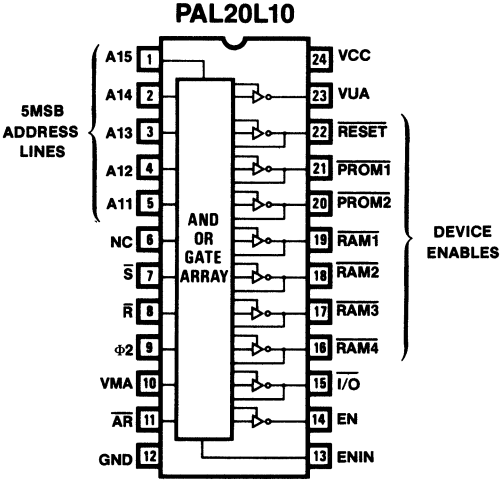
TTL Implementation



```
PAL20L10                                                    PAL DESIGN SPECIFICATION
P7073                                                         COLI/SACKETT 09/14/82
MC6800 MICROPROCESSOR INTERFACE
MMI SUNNYVALE, CALIFORNIA
A15  A14 A13  A12   A11    NC    /S    /R     PH2    VMA  /AR GND
ENIN EN  /IO /RAM4 /RAM3 /RAM2 /RAM1 /PROM2 /PROM1 /RESET VUA VCC


IF(VCC)  PROM1 =  A15* A14* A13* A12     * VMA* PH2*/RESET    ;PROM1, F000-FFFF

IF(VCC)  PROM2 =  A15* A14* A13*/A12     * VMA* PH2*/RESET    ;PROM2, E000-EFFF

IF(VCC)  RAM1  = /A15*/A14*/A13*/A12*/A11* VMA* PH2*/RESET    ;RAM1,  0000-07FF

IF(VCC)  RAM2  = /A15*/A14*/A13*/A12* A11* VMA* PH2*/RESET    ;RAM2,  0800-0FFF

IF(VCC)  RAM3  = /A15*/A14*/A13* A12*/A11* VMA* PH2*/RESET    ;RAM3,  1000-17FF

IF(VCC)  RAM4  = /A15*/A14*/A13* A12* A11* VMA* PH2*/RESET    ;RAM4,  1800-1FFF

IF(VCC)  IO    =  A15* A14*/A13* A12* A11* VMA* PH2*/RESET    ;I/O,   D800-DFFF

IF(VCC) /EN    = /PROM1*/PROM2*/RAM1*/RAM2*/RAM3*/RAM4*/IO* VMA*/RESET ;EN=/VUA

IF(VCC) /VUA   =  ENIN          ;ASSERTIVE HIGH VUA SIGNAL (INVERT EN FEEDBACK)

IF(VCC)  RESET =  S             ;SET
               + /R * RESET     ;RESET
               + /AR* RESET     ;AUTO RESET


FUNCTION TABLE

A15 A14 A13 A12 A11 /S /R /AR /RESET PH2 VMA /PROM1 /PROM2 /RAM1 /RAM2 /RAM3
/RAM4 /IO EN ENIN VUA

;ADDR1  S-R      /RE           PROM  --RAM--       ENABLE
;54321 /S /R /AR SET  PH2 VMA  1  2  1 2 3 4  I/O  OUT IN VUA   COMMENT
-------------------------------------------------------------------------------
 HHHHX  L  H  L   L    L   H   H  H  H H H H   H    H  H   L    RESET (/S=L)
 HHHHX  H  L  H   L    L   H   H  H  H H H H   H    H  H   L    AUTO-RESET
 HHHHX  H  L  L   H    L   H   H  H  H H H H   H    L  L   H    NO SELECT PH2=L
 HHHHX  H  L  L   H    H   L   H  H  H H H H   H    H  H   L    NO SELECT VMA=L
 HHHHX  H  L  L   H    H   H   L  H  H H H H   H    H  H   L    SELECT PROM1
 HHHLX  H  L  L   H    H   H   H  L  H H H H   H    H  H   L    SELECT PROM2
 LLLLL  H  L  L   H    H   H   H  H  L H H H   H    H  H   L    SELECT RAM1
 LLLLH  H  L  L   H    H   H   H  H  H L H H   H    H  H   L    SELECT RAM2
 LLLHL  H  L  L   H    H   H   H  H  H H L H   H    H  H   L    SELECT RAM3
 LLLHH  H  L  L   H    H   H   H  H  H H H L   H    H  H   L    SELECT RAM4
 HHLHH  H  L  L   H    H   H   H  H  H H H H   L    H  H   L    SELECT I/O PORT
-------------------------------------------------------------------------------
```

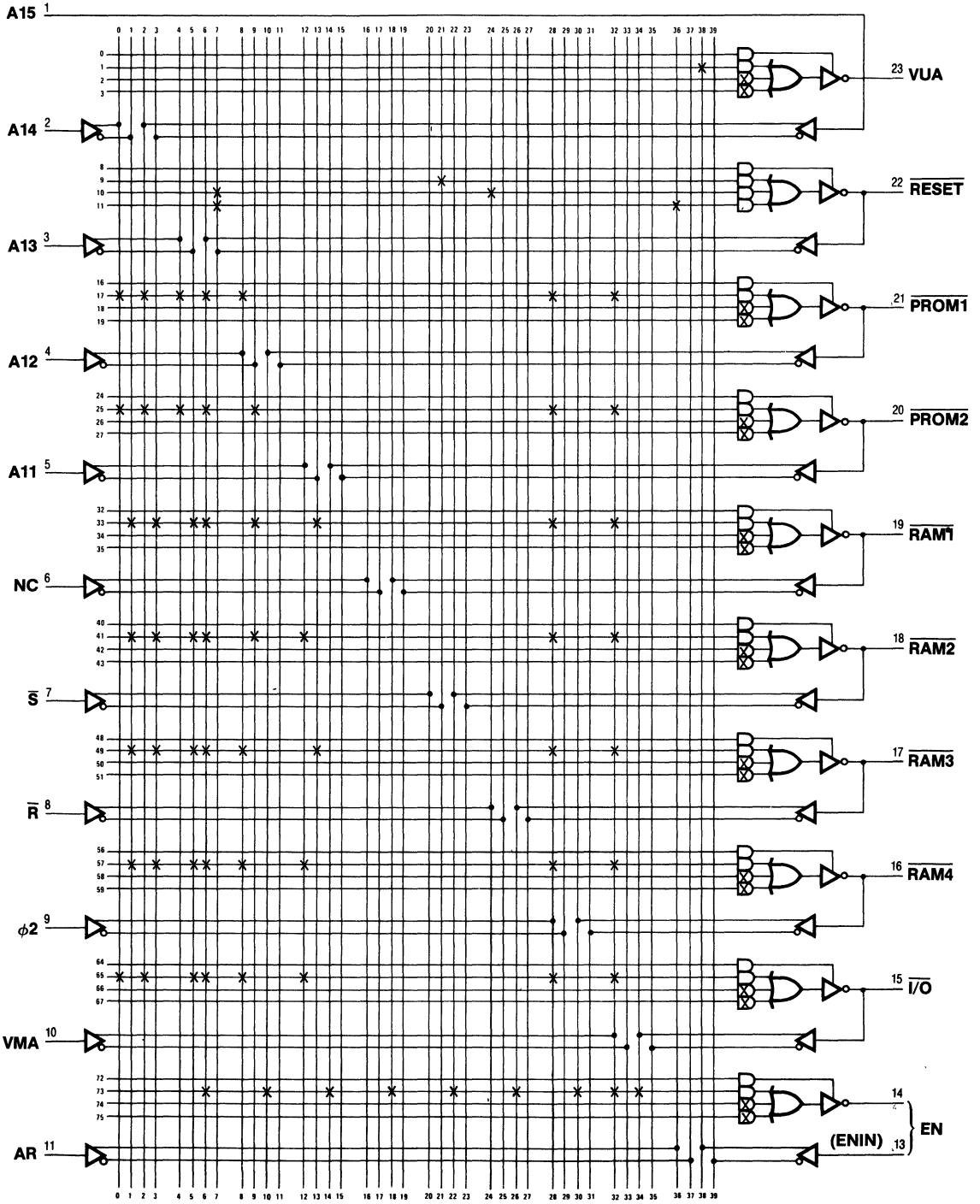
DESCRIPTION

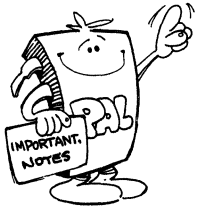
THIS PAL20L10 INTERFACES BETWEEN THE MOTOROLA MC6800 MICROPROCESSOR AND ITS SYSTEM COMPONENTS ON A SINGLE BOARD COMPUTER. THE FUNCTIONS IT PERFORMS, PREVIOUSLY DONE WITH RANDOM LOGIC ARE: ADDRESS DECODING, MEMORY AND I/O SELECT, RESET SIGNAL GENERATION, AND CONTROL OF THE BUFFER WHICH INTERFACES THE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



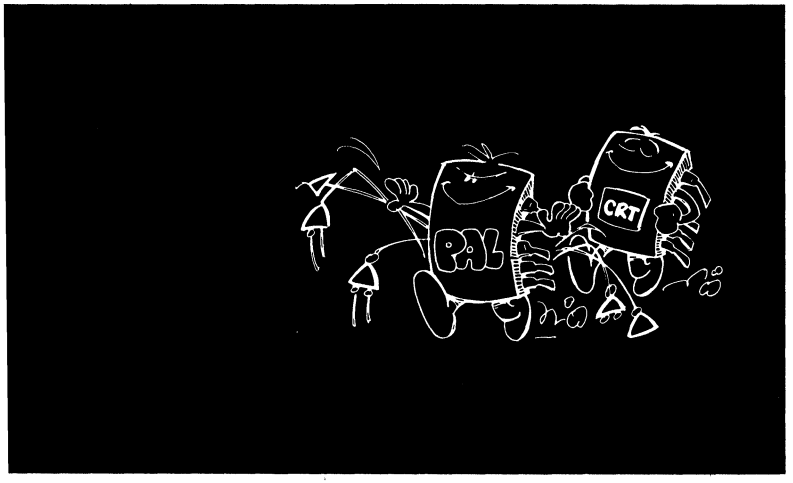
MC6800 Microprocessor Interface

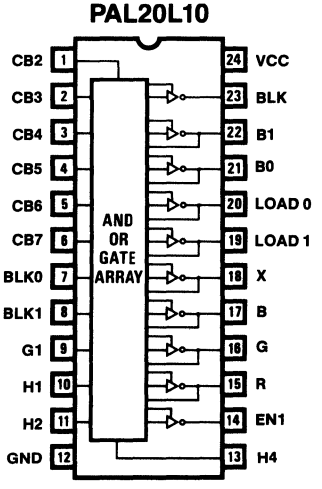
Logic Diagram PAL20L10




IMPORTANT
NOTES

Video Logic


PAL
CRT


PAL20L10
CB2 1
CB3 2
CB4 3
CB5 4
CB6 5
CB7 6
BLK0 7
BLK1 8
G1 9
H1 10
H2 11
GND 12
24 VCC
23 BLK
22 B1
21 B0
20 LOAD 0
19 LOAD 1
18 X
17 B
16 G
15 R
14 EN1
13 H4
AND
OR
GATE
ARRAY

Functional Description

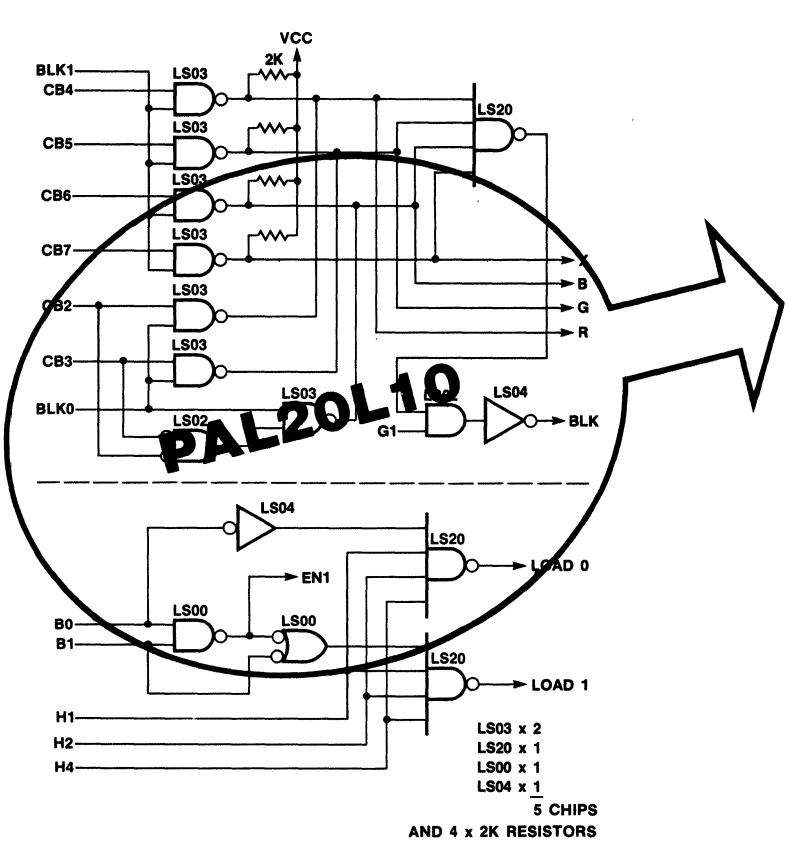
Many microcomputer systems use a CRT monitor or conventional television rather than a VDU perhaps because of cost reduction or because a color display is necessary.

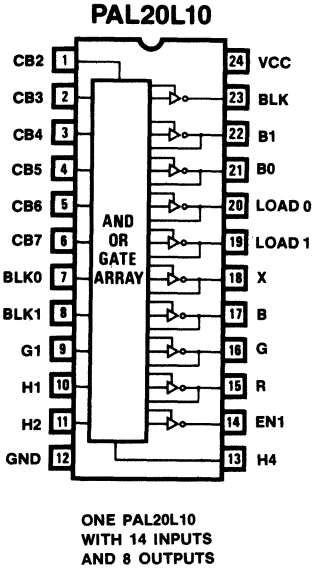
The circuit diagram shows all of the TTL chips used on a video interface board.

The top portion is the color and blanking drive decoding and the lower is the enable logic for PROMs and registers containing driving data. The LS03s are open collector drivers and the two bank drive decoders are wire-ORed using 2kΩ resistors.

PAL Implementation

All of the logic can be put into one PAL20L10, also eliminating the pull-up resistors. This provides a dramatic reduction in component count and board size. The wire-OR configuration is an obvious candidate for PAL implementation, since the logic function /F = A * B + C * D is exactly in the PAL configuration.





ONE PAL20L10
WITH 14 INPUTS
AND 8 OUTPUTS

```
PAL20L10                                          PAL DESIGN SPECIFICATION
VIDLOG                                              HARRY HUGHES 02/18/81
VIDEO LOGIC
MMI ENGLAND
CB2 CB3 CB4 CB5 CB6 CB7 BLK0 BLK1 G1 H1 H2 GND
H4 EN1 R G B X LOAD1 LOAD0 B0 B1 BLK VCC

IF (VCC) /LOAD0 = /B0* H1* H2* H4

IF (VCC) /LOAD1 =  H1* H2* H4*/B1
                +  H1* H2* H4* B1* B0

IF (VCC) /EN1   =  B1* B0

IF (VCC) /R     =  CB4* BLK1
                +  CB2* BLK0

IF (VCC) /G     =  CB5* BLK1
                +  CB3* BLK0

IF (VCC) /B     =  CB6* BLK1
                + /CB2*/CB3* BLK0

IF (VCC) /X     =  CB7* BLK1

IF (VCC) /BLK   = /G1*/BLK1*/BLK0
                + /G1*/BLK0*/CB4*/CB5*/CB6*/CB7
```

FUNCTION TABLE

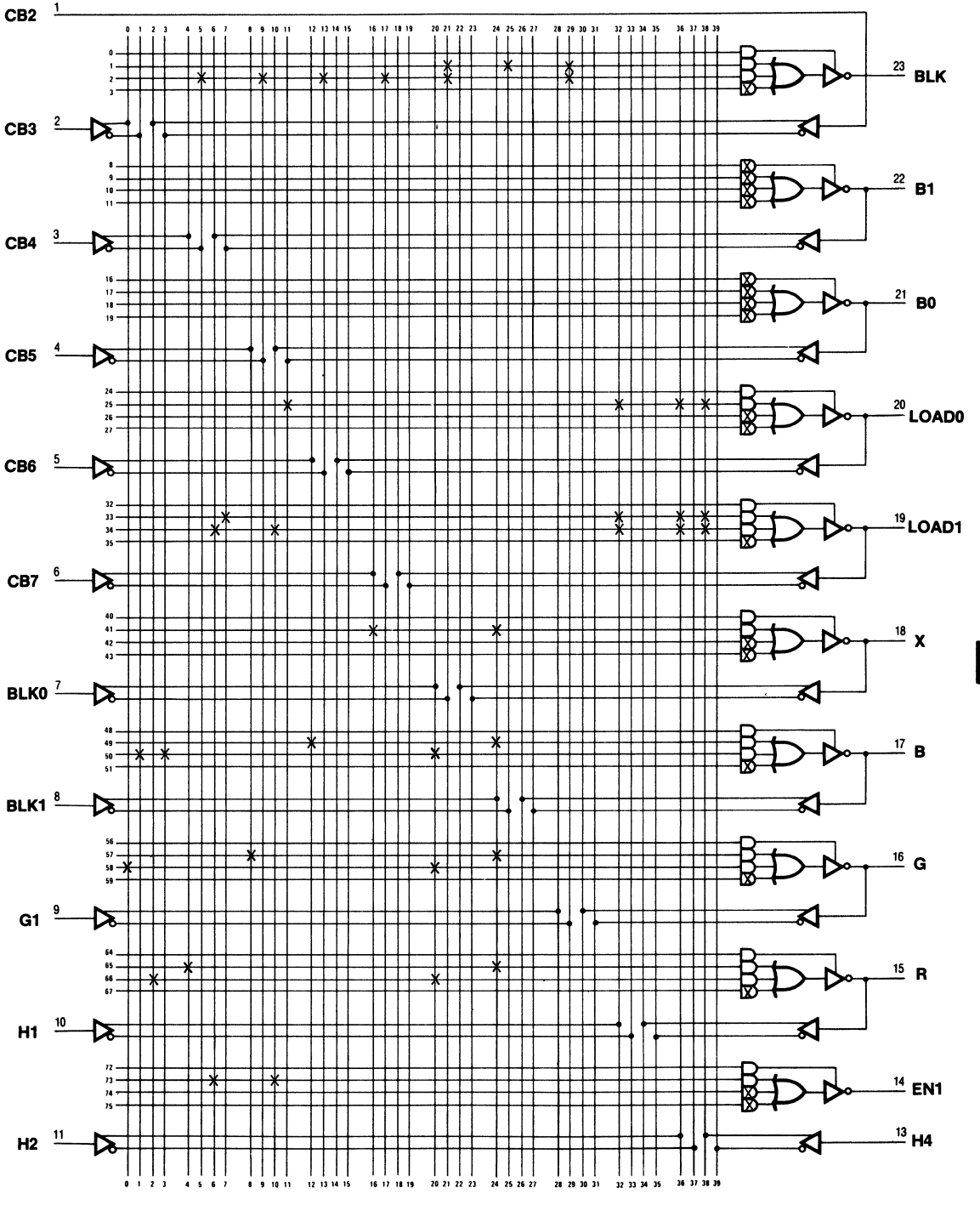
CB2 CB3 CB4 CB5 CB6 CB7 BLK0 BLK1 G1 H1 H2 H4 B1 B0 EN1 R G B X LOAD1 LOAD0 BLK

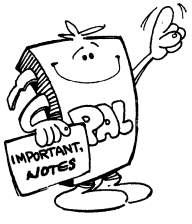
;CCCCCC	-BLANK-					E									
;BBBBBB	BLK	BLK	G	HHH	BB	N	R	G	B	X	LOAD		BLK	COMMENTS	
;234567	0	1	1	124	10	1					1	0			
XXXXXX	X	X	X	HHH	HL	X	X	X	X	X	H	L	X	LOAD0	
XXXXXX	X	X	X	HHH	LL	X	X	X	X	X	L	L	X	LOAD1	
XXXXXX	X	X	X	HHH	HH	L	X	X	X	X	L	H	X	LOAD1	
XXXXXX	X	X	X	LLL	HH	L	X	X	X	X	H	H	X	EN1	
XXHXXX	X	H	X	XXX	XX	X	L	X	X	X	X	X	X	R	
HXXXXX	H	X	X	XXX	XX	X	L	X	X	X	X	X	X	R	
XXXHXX	X	H	X	XXX	XX	X	X	L	X	X	X	X	X	G	
XHXXXX	H	X	X	XXX	XX	X	X	L	X	X	X	X	X	G	
XXXXHX	X	H	X	XXX	XX	X	X	X	L	X	X	X	X	B	
LLXXXX	H	X	X	XXX	XX	X	X	X	L	X	X	X	X	B	
XXXXXH	X	H	X	XXX	XX	X	X	X	X	L	X	X	X	X	
XXXXXX	L	L	L	XXX	XX	X	X	X	X	X	X	X	L	BLK	
XXLLLL	L	X	L	XXX	XX	X	X	X	X	X	X	X	L	BLK	

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

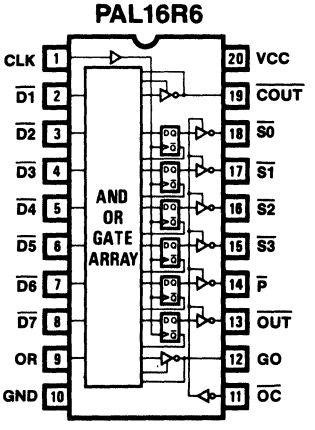
Video Logic

Logic Diagram PAL20L10




IMPORTANT NOTES

PAL/FIFO Performs UART Output Functions



A PAL 16R6 and a FIFO Perform UART Output Functions

Features:

- **Simple X1 clocking requirements.**
- **Automatic Parity generation (can be programmed to provide odd, even, mark, space parity or 8 data bit).**
- **Interfaces directly with MMI 67401 FIFO memory and automatically handles control handshake.**
- **Operates in single byte or burst mode.**
- **Automatic reset to 'Idle' mode on error.**

Operation

The clock input (pin 1) is the baud rate clock. Serial output data is centered on the negative edge of this same clock.

UART operation commences when the GO (pin 12) line is asserted (H1) and the FIFO is not empty (OR=HI).

The parallel input data is clocked out of the FIFO by the UART S3 output (pin 15) and held valid on the D1-D7 inputs (pins 2-8).

The UART generates a start bit (MARK = HI = 1) and than shifts out the data word, LSB first (pin 13). During this operation the parity bit is generated and appended to the data word.

The UART then generates a stop bit (SPACE = LO = O).

If the GO bit is still set, the UART continues by clocking the next byte out of the FIFO. This continues untill either the GO is low or OR (FIFO Output Ready, pin 9) is low, indicating the FIFO is empty.

If the UART is not being used with a FIFO, OR must be held HI to operate correctly.

The PAL16R6 may be reprogrammed (i.e. Design specification modified) to accept 8 data bits and no parity. The data bits may be output as mark = 1 or space = 1 (by design).

During normal operation the UART will transmit a complete character before sampling the GO line. The UART may be reset to 'Idle' at any time, however, by taking both the GO and the OR inputs low.

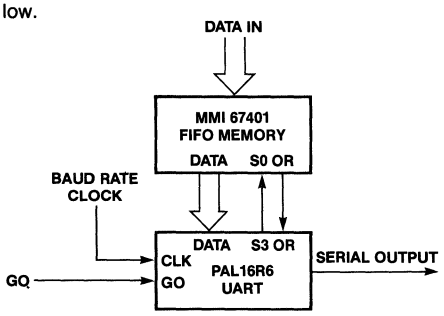


Fig. 1. UART Connection to FIFO

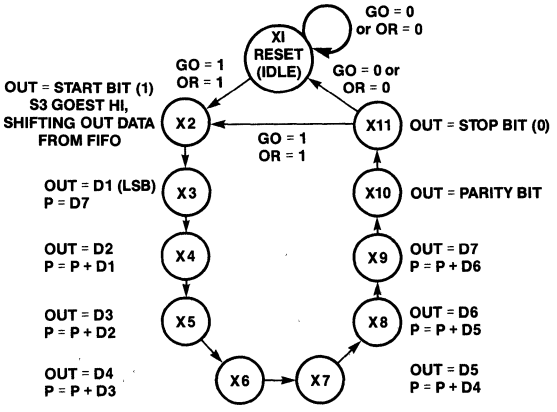


Fig. 2. State Diagram

The State Transition Table Yields the Following Equations:

$$S0 = \overline{GO}*S0*S1*\overline{S2}*S3 + GO*OR*S0*S1*\overline{S2}*S3 + \overline{S0}*S1*\overline{S2}*\overline{S3} + \overline{S0}*\overline{S1}*\overline{S2}*\overline{S3} + \overline{S0}*S1*S2*\overline{S3} + \overline{S0}*S1*\overline{S2}*S3 + GO*\overline{S0}*S1*S2*S3 + GO*OR*\overline{S0}*S1*\overline{S2}*S3$$

$$S1 = \overline{GO}*S0*S1*\overline{S2}*S3 + S0*\overline{S1}*\overline{S2}*S3 + \overline{S0}*S1*\overline{S2}*S3 + S0*\overline{S1}*\overline{S2}*\overline{S3} + \overline{S0}*S1*S2*\overline{S3} + S0*\overline{S1}*\overline{S2}*S3 + GO*OR*\overline{S0}*S1*\overline{S2}*S3 + \overline{GO}*\overline{S0}*S1*\overline{S2}*S3$$

$$S2 = \overline{GO}*S0*S1*\overline{S2}*S3 + S0*S1*\overline{S2}*S3 + \overline{S0}*\overline{S1}*\overline{S2}*\overline{S3} + S0*\overline{S1}*\overline{S2}*\overline{S3} + \overline{S0}*S1*S2*\overline{S3} + \overline{S0}*S1*\overline{S2}*S3 + \overline{GO}*\overline{S0}*S1*\overline{S2}*S3$$

$$S3 = \overline{GO}*\overline{S0}*S1*\overline{S2}*S3 + \overline{S0}*\overline{S1}*\overline{S2}*\overline{S3} + S0*\overline{S1}*\overline{S2}*S3 + \overline{GO}*S0*S1*\overline{S2}*S3$$

Minimise by inspection and add reset term.

$$S0 = \overline{GO}*S0*S1*\overline{S2}*S3 + GO*OR*S0*S1*\overline{S2}*S3 + S0*\overline{S2}*\overline{S3} + \overline{S0}*S1*\overline{S2}*\overline{S3} + \overline{S0}*S1*S2*\overline{S3} + \overline{GO}*\overline{S0}*S1*\overline{S2}*S3 + \overline{GO}*OR*S0*S1*\overline{S2}*S3 + \overline{GO}*\overline{OR}$$

$$S1 = \overline{GO}*S0*S1*\overline{S2}*S3 + \overline{S0}*S1*\overline{S3} + S0*\overline{S1}*\overline{S3} + S0*\overline{S1}*\overline{S2} + \overline{GO}*S0*S1*\overline{S2}*S3 + \overline{GO}*\overline{OR}$$

$$S2 = \overline{GO}*S0*S1*\overline{S2}*S3 + S0*S1*\overline{S2}*\overline{S3} + \overline{S0}*S0*\overline{S3} + \overline{S1}*S2*\overline{S3} + \overline{GO}*S0*S1*\overline{S2}*S3 + \overline{GO}*\overline{OR}$$

$$S3 = \overline{GO}*S0*S1*\overline{S2}*S3 + \overline{S0}*\overline{S1}*\overline{S2}*\overline{S3} + S0*\overline{S1}*\overline{S2}*S3 + \overline{GO}*S0*S1*\overline{S2}*S3 + \overline{GO}*\overline{OR}$$

Note. S3 is SO (shift out) input to FIFO memory
P is parity bit giving even parity on 8 bits total.
Parity is accumulated during the time D1 - D7 are shifted out

Inputs		Present State				Next State				Outputs		Comments
OR	GO	SO	S1	S2	S3	SO	S1	S2	S3	OUT	P	
X	O	1	1	1	1	1	1	1	1	O	X	Reset, Idle, Marking
1	1	1	1	1	1	1	O	O	O	1	X	Start Bit Out. LO, Space
X	X	1	O	O	O	O	1	O	O	D1	D7	Shift Out D1 (LSB)
X	X	O	1	O	O	1	1	O	O	D2	PO/OUT	Shift Out D2
X	X	1	1	O	O	O	O	1	O	D3	PO/OUT	Shift Out D3
X	X	O	O	1	O	1	O	1	O	D4	PO/OUT	Shift Out D4
X	X	1	O	1	O	O	1	1	O	D5	PO/OUT	Shift Out D5
X	X	O	1	1	O	1	1	1	O	D6	PO/OUT	Shift Out D6
X	X	1	1	1	O	O	O	O	O	D7	PO/OUT	Shift Out D7 (MSB)
X	X	O	O	O	O	1	O	O	1	P	X	Shift Out P (Parity)
X	X	1	O	O	1	O	1	O	1	O	X	Stop Bit Out, HI, MARK
1	O	O	1	O	1	1	1	1	1	O	X	Idle (Reset) If GO = LO
1	1	O	1	O	1	1	O	O	O		X	Get Next Byte If GO = HI
O	O	X	X	X	X	1	1	1	1	O	X	Reset

State Transition Table

Data Inputs = D1 - D7

S3 is the Shift Out (SO) control for the FIFO.

X = Don't Care

Serial Output Bits and Parity Bit Equations

$$COUT = S0*S1*S2*S3 + \overline{S0}*S1*\overline{S2}*S3 + P*S0*\overline{S1}*\overline{S2}*S3$$

$$OUT = \overline{COUT} + \overline{S0}*S1*\overline{S2}*\overline{S3}*D1 + S0*S1*\overline{S2}*\overline{S3}*D2 + \overline{S0}*\overline{S1}*S2*\overline{S3}*D3 + S0*\overline{S1}*S2*\overline{S3}*D4 + \overline{S0}*S1*S2*\overline{S3}*D5 + S0*S1*S2*\overline{S3}*D6 + \overline{S0}*\overline{S1}*S2*\overline{S3}*D7$$

$$P = \overline{S0}*S1*\overline{S2}*\overline{S3}*D7 + S0*S1*\overline{S2}*\overline{S3}*(P + OUT) + \overline{S0}*\overline{S1}*S2*\overline{S3}*(P + OUT) + S0*\overline{S1}*S2*\overline{S3}*(P + OUT) + \overline{S0}*S1*S2*\overline{S3}*(P + OUT) + S0*\overline{S1}*S2*\overline{S3}*(P + OUT) + \overline{S0}*\overline{S1}*S2*S3*(P + OUT) \quad (1)$$

$$P = S2*\overline{S3}*(P + OUT) + \overline{S0}*S1*\overline{S2}*\overline{S3}*(P + OUT) + \overline{S0}*\overline{S1}*S2*\overline{S3}*(P + OUT) + \overline{S0}*\overline{S1}*\overline{S2}*S3*D7 \quad (1)$$

$$P = \overline{S0}*S1*\overline{S2}*\overline{S3}*D7 + S2*\overline{S3}*(\overline{P}*OUT + P*\overline{OUT}) + S0*\overline{S1}*\overline{S2}*\overline{S3}*(\overline{P}*OUT + P*\overline{OUT}) + \overline{S0}*\overline{S1}*\overline{S2}*\overline{S3}*(\overline{P}*OUT + P*\overline{OUT})$$

$$\therefore\ P = \overline{S0}*S1*\overline{S2}*\overline{S3}*D7 + *S2*\overline{S3}*\overline{P}*OUT + S2*\overline{S3}*P*\overline{OUT} + S0*S1*\overline{S2}*\overline{S3}*\overline{P}*OUT + S0*S1*\overline{S2}*\overline{S3}*P*\overline{OUT} + \overline{S0}*\overline{S1}*\overline{S2}*\overline{S3}*\overline{P}*OUT + \overline{S0}*\overline{S1}*\overline{S2}*\overline{S3}*P*\overline{OUT}$$

```
PAL16R6                                                    PAL DESIGN SPECIFICATION
P7099                                          HARRY HUGHES, JERRY GREINER 12/08/82
PUART (PROGRAMMABLE UART WITH X1 CLOCK I/P AND AUTO FIFO INTERFACE)
MMI HAMPSHIRE, UNITED KINGDOM/ SUNNYVALE, CALIFORNIA
CLK /D1 /D2  /D3 /D4 /D5 /D6 /D7  OR   GND
/OC  GO /OUT /P  /S3 /S2 /S1 /S0 /COUT VCC

S0  := /GO* S0* S1* S2* S3
     +  GO* S0* S1* S2* S3* OR
     +     /S0*    /S2*/S3
     +     /S0*/S1* S2*/S3
     +     /S0* S1* S2*/S3
     + /GO*/S0* S1*/S2* S3
     +  GO*/S0* S1*/S2* S3* OR
     + /GO*                /OR

S1  := /GO* S0* S1* S2* S3
     +     /S0* S1*    /S3
     +      S0*/S1*    /S3
     +      S0*/S1*/S2
     + /GO*/S0* S1*/S2* S3
     + /GO*                /OR

S2  := /GO* S0* S1* S2* S3
     +      S0* S1*/S2*/S3
     +     /S0*     S2*/S3
     +         /S1* S2*/S3
     + /GO*/S0* S1*/S2* S3
     + /GO*                /OR

S3  := /GO* S0* S1* S2* S3
     +     /S0*/S1*/S2*/S3
     +      S0*/S1*/S2* S3
     + /GO*/S0* S1*/S2* S3
     + /GO*                /OR

OUT := /COUT
     + /S0* S1*/S2*/S3* D1
     +  S0* S1*/S2*/S3* D2
     + /S0*/S1* S2*/S3* D3
     +  S0*/S1* S2*/S3* D4
     + /S0* S1* S2*/S3* D5
     +  S0* S1* S2*/S3* D6
     + /S0*/S1*/S2*/S3* D7

P   := /S0* S1*/S2*/S3* D7
     +  S0* S1*/S2*/S3*/P* OUT
     +  S0* S1*/S2*/S3* P*/OUT
     + /S0*/S1*/S2*/S2*/P* OUT
     + /S0*/S1*/S2*/S3* P*/OUT
     +          S2*/S3*/P* OUT
     +          S2*/S3* P*/OUT

IF(VCC) COUT :=  S0* S1* S2* S3
              + /S0* S1*/S2* S3
              +  S0*/S1*/S2* S3* P
```

FUNCTION TABLE

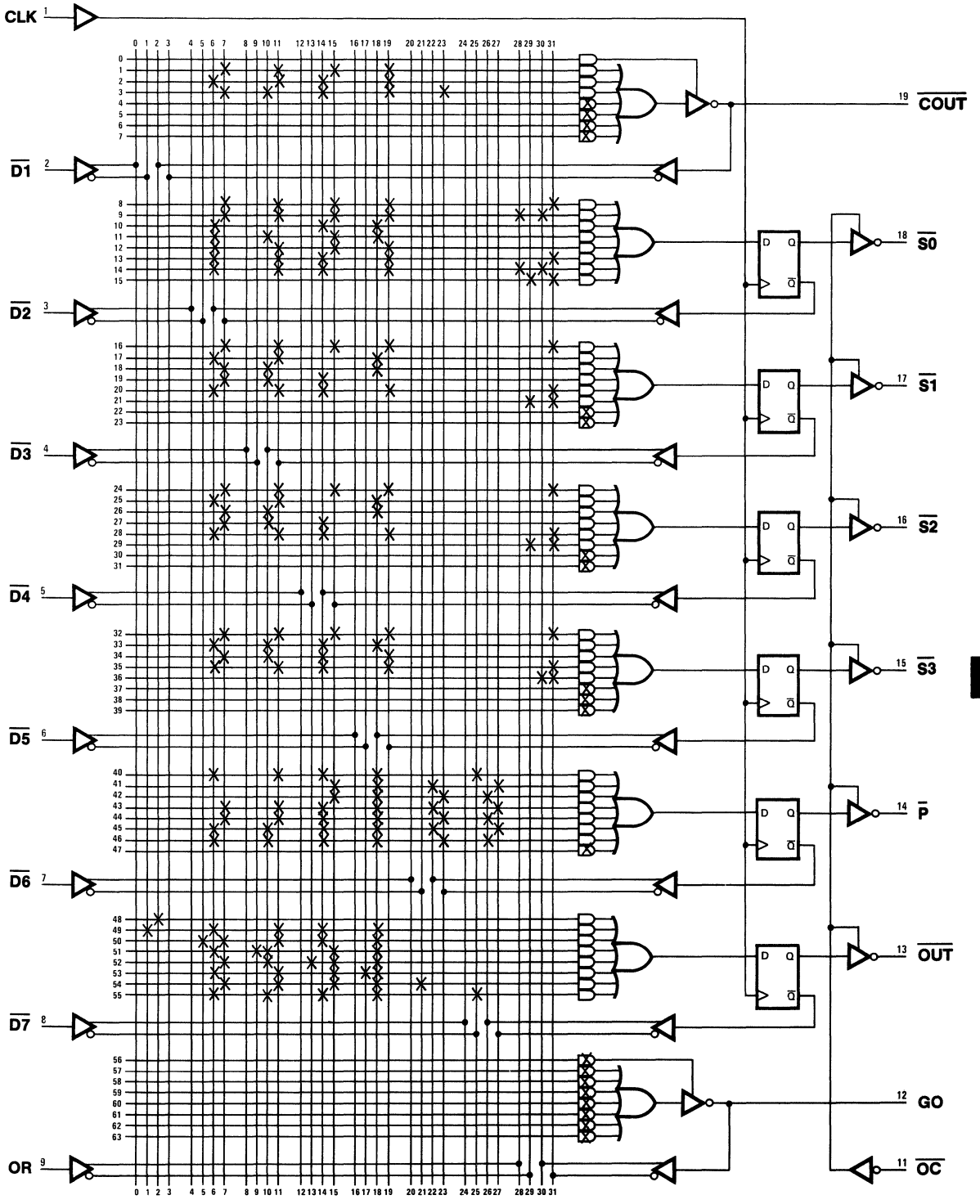
```
/OC CLK GO OR /D7 /D6 /D5 /D4 /D3 /D2 /D1 /S3 /S2 /S1 /S0 /P /OUT /COUT
;-------INPUTS--------   ---------OUTPUTS---------
;                ///////      ////
;                DDDDDDD      SSSS
;/OC CLK GO OR 7654321      3210 /P /OUT /COUT   COMMENTS
-------------------------------------------------------------------------------
  L   C  L  X  XXXXXXX      LLLL  X   H    L     RESET, IDLE, MARKING
  L   C  H  H  HLHLHLH      HHHL  X   H    X     START BIT OUT
  L   C  H  H  HLHLHLH      HHLH  X   H    X     PREPARE TO SHIFT D1
  L   C  H  H  HLHLHLH      HHLL  H   H    X     SHIFT OUT D1,D2 READY TO SHIFT
  L   C  H  H  HLHLHLH      HLHH  H   L    X     SHIFT OUT D2,D3 READY TO SHIFT
  L   C  H  H  HLHLHLH      HLHL  L   H    X     SHIFT OUT D3,D4 READY TO SHIFT
  L   C  H  H  HLHLHLH      HLLH  L   L    X     SHIFT OUT D4,D5 READY TO SHIFT
  L   C  H  H  HLHLHLH      HLLL  H   H    X     SHIFT OUT D5,D6 READY TO SHIFT
  L   C  H  H  HLHLHLH      HHHH  H   L    X     SHIFT OUT D6,D7 READY TO SHIFT
  L   C  H  H  HLHLHLH      LHHL  L   H    L     SHIFT OUT D7,PARITY BIT READY
  L   C  H  H  HLHLHLH      LHLH  X   H    L     SHIFT OUT PARITY (P)
  L   C  L  H  XXXXXXX      LLLL  X   H    L     STOP BIT OUT
  L   C  L  H  XXXXXXX      LLLL  X   H    L     IDLE (RESET) IF GO=LOW
  L   C  L  H  XXXXXXX      LLLL  X   H    L     IDLE
  L   C  H  H  LHLHLHL      HHHL  X   H    X     START BIT
  L   C  H  H  LHLHLHL      HHLH  X   H    X     PREPARE TO SHIFT D1
  L   C  H  H  LHLHLHL      HHLL  L   L    X     SHIFT OUT D1,D2 READY TO SHIFT
  L   C  H  H  LHLHLHL      HLHH  H   H    X     SHIFT OUT D2,D3 READY TO SHIFT
  L   C  H  H  LHLHLHL      HLHL  H   L    X     SHIFT OUT D3,D4 READY TO SHIFT
  L   C  H  H  LHLHLHL      HLLH  L   H    X     SHIFT OUT D4,D5 READT TO SHIFT
  L   C  H  H  LHLHLHL      HLLL  L   L    X     SHIFT OUT D5,D6 READY TO SHIFT
  L   C  H  H  LHLHLHL      HHHH  H   H    X     SHIFT OUT D6,D7 READY TO SHIFT
  L   C  H  H  LHLHLHL      LHHL  H   L    H     SHIFT OUT D7,PARITY BIT READY
  L   C  H  H  XXXXXXX      LHLH  X   H    L     SHIFT OUT PARITY(P)
  L   C  H  H  LLLLLLL      HHHL  X   H    X     STOP BIT OUT;START BIT,NO IDLE
  L   C  H  H  LLLLLLL      HHLH  X   H    X     PREPARE TO SHIFT D1
  L   C  H  H  LLLLLLL      HHLL  L   L    X     SHIFT OUT D1,D2 READY TO SHIFT
  L   C  L  L  XXXXXXX      LLLL  X   L    L     RESET
  H   X  X  X  XXXXXXX      ZZZZ  Z   Z    L     HI-Z
-------------------------------------------------------------------------------
```

DESCRIPTION

SEE TEXT.

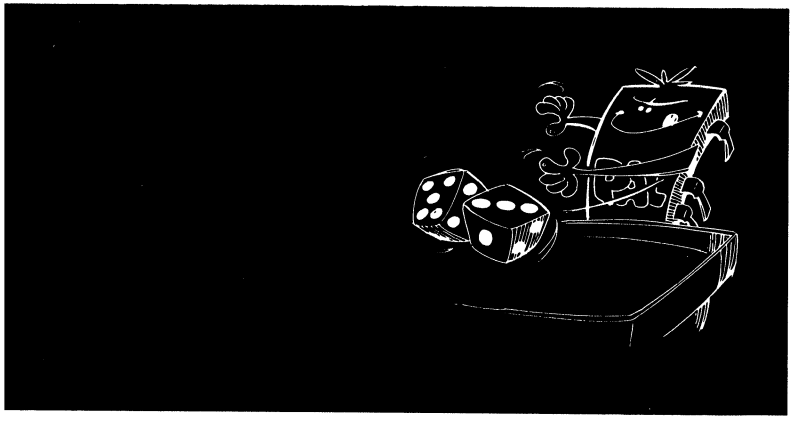
Programable UART

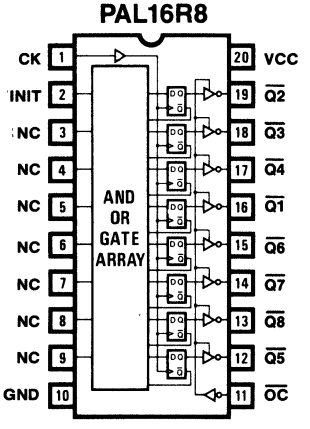
Logic Diagram PAL16R6



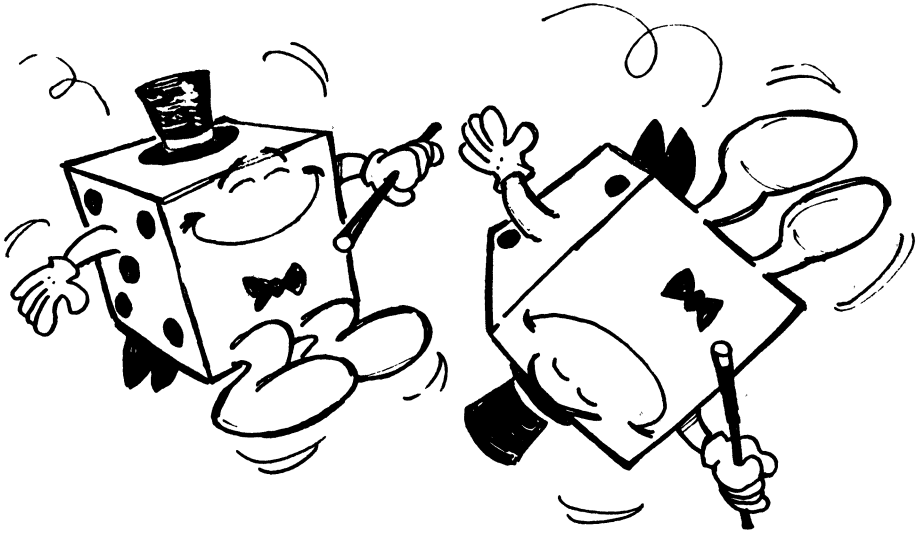

IMPORTANT
NOTES

Electronic Dice Game





CLASSIC DIE



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.

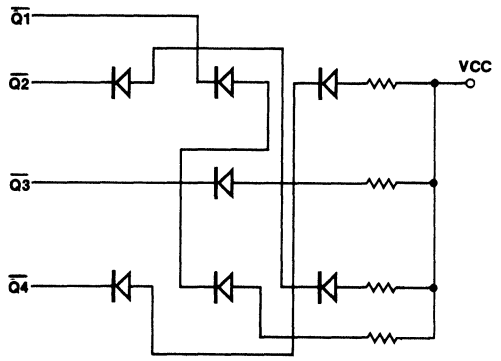


Figure 1

Looking at the Karnaugh 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.

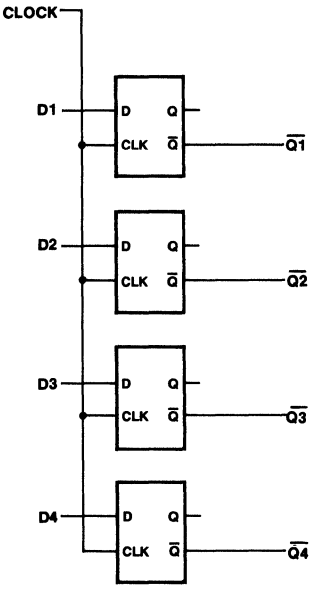


Figure 2

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;

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

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

	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

D_1 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

D_2 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

D_3 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

D_4 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

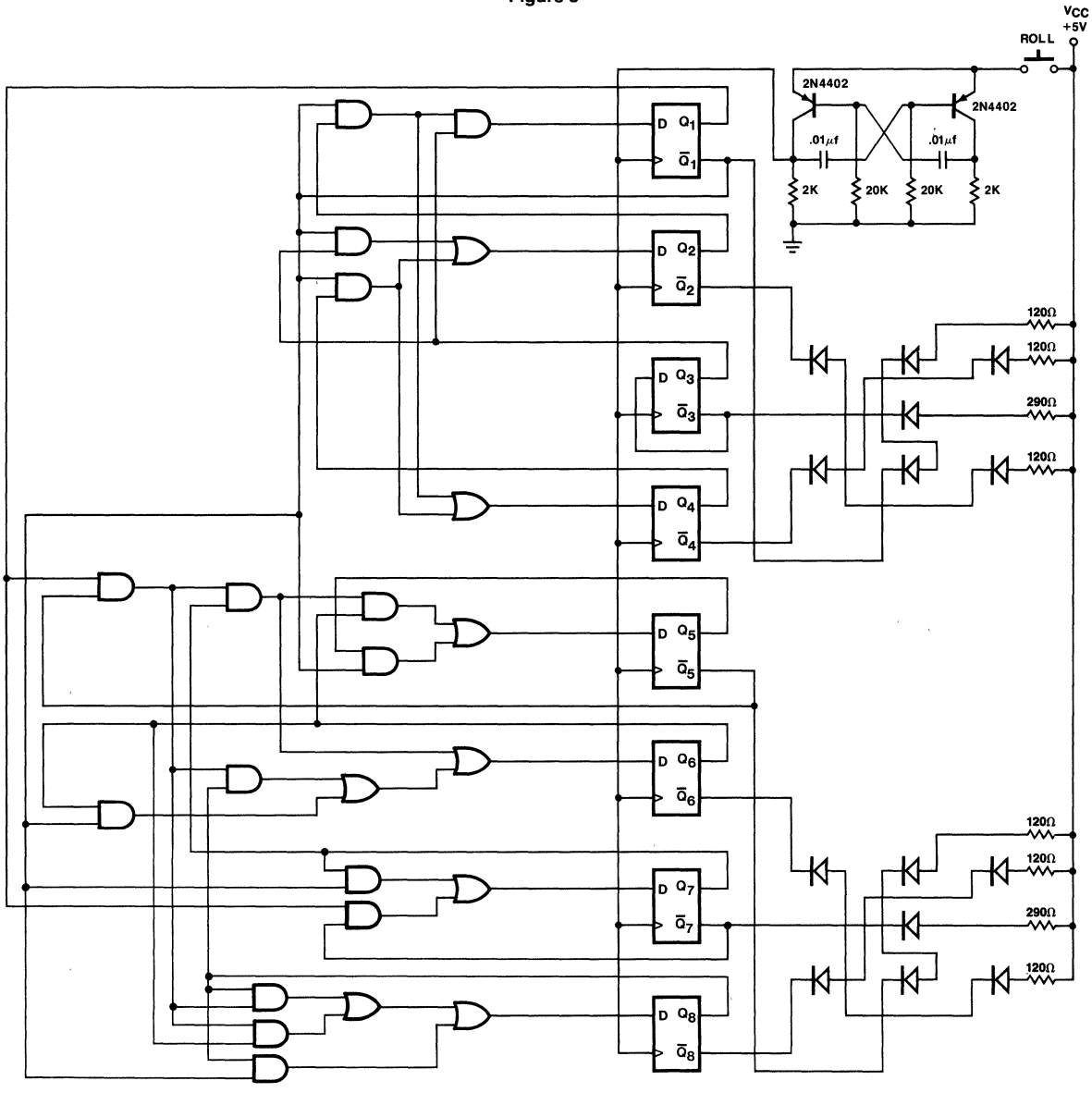


Figure 4

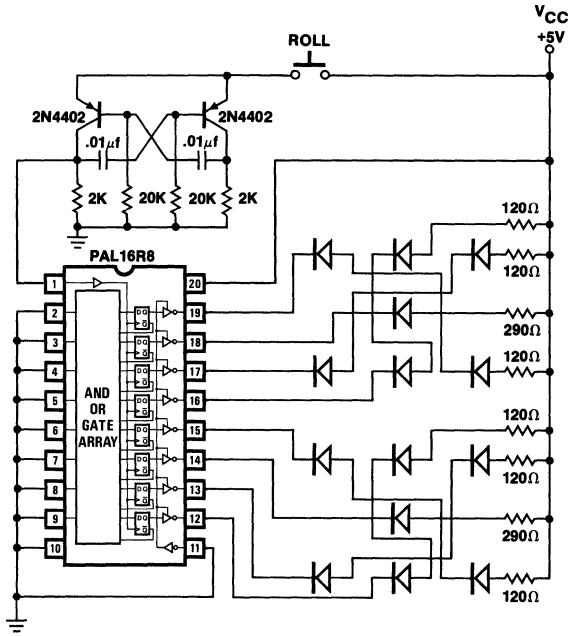
However, since this is a synchronous design the clocks of the two die are common. If the same equations are used for both die there will be only six different states. To get around this the first die is allowed to go through each of the six states incrementing with each clock. The second die is inhibited 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 PAL Devices

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. Note that the pinout is chosen to convenience PC board layout which is shown below.



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

```
PAL16R8                                          PAL DESIGN SPECIFICATION
EDG                                                 VETTER/COLI 07/06/81
ELECTRONIC DICE GAME
MMI SUNYVALE, CALIFORNIA
CK INIT NC NC NC NC NC NC NC GND
/OC /Q5 /Q8 /Q7 /Q6 /Q1 /Q4 /Q3 /Q2 VCC


Q1 :=  /Q1* Q2* Q3
    +   INIT

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

Q3 :=  /Q3*/INIT

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

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

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

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

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

FUNCTION TABLE

```
CK /OC INIT /Q1 /Q2 /Q3 /Q4 /Q5 /Q6 /Q7 /Q8

;-CONTROLS-     /OUTPUTS      COMMENTS
;C /            QQQQQQQQ    NUMBER DISPLAYED
;K OC  INIT     12345678    DIE 2  DIE 1
-------------------------------------------------
 C L    H       LHHHHHHH    INITIALIZE COUNTER
 C L    L       HHLHHHLH      1      1
 C L    L       HLHHHHLH      2      1
 C L    L       HHLLHHLH      3      1
 C L    L       HLHLHHLH      4      1
 C L    L       HLLLHHLH      5      1
 C L    L       LLHLHHLH      6      1
;
 C L    L       HHLHHLHH      1      2
 C L    L       HLHHHLHH      2      2
 C L    L       HHLLHLHH      3      2
 C L    L       HLHLHLHH      4      2
 C L    L       HLLLHLHH      5      2
 C L    L       LLHLHLHH      6      2
;
 C L    L       HHLHHHLL      1      3
 C L    L       HLHHHHLL      2      3
 C L    L       HHLLHHLL      3      3
 C L    L       HLHLHHLL      4      3
 C L    L       HLLLHHLL      5      3
 C L    L       LLHLHHLL      6      3
;
 C L    L       HHLHHLHL      1      4
 C L    L       HLHHHLHL      2      4
 C L    L       HHLLHLHL      3      4
 C L    L       HLHLHLHL      4      4
 C L    L       HLLLHLHL      5      4
 C L    L       LLHLHLHL      6      4
;
 C L    L       HHLHHLLL      1      5
 C L    L       HLHHHLLL      2      5
 C L    L       HHLLHLLL      3      5
 C L    L       HLHLHLLL      4      5
 C L    L       HLLLHLLL      5      5
 C L    L       LLHLHLLL      6      5
;
 C L    L       HHLHLLHL      1      6
 C L    L       HLHHLLHL      2      6
 C L    L       HHLLLLHL      3      6
 C L    L       HLHLLLHL      4      6
 C L    L       HLLLLLHL      5      6
 C L    L       LLHLLLHL      6      6
;
 C H    X       ZZZZZZZZ    TEST HI-Z
-------------------------------------------------
```

DESCRIPTION

THE DUAL MODULO-SIX COUNTER INCREMENTS ON THE RISING EDGE OF THE CLOCK (CK). THE THREE-STATE OUTPUTS ARE HIGH-Z WHEN THE OUTPUT CONTROL LINE (/OC) IS HIGH AND ENABLED WHEN THE OUTPUT CONTROL LINE (/OC) IS LOW.

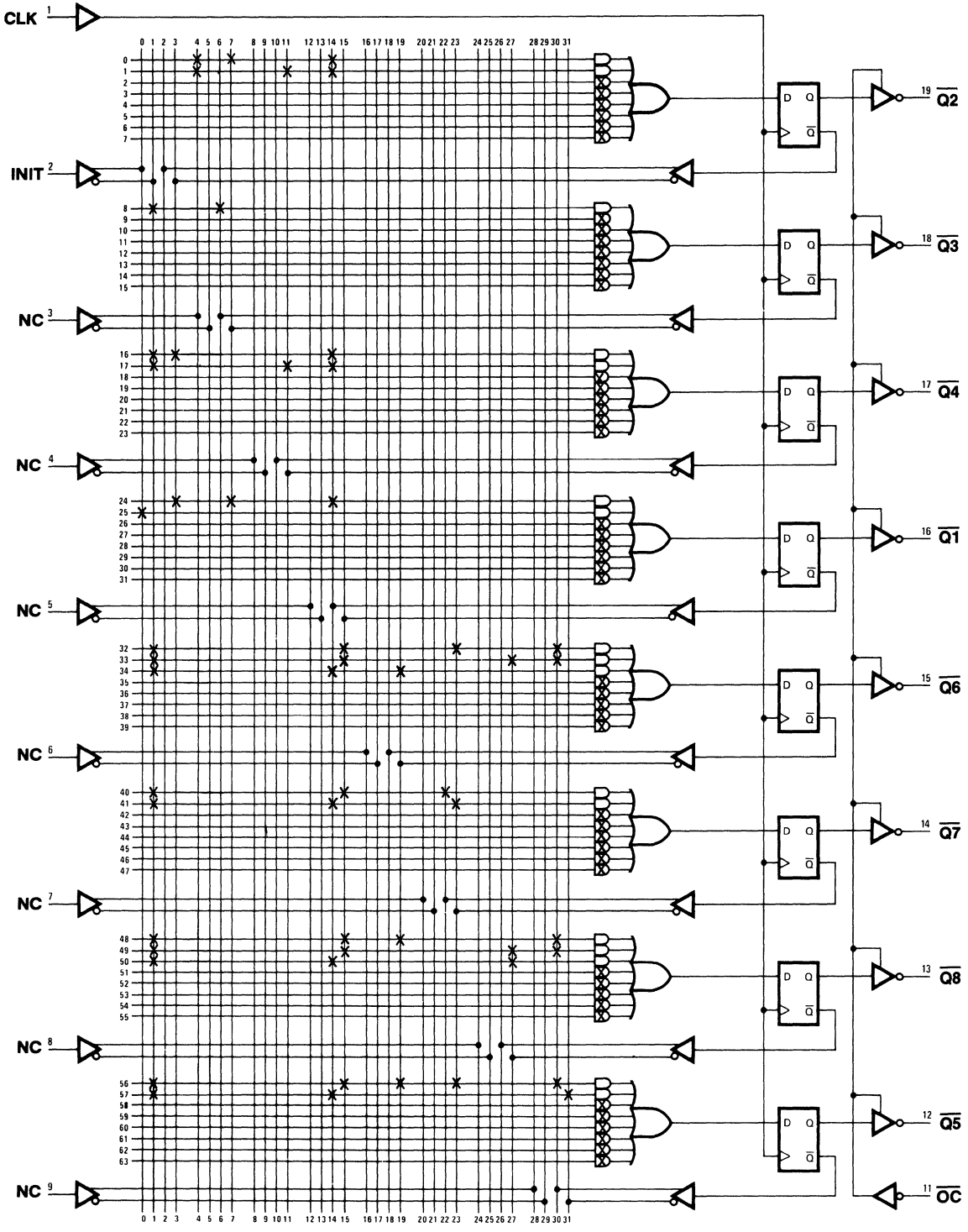
THE "INIT" LINE IS NEEDED TO INITIALIZE THE COUNTER SO THAT THE FUNCTION TABLE SIMULATION COULD BE PERFORMED AND THE PART COULD BE TESTED AT THE TIME OF FABRICATION. THIS LINE AS WELL AS ALL OTHER UNUSED INPUTS SHOULD BE TIED TO GND.

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

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

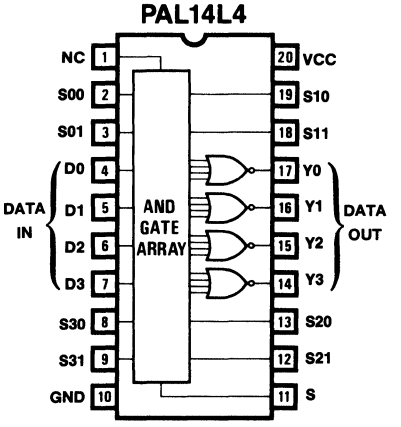
Electronic Dice Game

Logic Diagram PAL16R8




IMPORTANT
NOTES

4-Bit Serial Switch



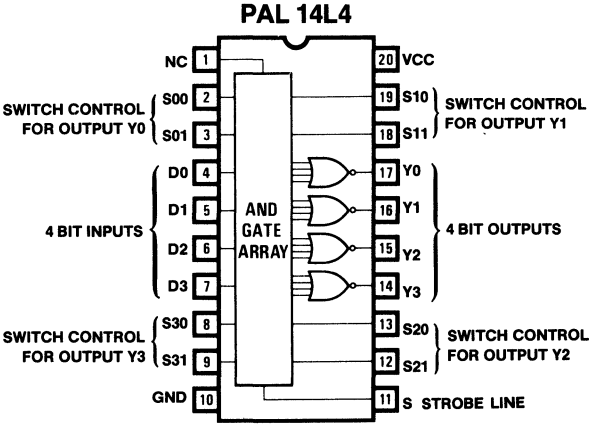
```
PAL14L4                                          PAL DESIGN SPECIFICATION
PD-100                                            N.SACHS/V.COLI 10/14/81
4-BIT SERIAL SWITCH
MMI SUNNYVALE, CALIFORNIA
NC S00 S01 D0 D1 D2 D3 S30 S31 GND
S  S21 S20 Y3 Y2 Y1 Y0 S11 S10 VCC


/Y3 =  /S*/D3* S31* S30        ;SWITCH D3 TO OUTPUT Y3 (S3=3)
    +  /S*/D2* S31*/S30        ;SWITCH D2 TO OUTPUT Y3 (S3=2)
    +  /S*/D1*/S31* S30        ;SWITCH D1 TO OUTPUT Y3 (S3=1)
    +  /S*/D0*/S31*/S30        ;SWITCH D0 TO OUTPUT Y3 (S3=0)

/Y2 =  /S*/D3* S21* S20        ;SWITCH D3 TO OUTPUT Y2 (S2=3)
    +  /S*/D2* S21*/S20        ;SWITCH D2 TO OUTPUT Y2 (S2=2)
    +  /S*/D1*/S21* S20        ;SWITCH D1 TO OUTPUT Y2 (S2=1)
    +  /S*/D0*/S21*/S20        ;SWITCH D0 TO OUTPUT Y2 (S2=0)

/Y1 =  /S*/D3* S11* S10        ;SWITCH D3 TO OUTPUT Y1 (S1=3)
    +  /S*/D2* S11*/S10        ;SWITCH D2 TO OUTPUT Y1 (S1=2)
    +  /S*/D1*/S11* S10        ;SWITCH D1 TO OUTPUT Y1 (S1=1)
    +  /S*/D0*/S11*/S10        ;SWITCH D0 TO OUTPUT Y1 (S1=0)

/Y0 =  /S*/D3* S01* S00        ;SWITCH D3 TO OUTPUT Y0 (S0=3)
    +  /S*/D2* S01*/S00        ;SWITCH D2 TO OUTPUT Y0 (S0=2)
    +  /S*/D1*/S01* S00        ;SWITCH D1 TO OUTPUT Y0 (S0=1)
    +  /S*/D0*/S01*/S00        ;SWITCH D0 TO OUTPUT Y0 (S0=0)
```

FUNCTION TABLE

```
S S31 S30 S21 S20 S11 S10 S01 S00 D3 D2 D1 D0 Y3 Y2 Y1 Y0

;   SWITCH CONTROL    INPUTS    OUTPUTS
;     S3 S2 S1 S0      DDDD      YYYY
;S    10 10 10 10      3210      3210        COMMENTS
-------------------------------------------------------------------------
 L    HH HL LH LL      LHLH      LHLH        D3=Y3, D2=Y2, D1=Y1, D0=Y0
 L    LL LH HL HH      LHLH      HLHL        D3=Y0, D2=Y1, D1=Y2, D0=Y3
 L    LL HL LH HH      LHLH      HHLL        D3=Y0, D2=Y2, D1=Y1, D0=Y3
 L    LL LH HL HH      LHLH      HLHL        D3=Y3, D2=Y1, D1=Y2, D0=Y0
 L    LL LL HH HH      HHLL      LLHH        D3=Y1,Y0  D0=Y3,Y2
 L    LL HH HH HH      HLLL      LHHH        D3=Y2,Y1,Y0  D0=Y3
 L    LL LL LL LL      HHHL      LLLL        D0=Y3,Y2,Y1,Y0
 L    LH LH LH LH      HHLH      LLLL        D1=Y3,Y2,Y1,Y0
 L    HL HL HL HL      HLHH      LLLL        D2=Y3,Y2,Y1,Y0
 L    HH HH HH HH      LHHH      LLLL        D3=Y3,Y2,Y1,Y0
 L    LL LL LL LL      LLLH      HHHH        D0=Y3,Y2,Y1,Y0
 L    LH LH LH LH      LLHL      HHHH        D1=Y3,Y2,Y1,Y0
 L    HL HL HL HL      LHLL      HHHH        D2=Y3,Y2,Y1,Y0
 L    HH HH HH HH      HLLL      HHHH        D3=Y3,Y2,Y1,Y0
 H    XX XX XX XX      LLLL      HHHH        TEST STROBE WITH D=L
 H    XX XX XX XX      HHHH      HHHH        TEST STROBE WITH D=H
-------------------------------------------------------------------------
```

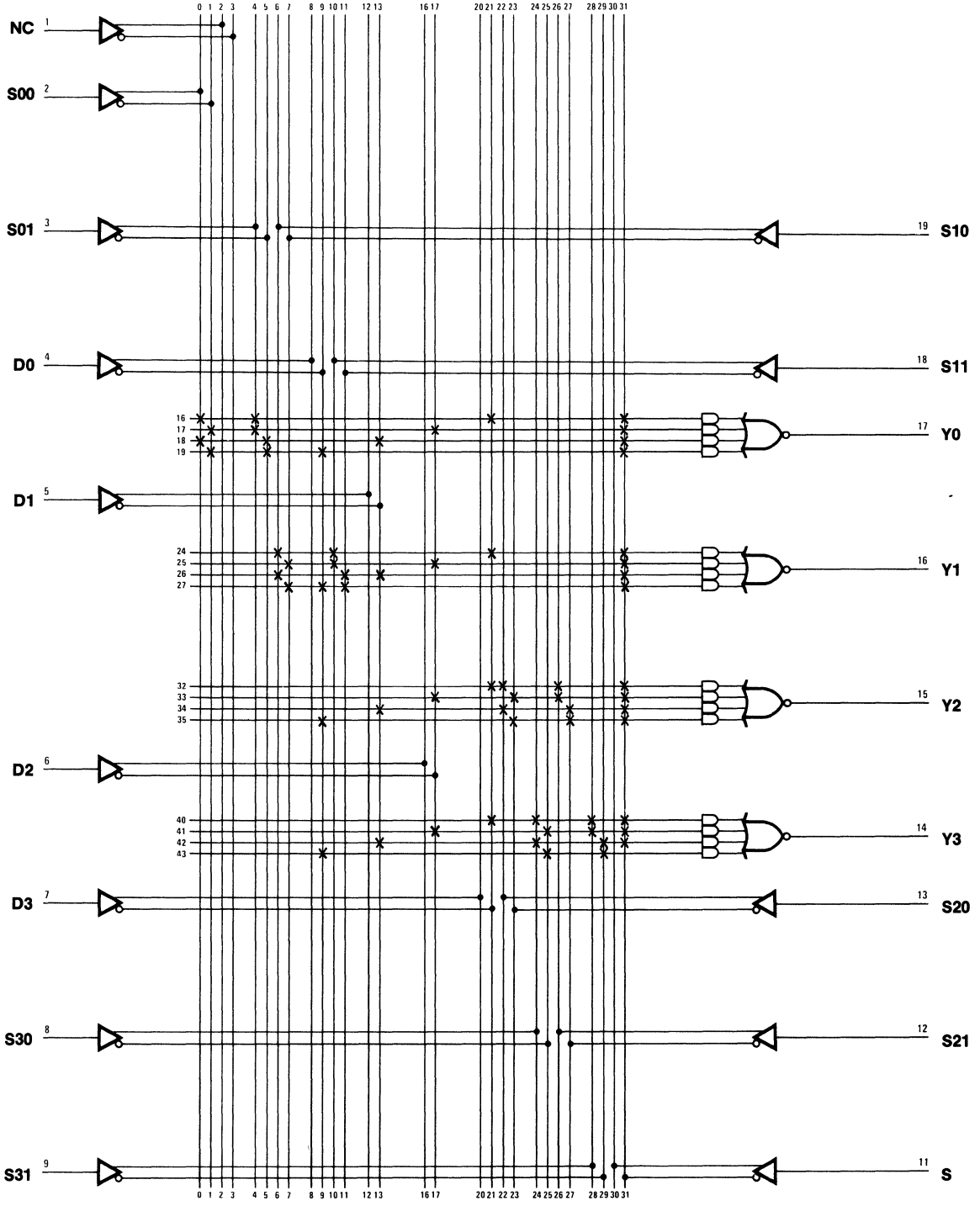
DESCRIPTION

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



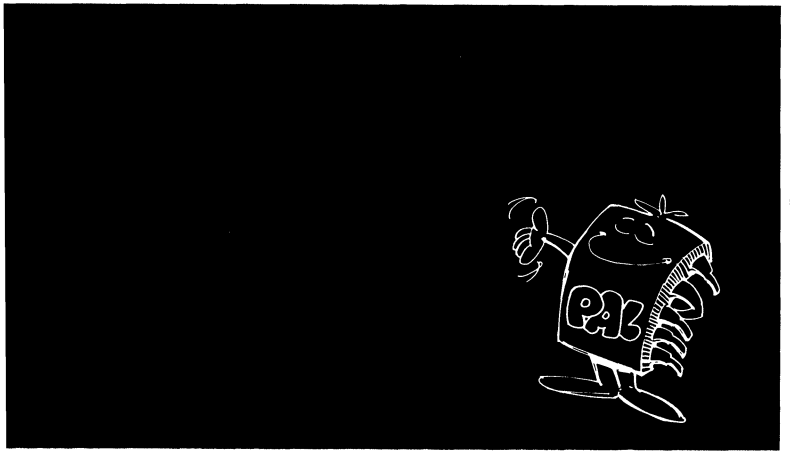
4-Bit Serial Switch

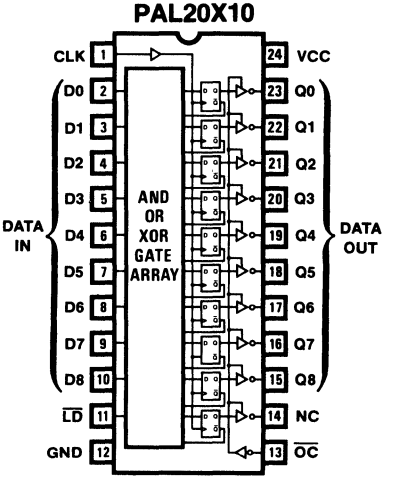
Logic Diagram PAL14L4




IMPORTANT
NOTES

9-Bit Register





```
PAL20X10                                          PAL DESIGN SPECIFICATION
PMSI405                                             BIRKNER/COLI 07/19/81
9-BIT REGISTER
MMI SUNNYVALE, CALIFORNIA
CLK D0 D1 D2 D3 D4 D5 D6 D7 D8 /LD GND
/OC NC Q8 Q7 Q6 Q5 Q4 Q3 Q2 Q1  Q0 VCC


/Q0 := /Q0*/LD      ;HOLD Q0
     + /D0* LD      ;LOAD D0

/Q1 := /Q1*/LD      ;HOLD Q1
     + /D1* LD      ;LOAD D1

/Q2 := /Q2*/LD      ;HOLD Q2
     + /D2* LD      ;LOAD D2

/Q3 := /Q3*/LD      ;HOLD Q3
     + /D3* LD      ;LOAD D3

/Q4 := /Q4*/LD      ;HOLD Q4
     + /D4* LD      ;LOAD D4

/Q5 := /Q5*/LD      ;HOLD Q5
     + /D5* LD      ;LOAD D5

/Q6 := /Q6*/LD      ;HOLD Q6
     + /D6* LD      ;LOAD D6

/Q7 := /Q7*/LD      ;HOLD Q7
     + /D7* LD      ;LOAD D7

/Q8 := /Q8*/LD      ;HOLD Q8
     + /D8* LD      ;LOAD D8
```

FUNCTION TABLE

/OC CLK /LD D8 D7 D6 D5 D4 D3 D2 D1 D0 Q8 Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0

;/ C /	DATA IN	DATA OUT	
;O L L	DDDDDDDDD	QQQQQQQQQ	
;C K D	876543210	876543210	COMMENT
L C L	LLLLLLLLL	LLLLLLLLL	LOAD ALL ZEROS
L C H	XXXXXXXXX	LLLLLLLLL	HOLD ALL ZEROS
L C L	HHHHHHHHH	HHHHHHHHH	LOAD ALL ONES
L C H	XXXXXXXXX	HHHHHHHHH	HOLD ALL ONES
L C L	LHLHLHLHL	LHLHLHLHL	LOAD EVEN CHECKERBOARD
L C H	XXXXXXXXX	LHLHLHLHL	HOLD EVEN CHECKERBOARD
L C L	HLHLHLHLH	HLHLHLHLH	LOAD ODD CHECKERBOARD
L C H	XXXXXXXXX	HLHLHLHLH	HOLD ODD CHECKERBOARD
H X X	XXXXXXXXX	ZZZZZZZZZ	TEST HI-Z

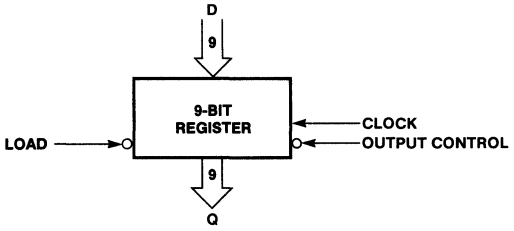
DESCRIPTION

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

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

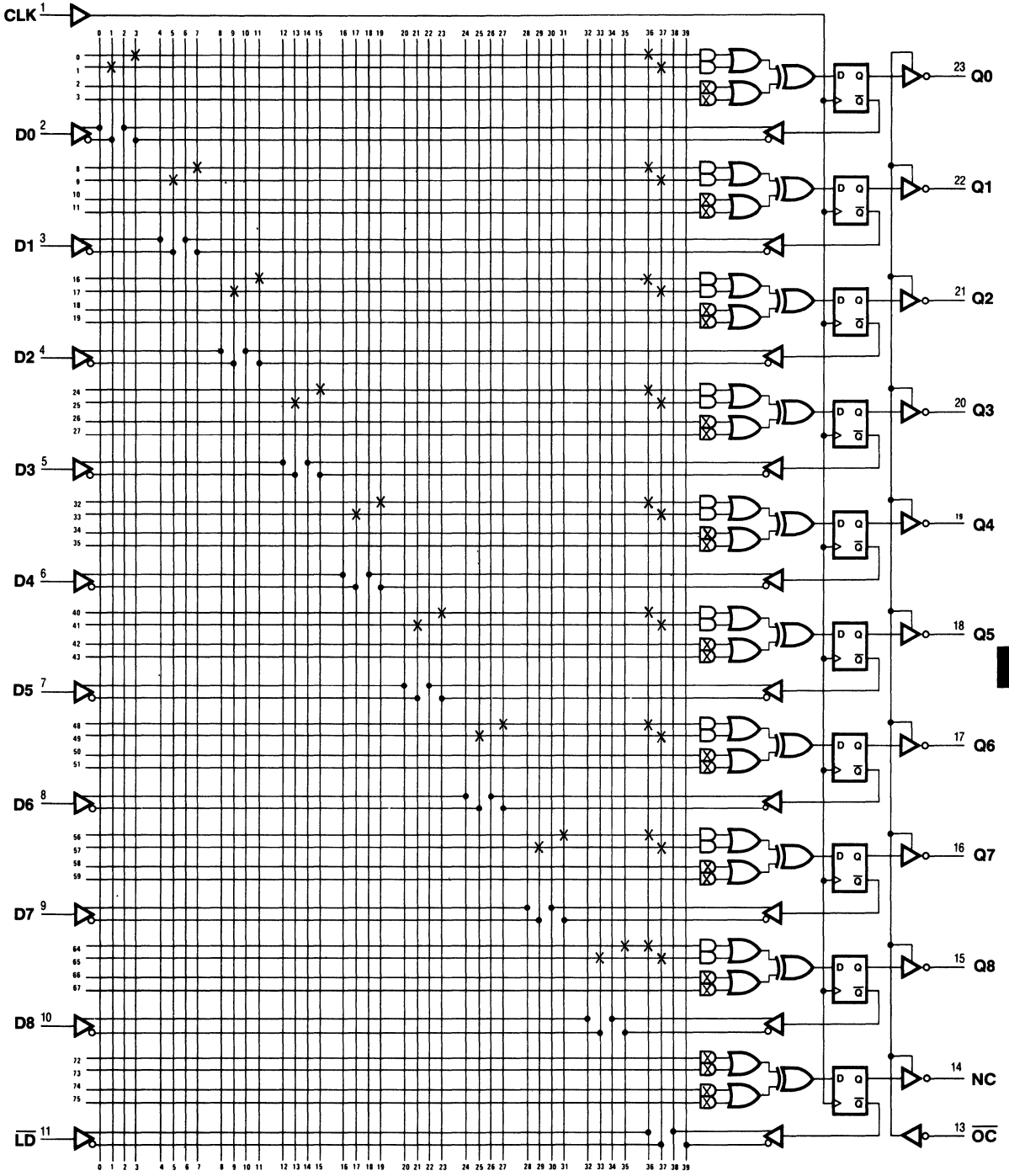
THESE OPERATIONS ARE EXERCISED IN THE FUNCTION TABLE AND SUMMARIZED IN OPERATIONS TABLE:

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



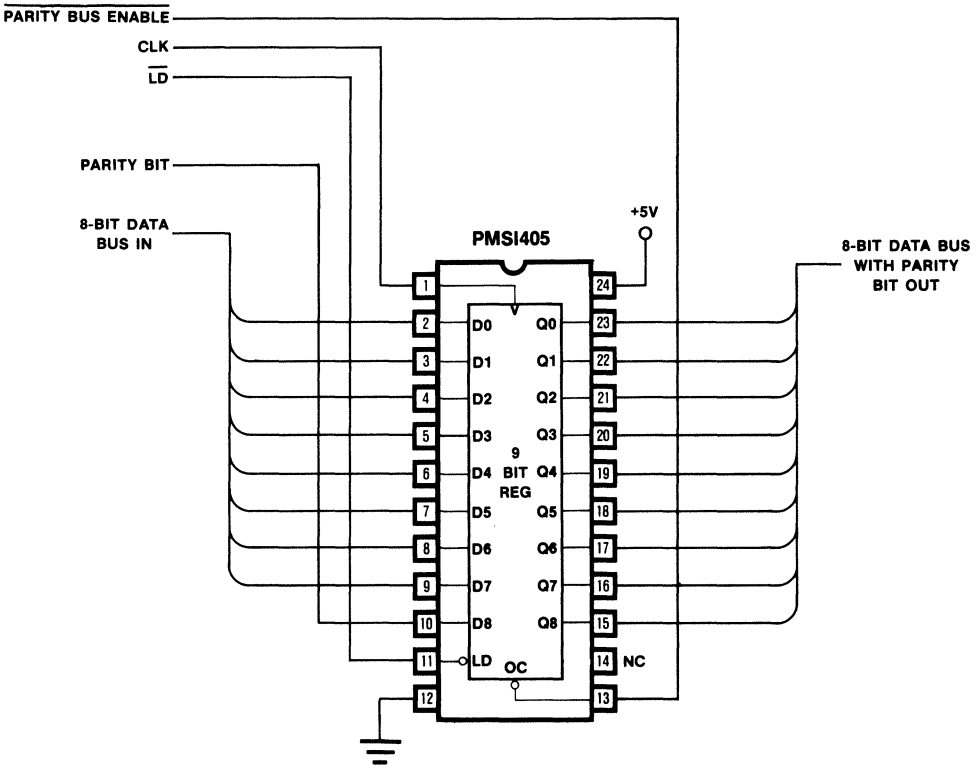
9-Bit Register

Logic Diagram PAL20X10

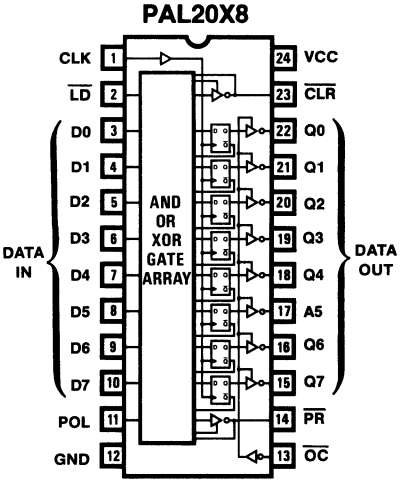


Application

Parity Bus Interface



Multifunction Octal Register



6

```
PAL20X8                                          PAL DESIGN SPECIFICATION
74LS380                                        BIRKNER/KAZMI/BLASCO 02/16/81
MULTIFUNCTION OCTAL REGISTER
MMI SUNNYVALE, CALIFORNIA
CLK /LD D0 D1 D2 D3 D4 D5 D6 D7  POL GND
/OC /PR Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0 /CLR VCC


/Q0 :=   CLR                                  ;CLEAR
     +  /CLR*/PR*/LD*/Q0                      ;HOLD
    :+: /CLR*/PR* LD* POL*/D0                 ;LOAD  D0 (TRUE)
     +  /CLR*/PR* LD*/POL* D0                 ;LOAD /D0 (COMP)

/Q1 :=   CLR                                  ;CLEAR
     +  /CLR*/PR*/LD*/Q1                      ;HOLD
    :+: /CLR*/PR* LD* POL*/D1                 ;LOAD  D1 (TRUE)
     +  /CLR*/PR* LD*/POL* D1                 ;LOAD /D1 (COMP)

/Q2 :=   CLR                                  ;CLEAR
     +  /CLR*/PR*/LD*/Q2                      ;HOLD
    :+: /CLR*/PR* LD* POL*/D2                 ;LOAD  D2 (TRUE)
     +  /CLR*/PR* LD*/POL* D2                 ;LOAD /D2 (COMP)

/Q3 :=   CLR                                  ;CLEAR
     +  /CLR*/PR*/LD*/Q3                      ;HOLD
    :+: /CLR*/PR* LD* POL*/D3                 ;LOAD  D3 (TRUE)
     +  /CLR*/PR* LD*/POL* D3                 ;LOAD /D3 (COMP)

/Q4 :=   CLR                                  ;CLEAR
     +  /CLR*/PR*/LD*/Q4                      ;HOLD
    :+: /CLR*/PR* LD* POL*/D4                 ;LOAD  D4 (TRUE)
     +  /CLR*/PR* LD*/POL* D4                 ;LOAD /D4 (COMP)

/Q5 :=   CLR                                  ;CLEAR
     +  /CLR*/PR*/LD*/Q5                      ;HOLD
    :+: /CLR*/PR* LD* POL*/D5                 ;LOAD  D5 (TRUE)
     +  /CLR*/PR* LD*/POL* D5                 ;LOAD /D5 (COMP)

/Q6 :=   CLR                                  ;CLEAR
     +  /CLR*/PR*/LD*/Q6                      ;HOLD
    :+: /CLR*/PR* LD* POL*/D6                 ;LOAD  D6 (TRUE)
     +  /CLR*/PR* LD*/POL* D6                 ;LOAD /D6 (COMP)

/Q7 :=   CLR                                  ;CLEAR
     +  /CLR*/PR*/LD*/Q7                      ;HOLD
    :+: /CLR*/PR* LD* POL*/D7                 ;LOAD  D7 (TRUE)
     +  /CLR*/PR* LD*/POL* D7                 ;LOAD /D7 (COMP)
```

```
FUNCTION TABLE

D7  D6  D5  D4  D3  D2  D1  D0  /CLR /PR /LD POL CLK /OC Q7  Q6  Q5  Q4  Q3  Q2  Q1  Q0

;   INPUTS              CONTROL             OUTPUTS     COMMENTS
; D7----D0   /CLR /PR /LD POL CLK /OC  Q7----Q0
-------------------------------------------------------------------------------
;CLEAR AND PRESET TESTS
  HHHHHHHH     L   L   L   H   C   L    LLLLLLLL   CLEAR (OVERRIDES PRESET/LOAD)
  LLLLLLLL     H   L   L   H   C   L    HHHHHHHH   PRESET (OVERRIDES LOAD)
  LLLLLLLL     L   L   L   L   C   L    LLLLLLLL   CLEAR (POL=L)
  HHHHHHHH     H   L   L   L   C   L    HHHHHHHH   PRESET (POL=L)
;LOAD DATA - WALKING ZEROES (TRUE DATA)
  HHHHHHHL     H   H   L   H   C   L    HHHHHHHL   LOAD HEX(FE)
  HHHHHHLH     H   H   L   H   C   L    HHHHHHLH   LOAD HEX(FD)
  HHHHHLHH     H   H   L   H   C   L    HHHHHLHH   LOAD HEX(FB)
  HHHHLHHH     H   H   L   H   C   L    HHHHLHHH   LOAD HEX(F7)
  HHHLHHHH     H   H   L   H   C   L    HHHLHHHH   LOAD HEX(EF)
  HHLHHHHH     H   H   L   H   C   L    HHLHHHHH   LOAD HEX(DF)
  HLHHHHHH     H   H   L   H   C   L    HLHHHHHH   LOAD HEX(BF)
  LHHHHHHH     H   H   L   H   C   L    LHHHHHHH   LOAD HEX(7F)
  HHHHHHHH     H   H   L   H   C   L    HHHHHHHH   LOAD HEX(FF)
;LOAD DATA - WALKING ONES (TRUE DATA)
  LLLLLLLH     H   H   L   H   C   L    LLLLLLLH   LOAD HEX(01)
  LLLLLLHL     H   H   L   H   C   L    LLLLLLHL   LOAD HEX(02)
  LLLLLHLL     H   H   L   H   C   L    LLLLLHLL   LOAD HEX(04)
  LLLLHLLL     H   H   L   H   C   L    LLLLHLLL   LOAD HEX(08)
  LLLHLLLL     H   H   L   H   C   L    LLLHLLLL   LOAD HEX(10)
  LLHLLLLL     H   H   L   H   C   L    LLHLLLLL   LOAD HEX(20)
  LHLLLLLL     H   H   L   H   C   L    LHLLLLLL   LOAD HEX(40)
  HLLLLLLL     H   H   L   H   C   L    HLLLLLLL   LOAD HEX(80)
  LLLLLLLL     H   H   L   H   C   L    LLLLLLLL   LOAD HEX(00)
;LOAD DATA - WALKING ONES (COMP DATA) WITH HOLD TESTS
  LLLLLLLL     H   H   H   L   C   L    LLLLLLLL   HOLD
  LLLLLLLL     H   H   L   L   C   L    HHHHHHHH   LOAD HEX(00) (COMP)
  LLLLLLLL     H   H   H   H   C   L    HHHHHHHH   HOLD
  LLLLLLLH     H   H   L   L   C   L    HHHHHHHL   LOAD HEX(01) (COMP)
  LLLLLLLL     H   H   H   L   C   L    HHHHHHHL   HOLD
  LLLLLLHL     H   H   L   L   C   L    HHHHHHLH   LOAD HEX(02) (COMP)
  HHHHHHHH     H   H   H   H   C   L    HHHHHHLH   HOLD
  LLLLLHLL     H   H   L   L   C   L    HHHHHLHH   LOAD HEX(04) (COMP)
  LLLLLLLL     H   H   H   L   C   L    HHHHHLHH   HOLD
  LLLLHLLL     H   H   L   L   C   L    HHHHLHHH   LOAD HEX(80) (COMP)
  HHHHHHHH     H   H   H   H   C   L    HHHHLHHH   HOLD
  LLLHLLLL     H   H   L   L   C   L    HHHLHHHH   LOAD HEX(10) (COMP)
  LLLLLLLL     H   H   H   L   C   L    HHHLHHHH   HOLD
  LLHLLLLL     H   H   L   L   C   L    HHLHHHHH   LOAD HEX(20) (COMP)
  HHHHHHHH     H   H   H   H   C   L    HHLHHHHH   HOLD
  LHLLLLLL     H   H   L   L   C   L    HLHHHHHH   LOAD HEX(40) (COMP)
  LLLLLLLL     H   H   H   L   C   L    HLHHHHHH   HOLD
  HLLLLLLL     H   H   L   L   C   L    LHHHHHHH   LOAD HEX(80) (COMP)
  HHHHHHHH     H   H   H   H   C   L    LHHHHHHH   HOLD
  LLLLLLLL     H   H   L   L   C   L    HHHHHHHH   LOAD HEX(00) (COMP)
  XXXXXXXX     X   X   X   X   X   H    ZZZZZZZZ   TEST HI-Z
-------------------------------------------------------------------------------
```

DESCRIPTION

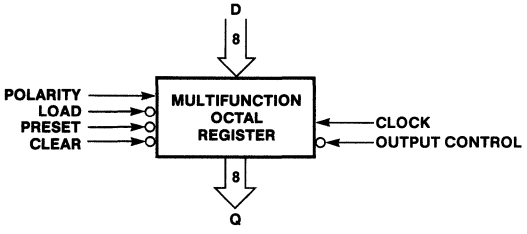
THIS IS AN 8-BIT SYNCHRONOUS REGISTER WITH PARALLEL LOAD, LOAD COMPLEMENT, PRESET, CLEAR, AND HOLD CAPABILITIES. FOUR CONTROL INPUTS (/LD,POL,/CLR,/PR) PROVIDE ONE OF FOUR OPERATIONS WHICH OCCUR SYNCHRONOUSLY WITH THE CLOCK (CLK).

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

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

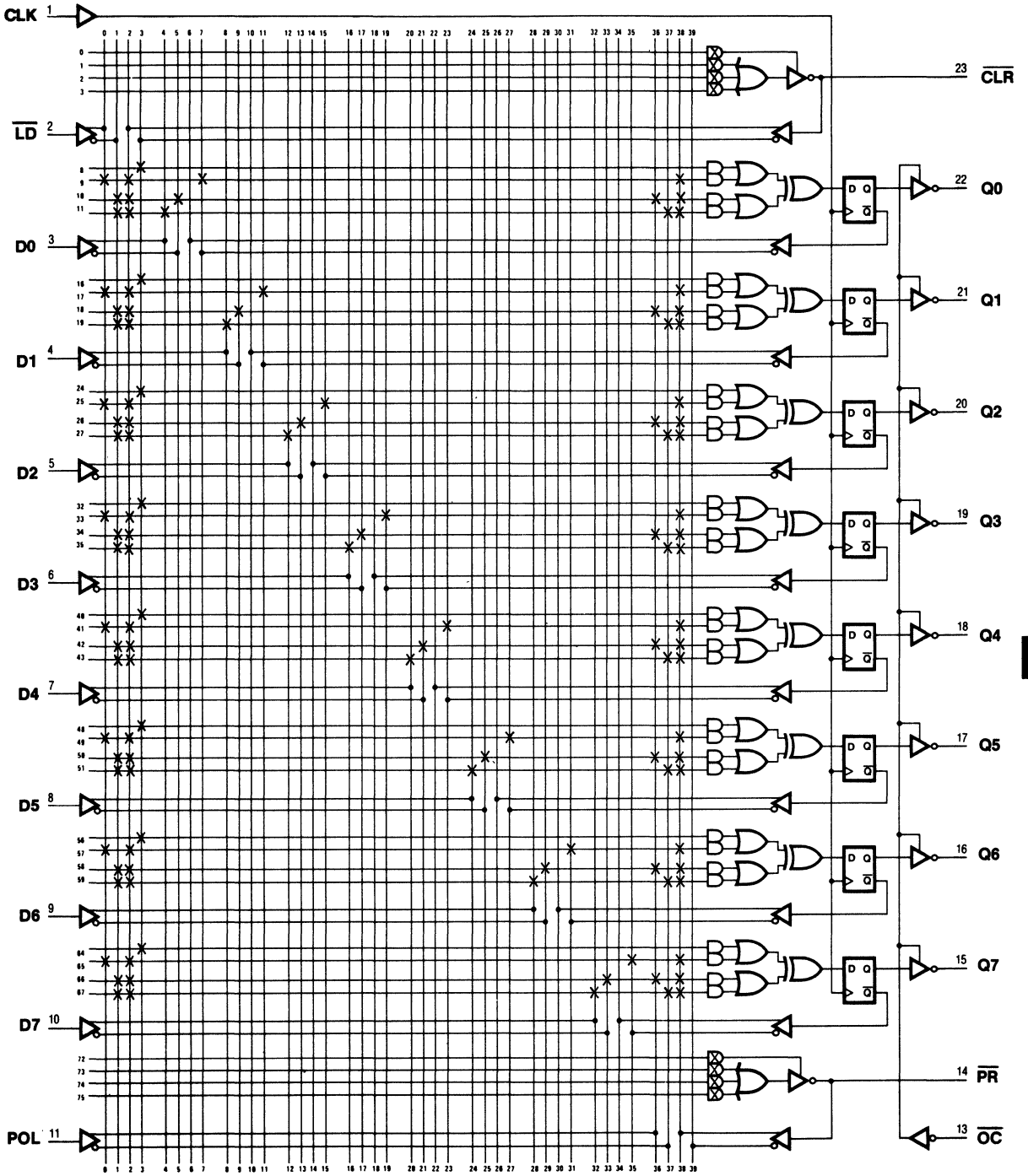
THESE OPERATIONS ARE EXERCISED IN THE FUNCTION TABLE AND SUMMARIZED IN THE OPERATIONS TABLE:

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



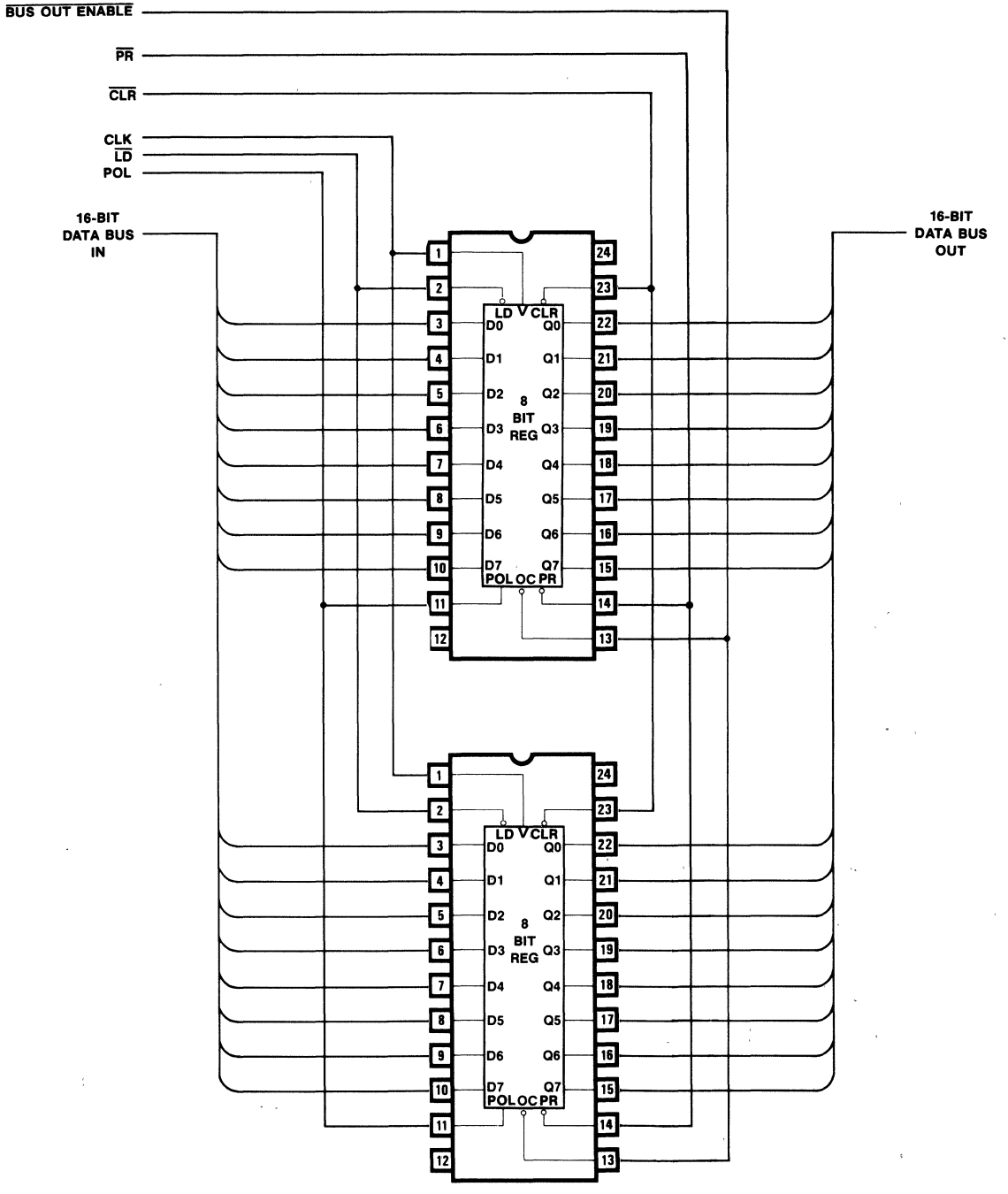
Multifunction Octal Register

Logic Diagram PAL20X8

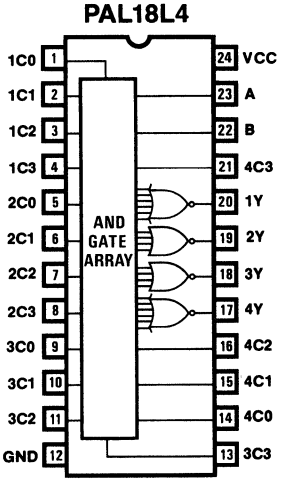


Application

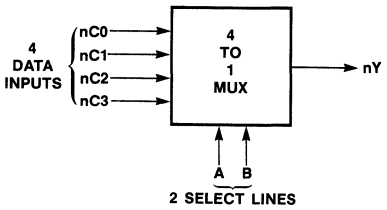
16-Bit Register



Quad 4:1 Mux


PAL18L4
1C0 1
1C1 2
1C2 3
1C3 4
2C0 5
2C1 6
2C2 7
2C3 8
3C0 9
3C1 10
3C2 11
GND 12
24 VCC
23 A
22 B
21 4C3
20 1Y
19 2Y
18 3Y
17 4Y
16 4C2
15 4C1
14 4C0
13 3C3
AND GATE ARRAY

```
PAL18L4                                          PAL DESIGN SPECIFICATION
74LS453                                        BIRKNER/KAZMI/BLASCO 03/10/81
QUAD 4:1 MULTIPLEXER
MMI SUNNYVALE, CALIFORNIA
1C0 1C1 1C2 1C3 2C0 2C1 2C2 2C3 3C0 3C1 3C2 GND
3C3 4C0 4C1 4C2  4Y  3Y  2Y  1Y 4C3  B   A  VCC


/1Y  =  /B*/A * /1C0                          ;SELECT INPUT 1C0
     +  /B* A * /1C1                          ;SELECT INPUT 1C1
     +   B*/A * /1C2                          ;SELECT INPUT 1C2
     +   B* A * /1C3                          ;SELECT INPUT 1C3

/2Y  =  /B*/A * /2C0                          ;SELECT INPUT 2C0
     +  /B* A * /2C1                          ;SELECT INPUT 2C1
     +   B*/A * /2C2                          ;SELECT INPUT 2C2
     +   B* A * /2C3                          ;SELECT INPUT 2C3

/3Y  =  /B*/A * /3C0                          ;SELECT INPUT 3C0
     +  /B* A * /3C1                          ;SELECT INPUT 3C1
     +   B*/A * /3C2                          ;SELECT INPUT 3C2
     +   B* A * /3C3                          ;SELECT INPUT 3C3

/4Y  =  /B*/A * /4C0                          ;SELECT INPUT 4C0
     +  /B* A * /4C1                          ;SELECT INPUT 4C1
     +   B*/A * /4C2                          ;SELECT INPUT 4C2
     +   B* A * /4C3                          ;SELECT INPUT 4C3
```



WHERE n = 1, 2, 3, or 4

FUNCTION TABLE

```
B A 1C0 1C1 1C2 1C3 2C0 2C1 2C2 2C3 3C0 3C1 3C2 3C3 4C0 4C1 4C2 4C3 1Y 2Y 3Y 4Y

; SEL  --------INPUTS--------  --OUTPUTS--      COMMENTS
;       1C    2C    3C    4C
; B A  0123  0123  0123  0123  1Y 2Y 3Y 4Y
-------------------------------------------------------------
  L L  LHHH  HHHH  HHHH  HHHH   L  H  H  H      1C0=0
  L L  HHHH  LHHH  HHHH  HHHH   H  L  H  H      2C0=0
  L L  HHHH  HHHH  LHHH  HHHH   H  H  L  H      3C0=0
  L L  HHHH  HHHH  HHHH  LHHH   H  H  H  L      4C0=0
  L L  HLLL  LLLL  LLLL  LLLL   H  L  L  L      1C0=1
  L L  LLLL  HLLL  LLLL  LLLL   L  H  L  L      2C0=1
  L L  LLLL  LLLL  HLLL  LLLL   L  L  H  L      3C0=1
  L L  LLLL  LLLL  LLLL  HLLL   L  L  L  H      4C0=1
  L L  HHHH  HHHH  HHHH  HHHH   H  H  H  H      TOGGLE LINES
  L H  HLHH  HHHH  HHHH  HHHH   L  H  H  H      1C1=0
  L H  HHHH  HLHH  HHHH  HHHH   H  L  H  H      2C1=0
  L H  HHHH  HHHH  HLHH  HHHH   H  H  L  H      3C1=0
  L H  HHHH  HHHH  HHHH  HLHH   H  H  H  L      4C1=0
  L H  LHLL  LLLL  LLLL  LLLL   H  L  L  L      1C1=1
  L H  LLLL  LHLL  LLLL  LLLL   L  H  L  L      2C1=1
  L H  LLLL  LLLL  LHLL  LLLL   L  L  H  L      3C1=1
  L H  LLLL  LLLL  LLLL  LHLL   L  L  L  H      4C1=1
  L H  HHHH  HHHH  HHHH  HHHH   H  H  H  H      TOGGLE LINES
  H L  HHLH  HHHH  HHHH  HHHH   L  H  H  H      1C2=0
  H L  HHHH  HHLH  HHHH  HHHH   H  L  H  H      2C2=0
  H L  HHHH  HHHH  HHLH  HHHH   H  H  L  H      3C2=0
  H L  HHHH  HHHH  HHHH  HHLH   H  H  H  L      4C2=0
  H L  LLHL  LLLL  LLLL  LLLL   H  L  L  L      1C2=1
  H L  LLLL  LLHL  LLLL  LLLL   L  H  L  L      2C2=1
  H L  LLLL  LLLL  LLHL  LLLL   L  L  H  L      3C2=1
  H L  LLLL  LLLL  LLLL  LLHL   L  L  L  H      4C2=1
  H L  HHHH  HHHH  HHHH  HHHH   H  H  H  H      TOGGLE LINES
  H H  HHHL  HHHH  HHHH  HHHH   L  H  H  H      1C3=0
  H H  HHHH  HHHL  HHHH  HHHH   H  L  H  H      2C3=0
  H H  HHHH  HHHH  HHHL  HHHH   H  H  L  H      3C3=0
  H H  HHHH  HHHH  HHHH  HHHL   H  H  H  L      4C3=0
  H H  LLLH  LLLL  LLLL  LLLL   H  L  L  L      1C3=1
  H H  LLLL  LLLH  LLLL  LLLL   L  H  L  L      2C3=1
  H H  LLLL  LLLL  LLLH  LLLL   L  L  H  L      3C3=1
  H H  LLLL  LLLL  LLLL  LLLH   L  L  L  H      4C3=1
  H H  HHHH  HHHH  HHHH  HHHH   H  H  H  H      TOGGLE LINES
 -------------------------------------------------------------
```

DESCRIPTION

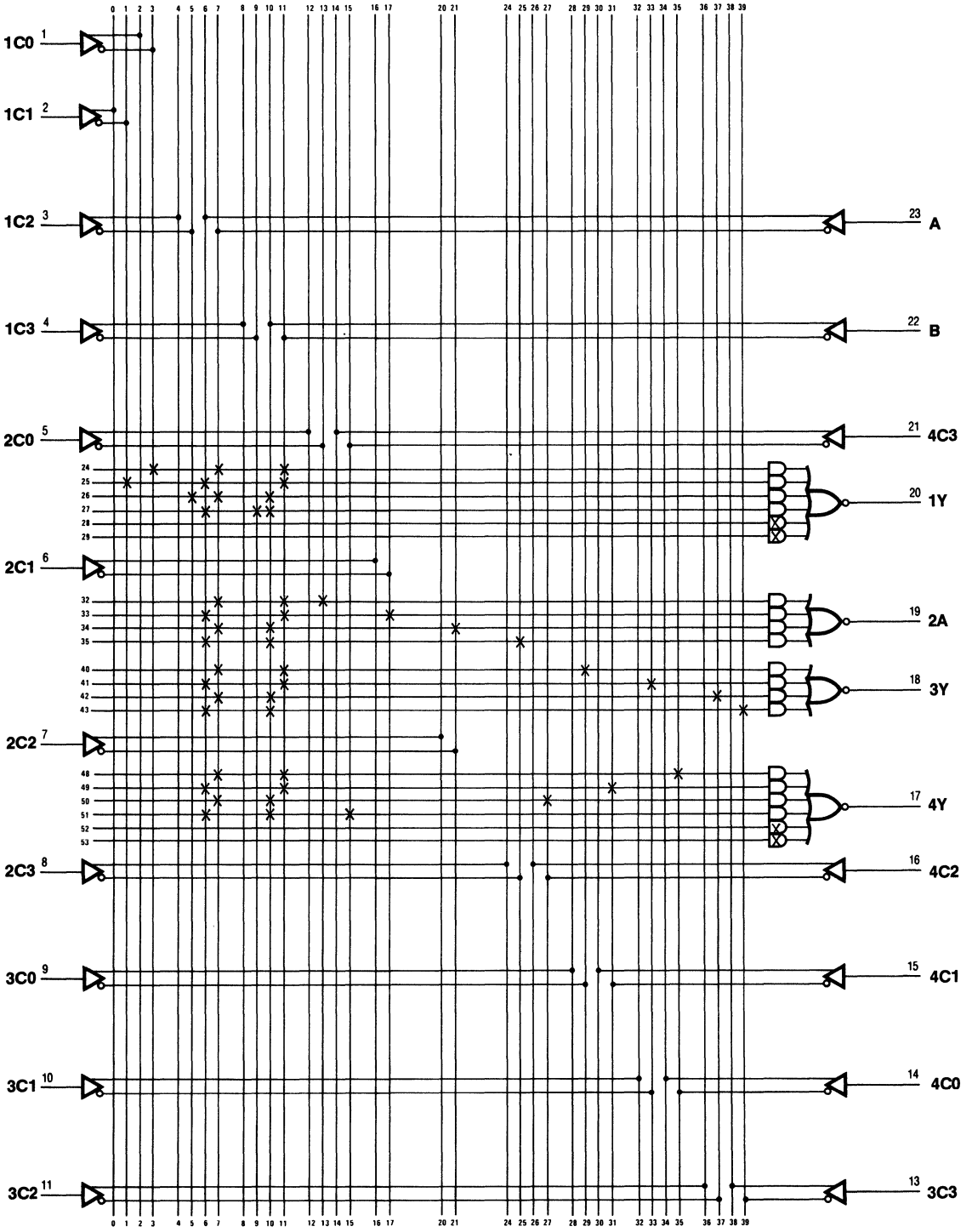
THIS IS AN EXAMPLE OF A QUAD 4-TO-1 MULTIPLEXER USING A PAL18L4. SELECT LINES A,B ARE ENCODED IN BINARY, WITH A REPRESENTING THE LSB.

OPERATIONS TABLE:

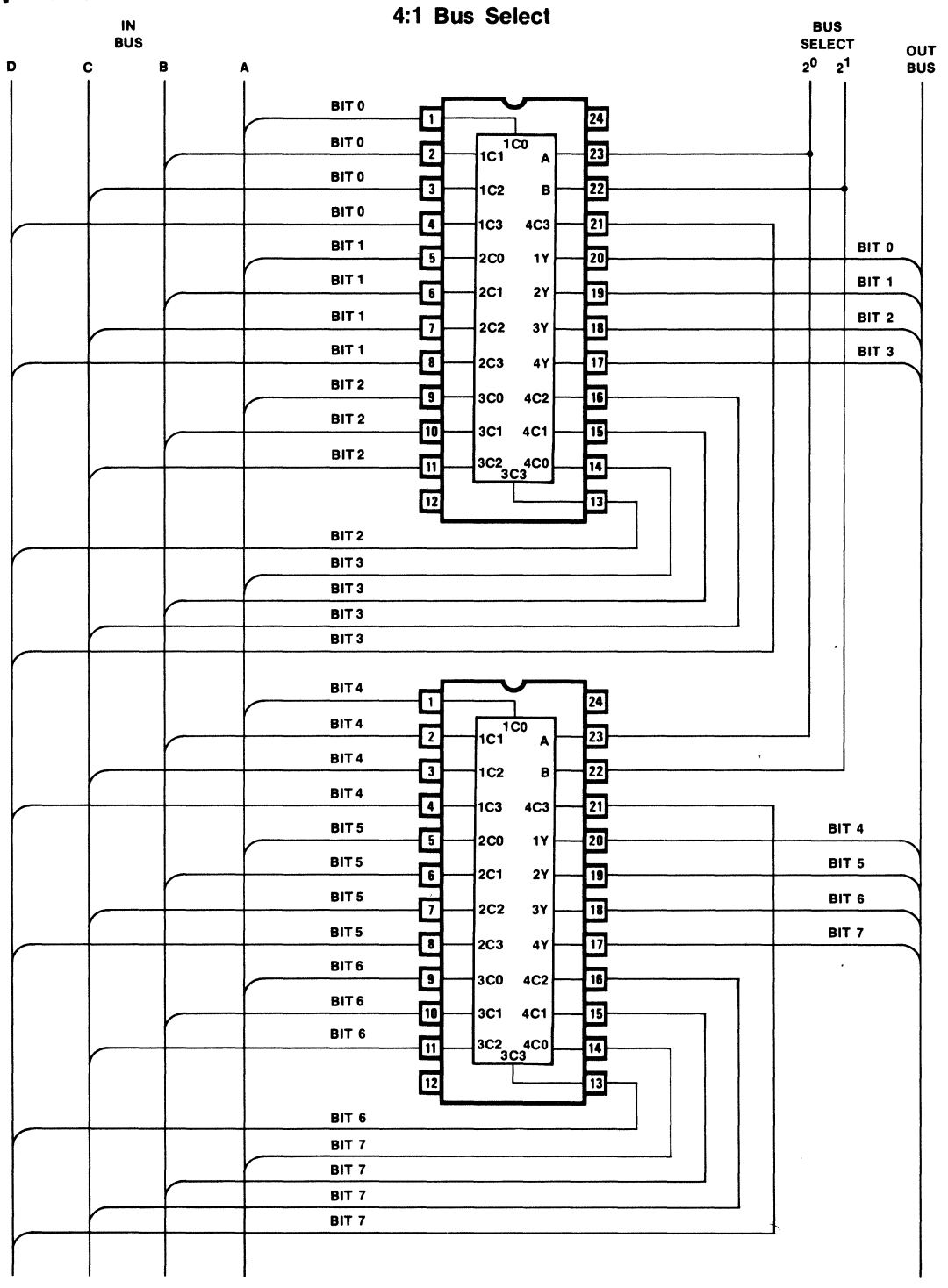
INPUT SELECT		OUTPUTS
B	A	Y
L	L	C0
L	H	C1
H	L	C2
H	H	C3

Quad 4:1 Multiplexer

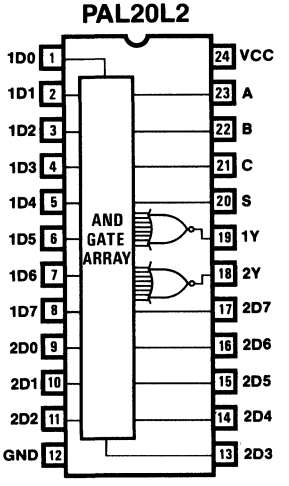
Logic Diagram PAL18L4



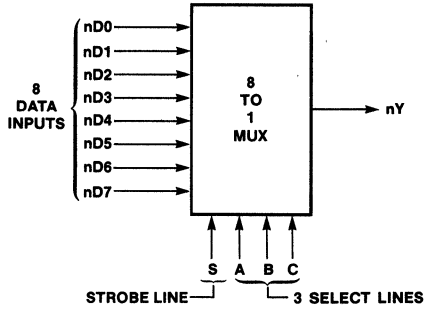
Application



Dual 8:1 Mux


PAL20L2
1D0 1
1D1 2
1D2 3
1D3 4
1D4 5
1D5 6
1D6 7
1D7 8
2D0 9
2D1 10
2D2 11
GND 12
24 VCC
23 A
22 B
21 C
20 S
19 1Y
18 2Y
17 2D7
16 2D6
15 2D5
14 2D4
13 2D3
AND GATE ARRAY

```
PAL20L2                                          PAL DESIGN SPECIFICATION
74LS451                                        BIRKNER/KAZMI/BLASCO 03/10/81
DUAL 8:1 MULTIPLEXER
MMI SUNNYVALE, CALIFORNIA
1D0 1D1 1D2 1D3 1D4 1D5 1D6 1D7 2D0 2D1 2D2 GND
2D3 2D4 2D5 2D6 2D7  2Y  1Y  S   C   B   A  VCC


/1Y  =  /S*/C*/B*/A * /1D0                    ;SELECT INPUT 1D0
     +  /S*/C*/B* A * /1D1                    ;SELECT INPUT 1D1
     +  /S*/C* B*/A * /1D2                    ;SELECT INPUT 1D2
     +  /S*/C* B* A * /1D3                    ;SELECT INPUT 1D3
     +  /S* C*/B*/A * /1D4                    ;SELECT INPUT 1D4
     +  /S* C*/B* A * /1D5                    ;SELECT INPUT 1D5
     +  /S* C* B*/A * /1D6                    ;SELECT INPUT 1D6
     +  /S* C* B* A * /1D7                    ;SELECT INPUT 1D7

/2Y  =  /S*/C*/B*/A * /2D0                    ;SELECT INPUT 2D0
     +  /S*/C*/B* A * /2D1                    ;SELECT INPUT 2D1
     +  /S*/C* B*/A * /2D2                    ;SELECT INPUT 2D2
     +  /S*/C* B* A * /2D3                    ;SELECT INPUT 2D3
     +  /S* C*/B*/A * /2D4                    ;SELECT INPUT 2D4
     +  /S* C*/B* A * /2D5                    ;SELECT INPUT 2D5
     +  /S* C* B*/A * /2D6                    ;SELECT INPUT 2D6
     +  /S* C* B* A * /2D7                    ;SELECT INPUT 2D7
```



FUNCTION TABLE

C B A 1D0 1D1 1D2 1D3 1D4 1D5 1D6 1D7 2D0 2D1 2D2 2D3 2D4 2D5 2D6 2D7 S 1Y 2Y

;SELECT			INPUTS 1D-	INPUTS 2D-		OUTPUTS		COMMENTS
;								
; C	B	A	01234567	01234567	S	1Y	2Y	
L	L	L	LHHHHHHH	LHHHHHHH	L	L	L	1D0=0 2D0=0
L	L	L	HLLLLLLL	LHHHHHHH	L	H	L	1D0=1 2D0=0
L	L	L	LHHHHHHH	HLLLLLLL	L	L	H	1D0=0 2D0=1
L	L	L	HLLLLLLL	HLLLLLLL	L	H	H	1D0=1 2D0=1
L	L	L	HHHHHHHH	HHHHHHHH	L	H	H	TOGGLE OTHER LINES
L	L	H	HLHHHHHH	HLHHHHHH	L	L	L	1D1=0 2D1=0
L	L	H	LHLLLLLL	HLHHHHHH	L	H	L	1D1=1 2D1=0
L	L	H	HLHHHHHH	LHLLLLLL	L	L	H	1D1=0 2D1=1
L	L	H	LHLLLLLL	LHLLLLLL	L	H	H	1D1=1 2D1=1
L	L	H	HHHHHHHH	HHHHHHHH	L	H	H	TOGGLE OTHER LINES
L	H	L	HHLHHHHH	HHLHHHHH	L	L	L	1D2=0 2D2=0
L	H	L	LLHLLLLL	HHLHHHHH	L	H	L	1D2=1 2D2=0
L	H	L	HHLHHHHH	LLHLLLLL	L	L	H	1D2=0 2D2=1
L	H	L	LLHLLLLL	LLHLLLLL	L	H	H	1D2=1 2D2=1
L	H	L	HHHHHHHH	HHHHHHHH	L	H	H	TOGGLE OTHER LINES
L	H	H	HHHLHHHH	HHHLHHHH	L	L	L	1D3=0 2D3=0
L	H	H	LLLHLLLL	HHHLHHHH	L	H	L	1D3=1 2D3=0
L	H	H	HHHLHHHH	LLLHLLLL	L	L	H	1D3=0 2D3=1
L	H	H	LLLHLLLL	LLLHLLLL	L	H	H	1D3=1 2D3=1
L	H	H	HHHHHHHH	HHHHHHHH	L	H	H	TOGGLE OTHER LINES
H	L	L	HHHHLHHH	HHHHLHHH	L	L	L	1D4=0 2D4=0
H	L	L	LLLLHLLL	HHHHLHHH	L	H	L	1D4=1 2D4=0
H	L	L	HHHHLHHH	LLLLHLLL	L	L	H	1D4=0 2D4=1
H	L	L	LLLLHLLL	LLLLHLLL	L	H	H	1D4=1 2D4=1
H	L	L	HHHHHHHH	HHHHHHHH	L	H	H	TOGGLE OTHER LINES
H	L	H	HHHHHLHH	HHHHHLHH	L	L	L	1D5=0 2D5=0
H	L	H	LLLLLHLL	HHHHHLHH	L	H	L	1D5=1 2D5=0
H	L	H	HHHHHLHH	LLLLLHLL	L	L	H	1D5=0 2D5=1
H	L	H	LLLLLHLL	LLLLLHLL	L	H	H	1D5=1 2D5=1
H	L	H	HHHHHHHH	HHHHHHHH	L	H	H	TOGGLE OTHER LINES
H	H	L	HHHHHHLH	HHHHHHLH	L	L	L	1D6=0 2D6=0
H	H	L	LLLLLLHL	HHHHHHLH	L	H	L	1D6=1 2D6=0
H	H	L	HHHHHHLH	LLLLLLHL	L	L	H	1D6=0 2D6=1
H	H	L	LLLLLLHL	LLLLLLHL	L	H	H	1D6=1 2D6=1
H	H	L	HHHHHHHH	HHHHHHHH	L	H	H	TOGGLE OTHER LINES
H	H	H	HHHHHHHL	HHHHHHHL	L	L	L	1D7=0 2D7=0
H	H	H	LLLLLLLH	HHHHHHHL	L	H	L	1D7=1 2D7=0
H	H	H	HHHHHHHL	LLLLLLLH	L	L	H	1D7=0 2D7=1
H	H	H	LLLLLLLH	LLLLLLLH	L	H	H	1D7=1 2D7=1
H	H	H	HHHHHHHH	HHHHHHHH	L	H	H	TOGGLE OTHER LINES
X	X	X	LLLLLLLL	LLLLLLLL	H	H	H	STROBE TEST 0
X	X	X	HHHHHHHH	HHHHHHHH	H	H	H	STROBE TEST 1

DESCRIPTION

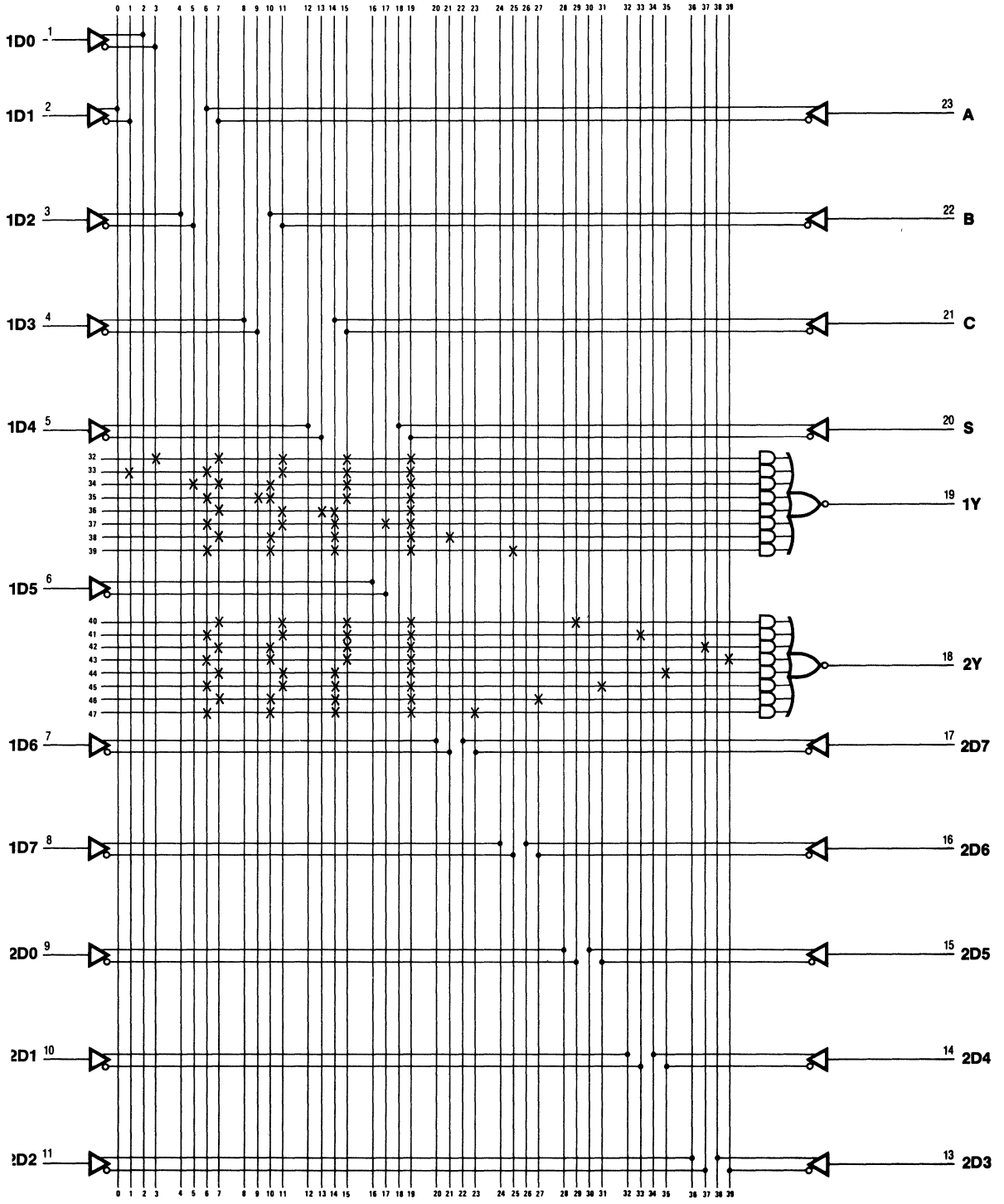
THIS IS AN EXAMPLE OF A DUAL 8-TO-1 MULTIPLEXER USING A PAL20L2. A STROBE LINE (S) IS PROVIDED TO GATE THE OUTPUTS OFF (Y=H) WHEN THE STROBE INPUT IS HIGH. THE SELECT LINES A,B,C ARE ENCODED IN BINARY, WITH A REPRESENTING THE LSB.

OPERATIONS TABLE:

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

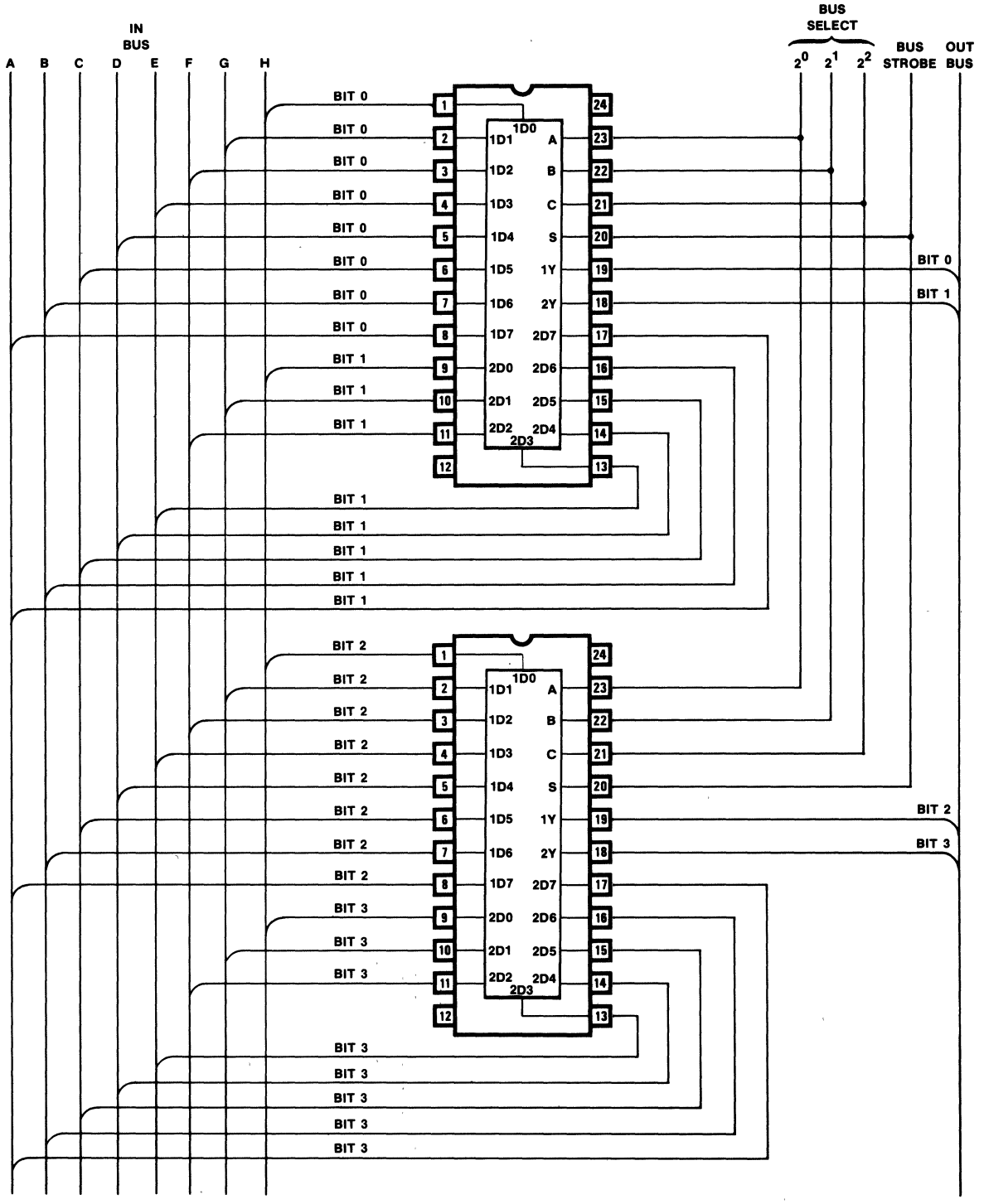
Dual 8:1 Multiplexer

Logic Diagram PAL20L2

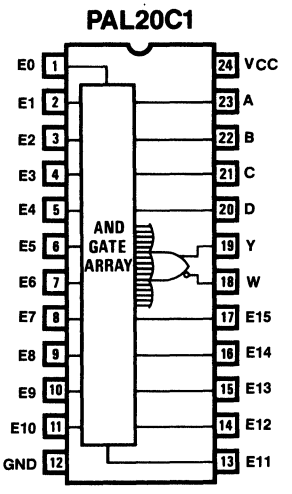


Application

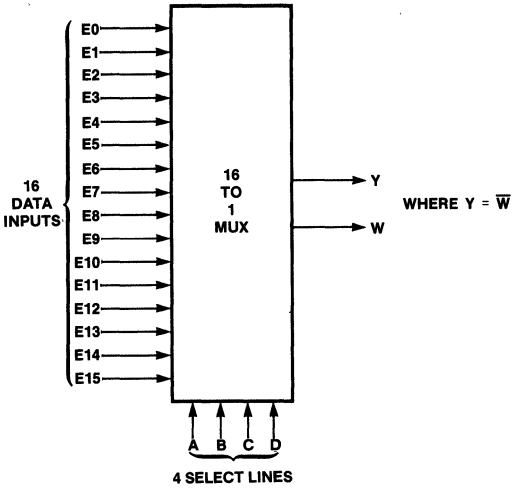
8:1 Bus Select



16:1 Mux


PAL20C1
E0 1
E1 2
E2 3
E3 4
E4 5
E5 6
E6 7
E7 8
E8 9
E9 10
E10 11
GND 12
24 VCC
23 A
22 B
21 C
20 D
19 Y
18 W
17 E15
16 E14
15 E13
14 E12
13 E11
AND GATE ARRAY

```
PAL20C1                                          PAL DESIGN SPECIFICATION
74LS450                                        BIRKNER/KAZMI/BLASCO 02/19/81
16:1 MULTIPLEXER
MMI SUNNYVALE, CALIFORNIA
E0   E1   E2   E3   E4   E5   E6   E7   E8   E9   E10 GND
E11  E12  E13  E14  E15  W    Y    D    C    B    A   VCC


Y   =   /D*/C*/B*/A * E0                    ;SELECT INPUT E0
    +   /D*/C*/B* A * E1                    ;SELECT INPUT E1
    +   /D*/C* B*/A * E2                    ;SELECT INPUT E2
    +   /D*/C* B* A * E3                    ;SELECT INPUT E3
    +   /D* C*/B*/A * E4                    ;SELECT INPUT E4
    +   /D* C*/B* A * E5                    ;SELECT INPUT E5
    +   /D* C* B*/A * E6                    ;SELECT INPUT E6
    +   /D* C* B* A * E7                    ;SELECT INPUT E7
    +    D*/C*/B*/A * E8                    ;SELECT INPUT E8
    +    D*/C*/B* A * E9                    ;SELECT INPUT E9
    +    D*/C* B*/A * E10                   ;SELECT INPUT E10
    +    D*/C* B* A * E11                   ;SELECT INPUT E11
    +    D* C*/B*/A * E12                   ;SELECT INPUT E12
    +    D* C*/B* A * E13                   ;SELECT INPUT E13
    +    D* C* B*/A * E14                   ;SELECT INPUT E14
    +    D* C* B* A * E15                   ;SELECT INPUT E15
```



FUNCTION TABLE

```
D C B A E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 E10 E11 E12 E13 E14 E15 Y W

;  SELECT    -----INPUTS-----   OUTPUTS     COMMENTS
;                      111111
; D C B A    0123456789012345    Y  W
-------------------------------------------------------------
  L L L L    LHHHHHHHHHHHHHHH    L  H       INPUT E0 = 0
  L L L L    HLLLLLLLLLLLLLLL    H  L       INPUT E0 = 1
  L L L L    HHHHHHHHHHHHHHHH    H  L       TOGGLE OTHER LINES
  L L L H    HLHHHHHHHHHHHHHH    L  H       INPUT E1 = 0
  L L L H    LHLLLLLLLLLLLLLL    H  L       INPUT E1 = 1
  L L L H    HHHHHHHHHHHHHHHH    H  L       TOGGLE OTHER LINES
  L L H L    HHLHHHHHHHHHHHHH    L  H       INPUT E2 = 0
  L L H L    LLHLLLLLLLLLLLLL    H  L       INPUT E2 = 1
  L L H L    HHHHHHHHHHHHHHHH    H  L       TOGGLE OTHER LINES
  L L H H    HHHLHHHHHHHHHHHH    L  H       INPUT E3 = 0
  L L H H    LLLHLLLLLLLLLLLL    H  L       INPUT E3 = 1
  L L H H    HHHHHHHHHHHHHHHH    H  L       TOGGLE OTHER LINES
  L H L L    HHHHLHHHHHHHHHHH    L  H       INPUT E4 = 0
  L H L L    LLLLHLLLLLLLLLLL    H  L       INPUT E4 = 1
  L H L L    HHHHHHHHHHHHHHHH    H  L       TOGGLE OTHER LINES
  L H L H    HHHHHLHHHHHHHHHH    L  H       INPUT E5 = 0
  L H L H    LLLLLHLLLLLLLLLL    H  L       INPUT E5 = 1
  L H L H    HHHHHHHHHHHHHHHH    H  L       TOGGLE OTHER LINES
  L H H L    HHHHHHLHHHHHHHHH    L  H       INPUT E6 = 0
  L H H L    LLLLLLHLLLLLLLLL    H  L       INPUT E6 = 1
  L H H L    HHHHHHHHHHHHHHHH    H  L       TOGGLE OTHER LINES
  L H H H    HHHHHHHLHHHHHHHH    L  H       INPUT E7 = 0
  L H H H    LLLLLLLHLLLLLLLL    H  L       INPUT E7 = 1
  L H H H    HHHHHHHHHHHHHHHH    H  L       TOGGLE OTHER LINES
  H L L L    HHHHHHHHLHHHHHHH    L  H       INPUT E8 = 0
  H L L L    LLLLLLLLHLLLLLLL    H  L       INPUT E8 = 1
  H L L L    HHHHHHHHHHHHHHHH    H  L       TOGGLE OTHER LINES
  H L L H    HHHHHHHHHLHHHHHH    L  H       INPUT E9 = 0
  H L L H    LLLLLLLLLHLLLLLL    H  L       INPUT E9 = 1
  H L L H    HHHHHHHHHHHHHHHH    H  L       TOGGLE OTHER LINES
  H L H L    HHHHHHHHHHLHHHHH    L  H       INPUT E10 = 0
  H L H L    LLLLLLLLLLHLLLLL    H  L       INPUT E10 = 1
  H L H L    HHHHHHHHHHHHHHHH    H  L       TOGGLE OTHER LINES
  H L H H    HHHHHHHHHHHLHHHH    L  H       INPUT E11 = 0
  H L H H    LLLLLLLLLLLHLLLL    H  L       INPUT E11 = 1
  H L H H    HHHHHHHHHHHHHHHH    H  L       TOGGLE OTHER LINES
  H H L L    HHHHHHHHHHHHLHHH    L  H       INPUT E12 = 0
  H H L L    LLLLLLLLLLLLHLLL    H  L       INPUT E12 = 1
  H H L L    HHHHHHHHHHHHHHHH    H  L       TOGGLE OTHER LINES
  H H L H    HHHHHHHHHHHHHLHH    L  H       INPUT E13 = 0
  H H L H    LLLLLLLLLLLLLHLL    H  L       INPUT E13 = 1
  H H L H    HHHHHHHHHHHHHHHH    H  L       TOGGLE OTHER LINES
  H H H L    HHHHHHHHHHHHHHLH    L  H       INPUT E14 = 0
  H H H L    LLLLLLLLLLLLLLHL    H  L       INPUT E14 = 1
  H H H L    HHHHHHHHHHHHHHHH    H  L       TOGGLE OTHER LINES
  H H H H    HHHHHHHHHHHHHHHL    L  H       INPUT E15 = 0
  H H H H    LLLLLLLLLLLLLLLH    H  L       INPUT E15 = 1
  H H H H    HHHHHHHHHHHHHHHH    H  L       TOGGLE OTHER LINES
-------------------------------------------------------------
```

DESCRIPTION

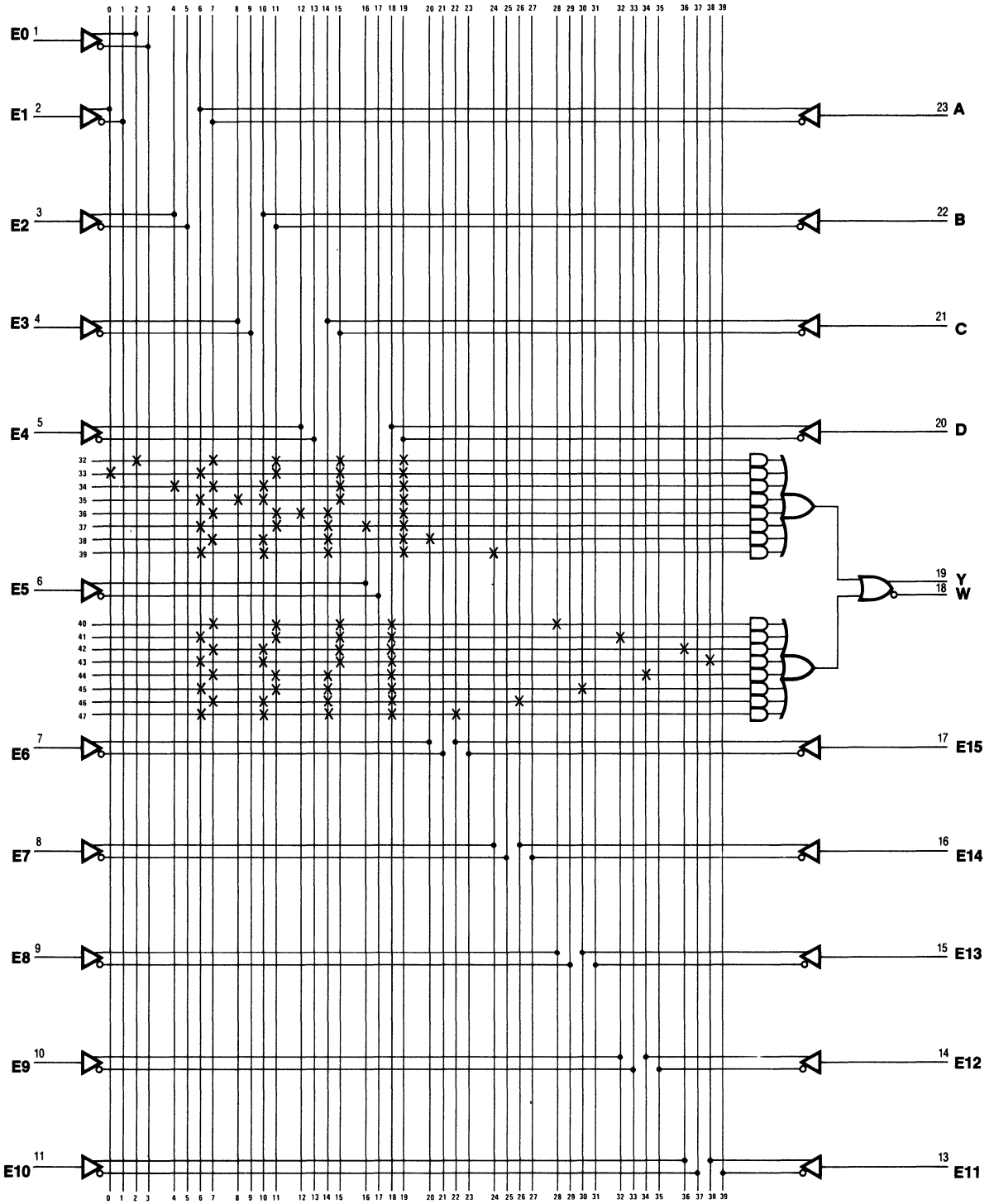
THIS IS AN EXAMPLE OF A 16-TO-1 MULTIPLEXER USING A PAL20C1. BOTH TRUE (Y) AND COMPLEMENT (W) OUTPUTS ARE PROVIDED. THE SELECT LINES A,B,C,D ARE ENCODED IN BINARY, WITH A REPRESENTING THE LSB AND D REPRESENTING THE MSB.

OPERATIONS TABLE:

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

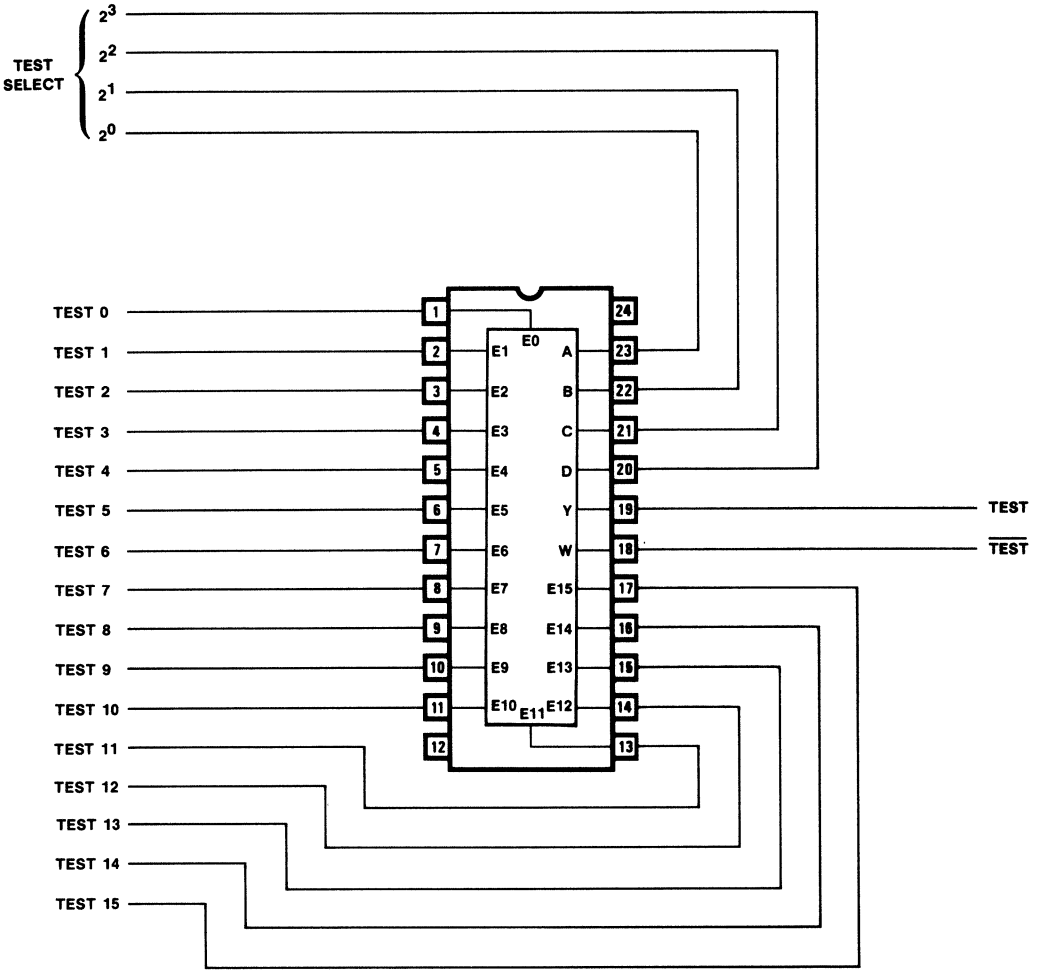
16:1 Multiplexer

Logic Diagram PAL20C1

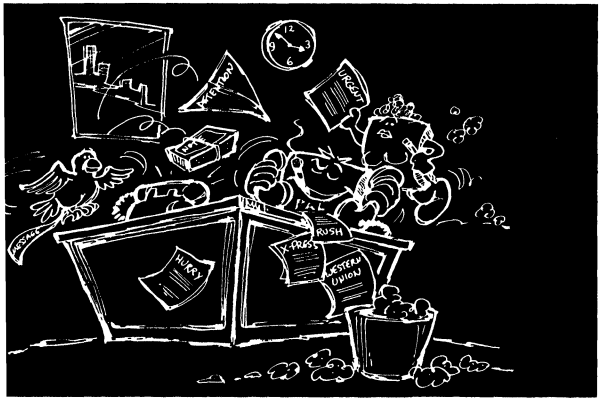


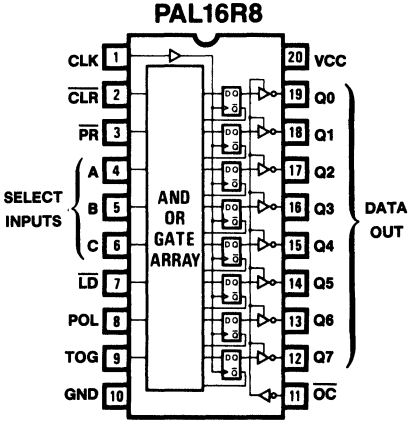
Application

Test Condition Mux



3-to-8 Demultiplexer with Control Storage





```
PAL16R8                                          PAL DESIGN SPECIFICATION
PMSI001                                            BIRKNER/COLI 04/07/82
3-TO-8 DEMULTIPLEXER WITH CONTROL STORAGE
MMI SUNNYVALE, CALIFORNIA
CLK /CLR /PR A  B  C  /LD POL TOG GND
/OC  Q7   Q6 Q5 Q4 Q3  Q2 Q1  Q0  VCC


/Q0 :=  CLR                          ;CLEAR Q0
     + /PR* LD*/POL*/C*/B*/A         ;LOAD COMPLEMENT (DECODE 000)
     + /PR* LD* POL*       A         ;LOAD TRUE
     + /PR* LD* POL*    B            ;LOAD TRUE
     + /PR* LD* POL* C               ;LOAD TRUE
     + /PR*/LD*/TOG*/Q0              ;HOLD
     + /PR*/LD* TOG* Q0              ;TOGGLE POLARITY

/Q1 :=  CLR                          ;CLEAR Q1
     + /PR* LD*/POL*/C*/B* A         ;LOAD COMPLEMENT (DECODE 001)
     + /PR* LD* POL*      /A         ;LOAD TRUE
     + /PR* LD* POL*    B            ;LOAD TRUE
     + /PR* LD* POL* C               ;LOAD TRUE
     + /PR*/LD*/TOG*/Q1              ;HOLD
     + /PR*/LD* TOG* Q1              ;TOGGLE POLARITY

/Q2 :=  CLR                          ;CLEAR Q2
     + /PR* LD*/POL*/C* B*/A         ;LOAD COMPLEMENT (DECODE 010)
     + /PR* LD* POL*       A         ;LOAD TRUE
     + /PR* LD* POL*   /B            ;LOAD TRUE
     + /PR* LD* POL* C               ;LOAD TRUE
     + /PR*/LD*/TOG*/Q2              ;HOLD
     + /PR*/LD* TOG* Q2              ;TOGGLE POLARITY

/Q3 :=  CLR                          ;CLEAR Q3
     + /PR* LD*/POL*/C* B* A         ;LOAD COMPLEMENT (DECODE 011)
     + /PR* LD* POL*      /A         ;LOAD TRUE
     + /PR* LD* POL*   /B            ;LOAD TRUE
     + /PR* LD* POL* C               ;LOAD TRUE
     + /PR*/LD*/TOG*/Q3              ;HOLD
     + /PR*/LD* TOG* Q3              ;TOGGLE POLARITY

/Q4 :=  CLR                          ;CLEAR Q4
     + /PR* LD*/POL* C*/B*/A         ;LOAD COMPLEMENT (DECODE 100)
     + /PR* LD* POL*       A         ;LOAD TRUE
     + /PR* LD* POL*    B            ;LOAD TRUE
     + /PR* LD* POL*/C               ;LOAD TRUE
     + /PR*/LD*/TOG*/Q4              ;HOLD
     + /PR*/LD* TOG* Q4              ;TOGGLE POLARITY

/Q5 :=  CLR                          ;CLEAR Q5
     + /PR* LD*/POL* C*/B* A         ;LOAD COMPLEMENT (DECODE 101)
     + /PR* LD* POL*      /A         ;LOAD TRUE
     + /PR* LD* POL*    B            ;LOAD TRUE
     + /PR* LD* POL*/C               ;LOAD TRUE
     + /PR*/LD*/TOG*/Q5              ;HOLD
     + /PR*/LD* TOG* Q5              ;TOGGLE POLARITY
```

```
/Q6 :=  CLR                          ;CLEAR Q6
     + /PR* LD*/POL* C* B*/A         ;LOAD COMPLEMENT (DECODE 110)
     + /PR* LD* POL*       A         ;LOAD TRUE
     + /PR* LD* POL*   /B            ;LOAD TRUE
     + /PR* LD* POL*/C               ;LOAD TRUE
     + /PR*/LD*/TOG*/Q6              ;HOLD
     + /PR*/LD* TOG* Q6              ;TOGGLE POLARITY

/Q7 :=  CLR                          ;CLEAR Q7
     + /PR* LD*/POL* C* B* A         ;LOAD COMPLEMENT (DECODE 111)
     + /PR* LD* POL*      /A         ;LOAD TRUE
     + /PR* LD* POL*   /B            ;LOAD TRUE
     + /PR* LD* POL*/C               ;LOAD TRUE
     + /PR*/LD*/TOG*/Q7              ;HOLD
     + /PR*/LD* TOG* Q7              ;TOGGLE POLARITY


FUNCTION TABLE

/OC CLK /CLR /PR /LD POL TOG C B A Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0

;CONTROL     FUNCTIONS     POLARITY  INPUT    OUTPUT
;/OC CLK    /CLR /PR /LD   POL  TOG   CBA     Q7----Q0     COMMENTS
-------------------------------------------------------------------------------
  L   C       L   L   L     H    L     XXX     LLLLLLLL     CLEAR  (OVERRIDES /PR)
  L   C       H   L   L     H    L     XXX     HHHHHHHH     PRESET (OVERRIDES /LD)
  L   C       H   H   L     H    X     LLL     LLLLLLLH     LOAD 0 (TRUE INPUT)
  L   C       H   H   L     H    X     LLH     LLLLLLHL     LOAD 1 (TRUE INPUT)
  L   C       H   H   L     H    X     LHL     LLLLLHLL     LOAD 2 (TRUE INPUT)
  L   C       H   H   L     H    X     LHH     LLLLHLLL     LOAD 3 (TRUE INPUT)
  L   C       H   H   L     H    X     HLL     LLLHLLLL     LOAD 4 (TRUE INPUT)
  L   C       H   H   L     H    X     HLH     LLHLLLLL     LOAD 5 (TRUE INPUT)
  L   C       H   H   L     H    X     HHL     LHLLLLLL     LOAD 6 (TRUE INPUT)
  L   C       H   H   L     H    X     HHH     HLLLLLLL     LOAD 7 (TRUE INPUT)
  L   C       H   H   H     X    L     XXX     HLLLLLLL     HOLD 7 (TRUE POLARITY)
  L   C       H   H   H     X    H     XXX     LHHHHHHH     HOLD (TOGGLE POLARITY)
  L   C       H   H   H     X    H     XXX     HLLLLLLL     HOLD (TOGGLE POLARITY)
  L   C       H   H   L     L    X     LLL     HHHHHHHL     LOAD 0 (COMP INPUT)
  L   C       H   H   L     L    X     LLH     HHHHHHLH     LOAD 1 (COMP INPUT)
  L   C       H   H   L     L    X     LHL     HHHHHLHH     LOAD 2 (COMP INPUT)
  L   C       H   H   L     L    X     LHH     HHHHLHHH     LOAD 3 (COMP INPUT)
  L   C       H   H   L     L    X     HLL     HHHLHHHH     LOAD 4 (COMP INPUT)
  L   C       H   H   L     L    X     HLH     HHLHHHHH     LOAD 5 (COMP INPUT)
  L   C       H   H   L     L    X     HHL     HLHHHHHH     LOAD 6 (COMP INPUT)
  L   C       H   H   L     L    X     HHH     LHHHHHHH     LOAD 7 (COMP INPUT)
  L   C       H   H   H     X    L     XXX     LHHHHHHH     HOLD 7 (TRUE POLARITY)
  L   C       H   H   H     X    H     XXX     HLLLLLLL     HOLD (TOGGLE POLARITY)
  L   C       H   H   H     X    H     XXX     LHHHHHHH     HOLD (TOGGLE POLARITY)
  H   X       X   X   X     X    X     XXX     ZZZZZZZZ     TEST HI-Z
-------------------------------------------------------------------------------
```

DESCRIPTION

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

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

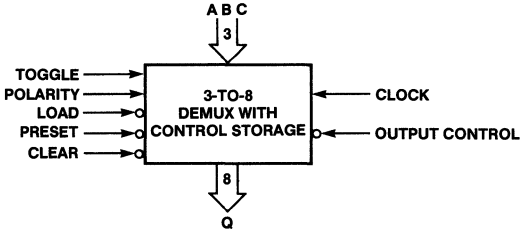
THE HOLD OPERATION HOLDS THE PREVIOUS VALUE IN THE REGISTER WHEN TOGGLE IS FALSE (TOG=L) REGARDLESS OF CLOCK TRANSITIONS. THE TOGGLE POLARITY OPERATION TOGGLES THE POLARITY OF THE DATA IN THE OUTPUT REGISTER WHEN TOGGLE IS TRUE (TOG=H).

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

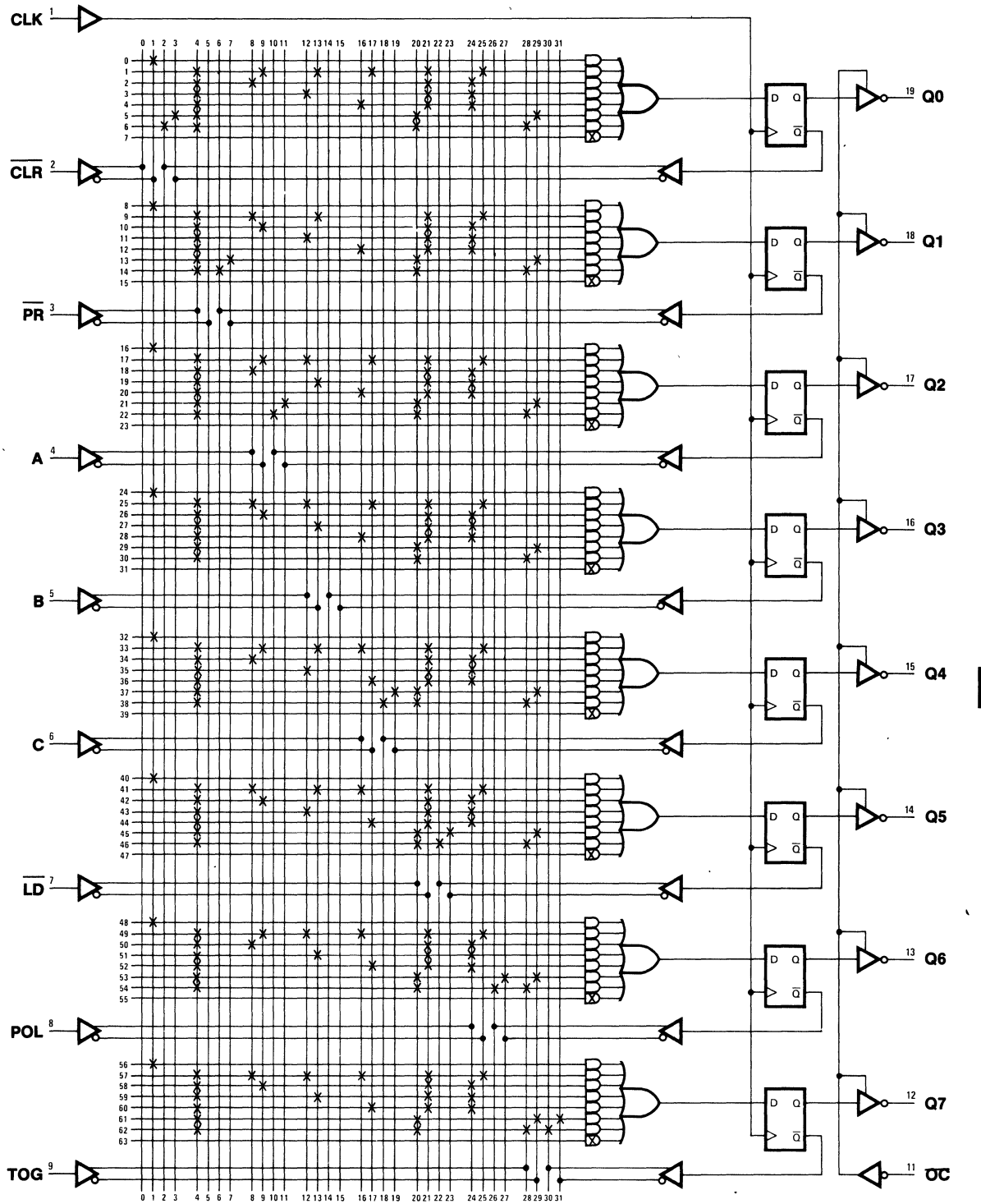
THE POLARITY OF "POL" MAY BE CHANGED IN THE LOGIC EQUATIONS TO SUIT SPECIFIC APPLICATIONS SO THAT CERTAIN OUTPUT POLARITIES ARE ASSERTIVE HIGH WHILE OTHERS ARE ASSERTIVE LOW.

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

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

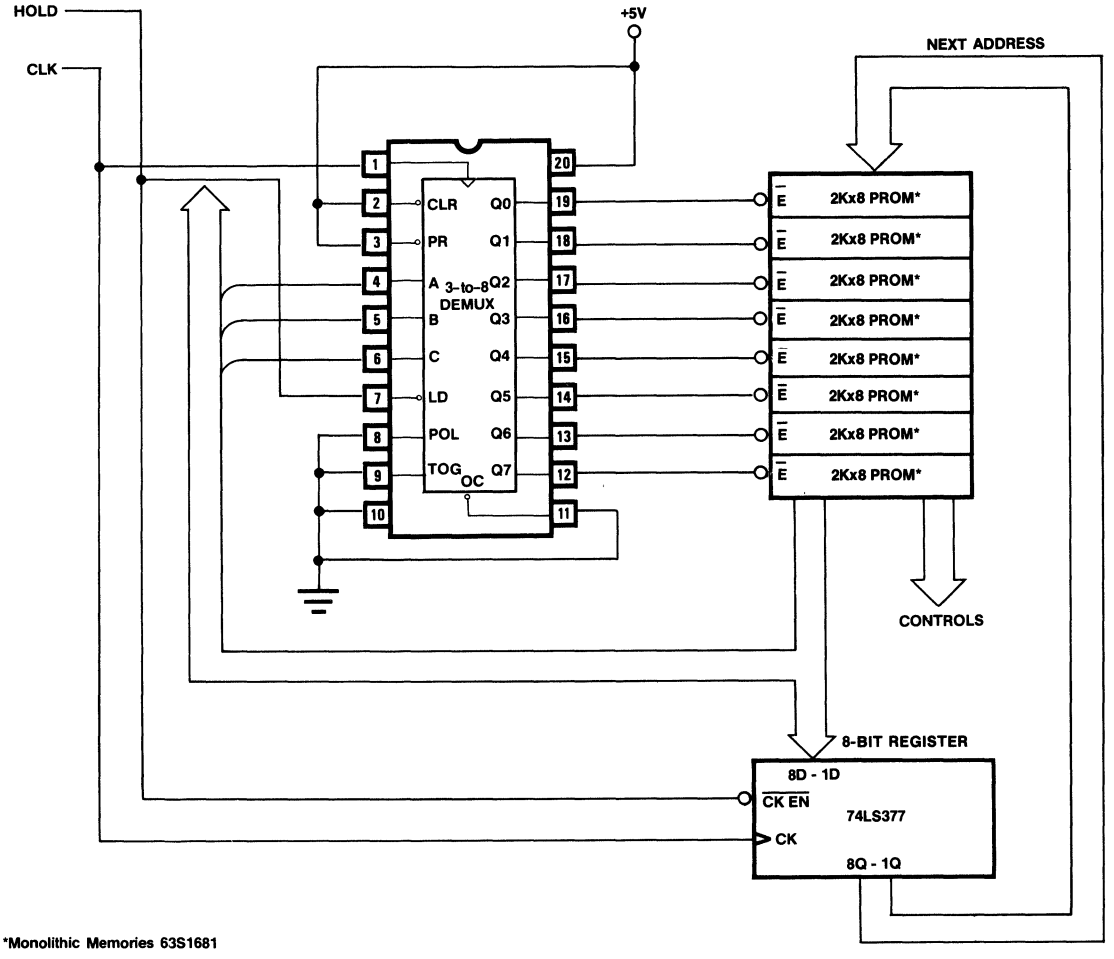


Logic Diagram PAL16R8



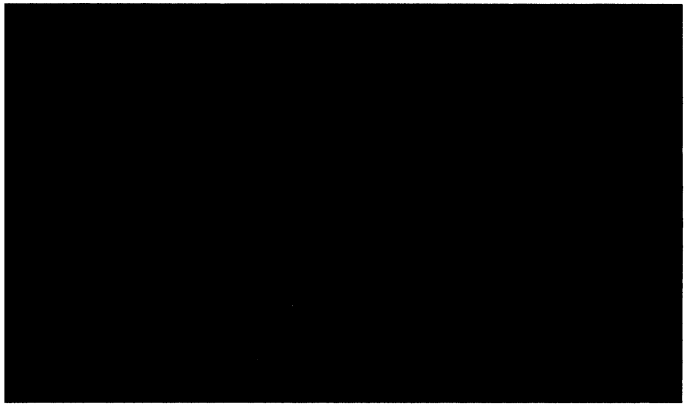
Application

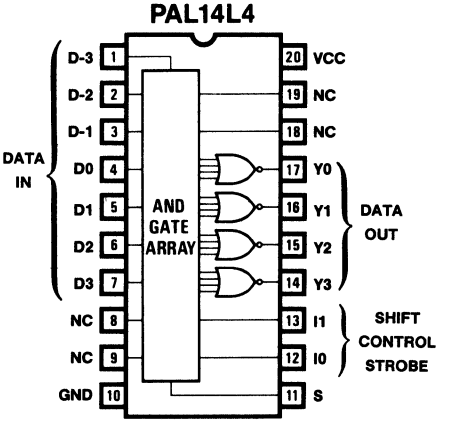
Pipeline Demux for Microprogram State Machine



*Monolithic Memories 63S1681

4-Bit Shifter





```
PAL14L4                                              PAL DESIGN SPECIFICATION
PD-101                                                 VINCENT COLI 10/05/81
4-BIT SHIFTER
MMI SUNNYVALE, CALIFORNIA
D-3 D-2 D-1 D0 D1 D2 D3 NC NC GND
 S  I0  I1  Y3 Y2 Y1 Y0 NC NC VCC


/Y0 = /S*/I1*/I0*/D0       ;NO SHIFT (D0)
    + /S*/I1* I0*/D-1      ;SHIFT 1 PLACE (D-1)
    + /S* I1*/I0*/D-2      ;SHIFT 2 PLACES (D-2)
    + /S* I1* I0*/D-3      ;SHIFT 3 PLACES (D-3)

/Y1 = /S*/I1*/I0*/D1       ;NO SHIFT (D1)
    + /S*/I1* I0*/D0       ;SHIFT 1 PLACE (D0)
    + /S* I1*/I0*/D-1      ;SHIFT 2 PLACES (D-1)
    + /S* I1* I0*/D-2      ;SHIFT 3 PLACES (D-2)

/Y2 = /S*/I1*/I0*/D2       ;NO SHIFT (D2)
    + /S*/I1* I0*/D1       ;SHIFT 1 PLACE (D1)
    + /S* I1*/I0*/D0       ;SHIFT 2 PLACES (D0)
    + /S* I1* I0*/D-1      ;SHIFT 3 PLACES (D-1)

/Y3 = /S*/I1*/I0*/D3       ;NO SHIFT (D3)
    + /S*/I1* I0*/D2       ;SHIFT 1 PLACE (D2)
    + /S* I1*/I0*/D1       ;SHIFT 2 PLACES (D1)
    + /S* I1* I0*/D0       ;SHIFT 3 PLACES (D0)


FUNCTION TABLE

S I1 I0 D3 D2 D1 D0 D-1 D-2 D-3 Y3 Y2 Y1 Y0

;CONTROL      INPUTS     OUTPUTS
;   I I      DDDD D-     YYYY
; S 1 0      3210 123    3210      COMMENTS
-------------------------------------------------------------
  L L L      HHHL HHH    HHHL      NO SHIFT L (I=0)
  L L H      HHHL HHH    HHLH      SHIFT L 1 PLACE (I=1)
  L H L      HHHL HHH    HLHH      SHIFT L 2 PLACES (I=2)
  L H H      HHHL HHH    LHHH      SHIFT L 3 PLACES (I=3)
  L L L      LLLH LLL    LLLH      NO SHIFT H (I=0)
  L L H      LLLH LLL    LLHL      SHIFT H 1 PLACE (I=1)
  L H L      LLLH LLL    LHLL      SHIFT H 2 PLACES (I=2)
  L H H      LLLH LLL    HLLL      SHIFT H 3 PLACES (I=3)
  L L L      LLLL HHH    LLLL      NO SHIFT L's (I=0)
  L L H      HLLL LHH    LLLL      SHIFT L's 1 PLACE (I=1)
  L H L      HHLL LLH    LLLL      SHIFT L's 2 PLACES (I=2)
  L H H      HHHL LLL    LLLL      SHIFT L's 3 PLACES (I=3)
  L L L      HHHH LLL    HHHH      NO SHIFT H's (I=0)
  L L H      LHHH HLL    HHHH      SHIFT H's 1 PLACE (I=1)
  L H L      LLHH HHL    HHHH      SHIFT H's 2 PLACES (I=2)
  L H H      LLLH HHH    HHHH      SHIFT H's 3 PLACES (I=3)
  H X X      LLLL LLL    HHHH      STROBE TEST WITH S=L
  H X X      HHHH HHH    HHHH      STROBE TEST WITH S=H
-------------------------------------------------------------
```

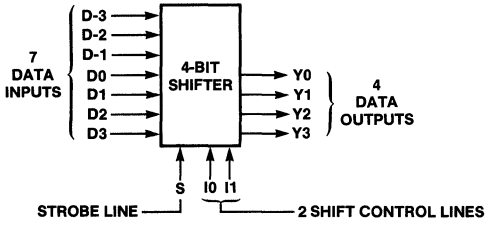
DESCRIPTION

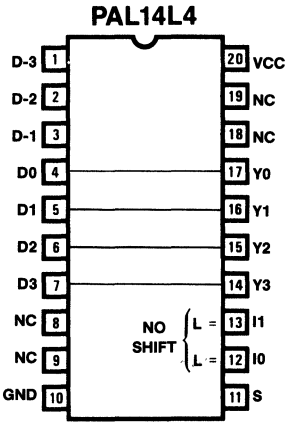
THE 4-BIT SHIFTER ACCEPTS A 4-BIT DATA WORD (D) AND SHIFTS THE WORD 0, 1, 2, OR 3 PLACES TO OUTPUTS (Y). THE NUMBER OF PLACES TO BE SHIFTED IS DETERMINED BY TWO CONTROL LINES (I1,I0) WHICH ARE ENCODED IN BINARY WITH I0 REPRESENTING THE LSB. A STROBE LINE (S) IS PROVIDED TO GATE THE OUTPUTS OFF (Y=H)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

OPERATIONS TABLE:

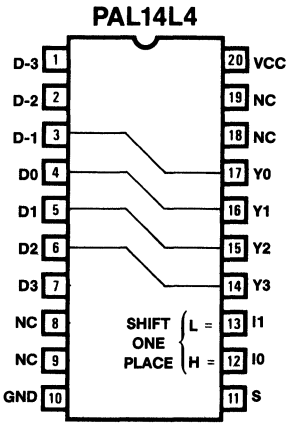
S	I1	I0	D3-D-3	Y3-Y0	OPERATION
H	X	X	X	H	STROBE HIGH
L	L	L	D	D	NO SHIFT
L	L	H	D	S(D)1	SHIFT 1 PLACE
L	H	L	D	S(D)2	SHIFT 2 PLACES
L	H	H	D	S(D)3	SHIFT 3 PLACES

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

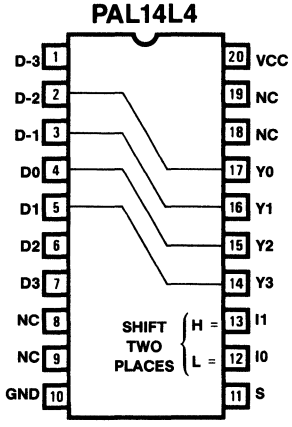



PAL14L4
D-3 1
D-2 2
D-1 3
D0 4
D1 5
D2 6
D3 7
NC 8
NC 9
GND 10
20 VCC
19 NC
18 NC
17 Y0
16 Y1
15 Y2
14 Y3
NO SHIFT
L = 13 I1
L = 12 I0
11 S

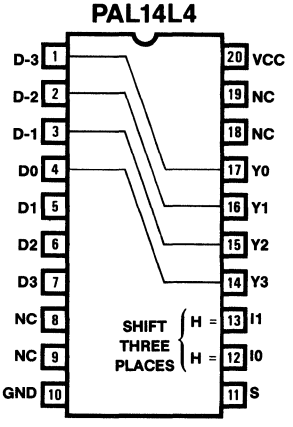
I = 0


PAL14L4
D-3 1
D-2 2
D-1 3
D0 4
D1 5
D2 6
D3 7
NC 8
NC 9
GND 10
20 VCC
19 NC
18 NC
17 Y0
16 Y1
15 Y2
14 Y3
SHIFT ONE PLACE
L = 13 I1
H = 12 I0
11 S

I = 1


PAL14L4
D-3 1
D-2 2
D-1 3
D0 4
D1 5
D2 6
D3 7
NC 8
NC 9
GND 10
20 VCC
19 NC
18 NC
17 Y0
16 Y1
15 Y2
14 Y3
SHIFT TWO PLACES
H = 13 I1
L = 12 I0
11 S

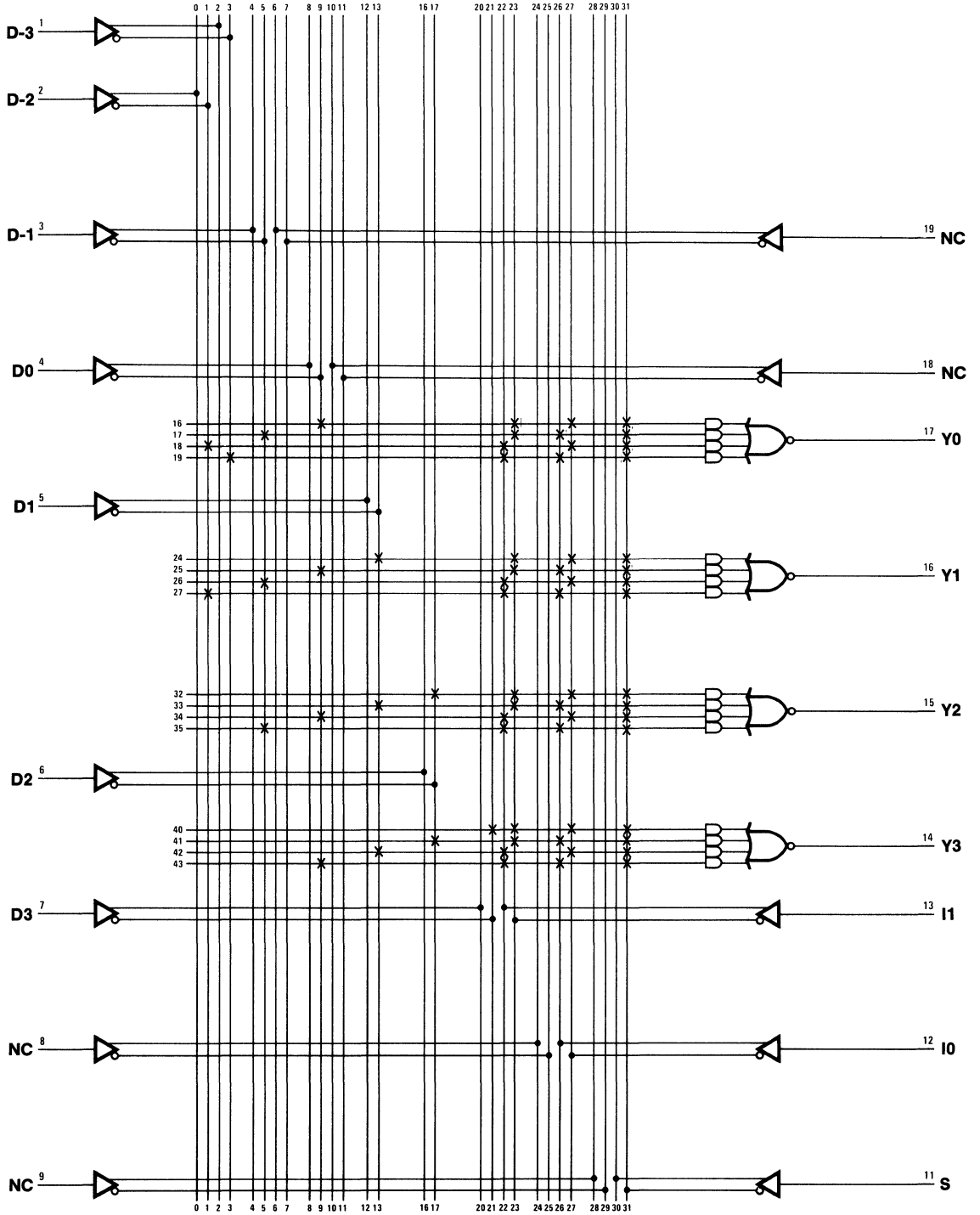
I = 2


PAL14L4
D-3 1
D-2 2
D-1 3
D0 4
D1 5
D2 6
D3 7
NC 8
NC 9
GND 10
20 VCC
19 NC
18 NC
17 Y0
16 Y1
15 Y2
14 Y3
SHIFT THREE PLACES
H = 13 I1
H = 12 I0
11 S

I = 3

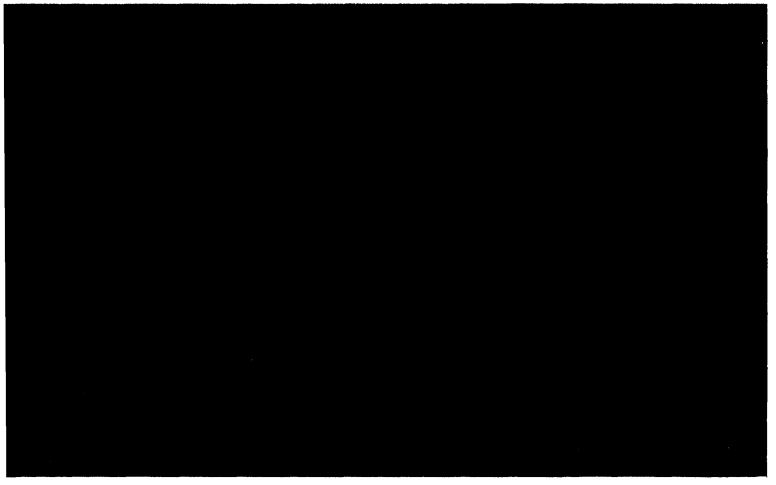
4-Bit Shifter

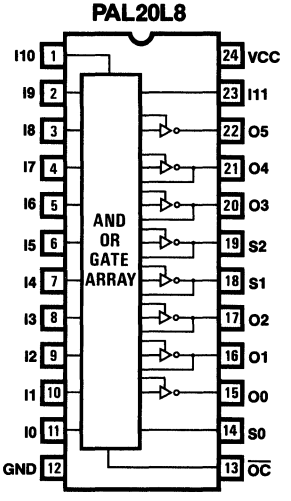
Logic Diagram PAL14L4




IMPORTANT
NOTES

6-Bit Right Shifter

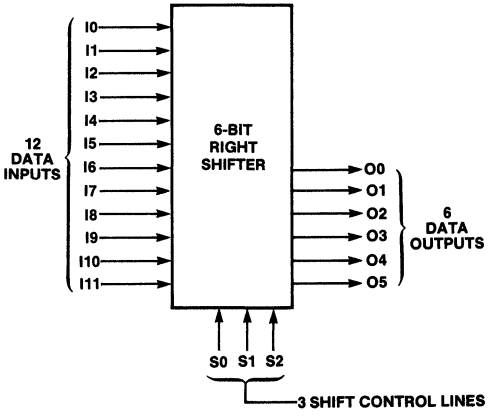




```
PAL20L8                                                PAL DESIGN SPECIFICATION
SHIFT6                                                       FRANK LEE 03/10/83
6-BIT RIGHT SHIFTER
MMI SUNNYVALE, CALIFORNIA
I10 I9 I8 I7 I6 I5 I4 I3 I2 I1 I0 GND
/OC S0 O0 O1 O2 S1 S2 O3 O4 O5 I11 VCC

IF(OC) /O5 =   S2* S1*/S0*/I11          ;SHIFT 6 BITS
           +   S2*/S1* S0*/I10          ;SHIFT 5 BITS
           +   S2*/S1*/S0*/I9           ;SHIFT 4 BITS
           +  /S2* S1* S0*/I8           ;SHIFT 3 BITS
           +  /S2* S1*/S0*/I7           ;SHIFT 2 BITS
           +  /S2*/S1* S0*/I6           ;SHIFT 1 BIT
           +  /S2*/S1*/S0*/I5           ;SHIFT 0 BIT

IF(OC) /O4 =   S2* S1*/S0*/I10          ;SHIFT 6 BITS
           +   S2*/S1* S0*/I9           ;SHIFT 5 BITS
           +   S2*/S1*/S0*/I8           ;SHIFT 4 BITS
           +  /S2* S1* S0*/I7           ;SHIFT 3 BITS
           +  /S2* S1*/S0*/I6           ;SHIFT 2 BITS
           +  /S2*/S1* S0*/I5           ;SHIFT 1 BIT
           +  /S2*/S1*/S0*/I4           ;SHIFT 0 BIT

IF(OC) /O3 =   S2* S1*/S0*/I9           ;SHIFT 6 BITS
           +   S2*/S1* S0*/I8           ;SHIFT 5 BITS
           +   S2*/S1*/S0*/I7           ;SHIFT 4 BITS
           +  /S2* S1* S0*/I6           ;SHIFT 3 BITS
           +  /S2* S1*/S0*/I5           ;SHIFT 2 BITS
           +  /S2*/S1* S0*/I4           ;SHIFT 1 BIT
           +  /S2*/S1*/S0*/I3           ;SHIFT 0 BIT

IF(OC) /O2 =   S2* S1*/S0*/I8           ;SHIFT 6 BITS
           +   S2*/S1* S0*/I7           ;SHIFT 5 BITS
           +   S2*/S1*/S0*/I6           ;SHIFT 4 BITS
           +  /S2* S1* S0*/I5           ;SHIFT 3 BITS
           +  /S2* S1*/S0*/I4           ;SHIFT 2 BITS
           +  /S2*/S1* S0*/I3           ;SHIFT 1 BIT
           +  /S2*/S1*/S0*/I2           ;SHIFT 0 BIT

IF(OC) /O1 =   S2* S1*/S0*/I7           ;SHIFT 6 BITS
           +   S2*/S1* S0*/I6           ;SHIFT 5 BITS
           +   S2*/S1*/S0*/I5           ;SHIFT 4 BITS
           +  /S2* S1* S0*/I4           ;SHIFT 3 BITS
           +  /S2* S1*/S0*/I3           ;SHIFT 2 BITS
           +  /S2*/S1* S0*/I2           ;SHIFT 1 BIT
           +  /S2*/S1*/S0*/I1           ;SHIFT 0 BIT

IF(OC) /O0 =   S2* S1*/S0*/I6           ;SHIFT 6 BITS
           +   S2*/S1* S0*/I5           ;SHIFT 5 BITS
           +   S2*/S1*/S0*/I4           ;SHIFT 4 BITS
           +  /S2* S1* S0*/I3           ;SHIFT 3 BITS
           +  /S2* S1*/S0*/I2           ;SHIFT 2 BITS
           +  /S2*/S1* S0*/I1           ;SHIFT 1 BIT
           +  /S2*/S1*/S0*/I0           ;SHIFT 0 BIT
```

FUNCTION TABLE

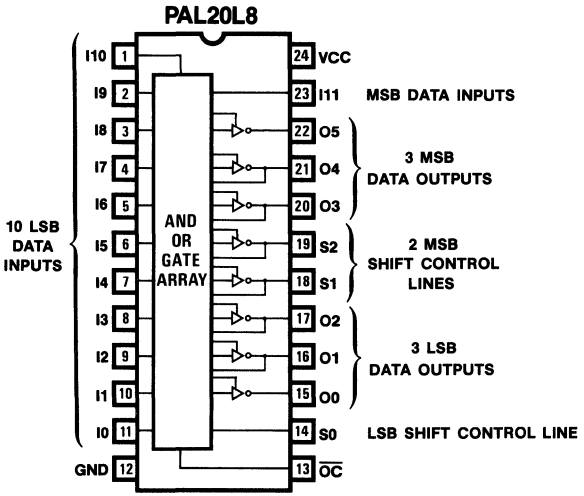
```
/OC S2 S1 S0 I11 I10 I9 I8 I7 I6 I5 I4 I3 I2 I1 I0 O5 O4 O3 O2 O1 O0

;/     II
;O SSS 11IIII IIIIII  OOOOOO
;C 210 109876 543210  543210     COMMENTS
-------------------------------------------------------------------
 H XXX XXXXXX XXXXXX  ZZZZZZ     HI-Z TEST
 L LLL LLHLLH HLLHHH  HLLHHH     SHIFT COUNT = 0
 L LLL HHLHHL LHHLLL  LHHLLL     SHIFT COUNT = 0
 L LLH LLHLLH HLLHHH  HHLLHH     SHIFT COUNT = 1
 L LLH HHLHHL LHHLLL  LLHHLL     SHIFT COUNT = 1
 L LHL LLHLLH HLLHHH  LHHLLH     SHIFT COUNT = 2
 L LHL HHLHHL LHHLLL  HLLHHL     SHIFT COUNT = 2
 L LHH LLHLLH HLLHHH  LLHHLL     SHIFT COUNT = 3
 L LHH HHLHHL LHHLLL  HHLLHH     SHIFT COUNT = 3
 L HLL LLHLLH HLLHHH  HLLHHL     SHIFT COUNT = 4
 L HLL HHLHHL LHHLLL  LHHLLH     SHIFT COUNT = 4
 L HLH LLHLLH HLLHHH  LHLLHH     SHIFT COUNT = 5
 L HLH HHLHHL LHHLLL  HLHHLL     SHIFT COUNT = 5
 L HHL LLHLLH HLLHHH  LLHLLH     SHIFT COUNT = 6
 L HHL HHLHHL LHHLLL  HHLHHL     SHIFT COUNT = 6
 L HHH LLHLLH HLLHHH  HHHHHH     SHIFT COUNT = 7
 L HHH HHLHHL LHHLLL  HHHHHH     SHIFT COUNT = 7
-------------------------------------------------------------------
```



DESCRIPTION

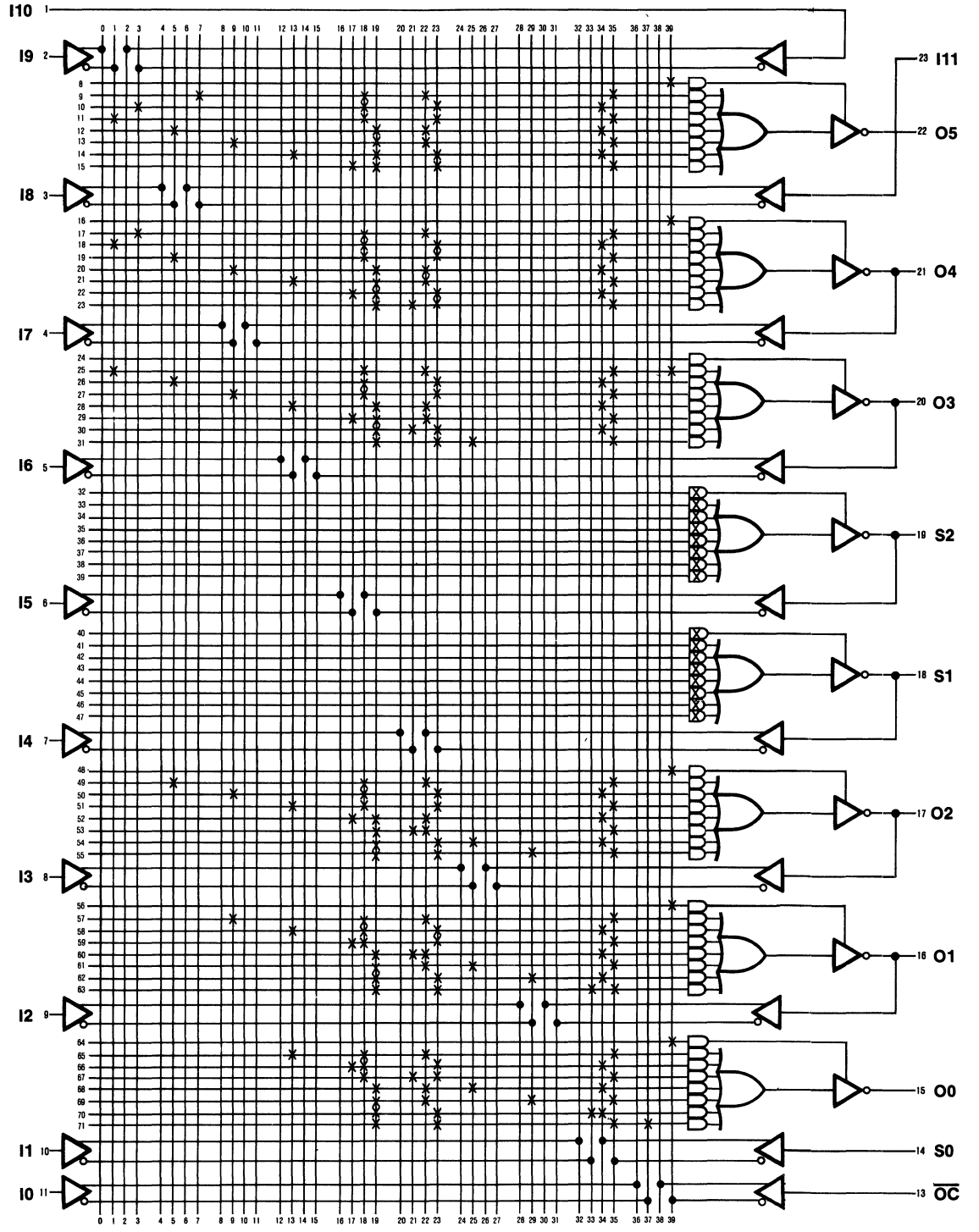
THE 6-BIT RIGHT SHIFTER IS A SHIFTER WHICH CAN ONLY SHIFT RIGHT TO A MAXIMUM OF 6 BITS AT A TIME. THERE ARE 12 INPUTS (I11-10) WHICH SHIFT INTO 6 POSITIONS (O5-O0). SHIFTING IS CONTROLLED BY A 3-BIT CONTROL (S2-S0). THERE IS ALSO AN OUTPUT ENABLE (/OC) FOR ALL OUTPUTS. ALL OUTPUTS ARE ACTIVE LOWS.

IN ADDITION, THE ABOVE TEST VECTORS (IN THE FUNCTION TABLE) ARE FOR 100% FAULT GRADING.



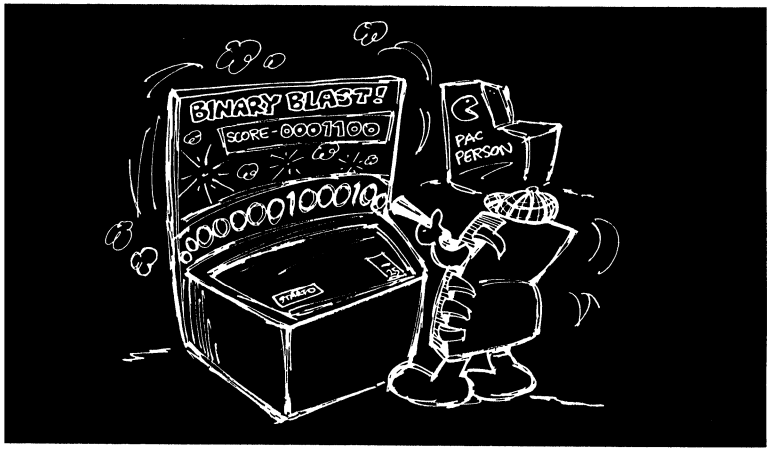
6-Bit Right Shifter

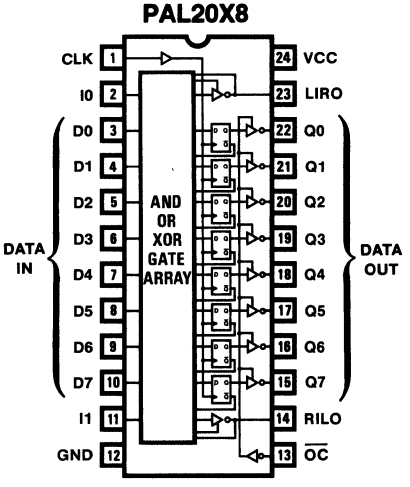
Logic Diagram PAL20L8




IMPORTANT.
NOTES

Octal Shift Register





6

```
PAL20X8                                          PAL DESIGN SPECIFICATION
74LS498                                               UDI GORDON 02/20/81
OCTAL SHIFT REGISTER
MMI SUNNYVALE, CALIFORNIA
CLK  I0  D0 D1 D2 D3 D4 D5 D6 D7  I1  GND
/OC RILO Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0 LIRO VCC


/Q0 :=  /I1*/I0*/Q0                    ;HOLD Q0
     +  /I1* I0*/Q1                    ;SHIFT RIGHT
    :+:  I1*/I0*/LIRO                  ;SHIFT LEFT
     +   I1* I0*/D0                    ;LOAD D0

/Q1 :=  /I1*/I0*/Q1                    ;HOLD Q1
     +  /I1* I0*/Q2                    ;SHIFT RIGHT
    :+:  I1*/I0*/Q0                    ;SHIFT LEFT
     +   I1* I0*/D1                    ;LOAD D1

/Q2 :=  /I1*/I0*/Q2                    ;HOLD Q2
     +  /I1* I0*/Q3                    ;SHIFT RIGHT
    :+:  I1*/I0*/Q1                    ;SHIFT LEFT
     +   I1* I0*/D2                    ;LOAD D2

/Q3 :=  /I1*/I0*/Q3                    ;HOLD Q3
     +  /I1* I0*/Q4                    ;SHIFT RIGHT
    :+:  I1*/I0*/Q2                    ;SHIFT LEFT
     +   I1* I0*/D3                    ;LOAD D3

/Q4 :=  /I1*/I0*/Q4                    ;HOLD Q4
     +  /I1* I0*/Q5                    ;SHIFT RIGHT
    :+:  I1*/I0*/Q3                    ;SHIFT LEFT
     +   I1* I0*/D4                    ;LOAD D4

/Q5 :=  /I1*/I0*/Q5                    ;HOLD Q5
     +  /I1* I0*/Q6                    ;SHIFT RIGHT
    :+:  I1*/I0*/Q4                    ;SHIFT LEFT
     +   I1* I0*/D5                    ;LOAD D5

/Q6 :=  /I1*/I0*/Q6                    ;HOLD Q6
     +  /I1* I0*/Q7                    ;SHIFT RIGHT
    :+:  I1*/I0*/Q5                    ;SHIFT LEFT
     +   I1* I0*/D6                    ;LOAD D6

/Q7 :=  /I1*/I0*/Q7                    ;HOLD Q7
     +  /I1* I0*/RILO                  ;SHIFT RIGHT
    :+:  I1*/I0*/Q6                    ;SHIFT LEFT
     +   I1* I0*/D7                    ;LOAD D7

IF(/I1*I0) /LIRO = /Q0                 ;LEFT IN RIGHT OUT

IF(I1*/I0) /RILO = /Q7                 ;RIGHT IN LEFT OUT
```

FUNCTION TABLE

I1 I0 D7 D6 D5 D4 D3 D2 D1 D0 CLK /OC RILO LIRO Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0

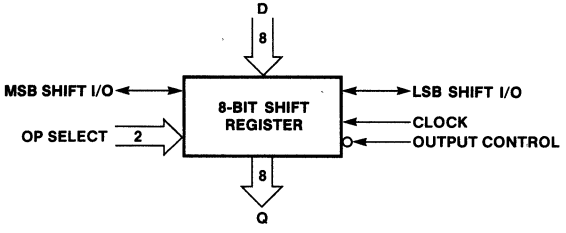
;INST	DATA IN D7----D0	CLK	/OC	RILO	LIRO	Q OUT Q7----Q0	COMMENTS
HH	LLLLLLLL	C	L	Z	Z	LLLLLLLL	LOAD ZEROS
LL	XXXXXXXX	C	L	Z	Z	LLLLLLLL	HOLD
HL	XXXXXXXX	C	L	L	H	LLLLLLLH	SHIFT LEFT IN A H
HL	XXXXXXXX	C	L	L	L	LLLLLLHL	SHIFT LEFT IN A L
HL	XXXXXXXX	C	L	L	L	LLLLLHLL	SHIFT LEFT IN A L
HL	XXXXXXXX	C	L	L	L	LLLLHLLL	SHIFT LEFT IN A L
HL	XXXXXXXX	C	L	L	L	LLLHLLLL	SHIFT LEFT IN A L
HL	XXXXXXXX	C	L	L	L	LLHLLLLL	SHIFT LEFT IN A L
HL	XXXXXXXX	C	L	L	L	LHLLLLLL	SHIFT LEFT IN A L
HL	XXXXXXXX	C	L	H	L	HLLLLLLL	SHIFT LEFT IN A L
HL	XXXXXXXX	C	L	L	L	LLLLLLLL	SHIFT LEFT IN A L
LL	XXXXXXXX	X	H	Z	Z	ZZZZZZZZ	TEST HI-Z
HH	HHHHHHHH	C	L	Z	Z	HHHHHHHH	LOAD ONES
LL	XXXXXXXX	C	L	Z	Z	HHHHHHHH	HOLD
LH	XXXXXXXX	C	L	L	H	LHHHHHHH	SHIFT RIGHT IN A L
LH	XXXXXXXX	C	L	H	H	HLHHHHHH	SHIFT RIGHT IN A H
LH	XXXXXXXX	C	L	H	H	HHLHHHHH	SHIFT RIGHT IN A H
LH	XXXXXXXX	C	L	H	H	HHHLHHHH	SHIFT RIGHT IN A H
LH	XXXXXXXX	C	L	H	H	HHHHLHHH	SHIFT RIGHT IN A H
LH	XXXXXXXX	C	L	H	H	HHHHHLHH	SHIFT RIGHT IN A H
LH	XXXXXXXX	C	L	H	H	HHHHHHLH	SHIFT RIGHT IN A H
LH	XXXXXXXX	C	L	H	L	HHHHHHHL	SHIFT RIGHT IN A H
LH	XXXXXXXX	C	L	H	H	HHHHHHHH	SHIFT RIGHT IN A H
LL	XXXXXXXX	X	H	Z	Z	ZZZZZZZZ	TEST HI-Z

DESCRIPTION

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

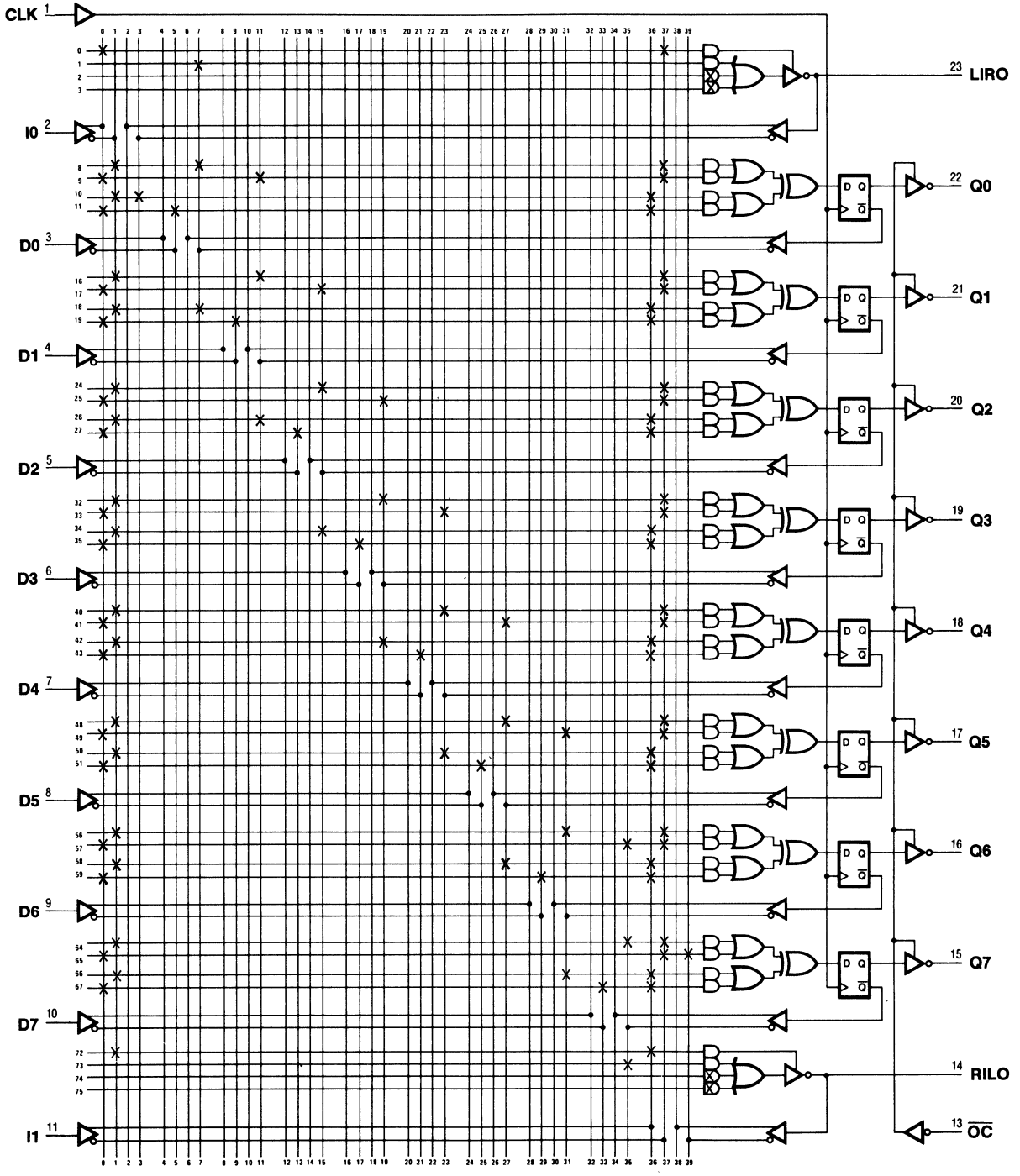
/OC	CLK	I1	I0	D7-D0	Q7-Q0	OPERATION
H	X	X	X	X	Z	HI-Z
L	C	L	L	X	L	HOLD
L	C	L	H	X	SR(Q)	SHIFT RIGHT
L	C	H	L	X	SL(Q)	SHIFT LEFT
L	C	H	H	D	D	LOAD

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



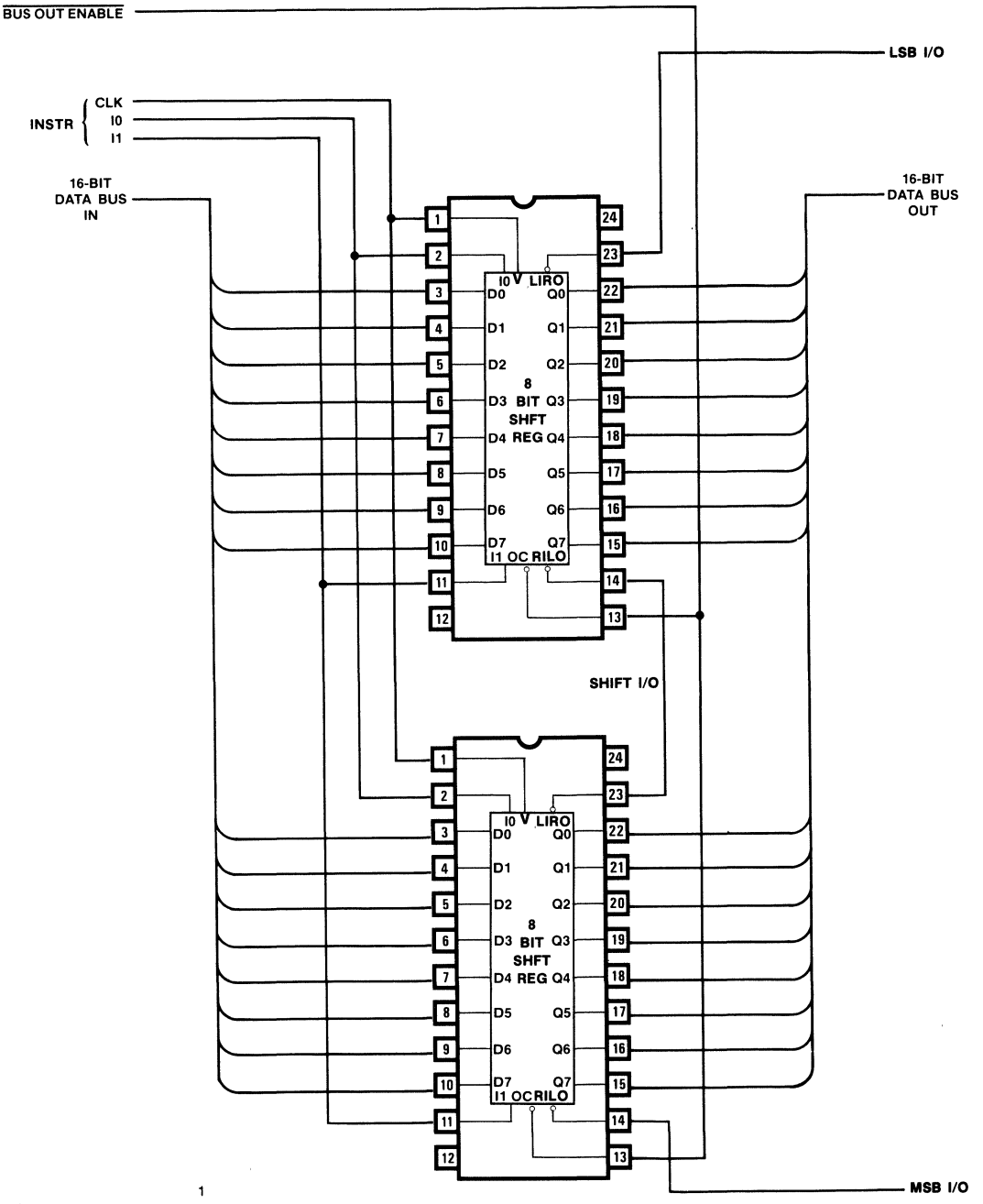
Octal Shift Register

Logic Diagram PAL20X8



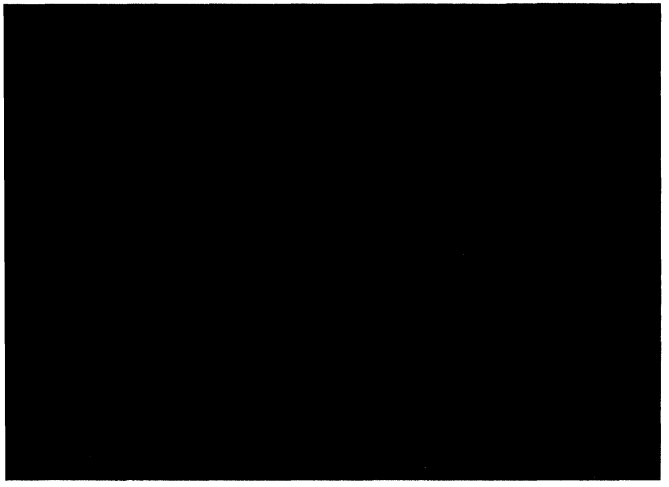
Application

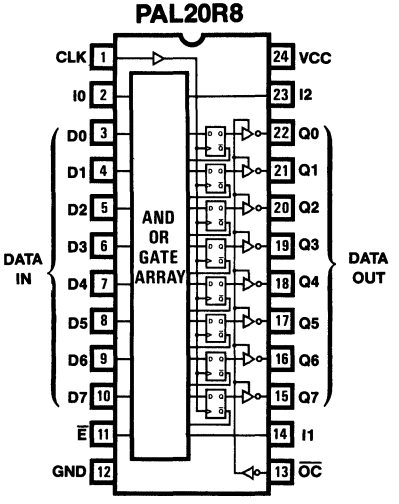
16-Bit Shift Register



NOTE: $f_{MAX} = \frac{1}{t_{PD\ CLK\ TO\ LIRO} + t_{SU}}$

Octal Registered Barrel Shifter





6

```
PAL20R8                                          PAL DESIGN SPECIFICATION
P7095                                               VINCENT COLI 09/10/82
OCTAL REGISTERED BARREL SHIFTER
MMI SUNNYVALE, CALIFORNIA
CLK I0 D0 D1 D2 D3 D4 D5 D6 D7 /E GND
/OC I1 Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0 I2 VCC


/Q0 := /E*/I2*/I1*/I0*/D0       ;SHIFT 0 PLACES
     + /E*/I2*/I1* I0*/D1       ;SHIFT 1 PLACES
     + /E*/I2* I1*/I0*/D2       ;SHIFT 2 PLACES
     + /E*/I2* I1* I0*/D3       ;SHIFT 3 PLACES
     + /E* I2*/I1*/I0*/D4       ;SHIFT 4 PLACES
     + /E* I2*/I1* I0*/D5       ;SHIFT 5 PLACES
     + /E* I2* I1*/I0*/D6       ;SHIFT 6 PLACES
     + /E* I2* I1* I0*/D7       ;SHIFT 7 PLACES

/Q1 := /E*/I2*/I1*/I0*/D1       ;SHIFT 0 PLACES
     + /E*/I2*/I1* I0*/D2       ;SHIFT 1 PLACES
     + /E*/I2* I1*/I0*/D3       ;SHIFT 2 PLACES
     + /E*/I2* I1* I0*/D4       ;SHIFT 3 PLACES
     + /E* I2*/I1*/I0*/D5       ;SHIFT 4 PLACES
     + /E* I2*/I1* I0*/D6       ;SHIFT 5 PLACES
     + /E* I2* I1*/I0*/D7       ;SHIFT 6 PLACES
     + /E* I2* I1* I0*/D0       ;SHIFT 7 PLACES

/Q2 := /E*/I2*/I1*/I0*/D2       ;SHIFT 0 PLACES
     + /E*/I2*/I1* I0*/D3       ;SHIFT 1 PLACES
     + /E*/I2* I1*/I0*/D4       ;SHIFT 2 PLACES
     + /E*/I2* I1* I0*/D5       ;SHIFT 3 PLACES
     + /E* I2*/I1*/I0*/D6       ;SHIFT 4 PLACES
     + /E* I2*/I1* I0*/D7       ;SHIFT 5 PLACES
     + /E* I2* I1*/I0*/D0       ;SHIFT 6 PLACES
     + /E* I2* I1* I0*/D1       ;SHIFT 7 PLACES

/Q3 := /E*/I2*/I1*/I0*/D3       ;SHIFT 0 PLACES
     + /E*/I2*/I1* I0*/D4       ;SHIFT 1 PLACES
     + /E*/I2* I1*/I0*/D5       ;SHIFT 2 PLACES
     + /E*/I2* I1* I0*/D6       ;SHIFT 3 PLACES
     + /E* I2*/I1*/I0*/D7       ;SHIFT 4 PLACES
     + /E* I2*/I1* I0*/D0       ;SHIFT 5 PLACES
     + /E* I2* I1*/I0*/D1       ;SHIFT 6 PLACES
     + /E* I2* I1* I0*/D2       ;SHIFT 7 PLACES

/Q4 := /E*/I2*/I1*/I0*/D4       ;SHIFT 0 PLACES
     + /E*/I2*/I1* I0*/D5       ;SHIFT 1 PLACES
     + /E*/I2* I1*/I0*/D6       ;SHIFT 2 PLACES
     + /E*/I2* I1* I0*/D7       ;SHIFT 3 PLACES
     + /E* I2*/I1*/I0*/D0       ;SHIFT 4 PLACES
     + /E* I2*/I1* I0*/D1       ;SHIFT 5 PLACES
     + /E* I2* I1*/I0*/D2       ;SHIFT 6 PLACES
     + /E* I2* I1* I0*/D3       ;SHIFT 7 PLACES
```

```
/Q5 := /E*/I2*/I1*/I0*/D5      ;SHIFT 0 PLACES
     + /E*/I2*/I1* I0*/D6      ;SHIFT 1 PLACES
     + /E*/I2* I1*/I0*/D7      ;SHIFT 2 PLACES
     + /E*/I2* I1* I0*/D0      ;SHIFT 3 PLACES
     + /E* I2*/I1*/I0*/D1      ;SHIFT 4 PLACES
     + /E* I2*/I1* I0*/D2      ;SHIFT 5 PLACES
     + /E* I2* I1*/I0*/D3      ;SHIFT 6 PLACES
     + /E* I2* I1* I0*/D4      ;SHIFT 7 PLACES

/Q6 := /E*/I2*/I1*/I0*/D6      ;SHIFT 0 PLACES
     + /E*/I2*/I1* I0*/D7      ;SHIFT 1 PLACES
     + /E*/I2* I1*/I0*/D0      ;SHIFT 2 PLACES
     + /E*/I2* I1* I0*/D1      ;SHIFT 3 PLACES
     + /E* I2*/I1*/I0*/D2      ;SHIFT 4 PLACES
     + /E* I2*/I1* I0*/D3      ;SHIFT 5 PLACES
     + /E* I2* I1*/I0*/D4      ;SHIFT 6 PLACES
     + /E* I2* I1* I0*/D5      ;SHIFT 7 PLACES

/Q7 := /E*/I2*/I1*/I0*/D7      ;SHIFT 0 PLACES
     + /E*/I2*/I1* I0*/D0      ;SHIFT 1 PLACES
     + /E*/I2* I1*/I0*/D1      ;SHIFT 2 PLACES
     + /E*/I2* I1* I0*/D2      ;SHIFT 3 PLACES
     + /E* I2*/I1*/I0*/D3      ;SHIFT 4 PLACES
     + /E* I2*/I1* I0*/D4      ;SHIFT 5 PLACES
     + /E* I2* I1*/I0*/D5      ;SHIFT 6 PLACES
     + /E* I2* I1* I0*/D6      ;SHIFT 7 PLACES
```

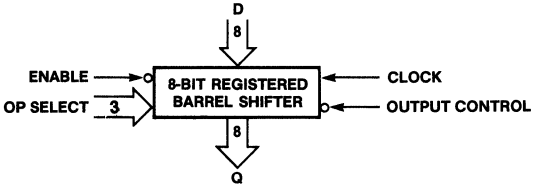
FUNCTION TABLE

CLK /OC E I2 I1 I0 D7 D6 D5 D4 D3 D2 D1 D0 Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0

;CHIP CONTROL			III	DDDDDDDD	QQQQQQQQ	
;CLK	/OC	/E	210	76543210	76543210	COMMENTS
C	L	L	LLL	HLLLLLLL	HLLLLLLL	BARREL SHIFT ONE H 0 PLACES
C	L	L	LLH	HLLLLLLL	LHLLLLLL	BARREL SHIFT ONE H 1 PLACES
C	L	L	LHL	HLLLLLLL	LLHLLLLL	BARREL SHIFT ONE H 2 PLACES
C	L	L	LHH	HLLLLLLL	LLLHLLLL	BARREL SHIFT ONE H 3 PLACES
C	L	L	HLL	HLLLLLLL	LLLLHLLL	BARREL SHIFT ONE H 4 PLACES
C	L	L	HLH	HLLLLLLL	LLLLLHLL	BARREL SHIFT ONE H 5 PLACES
C	L	L	HHL	HLLLLLLL	LLLLLLHL	BARREL SHIFT ONE H 6 PLACES
C	L	L	HHH	HLLLLLLL	LLLLLLLH	BARREL SHIFT ONE H 7 PLACES
C	L	L	LLL	LHHHHHHH	LHHHHHHH	BARREL SHIFT ONE L 0 PLACES
C	L	L	LLH	LHHHHHHH	HLHHHHHH	BARREL SHIFT ONE L 1 PLACES
C	L	L	LHL	LHHHHHHH	HHLHHHHH	BARREL SHIFT ONE L 2 PLACES
C	L	L	LHH	LHHHHHHH	HHHLHHHH	BARREL SHIFT ONE L 3 PLACES
C	L	L	HLL	LHHHHHHH	HHHHLHHH	BARREL SHIFT ONE L 4 PLACES
C	L	L	HLH	LHHHHHHH	HHHHHLHH	BARREL SHIFT ONE L 5 PLACES
C	L	L	HHL	LHHHHHHH	HHHHHHLH	BARREL SHIFT ONE L 6 PLACES
C	L	L	HHH	LHHHHHHH	HHHHHHHL	BARREL SHIFT ONE L 7 PLACES
C	L	H	XXX	XXXXXXXX	HHHHHHHH	PRESET (/E=H)
X	H	X	XXX	XXXXXXXX	ZZZZZZZZ	TEST HI-Z

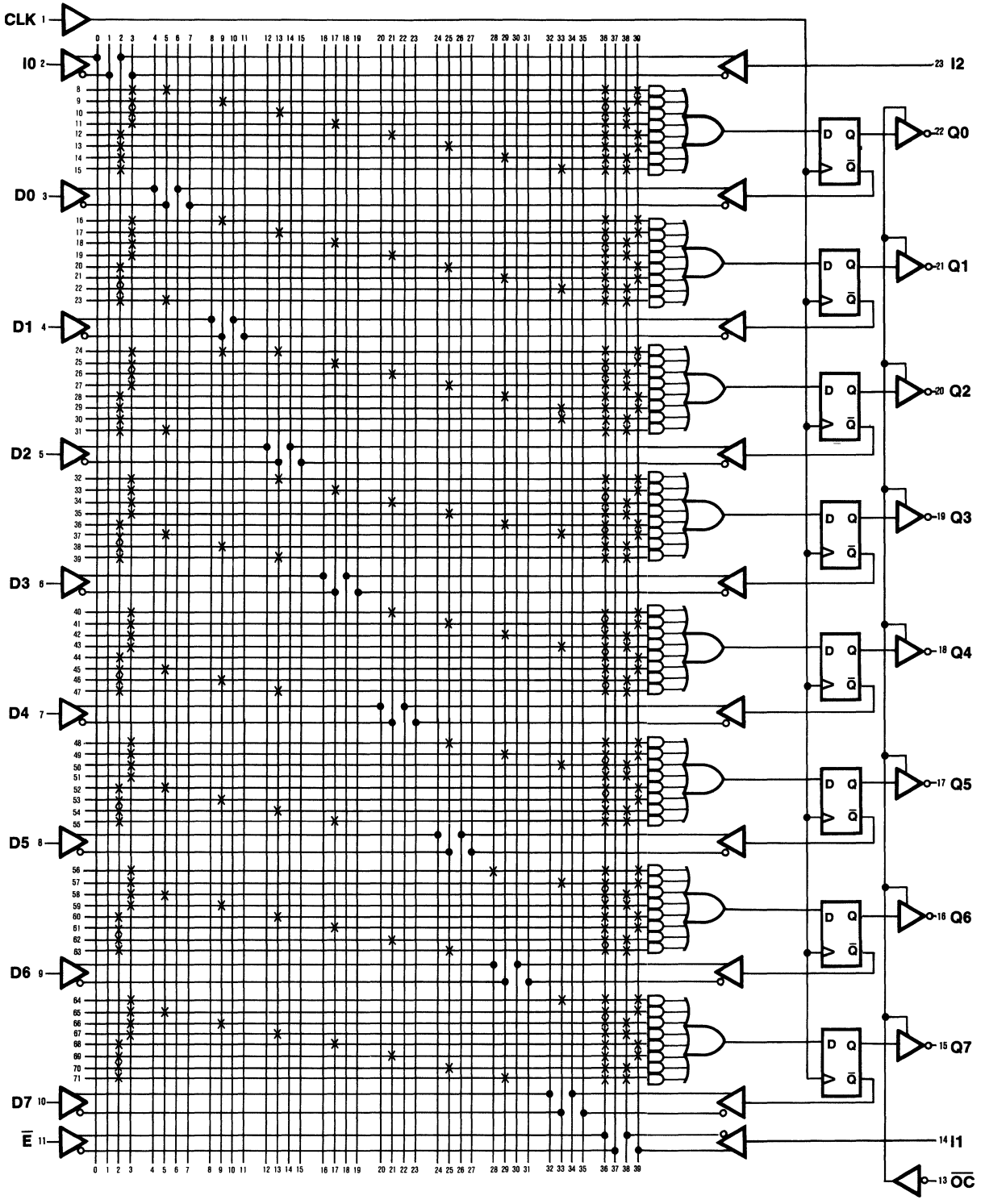
DESCRIPTION

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



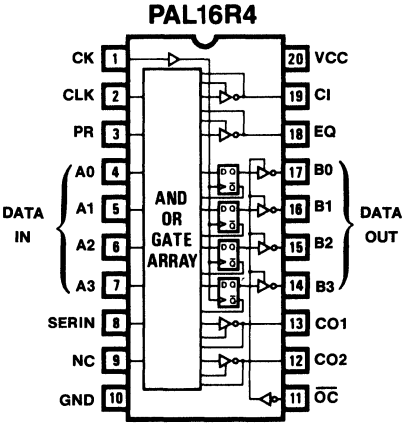
Octal Registered Barrel Shifter

Logic Diagram PAL20R8




IMPORTANT
NOTES

4-Bit
Shift Register/Comparator



6

Functional Description

Frequently it is necessary to take a serial bit stream and convert it to a parallel form for storage. It is also necessary, in some cases, to monitor the same bit stream for certain patterns e.g., floppy disk file headers, RS232 ASCII characters, etc.

Using a PAL it is possible to combine these two functions.

Circuit Description

The circuit shows a 16 bit serial/parallel convertor, with three-state outputs and a compare true output (EQ). Four PAL16R4s are cascaded for this design. There is also a synchronous preset (PR) pin available. The circuit takes positive true Non-Return-To-Zero (NRZ) data with a central positive edge clock and converts it into parallel data. When the A and B inputs are equal, a negative going clock is output, with the negative (leading) edge in the center of the output data.

PAL Implementation

A PAL16R4 is used, the four flip-flops comprising the shift register and the other gates are used to compare the data and output on appropriate pulse if the compare is true. For the shift register the equations are:

B0 := /SERIN*/PR
B1 := /B0*/PR
B2 := /B1*/PR
B3 := /B2*/PR

The inputs are inverted because of the inverting output buffer on each stage. To compare two inputs A and B, the boolean equivalence operator is A ≡ B (or A :*: B).

The equations using AND-OR-INVERT logic are:

A1, A2 ≡ B1, B2 = (/A1*/B1)*(/A2*/B2) + (A1*B1)*(/A2*/B2)
+ (/A1*/B1)*(A2*B2) + (A1*B1)*(A2*B2)

So for 3 inputs we require 8 sum of products and so on. This means that we have to cascade 2 stages in the 16R4 as there are only 7 product terms available per output. Hence, compare 1 (CO1) is:

= A0*B0*A1*B1*/CI
+ A0*B0*/A1*B1*/CI
+ /A0*/B0*A1*B1*/CI
+ /A0*/B0*/A1*/B1*/CI

/CI = COMPARE IN FROM PREVIOUS STAGE

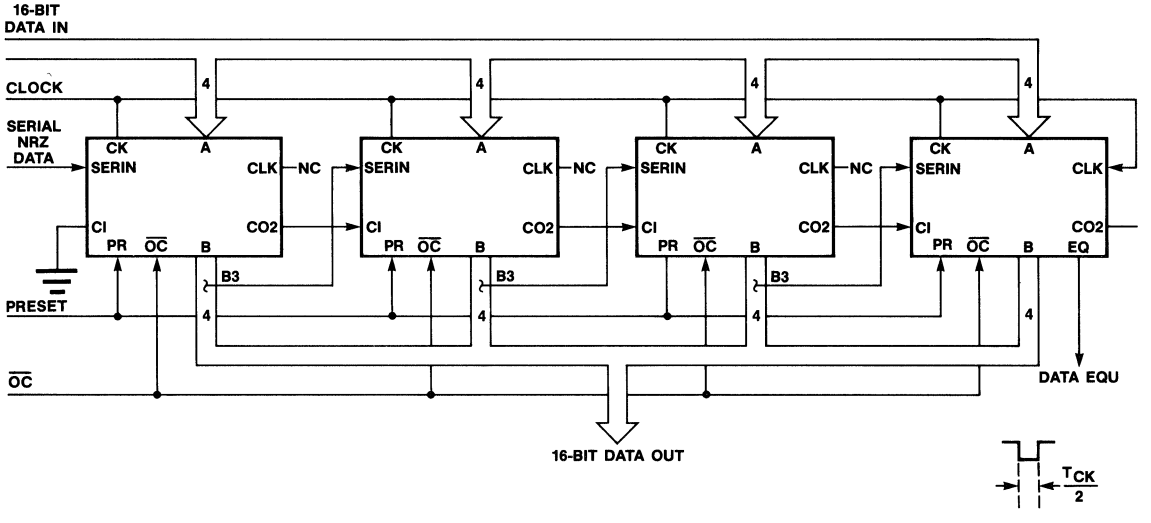
Compare Output 2 (CO2) is:
= A2*B2*A3*B3*/C*1 and so on.

On the final stage, the clock is inverted and gated with CO2. This causes a negative going pulse to be output when the A and B inputs are equal, the leading edge being at data center. This pulse is the EQ output. For a 16 bit register, the final CO2 output can have a maximum delay of 4x2x25ns = 200ns, so for a EQ pulse at data center the maximum clock frequency would be 5 MHz.

But, by using a carry-look-ahead technique, this could be increased to 7.2 MHz. In practice clock frequencies of greater than 10 MHz could be used (causing the EQ pulse leading edge to shift off center).

Conclusion

By feeding the outputs of the shift register into a bank of FIFOs (67401 or similar) a practical serial communication channel can be buffered into a microcomputer system. Using PAL units the design can be tailored for use in different communication systems.



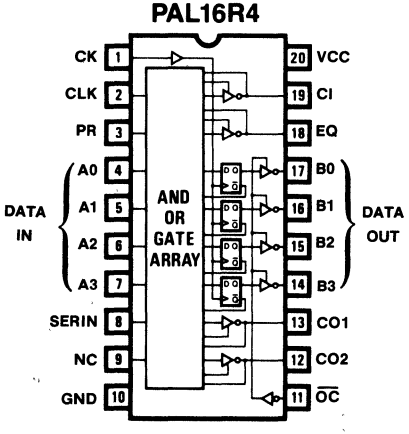
4-BIT SERIAL/PARALLEL SHIFT REGISTER/COMPARATOR WITH PRESET

```
PAL16R4                                          PAL DESIGN SPECIFICATION
SHFT4                                               HARRY HUGHES 02/18/81
4-BIT SHIFT REGISTER/COMPARATOR
MMI ENGLAND
 CK CLK PR  A0 A1 A2 A3 SERIN NC GND
/OC CO2 CO1 B3 B2 B1 B0  EQ   CI VCC


/B0 := /SERIN*/PR                        ;SHIFT LEFT (SERIAL IN)

/B1 := /B0*/PR                           ;SHIFT LEFT

/B2 := /B1*/PR                           ;SHIFT LEFT

/B3 := /B2*/PR                           ;SHIFT LEFT

IF (VCC) /CO1 =  A0* B0* A1* B1*/CI      ;COMPARE  A0= B0 AND  A1= B1
              +  A0* B0*/A1*/B1*/CI      ;COMPARE  A0= B0 AND /A1=/B1
              + /A0*/B0* A1* B1*/CI      ;COMPARE /A0=/B0 AND  A1= B1
              + /A0*/B0*/A1*/B1*/CI      ;COMPARE /A0=/B0 AND /A1=/B1

IF (VCC) /CO2 =  A2* B2* A3* B3*/CO1     ;COMPARE  A2= B2 AND  A3= B3
              +  A2* B2*/A3*/B3*/CO1     ;COMPARE  A2= B2 AND /A3=/B3
              + /A2*/B2* A3* B3*/CO1     ;COMPARE /A2=/B2 AND  A3= B3
              + /A2*/B2*/A3*/B3*/CO1     ;COMPARE /A2=/B2 AND /A3=/B3

IF (VCC)  /EQ = /CLK*/CO2                ;COMPARE TRUE PULSE


FUNCTION TABLE

SERIN A3 A2 A1 A0 CK /OC PR CI CO1 CO2 CLK EQ B3 B2 B1 B0

;-INPUTS-    -------CONTROL-------   OUTPUTS
;SER AAAA    C /  P  C  C  C  C  E    BBBB
;IN  3210    K OC R  I  01 02 LK Q    3210     COMMENTS
-------------------------------------------------------------------------------
  X  XXXX    C L  H  X  X  X  X  X    HHHH     PRESET
  L  HHHL    C L  L  L  L  L  L  L    HHHL     SHIFT LEFT IN A L (A=B)
  L  HHLL    C L  L  L  L  L  L  L    HHLL     SHIFT LEFT IN A L (A=B)
  L  HLLL    C L  L  L  L  L  L  L    HLLL     SHIFT LEFT IN A L (A=B)
  L  LLLL    C L  L  L  L  L  L  L    LLLL     SHIFT LEFT IN A L (A=B)
  H  LLLH    C L  L  L  L  L  L  L    LLLH     SHIFT LEFT IN A H (A=B)
  H  LLHH    C L  L  L  L  L  L  L    LLHH     SHIFT LEFT IN A H (A=B)
  H  LHHH    C L  L  L  L  L  L  L    LHHH     SHIFT LEFT IN A H (A=B)
  H  HHHH    C L  L  L  L  L  L  L    HHHH     SHIFT LEFT IN A H (A=B)
  X  XXXX    L L  X  H  H  H  X  H    XXXX     PREVIOUS STAGE COMPARE NOT TRUE
  X  XXXX    L L  X  X  X  X  H  H    XXXX     COMPARE TRUE PULSE INACTIVE
  H  HHHH    L L  L  L  L  L  L  L    HHHH     COMPARE TRUE
  X  XXXX    X H  X  X  X  X  X  X    ZZZZ     TEST HI-Z
-------------------------------------------------------------------------------
```

DESCRIPTION

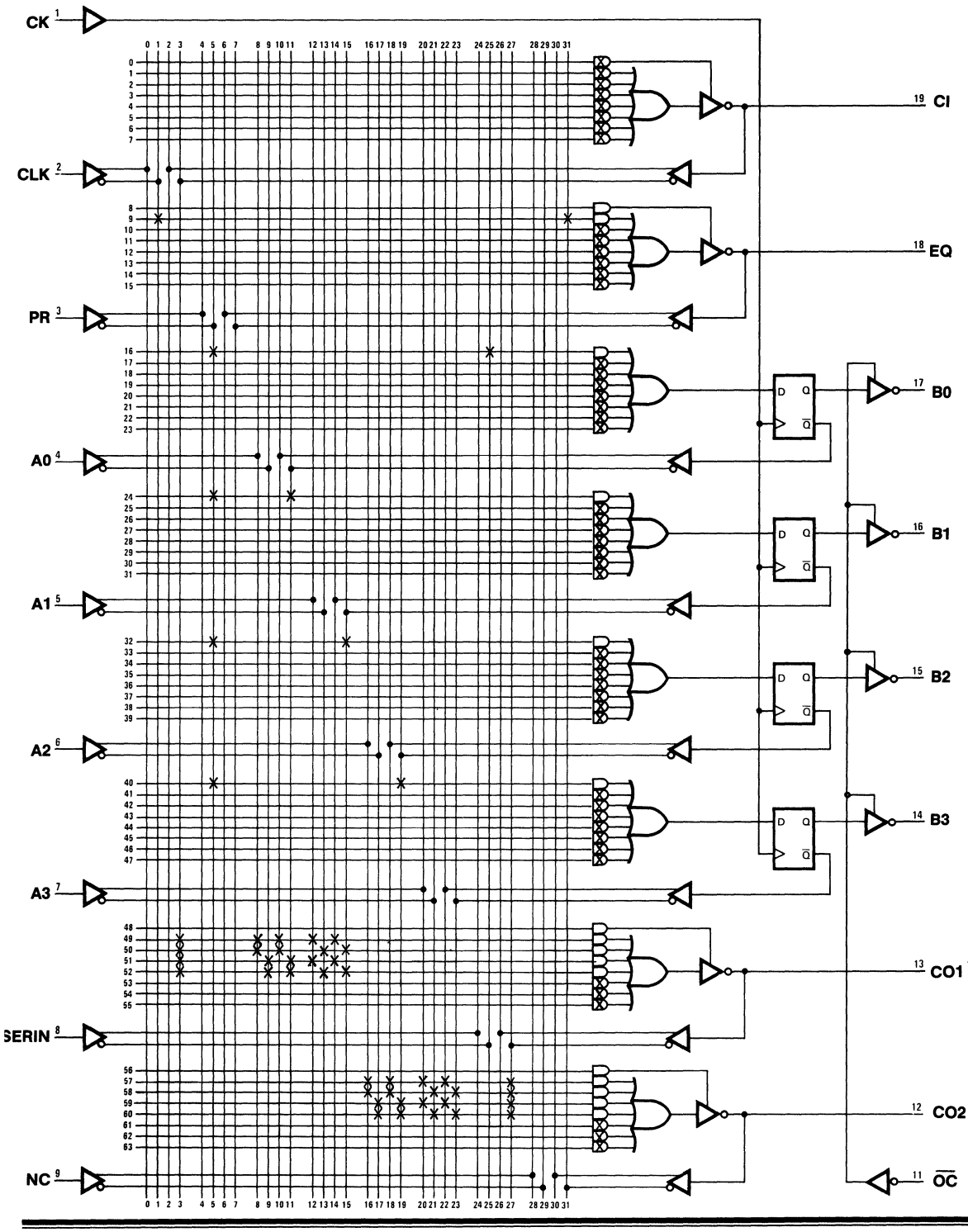
THIS 4-BIT SHIFT REGISTER/COMPARATOR ACCEPTS POSITIVE NRZ INPUT DATA ON THE RISING EDGE OF THE CLOCK (CK) AND PRODUCES A PARALLEL OUTPUT (B) WITH A 'COMPARE TRUE PULSE' ON THE NEGATIVE EDGE OF THE CLOCK (CLK).

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



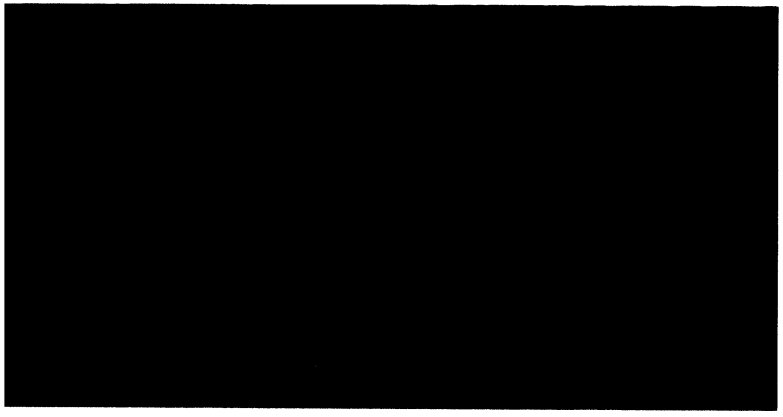
4-Bit Shift Register/Comparator

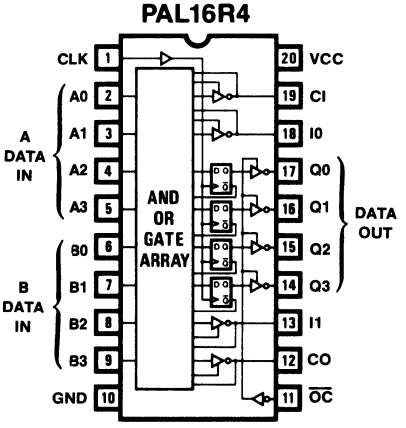
Logic Diagram PAL16R4




IMPORTANT
NOTES

4-Bit Counter with 2 Input Mux





```
PAL16R4                                          PAL DESIGN SPECIFICATION
P7022                                              BIRKNER/COLI 07/22/81
4-BIT COUNTER WITH 2 INPUT MUX
MMI SUNNYVALE, CALIFORNIA
CLK A0 A1 A2 A3 B0 B1 B2 B3 GND
/OC CO I1 Q3 Q2 Q1 Q0 I0 CI VCC


/Q0 := /I1*/I0*/Q0                  ;HOLD Q0 (LSB)
     + /I1* I0*/A0                  ;LOAD A0
     +  I1*/I0*/B0                  ;LOAD B0
     +  I1* I0*/CI*/Q0              ;HOLD Q0 IF NO CARRY IN
     +  I1* I0* CI* Q0              ;COUNT IF CARRY IN AND Q0=H

/Q1 := /I1*/I0*/Q1                  ;HOLD Q1
     + /I1* I0*/A1                  ;LOAD A1
     +  I1*/I0*/B1                  ;LOAD B1
     +  I1* I0*/CI*/Q1              ;HOLD Q1 IF NO CARRY IN
     +  I1* I0*/Q0*/Q1              ;HOLD Q1 IF Q0=L
     +  I1* I0* CI* Q0* Q1          ;COUNT IF CARRY IN AND Q0,Q1=H

/Q2 := /I1*/I0*/Q2                  ;HOLD Q2
     + /I1* I0*/A2                  ;LOAD A2
     +  I1*/I0*/B2                  ;LOAD B2
     +  I1* I0*/CI*/Q2              ;HOLD Q2 IF NO CARRY IN
     +  I1* I0*/Q0*/Q2              ;HOLD Q2 IF Q0=L
     +  I1* I0*/Q1*/Q2              ;HOLD Q2 IF Q1=L
     +  I1* I0* CI* Q0* Q1* Q2      ;COUNT IF CARRY IN AND Q0,Q1,Q2=H

/Q3 := /I1*/I0*/Q3                  ;HOLD Q3 (MSB)
     + /I1* I0*/A3                  ;LOAD A3
     +  I1*/I0*/B3                  ;LOAD B3
     +  I1* I0*/CI*/Q3              ;HOLD Q3 IF NO CARRY IN
     +  I1* I0*/Q0*/Q3              ;HOLD Q3 IF Q0=L
     +  I1* I0*/Q1*/Q3              ;HOLD Q3 IF Q1=L
     +  I1* I0*/Q2*/Q3              ;HOLD Q3 IF Q2=L
     +  I1* I0* CI* Q0* Q1* Q2* Q3  ;COUNT IF CARRY IN AND Q0,Q1,Q2,Q3=H

IF(VCC) /CO = /CI+/Q0+/Q1+/Q2+/Q3   ;CARRY OUT IF CARRY IN AND Q0,Q1,Q2,Q3=H
```

FUNCTION TABLE

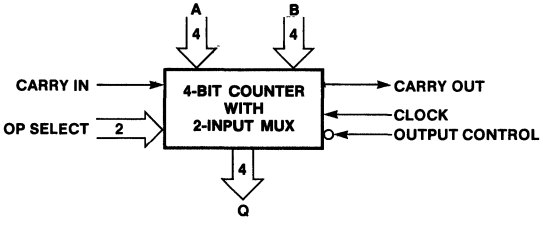
CLK /OC I1 I0 A3 A2 A1 A0 B3 B2 B1 B0 CI CO Q3 Q2 Q1 Q0

;				--INPUTS--				OUTPUT	
;CONTROL		INSTR		AAAA	BBBB	CARRY		QQQQ	COMMENTS
;CLK	/OC	I1	I0	3210	3210	CI	CO	3210	(HEX VALUES)
C	L	L	H	LLLL	HHHH	X	X	LLLL	LOAD A (0)
C	L	H	L	LLLL	HHHH	X	X	HHHH	LOAD B (F)
C	L	L	H	LLLH	LHHH	X	X	LLLH	LOAD A (1)
C	L	H	L	LLLH	LHHH	X	X	LHHH	LOAD B (7)
C	L	L	H	LLHL	HLHH	X	X	LLHL	LOAD A (3)
C	L	H	L	LLHL	HLHH	X	X	HLHH	LOAD B (B)
C	L	L	H	LHLL	HHLH	X	X	LHLL	LOAD A (4)
C	L	H	L	LHLL	HHLH	X	X	HHLH	LOAD B (D)
C	L	L	H	HLLL	HHHL	X	X	HLLL	LOAD A (8)
C	L	H	L	HLLL	HHHL	X	X	HHHL	LOAD B (E)
C	L	L	H	LLLL	HHHH	X	X	LLLL	LOAD A (0)
C	L	H	L	LLLL	HHHH	X	X	HHHH	LOAD B (F)
C	L	H	L	XXXX	LLLH	X	X	LLLH	LOAD B (1)
C	L	H	H	XXXX	XXXX	H	L	LLHL	INCREMENT
C	L	H	L	XXXX	LLHH	X	X	LLHH	LOAD B (3)
C	L	H	H	XXXX	XXXX	H	L	LHLL	INCREMENT
C	L	L	H	LHHH	XXXX	X	X	LHHH	LOAD A (7)
C	L	H	H	XXXX	XXXX	H	L	HLLL	INCREMENT
C	L	L	H	HHHH	XXXX	X	X	HHHH	LOAD A (F)
C	L	H	H	XXXX	XXXX	H	L	LLLL	INCREMENT (ROLL OVER)
C	L	H	L	XXXX	HHLL	X	X	HHLL	LOAD B (C)
C	L	H	H	XXXX	XXXX	H	L	HHLH	INCREMENT (D)
C	L	L	L	XXXX	XXXX	H	L	HHLH	HOLD (D)
C	L	H	H	XXXX	XXXX	H	L	HHHL	INCREMENT (E)
C	L	H	H	XXXX	XXXX	H	H	HHHH	INCREMENT (F) (CARRY OUT)
C	L	H	H	XXXX	XXXX	H	L	LLLL	INCREMENT (0) (ROLL OVER)
C	L	H	H	XXXX	XXXX	H	L	LLLH	INCREMENT (1)
C	L	H	H	XXXX	XXXX	L	L	LLLH	HOLD (NO CARRY IN)
C	L	H	H	XXXX	XXXX	H	L	LLHL	INCREMENT (2)
X	H	X	X	XXXX	XXXX	X	X	ZZZZ	TEST HI-Z

DESCRIPTION

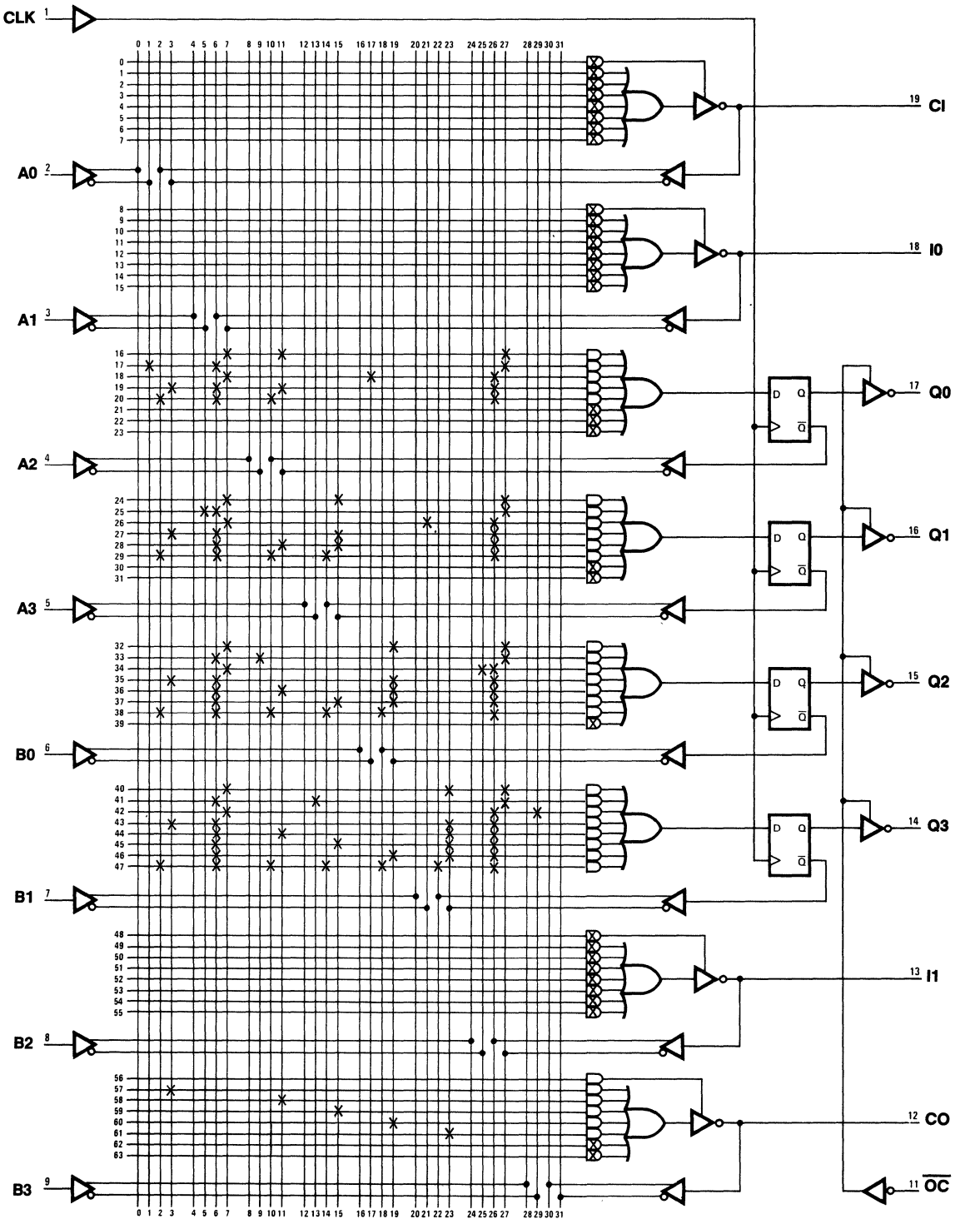
THE 4-BIT COUNTER LOADS A OR B FROM THE MUX, INCREMENTS, OR HOLDS ON THE RISING EDGE OF THE CLOCK.

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



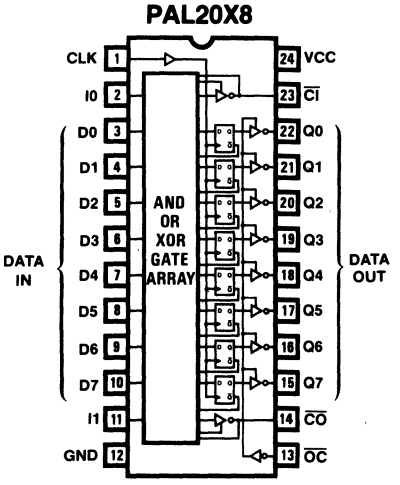
4-Bit Counter with 2 Input Mux

Logic Diagram PAL16R4




IMPORTANT NOTES

Octal Counter



6

```
PAL20X8                                          PAL DESIGN SPECIFICATION
74LS461                                        BIRKNER/KAZMI/BLASCO 02/10/81
OCTAL COUNTER
MMI SUNNYVALE, CALIFORNIA
CLK  I0 D0 D1 D2 D3 D4 D5 D6 D7  I1 GND
/OC /CO Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0 /CI VCC


/Q0 :=  /I1*/I0                              ;CLEAR LSB
     +       I0*/Q0                          ;COUNT/HOLD
    :+:  I1*/I0*/D0                          ;LOAD D0 (LSB)
     +   I1* I0* CI                          ;COUNT

/Q1 :=  /I1*/I0                              ;CLEAR
     +       I0*/Q1                          ;COUNT/HOLD
    :+:  I1*/I0*/D1                          ;LOAD D1
     +   I1* I0* CI*Q0                       ;COUNT

/Q2 :=  /I1*/I0                              ;CLEAR
     +       I0*/Q2                          ;COUNT/HOLD
    :+:  I1*/I0*/D2                          ;LOAD D2
     +   I1* I0* CI*Q0*Q1                    ;COUNT

/Q3 :=  /I1*/I0                              ;CLEAR
     +       I0*/Q3                          ;COUNT/HOLD
    :+:  I1*/I0*/D3                          ;LOAD D3
     +   I1* I0* CI*Q0*Q1*Q2                 ;COUNT

/Q4 :=  /I1*/I0                              ;CLEAR
     +       I0*/Q4                          ;COUNT/HOLD
    :+:  I1*/I0*/D4                          ;LOAD D4
     +   I1* I0* CI*Q0*Q1*Q2*Q3              ;COUNT

/Q5 :=  /I1*/I0                              ;CLEAR
     +       I0*/Q5                          ;COUNT/HOLD
    :+:  I1*/I0*/D5                          ;LOAD D5
     +   I1* I0* CI*Q0*Q1*Q2*Q3*Q4           ;COUNT

/Q6 :=  /I1*/I0                              ;CLEAR
     +       I0*/Q6                          ;COUNT/HOLD
    :+:  I1*/I0*/D6                          ;LOAD D6
     +   I1* I0* CI*Q0*Q1*Q2*Q3*Q4*Q5        ;COUNT

/Q7 :=  /I1*/I0                              ;CLEAR MSB
     +       I0*/Q7                          ;COUNT/HOLD
    :+:  I1*/I0*/D7                          ;LOAD D7 (MSB)
     +   I1* I0* CI*Q0*Q1*Q2*Q3*Q4*Q5*Q6     ;COUNT

IF (VCC)  CO = CI*Q0*Q1*Q2*Q3*Q4*Q5*Q6*Q7    ;CARRY OUT
```

FUNCTION TABLE

CLK /OC I1 I0 D7 D6 D5 D4 D3 D2 D1 D0 /CI /CO Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0

;CONTROL		INSTR		-INPUT-- DDDDDDDD	CARRY		-OUTPUT- QQQQQQQQ	COMMENTS
;CLK	/OC	I1	I0	76543210	/CI	/CO	76543210	(HEX VALUES)
C	L	H	L	LLLLLLLH	X	H	LLLLLLLH	LOAD (01)
C	L	H	H	XXXXXXXX	L	H	LLLLLLHL	INCREMENT
C	L	H	L	LLLLLLHH	X	H	LLLLLLHH	LOAD (03)
C	L	H	H	XXXXXXXX	L	H	LLLLLHLL	INCREMENT
C	L	H	L	LLLLLHHH	X	H	LLLLLHHH	LOAD (07)
C	L	H	H	XXXXXXXX	L	H	LLLLHLLL	INCREMENT
C	L	H	L	LLLLHHHH	X	H	LLLLHHHH	LOAD (0F)
C	L	H	H	XXXXXXXX	L	H	LLLHLLLL	INCREMENT
C	L	H	L	LLLHHHHH	X	H	LLLHHHHH	LOAD (1F)
C	L	H	H	XXXXXXXX	L	H	LLHLLLLL	INCREMENT
C	L	H	L	LLHHHHHH	X	H	LLHHHHHH	LOAD (3F)
C	L	H	H	XXXXXXXX	L	H	LHLLLLLL	INCREMENT
C	L	H	L	LHHHHHHH	X	H	LHHHHHHH	LOAD (7F)
C	L	H	H	XXXXXXXX	L	H	HLLLLLLL	INCREMENT
C	L	H	L	HHHHHHHH	L	L	HHHHHHHH	LOAD (FF)
C	L	H	H	XXXXXXXX	L	H	LLLLLLLL	INCREMENT (ROLL OVER)
C	L	H	L	HHHHHHHH	L	L	HHHHHHHH	LOAD (FF)
C	L	H	L	HHHHHHHL	X	H	HHHHHHHL	LOAD (FE)
C	L	H	L	HHHHHHLH	X	H	HHHHHHLH	LOAD (FD)
C	L	H	L	HHHHHLHH	X	H	HHHHHLHH	LOAD (FB)
C	L	H	L	HHHHLHHH	X	H	HHHHLHHH	LOAD (F7)
C	L	H	L	HHHLHHHH	X	H	HHHLHHHH	LOAD (EF)
C	L	H	L	HHLHHHHH	X	H	HHLHHHHH	LOAD (DF)
C	L	H	L	HLHHHHHH	X	H	HLHHHHHH	LOAD (BF)
C	L	H	L	LHHHHHHH	X	H	LHHHHHHH	LOAD (7F)
C	L	H	L	HHHHHHHH	L	L	HHHHHHHH	LOAD (FF)
C	L	L	L	XXXXXXXX	X	H	LLLLLLLL	CLEAR
C	L	H	H	XXXXXXXX	L	H	LLLLLLLH	INCREMENT TO (01)
C	L	H	H	XXXXXXXX	L	H	LLLLLLHL	INCREMENT TO (02)
C	L	H	H	XXXXXXXX	L	H	LLLLLLHH	INCREMENT TO (03)
C	L	H	H	XXXXXXXX	L	H	LLLLLHLL	INCREMENT TO (04)
C	L	H	H	XXXXXXXX	L	H	LLLLLHLH	INCREMENT TO (05)
C	L	H	H	XXXXXXXX	L	H	LLLLLHHL	INCREMENT TO (06)
C	L	H	H	XXXXXXXX	L	H	LLLLLHHH	INCREMENT TO (07)
C	L	H	H	XXXXXXXX	L	H	LLLLHLLL	INCREMENT TO (08)
C	L	H	H	XXXXXXXX	L	H	LLLLHLLH	INCREMENT TO (09)
C	L	H	H	XXXXXXXX	L	H	LLLLHLHL	INCREMENT TO (0A)
C	L	H	H	XXXXXXXX	L	H	LLLLHLHH	INCREMENT TO (0B)
C	L	H	H	XXXXXXXX	L	H	LLLLHHLL	INCREMENT TO (0C)
C	L	H	L	HHHHHHHL	X	H	HHHHHHHL	LOAD (FE)
C	L	H	H	XXXXXXXX	L	L	HHHHHHHH	INCREMENT TO (FF) /CO=L
C	L	H	H	XXXXXXXX	H	H	HHHHHHHH	CI INHIBITS COUNT AND CO
C	L	L	H	LLLLLLLL	L	L	HHHHHHHH	HOLD SEL INHIBITS COUNT ONLY
C	L	H	H	HHHHHHHH	L	H	LLLLLLLL	INCREMENT TO (00)
X	H	X	X	XXXXXXXX	X	X	ZZZZZZZZ	TEST HI-Z

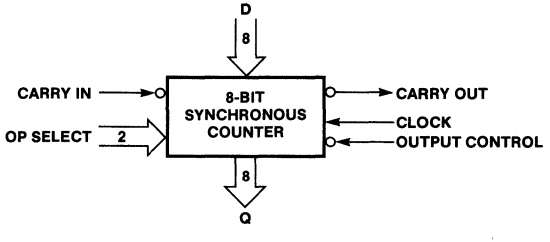
DESCRIPTION

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

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

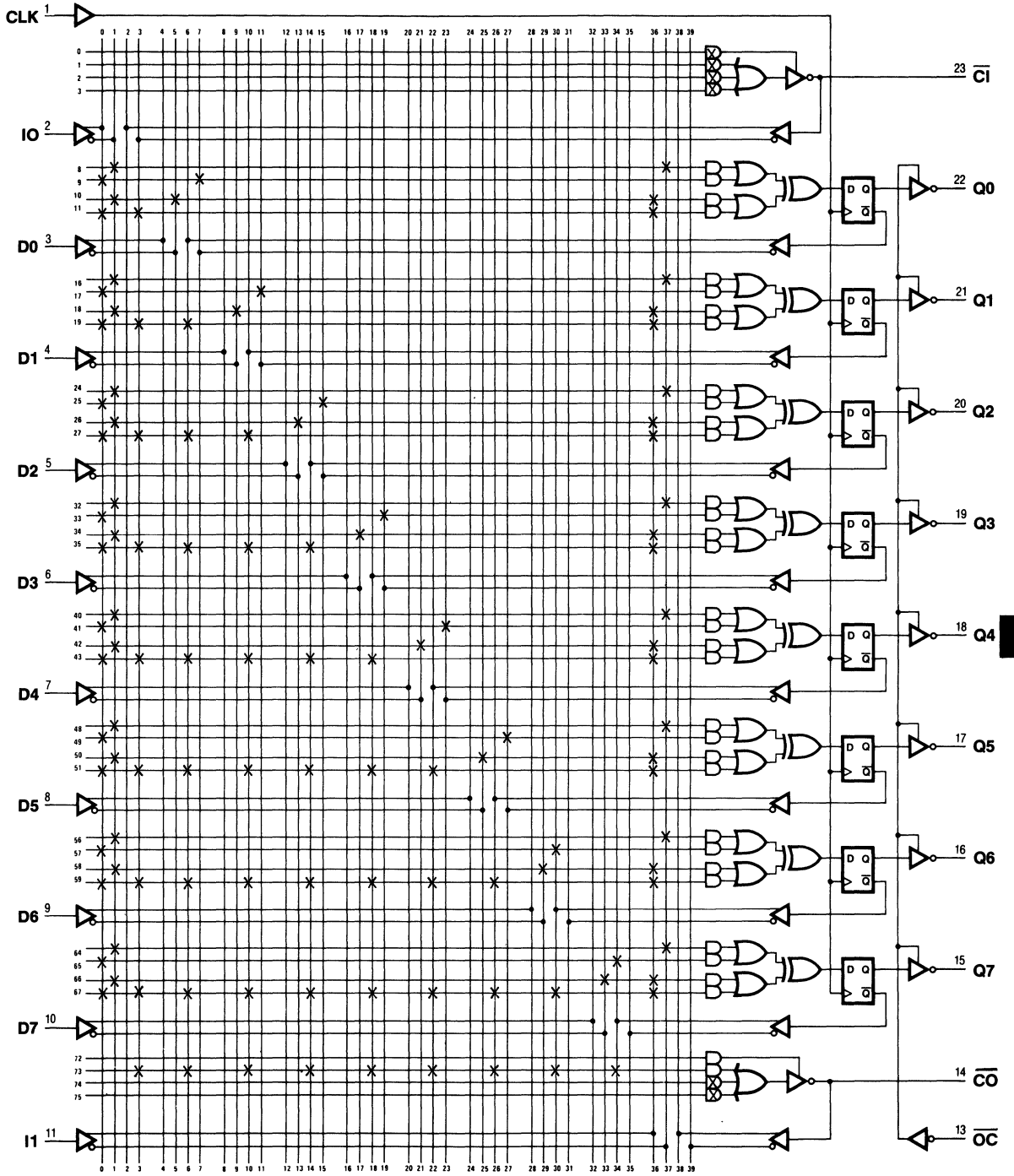
/OC	CLK	I1	I0	/CI	D7-D0	Q7-Q0	OPERATION
H	X	X	X	X	X	Z	HI-Z
L	C	L	L	X	X	L	CLEAR
L	C	L	H	X	X	Q	HOLD
L	C	H	L	X	D	D	LOAD
L	C	H	H	H	X	Q	HOLD
L	C	H	H	L	X	Q PLUS 1	INCREMENT

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



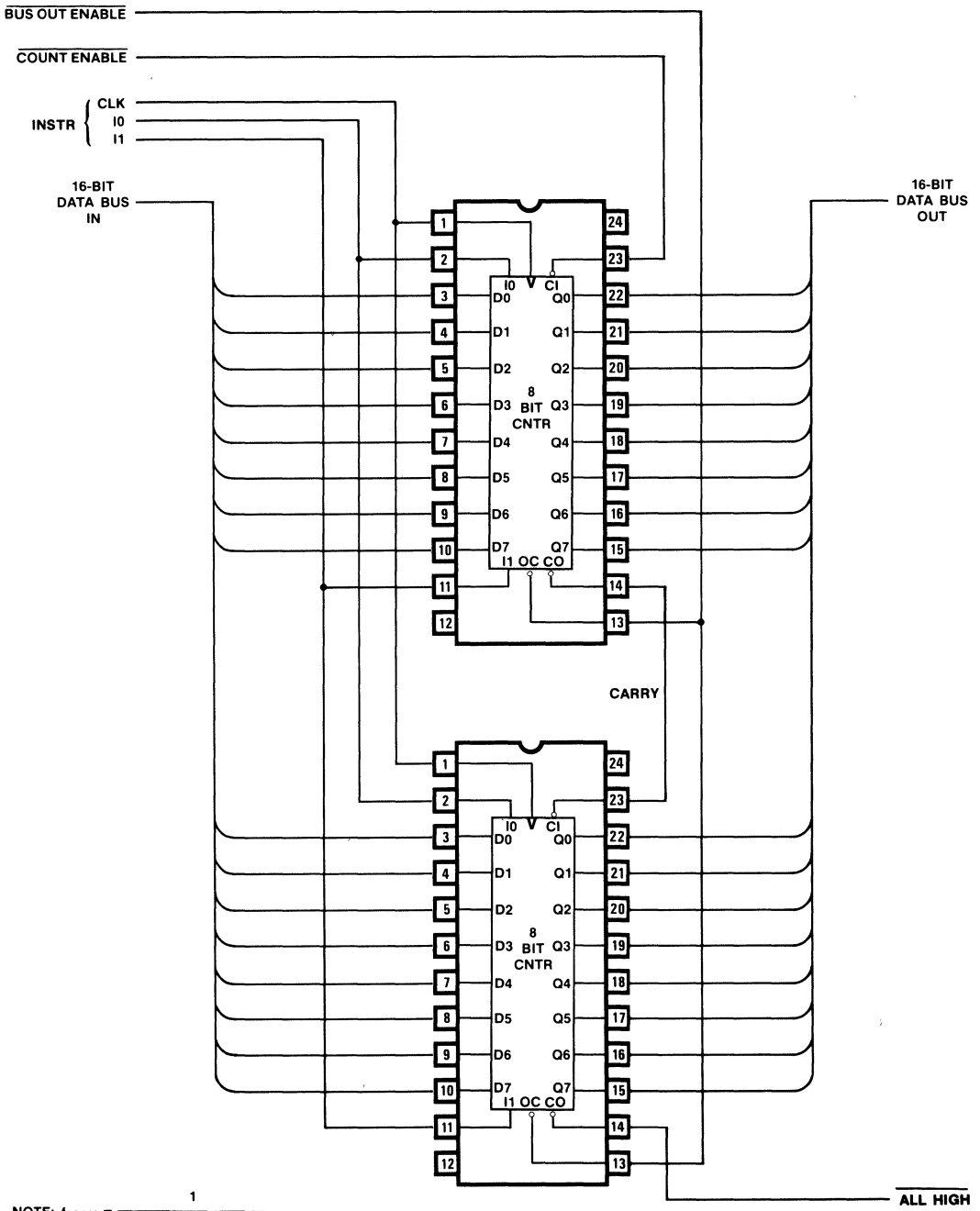
Octal Counter

Logic Diagram PAL20X8



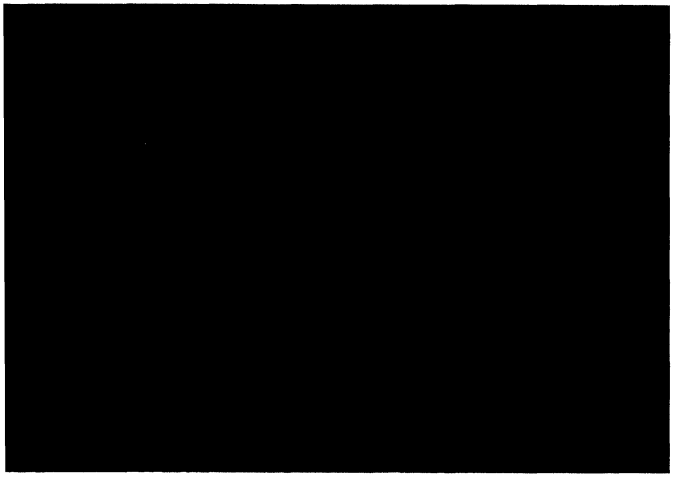
Application

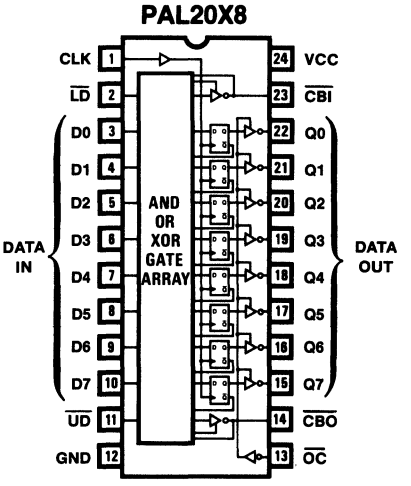
16-Bit Counter



NOTE: $f_{MAX} = \frac{1}{t_{PD\ CLK\ TO\ CO} + t_{SU}}$

Octal Up/Down Counter





```
PAL20X8                                          PAL DESIGN SPECIFICATION
PMSI401                                             VINCENT COLI 07/24/81
OCTAL UP/DOWN COUNTER
MMI SUNNYVALE, CALIFORNIA
CLK /LD  D0 D1 D2 D3 D4 D5 D6 D7 /UD  GND
/OC /CBO Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0 /CBI VCC


/Q0 :=  /LD*/Q0                                          ;HOLD Q0
     +   LD*/D0                                          ;LOAD D0 (LSB)
    :+: /LD* UD* CBI                                     ;INCREMENT
     +  /LD*/UD* CBI                                     ;DECREMENT

/Q1 :=  /LD*/Q1                                          ;HOLD Q1
     +   LD*/D1                                          ;LOAD D1
    :+: /LD* UD* CBI* Q0                                 ;INCREMENT
     +  /LD*/UD* CBI*/Q0                                 ;DECREMENT

/Q2 :=  /LD*/Q2                                          ;HOLD Q2
     +   LD*/D2                                          ;LOAD D2
    :+: /LD* UD* CBI* Q0* Q1                             ;INCREMENT
     +  /LD*/UD* CBI*/Q0*/Q1                             ;DECREMENT

/Q3 :=  /LD*/Q3                                          ;HOLD Q3
     +   LD*/D3                                          ;LOAD D3
    :+: /LD* UD* CBI* Q0* Q1* Q2                         ;INCREMENT
     +  /LD*/UD* CBI*/Q0*/Q1*/Q2                         ;DECREMENT

/Q4 :=  /LD*/Q4                                          ;HOLD Q4
     +   LD*/D4                                          ;LOAD D4
    :+: /LD* UD* CBI* Q0* Q1* Q2* Q3                     ;INCREMENT
     +  /LD*/UD* CBI*/Q0*/Q1*/Q2*/Q3                     ;DECREMENT

/Q5 :=  /LD*/Q5                                          ;HOLD Q5
     +   LD*/D5                                          ;LOAD D5
    :+: /LD* UD* CBI* Q0* Q1* Q2* Q3* Q4                 ;INCREMENT
     +  /LD*/UD* CBI*/Q0*/Q1*/Q2*/Q3*/Q4                 ;DECREMENT

/Q6 :=  /LD*/Q6                                          ;HOLD Q6
     +   LD*/D6                                          ;LOAD D6
    :+: /LD* UD* CBI* Q0* Q1* Q2* Q3* Q4* Q5             ;INCREMENT
     +  /LD*/UD* CBI*/Q0*/Q1*/Q2*/Q3*/Q4*/Q5             ;DECREMENT

/Q7 :=  /LD*/Q7                                          ;HOLD Q7
     +   LD*/D7                                          ;LOAD D7 (MSB)
    :+: /LD* UD* CBI* Q0* Q1* Q2* Q3* Q4* Q5* Q6         ;INCREMENT
     +  /LD*/UD* CBI*/Q0*/Q1*/Q2*/Q3*/Q4*/Q5*/Q6         ;DECREMENT

IF(VCC) CBO =  UD* CBI* Q0* Q1* Q2* Q3* Q4* Q5* Q6* Q7   ;CARRY OUT
           + /UD* CBI*/Q0*/Q1*/Q2*/Q3*/Q4*/Q5*/Q6*/Q7   ;BORROW OUT
```

FUNCTION TABLE

CLK /OC /LD /UD D7 D6 D5 D4 D3 D2 D1 D0 /CBI /CBO Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0

;				-INPUTS-			OUTPUTS	
;----CONTROL----				DDDDDDDD	CARR/BORR		QQQQQQQQ	COMMENTS
;CLK	/OC	/LD	/UD	76543210	/CBI	/CBO	76543210	(HEX VALUES)
C	L	L	X	LLLLLLLH	X	X	LLLLLLLH	LOAD (01)
C	L	H	L	XXXXXXXX	L	H	LLLLLLHL	INCREMENT
C	L	H	H	XXXXXXXX	L	H	LLLLLLLH	DECREMENT
C	L	L	X	LLLLLLHH	X	X	LLLLLLHH	LOAD (03)
C	L	H	L	XXXXXXXX	L	H	LLLLLHLL	INCREMENT
C	L	H	H	XXXXXXXX	L	H	LLLLLLHH	DECREMENT
C	L	L	X	LLLLLHHH	X	X	LLLLLHHH	LOAD (07)
C	L	H	L	XXXXXXXX	L	H	LLLLHLLL	INCREMENT
C	L	H	H	XXXXXXXX	L	H	LLLLLHHH	DECREMENT
C	L	L	X	LLLLHHHH	X	X	LLLLHHHH	LOAD (0F)
C	L	H	L	XXXXXXXX	L	H	LLLHLLLL	INCREMENT
C	L	H	H	XXXXXXXX	L	H	LLLLHHHH	DECREMENT
C	L	L	X	LLLHHHHH	X	X	LLLHHHHH	LOAD (1F)
C	L	H	L	XXXXXXXX	L	H	LLHLLLLL	INCREMENT
C	L	H	H	XXXXXXXX	L	H	LLLHHHHH	DECREMENT
C	L	L	X	LLHHHHHH	X	X	LLHHHHHH	LOAD (3F)
C	L	H	L	XXXXXXXX	L	H	LHLLLLLL	INCREMENT
C	L	H	H	XXXXXXXX	L	H	LLHHHHHH	DECREMENT
C	L	L	X	LHHHHHHH	X	X	LHHHHHHH	LOAD (7F)
C	L	H	L	XXXXXXXX	L	H	HLLLLLLL	INCREMENT
C	L	H	H	XXXXXXXX	L	H	LHHHHHHH	DECREMENT
C	L	L	X	HHHHHHHH	X	X	HHHHHHHH	LOAD (FF)
C	L	L	X	LHHHHHHH	X	X	LHHHHHHH	LOAD (7F)
C	L	L	X	HLHHHHHH	X	X	HLHHHHHH	LOAD (BF)
C	L	L	X	HHLHHHHH	X	X	HHLHHHHH	LOAD (DF)
C	L	L	X	HHHLHHHH	X	X	HHHLHHHH	LOAD (EF)
C	L	L	X	HHHHLHHH	X	X	HHHHLHHH	LOAD (F7)
C	L	L	X	HHHHHLHH	X	X	HHHHHLHH	LOAD (FB)
C	L	L	X	HHHHHHLH	X	X	HHHHHHLH	LOAD (FD)
C	L	L	X	HHHHHHHL	X	X	HHHHHHHL	LOAD (FE)
C	L	H	L	XXXXXXXX	L	L	HHHHHHHH	INCREMENT TO (FF) (/CBO=L)
C	L	H	L	XXXXXXXX	L	H	LLLLLLLL	INCREMENT TO (00) (ROLL OVER)
C	L	H	L	XXXXXXXX	L	H	LLLLLLLH	INCREMENT TO (01)
C	L	H	L	XXXXXXXX	L	H	LLLLLLHL	INCREMENT TO (02)
C	L	H	L	XXXXXXXX	H	H	LLLLLLHL	HOLD
C	L	H	L	XXXXXXXX	L	H	LLLLLLHH	INCREMENT TO (03)
C	L	H	H	XXXXXXXX	L	H	LLLLLLHL	DECREMENT TO (02)
C	L	H	H	XXXXXXXX	L	H	LLLLLLLH	DECREMENT TO (01)
C	L	H	H	XXXXXXXX	L	L	LLLLLLLL	DECREMENT TO (00) (/CBO=L)
C	L	H	H	XXXXXXXX	L	H	HHHHHHHH	DECREMENT TO (FF) (ROLL UNDER)
C	L	H	H	XXXXXXXX	L	H	HHHHHHHL	DECREMENT TO (FE)
C	L	H	H	XXXXXXXX	L	H	HHHHHHLH	DECREMENT TO (FD)
C	L	H	H	XXXXXXXX	H	H	HHHHHHLH	HOLD
C	L	H	H	XXXXXXXX	L	H	HHHHHHLL	DECREMENT TO (FC)
X	H	X	X	XXXXXXXX	X	X	ZZZZZZZZ	TEST HI-Z

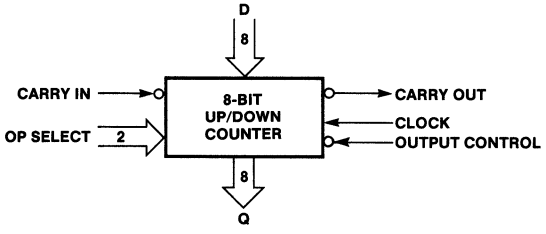
DESCRIPTION

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

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

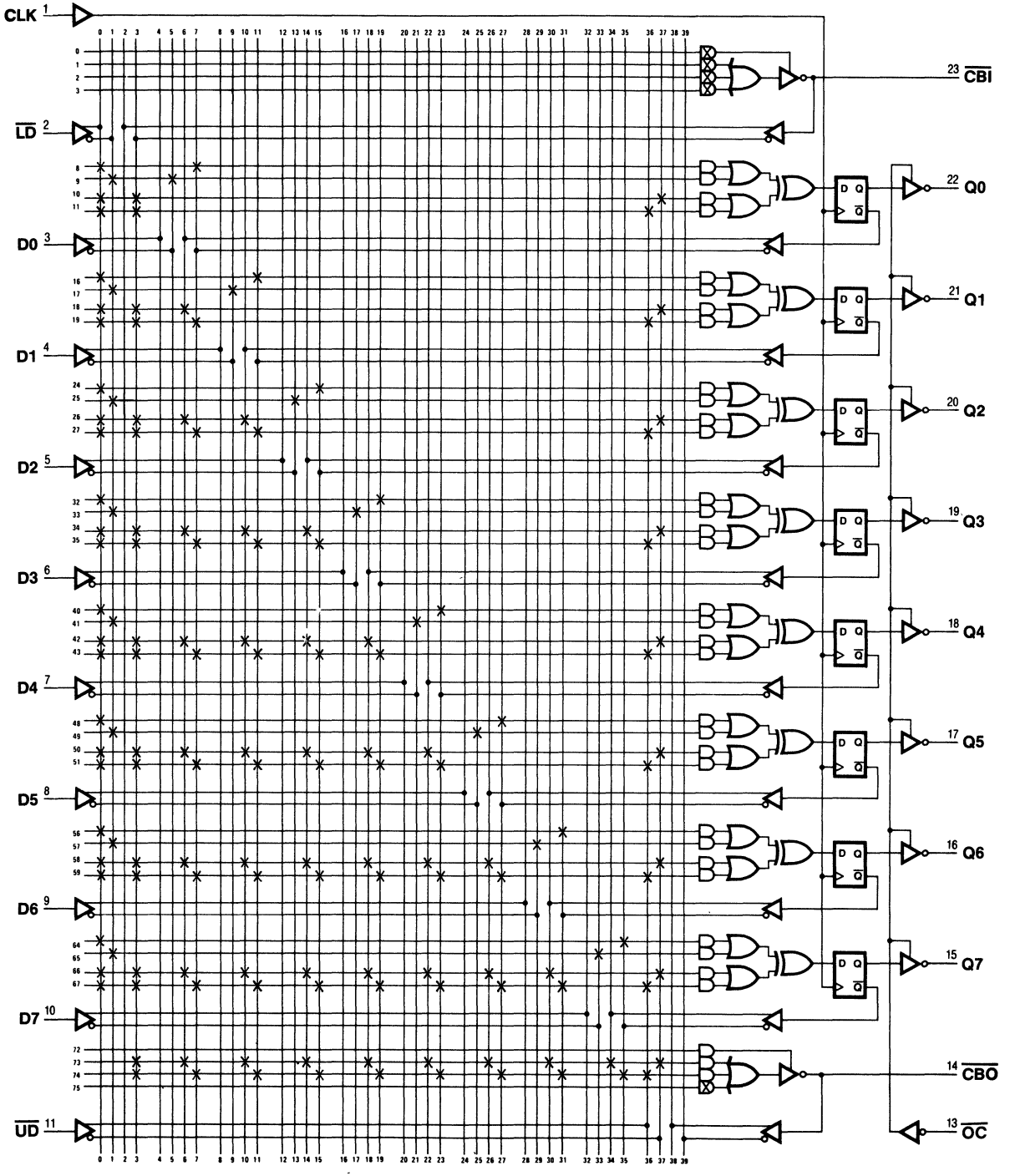
/OC	CLK	/LD	/UD	/CBI	D7-D0	Q7-Q0	OPERATION
H	X	X	X	X	X	Z	HI-Z
L	C	L	X	X	D	L	LOAD
L	C	H	L	H	X	Q	HOLD
L	C	H	L	L	X	Q PLUS 1	INCREMENT
L	C	H	H	H	X	Q	HOLD
L	C	H	H	L	X	Q MINUS 1	DECREMENT

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



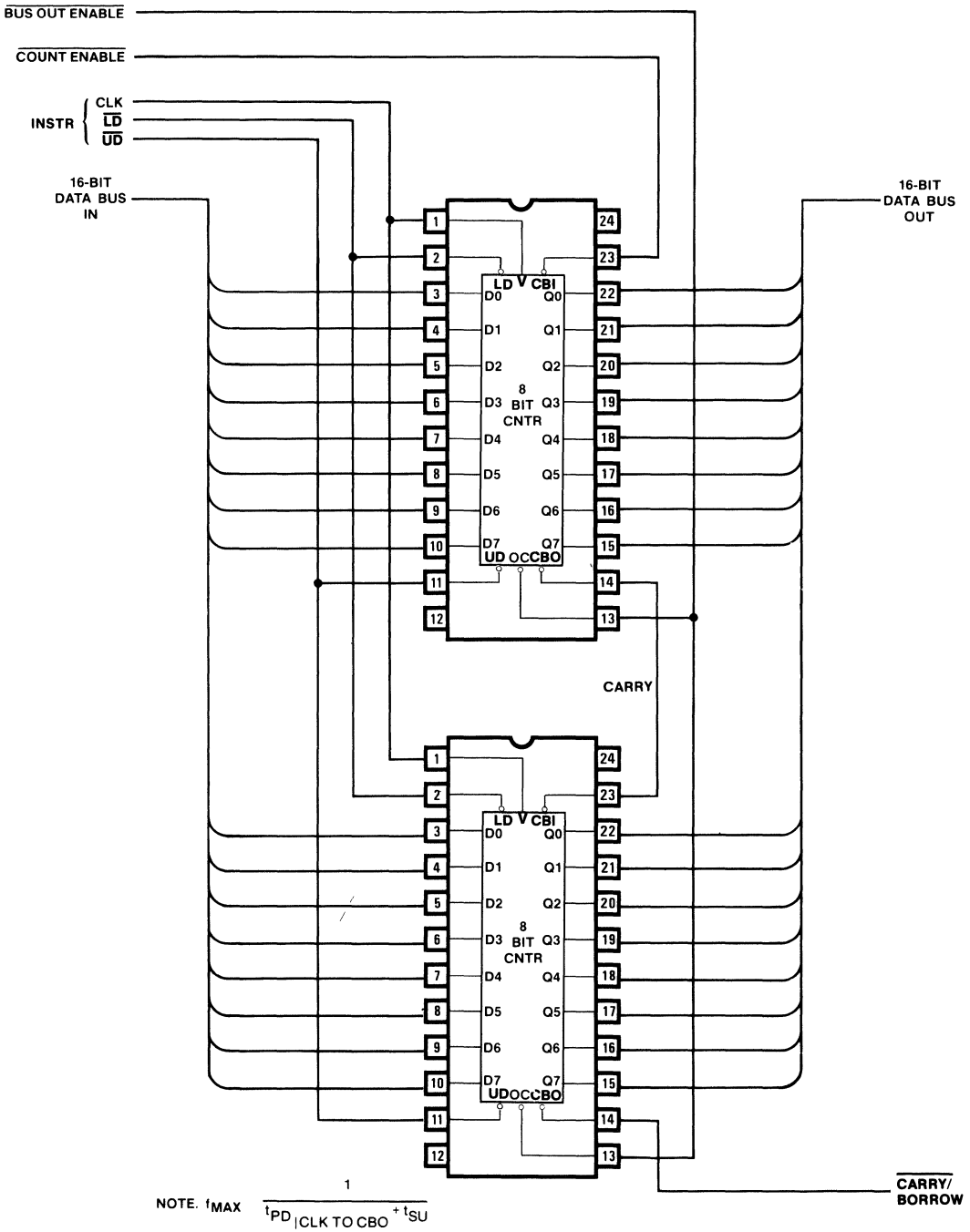
Octal Up/Down Counter

Logic Diagram PAL20X8

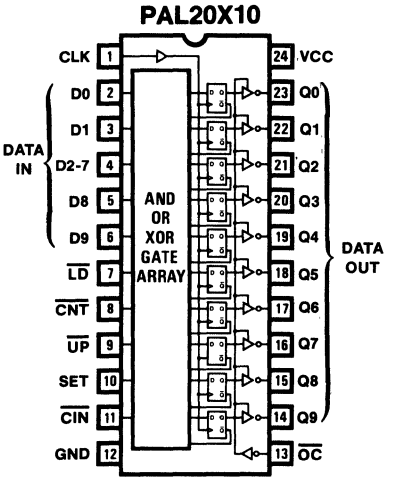


Application

16-Bit Up/Down Counter



10-Bit Counter



```
PAL20X10                                                PAL DESIGN SPECIFICATION
74LS491                                                   JOHN BIRKNER 04/01/81
10-BIT COUNTER
MMI SUNNYVALE, CALIFORNIA
CLK D0 D1 D2-7 D8 D9 /LD /CNT /UP SET /CIN GND
/OC Q9 Q8 Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0 VCC

/Q0 :=  /SET* LD*/D0                                              ;LOAD D0
     +  /SET*/LD*/Q0                                              ;HOLD (LSB)
    :+: /SET*/LD* CNT* CIN* UP                                    ;INCREMENT
     +  /SET*/LD* CNT* CIN*/UP                                    ;DECREMENT

/Q1 :=  /SET* LD*/D1                                              ;LOAD D1
     +  /SET*/LD*/Q1                                              ;HOLD
    :+: /SET*/LD* CNT* CIN* UP* Q0                                ;INCREMENT
     +  /SET*/LD* CNT* CIN*/UP*/Q0                                ;DECREMENT

/Q2 :=  /SET* LD*/D2-7                                            ;LOAD D2-7
     +  /SET*/LD*/Q2                                              ;HOLD
    :+: /SET*/LD* CNT* CIN* UP* Q0* Q1                            ;INCREMENT
     +  /SET*/LD* CNT* CIN*/UP*/Q0*/Q1                            ;DECREMENT

/Q3 :=  /SET* LD*/D2-7                                            ;LOAD D2-7
     +  /SET*/LD*/Q3                                              ;HOLD
    :+: /SET*/LD* CNT* CIN* UP* Q0* Q1* Q2                        ;INCREMENT
     +  /SET*/LD* CNT* CIN*/UP*/Q0*/Q1*/Q2                        ;DECREMENT

/Q4 :=  /SET* LD*/D2-7                                            ;LOAD D2-7
     +  /SET*/LD*/Q4                                              ;HOLD
    :+: /SET*/LD* CNT* CIN* UP* Q0* Q1* Q2* Q3                    ;INCREMENT
     +  /SET*/LD* CNT* CIN*/UP*/Q0*/Q1*/Q2*/Q3                    ;DECREMENT

/Q5 :=  /SET* LD*/D2-7                                            ;LOAD D2-7
     +  /SET*/LD*/Q5                                              ;HOLD
    :+: /SET*/LD* CNT* CIN* UP* Q0* Q1* Q2* Q3* Q4                ;INCREMENT
     +  /SET*/LD* CNT* CIN*/UP*/Q0*/Q1*/Q2*/Q3*/Q4                ;DECREMENT

/Q6 :=  /SET* LD*/D2-7                                            ;LOAD D2-7
     +  /SET*/LD*/Q6                                              ;HOLD
    :+: /SET*/LD* CNT* CIN* UP* Q0* Q1* Q2* Q3* Q4* Q5            ;INCREMENT
     +  /SET*/LD* CNT* CIN*/UP*/Q0*/Q1*/Q2*/Q3*/Q4*/Q5            ;DECREMENT

/Q7 :=  /SET* LD*/D2-7                                            ;LOAD D2-7
     +  /SET*/LD*/Q7                                              ;HOLD
    :+: /SET*/LD* CNT* CIN* UP* Q0* Q1* Q2* Q3* Q4* Q5* Q6        ;INCREMENT
     +  /SET*/LD* CNT* CIN*/UP*/Q0*/Q1*/Q2*/Q3*/Q4*/Q5*/Q6        ;DECREMENT

/Q8 :=  /SET* LD*/D8                                              ;LOAD D8
     +  /SET*/LD*/Q8                                              ;HOLD
    :+: /SET*/LD* CNT* CIN* UP* Q0* Q1* Q2* Q3* Q4* Q5* Q6* Q7    ;INCREMENT
     +  /SET*/LD* CNT* CIN*/UP*/Q0*/Q1*/Q2*/Q3*/Q4*/Q5*/Q6*/Q7    ;DECREMENT

/Q9 :=  /SET* LD*/D9                                              ;LOAD D9
     +  /SET*/LD*/Q9                                              ;HOLD (MSB)
    :+: /SET*/LD* CNT* CIN* UP* Q0* Q1* Q2* Q3* Q4* Q5* Q6* Q7* Q8 ;INCREMENT
     +  /SET*/LD* CNT* CIN*/UP*/Q0*/Q1*/Q2*/Q3*/Q4*/Q5*/Q6*/Q7*/Q8 ;DECREMENT
```

FUNCTION TABLE

CLK /OC SET /LD /CNT /CIN /UP D9 D8 D2-7 D1 D0 Q9 Q8 Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0

;			/	/			-DATA IN-			-DATA OUT-	
;C	/ S	/ C	C	/	-DATA		IN-		-DATA OUT-		
;L	O	E	L	N	I	U	DD	D	DD	QQQQQQQQQQ	
;K	C	T	D	T	N	P	98	2-7	10	9876543210	COMMENT
C	L	H	X	X	X	X	XX	X	XX	HHHHHHHHHH	SET (SET=H)
C	L	L	L	X	X	X	LL	L	LL	LLLLLLLLLL	CLEAR (D=L)
C	L	L	L	X	X	X	HH	H	HH	HHHHHHHHHH	SET (D=H)
C	L	L	H	H	X	X	LL	L	LL	HHHHHHHHHH	HOLD (/CNT=H)
C	L	L	L	X	X	X	LL	L	LL	LLLLLLLLLL	CLEAR (D=L)
C	L	L	H	H	X	X	XX	X	XX	LLLLLLLLLL	HOLD (/CNT=H)
C	L	L	H	L	H	X	XX	X	XX	LLLLLLLLLL	HOLD (/CIN=H)
C	L	L	H	L	L	L	XX	X	XX	LLLLLLLLLH	COUNT UP (NOTE 5 CNTRLS LOW NEAR GND)
C	L	L	H	L	L	L	XX	X	XX	LLLLLLLLHL	COUNT UP
C	L	L	H	L	L	L	XX	X	XX	LLLLLLLLHH	COUNT UP
C	L	L	H	L	L	L	XX	X	XX	LLLLLLLHLL	COUNT UP
C	L	L	H	L	L	H	XX	X	XX	LLLLLLLLHH	COUNT DOWN
C	L	L	H	L	L	H	XX	X	XX	LLLLLLLLHL	COUNT DOWN
C	L	L	H	L	L	H	XX	X	XX	LLLLLLLLLH	COUNT DOWN
C	L	L	H	L	L	H	XX	X	XX	LLLLLLLLLL	COUNT DOWN
C	L	L	H	L	L	H	XX	X	XX	HHHHHHHHHH	COUNT DOWN (ROLL UNDER)
C	L	L	H	L	L	H	XX	X	XX	HHHHHHHHHL	COUNT DOWN
C	L	L	H	L	L	H	XX	X	XX	HHHHHHHHLH	COUNT DOWN
C	L	L	H	L	L	H	XX	X	XX	HHHHHHHHLL	COUNT DOWN
C	L	L	H	L	L	H	XX	X	XX	HHHHHHHLHH	COUNT DOWN
C	L	L	H	L	L	H	XX	X	XX	HHHHHHHLHL	COUNT DOWN
C	L	L	H	L	L	H	XX	X	XX	HHHHHHHLLH	COUNT DOWN
C	L	L	H	L	L	H	XX	X	XX	HHHHHHHLLL	COUNT DOWN
C	L	L	L	X	X	X	LL	L	LL	LLLLLLLLLL	LOAD
C	L	L	H	L	L	L	XX	X	XX	LLLLLLLLLH	COUNT UP
C	L	L	H	H	X	X	XX	X	XX	LLLLLLLLLH	HOLD (/CNT=H)
C	L	L	H	L	L	H	XX	X	XX	LLLLLLLLLL	COUNT DOWN
C	L	L	L	X	X	X	LL	H	LL	LLHHHHHHLL	LOAD
C	L	L	H	L	L	L	XX	X	XX	LLHHHHHHLH	COUNT UP
C	L	L	H	H	X	X	XX	X	XX	LLHHHHHHLH	HOLD (/CNT=H)
C	L	L	H	L	L	H	XX	X	XX	LLHHHHHHLL	COUNT DOWN
C	L	L	L	X	X	X	HH	L	HH	HHLLLLLLHH	LOAD
C	L	L	H	L	L	L	XX	X	XX	HHLLLLLHLL	COUNT UP
C	L	L	H	H	X	X	XX	X	XX	HHLLLLLHLL	HOLD (/CNT=H)
C	L	L	H	L	L	H	XX	X	XX	HHLLLLLLHH	COUNT DOWN
C	L	L	L	X	X	X	LH	L	HL	LHLLLLLLHL	LOAD
C	L	L	H	L	L	L	XX	X	XX	LHLLLLLLHH	COUNT UP
C	L	L	H	H	X	X	XX	X	XX	LHLLLLLLHH	HOLD (/CNT=H)
C	L	L	H	L	L	H	XX	X	XX	LHLLLLLLHL	COUNT DOWN
C	L	L	L	X	X	X	HL	H	LH	HLHHHHHHLH	LOAD
C	L	L	H	L	L	L	XX	X	XX	HLHHHHHHHL	COUNT UP
C	L	L	H	H	X	X	XX	X	XX	HLHHHHHHHL	HOLD (/CNT=H)
C	L	L	H	L	L	H	XX	X	XX	HLHHHHHHLH	COUNT DOWN
C	L	L	L	X	X	X	HH	H	HH	HHHHHHHHHH	LOAD
C	L	L	H	L	L	L	HH	H	HH	LLLLLLLLLL	COUNT UP (ROLL OVER)
X	H	X	X	X	X	X	XX	X	XX	ZZZZZZZZZZ	TEST HI-Z

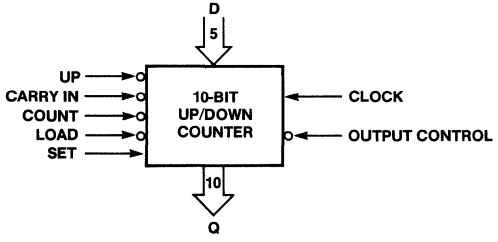
DESCRIPTION

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

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

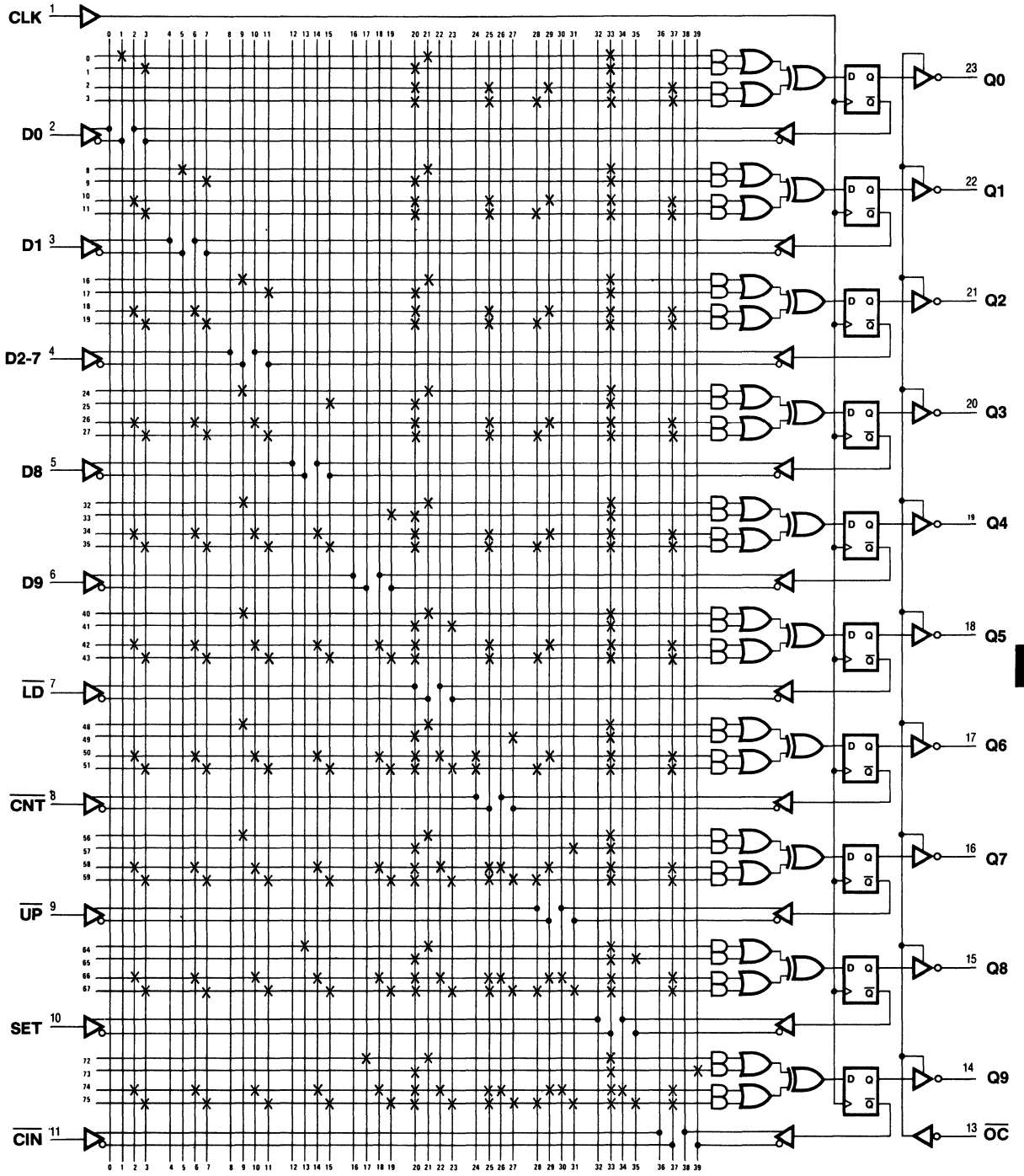
THESE OPERATIONS ARE EXERCISED IN THE FUNCTION TABLE AND SUMMARIZED IN THE OPERATIONS TABLE:

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



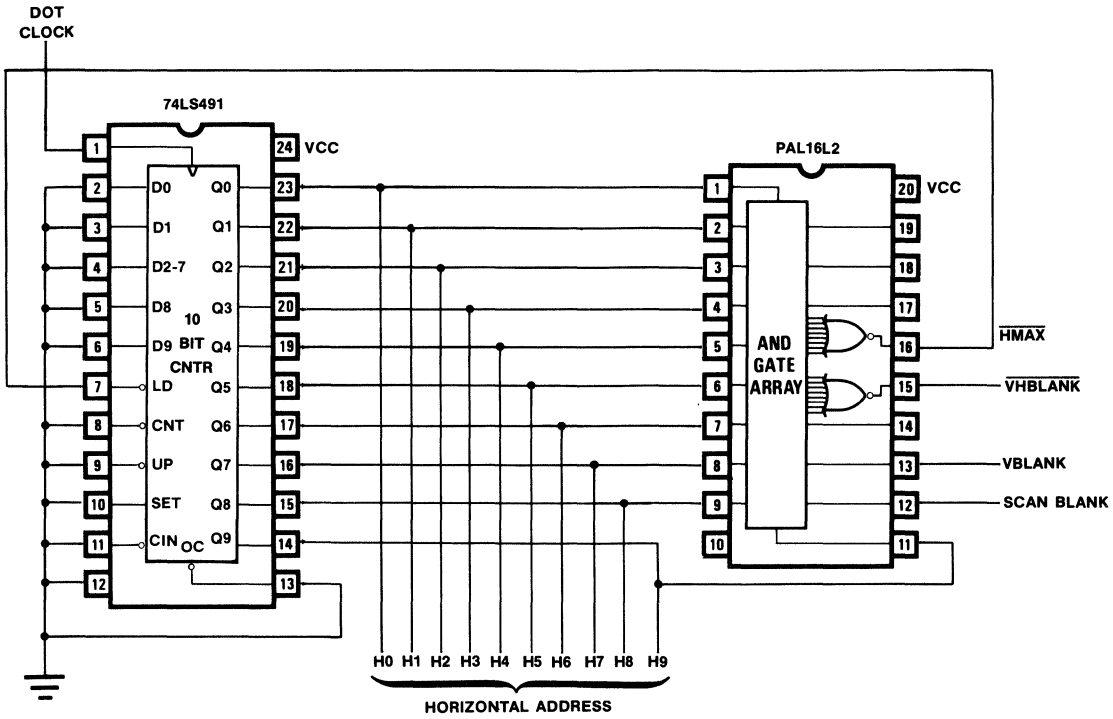
10-Bit Counter

Logic Diagram PAL20X10



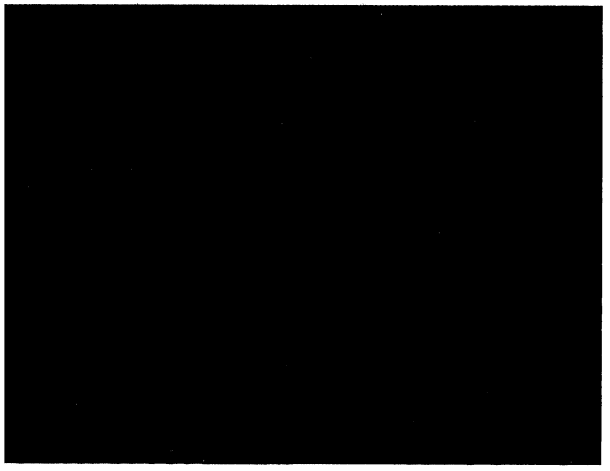
Application

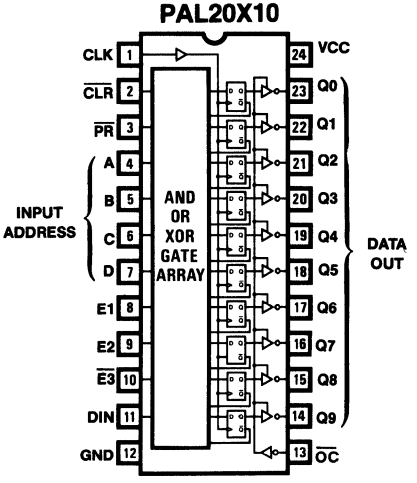
CRT Horizontal Timing and Blanking



NOTE PAL16L2 design is application dependent

10-Bit Addressable Register





6

```
PAL20X10                                                    PAL DESIGN SPECIFICATION
PMSI407                                                       DANESH TAVANA 04/05/82
10-BIT ADDRESSABLE REGISTER
MMI SUNNYVALE, CALIFORNIA
CLK /CLR /PR A  B  C  D  E1 E2 /E3 DIN GND
/OC  Q9   Q8 Q7 Q6 Q5 Q4 Q3 Q2  Q1 Q0  VCC


/Q0 :=   CLR                                          ;CLEAR (LSB)
     +  /PR*/Q0                                       ;HOLD (/Q0)
    :+: /PR*/CLR* E1* E2* E3*/D*/C*/B*/A*/Q0* DIN     ;LOAD ( DIN:+:/Q0) IF /Q0=H
     +  /PR*/CLR* E1* E2* E3*/D*/C*/B*/A* Q0*/DIN     ;LOAD (/DIN:+:/Q0) IF /Q0=L

/Q1 :=   CLR                                          ;CLEAR
     +  /PR*/Q1                                       ;HOLD (/Q1)
    :+: /PR*/CLR* E1* E2* E3*/D*/C*/B* A*/Q1* DIN     ;LOAD ( DIN:+:/Q1) IF /Q1=H
     +  /PR*/CLR* E1* E2* E3*/D*/C*/B* A* Q1*/DIN     ;LOAD (/DIN:+:/Q1) IF /Q1=L

/Q2 :=   CLR                                          ;CLEAR
     +  /PR*/Q2                                       ;HOLD (/Q2)
    :+: /PR*/CLR* E1* E2* E3*/D*/C* B*/A*/Q2* DIN     ;LOAD ( DIN:+:/Q2) IF /Q2=H
     +  /PR*/CLR* E1* E2* E3*/D*/C* B*/A* Q2*/DIN     ;LOAD (/DIN:+:/Q2) IF /Q2=L

/Q3 :=   CLR                                          ;CLEAR
     +  /PR*/Q3                                       ;HOLD (/Q3)
    :+: /PR*/CLR* E1* E2* E3*/D*/C* B* A*/Q3* DIN     ;LOAD ( DIN:+:/Q3) IF /Q3=H
     +  /PR*/CLR* E1* E2* E3*/D*/C* B* A* Q3*/DIN     ;LOAD (/DIN:+:/Q3) IF /Q3=L

/Q4 :=   CLR                                          ;CLEAR
     +  /PR*/Q4                                       ;HOLD (/Q4)
    :+: /PR*/CLR* E1* E2* E3*/D* C*/B*/A*/Q4* DIN     ;LOAD ( DIN:+:/Q4) IF /Q4=H
     +  /PR*/CLR* E1* E2* E3*/D* C*/B*/A* Q4*/DIN     ;LOAD (/DIN:+:/Q4) IF /Q4=L

/Q5 :=   CLR                                          ;CLEAR
     +  /PR*/Q5                                       ;HOLD (/Q5)
    :+: /PR*/CLR* E1* E2* E3*/D* C*/B* A*/Q5* DIN     ;LOAD ( DIN:+:/Q5) IF /Q5=H
     +  /PR*/CLR* E1* E2* E3*/D* C*/B* A* Q5*/DIN     ;LOAD (/DIN:+:/Q5) IF /Q5=L

/Q6 :=   CLR                                          ;CLEAR
     +  /PR*/Q6                                       ;HOLD (/Q6)
    :+: /PR*/CLR* E1* E2* E3*/D* C* B*/A*/Q6* DIN     ;LOAD ( DIN:+:/Q6) IF /Q6=H
     +  /PR*/CLR* E1* E2* E3*/D* C* B*/A* Q6*/DIN     ;LOAD (/DIN:+:/Q6) IF /Q6=L

/Q7 :=   CLR                                          ;CLEAR
     +  /PR*/Q7                                       ;HOLD (/Q7)
    :+: /PR*/CLR* E1* E2* E3*/D* C* B* A*/Q7* DIN     ;LOAD ( DIN:+:/Q7) IF /Q7=H
     +  /PR*/CLR* E1* E2* E3*/D* C* B* A* Q7*/DIN     ;LOAD (/DIN:+:/Q7) IF /Q7=L

/Q8 :=   CLR                                          ;CLEAR
     +  /PR*/Q8                                       ;HOLD (/Q8)
    :+: /PR*/CLR* E1* E2* E3* D*/C*/B*/A*/Q8* DIN     ;LOAD ( DIN:+:/Q8) IF /Q8=H
     +  /PR*/CLR* E1* E2* E3* D*/C*/B*/A* Q8*/DIN     ;LOAD (/DIN:+:/Q8) IF /Q8=L

/Q9 :=   CLR                                          ;CLEAR (MSB)
     +  /PR*/Q9                                       ;HOLD (/Q9)
    :+: /PR*/CLR* E1* E2* E3* D*/C*/B* A*/Q9* DIN     ;LOAD ( DIN:+:/Q9) IF /Q9=H
     +  /PR*/CLR* E1* E2* E3* D*/C*/B* A* Q9*/DIN     ;LOAD (/DIN:+:/Q9) IF /Q9=L
```

FUNCTION TABLE

/OC CLK /CLR /PR /E3 E2 E1 D C B A DIN Q9 Q8 Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0

;																					
;												-----OUTPUTS-------									
;CONTROL		-----FUNCTIONS----					---INPUTS--					Q	Q	Q	Q	Q	Q	Q	Q	Q	Q
;/OC	CLK	/CLR	/PR	/E3	E2	E1	D	C	B	A	DIN	9	8	7	6	5	4	3	2	1	0	COMMENTS
L	C	L	L	X	X	X	X	X	X	X	X	L	L	L	L	L	L	L	L	L	L	/CLR OVRRD /PR
L	C	H	L	X	X	X	X	X	X	X	X	H	H	H	H	H	H	H	H	H	H	/PR OVRRD ENABLE
L	C	H	H	L	H	H	L	L	L	L	L	H	H	H	H	H	H	H	H	H	L	LOAD Q0 WITH DIN
L	C	H	H	L	H	H	L	L	L	H	L	H	H	H	H	H	H	H	H	L	L	LOAD Q1 WITH DIN
L	C	H	H	L	H	H	L	L	H	L	L	H	H	H	H	H	H	H	L	L	L	LOAD Q2 WITH DIN
L	C	H	H	L	H	H	L	L	H	H	L	H	H	H	H	H	H	L	L	L	L	LOAD Q3 WITH DIN
L	C	H	H	L	H	H	L	H	L	L	L	H	H	H	H	H	L	L	L	L	L	LOAD Q4 WITH DIN
L	C	H	H	L	H	H	L	H	L	H	L	H	H	H	H	L	L	L	L	L	L	LOAD Q5 WITH DIN
L	C	H	H	L	H	H	L	H	H	L	L	H	H	H	L	L	L	L	L	L	L	LOAD Q6 WITH DIN
L	C	H	H	L	H	H	L	H	H	H	L	H	H	L	L	L	L	L	L	L	L	LOAD Q7 WITH DIN
L	C	H	H	L	H	H	H	L	L	L	L	H	L	L	L	L	L	L	L	L	L	LOAD Q8 WITH DIN
L	C	H	H	L	H	H	H	L	L	H	L	L	L	L	L	L	L	L	L	L	L	LOAD Q9 WITH DIN
L	C	H	H	L	H	H	H	L	L	H	H	H	L	L	L	L	L	L	L	L	L	LOAD Q9 WITH DIN
L	C	H	H	L	H	H	H	L	L	L	H	H	H	L	L	L	L	L	L	L	L	LOAD Q8 WITH DIN
L	C	H	H	L	H	H	L	H	H	H	H	H	H	H	L	L	L	L	L	L	L	LOAD Q7 WITH DIN
L	C	H	H	L	H	H	L	H	H	L	H	H	H	H	H	L	L	L	L	L	L	LOAD Q6 WITH DIN
L	C	H	H	L	H	H	L	H	L	H	H	H	H	H	H	H	L	L	L	L	L	LOAD Q5 WITH DIN
L	C	H	H	L	H	H	L	H	L	L	H	H	H	H	H	H	H	L	L	L	L	LOAD Q4 WITH DIN
L	C	H	H	L	H	H	L	L	H	H	H	H	H	H	H	H	H	H	L	L	L	LOAD Q3 WITH DIN
L	C	H	H	L	H	H	L	L	H	L	H	H	H	H	H	H	H	H	H	L	L	LOAD Q2 WITH DIN
L	C	H	H	L	H	H	L	L	L	H	H	H	H	H	H	H	H	H	H	H	L	LOAD Q1 WITH DIN
L	C	H	H	L	H	H	L	L	L	L	H	H	H	H	H	H	H	H	H	H	H	LOAD Q0 WITH DIN
L	C	H	H	L	L	L	X	X	X	X	X	H	H	H	H	H	H	H	H	H	H	HOLD STATE
L	C	H	H	L	H	H	L	L	L	L	L	H	H	H	H	H	H	H	H	H	L	LOAD Q0 WITH DIN
L	C	H	H	L	L	H	X	X	X	X	X	H	H	H	H	H	H	H	H	H	L	HOLD STATE
L	C	H	H	L	H	H	L	L	H	L	L	H	H	H	H	H	H	H	L	H	L	LOAD Q2 WITH DIN
L	C	H	H	L	H	L	X	X	X	X	X	H	H	H	H	H	H	H	L	H	L	HOLD STATE
L	C	H	H	L	H	H	L	H	L	L	L	H	H	H	H	H	L	H	L	H	L	LOAD Q4 WITH DIN
L	C	H	H	H	L	L	X	X	X	X	X	H	H	H	H	H	L	H	L	H	L	HOLD STATE
L	C	H	H	L	H	H	L	H	H	L	L	H	H	H	L	H	L	H	L	H	L	LOAD Q6 WITH DIN
L	C	H	H	H	L	H	X	X	X	X	X	H	H	H	L	H	L	H	L	H	L	HOLD STATE
L	C	H	H	L	H	H	H	L	L	L	L	H	L	H	L	H	L	H	L	H	L	LOAD Q8 WITH DIN
L	C	H	H	H	H	H	X	X	X	X	X	H	L	H	L	H	L	H	L	H	L	HOLD STATE
H	X	X	X	X	X	X	X	X	X	X	X	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	TEST HI-Z

DESCRIPTION

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

CLEAR OVERRIDES PRESET AND ENABLE, PRESET OVERRIDES ENABLE.

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

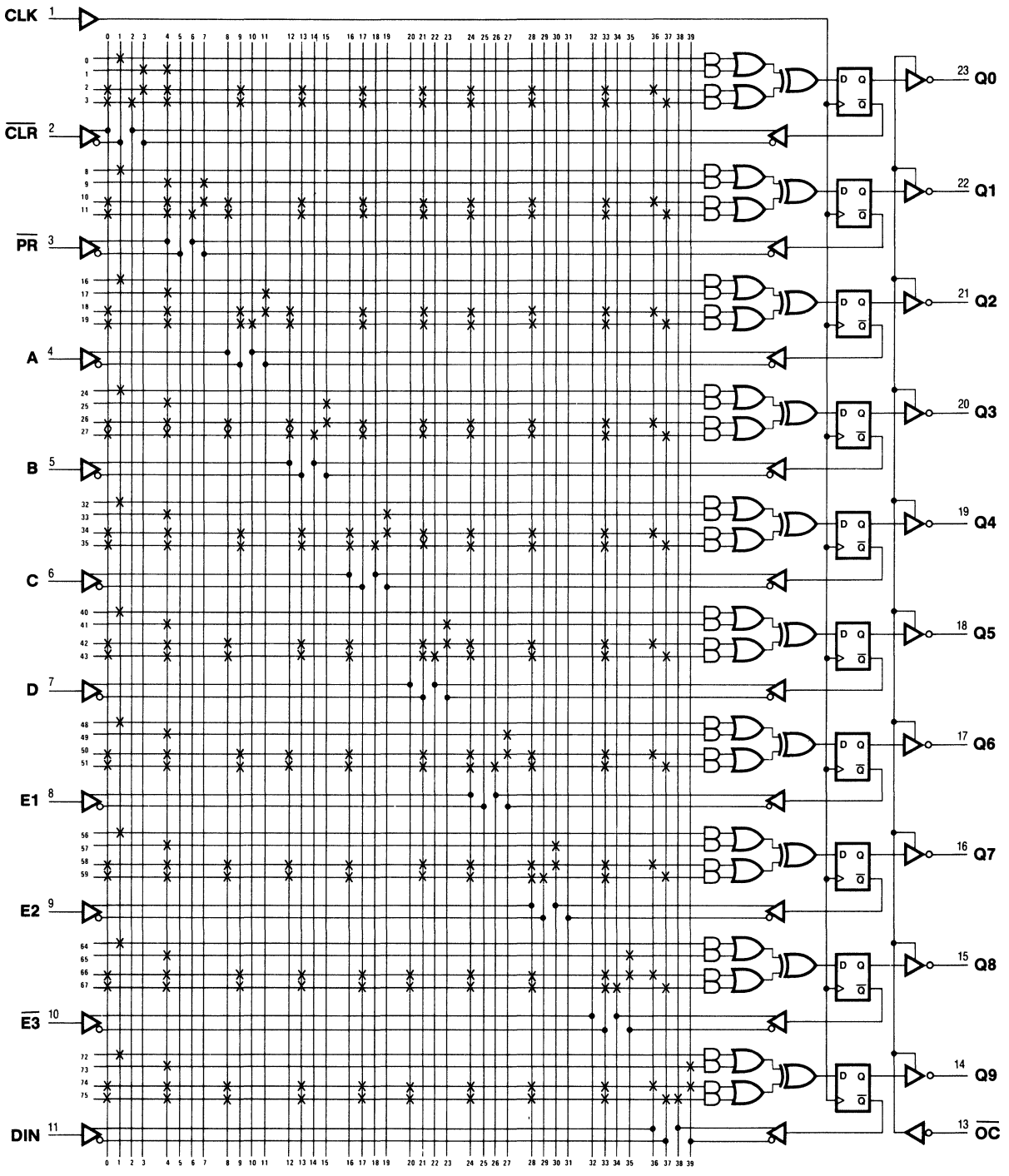
/OC	CLK	/CLR	/PR	/E3	E2	E1	D	C	B	A	DIN	Q9-Q0	OPERATION
H	X	X	X	X	X	X	X	X	X	X	X	Z	HI-Z
L	C	L	L	X	X	X	X	X	X	X	X	L	CLEAR
L	C	H	L	X	X	X	X	X	X	X	X	H	PRESET
L	C	H	H	L	L	L	X	X	X	X	X	Q	HOLD PREVIOUS STATES
L	C	H	H	L	L	H	X	X	X	X	X	Q	HOLD PREVIOUS STATES
L	C	H	H	L	H	L	X	X	X	X	X	Q	HOLD PREVIOUS STATES
L	C	H	H	L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

OUTPUT SELECT TABLE

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

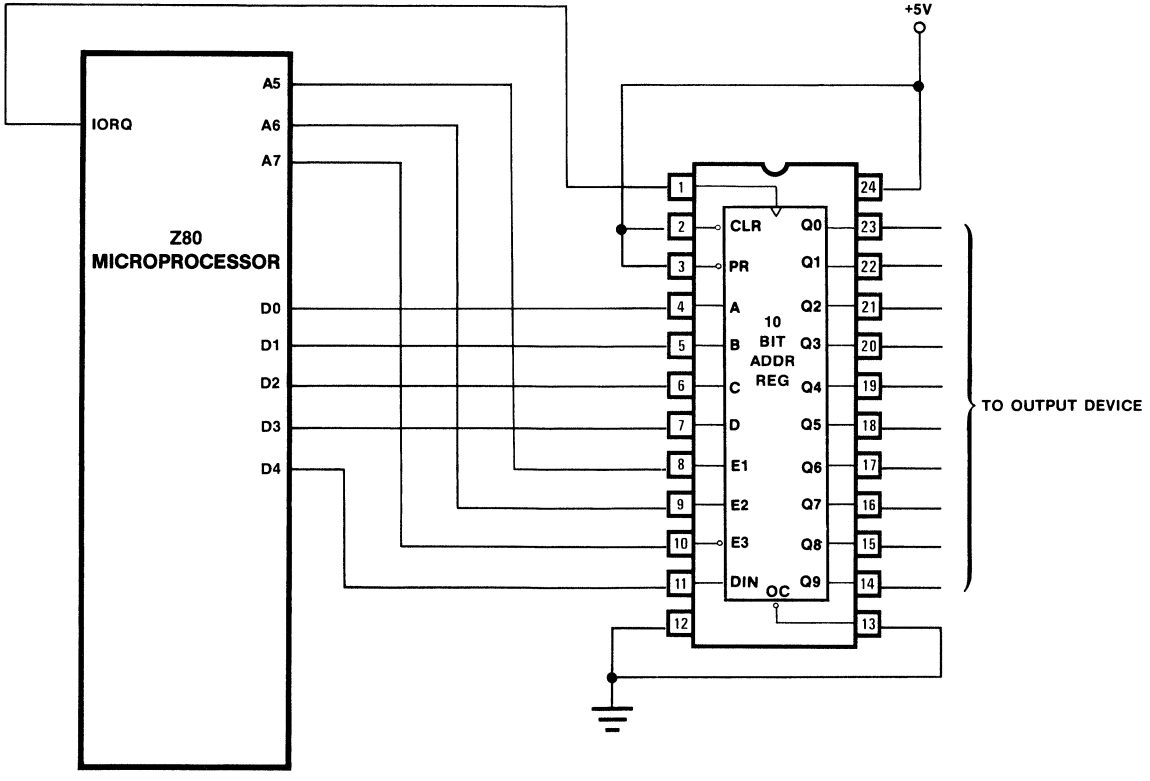
10-Bit Addressable Register

Logic Diagram PAL20X10

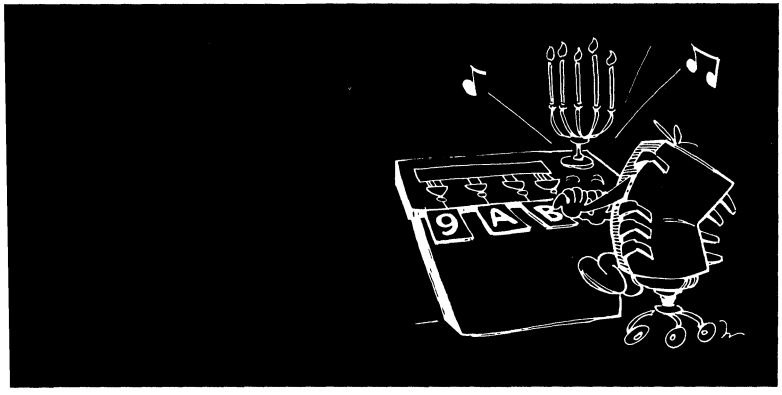


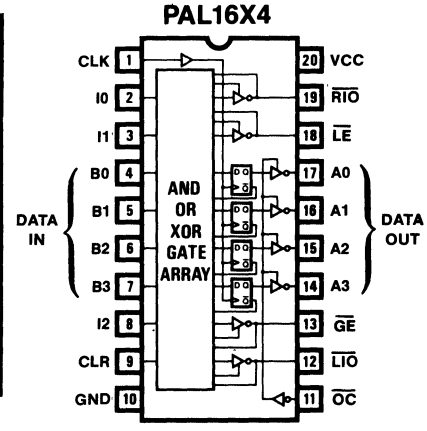
Application

10-Bit Addressable Register



4-Bit Up/Down Counter with Shift Register and Comparator





6

```
PAL16X4                                                    PAL DESIGN SPECIFICATION
SCNT4C                                                       BIRKNER/COLI 10/31/81
4-BIT UP/DOWN COUNTER WITH SHIFT REGISTER AND COMPARATOR
MMI SUNNYVALE, CALIFORNIA
CLK I0 I1 B0 B1 B2 B3 I2 CLR GND /OC /LIO /GE A3 A2 A1 A0 /LE /RIO VCC

IF (/I2* I1*/I0) RIO = (A0)                                ;SHIFT RIGHT OUT

/A0 :=  /I2*/I1*/I0*(/A0)*/CLR                             ;HOLD A0
     +  /I2*/I1* I0*(/B0)*/CLR                             ;LOAD B0 (LSB)
     +  /I2* I1*/I0*(/A1)*/CLR                             ;SHIFT RIGHT
     +   I2* I1*    (/A0)*/CLR                            ;HOLD (INC AND DEC)
    :+:  I2*/I1*/I0*/RIO                                   ;SHIFT LEFT
     +   I2* I1*/I0* RIO                                   ;INCREMENT IF CARRY IN
     +   I2* I1* I0* RIO                                   ;DECREMENT IF BORROW IN
     +   CLR                                               ;CLEAR LSB

/A1 :=  /I2*/I1*/I0*(/A1)*/CLR                             ;HOLD A1
     +  /I2*/I1* I0*(/B1)*/CLR                             ;LOAD B1
     +  /I2* I1*/I0*(/A2)*/CLR                             ;SHIFT RIGHT
     +   I2* I1*    (/A1)*/CLR                            ;HOLD (INC AND DEC)
    :+:  I2*/I1*/I0*(/A0)                                  ;SHIFT LEFT
     +   I2* I1*/I0*( A0)* RIO                             ;INCREMENT IF CARRY IN
     +   I2* I1* I0*(/A0)* RIO                             ;DECREMENT IF BORROW IN
     +   CLR                                               ;CLEAR

/A2 :=  /I2*/I1*/I0*(/A2)*/CLR                             ;HOLD A2
     +  /I2*/I1* I0*(/B2)*/CLR                             ;LOAD B2
     +  /I2* I1*/I0*(/A3)*/CLR                             ;SHIFT RIGHT
     +   I2* I1*    (/A2)*/CLR                            ;HOLD (INC AND DEC)
    :+:  I2*/I1*/I0*(/A1)                                  ;SHIFT LEFT
     +   I2* I1*/I0*( A1)*( A0)* RIO                       ;INCREMENT IF CARRY IN
     +   I2* I1* I0*(/A1)*(/A0)* RIO                       ;DECREMENT IF BORROW IN
     +   CLR                                               ;CLEAR

/A3 :=  /I2*/I1*/I0*(/A3)*/CLR                             ;HOLD A3
     +  /I2*/I1* I0*(/B3)*/CLR                             ;LOAD B3 (MSB)
     +  /I2* I1*/I0* /LIO*/CLR                             ;SHIFT RIGHT
     +   I2* I1*    (/A3)*/CLR                            ;HOLD (INC AND DEC)
    :+:  I2*/I1*/I0*(/A2)                                  ;SHIFT LEFT
     +   I2* I1*/I0*( A2)*( A1)*( A0)* RIO                 ;INCREMENT IF CARRY IN
     +   I2* I1* I0*(/A2)*(/A1)*(/A0)* RIO                 ;DECREMENT IF BORROW IN
     +   CLR                                               ;CLEAR MSB

IF ( I2) LIO = I2*/I1*/I0*( A3)                            ;SHIFT LEFT OUT
             + I2* I1*/I0*( A3)*( A2)*( A1)*( A0)* RIO     ;CARRY OUT
             + I2* I1* I0*(/A3)*(/A2)*(/A1)*(/A0)* RIO     ;BORROW OUT

IF (VCC)  LE = (A3*/B3)                                    ;B3 LT A3
             + (A3:*:B3)*(A2*/B2)                          ;B2 LT A2
             + (A3:*:B3)*(A2:*:B2)*(A1*/B1)                ;B1 LT A1
             + (A3:*:B3)*(A2:*:B2)*(A1:*:B1)*(A0*/B0)      ;B0 LT A0
             + (A3:*:B3)*(A2:*:B2)*(A1:*:B1)*(A0:*:B0)     ;B  EQ A

IF (VCC)  GE = (/A3*B3)                                    ;B3 GT A3
             + (A3:*:B3)*(/A2*B2)                          ;B2 GT A2
             + (A3:*:B3)*(A2:*:B2)*(/A1*B1)                ;B1 GT A1
             + (A3:*:B3)*(A2:*:B2)*(A1:*:B1)*(/A0*B0)      ;B0 GT A0
             + (A3:*:B3)*(A2:*:B2)*(A1:*:B1)*(A0:*:B0)     ;B  EQ A
```

FUNCTION TABLE

CLK /OC CLR I2 I1 I0 B3 B2 B1 B0 /GE /LE /LIO /RIO A3 A2 A1 A0

;			INST	INPUT					OUTPUT	
;--CONTROL--			III	BBBB	STATUS		---I/O---		AAAA	COMMENTS
;CLK	/OC	CLR	210	3210	/GE	/LE	/LIO	/RIO	3210	(HEX VALUES)
C	L	H	XXX	XXXX	X	X	Z	Z	LLLL	CLEAR
C	L	L	LHL	XXXX	X	X	L	H	HLLL	SHIFT RIGHT IN A H
C	L	L	LHL	XXXX	X	X	H	H	LHLL	SHIFT RIGHT IN A L
C	L	L	LLL	LHLH	L	H	Z	Z	LHLL	COMPARE B GT A
C	L	L	LHL	XXXX	X	X	H	H	LLHL	SHIFT RIGHT IN A L
C	L	L	LLL	LLHL	L	L	Z	Z	LLHL	COMPARE B EQ A
C	L	L	LHL	XXXX	X	X	H	L	LLLH	SHIFT RIGHT IN A L
C	L	L	LHL	XXXX	X	X	H	H	LLLL	SHIFT RIGHT IN A L
C	L	L	HLH	XXXX	X	X	H	Z	HHHH	SET
C	L	L	LLL	XXXX	X	X	Z	Z	HHHH	HOLD
C	L	L	HLL	XXXX	X	X	L	H	HHHL	SHIFT LEFT IN A L
C	L	L	HLL	XXXX	X	X	L	L	HHLH	SHIFT LEFT IN A H
C	L	L	HLL	XXXX	X	X	L	L	HLHH	SHIFT LEFT IN A H
C	L	L	HLL	XXXX	X	X	H	L	LHHH	SHIFT LEFT IN A H
C	L	L	HLL	XXXX	X	X	L	L	HHHH	SHIFT LEFT IN A H
C	L	L	LLH	LLLH	X	X	Z	Z	LLLH	LOAD (1)
C	L	L	HHL	XXXX	X	X	H	L	LLHL	INCREMENT
C	L	L	HHH	XXXX	X	X	H	L	LLLH	DECREMENT
C	L	L	LLH	LLHH	X	X	Z	Z	LLHH	LOAD (3)
C	L	L	HHL	XXXX	X	X	H	L	LHLL	INCREMENT
C	L	L	HHH	XXXX	X	X	H	L	LLHH	DECREMENT
C	L	L	LLH	LHHH	X	X	Z	Z	LHHH	LOAD (7)
C	L	L	HHL	XXXX	X	X	H	L	HLLL	INCREMENT
C	L	L	HHH	XXXX	X	X	H	L	LHHH	DECREMENT
C	L	L	LLH	HHHH	X	X	Z	Z	HHHH	LOAD (F)
C	L	L	LLH	LHHH	X	X	Z	Z	LHHH	LOAD (7)
C	L	L	LLH	HLHH	X	X	Z	Z	HLHH	LOAD (B)
C	L	L	LLH	HHLH	X	X	Z	Z	HHLH	LOAD (D)
C	L	L	LLH	HHHL	X	X	Z	Z	HHHL	LOAD (E)
C	L	L	LLH	HLHL	X	X	Z	Z	HLHL	LOAD (A)
C	L	L	HHL	XXXX	X	X	H	L	HLHH	INCREMENT TO (B)
C	L	L	HHL	XXXX	X	X	H	L	HHLL	INCREMENT TO (C)
C	L	L	LLL	HLHH	H	L	Z	Z	HHLL	COMPARE B LT A
C	L	L	HHL	XXXX	X	X	H	L	HHLH	INCREMENT TO (D)
C	L	L	HHL	XXXX	X	X	H	H	HHLH	HOLD NO CARRY IN (/RIO=H)
C	L	L	HHL	XXXX	X	X	H	L	HHHL	INCREMENT TO (E)
C	L	L	HHL	XXXX	X	X	L	L	HHHH	INCREMENT TO (F)CARRY OUT
C	L	L	HHL	XXXX	X	X	H	L	LLLL	INCREMENT TO (0)ROLL OVER
C	L	L	LHH	XXXX	X	X	Z	Z	HHHH	SET
C	L	L	HHH	XXXX	X	X	H	L	HHHL	DECREMENT TO (E)
C	L	L	LLH	LLHH	X	X	Z	Z	LLHH	LOAD (3)
C	L	L	HHH	XXXX	X	X	H	L	LLHL	DECREMENT TO (2)
L	L	L	HHH	LLHL	X	X	H	H	LLHL	COMPARE B EQ A
C	L	L	HHH	XXXX	X	X	H	L	LLLH	DECREMENT TO (1)
C	L	L	HHH	XXXX	X	X	L	L	LLLL	DECREMENT TO (0)BOROW OUT
C	L	L	HHH	XXXX	X	X	H	L	HHHH	DECREMENT TO (F)ROLL UNDR
X	H	X	XXX	XXXX	X	X	X	X	ZZZZ	TEST HI-Z

DESCRIPTION

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

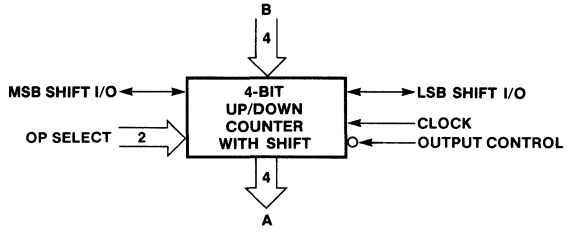
THE COMPARISION OPERATIONS (/GE AND /LE) WILL COMPARE THE INPUT DATA (B) WITH THE DATA IN THE REGISTERS (A) AND SUPPLY THE FOLLOWING STATUS:

COMPARISION	/GE	/LE
B IS GREATER THAN A	L	H
B IS EQUAL TO A	L	L
B IS LESS THAN A	H	L

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

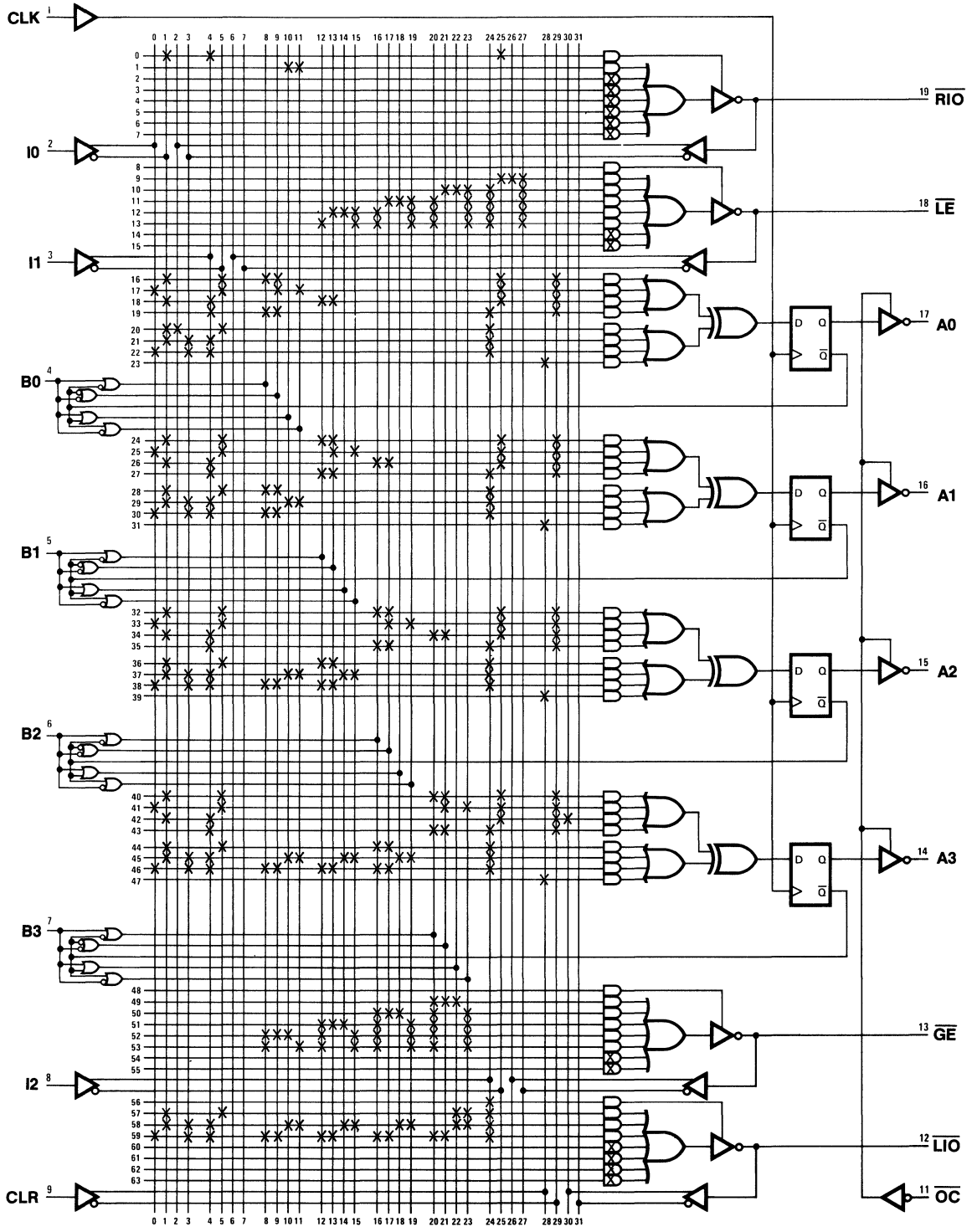
THESE OPERATIONS ARE EXERCISED IN THE FUNCTION TABLE AND SUMMARIZED IN THE OPERATIONS TABLE BELOW:

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



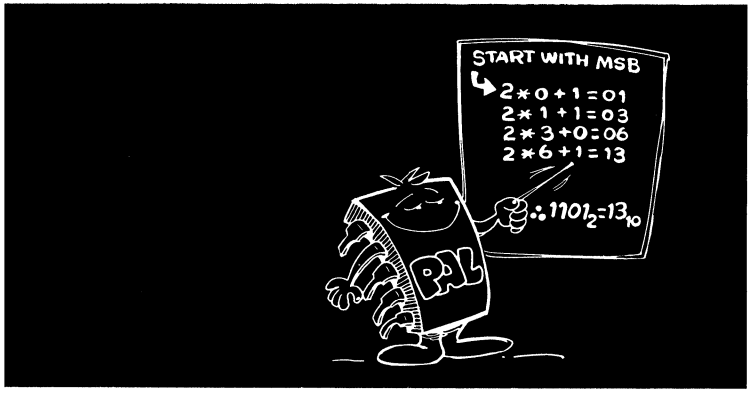
4-Bit Up/Down Counter with Shift Register and Comparator

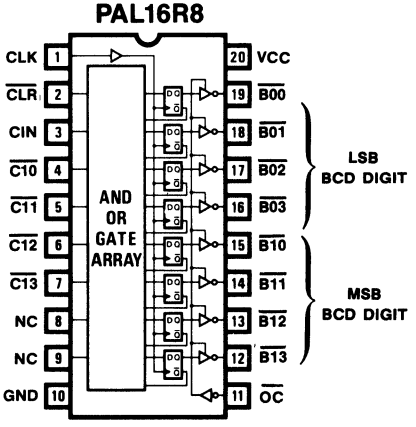
Logic Diagram PAL16X4




IMPORTANT
NOTES

Binary to BCD 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.

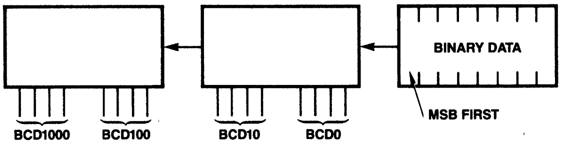


Figure 1. Conceptual Diagram of Binary to BCD Converter

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

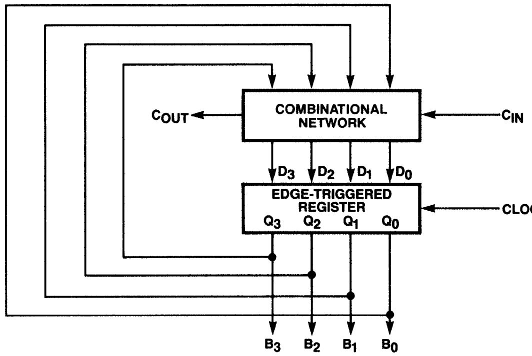


Figure 3, BCD Converter Building Block.

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.

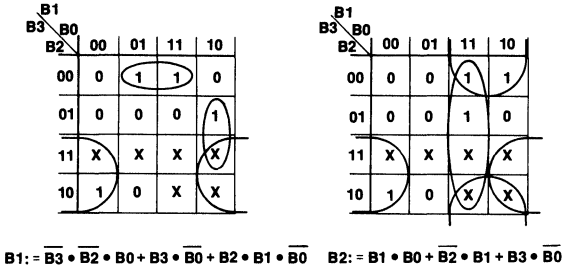


Figure 4. Karnaugh Maps

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.

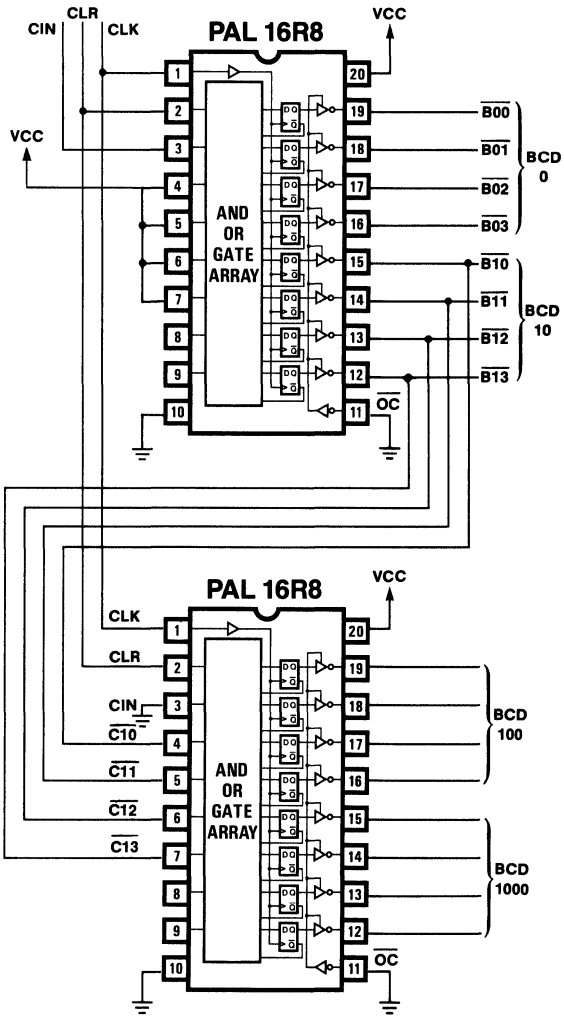


Figure 5. Logic Schematic

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

```
PAL16R8                                          PAL DESIGN SPECIFICATION
BBCD                                             S. WASER/V. COLI 09/14/81
BINARY TO BCD CONVERTER
MMI SUNNYVALE, CALIFORNIA
CLK /CLR CIN /C10 /C11 /C12 /C13 NC NC GND
/OC /B13 /B12 /B11 /B10 /B03 /B02 /B01 /B00 VCC


B00 := /CLR* CIN                          ;CONVERT B00 (LSB)
     + /CLR* C13
     + /CLR* C12* C10
     + /CLR* C12* C11

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

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

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

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

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

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

B13 := /CLR* B12*/B11*/B10                ;CONVERT B13 (MSB)
     + /CLR* B13* B10
```

FUNCTION TABLE

CIN C13 C12 C11 C10 /CLR CLK /OC B13 B12 B11 B10 B03 B02 B01 B00

;INPUTS	CONTROL	-OUTPUTS-	
; CARRY	/	(BCD)	
;C CCCC	C C /	BBBB BBBB	
;I 1111	L L O	1111 0000	
;N 3210	R K C	3210 3210	COMMENTS
X XXXX	L C L	LLLL LLLL	CLEAR (0)
H LLLL	H C L	LLLL LLLH	SERIAL INPUT A H (1)
H LLLL	H C L	LLLL LLHH	SERIAL INPUT A H (3)
L LLLL	H C L	LLLL LHHL	SERIAL INPUT A L (6)
H LLLL	H C L	LLLH LLHH	SERIAL INPUT A H (13)
X XXXX	L C L	LLLL LLLL	CLEAR (0)
H LLLL	H C L	LLLL LLLH	SERIAL INPUT A H (1)
L LLLL	H C L	LLLL LLHL	SERIAL INPUT A L (2)
L LLLL	H C L	LLLL LHLL	SERIAL INPUT A L (4)
L LLLL	H C L	LLLL HLLL	SERIAL INPUT A L (8)
H LLLL	H C L	LLLH LHHH	SERIAL INPUT A H (17)
L LLLL	H C L	LLHH LHLL	SERIAL INPUT A L (34)
L LLLL	H C L	LHHL HLLL	SERIAL INPUT A L (68)
X XXXX	L C L	LLLL LLLL	CLEAR (0)
H LLLL	H C L	LLLL LLLH	SERIAL INPUT A H (1)
L LLLL	H C L	LLLL LLHL	SERIAL INPUT A L (2)
H LLLL	H C L	LLLL LHLH	SERIAL INPUT A H (5)
H LLLL	H C L	LLLH LLLH	SERIAL INPUT A H (11)
L LLLL	H C L	LLHL LLHL	SERIAL INPUT A L (22)
H LLLL	H C L	LHLL LHLH	SERIAL INPUT A H (45)
L LLLL	H C L	HLLH LLLL	SERIAL INPUT A L (90)
X XXXX	L C L	LLLL LLLL	CLEAR REGISTERS TO TEST CASCADABILITY
L HLLH	H C L	LLLL LLLH	SERIAL INPUT A L (178) (TEST BCD 100, 1000)
L LHHH	H C L	LLLL LLHH	SERIAL INPUT A L (360) (TEST BCD 100, 1000)
L LHHL	H C L	LLLL LHHH	SERIAL INPUT A L (720) (TEST BCD 100, 1000)
L LLHL	H C L	LLLH LHLL	SERIAL INPUT A L (1440) (TEST BCD 100, 1000)
L LHLL	H C L	LLHL HLLL	SERIAL INPUT A L (2880) (TEST BCD 100, 1000)
L HLLL	H C L	LHLH LHHH	SERIAL INPUT A L (5760) (TEST BCD 100, 1000)
X XXXX	L C L	LLLL LLLL	CLEAR (0)
H LLLL	H C L	LLLL LLLH	SERIAL INPUT A H (1)
H LLLL	H C L	LLLL LLHH	SERIAL INPUT A H (3)
H LLLL	H C L	LLLL LHHH	SERIAL INPUT A H (7)
H LLLL	H C L	LLLH LHLH	SERIAL INPUT A H (15)
H LLLL	H C L	LLHH LLLH	SERIAL INPUT A H (31)
H LLLL	H C L	LHHL LLHH	SERIAL INPUT A H (63)
X XXXX	L C L	LLLL LLLL	CLEAR REGISTERS TO TEST CASCADABILITY
L LHHL	H C L	LLLL LLLH	SERIAL INPUT A H (127) (TEST BCD 100, 1000)
L LLHL	H C L	LLLL LLHL	SERIAL INPUT A H (255) (TEST BCD 100, 1000)
L LHLH	H C L	LLLL LHLH	SERIAL INPUT A H (511) (TEST BCD 100, 1000)
L LLLH	H C L	LLLH LLLL	SERIAL INPUT A H (1023) (TEST BCD 100, 1000)
L LLHL	H C L	LLHL LLLL	SERIAL INPUT A H (2047) (TEST BCD 100, 1000)
L LHLL	H C L	LHLL LLLL	SERIAL INPUT A H (4097) (TEST BCD 100, 1000)
L HLLH	H C L	HLLL LLLH	SERIAL INPUT A H (8196) (TEST BCD 100, 1000)
X XXXX	X X H	ZZZZ ZZZZ	TEST HI-Z

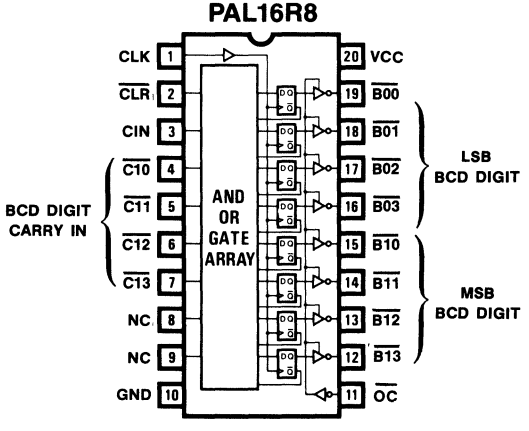
DESCRIPTION

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

THE INPUT BINARY DATA IS SHIFTED LEFT (STARTING WITH THE MSB) INTO THE BCD REGISTER. THIS TECHNIQUE IS KNOWN AS COULEUR'S TECHNIQUE (BIDEC).

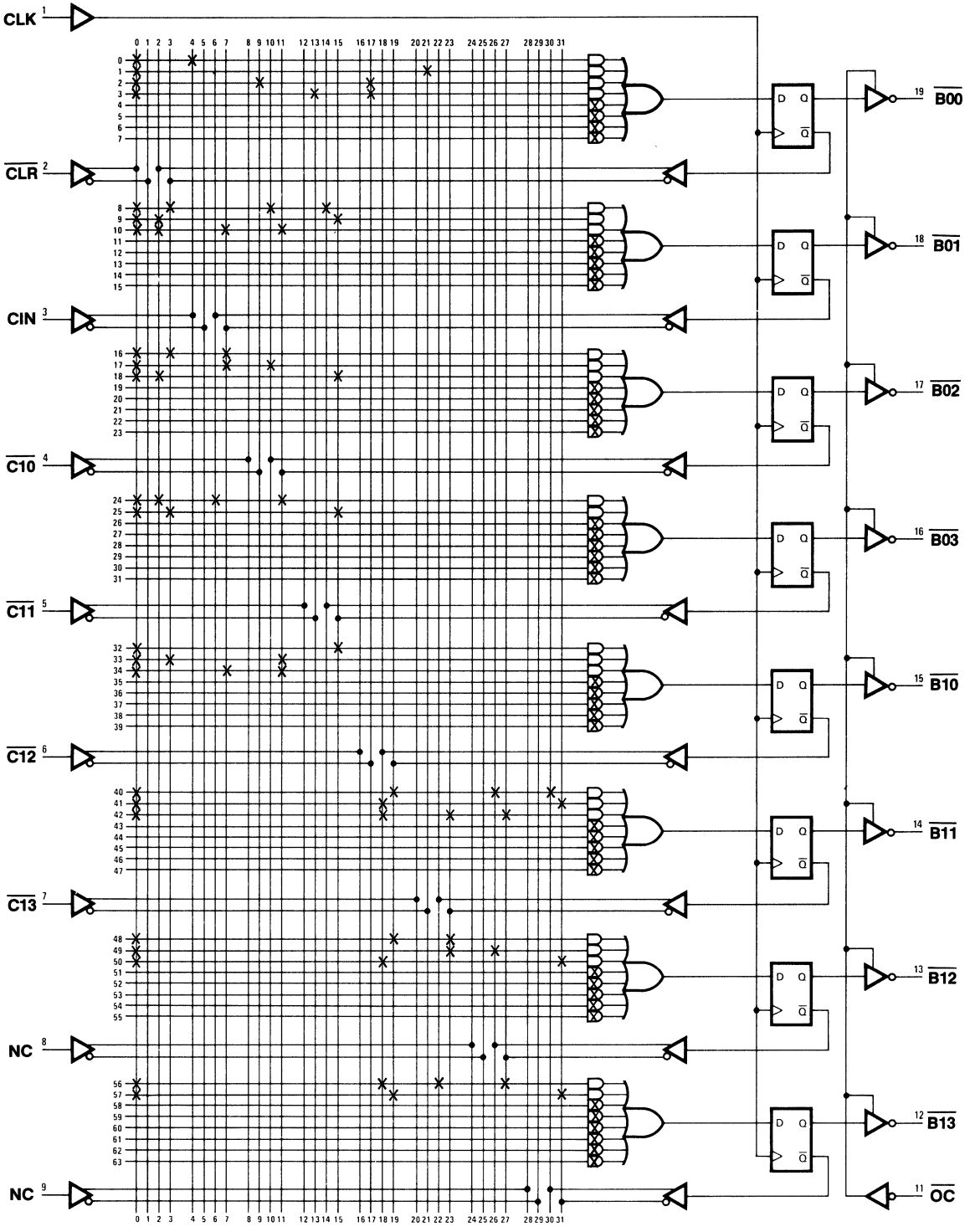
THE COMBITORIAL NETWORK IS DESIGNED FROM THE FOLLOWING NEXT-STATE TABLE:

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



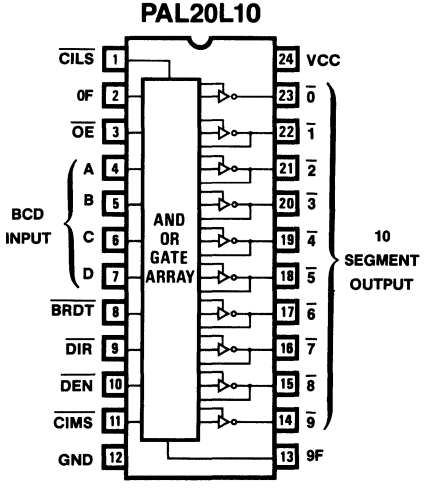
Binary to BCD Converter

Logic Diagram PAL16R8




IMPORTANT
NOTES

BCD to Decimal Decoder/ Bargraph Display Driver



6

```
PAL20L10                                              PAL DESIGN SPECIFICATION
PATTERN 003A                                          DAN JESSWEIN    12/29/82
BCD TO DECIMAL DECODER/BARGRAPH DISPLAY DRIVER
MMI FIELD ENGINEERING, CUPERTINO, CALIFORNIA
/CILS 0F /OE A B C D /BRDT /DIR /DEN /CIMS GND
9F  /9  /8  /7  /6  /5  /4  /3  /2  /1  /0 VCC


IF (OE) 0 = /A * /B * /C * /D * DEN          ;DECODE  0, DOTMODE
          +  BRDT * /DIR * 1                 ;DECODE  0, BARUP
          +  BRDT *  DIR * CILS              ;DECODE  0, BARDOWN

IF (OE) 1 =  A * /B * /C * /D *  DEN         ;DECODE  1, DOTMODE
          +  BRDT * /DIR * 2                 ;DECODE  1, BARUP
          +  BRDT *  DIR * /0F               ;DECODE  1, BARDOWN

IF (OE) 2 = /A *  B * /C * /D *  DEN         ;DECODE  2, DOTMODE
          +  BRDT * /DIR * 3                 ;DECODE  2, BARUP
          +  BRDT * DIR * 1                  ;DECODE  2, BARDOWN

IF (OE) 3 =  A *  B * /C * /D *  DEN         ;DECODE  3, DOTMODE
          +  BRDT * /DIR * 4                 ;DECODE  3, BARUP
          +  BRDT *  DIR * 2                 ;DECODE  3, BARDOWN

IF (OE) 4 = /A * /B *  C * /D *  DEN         ;DECODE  4, DOTMODE
          +  BRDT * /DIR * 5                 ;DECODE  4, BARUP
          +  BRDT *  DIR * 3                 ;DECODE  4, BARDOWN

IF (OE) 5 =  A * /B *  C * /D *  DEN         ;DECODE  5, DOTMODE
          +  BRDT * /DIR * 6                 ;DECODE  5, BARUP
          +  BRDT *  DIR * 4                 ;DECODE  5, BARDOWN

IF (OE) 6 = /A *  B *  C * /D *  DEN         ;DECODE  6, DOTMODE
          +  BRDT * /DIR * 7                 ;DECODE  6, BARUP
          +  BRDT *  DIR * 5                 ;DECODE  6, BARDOWN

IF (OE) 7 =  A *  B *  C * /D *  DEN         ;DECODE  7, DOTMODE
          +  BRDT * /DIR * 8                 ;DECODE  7, BARUP
          +  BRDT *  DIR * 6                 ;DECODE  7, BARDOWN

IF (OE) 8 = /A * /B * /C *  D *  DEN         ;DECODE  8, DOTMODE
          +  BRDT * /DIR * /9F               ;DECODE  8, BARUP
          +  BRDT *  DIR * 7                 ;DECODE  8, BARDOWN

IF (OE) 9 =  A * /B * /C *  D *  DEN         ;DECODE  9, DOTMODE
          +  BRDT * /DIR * CIMS              ;DECODE  9, BARUP
          +  BRDT *  DIR * 8                 ;DECODE  9, BARDOWN
```

FUNCTION TABLE

/OE /DEN /BRDT /DIR /CILS /CIMS D C B A 0F 9F /0 /1 /2 /3 /4 /5 /6 /7 /8 /9

;	/	//				
; /	B/	CC				
;/D	RD	II				
;OE	DI	LM		09	//////////	
;EN	TR	SS	DCBA	FF	0123456789	;COMMENTS
LL	HX	XX	LLLL	LH	LHHHHHHHHH	;TEST DOTMODE
LL	HX	XX	LLLH	HH	HLHHHHHHHH	; " " "
LL	HX	XX	LLHL	HH	HHLHHHHHHH	; " " "
LL	HX	XX	LLHH	HH	HHHLHHHHHH	; " " "
LL	HX	XX	LHLL	HH	HHHHLHHHHH	; " " "
LL	HX	XX	LHLH	HH	HHHHHLHHHH	; " " "
LL	HX	XX	LHHL	HH	HHHHHHLHHH	; " " "
LL	HX	XX	LHHH	HH	HHHHHHHLHH	; " " "
LL	HX	XX	HLLL	HH	HHHHHHHHLH	; " " "
LL	HX	XX	HLLH	HL	HHHHHHHHHL	; " " "
LL	HX	XX	HLHL	HH	HHHHHHHHHH	;TEST INVALID BCD
LL	HX	XX	HLHH	HH	HHHHHHHHHH	; " " " "
LL	HX	XX	HHLL	HH	HHHHHHHHHH	; " " " "
LL	HX	XX	HHLH	HH	HHHHHHHHHH	; " " " "
LL	HX	XX	HHHL	HH	HHHHHHHHHH	; " " " "
LL	HX	XX	HHHH	HH	HHHHHHHHHH	; " " " "
LL	LH	XH	LLLL	LH	LHHHHHHHHH	;TEST BARUP
LL	LH	XH	LLLH	LH	LLHHHHHHHH	; " " "
LL	LH	XH	LLHL	LH	LLLHHHHHHH	; " " "
LL	LH	XH	LLHH	LH	LLLLHHHHHH	; " " "
LL	LH	XH	LHLL	LH	LLLLLHHHHH	; " " "
LL	LH	XH	LHLH	LH	LLLLLLHHHH	; " " "
LL	LH	XH	LHHL	LH	LLLLLLLHHH	; " " "
LL	LH	XH	LHHH	LH	LLLLLLLLHH	; " " "
LL	LH	XH	HLLL	LH	LLLLLLLLLH	; " " "
LL	LH	XH	HLLH	LL	LLLLLLLLLL	; " " "
LX	LH	XL	XXXX	LL	LLLLLLLLLL	;TEST CIMS
LX	LL	LX	XXXX	LL	LLLLLLLLLL	;TEST CILS
LL	LL	HX	LLLL	LL	LLLLLLLLLL	;TEST BARDOWN
LL	LL	HX	LLLH	HL	HLLLLLLLLL	; " " "
LL	LL	HX	LLHL	HL	HHLLLLLLLL	; " " "
LL	LL	HX	LLHH	HL	HHHLLLLLLL	; " " "
LL	LL	HX	LHLL	HL	HHHHLLLLLL	; " " "
LL	LL	HX	LHLH	HL	HHHHHLLLLL	; " " "
LL	LL	HX	LHHL	HL	HHHHHHLLLL	; " " "
LL	LL	HX	LHHH	HL	HHHHHHHLLL	; " " "
LL	LL	HX	HLLL	HL	HHHHHHHHLL	; " " "
LL	LL	HX	HLLH	HL	HHHHHHHHHL	; " " "
LH	XX	HH	XXXX	HH	HHHHHHHHHH	;TEST /DEN
HX	XX	XX	XXXX	HH	ZZZZZZZZZZ	;TEST HI-Z

NOTE:
Outputs /0 and /9 are fed back by external connection to 0F and 9F, respectively.

DESCRIPTION

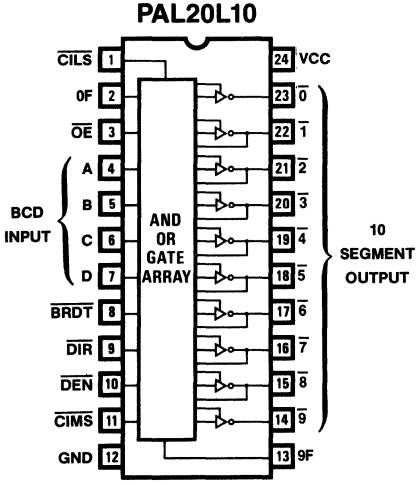
This PAL20L10 is intended for use as a ten segment bargraph display driver.

The device features three basic modes of operation:

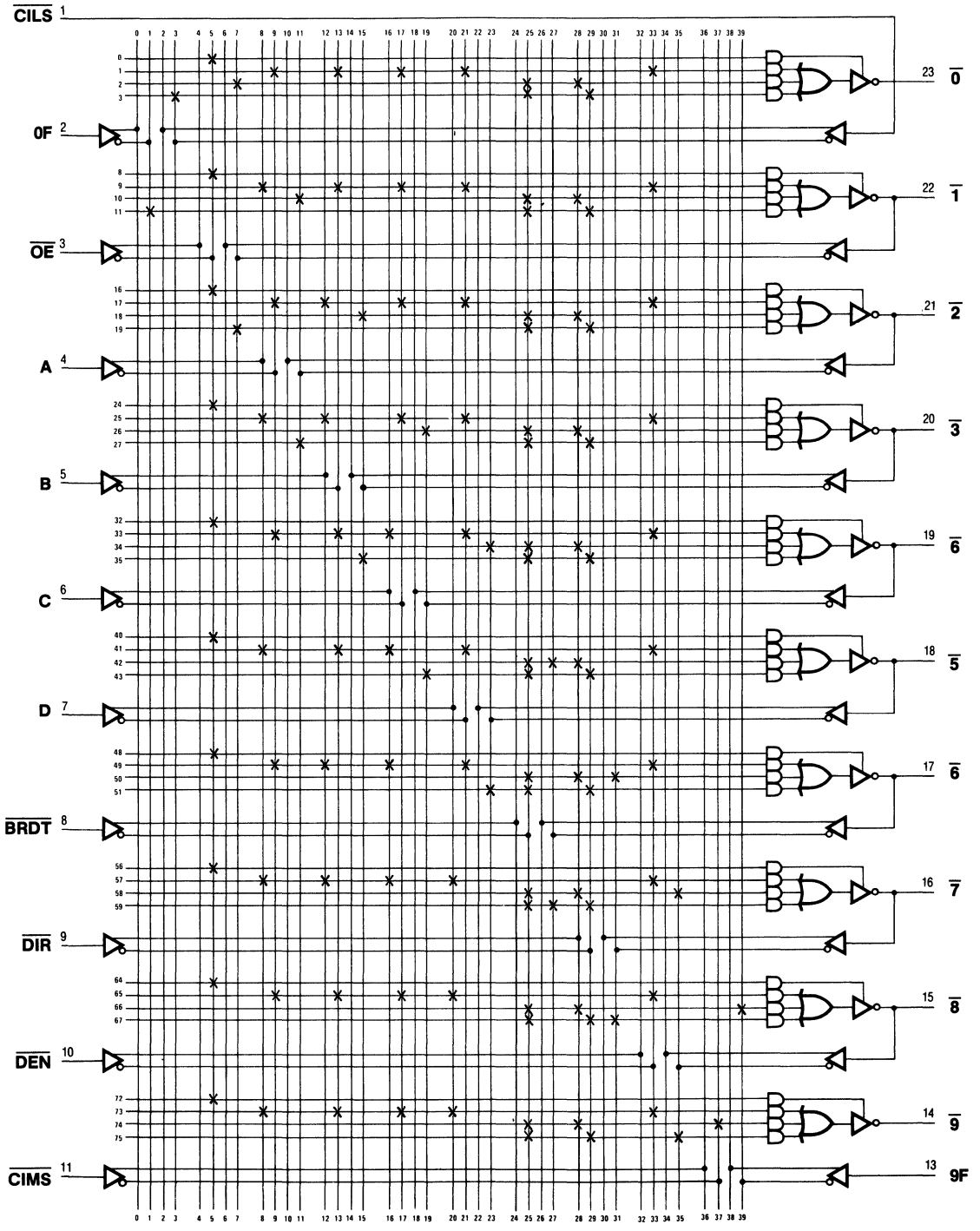
1. With the decade enable input ($\overline{DEN}$) held low and the bar/dot input ($\overline{BRDT}$) held high, the device will function in the DOTMODE, activating the one output corresponding to the given BCD input data. Invalid BCD are ignored.
2. The BARUP mode is selected by taking the $\overline{BRDT}$ input low and maintaining the bardirection input ($\overline{DIR}$) high. This mode provides activation of all lower significant segments in addition to the corresponding BCD value. Here the circuit produces a display that increases in "length" with the absolute value of the input.
3. Similarly, the BARDOWN mode provides a display that DECREASES in length with input magnitude. This mode is selected whenever the $\overline{BRDT}$ and $\overline{DIR}$ inputs are low.

Note that in both BAR modes the $\overline{DEN}$ input need not be asserted for the carry inputs ($\overline{CIMS}$ and $\overline{CILS}$) to function. This allows easy cascadability of the devices where 20 to 100 segments are required. Of course, designs with more than 100 segments are easily implemented with additional decoding circuitry.

Multiplexing between the BARUP, BARDOWN and DOT modes provides a multifunction display allowing easy visual comparison The output enable input ($\overline{OE}$) can be used to intensity modulate or "blank" the display.

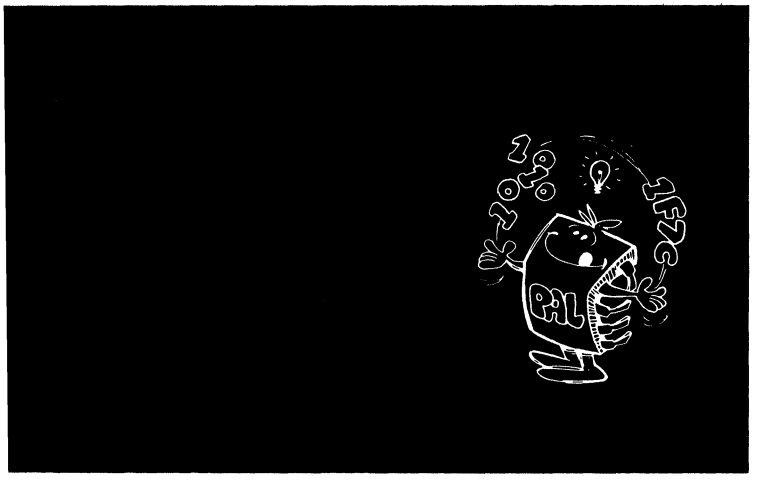


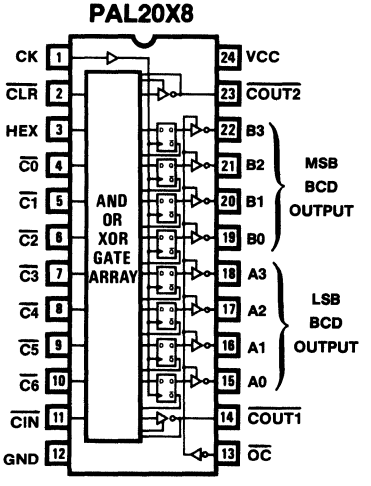
BCD To Decimal Decoder/Bargraph Display Driver Logic Diagram PAL20L10




IMPORTANT
NOTES

BCD/Hex Counter





```
PAL20X8                                                    PAL DESIGN SPECIFICATION
BHEX                                                          SAEED KAZMI 04/28/81
BCD/HEX COUNTER
MMI  SUNNYVALE, CALIFORNIA
CK /CLR HEX /C0 /C1 /C2 /C3 /C4 /C5 /C6 /CIN GND
/OC /COUT1 A0 A1 A2 A3 B0 B1 B2 B3 /COUT2 VCC


IF (VCC) COUT2 = /CLR*B0*B1*B2*B3                    ;HEX COUNT CARRY OUT
               + /CLR* /HEX*B0*B3                    ;BCD COUNT CARRY OUT

/B3 :=  CLR                                          ;CLEAR
     + /CLR*/B3                                      ;HOLD, MSB OF STAGE 2
    :+:/CLR*C0*C1*C2*C3*C4*C5*C6*CIN*B0*B1*B2        ;INCREMENT
     + /CLR* /HEX*B0*B3                              ;CLEAR IF BCD & COUNT=9

/B2 :=  CLR
     + /CLR*/B2
    :+:/CLR*C0*C1*C2*C3*C4*C5*C6*CIN*B0*B1

/B1 :=  CLR
     + /CLR*/B1
    :+:/CLR*  HEX*C0*C1*C2*C3*C4*C5*C6*CIN*B0
     + /CLR* /HEX*C0*C1*C2*C3*C4*C5*C6*CIN*B0*/B3    ;CLEAR IF BCD & COUNT=9

/B0 :=  CLR
     + /CLR*/B0                                      ;HOLD, LSB OF STAGE 2
    :+:/CLR*C0*C1*C2*C3*C4*C5*C6*CIN

/A3 :=  CLR
     + /CLR*/A3                                      ;HOLD, MSB OF STAGE 1
    :+:/CLR*C0*C1*C2*C3*C4*C5*C6*A0*A1*A2
     + /CLR* /HEX*A0*A3                              ;CLEAR IF BCD & COUNT=9

/A2 :=  CLR
     + /CLR*/A2
    :+:/CLR*C0*C1*C2*C3*C4*C5*C6*A0*A1

/A1 :=  CLR
     + /CLR*/A1
    :+:/CLR*  HEX*C0*C1*C2*C3*C4*C5*C6*A0
     + /CLR* /HEX*C0*C1*C2*C3*C4*C5*C6*A0*/A3        ;CLEAR IF BCD & COUNT=9

/A0 :=  CLR
     + /CLR*/A0                                      ;HOLD, LSB OF STAGE 1
    :+:/CLR*C0*C1*C2*C3*C4*C5*C6

IF (VCC) COUT1 = /CLR* HEX*A0*A1*A2*A3               ;HEX COUNT, INT. CARRY
               + /CLR*/HEX*A0*A3                     ;BCD COUNT, INT. CARRY
```

FUNCTION TABLE

HEX CLR CK OC C0 C1 C2 C3 C4 C5 C6
COUT2 B3 B2 B1 B0 COUT1 A3 A2 A1 A0 CIN

;H	C	C	O	C	C	C	C	C	C	C	C	B	B	B	B	C	A	A	A	A	C	
;E	L	K	C	0	1	2	3	4	5	6	O	3	2	1	0	O	3	2	1	0	I	
;X	R										U					U					N	
;											T					T						
;											2					1						COMMENT
X	H	C	H	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	CLEAR
X	H	C	H	H	H	H	H	H	H	H	L	L	L	L	L	L	L	L	L	L	H	HOLD, HEX COUNT 00
H	L	C	H	H	H	H	H	H	H	H	L	L	L	L	H	L	L	L	L	H	H	HEX COUNT 11
H	L	C	H	H	H	H	H	H	H	H	L	L	L	H	L	L	L	L	H	L	H	22
H	L	C	H	H	H	H	H	H	H	H	L	L	L	H	H	L	L	L	H	H	H	33
H	L	C	H	H	H	H	H	H	H	H	L	L	H	L	L	L	L	H	L	L	H	44
H	L	C	H	H	H	H	H	H	H	H	L	L	H	L	H	L	L	H	L	H	H	55
H	L	C	H	H	H	H	H	H	H	H	L	L	H	H	L	L	L	H	H	L	H	66
H	L	C	H	H	H	H	H	H	H	H	L	L	H	H	H	L	L	H	H	H	H	77
H	L	C	H	H	H	H	H	H	H	H	L	H	L	L	L	L	H	L	L	L	H	88
H	L	C	H	H	H	H	H	H	H	H	L	H	L	L	H	L	H	L	L	H	H	99
H	L	C	H	H	H	H	H	H	H	H	L	H	L	H	L	L	H	L	H	L	H	AA
H	L	C	H	H	H	H	H	H	H	H	L	H	L	H	H	L	H	L	H	H	H	BB
H	L	C	H	H	H	H	H	H	H	H	L	H	H	L	L	L	H	H	L	L	H	CC
H	L	C	H	H	H	H	H	H	H	H	L	H	H	L	H	L	H	H	L	H	H	DD
H	L	C	H	H	H	H	H	H	H	H	L	H	H	H	L	L	H	H	H	L	H	EE
H	L	C	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	FF
H	L	C	H	H	H	H	H	H	H	H	L	L	L	L	L	L	L	L	L	L	H	BCD COUNT 00
L	L	C	H	H	H	H	H	H	H	H	L	L	L	L	H	L	L	L	L	H	H	11
L	L	C	H	H	H	H	H	H	H	H	L	L	L	H	L	L	L	L	H	L	H	22
L	L	C	H	H	H	H	H	H	H	H	L	L	L	H	H	L	L	L	H	H	H	33
L	L	C	H	H	H	H	H	H	H	H	L	L	H	L	L	L	L	H	L	L	H	44
L	L	C	H	H	H	H	H	H	H	H	L	L	H	L	H	L	L	H	L	H	H	55
L	L	C	H	H	H	H	H	H	H	H	L	L	H	H	L	L	L	H	H	L	H	66
L	L	C	H	H	H	H	H	H	H	H	L	L	H	H	H	L	L	H	H	H	H	77
L	L	C	H	H	H	H	H	H	H	H	L	H	L	L	L	L	H	L	L	L	H	88
L	L	C	H	H	H	H	H	H	H	H	H	L	L	H	H	H	L	L	H	H		99
L	L	C	H	H	H	H	H	H	H	H	L	L	L	L	L	L	L	L	L	L	H	100

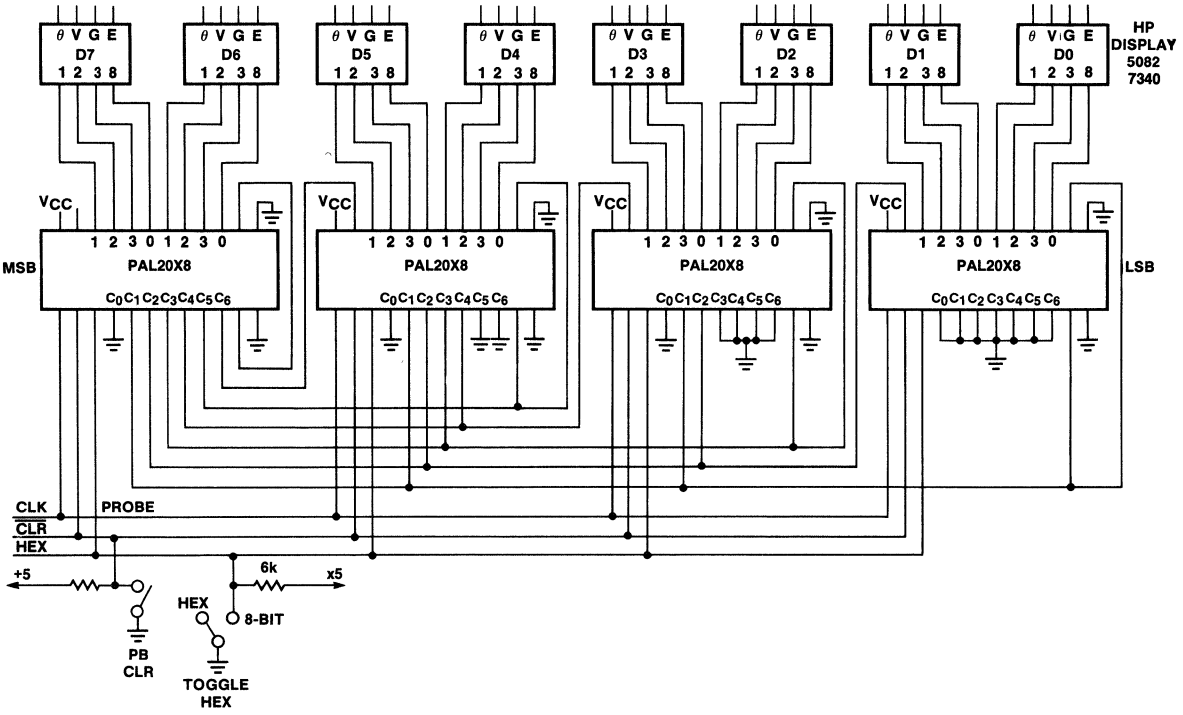
DESCRIPTION

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

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

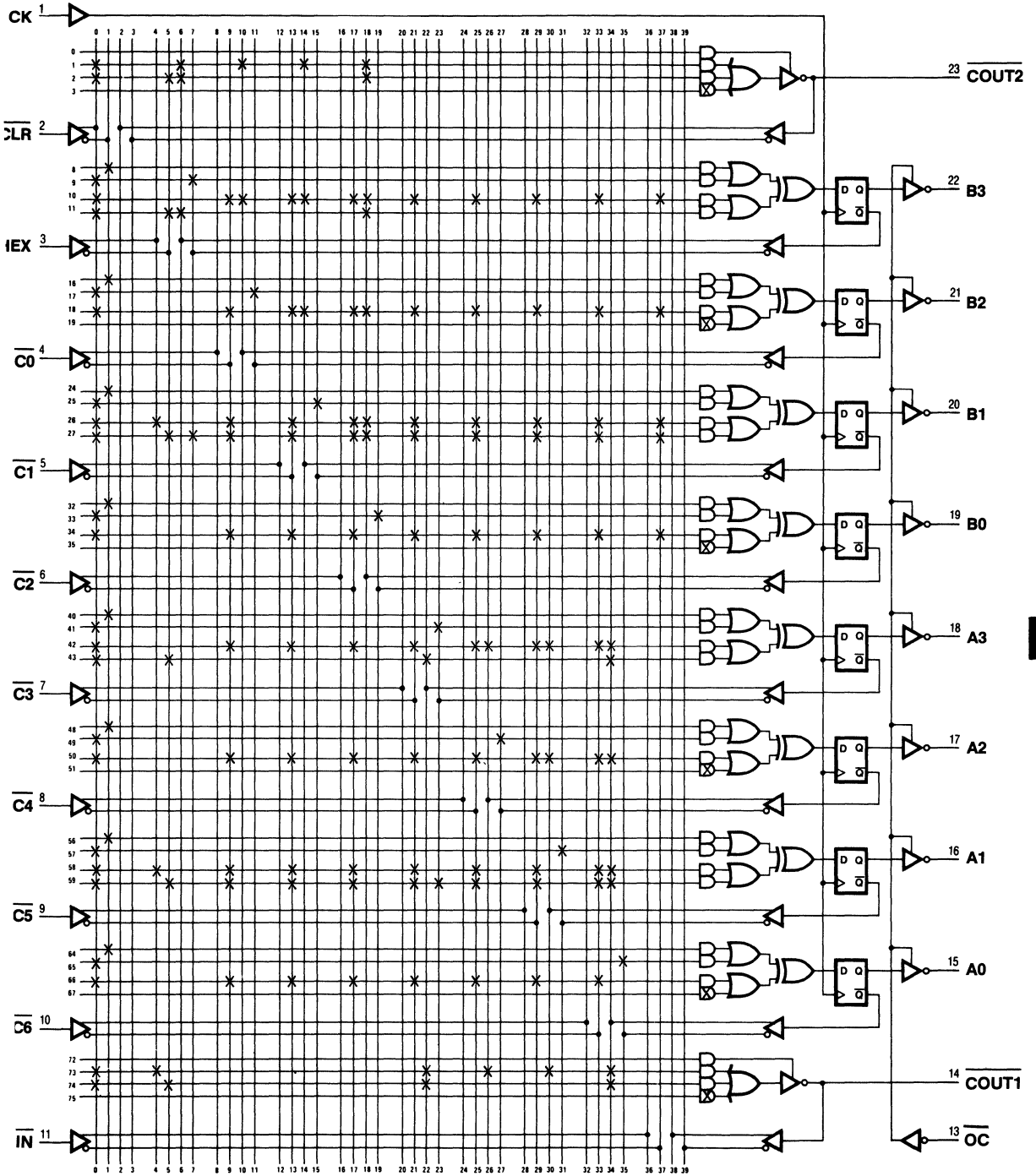
THIS COUNTER OPERATES AT 10 MHz AND CAN PERFORM THE FOLLOWING OPERATIONS:

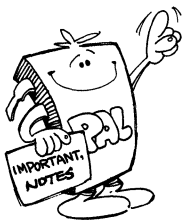
HEX	CLR	OPERATION
X	H	CLEAR
L	L	COUNT BCD
H	L	COUNT HEX



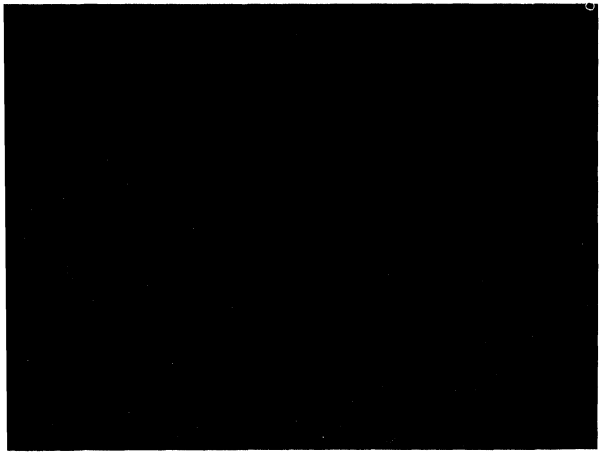
BCD/HEX Counter

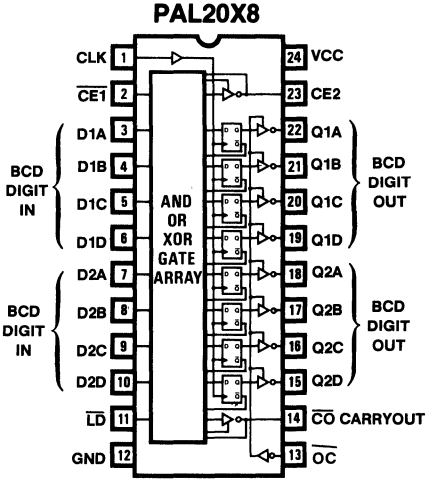
Logic Diagram PAL20X8




IMPORTANT
NOTES

2-Digit BCD Counter





6

```
PAL20X8                                                PAL DESIGN SPECIFICATION
PMSI403                                                   VINCENT COLI 06/21/82
2-DIGIT BCD COUNTER
MMI SUNNYVALE, CALIFORNIA
CLK /CE1 D1A D1B D1C D1D D2A D2B D2C D2D /LD GND
/OC /CO  Q2D Q2C Q2B Q2A Q1D Q1C Q1B Q1A CE2 VCC


/Q1A :=  /LD*/Q1A                                  ;HOLD FIRST BIT OF 1 DECIMAL
      +   LD*/D1A                                  ;LOAD D1A
     :+: /LD* CE1* CE2                             ;COUNT

/Q1B :=  /LD*/Q1B                                  ;HOLD SECOND BIT OF 1 DECIMAL
      +   LD*/D1B                                  ;LOAD D1B
     :+: /LD* CE1* CE2* Q1A*/Q1D                   ;COUNT

/Q1C :=  /LD*/Q1C                                  ;HOLD THIRD BIT OF 1 DECIMAL
      +   LD*/D1C                                  ;LOAD D1C
     :+: /LD* CE1* CE2* Q1A* Q1B                   ;COUNT

/Q1D :=  /LD*/Q1D                                  ;HOLD FOURTH BIT OF 1 DECIMAL
      +   LD*/D1D                                  ;LOAD D1D
     :+: /LD* CE1* Q1A* Q1B* Q1C                   ;COUNT
      +  /LD* CE1* CE2* Q1A* Q1D                   ;ROLL OVER

/Q2A :=  /LD*/Q2A                                  ;HOLD FIRST BIT OF 10 DECIMAL
      +   LD*/D2A                                  ;LOAD D2A
     :+: /LD* CE1* CE2* Q1A* Q1D                   ;COUNT

/Q2B :=  /LD*/Q2B                                  ;HOLD SECOND BIT OF 10 DECIMAL
      +   LD*/D2B                                  ;LOAD D2B
     :+: /LD* CE1* CE2* Q1A* Q1D* Q2A*/Q2D         ;COUNT

/Q2C :=  /LD*/Q2C                                  ;HOLD THIRD BIT OF 10 DECIMAL
      +   LD*/D2C                                  ;LOAD D2C
     :+: /LD* CE1* CE2* Q1A* Q1D* Q2A* Q2B         ;COUNT

/Q2D :=  /LD*/Q2D                                  ;HOLD FOURTH BIT OF 10 DECIMAL
      +   LD*/D2D                                  ;LOAD D2D
     :+: /LD* CE1* CE2* Q1A* Q1D* Q2A* Q2B* Q2C    ;COUNT
      +  /LD* CE1* CE2* Q1A* Q1D* Q2A* Q2D         ;ROLL OVER

IF (VCC)  CO =  Q1A* Q1D* Q2A* Q2D                 ;CARRY OUT (10 DECIMAL)
```

FUNCTION TABLE

CLK /OC /LD /CE1 CE2 D2D D2C D2B D2A D1D D1C D1B D1A /CO
Q2D Q2C Q2B Q2A Q1D Q1C Q1B Q1A

; CHIP					--INPUTS---			--OUTPUTS--		
;CONTROL		INSTRUCTIONS			BCD 2	BCD 1		BCD 2	BCD 1	COMMENT
;CLK	/OC	/LD	/CE1	CE2	DCBA	DCBA	/CO	DCBA	DCBA	(DECIMAL VALUE)
C	L	L	L	H	LLLL	LLLL	H	LLLL	LLLL	LOAD (00) /LD=L
C	L	H	H	H	XXXX	XXXX	H	LLLL	LLLL	HOLD (00)/CE1=H
C	L	H	L	H	XXXX	XXXX	H	LLLL	LLLH	COUNT (01)
C	L	H	L	H	XXXX	XXXX	H	LLLL	LLHL	COUNT (02)
C	L	H	L	H	XXXX	XXXX	H	LLLL	LLHH	COUNT (03)
C	L	H	L	H	XXXX	XXXX	H	LLLL	LHLL	COUNT (04)
C	L	H	L	H	XXXX	XXXX	H	LLLL	LHLH	COUNT (05)
C	L	H	L	H	XXXX	XXXX	H	LLLL	LHHL	COUNT (06)
C	L	H	L	H	XXXX	XXXX	H	LLLL	LHHH	COUNT (07)
C	L	H	L	H	XXXX	XXXX	H	LLLL	HLLL	COUNT (08)
C	L	H	L	H	XXXX	XXXX	H	LLLL	HLLH	COUNT (09)
C	L	H	L	H	XXXX	XXXX	H	LLLH	LLLL	COUNT (10)
C	L	H	L	H	XXXX	XXXX	H	LLLH	LLLH	COUNT (11)
C	L	L	H	L	LLLH	HLLH	H	LLLH	HLLH	LOAD (19)
C	L	H	L	H	XXXX	XXXX	H	LLHL	LLLL	COUNT (20)
C	L	L	H	L	LLHL	HLLH	H	LLHL	HLLH	LOAD (29)
C	L	H	L	H	XXXX	XXXX	H	LLHH	LLLL	COUNT (30)
C	L	L	H	L	LLHH	HLLH	H	LLHH	HLLH	LOAD (39)
C	L	H	L	H	XXXX	XXXX	H	LHLL	LLLL	COUNT (40)
C	L	L	H	L	LHLL	HLLH	H	LHLL	HLLH	LOAD (49)
C	L	H	L	H	XXXX	XXXX	H	LHLH	LLLL	COUNT (50)
C	L	L	H	L	LHLH	HLLH	H	LHLH	HLLH	LOAD (59)
C	L	H	L	H	XXXX	XXXX	H	LHHL	LLLL	COUNT (60)
C	L	L	H	L	LHHL	HLLH	H	LHHL	HLLH	LOAD (69)
C	L	H	L	H	XXXX	XXXX	H	LHHH	LLLL	COUNT (70)
C	L	L	H	L	LHHH	HLLH	H	LHHH	HLLH	LOAD (79)
C	L	H	L	H	XXXX	XXXX	H	HLLL	LLLL	COUNT (80)
C	L	L	H	L	HLLL	HLLH	H	HLLL	HLLH	LOAD (89)
C	L	H	L	H	XXXX	XXXX	H	HLLH	LLLL	COUNT (90)
C	L	L	H	L	HLLH	HLLH	L	HLLH	HLLH	LOAD (99) /CO=L
C	L	H	L	H	XXXX	XXXX	H	LLLL	LLLL	COUNT (00)
C	L	H	L	L	XXXX	XXXX	H	LLLL	LLLL	HOLD (00) CE2=L
X	H	X	X	X	XXXX	XXXX	X	ZZZZ	ZZZZ	TEST HI-Z

DESCRIPTION

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

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

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

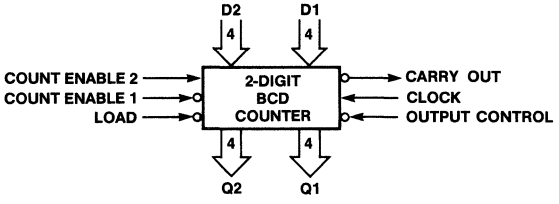
TWO OR MORE BCD COUNTERS CAN BE CASCADED TO IMPLEMENT LARGER BCD COUNTERS BY CONNECTING CARRY OUT (/CO) OF THE FIRST STAGE TO COUNT ENABLE (/CE1) OF THE SECOND STAGE.

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

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

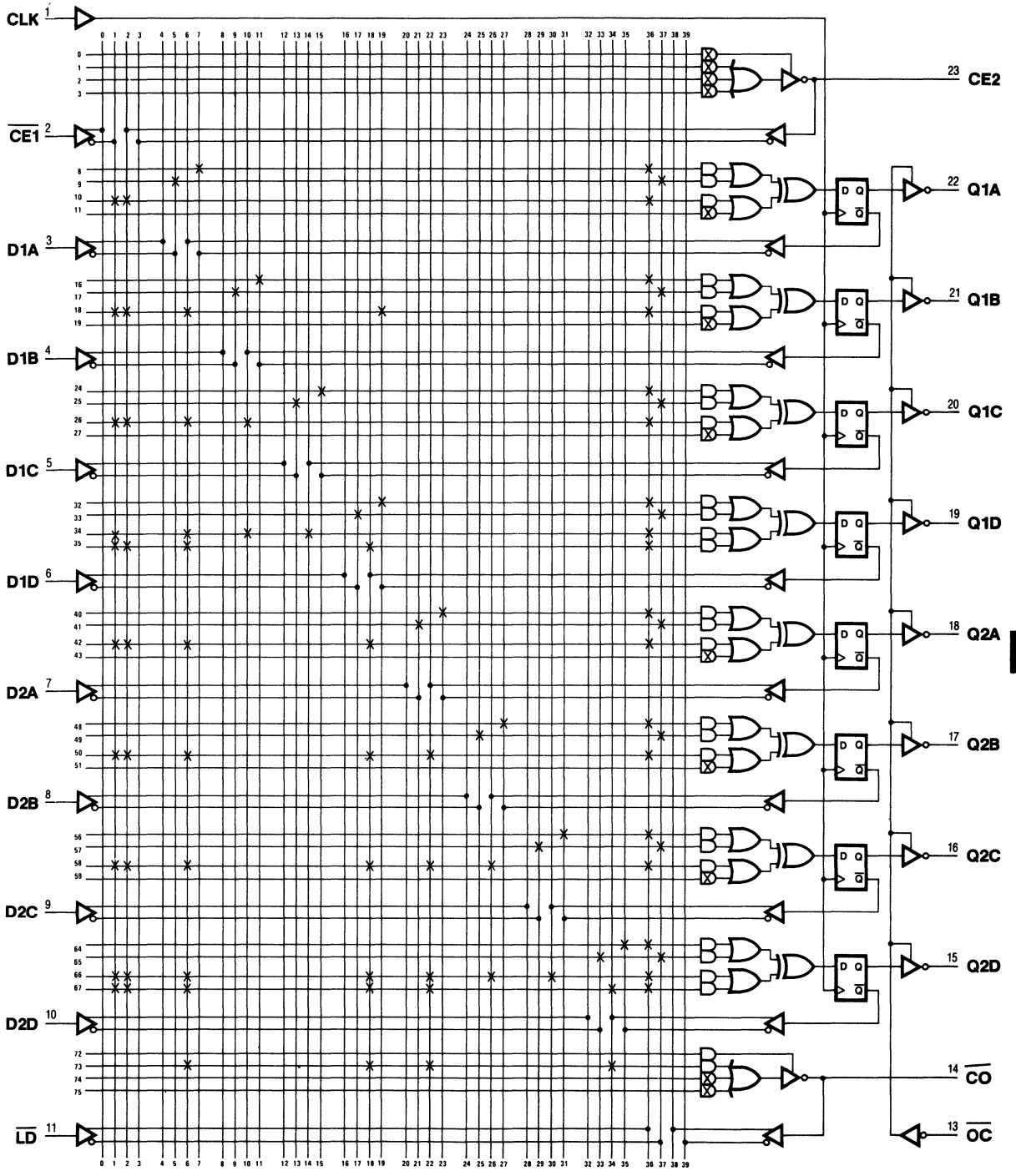
THESE OPERATIONS ARE EXERCISED IN THE FUNCTION TABLE AND SUMMARIZED IN THE OPERATIONS TABLE:

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



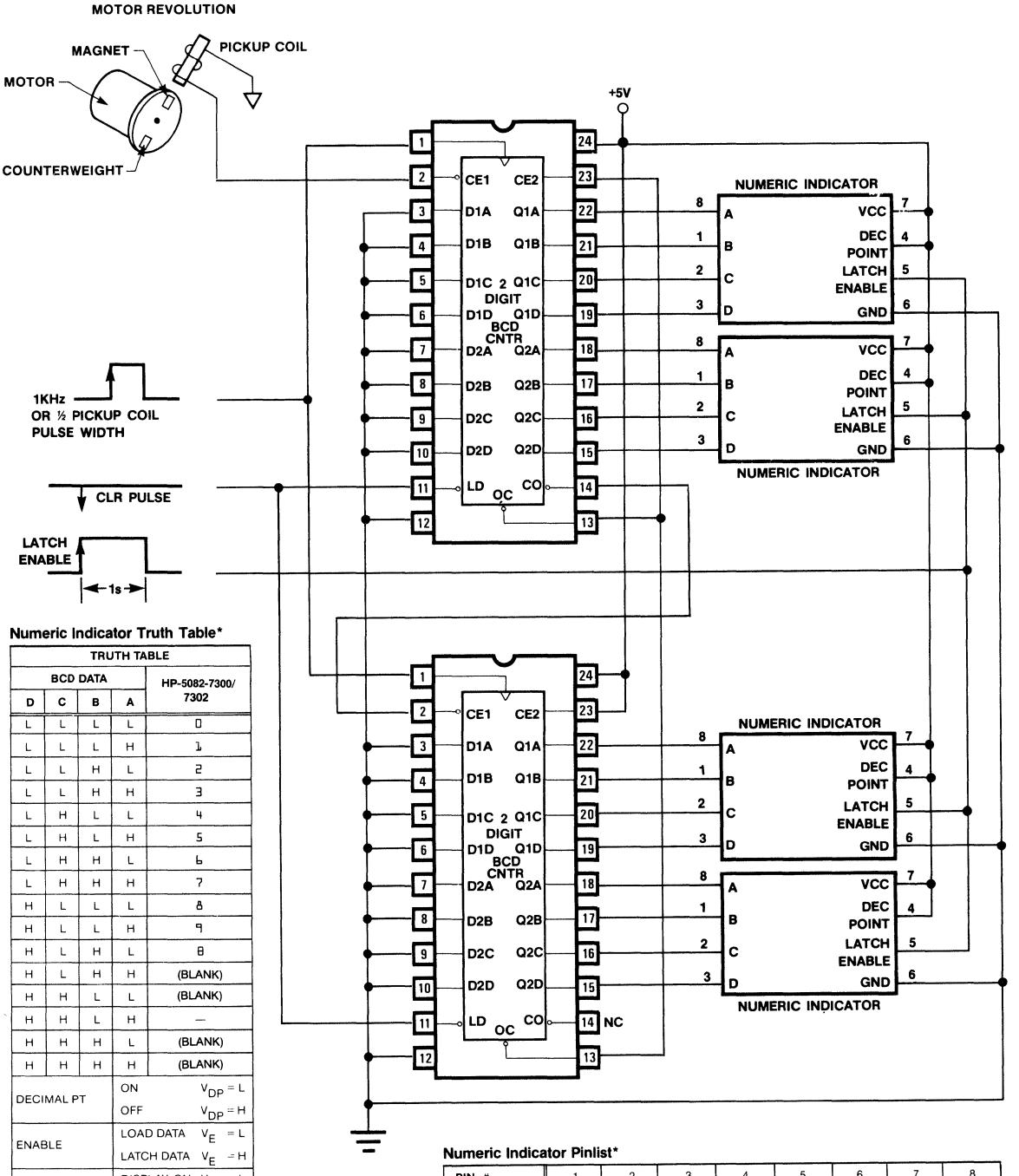
2-Digit BCD Counter

Logic Diagram PAL20X8



Application

Motor Revolution Per Second Indicator



Numeric Indicator Truth Table*

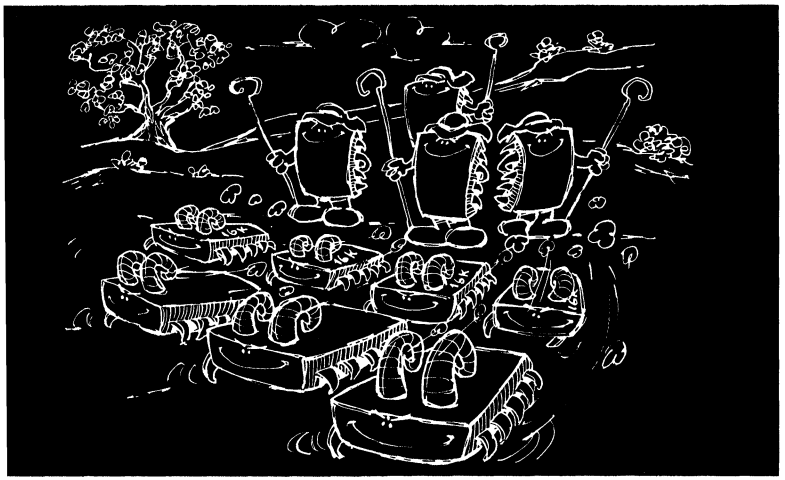
TRUTH TABLE				
BCD DATA				HP-5082-7300/ 7302
D	C	B	A	
L	L	L	L	0
L	L	L	H	1
L	L	H	L	2
L	L	H	H	3
L	H	L	L	4
L	H	L	H	5
L	H	H	L	6
L	H	H	H	7
H	L	L	L	8
H	L	L	H	9
H	L	H	L	8
H	L	H	H	(BLANK)
H	H	L	L	(BLANK)
H	H	L	H	—
H	H	H	L	(BLANK)
H	H	H	H	(BLANK)
DECIMAL PT				ON V_{DP} = L OFF V_{DP} = H
ENABLE				LOAD DATA V_E = L LATCH DATA V_E = H
BLANKING				DISPLAY-ON V_B = L DISPLAY-OFF V_B = H

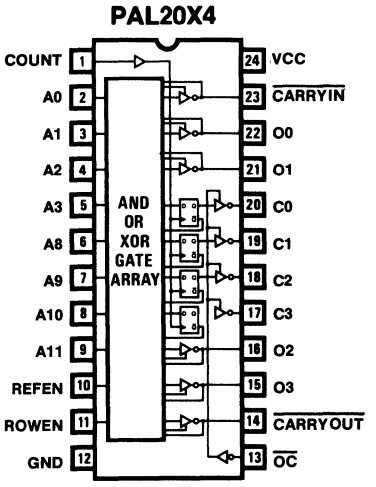
Numeric Indicator Pinlist*

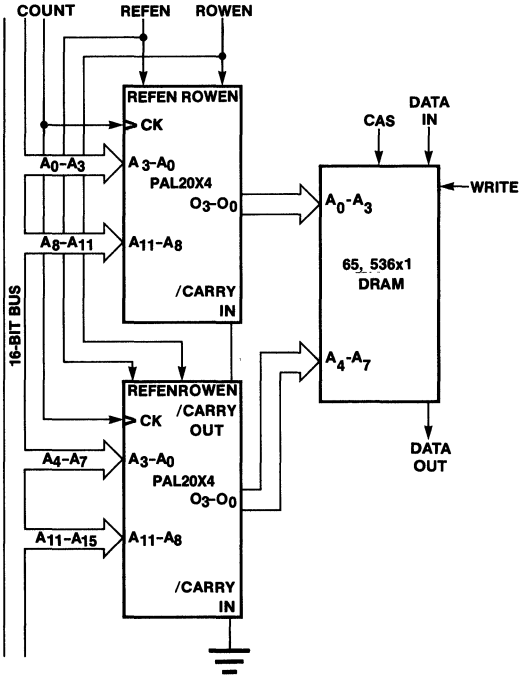
PIN #	1	2	3	4	5	6	7	8
FUNCTION	B	C	D	Decimal Point	Latch Enable	GND	VCC	A

*Consult the Hewlett-Packard "Optoelectronics Designer's Catalog" for more details

64K Dynamic RAM Refresh Controller






COUNT
REFEN
ROWEN
REFEN ROWEN
CK
A0-A3
A3-A0
PAL20X4
O3-O0
A8-A11
A11-A8
/CARRY
IN
16-BIT BUS
REFENROWEN
/CARRY
OUT
CK
A4-A7
A3-A0
PAL20X4
O3-O0
A11-A15
A11-A8
/CARRY
IN
DATA
IN
CAS
WRITE
A0-A3
65,536x1
DRAM
A4-A7
DATA
OUT

```
PAL20X4                                          PAL DESIGN SPECIFICATION
P7012                                            MIKE VOLPIGNO   07/15/81
64K DYNAMIC RAM REFRESH CONTROLLER
MMI FIELD APPLICATIONS ENGINEER NEWTON, MASSACHUSETTS
COUNT A0 A1 A2 A3 A8 A9 A10 A11 REFEN ROWEN GND
/OC /CARRYOUT O3 O2 C3 C2 C1 C0 O1 O0 /CARRYIN VCC


        /C0 :=  /C0                          ;INCR REF ADDRESS COUNTER
             +   REFEN*/ROWEN                ;SET REF COUNTER
            :+:  CARRYIN                     ;INCR REF COUNTER
             +   REFEN*/ROWEN                ;SET REF COUNTER

        /C1 :=  /C1                          ;INCR REF COUNTER
             +   REFEN*/ROWEN                ;SET REF COUNTER
            :+:  CARRYIN* C0                 ;INCR REF COUNTER
             +   REFEN*/ROWEN                ;SET REF COUNTER

        /C2 :=  /C2                          ;INCR REF COUNTER
             +   REFEN*/ROWEN                ;SET REF COUNTER
            :+:  CARRYIN* C0* C1             ;INCR REF COUNTER
             +   REFEN*/ROWEN                ;SET REF COUNTER

        /C3 :=  /C3                          ;INCR REF COUNTER
             +   REFEN*/ROWEN                ;SET REF COUNTER
            :+:  CARRYIN* C0* C1* C2         ;INCR REF COUNTER
             +   REFEN*/ROWEN                ;SET REF COUNTER

IF (VCC)  /O0 =  /A0*/REFEN* ROWEN           ;SELECT LOWER ADDRESS
             +   /A8*/REFEN*/ROWEN           ;SELECT UPPER ADDRESS
             +   /C0* REFEN* ROWEN           ;SELECT REFRESH ADDRESS

IF (VCC)  /O1 =  /A1*/REFEN* ROWEN           ;SELECT LOWER ADDRESS
             +   /A9*/REFEN*/ROWEN           ;SELECT UPPER ADDRESS
             +   /C1* REFEN* ROWEN           ;SELECT REFRESH ADDRESS

IF (VCC)  /O2 =  /A2* /REFEN* ROWEN          ;SELECT LOWER ADDRESS
             +   /A10*/REFEN*/ROWEN          ;SELECT UPPER ADDRESS
             +   /C2*  REFEN* ROWEN          ;SELECT REFRESH ADDRESS

IF (VCC)  /O3 =  /A3* /REFEN* ROWEN          ;SELECT LOWER ADDRESS
             +   /A11*/REFEN*/ROWEN          ;SELECT UPPER ADDRESS
             +   /C3*  REFEN* ROWEN          ;SELECT REFRESH ADDRESS

IF (VCC) CARRYOUT =  CARRYIN* C0* C1* C2* C3 ;CARRY OUT
```

FUNCTION TABLE

A0 A1 A2 A3 A8 A9 A10 A11 COUNT /OC REFEN ROWEN /CARRYIN /CARRYOUT
O3 O2 O1 O0 C3 C2 C1 C0

;--DATA-					--CARRIES--				
;AAAAAAAA	----CONTROLS---				/	/			
; 11	COU	/	REF	ROW	CARRY	CARRY	OOOO	CCCC	
;32101098	NT	OC	EN	EN	IN	OUT	3210	3210	COMMENTS
XXXXXXXX	C	L	H	L	H	H	HHHH	HHHH	SET REF CNTR
XXXXLLLL	C	L	L	L	H	H	LLLL	HHHH	UPPER ADDR LOW
XXXXHHHH	C	L	L	L	H	H	HHHH	HHHH	UPPER ADDR HI
LLLLXXXX	C	L	L	H	H	H	LLLL	HHHH	LOWER ADDR LOW
HHHHLLLL	C	L	L	H	H	H	HHHH	HHHH	LOWER ADDR HI
XXXXXXXX	C	L	H	H	L	H	LLLL	LLLL	INCR REF ADDR
XXXXXXXX	C	L	H	H	L	H	LLLH	LLLH	
XXXXXXXX	C	L	H	H	L	H	LLHL	LLHL	
XXXXXXXX	C	L	H	H	L	H	LLHH	LLHH	
XXXXXXXX	C	L	H	H	L	H	LHLL	LHLL	
XXXXXXXX	C	L	H	H	L	H	LHLH	LHLH	
XXXXXXXX	C	L	H	H	L	H	LHHL	LHHL	
XXXXXXXX	C	L	H	H	L	H	LHHH	LHHH	
XXXXXXXX	C	L	H	H	L	H	HLLL	HLLL	
XXXXXXXX	C	L	H	H	L	H	HLLH	HLLH	
XXXXXXXX	C	L	H	H	L	H	HLHL	HLHL	
XXXXXXXX	C	L	H	H	L	H	HLHH	HLHH	
XXXXXXXX	C	L	H	H	L	H	HHLL	HHLL	
XXXXXXXX	C	L	H	H	L	H	HHLH	HHLH	
XXXXXXXX	C	L	H	H	L	H	HHHL	HHHL	
XXXXXXXX	C	L	H	H	L	L	HHHH	HHHH	CARRY OUT
XXXXXXXX	X	H	X	X	X	X	XXXX	ZZZZ	TEST HI-Z

DESCRIPTION

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

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

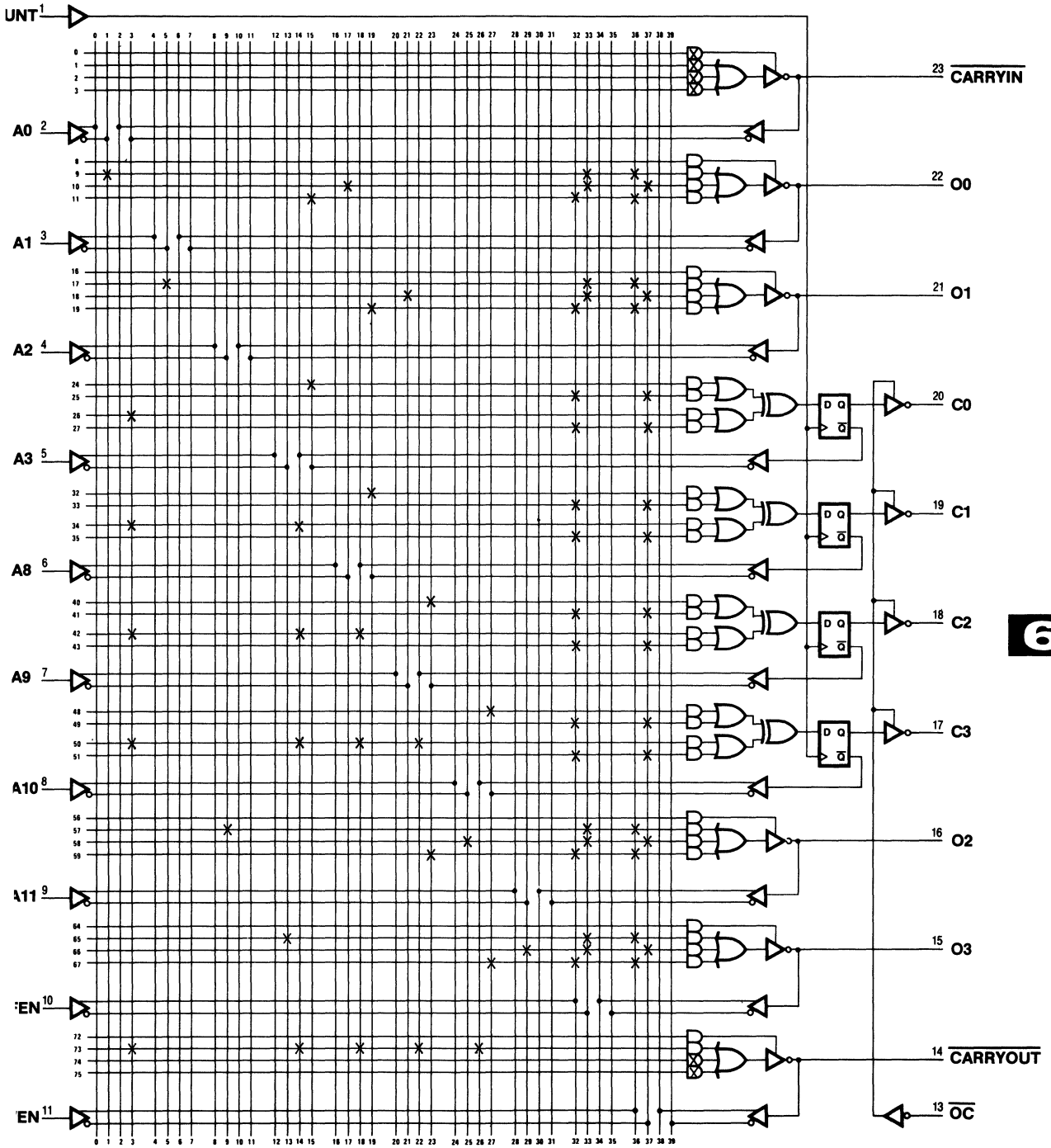
AN ADDRESS COUNTER (C3-C0) IS SELECTED DURING REFRESH WHEN ROW ENABLE (ROWEN) IS HIGH.

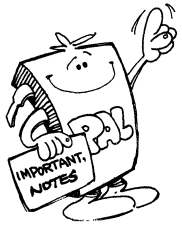
THESE OPERATIONS ARE EXERCISED IN THE FUNCTION TABLE AND SUMMERIZED IN THE OPERATIONS TABLE:

/OC	COUNT	REFEN	ROWEN	O3-O0	OPERATION
H	X	X	X	Z	HI-Z
L	C	L	L	A3-A0	SELECT LOW ADDR BITS
L	C	L	H	A11-A8	SELECT UPPER ADDR BITS
L	C	H	H	C3-C0	SELECT REFRESH ADDR BITS
L	C	H	L	H	SET REFRESH COUNTER

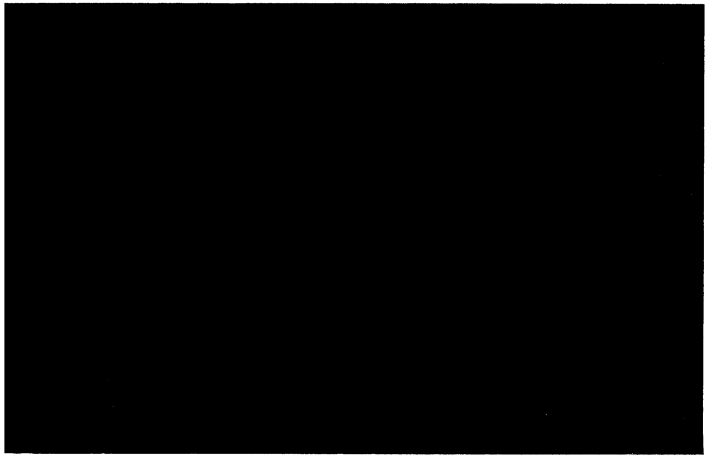
64K Dynamic RAM Refresh Controller

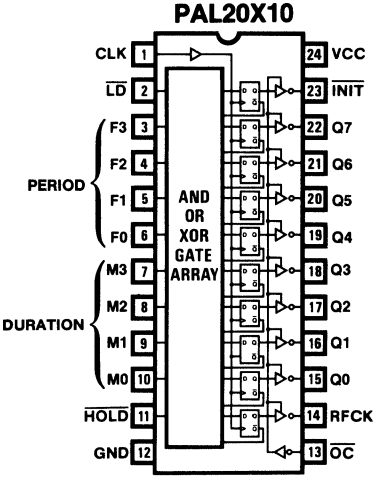
Logic Diagram PAL20X4




IMPORTANT
NOTES

Refresh Clock Generator





6

```
PAL20X10                                                    PAL DESIGN SPECIFICATION
RFCKGEN                                                          FRANK LEE 08/03/82
REFRESH CLOCK GENERATOR
MMI SUNNYVALE, CALIFORNIA
CLK /LD F3 F2 F1 F0 M3 M2 M1 M0 /HOLD GND
/OC RFCK Q0 Q1 Q2 Q3 Q4 Q5 Q6 Q7 /INIT VCC


INIT   :=  /Q7*/Q6*/Q5*/Q4*/Q3*/Q2*/Q1*/Q0          ;END OF COUNT
        +  LD                                       ;LOAD

/Q0    :=  /INIT*/Q0                                ;DECREMENT
       :+: /INIT*/HOLD                              ;HOLD

/Q1    :=  /INIT*/Q1                                ;DECREMENT
       :+: /INIT*/HOLD*/Q0                          ;HOLD

/Q2    :=   INIT                                    ;LOAD IN 0
        +  /INIT*/Q2                                ;DECREMENT
       :+: /INIT*/HOLD*/Q0*/Q1                      ;HOLD

/Q3    :=  /INIT*/Q3                                ;DECREMENT
       :+: /INIT*/HOLD*/Q0*/Q1*/Q2                  ;HOLD

/Q4    :=   INIT*/F0                                ;LOAD IN F0
        +  /INIT*/Q4                                ;DECREMENT
       :+: /INIT*/HOLD*/Q0*/Q1*/Q2*/Q3              ;HOLD

/Q5    :=   INIT*/F1                                ;LOAD IN F1
        +  /INIT*/Q5                                ;DECREMENT
       :+: /INIT*/HOLD*/Q0*/Q1*/Q2*/Q3*/Q4          ;HOLD

/Q6    :=   INIT*/F2                                ;LOAD IN F2
        +  /INIT*/Q6                                ;DECREMENT
       :+: /INIT*/HOLD*/Q0*/Q1*/Q2*/Q3*/Q4*/Q5      ;HOLD

/Q7    :=   INIT*/F3                                ;LOAD IN F3
        +  /INIT*/Q7                                ;DECREMENT
       :+: /INIT*/HOLD*/Q0*/Q1*/Q2*/Q3*/Q4*/Q5*/Q6  ;HOLD

/RFCK  :=  M3*/Q7*/Q6* Q5                           ;32 LOW STATES
        +  M2*/Q7*/Q6*/Q5* Q4                       ;16 LOW STATES
       :+: M1*/Q7*/Q6*/Q5*/Q4* Q3                   ;8 LOW STATES
        +  M0*/Q7*/Q6*/Q5*/Q4*/Q3* Q2               ;4 LOW STATES
```

FUNCTION TABLE

```
CLK /OC /LD /HOLD F3 F2 F1 F0 M3 M2 M1 M0
/INIT Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0 RFCK

;      /            /
;      H            I          R
;C / / O            N          F
;L O L L FFFF MMMM   I QQQQQQQQ C
;K C D D 3210 3210   T 76543210 K    COMMEMTS
-------------------------------------------------------------------------
 C L L X LLHH LLHH   L XXXXXXXX X    READY TO LOAD COUNTER
 C L L H LLHH LLHH   L LLHHHLHH X    READY TO LOAD COUNTER
 C L H H LLHH LLHH   H LLHHHLHH H    COUNTER LOADED, READY TO COUNT DOWN
 C L H H LLHH LLHH   H LLHHHLHL H    COUNT DOWN
 C L H H LLHH LLHH   H LLHHHLLH H    COUNT DOWN
 C L L H LLLH LHHL   L LLHHHLLL H    READY TO LOAD COUNTER, COUNT DOWN
 C L H H LLLH LHHL   H LLLHHLHH H    COUNTER LOADED
 C L H H LLLH LHHL   H LLLHHLHL L    COUNT DOWN
 C L H H LLLH LHHL   H LLLHHLLH L    COUNT DOWN
 C L H H LLLH LHHL   H LLLHHLLL L    COUNT DOWN
 C L L H LLLH LHHL   L LLLHLHHH L    READY TO LOAD COUNTER, COUNT DOWN
 C L H H LLLL LHHL   H LLLLHLHH L    COUNTER LOADED
 C L H H LLLL LHHL   H LLLLHLHL L    COUNT DOWN
 C L H H LLLL LHHL   H LLLLHLLH L    COUNT DOWN
 C L H H LLLL LHHL   H LLLLHLLL L    COUNT DOWN
 C L H H LLLL LHHL   H LLLLLHHH L    COUNT DOWN
 C L H H LLLL LHHL   H LLLLLHHL H    COUNT DOWN, RFCK HIGH
 C L H H LLLL LHHL   H LLLLLHLH H    COUNT DOWN, RFCK HIGH
 C L H L LLLL LHHL   H LLLLLHLH H    HOLD
 C L H H LLLL LHHL   H LLLLLHLL H    COUNT DOWN
 C L H H LLLL LHHL   H LLLLLLHH H    COUNT DOWN
 C L H H LLLL LHHL   H LLLLLLHL H    COUNT DOWN
 C L H H LLLL LHHL   H LLLLLLLH H    COUNT DOWN
 C L H H LLLL LHHL   H LLLLLLLL H    COUNT DOWN
 C L H H LLLL LHHL   L HIHHHHHH H    COUNT DOWN, READY TO LOAD COUNTER
 C H X X XXXX XXXX   Z ZZZZZZZZ Z    HIGH IMPEDANCE TEST
-------------------------------------------------------------------------
```

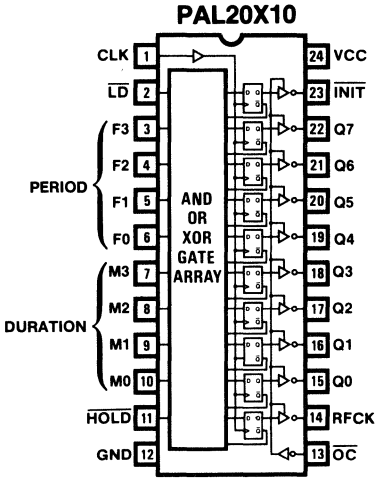
DESCRIPTION

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

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

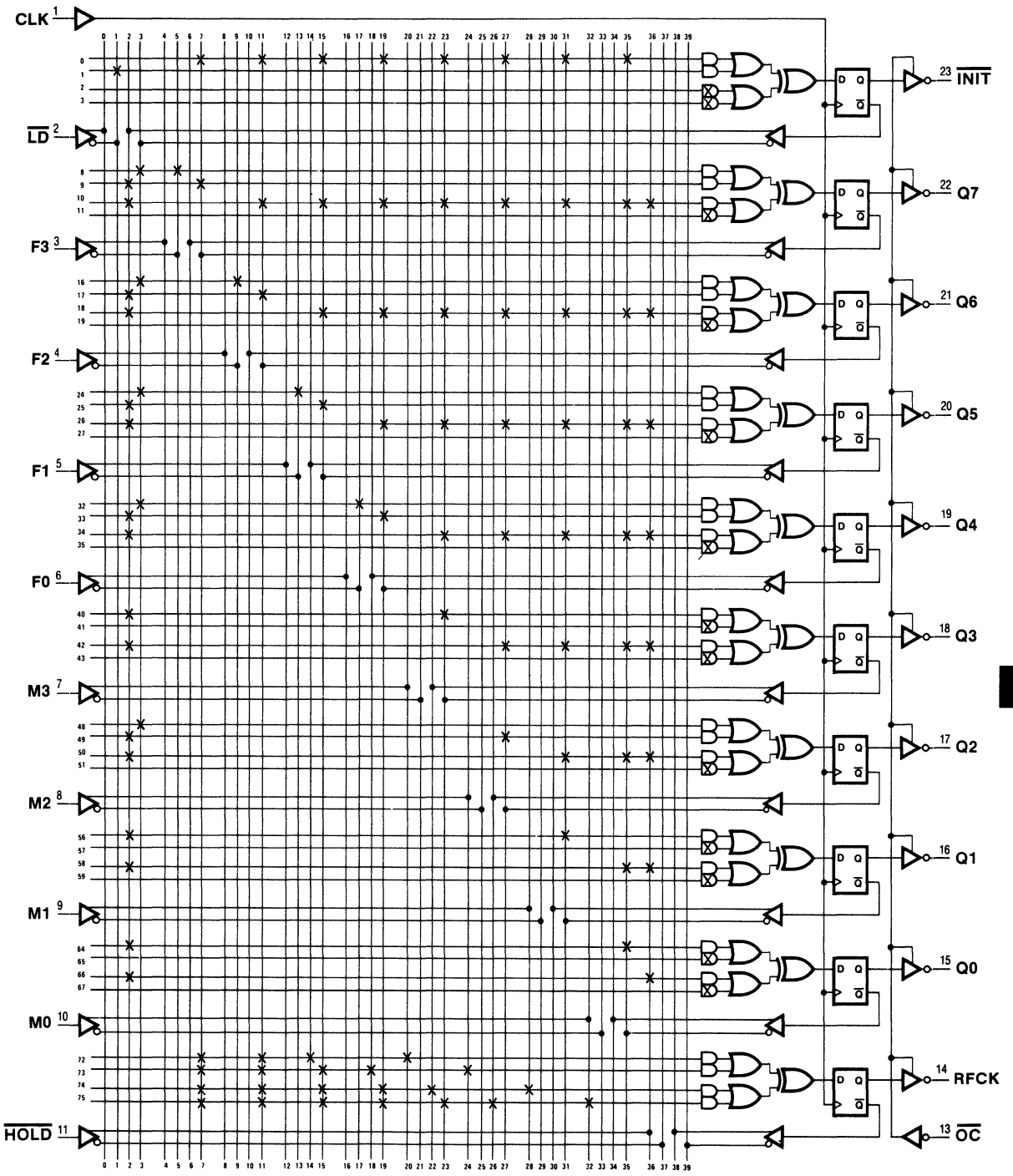
MMMM 3210	RFCK LOW DURATION (CYCLES)
0000	0
0001	4
0010	8
0011	12
0100	16
0101	20*
0110	24
0111	28
1000	32
1001	36*
1010	40*
1011	44*
1100	48
1101	52*
1110	56
1111	60

*NOT ALLOWED DUE TO BAD WAVEFORMS



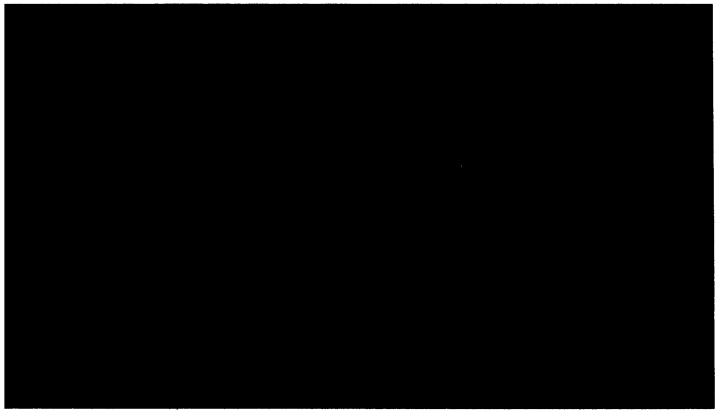
Refresh Clock Generator

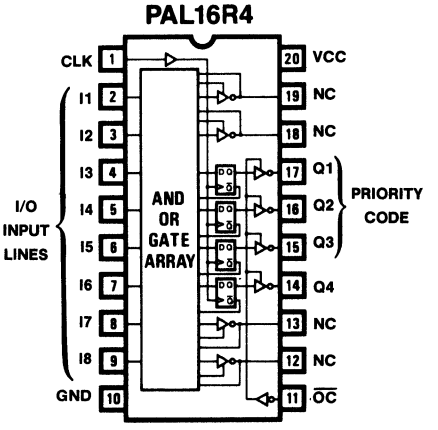
Logic Diagram PAL20X10




IMPORTANT
NOTES

8-Bit I/O Priority Interrupt Encoder With Registers





6

```
PAL16R4                                                PAL DESIGN SPECIFICATION
8BITENC                                                  VINCENT COLI 06/28/81
8-BIT I/O PRIORITY INTERRUPT ENCODER WITH REGISTERS
MMI SUNNYVALE, CALIFORNIA
CLK I1 I2 I3 I4 I5 I6 I7 I8 GND
/OC 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   ;INTERRUPT FLAG


FUNCTION TABLE

I8 I7 I6 I5 I4 I3 I2 I1 CLK /OC Q4 Q3 Q2 Q1

;-INPUTS-   CONTROL   OUTPUTS   COMMENTS
;IIIIIIII   CLK /OC    QQQQ
;87654321              4321
--------------------------------------------------------------------------
 LXXXXXXH    C   L     LHHH     I1 INTERRUPT (HIGHEST PRIORITY DEVICE)
 LXXXXXHL    C   L     LHHL     I2 INTERRUPT
 LXXXXHLL    C   L     LHLH     I3 INTERRUPT
 LXXXHLLL    C   L     LHLL     I4 INTERRUPT
 LXXHLLLL    C   L     LLHH     I5 INTERRUPT
 LXHLLLLL    C   L     LLHL     I6 INTERRUPT
 LHLLLLLL    C   L     LLLH     I7 INTERRUPT
 HLLLLLLL    C   L     LLLL     I8 INTERRUPT (LOWEST PRIORITY DEVICE)
 LLLLLLLL    C   L     HHHH     INTERRUPT FLAG
 XXXXXXXX    X   H     ZZZZ     TEST HI-Z
--------------------------------------------------------------------------
```

DESCRIPTION

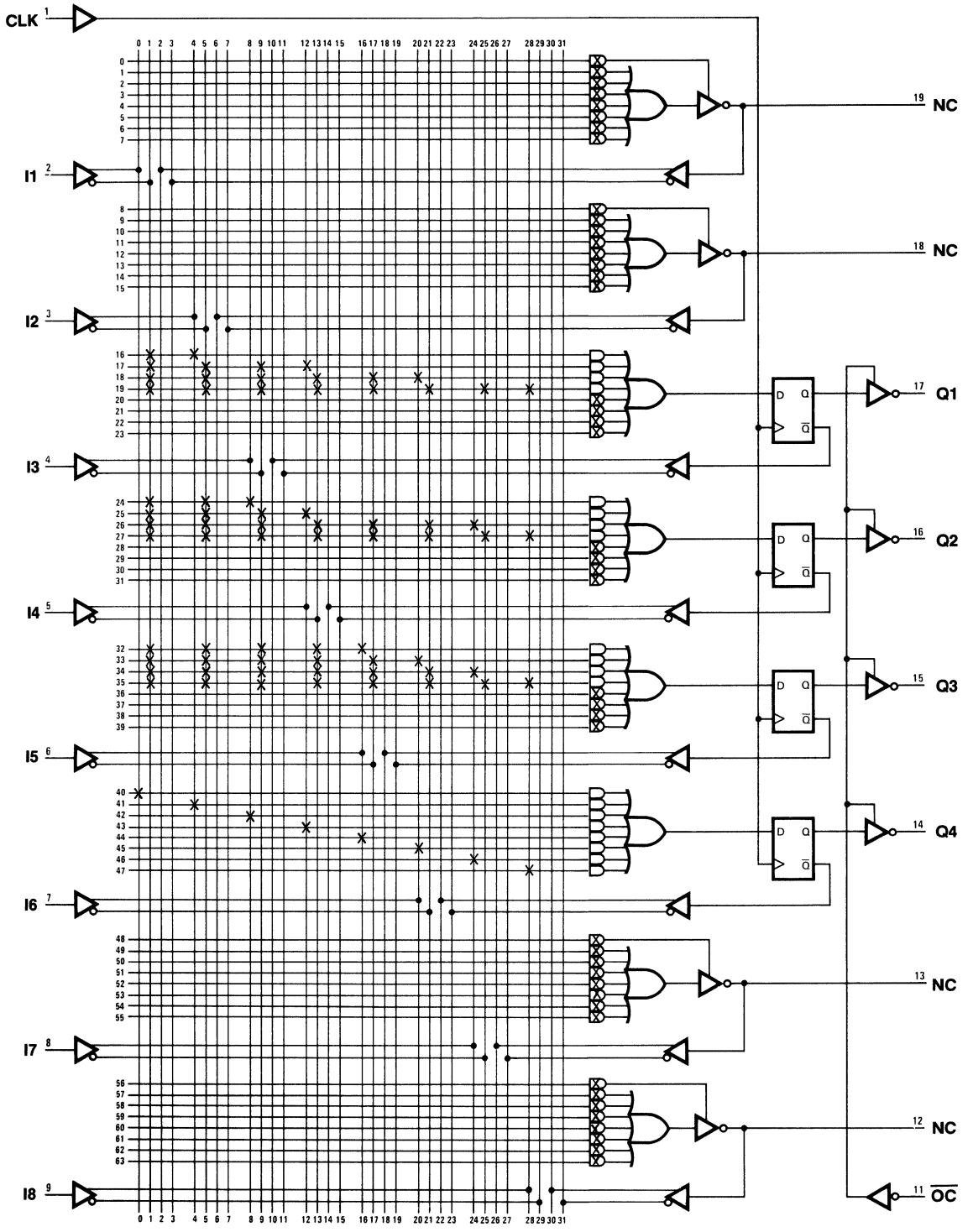
THE I/O PRIORITY INTERRUPT ENCODER PRIORITIZES 8 I/O LINES (I1 THRU I8) PRODUCING 111 (Q3, Q2, AND Q1 RESPECTIVELY) FOR THE HIGHEST PRIORITY I/O DEVICE (I1) AND 000 FOR AN INTERRUPT FROM THE LOWEST PRIORITY I/O DEVICE (I8).

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

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

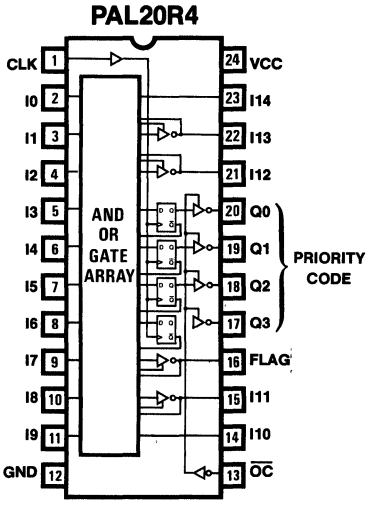
8-Bit I/O Priority Interrupt Encoder with Registers

Logic Diagram PAL16R4




IMPORTANT
NOTES

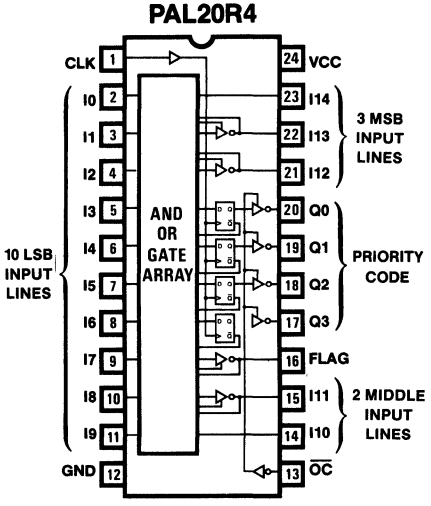
15-Input Registered Priority Encoder



```
PAL20R4                                                PAL DESIGN SPECIFICATION
P7090                                                    COLI/VOLPIGNO 10/13/82
15-INPUT REGISTERED PRIORITY ENCODER
MMI SUNNYVALE, CALIFORNIA
CLK I0  I1   I2  I3 I4 I5 I6 I7  I8  I9  GND
/OC I10 I11 FLAG Q3 Q2 Q1 Q0 I12 I13 I14 VCC


/Q0 := /I0* I1
     + /I0*/I1*/I2* I3
     + /I0*/I1*/I2*/I3*/I4* I5
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6* I7
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8* I9
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10* I11
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10*/I11*/I12* I13
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10*/I11*/I12*/I13*/I14

/Q1 := /I0*/I1* I2
     + /I0*/I1*/I2* I3
     + /I0*/I1*/I2*/I3*/I4*/I5* I6
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6* I7
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9* I10
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10* I11
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10*/I11*/I12*/I13* I14
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10*/I11*/I12*/I13*/I14

/Q2 := /I0*/I1*/I2*/I3* I4
     + /I0*/I1*/I2*/I3*/I4* I5
     + /I0*/I1*/I2*/I3*/I4*/I5* I6
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6* I7
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10*/I11* I12
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10*/I11*/I12* I13
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10*/I11*/I12*/I13* I14
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10*/I11*/I12*/I13*/I14

/Q3 := /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7* I8
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8* I9
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9* I10
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10* I11
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10*/I11* I12
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10*/I11*/I12* I13
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10*/I11*/I12*/I13* I14
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10*/I11*/I12*/I13*/I14

/FLAG = /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10*/I11*/I12*/I13*/I14
```

FUNCTION TABLE

```
CLK /OC I14 I13 I12 I11 I10 I9 I8 I7 I6 I5 I4 I3 I2 I1 I0 Q3 Q2 Q1 Q0 FLAG

; CHIP     FIFTEEN  INPUTS    --OUTPUTS--
;CONTROL   11111               QQQQ
;CLK /OC   432109876543210     3210    FLAG  COMMENTS
-------------------------------------------------------------------------------
  C   L    XXXXXXXXXXXXXXH     HHHH    H     I0  INTERRUPT (HIGHEST PRIORITY INPUT)
  C   L    XXXXXXXXXXXXXHL     HHHL    H     I1  INTERRUPT
  C   L    XXXXXXXXXXXXHLL     HHLH    H     I2  INTERRUPT
  C   L    XXXXXXXXXXXHLLL     HHLL    H     I3  INTERRUPT
  C   L    XXXXXXXXXXHLLLL     HLHH    H     I4  INTERRUPT
  C   L    XXXXXXXXXHLLLLL     HLHL    H     I5  INTERRUPT
  C   L    XXXXXXXXHLLLLLL     HLLH    H     I6  INTERRUPT
  C   L    XXXXXXXHLLLLLLL     HLLL    H     I7  INTERRUPT
  C   L    XXXXXXHLLLLLLLL     LHHH    H     I8  INTERRUPT
  C   L    XXXXXHLLLLLLLLL     LHHL    H     I9  INTERRUPT
  C   L    XXXXHLLLLLLLLLL     LHLH    H     I10 INTERRUPT
  C   L    XXXHLLLLLLLLLLL     LHLL    H     I11 INTERRUPT
  C   L    XXHLLLLLLLLLLLL     LLHH    H     I12 INTERRUPT
  C   L    XHLLLLLLLLLLLLL     LLHL    H     I13 INTERRUPT
  C   L    HLLLLLLLLLLLLLL     LLLH    H     I14 INTERRUPT (LOWEST PRIORITY INPUT)
  C   L    LLLLLLLLLLLLLLL     LLLL    L     NO  INTERRUPT
  X   H    XXXXXXXXXXXXXXX     ZZZZ    X     TEST HI-Z
-------------------------------------------------------------------------------
```

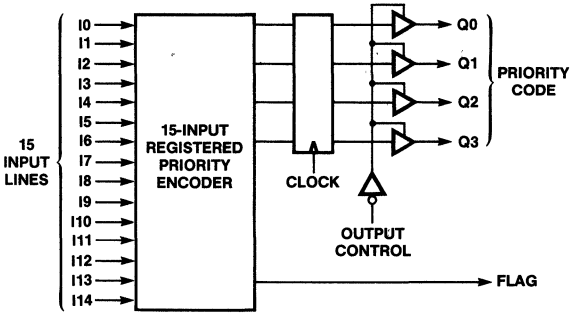


DESCRIPTION

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

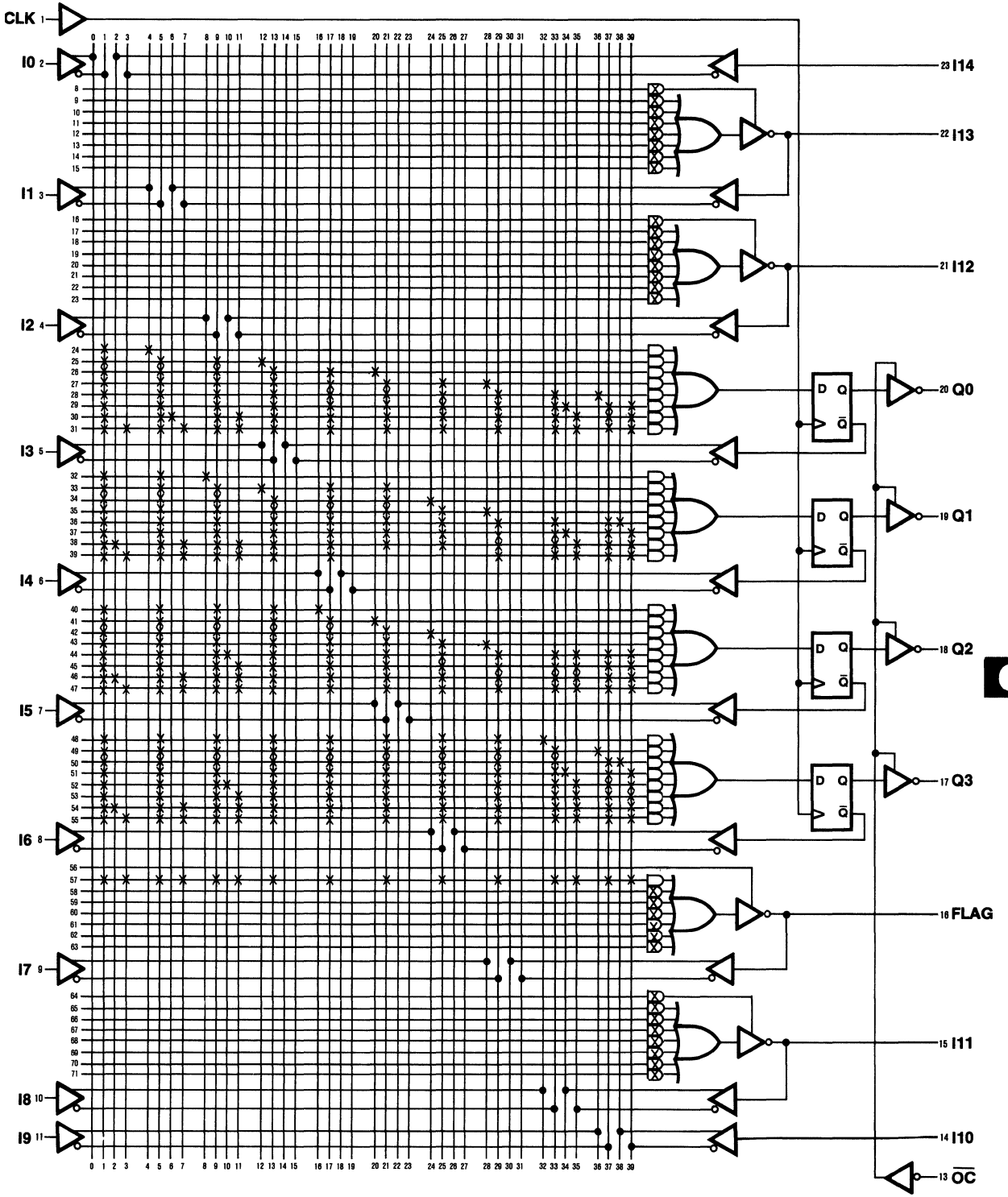
OPERATIONS TABLE

/OC	CLK	I14-I0	Q3-Q0	FLAG	OPERATION
H	X	X	Z	X	HI-Z
L	C	L	0	L	NO INTERRUPT
L	C	I0 =H	15	H	I0 INTERRUPT (HIGHEST PRIORITY INPUT)
L	C	I1 =H	14	H	I1 INTERRUPT
L	C	I2 =H	13	H	I2 INTERRUPT
L	C	I3 =H	12	H	I3 INTERRUPT
L	C	I4 =H	11	H	I4 INTERRUPT
L	C	I5 =H	10	H	I5 INTERRUPT
L	C	I6 =H	9	H	I6 INTERRUPT
L	C	I7 =H	8	H	I7 INTERRUPT
L	C	I8 =H	7	H	I8 INTERRUPT
L	C	I9 =H	6	H	I9 INTERRUPT
L	C	I10=H	5	H	I10 INTERRUPT
L	C	I11=H	4	H	I11 INTERRUPT
L	C	I12=H	3	H	I12 INTERRUPT
L	C	I13=H	2	H	I13 INTERRUPT
L	C	I14=H	1	H	I14 INTERRUPT (LOWEST PRIORITY INPUT)



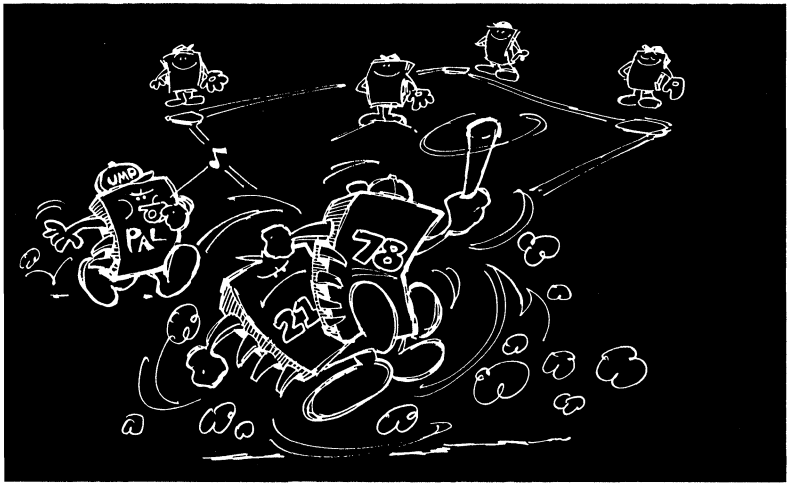
15-Input Registered Priority Encoder

Logic Diagram PAL20R4

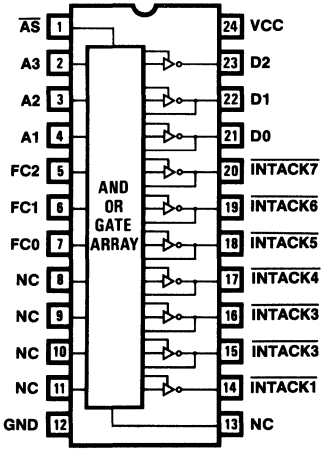



IMPORTANT
NOTES

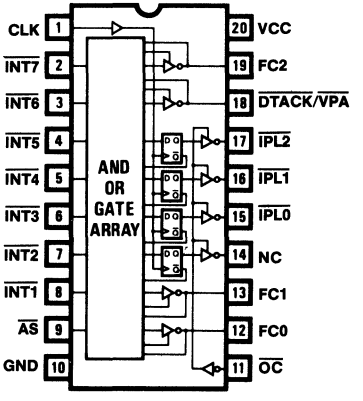
68000 Interrupt Controller



PAL20L10



PAL16R4



6

Introduction

Commercial and industrial microprocessor based systems typically consist of a processor interfaced with many peripherals which randomly require service. To fully utilize the processor's computing potentials sources of hardware interrupt must be used to free the processor from software routines. The 68000 16-bit microprocessor has 256 different exception processing vectors which point to a predetermined location in the program memory space. There are 192 external user interrupt vectors and seven autovector interrupts. Table 1 shows the exception vector assignments. The interrupt structure of the 68000 will be discussed in more detail along with two methods of designing an interrupt controller using PAL device (Programmable Array Logic).

TABLE 1
EXCEPTION VECTOR ASSIGNMENT

Vector Number(s)	Address			Assignment
	Dec	Hex	Space	
0	0	000	SP	Reset: Initial SSP
—	4	004	SP	Reset: Initial PC
2	8	008	SD	Bus Error
3	12	00C	SD	Address Error
4	16	010	SD	Illegal Instruction
5	20	014	SD	Zero Divide
6	24	018	SD	CHK Instruction
7	28	01C	SD	TRAPV Instruction
8	32	020	SD	Privilege Violation
9	36	024	SD	Trace
10	40	028	SD	Line 1010 Emulator
11	44	02C	SD	Line 1111 Emulator
12*	48	030	SD	(Unassigned, reserved)
13*	52	034	SD	(Unassigned, reserved)
14*	56	038	SD	(Unassigned, reserved)
15	60	03C	SD	Uninitialized Interrupt Vector
16-23*	64	04C	SD	(Unassigned, reserved)
	95	05F		—
24	96	060	SD	Spurious Interrupt
25	100	064	SD	Level 1 Interrupt Autovector
26	104	068	SD	Level 2 Interrupt Autovector
27	108	06C	SD	Level 3 Interrupt Autovector
28	112	070	SD	Level 4 Interrupt Autovector
29	116	074	SD	Level 5 Interrupt Autovector
30	120	078	SD	Level 6 Interrupt Autovector
31	124	07C	SD	Level 7 Interrupt Autovector
32-47	128	080	SD	TRAP Instruction Vectors
	191	0BF		—
48-63*	192	0C0	SD	(Unassigned, reserved)
	255	0FF		—
64-255	256	100	SD	User Interrupt Vectors
	1023	3FF		—

*Vector numbers 12, 13, 14, 16 through 23 and 48 through 63 are reserved for future enhancements by Motorola No user peripheral devices should be assigned these numbers.

68000 Exception Processing And Pin Description

The interrupt structure of the 68000 can grant up to 256 types of interrupt requests. These interrupts or exceptions are generated either by external or internal causes and are serviced by directing the flow of program to an exception processing routine. Table 1 lists these exceptions and their respective address in memory. Exception vectors are locations in memory where the processor fetches the address of a routine or subprogram which will service that exception. The exception vector is either internally or externally generated depending on the cause of the interrupt. The externally generated exceptions are the interrupts which are placed on the interrupt control pins ($\overline{IPL2}$, $\overline{IPL1}$, $\overline{IPL0}$), bus errors, and reset requests. The interrupts placed on control pins $\overline{IPL2}$-$\overline{IPL0}$ are requests from peripheral devices. The internally generated exceptions come from instructions, or from address error or tracing. These different types of exceptions and their priority is shown in Table 2. We will focus on group 1 interrupt exceptions.

TABLE 2
EXCEPTION GROUPING AND PRIORITY

Group	Exception	Processing
0	Reset Bus Error Address Error	Exception processing begins within two clock cycles.
1	Trace **Interrupt** Illegal Privilege	Exception processing begins before the next instruction
2	TRAP, TRAPV, CHK, Zero Divide	Exception processing is started by normal instruction execution.

The pinout of the 68000 is shown in Figure 1. The three interrupt control pins ($\overline{IPL2}$, $\overline{IPL1}$, $\overline{IPL0}$) are asynchronous active low inputs which indicate the encoded priority level of

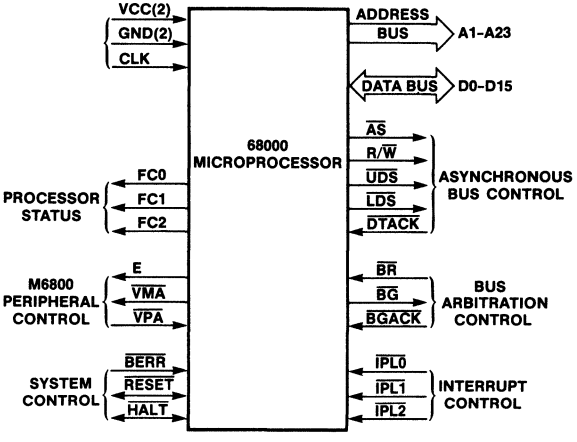


Figure 1

the device requesting an interrupt. The least significant bit is $\overline{IPL0}$ and the most significant bit is $\overline{IPL2}$. Level seven, ($\overline{IPL2}$, $\overline{IPL1}$, $\overline{IPL0}$) = (000), has the highest priority, while level zero, (111), indicates that no interrupts are requested. All lower or equal priority interrupts are masked during the interrupt service routine of the present interrupting device. There are two ways by which a peripheral device can interrupt the normal flow of program: autovectoring or external vector number generation. The function code pins (FC2, FC1, and FC0) all become high during an interrupt acknowledge cycle. The interrupt acknowledge cycle always follows an interrupt request only after the present instruction cycle is completed. Thus interrupt acknowlege cycles come in between the instruction cycles and no information is lost.

The signals which are used during an external interrupt request are listed below with a description of their behavior and function.

ADDRESS BUS: The 24-bit address bus holds the address of the data to be accessed During an interrupt acknowledge cycle the lower three bits (A3 A2 A1) hold the encoded level of the interrupt being serviced. If a level 5 interrupt is being serviced then (A3 A2 A1) will be (101) respectively.

DATA BUS: The lower 8-bits of this 16-bit data bus must contain the vector number during a user interrupt acknowledge cycle. If an auto-vector routine is in process then the data bus is ignored.

$\overline{AS}$: The processor asserts address strobe anytime there is a valid data on the address bus. It remains asserted for as long as address is valid.

R/$\overline{W}$: This signal defines the data bus transfer as a read or a write cycle. During an interrupt acknowledge cycle the processor is in a read mode.

$\overline{UDS}$, $\overline{LDS}$: The upper & lower data strobes are asserted when the processor is in read or write instruction cycle. Upper strobe enables the most significant byte of data while the lower strobe enables the least significant.

$\overline{DTACK}$: Data transfer acknowledge is an externally generated signal which tells the processor that valid data is present on the data bus If $\overline{DTACK}$ is not asserted before the falling edge of S4 (S4 is the fourth CPU clock state in a seven state instruction cycle) then wait states are introduced.

$\overline{IPL2}$-$\overline{IPL0}$: The three interrupt control pins are inputs which contain the encoded priority level of the external interrupting device Note that these are active low pins where (000) is level seven encoded.

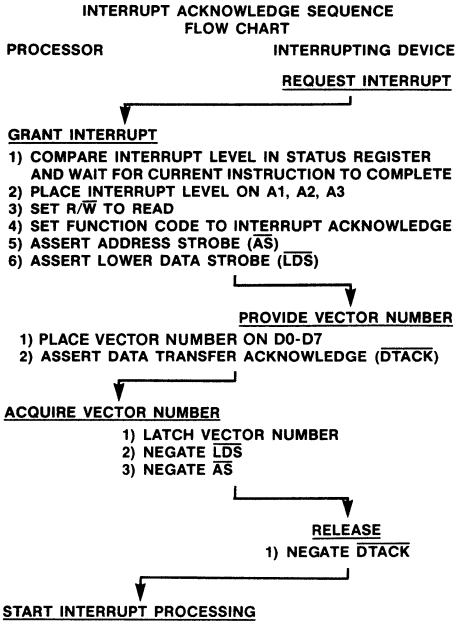
FC2-FC0: Function code input pins indicate the processors status. When they are all high the processor is in an interrupt acknowledge cycle.

E,$\overline{VMA}$,$\overline{VPA}$: The enable, valid memory address, and valid peripheral address are the standard 6800 family signals. $\overline{VPA}$ is also used when auto-vectoring.

The following two design examples use PAL units as a simple and cost effective interrupt controller interface to the 68000. The first method introduces external vector generation while the second design example will show how auto-vectoring can be done on the 68000.

Prioritized Individually Vectored Interrupt (Method 1)

As shown in the interrupt acknowledge sequence flow chart and timing diagram (Fig. 2), a peripheral device must interrupt the processor by placing the encoded level of priority on the interrupt control pins ($\overline{IPL2}$-$\overline{IPL0}$). The processor then grants the interrupt after the completion of the current instruction cycle. The encoded priority level of the interrupt is placed on address bus bits A3-A1 during the interrupt acknowledge cycle. The status code (FC2-FC0) become high which is an indication of an interrupt acknowledge cycle. Once the lower data strobe ($\overline{LDS}$) is asserted the external device must place an eight bit vector on the least significant byte of the data bus. This data is latched in S7.



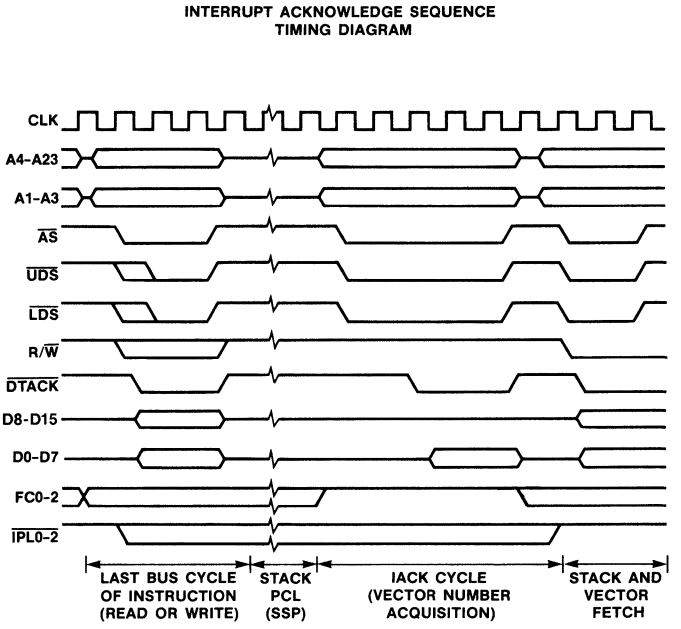


Figure 2

As shown on Table 1 vectors 64-255 are assigned to the user interrupt vector numbers. These vectors must be generated externally and placed on the data bus during an interrupt acknowledge cycle. In our interrupt controller we arbitrarily choose vectors 249-255 as the vectors assigned to the interrupting peripheral devices. Table 3a shows this assignment and how the vector can be decoded from the address bits A1-A3.

Priority level	Vector assigned	Data								Address		
Device #	Decimal #	D7	D6	D5	D4	D3	D2	D1	D0	A3	A2	A1
1 (low priority)	249	1	1	1	1	1	0	0	1	0	0	1
2	250	1	1	1	1	1	0	1	0	0	1	0
3	251	1	1	1	1	1	0	1	1	0	1	1
4	252	1	1	1	1	1	1	0	0	1	0	0
5	253	1	1	1	1	1	1	0	1	1	0	1
6	254	1	1	1	1	1	1	1	0	1	1	0
7 (high priority)	255	1	1	1	1	1	1	1	1	1	1	1

TABLE 3a

The two PAL units used on this design example generate the interrupt vectors and all the necessary control signals. The various signals and their implementation on the PAL units are explained below.

$\overline{\text{INT7}}$-$\overline{\text{INT0}}$: Any of the seven peripheral devices can request an interrupt by asserting one of these inputs. The interrupt must remain asserted until acknowledged by the CPU.

FC2-FC0 and $\overline{\text{AS}}$: Function code or processor status code become logical high during an interrupt acknowledge cycle. Address strobe is asserted anytime valid address is on the bus. $\overline{\text{DTACK}}$ and Data output control are decoded from these four outputs of the 68000.

A1-A3: The three least significant bits of the address bus contain the encoded level of the interrupting device. These signals are used in generating the vector number which is to be put on the data bus.

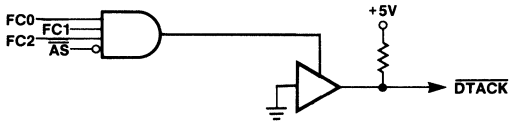
$\overline{\text{RESET}}$: Reset is an input which clears all outputs, and is used for system initialization.

$\overline{\text{IPL2}}$-$\overline{\text{IPL0}}$: These PAL outputs are active low synchronous signals which interface directly to the CPU. They represent the encoded level of the highest priority interrupt that is requested. Table 3b shows the truth table of our priority encoder implemented on a PAL.

Interrupt request input from peripheral							Encoded int. level		
$\overline{\text{INT7}}$	$\overline{\text{INT6}}$	$\overline{\text{INT5}}$	$\overline{\text{INT4}}$	$\overline{\text{INT3}}$	$\overline{\text{INT2}}$	$\overline{\text{INT1}}$	$\overline{\text{IPL2}}$	$\overline{\text{IPL1}}$	$\overline{\text{IPL0}}$
0	X	X	X	X	X	X	0	0	0
1	0	X	X	X	X	X	0	0	1
1	1	0	X	X	X	X	0	1	0
1	1	1	0	X	X	X	0	1	1
1	1	1	1	0	X	X	1	0	0
1	1	1	1	1	0	X	1	0	1
1	1	1	1	1	1	0	1	1	0
1	1	1	1	1	1	1	1	1	1

TABLE 3b

$\overline{\text{DTACK}}$: Data transfer acknowledge must be asserted by outside circuitry during a data transfer operation. The logic diagram shown below illustrates how /DTACK is derived from address strobe and processor status signals.



D7-D0: The three least significant bits of the data output can be decoded straight from the least significant address bits A3 A2 A1. That is D3=A3 D2=A2 D1=A1. The other five bits of data can be held high with pull up resistors. Output of the three data bits become enabled by using the same scheme which was used in enabling the $\overline{\text{DTACK}}$ output.

$\overline{\text{INTACK7}}$-$\overline{\text{INTACK1}}$: Only one of these signals will be asserted during the interrupt acknowledge cycle. This signal feeds back into the interrupting device to tell that device that its interrupt has been acknowledged. We can use the 3-bit addresses to decode these signals as shown in Table 3a.

			$\overline{\text{INTACK}}$						
A3	A2	A1	7	6	5	4	3	2	1
0	0	0	1	1	1	1	1	1	1
0	0	1	1	1	1	1	1	1	0
0	1	0	1	1	1	1	1	0	1
0	1	1	1	1	1	1	0	1	1
1	0	0	1	1	1	0	1	1	1
1	0	1	1	1	0	1	1	1	1
1	1	0	1	0	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1

TABLE 3c

The logic diagram of the controller is shown in Figure 3. The controller can operate without any wait states at the fastest CPU clock rate of 12 5 MHZ as shown in the timing diagram of Fig. 4. The appendix contains the fuse plot which translates into a logic schematic for these two PALs.

Auto-vectored Interrupts (Method 2)

The seven auto-vector interrupts are assigned vector numbers 25 through 31. Interrupts are requested by placing the encoded level of the request on the interrupt control pins ($\overline{\text{IPL2}}$-$\overline{\text{IPL0}}$). The processor responds to this request by placing the requested level on the processor's status register and by inhibiting all requests of lower priority. When the pending instruction cycle is completed an interrupt acknowledge cycle takes place. If Valid Peripheral Address, ($\overline{\text{VPA}}$) is asserted before the falling edge of S4 then an auto-vector routine takes place. The data or the vector number is generated internally depending on the priority level of the interrupt request; the vector assigned is shown on the table at the beginning of this report. The Auto-vector timing diagram and a very simple and practical interrupt controller implemented on a PAL is shown in Fig. 4. The PAL DESIGN SPECIFICATIONS are included in the appendix. Note that $\overline{\text{VPA}}$ is generated by enabling the PAL output when FC2-FC1 are high and $\overline{\text{AS}}$ is asserted. The appendix contains the fuse plots for this PAL design. Note that the fuse plots translates directly to a logic diagram.

Interrupt Controller Logic Diagram
External Vector Generation

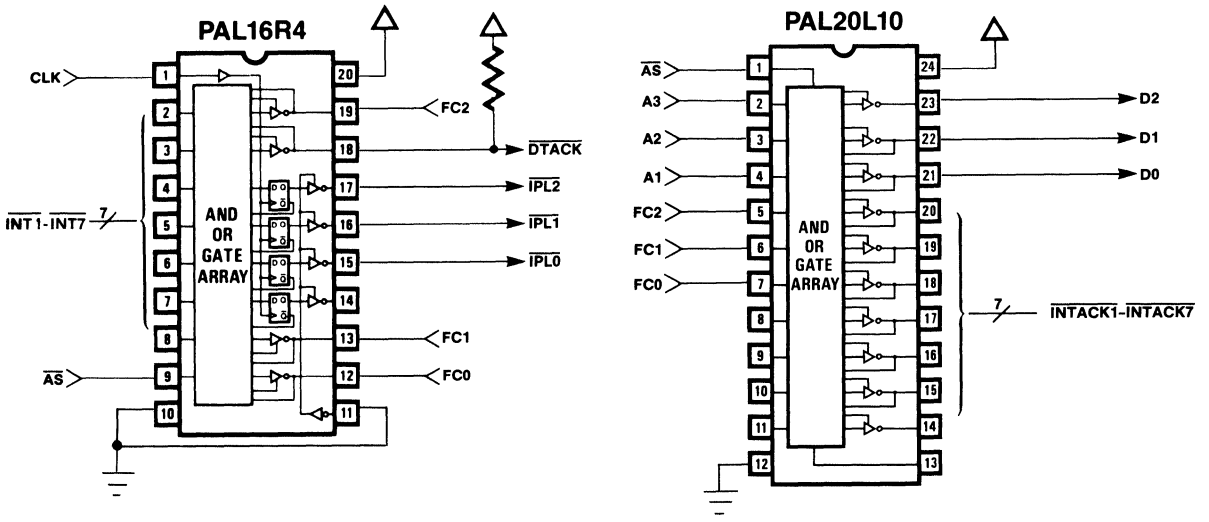
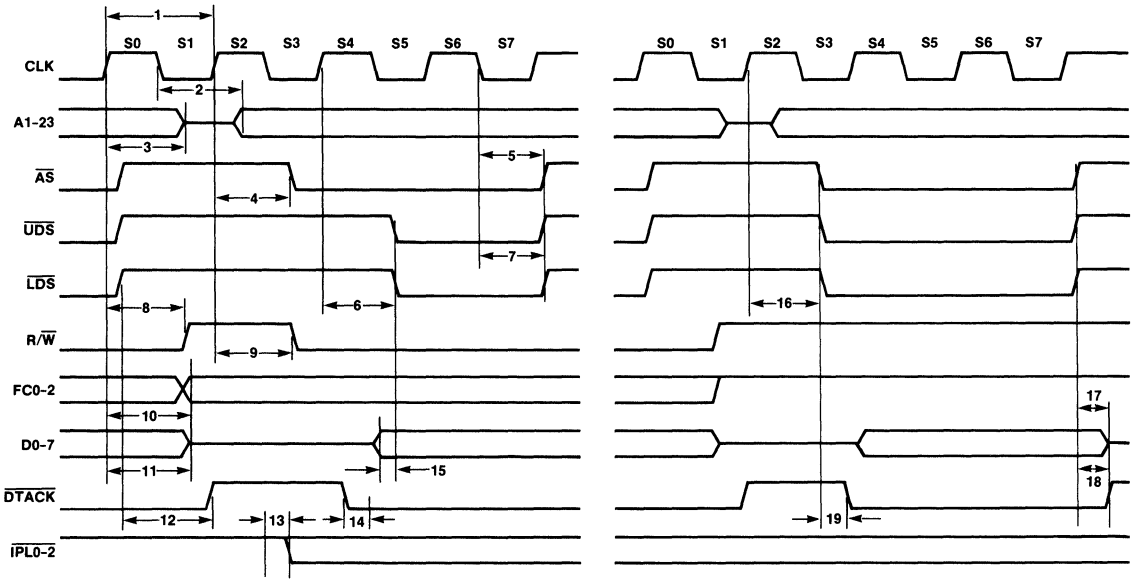


Figure 3



1 CLOCK PERIOD (80 ns)
2 CLOCK LOW TO ADDRESS (55 ns)
3 CLOCK HIGH TO ADDRESS Hi-Z (60 ns)
4 CLOCK HIGH TO $\overline{AS}$ LOW (55 ns)
5 CLOCK LOW TO $\overline{AS}$ HIGH (50 ns)
6 CLOCK HIGH TO $\overline{DS}$ LOW (55 ns)
7 CLOCK LOW TO $\overline{DS}$ HIGH (50 ns)
8 CLOCK HIGH TO R/$\overline{W}$ HIGH (60 ns)
9 CLOCK HIGH TO R/$\overline{W}$ LOW (60 ns)
10 CLOCK HIGH TO FC VALID (55 ns)
11 CLOCK HIGH TO DATA Hi-Z (60 ns)
12 $\overline{AS}$ HIGH TO $\overline{DTACK}$ HIGH (70 ns)
13 PAL's CLOCK TO OUT (25 ns)
14 MINIMUM SETUP TIME (20 ns)
15 DATA VALID TO $\overline{DS}$ LOW (15 ns)
16 CLOCK HIGH TO $\overline{AS}$, $\overline{DS}$ LOW (55 ns)
17 PAL's INPUT TO Hi-Z (25 ns)
18 PAL's INPUT TO Hi-Z (25 ns)
19 PAL's INPUT TO Hi-Z (25 ns)

Figure 4

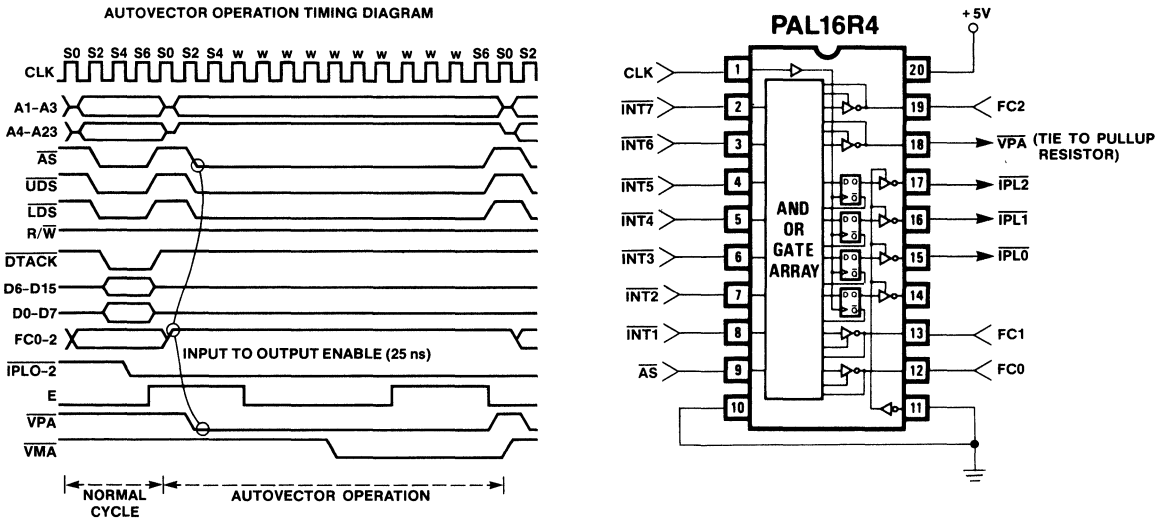


Figure 5

Conclusions

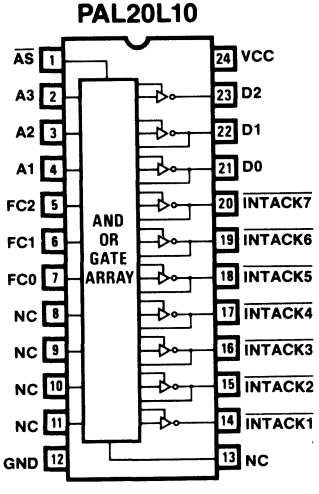
We have just seen how to implement an interrupt controller circuit using one or two PAL devices. This circuit can operate at the maximum operating frequency of the 68000 which is 12.5 MAZ. Since the most critical timing of the circuit is the PAL units's input to Hi-Z we see that we can operate the circuit with a wide spectrum of frequencies and with slower PAL units. To guarantee operation, timing spec. #16 and #19 on Figure 4 must add up to assert DTACK 20ns prior to falling edge of S4. Thus 55ns + (PAL unit's input to Hi-Z) should be less than or equal to 100ns for a 12.5 MHZ clock. We see that this qualifies fast, regular and half power PAL devices for this circuit. At slower CPU clock rates even the quarter-power PAL devices may be used.

References

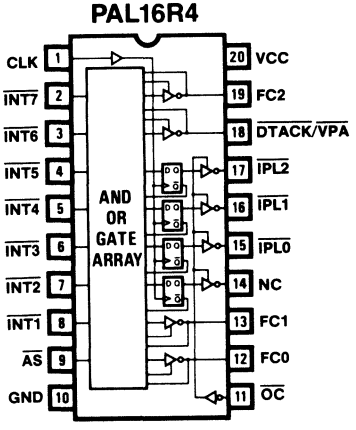
1. J. Birkner, V. Coli, "Programmable Array Logic Handbook," Monolithic Memories Inc.
2. MC68000 Data Sheet, Motorola Semiconductors. Austin, Texas 78721
3. Rex Davis, "Prioritized Individually Vectored Interrupts for Multiple Peripheral Systems with the MC68000," Motorola Application Note AN-819

Appendix

Interrupt Controller



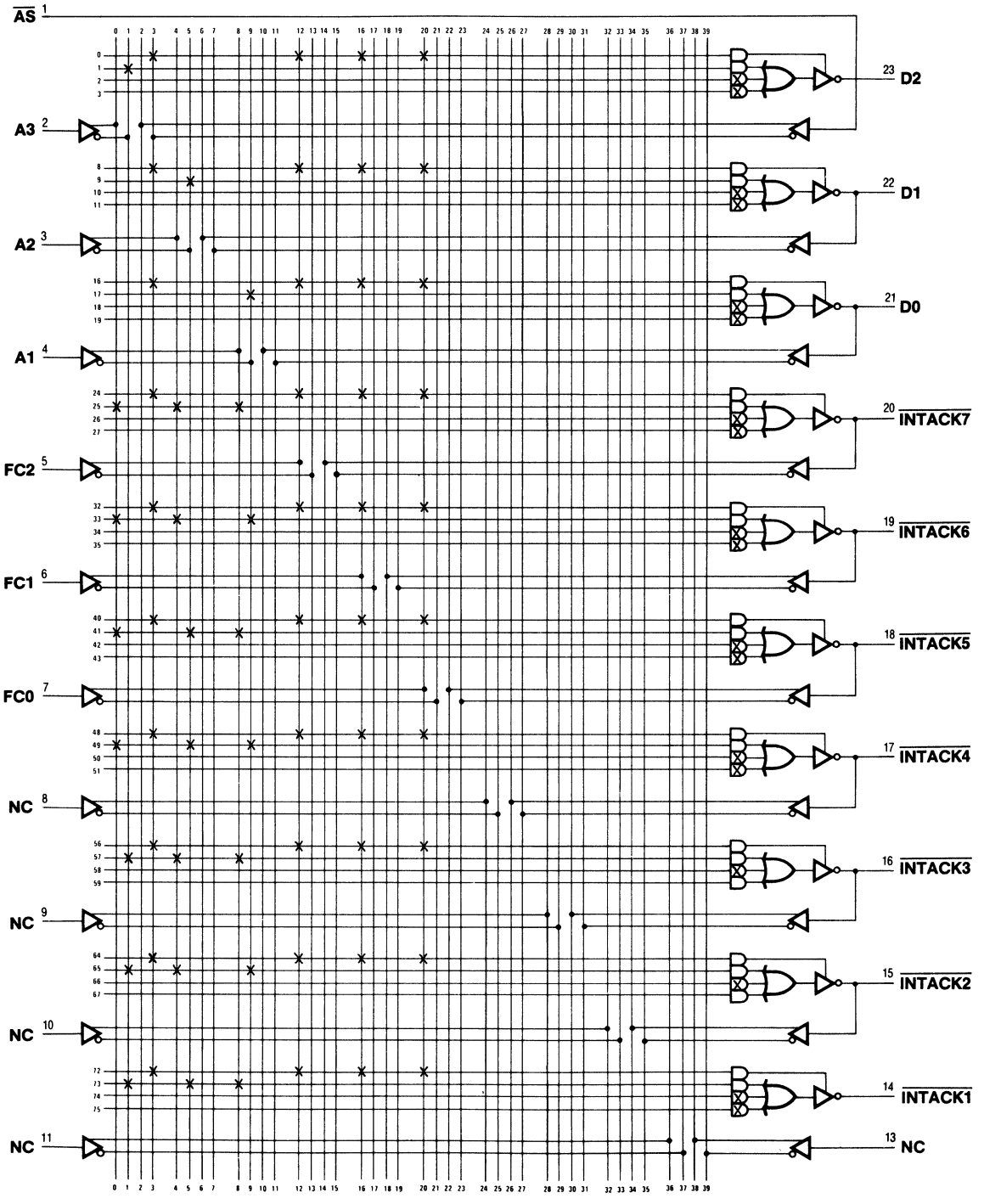
Interrupt Controller



```
PAL20L10                                                    PAL DESIGN SPECIFICATIONS
INTC.DAT                                                    DANESH TAVANA     9/25/82
INTERRUPT CONTROLLER
MMI, SUNNYVALE
/AS    A3       A2       A1       FC2      FC1      FC0      NC    NC NC NC GND
 NC /INTACK1 /INTACK2 /INTACK3 /INTACK4 /INTACK5 /INTACK6 /INTACK7 D0 D1 D2 VCC


IF (FC2* FC1* FC0* AS) /D2 =/A3

IF (FC2* FC1* FC0* AS) /D1 =/A2

IF (FC2* FC1* FC0* AS) /D0 =/A1

IF (FC2* FC1* FC0* AS) INTACK7 = A3* A2* A1

IF (FC2* FC1* FC0* AS) INTACK6 = A3* A2*/A1

IF (FC2* FC1* FC0* AS) INTACK5 = A3*/A2* A1

IF (FC2* FC1* FC0* AS) INTACK4 = A3*/A2*/A1

IF (FC2* FC1* FC0* AS) INTACK3 =/A3* A2* A1

IF (FC2* FC1* FC0* AS) INTACK2 =/A3* A2*/A1

IF (FC2* FC1* FC0* AS) INTACK1 =/A3*/A2* A1



FUNCTION TABLE
/AS FC2 FC1 FC0 A3 A2 A1 D2 D1 D0
/INTACK7 /INTACK6 /INTACK5 /INTACK4 /INTACK3 /INTACK2 /INTACK1
;                                   /  /  /  /  /  /  /
;                                   I  I  I  I  I  I  I
;                                   N  N  N  N  N  N  N
;                                   T  T  T  T  T  T  T
;                                   A  A  A  A  A  A  A
;/  F  F  F                         C  C  C  C  C  C  C
;A  C  C  C    A  A  A    D  D  D    K  K  K  K  K  K  K
;S  2  1  0    3  2  1    2  1  0    7  6  5  4  3  2  1       COMMENTS
-------------------------------------------------------------------------------
 L  H  H  H    L  L  L    L  L  L    H  H  H  H  H  H  H       NO INTERRUPT
 L  H  H  H    L  L  H    L  L  H    H  H  H  H  H  H  L       LEVEL 1
 L  H  H  H    L  H  L    L  H  L    H  H  H  H  H  L  H       LEVEL 2
 L  H  H  H    L  H  H    L  H  H    H  H  H  H  L  H  H       LEVEL 3
 L  H  H  H    H  L  L    H  L  L    H  H  H  L  H  H  H       LEVEL 4
 L  H  H  H    H  L  H    H  L  H    H  H  L  H  H  H  H       LEVEL 5
 L  H  H  H    H  H  L    H  H  L    H  L  H  H  H  H  H       LEVEL 6
 L  H  H  H    H  H  H    H  H  H    L  H  H  H  H  H  H       LEVEL 7
 H  X  X  X    X  X  X    Z  Z  Z    Z  Z  Z  Z  Z  Z  Z       OUTPUT HI-Z
 L  H  H  L    X  X  X    Z  Z  Z    Z  Z  Z  Z  Z  Z  Z       OUTPUT HI-Z
 L  H  L  H    X  X  X    Z  Z  Z    Z  Z  Z  Z  Z  Z  Z       OUTPUT HI-Z
 L  L  H  H    X  X  X    Z  Z  Z    Z  Z  Z  Z  Z  Z  Z       OUTPUT HI-Z
-------------------------------------------------------------------------------
```

PAL1 — Interrupt Controller

Logic Diagram PAL20L10



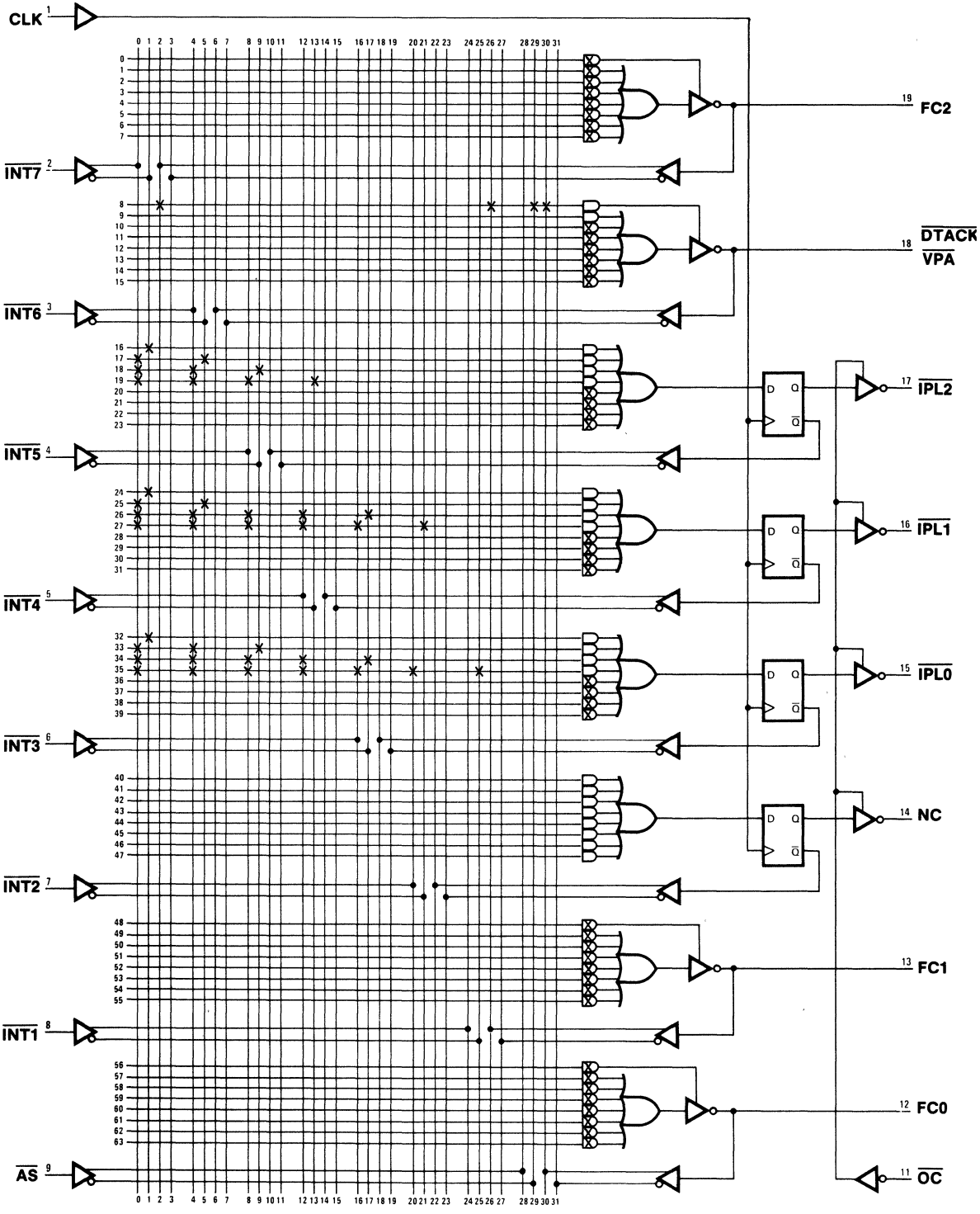
```
PAL16R4                                          PAL DESIGN SPECIFICATIONS
INTA.DAT                                         DANESH TAVANA
INTERRUPT CONTROLLER
MMI SUNNYVALE, CA
CLK /INT7 /INT6 /INT5 /INT4 /INT3 /INT2 /INT1  /AS GND
/OC  FC0   FC1    NC  /IPL0 /IPL1 /IPL2 /DTACK FC2 VCC

IF (FC2* FC1* FC0* AS) DTACK = VCC                    ;ASSERT IF OUTPUT ENABLE

IPL2 :=  INT7                                         ;PRIORITY ENCODED BIT
      + /INT7* INT6
      + /INT7*/INT6* INT5
      + /INT7*/INT6*/INT5* INT4

IPL1 :=  INT7                                         ;PRIORITY ENCODED BIT
      + /INT7* INT6
      + /INT7*/INT6*/INT5*/INT4* INT3
      + /INT7*/INT6*/INT5*/INT4*/INT3* INT2

IPL0 :=  INT7                                         ;PRIORITY ENCODED BIT
      + /INT7*/INT6* INT5
      + /INT7*/INT6*/INT5*/INT4* INT3
      + /INT7*/INT6*/INT5*/INT4*/INT3*/INT2* INT1
```

FUNCTION TABLE

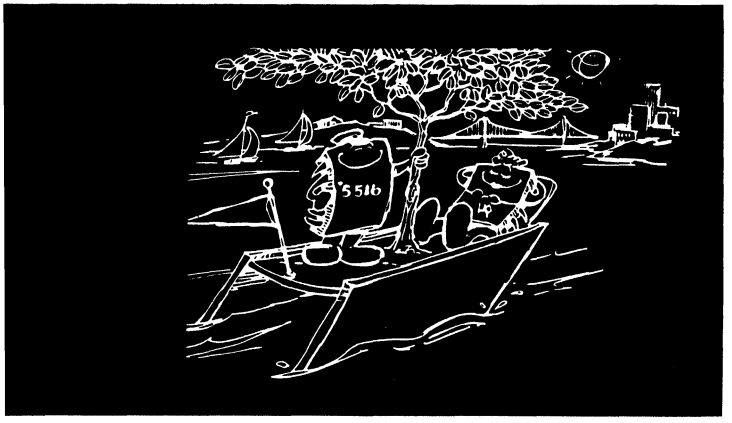
```
CLK /INT7 /INT6 /INT5 /INT4 /INT3 /INT2 /INT1
FC2 FC1 FC0 /AS /IPL2 /IPL1 /IPL0 /DTACK
;                                    /
;   / / / / / / /            / / /  D
;   I I I I I I I            I I I  T
;C  N N N N N N N   F F F /   P P P  A
;L  T T T T T T T   C C C A   L L L  C
;K  7 6 5 4 3 2 1   2 1 0 S   2 1 0  K          COMMENTS
-------------------------------------------------------------------------------
 C  L X X X X X X   H H H L   L L L  L          ASSERT /DTACK
 C  H L X X X X X   H H H H   L L H  Z          INTERRUPT LEVEL 7
 C  H H L X X X X   L L H L   L H L  Z          INTERRUPT LEVEL 6
 C  H H H L X X X   L H L L   L H H  Z          INTERRUPT LEVEL 5
 C  H H H H L X X   L H H L   H L L  Z          INTERRUPT LEVEL 4
 C  H H H H H L X   H L L L   H L H  Z          INTERRUPT LEVEL 3
 C  H H H H H H L   H L H L   H H L  Z          INTERRUPT LEVEL 2
 C  H H H H H H H   H H L L   H H H  Z          INTERRUPT LEVEL 1
-------------------------------------------------------------------------------
```

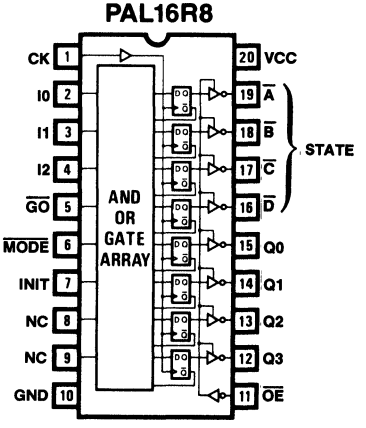
PAL2 — Interrupt Controller

Logic Diagram PAL16R4



State Counter for Multiplier/Divider*





6

* This paper is a slightly modified version of the entitled "Using A PAL to Emulate the Internal State Counter of the Monolithic Memories 'S516 LSI Multiplier/Divider" which appeared in the Conference Proceedings of the 8th West Coast Computer Faire, 18-20 March 1983.

Using A Pal to Emulate the Internal State Counter of the MMI 'S516 LSI Multiplier/Divider

Introduction

Specialized arithmetic hardware can be used with a standard microprocessor to significantly improve the number crunching power of a typical microcomputer. Such arithmetic hardware is available in the Monolithic Memories SN54/74S516 16-bit Multiplier/Divider/Accumulator. An introduction to this co-processor as well as a basic microprocessor interface circuit are presented in this paper. Also provided are programming examples and a state counter implemented in Programmable Array Logic which can be useful when debugging complex 'S516 software routines. The complete design of this state counter is included as a tutorial in state machine synthesis.

The Arithmetic Co-Processor

The Monolithic Memories SN54/74S516 is a 16-bit Multiplier/Divider/Accumulator with bus oriented Input/Output. The Logic Symbol for this device are given in Fig. 1. The 'S516 offers 28 different multiply options and 13 different divide options including:

- positive and negative multiply
- positive and negative accumulation
- multiplication by a constant
- single-length or double-length division
- division of a previously generated result
- division by a constant
- continued division of a remainder or quotient
- integer or fractional arithmetic mode

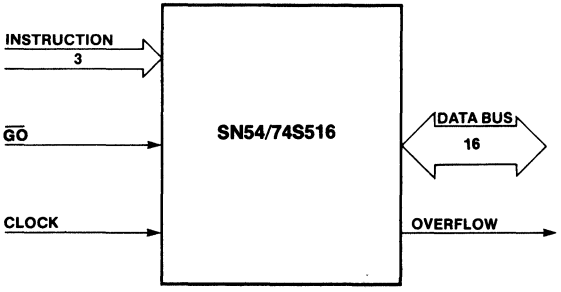


Figure 1. 'S516 Logic Symbol. The Multiplier can be thought of as an arithmetic processor in a black box in which there are three instruction inputs (I2-I0), the chip activation input ($\overline{GO}$), the clock pulse input (CK), the overflow flag (OVR), and the 16-bit I/O Data Bus (B15-B0).

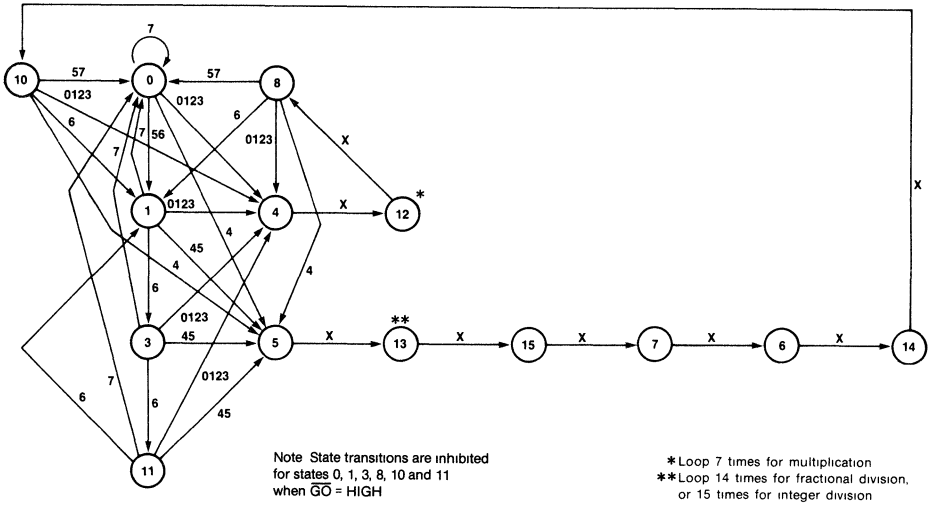


Figure 2. 'S516 State Transition Diagram. The multiplier transitions through these states as it performs its various arithmetic operations. The numbers inside the circles indicate the state of the machine (D,C,B,A). The numbers written next to the state-transition path lines are the possible instruction inputs (I2-I0).

The 'S516 sequences through 14 unique states synchronous with the clock input (CK) in order to execute these operations. The state transition diagram graph for the device is given in Fig. 2. A specific operation is selected by a 3-bit instruction input (I2-I0) sequence that must be given prior to the 'S516s trnasition through different states. This allows for easy interface with an 8-bit or 16-bit microprocessor.

Here's how to do it, assign the multiplier's three instruction lines to the least-significant bits on the address bus, and route these bits to the instruction input. The remaining bits of the address bus can be decoded using a Programmable Array Logic (PAL) device that controls the 'S516. The PAL decodes these bits in order to generate the chip activation input (GO) when the 'S516 is selected. The 'S516 bidirectional pins (B15-B0) are connected to this microprocessor data bus. A minimum microprocessor interface circuit is given in Fig. 3. Consult reference 1 or 2 for the details of a microprocessor interface circuit.

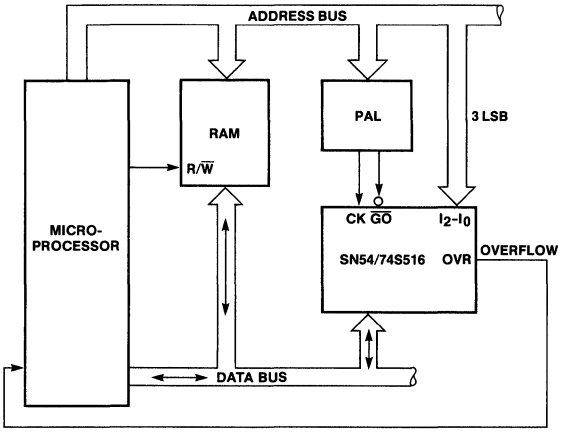


Figure 3. Minimum Microprocessor Interface. The 'S516 sits on the μP data bus and monitors the 3 LSBs from the address bus. The PAL monitors the remaining address bits.

You now can write an assembly program that not only allows the μP to select the multiplier, but also tells the μP what operation to perform. For example, if the 'S516 is assigned to octal address 100, any address code from 100 through 107 selects the multiplier. Let's say you want to multiply two numbers which are stored in address locations 108 and 109, and you want the double-length result placed in addresses 110 and 111. Finally, assume the μP MOVE instruction takes the general form:

MOV SA, DA

where SA is the source address and DA is the destination address. The following segment of assembly language code causes the microprocessor to move the data on and off the data bus, while the 74S516 performs the double length multiplication (Z,W = X*Y):

Assembly Program		SN54/74S516 Instruction
MOV	108,106	Load X (instruction 6)
MOV	109,100	Load Y (instruction 0)
NOP		Multiply X*Y (if needed; depends on μP speed)
MOV	107,110	Read Z, 16 MSB of product (instruction 7)
MOV	107,111	Read W, 16 LSB of product (instruction 7)

Software to perform division and more complex, chained operations can be written in a similar fashion. Programming examples are given in Apendix I. When evaluating the system and debugging the software, it is often desirable to observe what states the 'S516 transitions through as it receives its instructions (via the software). A state counter which can be used in such an evaluation circuit is now presented. This circuit is implemented with a single PAL chip. Basically, the PAL state counter monitors the control inputs to the 'S516 in order to display the state of the machine on four LEDs.

The PAL State Counter

The PAL used in the design is the PAL16R8. This device offers eight inputs and eight registered outputs with internal feedback to the PAL array. The true or complement of each input or feedback can be logically ANDed with each other. This is done by selectively blowing fuses in any of eight product lines are summed at an OR gate and are available for input to the D-input output register (dual rank flip-flop) on the rising edge of the chip common clock. This architecture allows direct inplementation of logic when expressed in sum of products form, and it is ideal for the construction of synchronous state machines. PALs provide an excellent alternative to discrete TTL chips. In this design, a single PAL16R8 replaces fifteen SSI/MSI TTL chips. Reference 3 is available as a complete guide on how to design with Programmable Array Logic.

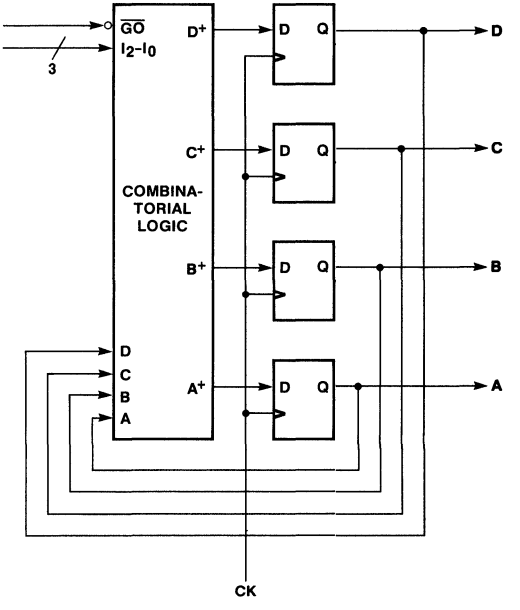


FIGURE 4. State Machine Representation of the 'S516. The next state of the machine (D+,C+,B+,A+) is a function of both the present state (D,C,B,A) and the instruction inputs (I2-I0 GO). The machine will transition synchronous with the clock (CK). Note that this is a Moore machine since the outputs are also the state variables.

In order to design the state counter, the 'S516 is considered to be a finite-state machine in which the next state is a function of both the present state and the instruction input. This is illustrated in Fig. 4. Referring to the state transition diagram (Fig. 2), the numbers inside the circles indicate the state of the machine. These state are represented with four bits (D,C,B,A), where D is the most-significant bit and A is the least significant bit. In the PAL hardware, the states are stored in D-type flip-flops which are gated with the sampled inputs on the rising edge of the clock. The new state is now stored in the flip-flop and is available for output. The numbers written next to the state transition path lines are the possible instruction inputs represented with three bits (I2-I0), where I2 is the most-significant bit and I0 is the least-significant bit. For example, when in state 0 and instruction 6 (110) is presented, followed by a clock pulse, the machine will be in state 5. Notice that states 2 and 9 do not appear in the state transition diagram. These states are considered undefined states and never can be entered under normal circuit operation. The machine will reside in each state for the duration of one clock cycle with two exceptions:

1) The device will remain in state 12, the multiplication loop, for seven clock pulses and state 13, the division loop, for either fourteen clock pulses if performing fractional arithmetic, or fifteen clock pulses if performing integer arithmetic.

2) The device will remain in states 0, 1, 3, 8, 10, or 11 indefinitely, regardless of clock pulses, if the chip activation input ($\overline{GO}$) is held high. This provides the system with the ability to request information to be placed on the data bus only when the data bus is free.

The state table, which is derived from the state transition diagram, is given in Fig. 5. This table is used to synthesize the state equations for the PAL. Note that each row in the state table corresponds to a state (or bubble from the state transition diagram). The left side of the table gives the present state, or where the machine now resides. The right side of the table gives the next state, or where the machine will go next. Since the next state is also dependent on eight possible instruction inputs, eight possible next states must be shown. The state equations are transcribed from the state transition diagram by considering which combinations of present states and inputs will provide an output (or next state) of one for the four state variables (D,C,B,A). For example, when in state 0 (S0), an input of 0, 1, 3, 2, or 4 will provide an output of one for state variable C. However, state variable D can never have an output of one regardless of any input when in state 0. The four state equations are given in Appendix II along with their corresponding Karnaugh maps. These Karnaugh maps and Boolean algebra are used to minimize the OR terms in the equations, so that they can fit in the eight product terms per output architecture provided in the PAL. Also, a 4-bit counter is implemented in the four remaining registers (Q3-Q0) in the PAL. This counter is used to control the state variables when in the multiplication or division loop. The counter is allowed to increment only when in states 12 or 13, and the state equations decode the terminal count indicating that the looping is complete.

PRESENT STATE		INPUT/NEXT STATE							
		INSTRUCTION INPUT							
	(000)	0 (001)	1 (011)	3 (010)	2 (110)	6 (111)	7 (101)	5 (100)	4
STATE	DCBA	DCBA	DCBA	DCBA	DCBA	DCBA	DCBA	DCBA	DCBA
S0†	0000	0100	0100	0100	0100	0001	0000	0001	0101
S1†	0001	0100	0100	0100	0100	0011	0000	0101	0101
S3†	0011	0100	0100	0100	0100	1011	0000	0101	0101
S4	0100	1100	1100	1100	1100	1100	1100	1100	1100
S5	0101	1101	1101	1101	1101	1101	1101	1101	1101
S6	0110	1110	1110	1110	1110	1110	1110	1110	1110
S7	0111	0110	0110	0110	0110	0110	0110	0110	0110
S8†	1000	0100	0100	0100	0100	0001	0000	0000	0101
S10†	1010	0100	0100	0100	0100	0000	0000	0000	0101
S11†	1011	0100	0100	0100	0100	0001	0000	0101	0101
S12	1100	1000	1000	1000	1000	1000	1000	1000	1000
S12*	1100	1100	1100	1100	1100	1100	1100	1100	1100
S13	1101	1111	1111	1111	1111	1111	1111	1111	1111
S13*	1101	1101	1101	1101	1101	1101	1101	1101	1101
S14	1110	1010	1010	1010	1010	1010	1010	1010	1010
S15	1111	0111	0111	0111	0111	0111	0111	0111	0111

* Multiple looping for state 12 or 13

† State transition is inhibited when $\overline{GO}$ = HIGH

Figure 5. State Table. This table is used to derive the state equations for the PAL. Note that each row in the state table corresponds to a state (or bubble from the state transition diagram, Fig. 2).

The state equations are then written using the PAL Design Specification format as shown in Fig. 6. In this form, this design information can be input to the FORTRAN computer program PALASM (acronym for PAL Assembler). PALASM assembles the PAL Design Specification, translating the logic equations into a fuse plot and various programming formats compatible with standard PROM and PAL programmers. PALASM also exercises the Function Table in the logic equation and reports design errors or provides test vectors.

It is easy to interface the PAL State Counter with the 'S516, as shown in Fig. 7. Just tie the clock input (CK), the three instruction inputs (12-10), and the chip activation input ($\overline{GO}$) of the PAL directly to the respective lines of the 'S516. The remaining PAL inputs, pins 6 through 9 should also be tied to ground as well as the output enable pin ($\overline{OE}$). Additionally, connect the four PAL state counter outputs (D,B,C,A) to the LEDs through series resistors (330-Ohm). The base of these LEDs are then tied to +5-Volts. The remaining PAL outputs, pins 12 through 15, should be left not connected. Now just plug in the power, i.e. VCC and GND, and you're ready to go!

There are a couple of more details worth mentioning about the PAL state counter:

First, as is the case for most sequential circuits, the PAL flip-flops can power up randomly to any state. Since this design is a true state machine, i.e. the present outputs are a function of the present inputs and the previous state, the designer should provide a way in order to force the flip-flops into a known state (preferably state 0) regardless of the initial state of the flip-flops. In this design, two ways are provided to do this. The PAL can be initialized to state 0 by driving the initialization pin high (INIT=H) followed by a clock pulse. However, in a system with the 'S516, the PAL can be initialized to state 0 along with the 'S516 by receiving 21 clock pulses with instruction code 7. In this case the PAL state counter is guaranteed to eventually end up in state 0 regardless of which state it powered up into and the initialization pin can be tied to ground.

Second, this PAL tracks the 'S516 exactly under all conditions except when chained integer division is to be performed. In this case, the MODE pin is connected to an external register which is set when the 'S516 passes through state 1 with instruction code 6 and cleared when the 'S516 passes through state 1 with instruction code 5.

```
PAL16R8                                                    PAL DESIGN SPECIFICATION
PATTERN NUMBER 1                                             VINCENT COLI 10/25/82
SN54/74S516 (16-BIT MULTIPLIER/DIVIDER/ACCUMULATOR CO-PROCESSOR) STATE COUNTER
MMI SUNNYVALE, CALIFORNIA
CK I0 I1 I2 /GO /MODE INIT NC NC GND /OE Q3 Q2 Q1 Q0 /D /C /B /A VCC


  A := /INIT* D* C    * A                    ;DECODE STATE 13 OR 15
     + /INIT    * C*/B* A                    ;DECODE STATE 5 OR 13
     + /INIT*/D*/C       * I2*/I1* I0* GO    ;DECODE STATE 0, 1, 3, OR 11
     + /INIT    */C    * A* I2*/I1* I0* GO   ;AND INSTRUCTION 5 AND /GO=L
     + /INIT    */C    * A            */GO   ;HOLD A IF STATE 1, 3, 11 AND /GO=H
     + /INIT    */C       * I2    */I0* GO   ;DECODE STATE 0,1,3,8,10,11 INST 4,6

  B := /INIT* D* C*/B* A* Q3* Q2* Q1         ;DECODE STATE 13 (DIVIDE LOOP)
     + /INIT    * C* B                       ;DECODE STATE 6, 7, 14, OR 15
     + /INIT*/D*/C    * A* I2* I1*/I0* GO    ;DECODE STATE 1 OR 3, INSTR 6, /GO=L
     + /INIT    */C    * B            */GO   ;HOLD B IF STATE 3, 10, 11 AND /GO=H

  C := /INIT* D* C*/B*/A*/Q3                 ;DECODE STATE 12 (MULTIPLY LOOP)
     + /INIT*/D* C                           ;DECODE STATE 4, 5, 6, OR 7
     + /INIT    * C    * A                  ;DECODE STATE 5, 7, 13, OR 15
     + /INIT    */C    * A* I2*/I1     * GO  ;DECODE STATE 1,3,11, INST 4,5 /GO=L
     + /INIT    */C       */I2         * GO  ;DECODE STATE 0, 1, 3, 8, 10, OR 11
     + /INIT    */C             */I1*/I0* GO  ;AND INST 0, 1, 2, 3, OR 4 AND /GO=L

  D := /INIT    * C*/B                       ;DECODE STATE 4, 5, 12, OR 13
     + /INIT    * C    */A                   ;DECODE STATE 4, 6, 12, OR 14
     + /INIT*/D*/C* B* A* I2* I1*/I0* GO      ;DECODE STATE 3 AND INST 6 AND /GO=L
     + /INIT* D*/C                   */GO    ;HOLD D IF STATE 8, 10, 11 AND /GO=H

/Q0 := /INIT    */C    * A            */Q0   ;HOLD Q0 IF STATE 1, 3, OR 11
     + /INIT*/D* C*/B* A* MODE       */Q0    ;HOLD IF STATE 5 AND /MODE=L
     + /INIT* D* C*/B                * Q0    ;COUNT IF STATE 12 OR 13
     + /INIT*/D* C*/B*/A                     ;CLEAR Q0 IF STATE 4
     + /INIT* D*/C    */A                    ;CLEAR Q0 IF STATE 8 OR 10
     + /INIT    * C* B                       ;CLEAR Q0 IF STATE 6, 7, 14,OR 15
     + /INIT*/D*/C*/B*/A         */I0        ;SET Q0=H IF STATE 0
     + /INIT*/D*/C*/B*/A     */I2            ;AND INSTRUCTION 5 (OR 7)

/Q1 :=       D* C*/B              */Q1*/Q0   ;HOLD IF STATE 12 OR 13 AND Q0=L
     +       D* C*/B              * Q1* Q0   ;COUNT IF STATE 12 OR 13
     +      /D       * A                     ; CLEAR Q1 IF STATE
     +        /C                            ;0, 1, 3, 5, 6, 7, 8,
     +           B                          ; 10, 11, 14, OR 15
     +  INIT                                 ;INITIALIZE Q1

/Q2 :=                      /Q2    */Q0      ;HOLD IF Q0=L
     +                      /Q2*/Q1          ;HOLD IF Q1=L
     +       D* C*/B      * Q2* Q1* Q0       ;COUNT IF STATE 12 OR 13
     +      /D                               ; CLEAR Q2 IF STATE
     +        /C                            ;0, 1, 3, 4, 5, 6, 7,
     +           B                          ;8, 10, 11, 14, OR 15
     +  INIT                                 ;INITIALIZE Q2
```

Figure 6a. PAL Design Specification. These logic equations are used by PALASM to generate the fuse plot, which programs the PAL to perform the state counter function.

```
/Q3 :=                      /Q3        */Q0   ;HOLD IF Q0=L
     +                      /Q3    */Q1,       ;HOLD IF Q1=L
     +                      /Q3*/Q2            ;HOLD IF Q2=L
     +       D* C*/B    * Q3* Q2* Q1* Q0      ;COUNT IF STATE 12 OR 13
     +      /D                                 ; CLEAR Q3 IF STATE
     +        /C                               ;0, 1, 3, 4, 5, 6, 7,
     +           B                             ;8, 10, 11, 14, OR 15
     +  INIT                                   ;INITIALIZE Q3

FUNCTION TABLE

CK /OE INIT /MODE /GO I2 I1 I0 Q3 Q2 Q1 Q0 D C B A

;CHIP CONTROL          INSTRUCTIONS   QQQQ    STATE
;CK /OE INIT   /MODE   /GO I2 I1 I0   3210    D C B A     COMMENTS (INSTR-STATE)
-------------------------------------------------------------------------------
 C   L   H        X      X  X  X  X   LLLH    L L L L     INITIALIZE        (X - 0)
 C   L   L        L      L  H  L  H   LLLH    L L L H     LOAD X (FRACT)    (5 - 1)
 C   L   L        L      L  L  X  X   LLLH    L L H L     LOAD Y   (0,1,2,3 - 4)
 C   L   L        L      L  X  X  X   LLHL    H H L L     MULT LOOP 1 XY    (X -12)
 C   L   L        L      L  X  X  X   LLHH    H H L L     MULT LOOP 2 XY    (X -12)
 C   L   L        L      L  X  X  X   LHLL    H H L L     MULT LOOP 3 XY    (X -12)
 C   L   L        L      L  X  X  X   LHLH    H H L L     MULT LOOP 4 XY    (X -12)
 C   L   L        L      L  X  X  X   LHHL    H H L L     MULT LOOP 5 XY    (X -12)
 C   L   L        L      L  X  X  X   LHHH    H H L L     MULT LOOP 6 XY    (X -12)
 C   L   L        L      L  X  X  X   HLLL    H H L L     MULT LOOP 7 XY    (X -12)
 C   L   L        L      L  X  X  X   HLLH    H H L L     MULT(EXIT LOOP)   (X - 8)
 C   L   L        L      L  H  L  H   LLLL    L L L L     ROUND Z,W         (5 - 0)
 C   L   L        L      L  H  H  H   LLLH    L L L L     READ Z            (7 - 0)
 C   L   L        L      L  H  L  H   LLLH    L L L H     LOAD X (FRACT)    (5 - 1)
 C   L   L        L      L  H  H  L   LLLH    L L H H     LOAD Z            (6 - 3)
 C   L   L        L      L  H  L  L   LLLH    L L H H     LOAD W            (4 - 5)
 C   L   L        L      L  X  X  X   LLLH    H H L H     DIVIDE LOOP  1    (X -13)
 C   L   L        L      L  X  X  X   LLHL    H H L H     DIVIDE LOOP  2    (X -13)
 C   L   L        L      L  X  X  X   LLHH    H H L H     DIVIDE LOOP  3    (X -13)
 C   L   L        L      L  X  X  X   LHLL    H H L H     DIVIDE LOOP  4    (X -13)
 C   L   L        L      L  X  X  X   LHLH    H H L H     DIVIDE LOOP  5    (X -13)
 C   L   L        L      L  X  X  X   LHHL    H H L H     DIVIDE LOOP  6    (X -13)
 C   L   L        L      L  X  X  X   LHHH    H H L H     DIVIDE LOOP  7    (X -13)
 C   L   L        L      L  X  X  X   HLLL    H H L H     DIVIDE LOOP  8    (X -13)
 C   L   L        L      L  X  X  X   HLLH    H H L H     DIVIDE LOOP  9    (X -13)
 C   L   L        L      L  X  X  X   HLHL    H H L H     DIVIDE LOOP 10    (X -13)
 C   L   L        L      L  X  X  X   HLHH    H H L H     DIVIDE LOOP 11    (X -13)
 C   L   L        L      L  X  X  X   HHLL    H H L H     DIVIDE LOOP 12    (X -13)
 C   L   L        L      L  X  X  X   HHLH    H H L H     DIVIDE LOOP 13    (X -13)
 C   L   L        L      L  X  X  X   HHHL    H H L H     DIVIDE LOOP 14    (X -13)
 C   L   L        L      L  X  X  X   HHHH    H H H H     DIV(EXIT LOOP)    (X -15)
 C   L   L        L      L  X  X  X   LLLL    L H H H     DIVIDE  Z,W/X     (X - 7)
 C   L   L        L      L  X  X  X   LLLL    L H H L     DIVIDE  Z,W/X     (X - 6)
 C   L   L        L      L  X  X  X   LLLL    H H H L     DIVIDE  Z,W/X     (X -14)
 C   L   L        L      L  X  X  X   LLLL    H L H L     DIVIDE  Z,W/X     (X -10)
 C   L   L        L      L  H  H  H   LLLL    L L L L     READ Z (QUOT)     (7 - 0)
 C   L   L        L      L  H  H  H   LLLH    L L L L     READ W (REM)      (7 - 0)
```

Figure 6b. PAL Function Table. The Function Table is used for design verification. PALASM exercises this table in the logic equation and reports design errors or provides test vectors. Note that both a multiplication and a division operation are tested.

If you do not require chained integer division, then simply connect the MODE pin to ground and the PAL state counter will automatically keep track of the required number of division loops for the integer and fractional modes. Consult the SN54/74S516 data sheet for more information on the integer and fractional arithmetic modes.

Summary

In this paper we have covered many topics. I only planned to provide the reader with an introduction to these topics. Hopefully interest has been stimulated in our products and new ideas. Additional information is available in the references or by contacting the author directly.

Acknowledgments

The first five programming examples were originally composed by Udi Gordon who is now designing military specification computers with Rolm Corporation. The Newton-Raphson programming example was composed by my colleague Suneel Rajpal and myself. The entire paper and especially the PAL state machine was critically reviewed by my office mate Danesh Tavana. The manuscript was prepared using the super microcomputer of Dick Blasco, an Electronics Design Consultant at R & L Associates based in San Jose, California. To these people and the rest of my PALs at Monolithic Memories, I owe gratitude.

References

1. "SN54/74S516 Co-Processor Supercharges 68000 Arithmetic," R. Blasco, V. Coli, C. Hastings, S. Rajpal, *MMI Application Note AN-114.*
2. "The Design and Application of a High- Speed Multiply/Divide Board for the STD Bus," M. Linse, G. Oliver, K. Bailey, M. Baxter, *MMI Application Note AN-115.*
3. *PAL Programmable Array Logic Handbook,* J. Birkner, V. Coli, Monolithic Memories, Inc.
4. "Big, Fast and Simple-Algorithms, Architecture, and Components for High-End Superminis," E. Gordon, C. Hastings, *MMI Applications Note An-111.*
5. *An Introduction to Arithmetic for Digital Designers,* S. Waser, M. J. Flynn, Holt, Rinehart & Winston, N.Y., 1982.

SN54/74S516 EVALUATION CIRCUIT

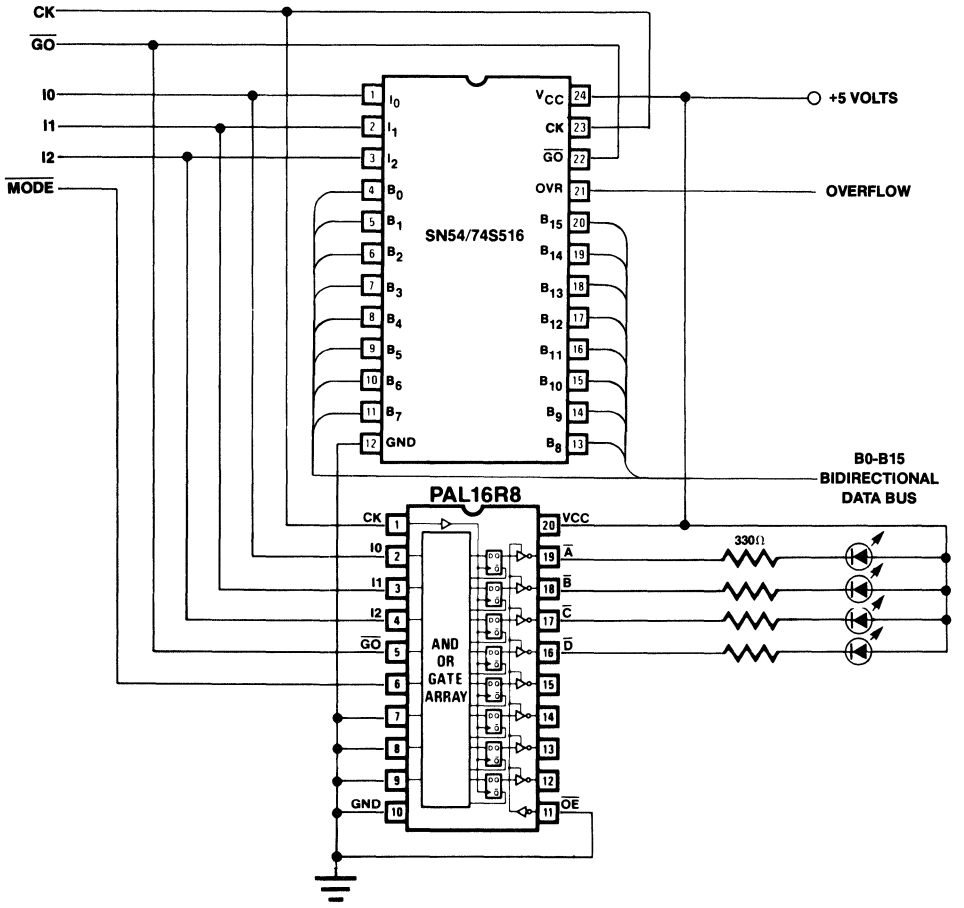


Figure 7. 'S516 Evaluation Circuit. The PAL16R8 monitors the control inputs to the 'S516 in order display the machine state in the LEDs. One PAL16R8 provides the logic and registers needed for the state counter. Only four resistors and four LEDs are needed to complete the circuit.

APPENDIX I

Programming Examples

In the following examples assume that each line with a separate instruction corresponds to one clock pulse except where otherwise indicated. Instruction codes are 0, 1, 2, 3, 4, 5, 6, 7 and X according to the usage explained in Fig. 2.

*means 8 clock cycles for Multiplication.
**means 19 clock cycles for Fractional Division, or
20 clock cycles for Integer Division

Programming Example 1

Calculating X • Y (A•B)

INST 6	X ← A
INST 0	Y ← B
INST X*	MULTIPLY, Perform X • Y (A•B)
INST 7	READ Z = 16 MSB OF (A•B)
INST 7	READ W = 16 LSB OF (A•B)

Programming Example 2

Calculating X1 • Y (A•C)
X1 is a previous multiplier value. It was previously loaded (in example 1) with A.

INST 0	Y ← C
INST X*	MULTIPLY, Perform X1 • Y (A•C)
INST 7	READ X = 16 MSB OF (A•C)
INST 7	READ W = 16 LSB OF (A•C)

Programming Example 3

Calculating $\sum_{i=1}^{N} X_i \cdot Y_i$ (A•B + C•D + E•F + ...)

In this case we read only after N multiplications. A new $X_i + 1$ is loaded during the multiplication process for X_iY_i.
Assume N = 3.
The sequence of instructions and operations for calculating

$$\sum_{i=1}^{3} X_i \cdot Y_i \text{ is: } (A \cdot B + C \cdot D + E \cdot F)$$

N = 1

INST 6	X ← A
INST 0	Y ← B
INST X*	MULTIPLY, Perform A•B

N = 2

INST 6	LOAD X ← C
	Z ← 16 MSB of (A•B)
	W ← 16 LSB of (A•B)
INST 2	Y ← D
INST X*	MULTIPLY, Perform C•D + (K_Z, K_W)

N = 3

INST 6	LOAD X ← E
	Z ← 16 MSB of (C•D + A•B)
	W ← 16 LSB of (C•D + A•B)
INST 2	Y ← F
INST X*	MULTIPLY, Perform E•F + (K_Z, K_W)
INST 7	READ Z = 16 MSB of (E•F + C•D + A•B)
INST 7	READ W = 16 LSB of (E•F + C•D + A•B)

Programming Example 4

Multiplication plus a constant (A•B + Constant)
Assume that the constant is a 32-bit 2s-complement number.

INST 6	X ← A
INST 6	Z ← C LOAD 16 MSB of constant
INST 6	W ← D LOAD 16 LSB of constant
INST 0	Y ← B
INST X*	MULTIPLY, Perform A•B + (Z, W)
INST 7	READ Z = 16 MSB of (A•B + (C, D))
INST 7	READ W = 16 LSB of (A•B + (C, D))

Programming Example 5

Dividing a 32-bit number by a 16-bit number ((B, C)/A)

INST 6	X ← A
INST 6	Z ← B
INST 4	W ← C
INST X**	DIVIDE, Perform (B, C)/A or ((Z, W)/X)
INST 7	READ the quotient Z = (B, C)/A
INST 7	READ the remainder W of (B, C)/A

Programming Example 6

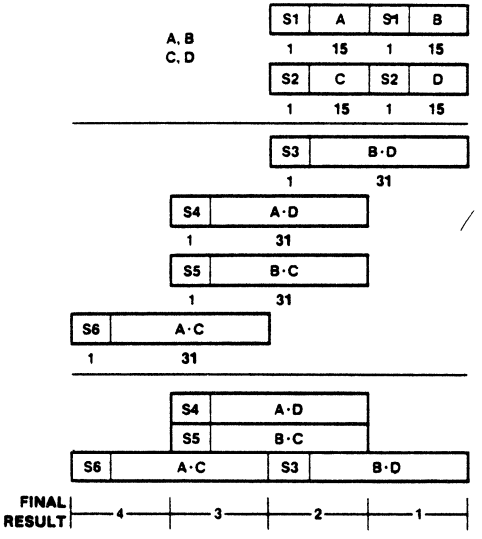
Double-precision multiplication ((A, B)•(C, D))
It is possible, using the 74S516, to multiply two numbers having up to 30 significant bits each.

Let S1 be the sign bit of the multiplier;
A be the 15 most-significant bits of the multiplier; and
B be the 15 least-significant bits of the multiplier.

S1 must be *duplicated* into the sign bit which goes with the 15 least significant bits, since the 15 most-significant bits and the 15 least-significant bits are used independently as two 2s-complement numbers.

Likewise let S2 be the sign bit of the multiplicand;
C be the 15 most-significant bits of the multiplicand;
and
D be the 15 least-significant bits of the multiplicand.

S2 must be duplicated into the sign bit of the least-significant half in this case also, just as S1 was. The final result consists of a sign bit, plus 62 significant numeric bits, plus one "empty" bit position. Such double-precision multiplication is common in floating-point operations. The following example will illustrate:



Programming Example 6 (Continued)

INST 6 X ← B
INST 0 Y ← D
INST X* MULTIPLY, Perform X·Y
INST 6 LOAD X ← D
Z = 16 MSB of B·D
W = 16 LSB of B·D
W = part 1 of final result and can be read now
INST 6 NOP
INST 2 Y ← A
INST X* MULTIPLY, Perform X·Y + $K_Z 2^{-15}$
INST 6 LOAD X ← C
Z = 16 MSB of (A·D + (B·D) · 2^{-15}
W = 16 LSB of (A·D + (B·D) · 2^{-15}
When B·D is shifted right 15 places, the sign bit S3 is extended.
INST 2 Y ← B
INST X* MULTIPLY, Perform X·Y + (K_Z, K_W)
INST 6 LOAD X ← A
Z = 16 MSB of (B·C + (A·D + (B·D) · 2^{-15})) = K_Z

W = 16 LSB of (B·C + (A·D + (B·D) · 2^{-15})) = K_W

W = part 2 of the final result and can be read now.

INST 6 NOP
INST 2 Y ← C
INST X* MULTIPLY, Perform X·Y + $K_Z 2^{-15}$
INST 7 READ part 4 of the final result.
Z = 16 MSB of
(A·C + (B·C + (A·D + (B·D) · 2^{-15}))·2^{-15})
INST 7 READ part 3 of the final result.
W = 16 LSB of
(A·C + (B·C + (A·D + (B·D) · 2^{-15}))·2^{-15})

Programming Example 7

Calculating a square root.

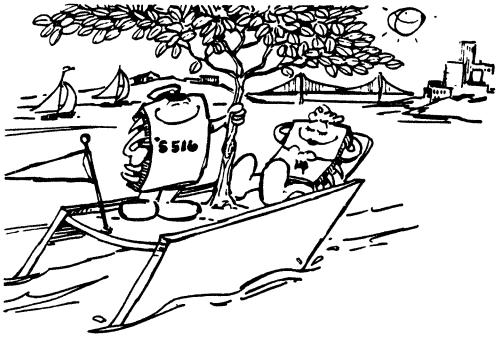
Using the 74S516, it is possible to perform a square root based on the Newton-Raphson approximation. The initial estimate of the square root, B, is supplied to the 'S516. This initial value can be obtained either from a ROM or by performing Boolean operations in the μP. Then the following equation is iteratively executed:

$$Qn+1 = \tfrac{1}{2}\left(Q_n + \frac{B}{Q_n}\right) = \tfrac{1}{2}Q_n + \frac{\frac{1}{2}B}{Q_n} \cong \sqrt{B}$$

Accurate results can be obtained in as few as three iterations since this function exhibits quadratic convergence. The exact square root can be obtained by comparing the present Q_n with the previous Q_n in the μP after every iteration. When $Q_{n+1} = Q_n$, then Q_{n+1} is the exact square root. Also, intermediate results can be input/output to a μP internal register in order to increase execution speed. Consult reference 4 or 5 for additional information on the Newton-Raphson approximation.

INST 5 X ← ½
INST 0 Y ← B = ½B
INST X* MULTIPLY, Perform Z,W ← X·Y
INST 7 NOP
INST 5 X ← Q_n = $\frac{\frac{1}{2}B}{Q_n}$
INST 4 NOP
INST X** DIVIDE, Perform Z,W ← K_Z/X
INST 5 ROUND Quotient
Z and W ← 0 = $\frac{1}{2}Q_n + \frac{\frac{1}{2}B}{Q_n}$
INST 2 Y ← ½
INST X MULTIPLY, Perform Z,W ← X1·Y + K_Z, K_W
REPEAT n times

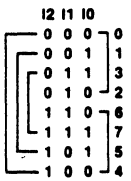
INST 7 READ Z
INST 7 READ W } Z,W = $Q_{n+1} \cong \sqrt{B}$



APPENDIX II

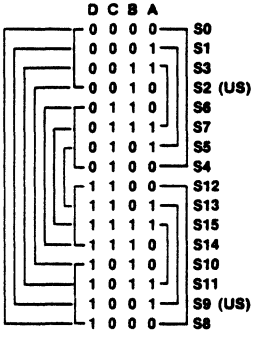
Derivation of Karnaugh Maps

INSTRUCTIONS



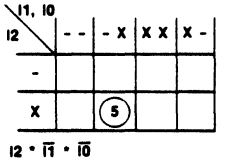
I2 \ I1, I0	- -	- X	XX	X -
-	0	1	3	2
X	4	5	7	6

STATE VARIABLES

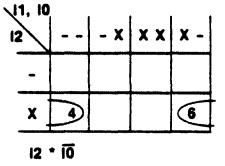


D,C \ B,A	- -	- X	X X	X -
- -	S0	S1	S3	US
- X	S4	S5	S7	S6
X X	S12	S13	S15	S14
X -	S8	US	S11	S10

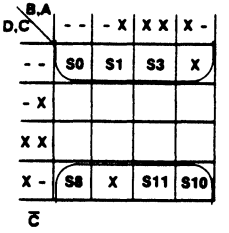
X = TRUE
- = COMPLEMENT
US = UNDEFINED STATE



$I2 \cdot \overline{I1} \cdot \overline{I0}$



$I2 \cdot \overline{I0}$

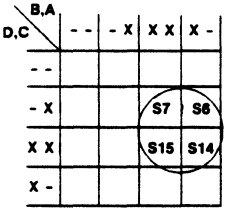


$\overline{C}$

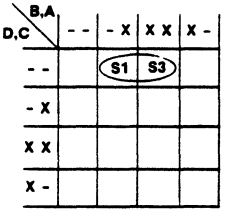
A: = D• C • A
+ C•/B• A
+ /D•/C •I2•/I1• I0 • GO
+ /C • A•I2•/I1• I0 • GO
+ /C •I2 •/I0 • GO

B: = S1 • 6 • GO + S3 • 6 • GO + S6 + S7
+ S13•CNTD + S14 + S15

B: = S13•CNTD
+ S6 + S7 + S14 + S15
+(S1+S3)•6• GO



C • B



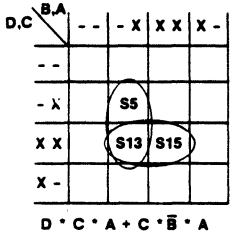
$\overline{D} \cdot \overline{C} \cdot A$

B: = D• C•/B•A •CNTD
+ C• B
+ /D•/C •A•I2•I1•/I0• GO

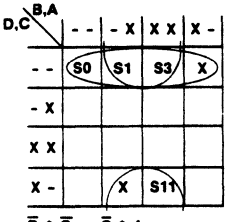
Derivation of State Equations

A: = S0•(6+5+4)•GO + S1•(6+5+4)• GO
+ S3•(6+5+4)• GO + S5 + S8•(6+4)•GO
+ S10•(6+4)•GO + S11•(6+5+4)• GO + S13
+ S15

A: = S5 + S13 + S15
+ (S0 + S1 + S3 + S11) • $\overline{5}$ • GO
+ (S0 + S1 + S3 + S8 + $\overline{S10}$ + S11) • (6 + 4) • GO



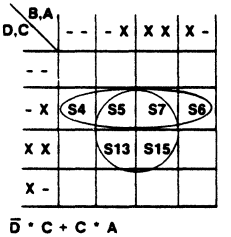
D • C • A + C • $\overline{B}$ • A



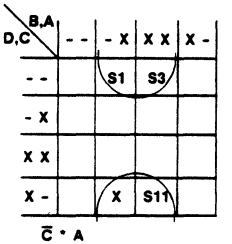
$\overline{D} \cdot \overline{C} + \overline{C} \cdot A$

C: = S0•(0+1+3+2+4)•GO + S1•(0+1+3+2+5+4)• GO
+ S3•(0+1+3+2+5+4)• GO + S4 + S5 + S6 + S7
+ S8•(0+1+3+2+4)•GO + S10•(0+1+3+2+4)•GO
+ S11•(0+1+3+2+5+4)• GO + S12•/CNTM + S13 + S15

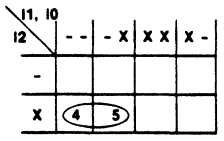
C. = S12•/CNTM
+ (S1 + S3 + S11) • (5 + 4) • GO
+ (S0 + S1 + S3 + S8 + S10 + S11) • (0+1+3+2+4)•GO



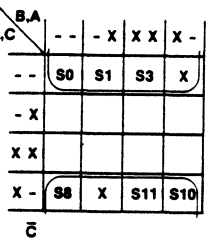
$\overline{D} \cdot C + C \cdot A$



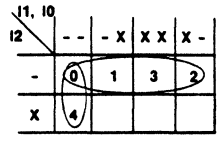
$\overline{C} \cdot A$



I2 * $\overline{I1}$



$\overline{C}$

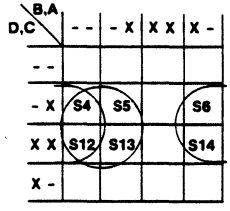


$\overline{I2}$ + $\overline{I1}$ * $\overline{I0}$

C: = D• C•/B•/A •/CNTM
+ /D• C
+ C • A
+ /C * A * 12*/I1*GO
+ /C •/I2 •GO
+ /C •/I1•/I0•GO

D: = S3•6•GO + S4 + S5 + S6 + S12 + S13 + S14

D: = S4 + S5 + S6 + S12 + S13 + S14 + S3•6 * GO



C * $\overline{B}$ + C * $\overline{A}$

D: = C•/B
+ C •/A
+ /D•/C• B• A•I2•I1•/I0 * GO

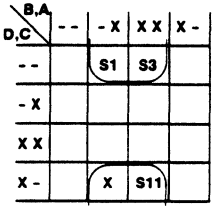
Note:
CNTM = While in multiplication loop.
CNTD = While in division loop.

Derivation of Hold States

D	C	B	A	
0	0	0	0	S0
0	0	0	1	S1
0	0	1	1	S3
1	0	0	0	S8
1	0	1	0	S10
1	0	1	1	S11

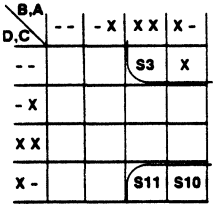
State Transitions are inhibited when $\overline{GO}$ = HIGH

A: = S1 + S3 + S11



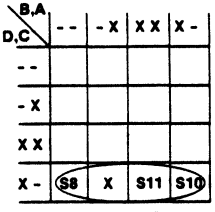
A: = $\overline{C}$ * A

B: = S3 + S10 + S11

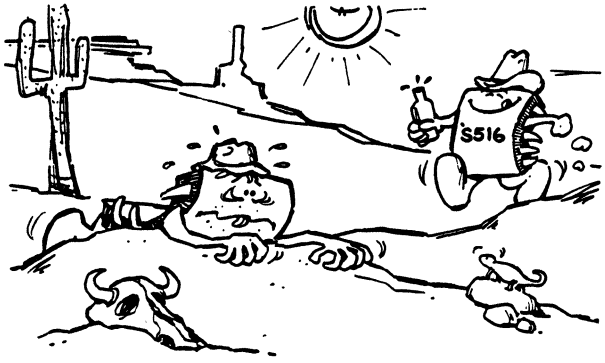


B: = $\overline{C}$ * B

D: = S8 + S10 + S11



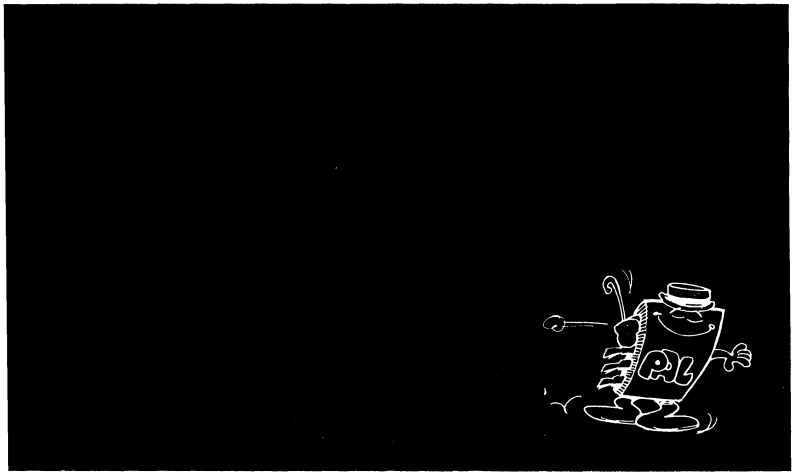
D: = D * $\overline{C}$

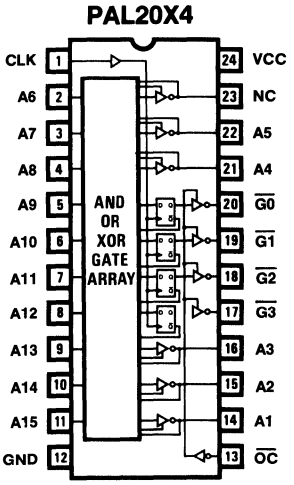


"... THE 745516 CAN DRAMATICALLY EXTEND THE LIFE CYCLE OF EXISTING MICROCOMPUTER SYSTEMS BASED ON MICROPROCESSORS WHICH EITHER DON'T HAVE MULTIPLICATION AND DIVISION INSTRUCTIONS, OR PERFORM THESE OPERATIONS RELATIVELY SLOWLY..."

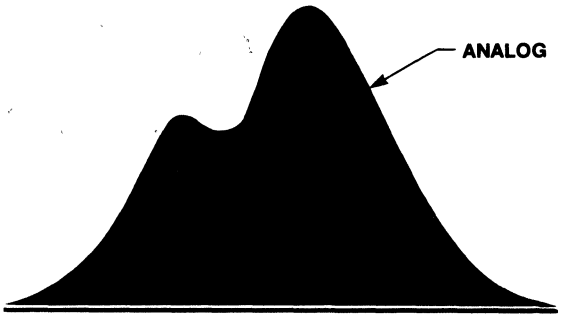
"... IN THIS DESIGN, A SINGLE PAL16R8 REPLACES FIFTEEN SSI/MSI TTL CHIPS..."

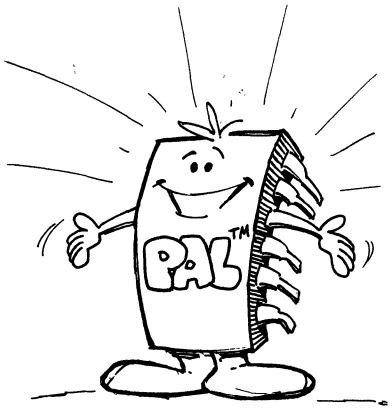
4-Bit Flash Gray A/D Converter

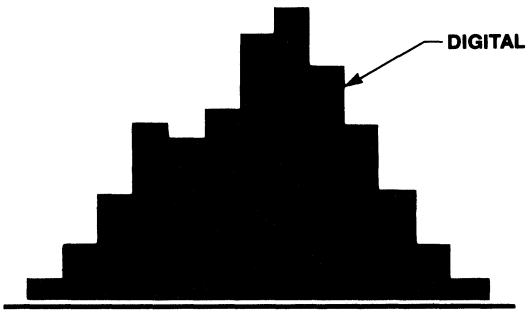


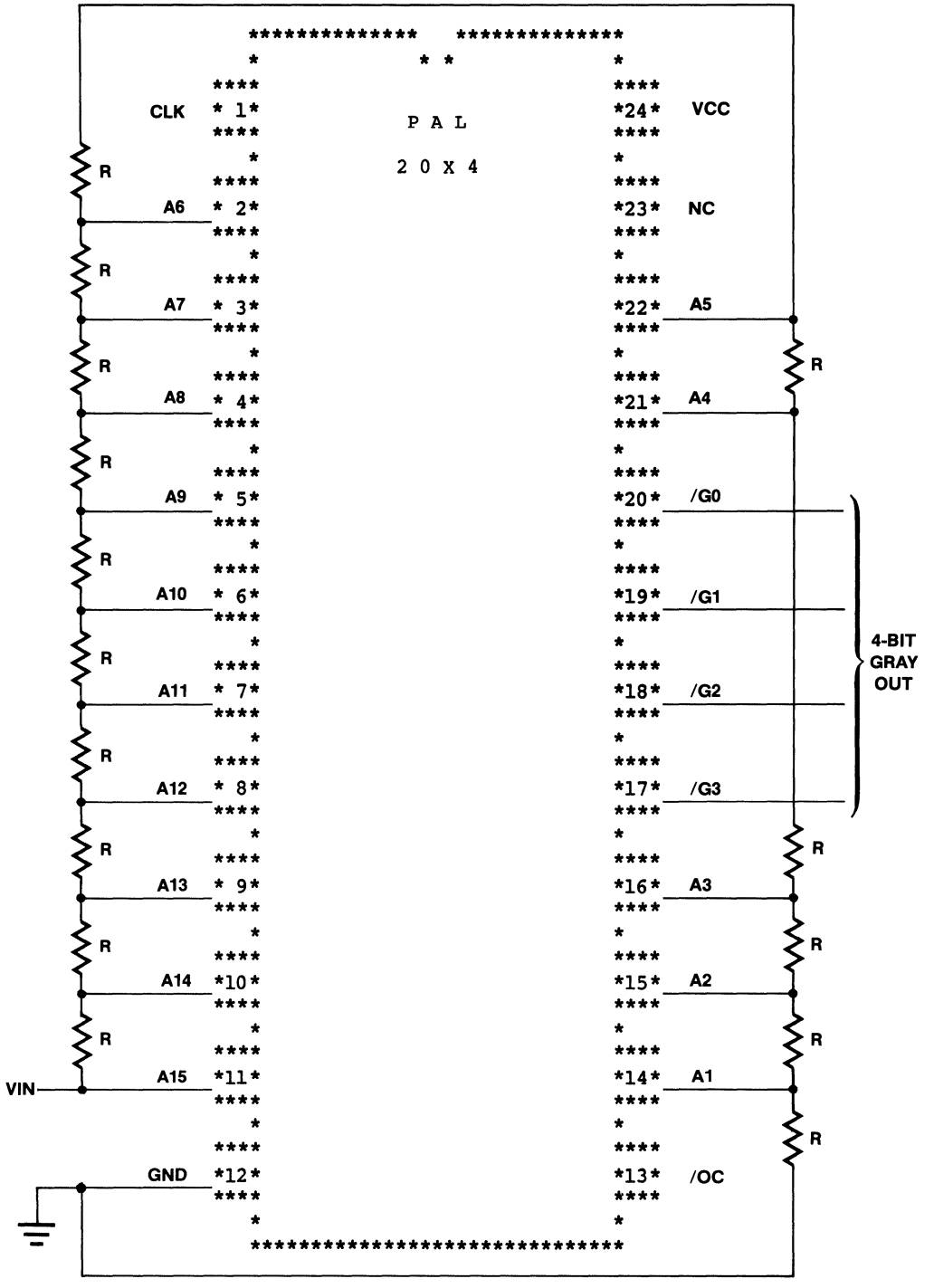


A/D Converter





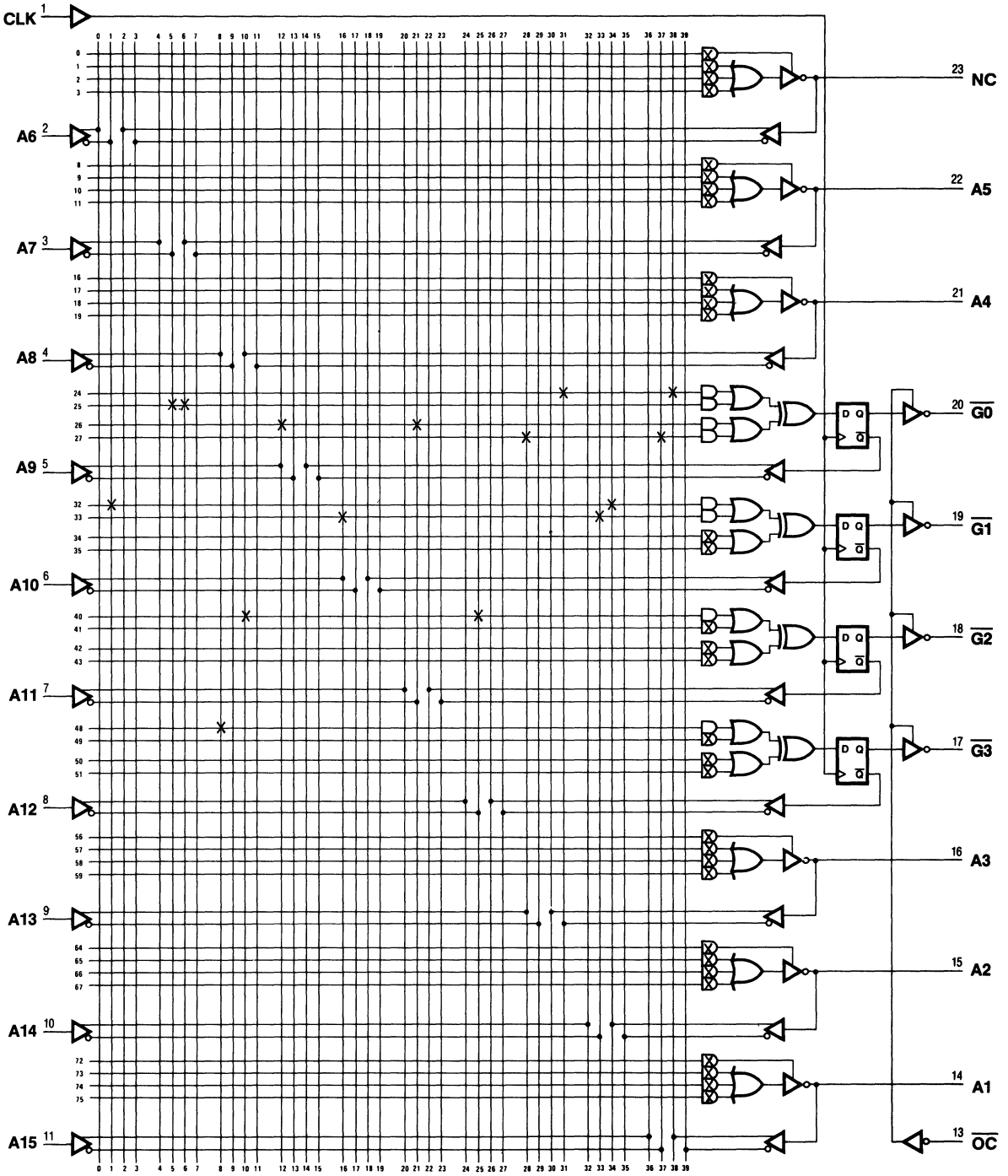


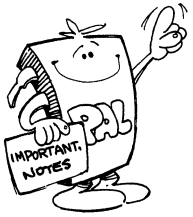

PAL
20X4
CLK
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
VIN
GND
VCC
NC
A5
A4
/G0
/G1
/G2
/G3
A3
A2
A1
/OC
4-BIT
GRAY
OUT
R

```
PAL20X4                                                PAL DESIGN SPECIFICATION
ADC4                                                     BIRKNER/COLI 07/29/81
4-BIT FLASH GRAY A/D CONVERTER
MMI SUNNYVALE, CALIFORNIA
CLK A6 A7 A8 A9 A10 A11 A12 A13 A14 A15 GND
/OC A1 A2 A3 /G3 /G2 /G1 /G0 A4 A5 NC VCC


G0 :=  /A3 * A1     ;CONVERT ANALOG TO G0 (LSB)
    +  /A7 * A5
   :+: /A11* A9
    +  /A15* A13

G1 :=  /A6 * A2     ;CONVERT ANALOG TO G1
    +  /A14* A10

G2 :=  /A12* A4     ;CONVERT ANALOG TO G2

G3 :=   A8          ;CONVERT ANALOG TO G3 (MSB)


FUNCTION TABLE

/OC CLK A15 A14 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 G3 G2 G1 G0

;          ANALOG  INPUTS      GRAY
;CONTROL   111111              OUTPUTS    COMMENTS
;/OC CLK   543210987654321     3210       FRACTION OF Vmax
-------------------------------------------------------------
  L   C    LLLLLLLLLLLLLLL     LLLL       V=0
  L   C    LLLLLLLLLLLLLLH     LLLH       V=1/16
  L   C    LLLLLLLLLLLLLHH     LLHH       V=1/8
  L   C    LLLLLLLLLLLLHHH     LLHL       V=3/16
  L   C    LLLLLLLLLLLHHHH     LHHL       V=1/4
  L   C    LLLLLLLLLLHHHHH     LHHH       V=5/16
  L   C    LLLLLLLLLHHHHHH     LHLH       V=3/8
  L   C    LLLLLLLLHHHHHHH     LHLL       V=7/16
  L   C    LLLLLLLHHHHHHHH     HHLL       V=1/2
  L   C    LLLLLLHHHHHHHHH     HHLH       V=9/16
  L   C    LLLLLHHHHHHHHHH     HHHH       V=5/8
  L   C    LLLLHHHHHHHHHHH     HHHL       V=11/16
  L   C    LLLHHHHHHHHHHHH     HLHL       V=3/4
  L   C    LLHHHHHHHHHHHHH     HLHH       V=13/16
  L   C    LHHHHHHHHHHHHHH     HLLH       V=7/8
  L   C    HHHHHHHHHHHHHHH     HLLL       V=15/16
  H   X    XXXXXXXXXXXXXXX     ZZZZ       TEST HI-Z
-------------------------------------------------------------


DESCRIPTION

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

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

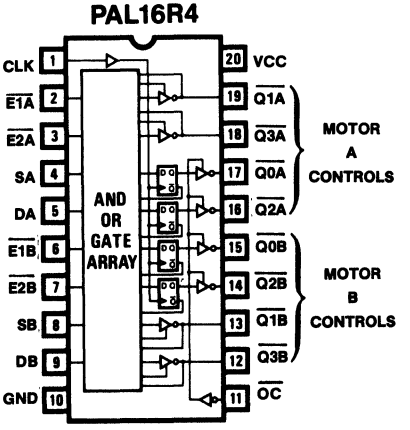
4-Bit Flash Gray A/D Converter

Logic Diagram PAL20X4




IMPORTANT
NOTES

Stepper Motor Controller



6

Functional Description

Stepper motors and linear actuators are used in a variety of applications requiring precise rotational and/or linear movement. Examples are printers, floppy disc drives, mechanical valves, etc. Stepper motors are two-phase permanent magnet motors which provide discrete angular movement every time the polarity of a winding is changed. In the case of linear actuators, the angular movement is converted to a linear movement via a load screw. In essence, they are dc motors without brushes, where the user provides commutation with external logic.

Circuit Operation

One type of drive circuit, unipolar drive, is shown in Figure 1 below. Two drive sequences are given in Tables 1A and 1B. Angular rotation is achieved by saturating the transistor drivers in the sequence shown in the appropriate table (full or half step). Now, assume the circuit of Figure 1 is connected to a stepper motor designed for 7.5° steps. By following the step sequence of Table 1A (full step), the shaft will rotate 7.5° each time the state is changed. If the sequence of Table 1B is followed, a 3.75° (half step) rotation will result for each change of state. For both step sequences, the direction can be reversed by stepping backwards through the table (step 4-3-2-1-4-etc.).

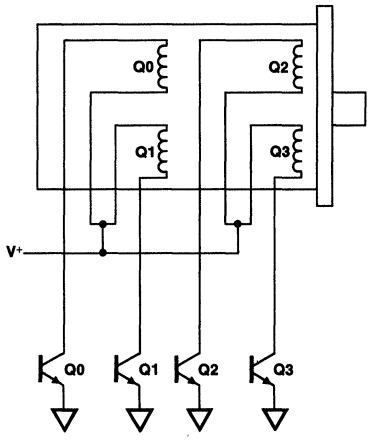


Figure 1

Table 1A
FULL STEP SEQUENCE

STEP	Q0	Q1	Q2	Q3
1	1	0	1	0
2	1	0	0	1
3	0	1	0	1
4	0	1	1	0
1	1	0	1	0

CLOCKWISE ROTATION ↓

↑ COUNTER-CLOCKWISE ROTATION

Table 1B
HALF STEP SEQUENCE

STEP	Q0	Q1	Q2	Q3
1	1	0	1	0
2	1	0	0	0
3	1	0	0	1
4	0	0	0	1
5	0	1	0	1
6	0	1	0	0
7	0	1	1	0
8	0	0	1	0
1	1	0	1	0

CLOCKWISE ROTATION ↓

↑ COUNTER-CLOCKWISE ROTATION

PAL Implementation

In this application, one PAL16R4 can be used to provide the logic levels required to drive two stepper motors in the full step mode. Due to the high current drive required (100-400 mA/phase), external inverting high current buffers would be used (ULN 2001 or equivalent). In the design, the following features are provided within the PAL:

- Enable/Disable inputs to enable stepping of either section. (/E inputs).
- Select clockwise or counter-clockwise rotation.
- Set the motor to logic state step 1.

A block diagram/pinout is shown in Figure 2.

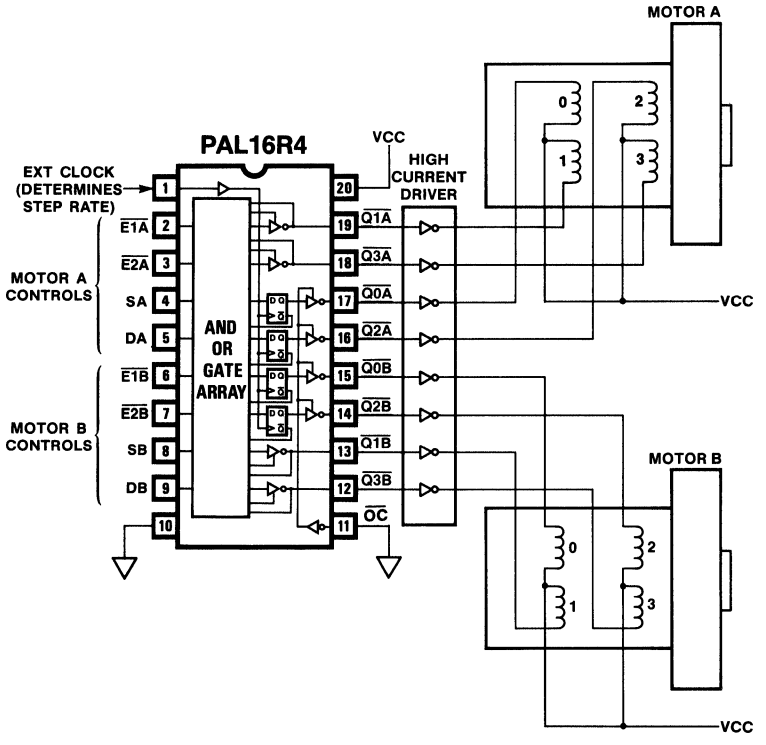


Figure 2

A function table for each motor control section is given below.

CLOCK	$\overline{E1}$	$\overline{E2}$	S	D	FUNCTION
X	1	X	X	X	Hold motor in current position
X	X	1	X	X	Hold motor in current position
↑	0	0	1	X	Set outputs to step 1 levels
↑	0	0	0	0	Step motor clockwise
↑	0	0	0	1	Step motor counter-clockwise

The full step sequence (Table 1A) can be simplified from 4 outputs to 2 outputs since Q1 = Q0 and Q3 = Q2. The sequences can then be expressed as follows:

	D = 0		D = 1	
STEP	Q0	Q2	Q0	Q2
1	1	1	1	1
2	1	0	0	1
3	0	0	0	0
4	0	1	1	0
1	1	1	1	1

$Q1 = \overline{Q0}$ $Q3 = \overline{Q2}$

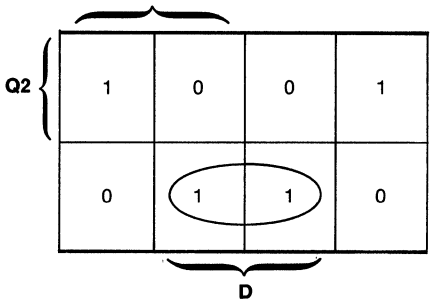
when S = 1:

Q0 = 1 Q1 = 0 Q2 = 1 Q3 = 0

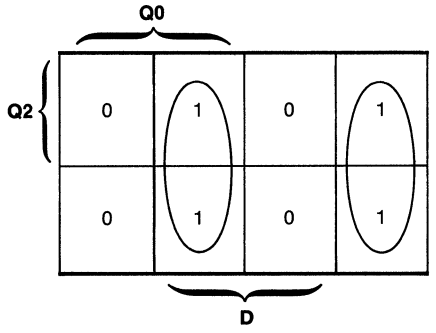
when E1 = 1 or E2 = 1

Q0 = q0 Q1 = q1 Q2 = q2 Q3 = q3

The step sequences can be converted to equations by use of a Karnaugh map.



$Q0 = Q2 \cdot \overline{D} + \overline{Q2} \cdot D$



$Q2 = Q0 \cdot D + \overline{Q0} \cdot \overline{D}$

Factor in E1 and E2:

$Q0 = \overline{E1} \cdot \overline{E2} \cdot Q2 \cdot \overline{D} + \overline{E1} \cdot \overline{E2} \cdot \overline{Q2} \cdot \overline{D}$

$Q2 = \overline{E1} \cdot \overline{E2} \cdot Q0 \cdot D + \overline{E1} \cdot \overline{E2} \cdot \overline{Q0} \cdot \overline{D}$

Express the set function as an equation:

$Q0 = \overline{E1} \cdot E2 \cdot S \qquad Q2 = \overline{E1} \cdot \overline{E2} \cdot S$

Express the hold function (when E1 or E2 = 1)

$Q0 = q0 \cdot E1 + q0 \cdot E2 \quad Q2 = q2 \cdot E1 + q2 \cdot E2$

Combining all the above:

$Q0 := \overline{E1} \cdot \overline{E2} \cdot S + Q0 \cdot E1 + Q0 \cdot E2 + \overline{E1} \cdot \overline{E2} \cdot Q2 \cdot \overline{D} + \overline{E1} \cdot \overline{E2} \cdot \overline{Q2} \cdot D$

$Q1 := Q0$

$Q2 := \overline{E1} \cdot \overline{E2} \cdot S + Q2 \cdot E1 + Q2 \cdot E2 + \overline{E1} \cdot \overline{E2} \cdot Q0 \cdot D + \overline{E1} \cdot \overline{E2} \cdot \overline{Q0} \cdot \overline{D}$

$Q3 := Q2$

Conclusion

Although this example could be used "as is" in a stepper motor application, the programmability of PAL device could allow for any desired modifications. Changes to the circuit might include:

1. Drive only one stepper motor, using a PAL16R6. The other flip-flops could be used as a programmable counter, allowing for different speed settings.
2. Drive only one stepper motor, using the extra inputs and outputs to handle other circuit functions.
3. Drive only one stepper motor, using a PAL16R6. The other flip-flops could be used as a 4-bit position counter.
4. The substitution of a PAL16R8, and another *inverting* buffer would allow the driving and control of four stepper motors.
5. Re-program for half-step operation.

```
PAL16R4                                          PAL DESIGN SPECIFICATION
SMC                                                 DAVE SACKETT 02/23/81
STEPPER MOTOR CONTROLLER
DEVOE COMPANY, INDIANAPOLIS, INDIANA
CLK /E1A /E2A SA DA /E1B /E2B SB DB GND
/OC /Q3B /Q1B /Q2B /Q0B /Q2A /Q0A /Q3A /Q1A VCC


Q0A :=  Q0A*/E1A                ;HOLD IF NOT E1
     +  Q0A*/E2A                ;HOLD IF NOT E2
     +  SA * E1A* E2A           ;STEP 1 IF SET
     + /Q2A* E1A* E2A* DA       ;LOAD /Q2A IF COUNTER-CLOCKWISE
     +  Q2A* E1A* E2A*/DA       ;LOAD  Q2A IF CLOCKWISE

IF (VCC) Q1A = /Q0A

Q2A :=  Q2A*/E1A                ;HOLD IF NOT E1
     +  Q2A*/E2A                ;HOLD IF NOT E2
     +  SA * E1A* E2A           ;STEP 1 IF SET
     +  Q0A* E1A* E2A* DA       ;LOAD  Q0A IF COUNTER-CLOCKWISE
     + /Q0A* E1A* E2A*/DA       ;LOAD /Q0A IF CLOCKWISE

IF (VCC) Q3A = /Q2A

Q0B :=  Q0B*/E1B                ;HOLD IF NOT E1
     +  Q0B*/E2B                ;HOLD IF NOT E2
     +  SB * E1B* E2B           ;STEP 1 IF SET
     + /Q2B* E1B* E2B* DB       ;LOAD /Q2B IF COUNTER-CLOCKWISE
     +  Q2B* E1B* E2B*/DB       ;LOAD  Q2B IF CLOCKWISE

IF (VCC) Q1B = /Q0B

Q2B :=  Q2B*/E1B                ;HOLD IF NOT E1
     +  Q2B*/E2B                ;HOLD IF NOT E2
     +  SB * E1B* E2B           ;STEP 1 IF SET
     +  Q0B* E1B* E2B* DB       ;LOAD  Q0B IF COUNTER-CLOCKWISE
     + /Q0B* E1B* E2B*/DB       ;LOAD /Q0B IF CLOCKWISE

IF (VCC) Q3B = /Q2B
```

FUNCTION TABLE

CLK /OC /E1A /E2A SA DA Q0A Q1A Q2A Q3A /E1B /E2B SB DB Q0B Q1B Q2B Q3B

;CHIP		STEPPER MOTOR A					STEPPER MOTOR B					
;C	/	CONTROL				STEP	CONTROL				STEP	
;L	O	E	E	S	D	QQQQ	E	E	S	D	QQQQ	
;K	C	1	2	A	A	0123	1	2	A	A	0123	COMMENTS
C	L	L	L	H	X	HLHL	L	L	H	X	HLHL	SET TO STEP 1
C	L	H	H	X	X	HLHL	H	H	X	X	HLHL	HOLD
C	L	L	L	L	L	HLLH	L	L	L	H	LHHL	STEP A CW, B CCW
C	L	L	L	L	L	LHLH	L	L	L	H	LHLH	STEP A CW, B CCW
C	L	L	L	L	L	LHHL	L	L	L	H	HLLH	STEP A CW, B CCW
C	L	L	L	L	L	HLHL	L	L	L	H	HLHL	STEP A CW, B CCW
C	L	L	L	L	L	HLLH	L	L	L	H	LHHL	STEP A CW, B CCW
C	L	L	L	L	L	LHLH	L	L	L	H	LHLH	STEP A CW, B CCW
C	L	L	L	L	H	HLLH	L	L	L	L	LHHL	STEP A CCW, B CW
C	L	H	L	L	H	HLLH	H	L	L	L	LHHL	HOLD
C	L	L	H	L	H	HLLH	L	H	L	L	LHHL	HOLD
C	L	L	H	H	H	HLLH	L	H	H	L	LHHL	HOLD

DESCRIPTION

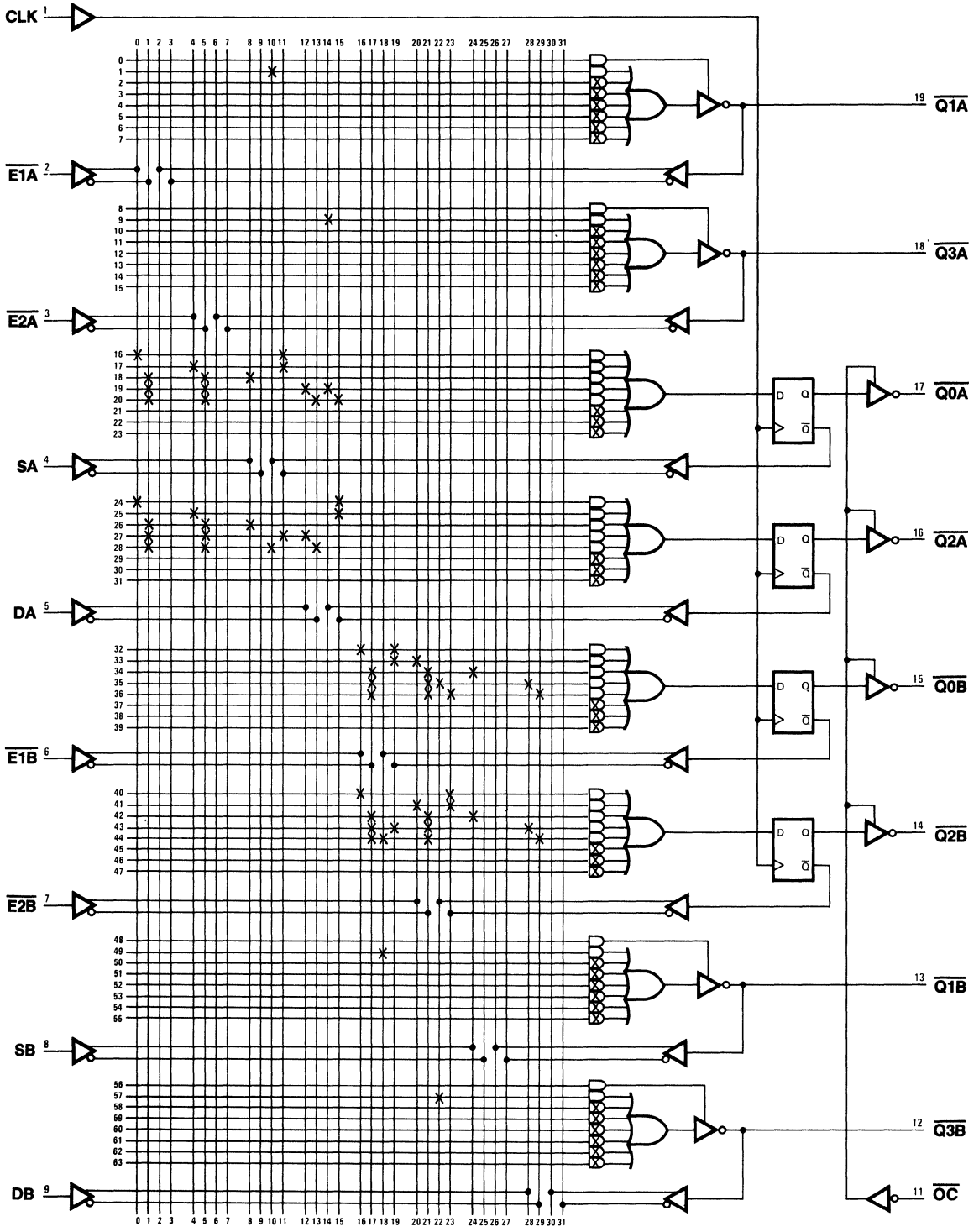
THIS PAL16R4 PROVIDES THE LOGIC LEVELS REQUIRED TO DRIVE TWO STEPPER MOTORS IN THE FULL STEP MODE.

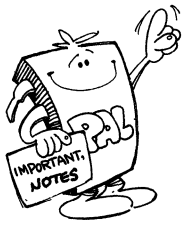
THE FOLLOWING OPERATIONS MAY BE PERFORMED FOR EACH STEPPER MOTOR CONTROLLER INDIVIDUALLY:

CLK	/E1	/E2	S	D	OPERATION
X	H	X	X	X	HOLD MOTOR IN CURRENT POSITION
X	X	H	X	X	HOLD MOTOR IN CURRENT POSITION
C	L	L	H	X	SET OUTPUTS TO STEP 1 LEVELS
C	L	L	L	L	STEP MOTOR CLOCKWISE
C	L	L	L	H	STEP MOTOR COUNTER-CLOCKWISE

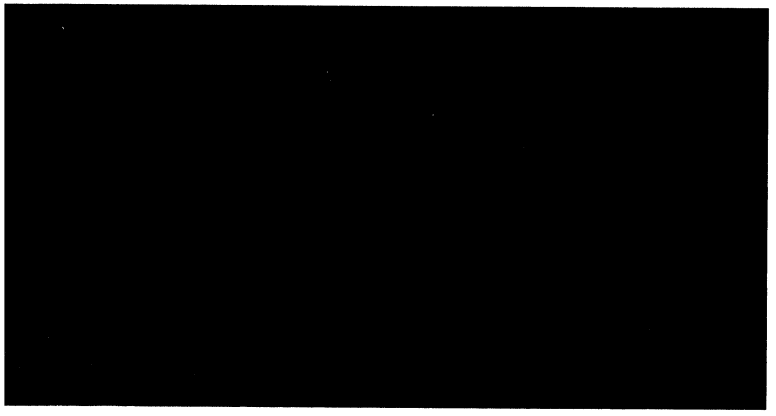
Stepper Motor Controller

Logic Diagram PAL16R4

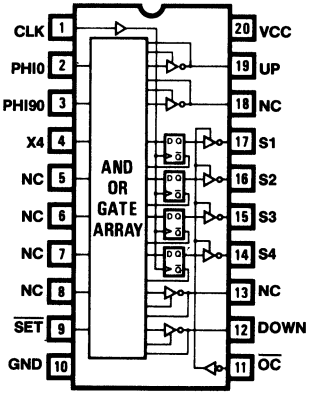



IMPORTANT
NOTES

Shaft Encoder

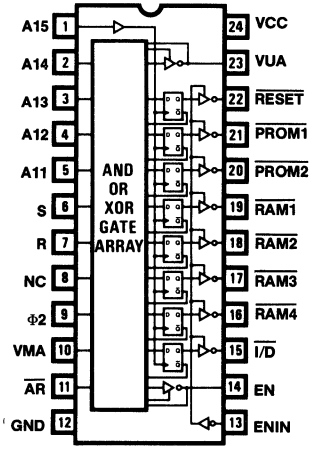


PAL16R4

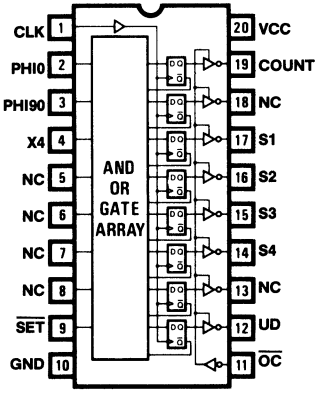


6

PAL20X8



PAL16R8



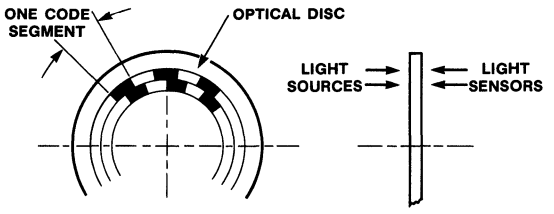
USING PAL® to GET RELIABLE DIGITAL DATA FROM SHAFT ENCODER CIRCUITS

The trend away from analog systems to digital numerical control systems focuses new attention on instruments that convert the analog output of position sensing devices into digital format.

The recording of the position of moveable parts of a machine requires high accuracy, as well as noise immunity, which is possible to attain through shaft encoder circuits of the type used in speed controllers and optical position sensing devices.

Principles of Shaft Encoding

Most common used are opto electrical encoders. The advantages of this type of circuits are direct shaft to digital encoding, a minimum number of power supplies, low power requirements, low cost and high speed.



Optical encoders measure shaft rotation by detecting the light which passes through a rotating code disc and a fixed slit.

It consists of a number of light sources and sensors whose paths are interrupted by a disc that has transparent and apaque areas.

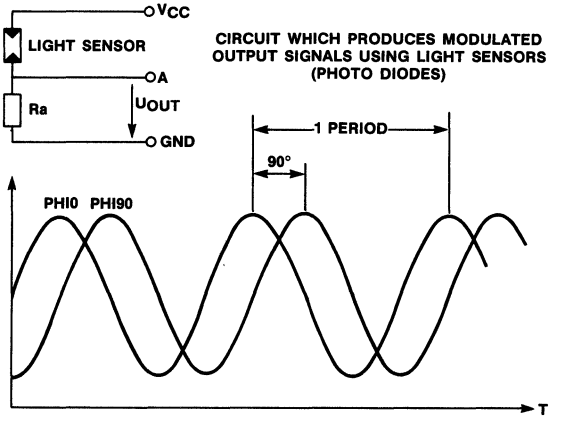
By using various concentric patterns, the shaft position can be defined. Light shines through the disc onto a sensor. The sensor turns on when the light passes through a transparent part, giving an output depending on the opacity of the segment. The combined output of all the segments is a digital information representing the disc and the shaft position.

— an **ABSOLUTE ENCODER** has a number of concentric tracks on the code disc, which provide a whole word parallel readout of shaft angle without the necessity of counting pulses. The code patterns on the disc are a kind of storage.

Therefore the readout is also present after a power interruption.

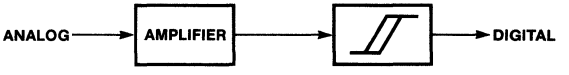
— **INCREMENTAL ENCODERs**, the type which will be described here, have a single code track on the disc, and angular position is determined by counting pulses produced by the modulated light falling onto the photodetectors.

Direction sensing is obtained by the use of quadrature signals which are provided by appropriate phasing of the paths.



Therefore, if the power is interrupted an INCREMENTAL ENCODER must have the zero position reestablished.

The two signals gained from the light sensors have to be amplified and converted into digital format using Schmitt-Triggers.



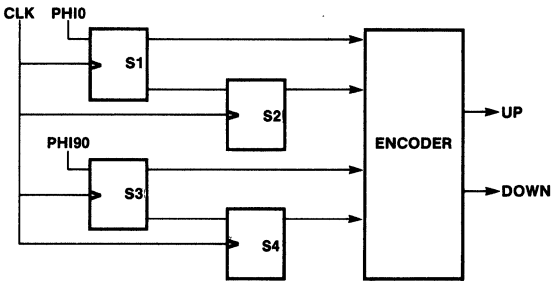
The two signals are dephased from each other by a quarter of a period. The direction of the movement can be determined if one signal leads or lags the other.

Out of his phase relationship the shaft encoder produced the information about direction and speed for the following UP/DOWN-counter.

To avoid random discrepancies during switching operations it is recommended to build shaft encoders using synchronous clocked circuitry.

6

The picture below shows a typical circuit diagram for SHAFT ENCODING:



The phasing of the two signals PHI0 and PHI90 is discriminated by the four registers and their outputs are encoded.

The principle of the above circuit is the time delay between the clock pulses for the two signals PHI0 and PHI90.

S1 = PHI0 clk S2 = PHI0 clk + 1

S3 = PHI90 clk S4 = PHI90 clk + 1

A transition in the signals PHI0 and PHI90 from L to H or vice versa causes a change in the data stored in the registers which is then encoded for direction sensing.

The logic equations for the encoder circuitry are given below

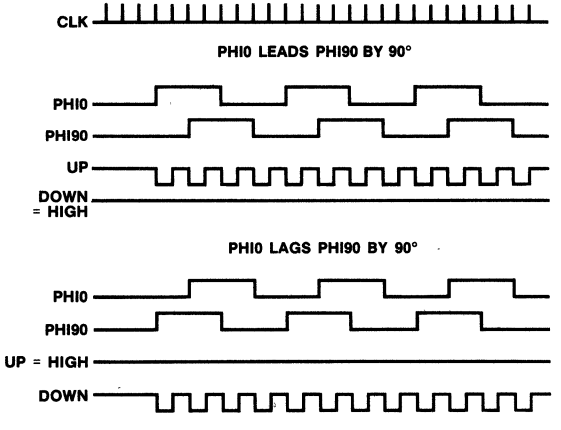
PHI0 leads PHI90:

S1 * S2 * S3 * /S4
/S1 * /S2 * /S3 * S4
S1 * /S2 * /S3 * /S4
/S1 * S2 * S3 * S4

PHI0 lags PHI90:

/S1 * /S2 * S3 * /S4
S1 * S2 * /S3 * S4
S1 * /S2 * S3 * S4
/S1 * S2 * /S3 * /S4

* stands for AND-Function.



Let us assume that when the shaft rotates in a clockwise direction, the input PHI0 leads the input PHI90 by 90° and the logic will generate pulses only at the UP-output.

On the other hand, when the shaft's rotation is in the counterclockwise direction the input PHI0 lags the input PHI90 by 90°.

In this case, four pulses per square wave are presented at the DOWN-output. Note that both the UP and DOWN outputs of the counter are normally held high.

To ensure that no shaft encoder transition is missed, the clock frequency should be at least $8 \times n \times s$, where n is the number of pulses produced by the encoder for each shaft revolution and s is the maximum speed in revolution per second to be expected.

Most common used are CLK frequencies in the range of 1MHz for optimum circuit operation.

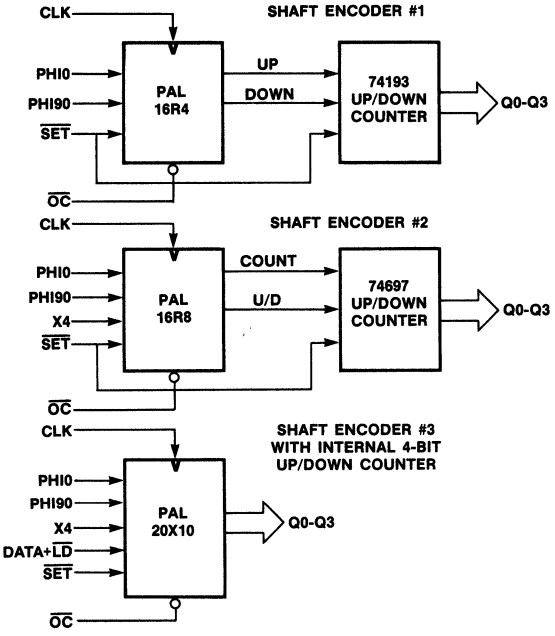
Synchronous two channel shaft encoders as described above are relatively insensitive to encoder phase errors. They use no temperamental Mono-Flops and can detect the occurence of illegal transition states generated by the circuit.

It is the property of such kind of circuits that interference on the input lines are ignored if they occur between two clock cycles.

Random noise on both input lines results in UP and DOWN count so that in the end the digital information remains unchanged.

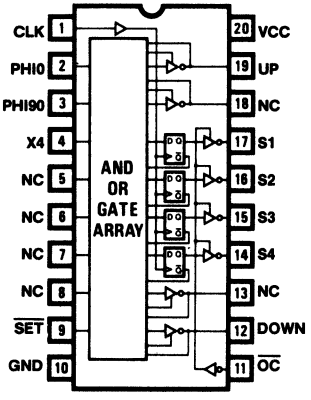
Hence synchronous SHAFT ENCODERS are extremely useful in electrically noisy environments.

The following PAL Applications are examples of such circuits using a single PAL.



```
PAL16R4                                                PAL DESIGN SPECIFICATION
P7016                                                     WILLY VOLDAN 09/09/82
SHAFT ENCODER No. 1
MMI GMBH MUNICH
CLK PHI0 PHI90 X4 NC NC NC NC /SET GND
/OC DOWN  NC   S4 S3 S2 S1 NC  UP  VCC


/S1 :=  /PHI0                                          ;CHECK FOR PHI0
     +   SET                                           ;INITIALIZE S1=L

/S2 :=  /S1                                            ;CHECK FOR S1
     +   SET                                           ;INITIALIZE S2=L

/S3 :=  /PHI90                                         ;CHECK FOR PHI90
     +   SET                                           ;INITIALIZE S3=L

/S4 :=  /S3                                            ;CHECK FOR S3
     +   SET                                           ;INITIALIZE S4=L

IF (VCC) /DOWN =  S1* S2* S3*/S4* PHI0* PHI90          ;PHI0 LEADS PHI90
               + /S1*/S2*/S3* S4*/PHI0*/PHI90          ;PHI0 LEADS PHI90
               +  S1*/S2*/S3*/S4* PHI0*/PHI90          ;PHI0 LEADS PHI90
               + /S1* S2* S3* S4*/PHI0* PHI90          ;PHI0 LEADS PHI90

IF (VCC) /UP   = /S1*/S2* S3*/S4*/PHI0* PHI90          ;PHI90 LEADS PHI0
               +  S1* S2*/S3* S4* PHI0*/PHI90          ;PHI90 LEADS PHI0
               +  S1*/S2* S3* S4* PHI0* PHI90          ;PHI90 LEADS PHI0
               + /S1* S2*/S3*/S4*/PHI0*/PHI90          ;PHI90 LEADS PHI0
```

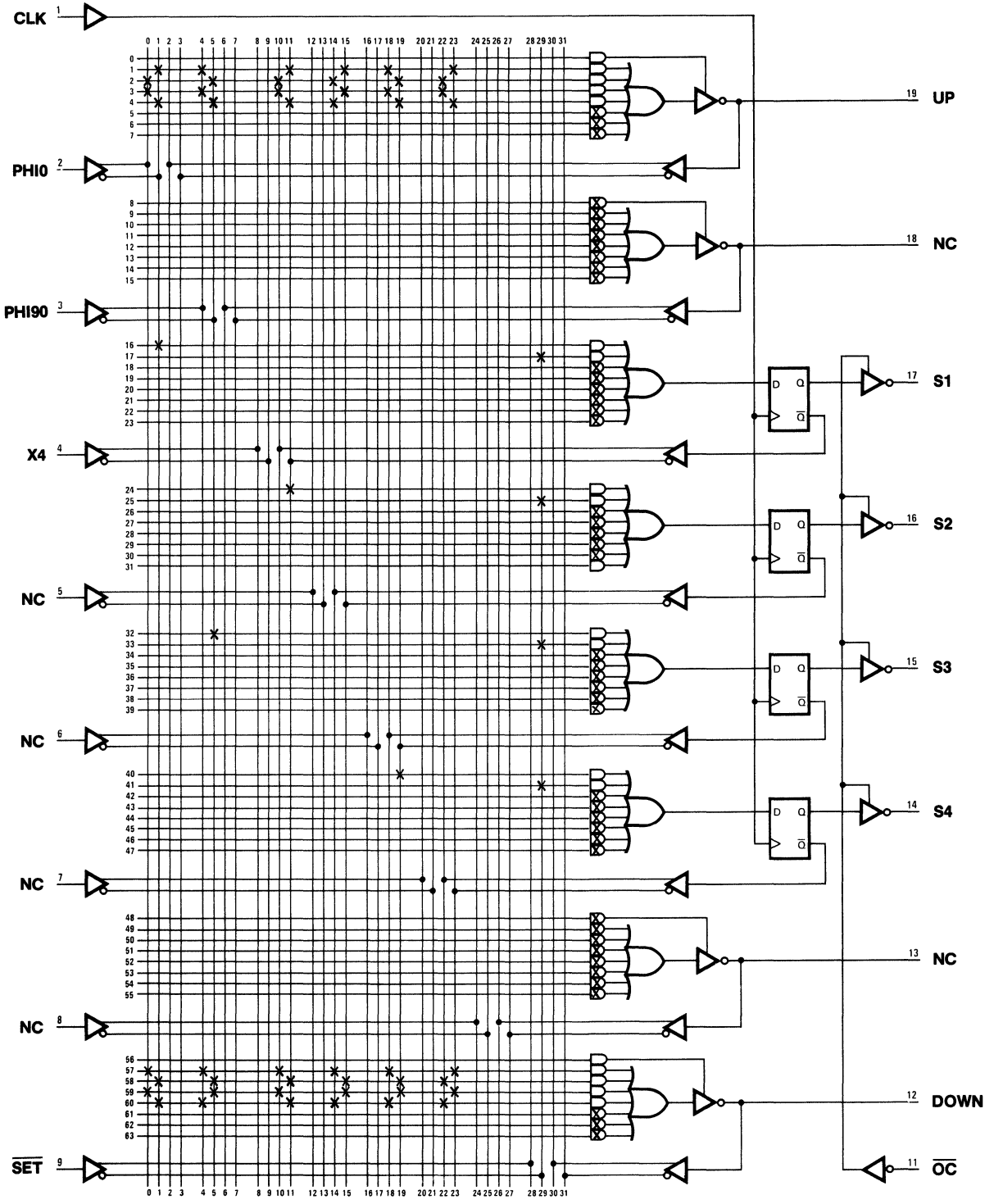

PAL16R4



```
FUNCTION TABLE
CLK /OC /SET PHI0 PHI90 S4 S3 S2 S1 UP DOWN
;---CONTROLS---       --INPUTS--      SSSS      OUTPUTS
;CLK  /OC  /SET       PHI0 PHI90      4321      UP DOWN       COMMENTS
---------------------------------------------------------------------------
  C    L    L          X    X         LLLL      H   H         CLEAR REGISTERS
  C    L    H          L    L         LLLL      H   H
  C    L    H          L    L         LLLL      H   H
  C    L    H          L    H         LHLL      L   H         COUNT UP
  C    L    H          L    H         HHLL      H   H
  C    L    H          H    H         HHLH      L   H         COUNT UP
  C    L    H          H    H         HHHH      H   H
  C    L    H          H    L         HLHH      L   H         COUNT UP
  C    L    H          H    L         LLHH      H   H
  C    L    H          L    L         LLHL      L   H         COUNT UP
  C    L    H          L    L         LLLL      H   H
  C    L    H          L    H         LHLL      L   H         COUNT UP
  C    L    H          L    H         HHLL      H   H
  C    L    H          H    H         HHLH      L   H         COUNT UP
  C    L    H          H    H         HHHH      H   H
  C    L    H          H    L         HLHH      L   H         COUNT UP
  C    L    H          H    L         LLHH      H   H
  C    L    H          L    L         LLHL      L   H         COUNT UP
  C    L    H          L    L         LLLL      H   H
  C    L    H          L    H         LHLL      L   H         COUNT UP
  C    L    H          L    H         HHLL      H   H
  C    L    H          H    H         HHLH      L   H         COUNT UP
  C    L    H          H    H         HHHH      H   H
  C    L    H          H    L         HLHH      L   H         COUNT UP
  C    L    H          H    L         LLHH      H   H
  C    L    H          L    L         LLHL      L   H         COUNT UP
  C    L    H          L    L         LLLL      H   H
  C    L    L          X    X         LLLL      H   H         CLEAR REGISTERS
  C    L    H          L    L         LLLL      H   H
  C    L    H          L    L         LLLL      H   H
  C    L    H          H    L         LLLH      H   L         COUNT DOWN
  C    L    H          H    L         LLHH      H   H
  C    L    H          H    H         LHHH      H   L         COUNT DOWN
  C    L    H          H    H         HHHH      H   H
  C    L    H          L    H         HHHL      H   L         COUNT DOWN
  C    L    H          L    H         HHLL      H   H
  C    L    H          L    L         HLLL      H   L         COUNT DOWN
  C    L    H          L    L         LLLL      H   H
  C    L    H          H    L         LLLH      H   L         COUNT DOWN
  C    L    H          H    L         LLHH      H   H
  C    L    H          H    H         LHHH      H   L         COUNT DOWN
  C    L    H          H    H         HHHH      H   H
  C    L    H          L    H         HHHL      H   L         COUNT DOWN
  C    L    H          L    H         HHLL      H   H
  C    L    H          L    L         HLLL      H   L         COUNT DOWN
  C    L    H          L    L         LLLL      H   H
  C    L    H          H    L         LLLH      H   L         COUNT DOWN
  C    L    H          H    L         LLHH      H   H
  C    L    H          H    H         LHHH      H   L         COUNT DOWN
  C    L    H          H    H         HHHH      H   H
  C    L    H          L    H         HHHL      H   L         COUNT DOWN
  C    L    H          L    H         HHLL      H   H
  C    L    H          L    L         HLLL      H   L         COUNT DOWN
  C    L    H          L    L         LLLL      H   H
  X    H    X          X    X         ZZZZ      X   X         TEST HI-Z
```

Shaft Encoder No. 1

Logic Diagram PAL16R4



```
PAL16R8                                                PAL DESIGN SPECIFICATION
P7095                                                    WILLY VOLDAN 09/09/82
SHAFT ENCODER No. 2
MMI GMBH MUNICH
CLK PHI0 PHI90 X4 NC NC NC NC /SET GND
/OC UD NC S4 S3 S2 S1 NC COUNT VCC


/S1    := /PHI0                                    ;CHECK FOR PHI0
        +  SET                                     ;INITIALIZE S1=L

/S3    := /PHI90                                   ;CHECK FOR PHI90
        +  SET                                     ;INITIALIZE S3=L

/S2    :=  S1                                      ;CHECK FOR /S1
        +  SET                                     ;INITIALIZE S2=L

/S4    :=  S3                                      ;CHECK FOR /S3
        +  SET                                     ;INITIALIZE S4=L

/COUNT :=  S1* S2*/S3* S4                          ;THIS OUTPUT ALTERNATES
        + /S1*/S2* S3*/S4                          ;BETWEEN HIGH AND LOW WITH
        + /S1* S2*/S3*/S4* X4                      ;HALF OR QUARTER THE
        +  S1*/S2* S3* S4* X4                      ;CLK FREQUENCY
        +  S1* S2* S3*/S4
        + /S1*/S2*/S3* S4
        + /S1* S2* S3* S4* X4
        +  S1*/S2*/S3*/S4* X4

/UD    := /S1* S2*/S3* S4                          ;THIS OUTPUT DETERMINES
        + /S1* S2* S3* S4                          ;IF SIGNAL PHI0 LEADS
        + /S1* S2* S3*/S4                          ;OR LAGS SIGNAL PHI90
        +  S1* S2* S3*/S4
        +  S1*/S2* S3*/S4
        +  S1*/S2*/S3*/S4
        +  S1*/S2*/S3* S4
        + /S1*/S2*/S3* S4
```

DESCRIPTION

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

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

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

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

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

FUNCTION TABLE

CLK /OC /SET PHI0 PHI90 X4 S1 S2 S3 S4 COUNT UD

;---CONTROLS---			--INPUTS--		X	SSSS	-OUTPUTS-		
;CLK	/OC	/SET	PHI0	PHI90	4	1234	COUNT	UD	COMMENTS
C	L	L	X	X	X	LLLL	H	H	CLEAR REGISTERS
C	L	H	L	L	L	LHLH	H	H	COUNT UP X4=L
C	L	H	H	L	L	HHLH	H	L	
C	L	H	H	L	L	HLLH	L	H	
C	L	H	H	H	L	HLHH	H	L	
C	L	H	H	H	L	HLHL	H	H	
C	L	H	L	H	L	LLHL	H	L	
C	L	H	L	H	L	LHHL	L	H	
C	L	H	L	L	L	LHLL	H	L	
C	L	H	L	L	L	LHLH	H	H	
C	L	H	H	L	L	HHLH	H	L	
C	L	H	H	L	L	HLLH	L	H	
C	L	H	H	H	L	HLHH	H	L	
C	L	H	H	H	L	HLHL	H	H	
C	L	H	L	H	L	LLHL	H	L	
C	L	H	L	H	L	LHHL	L	H	
C	L	H	L	L	H	LHLL	H	L	COUNT UP X4=L
C	L	H	L	L	H	LHLH	L	H	
C	L	H	H	L	H	HHLH	H	L	
C	L	H	H	L	H	HLLH	L	H	
C	L	H	H	H	H	HLHH	H	L	
C	L	H	H	H	H	HLHL	L	H	
C	L	H	L	H	H	LLHL	H	L	
C	L	H	L	H	H	LHHL	L	H	
C	L	H	L	L	H	LHLL	H	L	
C	L	H	L	L	H	LHLH	L	H	
C	L	H	H	L	H	HHLH	H	L	
C	L	H	H	L	H	HLLH	L	H	
C	L	H	H	H	H	HLHH	H	L	
C	L	H	H	H	H	HLHL	L	H	
C	L	L	X	X	X	LLLL	H	L	CLEAR REGISTERS
C	L	H	L	L	L	LHLH	H	H	COUNT DOWN X4=L
C	L	H	L	H	L	LHHH	H	L	
C	L	H	L	H	L	LHHL	H	L	
C	L	H	H	H	L	HHHL	H	L	
C	L	H	H	H	L	HLHL	L	L	
C	L	H	H	L	L	HLLL	H	L	
C	L	H	H	L	L	HLLH	H	L	
C	L	H	L	L	L	LLLH	H	L	
C	L	H	L	L	L	LHLH	L	L	
C	L	H	L	H	L	LHHH	H	L	
C	L	H	L	H	L	LHHL	H	L	
C	L	H	H	H	L	HHHL	H	L	
C	L	H	H	H	L	HLHL	L	L	
C	L	H	H	L	L	HLLL	H	L	
C	L	H	H	L	L	HLLH	H	L	
C	L	H	L	L	H	LLLH	H	L	COUNT DOWN X4=H
C	L	H	L	L	H	LHLH	L	L	

;---CONTROLS---			--INPUTS--		X	SSSS	-OUTPUTS-		
;CLK	/OC	/SET	PHI0	PHI90	4	1234	COUNT	UD	COMMENTS
C	L	H	L	H	H	LHHH	H	L	
C	L	H	L	H	H	LHHL	L	L	
C	L	H	H	H	H	HHHL	H	L	
C	L	H	H	H	H	HLHL	L	L	
C	L	H	H	L	H	HLLL	H	L	
C	L	H	H	L	H	HLLH	L	L	
C	L	H	L	L	H	LLLH	H	L	
C	L	H	L	L	H	LHLH	L	L	
C	L	H	L	H	H	LHHH	H	L	
C	L	H	L	H	H	LHHL	L	L	
C	L	H	H	H	H	HHHL	H	L	
C	L	H	H	H	H	HLHL	L	L	
X	H	X	X	X	X	ZZZZ	Z	Z	TEST HI-Z

DESCRIPTION

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

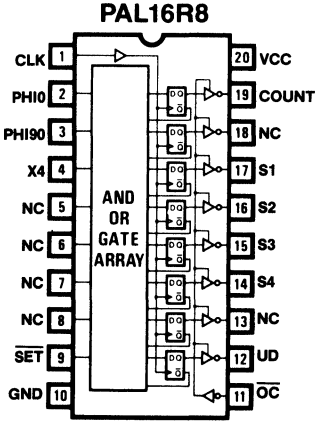
THE "COUNT" OUTPUT OF THE PAL IS NORMALLY HIGH. DURING SHAFT ENCODING THIS OUTPUT ALTERNATES BETWEEN HIGH AND LOW.

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

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

THE SHAFT ENCODER FEATURES THE CONFIGURATION AND OUTPUT POLARITY TO DRIVE AN 74S697 TYPE UP/DOWN COUNTER.

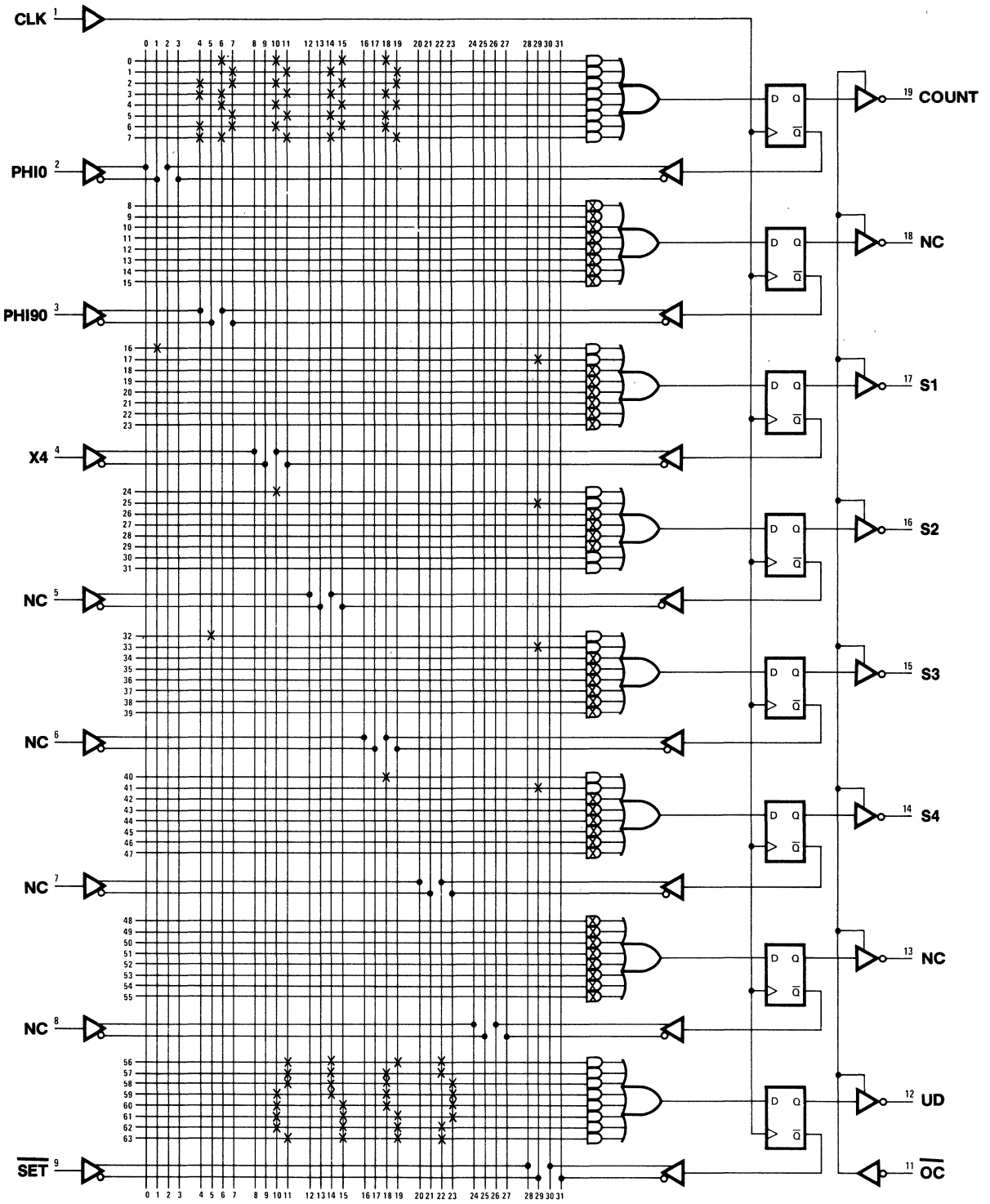
THIS DESIGN WITH GLITCHFREE OUTPUTS WILL BE EXTREMELY USEFUL IN ELECTRICALLY NOISY ENVIRONMENTS. THE PINNING IS GIVEN AS A FIRST PROPOSAL AND CAN BE CHANGED ACCORDING TO THE PC-BOARD LAYOUT.



6

Shaft Encoder No. 2

Logic Diagram PAL16R8



```
PAL20X10                                                PAL DESIGN SPECIFICATION
P7096                                                     WILLY VOLDAN 09/09/82
SHAFT ENCODER No. 3 (WITH INTERNAL 4-BIT UP/DOWN COUNTER)
MMI GMBH MUNICH
CLK PHI0 PHI90 X4 /LD NC D3 D2 D1 D0 /SET GND
/OC DOWN S4 S3 S2 S1 Q3 Q2 Q1 Q0 UP VCC

  /S1 :=  /PHI0                                 ;CHECK FOR PHI0
       +   SET                                  ;INITIALIZE S1=L

  /S2 :=  /S1                                   ;CHECK FOR S1
       +   SET                                  ;INITIALIZE S2=L

  /S3 :=  /PHI90                                ;CHECK FOR PHI90
       +   SET                                  ;INITIALIZE S3=L

  /S4 :=  /S3                                   ;CHECK FOR S3
       +   SET                                  ;INITIALIZE S4=L

/DOWN :=   S1* S2* S3*/S4* PHI0* PHI90* X4      ;PHI0 LEADS PHI90 - COUNT=FREQ/2
       +  /S1*/S2*/S3* S4*/PHI0*/PHI90* X4      ;PHI0 LEADS PHI90 - COUNT=FREQ/2
      :+:  S1*/S2*/S3*/S4* PHI0*/PHI90          ;PHI0 LEADS PHI90 - COUNT=FREQ/4
       +  /S1* S2* S3* S4*/PHI0* PHI90          ;PHI0 LEADS PHI90 - COUNT=FREQ/4

/UP   :=  /S1*/S2* S3*/S4*/PHI0* PHI90          ;PHI90 LEADS PHI0 - COUNT=FREQ/4
       +   S1* S2*/S3* S4* PHI0*/PHI90          ;PHI90 LEADS PHI0 - COUNT=FREQ/4
      :+:  S1*/S2* S3* S4* PHI0* PHI90* X4      ;PHI90 LEADS PHI0 - COUNT=FREQ/2
       +  /S1* S2*/S3*/S4*/PHI0*/PHI90* X4      ;PHI90 LEADS PHI0 - COUNT=FREQ/2

/Q0   :=  /SET* LD*/D0                          ;LOAD D0 (LSB)
       +  /SET*/LD*/Q0                          ;HOLD Q0
      :+: /SET*/LD* UP*/DOWN                    ;DECREMENT
       +  /SET*/LD*/UP* DOWN                    ;INCREMENT

/Q1   :=  /SET* LD*/D1                          ;LOAD D1
       +  /SET*/LD*/Q1                          ;HOLD Q1
      :+: /SET*/LD* UP*/DOWN*/Q0                ;DECREMENT
       +  /SET*/LD*/UP* DOWN* Q0                ;INCREMENT

/Q2   :=  /SET* LD*/D2                          ;LOAD D2
       +  /SET*/LD*/Q2                          ;HOLD Q2
      :+: /SET*/LD* UP*/DOWN*/Q0*/Q1            ;DECREMENT
       +  /SET*/LD*/UP* DOWN* Q0* Q1            ;INCREMENT

/Q3   :=  /SET* LD*/D3                          ;LOAD D3 (MSB)
       +  /SET*/LD*/Q3                          ;HOLD Q3
      :+: /SET*/LD* UP*/DOWN*/Q0*/Q1*/Q2        ;DECREMENT
       +  /SET*/LD*/UP* DOWN* Q0* Q1* Q2        ;INCREMENT
```

FUNCTION TABLE

CLK /OC /SET /LD X4 PHI0 PHI90 S1 S2 S3 S4 UP DOWN D3 D2 D1 D0 Q3 Q2 Q1 Q0

;CLK	----CONTROLS---- / OC	/ SET	/ LD	X4	INPUT PHI 0	90	SSSS 1234	UP	DOWN	DDDD 3210	QQQQ 3210	COMMENTS (Q HEX VALUE)
C	L	L	X	X	X	X	LLLL	H	H	XXXX	HHHH	INITIALIZE REGISTERS (F)
C	L	H	L	X	X	X	XXXX	X	X	HLHL	HLHL	LOAD (A)
C	L	H	H	L	L	L	LLLL	H	H	XXXX	HLHL	HOLD (A)
C	L	H	H	L	H	L	HLLL	H	H	XXXX	HLHL	HOLD (A) PHI0 LEADS PHI90
C	L	H	H	L	H	L	HHLL	H	L	XXXX	HLHL	HOLD (A) X4=H - FREQ/4
C	L	H	H	L	H	H	HHHL	H	H	XXXX	HLLH	DECREMENT (9)
C	L	H	H	L	H	H	HHHH	H	H	XXXX	HLLH	HOLD (9)
C	L	H	H	L	L	H	LHHH	H	H	XXXX	HLLH	HOLD (9)
C	L	H	H	L	L	H	LLHH	H	L	XXXX	HLLH	HOLD (9)
C	L	H	H	L	L	L	LLLH	H	H	XXXX	HLLL	DECREMENT (8)
C	L	H	H	L	L	L	LLLL	H	H	XXXX	HLLL	HOLD (8)
C	L	H	H	L	H	L	HLLL	H	H	XXXX	HLLL	HOLD (8)
C	L	H	H	L	H	L	HHLL	H	L	XXXX	HLLL	HOLD (8)
C	L	H	H	L	H	H	HHHL	H	H	XXXX	LHHH	DECREMENT (7)
C	L	H	H	L	H	H	HHHH	H	H	XXXX	LHHH	HOLD (7)
C	L	H	H	L	L	H	LHHH	H	H	XXXX	LHHH	HOLD (7)
C	L	H	H	L	L	H	LLHH	H	L	XXXX	LHHH	HOLD (7)
C	L	H	H	H	L	L	LLLH	H	H	XXXX	LHHL	DECREMENT (6)
C	L	H	H	H	L	L	LLLL	H	L	XXXX	LHHL	HOLD (6) X4=H - FREQ/2
C	L	H	H	H	H	L	HLLL	H	H	XXXX	LHLH	DECREMENT (5)
C	L	H	H	H	H	L	HHLL	H	L	XXXX	LHLH	HOLD (5)
C	L	H	H	H	H	H	HHHL	H	H	XXXX	LHLL	DECREMENT (4)
C	L	H	H	H	H	H	HHHH	H	L	XXXX	LHLL	HOLD (4)
C	L	H	H	H	L	H	LHHH	H	H	XXXX	LLHH	DECREMENT (3)
C	L	H	H	H	L	H	LLHH	H	L	XXXX	LLHH	HOLD (3)
C	L	H	H	H	L	L	LLLH	H	H	XXXX	LLHL	DECREMENT (2)
C	L	H	H	H	L	L	LLLL	H	L	XXXX	LLHL	HOLD (2)
C	L	H	H	H	H	L	HLLL	H	H	XXXX	LLLH	DECREMENT (1)
C	L	H	H	H	H	L	HHLL	H	L	XXXX	LLLH	HOLD (1)
C	L	H	H	H	H	H	HHHL	H	H	XXXX	LLLL	DECREMENT (0)
C	L	H	H	H	H	H	HHHH	H	L	XXXX	LLLL	HOLD (0)
C	L	H	H	H	L	H	LHHH	H	H	XXXX	HHHH	DECREMENT (F)(ROLL UNDER)
C	L	H	H	H	L	H	LLHH	H	L	XXXX	HHHH	HOLD (F)
C	L	H	H	H	L	L	LLLH	H	H	XXXX	HHHL	DECREMENT (E)
C	L	H	L	X	X	X	XXXX	X	X	LHLH	LHLH	LOAD (5)
C	L	H	H	L	L	L	LLLL	H	H	XXXX	LHLH	HOLD (5)
C	L	H	H	L	L	H	LLHL	H	H	XXXX	LHLH	HOLD (5) PHI90 LEADS PHI0
C	L	H	H	L	L	H	LLHH	L	H	XXXX	LHLH	HOLD (5) X4=L - FREQ/4
C	L	H	H	L	H	H	HLHH	H	H	XXXX	LHHL	INCREMENT (6)
C	L	H	H	L	H	H	HHHH	H	H	XXXX	LHHL	HOLD (6)
C	L	H	H	L	H	L	HHLH	H	H	XXXX	LHHL	HOLD (6)
C	L	H	H	L	H	L	HHLL	L	H	XXXX	LHHL	HOLD (6)
C	L	H	H	L	L	L	LHLL	H	H	XXXX	LHHH	INCREMENT (7)
C	L	H	H	L	L	L	LLLL	H	H	XXXX	LHHH	HOLD (7)
C	L	H	H	L	L	H	LLHL	H	H	XXXX	LHHH	HOLD (7)
C	L	H	H	L	L	H	LLHH	L	H	XXXX	LHHH	HOLD (7)
C	L	H	H	L	H	H	HLHH	H	H	XXXX	HLLL	INCREMENT (8)
C	L	H	H	L	H	H	HHHH	H	H	XXXX	HLLL	HOLD (8)
C	L	H	H	L	H	L	HHLH	H	H	XXXX	HLLL	HOLD (8)

```
;----CONTROLS----  INPUT
;    /  /  /        PHI    SSSS          DDDD  QQQQ     COMMENTS
;CLK OC SET LD X4  0  90  1234  UP DOWN  3210  3210    (Q HEX VALUE)
-------------------------------------------------------------------------------
 C   L  H   H  L    H L   HHLL  L   H    XXXX  HLLL    HOLD (8)
 C   L  H   H  H    L L   LHLL  H   H    XXXX  HLLH    INCREMENT (9)
 C   L  H   H  H    L L   LLLL  L   H    XXXX  HLLH    HOLD (9)  X4=H - FREQ/2
 C   L  H   H  H    L H   LLHL  H   H    XXXX  HLHL    INCREMENT (A)
 C   L  H   H  H    L H   LLHH  L   H    XXXX  HLHL    HOLD (A)
 C   L  H   H  H    H H   HLHH  H   H    XXXX  HLHH    INCREMENT (B)
 C   L  H   H  H    H H   HHHH  L   H    XXXX  HLHH    HOLD (B)
 C   L  H   H  H    H L   HHLH  H   H    XXXX  HHLL    INCREMENT (C)
 C   L  H   H  H    H L   HHLL  L   H    XXXX  HHLL    HOLD (C)
 C   L  H   H  H    L L   LHLL  H   H    XXXX  HHLH    INCREMENT (D)
 C   L  H   H  H    L L   LLLL  L   H    XXXX  HHLH    HOLD (D)
 C   L  H   H  H    L H   LLHL  H   H    XXXX  HHHL    INCREMENT (E)
 C   L  H   H  H    L H   LLHH  L   H    XXXX  HHHL    HOLD (E)
 C   L  H   H  H    H H   HLHH  H   H    XXXX  HHHH    INCREMENT (F)
 C   L  H   H  H    H H   HHHH  L   H    XXXX  HHHH    HOLD (F)
 C   L  H   H  H    H L   HHLH  H   H    XXXX  LLLL    INCREMENT (0)  (ROLL OVER)
 C   L  H   H  H    H L   HHLL  L   H    XXXX  LLLL    HOLD (0)
 C   L  H   H  H    L L   LHLL  H   H    XXXX  LLLH    INCREMENT (1)
 C   L  H   L  X    X X   XXXX  X   X    LLLH  LLLH    LOAD (1)
 C   L  H   L  X    X X   XXXX  X   X    LLHH  LLHH    LOAD (3)
 C   L  H   L  X    X X   XXXX  X   X    LHLH  LHLH    LOAD (5)
 C   L  H   L  X    X X   XXXX  X   X    LHHH  LHHH    LOAD (7)
 C   L  H   L  X    X X   XXXX  X   X    HLLH  HLLH    LOAD (9)
 C   L  H   L  X    X X   XXXX  X   X    HLHH  HLHH    LOAD (B)
 C   L  H   L  X    X X   XXXX  X   X    HHLH  HHLH    LOAD (D)
 C   L  H   L  X    X X   XXXX  X   X    HHHH  HHHH    LOAD (F)
 X   H  X   X  X    X X   ZZZZ  Z   Z    XXXX  ZZZZ    TEST HI-Z
-------------------------------------------------------------------------------
```

DESCRIPTION

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

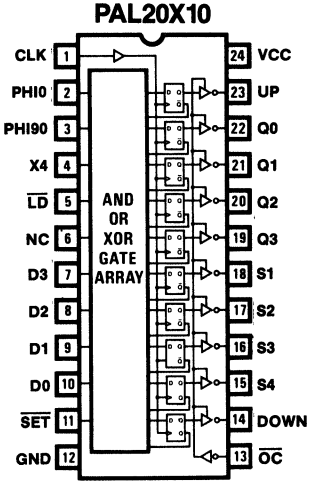
BOTH THE "UP" AND "DOWN" OUTPUTS OF THE PAL ARE NORMALLY AT HIGH.

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

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

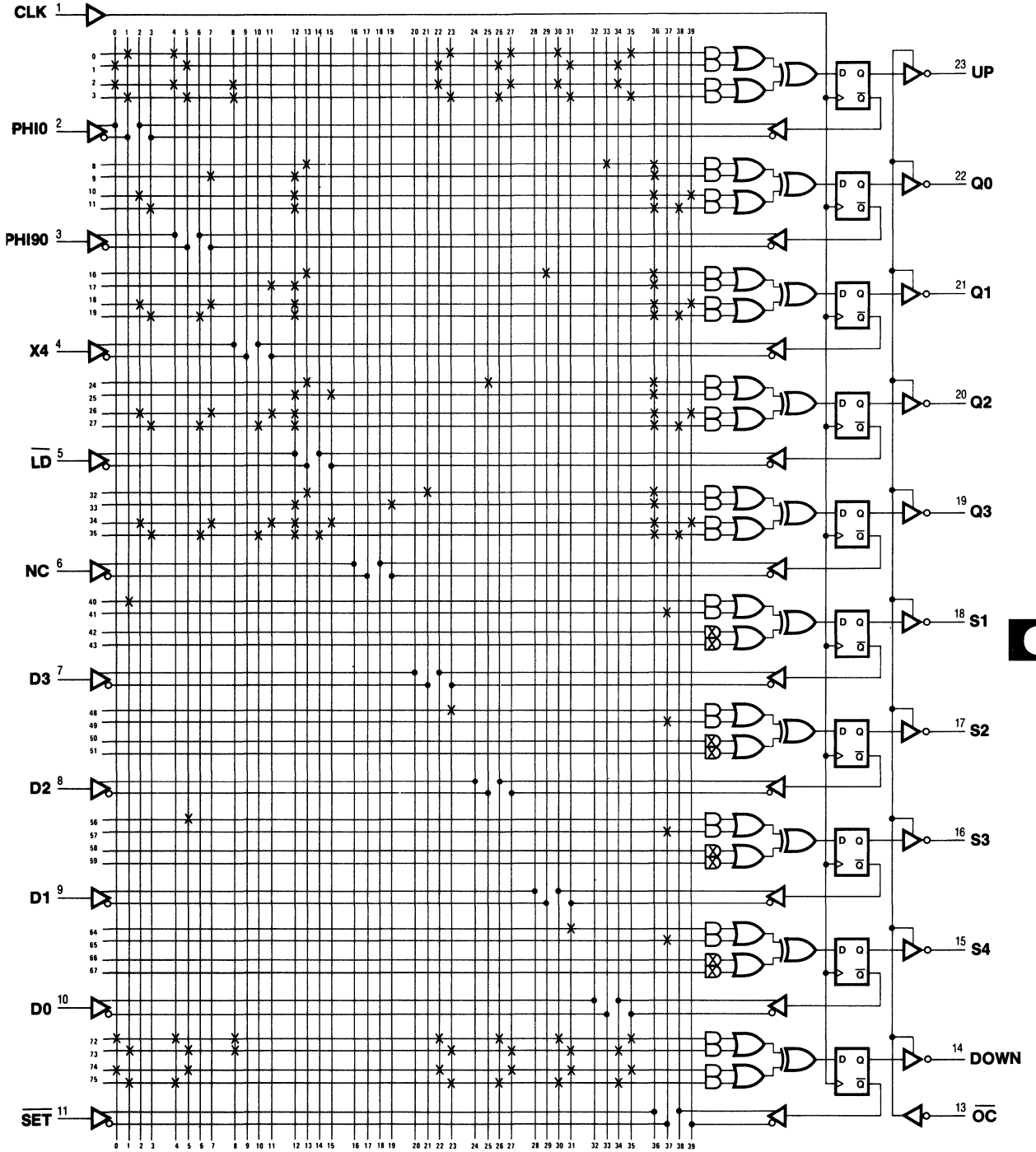
THE INTERNAL 4-BIT SYNCHRONOUS COUNTER HAS COUNT-UP, COUNT DOWN CAPABILITIES. ALSO, THE COUNTER CAN PARALLEL LOAD AND HOLD DATA INDEPENDENTLY OF THE SHAFT ENCODER SECTION. THE REGISTERS ARE SYNCHRONOUSLY INITIALIZED WHEN /SET IS HELD LOW.

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



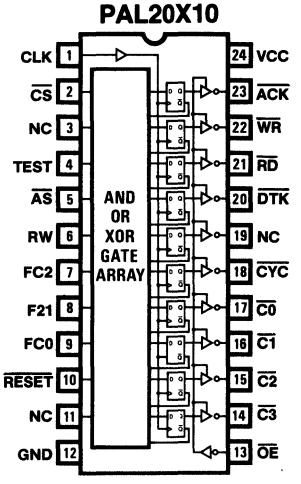
Shaft Encoder No. 3
(With Internal 4-Bit Up/Down Counter)

Logic Diagram PAL20X10




IMPORTANT
NOTES

Interface Controller for 68000 μP to Zilog 8500 Peripherals



6

```
PAL20X10                                                    PAL DESIGN SPECIFICATION
PMSI408                                                JANE LEE(MMI)/KEN THOMAS(ZILOG)
INTERFACE CONTROLLER FOR 68000 uP TO ZILOG 85XX PERIPHERALS
MMI SUNNYVALE, CALIFORNIA
CLK /CS  NC TEST /AS  RW  FC2  FC1  FC0 /RESET  NC  GND
/OC /C3 /C2 /C1  /C0 /CYC NC  /DTK /RD  /WR    /ACK VCC


C0  :=  /C0*/TEST                                          ;COUNT/HOLD (LSB)

C1  :=  /RESET* AS* C1                                     ;HOLD
    :+: /RESET* AS* C0                                     ;DECREMENT

C2  :=  /RESET* AS* C2                                     ;HOLD
    :+: /RESET* AS* C0* C1                                 ;DECREMENT

C3  :=  /RESET* AS* C3                                     ;HOLD
    :+: /RESET* AS* C0* C1* C2                             ;DECREMENT

DTK :=  /RESET*/ACK* CYC* C3*/C2*/C1* C0* CS               ;DTACK FOR RD/WR CYCLE
     +  /RESET* ACK* CYC* C3*/C2* C1*/C0                   ;DTACK FOR INTERRUPT
                                                           ;CYCLE

CYC :=  /RESET* AS*/CYC* C0                                ;START NEW CYCLE
     +  /RESET* AS* CYC                                    ;HOLD PROCESS OF CYCLE
    :+: /RESET* CYC* DTK                                   ;END OF CYCLE

RD  :=  /RESET* CYC*/ACK* RW* C3*/C2* CS                   ;NORMAL READ OPERATION
     +  /RESET* CYC*/ACK* RW*/C3* C2* C1* C0* CS           ;NORMAL READ OPERATION
    :+: /RESET* CYC* ACK* RW* C3                           ;READ DURING INTERRUPT
     +   RESET                                             ;RESET

WR  :=  /RESET* CYC*/ACK*/RW* C3*/C2* CS                   ;WRITE
     +  /RESET* CYC*/ACK*/RW*/C3* C2* C1* C0* CS           ;WRITE
    :+:  RESET                                             ;RESET

ACK :=  /RESET* FC0* FC1* FC2* AS* CYC*/C0                 ;START OF INTERRUPT
                                                           ;ACKNOWLEDGE
     +  /RESET* FC0* FC1* FC2* CYC                         ;HOLDS INTERRUPT
```

FUNCTION TABLE

CLK /OC /RESET /C3 /C2 /C1 /C0 TEST /CS FC2 FC1 FC0 RW /AS /CYC /RD /WR /DTK /ACK

;							/	//		
;C	/		T/	FFF		/	C//	DA		
;L	O/	CCCC	SC	CCC	R	A	YRW	TC		
;K	ER	3210	TS	210	W	S	CDR	KK	CYCLE	COMMENTS
C	LL	HHHH	HH	LLL	H	H	HLL	HH	1	INITIALIZATION
C	LH	HHHL	LH	LLL	H	H	HHH	HH	2	
C	LH	HHHH	LH	LLL	H	H	HHH	HH	3	
C	LH	HHHL	LL	LLL	H	H	HHH	HH	4	
C	LH	HHLH	LL	LLL	H	L	LHH	HH	5	BEGINNING OF WAIT FOR READ CYCLE
C	LH	HHLL	LL	LLL	H	L	LHH	HH	6	
C	LH	HLHH	LL	LLL	H	L	LHH	HH	7	
C	LH	HLHL	LL	LLL	H	L	LHH	HH	8	
C	LH	HLLH	LL	LLL	H	L	LHH	HH	9	
C	LH	HLLL	LL	LLL	H	L	LHH	HH	10	
C	LH	LHHH	LL	LLL	H	L	LLH	HH	11	/RD GOES ACTIVE
C	LH	LHHL	LL	LLL	H	L	LLH	HH	12	
C	LH	LHLH	LL	LLL	H	L	LLH	LH	13	ACKNOWLEDGE END OF DATA TRANSFER
C	LH	LHLL	LL	LLL	H	L	HLH	HH	14	END OF READ CYCLE
C	LH	HHHH	LH	LLL	H	H	HHH	HH	15	INITIALIZATION
C	LH	HHHL	LH	LLL	H	H	HHH	HH	16	
C	LH	HHLH	LL	LLL	L	L	LHH	HH	17	BEGINNING OF WAIT FOR WRITE CYCLE
C	LH	HHLL	LL	LLL	L	L	LHH	HH	18	
C	LH	HLHH	LL	LLL	L	L	LHH	HH	19	
C	LH	HLHL	LL	LLL	L	L	LHH	HH	20	
C	LH	HLLH	LL	LLL	L	L	LHH	HH	21	
C	LH	HLLL	LL	LLL	L	L	LHH	HH	22	
C	LH	LHHH	LL	LLL	L	L	LHL	HH	23	/WR GOES ACTIVE
C	LH	LHHL	LL	LLL	L	L	LHL	HH	24	
C	LH	LHLH	LL	LLL	L	L	LHL	LH	25	ACKNOWLEDGE END OF DATA TRANSFER
C	LH	LHLL	LL	LLL	L	L	HHL	HH	26	END OF WRITE CYLE
C	LH	HHHH	LH	LLL	H	H	HHH	HH	27	INITIALIZATION
C	LH	HHHL	LH	LLL	H	H	HHH	HH	28	
C	LH	HHLH	LH	HHH	H	L	LHH	HH	29	INTERRUPT CYCLE (WAITS RD SIGNAL)
C	LH	HHLL	LH	HHH	H	L	LHH	HL	30	
C	LH	HLHH	LH	HHH	H	L	LHH	HL	31	
C	LH	HLHL	LH	HHH	H	L	LHH	HL	32	
C	LH	HLLH	LH	HHH	H	L	LHH	HL	33	
C	LH	HLLL	LH	HHH	H	L	LHH	HL	34	
C	LH	LHHH	LH	HHH	H	L	LHH	HL	35	
C	LH	LHHL	LH	HHH	H	L	LLH	HL	36	/RD GOES ACTIVE
C	LH	LHLH	LH	HHH	H	L	LLH	HL	37	
C	LH	LHLL	LH	HHH	H	L	LLH	LL	38	
C	LH	LLHH	LH	HHH	H	L	HLH	HL	39	
C	LH	HHHL	LH	LLL	H	H	HHH	HH	40	END OF CYCLE

DESCRIPTION

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

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

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

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

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

FUNCTIONAL DESCRIPTION

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

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

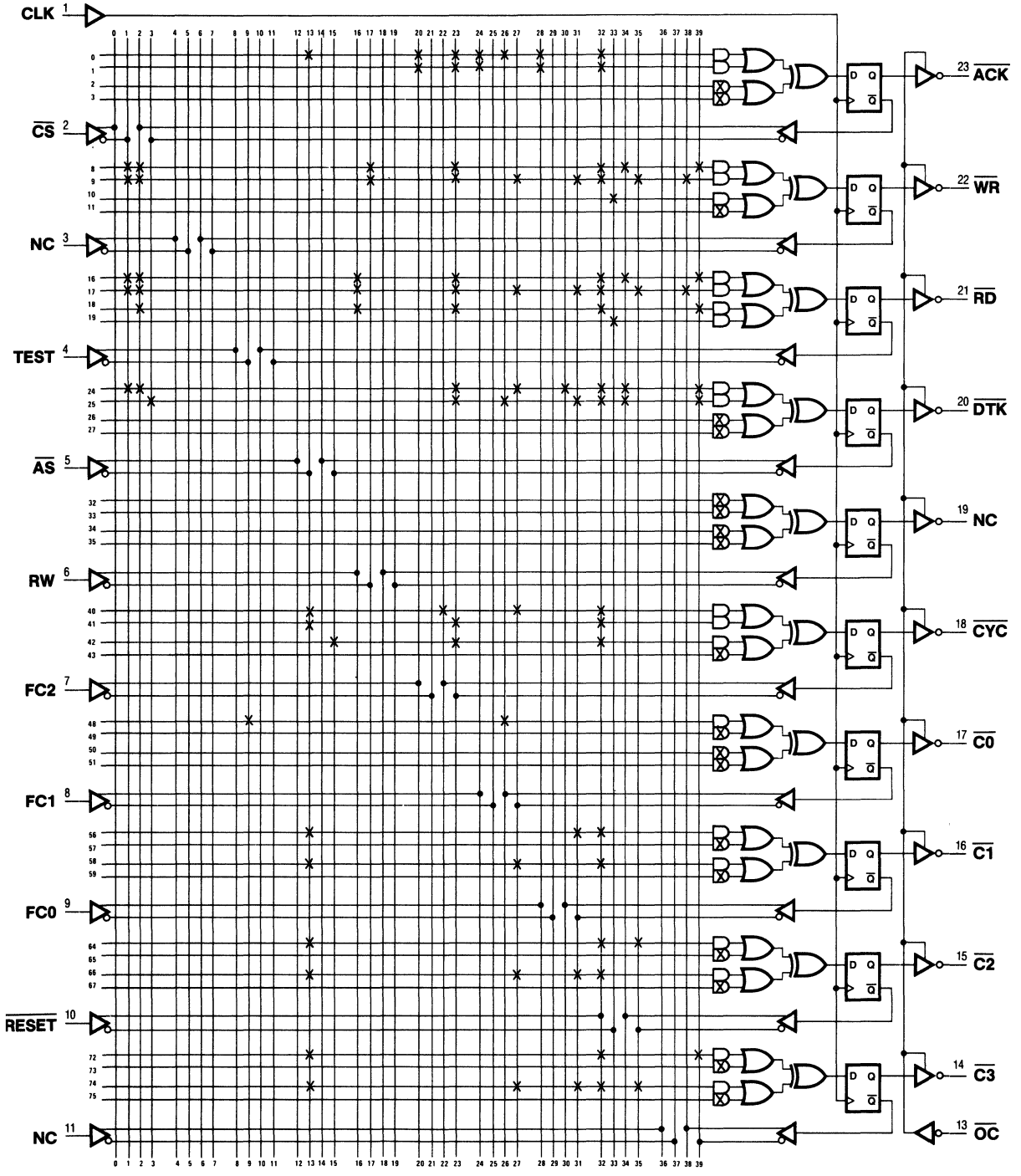
/C0, /C1, /C2, AND /C3 ARE PART OF A 4-BIT SYNCHRONOUS DOWN COUNTER WHERE /C3 IS THE MSB AND /C0 IS THE LSB. /C0 IS SIMPLY A DIVIDE BY 2 OF THE 68000 CLOCK USED BY THE PERIPHERAL FOR PCLK. THE OTHER THREE SIGNALS, C1, C2, C3 WITH /CYC ARE USED INTERNAL TO THE PAL FOR GENERATING THE ABOVE CONTROL SIGNALS.

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

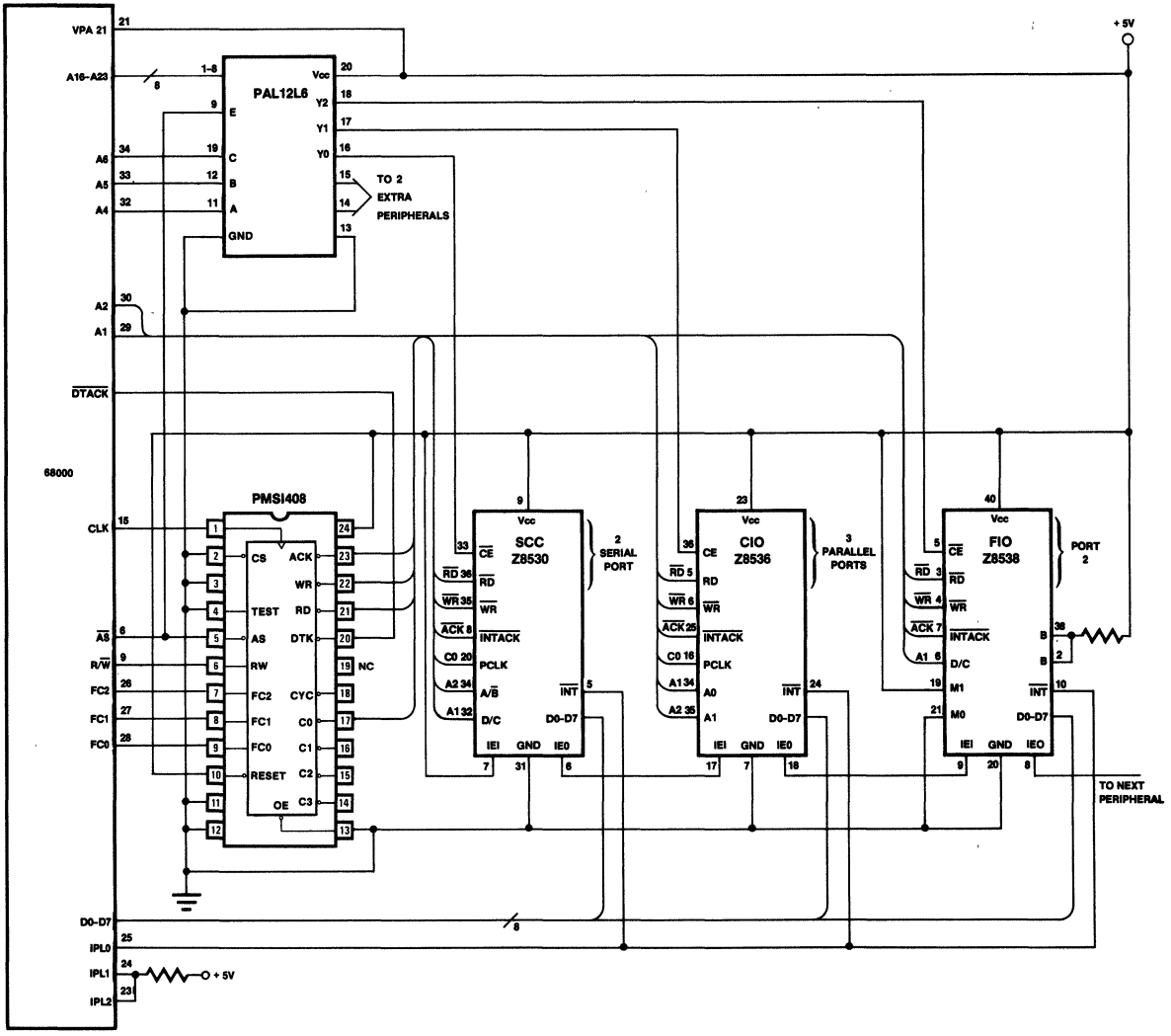
THE CONTROL SIGNALS, /RD, /WR, /ACK, AND /DTK ARE GENEREATED THROUGH THE PAL AND QUALIFIED WITH THE COUNTER TO GENERATE EDGES THAT MEET TIME RESTRAITS.

Interface Controller for 68000 μP to Zilog 8500 Peripherals

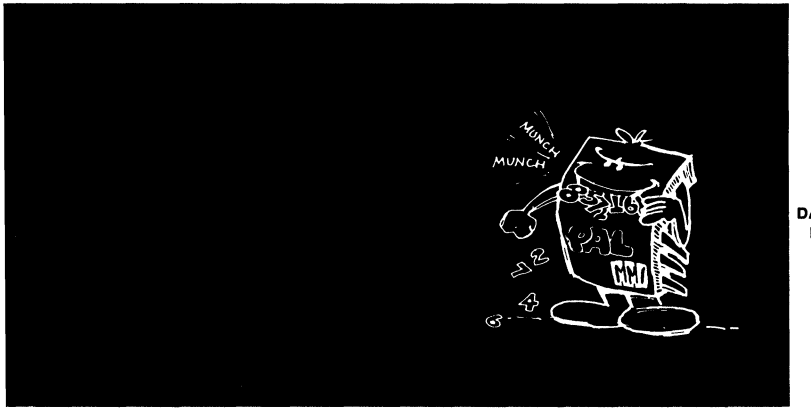
Logic Diagram PAL20X10

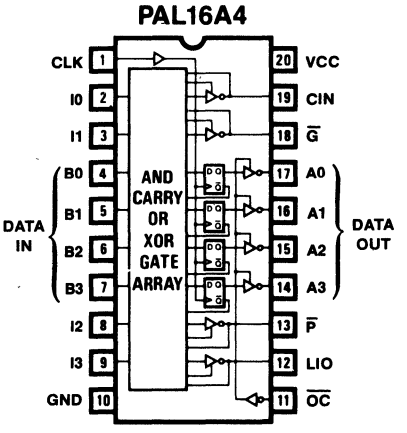


Zilog 8500 Interface with the 68000 Microprocessor



ALU/Accumulator





```
PAL16A4                                          PAL DESIGN SPECIFICATION
P7014                                              BIRKNER/COLI 07/15/81
ALU/ACCUMULATOR
MMI SUNNYVALE, CALIFORNIA
CLK I0  I1 B0 B1 B2 B3 I2 I3  GND
/OC LIO /P A3 A2 A1 A0 /G CIN VCC


CARRY0 .EQU   /I3*/I2*/I1*/I0 * CIN

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

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

CARRY3 .EQU   /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


/A0 :=  /I3*/I2*/I1*/I0*(A0:*:B0)   ;A0 PLUS B0
     +  /I3*/I2* I0*(/A0)           ;HOLD A0   (A0 AND)
     +  /I3*/I2* I1*(/B0)           ;LOAD B0   (  B0  )
     +  /I3* I2*/I1*/I0*(B0)        ;LOAD /B0
    :+: /I3* I2*/I1* I0*(/A0*/B0)   ;A0 OR B0
     +  /I3* I2* I1*/I0*/CIN        ;SHIFT LEFT A0
     +  /I3* I2* I1* I0*(/A1)       ;SHIFT RIGHT A0
     +   I3*(/A0)                   ;HOLD A0 (LSB)
     +   CARRY0                     ;A0 PLUS B0 PLUS 1

/A1 :=  /I3*/I2*/I1*/I0*(A1:*:B1)   ;A1 PLUS B1
     +  /I3*/I2* I0*(/A1)           ;HOLD A1   (A1 AND)
     +  /I3*/I2* I1*(/B1)           ;LOAD B1   (  B1  )
     +  /I3* I2*/I1*/I0*(B1)        ;LOAD /B1
    :+: /I3* I2*/I1* I0*(/A1*/B1)   ;A1 OR B1
     +  /I3* I2* I1*/I0*(/A0)       ;SHIFT LEFT A1
     +  /I3* I2* I1* I0*(/A2)       ;SHIFT RIGHT A1
     +   I3*(/A1)                   ;HOLD A1
     +   CARRY1                     ;A1 PLUS B1 PLUS 1

/A2 :=  /I3*/I2*/I1*/I0*(A2:*:B2)   ;A2 PLUS B2
     +  /I3*/I2* I0*(/A2)           ;HOLD A2   (A2 AND)
     +  /I3*/I2* I1*(/B2)           ;LOAD B2   (  B2  )
     +  /I3* I2*/I1*/I0*(B2)        ;LOAD /B2
    :+: /I3* I2*/I1* I0*(/A2*/B2)   ;A2 OR B2
     +  /I3* I2* I1*/I0*(/A1)       ;SHIFT LEFT A2
     +  /I3* I2* I1* I0*(/A3)       ;SHIFT RIGHT A2
     +   I3*(/A2)                   ;HOLD A2
     +   CARRY2                     ;A2 PLUS B2 PLUS 1
```

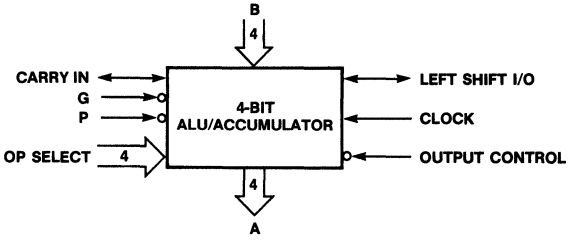
```
/A3 :=  /I3*/I2*/I1*/I0*(A3:*:B3)    ;A3 PLUS B3
     +  /I3*/I2* I0*(/A3)            ;HOLD A3    (A3 AND)
     +  /I3*/I2* I1*(/B3)            ;LOAD B3    (  B3  )
     +  /I3* I2*/I1*/I0*(B3)         ;LOAD /B3
    :+: /I3* I2*/I1* I0*(/A3*/B3)    ;A3 OR B3
     +  /I3* I2* I1*/I0*(/A2)        ;SHIFT LEFT A3
     +  /I3* I2* I1* I0*/LIO         ;SHIFT RIGHT A3
     +   I3*(/A3)                    ;HOLD A3 (MSB)
     +   CARRY3                      ;A3 PLUS B3 PLUS 1

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)    ;SHIFT LEFT OUT

IF (/I3* I2* I1* I0) /CIN = (/A0)    ;SHIFT RIGHT OUT
```

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.

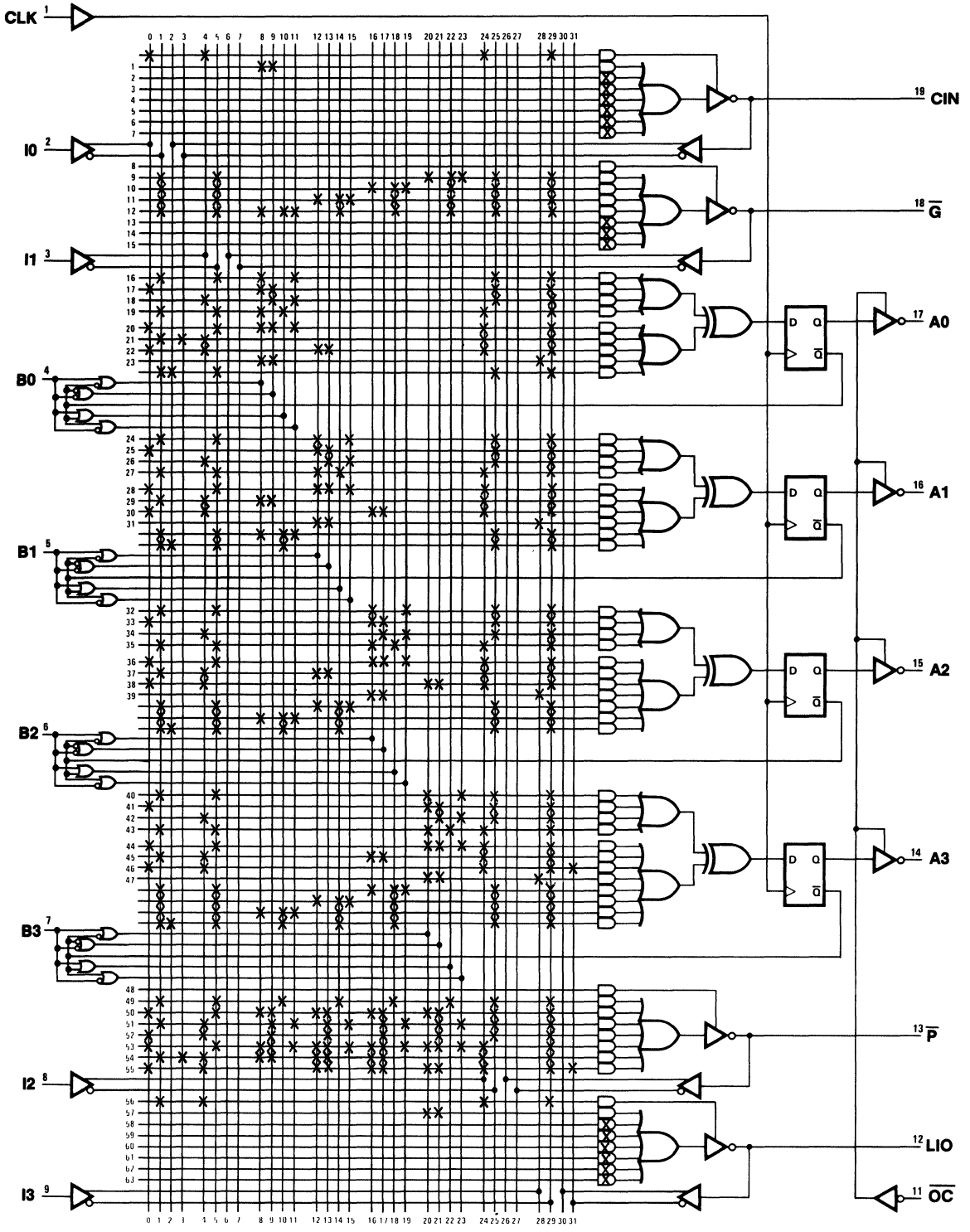
OPERATIONS TABLE:

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



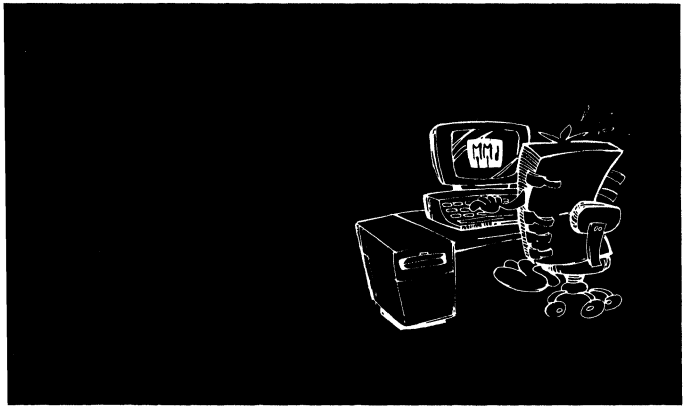
ALU/Accumulator

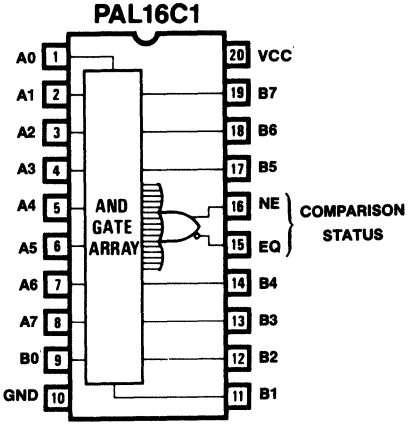
Logic Diagram PAL16A4




IMPORTANT
NOTES

Octal Comparator

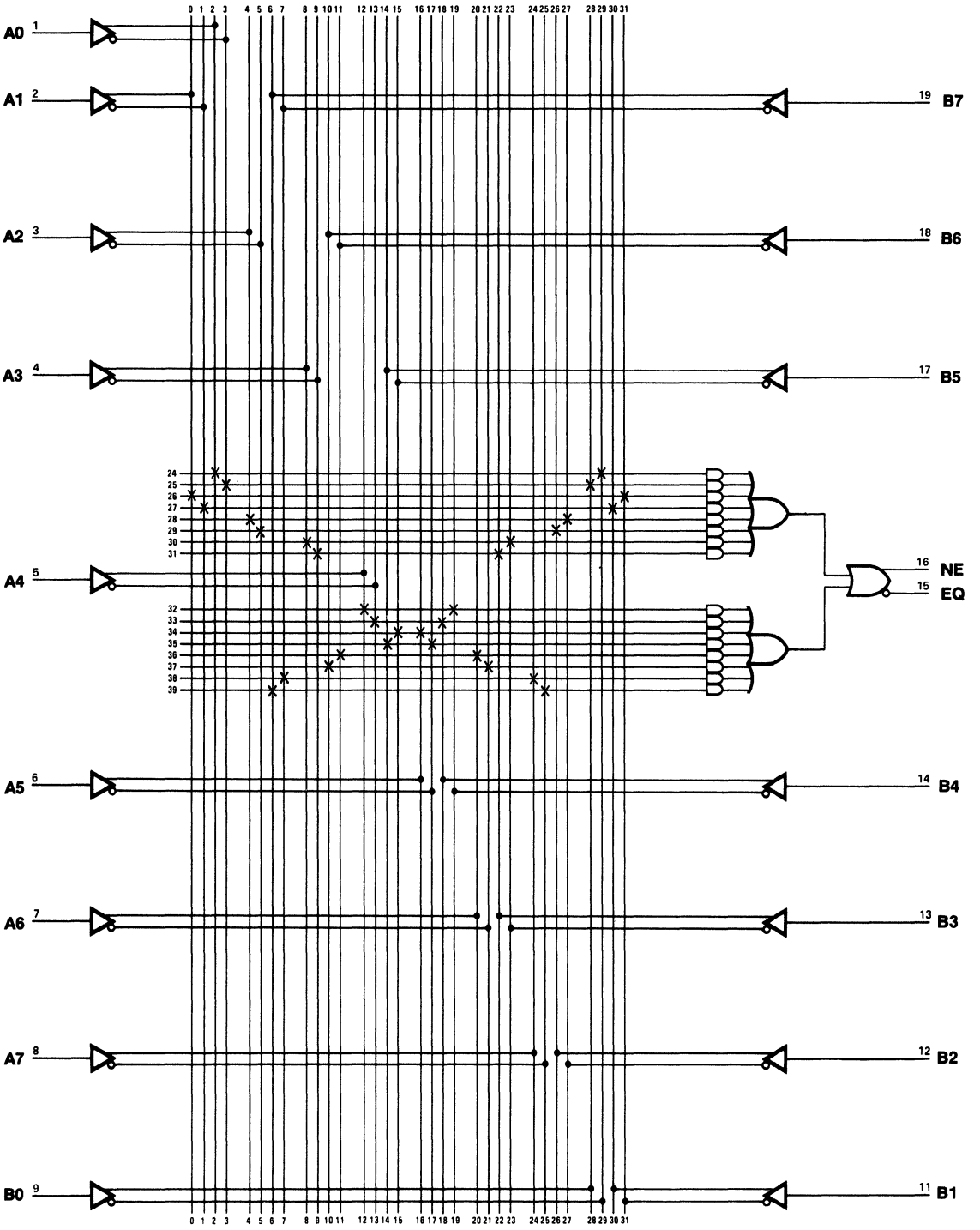




```
PAL16C1                                                PAL DESIGN SPECIFICATION
COMP8                                                    BIRKNER/COLI 07/21/81
OCTAL COMPARATOR
MMI SUNNYVALE,CALIFORNIA
A0 A1 A2 A3 A4 A5 A6 A7 B0 GND
B1 B2 B3 B4 EQ NE B5 B6 B7 VCC


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


FUNCTION TABLE

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

;INPUT  A  INPUT  B  OUTPUTS
;76543210  76543210   NE EQ    COMMENTS
-------------------------------------------------------------
 HLLLLLLL  LLLLLLLL   H  L    A7=H, B7=L
 LHLLLLLL  LLLLLLLL   H  L    A6=H, B6=L
 LLHLLLLL  LLLLLLLL   H  L    A5=H, B5=L
 LLLHLLLL  LLLLLLLL   H  L    A4=H, A5=L
 LLLLHLLL  LLLLLLLL   H  L    A3=H, B3=L
 LLLLLHLL  LLLLLLLL   H  L    A2=H, B2=L
 LLLLLLHL  LLLLLLLL   H  L    A1=H, B1=L
 LLLLLLLH  LLLLLLLL   H  L    A0=H, B0=L
 LLLLLLLL  HLLLLLLL   H  L    A7=L, B7=H
 LLLLLLLL  LHLLLLLL   H  L    A6=L, B6=H
 LLLLLLLL  LLHLLLLL   H  L    A5=L, B5=H
 LLLLLLLL  LLLHLLLL   H  L    A4=L, B4=H
 LLLLLLLL  LLLLHLLL   H  L    A3=L, B3=H
 LLLLLLLL  LLLLLHLL   H  L    A2=L, B2=H
 LLLLLLLL  LLLLLLHL   H  L    A1=L, B1=H
 LLLLLLLL  LLLLLLLH   H  L    A0=L, B0=H
 LLLLLLLL  LLLLLLLL   L  H    TEST ALL L'S
 HHHHHHHH  HHHHHHHH   L  H    TEST ALL H'S
 HLHLHLHL  HLHLHLHL   L  H    TEST EVEN CHECKERBOARD
 LHLHLHLH  LHLHLHLH   L  H    TEST ODD  CHECKERBOARD
 HHLLHHLL  HHLLHHLL   L  H    TEST EVEN DOUBLE CHECKERBOARD
 LLHHLLHH  LLHHLLHH   L  H    TEST ODD  DOUBLE CHECKERBOARD
-------------------------------------------------------------


DESCRIPTION

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

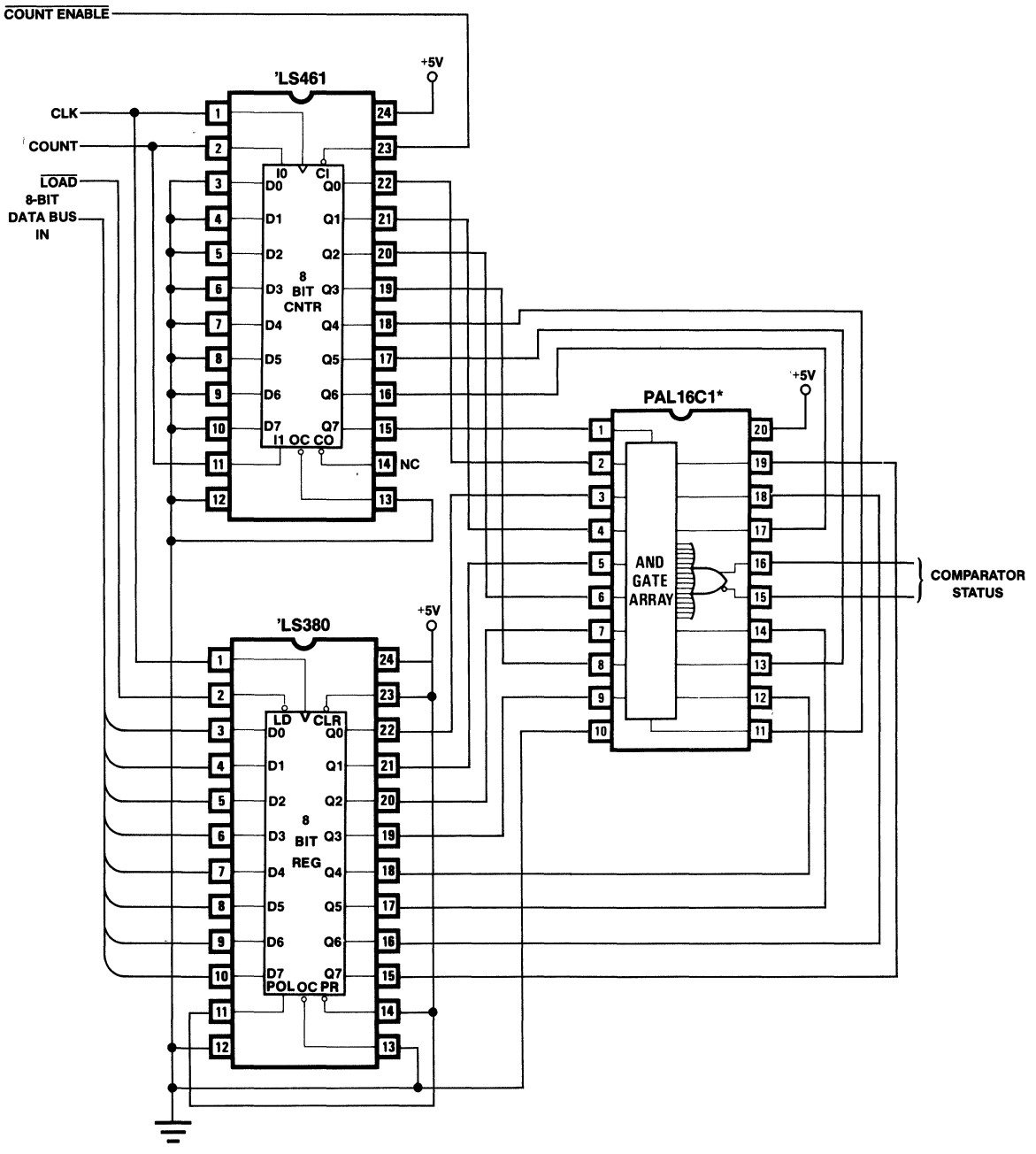
Octal Comparator

Logic Diagram PAL16C1



Application

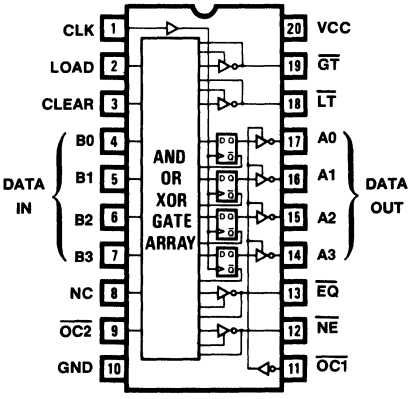
Octal Counter/Register Comparator



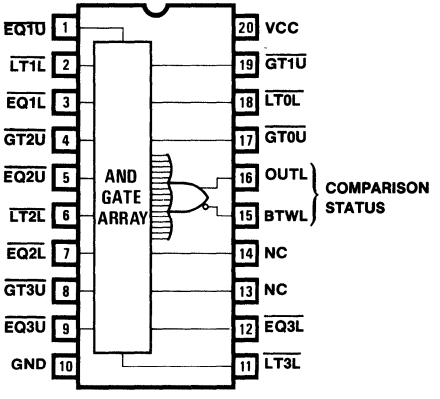
*Note different pinout.

Between Limits Comparator

PAL16X4

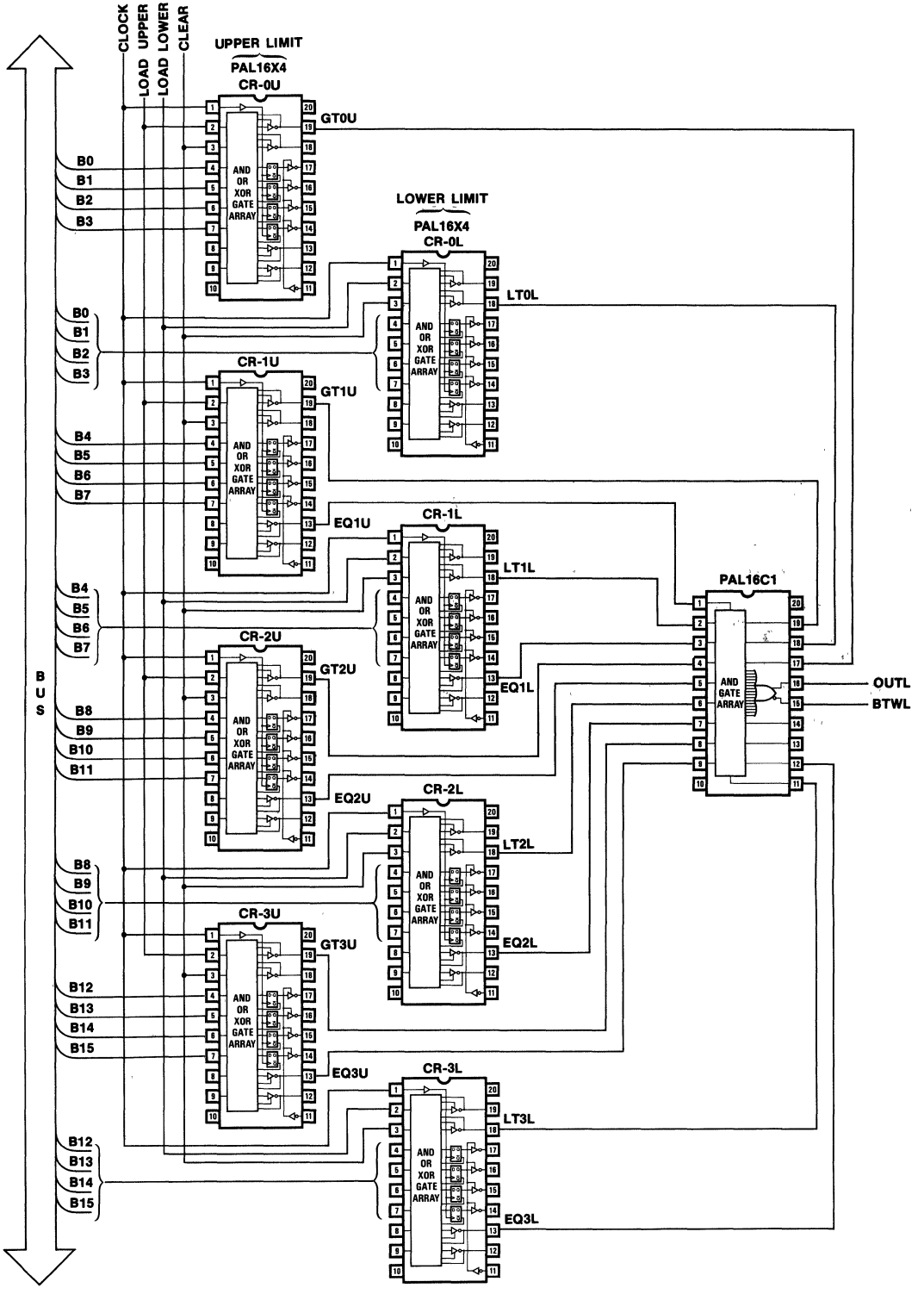


PAL16C1

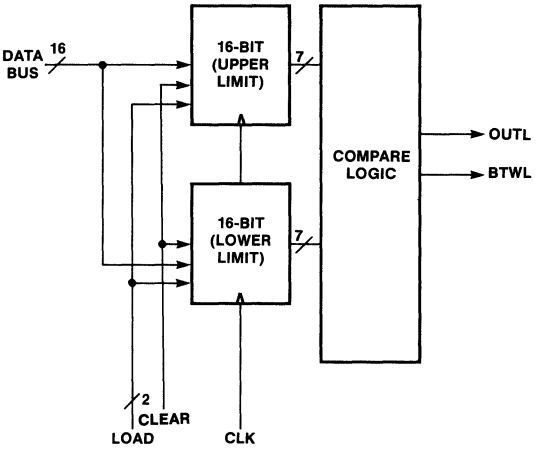


6

Between Limits Comparator — Logic Schematic



```
PAL16X4                                                PAL DESIGN SPECIFICATION
BLR                                                       BIRKNER/COLI 07/12/81
BETWEEN LIMITS COMPARATOR/REGISTER
MMI SUNNYVALE, CALIFORNIA
 CLK LOAD CLEAR B0 B1 B2 B3  NC /OC2 GND
/OC1 /NE   /EQ  A3 A2 A1 A0 /LT /GT  VCC


IF(OC2)  LT = (A3*/B3)                                          ;B3=L, A3=H
           + (A3:*:B3) * (A2*/B2)                               ;B2=L, A2=H
           + (A3:*:B3) * (A2:*:B2) * (A1*/B1)                   ;B1=L, A1=H
           + (A3:*:B3) * (A2:*:B2) * (A1:*:B1) * (A0*/B0)       ;B0=L, A0=H

IF(OC2)  GT = (/A3*B3)                                          ;B3=H, A3=L
           + (A3:*:B3) * (/A2*B2)                               ;B2=H, A2=L
           + (A3:*:B3) * (A2:*:B2) * (/A1*B1)                   ;B1=H, A1=L
           + (A3:*:B3) * (A2:*:B2) * (A1:*:B1) * (/A0*B0)       ;B0=H, A0=L

IF(OC2)  EQ = (A3:*:B3) * (A2:*:B2) * (A1:*:B1) * (A0:*:B0)     ;COMPARE EQUAL

IF(OC2)  NE = (A3:+:B3) + (A2:+:B2) + (A1:+:B1) + (A0:+:B0)     ;COMPARE NOT EQUAL

      /A3 := (/A3)*/LOAD      ;HOLD  REG A3
           + (/B3)* LOAD      ;LOAD  REG A3
           +  CLEAR           ;CLEAR REG A3

      /A2 := (/A2)*/LOAD      ;HOLD  REG A2
           + (/B2)* LOAD      ;LOAD  REG A2
           +  CLEAR           ;CLEAR REG A2

      /A1 := (/A1)*/LOAD      ;HOLD  REG A1
           + (/B1)* LOAD      ;LOAD  REG A1
           +  CLEAR           ;CLEAR REG A1

      /A0 := (/A0)*/LOAD      ;HOLD  REG A0
           + (/B0)* LOAD      ;LOAD  REG A0
           +  CLEAR           ;CLEAR REG A0
```



FUNCTION TABLE

CLK /OC1 /OC2 LOAD CLEAR B3 B2 B1 B0 A3 A2 A1 A0 LT EQ NE GT

;CONTROL ; /OC ;CLK 1 2	OPERATIONS LOAD CLEAR	BUS BBBB 3210	REG AAAA 3210	--STATUS--- LT EQ NE GT	COMMENTS (HEX VALUES)
C L X	X H	XXXX	LLLL	X X X X	CLEAR REG
C L X	H L	LLLL	LLLL	X X X X	LOAD REG (0)
X L L	L L	LLLL	LLLL	L H L L	COMPARE (0 EQ 0)
X L L	L L	LLLH	LLLL	L L H H	COMPARE (1 GT 0)
X L X	L L	XXXX	LLLL	X X X X	READ REG (0)
C L X	X H	XXXX	LLLL	X X X X	CLEAR REG
C L X	H L	LHLH	LHLH	X X X X	LOAD REG (5)
X L L	L L	LLLL	LHLH	H L H L	COMPARE (0 LT 5)
X L L	L L	LHLH	LHLH	L H L L	COMPARE (5 EQ 5)
X L L	L L	HHHH	LHLH	L L H H	COMPARE (F GT 5)
X L X	L L	XXXX	LHLH	X X X X	READ REG (5)
C L X	X H	XXXX	LLLL	X X X X	CLEAR REG
C L X	H L	HLHL	HLHL	X X X X	LOAD REG (A)
X L L	L L	LHLL	HLHL	H L H L	COMPARE (4 LT A)
X L L	L L	HLHL	HLHL	L H L L	COMPARE (A EQ A)
X L L	L L	HLHH	HLHL	L L H H	COMPARE (B GT A)
X L X	L L	XXXX	HLHL	X X X X	READ REG (A)
C L X	X H	XXXX	LLLL	X X X X	CLEAR REG
C L X	H L	HHHH	HHHH	X X X X	LOAD REG (F)
X L L	L L	HHHL	HHHH	H L H L	COMPARE (E LT F)
X L L	L L	HHHH	HHHH	L H L L	COMPARE (F EQ F)
X L X	L L	XXXX	HHHH	X X X X	READ REG (F)
C L X	L L	XXXX	HHHH	X X X X	HOLD (F)
X H X	X X	XXXX	ZZZZ	X X X X	TEST HI-Z (/OC1=H)
X X H	X X	XXXX	XXXX	Z Z Z Z	TEST HI-Z (/OC2=H)

DESCRIPTION

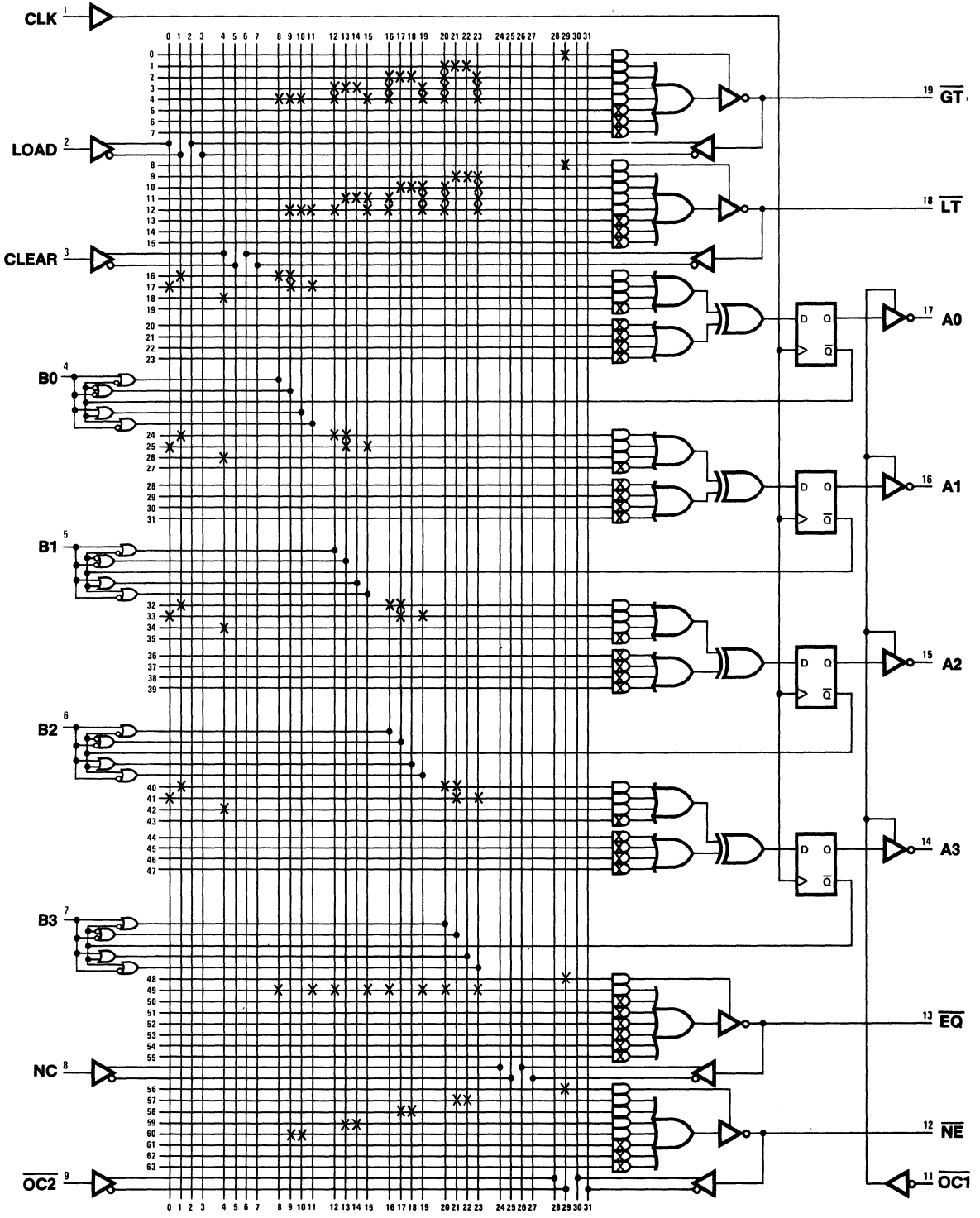
THE DEVICE CONTINUOUSLY COMPARES THE VALUE OF BUS (B3-B0) WITH THE VALUE OF THE REGISTER (A3-A0) AND REPORTS THE STATUS ON OUTPUTS LT, EQ, NE, AND GT:

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

/OC1	/OC2	CLK	LOAD	CLEAR	STATUS LT EQ NE GT	BUS B3-B0	REG A3-A0	OPERATION
H	X	X	X	X	X X X X	X	Z	REG HI-Z
X	H	X	X	X	Z Z Z Z	X	X	STATUS HI-Z
L	X	X	L	L	X X X X	X	A	READ REG
X	X	C	H	L	X X X X	B	B	LOAD REG
X	X	C	L	L	X X X X	X	A	HOLD
X	X	C	X	H	X X X X	X	L	CLEAR REG
X	L	X	L	L	STATUS	B	A	COMPARE

Between Limits Comparator/Register

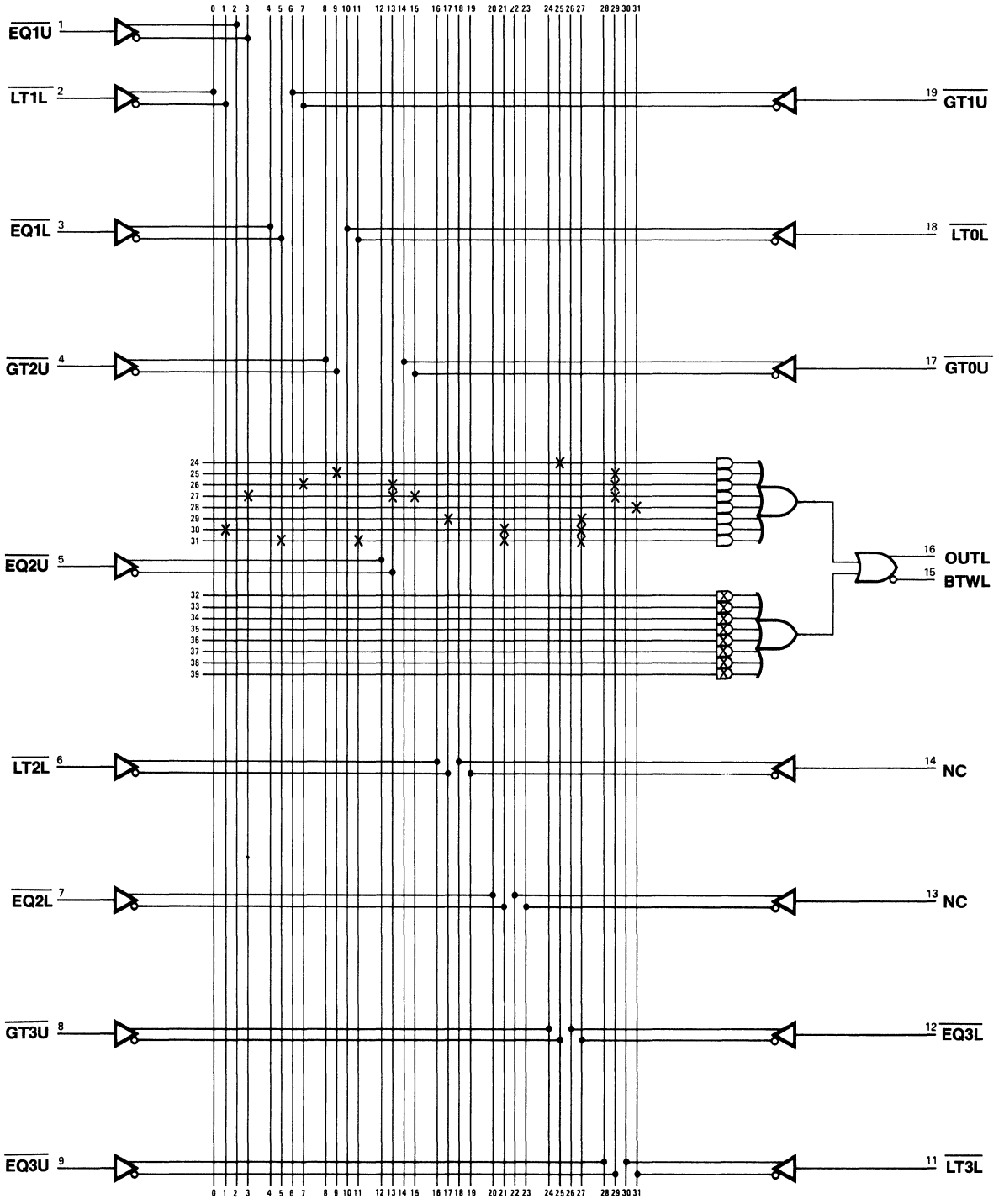
Logic Diagram PAL16X4



```
PAL16C1                                              PAL DESIGN SPECIFICATION
BLL                                                    BIRKNER/COLI 07/12/81
BETWEEN LIMITS COMPARATOR/LOGIC
MMI SUNNYVALE, CALIFORNIA
/EQ1U /LT1L /EQ1L /GT2U /EQ2U /LT2L /EQ2L /GT3U /EQ3U GND
/LT3L /EQ3L   NC    NC   BTWL  OUTL /GT0U /LT0L /GT1U VCC


OUTL =  GT3U                          ;GT CR-3U
     +  EQ3U * GT2U                   ;EQ CR-3U, GT CR-2U
     +  EQ3U * EQ2U * GT1U            ;EQ CR-3U, EQ CR-2U, GT CR-1U
     +  EQ3U * EQ2U * EQ1U * GT0U     ;EQ CR-3U, EQ CR-2U, EQ CR-1U, GT CR-0U
     +  LT3L                          ;LT CR-3L
     +  EQ3L * LT2L                   ;EQ CR-3L, LT CR-2L
     +  EQ3L * EQ2L * LT1L            ;EQ CR-3L, EQ CR-2L, LT CR-1L
     +  EQ3L * EQ2L * EQ1L * LT0L     ;EQ CR-3L, EQ CR-2L, EQ CR-1L, LT CR-0L


FUNCTION TABLE

GT3U EQ3U EQ3L LT3L GT2U EQ2U EQ2L LT2L GT1U EQ1U EQ1L LT1L GT0U LT0L OUTL BTWL

; COMPARATOR REGISTERS
;CR-3  CR-2  CR-1  CR-0
;GEEL  GEEL  GEEL  G  L
;TQQT  TQQT  TQQT  T  T     OUTPUTS
;UULL  UULL  UULL  U  L    OUTL  BTWL     COMMENTS
-------------------------------------------------------------------------
 HLLL  XXXX  XXXX  X  X      H     L       OUT OF LIMITS (GT3U=H)
 LHLL  HLLL  XXXX  X  X      H     L       OUT OF LIMITS (GT2U=H)
 LHLL  LHLL  HLLL  X  X      H     L       OUT OF LIMITS (GT1U=H)
 LHLL  LHLL  LHLL  H  L      H     L       OUT OF LIMITS (GT0U=H)
 LHLL  LHLL  LHLL  L  H      L     H       BETWEEN LIMITS
 LHLL  LHLL  LLHL  X  X      L     H       BETWEEN LIMITS
 LHLL  LHLL  LLLH  X  X      L     H       BETWEEN LIMITS
 LHLL  LLHL  XXXX  X  X      L     H       BETWEEN LIMITS
 LHLL  LLLH  XXXX  X  X      L     H       BETWEEN LIMITS
 LLHL  HLLL  XXXX  X  X      L     H       BETWEEN LIMITS
 LLHL  LHLL  XXXX  X  X      L     H       BETWEEN LIMITS
 LLHL  LLHL  HLLL  X  X      L     H       BETWEEN LIMITS
 LLHL  LLHL  LHLL  X  X      L     H       BETWEEN LIMITS
 LLHL  LLHL  LLHL  H  L      L     H       BETWEEN LIMITS
 LLHL  LLHL  LLHL  L  H      H     L       OUT OF LIMITS (LT0L=H)
 LLHL  LLHL  LLLH  X  X      H     L       OUT OF LIMITS (LT1L=H)
 LLHL  LLLH  XXXX  X  X      H     L       OUT OF LIMITS (LT2L=H)
 LLLH  XXXX  XXXX  X  X      H     L       OUT OF LIMITS (LT3L=H)
-------------------------------------------------------------------------


DESCRIPTION

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

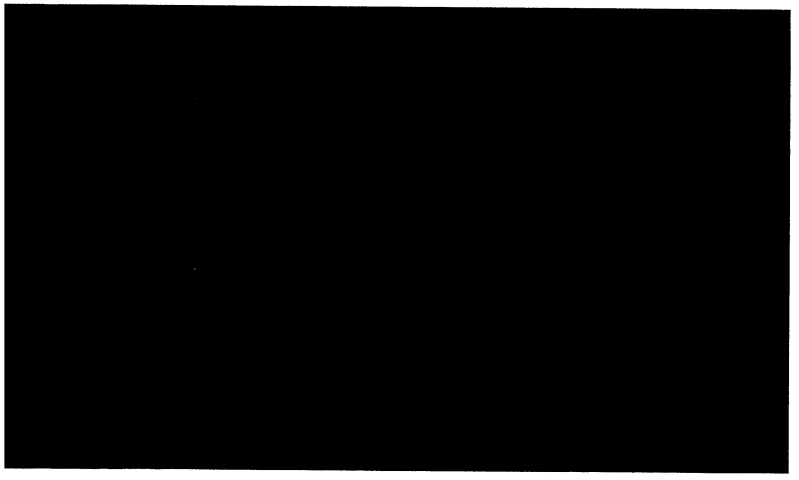
Between Limits Comparator/Logic

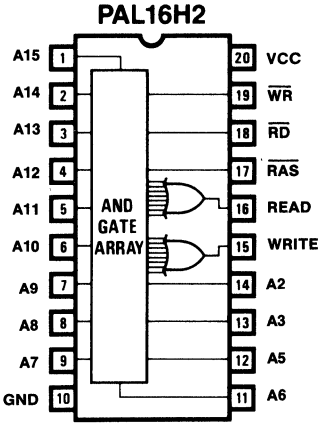
Logic Diagram PAL16C1

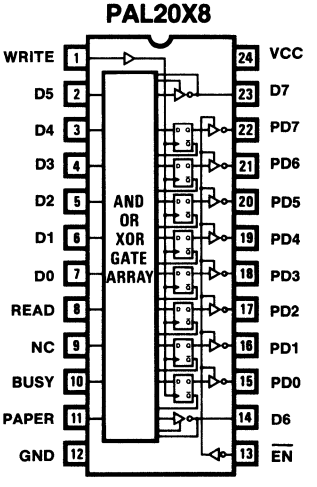



IMPORTANT
NOTES

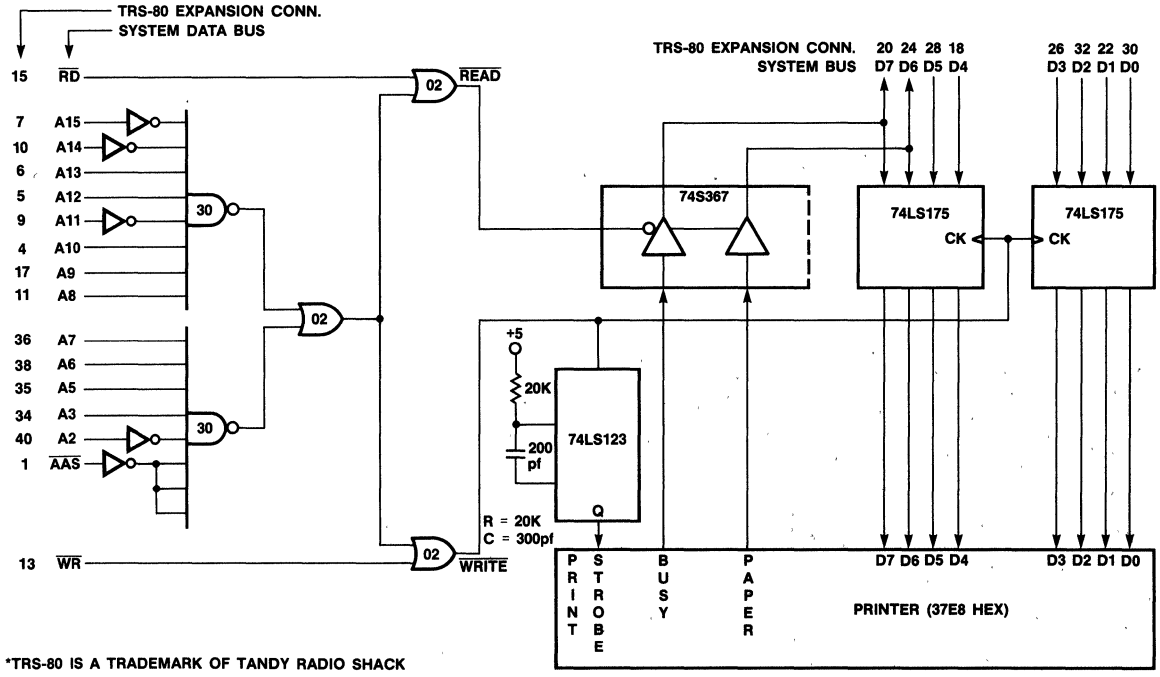
Memory Mapped Printer



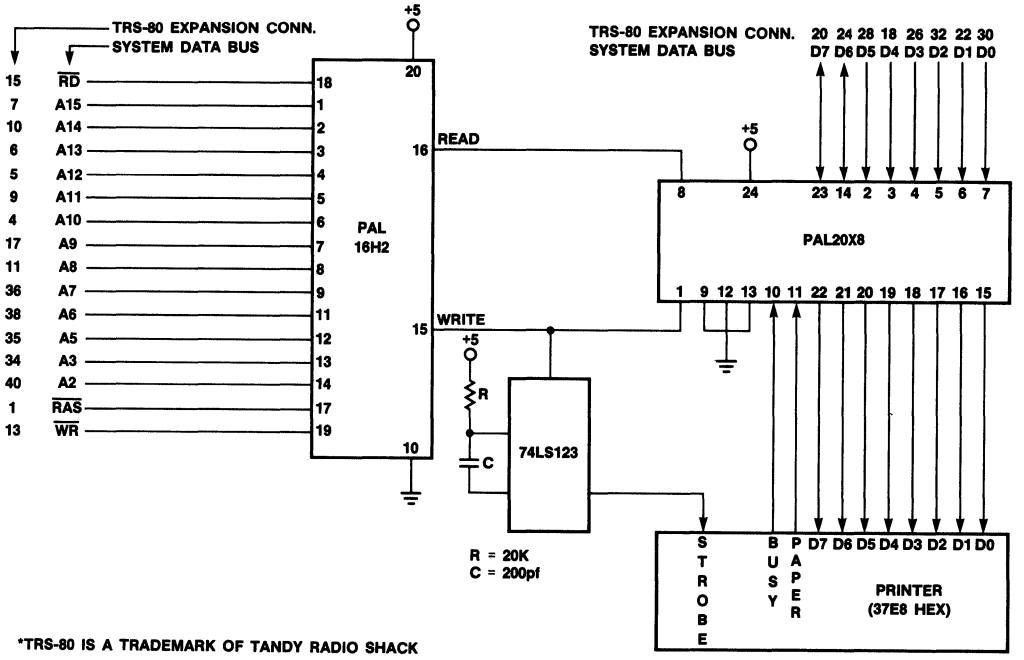




6



*TRS-80 IS A TRADEMARK OF TANDY RADIO SHACK



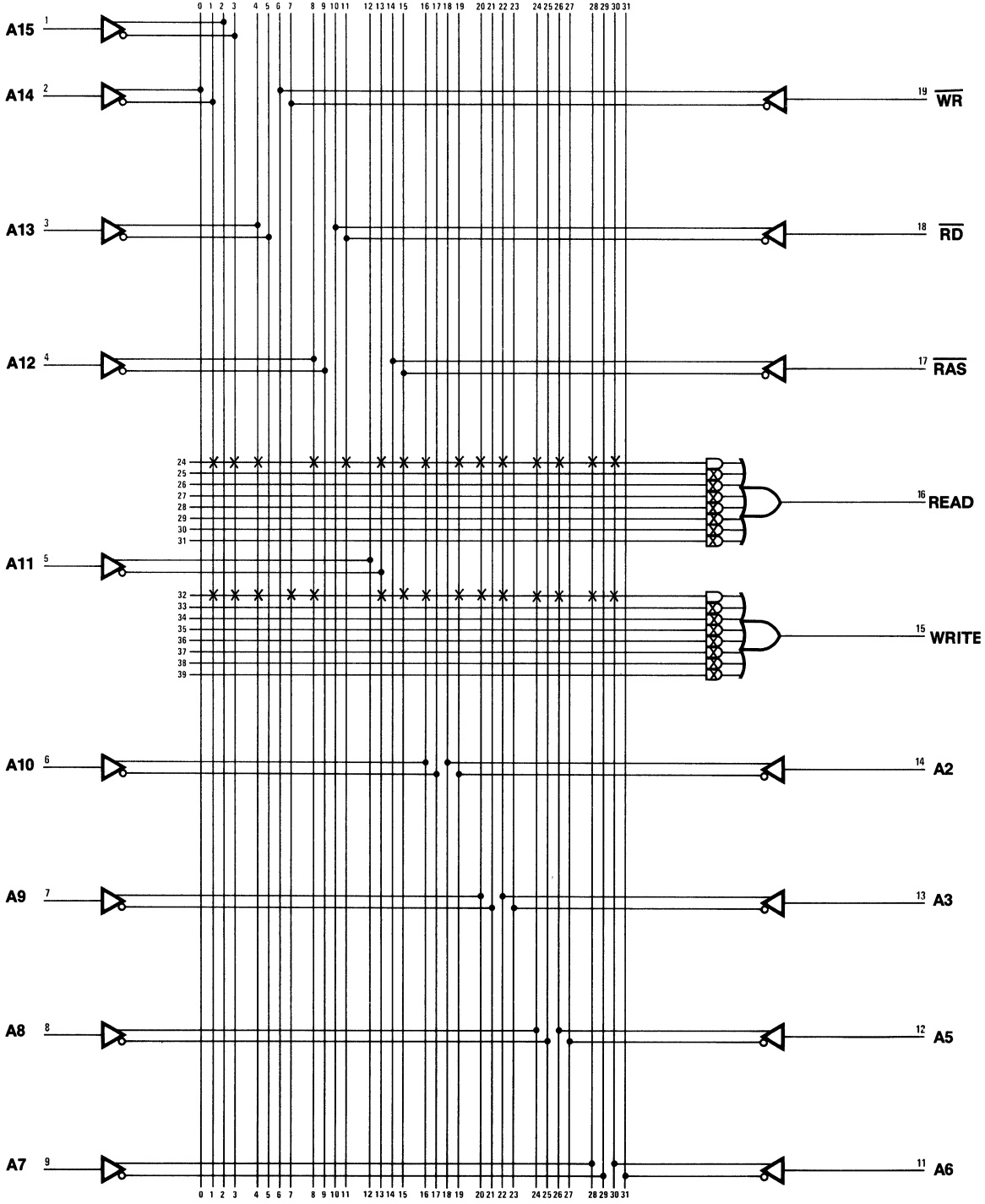
*TRS-80 IS A TRADEMARK OF TANDY RADIO SHACK

6

```
PAL16H2                                          PAL DESIGN SPECIFICATION
MMPD                                             DICK JONES      07/07/81
MEMORY MAPPED PRINTER DECODER
MMI FIELD APPLICATIONS ENGINEER LOMBARD, ILLINOIS
A15 A14 A13 A12 A11 A10 A9 A8 A7 GND
A6 A5 A3 A2 WRITE READ /RAS /RD /WR VCC

WRITE = /A15*/A14*A13*A12*/A11*A10*A9*A8*A7*A6*A5*A3*/A2*RAS*WR

 READ = /A15*/A14*A13*A12*/A11*A10*A9*A8*A7*A6*A5*A3*/A2*RAS*RD


FUNCTION TABLE

A15 A14 A13 A12 A11 A10 A9 A8 A7 A6 A5 A3 A2 RAS RD WR WRITE READ

;AAAA AAAA AAA AA RRW    WR
;1111 1198 765 32 ADR    RE
;5432 10          S      IA
;                        TD
;                        E       COMMENTS
-------------------------------------------------------
 LLHH LHHH HHH HL HLH    HL      WRITE WHEN 37E8 HEX
 LLHH LHHH HHH HL HHL    LH      READ WHEN 37E8 HEX
 XXXX XXXX XXX XX LXX    LL      RAS NOT PRESENT
 XXXX XXXX XXX XX HLL    LL      NOT READ OR WRITE
-------------------------------------------------------


DESCRIPTION

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

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

Memory Mapped Printer Decoder

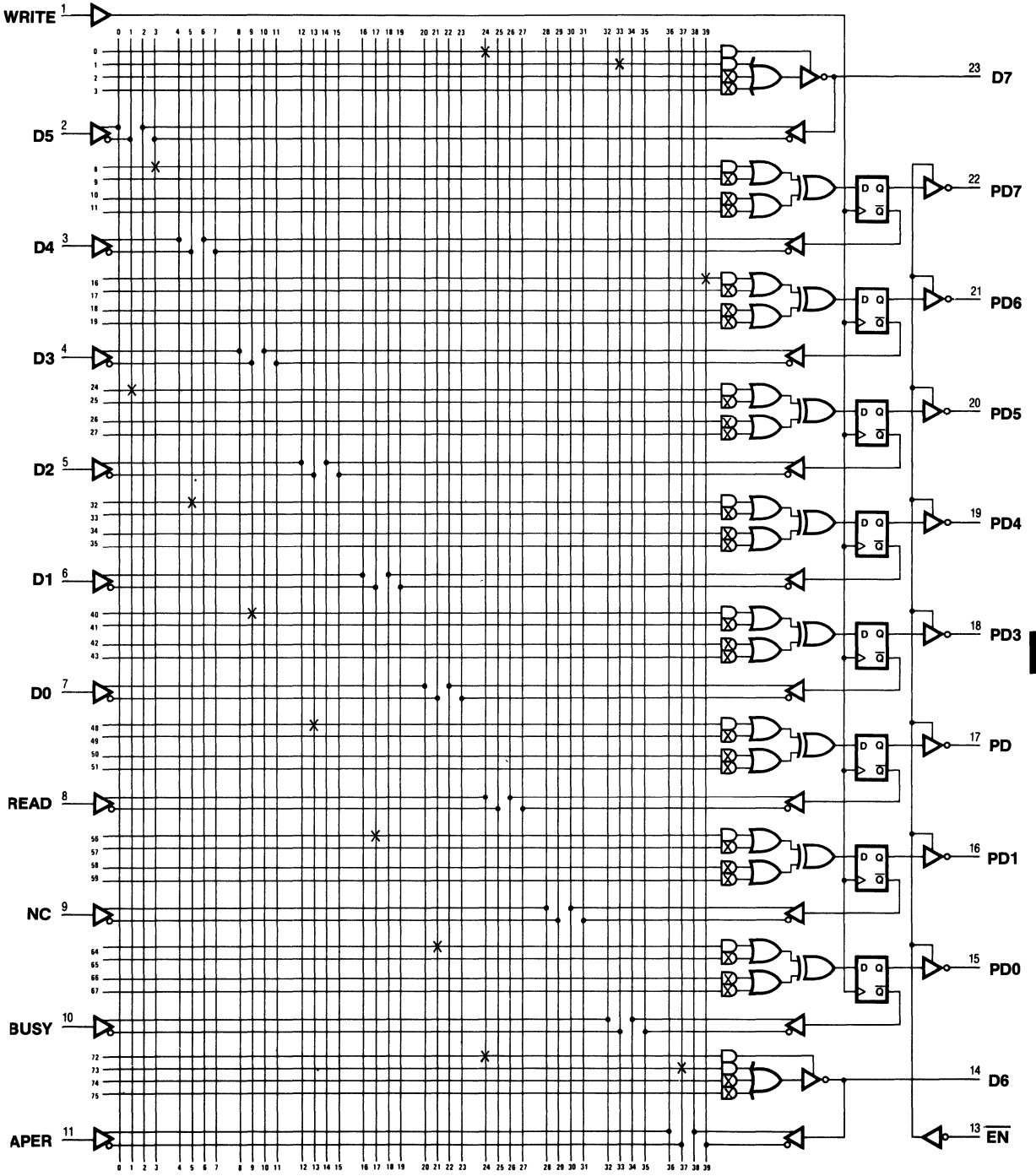
Logic Diagram PAL16H2

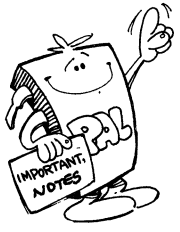


```
PAL20X8                                    PAL DESIGN SPECIFICATION
PDRM                                       DICK JONES      07/07/81
PRINTER DATA REGISTER/MUX
MMI FIELD APPLICATIONS ENGINEER LOMBARD, ILLINOIS
WRITE D5 D4 D3 D2 D1 D0 READ NC BUSY PAPER GND
/EN D6 PD0 PD1 PD2 PD3 PD4 PD5 PD6 PD7 D7 VCC


IF (READ) /D7  = /BUSY       ;READ PRINTER STATUS - BUSY

         /PD7 := /D7         ;LOAD MSB TO PRINTER

         /PD6 := /D6         ;LOAD PRINTER

         /PD5 := /D5         ;LOAD PRINTER

         /PD4 := /D4         ;LOAD PRINTER

         /PD3 := /D3         ;LOAD PRINTER

         /PD2 := /D2         ;LOAD PRINTER

         /PD1 := /D1         ;LOAD PRINTER

         /PD0 := /D0         ;LOAD LSB TO PRINTER

IF (READ) /D6  = /PAPER      ;READ PRINTER STATUS - PAPER OUT

FUNCTION TABLE

WRITE BUSY PAPER READ D7 D6 D5 D4 D3 D2 D1 D0 PD7 PD6 PD5 PD4 PD3 PD2 PD1 PD0

;WBPR DDDDDDDD PPPPPPPP
;RUAE 76543210 DDDDDDDD
;ISPA          76543210
;TYED
;E R                          COMMENTS
-------------------------------------------------
 LHLH HLXXXXXX XXXXXXXX       READ PRNTR STATUS
 LLHH LHXXXXXX XXXXXXXX       READ PRNTR STATUS
 CXXL ZZXXXXXX XXXXXXXX       TEST HI-Z
 CXXL LLLLLLLL LLLLLLLL       LOAD DATA TO PRINTER
 CXXL HLHLHLHL HLHLHLHL       LOAD DATA TO PRINTER
 CXXL HHHHHHHH HHHHHHHH       LOAD DATA TO PRINTER
-------------------------------------------------

 DESCRIPTION

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

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

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

Printer Data Register/Mux

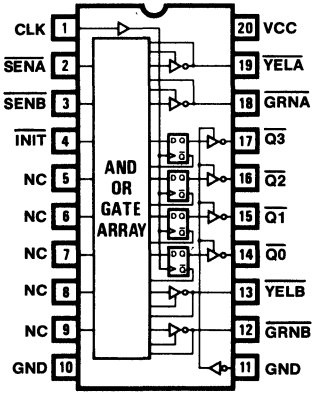
Logic Diagram PAL20X8



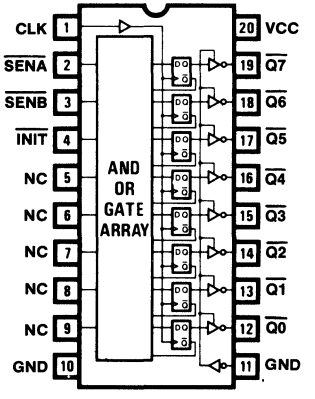

IMPORTANT
NOTES

Traffic Signal Controller

PAL16R4



PAL16R8



6

An Example of Sequential Logic Design with PALs

Figure 1 illustrates a simple traffic intersection consisting of two one-way streets, direction A and direction B. Each direction has a signal consisting of red, yellow and green lamps which are activated with appropriately named active low signals. Also, each direction has a sensor which provides an active low signal indicating the presence of an oncoming vehicle. Our controller is to manage this intersection with the sensors as inputs and the lamps as outputs, as shown in Figure 2.

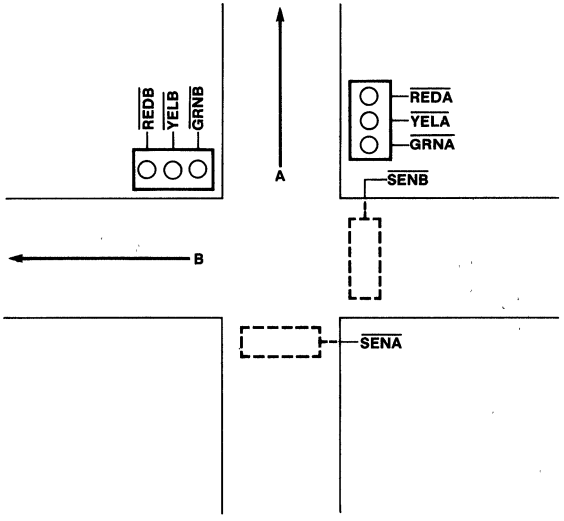


Figure 1. Traffic Intersection

Figure 2 also includes the system clock and an initialize (or reset) signal, which drives the controller to a pre-defined initial state. This raises two important issues in designing sequential logic with PALs. First, all circuit implementations of sequential logic with PALs are totally synchronous. This implies that all state variables (flip-flops) change at the same time, precisely after the rising edge of the clock. Second, PAL sequential logic designs should include a means for initialization to implement test programs and ensure reliable circuit operation.

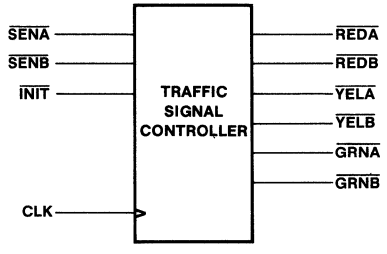


Figure 2. Traffic Signal Controller

The specifies of the controller operations are detailed with a state graph, shown in Figure 3.

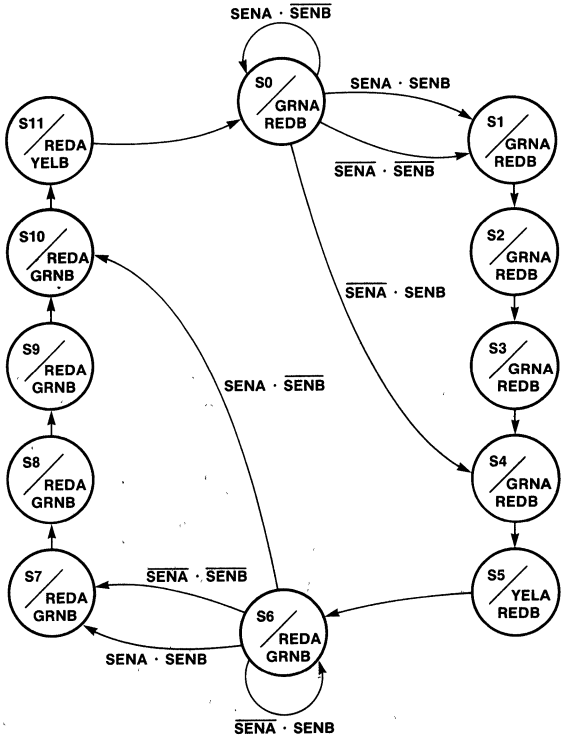


Figure 3. State Graph — Traffic Signal Controller

In this format, each bubble represents a stable state, i.e., an output configuration, lasting at least one clock cycle. Inside the bubble is the name of the state (S0-S11) and the outputs associated with that state. This particular circuit's outputs are specified to be a function of the state of the flip-flops only, and are not directly affected by the inputs. For the sake of simplicity in the state graph, the transitions involving INIT are omitted; INIT simply drives the circuit to S0 from any state, regardless of other inputs.

Another method of expressing the state graph information is with a state table, shown in Figure 4. Each row in the state table corresponds to a state (or bubble from the state graph). The first four columns give the next state for each of the possible input combinations. The output is also given for each state in the table. The state graph may be omitted but a state table is generally required to design the circuit.

The next step in the design is to assign state variables. This process also involves selecting the PAL or PALs to be used. Referring to Figures 2 and 3, the circuit requirements are seen to be 4 input, 6 output and 12 internal states. Since the intput/output pin requirements do not impose restrictions in either a 20 or 24 pin package, the states requirement will be addressed first. A PAL16R4 can implement the 12 states with its 4 flip flops. However, only 4 combinatorial outputs are available. This means that the flip-flop outputs (or state variables) will need to

be utilized as circuit outputs as well. One such approach is to take advantage of the fact that REDA = $\overline{\text{REDB}}$ and implementing REDA with one flip-flop, and REDB with an external inverter. Such a state variable assignment and resulting transition table is in Figure 5. This is generated by substituting the variable assignments into the state table.

CURRENT STATE \ NEXT STATE	INPUT SENA, SENB 00	01	10	11	OUTPUT REDA	YELA	GRNA	REDB	YELB	GRNB
S0	S1	S4	S0	S1	0	0	1	1	0	0
S1	S2	S2	S2	S2	0	0	1	1	0	0
S2	S3	S3	S3	S3	0	0	1	1	0	0
S3	S4	S4	S4	S4	0	0	1	1	0	0
S4	S5	S5	S5	S5	0	0	1	1	0	0
S5	S6	S6	S6	S6	0	1	0	1	0	0
S6	S7	S6	S10	S7	1	0	0	0	0	1
S7	S8	S8	S8	S8	1	0	0	0	0	1
S8	S9	S9	S9	S9	1	0	0	0	0	1
S9	S10	S10	S10	S10	1	0	0	0	0	1
S10	S11	S11	S11	S11	1	0	0	0	0	1
S11	S0	S0	S0	S0	1	0	0	0	1	0

Figure 4. State Table — Traffic Signal Controller

CURRENT STATE	INPUT/NEXT STATE 00	01	10	11	OUTPUTS
0 0 0 0	0 0 0 1	0 1 0 0	0 0 0 0	0 0 0 1	0 0 1 1 0 0
0 0 0 1	0 0 1 0	0 0 1 0	0 0 1 0	0 0 1 0	0 0 1 1 0 0
0 0 1 0	0 0 1 1	0 0 1 1	0 0 1 1	0 0 1 1	0 0 1 1 0 0
0 0 1 1	0 1 0 0	0 1 0 0	0 1 0 0	0 1 0 0	0 0 1 1 0 0
0 1 0 0	0 1 0 1	0 1 0 1	0 1 0 1	0 1 0 1	0 0 1 1 0 0
0 1 0 1	1 0 0 0	1 0 0 0	1 0 0 0	1 0 0 0	0 1 0 1 0 0
1 0 0 0	1 0 0 1	1 0 0 0	1 1 0 0	1 0 0 1	1 0 0 0 0 1
1 0 0 1	1 0 1 0	1 0 1 0	1 0 1 0	1 0 1 0	1 0 0 0 0 1
1 0 1 0	1 0 1 1	1 0 1 1	1 0 1 1	1 0 1 1	1 0 0 0 0 1
1 0 1 1	1 1 0 0	1 1 0 0	1 1 0 0	1 1 0 0	1 0 0 0 0 1
1 1 0 0	1 1 0 1	1 1 0 1	1 1 0 1	1 1 0 1	1 0 0 0 0 1
1 1 0 1	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	1 0 0 0 1 0

STATE	Q3 Q2 Q1 Q0
S0	0 0 0 0
S1	0 0 0 1
S2	0 0 1 0
S3	0 0 1 1
S4	0 1 0 0
S5	0 1 0 1
S6	1 0 0 0
S7	1 0 0 1
S8	1 0 1 0
S9	1 0 1 1
S10	1 1 0 0
S11	1 1 0 1

Input = SENA, SENB
Current/Next State = Q3, Q2, Q1, Q0/$Q3^+$, $Q2^+$, $Q1^+$, $Q0^+$
Output = REDA, YELA, GRNA, REDB, YELB, GRNB

Figure 5. State Assignment No. 1/Transition Table

From this table, Karnaugh maps for D flip flop next state equations and output functions can be written, by transcription from the transition table, and minimized equations derived. This is shown in Figures 6 through 10. The state machine and PAL implementation follows. Note that the INIT term is simply AND'ed with all next state products to generate the initialize function.

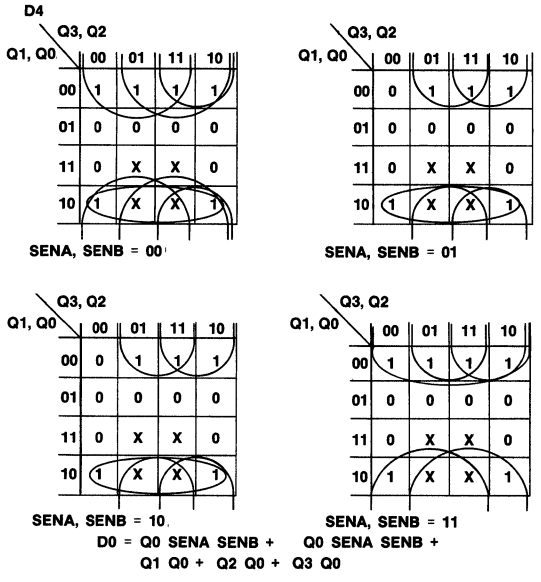


Figure 6. K Maps for Q0+

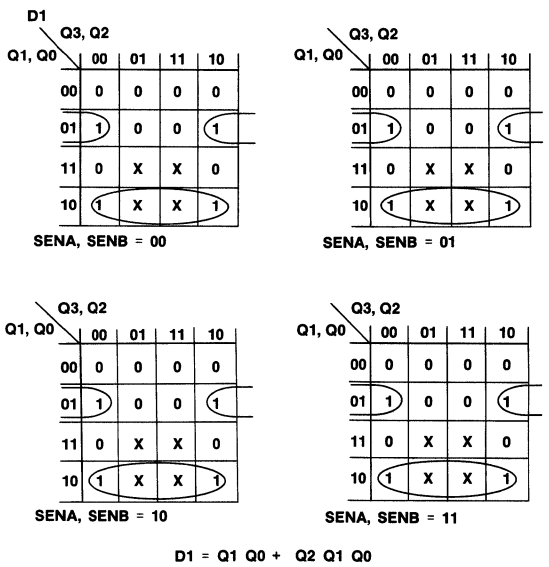


Figure 7. K Maps for Q1+

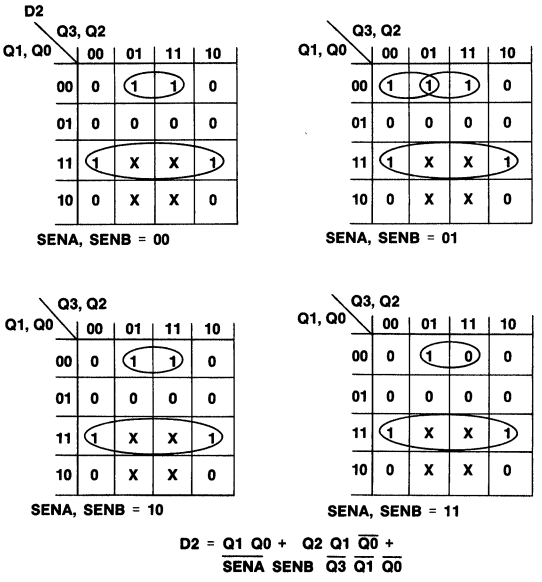


Figure 8. K Maps for Q2+

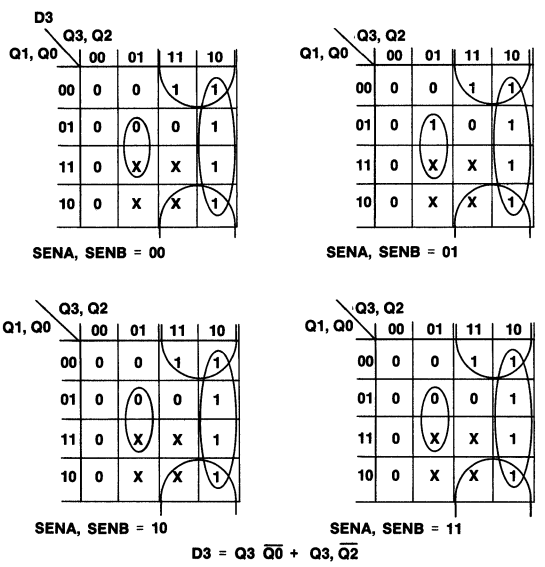


Figure 9. K Maps for Q3+

REDA

Q1, Q0 \ Q3, Q2	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	0	X	X	1
10	0	X	X	1

REDA = Q3

YELA

Q1, Q0 \ Q3, Q2	00	01	11	10
00	0	0	0	0
01	0	1	0	0
11	0	X	X	0
10	0	X	X	0

YELA = $\overline{Q3}$ Q2 Q0

GRNA

Q1, Q0 \ Q3, Q2	00	01	11	10
00	1	1	0	0
01	1	0	0	0
11	1	X	X	0
10	1	X	X	0

GRNA = $\overline{Q3}$ $\overline{Q2}$ + $\overline{Q3}$ $\overline{Q1}$ $\overline{Q0}$

REDB

Q1, Q0 \ Q3, Q2	00	01	11	10
00	1	1	0	0
01	1	1	0	0
11	1	X	X	0
10	1	X	X	0

REDB = $\overline{Q3}$

YELB

Q1, Q0 \ Q3, Q2	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	0	X	X	0
10	0	X	X	0

YELB = Q3 Q2 $\overline{Q0}$

GRNB

Q1, Q0 \ Q3, Q2	00	01	11	10
00	0	0	1	1
01	0	0	0	1
11	0	X	X	1
10	0	X	X	1

GRNB = Q3 $\overline{Q2}$ + Q3 $\overline{Q1}$ $\overline{Q0}$

Figure 10. K Maps for Outputs

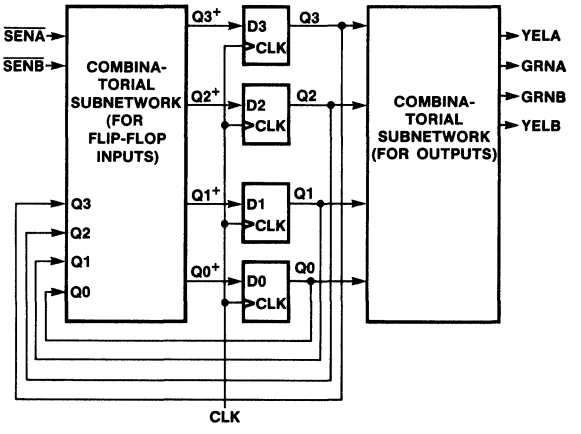


Figure 11. State Machine Traffic Signal Controller No. 1

```
PAL16R4                                                PAL DESIGN SPECIFICATION
TRACNT1                                                B. BRAFMAN 03/14/81
TRAFFIC SIGNAL CONTROLLER No. 1
MMI FIELD APPLICATIONS ENGINEER YORBA LINDA, CALIFORNIA
CLK /SENA /SENB /INIT NC NC NC NC NC GND
GND /GRNB /YELB /Q0 /Q1 /Q2 /Q3 /GRNA /YELA VCC


IF (VCC) YELA  =  /Q3 *   Q2 *   Q0                ;"A" DIRECTION YELLOW LITE

IF (VCC) GRNA  =  /Q3 *  /Q2                       ;"A" DIRECTION GREEN
                +  /Q3 *  /Q1 *  /Q0

Q3  :=   Q3   *  /Q0   * /INIT
     +   Q3   *  /Q2   * /INIT
     +  /Q3   *   Q2   *   Q0   * /INIT

Q2  :=   Q1   *   Q0   * /INIT
     +   Q2   *  /Q1   *  /Q0   * /INIT
     +  /Q3   *  /Q1   *  /Q0   * /SENA *  SENB * /INIT

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

Q0  :=  /Q0   *  /SENA *  /SENB * /INIT        ;PRODUCTS ARE FROM K-MAP
     +  /Q0   *   SENA *   SENB * /INIT
     +   Q1   *   /Q0  *  /INIT
     +   Q2   *   /Q0  *  /INIT
     +   Q3   *   /Q0  *  /INIT


IF (VCC) GRNB  =   Q3 *  /Q2                       ;"B" DIRECTION GREEN
                +  Q3 *  /Q1 *  /Q0

IF (VCC) YELB  =   Q3 *   Q2 *   Q0                ;"B" DIRECTION YELLOW
```

FUNCTION TABLE

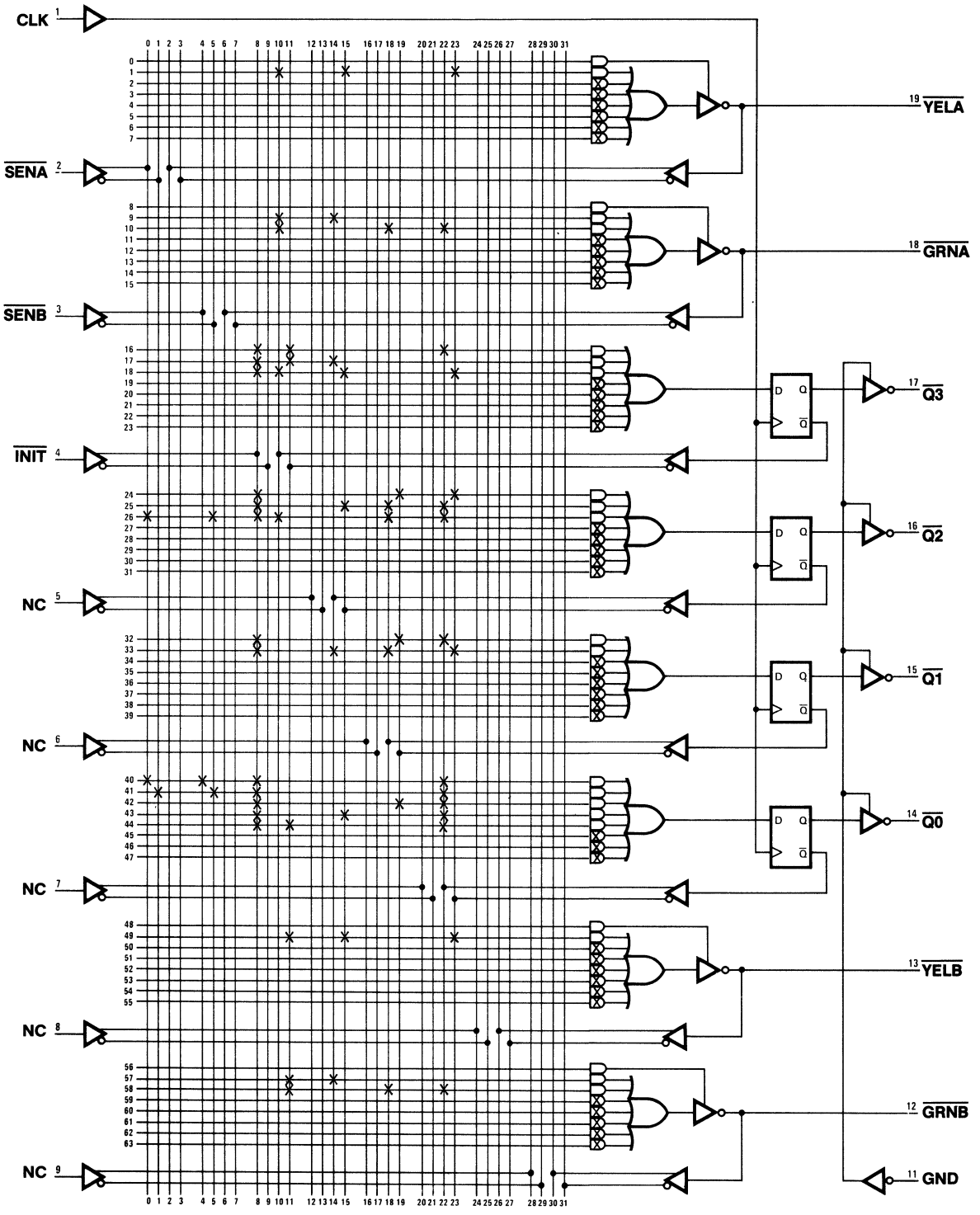
/INIT	CLK	SENA	SENB	Q3	Q2	Q1	Q0	GRNA	YELA	GRNB	YELB	;COMMENTS
L	C	X	X	L	L	L	L	X	X	X	X	S0 (INIT)
H	C	L	L	L	L	L	H	H	L	L	L	S1
H	C	X	X	L	L	H	L	H	L	L	L	S2
H	C	X	X	L	L	H	H	H	L	L	L	S3
H	C	X	X	L	H	L	L	H	L	L	L	S4
H	C	X	X	L	H	L	H	L	H	L	L	S5
H	C	X	X	H	L	L	L	L	L	H	L	S6
H	C	L	L	H	L	L	H	L	L	H	L	S7
H	C	X	X	H	L	H	L	L	L	H	L	S8
H	C	X	X	H	L	H	H	L	L	H	L	S9
H	C	X	X	H	H	L	L	L	L	H	L	S10
H	C	X	X	H	H	L	H	L	L	L	H	S11
H	C	X	X	L	L	L	L	H	L	L	L	S0
H	C	L	H	L	H	L	L	H	L	L	L	S4
L	C	X	X	L	L	L	L	X	X	X	X	S0 (INIT)
H	C	H	L	L	L	L	L	H	L	L	L	S0
H	C	H	H	L	L	L	H	H	L	L	L	S1

DESCRIPTION

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

Traffic Signal Controller No. 1

Logic Diagram PAL16R4



Another approach to the circuit implementation is to assign one flip-flop (state variable) per output. When this is done with PALs, no penalty in chip count need be paid. In addition, such an implementation offers higher throughput than one which uses combinatorial outputs as one layer of delay is eliminated. Once again referring to Figure 4, we see that the outputs do not change in 2 sets of 5 states. Thus, a 3-bit counter is needed to count the states in which the outputs are common. This leaves only 5 flip-flops in a PAL16R8, so we will use the REDA = REDB trick once again. Figure 13 shows a new state assignment to re-implement the circuit with this approach.

CURRENT STATE	INPUT/NEXT STATE																											
	00								01								10								11			
	Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0	Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0	Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0	Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0
S0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	1	0	0	0	0	1
S1	0	0	1	0	0	0	0	1	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0
S2	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	1	0	0	1	0	0	0	1	1	0	0	1	0	0	0	1	1
S3	0	0	1	0	0	0	1	1	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0	0
S4	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0
S5	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0
S6	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	1	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	1
S7	1	0	0	0	1	0	0	1	1	0	0	0	1	0	1	0	1	0	0	0	1	0	1	0	1	0	0	0	1	0	1	0
S8	1	0	0	0	1	0	1	0	1	0	0	0	1	0	1	1	1	0	0	0	1	0	1	1	1	0	0	0	1	0	1	1
S9	1	0	0	0	1	0	1	1	1	0	0	0	1	1	0	0	1	0	0	0	1	1	0	0	1	0	0	0	1	1	0	0
S10	1	0	0	0	1	1	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0
S11	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0

Q7 = REDA = $\overline{\text{REDB}}$ Q5 = GRNA Q3 = GRNB Q2, Q1, Q0 = COUNTER

Q6 = YELA Q4 = YELB

Figure 13. State Assignment No. 2/Transition Table

This transition table would translate into 10 variable Karnaugh maps. Instead, we can simply write down the equations for each next state variable and minimize by inspection (especially if the 8 product per output restriction is exceeded). Initially we will use state names rather than state variables.

By inspection, Q7+, Q6+, Q5+, Q4+, Q3+, Q1+, not depend on SENA and SENB:

Q7+ = S5 + S6 + S7 + S8 + S9 + S10

Q6+ = S4

Q5+ = S0 + S1 = S2 + S3 + S11

Q4+ = S10

Q3+ = S5 + S6 + S7 + S8 + S9

Q1+ = S1 + S2 + S7 + S8

Q2+ and Q0+ do depend on SENA and SENB:

Q2+ = /SENA * /SENB * (S3 + S9) [2 products]
+ /SENA * SENB * (S0 + S3 + S9) [3 products]
+ SENA * /SENB * (S3 + S6 + S9) [3 products]
+ SENA * SENB * (S3 + S9) [2 products]

Since Q2+ needs 10 products as written, we minimize by Boolean manipulation:

Q2+ = /SENA * /SENB * (S3 + S9) } /SENA * (S3 + S9)
+ /SENA * SENB * (S3 + S9) }
+ /SENA * SENB + (S0) } S3 + S9
+ SENA * /SENB * (S3 + S9) } SENA * (S3 + S9)
+ SENA * SENB * (S3 + S9) }
+ SENA * SENB * (S6)

Thus,

Q2+ = S3 + S9
+ /SENA * SENB * S0
+ SENA * SENB * S6

Next,

Q0+ = /SENA * /SENB * (S0 + S2 + S6 + S8) [4 products]
+ /SENA * SENB * (S2 + S8) [2 products]
+ SENA * /SENB * (S2 + S8) [2 products]
+ SENA * SENB * (S0 + S2 + S6 + S8) [4 products]

Q0+ = /SENA * /SENB * (S0 + S6)
+ /SENA * /SENB * (S2 + S8) } /SENA * (S2 + S8)
+ /SENA * SENB * (S2 + S8) }
+ SENA * /SENB * (S2 + S8) } SENA * (S2 + S8) } S2 + S8
+ SENA * SENB * (S2 + S8) }
+ SENA * SENB * (S0 + S6)

Q0+ + S2 + S8
+ /SENA * /SENB * (S0 + S6)
+ SENA * SENB * (S0 + S6)

The full equations (note that minimization is optional for less than 8 product sums):

$Q7^+ = /Q7 * Q6 * /Q5 * /Q4 * /Q3 * /Q2 * /Q1 * /Q0$
$\quad + Q7 * /Q6 * /Q5 * /Q4 * Q3 * /Q2 * /Q1 * /Q0$
$\quad + Q7 * /Q6 * /Q5 * /Q4 * Q3 * /Q2 * /Q1 * Q0$
$\quad + Q7 * /Q6 * /Q5 * /Q4 * Q3 * /Q2 * Q1 * /Q0$
$\quad + Q7 * /Q6 * /Q5 * /Q4 * Q3 * /Q2 * Q1 * Q0$
$\quad + Q7 * /Q6 * /Q5 * /Q4 * Q3 * Q2 * /Q1 * /Q0$

$Q6^+ = /Q7 * /Q6 * Q5 * /Q4 * /Q3 * Q2 * /Q1 * /Q0$

$Q5^+ = /Q7 * /Q6 * Q5 * /Q4 * /Q3 * /Q2 * /Q1 * /Q0$
$\quad + /Q7 * /Q6 * Q5 * /Q4 * /Q3 * /Q2 * /Q1 * Q0$
$\quad + /Q7 * /Q6 * Q5 * /Q4 * /Q3 * /Q2 * Q1 * /Q0$
$\quad + /Q7 * /Q6 * Q5 * /Q4 * /Q3 * /Q2 * Q1 * Q0$
$\quad + Q7 * /Q6 * /Q5 * Q4 * /Q3 * /Q2 * /Q1 * /Q0$

$Q4^+ = Q7 * /Q6 * /Q5 * /Q4 * Q3 * Q2 * /Q1 * /Q0$

$Q3^+ = /Q7 * Q6 * /Q5 * /Q4 * /Q3 * /Q2 * /Q1 * /Q0$
$\quad + Q7 * /Q6 * /Q5 * /Q4 * Q3 * /Q2 * /Q1 * /Q0$
$\quad + Q7 * /Q6 * /Q5 * /Q4 * Q3 * /Q2 * /Q1 * Q0$
$\quad + Q7 * /Q6 * /Q5 * /Q4 * Q3 * /Q2 * Q1 * /Q0$
$\quad + Q7 * /Q6 * /Q5 * /Q4 * Q3 * /Q2 * Q1 * Q0$

$Q2^+ = /Q7 * /Q6 * Q5 * /Q4 * /Q3 * /Q2 * Q1 * Q0$
$\quad + Q7 * /Q6 * /Q5 * /Q4 * Q3 * /Q2 * Q1 * Q0$
$\quad + /Q7 * /Q6 * Q5 * /Q4 * /Q3 * /Q2 * /Q1 * /Q0$
$\quad * /SENA * SENB$
$\quad + Q7 * /Q6 * /Q5 * /Q4 * Q3 * /Q2 * /Q1 * /Q0$
$\quad * SENA * SENB$

$Q1^+ = /Q7 * /Q6 * Q5 * /Q4 * /Q3 * /Q2 * /Q1 * Q0$
$\quad + /Q7 * /Q6 * Q5 * /Q4 * /Q3 * /Q2 * Q1 * /Q0$
$\quad + Q7 * /Q6 * /Q5 * /Q4 * Q3 * /Q2 * /Q1 * Q0$
$\quad + Q7 * /Q6 * /Q5 * /Q4 * Q3 * /Q2 * Q1 * /Q0$

$Q0^+ = /Q7 * /Q6 * Q5 * /Q4 * /Q3 * /Q2 * Q1 * /Q0$
$\quad + Q7 * /Q6 * /Q5 * /Q4 * Q3 * /Q2 * Q1 * /Q0$
$\quad + /Q7 * /Q6 * Q5 * /Q4 * /Q3 * /Q2 * /Q1 * /Q0$
$\quad * /SENA * /SENB$
$\quad + Q7 * /Q6 * /Q5 * /Q4 * Q3 * /Q2 * /Q1 * /Q0$
$\quad * /SENA * /SENB$
$\quad + /Q7 * /Q6 * Q5 * /Q4 * /Q3 * /Q2 * /Q1 * /Q0$
$\quad * SENA * SENB$

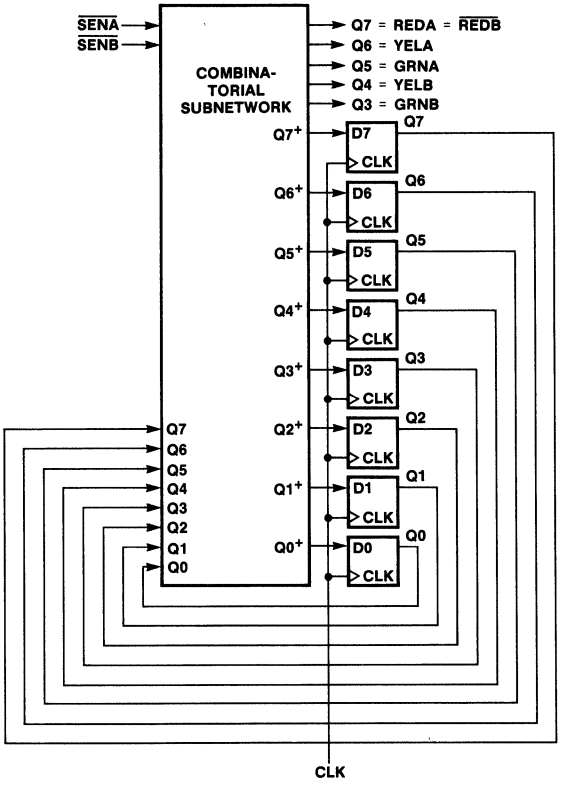


Figure 14. State Machine Traffic Signal Controller No. 2 (Note flip-flops are directly circuit outputs)

```
PAL16R8                                                    PAL DESIGN SPECIFICATION
TRACNT2                                                    B. BRAFMAN 07/16/81
TRAFFIC SIGNAL CONTROLLER No. 2
MMI FIELD APPLICATIONS ENGINEER YORBA LINDA, CALIFORNIA
CLK /SENA /SENB /INIT NC  NC NC NC NC GND
GND /Q0 /Q1 /Q2 /Q3 /Q4 /Q5 /Q6 /Q7 VCC


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

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

Q5 := /Q7 * /Q6 *  Q5 * /Q4 * /Q3 * /Q2 * /Q1 * /Q0
    + /Q7 * /Q6 *  Q5 * /Q4 * /Q3 * /Q2 * /Q1 *  Q0
    + /Q7 * /Q6 *  Q5 * /Q4 * /Q3 * /Q2 *  Q1 * /Q0
    + /Q7 * /Q6 *  Q5 * /Q4 * /Q3 * /Q2 *  Q1 *  Q0
    +  Q7 * /Q6 * /Q5 *  Q4 * /Q3 * /Q2 * /Q1 * /Q0
    +  INIT                ; PRESET THIS BIT FOR INITIALIZATION TO STATE S0

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

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

Q2 := /Q7 * /Q6 *  Q5 * /Q4 * /Q3 * /Q2 *  Q1 *  Q0 * /INIT
    +  Q7 * /Q6 * /Q5 * /Q4 *  Q3 * /Q2 *  Q1 *  Q0 * /INIT
    + /Q7 * /Q6 *  Q5 * /Q4 * /Q3 * /Q2 * /Q1 * /Q0 * /INIT * /SENA *  SENB
    +  Q7 * /Q6 * /Q5 * /Q4 *  Q3 * /Q2 * /Q1 * /Q0 * /INIT *  SENA *  SENB

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

Q0 := /Q7 * /Q6 *  Q5 * /Q4 * /Q3 * /Q2 *  Q1 * /Q0 * /INIT
    +  Q7 * /Q6 * /Q5 * /Q4 *  Q3 * /Q2 *  Q1 * /Q0 * /INIT
    + /Q7 * /Q6 *  Q5 * /Q4 * /Q3 * /Q2 * /Q1 * /Q0 * /INIT * /SENA * /SENB
    +  Q7 * /Q6 * /Q5 * /Q4 *  Q3 * /Q2 * /Q1 * /Q0 * /INIT * /SENA * /SENB
    + /Q7 * /Q6 *  Q5 * /Q4 * /Q3 * /Q2 * /Q1 * /Q0 * /INIT *  SENA *  SENB
    +  Q7 * /Q6 * /Q5 * /Q4 *  Q3 * /Q2 * /Q1 * /Q0 * /INIT *  SENA *  SENB
```

FUNCTION TABLE

/INIT	CLK	SENA	SENB	Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0	;COMMENTS
L	C	X	X	L	L	H	L	L	L	L	L	S0 (INIT)
H	C	L	L	L	L	H	L	L	L	L	H	S1
H	C	X	X	L	L	H	L	L	L	H	L	S2
H	C	X	X	L	L	H	L	L	L	H	H	S3
H	C	X	X	L	L	H	L	L	H	L	L	S4
H	C	X	X	L	H	L	L	L	L	L	L	S5
H	C	X	X	H	L	L	L	H	L	L	L	S6
H	C	L	L	H	L	L	L	H	L	L	H	S7
H	C	X	X	H	L	L	L	H	L	H	L	S8
H	C	X	X	H	L	L	L	H	L	H	H	S9
H	C	X	X	H	L	L	L	H	H	L	L	S10
H	C	X	X	H	L	L	H	L	L	L	L	S11
H	C	X	X	L	L	H	L	L	L	L	L	S0
H	C	L	H	L	L	H	L	L	H	L	L	S4
L	C	X	X	L	L	H	L	L	L	L	L	S0 (INIT)
H	C	H	L	L	L	H	L	L	L	L	L	S0
H	C	H	H	L	L	H	L	L	L	L	H	S1

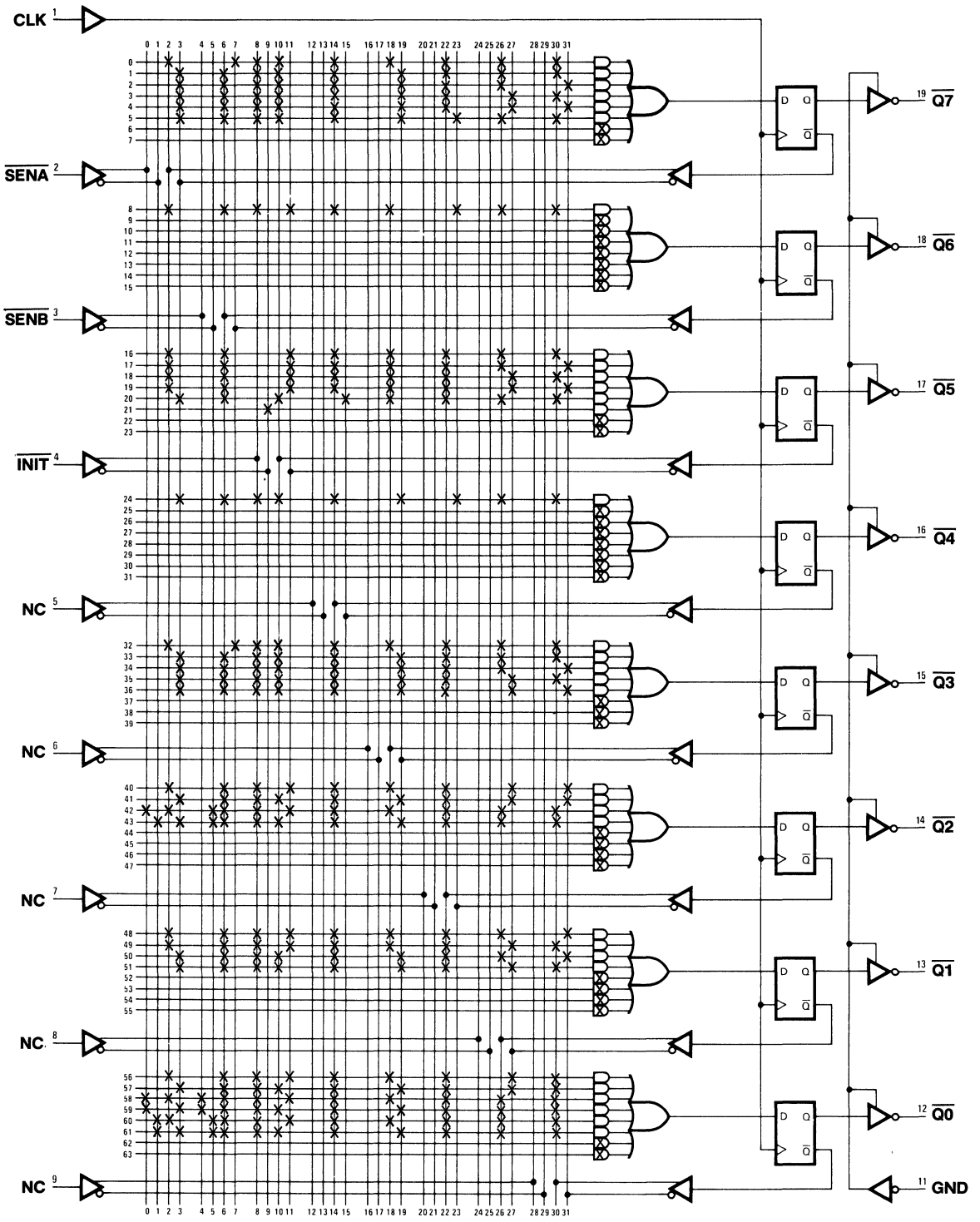
DESCRIPTION

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

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

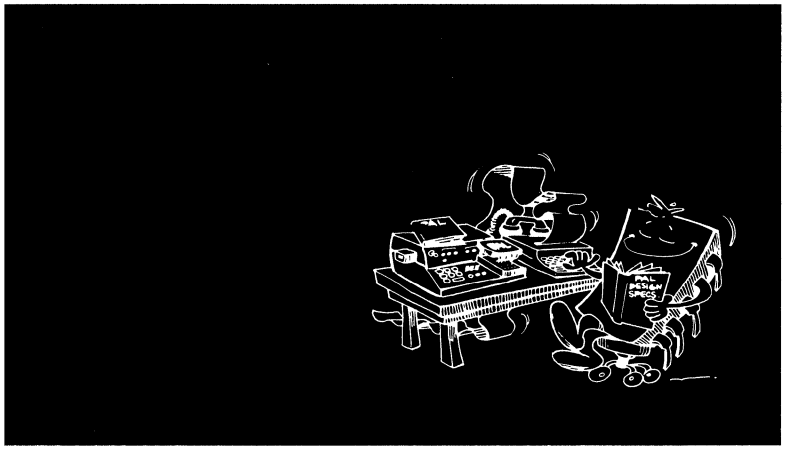
Traffic Signal Controller No. 2

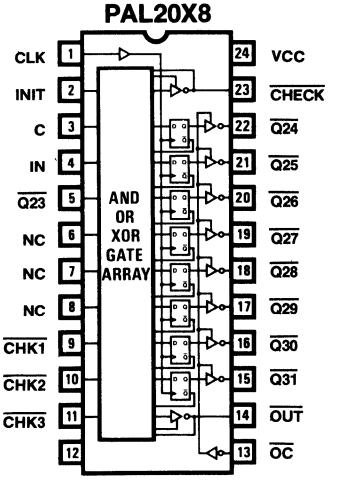
Logic Diagram PAL16R8

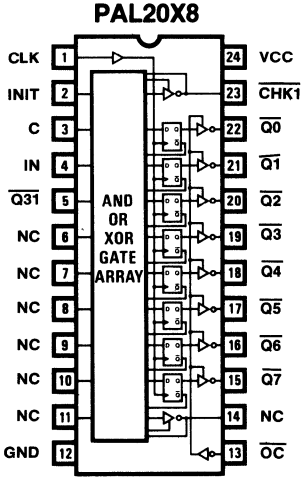


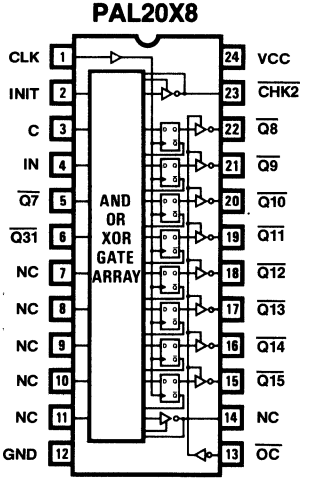

IMPORTANT
NOTES

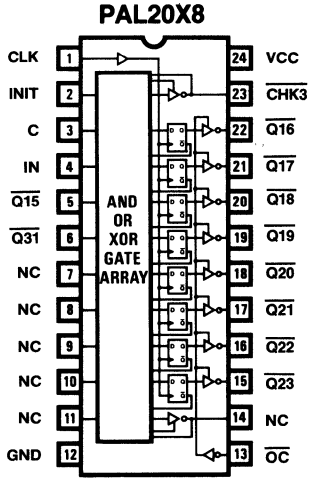
32-Bit CRC (Cyclical Redundancy Checking) Error Detection











CYCLIC REDUNDANCY CHECK (CRC) USING PALS

Programmable Array Logic Devices Provide Efficient Implementation of Popular Local Network Error Checking Protocol

There is a growing interest in providing data communication links to connect several processors and peripherals into one local area network. One of the most popular networks is the Ethernet. To insure reliable communications in the network an efficient error detection scheme is required. The Ethernet protocol specifies a 32-bit cycle redundancy check (CRC).

The following application opens with a tutorial on the CRC and then shows a detailed implementation of the Ethernet CRC using PAL. The use of fuse programmable devices allows easy modification to accommodate other data communications protocols as well as other applications (CRC in disk drives, etc.).

Introduction

The growing number of high speed digital links and the need for reliable communication require implementation of efficient error detection schemes. A 32-bit CRC using PAL circuits meets these requirements.

What is CRC ?

CRC is the acronym for Cyclic Redundancy Check, an error detection technique widely used in serial communication systems from computer to computer or from computer to peripheral devices. This technique operates on serial bits of information treated as the coefficients of a binary polynomial, P(x), and processes these bits in modulo-2 arithmetic.

The basic coding concept of the CRC is to modify the polynomial, P(x), so that it is exactly divisible by a fixed polynomial, G(x); the divisor G(x) is referred to as the generator polynomial. The modified polynomial, M(x), is transmitted. M(x) is divided by the same G(x) when received or fetched. If the remainder is zero, all bits are assumed to be correct; otherwise, a flag is set to indicate an error.

An Example Using CRC

A 6-bit message (101110) can be represented in polynomial form as:

$$P(x) = 1 + 0x + 1x^2 + 1x^3 + 1x^4 + 0x^5$$

or

$$P(x) = x^4 + x^3 + x^2 + 1$$

The highest power of x is attached to the least significant bit (LSB). The LSB is the first bit transmitted in communication channels. In general P(x) will not be exactly divisible by G(x); the division will generate a quotient, Q(x), and a remainder, R(x). P(x) is prescaled to insure that the order of P(x) is greater than the order of G(x) so that the remainder is always different than the message itself.

Specifically, P(x) is prescaled by x^n where n is the degree of G(x).

$$x^n P(x) = Q(x)\, G(x) + R(x) \qquad (1)$$

For a 3-bit CRC and $G(x) = x^3 + 1$ equation (1) becomes:

$$x^3 (x^4 + x^3 + x^2 + 1) = (x^4 + x^3 + x^2 + x)(x^3 + 1) + (x^2 + x)$$

The operation is performed using modulo-2 arithmetic where the sum and difference are synonymous.
Equation 1 can equivalently be written as:

$$x^n P(x) - R(x) = x^n P(x) + R(x) = Q(x)\, G(x) = M(x)$$

M(x) is exactly divisible by G(x) and it is M(x) that is transmitted. The message M(x) is formed by adding the remaining bits, R(x), of a fixed length n to the message bits. Because the message was prescaled, addition is equivalent to appending the remainder at the end of the data bits. For the example given (remembering that LSB is sent first) the information transmitted is: 011 101110. Three redundancy bits are appended to facilitate error detection.

In conclusion when performing CRC, the transmitter will generate and append R(x) while the receiver will verify the exact division of M(x) by G(x) and signal the occurrence of an error.

Why CRC ?

Compared to other error detection schemes such as parity checking CRC is more powerful:

- all errors within n successive bits are detected
- for even G(x) all errors with an odd number of bits in error are detected (50% of all possible random errors)
- all error patterns that are not divisible by G(x) are detected

CRC is also more efficient (for large frame of information). Efficiency is defined as the number of data bits divided by the total number of bits transmitted. For example: in the Ethernet

specification the number of data bits ranges from 60 to 1500 bytes, the total number of bits transmitted will contain only an extra 32-bits (if 32-bit CRC is used) instead of an extra 60 to 1500 parity bits (if we assume one parity bit per byte of information is used.)

Implementation

A binary division that ignores the quotient but retains the remainder, R(x), is conventionally implemented with a shift register. A feedback path from the output of the last stage returns to the input of the stages corresponding to the powers of x that have non zero coefficient in the generating polynomial. At each such input, a modulo-2 adder (Exclusive OR) combines the feedback signal with the output from the previous stage.

This is equivalent to a shift subtract on each clock cycle. The example in Fig. 1 shows a 32-bit CRC where G (x), the generator polynominal, (used in ETHERNET and AUTOBAUND II) is given by:

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

Transmitter operation:

1. Initialize shift register to all Hs. (INIT = H)
2. Data is shifted in the feedback shift register to generate R(x) and fed to the output. (Control C = H)
3. After last data bit has been processed, the complimented R(x) is shifted out for transmission. (Control C = L)

Receiver operation:

1. Initialize shift register to all Hs.
2. Data is shifted in the feedback shift register to regenerate R(x).
3. After last data bit has been processed, Control remains High and the complement of R(x) is shifted (as generated by transmitter).

If the two R(x) match, the final content of the shift register is:

X31 X0
11000111000001001101110101111011 (residue)

4) This value is tested. (Check = L indicates no error, as shown in Fig. 2)

The same hardware can perform both the transmitter and receiver function if the respective inputs are multiplexed.

PAL Configuration

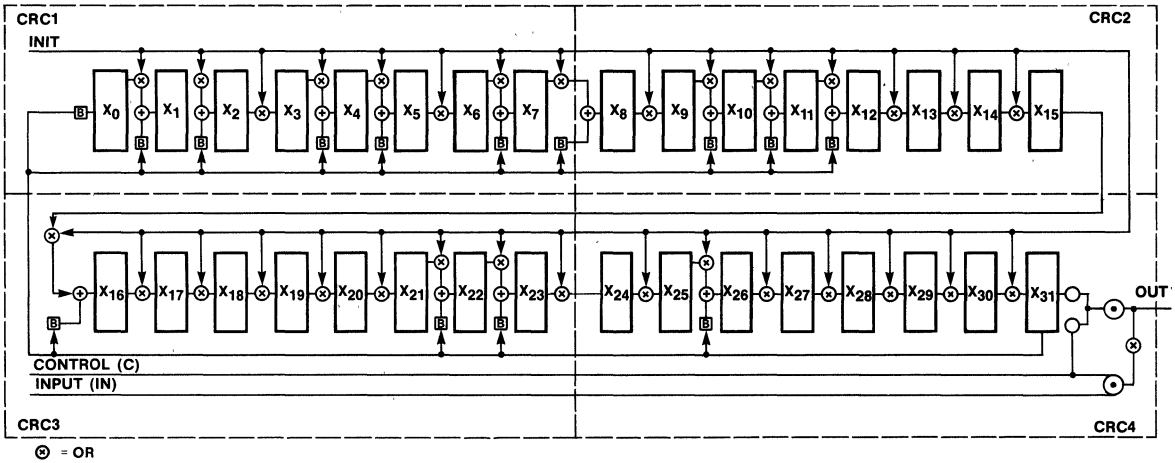
Four PAL circuits are used to implement the 32-bit CRC. The PAL interconnections are shown in Fig. 2. Eight bits of the 32-bit shift register are distributed in each PAL. The logic to check for the residue is distributed among the four PAL circuits as shown in Fig. 3.

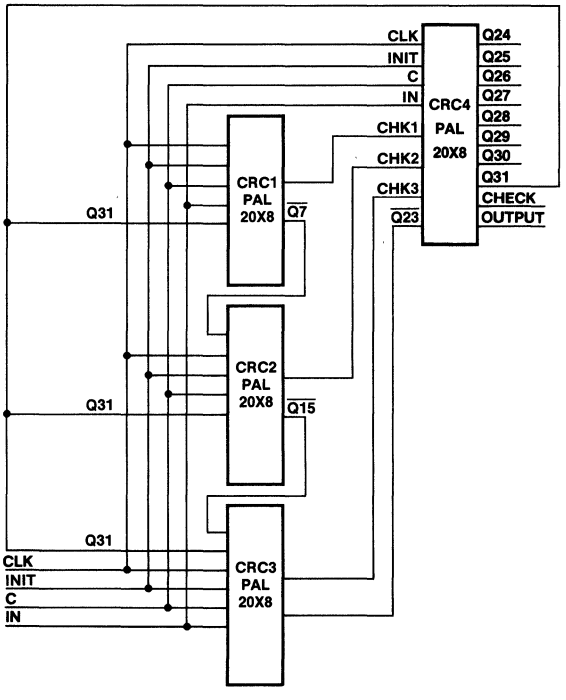
Summary

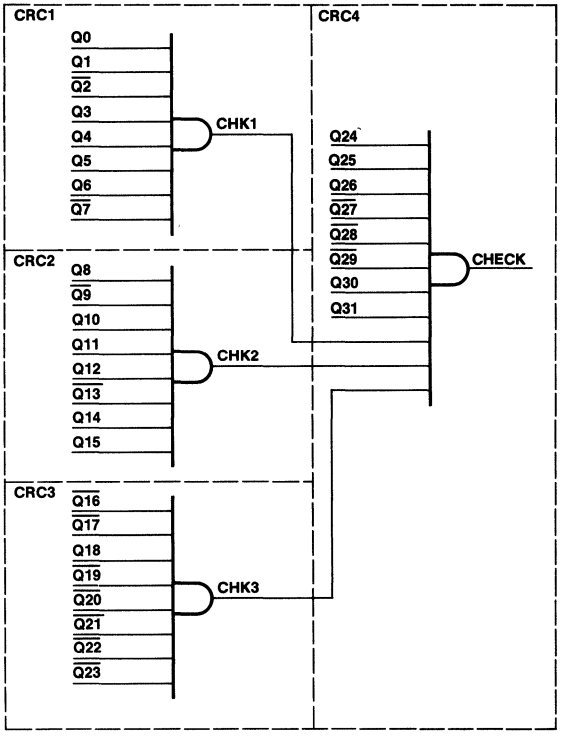
The lack of standardization in data communication equipment makes the use of PAL devices very attractive due to their flexibility and ease of design modification.

REFERENCES

The Ethernet, A Local Area Network, Data Link Layer and Physical Layer Specifications, version 1.0 (joint publication of Digital Equipment Corp., Intel Corp., and Xerox Corp., 1980).


CRC1
INIT
CRC2
CRC3
CRC4
CONTROL (C)
INPUT (IN)
OUT
⊗ = OR
⊕ = XOR
B = (IN ⊕ Q31) · C · INIT ≡ IN · Q31 · C · INIT + IN · Q31 · C · INIT
⊙ = AND
○ = NOT
D Q
CK


CLK
INIT
C
IN
CHK1
CHK2
CHK3
Q31
CRC1 PAL 20X8
CRC2 PAL 20X8
CRC3 PAL 20X8
CRC4 PAL 20X8
Q7
Q15
Q23
Q24
Q25
Q26
Q27
Q28
Q29
Q30
Q31
CHECK
OUTPUT

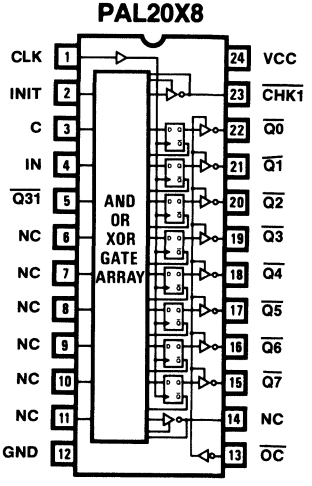

CRC1
CRC4
Q0
Q1
Q2
Q3
Q4
Q5
Q6
Q7
CHK1
Q24
Q25
Q26
Q27
Q28
Q29
Q30
Q31
CHECK
CRC2
Q8
Q9
Q10
Q11
Q12
Q13
Q14
Q15
CHK2
CRC3
Q16
Q17
Q18
Q19
Q20
Q21
Q22
Q23
CHK3

```
PAL20X8                                                    PAL DESIGN SPECIFICATION
CRC1                                                          NADIA SACHS 08/14/81
32-BIT CRC (CYCLICAL REDUNDANCY CHECKING) ERROR DETECTION, CHIP 1
MMI SUNNYVALE, CALIFORNIA
CLK INIT C IN /Q31 NC NC NC NC NC NC GND
/OC NC /Q7 /Q6 /Q5 /Q4 /Q3 /Q2 /Q1 /Q0 /CHK1 VCC


IF(VCC) CHK1 =  Q0* Q1*/Q2* Q3* Q4* Q5* Q6*/Q7   ;CHECK BIT 1

Q0  :=   Q31* C                                  ;SHIFT IF C
     +   INIT                                    ;INITIALIZE
    :+: /INIT* IN* C                             ;MODULO-2
     +  /INIT* IN* C                             ;ADDITION

Q1  :=   Q0                                      ;SHIFT
     +   INIT                                    ;INITIALIZE
    :+: /INIT* C* IN*/Q31                        ;MODULO-2
     +  /INIT* C*/IN* Q31                        ;ADDITION

Q2  :=   Q1                                      ;SHIFT
     +   INIT                                    ;INITIALIZE
    :+: /INIT* C* IN*/Q31                        ;MODULO-2
     +  /INIT* C*/IN* Q31                        ;ADDITION

Q3  :=   Q2                                      ;SHIFT
     +   INIT                                    ;INITIALIZE

Q4  :=   Q3                                      ;SHIFT
     +   INIT                                    ;INITIALIZE
    :+: /INIT* C* IN*/Q31                        ;MODULO-2
     +  /INIT* C*/IN* Q31                        ;ADDITION

Q5  :=   Q4                                      ;SHIFT
     +   INIT                                    ;INITIALIZE
    :+: /INIT* C* IN*/Q31                        ;MODULO-2
     +  /INIT* C*/IN* Q31                        ;ADDITION

Q6  :=   Q5                                      ;SHIFT
     +   INIT                                    ;INITIALIZE

Q7  :=   Q6                                      ;SHIFT
     +   INIT                                    ;INITIALIZE
    :+: /INIT* C* IN*/Q31                        ;MODULO-2
     +  /INIT* C*/IN* Q31                        ;ADDITION
```

FUNCTION TABLE

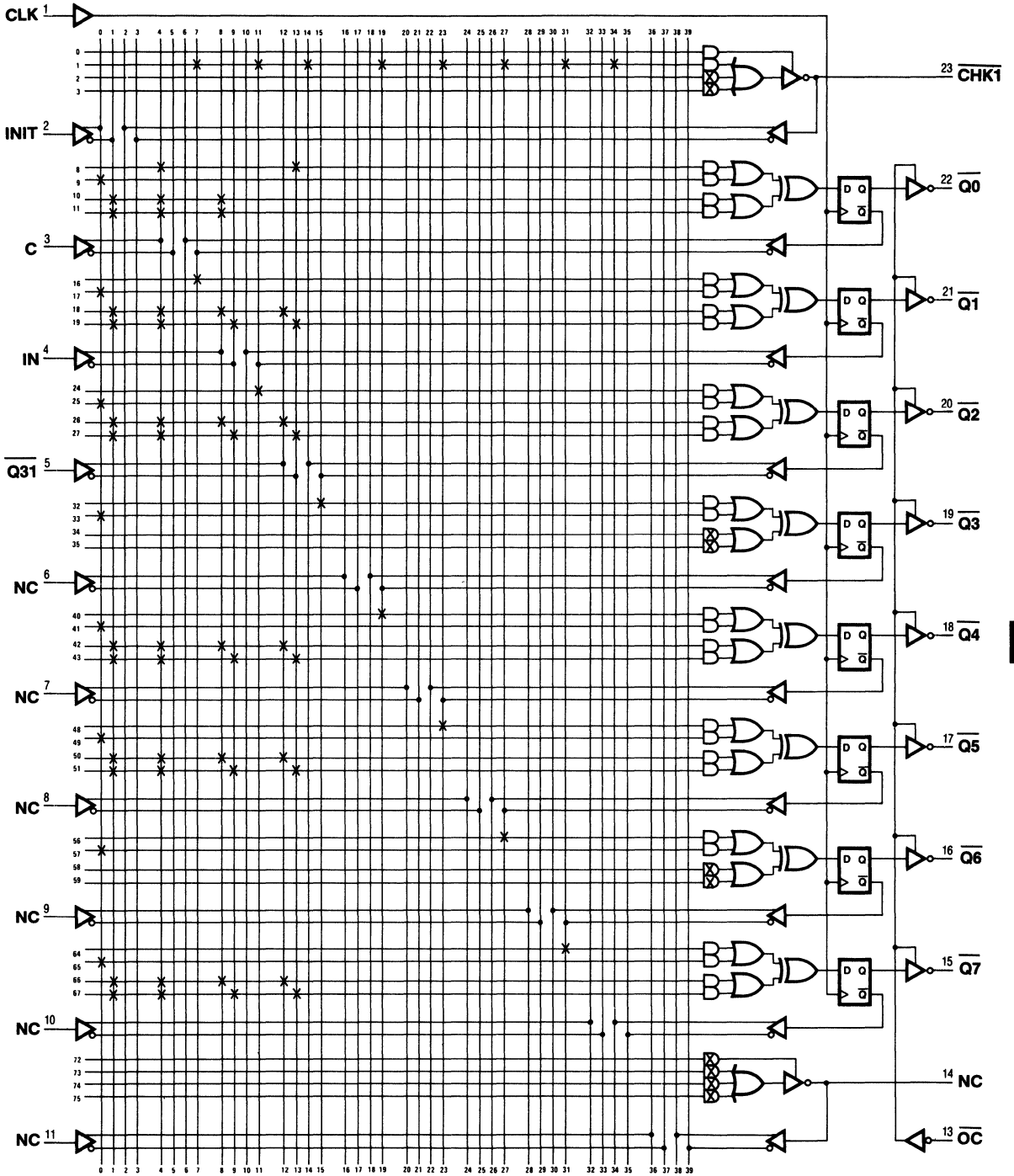
CLK /OC INIT C IN Q31 Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0 CHK1

;					Q3	Q Q Q Q Q Q Q Q	
;CLK	/OC	INIT	C	IN	1	7 6 5 4 3 2 1 0	CHK1
C	L	H	X	X	X	H H H H H H H H	X
C	L	L	H	L	H	L H L L H L L H	X
C	L	L	H	L	H	L L H L L H L H	X
C	L	L	H	L	H	H H H H H H L H	X
C	L	L	H	L	H	L H L L H H L H	X
C	L	L	H	L	H	L L H L H H L H	X
C	L	L	H	L	H	H H H L H H L H	X
C	L	L	H	L	L	H H L H H L H L	X
C	L	L	H	L	L	H L H H L H L L	X
C	L	L	H	L	L	L H H L H L L L	X
C	L	L	H	L	H	L H H L L H H H	X
C	L	L	H	L	L	H H L L H H H L	X
C	L	L	H	L	L	H L L H H H H L	X
C	L	L	H	L	H	H L L L H H H H	X
C	L	L	H	L	H	H L H L H L L H	X
C	L	L	H	L	H	H H H L L H L H	X
C	L	L	H	L	H	L H H H H H L H	X
C	L	L	H	L	L	H H H H H H L H	X
C	L	L	H	L	L	H H H H H L H L	X
C	L	L	H	L	L	H H H L H L L L	X
C	L	L	H	L	L	H H L H L L L L	X
C	L	L	H	L	L	H L H L L L L L	X
C	L	L	H	L	L	L H L L L L L L	X
C	L	L	H	L	L	H L L L L L L L	X
C	L	L	H	L	L	L L L L L L L L	X
C	L	L	H	L	H	H L H H H H H H	X
C	L	L	H	L	L	L H H L H H H L	X
C	L	L	H	L	H	L H H L H L H H	X
C	L	L	H	L	H	L H H L L L L H	X
C	L	L	H	L	L	H H L L L L H L	X
C	L	L	H	L	H	L L H H L L H H	X
C	L	L	H	L	L	L H H L L H H L	X
C	L	L	H	L	H	L H H H H H H H	H



32-Bit CRC, Chip 1

Logic Diagram PAL20X 8



```
PAL20X8                                                    PAL DESIGN SPECIFICATION
CRC2                                                          NADIA SACHS 08/14/81
32-BIT CRC (CYCLICAL REDUNDANCY CHECKING) ERROR DETECTION, CHIP 2
MMI SUNNYVALE, CALIFORNIA
CLK INIT C IN /Q7 /Q31 NC NC NC NC NC GND
/OC NC /Q15 /Q14 /Q13 /Q12 /Q11 /Q10 /Q9 /Q8 /CHK2 VCC


IF(VCC) CHK2 =  Q8*/Q9* Q10* Q11* Q12*/Q13* Q14* Q15    ;CHECK BIT 2

Q8  :=   Q7                                             ;SHIFT
     +   INIT                                           ;INITIALIZE
    :+: /INIT* C* IN*/Q31                               ;MODULO-2
     +  /INIT* C*/IN* Q31                               ;ADDITION

Q9  :=   Q8                                             ;SHIFT
     +   INIT                                           ;INITIALIZE

Q10 :=   Q9                                             ;SHIFT
     +   INIT                                           ;INITIALIZE
    :+: /INIT* C* IN*/Q31                               ;MODULO-2
     +  /INIT* C*/IN* Q31                               ;ADDITION

Q11 :=   Q10                                            ;SHIFT
     +   INIT                                           ;INITIALIZE
    :+: /INIT* C* IN*/Q31                               ;MODULO-2
     +  /INIT* C*/IN* Q31                               ;ADDITION

Q12  :=  Q11                                            ;SHIFT
     +   INIT                                           ;INITIALIZE
    :+: /INIT* C* IN*/Q31                               ;MODULO-2
     +  /INIT* C*/IN* Q31                               ;ADDITION

Q13 :=   Q12                                            ;SHIFT
     +   INIT                                           ;INITIALIZE

Q14 :=   Q13                                            ;SHIFT
     +   INIT                                           ;INITIALIZE

Q15 :=   Q14                                            ;SHIFT
     +   INIT                                           ;INITIALIZE
```

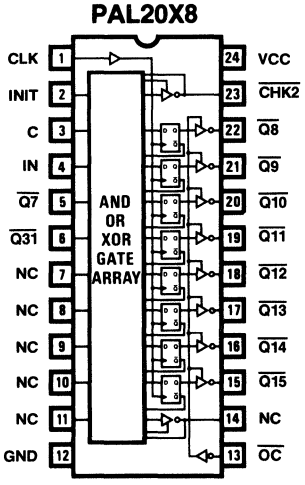
FUNCTION TABLE

CLK /OC INIT C IN Q7 Q31 Q15 Q14 Q13 Q12 Q11 Q10 Q9 Q8 CHK2

;						Q	QQQQQQQQ	
;					Q	3	111111	
;CLK	/OC	INIT	C	IN	7	1	54321098	CHK2
C	L	H	X	X	X	X	HHHHHHHH	X
C	L	L	H	L	H	H	HHHLLLHL	L
C	L	L	H	L	L	H	HHLHHLLH	L
C	L	L	H	L	L	H	HLHLHHHH	X
C	L	L	H	L	H	H	LHLLLLHL	X
C	L	L	H	L	L	H	HLLHHLLH	X
C	L	L	H	L	L	H	LLHLHHHH	X
C	L	L	H	L	H	L	LHLHHHHH	X
C	L	L	H	L	H	L	HLHHHHHH	X
C	L	L	H	L	H	L	LHHHHHHH	X
C	L	L	H	L	L	H	HHHLLLHH	X
C	L	L	H	L	L	L	HHLLLHHL	X
C	L	L	H	L	H	L	HLLLHHLH	X
C	L	L	H	L	H	H	LLLLLHHL	X
C	L	L	H	L	H	H	LLLHLLLL	X
C	L	L	H	L	H	H	LLHHHHLL	X
C	L	L	H	L	H	H	LHHLLHLL	X
C	L	L	H	L	L	L	HHLLHLLL	X
C	L	L	H	L	H	L	HLLHLLLH	X
C	L	L	H	L	H	L	LLHLLLHH	X
C	L	L	H	L	H	L	LHLLLHHH	X
C	L	L	H	L	H	L	HLLLHHHH	X
C	L	L	H	L	H	L	LLLHHHHH	X
C	L	L	H	L	L	L	LLHHHHHL	X
C	L	L	H	L	H	L	LHHHHHLH	X
C	L	L	H	L	L	H	HHHLLHHH	X
C	L	L	H	L	H	L	HHLLHHHH	X
C	L	L	H	L	L	H	HLLLLLHH	X
C	L	L	H	L	L	H	LLLHHLHH	X
C	L	L	H	L	L	L	LLHHLHHL	X
C	L	L	H	L	H	H	LHHHLLLL	X
C	L	L	H	L	L	L	HHHLLLLL	X
C	L	L	H	L	L	H	HHLHHHLH	H

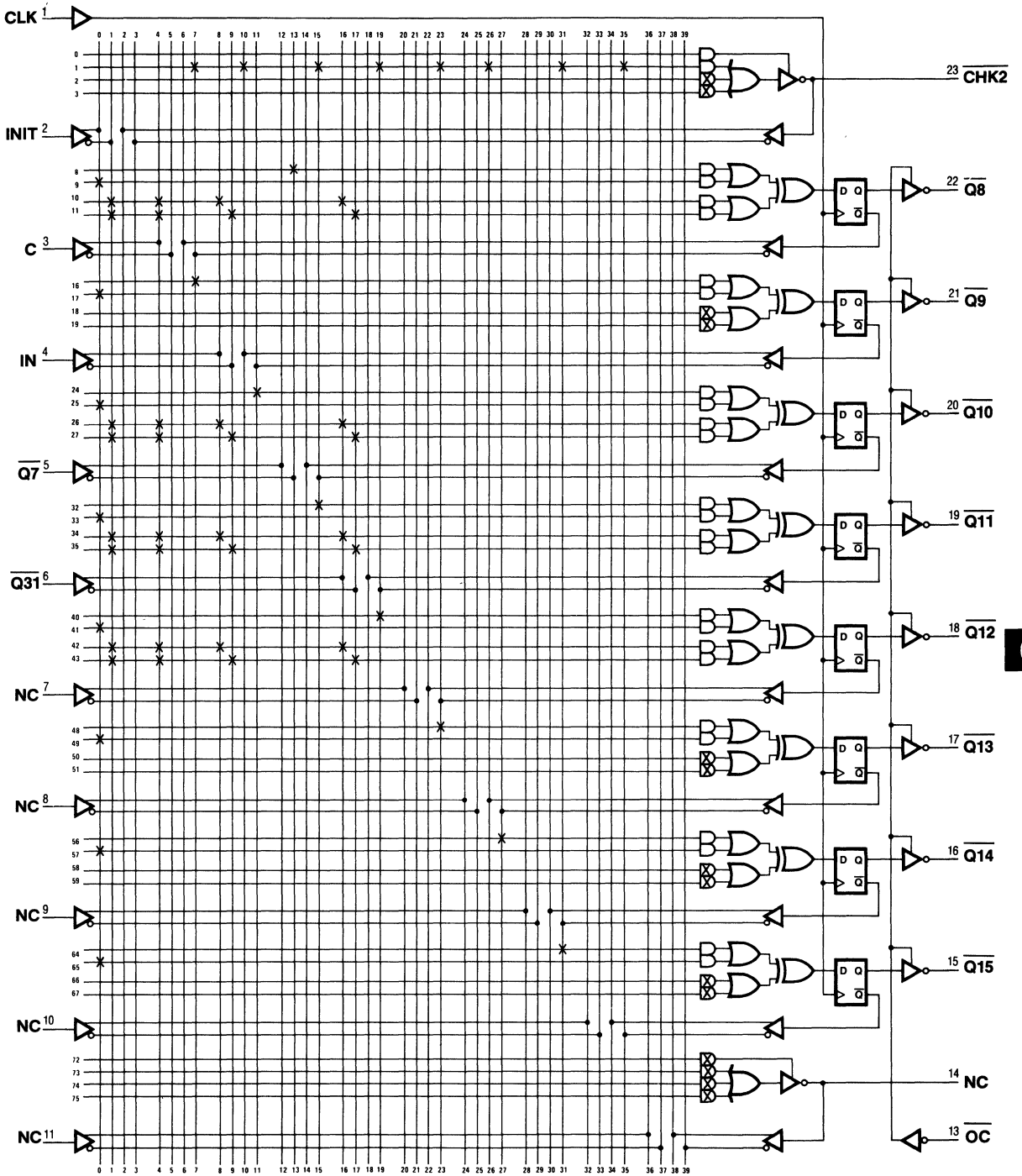
DESCRIPTION

SECOND 8-BIT SHIFT REGISTER AND CHECK.



32-Bit CRC, Chip 2

Logic Diagram PAL20X 8



```
PAL20X8                                                    PAL DESIGN SPECIFICATION
CRC3                                                          NADIA SACHS 08/14/81
32-BIT CRC (CYCLICAL REDUNDANCY CHECKING) ERROR DETECTION CHIP 3
MMI SUNNYVALE,CALIFORNIA
CLK INIT C IN /Q15 /Q31 NC NC NC NC NC GND
/OC NC /Q23 /Q22 /Q21 /Q20 /Q19 /Q18 /Q17 /Q16 /CHK3 VCC


IF(VCC) CHK3  = /Q16*/Q17* Q18*/Q19*/Q20*/Q21*/Q22*/Q23    ;CHECK BIT 3

Q16  :=   Q15                                              ;SHIFT
      +   INIT                                             ;INITIALIZE
     :+: /INIT* C* IN*/Q31                                 ;MODULO-2
      +  /INIT* C*/IN* Q31                                 ;ADDITION

Q17  :=   Q16                                              ;SHIFT
      +   INIT                                             ;INITIALIZE

Q18  :=   Q17                                              ;SHIFT
      +   INIT                                             ;INITIALIZE

Q19  :=   Q18                                              ;SHIFT
      +   INIT                                             ;INITIALIZE

Q20  :=   Q19                                              ;SHIFT
      +   INIT                                             ;INITIALIZE

Q21  :=   Q20                                              ;SHIFT
      +   INIT                                             ;INITIALIZE

Q22  :=   Q21                                              ;SHIFT
      +   INIT                                             ;INITIALIZE
     :+: /INIT* C* IN*/Q31                                 ;MODULO-2
      +  /INIT* C*/IN* Q31                                 ;ADDITION

Q23  :=   Q22                                              ;SHIFT
      +   INIT                                             ;INITIALIZE
     :+: /INIT* C* IN*/Q31                                 ;MODULO-2
      +  /INIT* C*/IN* Q31                                 ;ADDITION
```

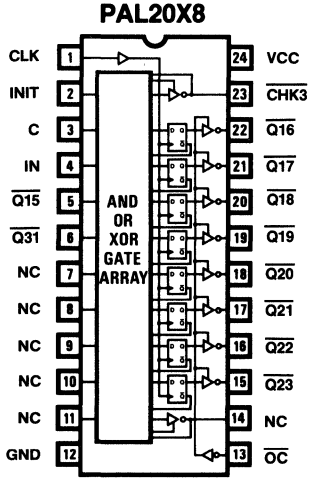
FUNCTION TABLE

CLK /OC INIT C IN Q15 Q31 Q23 Q22 Q21 Q20 Q19 Q18 Q17 Q16 CHK3

;					Q	Q	QQQQQQQQ	
;					1	3	22221111	
;CLK	/OC	INIT	C	IN	5	1	32109876	CHK3
C	L	H	X	X	X	X	HHHHHHHH	X
C	L	L	H	L	H	H	LLHHHHHL	L
C	L	L	H	L	H	H	HLHHHHLL	L
C	L	L	H	L	H	H	HLHHHLLL	X
C	L	L	H	L	H	H	HLHHLLLL	X
C	L	L	H	L	L	H	HLHLLLLH	X
C	L	L	H	L	H	H	HLLLLLHL	X
C	L	L	H	L	L	L	LLLLLHLL	X
C	L	L	H	L	L	L	LLLLHLLL	X
C	L	L	H	L	H	L	LLLHLLLH	X
C	L	L	H	L	L	H	HHHLLLHH	X
C	L	L	H	L	H	L	HHLLLHHH	X
C	L	L	H	L	H	L	HLLLHHHH	X
C	L	L	H	L	H	H	HHLHHHHL	X
C	L	L	H	L	L	H	LHHHHHLH	X
C	L	L	H	L	L	H	LLHHHLHH	X
C	L	L	H	L	L	H	HLHHLHHH	X
C	L	L	H	L	L	L	LHHLHHHL	X
C	L	L	H	L	H	L	HHLHHHLH	X
C	L	L	H	L	H	L	HLHHHLHH	X
C	L	L	H	L	L	L	LHHHLHHL	X
C	L	L	H	L	L	L	HHHLHHLL	X
C	L	L	H	L	H	L	HHLHHLLH	X
C	L	L	H	L	L	L	HLHHLLHL	X
C	L	L	H	L	L	L	LHHLLHLL	X
C	L	L	H	L	L	H	LLLLHLLH	X
C	L	L	H	L	H	L	LLLHLLHH	X
C	L	L	H	L	H	H	HHHLLHHL	X
C	L	L	H	L	H	H	LLLLHHLL	X
C	L	L	H	L	L	L	LLLHHLLL	X
C	L	L	H	L	L	H	HHHHLLLH	X
C	L	L	H	L	L	L	HHHLLLHL	X
C	L	L	H	L	H	H	LLLLLHLL	H

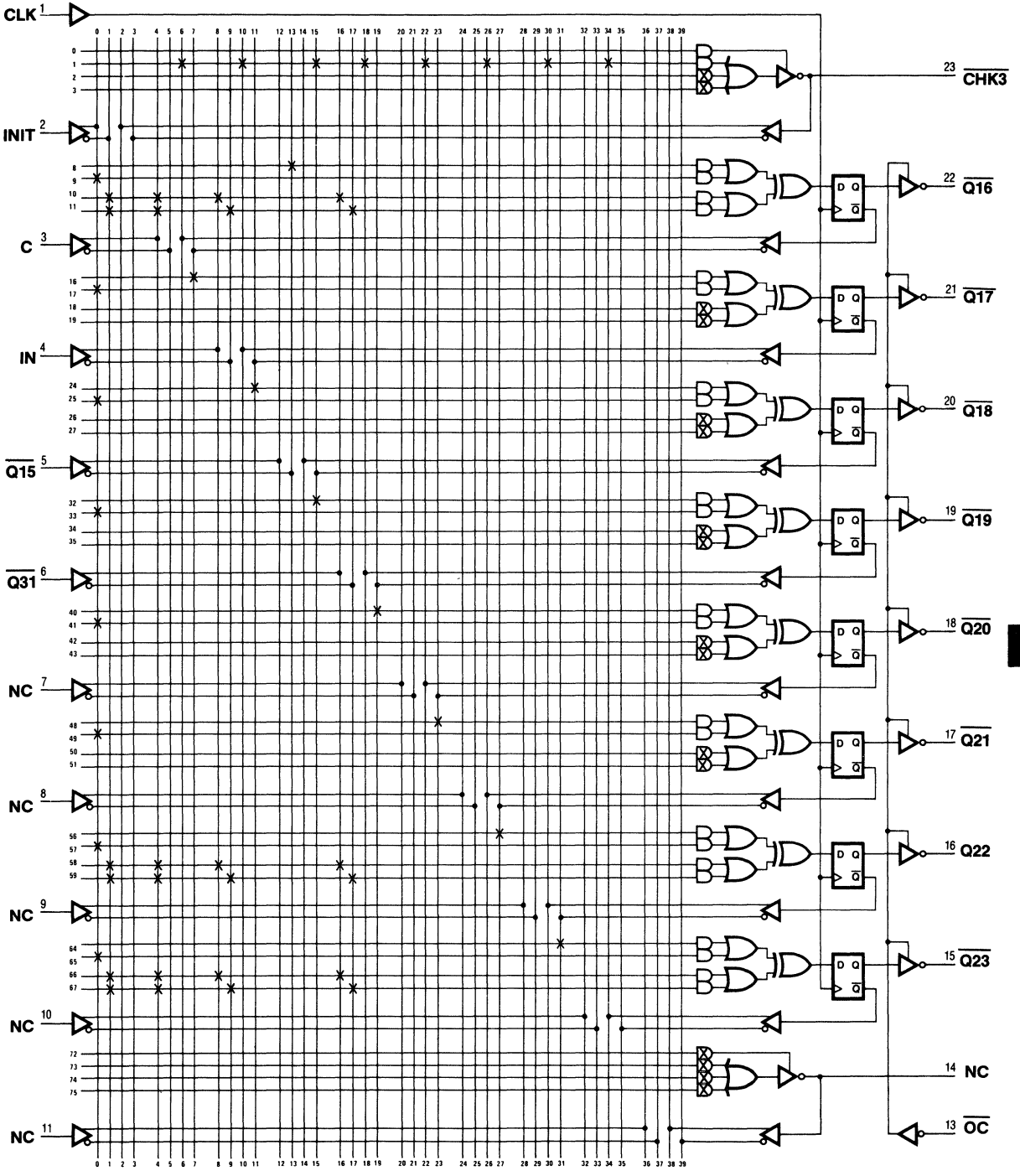
DESCRIPTION

THIRD 8-BIT SHIFT REGISTER AND CHECK.



32-Bit CRC, Chip 3

Logic Diagram PAL20X 8



```
PAL20X8                                                    PAL DESIGN SPECIFICATION
CRC4                                                          NADIA SACHS 08/14/81
32-BIT CRC (CYCLICAL REDUNDANCY CHECKING) ERROR DETECTION, CHIP 4
MMI SUNNYVALE, CALIFORNIA
CLK INIT C IN /Q23 NC NC NC /CHK1 /CHK2 /CHK3 GND
/OC /OUT /Q31 /Q30 /Q29 /Q28 /Q27 /Q26 /Q25 /Q24 /CHECK VCC


IF(VCC) CHECK  =   CHK1* CHK2* CHK3* Q24* Q25* Q26*/Q27   ;CHECK
                *                      /Q28*/Q29* Q30* Q31   ;ERROR

Q24    :=   Q23                                            ;SHIFT
        +   INIT                                           ;INITIALIZE

Q25    :=   Q24                                            ;SHIFT
        +   INIT                                           ;INITIALIZE

Q26    :=   Q25                                            ;SHIFT
        +   INIT                                           ;INITIALIZE
       :+: /INIT* C* IN*/Q31                               ;MODULO-2
        +  /INIT* C*/IN* Q31                               ;ADDITION

Q27    :=   Q26                                            ;SHIFT
        +   INIT                                           ;INITIALIZE

Q28    :=   Q27                                            ;SHIFT
        +   INIT                                           ;INITIALIZE

Q29    :=   Q28                                            ;SHIFT
        +   INIT                                           ;INITIALIZE

Q30    :=   Q29                                            ;SHIFT
        +   INIT                                           ;INITIALIZE

Q31    :=   Q30                                            ;SHIFT
        +   INIT                                           ;INITIALIZE

IF(VCC) OUT =  Q31*/C                                      ;SERIAL
             + /IN * C                                     ;OUT
```

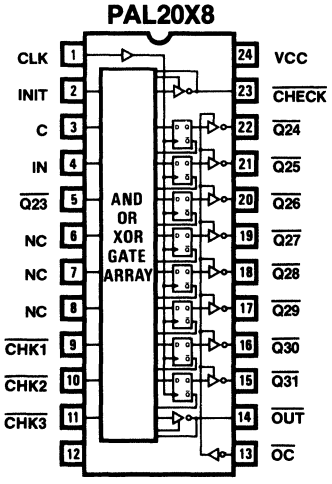
FUNCTION TABLE

CLK /OC INIT C IN CHK1 CHK2 CHK3 Q23 Q31 Q30 Q29 Q28 Q27 Q26 Q25 Q24 OUT CHECK

;						Q	QQQQQQQQ		
;					CHK	2	33222222		
;CLK	/OC	INIT	C	IN	123	3	10987654	OUT	CHECK
C	L	H	X	L	XXX	X	HHHHHHHH	X	X
C	L	L	H	L	XXX	H	HHHHHLHH	H	X
C	L	L	H	L	XXX	L	HHHHLLHL	H	X
C	L	L	H	L	XXX	H	HHHLLLLH	H	X
C	L	L	H	L	XXX	H	HHLLLHHH	H	X
C	L	L	H	L	XXX	H	HLLLHLHH	H	X
C	L	L	H	L	XXX	H	LLLHLLHH	H	X
C	L	L	H	L	XXX	H	LLHLLHHH	H	X
C	L	L	H	L	XXX	L	LHLLHHHL	H	X
C	L	L	H	L	XXX	L	HLLHHHLL	H	X
C	L	L	H	L	XXX	L	LLHHHHLL	H	X
C	L	L	H	L	XXX	H	LHHHHLLH	H	X
C	L	L	H	L	XXX	H	HHHHLLHH	H	X
C	L	L	H	L	XXX	H	HHHLLLHH	H	X
C	L	L	H	L	XXX	H	HHLLLLHH	H	X
C	L	L	H	L	XXX	L	HLLLLLHL	H	X
C	L	L	H	L	XXX	L	LLLLLLLL	H	X
C	L	L	H	L	XXX	H	LLLLLLLH	H	X
C	L	L	H	L	XXX	L	LLLLLLHL	H	X
C	L	L	H	L	XXX	H	LLLLLHLH	H	X
C	L	L	H	L	XXX	H	LLLLHLHH	H	X
C	L	L	H	L	XXX	L	LLLHLHHL	H	X
C	L	L	H	L	XXX	H	LLHLHHLH	H	X
C	L	L	H	L	XXX	H	LHLHHLHH	H	X
C	L	L	H	L	XXX	H	HLHHLHHH	H	X
C	L	L	H	L	XXX	L	LHHLHLHL	H	X
C	L	L	H	L	XXX	L	HHLHLHLL	H	X
C	L	L	H	L	XXX	L	HLHLHHLL	H	X
C	L	L	H	L	XXX	H	LHLHHHLH	H	X
C	L	L	H	L	XXX	L	HLHHHLHL	H	X
C	L	L	H	L	XXX	L	LHHHLLLL	H	X
C	L	L	H	L	XXX	H	HHHLLLLH	H	X
C	L	L	H	L	HHH	H	HHLLLHHH	H	H
C	L	L	L	X	XXX	X	HLLLHHHX	H	X
C	L	L	L	X	XXX	X	LLLHHHXX	L	X

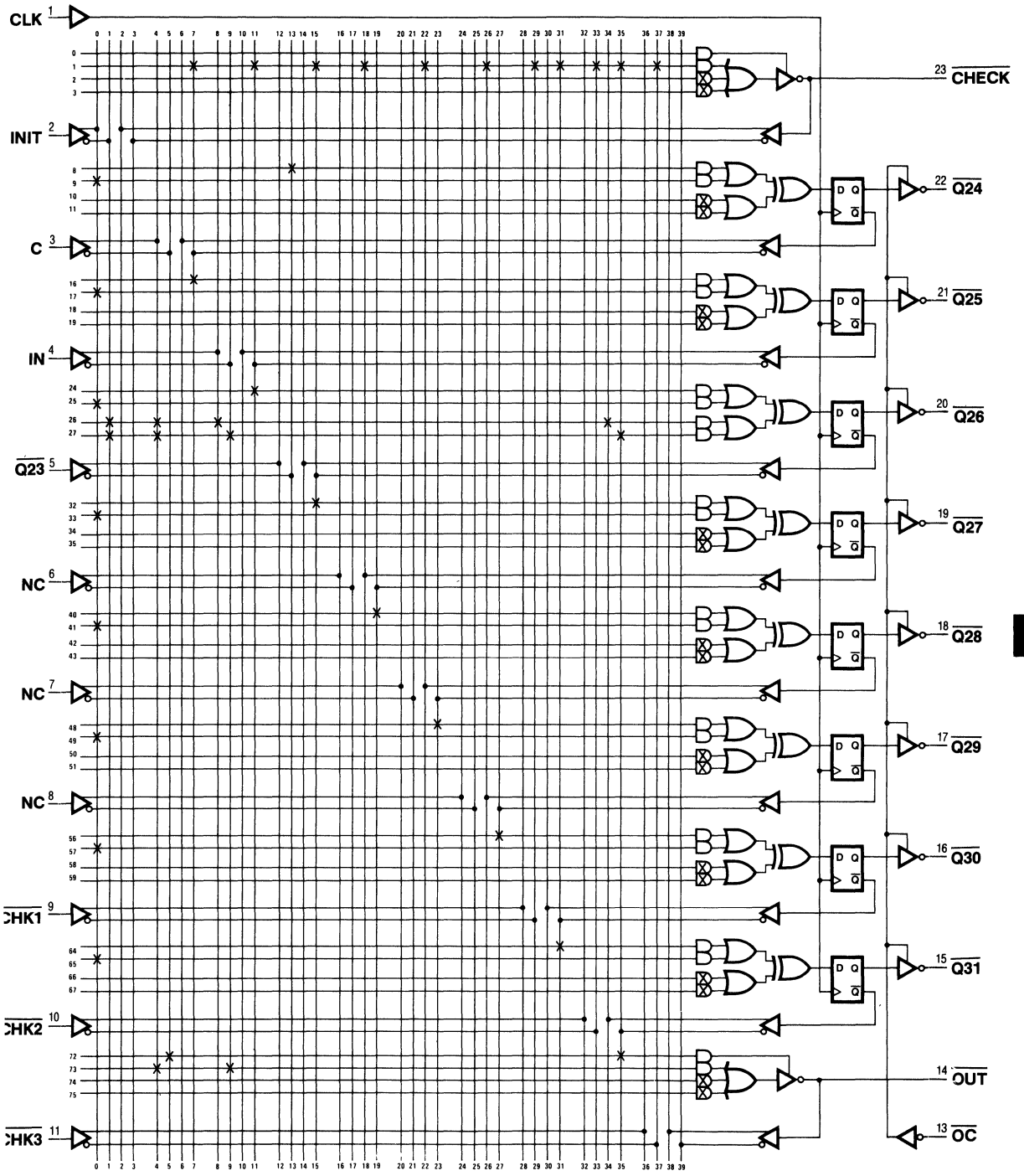
DESCRIPTION

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



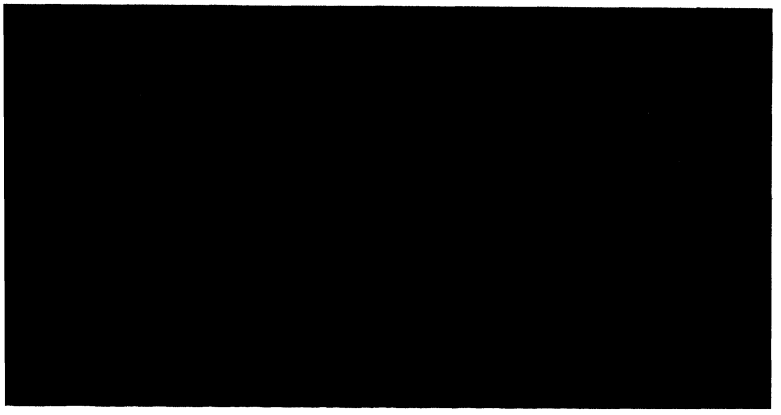
32-Bit CRC, Chip 4

Logic Diagram PAL20X8

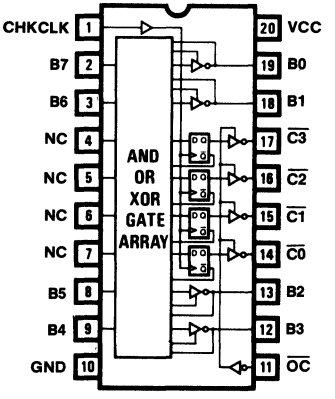



IMPORTANT
NOTES

8-Bit Error Detection and Correction

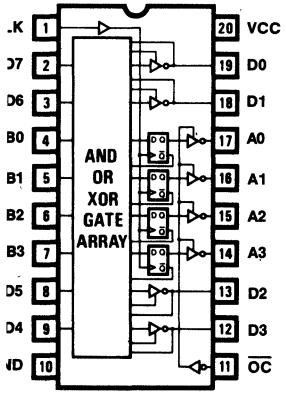


PAL16X4

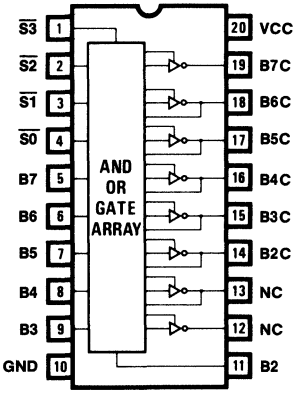


6

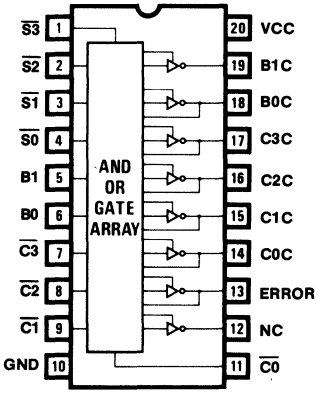
PAL16X4



PAL16L8



PAL16L8



Single bit error detection and correction for an 8-bit data word requires 4 check bits, making a 12-bit code word. The simplest code to design is a 12-bit Hamming code. To arrive at the code, we set up the following matrix:

	B7	B6	B5	B4	B3	B2	B1	B0	C3	C2	C1	C0
S3	X	X	X	X					X			
S2	X				X	X	X			X		
S1		X	X		X	X		X			X	
S0		X		X	X		X	X				X

The vertical columns are in a counting pattern, excluding the single bit values of 8, 4, 2, 1. The single bit values are assigned to the check bits C3-C0. By reading horizontally across the rows of the matrix, we get the check equations by exclusive OR'ing the data bits with X's in that row and equating that to the check bit with an X in that row:

$$C3 = B7 \oplus B6 \oplus B5 \oplus B4$$
$$C2 = B7 \oplus B3 \oplus B2 \oplus B1$$
$$C1 = B6 \oplus B5 \oplus B3 \oplus B2 \oplus B0$$
$$C0 = B6 \oplus B4 \oplus B3 \oplus B1 \oplus B0$$

The check bits are stored along with the data bits in the following message format:

M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1
B7	B6	B5	B4	C3	B3	B2	B1	C2	B0	C1	C0

When a read occurs, the check bits are recalculated and compared with the stored bits to generate the 4 bit syndrome:

$$S3 = B7 \oplus B6 \oplus B5 \oplus B4 \oplus C3$$
$$S2 = B7 \oplus B3 \oplus B2 \oplus B1 \oplus C2$$
$$S1 = B6 \oplus B5 \oplus B3 \oplus B2 \oplus B0 \oplus C1$$
$$S0 = B6 \oplus B4 \oplus B3 \oplus B1 \oplus B0 \oplus C0$$

The 4 syndrome bits indicate the location of any single bit errors in the 12-bit message format which may than be corrected by inversion.

The Hamming code works by introducing enough other code words to create a difference of exactly 3 bits between legal code words. All other code words are illegal. If, in storage, one bit flips, the result is an illegal word. In addition, there is only one word in the set of legal code words from which it could have come, hence the correction.

The Hamming matrix and resultant sets of check bit and syndrome equations are selected so that when a single bit error occurs, the syndrome gives the position of that bit (either data or check) in the 12-bit message format.

Example **B7** **B0**

random data word: 1 1 0 1 1 1 0 0

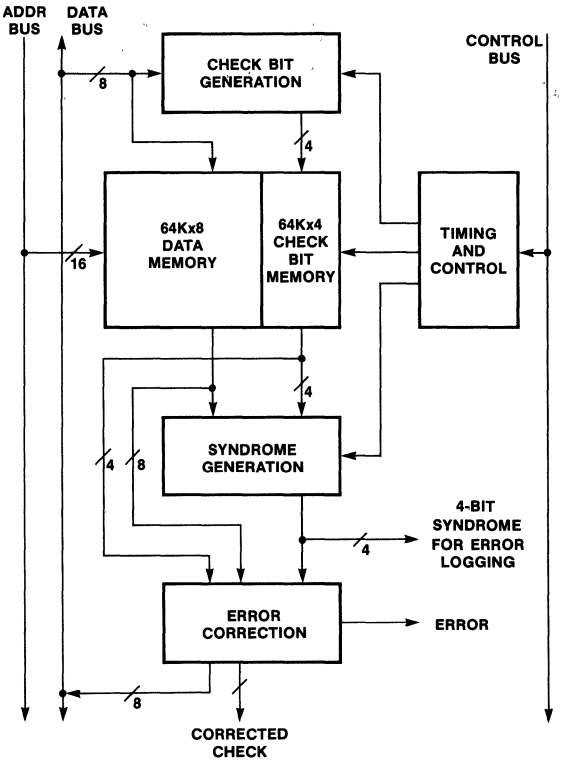
check bits:

$$C3 = B7 \oplus B6 \oplus B5 \oplus B4$$
$$= 1 \oplus 1 \oplus 0 \oplus 1 = 1$$
$$C2 = B7 \oplus B3 \oplus B2 \oplus B1$$
$$= 1 \oplus 1 \oplus 1 \oplus 0 = 1$$
$$C1 = B6 \oplus B5 \oplus B3 \oplus B2 \oplus B0$$
$$= 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 1$$
$$C0 = B6 \oplus B4 \oplus B3 \oplus B1 \oplus B0$$
$$= 1 \oplus 1 \oplus 1 \oplus 0 \oplus 0 = 1$$

message:

M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1
1	1	0	1	1	1	1	0	1	0	1	1
B7	B6	B5	B4	C3	B3	B2	B1	C2	B0	C1	C0

EDAC System Block Diagram



assume no error:

$S3 = B7 \oplus B6 \oplus B5 \oplus B4 \oplus C3$
$= 1 \oplus 1 \oplus 0 \oplus 1 \oplus 1 = 0$

$S2 = B7 \oplus B3 \oplus B2 \oplus B1 \oplus C2$
$= 1 \oplus 1 \oplus 1 \oplus 0 \oplus 1 = 0$

$S1 = B6 \oplus B5 \oplus B3 \oplus B2 \oplus B0 \oplus C1$
$= 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 \oplus 1 = 0$

$S0 = B6 \oplus B4 \oplus B3 \oplus B1 \oplus B0 \oplus C0$
$= 1 \oplus 1 \oplus 1 \oplus 0 \oplus 0 \oplus 1 = 0$

syndrome 0000 indicates no error

assume data bit B7 flips (1 → 0):

$S3 = \underline{0} \oplus 1 \oplus 0 \oplus 1 \oplus 1 = 1$
$S2 = \underline{0} \oplus 1 \oplus 1 \oplus 0 \oplus 1 = 1$
$S1 = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 \oplus 1 = 0$
$S0 = 1 \oplus 1 \oplus 1 \oplus 0 \oplus 0 \oplus 1 = 0$

syndrome 1100 indicates M12 or B7 is in error

assume check bit C3 flips (1 → 0):

$S3 = 1 \oplus 1 \oplus 0 \oplus 1 \oplus \underline{0} = 1$
$S2 = 1 \oplus 1 \oplus 1 \oplus 0 \oplus 1 = 0$
$S1 = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 \oplus 1 = 0$
$S0 = 1 \oplus 1 \oplus 1 \oplus 0 \oplus 0 \oplus 1 = 0$

syndrome 1000 indicates M8 or C3 is in error

While the check bits can be generated with a 256x4 PROM if the tolerances are loose, high performance systems will need to latch or register the check bits to meet cycle time requirements. Similarly, the syndrome bits can be generated with a 4096x4 PROM but in both cases the PAL16X4 is the high performance choice.

The worst case equations are S1 and S0 with a 6 term exclusive OR. We use 2 properties of the exclusive OR to fit the equations into the 16X4:

1. Associativity
   $A \oplus B \oplus C = (A \oplus B) \oplus C$
2. $A \oplus B \oplus C = A\overline{B}\,\overline{C} + \overline{A}B\overline{C} + \overline{A}\,\overline{B}C + ABC$

Since a 3 term exclusive OR can be realized with a 4 product sum in sum of products form, a 6 term exclusive OR can be realized by exclusive OR of 2 4 product sums. This is exactly the 16X4 configuration.

Check Bit Equations

$C3 = B7 \oplus B6 \oplus B5 \oplus B4$
$= (B7 \cdot \overline{B6} + \overline{B7} \cdot B6) \oplus (B5 \cdot \overline{B4} + \overline{B5} \cdot B4)$

$C2 = B7 \oplus B3 \oplus B2 \oplus B1$
$= (B7 \cdot \overline{B3} + \overline{B7} \cdot B3) \oplus (B2 \cdot \overline{B1} + \overline{B2} \cdot B1)$

$C1 = B6 \oplus B5 \oplus B3 \oplus B2 \oplus B0$
$= (B6 \cdot B5 \cdot B3 + \overline{B6} \cdot \overline{B5} \cdot B3 + \overline{B6} \cdot B5 \cdot \overline{B3} + B6 \cdot \overline{B5} \cdot \overline{B3}) \oplus (B2 \cdot \overline{B0} + \overline{B2} \cdot \quad)$

$C0 = B6 \oplus B4 \oplus B3 \oplus B1 \oplus B0$
$= (B6 \cdot B4 \cdot B3 + \overline{B6} \cdot \overline{B4} \cdot B3 + \overline{B6} \cdot B4 \cdot \overline{B3} + B6 \cdot \overline{B4} \cdot \overline{B3}) \oplus (B1 \cdot \overline{B0} + \overline{B1} \cdot B0)$

Syndrome Bit Equations

$S3 = B7 \oplus B6 \oplus B5 \oplus B4 \oplus C3$
$= (B7 \cdot B6 \cdot B5 + \overline{B7} \cdot \overline{B6} \cdot B5 + \overline{B7} \cdot B6 \cdot \overline{B5} + B7 \cdot \overline{B6} \cdot \overline{B5}) \oplus (B4 \cdot \overline{C3} + \overline{B4} \cdot C3)$

$S2 = B7 \oplus B3 \oplus B2 \oplus B1 \oplus C2$
$= (B7 \cdot B3 \cdot B2 + \overline{B7} \cdot \overline{B3} \cdot B2 + \overline{B7} \cdot B3 \cdot \overline{B2} + B7 \cdot \overline{B3} \cdot \overline{B2}) \oplus (B1 \cdot \overline{C2} + \overline{B1} \cdot C2)$

$S1 = B6 \oplus B5 \oplus B3 \oplus B2 \oplus B0 \oplus C1$
$= (B6 \cdot B5 \cdot B3 + \overline{B6} \cdot \overline{B5} \cdot B3 + \overline{B6} \cdot B5 \cdot \overline{B3} + B6 \cdot \overline{B5} \cdot \overline{B3}) \oplus (\overline{B2} \cdot \overline{B0} \cdot C1 + \overline{B2} \cdot \overline{B0} \cdot C1 + B2 \cdot B0 \cdot \overline{C1} + B2 \cdot \overline{B0}$

$S0 = B6 \oplus B4 \oplus B3 \oplus B1 \oplus B0 \oplus C0$
$= (B6 \cdot B4 \cdot B3 + \overline{B6} \cdot \overline{B4} \cdot B3 + \overline{B6} \cdot B4 \cdot \overline{B3} + B6 \cdot \overline{B4} \cdot \overline{B3}) \oplus (B1 \cdot B0 \cdot C0 + \overline{B1} \cdot \overline{B0} \cdot C0 + \overline{B1} \cdot B0 \cdot \overline{C0} + B1 \cdot \overline{B0} \quad)$

The error correction block decodes the 4 syndrome bits, and, if they are not 0000, inverts the indicated bit in the message format. The equations:

$M12\ (= B7C) = S3 \cdot S2 \cdot \overline{S1} \cdot \overline{S0} \oplus B7$
$= S3 \cdot S2 \cdot \overline{S1} \cdot \overline{S0} \cdot \overline{B7} + \overline{(S3 \cdot S2 \cdot \overline{S1} \cdot \overline{S0})} \cdot B7$
$= S3 \cdot S2 \cdot \overline{S1} \cdot \overline{S0} \cdot \overline{B7} + \overline{S3} \cdot B7 + S2 \cdot B7 + S1 \cdot B7 + \overline{S0} \cdot B7$

$M11\ (= B6C) = S3 \cdot \overline{S2} \cdot S1 \cdot S0 \cdot \overline{B6} \cdot \overline{S3} \cdot B6 \cdot S2 \cdot B6 + \overline{S1} \cdot B6 \cdot \overline{S0} \cdot B6$

$M10\ (= B5C) = S3 \cdot \overline{S2} \cdot S1 \cdot \overline{S0} \cdot \overline{B5} + \overline{S3} \cdot B5 + S2 \cdot B5 + \overline{S1} \cdot B5 + S0 \cdot B5$

$M9\ (= B4C) = S3 \cdot \overline{S2} \cdot \overline{S1} \cdot S0 \cdot \overline{B4} + \overline{S3} \cdot B4 + S2 \cdot B4 + S1 \cdot B4 + \overline{S0} \cdot B4$

$M8\ (= C3C) = S3 \cdot \overline{S2} \cdot \overline{S1} \cdot \overline{S0} \cdot \overline{C3} + \overline{S3} \cdot C3 + S2 \cdot C3 + S1 \cdot C3 + S0 \cdot C3$

$M7\ (= B3C) = \overline{S3} \cdot S2 \cdot S1 \cdot S0 \cdot \overline{B3} + S3 \cdot B3 + \overline{S2} \cdot B3 + \overline{S1} \cdot B3 + \overline{S0} \cdot B3$

$M6\ (= B2C) = \overline{S3} \cdot S2 \cdot S1 \cdot \overline{S0} \cdot \overline{B2} + S3 \cdot B2 + \overline{S2} \cdot B2 + \overline{S1} \cdot B2 + S0 \cdot B2$

$M5\ (= B1C) = \overline{S3} \cdot S2 \cdot \overline{S1} \cdot S0 \cdot \overline{B1} + S3 \cdot B1 + \overline{S2} \cdot B1 + S1 \cdot B1 + \overline{S0} \cdot B1$

$M4\ (= C2C) = \overline{S3} \cdot S2 \cdot \overline{S1} \cdot \overline{S0} \cdot \overline{C2} + S3 \cdot C2 + \overline{S2} \cdot C2 + S1 \cdot C2 + S0 \cdot C2$

$M3\ (= B0C) = \overline{S3} \cdot \overline{S2} \cdot S1 \cdot S0 \cdot \overline{B0} + S3 \cdot B0 + S2 \cdot B0 + \overline{S1} \cdot B0 + \overline{S0} \cdot B0$

$M2\ (= C1C) = \overline{S3} \cdot \overline{S2} \cdot S1 \cdot \overline{S0} \cdot \overline{C1} + S3 \cdot C1 + S2 \cdot C1 + \overline{S1} \cdot C1 + S0 \cdot C1$

$M1\ (= C0C) = \overline{S3} \cdot \overline{S2} \cdot \overline{S1} \cdot S0 \cdot \overline{C0} + S3 \cdot C0 + S2 \cdot C0 + S1 \cdot C0 + \overline{S0} \cdot C0$

$ERROR = \overline{S3 \cdot S2 \cdot \overline{S1} \cdot \overline{S0}}$

ERROR is an active high error indicator available for error logging along with the 4 syndrome bits.

To use 2 PAL16L8's for the error correction block, we need only invert the message bits to get active true outputs.

```
PAL16X4                                          PAL DESIGN SPECIFICATION
CBG                                                   B. BRAFMAN 02/16/81
CHECK BIT GENERATOR
MMI FIELD APPLICATIONS ENGINEER YORBA LINDA, CALIFORNIA
CHKCLK B7 B6 NC NC NC NC B5 B4 GND
/OC B3 B2 /C0 /C1 /C2 /C3 B1 B0 VCC


C3 :=   B7*/B6              ;B7 :+: B6
    +  /B7* B6              ;   :+:
    +   GND                 ;DO NOT BLOW THIS PRODUCT LINE
    +   GND                 ;DO NOT BLOW THIS PRODUCT LINE
   :+:  B5*/B4              ;B5 :+: B4
    +  /B5* B4

C2 :=   B7*/B3              ;B7 :+: B3
    +  /B7* B3              ;   :+:
    +   GND                 ;DO NOT BLOW THIS PRODUCT LINE
    +   GND                 ;DO NOT BLOW THIS PRODUCT LINE
   :+:  B2*/B1              ;B2 :+: B1
    +  /B2* B1

C1 :=   B6* B5* B3          ;B6 :+: B5 :+: B3
    +  /B6*/B5* B3
    +  /B6* B5*/B3          ;   :+:
    +   B6*/B5*/B3
   :+:  B2*/B0              ;B2 :+: B0
    +  /B2* B0

C0 :=   B6* B4* B3          ;B6 :+: B4 :+: B3
    +  /B6*/B4* B3
    +  /B6* B4*/B3          ;   :+:
    +   B6*/B4*/B3
   :+:  B1*/B0              ;B1 :+: B0
    +  /B1* B0
```

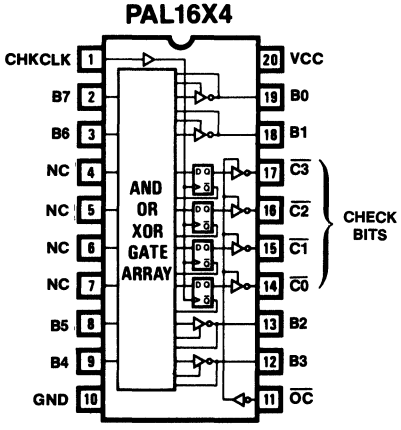
FUNCTION TABLE

CHKCLK /OC B7 B6 B5 B4 B3 B2 B1 B0 C3 C2 C1 C0

;CONTROL ;CHK	/ OC	8 BIT IN BBBBBBBB 76543210	CORR CCCC 3210	COMMENTS
C	L	HHHHHHHH	LLHH	ALL ONES DATA
C	L	LHHHHHHH	HHHH	SHIFT A ZERO ACROSS
C	L	HLHHHHHH	HLLL	
C	L	HHLHHHHH	HLLH	
C	L	HHHLHHHH	HLHL	
C	L	HHHHLHHH	LHLL	
C	L	HHHHHLHH	LHLH	
C	L	HHHHHHLH	LHHL	
C	L	HHHHHHHL	LLLL	
C	L	LLLLLLLL	LLLL	ALL ZEROS DATA
C	L	HLLLLLLL	HHLL	SHIFT A ONE ACROSS
C	L	LHLLLLLL	HLHH	
C	L	LLHLLLLL	HLHL	
C	L	LLLHLLLL	HLLH	
C	L	LLLLHLLL	LHHH	
C	L	LLLLLHLL	LHHL	
C	L	LLLLLLHL	LHLH	
C	L	LLLLLLLH	LLHH	

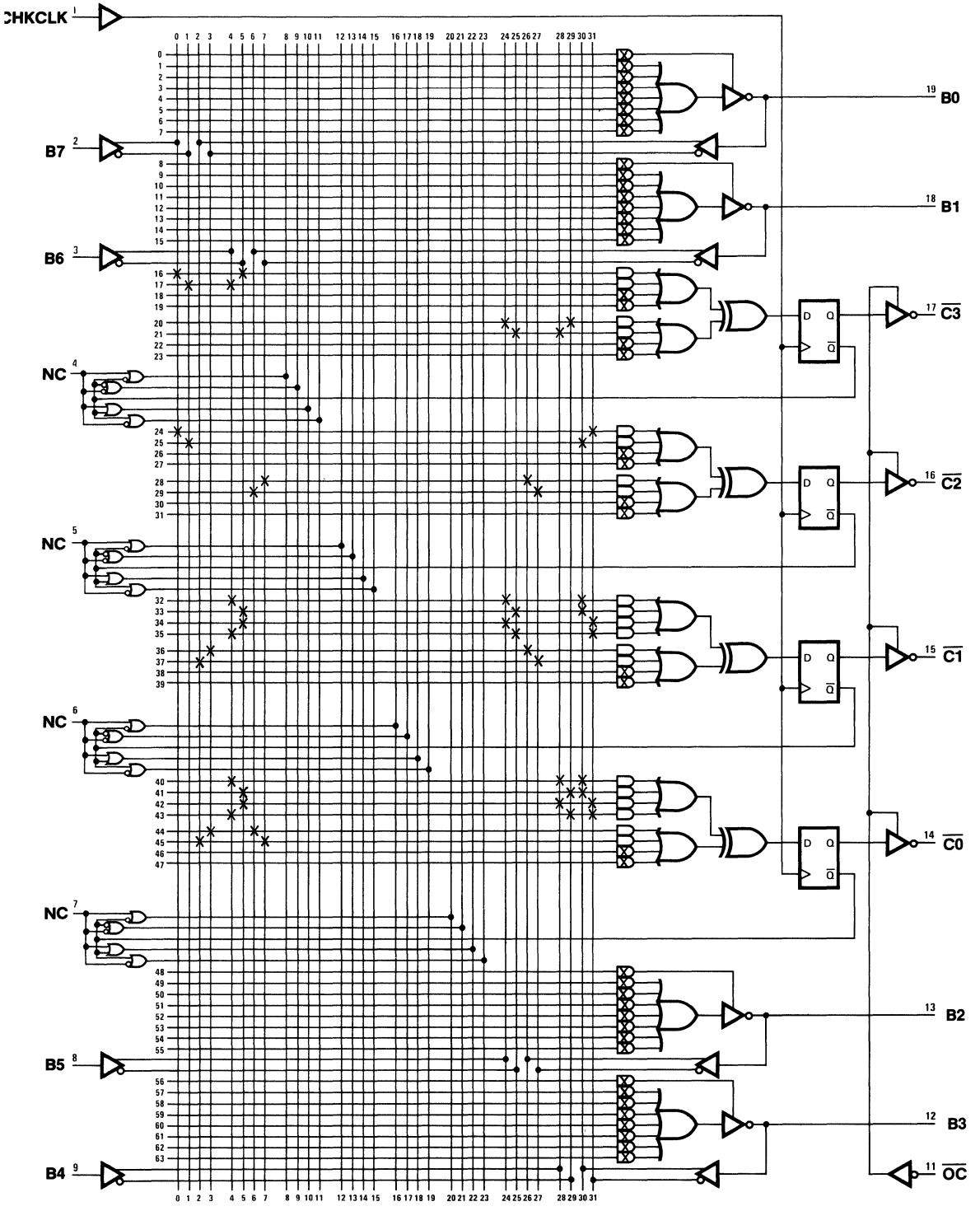
DESCRIPTION

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



Check Bit Generator

Logic Diagram PAL16X4



```
PAL16X4                                                PAL DESIGN SPECIFICATION
SBG                                                        B. BRAFMAN 03/13/81
SYNDROME BIT GENERATOR
MMI FIELD APPLICATIONS ENGINEER YORBA LINDA, CALIFORNIA
SYNCLK D7 D6 B0 B1 B2 B3 D5 D4 GND
/OC D3 D2 A3 A2 A1 A0 D1 D0 VCC


;    IN THE ABOVE PIN LIST, THE FOLLOWING SUBSTITUTIONS HAVE BEEN
;    MADE TO ACCOMODATE THE SPECIFIC FORMAT (FIXED SYMBOLS) FOR THE
;    ARITHMETIC PALS IN PALASM:

;         D7 MEANS B7        B0 MEANS /C3  (CHECK BIT 3)
;         D6 MEANS B6        B1 MEANS /C2  (CHECK BIT 2)
;         D5 MEANS B5        B2 MEANS /C1  (CHECK BIT 1)
;         D4 MEANS B4        B3 MEANS /C0  (CHECK BIT 0)
;         D3 MEANS B3        A0 MEANS /S3  (SYNDROME BIT 3)
;         D2 MEANS B2        A1 MEANS /S2  (SYNDROME BIT 2)
;         D1 MEANS B1        A2 MEANS /S1  (SYNDROME BIT 1)
;         D0 MEANS B0        A3 MEANS /S0  (SYNDROME BIT 0)

;  B0-B7 ARE THE BITS OF THE DATA WORD.
;  THE SUBSTITUTIONS APPLY BELOW WITH THE EXCEPTION OF COMMENTS.


/A0 :=    D7* D6* D5              ;B7 :+: B6 :+: B5
     +   /D7*/D6* D5
     +   /D7* D6*/D5              ;   :+:
     +    D7*/D6*/D5
    :+:   D4*(/B0)                ;B4 :+: C3
     +   /D4*( B0)

/A1 :=    D7* D3* D2              ;B7 :+: B3 :+: B2
     +   /D7*/D3* D2
     +   /D7* D3*/D2              ;   :+:
     +    D7*/D3*/D2
    :+:   D1*(/B1)                ;B1 :+: C2
     +   /D1*( B1)

/A2 :=    D6* D5* D3              ;B6 :+: B5 :+: B3
     +   /D6*/D5* D3
     +   /D6* D5*/D3              ;   :+:
     +    D6*/D5*/D3
    :+:   D2* D0*( B2)            ;B2 :+: B0 :+: C1
     +   /D2*/D0*( B2)
     +   /D2* D0*(/B2)
     +    D2*/D0*(/B2)

/A3 :=    D6* D4* D3              ;B6 :+: B4 :+: B3
     +   /D6*/D4* D3
     +   /D6* D4*/D3              ;   :+:
     +    D6*/D4*/D3
    :+:   D1* D0*( B3)            ;B1 :+: B0 :+: C0
     +   /D1*/D0*( B3)
     +   /D1* D0*(/B3)
     +    D1*/D0*(/B3)
```

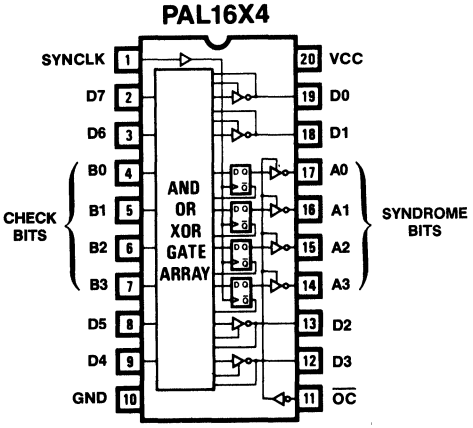
FUNCTION TABLE

SYNCLK /OC D7 D6 D5 D4 D3 D2 D1 D0 B0 B1 B2 B3 /A0 /A1 /A2 /A3

;CONTROL		--DATA-		////	
;SYN	/	DDDDDDDD	BBBB	AAAA	
;CLK	OC	76543210	0123	0123	COMMENTS
C	L	HHHHHHHH	LLHH	LLLL	NO ERROR
C	L	LHHHHHHH	LLHH	HHLL	D7 ERROR
C	L	HLHHHHHH	LLHH	HLHH	D6 ERROR
C	L	HHLHHHHH	LLHH	HLHL	D5 ERROR
C	L	HHHLHHHH	LLHH	HLLH	D4 ERROR
C	L	HHHHLHHH	LLHH	LHHH	D3 ERROR
C	L	HHHHHLHH	LLHH	LHHL	D2 ERROR
C	L	HHHHHHLH	LLHH	LHLH	D1 ERROR
C	L	HHHHHHHL	LLHH	LLHH	D0 ERROR
C	L	HHHHHHHH	HLHH	HLLL	B0 ERROR
C	L	HHHHHHHH	LLHH	LLLL	NO ERROR
C	L	HHHHHHHH	LLLH	LLHL	B2 ERROR
C	L	HHHHHHHH	LLHL	LLLH	B1 ERROR
C	L	LLLLLLLL	LLLL	LLLL	NO ERROR
C	L	HLLLLLLL	LLLL	HHLL	D7 ERROR
C	L	LHLLLLLL	LLLL	HLHH	D6 ERROR
C	L	LLHLLLLL	LLLL	HLHL	D5 ERROR
C	L	LLLHLLLL	LLLL	HLLH	D4 ERROR
C	L	LLLLHLLL	LLLL	LHHH	D3 ERROR
C	L	LLLLLHLL	LLLL	LHHL	D2 ERROR
C	L	LLLLLLHL	LLLL	LHLH	D1 ERROR
C	L	LLLLLLLH	LLLL	LLHH	D0 ERROR
C	L	LLLLLLLL	HLLL	HLLL	B0 ERROR
C	L	LLLLLLLL	LHLL	LHLL	B1 ERROR
C	L	LLLLLLLL	LLHL	LLHL	B2 ERROR
C	L	LLLLLLLL	LLLH	LLLH	B3 ERROR

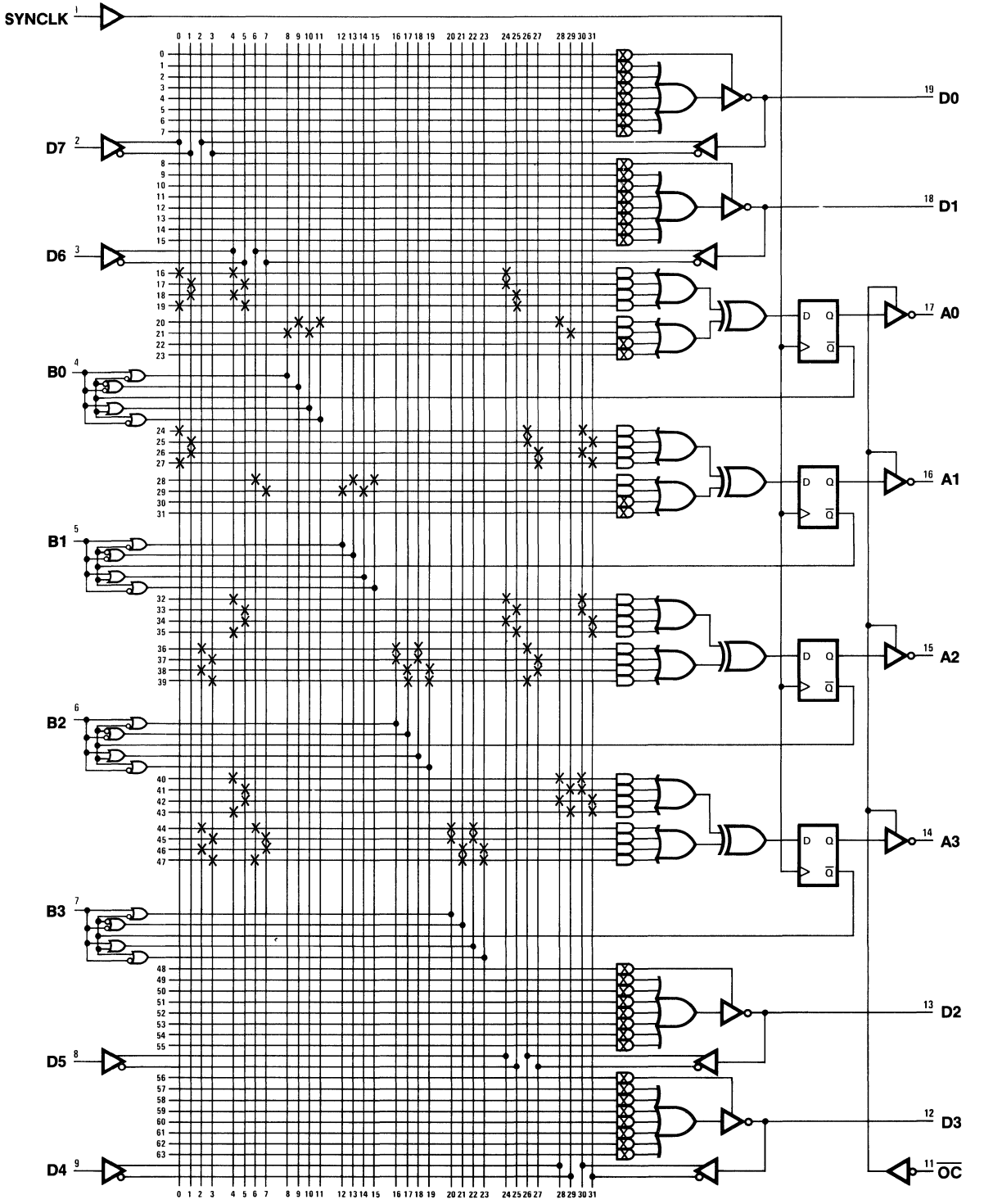
DESCRIPTION

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



Syndrome Bit Generator

Logic Diagram PAL16X4

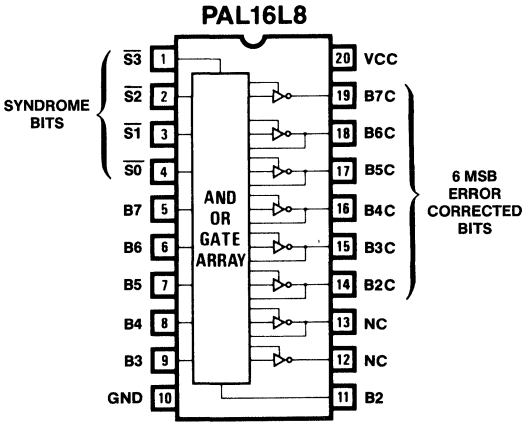


```
PAL16L8                                          PAL DESIGN SPECIFICATION
ECU1                                                  B. BRAFMAN 03/13/81
ERROR CORRECTION UNIT No. 1
MMI FIELD APPLICATIONS ENGINEER YORBA LINDA, CALIFORNIA
/S3 /S2 /S1 /S0 B7  B6  B5  B4  B3  GND
 B2  NC  NC B2C B3C B4C B5C B6C B7C VCC


IF (VCC) /B7C =  S3* S2*/S1*/S0* B7      ;CORRECTION OF B7
              + /S3*/B7                  ;NO CORRECTION
              + /S2*/B7                  ;NO CORRECTION
              +  S1*/B7                  ;NO CORRECTION
              +  S0*/B7                  ;NO CORRECTION

IF (VCC) /B6C =  S3*/S2* S1* S0* B6      ;CORRECTION OF B6
              + /S3*/B6                  ;NO CORRECTION
              +  S2*/B6                  ;NO CORRECTION
              + /S1*/B6                  ;NO CORRECTION
              + /S0*/B6                  ;NO CORRECTION

IF (VCC) /B5C =  S3*/S2* S1*/S0* B5      ;CORRECTION OF B5
              + /S3*/B5                  ;NO CORRECTION
              +  S2*/B5                  ;NO CORRECTION
              + /S1*/B5                  ;NO CORRECTION
              +  S0*/B5                  ;NO CORRECTION

IF (VCC) /B4C =  S3*/S2*/S1* S0* B4      ;CORRECTION OF B4
              + /S3*/B4                  ;NO CORRECTION
              +  S2*/B4                  ;NO CORRECTION
              +  S1*/B4                  ;NO CORRECTION
              + /S0*/B4                  ;NO CORRECTION

IF (VCC) /B3C = /S3* S2* S1* S0* B3      ;CORRECTION OF B3
              +  S3*/B3                  ;NO CORRECTION
              + /S2*/B3                  ;NO CORRECTION
              + /S1*/B3                  ;NO CORRECTION
              + /S0*/B3                  ;NO CORRECTION

IF (VCC) /B2C = /S3* S2* S1*/S0* B2      ;CORRECTION OF B2
              +  S3*/B2                  ;NO CORRECTION
              + /S2*/B2                  ;NO CORRECTION
              + /S1*/B2                  ;NO CORRECTION
              +  S0*/B2                  ;NO CORRECTION
```

```
FUNCTION TABLE

S3 S2 S1 S0 B7 B6 B5 B4 B3 B2 B7C B6C B5C B4C B3C B2C

;SYNDROME     INPUT      CORRECTED
;  SSSS       BBBBBB       BBBBBB
;  3210       765432       765432          COMMENTS
-------------------------------------------------
   LLLL       HHHHHH       HHHHHH          NO ERROR
   LLLL       LLLLLL       LLLLLL          NO ERROR
   HHLL       HHHHHH       LHHHHH          CORRECT B7
   HHLL       LLLLLL       HLLLLL          CORRECT B7
   HLHH       HHHHHH       HLHHHH          CORRECT B6
   HLHH       LLLLLL       LHLLLL          CORRECT B6
   HLHL       HHHHHH       HHLHHH          CORRECT B5
   HLHL       LLLLLL       LLHLLL          CORRECT B5
   HLLH       HHHHHH       HHHLHH          CORRECT B4
   HLLH       LLLLLL       LLLHLL          CORRECT B4
   LHHH       HHHHHH       HHHHLH          CORRECT B3
   LHHH       LLLLLL       LLLLHL          CORRECT B3
   LHHL       HHHHHH       HHHHHL          CORRECT B2
   LHHL       LLLLLL       LLLLLH          CORRECT B2
-------------------------------------------------
```

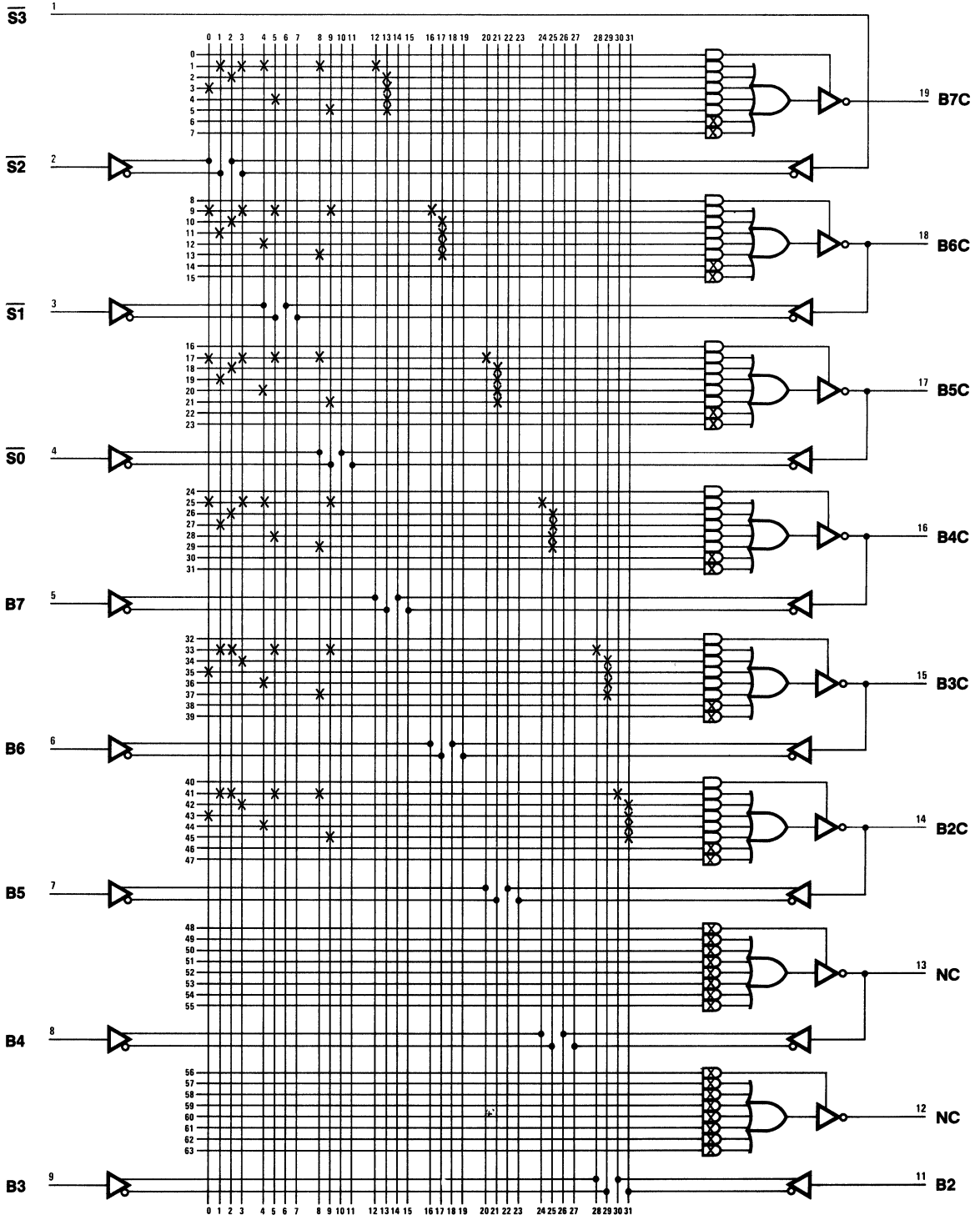
DESCRIPTION

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



Error Correction Unit No. 1

Logic Diagram PAL16L8

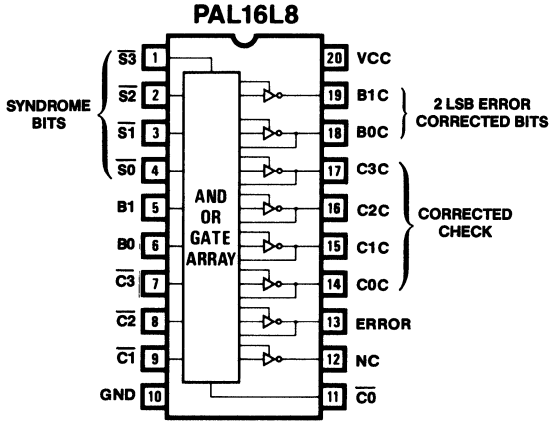


```
PAL16L8                                          PAL DESIGN SPECIFICATION
ECU2                                                  B. BRAFMAN 03/13/81
ERROR CORRECTION UNIT No. 2
MMI FIELD APPLICATIONS ENGINEER  YORBA LINDA, CALIFORNIA
/S3 /S2 /S1 /S0 B1 B0 /C3 /C2 /C1 GND
/C0 NC ERROR C0C C1C C2C C3C B0C B1C VCC


IF (VCC) /B1C = /S3* S2*/S1* S0* B1     ;CORRECTION OF B1
               +  S3*/B1                ;NO CORRECTION
               + /S2*/B1                ;NO CORRECTION
               +  S1*/B1                ;NO CORRECTION

IF (VCC) /B0C = /S3*/S2* S1* S0* B0     ;CORRECTION OF B0
               +  S3*/B0                ;NO CORRECTION
               +  S2*/B0                ;NO CORRECTION
               + /S1*/B0                ;NO CORRECTION
               + /S0*/B0                ;NO CORRECTION

IF (VCC) /C3C =  S3*/S2*/S1*/S0* C3     ;CORRECTION OF C3
               + /S3*/C3                ;NO CORRECTION
               +  S2*/C3                ;NO CORRECTION
               +  S1*/C3                ;NO CORRECTION
               +  S0*/C3                ;NO CORRECTION

IF (VCC) /C2C = /S3* S2*/S1*/S0* C2     ;CORRECTION OF C2
               +  S3*/C2                ;NO CORRECTION
               + /S2*/C2                ;NO CORRECTION
               +  S1*/C2                ;NO CORRECTION
               +  S0*/C2                ;NO CORRECTION

IF (VCC) /C1C = /S3*/S2* S1*/S0* C1     ;CORRECTION OF C1
               +  S3*/C1                ;NO CORRECTION
               +  S2*/C1                ;NO CORRECTION
               + /S1*/C1                ;NO CORRECTION
               +  S0*/C1                ;NO CORRECTION

IF (VCC) /C0C = /S3*/S2*/S1* S0* C0     ;CORRECTION OF C0
               +  S3*/C0                ;NO CORRECTION
               +  S2*/C0                ;NO CORRECTION
               +  S1*/C0                ;NO CORRECTION
               + /S0*/C0                ;NO CORRECTION

IF (VCC) /ERROR = /S3*/S2*/S1*/S0       ;NO ERROR!
```

```
FUNCTION TABLE

S3 S2 S1 S0 B1 B0 C3 C2 C1 C0 B1C B0C C3C C2C C1C C0C ERROR

;                    BB   CCCC
;SSSS   BB   CCCC    10   3210
;3210   10   3210    CC   CCCC   ERROR      COMMENTS
-----------------------------------------------------
 LLLL   HH   XXXX    HH   XXXX     L        NO ERROR
 LLLL   XX   HHHH    XX   HHHH     L        NO ERROR
 LLLL   LL   XXXX    LL   XXXX     L        NO ERROR
 LLLL   XX   LLLL    XX   LLLL     L        NO ERROR
 LHLH   HH   XXXX    LH   XXXX     H        CORRECT B1
 LHLH   LL   XXXX    HL   XXXX     H        CORRECT B1
 LLHH   HH   XXXX    HL   XXXX     H        CORRECT B0
 LLHH   LL   XXXX    LH   XXXX     H        CORRECT B0
 HLLL   XX   HHHH    XX   LHHH     H        CORRECT C3
 HLLL   XX   LLLL    XX   HLLL     H        CORRECT C3
 LHLL   XX   HHHH    XX   HLHH     H        CORRECT C2
 LHLL   XX   LLLL    XX   LHLL     H        CORRECT C2
 LLHL   XX   HHHH    XX   HHLH     H        CORRECT C1
 LLHL   XX   LLLL    XX   LLHL     H        CORRECT C1
 LLLH   XX   HHHH    XX   HHHL     H        CORRECT C0
 LLLH   XX   LLLL    XX   LLLH     H        CORRECT C0
-----------------------------------------------------
```

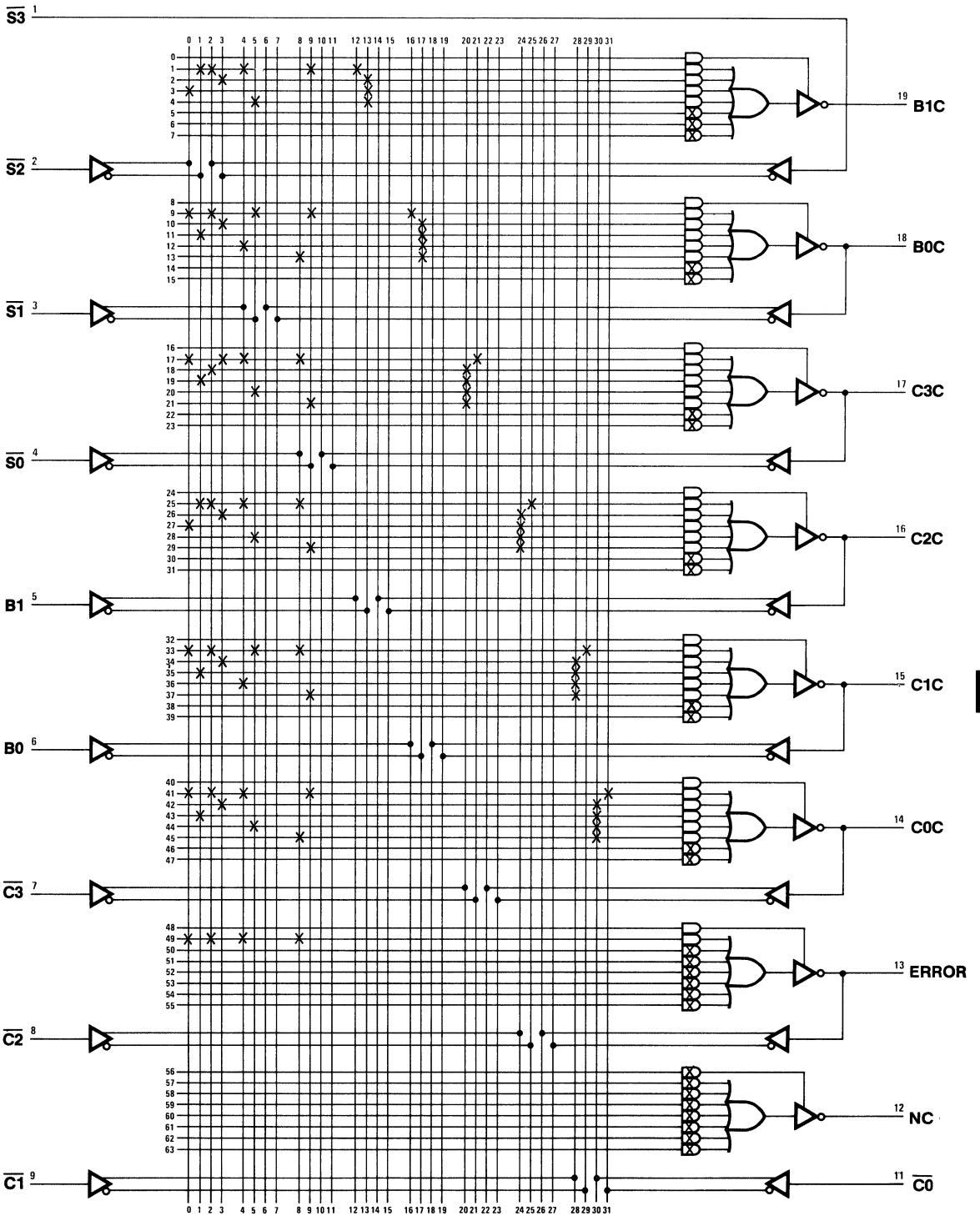
DESCRIPTION

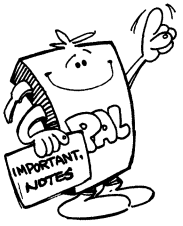
THIS PAL PERFORMS ERROR CORRECTION ON BITS B0-B1 AND CHECKS BITS C0-C3 BASED ON THE 4 BIT ERROR SYNDROME S0-S3.

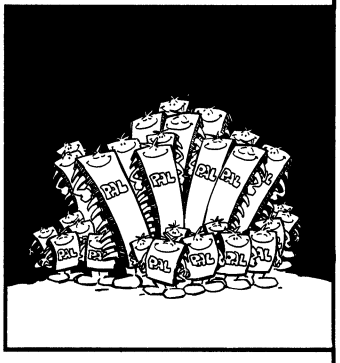


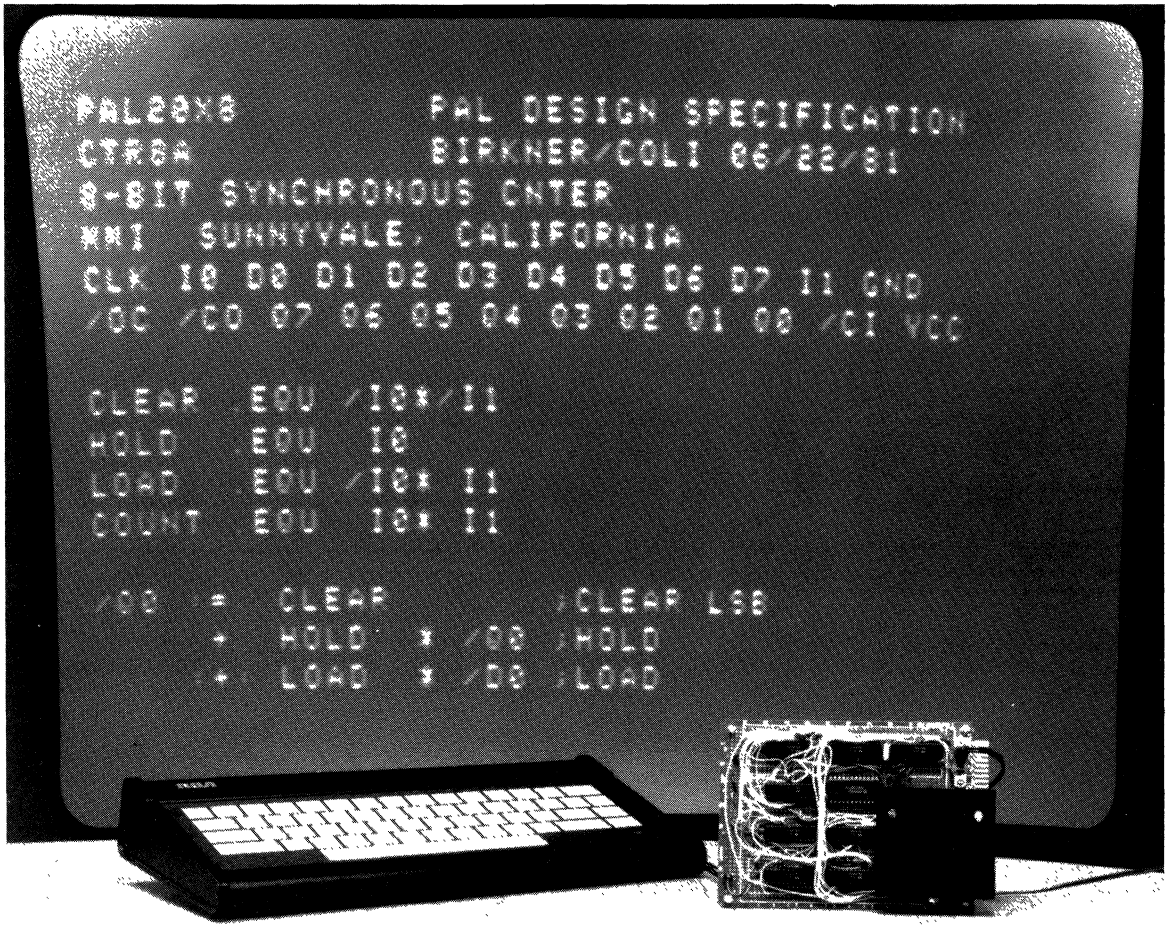
Error Correction Unit No. 2

Logic Diagram PAL16L8




IMPORTANT
NOTES





Introduction to Video Section

Computer graphics is a rapidly growing field in the computer industry. It enables communication between the human being and the computer. The human eye can absorb information displayed in diagrams, numbers, letters and images much faster than it can absorb tables with numbers only. Computer graphics is penetrating into almost every professional field and the "video games" syndrome is spreading into more and more living rooms.

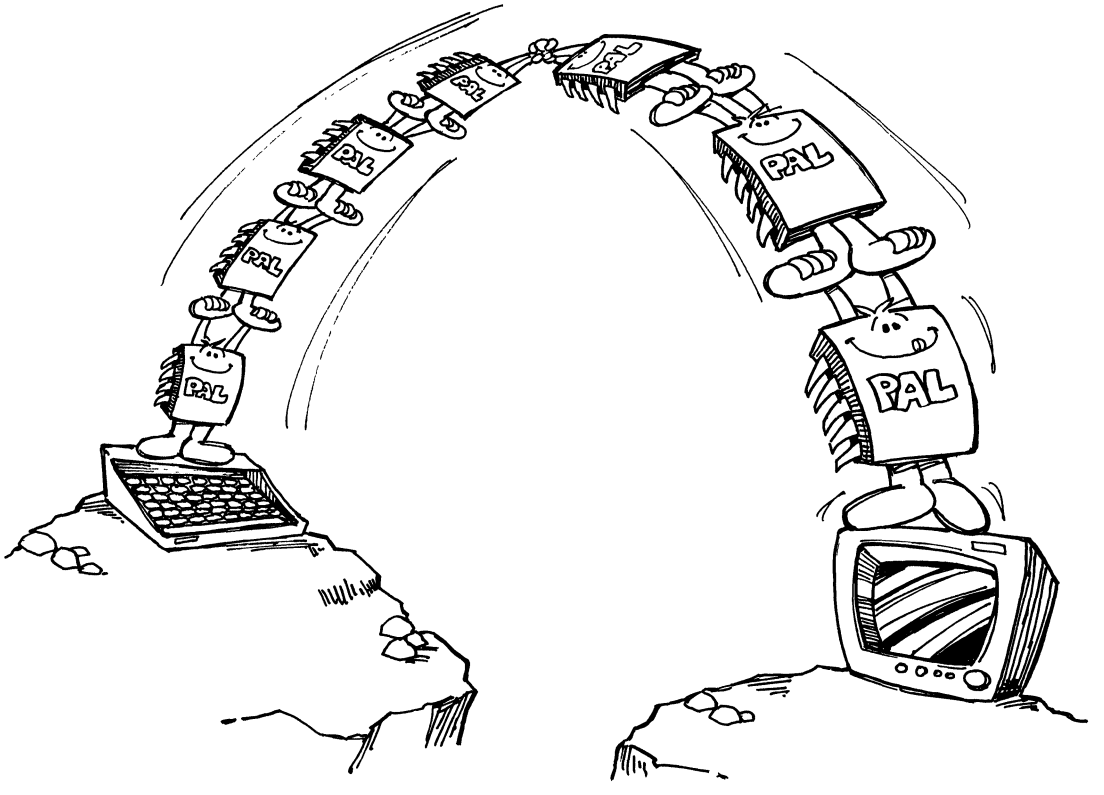
The PAL family teams up to help the wide-spread use of computer graphics. By using high speed PAL devices, the designer can implement simulations and real time systems in an efficient way.

The following designs are examples of the use of PAL devices in the video world.

1. **10 BIT COUNTER [SN54/74LS491]** This is ideal for video synch generation for CRT graphics including video games. This counter provides the vertical and horizontal coordinates required to address the graphic data. Video resolution is usually 9 or 10 bits in the vertical and horizontal which in the past was supplied by three LS161's. These, of course, can be replaced by one LS491. This counter can also count down which allows screen coordinate reversal.
2. Analog to digital and digital to analog converters, change analog signal into an equivalent digital code and vice versa. This function is commonly required in a digital system when the analog information has to be stored and then shown as a displacement on a CRT.
3. The video logic design shows the reduction in the number of parts on a video-interface board using PAL circuits.
4. The video-controller board is an example of how to implement a video system using PAL circuits.

Implementing a Video Controller Using Programmable Array Logic*

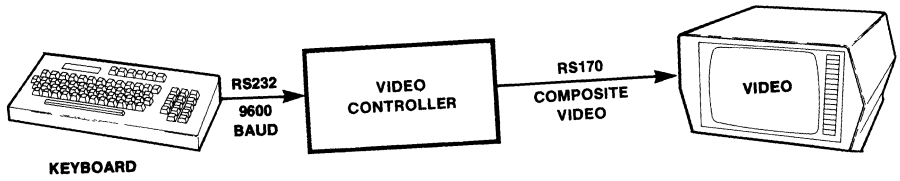
by Ehud "Udi" Gordon



7

The use of video is definitely a sign of the times and will be with us into the future. The video-controller board is the bridge between the outside world and the screen. There are many possibilities to implement the video-controller board, but an efficient one uses PALs. The following will simplify the construction of a video-controller board and will assist the designer in eliminating board space problems. PALs are the backbone of this application.

* An abbreviated version of this paper appeared with the same title in the Conference Proceedings of the 7th West Coast Computer Faire, March 1982.



Overview

There are words that belong to specific concepts within the video industry. For a clear understanding of how the video-controller board works we need to define some of them:

FRAME — A single complete picture

SCANNING — The process of analyzing successively, according to a predetermined method, the light values of picture elements which constitute the total picture area

FRAME FREQUENCY — The number of times per second the picture area is completely scanned

The frame frequency has been standardized at 30 frames per second, which gives a satisfactory illusion of motion continuity, although the individual picture elements are transmitted one after the other. A frequency of 30 frames per second is equal to half the power-line frequency in most locations. Using this standardized rate, the entire screen is illuminated 30 times per second. This is too slow to give the impression of a truly steady light, so a flickering occurs.

To overcome the flickering problem it is common to use the **INTERLACED SCANNING METHOD**. In this method the screen is divided into 525 lines and into two fields. There is an odd numbered field and an even numbered field. Each of these fields is made up of one-half of the screen by dividing the frame into odd and even numbered lines. First, all the odd numbered lines 1, 3, 5, ... are scanned from top to bottom. After scanning a total of 262.5 odd numbered lines, the even numbered lines are scanned from top to bottom, for the total of 525 lines. Each scanning period from top to bottom is 1/60 of a second and is called a **FIELD**. Two fields constitute a frame.

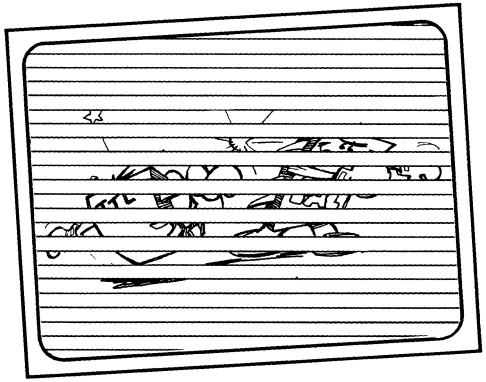


Figure 1a. Odd Numbered Lines Scanned Generate the Odd Field

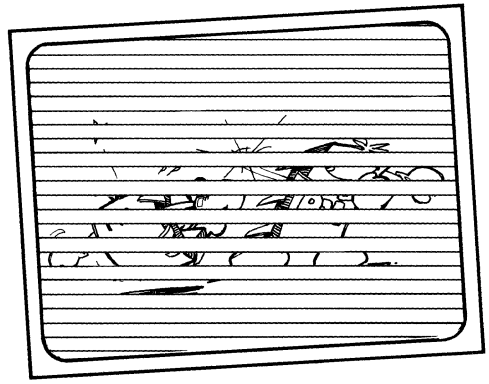


Figure 1b. Even Numbered Lines Scanned Generate the Even Field

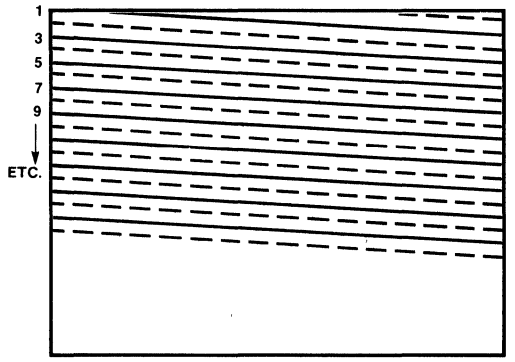


Figure 1c. The Solid Lines 1, 3, 5, . . . Represent the Odd Field. The Dashed Lines 2, 4, 6 . . . Represent the Even Field

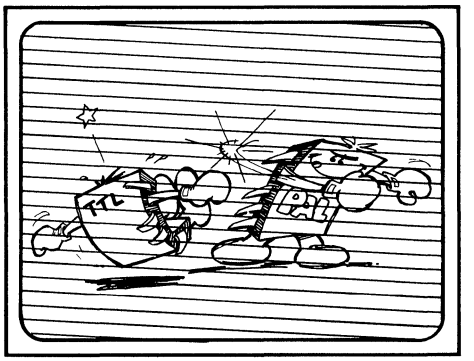


Figure 1d. The Odd and Even Fields Combined Generate One Frame

FIELD FREQUENCY — The number of times per second the field is completely scanned

The field frequency has been standardized at 60 fields per second. At this rate, the screen gets illuminated 60 times per second which results in the disappearance of the flickering. (Notice that the frame frequency is still 1/30 of a second.)

At the end of each scan line the electron beam moves rapidly from right to left on the screen which gives the horizontal retrace. Then it moves across the next odd/even numbered scan line and returns retracing right to left. While the beam retraces, blanking pulses are transmitted which blank out the total screen for this time period.

A HORIZONTAL SYNCHRONIZED PULSE is used to time the start of each horizontal scan line. The rate of the horizontal synchronizing pulse is equal to the total number of lines per frame multiplied by the number of frames transmitted per second:

$$525 \text{ lines} * 30 \text{ frames} = 15750 \text{ pulses/second}$$

A VERTICAL SYNCHRONIZED PULSE is used to time the start of each field. It occurs at the end of each field and causes the beam to go from bottom to top of the screen. During the vertical retrace, blanking pulses are transmitted which blank out the total screen for this time period. The rate of the vertical synchronizing pulse is equal to the number of fields transmitted per second: 60 pulses/second.

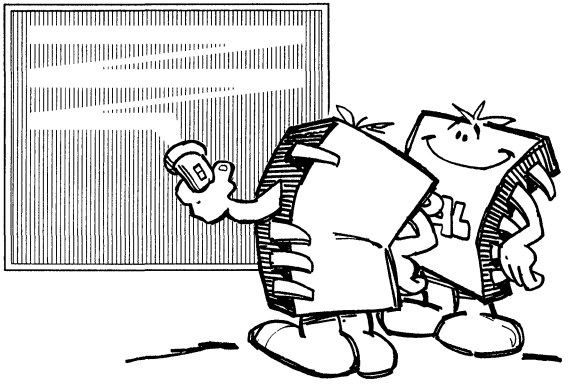


Figure 2. Horizontal Scanned Lines With the Retraces

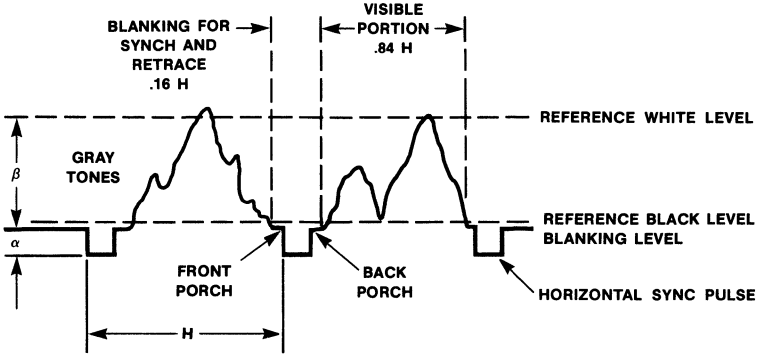


Figure 3. Composite Video Signal

$\beta = 1.0 \pm 0.05$ Volts
$\alpha = 0.4 \pm 0.05\beta$ Volts
H = Time From Start of One Line to Start of Next Line

A picture signal is defined as voltage levels which when combined with the synchronizing pulses generate **A COMPOSITE VIDEO SIGNAL**. There are two reference voltage levels; i.e., a high voltage level = white and a low voltage level = black (see Figure 3). In between these reference levels are the gray tones. The synchronizing pulses are below the black level so they do not produce light.

The horizontal and vertical pulses along with the picture signals constitute a frame where 15,750 horizontal lines are scanned every second making one complete horizontal cycle of approximately 63.5 microsceonds (commonly called 1H). Because of the front and the back porches (see Figure 3) that are used for retrace time and synchronization, the visible portion lasts approximately 54 microseconds. The time from the start of one field to the start of the next field is 262.5H or 1V.

Figure 6 illustrates the relationship between (a) the horizontal and vertical sweeps and (b) the line and the field blanking. A field blanking signal starts at Point A which is the initiation of the vertical retrace where the electron beam is at the bottom of the screen. During the vertical retrace period, the beam moves back and forth across the screen until it reaches the top of the screen (see Figure 5). The field remains blank until the beam reaches Point B. From Point B the beam is visible until it nears the right edge of the screen. At this time, horizontal blanking occurs and the beam moves from the right edge to the left edge of the screen. Horizontal blanking occurs 263 times for each field. At Point C, a new vertical retrace for the next field is initiated. At Point D, the beam finishes the scan of the entire screen for one frame. This point corresponds to Point A in the first field, so a field blanking occurs again.

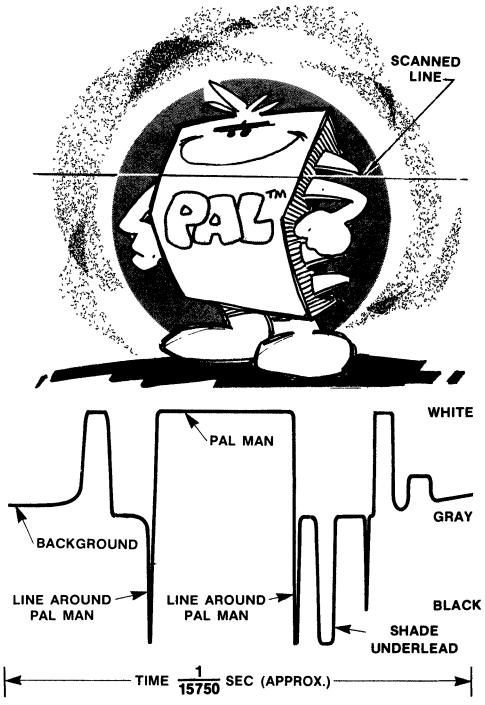


Figure 4. A Waveform of Video Voltage Produced by Scanning One Line on a Televised Screen

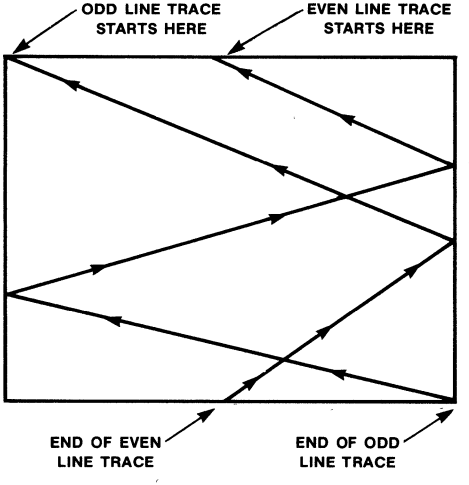


Figure 5. Vertical Retraces for the Odd and Even Fields

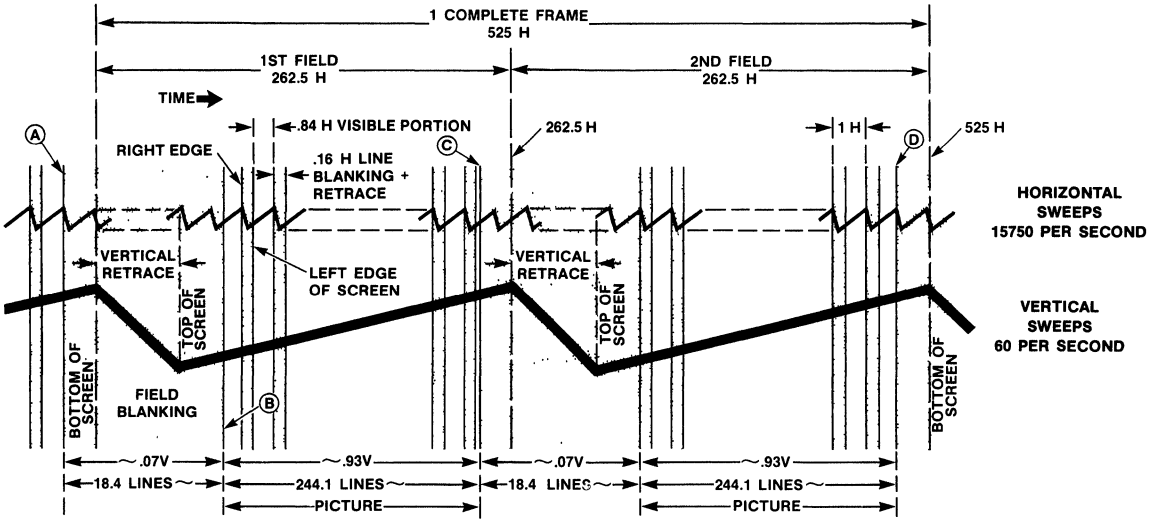


Figure 6. Vertical and Horizontal Sweeps, and Blanking Within a Frame

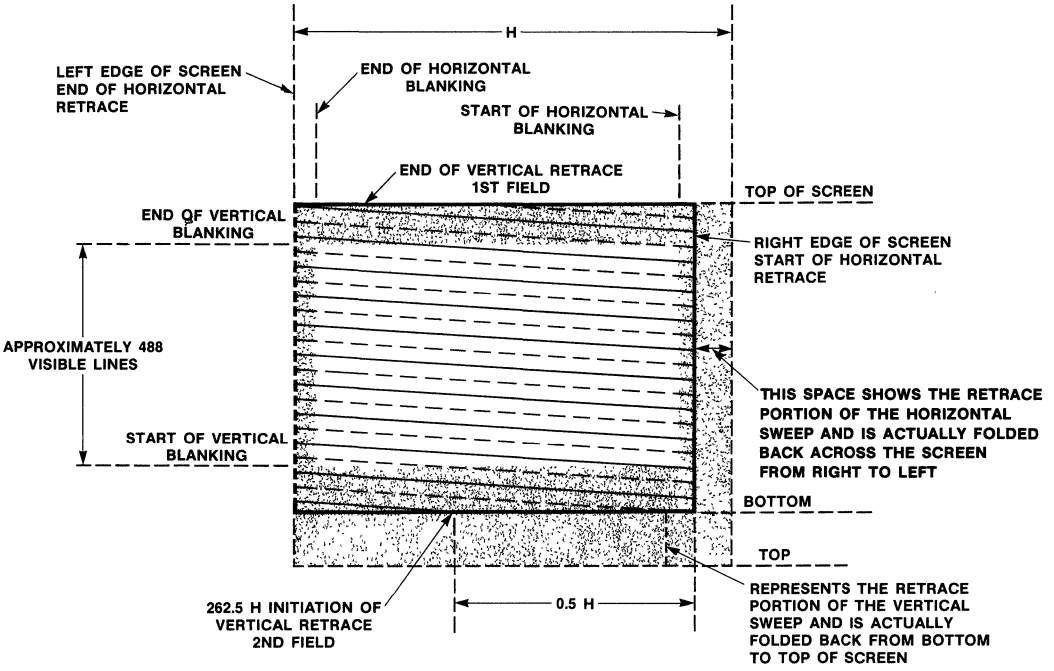


Figure 7. Effect of Vertical and Horizontal Blanking on the Screen

Figure 7 shows the actual movement of the beam and the effect of the line and the field blanking on the screen. The shaded portions at the top and bottom of the screen represent field blanking, while the shaded portion at the left and right edges of the screen represent line blanking. The combination of the shaded portions with the unshaded portions represent the available screen size, if no retracing were necessary.

From a single scan line to a complete picture which defines a frame, we have presented the concept and the process for sending information (a composite video signal) to a video receiver.

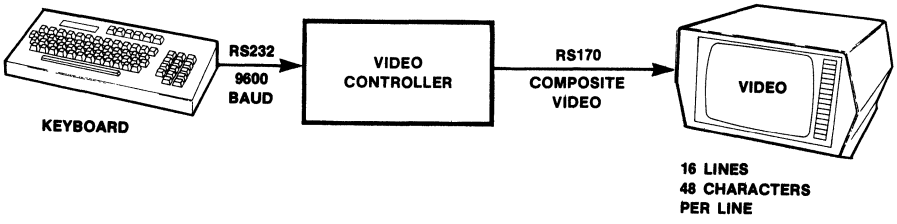


Figure 8. A Very Simplified Diagram

From the Keyboard to the Video Controller

The input to our video controller comes from a keyboard terminal or a computer. The information is transferred via an RS232 port to the video controller, one bit at a time. Each character is represented in ASCII code and is detected by the video controller only if a start signal was first issued.

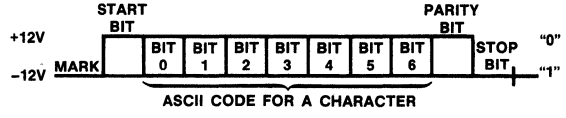


Figure 9. Each Character Consists of 10 Bits
- **1 Start Bit**
- **7 ASCII Bits**
- **1 Parity Bit**
- **1 Stop Bit**

The design is based on using 260 scan lines and an 8 MHz crystal. In order to obtain the standard horizontal period of 63.5 microseconds, each horizontal line is divided into 512 cycles giving a line period of 512/8 = 64.0 microseconds. The frequency of the electron beam is equal to the power line frequency (60 Hz) so in a clear environment (without noise) the picture is clear and steady.

$$\text{FRAME RATE} = \frac{8\ \text{MHz}}{260\ \text{scan lines} * 512\ \text{cycles per line}} = 60.09\ \text{Hz}$$

These figures and calculations are explained and diagrammed in detail in the section on implementation.

From the Video Controller to the Video Screen

Our video controller uses a **NON-INTERLACED METHOD** for scanning (see Figrue 10). In this method, the traced lines are adjacent and each frame consists of 260 scanned lines. Each field is equal to a frame which changes the definition of a field so that a complete frame is now transmitted in 1/60 of a second.

Figure 11 shows the horizontal and the vertical sweeps which cut in half the common frame frequency and now generate two frames for the same amount of time. At the end of each horizontal line, the electron beam moves rapidly from the right edge to the left edge of the screen while blanking pulses are transmitted (Points C-D in Figure 11).

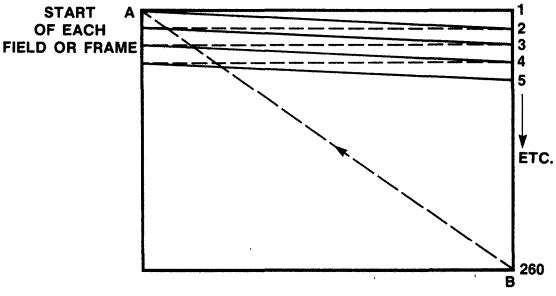


Figure 10. Non-interlaced Scanning. 260 Solid Lines Represent a Frame. The Dashed Lines are the Retrace Lines.

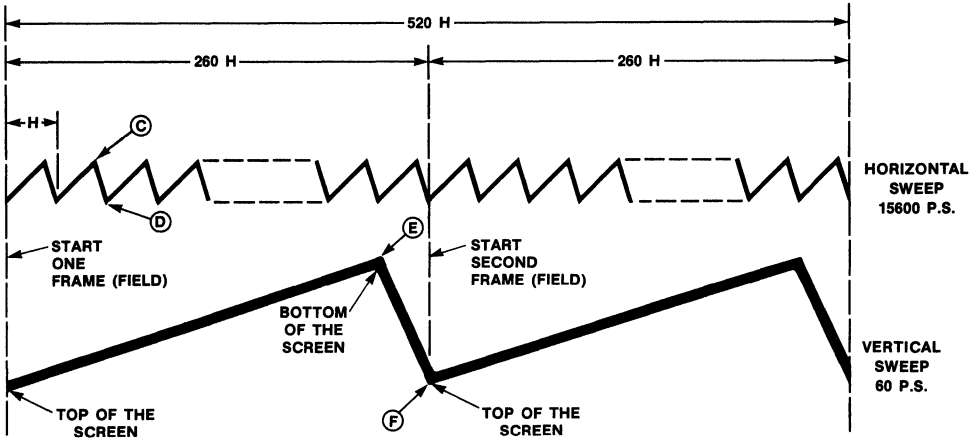


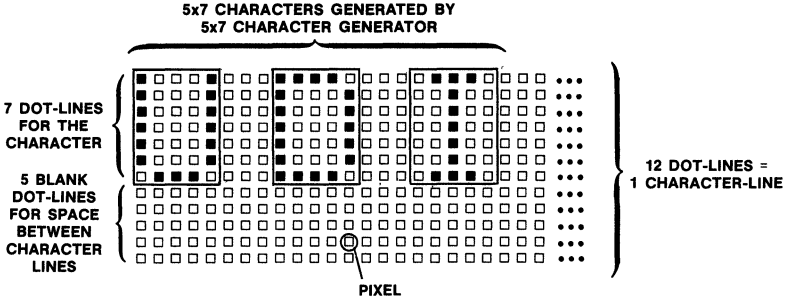
Figure 11. 520 Horizontal Lines Generate Two Fields Which Constitute Two Frames In A Non-Interlaced Scanning

At the end of 260 horizontal lines, the electron beam is blanked out while it goes back from the bottom to the top of the screen (Points E–F in Figure 11).

The video controller can display a maximum of 16 visible character-lines with 48 visible characters per character-line. Each character is produced by a 5×7 character generator. A character-line on the screen consists of twelve dot-lines: seven dot-lines for the character and five blank dot-lines for space between the character-lines. Each dot in a dot-line is called a **PIXEL**.

Figure 12 shows one character-line with the characters "U, D, I" and all the pixels that built these three characters on the screen. These pixels or character elements are sent one after the other in an orderly sequence across the dot-line. Twelve dot-lines are rapidly transmitted to create one character-line. This is called **SUCCESSIVE METHOD OF TRANSMISSION** (see Figure 13).

Figure 12. One Character-line With Three Characters



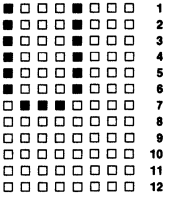
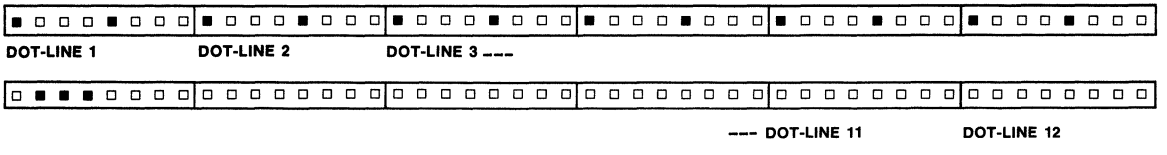


Figure 13. Successive Method of Transmission. For Simplicity Assume Only One Character 'U'. Each Dot-line is Sent in Orderly Sequence.



Implementation

Our video-controller board is divided into two subsystems which are connected by a RAM. The RAM is time shared by the two subsystems and provides the effect of a multi-port RAM. An 8-bit ASCII code is entered into the system through a "write only" port which is connected by the RS232 interface. The code is written into the RAM in locations indicated by pointers "SCROL" and "CURS". We don't write every cycle, but we read every cycle. The code is read from the RAM from locations indicated by pointers "LINES" and "CHAR" and transferred through a "read only" port to a character generator that generates the pixels for the screen. The "read" is done continuously so the picture looks steady. "Write" is done upon receipt of a special signal.

To build the video-controller board we used a RAM, a character generator and counters that are used as pointers/special module counters. Monolithic Memories' PAL units provide an excellent replacement for TTL counters. They can be programmed to count in any desired mode. The PAL meets the requirements of the standard TTL SSI/MSI of 25 nsec propagation delay while reducing the chip count on the board. The PAL can generate non-standard functions which are not available on standard chips. By customizing the PAL to give only the functions that are needed, the user can save board space. The following is a detailed design of a video controller using Monolithic Memories PAL devices.

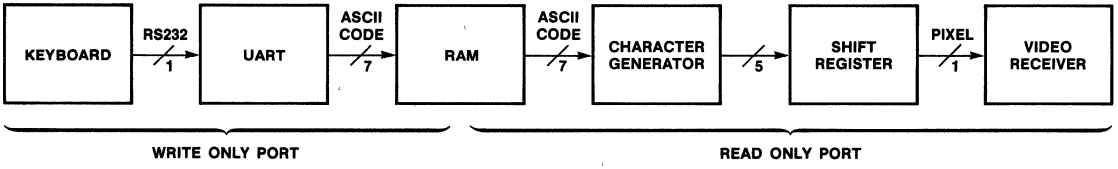


Figure 14. The RAM Divides the Board into Two Subsystems:
a) Write Only Subsystem; From the Keyboard to the RAM.
b) Read Only Subsystem; From the RAM to the Video Receiver.

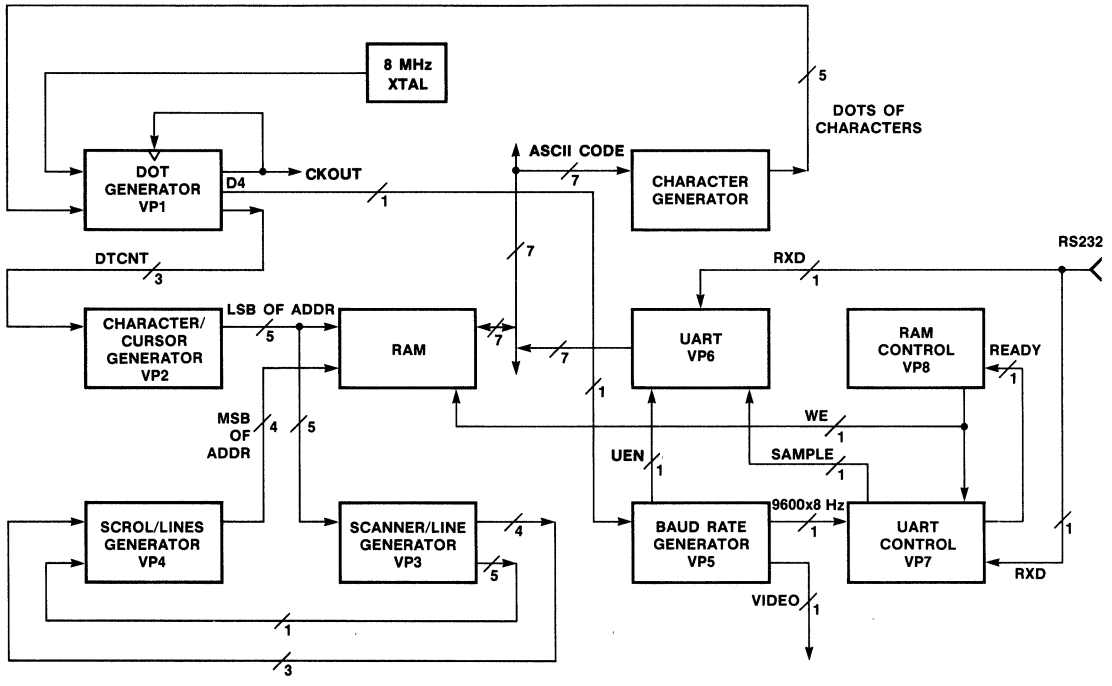
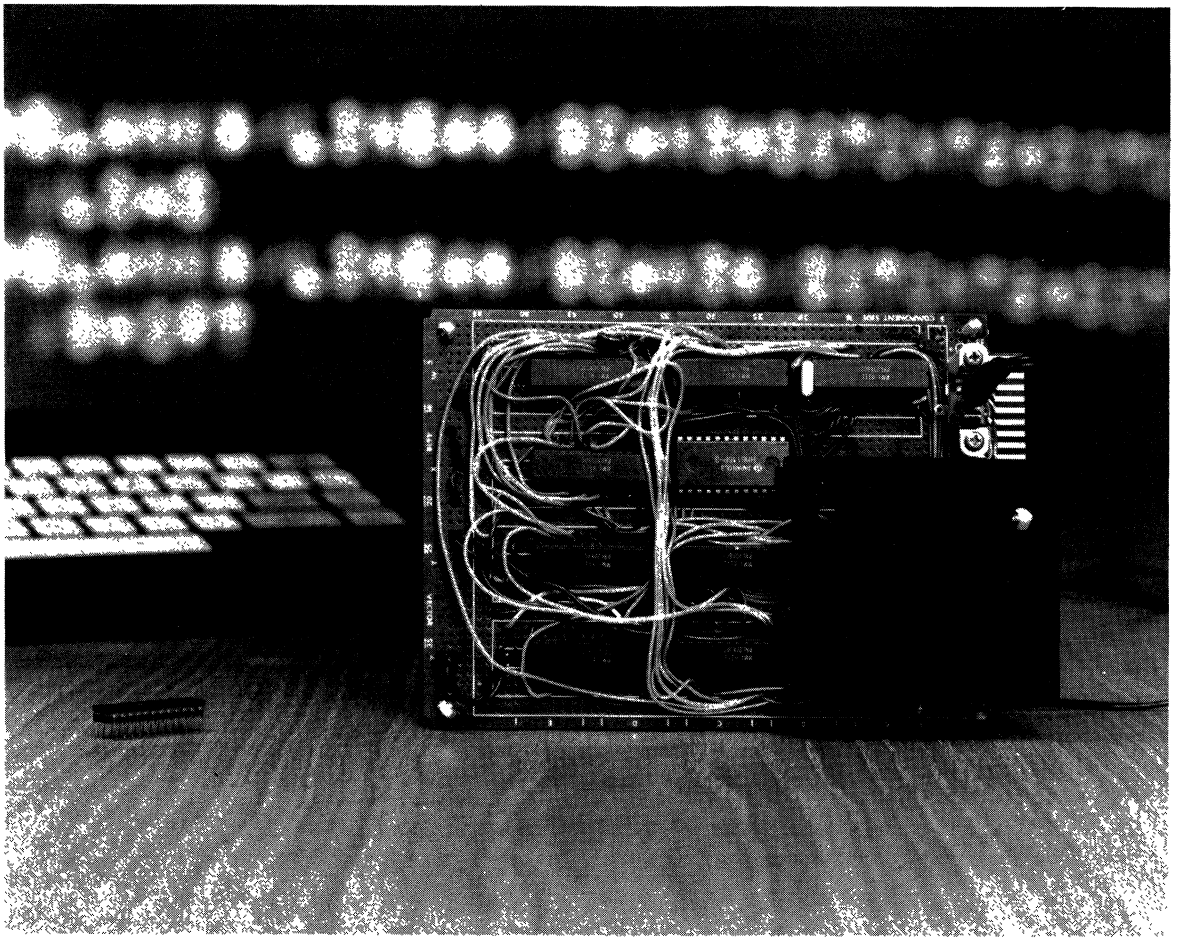
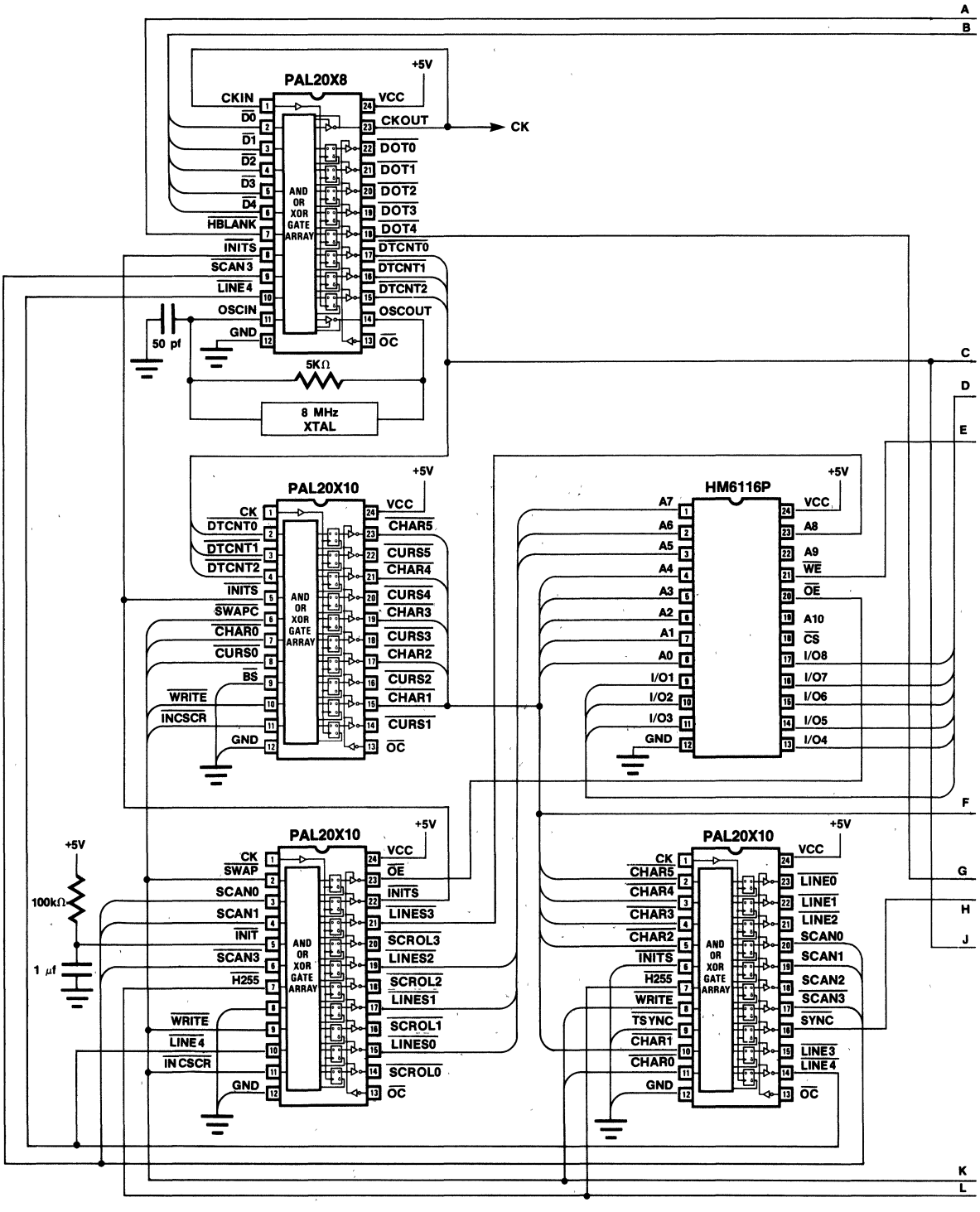


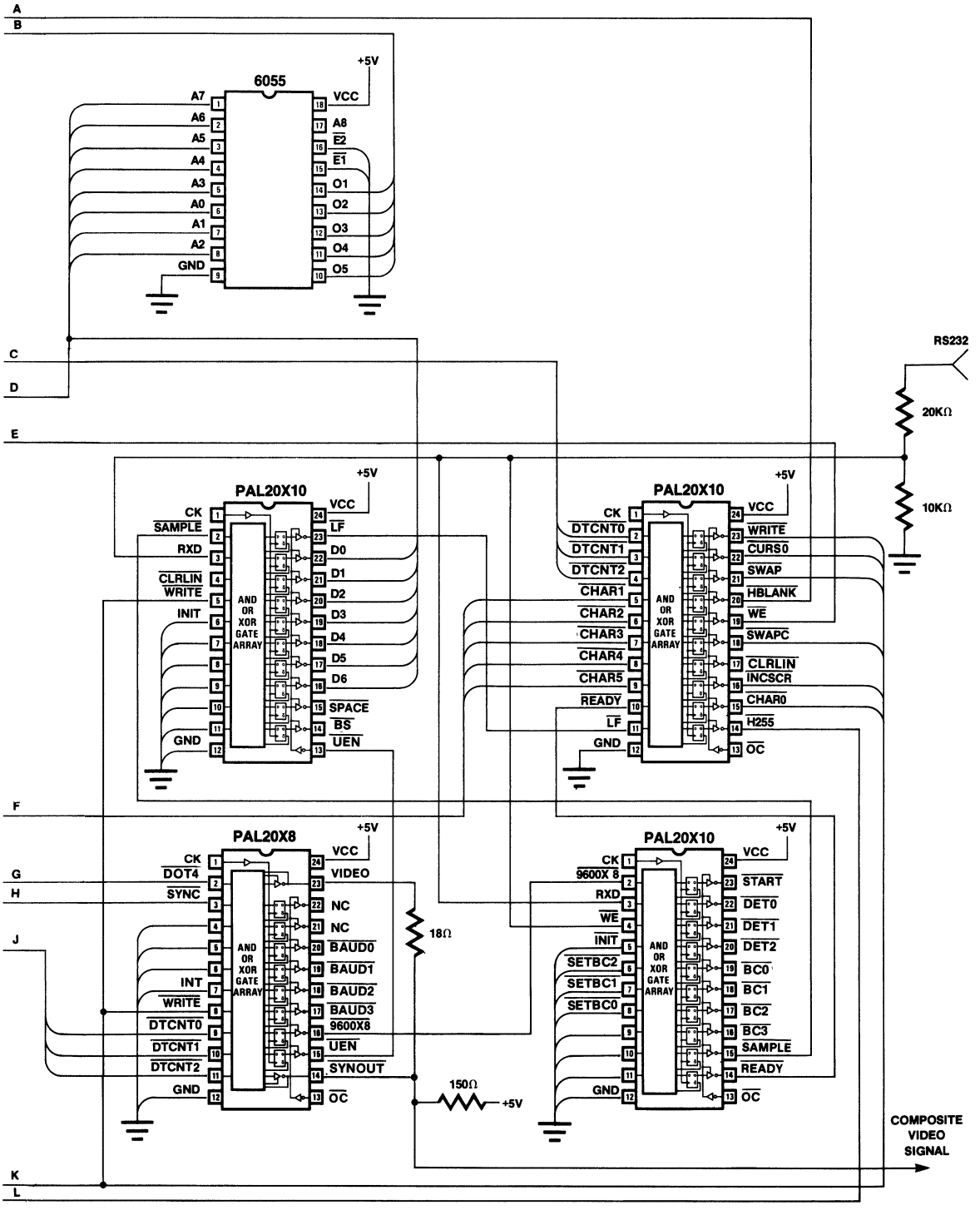
Figure 15. Block Diagram of the Video Controller

NOTES.

1 VP1 is a shift register for the dots in a dot-line of each character "DTCNT" counts 8 dots for each character (PAL20X8)
2 VP2 generates the 5 least significant bits of the address to the RAM (PAL20X10)· "CHAR", when reading from the RAM, "CURS", when writing into the RAM
3 VP3 counts 12 dot-lines per 1 character-line and the number of lines in the whole screen (PAL20X10)
4 VP4 generates the 4 most significant bits of the address to the RAM (PAL20X10) "LINES", when reading from the RAM, "SCROL", when writing into the RAM
5 VP5 generates the baud rate and the composite video signal (PAL20X8)
6 VP6 is a shift register It loads the ASCII bits in serial form from the RXD line (PAL20X8)
7 VP7 generates the "SAMPLE" pulses, the "READY" signal when a code for a character is in the UART, and detects a false "START" signal (PAL20X10)
8 VP8 controls the RAM (PAL20X10).
9 Monolithic Memories' 6055 is used as a character generator
10 Hitachi HM6116P· 2048 words x 8 bits high speed static CMOS RAM is used in our design

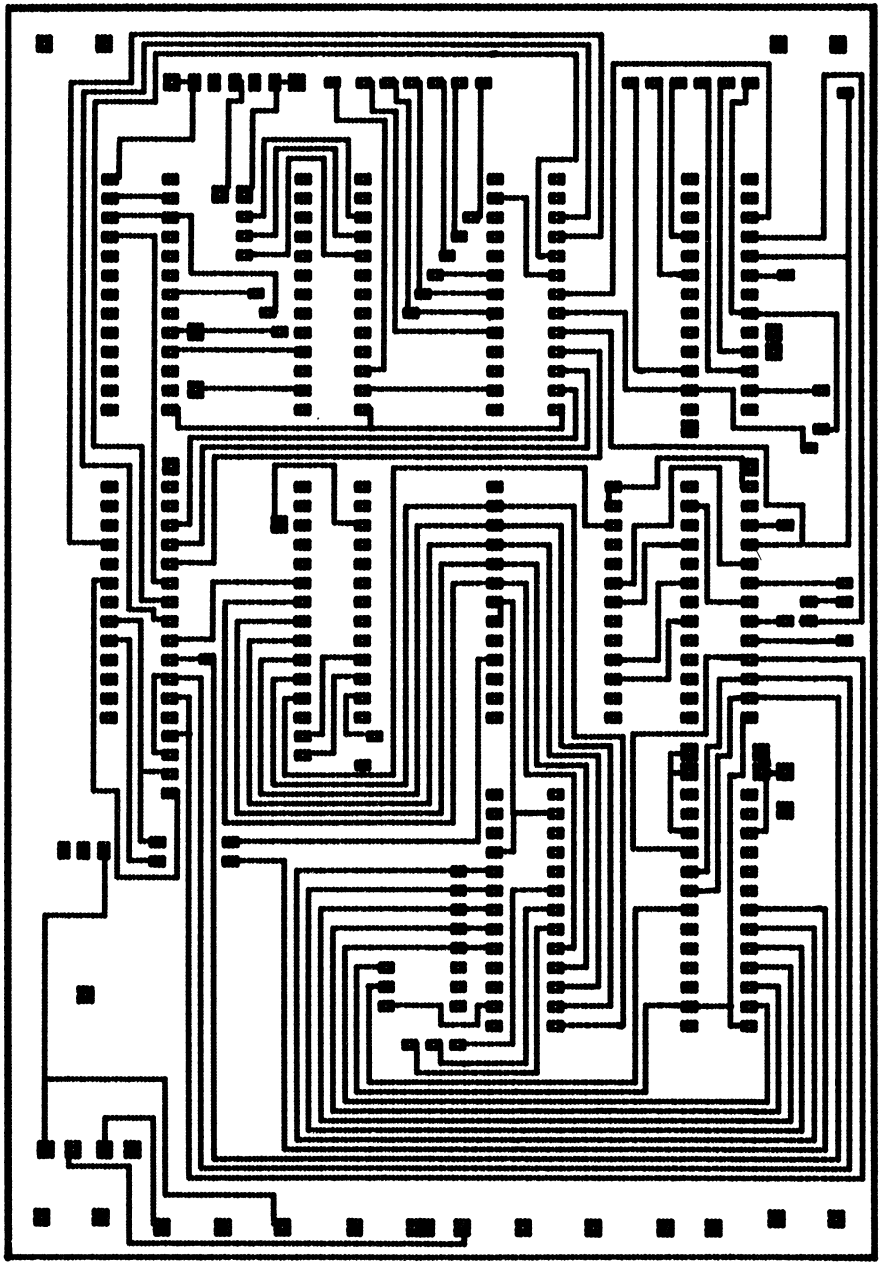



PAL20X8
+5V
CKIN
D0
D1
D2
D3
D4
HBLANK
INITS
SCAN3
LINE4
OSCIN
GND
VCC
CKOUT
DOT0
DOT1
DOT2
DOT3
DOT4
DTCNT0
DTCNT1
DTCNT2
OSCOUT
OC
AND OR XOR GATE ARRAY
CK
50 pf
5KΩ
8 MHz XTAL
PAL20X10
CK
DTCNT0
DTCNT1
DTCNT2
INITS
SWAPC
CHAR0
CURS0
BS
WRITE
INCSCR
GND
VCC
CHAR5
CURS5
CHAR4
CURS4
CHAR3
CURS3
CHAR2
CURS2
CHAR1
CURS1
OC
HM6116P
A7
A6
A5
A4
A3
A2
A1
A0
I/O1
I/O2
I/O3
GND
VCC
A8
A9
WE
OE
A10
CS
I/O8
I/O7
I/O6
I/O5
I/O4
PAL20X10
CK
SWAP
SCAN0
SCAN1
INIT
SCAN3
H255
WRITE
LINE4
INCSCR
GND
VCC
OE
INITS
LINES3
SCROL3
LINES2
SCROL2
LINES1
SCROL1
LINES0
SCROL0
OC
100kΩ
1 μf
PAL20X10
CK
CHAR5
CHAR4
CHAR3
CHAR2
INITS
H255
WRITE
TSYNC
CHAR1
CHAR0
GND
VCC
LINE0
LINE1
LINE2
SCAN0
SCAN1
SCAN2
SCAN3
SYNC
LINE3
LINE4
OC
A
B
C
D
E
F
G
H
J
K
L

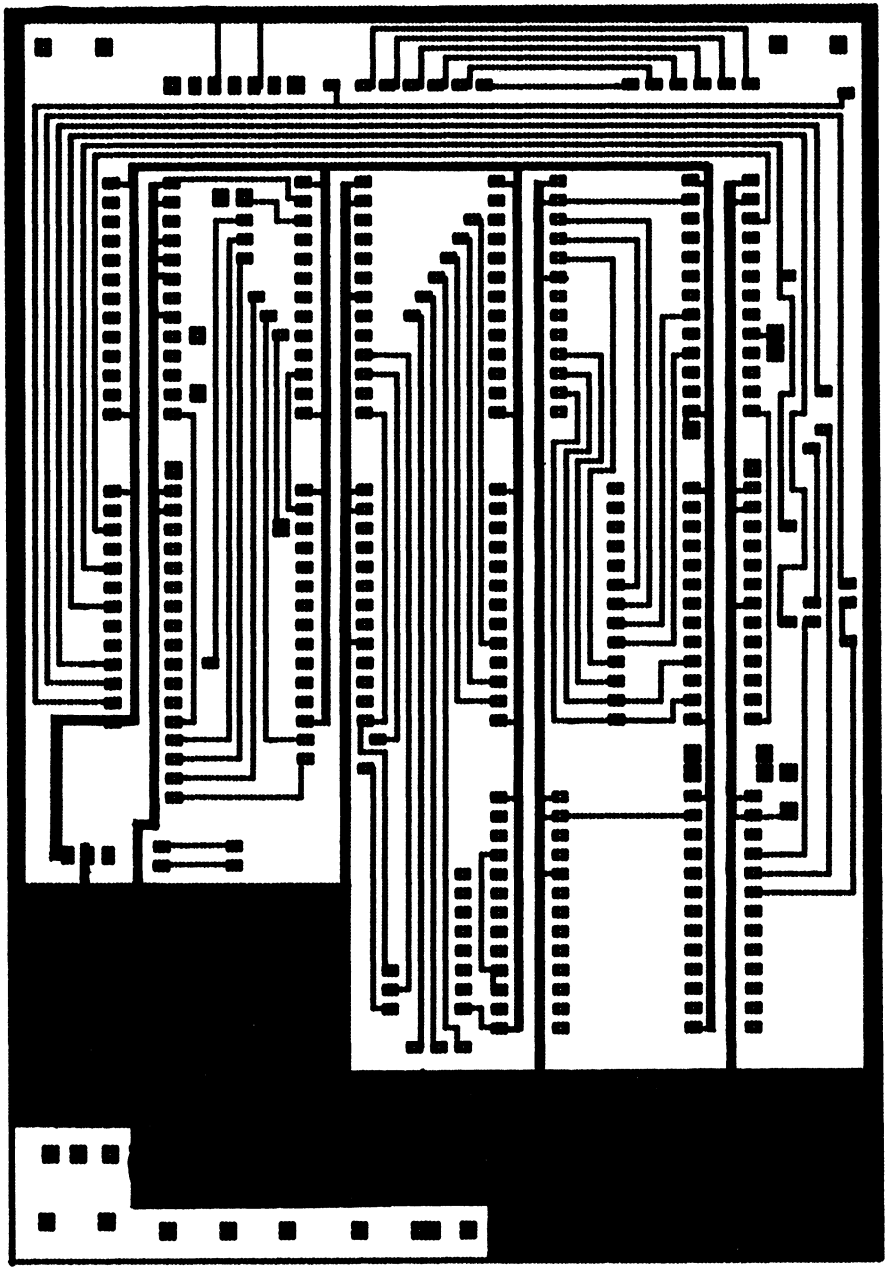

A
B
6055
+5V
A7
A6
A5
A4
A3
A0
A1
A2
GND
VCC
A8
E2
E1
O1
O2
O3
O4
O5
C
D
E
RS232
20KΩ
10KΩ
PAL20X10
CK
SAMPLE
RXD
CLRLIN
WRITE
INIT
GND
VCC
LF
D0
D1
D2
D3
D4
D5
D6
SPACE
BS
UEN
AND OR XOR GATE ARRAY
DTCNT0
DTCNT1
DTCNT2
CHAR1
CHAR2
CHAR3
CHAR4
CHAR5
READY
WRITE
CURS0
SWAP
HBLANK
WE
SWAPC
CLRLIN
INCSCR
CHAR0
H255
OC
F
PAL20X8
DOT4
SYNC
INT
DTCNT0
DTCNT1
DTCNT2
VIDEO
NC
NC
BAUD0
BAUD1
BAUD2
BAUD3
9600X8
UEN
SYNOUT
18Ω
150Ω
G
H
J
9600X 8
RXD
WE
INIT
SETBC2
SETBC1
SETBC0
START
DET0
DET1
DET2
BC0
BC1
BC2
BC3
SAMPLE
READY
COMPOSITE VIDEO SIGNAL
K
L

7

Solder Side



Component Side

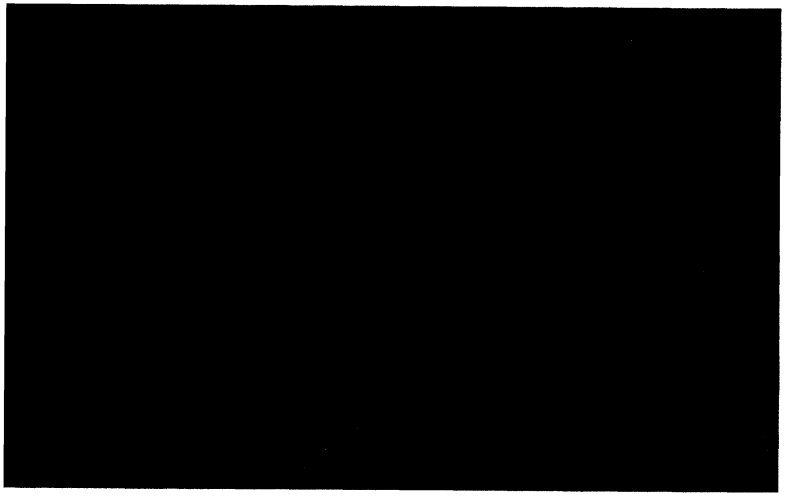


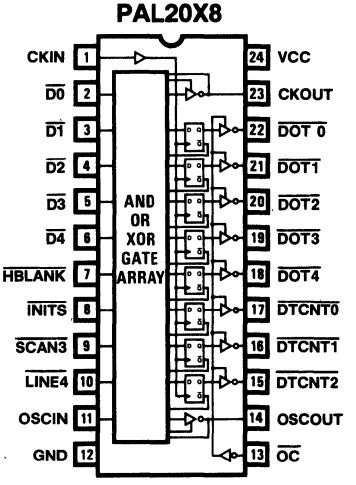
Video Controller Specifications

The Video controller provides a 16 line by 48 character display for use with standard CRT monitors and televisions. The controller "listens" to standard RS232C serial data via a 25 pin DB-25 plug/socket pair inserted into any Computer/Terminal RS232C interface. The controller stores the serial data in a 2Kx8 RAM and continuously displays it on the RS170 composite video output. Eight PALs and one character generator provide the control circuitry, mounted on a 5x7 inch PCB.

lines per frame	16
characters per line	48
character format	5x7
ASCII character set	64 (upper case)
scanning method	non-interlace
scans per frame	260 (312 optional)
scans per line	12
frame period	16.64 milli seconds (19.968 optional)
scan period	64 micro seconds
character period	1 micro second
dot period	125 nano seconds
input specification	RS232C
speed	9600 Baud (4800,2400,1200 optional)
format	7 bits, mark parity, one stop bit
low level	-3 to -12 Volts
high level	+6 to +12 Volts
input impedance	30K Ohm
input connector	25 pin DB-25P/DB-25S
output specification	RS170 (composite video)
sync pulse	4 micro seconds
horizontal blanking	16 micro seconds
sync level	0.4 Volts
black level	0.8 Volts
white level	1.8 Volts
output drive	75 Ohm termination
output connector	RCA Audio Jack
power	12 Watts
input voltage	100-120 Volts AC (200-240 optional)
line frequency	50/60 Hz
card size	5x7x1 inches

Dot Generator





```
PAL20X8                                          PAL DESIGN SPECIFICATION
VP1                                              BIRKNER/KAZMI/UDI 7/7/81
DOT GENERATOR
MMI SUNNYVALE, CALIFORNIA
CKIN /D0 /D1 /D2 /D3 /D4 /HBLANK /INITS /SCAN3 /LINE4 OSCIN GND
/OC OSCOUT /DTCNT2 /DTCNT1 /DTCNT0 /DOT4 /DOT3 /DOT2 /DOT1 /DOT0
CKOUT VCC


IF(VCC) /CKOUT = /OSCOUT

 DOT0     :=  /HBLANK*/LINE4*/SCAN3*DTCNT2*DTCNT1*DTCNT0*D0     ;LOAD

 DOT1     :=  /HBLANK*/LINE4*/SCAN3*DTCNT2*DTCNT1*DTCNT0*D1     ;LOAD
           +  /HBLANK*/LINE4*/SCAN3*/DTCNT2*DOT0                ;SHIFT

 DOT2     :=  /HBLANK*/LINE4*/SCAN3*DTCNT2*DTCNT1*DTCNT0*D2     ;LOAD
           +  /HBLANK*/LINE4*/SCAN3*/DTCNT2*DOT1                ;SHIFT

 DOT3     :=  /HBLANK*/LINE4*/SCAN3*DTCNT2*DTCNT1*DTCNT0*D3     ;LOAD
           +  /HBLANK*/LINE4*/SCAN3*/DTCNT2*DOT2                ;SHIFT

 DOT4     :=  /HBLANK*/LINE4*/SCAN3*DTCNT2*DTCNT1*DTCNT0*D4     ;LOAD
           +  /HBLANK*/LINE4*/SCAN3*/DTCNT2*DOT3                ;SHIFT

 DTCNT0   :=  /INITS*DTCNT0                                     ;HOLD
           +  /INITS*DTCNT0                                     ;EXTEND
          :+: /INITS                                            ;INC

 DTCNT1   :=  /INITS*DTCNT1                                     ;HOLD
           +  /INITS*DTCNT1                                     ;EXTEND
          :+: /INITS*DTCNT0                                     ;INC

 DTCNT2   :=  /INITS*DTCNT2                                     ;HOLD
           +  /INITS*DTCNT2                                     ;EXTEND
          :+: /INITS*DTCNT0*DTCNT1                              ;INC

IF(VCC) /OSCOUT = OSCIN
```

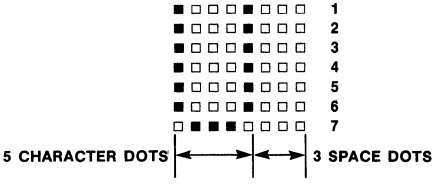
FUNCTION TABLE

CKIN D4 D3 D2 D1 D0 HBLANK INITS SCAN3 LINE4 OSCIN /OC OSCOUT DTCNT2
DTCNT1 DTCNT0 DOT4 DOT3 DOT2 DOT1 DOT0 CKOUT

; C K I N	DATA IN D4---D0	H B L A N K	I N I T S	S C A N 3	L I N E 4	O S C I N	/ O C	O S C O U T	DTCNT 210	DOT 43210	C K O U T	COMMENTS
C	XXXXX	X	H	X	X	X	L	X	LLL	XXXXX	X	INITIALIZE DTCNT
C	XXXXX	X	L	X	X	X	L	X	LLH	XXXXX	X	INC DTCNT
C	XXXXX	X	L	X	X	X	L	X	LHL	XXXXX	X	INC DTCNT
C	XXXXX	X	L	X	X	L	L	H	LHH	XXXXX	H	INC DTCNT, OSCILIN = L
C	XXXXX	X	L	X	X	H	L	L	HLL	XXXXX	L	INC DTCNT, OSCILIN = H
C	XXXXX	X	L	X	X	X	L	X	HLH	XXXXX	X	INC DTCNT
C	XXXXX	X	L	X	X	X	L	X	HHL	XXXXX	X	INC DTCNT
C	XXXXX	X	L	X	X	X	L	X	HHH	XXXXX	X	INC DTCNT
C	LHLHL	L	L	L	L	X	L	X	LLL	LHLHL	X	DATA IS LOADED
C	XXXXX	L	L	L	L	X	L	X	LLH	HLHLL	X	OUTPUT TO SCREEN = H
C	XXXXX	L	L	L	L	X	L	X	LHL	LHLLL	X	OUTPUT TO SCREEN = L
C	XXXXX	L	L	L	L	X	L	X	LHH	HLLLL	X	OUTPUT TO SCREEN = H
C	XXXXX	L	L	L	L	X	L	X	HLL	LLLLL	X	OUTPUT TO SCREEN = L
C	XXXXX	L	L	L	L	X	L	X	HLH	LLLLL	X	SEND BLANK TO SCREEN
C	XXXXX	L	L	L	L	X	L	X	HHL	LLLLL	X	SEND BLANK TO SCREEN
C	XXXXX	L	L	L	L	X	L	X	HHH	LLLLL	X	SEND BLANK TO SCREEN
C	HLHLH	L	L	L	L	X	L	X	LLL	HLHLH	X	NEW DATA IS LOADED
C	XXXXX	L	L	L	L	X	L	X	LLH	LHLHL	X	OUTPUT TO SCREEN = L

DESCRIPTION

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



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

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

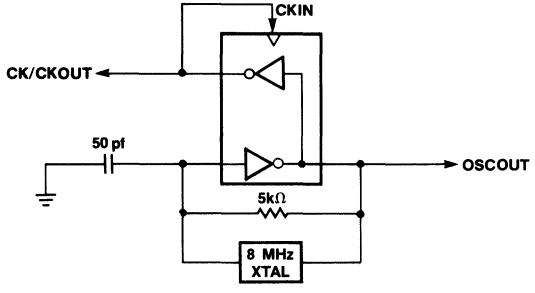
"HBLANK" IS A SIGNAL FOR THE END OF ONE DOT LINE (48 CHARACTERS).

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

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

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

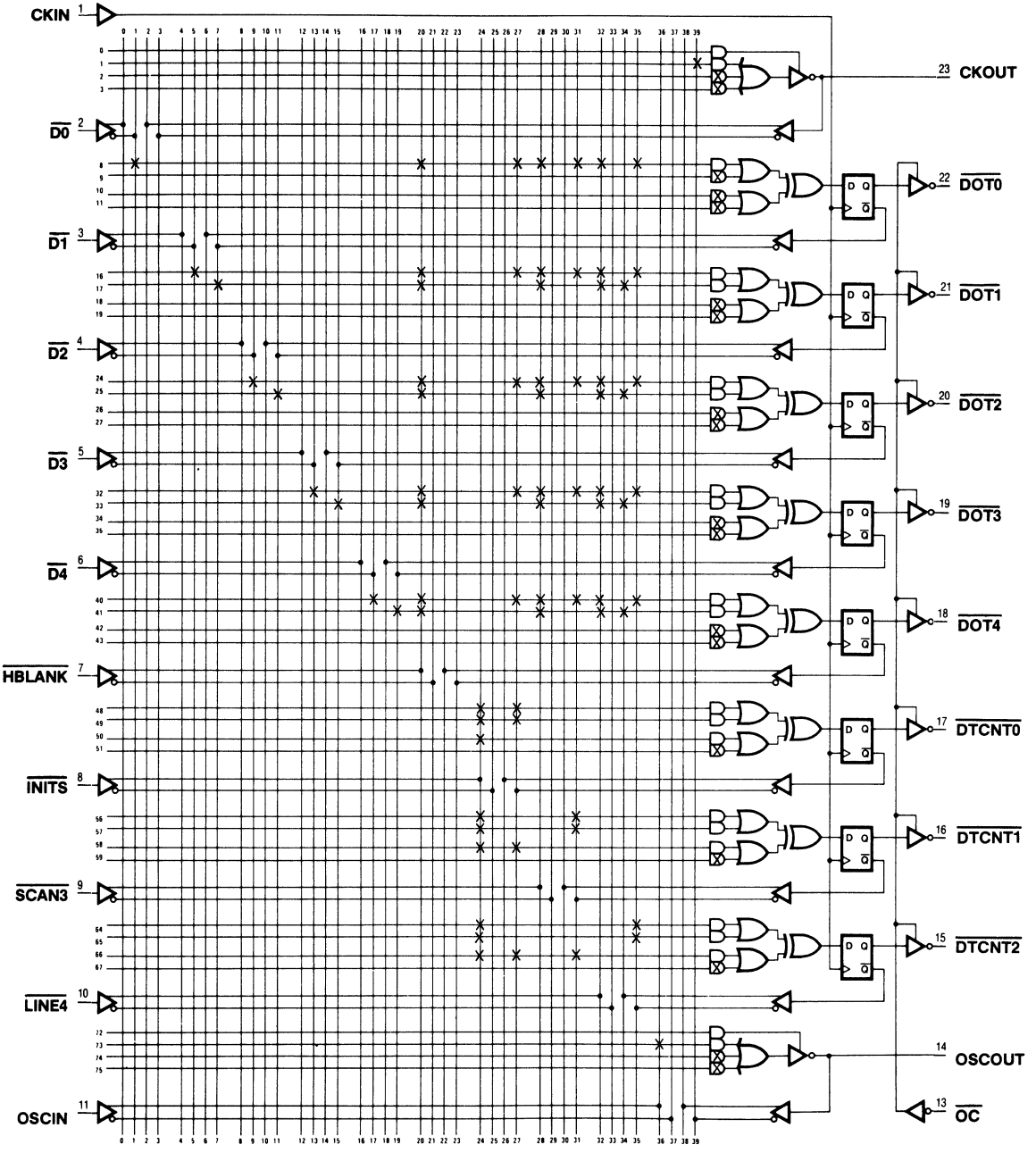
WHEN "LINE4" AND/OR "SCAN3" ARE SET, "DOT" SENDS BLANK DOTS TO THE SCREEN FOR GENERATING SPACES BETWEEN CHARACTER LINES AND FOR THE MARGINS OF THE SCREEN. THE OUTPUT FROM THE REGISTER TO THE SCREEN IS THROUGH "DOT4".



CLOCK GENERATOR

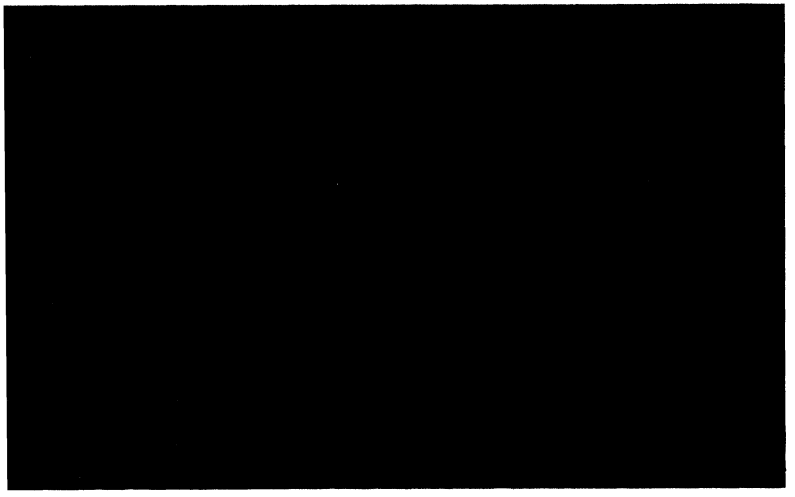
Dot Generator

Logic Diagram PAL20X 8

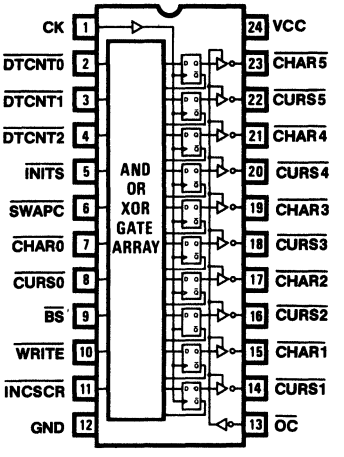



IMPORTANT
NOTES

CHAR/CURS Generator



PAL20X10



```
PAL20X10                                        PAL DESIGN SPECIFICATION
VP2                                             BIRKNER/KAZMI/UDI 7/9/81
CHAR/CURS GENERATOR
MMI SUNNYVALE, CALIFORNIA
CK /DTCNT0 /DTCNT1 /DTCNT2 /INITS /SWAPC /CHAR0 /CURS0 /BS /WRITE /INCSCR GND
/OC /CURS1 /CHAR1 /CURS2 /CHAR2 /CURS3 /CHAR3 /CURS4 /CHAR4 /CURS5 /CHAR5 VCC

 CHAR1 :=  SWAPC*/INITS*CURS1                          ;SWAP WITH CURS
        + /SWAPC*/INITS*DTCNT0*DTCNT1*DTCNT2*CHAR0     ;INC
       :+:/SWAPC*/INITS*CHAR1                          ;HOLD

 CHAR2 :=  SWAPC*/INITS*CURS2                          ;SWAP WITH CURS
        + /SWAPC*/INITS*DTCNT0*DTCNT1*DTCNT2           ;INC
                * CHAR0*CHAR1
       :+:/SWAPC*/INITS*CHAR2                          ;HOLD

 CHAR3 :=  SWAPC*/INITS*CURS3                          ;SWAP WITH CURS
        + /SWAPC*/INITS*DTCNT0*DTCNT1*DTCNT2           ;INC
                * CHAR0*CHAR1*CHAR2
       :+:/SWAPC*/INITS*CHAR3                          ;HOLD

 CHAR4 :=  SWAPC*/INITS*CURS4                          ;SWAP WITH CURS
        + /SWAPC*/INITS*DTCNT0*DTCNT1*DTCNT2           ;INC
                * CHAR0*CHAR1*CHAR2*CHAR3
       :+:/SWAPC*/INITS*CHAR4                          ;HOLD

 CHAR5 :=  SWAPC*/INITS*CURS5                          ;SWAP WITH CURS
        + /SWAPC*/INITS*DTCNT0*DTCNT1*DTCNT2           ;INC
                * CHAR0*CHAR1*CHAR2*CHAR3*CHAR4
       :+:/SWAPC*/INITS*CHAR5                          ;HOLD

 CURS1 :=  SWAPC*/INITS*/INCSCR*CHAR1                  ;SWAP WITH CHAR
        + /SWAPC*/INITS*/INCSCR*CURS1                  ;HOLD
       :+:/SWAPC*/INITS*/INCSCR                        ;INC
                * WRITE* DTCNT2*DTCNT1*/DTCNT0*CURS0

 CURS2 :=  SWAPC*/INITS*/INCSCR*CHAR2                  ;SWAP WITH CHAR
        + /SWAPC*/INITS*/INCSCR*CURS2                  ;HOLD
       :+:/SWAPC*/INITS*/INCSCR                        ;INC
                * WRITE* DTCNT2*DTCNT1*/DTCNT0
                * CURS0* CURS1

 CURS3 :=  SWAPC*/INITS*/INCSCR*CHAR3                  ;SWAP WITH CHAR
        + /SWAPC*/INITS*/INCSCR*CURS3                  ;HOLD
       :+:/SWAPC*/INITS*/INCSCR                        ;INC
                * WRITE* DTCNT2*DTCNT1*/DTCNT0
                * CURS0* CURS1*CURS2

 CURS4 :=  SWAPC*/INITS*/INCSCR*CHAR4                  ;SWAP WITH CHAR
        + /SWAPC*/INITS*/INCSCR*CURS4                  ;HOLD
       :+:/SWAPC*/INITS*/INCSCR                        ;INC
                * WRITE* DTCNT2*DTCNT1*/DTCNT0
                * CURS0* CURS1*CURS2*CURS3

 CURS5 :=  SWAPC*/INITS*/INCSCR*CHAR5                  ;SWAP WITH CHAR
        + /SWAPC*/INITS*/INCSCR*CURS5                  ;HOLD
       :+:/SWAPC*/INITS*/INCSCR                        ;INC
                * WRITE* DTCNT2*DTCNT1*/DTCNT0
                * CURS0* CURS1*CURS2*CURS3*CURS4
```

FUNCTION TABLE

```
CK    DTCNT2 DTCNT1 DTCNT0 INITS SWAPC BS    WRITE INCSCR /OC   CURS5 CURS4
CURS3 CURS2  CURS1  CURS0  CHAR5 CHAR4 CHAR3 CHAR2 CHAR1  CHAR0

;                          I
;           I  S       W  N
;           N  W       R  C
;           I  A       I  S  /
; C  DTCNT  T  P  B    T  C  O   CURS      CHAR
; K   210   S  C  S    E  R  C  54321 0   54321 0   COMMENTS
-------------------------------------------------------------------------
  C   XXX   H  X  X    X  X  L  LLLLL L   LLLLL L   INITIALIZE CHAR AND CURS
  C   LLL   L  L  X    X  L  L  LLLLL L   LLLLL H   INCREMENT CHAR, HOLD CURS
  C   LLH   L  L  X    X  L  L  LLLLL L   LLLLL H   HOLD CHAR AND CURS
  C   LHL   L  L  X    X  L  L  LLLLL L   LLLLL H   HOLD CHAR AND CURS
  C   LHH   L  L  X    X  L  L  LLLLL L   LLLLL H   HOLD CHAR AND CURS
  C   HLL   L  L  X    X  L  L  LLLLL L   LLLLL H   HOLD CHAR AND CURS
  C   HLH   L  L  X    X  L  L  LLLLL L   LLLLL H   HOLD CHAR AND CURS
  C   HHL   L  L  X    X  L  L  LLLLL L   LLLLL H   HOLD CHAR AND CURS
  C   HHH   L  L  X    X  L  L  LLLLL L   LLLLH H   INCREMENT CHAR
  C   LLL   L  L  X    X  L  L  LLLLL L   LLLLH L   HOLD CHAR AND CURS
  C   LLH   L  L  X    X  L  L  LLLLL L   LLLLH L   HOLD CHAR AND CURS
  C   HHH   L  L  X    X  L  L  LLLLL L   LLLLH L   INCREMENT CHAR
  C   LLL   L  L  X    X  L  L  LLLLL L   LLLLH H   CHAR = 3
  C   LLH   L  L  X    X  L  L  LLLLL L   LLLLH H   HOLD CHAR AND CURS
  C   LHL   L  L  X    X  L  L  LLLLL L   LLLLH H   HOLD CHAR AND CURS
  C   LHH   L  H  X    X  L  L  LLLLH H   LLLLL L   SWAP CHAR AND CURS
  C   HLL   L  L  X    X  L  L  LLLLH H   LLLLL H   SET CHAR0
  C   HHH   L  L  X    X  L  L  LLLLH H   LLLLH H   INCREMENT CHAR TO 3
  C   HHH   L  L  X    X  L  L  LLLLH H   LLLHL H   INCREMENT CHAR, HOLD CURS
  C   LLL   L  L  X    X  L  L  LLLLH H   LLLHL L   CHAR = 4
  C   HHL   L  L  X    L  L  L  LLLLH H   LLLHL L   HOLD CHAR AND CURS
  C   HHH   L  L  X    X  L  L  LLLLH H   LLLHL L   INCREMENT CHAR
  C   LLL   L  L  X    X  L  L  LLLLH H   LLLHL H   CHAR = 5
  C   HHH   L  L  X    X  L  L  LLLLH H   LLLHH H   INCREMENT CHAR
  C   LLL   L  L  X    X  L  L  LLLLH H   LLLHH L   CHAR = 6
  C   LLL   L  L  X    X  L  L  LLLLH H   LLLHH H   CHAR = 7
  C   HHH   L  L  X    X  L  L  LLLLH H   LLHLL H   INCREMENT CHAR
  C   HHH   L  L  X    X  L  L  LLLLH H   LLHLL L   CHAR = 8
  C   HHH   L  L  X    X  L  L  LLLLH H   LLHLH H   INCREMENT CHAR TO 11
  C   LLL   L  L  X    X  L  L  LLLLH H   LLHLH H   CHAR = 11
  C   HHH   L  L  X    X  L  L  LLLLH H   LLHHL H   SET CHAR0, INC CHAR BY 2
  C   HHH   L  L  X    X  L  L  LLLLH H   LLHHH H   CHAR = 15
  C   HHH   L  L  X    X  L  L  LLLLH H   LHLLL H   CHAR = 17
  C   HHL   L  L  X    H  L  L  LLLHL H   LHLLL H   INCREMENT CURS
  C   HHH   L  L  X    X  L  L  LLLHL L   LHLLH H   CHAR = 19, CURS = 4
  C   HHL   L  L  X    H  L  L  LLLHH H   LHLLH H   CURS = 7
  C   HHH   L  L  X    X  L  L  LLLHH H   LHLHL H   CHAR = 21
  C   HHL   L  L  X    H  L  L  LLHLL H   LHLHL H   CURS = 9
  C   HHH   L  L  X    X  L  L  LLHLL H   LHLHH H   CHAR = 23
  C   HHL   L  L  X    H  L  L  LLHLH H   LHLHH H   CURS = 11
  C   HHH   L  L  X    X  L  L  LLHLH H   LHHLL H   CHAR = 25
  C   HHL   L  L  X    H  L  L  LLHHL H   LHHLL H   CURS = 13
```

C	HHH	L	L	X	X	L	L	LLHHL H	LHHLH H	CHAR = 27
C	HHL	L	L	X	H	L	L	LLHHH H	LHHLH H	CURS = 15
C	HHH	L	L	X	X	L	L	LLHHH H	LHHHL H	CHAR = 29
C	HHH	L	L	X	X	L	L	LLHHH H	LHHHH H	CHAR = 31
C	HHH	L	L	X	X	L	L	LLHHH H	HLLLL H	CHAR = 33
C	HHH	L	L	X	X	L	L	LLHHH H	HLLLH H	CHAR = 35
C	HHH	L	L	X	X	L	L	LLHHH H	HLLHL H	CHAR = 37
C	HHH	L	L	X	X	L	L	LLHHH H	HLLHH H	CHAR = 39
C	HHH	L	L	X	X	L	L	LLHHH H	HLHLL H	CHAR = 41
C	HHH	L	L	X	X	L	L	LLHHH H	HLHLH H	CHAR = 43
C	HHH	L	L	X	X	L	L	LLHHH H	HLHHL H	CHAR = 45
C	HHH	L	L	X	X	L	L	LLHHH H	HLHHH H	CHAR = 47
C	HHH	L	L	X	X	L	L	LLHHH H	HHLLL H	CHAR = 49
C	HHH	L	L	X	X	L	L	LLHHH H	HHLLH H	CHAR = 51
C	HHH	L	L	X	X	L	L	LLHHH H	HHLHL H	CHAR = 53
C	HHH	L	L	X	X	L	L	LLHHH H	HHLHH H	CHAR = 55
C	HHH	L	L	X	X	L	L	LLHHH H	HHHLL H	CHAR = 57
C	HHH	L	L	X	X	L	L	LLHHH H	HHHLH H	CHAR = 59
C	HHH	L	L	X	X	L	L	LLHHH H	HHHHL H	CHAR = 61
C	HHH	L	L	X	X	L	L	LLHHH H	HHHHH H	CHAR = 63
C	LLL	L	L	X	X	L	L	LLHHH H	HHHHH H	CHAR = 63, DTCNT = 0
C	LLH	L	L	X	X	L	L	LLHHH H	HHHHH H	CHAR = 63, DTCNT = 1
C	LHL	L	L	X	X	L	L	LLHHH H	HHHHH H	CHAR = 63, DTCNT = 2
C	LHH	L	L	X	X	L	L	LLHHH H	HHHHH H	CHAR = 63, DTCNT = 3
C	HLL	L	L	X	X	L	L	LLHHH H	HHHHH H	CHAR = 63, DTCNT = 4
C	HLH	L	L	X	X	L	L	LLHHH H	HHHHH H	CHAR = 63, DTCNT = 5
C	HHL	L	L	X	H	L	L	LHLLL H	HHHHH H	INCREMENT CURS
C	HHH	L	L	X	X	L	L	LHLLL L	LLLLL H	CURS = 14, INCREMENT CHAR
C	HHH	L	L	X	X	L	L	LHLLL L	LLLLL L	CURS = 14, CHAR = 0

DESCRIPTION

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

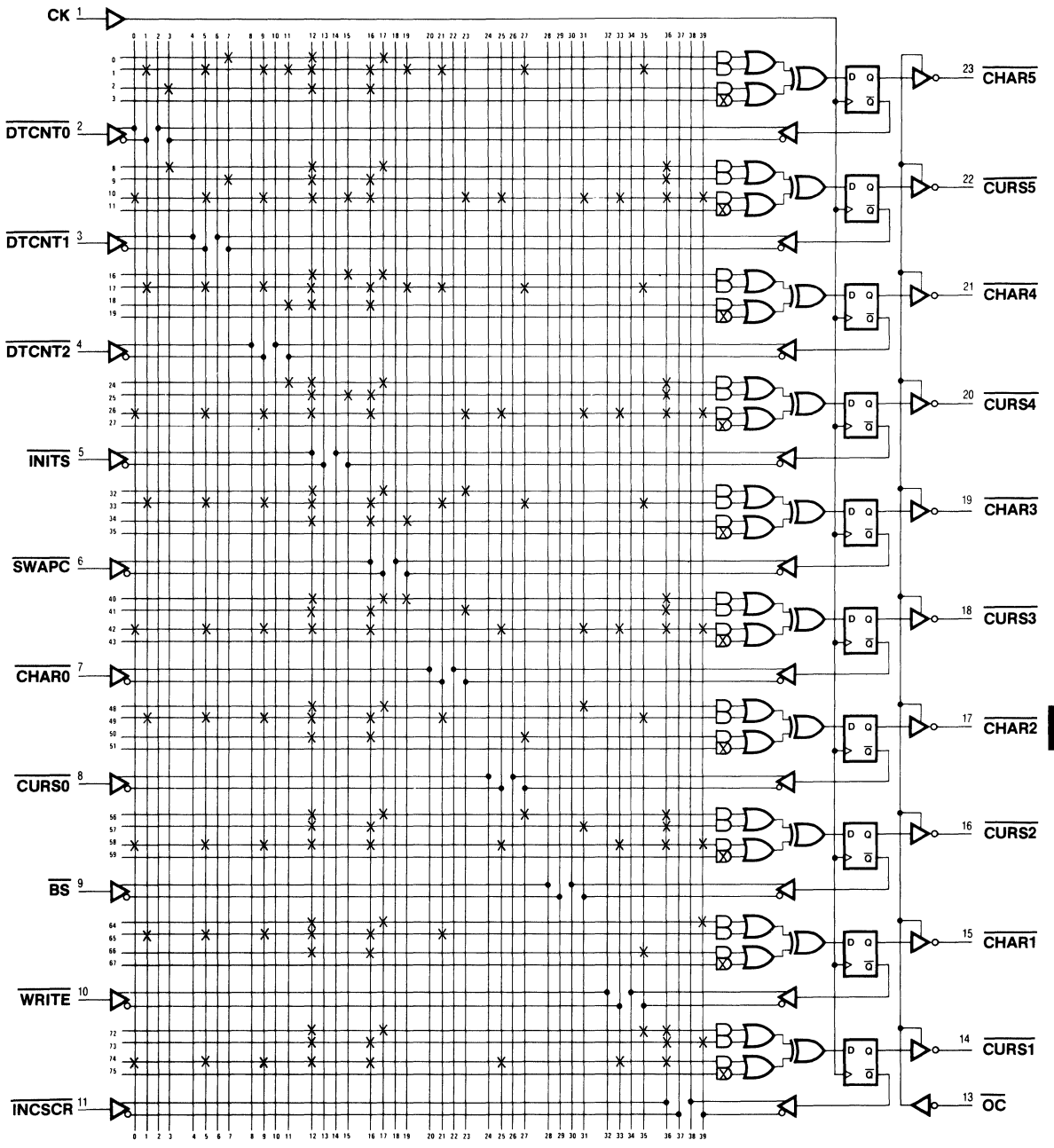
"CHAR" IS INCREMENTED AT THE END OF 8 PIXELS MEANING IT COUNTS AFTER EACH CHARACTER. IT COUNTS FROM 0 TO 63 ALTHOUGH ONLY 48 CHARACTERS ARE VISIBLE. THE HORIZONTAL SYNC PULSE IS GIVEN BETWEEN CHARACTERS 56 AND 59. DURING THE COUNTS OF CHAR FROM 48 TO 63, BLACK SIGNALS ARE TRANSMITTED TO THE SCREEN. "CURS"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

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

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

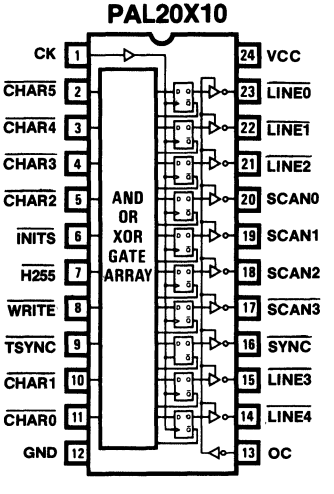
CHAR/CURS Generator

Logic Diagram PAL20X10




IMPORTANT
NOTES

SCAN/LINE Generator



7

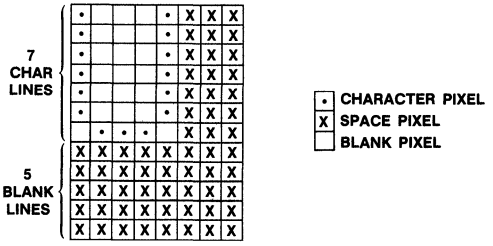
```
PAL20X10                                          PAL DESIGN SPECIFICATION
VP3                                               BIRKNER/UDI      7/13/81
SCAN/LINE GENERATOR
MMI SUNNYVALE, CALIFORNIA
 CK /CHAR5 /CHAR4 /CHAR3 /CHAR2 /INITS /H255 /WRITE /TSYNC /CHAR1 /CHAR0 GND
/OC /LINE4 /LINE3 /SYNC /SCAN3  SCAN2  SCAN1  SCAN0 /LINE2 /LINE1 /LINE0 VCC


/SCAN0   :=  INITS                                          ;INITIALIZE
          + /INITS*/SCAN0                                   ;HOLD
         :+:/INITS*H255                                     ;INCREMENT

/SCAN1   :=  INITS                                          ;INITIALIZE
          + /INITS*/SCAN1                                   ;HOLD
         :+:/INITS*H255*SCAN0                               ;INCREMENT

/SCAN2   :=  INITS                                          ;INITIALIZE
          + /INITS*/SCAN2                                   ;HOLD
         :+:/INITS*H255*SCAN0*SCAN1*/SCAN3                  ;INC IN MODULUS 12

 SCAN3   := /INITS*SCAN3                                    ;HOLD
          + /INITS*H255*LINE4*LINE2*LINE0                   ;DETECT SCAN LINE 260
                *SCAN2*SCAN1*SCAN0                          ;FOR VERTICAL RETRACE
         :+:/INITS*H255*SCAN0*SCAN1*SCAN2                   ;INCREMENT
          + /INITS*H255*SCAN3*SCAN1*SCAN0                   ;MODULE 12 CORRECTION
                                                            ;INITIAL WHEN INITS=H
LINE0    := /INITS*LINE0                                    ;HOLD
          + /TSYNC                                          ;TEST SYNC
         :+:/INITS*H255*SCAN3*SCAN1*SCAN0                   ;INCREMENT
          + /INITS*H255*LINE4*LINE2*LINE0                   ;DETECT LINE 21 7/12
                *SCAN2*SCAN1*SCAN0                          ;FOR VERTICAL RETRACE

LINE1    := /INITS*LINE1                                    ;HOLD
          + /TSYNC                                          ;TEST SYNC
         :+:/INITS*H255*SCAN3*SCAN1*SCAN0*LINE0             ;INCREMENT

LINE2    := /INITS*LINE2                                    ;HOLD
          + /INITS*LINE2                                    ;EXTEND
         :+:/INITS*H255*SCAN3*SCAN1*SCAN0*LINE0*LINE1       ;INCREMENT
          + /INITS*H255*LINE4*LINE2*LINE0                   ;DETECT LINE 21 7/12
                *SCAN2*SCAN1*SCAN0                          ;FOR VERTICAL RETRACE

LINE3    := /INITS*LINE3                                    ;HOLD
          + /INITS*LINE3                                    ;EXTEND
         :+:/INITS*H255*SCAN3*SCAN1*SCAN0                   ;INCREMENT
                *LINE2*LINE1*LINE0

LINE4    := /INITS*LINE4                                    ;HOLD
          + /TSYNC                                          ;TEST SYNC
         :+:/INITS*H255*SCAN3*SCAN1*SCAN0                   ;INCREMENT
                *LINE3*LINE2*LINE1*LINE0
          + /INITS*H255*LINE4*LINE2*LINE0                   ;DETECT LINE 21 7/12
                *SCAN2*SCAN1*SCAN0                          ;FOR VERTICAL RETRACE

 SYNC    :=   CHAR5*CHAR4*/CHAR3*CHAR2*/WRITE               ;CHAR 52-55 HORIZ SYNC
          +   LINE0*LINE1*/LINE2*LINE4*/SCAN2*/SCAN3*SYNC;VERTICAL SYNC
         :+:  LINE0*LINE1*/LINE2*LINE4*/SCAN2*/SCAN3*SYNC;WHEN LINE=19 SCAN 0-3
              *CHAR5*CHAR4*/CHAR3*/CHAR2                    ;AND CHAR 48-51
```

FUNCTION TABLE

```
CK    CHAR5 CHAR4 CHAR3 CHAR2 CHAR1 CHAR0 TSYNC INITS H255  WRITE
/OC   LINE4 LINE3 LINE2 LINE1 LINE0 SYNC  SCAN3 SCAN2 SCAN1 SCAN0

;            T  I     W
;            S  N  H  R           S
;            Y  I  2  I  /        Y
; C   CHAR   N  T  5  T  O   LINE  N  SCAN
; K  543210  C  S  5  E  C  43210  C  3210   COMMENTS
-------------------------------------------------------------------------------
  C  XXXXXX  H  H  X  X  L  LLLLL  X  LLLL   INITIALIZE COUNTERS
  C  XXXXXX  H  L  H  X  L  LLLLL  X  LLLH   INC SCAN
  C  XXXXXX  H  L  H  X  L  LLLLL  X  LLHL   INC SYNC
  C  XXXXXX  H  L  H  X  L  LLLLL  X  LLHH   INC SYNC
  C  XXXXXX  H  L  H  X  L  LLLLL  X  LHLL   INC SCAN
  C  XXXXXX  H  L  H  X  L  LLLLL  X  LHLH   INC SCAN
  C  XXXXXX  H  L  H  X  L  LLLLL  X  LHHL   INC SCAN
  C  XXXXXX  H  L  H  X  L  LLLLL  X  LHHH   INC SCAN
  C  XXXXXX  H  L  H  X  L  LLLLL  X  HLLL   INC SCAN
  C  XXXXXX  H  L  H  X  L  LLLLL  X  HLLH   INC SCAN
  C  XXXXXX  H  L  H  X  L  LLLLL  X  HLHL   INC SCAN
  C  XXXXXX  H  L  H  X  L  LLLLL  X  HLHH   INC SCAN
  C  XXXXXX  H  L  H  X  L  LLLLH  X  LLLL   INC LINE, INC SCAN MODULE 12
  C  XXXXXX  H  L  H  X  L  LLLLH  X  LLLH   LINE = 1, INC SCAN
  C  XXXXXX  H  L  H  X  L  LLLLH  X  LLHL   LINE = 1, INC SCAN
  C  XXXXXX  H  L  H  X  L  LLLLH  X  LLHH   LINE = 1, INC SCAN
  C  XXXXXX  H  H  X  X  L  LLLLL  X  LLLL   INITIALIZE COUNTERS
  C  HHLLXX  L  L  L  X  L  HLLHH  L  LLLL   LINE = 19, FOR TESTING VERTIC SYNC
  C  HHLLXX  H  L  H  X  L  HLLHH  L  LLLH   VERTICAL SYNC
  C  HHLLXX  H  L  H  X  L  HLLHH  L  LLHL   VERTICAL SYNC
  C  HHLLXX  H  L  H  X  L  HLLHH  L  LLHH   VERTICAL SYNC
  C  HHLLXX  H  L  H  X  L  HLLHH  L  LHLL   INC SCAN
  C  HHLLXX  H  L  H  X  L  HLLHH  L  LHLH   INC SCAN
  C  HHLHXX  H  L  L  L  L  XXXXX  H  XXXX   HORIZONTAL SYNC
-------------------------------------------------------------------------------
```

DESCRIPTION

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

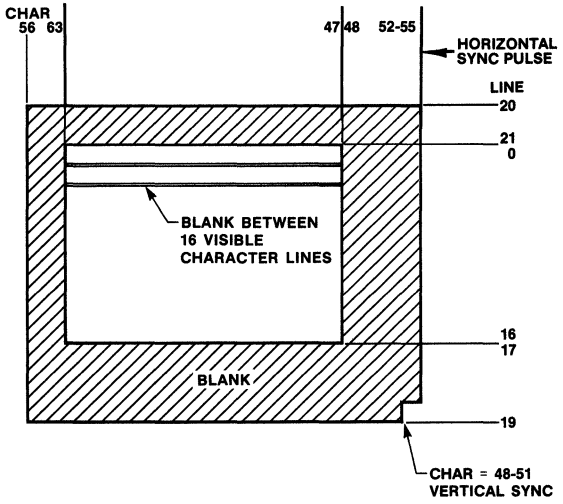


"SCAN" IS A MODULE 12 COUNTER THAT COUNTS THE NUMBER OF THE DOT LINES FOR EACH CHARACTER.

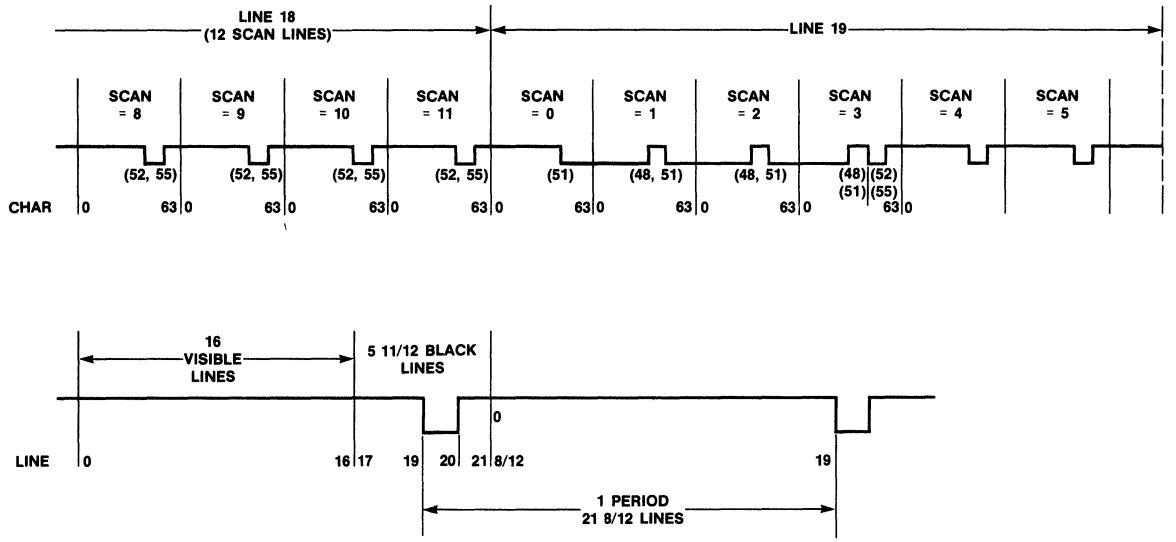
"LINE" COUNTS THE NUMBER OF THE CHARACTER LINES. EACH CHARACTER LINE IS 12 SCAN LINES. THE COUNTER COUNTS UNTIL 21 ALTHOUGH ONLY 16 LINES ARE VISIBLE ON THE SCREEN.

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

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

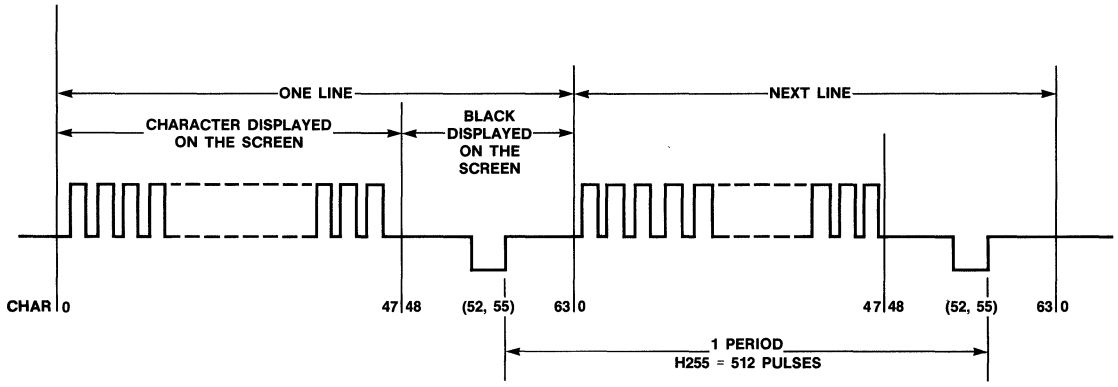


Vertical Sync



12 dot-lines generate one-character line. The total number of scanned lines is 260 or 21 8/12 character-lines. Out of the 21 8/12 character lines only 16 lines are visible. The vertical sync pulse is asserted during Line 19.

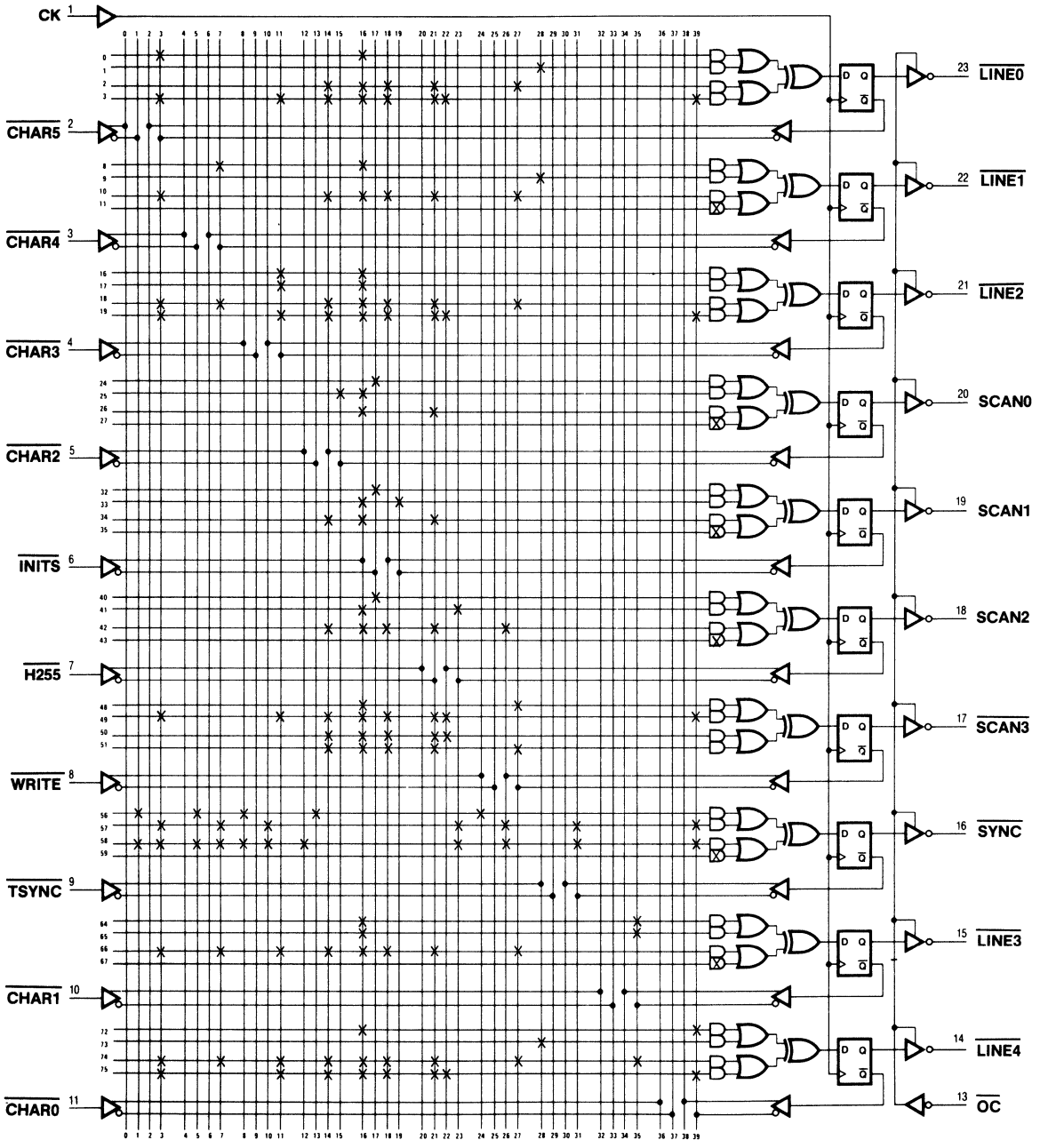
Horizontal Sync



One character is represented in 8 pulses. Then the 512 pulses cause "CHAR" to count from 0 to 63. Out of the 64 characters only 48 characters are visible on the screen. The horizontal sync pulse is asserted during "CHAR" position 52 through and including "CHAR" 55.

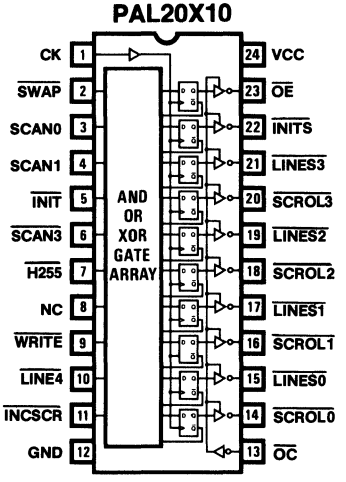
SCAN/LINE Generator

Logic Diagram PAL20X10




IMPORTANT
NOTES

LINES/SCROL Generator



7

```
PAL20X10                                        PAL DESIGN SPECIFICATION
VP4                                             BIRKNER/UDI      7/14/81
LINES/SCROL GENERATOR
MMI SUNNYVALE, CALIFORNIA
 CK /SWAP    SCAN0   SCAN1  /INIT   /SCAN3   /H255   NC      /WRITE  /LINE4
/INCSCR GND /OC     /SCROL0 /LINES0 /SCROL1 /LINES1 /SCROL2 /LINES2 /SCROL3
/LINES3 /INITS /OE  VCC

LINES0  :=  /SWAP*/INITS*LINES0                     ;HOLD
         +   SWAP*/INITS*SCROL0                     ;SWAP WITH SCROL
        :+: /SWAP*/INITS*H255*/LINE4                ;INC (12 DOT LINES)
                 * SCAN3*SCAN1*SCAN0
         +  /SWAP*/INITS*INCSCR                     ;INC (LF)

SCROL0  :=  /SWAP*/INITS*SCROL0                     ;HOLD
         +   SWAP*/INITS*LINES0                     ;SWAP WITH LINES
        :+: /SWAP*/INITS*INCSCR                     ;INC (LF OR CHAR = 47)
         +         INITS                            ;INITIALIZE

LINES1  :=  /SWAP*/INITS*LINES1                     ;HOLD
         +   SWAP*/INITS*SCROL1                     ;SWAP WITH SCROL
        :+: /SWAP*/INITS*H255*/LINE4                ;INC (12 DOT LINES)
                 * SCAN3*SCAN1*SCAN0*LINES0
         +  /SWAP*/INITS*INCSCR*LINES0              ;INC (LF)

SCROL1  :=  /SWAP*/INITS*SCROL1                     ;HOLD
         +   SWAP*/INITS*LINES1                     ;SWAP WITH LINES
        :+: /SWAP*/INITS*INCSCR*SCROL0              ;INC (LF OR CHAR = 47)
         +         INITS                            ;INITIALIZE

LINES2  :=  /SWAP*/INITS*LINES2                     ;HOLD
         +   SWAP*/INITS*SCROL2                     ;SWAP WITH SCROL
        :+: /SWAP*/INITS*H255*/LINE4                ;INC (12 DOT LINES)
                 * SCAN3*SCAN1*SCAN0
                 *LINES1*LINES0
         +  /SWAP*/INITS*INCSCR                     ;INC (LF)
                 *LINES1*LINES0

SCROL2  :=  /SWAP*/INITS*SCROL2                     ;HOLD
         +   SWAP*/INITS*LINES2                     ;SWAP WITH LINES
        :+: /SWAP*/INITS*INCSCR*SCROL0*SCROL1       ;INC (LF OR CHAR = 47)
         +         INITS                            ;INITIALIZE

LINES3  :=  /SWAP*/INITS*LINES3                     ;HOLD
         +   SWAP*/INITS*SCROL3                     ;SWAP WITH SCROL
        :+: /SWAP*/INITS*H255*/LINE4                ;INC (12 DOT LINES)
                 * SCAN3*SCAN1*SCAN0
                 *LINES2*LINES1*LINES0
         +  /SWAP*/INITS*INCSCR                     ;INC (LF)
                 *LINES2*LINES1*LINES0

SCROL3  :=  /SWAP*/INITS*SCROL3                     ;HOLD
         +   SWAP*/INITS*LINES3                     ;SWAP WITH LINES
        :+: /SWAP*/INITS*INCSCR                     ;INC (LF OR CHAR = 47)
                 *SCROL2*SCROL1*SCROL0
         +         INITS                            ;INITIALIZE

INITS   :=  INIT*H255                               ;INITIALIZATION SIGNAL

OE      := /WRITE                                   ;ENABLE THREE STATE OF RAM
```

FUNCTION TABLE

```
CK    SWAP   SCAN3  SCAN1  SCAN0  INIT   H255   WRITE LINE4  INCSCR /OC
SCROL3 SCROL2 SCROL1 SCROL0 LINES3 LINES2 LINES1 LINES0  INITS   OE

;                          I
;                      W   L   N                       I
;     S          I   H   R   I   C                       N
;     W          N   2   I   N   S   /                   I
; C   A   SCAN   I   5   T   E   C   O   SCROL   LINES   T   O
; K   P    310   T   5   E   4   R   C    3210    3210   S   E   COMMENTS
-------------------------------------------------------------------------------
  C   X    XXX   H   H   X   X   X   L    XXXX    XXXX   H   X   SET INITIALIZE BIT
  C   X    XXX   L   X   X   X   X   L    HHHH    LLLL   L   X   INITIALIZE COUNTERS
  C   L    XXX   L   L   X   X   L   L    HHHH    LLLL   L   X   INITS = L
  C   L    HHH   L   H   X   L   L   L    HHHH    LLLH   L   X   INC LINES, HOLD SCROL
  C   L    HHH   L   H   X   L   L   L    HHHH    LLHL   L   X   INC LINES, HOLD SCROL
  C   L    HHH   L   H   X   L   L   L    HHHH    LLHH   L   X   INC LINES, SCAN = 11
  C   L    HHH   L   H   X   L   L   L    HHHH    LHLL   L   X   INC LINES, SCAN = 11
  C   L    XXX   L   H   X   X   H   L    LLLL    LHLH   L   X   INC LINES & SCROL: LF
  C   L    XXX   L   X   X   X   H   L    LLLH    LHHL   L   X   INC LINES & SCROL: LF
  C   L    XXX   L   L   X   X   L   L    LLLH    LHHL   L   X   HOLD LINES & SCROL
  C   H    XXX   L   L   X   X   L   L    LHHL    LLLH   L   X   SWAP LINES & SCROL
  C   L    XXX   L   X   X   X   H   L    LHHH    LLHL   L   X   INC LINES & SCROL
  C   X    XXX   X   X   L   X   X   L    XXXX    XXXX   X   H   SET OE
  C   X    XXX   X   X   H   X   X   L    XXXX    XXXX   X   L   CLEAR OE
-------------------------------------------------------------------------------
```

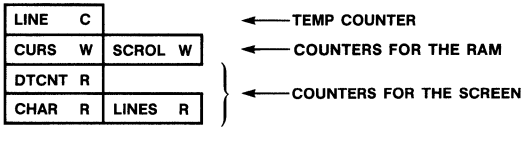
DESCRIPTION

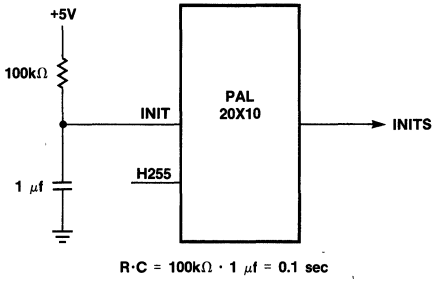
"SCROL" AND "LINES" ARE COUNTERS AND POINTERS TO THE RAM. "LINES" IS A POINTER TO THE LINE THAT IS READ FROM THE RAM. "SCROL" IS A POINTER TO THE LOCATION IN THE RAM WHERE A NEW LINE CAN BE STORED. BOTH OF THEM COUNT UP TO A MAXIMUM OF 16 LINES.

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

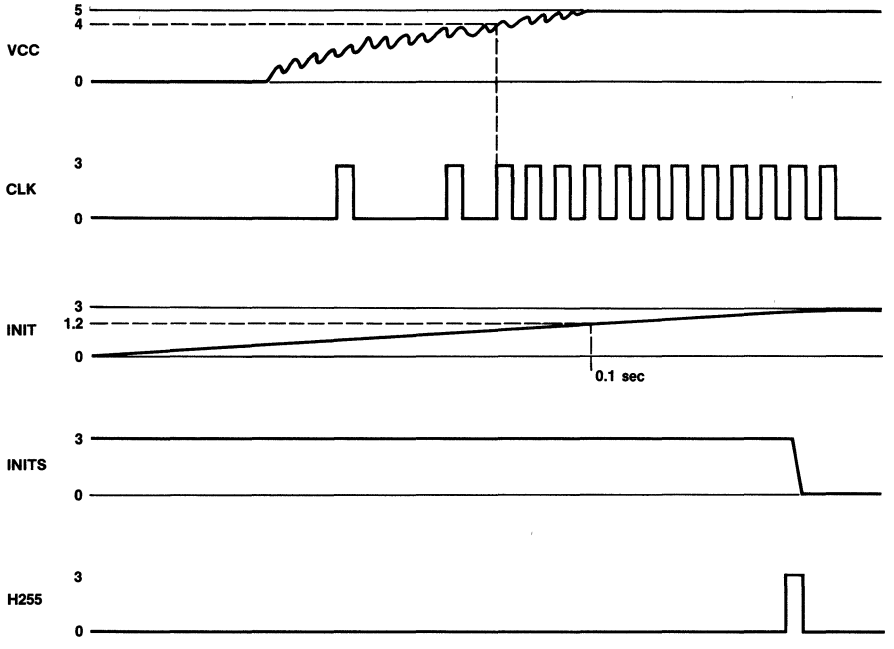
THE NEXT FIGURE SHOWS ALL THE POINTERS THAT HAVE BEEN DESCRIBED.

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



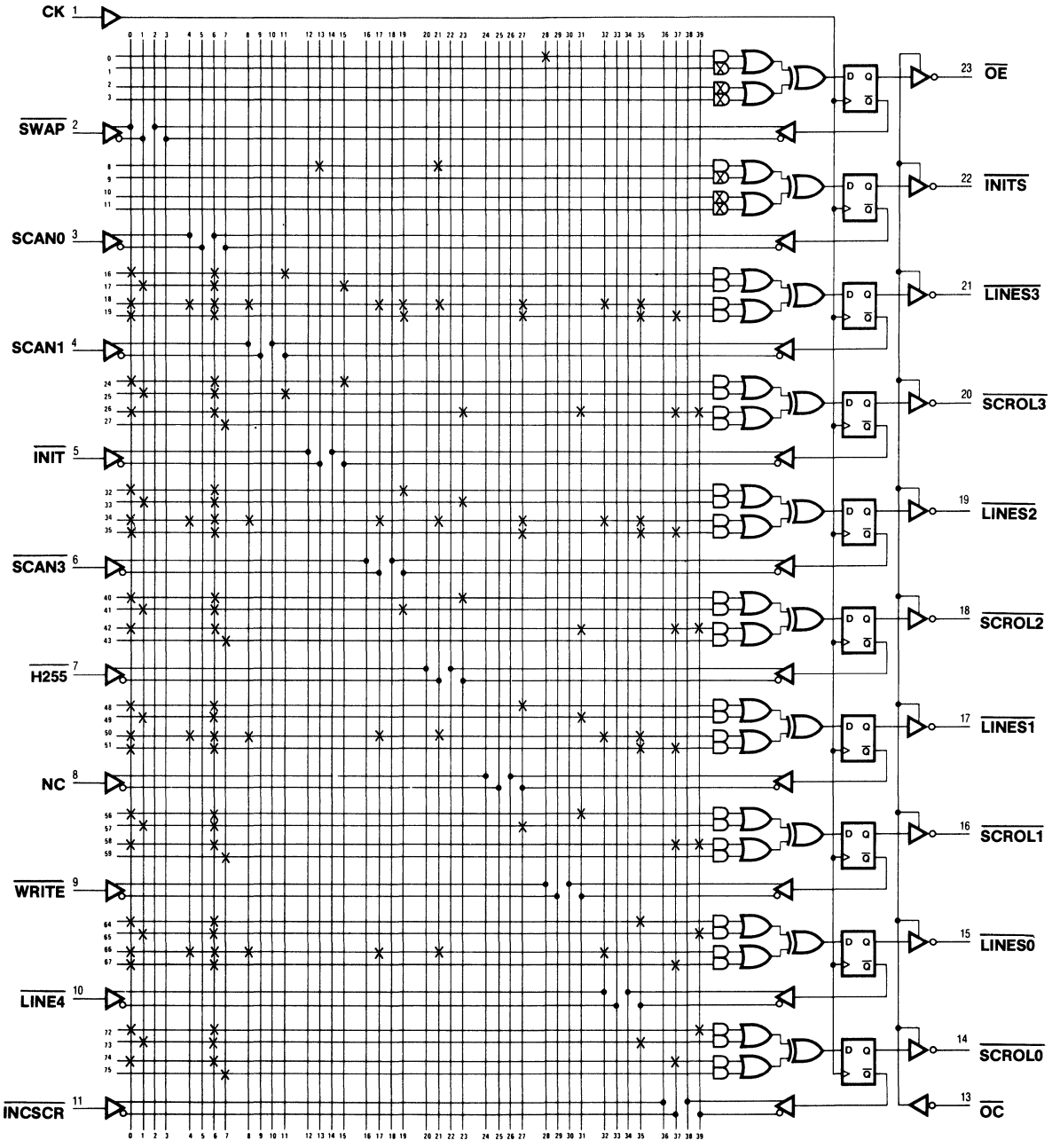


Initialization of the system when power (+5V) is turned on.



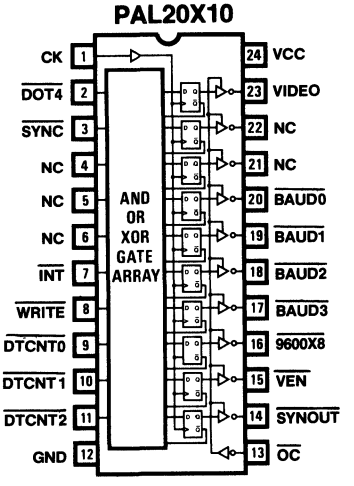
LINES/SCROL Generator

Logic Diagram PAL20X10




IMPORTANT
NOTES

Composite Video/Baud Rate Generator



7

```
PAL20X8                                          PAL DESIGN  SPECIFICATION
VP5                                              BIRKNER/KAZMI/UDI 7/14/81
COMPOSITE VIDEO/BAUD RATE GENERATOR
MMI SUNNYVALE, CALIFORNIA
CK /DOT4 /SYNC   NC  NC  NC   INT  /WRITE /DTCNT0 /DTCNT1 /DTCNT2 GND
/OC /SYNOUT  /UEN /9600X8 /BAUD3 /BAUD2 /BAUD1 /BAUD0 NC NC VIDEO VCC


IF( SYNC )  SYNOUT =  SYNC                       ;SYNC PULSE

IF( /DOT4 ) /VIDEO = /DOT4                       ;PIXEL TO THE SCREEN

9600X8 := INT                                    ;TO INITIALIZE COUNTER BAUD
        + BAUD3*BAUD2*DTCNT2*DTCNT1*/DTCNT0      ;104/8 = 13, MODULE 102
                                                 ;COUNTS 103 = 12 6/8
BAUD0 := /9600X8*BAUD0                           ;HOLD
       + /9600X8*BAUD0                           ;EXTEND
      :+:/9600X8*DTCNT2*DTCNT1*DTCNT0            ;INC

BAUD1 := /9600X8*BAUD1                           ;HOLD
       + /9600X8*BAUD1                           ;EXTEND
      :+:/9600X8*DTCNT2*DTCNT1*DTCNT0*BAUD0      ;INC

BAUD2 := /9600X8*BAUD2                           ;HOLD
       + /9600X8*BAUD2                           ;EXTEND
      :+:/9600X8*DTCNT2*DTCNT1*DTCNT0            ;INC
                *BAUD1*BAUD0

BAUD3 := /9600X8*BAUD3                           ;HOLD
       + /9600X8*BAUD3                           ;EXTEND
      :+:/9600X8*DTCNT2*DTCNT1*DTCNT0            ;INC
                *BAUD2*BAUD1*BAUD0

UEN   := WRITE*/DTCNT2*DTCNT1                    ;DTCNT = 2,3
       + WRITE* DTCNT2*/DTCNT1*/DTCNT0           ;DTCNT = 4
```

FUNCTION TABLE

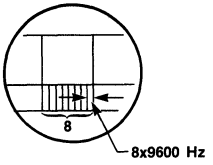
CK DOT4 SYNC INT WRITE DTCNT2 DTCNT1 DTCNT0 /OC SYNOUT UEN
9600X8 BAUD3 BAUD2 BAUD1 BAUD0 VIDEO

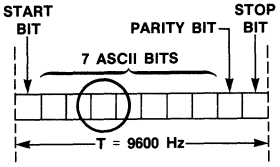
;												
;				W				S		9		
;		D	S	R				Y		6		V
;		O	Y	I		/	O	N		0		I
;	C	T	N	T	DTCNT	E	U	U	E	0	BAUD	D	E
;	K	4	C	E	210	N	T	T	N	8	3210	O	COMMENTS
	L	X	H	X	XXX	L	H	X	X	XXXX	X	CHECK SYNOUT FOR H
	L	X	L	X	XXX	L	L	X	X	XXXX	X	CHECK SYNOUT FOR L
	L	L	X	X	XXX	L	X	X	X	XXXX	L	VIDEO = L
	L	H	X	X	XXX	L	X	X	X	XXXX	H	VIDEO = H
	C	X	X	H	X	LLL	L	X	X	H	XXXX	X	SET 9600X8
	C	X	X	L	X	LLL	L	X	X	L	LLLL	X	INITIALIZE BAUD COUNTER
	C	X	X	L	X	HHH	L	X	X	L	LLLH	X	INC BAUD
	C	X	X	L	X	HHH	L	X	X	L	LLHL	X	INC BAUD
	C	X	X	L	X	HHH	L	X	X	L	LLHH	X	INC BAUD
	C	X	X	L	X	HHH	L	X	X	L	LHLL	X	INC BAUD
	C	X	X	L	X	HLH	L	X	X	L	LHLL	X	HOLD BAUD: DTCNT NEQ HHH
	C	X	X	L	X	HHH	L	X	X	L	LHLH	X	INC BAUD
	C	X	X	L	X	HHH	L	X	X	L	LHHL	X	INC BAUD
	C	X	X	L	X	HHH	L	X	X	L	LHHH	X	INC BAUD
	C	X	X	L	X	HHH	L	X	X	L	HLLL	X	INC BAUD
	C	X	X	L	X	HHH	L	X	X	L	HLLH	X	INC BAUD
	C	X	X	L	X	HHH	L	X	X	L	HLHL	X	INC BAUD
	C	X	X	L	X	HHH	L	X	X	L	HLHH	X	INC BAUD
	C	X	X	L	X	HHH	L	X	X	L	HHLL	X	INC BAUD
	C	X	X	L	H	LLL	L	X	L	L	HHLL	X	HOLD BAUD
	C	X	X	L	H	LLH	L	X	L	L	HHLL	X	HOLD BAUD
	C	X	X	L	H	LHL	L	X	H	L	HHLL	X	HOLD BAUD, SET UEN
	C	X	X	L	H	LHH	L	X	H	L	HHLL	X	HOLD BAUD, SET UEN
	C	X	X	L	H	HLL	L	X	H	L	HHLL	X	HOLD BAUD, SET UEN
	C	X	X	L	H	HLH	L	X	L	L	HHLL	X	HOLD BAUD
	C	X	X	L	H	HHL	L	X	L	H	HHLL	X	SET 9600X8, HOLD BAUD
	C	X	X	L	H	HHH	L	X	L	L	LLLL	X	INITIALIZE BAUD

DESCRIPTION

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

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

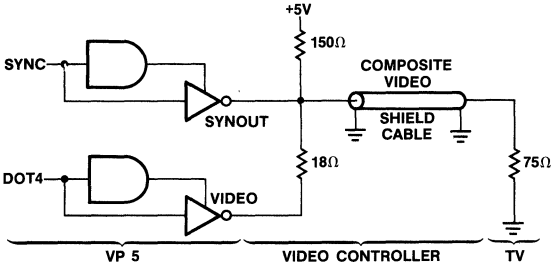




T = 1/76800 = 13 MICROSECOND
THE CLOCK FREQUENCY IS 8000000 Hz.
WE NEED TO DIVIDE THE CLOCK FREQUENCY BY 104 TO GENERATE A FREQUENCY OF 76800 Hz.

8000000/76800 = 104 = 13*8

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



$\frac{18}{150 + 18} * 4.6 = 0.49V$

$\frac{75}{150 + 75} * 5 = 1.66V$ When SYNC = H the output is at 0.4V.

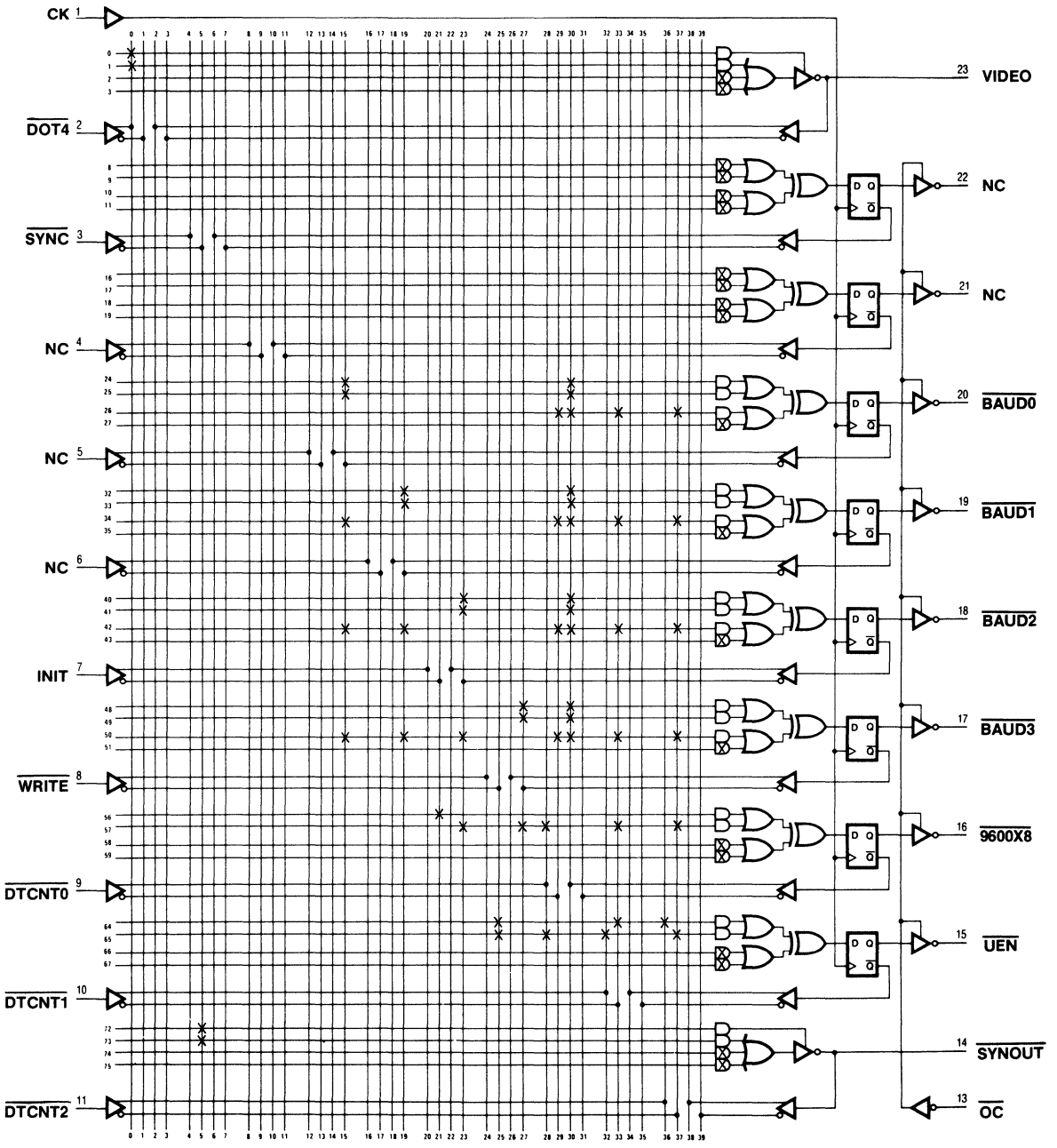
DIGITAL TO ANALOG CONVERTER

SYNOUT	VIDEO	COMPOSITE VIDEO
L	L	1.7V
L	H	0.9V
H	L	0.4V
H	H	0.4V

COMPOSITE VIDEO SIGNAL

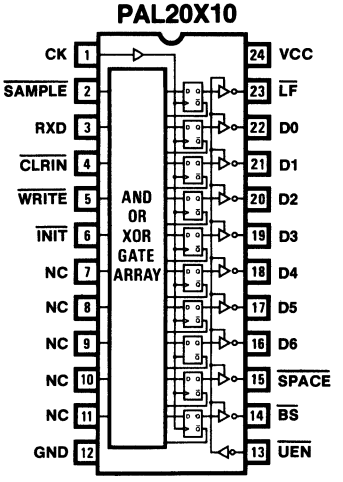
Composite Video/Baud Rate Generator

Logic Diagram PAL20X8




IMPORTANT
NOTES

UART Shift Register and Control Key Detect



7

```
PAL20X10                                        PAL DESIGN SPECIFICATION
VP6                                             BIRKNER/UDI      21/7/81
UART SHIFT REGISTER AND CONTROL KEY DETECT
MMI SUNNYVALE, CALIFORNIA
CK /SAMPLE RXD /CLRLIN /WRITE  INIT NC NC NC NC NC GND
/UEN /BS  /SPACE  D6  D5  D4  D3  D2  D1  D0  /LF  VCC


/D0 := /D0*/SAMPLE                          ;HOLD
     + SPACE                                ;SET SPACE CODE
    :+:/D1* SAMPLE                          ;SHIFT

/D1 := /D1*/SAMPLE                          ;HOLD
     + SPACE                                ;SET SPACE CODE
    :+:/D2* SAMPLE                          ;SHIFT

/D2 := /D2*/SAMPLE                          ;HOLD
     + SPACE                                ;SET SPACE CODE
    :+:/D3* SAMPLE                          ;SHIFT

/D3 := /D3*/SAMPLE                          ;HOLD
     + SPACE                                ;SET SPACE CODE
    :+:/D4* SAMPLE                          ;SHIFT

/D4 := /D4*/SAMPLE                          ;HOLD
     + SPACE                                ;SET SPACE CODE
    :+:/D5* SAMPLE                          ;SHIFT

/D5 := /INIT*/D5*/SAMPLE*/SPACE             ;SET SPACE CODE INSTEAD
     + /D5*/SAMPLE*/SPACE                   ;OF ANY CONTROL CODE
    :+:/INIT*/D6*SAMPLE*/SPACE              ;SHIFT

/D6 := /INIT*/D6*/SAMPLE                    ;HOLD
     + SPACE                                ;SET SPACE CODE
    :+:/INIT* RXD*SAMPLE                    ;DATA IS SHIFTED IN


SPACE := WRITE*/D6*/D5                      ;DETECT CTRL CHAR
       + CLRLIN                             ;AND CLEAR LINE

 LF := /D6*/D5*/D4*D3*/D2*D1*/D0            ;LINE FEED = HEX 0A
          *WRITE                            ;LATCH ON WRITE
       + LF*WRITE                           ;HOLD DURING WRITE
```

FUNCTION TABLE

CK SAMPLE RXD CLRLIN WRITE INIT /UEN /BS SPACE D6 D5 D4 D3 D2 D1 D0 LF

; CK	S AM PLE	RXD	CLR LIN	WRI TE	INI T	/UEN	/BS	SPA CE	DATA D6-----D0	OUT	L F	COMMENTS
C	X	X	L	L	L	L	X	L	XXXXXXX		X	INITIALIZE "SPACE"
C	H	X	L	L	H	L	X	L	HHXXXXX		X	INITIALIZE D6 AND D5
C	X	X	L	L	L	L	X	L	XXXXXXX		X	HOLD "SPACE" LO
C	H	L	L	X	L	L	X	L	HXXXXXX		X	SHIFT IN 1'ST BIT OF "U" CODE
C	H	H	L	X	L	L	X	L	LHXXXXX		X	SHIFT IN 2'ND BIT OF "U" CODE
C	H	L	L	X	L	L	X	L	HLHXXXX		X	SHIFT IN 3'RD BIT OF "U" CODE
C	H	H	L	X	L	L	X	L	LHLHXXX		X	SHIFT IN 4'TH BIT OF "U" CODE
C	H	L	L	X	L	L	X	L	HLHLHXX		X	SHIFT IN 5'TH BIT OF "U" CODE
C	H	H	L	X	L	L	X	L	LHLHLHX		X	SHIFT IN 6'TH BIT OF "U" CODE
C	H	L	L	X	L	L	X	L	HLHLHLH		X	OUTPUT "U" CODE, STORE IN RAM
C	H	H	L	X	L	L	X	L	LHLHLHL		X	SHIFT IN 1'ST BIT OF "LF" CODE
C	H	L	L	X	L	L	X	L	HLHLHLH		X	SHIFT IN 2'ND BIT OF "LF" CODE
C	H	H	L	X	L	L	X	L	LHLHLHL		X	SHIFT IN 3'RD BIT OF "LF" CODE
C	H	L	L	X	L	L	X	L	HLHLHLH		X	SHIFT IN 4'TH BIT OF "LF" CODE
C	H	H	L	X	L	L	X	L	LHLHLHL		X	SHIFT IN 5'TH BIT OF "LF" CODE
C	H	H	L	X	L	L	X	L	LLHLHLH		X	SHIFT IN 6'TH BIT OF "LF" CODE
C	H	H	L	L	L	L	X	L	LLLHLHL		X	DETECT "LF" CODE
C	L	X	L	H	L	L	X	H	LLLHLHL		H	HOLD DURING WRITE
C	L	X	L	L	L	L	X	L	LHLLLLL		X	WRITE "SPACE" CODE INTO RAM

7

DESCRIPTION

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

ASCII Code System and Character Set

D6, D5 — CONTROL CHARACTERS ARE DETECTED WHEN D6, D5 = 0,0

8 7 6 5 / 4 3 2 1	0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1
0 0 0 0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	STX	DC2	"	2	B	R	b	r
0 0 1 1	ETX	DC3	#	3	C	S	c	s
0 1 0 0	EOT	DC4	$	4	D	T	d	t
0 1 0 1	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	ACK	SYN	&	6	F	V	f	v
0 1 1 1	BEL	ETB	'	7	G	W	g	w
1 0 0 0	BS	CAN	(	8	H	X	h	x
1 0 0 1	HT	EM	)	9	I	Y	i	y
1 0 1 0	LF	SUB	*	:	J	Z	j	z
1 0 1 1	VT	ESC	+	;	K	[	k	{
1 1 0 0	FF	FS	,	<	L	\	l	|
1 1 0 1	CR	GS	-	=	M	]	m	}
1 1 1 0	SO	RS	.	>	N	^	n	~
1 1 1 1	SI	US	/	?	O	—	o	DEL

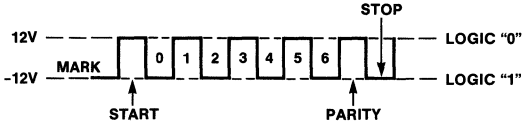
Printable Characters

Printer Control Characters

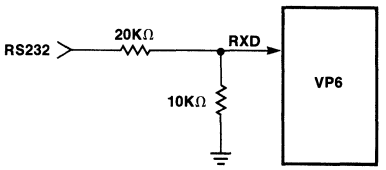
Codes Generated and Transmitted by the Terminal

ASR Control Codes

Extended Line Control



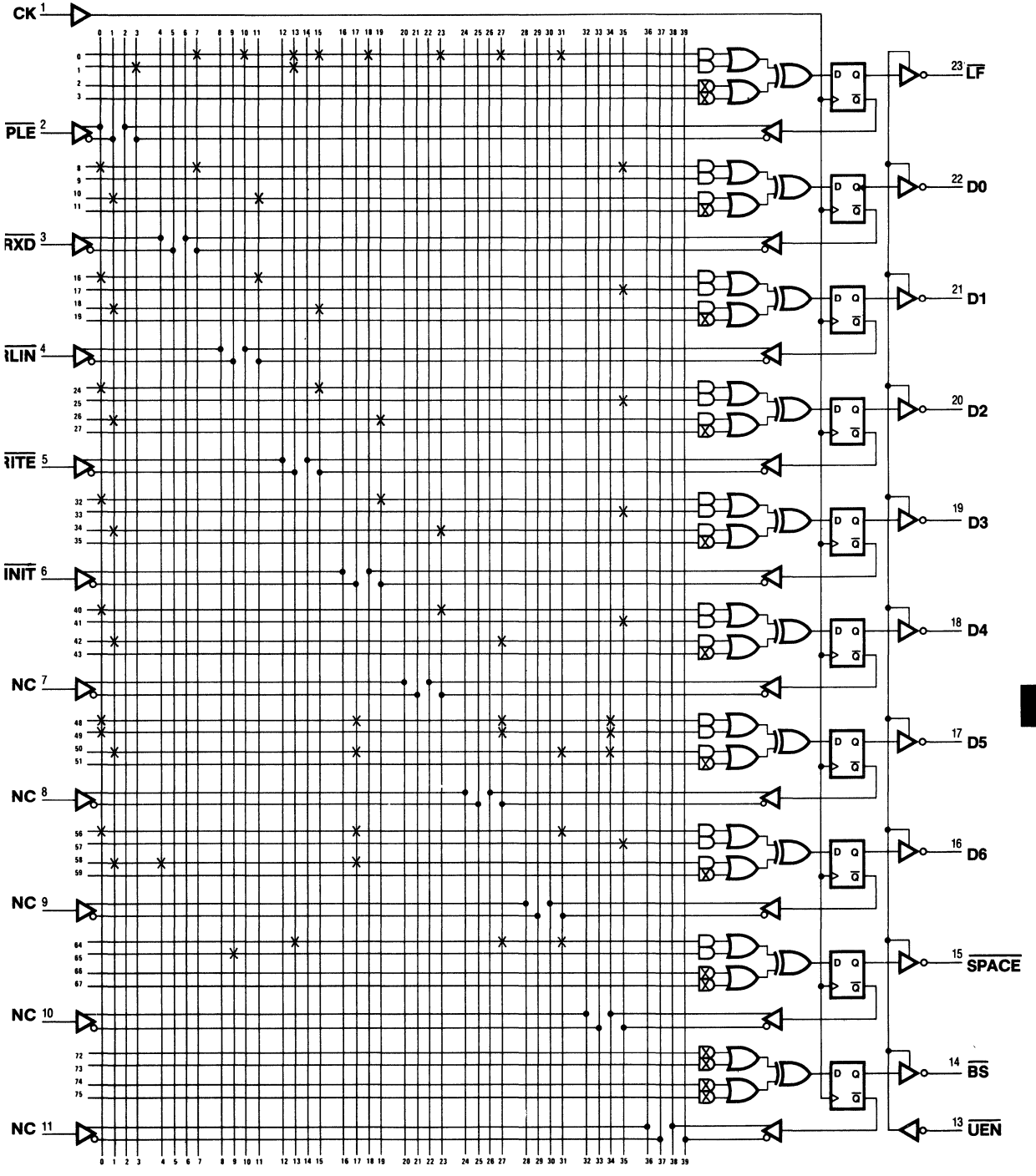
POLARITY CONVENTION FOR RS232

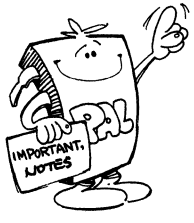


The voltages of the RS232 signals are between +12V and -12V. In order to get a logic one at 4V we built the above circuit.

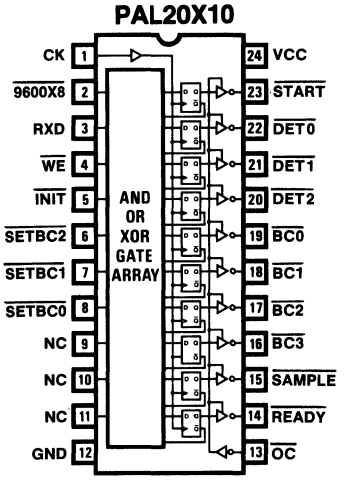
UART Shift Register and Control Detect

Logic Diagram PAL20X10




IMPORTANT
NOTES

UART Control



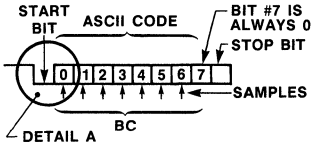
7

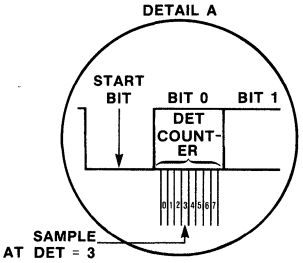
```
PAL20X10                                          PAL DESIGN SPECIFICATION
VP7                                               BIRKNER/UDI      7/28/81
UART CONTROL
MMI SUNNYVALE, CALIFORNIA
CK /9600X8  RXD /WE  /INIT /SETBC2 /SETBC1 /SETBC0  NC  NC  NC  GND
/OC /READY /SAMPLE /BC3 /BC2 /BC1 /BC0 /DET2 /DET1 /DET0 /START VCC


START := /READY*START                             ;HOLD
       + /READY*RXD                               ;DETECT START BIT
      :+:/READY*START*BC3*BC2*BC1*BC0             ;FILTER FALSE START
                    */DET2*/DET1*DET0*/RXD        ;CANCEL START IF NO RXD

DET0 := START*DET0                                ;HOLD
      + START*DET0                                ;EXTEND
     :+:START*9600X8                              ;CARRY

DET1 := START*DET1                                ;HOLD
      + START*DET1                                ;EXTEND
     :+:START*9600X8*DET0                         ;CARRY

DET2 := START*DET2                                ;HOLD
      + START*DET2                                ;EXTEND
     :+:START*9600X8*DET0*DET1                    ;CARRY

BC0  := /SETBC0*BC0                               ;HOLD
      + /SETBC0*READY                             ;SET BC TO -1 ON READY
     :+:/SETBC0*START*9600X8*DET0*DET1*DET2       ;CARRY
      +  SETBC0                                   ;SET BC TO 7 FOR TESTING

BC1  := /SETBC1*BC1                               ;HOLD
      + /SETBC1*READY                             ;SET BC TO -1 ON READY
     :+:/SETBC1*START*9600X8*DET0*DET1*DET2       ;CARRY
                          *BC0
      +  SETBC1                                   ;SET BC TO 7 FOR TESTING

BC2  := /SETBC2*BC2                               ;HOLD
      + /SETBC2*READY                             ;SET BC TO -1 ON READY
     :+:/SETBC2*START*9600X8*DET0*DET1*DET2       ;CARRY
                          *BC0*BC1
      +  SETBC2                                   ;SET BC TO 7 FOR TESTING

BC3  := BC3                                       ;HOLD
      + READY                                     ;SET BC TO -1 ON READY
     :+:START*9600X8*DET0*DET1*DET2               ;CARRY
                          *BC0*BC1*BC2

SAMPLE := START*9600X8*/DET2*DET1*/DET0*/BC3      ;DET=2 & BC=0..7
        + START*9600X8*/DET2*DET1*/DET0*/BC3      ;EXTEND
       :+:START*9600X8*/DET2*DET1*/DET0           ;CANCEL BC = 7
                    */BC3*BC2*BC1*BC0

READY := /INIT*/WE*READY                          ;HOLD
       + /INIT*/WE*START*BC3*/BC2*/BC1*/BC0       ;SET ON BC=8
      :+: INIT                                    ;INITIAL READY, START & BC
```

FUNCTION TABLE

```
CK 9600X8 RXD /WE INIT SETBC2 SETBC1 SETBC0 /OC READY
SAMPLE  BC3  BC2  BC1  BC0   DET2  DET1  DET0   START

;    9                          S
;    6                    R     A              S
;    0        I           E     M              T
;    0  R  /  N        /  A     P              A
; C  X  X  W  I  SETBC  O  D     L   BC   DET   R
; K  8  D  E  T   210   C  Y     E  3210  210   T   COMMENTS
-------------------------------------------------------------------------------
  C  X  X  X  H   XXX   L  H     X  XXXX  XXX   X   INITIAL READY
  C  X  L  H  L   XXX   L  H     X  HHHH  XXX   L   CLEAR START & INITIAL BC
  C  X  L  L  L   XXX   L  L     X  HHHH  LLL   L   INITIAL DET & CLEAR READY
  C  X  H  L  L   XXX   L  L     X  HHHH  LLL   H   SET START
  C  H  L  L  L   LLL   L  L     L  HHHH  LLH   H   NO RXD, DETECT FALSE START
  C  H  L  L  L   LLL   L  L     L  HHHH  LHL   L   CLEAR START (NOISE ON RXD)
  C  H  H  L  L   LLL   L  L     L  HHHH  LLL   H   INITIALIZE DET
  C  H  H  L  L   LLL   L  L     L  HHHH  LLH   H   NOW IT IS A REAL SIGNAL
  C  H  H  L  L   LLL   L  L     L  HHHH  LHL   H   INC DET, NO SAMPLING
  C  H  H  L  L   LLL   L  L     L  HHHH  LHH   H   NO SAMPLE, START BIT OF INFO
  C  H  H  L  L   LLL   L  L     L  HHHH  HLL   H   INC DET, NO SAMPLING
  C  H  H  L  L   LLL   L  L     L  HHHH  HLH   H   INC DET, NO SAMPLING
  C  H  H  L  L   LLL   L  L     L  HHHH  HHL   H   INC DET, NO SAMPLING
  C  H  H  L  L   LLL   L  L     L  HHHH  HHH   H   INC DET, NO SAMPLING
  C  H  H  L  L   LLL   L  L     L  LLLL  LLL   H   INC DET AND BIT COUNTER
  C  H  H  L  L   LLL   L  L     L  LLLL  LLH   H   1'ST BIT OF ASCII CODE
  C  H  H  L  L   LLL   L  L     L  LLLL  LHL   H   INC DET
  C  H  H  L  L   LLL   L  L     H  LLLL  LHH   H   SAMPLE THE 1'ST INFO BIT
  C  H  H  L  L   LLL   L  L     L  LLLL  HLL   H   INC DET
  C  H  H  L  L   LLL   L  L     L  LLLL  HLH   H   INC DET
  C  H  H  L  L   LLL   L  L     L  LLLL  HHL   H   INC DET
  C  H  H  L  L   LLL   L  L     L  LLLL  HHH   H   INC DET
  C  H  H  L  L   LLL   L  L     L  LLLH  LLL   H   INC DET AND BIT COUNTER
  C  H  H  L  L   LLL   L  L     L  LLLH  LLH   H   2'ND BIT OF ASCII CODE
  C  H  H  L  L   LLL   L  L     L  LLLH  LHL   H   INC DET
  C  H  H  L  L   LLL   L  L     H  LLLH  LHH   H   SAMPLE 2'ND BIT OF INFO
  C  H  H  L  L   LLL   L  L     L  LLLH  HLL   H   INC DET
  C  H  H  L  L   LLL   L  L     L  LLLH  HLH   H   INC DET
  C  H  H  L  L   LLL   L  L     L  LLLH  HHL   H   INC DET
  C  H  H  L  L   LLL   L  L     L  LLLH  HHH   H   INC DET
  C  H  H  L  L   HLH   L  L     L  LHHH  LLL   H   SET BIT COUNTER TO 7
  C  H  H  L  L   LLL   L  L     L  LHHH  LLH   H   7'TH BIT OF ASCII CODE
  C  H  H  L  L   LLL   L  L     L  LHHH  LHL   H   INC DET
  C  H  H  L  L   LLL   L  L     L  LHHH  LHH   H   NO SAMPLE 7'TH BIT ALWAYS 0
  C  H  H  L  L   LLL   L  L     L  LHHH  HLL   H   INC DET
  C  H  H  L  L   LLL   L  L     L  LHHH  HLH   H   INC DET
  C  H  H  L  L   LLL   L  L     L  LHHH  HHL   H   INC DET
  C  H  H  L  L   LLL   L  L     L  LHHH  HHH   H   INC DET
  C  H  H  L  L   LLL   L  L     L  HLLL  LLL   H   INC DET AND BIT COUNTER
  C  H  H  H  L   LLL   L  H     L  HLLL  LLH   H   SET THE READY SIGNAL
  C  H  H  L  L   LLL   L  L     L  HHHH  LHL   L   SET START BIT & INITIAL BC
  C  H  H  L  L   LLL   L  L     L  HHHH  LLL   H   REPEATE FOR NEXT CHARACTER
-------------------------------------------------------------------------------
```

DESCRIPTION





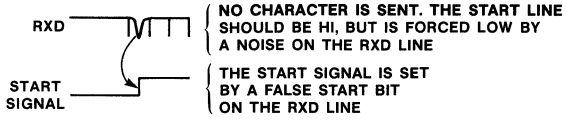
"BC" IS A COUNTER FOR THE ASCII CODE BITS. A SAMPLE FROM EACH ASCII BIT IS TAKEN WHEN "DET" = 3.

THE "START" SIGNAL IS SET WHEN A START BIT IS DETECTED ON THE RXD LINE AND REMAINS SET UNTIL THE LAST ASCII BIT OF THE CHARACTER THAT IS SAMPLED. AFTER ALL THE 7 BITS OF THE ASCII CODE WERE SAMPLED, A WRITE SIGNAL IS SENT TO THE RAM AND THE 7 BITS (A CODE FOR A CHARACTER) ARE WRITTEN IN PARALLEL INTO THE RAM.

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

BUT

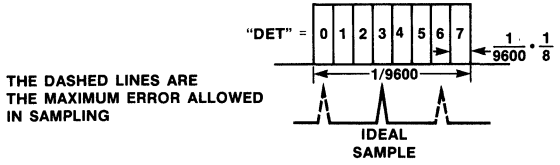
ASSUME THAT A NOISE OCCURS ON THE RXD LINE WHICH SET THE "START" SIGNAL.



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

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

SINCE BOTH A TRANSMITTER AND A RECEIVER ARE USED IN OBTAINING INFORMATION,
AN ERROR CAN OCCUR THAT INVOLVES BOTH OF THESE COMPONENTS.
ASSUME: ERROR IN TRANSMITTING FREQUENCY = EX
ERROR IN RECEIVING FREQUENCY = ER
THEN THE TOTAL ERROR FOR ONE BIT OF INFORMATION IS (EX + ER).
AND THE TOTAL ERROR FOR THE N'TH BIT OF INFORMATION IS N*(EX + ER).
WHEN COUNTER "DET" IS EQUAL 3, WE HAVE THE IDEAL BIT FOR SAMPLING.
THE MAXIMUM ERROR THAT IS ALLOWED DUE TO THE TOLERANCES IN THE FREQUENCIES
IN BOTH TRANSMITTER AND RECEIVER, WILL BE WHEN THE SAMPLE IS TAKEN AT
"DET" = 0 OR AT "DET" = 6, AS SHOWN HERE:



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

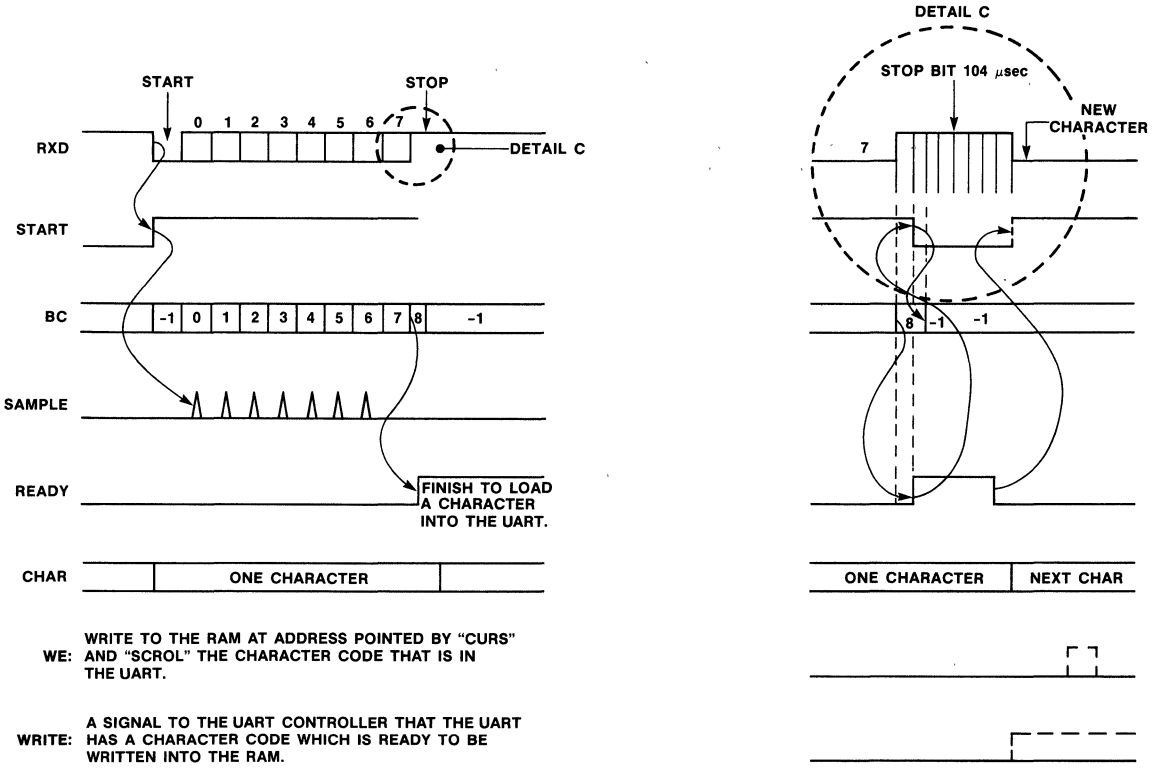
THE TOTAL ERROR ALLOWED FOR THE 7'TH BIT OF INFORMATION IS:

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

THE TOTAL ERROR ALLOWED FOR ONE BIT OF INFORMATION IS:

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

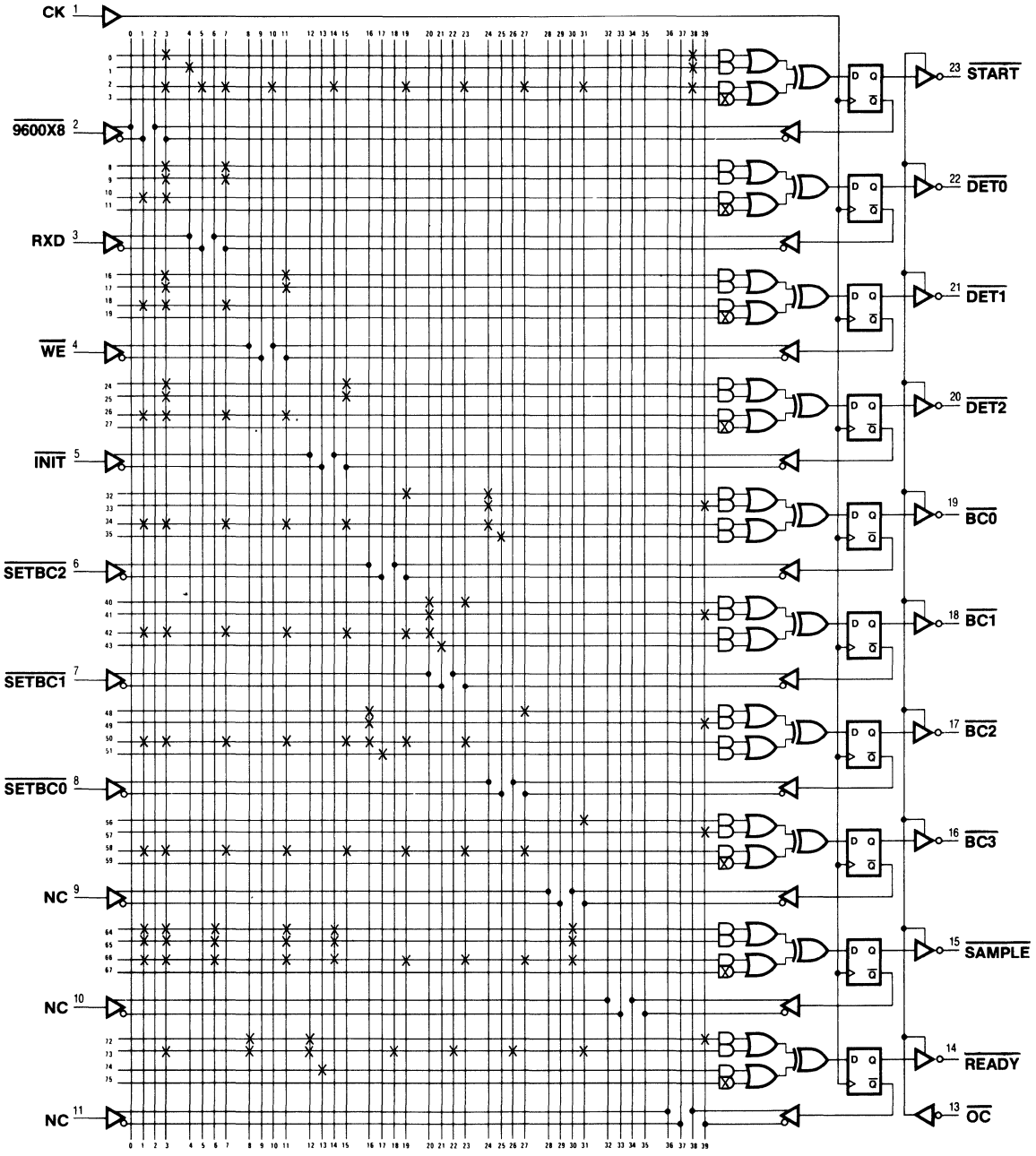
THE FREQUENCY RATE BETWEEN RS232 AND THE RXD LINE FOR EVERY BIT OF
INFORMATION SHOULD BE BETWEEN 10150 Hz AND 9130 Hz.



Timing diagram for detecting a character and writing it into the RAM. Signals "WRITE" and "WE" are explained and derived in the next PAL specification VP8.

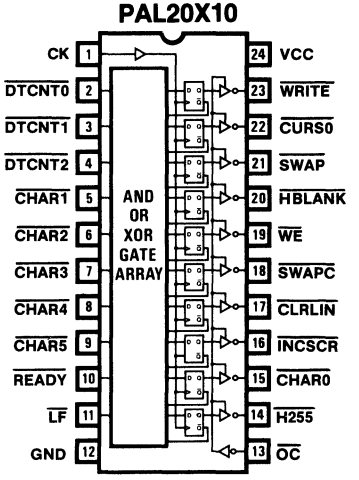
UART Control

Logic Diagram PAL20X10




IMPORTANT NOTES

RAM Control



7

```
PAL20X10                                          PAL DESIGN SPECIFICATION
VP8                                               BIRKNER/UDI      7/29/81
RAM CONTROL
MMI SUNNYVALE, CALIFORNIA
CK /DTCNT0 /DTCNT1 /DTCNT2 /CHAR1 /CHAR2 /CHAR3 /CHAR4 /CHAR5 /READY /LF GND
/OC /H255 /CHAR0 /INCSCR /CLRLIN /SWAPC /WE /HBLANK /SWAP /CURS0 /WRITE  VCC


WRITE:=READY*/DTCNT2*/DTCNT1*/DTCNT0*/H255                ;SET WHEN DTCNT=0
          *CHAR5*CHAR4*CHAR3*CHAR2*CHAR1*CHAR0            ;& CHAR=63
      + WRITE*/DTCNT2*/H255                               ;HOLD AT 0,1,2,3
     :+:WRITE* DTCNT2*/DTCNT1*/H255                       ;HOLD AT 4,5
      + CLRLIN*/DTCNT2*/DTCNT1*/DTCNT0                    ;SET AT DTCNT=0

SWAP := WRITE*/DTCNT2*/DTCNT1*DTCNT0                      ;SWAP AT DTCNT=1
      + WRITE* DTCNT2*/DTCNT1*/DTCNT0                     ;SWAP BACK AT 4

SWAPC:= WRITE*/DTCNT2*/DTCNT1*DTCNT0*/CLRLIN              ;SWAP AT DTCNT=1
      + WRITE* DTCNT2*/DTCNT1*/DTCNT0*/CLRLIN             ;SWAP BACK AT 4
     :+:CHAR5*CHAR4*/CHAR3*LF                             ;TEST CONDITION

WE   := WRITE*/DTCNT2*DTCNT1*DTCNT0                       ;ENABLE WRITE

INCSCR  := WRITE*DTCNT2*/DTCNT1*DTCNT0*/CLRLIN*/H255*LF   ;DETECT LINEFEED
         + WRITE* CHAR5*/CHAR4*CHAR3*CHAR2*CHAR1*CHAR0    ;CURS=47, LAST
                */CLRLIN*/H255                            ;VISIBLE CHAR ON
                * DTCNT2*/DTCNT1*DTCNT0                   ;LINE. DTCNT=5
        :+:CHAR5*CHAR4*/CHAR3*LF                          ;TEST CONDITION

H255    := CHAR5*CHAR4*CHAR3*CHAR2*CHAR1*CHAR0            ;END OF LINE
           *DTCNT2*DTCNT1*/DTCNT0                         ;DTCNT=6

CLRLIN := INCSCR                                          ;SET ON LINE END
        + CLRLIN*CHAR5*CHAR4*CHAR3*CHAR2*CHAR1*CHAR0      ;HOLD THIS WRITE
       :+:CLRLIN*/CHAR5                                   ;HOLD, CHAR=0-31
        + CLRLIN*/CHAR4                                   ;HOLD, CHAR=32-47

HBLANK := CHAR5*CHAR4                                     ;CHAR=48-63
        + CHAR5*CHAR4                                     ;EXTEND
       :+:CHAR5*CHAR4*/CHAR3*/CHAR2*/CHAR1*/CHAR0*/DTCNT2 ;CANCEL 48-48.5
        + CHAR5*CHAR4*/CHAR3*/CHAR2*/CHAR1*/CHAR0*/DTCNT1 ;CANCEL 48.5-48.75

CHAR0 :=  SWAPC*CURS0                                     ;SWAP WITH CURS
       + /SWAPC*CHAR0                                     ;HOLD
      :+:/SWAPC*DTCNT0*DTCNT1*DTCNT2                      ;INC

CURS0 :=  SWAPC*/INCSCR*CHAR0                             ;SWAP WITH CHAR
       + /SWAPC*/INCSCR*CURS0                             ;HOLD
      :+:/SWAPC*/INCSCR*WRITE*/CLRLIN*DTCNT2*DTCNT1*/DTCNT0 ;INC
```

FUNCTION TABLE

```
CK DTCNT2 DTCNT1 DTCNT0 CHAR5 CHAR4 CHAR3 CHAR2 CHAR1 CHAR0 READY
LF /OC   H255   INCSCR CLRLIN  SWAPC  /WE HBLANK  SWAP CURS0  WRITE
```

; C K	DTCNT 210	CHAR 54321	0	RE AD Y	L F	/ OC	H2 5C 5	INC SCR	CLR LIN	SWA PC	/WE	HBL ANK	SWP AP	CUR S0	WRI TE	COMMENTS
C	LLL	HHLXX	X	X	H	L	X	H	X	H	X	X	X	X	X	SET INCSCR,SWAPC
C	HHL	HHLXX	X	X	H	L	X	H	H	H	X	X	X	L	X	CLRLIN=H,CURS=L
C	LLL	HHLXX	X	X	H	L	L	H	H	H	X	X	X	L	H	SET WRITE
C	LLH	LLLXX	X	X	X	L	L	L	L	L	X	X	X	L	H	INCSCS,CLRLIN=L
C	HHL	LLXXX	X	L	X	L	L	L	L	L	X	X	X	X	L	SWAP CURS0,CHAR0
C	HHL	HHLXX	X	X	H	L	X	H	X	H	X	X	X	X	X	SET INCSCR,CWAPC
C	HHL	HHLXX	X	X	H	L	X	H	H	H	X	X	X	L	X	CLRLIN=H
C	LLL	HHLXX	X	X	H	L	L	H	H	H	X	X	X	X	H	SET WRITE SIGNAL
C	LLH	XXXXX	X	L	X	L	L	X	X	X	X	X	X	X	H	HOLD WRITE
C	LHL	XXXXX	X	L	X	L	L	X	X	X	H	X	X	X	H	HOLD WRITE
C	LHH	XXXXX	X	L	X	L	L	X	X	X	L	X	X	X	H	ENABLE WRITE
C	HLL	XXXXX	X	L	X	L	L	X	X	X	H	X	X	X	H	HOLD WRITE
C	HLH	XXXXX	X	L	X	L	L	X	X	X	X	X	X	X	H	HOLD WRITE
C	HHL	XXXXX	X	L	X	L	L	X	X	L	X	X	X	X	L	CLEAR WRITE SIGNAL
C	HHH	XXXXX	H	L	X	L	L	L	X	L	X	X	X	X	L	CHAR0=H, INCSCR=L
C	LLL	XXXXX	H	L	X	L	L	L	X	L	X	X	X	X	L	HOLD
C	LLH	XXXXX	H	L	X	L	L	L	X	L	X	X	X	X	L	HOLD
C	LHL	XXXXX	H	L	X	L	L	L	X	L	X	X	X	X	L	HOLD
C	LHH	XXXXX	H	L	X	L	L	L	X	L	X	X	X	X	L	HOLD
C	HLL	XXXXX	H	L	X	L	L	L	X	L	X	X	X	X	L	HOLD
C	HLH	XXXXX	H	L	X	L	L	L	X	L	X	X	X	X	L	HOLD
C	HHL	HHHHH	H	L	X	L	H	L	X	L	X	X	X	X	X	END OF LINE
C	HHH	HHLXX	L	L	L	L	L	L	L	L	X	X	X	X	X	CLEAR CLRLIN & H255
C	LLL	HHHXX	L	H	L	L	L	L	L	L	X	H	X	X	L	SET HBLANK
C	HHH	HHLXX	H	L	L	L	L	L	L	L	X	H	X	L	L	SET CURS0 LO
C	LLL	HHLLL	H	H	L	L	L	L	L	L	X	H	X	L	X	HBLANK=L
C	LHH	HHLXX	H	L	L	L	L	L	L	L	X	X	L	L	L	HOLD
C	HHL	HHLXX	H	L	H	L	L	H	L	H	X	X	L	L	L	SET INCSCR
C	XXX	HHLXX	L	L	L	L	L	L	H	X	X	X	X	L	L	SET CLRLIN
C	LLL	LLXXX	L	L	L	L	L	L	H	X	X	X	X	L	H	HOLD CLRLIN,SET WRITE
C	LLL	LLXXX	L	L	L	L	L	L	H	X	X	X	X	L	L	CLR WRITE,HOLD CLRLIN
C	LLL	HHLXX	L	L	L	L	L	L	L	X	X	X	X	L	H	CLEAR CLRLIN
C	LLH	XXXXX	L	L	L	L	L	L	L	H	X	X	H	L	H	SET SWAP & SWAPC

DESCRIPTION

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

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

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

"UEN" ENABLES THE UART.

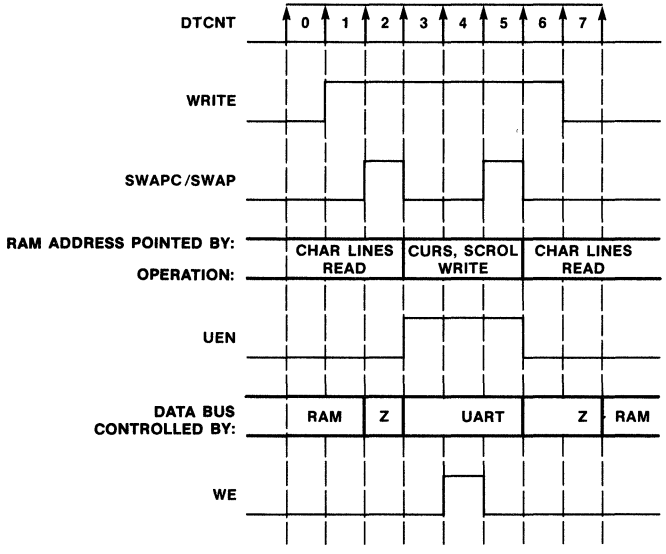
"WE" WRITES TO THE RAM THE DATA THAT IS IN THE UART REGISTER.

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

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

"H255" DETECTS END OF LINE.

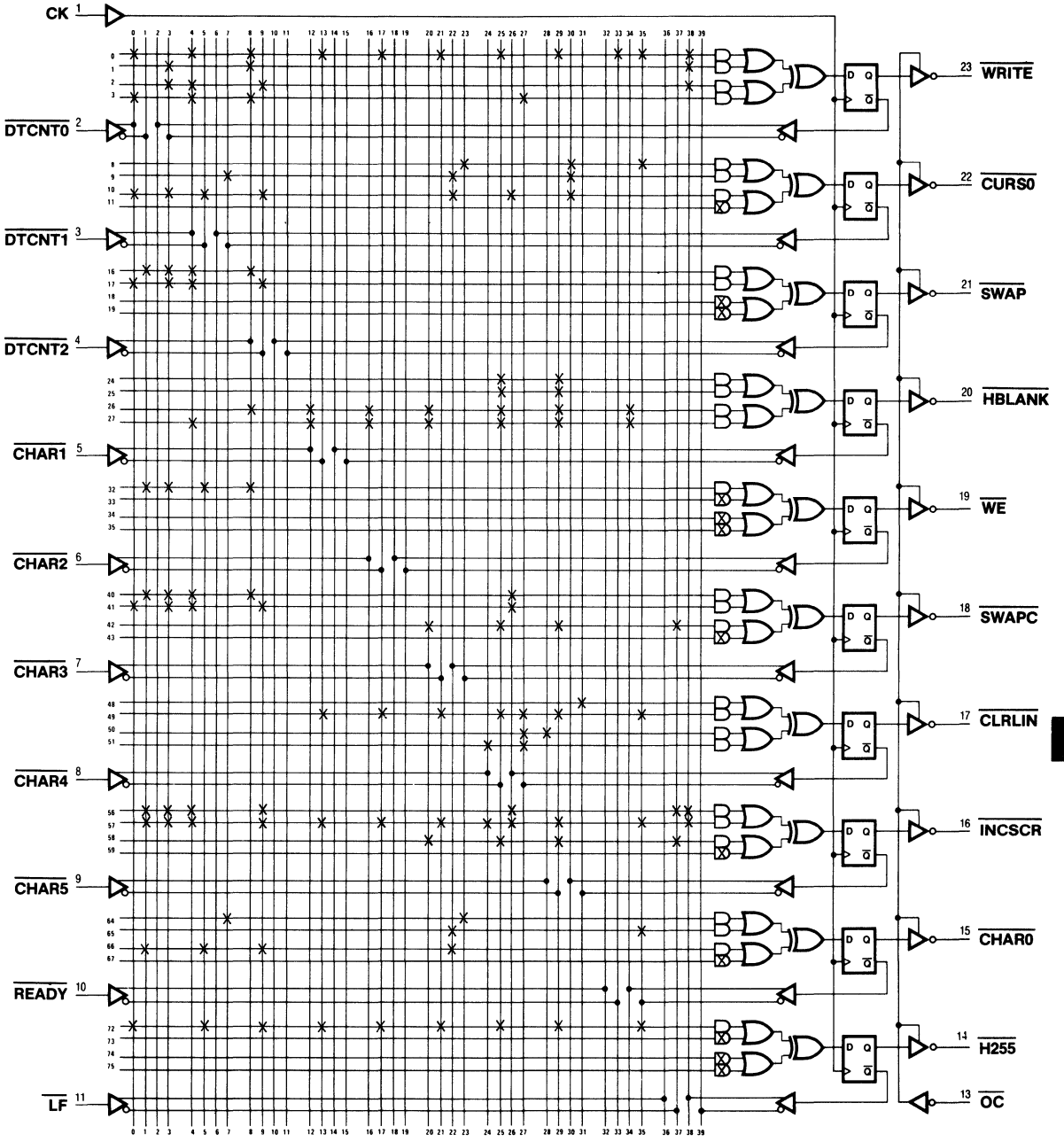
"HBLANK" BLANK OUT THE SCREEN WHEN "CHAR" IS BETWEEN 48 AND 63.



WRITE and READ signals to the RAM for one character code (when CHAR = 63). (Z = THREE STATE)

RAM Control

Logic Diagram PAL20X10



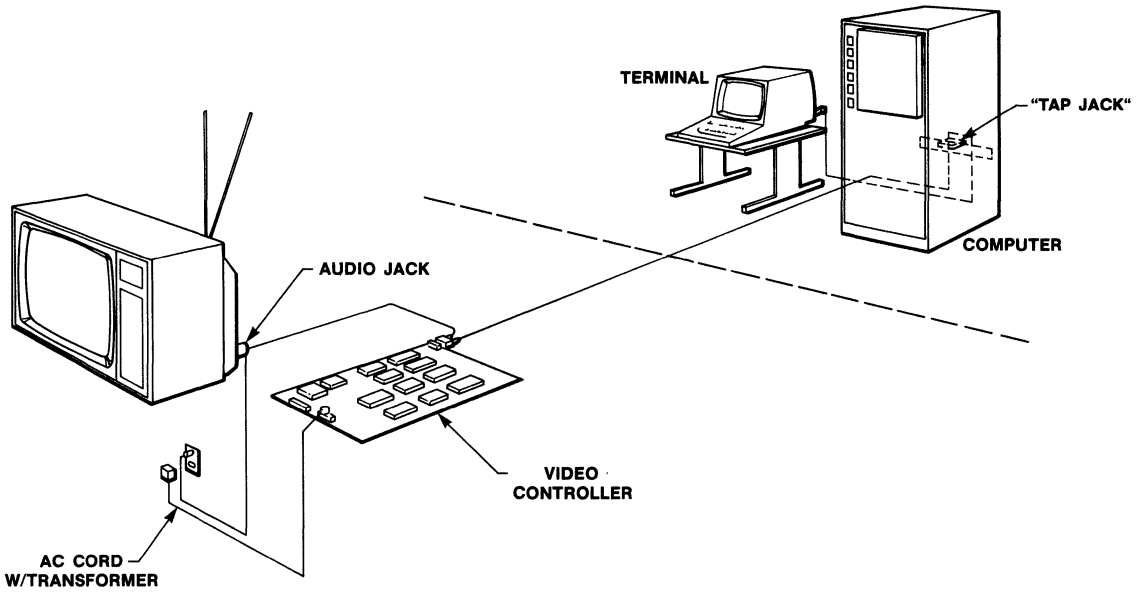
Summary

This paper conveys the message that a video-controller board can be designed in an efficient way by using PALs. The above video controller was developed and is used by Monolithic Memories, Inc.

We have presented the basic vocabulary and processes that are presently common to the industry with an explanation of how we developed our video-controller board.

Boolean transfer function equations, function tables, descriptions, simulations and fuse-plots are all offered within our PAL specifications. These were developed on the company in-house computer and are available from Monolithic Memories.

Video Controller Computer Interface



References

1. *Color TV Training Manual*
by Howard W. Sams
Howard W. Sams & Co., Inc.
The Bobb-Merrill Co., Inc.

2. *Television Fundamentals Theory, Circuit and Servicing*
Fowler and Lippert
McGraw Hill

3. *Principles of Interactive Computer Graphics*
by William M. Newman and Robert F. Sproull
McGraw Hill

4. *John Birkner, personal Communication, 1981*

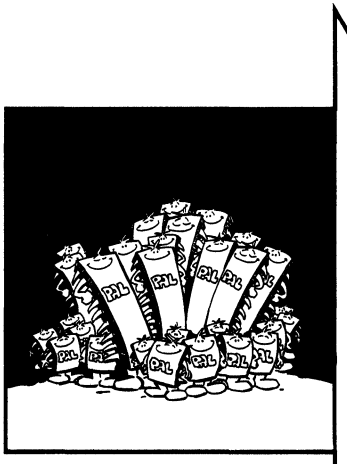


Table of Contents

High Speed/Low Cost Fuse Link Arrays Compete with TTL 74S/LS

John Birkner/Wescon 78

Random Logic has remained virtually impenetrable by today's state of the art LSI technology. This fact is dramatically evidenced by the high percentage of TTL 74S/LS logic found in currently introduced small board computers based on MOS microprocessors, and also high performance minicomputers based on bipolar bit slice chip sets

LSI has vastly reduced the well defined, regular logic functions found in these systems because of the wide appeal, thus, high volume and low cost. When the regular functions are applied to the specific needs of a particular systems manufacturer, however, individual interfacing requirements demand additional logic not available in LSI functions. The present solution is to use standard SSI/MSI logic functions at a high cost in chip count, assembly/test labor and reliability.

To answer the need for LSI Random Logic, a family of fuse Programmable Array Logic devices (PAL) has recently been introduced. This family meets the speed requirements of competing TTL SSI/MSI with typical 25ns propogation and set up delays while reducing chip count 4 to 1.

Total system cost using PALs is comparable with SSI/MSI, yet design time and inventory costs are reduced. The 15 part family utilizes the latest improvements in bipolar Schottky fusible link technology with the proven reliability track record of bipolar PROMs. Titanium-tungsten fuses and emitter follower arrays are used to achieve the 25ns delays through three levels of logic.

DEFINING THE PROBLEM

Increasing the logic function density per chip is the name of the game in the integrated circuit industry. The standard procedure is to identify a logical block which is common to many applications and users, and then to integrate that function onto a single chip. The problem with integrating the random and irregular functions which are so prevalent in today's digital systems is to identify a common denominator which can provide a generalized solution that meets cost and performance goals.

A Decade of TTL

The growth of TTL SSI/MSI during the last decade provides a wealth of knowledge about the elements that designers need to build digital systems. This family of devices was defined by many users and manufacturers over many years and is, therefore, an evolutionary, emperical data base from which we can extrapolate generalized logical primatives for LSI implementation.

The Primitives

SSI/MSI functions can be classified as either combinatorial or sequential. Combinatorial includes the basic gates, AND-OR-INVRT gates, Multiplexors, de-multiplexors and priority encoders. More complex combinatorial includes arithmetic functions such as comparitors, adders, and ALUs. The common denominator among the simple combinatorial elements is the sum of products An additional primative in the arithmetic elements is a high density of Exclusive-ORs.

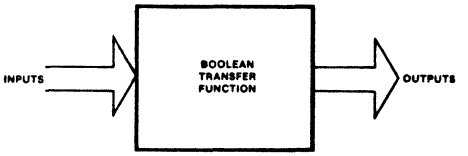


Figure 1: Combinatorial Function

Sequential includes all the registered functions such as counters, shift registers, registered multipliexors, accumulators, and registered encoders. The common denominator of these sequential elements is a register driven by a combinatorial element whose inputs are both external and internal from the register All of the sequential devices may be described as simple state machines.

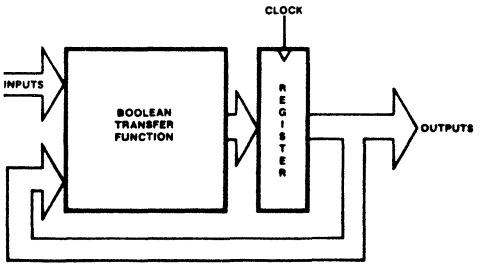


Figure 2: Sequential Function

Both the combinatorial and the sequential functions may have the optional characteristic of three-state outputs. That is, the output function may be disconnected as a driving source by placing the outputs in a high impedance state (High-Z)

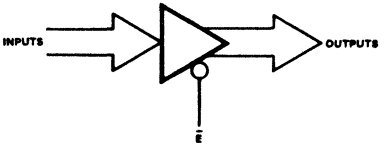


Figure 3: Three-State Output Gate

8

Definition of Terms

Before we can rigorously define the basic logical primitives, we first need a precise language which unambiguously describes the transfer functions.

Definitions:

= Equality, after the propogation time, t_{PD}, from any input change.

:= Replaced by, after the propogation delay, t_{PD}, from the low to high transition of the clock.

IF Conditional Equality, after the propogation delay, t_{PD}, from the enabling condition. Otherwide, high impedance (High-Z).

/ Complement, Boolean operator, placed to the left of a Boolean variable.

* AND, Boolean operator.

+ OR, Boolean operator.

:+: XOR, Boolean operator.

( ) Parenthetical separators.

hierarchical order / * + :+:
where / is evaluated first.

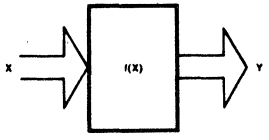
Given these definitions, we can formulate rigorous logical descriptions of standard SSI/MSI functions.

Rigorous Statement of Primitives

We can now state, in equation form, a generalized set of equations suitable for LSI implementation.

For an input vector X and an output vector Y, a combinatorial function is described as,

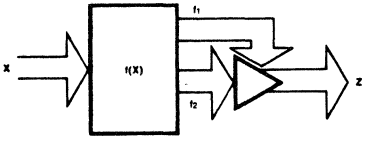
$$Y = f(X)$$



where f is Boolean sum of products transfer function.

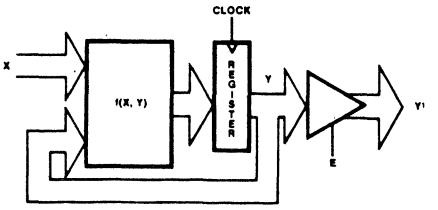
A three state combinatorial function is described as,

$$IF\ (\ f_1(X)\ )\quad Z = f_2(X)$$



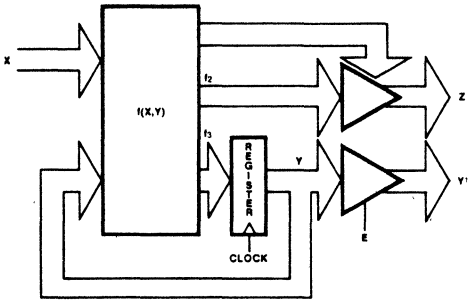
A sequential function is described as,

$$Y := f(X, Y)$$
$$IF(E)\ Y^1 = Y$$



Combining sequential, combinatorial and three-state we now have,

$$IF\ (\ f_1(X, Y)\ )\ Z = f_2(X, Y)$$
$$Y := f_3(X, Y)$$
$$IF\ (E)\quad Y^1 = Y$$



The above generalized Boolean functions are a super set of virtually all of the standard SSI/MSI functions.

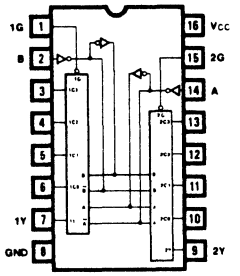
For an LSI implementation to provide a better solution than existing SSI/MSI it must meet the following requirements:

1. Reduce package count
2. Match or improve performance.
3. Reduce System Cost.

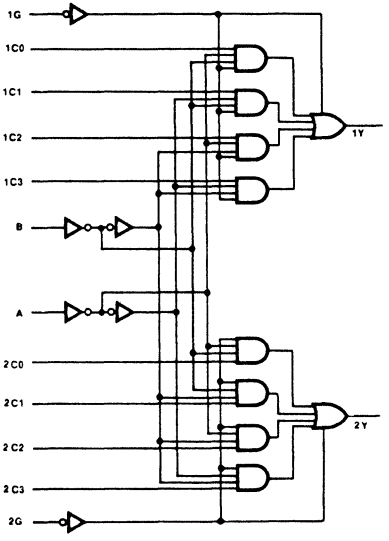
The Programmable Array Logic (PAL) family was designed to meet these requirements.

COMBINATORIAL EXAMPLE:

4 to 1 Mux with three-state Outputs (74LS253)



4 to 1 Mux Logic Symbol



4 to 1 Mux Logic Diagram

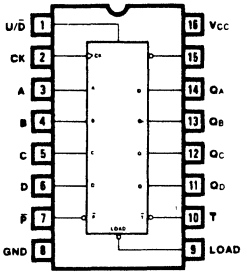
EQUATIONS:

IF(/1G) 1Y = /A* /B* 1C0 +
A*/B*1C1 +
/A* B*1C2 +
A* B*1C3

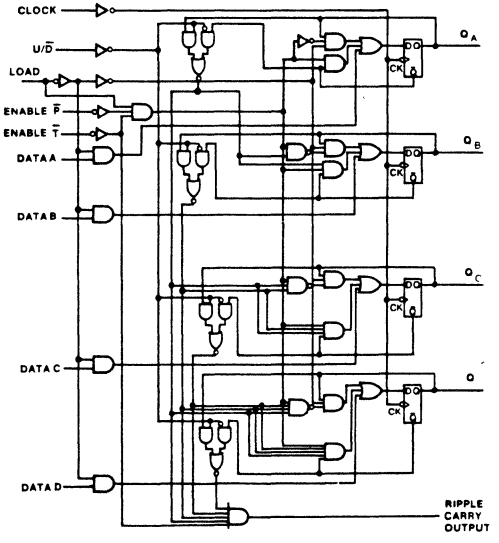
IF(/2G) 2Y = /A*/B*2C0 +
A*/B*2C1 +
/A*B*2C2 +
A*B*2C3

SEQUENTIAL EXAMPLE:

4 BIT UP/DOWN COUNTER (74LS169)



Up/Down Counter Logic Symbol



Up/Down Counter Logic Diagram

EQUATIONS:

QA : = LOAD*A + /LOAD*QA :+:
/LOAD*P*T
QB : = LOAD*B + /LOAD*QB :+:
/LOAD*P*T*UD*QA +
/LOAD*P*T/UD*/QA
QC : = LOAD*C + /LOAD*QC :+:
/LOAD*P*T*UD*QA*QB +
/LOAD*P*T*/UD*/QA*/QB
QD : = LOAD*D + /LOAD*QD :+:
/LOAD*P*T*UD*QA*QB*QC +
/LOAD*P*T*/UD*/QA*/QB*/QC
/CARRY = T*UD*QA*QB*QC*QD + T*/UD*
/QA*/QB*/QC*/QD

8

THE PROGRAMMABLE LSI SOLUTION

A family of 15 PAL devices are manufactured by Monolithic Memories, Inc. which provide a range of logic functions including combinatorial, three-state, sequential, and arithmetic sequential. The TTL devices utilize a platinum Silicide Schottky process and Titanium Tungsten fuse links to form a single programmable emitter follower AND array which drives a fixed OR array as shown in Figure 4.

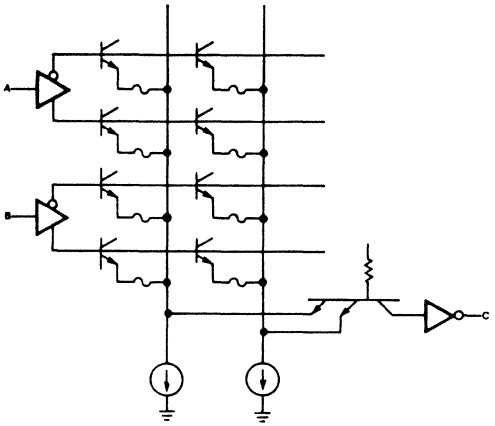


Figure 4: AND-OR Circuit Diagram

The AND-OR array is described logically by a distributed AND gate symbol to facilitate ease of drawing. (Conventional AND symbol with single input rail where X indicates a diode and a fuse)

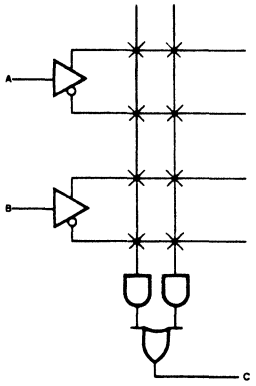


Figure 5: AND-OR Logic Diagram

Basic Combinatorial Cell

The basic combinatorial cell implements the sum of products over a range of input-output pin ratios. All combinatorial output drivers conform to the Low Power Schottky TTL electrical characteristics for totempole drivers. e.g., I_{OL} = 8mA

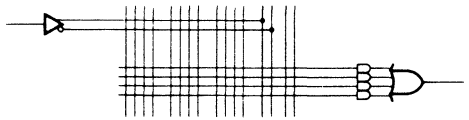


Figure 6: Combinatorial Cell

Programmable I/O

A more complex combinatorial cell incorporates three-state drivers where the enable function is programmable. This allows a pin to be allocated as an input, an output, or dynamically I/O. All three-state outputs are designed to meet the Low Power Schottky TTL three-state bus standard of 24mA I_{OL}

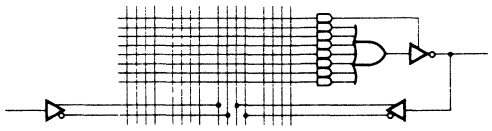


Figure 7: Programmable I/O Cell

Registered Feedback Cell

A D-type Register at the output of a combinatorial cell forms the state machine primative. This cell can be used to implement regular state sequences such as count and shift, but more importantly, it can be used to implement random state sequences not available as standard functions

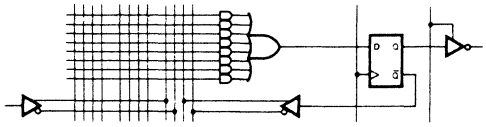


Figure 8: Registered Feedback Cell

Arithmetic Cell

The most complex of the PAL cells includes an exclusive OR and a gated feedback to implement basic arithmetic operations such as greater than, less than, add and subtract

The register forms an accumulator while the combinatorial network forms an ALU.

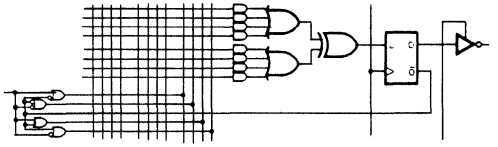


Figure 9 Arithmetic Cell

Structured Pinouts

The 15 PALs are packaged in the popular 20 pin 300 mil wide Skinny DIP™ Input pins are generally to the left while output pins and I/O pins are to the right This feature conveniences PC board layout and allows the Pinout/Logic Symbol to be a Block diagram

PALs: Programmable Logic Functions Help Minimize Hardware

John Birkner/Wescon 79

Taking advantage of PROM on-the-spot programmability, the PALs offer a range of logic replacement capabilities — from random gates to complex arithmetic circuits. Using a programmable AND array to feed a fixed OR array, PALs permit compact realizations of sum-of-products expressions. This paper will provide an overview of the PAL family, a basic understanding of the PAL structure, some guidelines as to how systems can be partitioned to take advantage of the PALs and a design example.

The Problem

A look at the three[1] billion dollar digital integrated circuit marketplace provides insight into the needs of systems designers. First, Figure 1 breaks out the dollar marketplace into four major areas.

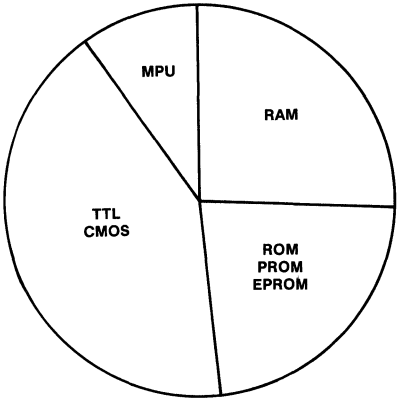


Figure 1. $ Digital IC Marketplace

As the average system should contain a similar percentage cost of silicon for MPU, RAM, ROM/PROM/EPROM, TTL/CMOS, it is of particular interest to note the high percentage of Random Logic; e.g., TTL/CMOS. In spite of the great microprocessor revolution, the systems marketplace spends 35% of its dollars on random logic. To make this point shockingly clear, Figure 2 shows the three billion *unit* marketplace for MPU, RAM, ROM/PROM/EPROM, TTL/CMOS.

The systems marketplace devotes 87% of its IC chip count to TTL/CMOS. Dramatic evidence of this fact is demonstrated by observing the high percentage of 16 and 20 pin DIPs on today's small board computers and microprocessor systems.

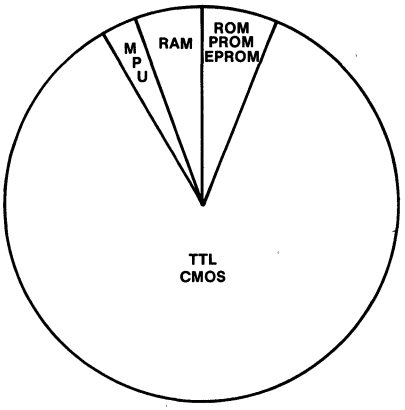


Figure 2. Unit Digital IC Marketplace

Comparison of Figure 1 and Figure 2 along with some quick arithmetic shows the ASP (Average Selling Price) of the MPU, RAM, ROM/PROM/EPROM is around $4.00, whereas, the ASP of TTL/CMOS is around $.50. As a ball-park cost of placing an IC on a board is around $1.00 (board cost, assembly and test), the major cost of using the TTL/CMOS is the overhead, not the silicon.

Programmable Logic

Semiconductor manufacturers have been busy designing a host of programmable logic devices to meet the challenge of integrating this last holdout to LSI, namely random logic gates/muxes/decoders/flip-flops. For the system designer, programmable logic holds the promise of balancing the level of integration of his digital ICs while still further reducing the printed circuit board real estate required per system. Further, programmable logic gives the user a custom IC which he can buy as an inexpensive high volume/multiple sourced virgin device, then customize on commonly available programmers.

The first and most common programmable logic device suitable for logic replacement is the bipolar PROM. Available in a wide variety of input/output pin ratios, the PROM transforms an input variable (address lines) to a desired output condition (data out) with a propagation delay in the range of 50 to 80 nanoseconds. Normally thought of as a memory, the PROM is a sum of products, boolean transfer function which will transform all possible input vectors to any desired output vector. Caution! PROM outputs glitch during the propagation delay of any input change. This is true as any input change causes the source of data to move from one product to another where there may be a gap, overlap, or a third product causing unpredictable outputs. Figure 3 shows a PROM architecture in sum of products form.

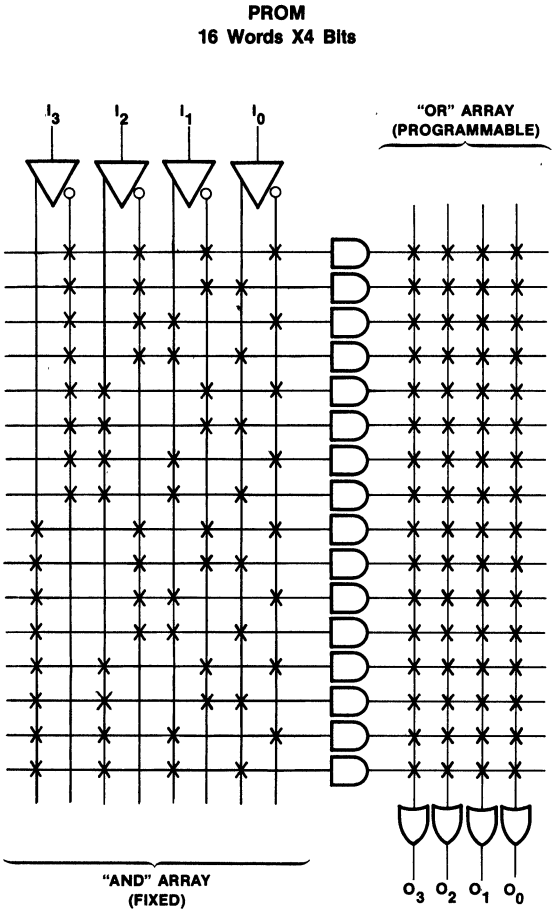


Figure 3. PROM Architecture

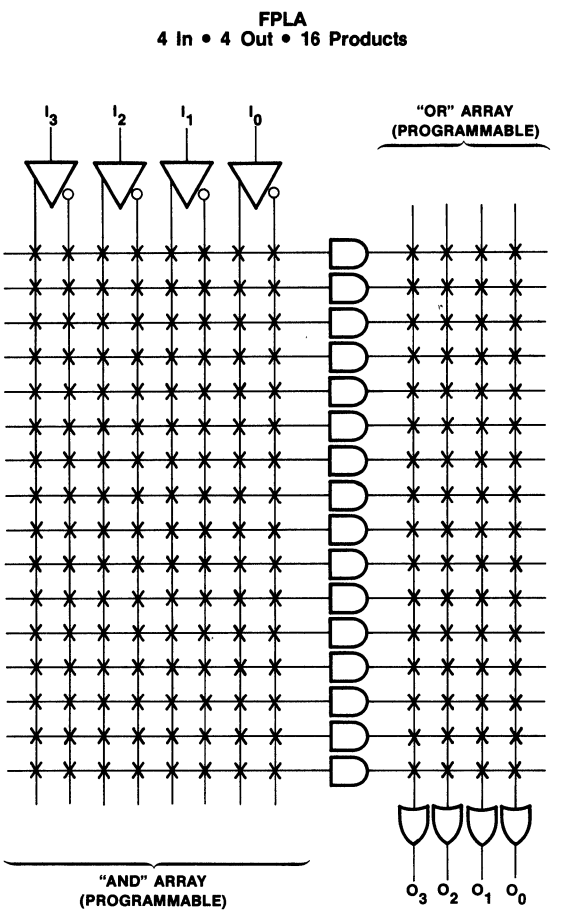


Figure 4. FPLA Architecture

8

The first programmable logic device designed specifically for logic replacement was the FPLA. The programmable AND array overcomes the previous glitch problem in the PROM and allows more input variables. Figure 4 shows an FPLA architecture with programmable AND and programmable OR arrays.

A recent entry into the programmable logic marketplace is the PAL^2 (Programmable Array Logic). The PAL architecture is a complement of the PROM architecture as the AND array is programmable and the OR array is fixed. The architecture affords simple programming in existing PROM programmers plus a fast propagation time of 40 nanoseconds maximum over the commercial V_{CC} and temperature ranges. In addition, PALs have additional output options of Registers and I/O as summarized in Figure 6. Figure 5 shows the PAL architecture with programmable AND array and fixed OR array.

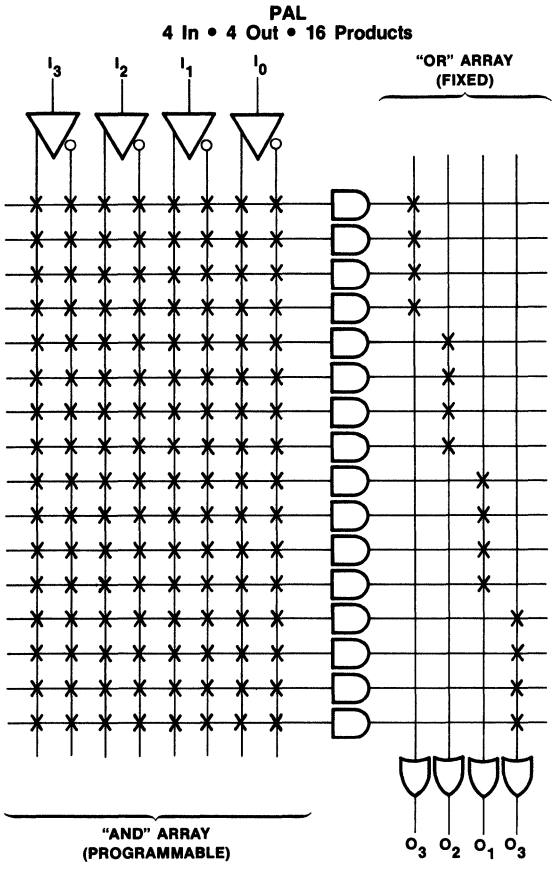


Figure 5. PAL Architecture

	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

Figure 6. Programmable Logic Summary

PALs — A Family of 15

Fifteen PALs provide a family approach to the replacement of random logic gates, muxs, decoders and flip-flops at a greater than 4 to 1 chip count reduction. The die sizes range from 13K square mils to 19K square mils which compare to 2K PROM and 4K PROM die sizes which sell in the $2.00 to $4.00 range. Two major semiconductor manufacturers are now supplying PALs with a third to appear shortly.

A description of the family is shown in Table 1 below.

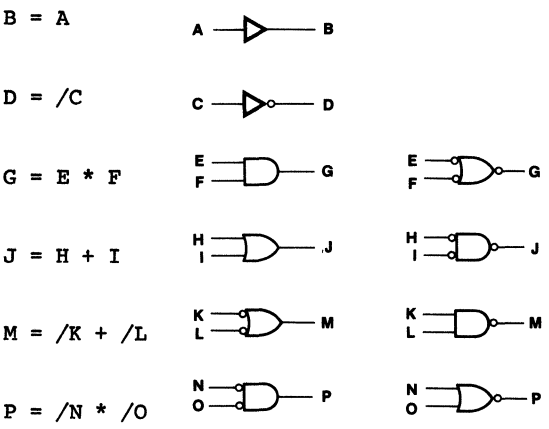
PART NUMBER	DESCRIPTION
PAL10H8	Octal 10 Input And-Or Gate Array
PAL12H6	Hex 12 Input And-Or Gate Array
PAL14H4	Quad 14 Input And-Or Gate Array
PAL16H2	Dual 16 Input And-Or Gate Array
PAL16C1	16 Input And-Or/And-Or-Invert Gate Array
PAL10L8	Octal 10 Input And-Or-Invert Gate Array
PAL12L6	Hex 12 Input 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

Table 1. PAL Family

Ranging in complexity from simple gates to complex arithmetic functions the PAL Family can functionally replace up to 90% of the 7400 Series TTL.

A Simple Example

To demonstrate a PAL implementation, consider the following Boolean functions:



Choosing the PAL10H8, these functions are now implemented as shown in the logic symbol of Figure 7.

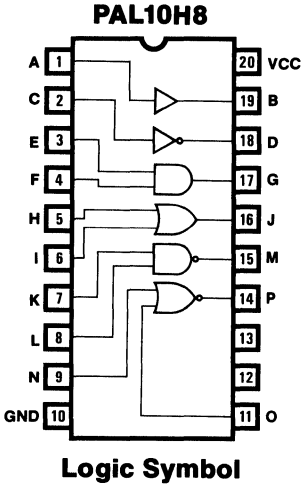


Figure 7. Example Gates Logic Symbol

The functions are described by the logic diagram of Figure 8. The buffer which connects Pin 1 (A) to Pin 19 (B) is formed by the single "X" placed at the intersection of product term 0 (top horizontal line) and input line 2 (far left vertical line). The "X" is a shorthand expression for an intact fuse connecting the product line (AND gate) to the input line thru an NPN transistor. The absence of an "X" indicates a blown fuse. The AND gate is formed by two "Xs" on the same line which signifies ANDing as all "Xed" inputs on the single rail AND symbol are ANDed together. The OR function is formed by two "Xs" on different product lines.

This example has shown the implementation of common gate structures. The versatility of the PAL is demonstrated by some not so common gate structures shown below. Figure 9 shows a PAL implementation of those gates.

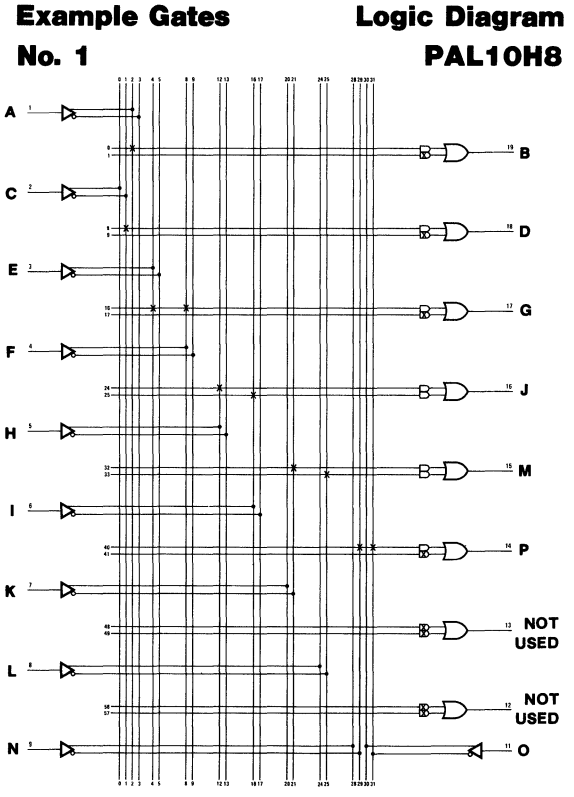
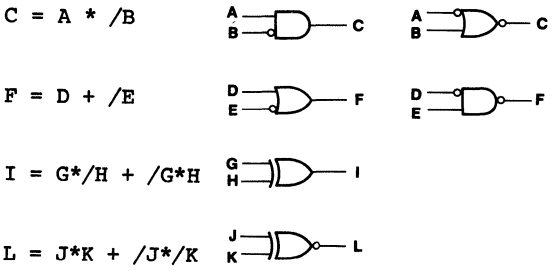


Figure 8. Example Gates Logic Diagram



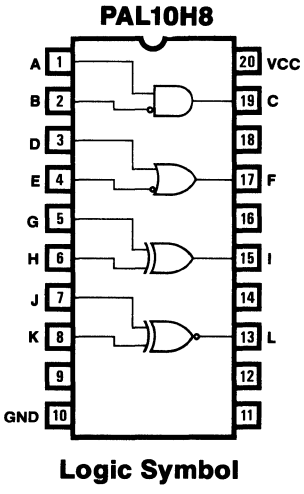


Figure 9. Example Gates Logic Symbol

8

Partitioning System Logic for PALs

The sum of products Boolean transfer function is the common denominator and basic primitive of all digital logic systems. From the sum of products, not only can all the gate functions be derived, but so also can latches, edge triggered registers, MUXs, encoders and decoders. The first nine PALs (PAL10H8 thru PAL16L2) implement a variety of input/output pin ratios and product terms per output. Input pin loading is less than 0.25 mA while output drive is 8 mA sink and 3.2 mA source. A typical gate configuration is shown in Figure 10.

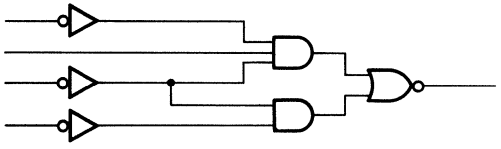


Figure 10. Typical Sum of Products

Three-state buses are commonly used in TTL systems to reduce interconnect densities by sharing signal lines. The PAL16L8 thru PAL16A4 output drivers are all three-state outputs which sink 24 mA, compatible with the low power Schottky three state standard. This 24 mA sink is sufficient to drive many standard buses including the Intel Multibus.

The PAL three-state outputs are of two types. First, the register outputs are controlled by a common enable pin for parallel bus enabling. Second, the combinatorial outputs are individually controlled by a product term. This feature allows an open collector configuration to be logically derived as required in bus-shared control signals such as memory transfer acknowledge. The latter type is shown in Figure 11.

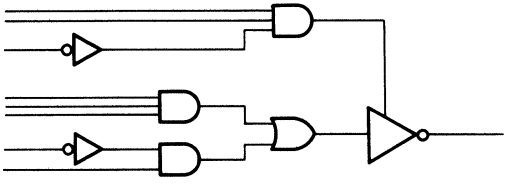


Figure 11. Three-State Driver with Individual Enable

Latch circuits can be implemented as shown in Figure 12. The PAL16L8 is commonly used for this function as six of the eight outputs are internally connected to an array input line. Edge triggered flip flops can be constructed from two latches where individual clocking is required.

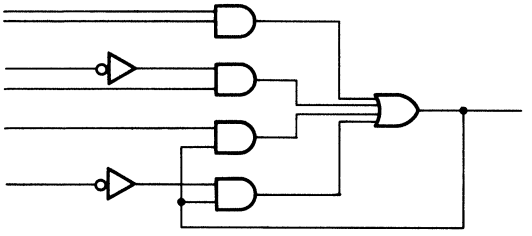


Figure 12. Latch Circuit

Control logic containing D-type or J-K flip-flops is common among memory, processor and controller interfaces. This logic can generally be classified as state-sequence logic or small-state machines. The PAL types PAL16R8, PAL16R6 and PAL16R4 contain output registers with feedback which can implement simple state-sequences. Figure 13 shows a typical control logic configuration.

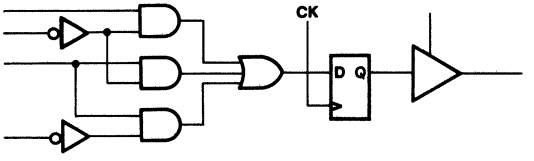


Figure 13. Registered Sum of Products

Design Example

A typical control logic problem is the memory to processor handshake on memory transfer used in many computer architectures. The processor makes a transfer request by activating a request line (REQ) and specifies a read or write operation on a Read/Write line (R/W).

During a read operation the processor waits for a Data Available signal at which time the data bus is sampled and the request line lowered, completing the cycle. During a write operation, the processor places data on the bus and waits for a write complete signal after the write cycle is finished. Upon write complete, the request line is lowered, completing the cycle. This handshaking operation is described in the timing diagram of Figure 14.

The memory-board logic to implement this function may be designed with gates and edge triggered flip flops as shown in Figure 15. This particular design would require about five SSI/MSI packages. The same design is implemented in Figure 16 by a single PAL16R8.

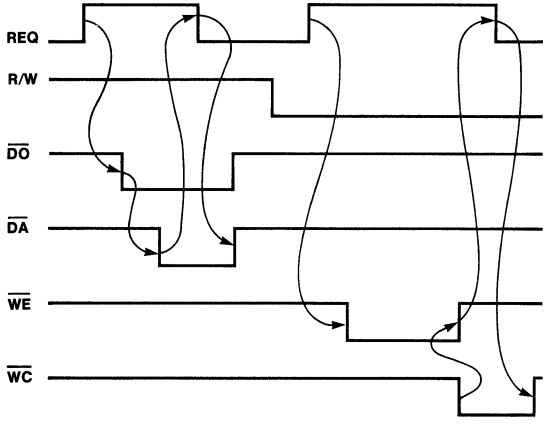


Figure 14. Memory Handshake Timing

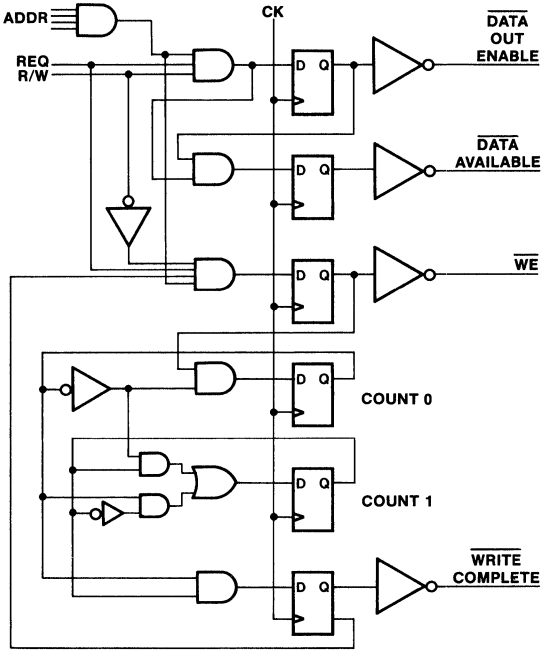


Figure 15. Memory Handshake Logic

Logic Diagram PAL16R8

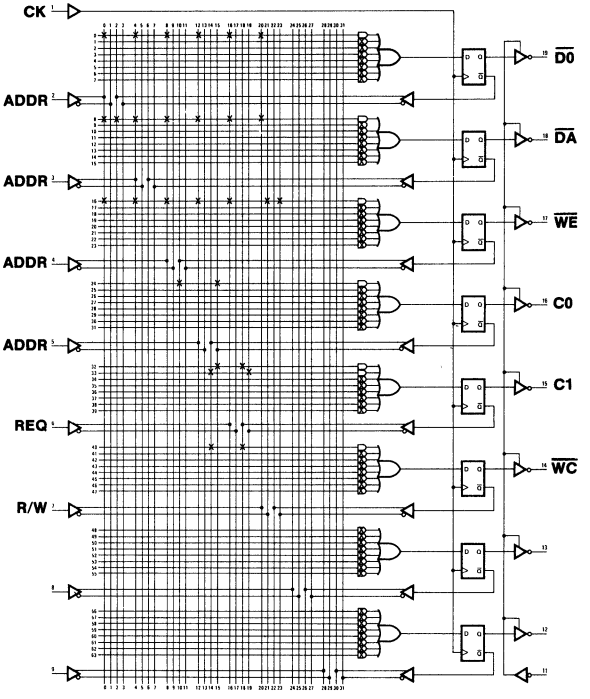


Figure 16. PAL Memory Handshake Logic

Bibliography

1. Semiconductor Industry Association, 1979, first four months report.

2. John Birkner, High Speed/Low Cost Fuse Link Arrays Compete with TTL 74S/LS, WESCON '78.

Gate Arrays Logjam Test Engineering

John Birkner/Wescon 80

Gate arrays give logic designers the ability to create numerous semi-custom ICs in a relatively short amount of time. The number of unique patterns may vary from 10 to 200 for a typical new systems design. When the new system is transferred to production, the Test Engineering department is faced with one to two work weeks per pattern. Systems manufacturers, using gate arrays for the first time, are cautioned to plan for the increased demands on Test Engineering, or they will find a logjam in testing. This paper examines the problems and solutions in testing gate arrays and semicustom logic.

Definitions

Many terms have recently appeared to describe the host of integrated circuits which are configured late in the manufacturing process. These ICs are most generally described as uncommitted logic. More specifically, there are classifications such as semicustom logic, gate arrays, programmable logic arrays, field programmable logic arrays, programmable array logic and hard array logic.

All of these devices provide the systems designer with common advantages; higher board density, lower power, higher speed, etc. For the purposes of this discussion, they will all be classified as gate arrays. The specific example chosen for this discussion will be hard array logic, HAL, and its programmable counterpart, PAL.

Replacing PCBs With Silicon

The problem of testing digital logic is steadily moving from the printed circuit board (PCB) to the LSI chip. In the case of gate arrays, the etched lines which connected the random assortment of SSI/MSI gates and flip flops is being lifted from the PCB and placed on silicon. Test engineers, who complain that the new gate arrays create a testing problem, forget that they once tested that same logic on the PCB.

Gate arrays don't create new testing problems; they just move the old testing problem from the PCB onto the silicon chip. This may be viewed as an opportunity to structure the testing problem in the more regular environment of a fixed package, structured array. We must be more clever, however, in generating our test programs without the aid of internal test points which were available on the PCB.

The Logjam

Testing LSI logic is certainly not a new problem. Standard LSI chips, like the microprocessor and the variety of special purpose controllers, are regularly tested. The new problem lies in large numbers of gate array patterns which can be quickly designed and fabricated.

For a single systems design, a few design engineers can create as many as 200 new and unique gate array patterns. At one week per pattern, this translates to four work years of test engineering time.

Test Vectors

Test programs for LSI devices consist of electrical, switching and functional tests. The electrical and switching characteristics are tested by the semiconductor manufacturer, and may be optionally tested by the user. The function test is supplied to the manufacturer by the user. It consists of a list of test vectors, which specify for each pin, instructions for driving high or low, or testing high or low, or not driving and not testing.

The time consuming job for the user is the generation of test vectors. At this time, work is under way toward automating the generation of test vectors, thus far with some success. Even with automated programs, manual intervention and human interaction make the task time consuming. In addition, the automated programs are not self starting. They need manually generated seed vectors.

Responsibility of the Designer

When a new system is transferred to production, the systems designer hands over the responsibility for the system to the test engineering department who now determines how and what tests should be performed to ensure proper operation of the system. At this pont the systems designer transmits the necessary information for understanding the system operation. Unfortunately, much information is lost at this point. All too often continuity is broken and while the systems designer is off designing the next great project, the test engineer is left to struggle with understanding how the system works. This struggle is exaggerated when system complexity is increased by the use of gate arrays.

It is the design engineer who best knows the operation of his gate array design, and it is the design engineer who can quickly specify a few seed vectors to give the test engineer a starting point or to give the automatic vector generator a starting point. The problem at hand is to entice designers into this task.

Tools for the Designer

The first step toward luring the designer into providing test vector information is to give him a standard format for specifying the vectors. The format should be easy to understand and convenient to write down either on paper or, preferably, into a computer terminal. Fortunately, there is a format that already exists, which with slight modification, will suit our purpose. The format is the well known "Function Table" format found in the Texas Instruments TTL *Databook* and accepted by JEDEC. Let us now redefine that format as follows:

Function Table

The function table begins with the keyword, "FUNCTION TABLE." It is followed by a pin list which symbolically

defines each pin. The pin list is followed by a dashed line, e.g.-------(length optional), which in turn is followed by a list of vectors, one vector per line. A vector is a sequence of states in order of the pin list followed by an optional comment. The vector list is followed by another dashed line.

Definition of States

```
H   HIGH LEVEL

L   LOW LEVEL

C   TRANSITION FROM LOW TO HIGH

X   IRRELEVANT

Z   OFF (HIGH IMPEDANCE)
```

With this definition, let us now write a function Table for a simple two input AND Gate having inputs A and B with output C.

```
FUNCTION TABLE

A  B  C
--------------
L  L  L
H  L  L
L  H  L
H  H  H
--------------
```

The second step in persuading the designer into providing test vectors is to combine the Function Table with the gate array design specification so as to provide one document with design and functional information.

The real objective here is to give the designer a format to mimic the data sheet format which he is accustomed to seeing for standard TTL components.

This is the approach taken to specify designs for Hard Array Logic, HAL, the mask version of Programmable Array Logic, PAL. The design specification is a computer file which contains part number, name of device, symbolic pin list, design equations and a Function Table.

Outsmarting the Designer

Providing just the format for specifying test vector information is not likely to seduce the busy systems designer into such a boring task. We have to be a bit more clever to snare him. Fortunately, we can charm him with a decoy, gate array simulation.

Gate Array Simulation

The systems designer is impressed with computer-aided tools that help him in his design task. What if he had a computer program which would simulate his gate array design and tell him if it works as he wants it to? Would he use it? This author is betting that he will.

A simulator has been added to the PAL Assembler, PALASM, which reads the Function Table and simulates the device as specified in the state equations which specify the transfer function. Inconsistencies between the Function Table and the transfer function are reported as errors. The simulator checks the operation of the gate array against the equations, thus verifying the designer's intention for the device. What the designer perhaps will never realize is that he has also provided seed vectors for test engineering.

Future Considerations

The test vector generation problem is generally considered to be one of starting with the transfer function and synthesizing the state table. The engineering problem is just the reverse. That is, the designer starts off with the state table and he synthesizes the transfer function. A challenge for gate array manufacturers is to provide their customers with computer-aided tools to automatically generate gate arrays from state tables.

High Level Language for Programmable Array Logic

John Birkner/Wescon 81

Introduction

Standards for silicon definition languages are under consideration by semiconductor manufacturers in order to provide a means of communication between vendor and customer for specifying semicustom ICs. The objective of an industry standard language is to provide a hardware-independent, manufacturer-independent, user-independent, unambiguous definition of an IC. So far, the debates in subcommittees, ranging from what symbol to use for an AND gate to whether there should be a standard at all, would indicate that many standards will evolve.

Digital machines are most efficiently defined by explicit Boolean state equations. These equations are unambiguous, they are easily simulated, and they can be readily expanded to systems of machines. This paper discusses such an equation-implemented language as applied to Programmable Array Logic.

Logic Schematics Fall Short

The predominant hardware description language of today is the logic schematic. The predominant hardware verification technique of today is the hardware prototype. These tools are inadequate for the development of silicon hardware.

The logic schematic may be a fine tool for human understanding of a digital system, but is is quite awkward to input it into a computer for generation of photolithographic plates used in making semicustom or custom ICs. Likewise, the hardware prototype is an awkward means of design verification of an IC. Differences in propagation delay, circuit implementation, and logic implementation degrade the effectiveness of the hardware emulation. Also, the manual interaction increases the probability of error, especially in high complexity logic systems. The cost of an error in making an IC is tens and thousands of dollars and months of delay. The cost of multiple errors is years of delay, which often kills the project.

TTL 7400-series logic schematics are commonly submitted to semiconductor vendors for implementation in semicustom or custom ICs. The semiconductor manufacturer then makes a choice as to whether to implement the function exactly as shown on the schematic, or to optimize the design to the particular circuit technique.

If the vendor implements the TTL function exactly as shown, the user will pay the price of extra die size. If the vendor optimizes the design, he must understand it, at a high cost of engineering development time. In either case, the vendor is subject to the risk that the IC version will not operate identically to the TTL version. Figure 1 demonstrates the risk of copying TTL logic directly. The 74LS161 four-bit counter is asynchronously set to zero when the count reaches 12. Will this circuit hazard operate the same when it is forged into one piece of silicon? Or, will it oscillate?

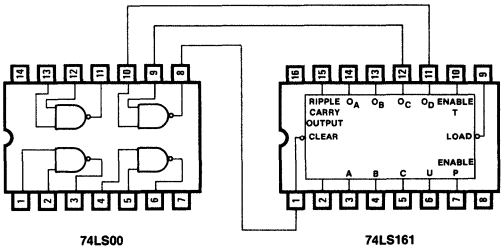


Figure 1. Common Circuit Hazard

Testability is another common trap for the semiconductor vendor. The user's TTL prototype may work fine in his system and also be virtually untestable. A frequent example is the clock phase generator which has no initialize control pin. Without a known initial state, the tester must sequence the device until a recognizable pattern is observed. Most IC testers do not have this capability.

Discipline for Semicustom Users

Successful use of semicustom logic is best achieved when the user understands the vendors manufacturing cycle and when the user participates in that cycle. The production of a semicustom IC is basically the same as that required for a standard IC, with the exception that the user takes part in some of the steps. The major steps are outlined below.

1. Define
2. Simulate
3. Build
4. Test

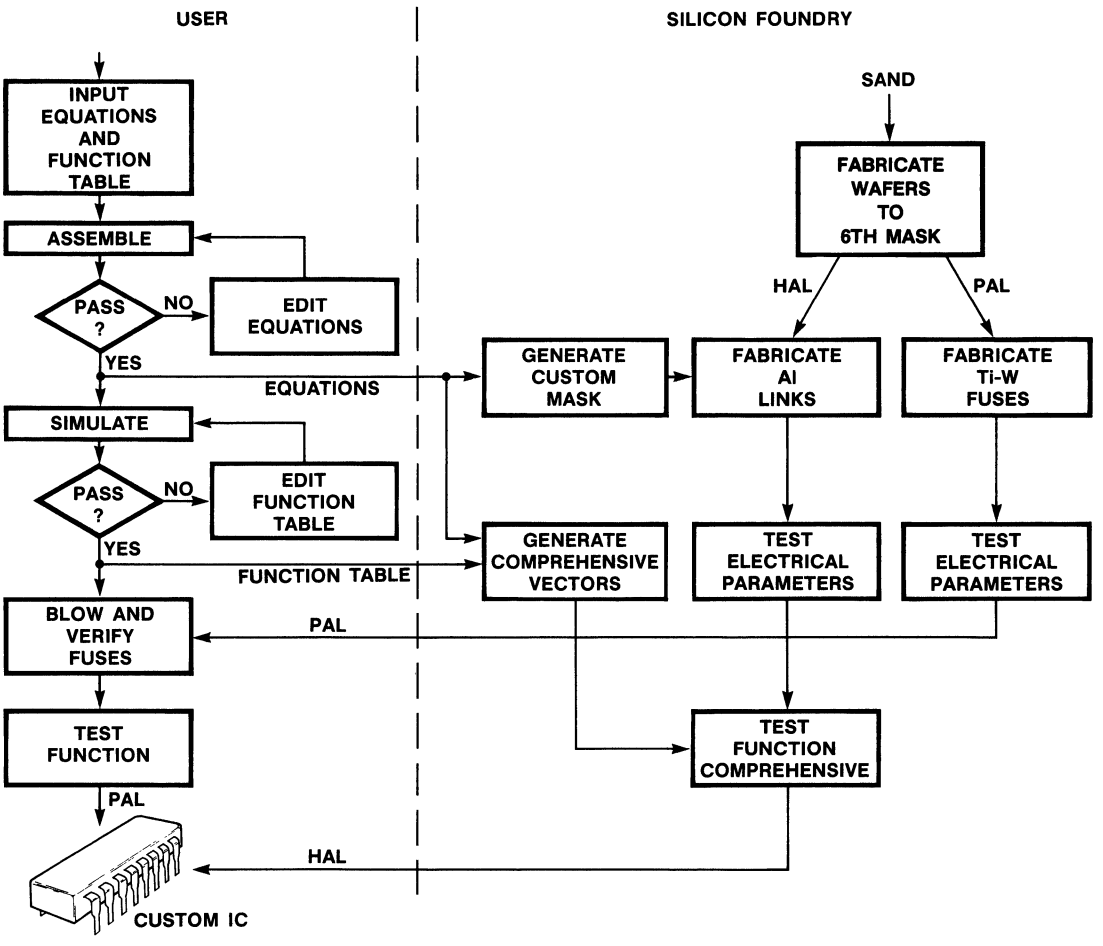


Figure 2. PAL/HAL Development Cycle

These four steps are demonstrated in the diagram above for the development of PAL, Programmable Array Logic, and HAL, Hard Array Logic. The user begins by defining his logic functions using Boolean logic equations. These equations are automatically assembled into hardware routing instructions by the PAL assembler, PALASM. Syntax errors are reported. The user edits and reassembles, repeating the process until no syntax errors are reported.

The user then inputs a Function Table to verify his design. PALASM simulates the PAL/HAL, using the equations, and executes the Function Table on the simulated device. Simulation errors are reported. The user corrects the Function Table and/or equations until no simulation errors occur. This completes the design verification process.

The automatic routing instructions from PALASM may now be used to generate a custom mask for fabrication of a HAL. Aluminum links implement the logic equations, providing a semicustom device. Comprehensive Test vectors are generated using the Function Table as seed vectors. One hundred percent coverage for stuck-high/low test conditions is usually achieved with fewer than 200 vectors.

HALs are cost-effective in volumes of 1000 or above. For lower volumes, the programmable version, PAL, may be programmed by the user on low-cost programmers. The HAL is unique in that it is a semicustom device that has a programmable prototype, the PAL.

Equations Simplify

"Our life is frittered away by detail . . . Simplify, simplify."

Thoreau

Hardware Description Languages most often use network topology to specify the interconnection of gates and flipflops. Propagation delays are individually specified; wire lists are generated; long lists of generally unreadable network interconnection are printed out.

Explicit Boolean equations are a much simpler method of writing logic functions. They are easily interpreted and can be readily commented. An example of an AND gate feeding an OR gate with a comment is shown below.

F = A*B + C ; This is a comment

The AND gate is symbolized by "*" and the OR gate is symbolized by "+".

Function Table Simulates

Equations are easily simulated and can be exercised by HIGHs and LOWs specified in a Function Table. A simulation of the above equations is implemented in the Function Table below.

```
FUNCTION TABLE

A B C  F
-------------------------
L L L  L TEST OUTPUT LOW
H H L  H TEST OUTPUT HIGH
L L H  H TEST OUTPUT HIGH
-------------------------
```

The simulation is now used to generate test vectors and may also be used as seed vectors for automatic generation of comprehensive test vectors until a 100% coverage of stuck-high/low conditions is achieved.

Design Verification

The design verification performed by a user in the PAL/HAL development cycle of Figure 2 is now demonstrated in a real example. What follows is a computer printout from the terminal of a PAL Development System during a development session. Two errors are intentionally made during the session. First, a syntax error, then a simulation error is made to demonstrate the corrective feedback in the development cycle.

EXAMPLE SESSION #1

```
+INPUT
PAL10L8                    PAL DESIGN SPECIFICATION
PN12345                    J. BIRKNER        6/9/81
EXAMPLE
MMI SUNNYVALE, CA
A B C X X X X X X GND X X X X X X X X /F VCC
F = A*B + E ;THIS IS A COMMENT
FUNCTION TABLE
A B C  F
-------------------------
L L L  L  TEST OUTPUT LOW
H H L  L  TEST OUTPUT HIGH
L L H  H  TEST OUTPUT HIGH
-------------------------

+ASSEMBLE
ASSEMBLY ERROR = E

FAIL
+EDIT
   PAL10L8                 PAL DESIGN SPECIFICATION
#  PN12345                 J. BIRKNER        6/9/81
#  EXAMPLE
#  MMI SUNNYVALE, CA
#  A B C X X X X X X GND X X X X X X X X /F VCC
#  F = A*B + E ;THIS IS A COMMENT
#CEF = A*B + C
   F = A*B + C ;THIS IS A COMMENT
#
#Q

+SIMULATE
01 000XXXXXXXXXXXXXXH1
VECTOR ERROR 02 F

FAIL
+EDIT
   PAL10L8                 PAL DESIGN SPECIFICATION
#  PN12345                 J. BIRKNER        6/9/81
#  EXAMPLE
#  MMI SUNNYVALE, CA
#  A B C X X X X X X GND X X X X X X X X /F VCC
#  F = A*B + C ;THIS IS A COMMENT
#  FUNCTION TABLE
#  A B C  F
#  -------------------------
#  L L L  L  TEST OUTPUT LOW
#  H H L  L  TEST OUTPUT HIGH
#CLH H L  H
   H H L  H  TEST OUTPUT HIGH
#
#Q

+SIMULATE
01 000XXXXXXXXXXXXXXH1
02 110XXXXXXXXXXXXXXL1
03 001XXXXXXXXXXXXXXL1


PASS
+
```

A more complex example is an 8-bit synchronous counter as shown in Figure 3. Example Session #2 reads, assembles and simulates the PAL Design Specification.

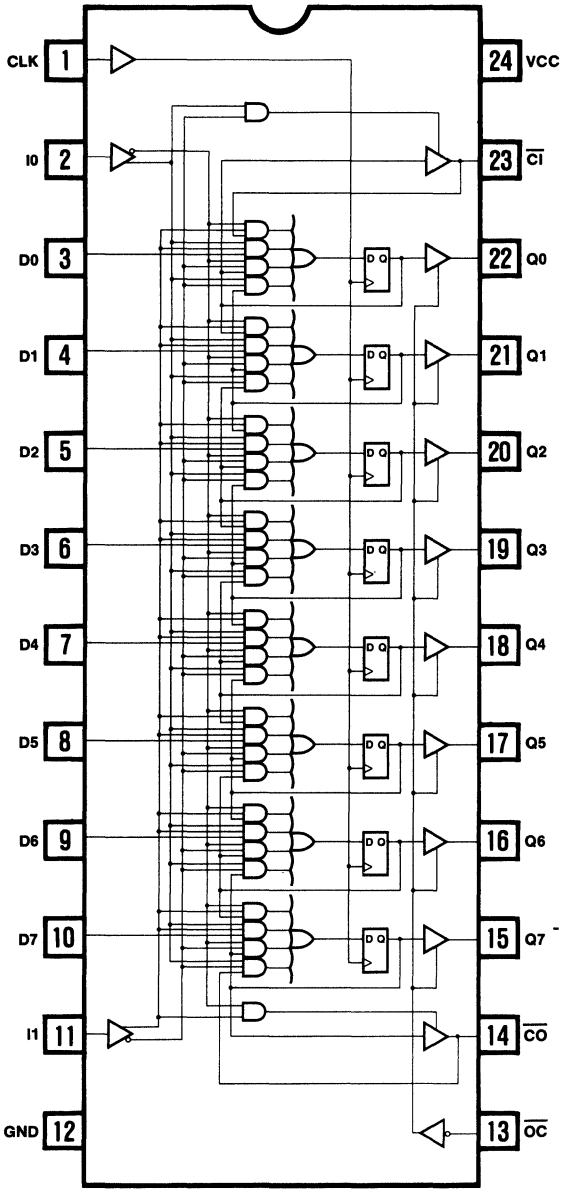


Figure 3. 8-Bit Synchronous Counter

EXAMPLE SESSION #2

```
+READ
PAL20X8                PAL DESIGN SPECIFICATION
CTR8A          BIRKNER/KAZMI/BLASCO     2/10/81
8-BIT SYNCHRONOUS COUNTER
MMI  SUNNYVALE, CALIFORNIA
CLK I0 D0 D1 D2 D3 D4 D5 D6 D7 I1 GND
/EN /CO Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0 /CI VCC

/Q0 :=  /I1*/I0                ;CLEAR LSB
     +       I0 * /Q0          ;COUNT/HOLD
    :+:  I1*/I0 * /D0          ;LOAD D0 (LSB)
     +   I1* I0 * CI           ;COUNT

/Q1 :=  /I1*/I0
     +       I0 * /Q1
    :+:  I1*/I0 * /D1
     +   I1* I0 * CI*Q0

/Q2 :=  /I1*/I0
     +       I0 * /Q2
    :+:  I1*/I0 * /D2
     +   I1* I0 * CI*Q0*Q1

/Q3 :=  /I1*/I0
     +       I0 * /Q3
    :+:  I1*/I0 * /D3
     +   I1* I0 * CI*Q0*Q1*Q2

/Q4 :=  /I1*/I0
     +       I0 * /Q4
    :+:  I1*/I0 * /D4
     +   I1* I0 * CI*Q0*Q1*Q2*Q3

/Q5 :=  /I1*/I0
     +       I0 * /Q5
    :+:  I1*/I0 * /D5
     +   I1* I0 * CI*Q0*Q1*Q2*Q3*Q4

/Q6 :=  /I1*/I0
     +       I0 * /Q6
    :+:  I1*/I0 * /D6
     +   I1* I0 * CI*Q0*Q1*Q2*Q3*Q4*Q5

/Q7 :=  /I1*/I0
     +       I0 * /Q7
    :+:  I1*/I0 * /D7
     +   I1* I0 * CI*Q0*Q1*Q2*Q3*Q4*Q5*Q6

IF (VCC)  CO = CI*Q0*Q1*Q2*Q3*Q4*Q5*Q6*Q7
```

8

```
FUNCTION TABLE

I1 I0 D7 D6 D5 D4 D3 D2 D1 D0 /CI CLK /EN
Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0 /CO

; SEL     INPUTS     CONTROL     OUTPUTS
;I1 I0   7------0  /CI CLK /EN  7------0 /CO
-------------------------------------------
  L L   HHHHHHHH   L   C   L   LLLLLLLL  H
  H L   LLLLLLLH   X   C   L   LLLLLLLH  H
  H H   XXXXXXXX   L   C   L   LLLLLLHL  H
  H L   LLLLLLHH   X   C   L   LLLLLLHH  H
  H H   XXXXXXXX   L   C   L   LLLLLHLL  H
  H L   LLLLLHHH   X   C   L   LLLLLHHH  H
  H H   XXXXXXXX   L   C   L   LLLLHLLL  H
  H L   LLLLHHHH   X   C   L   LLLLHHHH  H
  H H   XXXXXXXX   L   C   L   LLLHLLLL  H
  H L   LLLHHHHH   X   C   L   LLLHHHHH  H
  H H   XXXXXXXX   L   C   L   LLHLLLLL  H
  H L   LLHHHHHH   X   C   L   LLHHHHHH  H
  H H   XXXXXXXX   L   C   L   LHLLLLLL  H
  H L   LHHHHHHH   X   C   L   LHHHHHHH  H
  H H   XXXXXXXX   L   C   L   HLLLLLLL  H
  H L   HHHHHHHH   L   C   L   HHHHHHHH  L
  H H   XXXXXXXX   L   C   L   LLLLLLLL  H
  H L   HHHHHHHH   L   C   L   HHHHHHHH  L
  H L   HHHHHHHL   X   C   L   HHHHHHHL  H
  H L   HHHHHHLH   X   C   L   HHHHHHLH  H
  H L   HHHHHLHH   X   C   L   HHHHHLHH  H
  H L   HHHHLHHH   X   C   L   HHHHLHHH  H
  H L   HHHLHHHH   X   C   L   HHHLHHHH  H
  H L   HHLHHHHH   X   C   L   HHLHHHHH  H
  H L   HLHHHHHH   X   C   L   HLHHHHHH  H
  H L   LHHHHHHH   X   C   L   LHHHHHHH  H
  H L   HHHHHHHH   L   C   L   HHHHHHHH  L
  L L   XXXXXXXX   X   C   L   LLLLLLLL  H
  H H   XXXXXXXX   L   C   L   LLLLLLLH  H
  H H   XXXXXXXX   L   C   L   LLLLLLHL  H
  H H   XXXXXXXX   L   C   L   LLLLLLHH  H
  H H   XXXXXXXX   L   C   L   LLLLLHLL  H
  H L   HHHHHHHL   X   C   L   HHHHHHHL  H
  H H   XXXXXXXX   L   C   L   HHHHHHHH  L
  H H   XXXXXXXX   H   C   L   HHHHHHHH  H
  L H   LLLLLLLL   L   C   L   HHHHHHHH  L
  H H   HHHHHHHH   L   C   L   LLLLLLLL  H
-------------------------------------------
```

```
+ASSEMBLE

PASS
+SIMULATE
01 C01111111110X0HLLLLLLLLL01
02 C0100000001X0HLLLLLLLLHX1
03 C1XXXXXXXX1X0HLLLLLLLHL01
04 C0110000001X0HLLLLLLHHX1
05 C1XXXXXXXX1X0HLLLLLHLL01
06 C0111000001X0HLLLLLHHHX1
07 C1XXXXXXXX1X0HLLLLHLLL01
08 C0111100001X0HLLLLHHHHX1
09 C1XXXXXXXX1X0HLLLHLLLL01
10 C0111110001X0HLLLHHHHHX1
11 C1XXXXXXXX1X0HLLHLLLLL01
12 C0111111001X0HLLHHHHHHX1
13 C1XXXXXXXX1X0HLHLLLLLL01
14 C0111111101X0HLHHHHHHHX1
15 C1XXXXXXXX1X0HHLLLLLLL01
16 C0111111111X0LHHHHHHHH01
17 C1XXXXXXXX1X0HLLLLLLLL01
18 C0111111111X0LHHHHHHHH01
19 C0011111111X0HHHHHHHHLX1
20 C0101111111X0HHHHHHHLHX1
21 C0110111111X0HHHHHHLHHX1
22 C0111011111X0HHHHHLHHHX1
23 C0111101111X0HHHHLHHHHX1
24 C0111110111X0HHHLHHHHHX1
25 C0111111011X0HHLHHHHHHX1
26 C0111111101X0HLHHHHHHHX1
27 C0111111111X0LHHHHHHHH01
28 C0XXXXXXXX0X0HLLLLLLLLX1
29 C1XXXXXXXX1X0HLLLLLLLH01
30 C1XXXXXXXX1X0HLLLLLLHL01
31 C1XXXXXXXX1X0HLLLLLLHH01
32 C1XXXXXXXX1X0HLLLLLHLL01
33 C0011111111X0HHHHHHHHLX1
34 C1XXXXXXXX1X0LHHHHHHHH01
35 C1XXXXXXXX1X0HHHHHHHHH11
36 C1000000000X0LHHHHHHHH01
37 C1111111111X0HLLLLLLLL01

PASS
+
```

Hard Array Logic Provides New TTL Standards

John Birkner and Vincent Coli/Southcon 82

Summary

HALs manufactured to perform standard functions add a new dimension to TTL. These new devices have features and benefits not found in the standard TTL family. They also provide an introduction to semicustom logic design with zero engineering effort. The user is always free to further customize these designs with a PAL if he chooses.

Product Definition

PAL—Programmable Array Logic

The PAL is a fuse-programmable small gate array. PALs are available in 20 and 24 pin packages with many different input/output pin ratios and active high/active low output polarities. High end members feature Registered Outputs with Feedback, Exclusive OR Gates, and Arithmetic Gated Feedback.

HAL—Hard Array Logic

The HAL family is a mask-programmed version of a PAL. The HAL is to a PAL just as a ROM is to a PROM. A standard wafer is fabricated to the 6th mask. Then a custom metal mask is used to fabricate Aluminum links for a HAL instead of the programmable Ti-W fuse array used in a PAL.

HMSI—HAL Medium Scale Integration

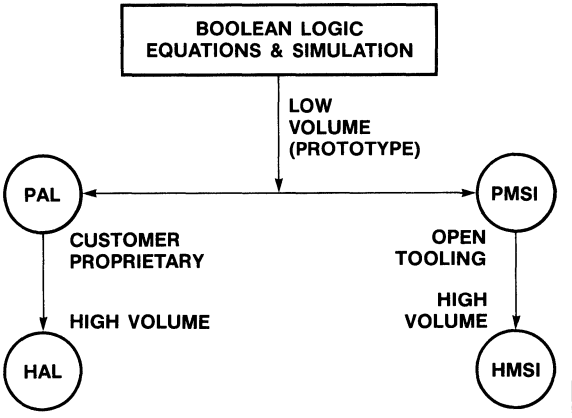
The HMSI family is derived from the PAL using HAL technology. These devices perform predetermined functions which are not available in the TTL family. Because they are produced in volume, the user receives the benefit of volume pricing. An industry standard 74LS part number is assigned to the device.

PMSI—PAL Medium Scale Integration

The PMSI family is derived in a similar fashion except this product is produced entirely from a PAL. A HAL mask is not generated and an industry standard 74LS part number is not assigned unless sales warrant it.

Design Methodology

The HMSI/PMSI design methodology is a part of a larger methodology described in the flow chart below A product is defined in the form of Boolean logic equations. It is then simulated using a Function Table. A user will prototype the device with a PAL and switch to HAL for high volume. When the design originates from the semiconductor manufacturer, it is prototyped in PAL and trial marketed as "open tooled" PMSI. If the market warrants high volume, then a HAL mask is generated and the PMSI device is promoted to HMSI.



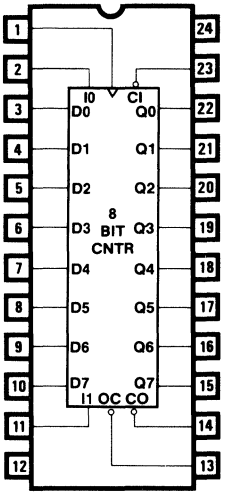
Data Sheets

HMSI products are documented both as a standard TTL chip and as a PAL. The standard TTL documentation consists of:

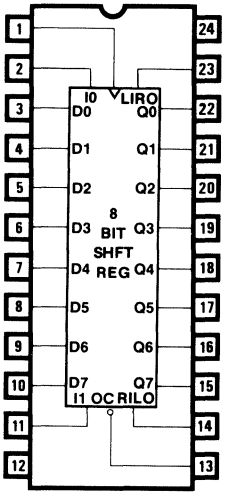
1. 74LSXXX part number.
2. Description of features.
3 Function Table which presents the functions of the device in a structured-tabular format.
4 Logic Symbol with pin identification and polarity.
5. Logic Diagram which is the MIL standard logic gate drawing.
6. Electrical and Switching characteristics.
7. Connection Diagram for a typical application of the device

Logic Symbols for the first seven HMSI functions are shown below:

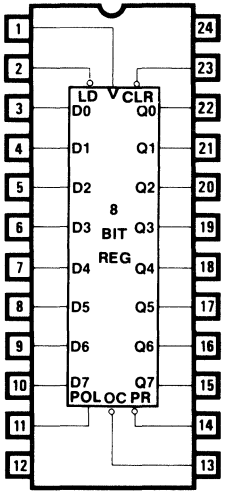
SN54/74LS461
Octal Counter



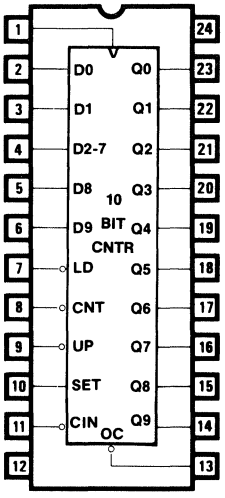
SN54/74LS498
Octal Shift Register



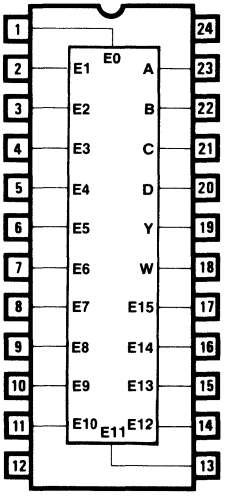
SN54/74LS380
Multifunction Register



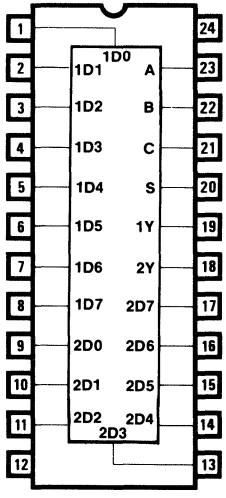
SN54/74LS491
10-Bit Counter



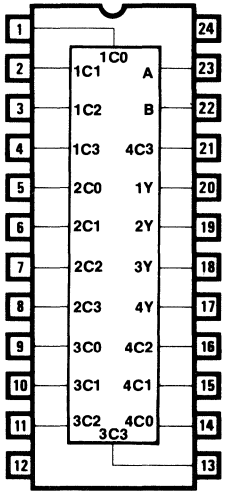
SN54/74LS450
16:1 Mux



SN54/74LS451
Dual 8:1 Mux



SN54/74LS453
Quad 4:1 Mux



The PAL documentation consists of the PAL Design Specification format, which is:

1. PAL part type.
2. Customer part name.
3. Pin identification list.
4. Boolean logic equations with optional comments.
5. Function Table which can be used to functionally test the device.
6. Description.

The purpose of the dual form of documentation is to provide the traditional logic designer with the standard data sheet which he is accustomed to, and to provide the PAL Design Specification which he can use with PALASM (PAL Assembler) to further customize PALs to perform his desired logic function.

HMSI Product Description

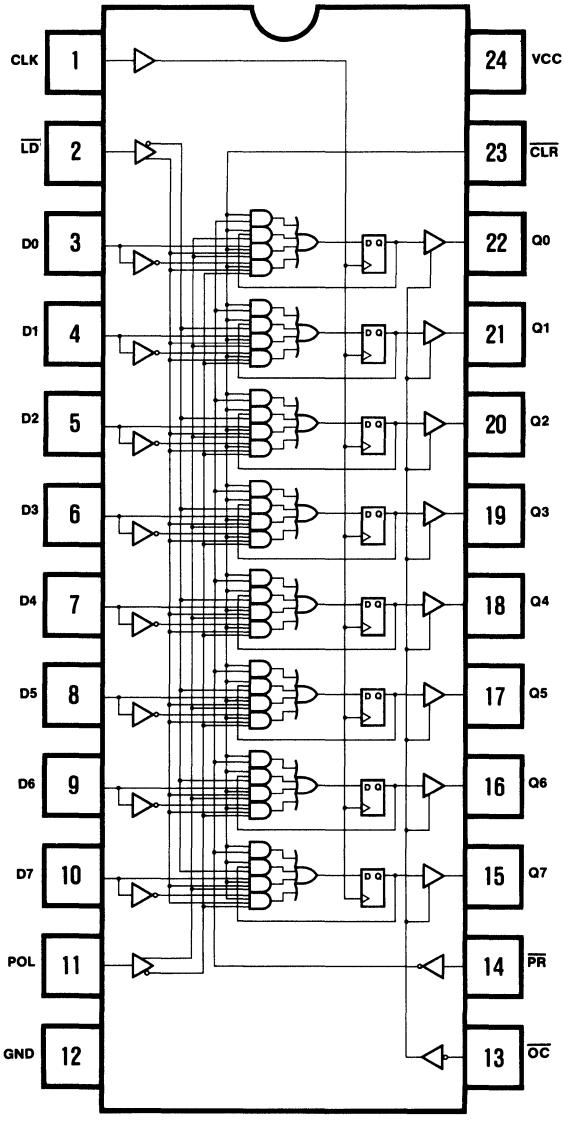
Selected members of the HMSI family will now be described.

Octal Register (SN54/74LS380)

The 8-bit register combines all of the three features found in the LS273 (CLEAR), LS374 (THREE-STATE), and LS273 (CLOCK ENABLE) into one 24-pin skinny DIP package. In addition to the above, this device has extra features: LOAD COMPLEMENT and PRESET, allowing inverting data operations. A Bus-structured pinout is also provided.

This convenient microprocessor interface lowers inventory requirements.

SN54/74LS380
Octal Register

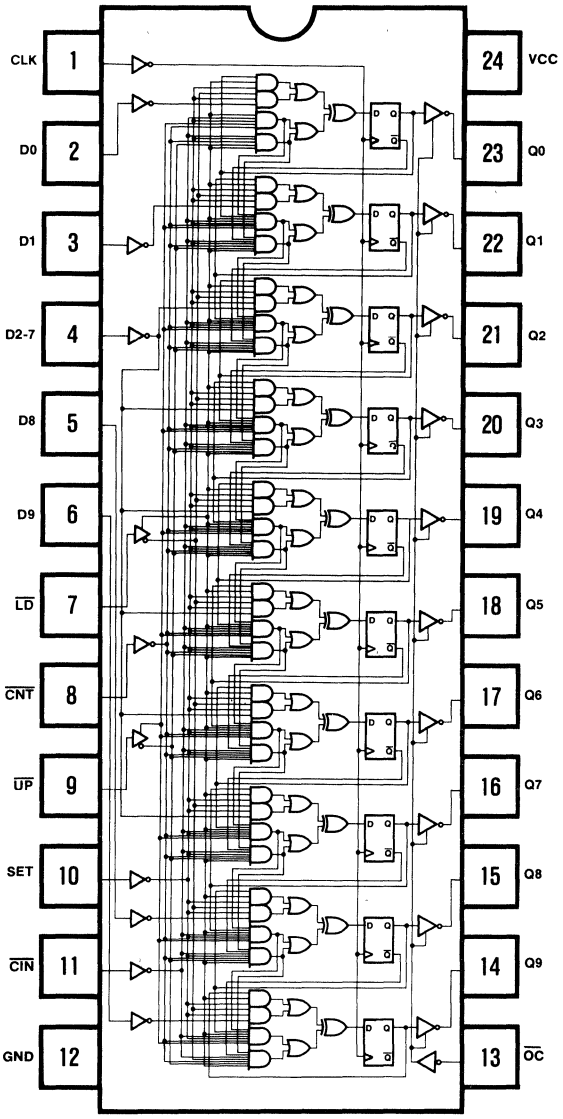


Function Table

OC	CLK	LD	POL	CLR	PR	D7-D0	Q7-Q0	OPERATION
H	X	X	X	X	X	X	Z	HI-Z
L	↑	X	X	L	X	X	L	CLEAR
L	↑	X	X	H	L	X	H	PRESET
L	↑	H	X	H	H	X	Q	HOLD
L	↑	L	H	H	H	D	D	LOAD true
L	↑	L	L	H	H	D	$\overline{D}$	LOAD comp

8

SN54/74LS491
10-Bit Up/Down Counter



10-Bit Up/Down Counter (SN54/74LS491)

The 10-bit up/down counter replaces three 4-bit counters (LS161). This is ideal for video synch generation for CRT graphics including video games. The counter provides the vertical and horizontal coordinates required to address the graphic data and 10-bits supplies ample video resolution. The count down capability allows for screen coordinate reversal.

SET, COUNT ENABLE and LOAD operations and a Bus-structured pinout with THREE-STATE outputs are also included.

A logic equation for Q1 is shown below:

$$
\begin{array}{lll}
\overline{Q1} := & \overline{SET}*LD*\overline{D1} & ;\text{LOAD D1} \\
+ & \overline{SET}*\overline{LD}*\overline{Q1} & ;\text{HOLD} \\
:+: & \overline{SET}*\overline{LD}*CNT*CIN*\overline{UP}*\overline{Q0} & ;\text{INC} \\
+ & \overline{SET}*\overline{LD}*CNT*CIN*\overline{UP}*\overline{Q0} & ;\text{DEC}
\end{array}
$$

This equation is interpreted as follows:
Not ($\overline{\ }$) Q1 is replaced by (:=)
not D1 if (*) not SET and (*) LOAD
or (+) itself if not SET and not LOAD
unless (:+:) INCREMENT and not SET and not LOAD
or DECREMENT and not SET and not LOAD

Function Table

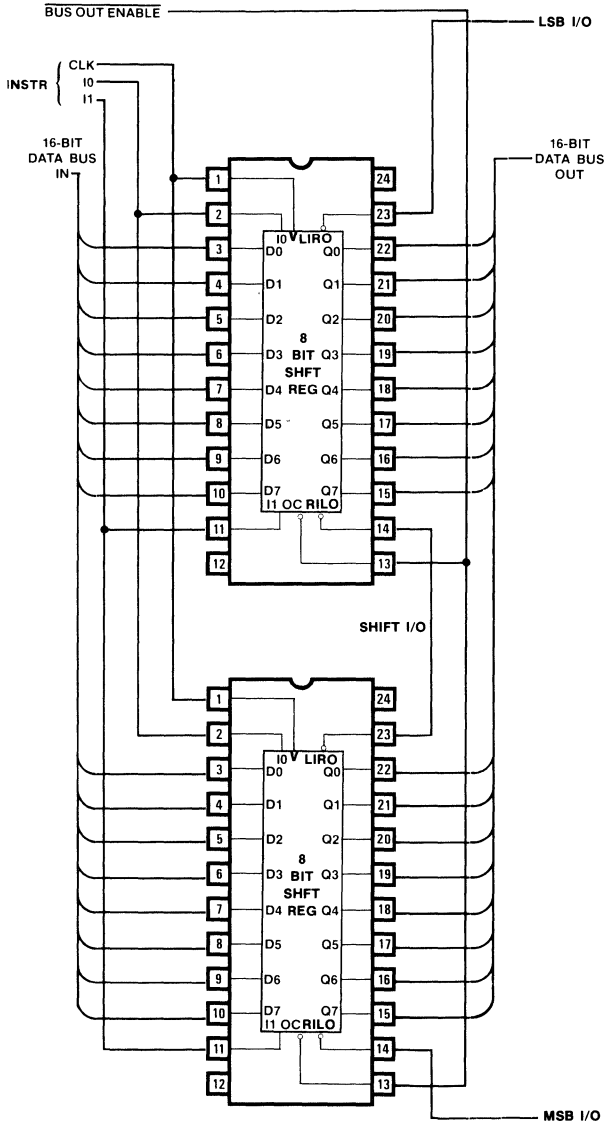
$\overline{OC}$	CLK	SET	$\overline{LD}$	$\overline{CNT}$	$\overline{CIN}$	$\overline{UP}$	D9-D0	Q9-Q0	OPERATION
H	X	X	X	X	X	X	X	Z	HI-Z
L	↑	H	X	X	X	X	X	H	Set all HIGH
L	↑	L	L	X	X	X	D	D	LOAD D
L	↑	L	H	H	X	X	X	Q	HOLD
L	↑	L	H	L	H	X	X	Q	HOLD
L	↑	L	H	L	L	L	X	Q plus 1	Count UP
L	↑	L	H	L	L	H	X	Q minus 1	Count DN

Octal Shift Register (SN54/74LS498)

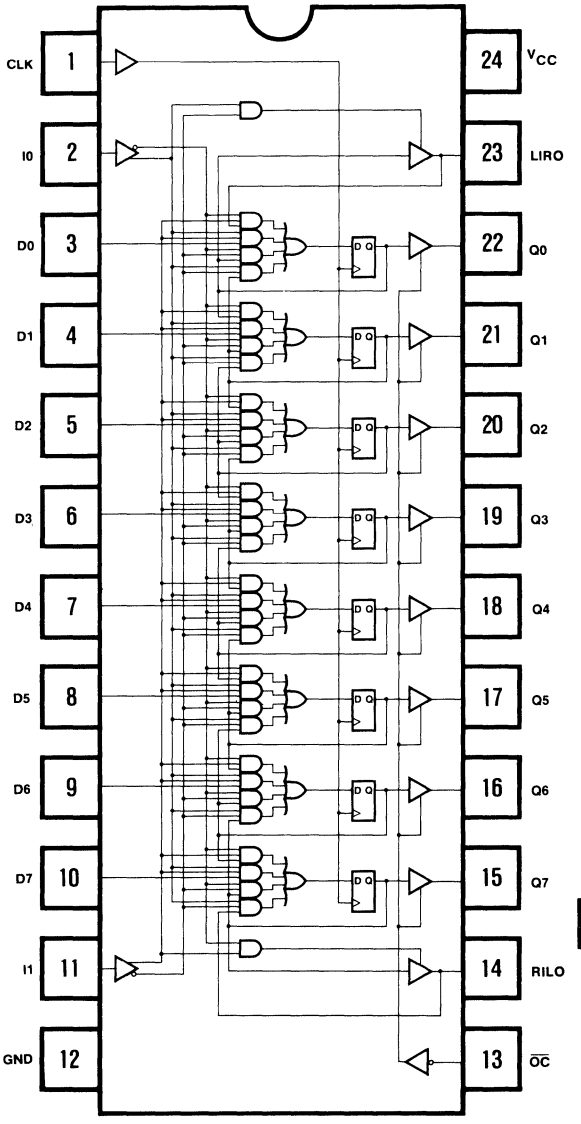
This 8-bit shift register replaces one LS198 fat DIP. It is bit-wide expandable and provides parallel to serial as well as serial to parallel data conversions with its parallel and serial input/output capabilities. SHIFT RIGHT, SHIFT LEFT and HOLD operations and a Bus-structured pinout with THREE-STATE outputs are also included

Note that LIRO and RILO pins are located to facilitate convenient connection when cascading the shift register to provide larger shift registers.

TYPICAL APPLICATION



SN54/74LS498
Octal Shift Register



Function Table

$\overline{OC}$	CLK	I1	I0	D7-D0	Q7-Q0	OPERATION
H	X	X	X	X	Z	HI-Z
L	↑	L	L	X	L	HOLD
L	↑	L	H	X	SR(Q)	SHIFT RIGHT
L	↑	H	L	X	SL(Q)	SHIFT LEFT
L	↑	H	H	D	D	LOAD

8

Quad 4:1 Multiplexer (SN54/74LS453)

The quad 4:1 Mux is twice the density of the LS153. The Mux selects one of four inputs as specified by two binary encoded inputs. The true data is output.

Function Table

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

Dual 8:1 Multiplexer (SN54/74LS451)

The dual 8:1 Mux is twice the density of the LS151. The Mux selects one of eight inputs as specified by three binary encoded select inputs. The true data is output when strobed by the strobe line.

16:1 Multiplexer (SN54/74LS450)

This device replaces one LS150 fat DIP. The 16:1 Mux selects one of sixteen inputs as specified by four binary encoded select inputs. In addition, both true and complement output data are provided.

OCTAL COUNTER (SN54/74LS461)

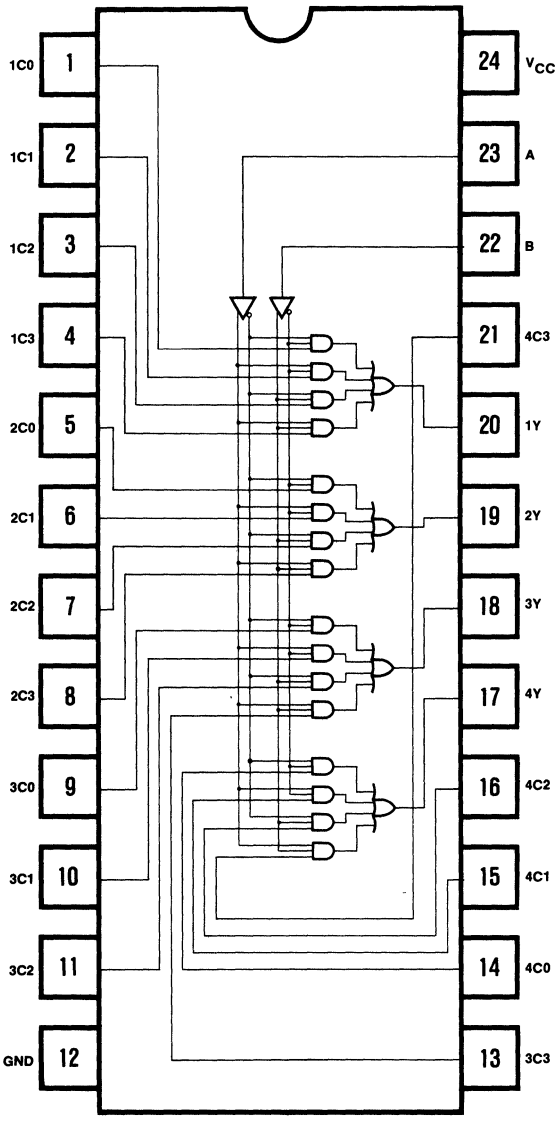
The 8-bit counter replaces two 4-bit counters (LS161). This counter is expandable in 8-bit increments and features CLEAR, LOAD, HOLD, and INCREMENT operations as selected by the two instruction lines. A Bus-structured pinout with THREE-STATE outputs is also provided.

The octal architecture makes this device ideal for microprogram-counter, DMA-controller, and general purpose counting applications.

Note that the operation codes were chosen such that only one instruction LOAD and INCREMENT as in a program counter (JUMP/INCREMENT).

Also carry-in and carry-out pins are located to facilitate convenient connection when cascading the counter to provide larger counters.

SN54/74LS453
Quad 4:1 Mux



PMSI Product Description

Two members of a growing PMSI family will now be described.

9-BIT REGISTER

The 9-bit register loads data into the register on the rising edge of the clock if the load line is asserted. The original data is held in the register if the load line is not asserted regardless of clock pulses. A Bus-structured pinout and THREE-STATE outputs are provided for bus driving capabilities.

The 9-bit architecture makes this device ideal for parity bus interfacing in microprogrammed systems.

Function Table

$\overline{OC}$	CLK	$\overline{LD}$	D8-D0	Q8-Q0	OPERATION
H	X	X	X	Z	HI-Z
L	↑	H	X	Q	HOLD
L	↑	L	D	D	LOAD

A logic equation for Q5 is shown below:

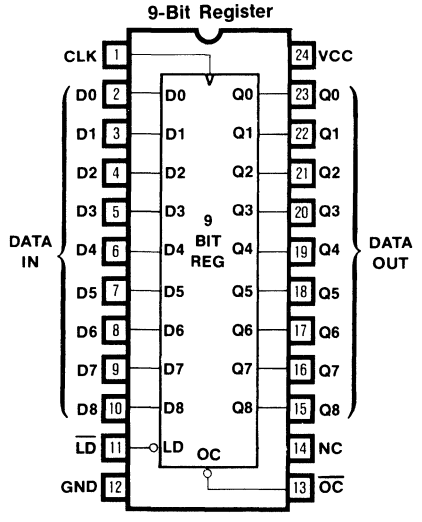
```
Q5  :=  Q5*LD          ;HOLD Q5
    +   D5* LD         ;LOAD D5
```

This equation is interpreted as follows:

Not (¯) Q5 is replaced by (:=)

itself if (*) not LOAD

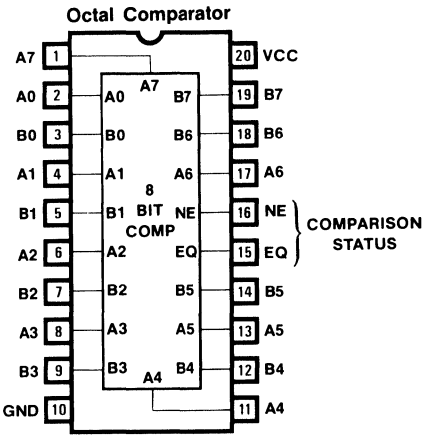
or (+) not D5 if LOAD



OCTAL COMPARATOR

The octal comparator establishes the equivalence of two 8-bit data strings. Both true and complementary outputs are provided.

This pinout is given as a first proposal. It can be changed according to PC board layout.



PAL DESIGN SPECIFICATION:

```
PAL16C1                              PAL DESIGN SPECIFICATION
PMSI003                                  BIRKNER/COLI 09/21/81
OCTAL COMPARATOR
MMI SUNNYVALE, CALIFORNIA
A7 A0 B0 A1 B1 A2 B2 A3 B3 GND
A4 B4 A5 B5 EQ NE A6 B6 B7 VCC
NE  =   A0*B0     +   A0* B0        ;A0 :+: B0
    +   A1*B1     +   A1* B1        ;A1 :+: B1
    +   A2*B2     +   A2* B2        ;A2 :+: B2
    +   A3*B3     +   A3* B3        ;A3 :+: B3
    +   A4*B4     +   A4* B4        ;A4 :+: B4
    +   A5*B5     +   A5* B5        ;A5 :+: B5
    +   A6*B6     +   A6* B6        ;A6 :+: B6
    +   A7*B7     +   A7* B7        ;A7 :+: B7
```

Function Table

A7 A6 A5 A4 B7 B6 B5 B4	A3 A2 A1 A0 B3 B2 B1 B0	NE	EQ	COMMENTS
	INPUT A / INPUT B	OUTPUTS		
7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0			
H L L L L L L L	L L L L L L L L	H	L	A7=H, B7=L
L H L L L L L L	L L L L L L L L	H	L	A6=H, B6=L
L L H L L L L L	L L L L L L L L	H	L	A5=H, B5=L
L L L H L L L L	L L L L L L L L	H	L	A4=H, B4=L
L L L L H L L L	L L L L L L L L	H	L	A3=H, B3=L
L L L L L H L L	L L L L L L L L	H	L	A2=H, B2=L
L L L L L L H L	L L L L L L L L	H	L	A1=H, B1=L
L L L L L L L H	L L L L L L L L	H	L	A0=H, B0=L
L L L L L L L L	H L L L L L L L	H	L	A7=L, B7=H
L L L L L L L L	L H L L L L L L	H	L	A6=L, B6=H
L L L L L L L L	L L H L L L L L	H	L	A5=L, B5=H
L L L L L L L L	L L L H L L L L	H	L	A4=L, B4=H
L L L L L L L L	L L L L H L L L	H	L	A3=L, B3=H
L L L L L L L L	L L L L L H L L	H	L	A2=L, B2=H
L L L L L L L L	L L L L L L H L	H	L	A1=L, B1=H
L L L L L L L L	L L L L L L L H	H	L	A0=L, B0=H
L L L L L L L L	L L L L L L L L	L	H	TEST ALL L
H H H H H H H H	H H H H H H H H	L	H	TEST ALL H
H L H L H L H L	H L H L H L H L	L	H	EVEN CHCKB
L H L H L H L H	L H L H L H L H	L	H	ODD CHCKB

Coming Attractions

As you can see from the nine devices presented in this paper, Hard Array Logic provides an opportunity to replace small and medium-scale TTL components with a chip count reduction while also providing additional features. The family will grow as many new PMSI are being planned.

The most important point to remember is that the designer is always able to customize the HMSI/PMSI device to further suit his functional requirements.

Macro's for Programmable Logic

John Birkner/Wescon 82

The increasing demand for semi-custom silicon is causing a corresponding increase in demand for efficient CAD tools to route that silicon. As computer software has evolved from machine code to assemblers and compilers, computer silicon is evolving from circuit schematics to logic network lists and silicon compilers.

Silicon Compilers

The designers of today's personal computers, business computers, word processors, main frames and video games must get their products to market in the minimum possible time. When they interconnect microprocessors and memories with semi-custom logic, the key to production release is integrity of design (no mistakes please). CAD tools help designers increase the integrity of their designs by providing explicit design specifications, syntax checking, and simulation feedback.

Duality of Definition

Designers have two approaches in specifying silicon systems. The first is the input-output or stimulus response of a system, usually described by a timing diagram, truth table or next state table. The second is the transfer function of a system, usually described by a logic schematic, network list, or logic equation.

The relationships between these two formats are complex and not well understood. The designer usually starts with a known stimulus response and he derives the transfer function. The test engineer (usually in another department or company) takes the transfer function and reverse engineers the stimulus response. For a particular system, the transfer function may be easier to derive than the stimulus response, or vice-versa. For a particular transfer function, there may be an infinite range of stimulus response. For a particuar stimulus response, there may be numerous solutions to the transfer function.

In this dilemma three facts are certain: the transfer function determines the geography of the silicon, the stimulus response determines the final test and a silicon compiler must develop both.

Macro's Increase Efficiency

Macro's allow the designer to specify his design with fewer statements. Fewer statements translate to fewer errors and a faster design cycle. Macro's allow the designer to take advantage of high level functions that are already checked out, so he doesn't have to re-invent the wheel.

Equations and Functions Tables

Existing CAD tools for programmable logic such as the PAL assembler, PALASM, by Monolithic Memories, use equations for the transfer function and function tables for stimulus response. The output from the assembler is the fuse map for silicon geography and test vectors for final test. This paper will suggest additional macro's to improve efficiency of both equations and function tables.

Transportability of Design

Upon completion of a system design, it is useful to be able to transport that investment in engineering effort across silicon vendors as well as different silicon implementations. A design in PALs, for instance, could be upgraded to a higher density PALs or to gate arrays if the right CAD tools are provided.

Natural Language

In selecting high level macro's, consideration is given to understandability thru familiarity with existing concepts and conventions. Readability and ease of use will ultimately determine adoption and success of new languages.

To study these alternatives, the next section will review fundamental logic elements expressed in terms of logic equations. Some assumptions will be made regarding format. The assumptions will later be examined and macro's will be suggested. The format assumes existing conventions as follows:

=	combinatorial assignment
:=	sequential assignment
*	AND
+	OR
:+:	XOR
:*:	NOT XOR
IF	three state
;	comment

The format assumes new conventions as follows:

..	sequential product term expansion
+..+	sequential sum term expansion
N..M	range from N to M
I=N..M	I steps from N to M
.PI	Product term expansion operator
.SUM	Sum term expansion operator

Fundamental Logic Elements

To study the need for equation macro's, fundamental logic elements will now be reviewed.

Combinatorial—Basic

```
A = B           ; A assigned B
A = /B          ; A assigned not B
Y = A*B         ; A AND B
Y = A + B       ; A OR B
Y = A :+: B     ; A XOR B
Y = A :*: B     ; NOT ( A XOR B )
Y = A*S + B*/S; Mux of A,B
```

Combinatorial—Comparison

```
EQ = .PI( I=0..7 A(I):*:B(I) ) ; Equal
NE = .SUM( I=0..7 A(I):+:B(I) ) ; Not equal
GT = .SUM( I=0..7 /A(I)*B(I) ; Greater than
          *.PI( J=I+1..& A(J):*:B(J) )
LT = .SUM( I=0..7 A(I)*/B(I) ; Less than
          *.PI( J=I+1..7 A(J):*:B(J) )
BTL = EQ              ; Between limits
    + LT:+:GT
```

Combinatorial—Basic Data Path

```
Y(I=0..7) = A(I);       Octal buffer
IF( OE )
  Y(I=0..7) = A(I)      ; Octal buffer with
                        ; with three state
```

Combinatorial—Arithmetic Data Path

```
SUM(I=0..7) = A(I):+:B(I)          ; 8-bit adder
            :+:.SUM( J=-1..I-1 A(J)*B(J)*
               .PI( K=J+1..I-1 A(K)+B(K) ) )
```

Sequential—Basic

```
Q := D                    ; D Flipflop
Q := Q*/K +/Q*J           ; JK Flipflop
Q := Q :+: Q*K + /Q*J     ; JK Flipflop
```

Sequential—Data Path

```
Q(I=0..7) := D(I)         ; Octal register
IF(OE)                    ; Octal register
Q(I=0..7) := D(I)         ; with three state
Q(I=0..7) := Q(I−1)       ; Octal shift register
                          ; Q(−1)=Shift-in
Q(I=0..7) := Q(I)         ; Octal counter
             :+: Q(I−1)*..*Q(0) )
Q(I=0..7) := /Q(I)*( /Q(I−1)*..*/Q(0) )
           + Q(I)*( Q(I−1)+..+ Q(0) )
                          ; Octal counter
```

Macro's for Equations

N..M Macro

The N..M macro allows a range to be specified from N to M. A byte, B, for example can be defined as,

B(7..0)

meaning,

B(7) B(6) B(5) B(4) B(3) B(2) B(1) B(0)

This macro saves space and typing time when specifying input/output pin lists.

.EQU Macro

The .EQU macro allows a product term to be specified only once, thereafter to be referred to by symbolic name. A common term such as a memory mapped decode may be symbolically named as follows:

DECODE .EQU IOENABLE*A15*/A14*A13*/A10

This macro saves space, saves typing and improves readability when product terms are repetitious.

I=N..M Macro

The I=N..M macro allows multiple equations set to be specified in one statement. The system of equations of a quad 2:1 mux can be defined as,

Y(I=3..0) = A(I)*S + B(I)*/S

This single statement generates four equations,

Y0 = A0*S + B0*/S
Y1 = A1*S + B1*/S
Y2 = A2*S + B2*/S
Y3 = A3*S + B3*/S

with the obvious savings in space, typing time and readability.

Macro's for Function Tables

Function tables specify the stimulus response for a system and produce test vectors for final test. When the number of nodes and/or pins exceeds about 60, the table becomes unprintable on the printed page. A common method of data compression for test vectors is to use a hexadecimal format.

HEX Macro

The HEX macro allows a function table variable of type N..M to be compressed to hexadecimal format. The allowable values are hex 0 thru hex F, and the special values of X and Z. A function table for an octal register may be expressed as,

```
FUNCTION TABLE
CLK D(7..0) Q(7..0)
;C
;L
;K D Q COMMENT
-------------------------
  C 00 00 all zeros
  C FF FF all ones
  C 55 55 checkerboard
  C AA AA checkerboard
-------------------------
```

which shows the obvious savings in space, typing time and readability.

Examples

The next page shows two examples of design specifications which utilize the proposed macros. The first is a basic example of an octal counter. The second is an octal FIFO which would require 38 complex equations without the proposed macro's.

Basic Example

```
PAL20X8                          PAL DESIGN SPECIFICATION
CNT8                                   J. BIRKNER  6/3/82
OCTAL COUNTER
MONOLITHIC MEMORIES SUNNYVALE, CA
CLK /LOAD D(0..7) /CLEAR GND
/OC /COUT /Q(7..0) /CIN VCC

COUNT /EQU /LOAD*/CLEAR ; EQUATE MACRO
;the above macro saves typing and increases readability
Q(I=0..7) := Q(I)*COUNT
           + D(I)*LOAD*/CLEAR
          :+:Q(I−1)*..*Q(0)*CIN*COUNT
;the above macro creates 8 equations
COUT = Q(7)*..*Q(0)*CIN
```

FUNCTION TABLE

;C ;L ;K	CLK R	CLEAR D	LOAD C	OC D	D(7..0) I	CIN Q	Q(7..0) O	COUNT COMMENT
C	H	L	H	XX	L	00	L	CLEAR
C	L	L	H	XX	L	00	L	HOLD
C	L	L	H	XX	H	01	L	COUNT
C	L	L	H	XX	H	02	L	COUNT
C	L	H	H	55	X	55	L	LOAD 55
C	L	H	H	AA	X	AA	L	LOAD AA
C	X	X	L	XX	X	ZZ	X	HIGH Z

DESCRIPTION

THE OCTAL COUNTER COUNTS, LOADS, CLEARS OR HOLDS SYNCHRONOUS WITH THE RISING EDGE OF THE CLOCK. CLEAR OVERIDES ALL OTHER OPERATIONS.

OPERATIONS TABLE

OC	CLEAR	LOAD	D	Q	OPERATION
L	X	X	X	Z	HIGH-Z
H	H	X	X	L	CLEAR
H	L	L	X	Q+1	INCREMENT
H	L	H	D	D	LOAD

8

FIFO Example

```
SYSTEM DESIGN SPECIFICATION
FIFO001  J. BIRKNER  8/4/81
OCTAL FIFO, 4-WORD DEEP
MONOLITHIC MEMORIES SUNNYVALE, CA
CLK /IR /SI D(0..7) /OC O(7..0) /OR /SO
F(3..0) Q(3..0,7..0)
F(I=0..3) :=    F(I)     */SO*/SI ;HOLD
              + F(I)     */SO* SI ;HOLD
              + F(I-1) */SO* SI ;SHIFT
              + F(I+1) */SO*/SI ;SHIFT
Q( I=0..3 J=0..7 ) :=
                Q(I J)     */SO*/SI
              + Q(I J)     */SO* SI* F(I)
              + D(J)       */SO* SI*/F(I)
              + D(J)       * SO* SI*/F(I+1)
              + Q(I+1 J) * SO* SI* F(I+1)
              + Q(I+1 J) * SO*/SI
IF( OC )
  O(I=0..7) = Q(I)  ;Three state enable
```

Function Table

CLK OC SI IR D(7..0) F(3) Q(3 7..0)
F(2) Q(2 7..0) F(1) Q(1 7..0) F(0)
Q(0 7..0) SO OR

;C ;L ;K	O C	S I	I R	D	3 FQ	2 FQ	1 FQ	0 FQ	O	S O	O R	
C	H	L	H	XX	LXX	XXX	XXX	XXX	XX	H	X	INT1
C	H	L	H	XX	LXX	LXX	XXX	XXX	XX	H	C	INT2
C	H	L	H	XX	LXX	LXX	LXX	XXX	XX	H	X	INT3
C	H	L	H	XX	LXX	LXX	LXX	LXX	XX	H	H	INT4
C	H	H	H	44	LXX	LXX	LXX	H44	44	L	H	SI44
C	H	H	H	55	LXX	LXX	H55	H44	44	L	H	SI44
C	H	H	H	66	LXX	H66	H55	H44	44	L	H	SI66
C	H	H	H	77	LXX	H77	H66	H55	55	H	H	SISO
C	H	H	L	88	H88	H77	H66	H55	55	L	H	SI88
C	H	H	L	99	H88	H77	H66	H55	55	L	H	SI??
C	H	L	H	XX	LXX	H88	H77	H66	66	H	H	SO
C	H	L	H	XX	LXX	LXX	H88	H77	77	H	H	SO
C	H	L	H	XX	LXX	LXX	LXX	H88	88	H	H	SO
C	H	L	H	XX	LXX	LXX	LXX	LXX	XX	H	L	SO??
C	L	L	H	XX	XXX	XXX	XXX	XXX	ZZ	H	L	HI-Z

Description

THE OCTAL 4-WORD DEEP FIFO IS A SYNCHRONOUS SHIFT REGISTER OF VARIABLE LENGTH DEPENDING ON INPUT/OUTPUT SHIFT RATES.

PAL: Quick Turnaround Alternative to Gate Arrays

Shlomo Waser

Abstract

The first part of the paper will describe the PAL family including the second generation to be introduced shortly, and the HAL, which is a gate array with fusible prototype.

The second part will address the PAL software support tools with emphasis on HDL (Hardware Description Language), design verification via function tables, simulation and automatic test vector generation.

Introduction

The Part Number Problem

As the capability of the semiconductor industry to integrate more and more logic into a single chip, a unique problem has arisen; how to define a common chip that will have enough volume to justify the tremendous design costs associated with high degree of integration. Let us illustrate this with an example: Back in the '60s when the integration capability was below 10 gates per chip, we made quad NAND gates and similar SSI. Everybody that made digital systems could use a quad NAND so there was no volume problem. But in the '80s, we can put thousands of gates per chip and the design cost is measured in millions of dollars. As long as we provide memories and microprocessors, there are always enough customers that have common requirements to request large volume. So in the '80s, we don't have a problem with memories and microprocessors, but we do have a problem with what to do with the rest of the required components. Especially the components that are unique to special systems and there is no chance that there will be enough volume generated to justify the large design cost.

The SSI/MSI Solution

One solution to the above problem is to use SSI gates and construct the desired function. This is an undesirable solution since it does not take advantage of the increased integration available nowadays.

The Custom Solution

If the special function has enough volume, a custom design is justified economically, provided the long design cycle associated with custom is acceptable.

The Programmable Solution

There is a big void between the undesirable SSI solutions and the uneconomical custom. This void is now being addressed by semi-custom approaches. Three semi-custom approaches are generally available (in increasing order with gate densities):

1. Fuse programmable logic (PAL, FPLA, and PROM)
2. Gate Arrays
3. Standard Cells

It is estimated that by 1990, half of the semiconductor market will be made of programmable system components. This paper will focus on only the PAL approach to semi-custom, but will compare the PAL to other alternatives.

The PAL Family

The First Generation: PAL20

The PAL is an abbreviation for Programmable Array Logic. The PAL was invented and patented by John Birkner of Monolithic Memories, Inc. The family is made of 15 different parts. All are housed in 20-pin DIP, and are fuse programmable by the user. Of the 15 different parts, 9 are combinatorial and 6 are registered for sequential operations. Figure 1 illustrates the use of a combinatorial PAL to perform address decoding in a typical microprocessor system.

Each PAL is made of one or more of the following cells:

- AND-OR/NOR cell used for combinatorial logic (Figure 2)
- Registered cell with internal feedback. Used to implement sequential logic (Figure 3).
- Programmable I/O cells enable the user to dynamically change the pin function from input to output and vice versa. This feature is very useful for bidirectional communication (Figure 4).

Figure 5 is a general block diagram of a state machine. The state of the machine is determined by the contents of the output register while the next state is a function of the present state and some input variables. Figure 6 depicts the implementation of this classic structure by a registered PAL. Note that the present state information is fed back to the combinatorial network via the internal feedback, thus conserving the use of precious pins.

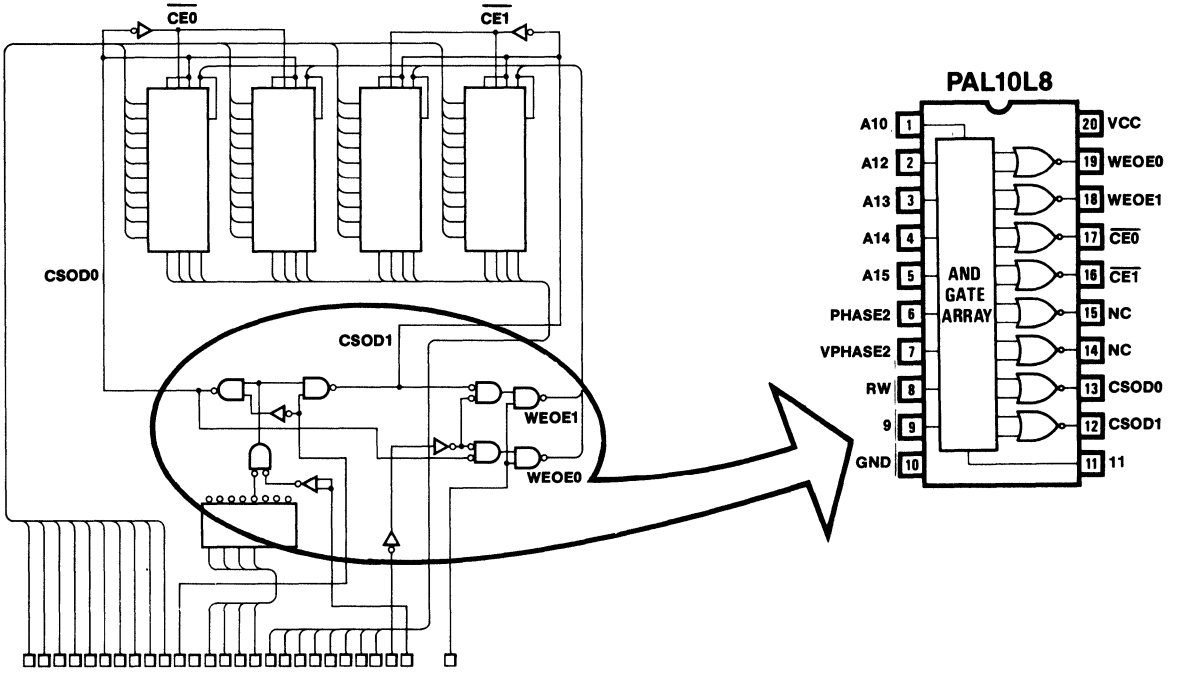


Figure 1. Using PAL to Replace Random Logic in 6800 Microprocessor Bus

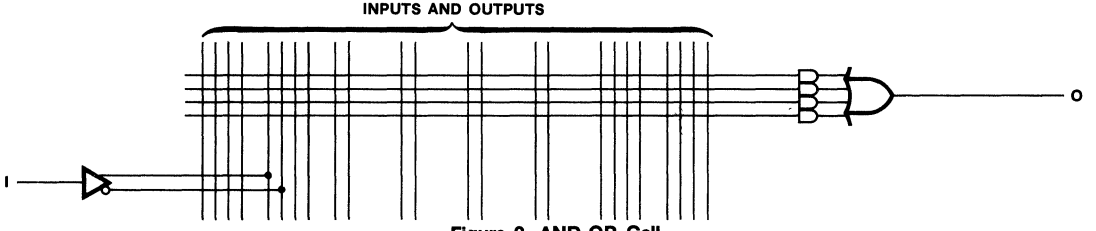


Figure 2. AND-OR Cell

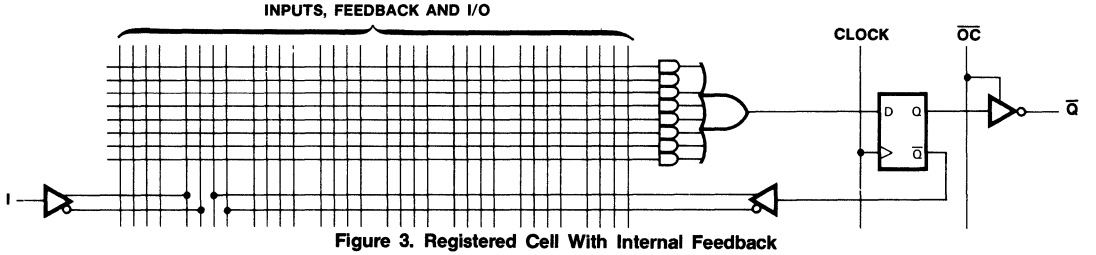


Figure 3. Registered Cell With Internal Feedback

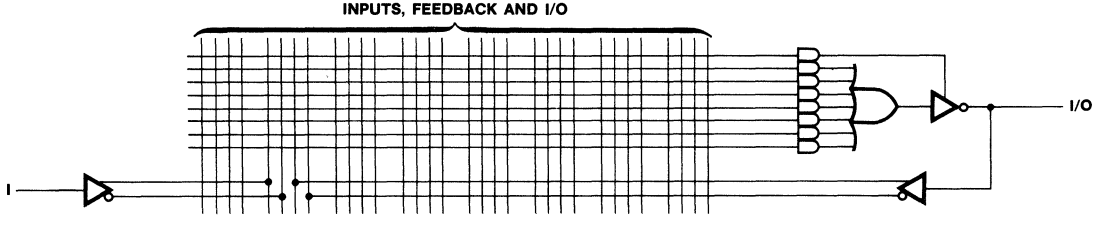


Figure 4. Programmable I/O Cell

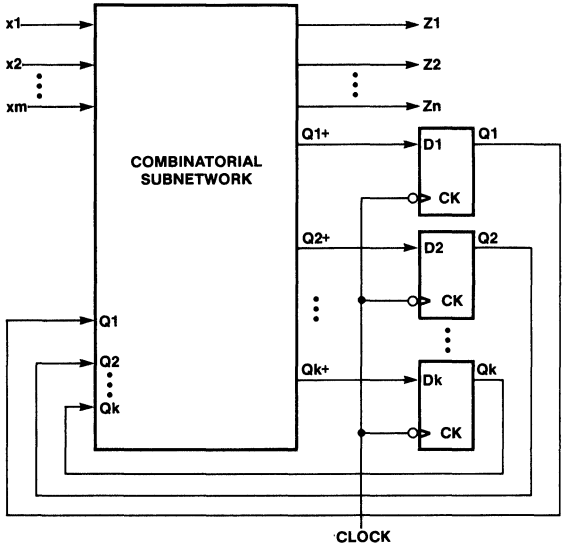


Figure 5. State Machine Block Diagram

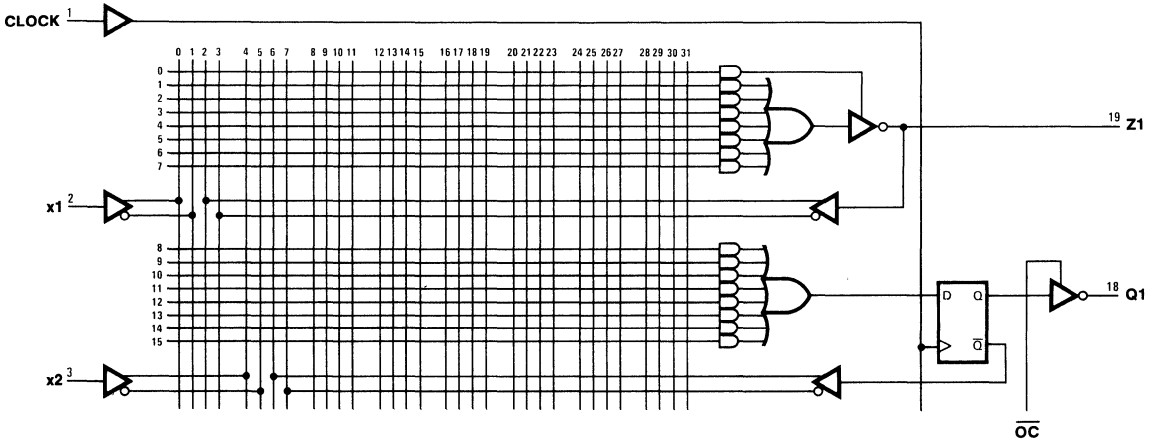


Figure 6. Section of a PAL16R6 Logic Diagram

The Second Generation: PAL24

The PAL24 is an evolution of the PAL20 and similarly has combinatorial and registered parts. However, in the combinatorial circuits, it is now possible to decode simultaneously up to 20 inputs. The registered parts now have as many as 10 outputs, but the important thing to note about the architecture of the registered part is the addition of a new cell: "AND-OR-EXOR" as illustrated in Figure 7. The significance of the cell is for implementing various types of counters. Figure 8 shows the logic diagram for the PAL20X8 that can be easily programmed to implement an octal counter.

PAL Performance: Present and Future

The most critical parameter for the PAL performance is the propagation delay for the combinatorial cells and the set-up time for the registered cells. These two parameters are equal in the PAL family and are listed below as function of time:

	1979	1981	1982	1983
$t_{PD} = t_{SU}$	40 ns	35 ns	25 ns	15 ns

These numbers are worst case for commercial temperature and supply ranges. Add 5 ns to get the worst case numbers for military temperature and supply.

PAL — Industry Standard

The PAL has been accepted as an industry standard and is being manufactured under license by National Semiconductor (NSC) and Advanced Micro Devices (AMD). In addition, it was announced by Texas Instrument (TI). It is very significant to note that TI and NSC, who are the leaders in the TTL market have recognized that PAL will eventually replace the present TTL. The PAL20 was adopted by the JEDEC 42.1 Committee as a de facto standard and it further proposed that the PAL24 be also adopted.

HAL — Hard Array Logic

The HAL to a PAL is the same as a ROM to a PROM. Instead of having a fuse mask, a metal mask is used. This technique reduces the PAL price almost by a factor of two. The HAL is similar to a gate array. Both are customized in the last fabrication stage by applying a unique metal mask to otherwise standard wafer. However, the HAL is the only gate array in the market with a fusible prototype.

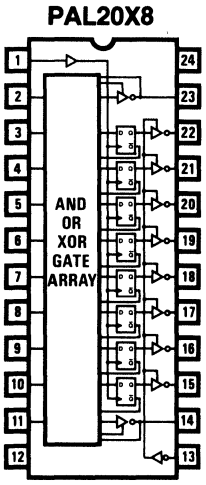


Figure 8. PAL20X8

PAL vs Other Alternatives

PAL Advantages Over SSI/MSI

The PAL can replace from 4-10 SSI/MSI and consequently it reduces the board space required for a function. It also increases reliability by moving the interconnections from the less reliable PC solder connections to the silicon chip. Since the PAL is programmable, it expedites the prototyping and debugging process as compared to conventional SSI/MSI. Last but not least, is the PAL security fuse which makes it impossible to copy the contents of a PAL.

PAL vs Gate Arrays

Gate arrays typically have higher gate density than a PAL. However, they have several drawbacks. The turn-around time for gate array is 2–4 months for the first silicon and typically two iterations are required to get a completely functional unit. Thus, the actual turn-around time is 4–8 months. By contrast, a PAL can be programmed in a few minutes and if several iterations are required, the total turn-around time is still measured in hours, not in months. Even if a customer selects to go with HAL (which has a turn-around time similar to gate arrays), he still does not lose time since he can use the PAL as his fusible prototype to build the first systems.

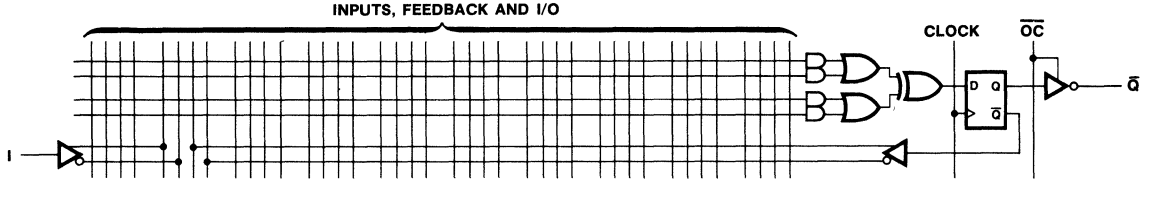


Figure 7. Registered Cell Using XOR Gate

Gate array development cost is estimated to be $20-$50K per circuit, this compared with less than $1K for HAL mask charges or nothing if only PAL is used.

Finally, at the time of writing this paper, the PAL is multi-sourced while gate arrays are single sourced.

PAL vs FPLA

Both approaches use AND array followed by OR array. In the FPLA, both arrays are programmable while in the PAL, only the AND array is programmable. The FPLA approach is academically more flexible but in practicality this flexibility costs time delay and power dissipation. The PAL with its programmable AND array is an engineering trade-off, since the majority of the TTL applications can be implemented by the single programmable array while increasing speed and saving power dissipation.

PROMs Complements the PAL

In few applications, we find that the PAL does not have sufficient product-terms to implement a certain function (especially arithmetic functions). In cases like this, the PROM has a clear advantage since it has all possible product-terms for a given number of inputs. For example: to build a 4x4 multiplier, only 8 inputs and 8 outputs are required; but the number of product terms is more than the 8 available in PAL. Thus, a 256x8 PROM can be used to implement this multiplier (since a 256 words PROM has 8 address lines).

PAL Design Specifications

Motivation

The PAL design specifications serve two functions which are very critical to any user who customizes his own chip:

(a) Documenting the customized function on a computerized data sheet form so that retrieving and updating are easily accomplished.

(b) Automating the design process, i.e., only logic equations are necessary in specifying a new function. The translation of these logic-equations to physcial fuse pattern should be transparent to the user.

The Specifications

The following items constitute the total specs:

- PAL part number
- Heading on user and function
- Pin list
- Logic equations
- Function table (optional)
- Description (optional)

Let us illustrate these specs by going though an example of a 24-pin PAL which is customized as an octal counter. This octal synchronous counter can perform 4 operations. Each operation is selected by a two bit code:

I1	I0	
0	0	Clear
0	1	Hold
1	0	Load
1	1	Increment

The counter has 8 data outputs which are called, "Q0" through "Q7" and 8 data inputs which are called "D0" through "D7", where "D0" is the LSB.

8

So now we are ready to go through the detailed specifications.
The first 4 lines are specified in the following way:

- Line 1 specifies that the device is PAL20X8
- Line 2 specifies user part number
- Line 3 is the device application name
- Line 4 is the user's company name, city and state

```
PAL20X8                                  PAL DESIGN SPECIFICATION
74LS461                                  BIRKNER/KAZMI/BLASCO 02/10/81
OCTAL COUNTER
MMI SUNNYVALE, CALIFORNIA
```

- Line 5 is the beginning of the pin list. In our case there are 24-pins:

  Pin 1 is the clock pin called "CLK"

  Pin 2 is a select line called "I0"

  Pin 3 is the LSB of the input called "D0," and so forth.

```
CLK  I0 D0 D1 D2 D3 D4 D5 D6 D7  I1 GND
/OC /CO Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0 /CI VCC
```

Logic-equations follow the pin list:

```
/Q0 :=  /I1*/I0                              ;CLEAR LSB
     +       I0 * /Q0                        ;COUNT/HOLD
    :+:  I1*/I0 * /D0                        ;LOAD D0 (LSB)
     +   I1* I0 * CI                         ;COUNT

/Q1 :=  /I1*/I0                              ;CLEAR
     +       I0 * /Q1                        ;COUNT/HOLD
    :+:  I1*/I0 * /D1                        ;LOAD D1
     +   I1* I0 * CI*Q0                      ;COUNT

/Q2 :=  /I1*/I0                              ;CLEAR
     +       I0 * /Q2                        ;COUNT/HOLD
    :+:  I1*/I0 * /D2                        ;LOAD D2
     +   I1* I0 * CI*Q0*Q1                   ;COUNT

/Q3 :=  /I1*/I0                              ;CLEAR
     +       I0 * /Q3                        ;COUNT/HOLD
    :+:  I1*/I0 * /D3                        ;LOAD D3
     +   I1* I0 * CI*Q0*Q1*Q2                ;COUNT

/Q4 :=  /I1*/I0                              ;CLEAR
     +       I0 * /Q4                        ;COUNT/HOLD
    :+:  I1*/I0 * /D4                        ;LOAD D4
     +   I1* I0 * CI*Q0*Q1*Q2*Q3             ;COUNT

/Q5 :=  /I1*/I0                              ;CLEAR
     +       I0 * /Q5                        ;COUNT/HOLD
    :+:  I1*/I0 * /D5                        ;LOAD D5
     +   I1* I0 * CI*Q0*Q1*Q2*Q3*Q4          ;COUNT

/Q6 :=  /I1*/I0                              ;CLEAR
     +       I0 * /Q6                        ;COUNT/HOLD
    :+:  I1*/I0 * /D6                        ;LOAD D6
     +   I1* I0 * CI*Q0*Q1*Q2*Q3*Q4*Q5       ;COUNT

/Q7 :=  /I1*/I0                              ;CLEAR MSB
     +       I0 * /Q7                        ;COUNT/HOLD
    :+:  I1*/I0 * /D7                        ;LOAD D7 (MSB)
     +   I1* I0 * CI*Q0*Q1*Q2*Q3*Q4*Q5*Q6    ;COUNT

IF (VCC)  CO = CI*Q0*Q1*Q2*Q3*Q4*Q5*Q6*Q7    ;CARRY OUT
```

Following the logic equations, a FUNCTION TABLE is given. Note that the FUNCTION TABLE includes some basic test information:

```
FUNCTION TABLE

CLK /OC I1 I0 D7 D6 D5 D4 D3 D2 D1 D0 /CI /CO Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0

;                    -INPUT--              -OUTPUT-
;CONTROL    INSTR    DDDDDDDD    CARRY     QQQQQQQQ     COMMENTS
;CLK /OC    I1 I0    76543210    /CI /CO   76543210    (HEX VALUES)
-------------------------------------------------------------------------------
;BASIC LOAD TEST AND INCREMENT-WITH-CARRY TESTS
  C   L      H  L    LLLLLLLH     X   H     LLLLLLLH    LOAD (01)
  C   L      H  H    XXXXXXXX     L   H     LLLLLLHL    INCREMENT
  C   L      H  L    LLLLLLHH     X   H     LLLLLLHH    LOAD (03)
  C   L      H  H    XXXXXXXX     L   H     LLLLLHLL    INCREMENT
  C   L      H  L    LLLLLHHH     X   H     LLLLLHHH    LOAD (07)
  C   L      H  H    XXXXXXXX     L   H     LLLLHLLL    INCREMENT
  C   L      H  L    LLLLHHHH     X   H     LLLLHHHH    LOAD (0F)
  C   L      H  H    XXXXXXXX     L   H     LLLHLLLL    INCREMENT
  C   L      H  L    LLLHHHHH     X   H     LLLHHHHH    LOAD (1F)
  C   L      H  H    XXXXXXXX     L   H     LLHLLLLL    INCREMENT
  C   L      H  L    LLHHHHHH     X   H     LLHHHHHH    LOAD (3F)
  C   L      H  H    XXXXXXXX     L   H     LHLLLLLL    INCREMENT
  C   L      H  L    LHHHHHHH     X   H     LHHHHHHH    LOAD (7F)
  C   L      H  H    XXXXXXXX     L   H     HLLLLLLL    INCREMENT
  C   L      H  L    HHHHHHHH     L   L     HHHHHHHH    LOAD (FF)
  C   L      H  H    XXXXXXXX     L   H     LLLLLLLL    INCREMENT (ROLL OVER)
;COMPLEMENT LOAD TESTS
  C   L      H  L    HHHHHHHH     L   L     HHHHHHHH    LOAD (FF)
  C   L      H  L    HHHHHHHL     X   H     HHHHHHHL    LOAD (FE)
  C   L      H  L    HHHHHHLH     X   H     HHHHHHLH    LOAD (FD)
  C   L      H  L    HHHHHLHH     X   H     HHHHHLHH    LOAD (FB)
  C   L      H  L    HHHHLHHH     X   H     HHHHLHHH    LOAD (F7)
  C   L      H  L    HHHLHHHH     X   H     HHHLHHHH    LOAD (EF)
  C   L      H  L    HHLHHHHH     X   H     HHLHHHHH    LOAD (DF)
  C   L      H  L    HLHHHHHH     X   H     HLHHHHHH    LOAD (BF)
  C   L      H  L    LHHHHHHH     X   H     LHHHHHHH    LOAD (7F)
  C   L      H  L    HHHHHHHH     L   L     HHHHHHHH    LOAD (FF)
;SHORT COUNT SEQUENCE - CONSECUTIVE COUNTS
  C   L      L  L    XXXXXXXX     X   H     LLLLLLLL    CLEAR
  C   L      H  H    XXXXXXXX     L   H     LLLLLLLH    INCREMENT TO (01)
  C   L      H  H    XXXXXXXX     L   H     LLLLLLHL    INCREMENT TO (02)
  C   L      H  H    XXXXXXXX     L   H     LLLLLLHH    INCREMENT TO (03)
  C   L      H  H    XXXXXXXX     L   H     LLLLLHLL    INCREMENT TO (04)
;COUNT HOLD CHECK - CO GATING CHECK
  C   L      H  L    HHHHHHHL     X   H     HHHHHHHL    LOAD (FE)
  C   L      H  H    XXXXXXXX     L   L     HHHHHHHH    INCREMENT TO (FF) /CO=L
  C   L      H  H    XXXXXXXX     H   H     HHHHHHHH    CI INHIBITS COUNT AND CO
  C   L      L  H    LLLLLLLL     L   L     HHHHHHHH    HOLD SEL INHIBITS COUNT ONLY
  C   L      H  H    HHHHHHHH     L   H     LLLLLLLL    INCREMENT TO (00)
;TEST OUTPUT CONTROL (/OC) FOR THREE-STATE OUTPUTS
  X   H      X  X    XXXXXXXX     X   X     ZZZZZZZZ    TEST HI-Z
-------------------------------------------------------------------------------
```

Finally, it is highly recommended that a description be a part of this computerized data sheet.

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

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

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

CAD Software for PAL

PALASM

PALASM is a software which translates the LOGIC EQUATIONS to a fuse pattern. The output of the PALASM is in a format compatible with either a PAL or a PROM programmer. The user needs to specify only his "block-box" description and the programmer will blow the corresponding fuse pattern.

PALASM also uses the FUNCTION TABLE to perform two critical operations for the semi-custom user:

- Design Verification — Each entry of the FUNCTION TABLE is checked against the logic equations and any inconsistency is flagged as an error.
- Test Vectors — Each entry of the FUNCTION TABLE is translated to a universal test vector, so there is a way of testing the customized PAL once it is fabricated

PALASM is written in FORTRAN IV and it resides on an IBM 370/168. Users can access the program by calling the NCSS time-share network. Additionally, Monolithic Memories makes the source code of PALASM available to users at no cost. PALASM is presently running on quite a few computers at customer sites. Several examples of such computers are the DEC PDP/11, DG NOVA, HP2100, MDS800, and many others.

An example of the test-vectors generated from the FUNCTION TABLE of the octal counter is given at right:

```
 1 C0100000001X0HLLLLLLLHX1
 2 C1XXXXXXXX1X0HLLLLLLHL01
 3 C0110000001X0HLLLLLLHHX1
 4 C1XXXXXXXX1X0HLLLLLHLL01
 5 C0111000001X0HLLLLLHHHX1
 6 C1XXXXXXXX1X0HLLLLHLLL01
 7 C0111100001X0HLLLLHHHHX1
 8 C1XXXXXXXX1X0HLLLHLLLL01
 9 C0111110001X0HLLLHHHHHX1
10 C1XXXXXXXX1X0HLLHLLLLL01
11 C0111111001X0HLLHHHHHHX1
12 C1XXXXXXXX1X0HLHLLLLLL01
13 C0111111101X0HLHHHHHHHX1
14 C1XXXXXXXX1X0HHLLLLLLL01
15 C0111111111X0LHHHHHHHH01
16 C1XXXXXXXX1X0HLLLLLLLL01
17 C0111111111X0LHHHHHHHH01
18 C0011111111X0HHHHHHHHLX1
19 C0101111111X0HHHHHHHLHX1
20 C0110111111X0HHHHHHLHHX1
21 C0111011111X0HHHHHLHHHX1
22 C0111101111X0HHHHLHHHHX1
23 C0111110111X0HHHLHHHHHX1
24 C0111111011X0HHLHHHHHHX1
25 C0111111101X0HLHHHHHHHX1
26 C0111111111X0LHHHHHHHH01
27 C0XXXXXXXX0X0HLLLLLLLLX1
28 C1XXXXXXXX1X0HLLLLLLLH01
29 C1XXXXXXXX1X0HLLLLLLHL01
30 C1XXXXXXXX1X0HLLLLLLHH01
31 C1XXXXXXXX1X0HLLLLLHLL01
32 C0011111111X0HHHHHHHHLX1
33 C1XXXXXXXX1X0LHHHHHHHH01
34 C1XXXXXXXX1X0HHHHHHHHH11
35 C1000000000X0LHHHHHHHH01
36 C1111111111X0HLLLLLLLL01
37 XXXXXXXXXXXX1XZZZZZZZZX1
```

Testability

The test vectors which resulted from the FUNCTION TABLE are referred to as basic test vector and they indeed guarantee that the part is functioning according to each entry in the FUNCTION TABLE. However, to increase the confidence level that the device will not "misbehave" (when unspecified inputs are applied) it is necessary to insure that all nodes are toggled during the testing. Monolithic Memories has an automatic test-vector generation program which is used to test HAL, the software uses the basic test-vectors as seed vector and from there it iteratively adds more test vectors until all nodes have been toggled.

It is important to note that for testability it is always necessary to have an initiallization mechanism, e.g., in the octal counter when I0 = I1 = 0, the counter is cleared to all zeros.

HMSI: Dedicated HAL Functions

In general, the PAL and HAL are programmed according to user specifications. However, as an aid to customers, we make certain dedicated 24-pin HAL. For example:

- Octal Counter
- 10-Bit Counter
- 16:1 MUX and several other "new" TTL functions

PAL and the Military

Bipolar Technology

The PALs are fabricated using bipolar Schottky technology which has been proven to be much more reliable at extreme temperatures than the MOS technology. The PAL is being programmed by blowing TiW fusible links which has proven reliability on PROMs. The fusible link technique is much more reliable than the EPROM or EEPROM which use a stored charge that leaks with time and has a questionable operation at the military temperature range.

Low-Volume

Low-volume requirement has been a continuous problem to the military market place, but with PALs obviously the low-volume requirement can be satisfied by customizing as little as one chip at a time.

Obsolete SSI/MSI

Anytime the demand for ICs exceeds the available supply, the semiconductor manufacturer stops producing the least profitable ICs. As the labor cost increases and the silicon cost decreases, SSI/MSI devices (which are labor intensive) are the first to become obsolete. With the fast turn-around time of the PAL. These TTL devices can be quickly reduced by PAL.

Leadless Packages

The PAL is also available in the space saving leadless chip carriers (LLC). These carriers are made according to JEDEC outline (Type B) with 50 mil centers and 75 mil package thickness. The 20-pin LCC is 350x350 mils and occupies only 40% of the space required by the equivalent of 20-pin DIP.

Military Processing

PALs, like all other Monolithic Memories' products, can be processed to meet the HI-REL requirements of MIL-STD-883B. This processing is available for both LCC and DIP packages and also for dice where they are visually inspected by method 2010B and are shipped in a standard waffle pack.

Summary

The PAL family is already a valuable semi-custom approach in the commercial market place. It will even be more valuable to the military market that has been looking for a solution to the problem of low-volume requirement.

The PAL, which is implemented in bipolar technology, presents a high reliability programmable solution to the demanding needs of the military market.

Programmable Array Logic Leads to Flexible Application of 8-Bit Wide Memories

Bernard Brafman

Introduction

The flexible application of memory devices in small microprocessor based systems have been enhanced by the introduction of 8-bit wide static random access memories. These devices have pinouts and operating characteristics that are similar to those of read only and programmable read only memories. Thus, the configuration of both read/write and read only memories can be simplified to meet changing system requirements. For example, this flexibility allows a single printed circuit board to be used in several different product configurations, from fixed program intensive (large read only memory requirements) to data intensive (large read/write memory requirements). Using programmable logic technology in conjunction with 8-bit wide memory devices adds even more flexibility and helps to reduce parts count.

To provide for the simple substitution of parts, a printed circuit (PC) board must be developed that allows interchangeability among the memory sockets of 8k-, 16k-, and 64k-memory devices, both random access memory (RAM) and programmable read only memory (P/ROM), or read only memory (ROM). A method must be created also to define flexibly the addressing of these parts to accommodate varying address space requirements. The first task is physical — e.g., some 64k devices have 28 pins while others have 24 pins. To be most flexible, the PC board will have a 28-pin socket with jumpers to allow the use of either 24- or 28-pin devices. Socket interchangeability also necessitates an examination of the timing requirements of the target memory devices and microprocessor, since differences exist in setup or prechange or hold times for address and data, and control signal interactions. Some systems will need additional gating or delays to ensure flexibility. Such timing information is easily found in application notes published by both microprocessor and memory manufacturers.

The second task is to implement decoding schemes for the selection of the appropriate devices in the memory array. One method uses hardwired discrete logic. This alternative is not viable because high configurability requires an excessive use of space consuming jumpers and also results in unused logic on the PC board. Programmable logic, on the other hand, implements the requisite configurably in fuses on silicon, not on the PC board, and consolidates several integrated circuit (IC) packages into one.

Design Tradeoffs

The choices in programmable logic are programmable logic arrays (PLAs), P/PROMs, and programmable array logic (PAL®). PLAs must be ruled out since their 2-fuse array structure, while awarding some versatility, uses up too much board space. PLAs are implemented in 0.6 in. (1.5 cm), 28-pin packages; the job can be done with smaller 0.3 in. (0.8 cm), 16- or 20-pin packages when P/ROMs or PALs are used.

While P/ROMs are frequently employed in such applications, there are compelling reasons to consider the use of PAL. First, any P/ROM with an adequate number of input and output pins to replace the required external logic consumes more power than a comparable PAL. For example, providing control signals for four memory devices requires four chip enables and at least one output enable signal. These signals must be derived from at least six address lines and generally three and often four control signals. In such a case, the programmable logic device must have ten inputs and five outputs. For a PROM, this would be an 8K, 20 or 24-pin device, organized as 1Kx8. Typically, a commercial 8K bipolar PROM has an I_{CC} of 190 mA over temperature, as compared with 90 mA for a 10-input, 8-output PAL. Even if external logic were employed, so that a 1K PROM, organized as 256x4 were used, the I_{CC} would by typically 130 mA over temperature.

PALs are not susceptible to the "glitching" characteristics of PROMs during the access time from address. To combat glitching, the PROM must be used with registers or, if the control signals are active low only, open collector with pullup resistors for the outputs. Generally, a processor control signal or its derivative serves as the clock for the register or output enable to the PROM, an example of increasing parts count and costs in an attempt to overcome an implementation detail. While the technique usually employed to generate the fuse pattern, or programming, for PROMs is manual, a useful and uniform design tool exists for configuring PALs. A FORTRAN IV program called PALASM (for PAL assembler) is available on the National CSS timeshare network or as source code at no charge for any machine that supports a FORTRAN environment. PALASM converts logic equations describing the function of the target device directly into a fuse pattern format that is compatible with the several PROM programmers which support PAL programming. This assembler allows simple, rapid, and complete design and documentation of PAL configurations in a format useful for communication between engineers. Finally, like PROMs, PALs have a masked counterpart, hard array logic (HAL). HALs are plug compatible with the corresponding PAL, and offer significant cost reductions for high volume applications.

Implementation Example

A typical design with memory device interchangeability, is the 6802 microprocessor based instrument required to have four sockets capable of accepting 2kx8 static RAMs, 2kx8 erasable programmable read only memories (EPROMs), or 4kx8 EPROMs. The four chip enable signals ($\overline{CE}_1$, $\overline{CE}_2$, $\overline{CE}_3$, and $\overline{CE}_4$), the common output enable (OE), and the common read/write control (R/$\overline{W}$) are compatible with the devices selected so that no special gating is needed. The programmable logic device will have as inputs the five high order address bits A_{11} through A_{15} and the control signals VMA (valid memory address) and E (enable). While R/$\overline{W}$ may be used directly for write enable ($\overline{WE}$), it is needed to generate $\overline{OE}$. (See Fig. 1.).

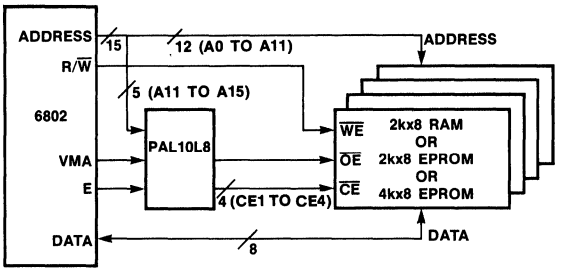
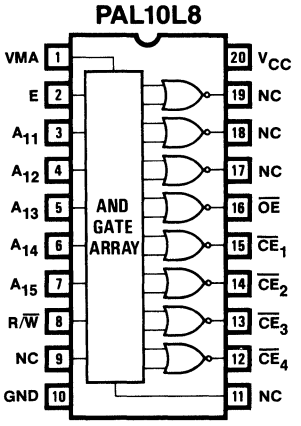


Fig. 1. 6802 based system. Block diagram outlines system's capability to accept 2kx8 static RAMs, 2kx8 EPROMs, or 4kx8 EPROMs.

The programmable logic device requires eight inputs and five outputs. PAL10L8 and PAL12L6 meet both the input and output pin requirements. The PAL10L8 has been selected for this discussion because the three outputs are left unused and are available for logic replacement. (See Fig. 2.). Internally, the PAL10L8 generates eight 2-term OR sums, each sum composed of two 10-term AND products. Each of the products may be connected to any of the input signals or its complement.



GND = GROUND
NC = NOT CONNECTED

Fig. 2. PAL10L8 pinout. Device generates eight 2-term OR sums, each composed of two 10-term AND products.

While there are 81 unique combinations of the three memory devices in the four sockets, there are only 16 distinct decoding schemes implementing either a 2k- or 4k-deep device in any of the four sockets; these 16 schemes are set forth in "Address Ranges of 16 Device Combinations." See "PAL Design Specifications," which can also include a function or truth table, for a description of the arbitrary pinout shown in Fig. 2; this pinout may be altered simply by switching the elements of the pin list on line 5 on both design specifications. Notice that the chip enables and output enable are inverted in the pin list and true in the equations. This is to keep expressions in the sum-of-products form for PALASM, which expects equations summed at the node before the inversion for active low PALs. Also, there are two unused inputs and three unused outputs that may be applied to reduce logic elsewhere in the system

ADDRESS RANGES FOR 16 COMBINATIONS OF DEVICES

CE_1	CE_2	CE_3	CE_4	CONFIGURATION
0000-07FF	0800-0FFF	1000-17FF	1800-1FFF	2k, 2k, 2k, 2k
0000-07FF	0800-0FFF	1000-17FF	1800-27FF	2k, 2k, 2k, 4k
0000-07FF	0800-0FFF	1000-1FFF	2000-27FF	2k, 2k, 4k, 2k
0000-07FF	0800-0FFF	1000-1FFF	2000-2FFF	2k, 2k, 4k, 4k
0000-07FF	0800-17FF	1800-1FFF	2000-27FF	2k, 4k, 2k, 2k
0000-07FF	0800-17FF	1800-1FFF	2000-2FFF	2k, 4k, 2k, 4k
0000-07FF	0800-17FF	1800-27FF	2800-2FFF	2k, 4k, 4k, 2k
0000-07FF	0800-17FF	1800-27FF	2800-37FF	2k, 4k, 4k, 4k
0000-0FFF	1000-17FF	1800-1FFF	2000-27FF	4k, 2k, 2k, 2k
0000-0FFF	1000-17FF	1800-1FFF	2000-2FFF	4k, 2k, 2k, 4k
0000-0FFF	1000-17FF	1800-27FF	2800-2FFF	4k, 2k, 4k, 2k
0000-0FFF	1000-17FF	1800-27FF	2800-37FF	4k, 2k, 4k, 4k
0000-0FFF	1000-1FFF	2000-27FF	2800-2FFF	4k, 4k, 2k, 2k
0000-0FFF	1000-1FFF	2000-27FF	2800-37FF	4k, 4k, 2k, 4k
0000-0FFF	1000-1FFF	2000-2FFF	3000-37FF	4k, 4k, 4k, 2k
0000-0FFF	1000-1FFF	2000-2FFF	3000-3FFF	4k, 4k, 4k, 4k

```
PAL10L8
PN1001                                              B. BRAFMAN 12/10/80
ADDRESS DECODER  (EXAMPLE 1)
MMI SUNNYVALE, CALIFORNIA
VMA E A11 A12 A13 A14 A15 RW NC GND      } PIN LIST OF SYMBOLIC NAMES
NC /CE4 /CE3 /CE2 /CE1 /OE NC NC NC VCC  }

CE1 = /A15 * /A14 * /A13 * /A12 * VMA * E  }
CE2 = /A15 * /A14 * /A13 *  A12 * VMA * E  }
CE3 = /A15 * /A14 *  A13 * /A12 * VMA * E  } LOGIC EQUATIONS
CE4 = /A15 * /A14 *  A13 *  A12 * VMA * E  }
OE  = E * RW

DESCRIPTION                                                    }
  THIS PART GENERATES CHIP ENABLES AND OUTPUT ENABLE FOR       }
  TWO 4k X 8 PROMS AND TWO 2k X 8 STATIC RAMS AS FOLLOWS:      }
          CE1--0000-0FFF                                       } OPERATION
          CE2--1000-1FFF                                       }
          CE3--2000-27FF                                       }
          CE4--2800-2FFF                                       }
```

```
PAL10L8
PN1002                                              B. BRAFMAN 12/10/80
ADDRESS DECODER  (EXAMPLE 2)
MMI SUNNYVALE, CALIFORNIA
VMA E A11 A12 A13 A14 A15 RW NC GND      } PIN LIST OF SYMBOLIC NAMES
NC /CE4 /CE3 /CE2 /CE1 /OE NC NC NC VCC  }

CE1 = /A15 * /A14 * /A13 * /A12 * /A11 * VMA * E  }
CE2 = /A15 * /A14 * /A13 * /A12 *  A11 * VMA * E  }
CE3 = /A15 * /A14 * /A13 *  A12 * /A11 * VMA * E  } LOGIC EQUATIONS
CE4 = /A15 * /A14 * /A13 *  A12 *  A11 * VMA * E  }
    + /A15 * /A14 *  A13 * /A12 *  A11 * VMA * E  }
OE  = E * RW

DESCRIPTION                                                    }
  THIS PART GENERATES CHIP ENABLES AND OUTPUT ENABLE FOR       }
  THREE 2k X 8 STATIC RAMS AND ONE 4k X 8 PROM AS FOLLOWS:     }
          CE1--0000-07FF                                       } OPERATION
          CE2--8000-0FFF                                       }
          CE3--1000-17FF                                       }
          CE4--1800-27FF                                       }
```

Summary

The combination of 8-bit wide memory devices and programmable array logic allows designers to implement highly flexible microprocessor based designs with minimal chip count and reduced overall costs. PALASM serves as a useful tool for expediting prototype cycles and documenting PAL designs; this can reduce future problems when changes must be made by designers unfamiliar with the original design. Finally, products with high volume in a specific configuration may switch over to mask programmable array logic, HAL, to reduce costs further.

PAL bumps eight chips from μP interface

The PAL, or programmable array logic device, can indeed be a good friend to the logic designer who is short on PC-board space for the eight or so SSI and MSI TTL logic chips that usually connect a single-board computer to its memory devices, I/O port adapters, and timer module. The example shown in the figure uses just a single PAL20L10 device to interface an MC6800 microprocessor-based board with its system components.

A fuse-programmable device, the PAL requires PAL assembler source code (see the listing) to assemble a fuse plot that is compatible with PAL and PROM programmers. Once programmed, the logic array decodes addresses, selects memory or I/O ports, and generates a $\overline{\text{RESET}}$ signal. In addition, it controls the buffer that interfaces the data bus with other boards in a system.

In the circuit shown, only the address lines A_{11} through A_{15} are significant. More complete decoding can be achieved by specifying additional address inputs to the PAL.

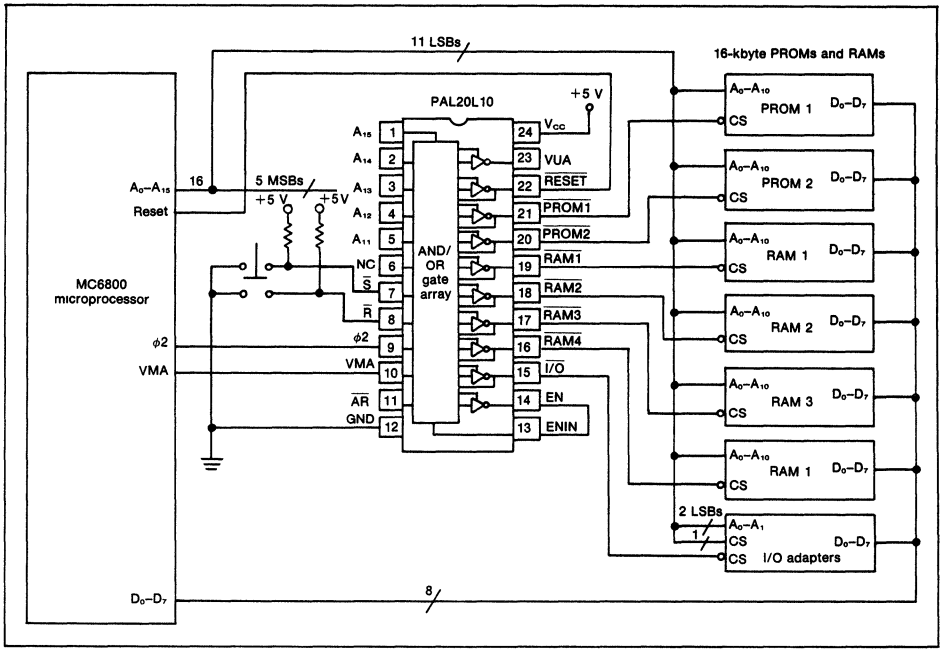
Any unique address can be fully decoded if the logic equation for that address is written so that the product term will be true when the correct address appears. The active low outputs provide the proper polarity for chip-select lines.

Because PALs supply true and complement innput polarities and because any address can be decoded by writing the appropriate equation, system components can be moved through the address space by editing the equations. Also, if all of the listed memory or I/O devices are not required or desired, address space can be freed by a minor equation change. In an upgraded version, pin 6 can be used for additional address decoding.

The PAL determines the presence of a system reset request either sent automatically by the system or by checking the status of its pushbutton-actuated reset pin, pin 8. A reset condition inhibits memory and I/O chip-select lines.

The Valid Unit Address line (VUA), which controls the buffer, is decoded when a memory or I/O line has been successfully selected.

Vincent J. Coli, Applications Engineer, Monolithic Memories Inc., 1165 E. Arques Ave., Sunnyvale, Calif. 94086.



Properly programmed, the programmable array logic device, or PAL, replaces the handful of TTL chips that are usually required to link a microprocessor to its memory and I/O devices.

Design specification

```
A15  A14  A13   A12    A11     NC      /S      /R                PH2      VMA     /AR   GND
ENIN EN   IO    /RAM4  /RAM3   /RAM2   /RAM1   /PROM2            /PROM1   /RESET  VUA   VCC

IF (VCC)      PROM1 =  A15*  A14*  A13*  A12       *             VMA* PH2*/RESET          ;PROM1,    F000-FFFF
IF (VCC)      PROM2 =  A15*  A14*  A13*  /A12      *             VMA* PH2*/RESET          ;PROM2,    E000-EFFF
IF (VCC)      RAM1  =  /A15* /A14* /A13* /A12* /A11*             VMA* PH2*/RESET          ;RAM1,     0000-07FF
IF (VCC)      RAM2  =  /A15* /A14* /A13* /A12*  A11*             VMA* PH2*/RESET          ,RAM2,     0800-0FFF
IF (VCC)      RAM3  =  /A15* /A14* /A13*  A12* /A11*             VMA* PH2*/RESET          ;RAM3,     1000-17FF
IF (VCC)      RAM4  =  /A15* /A14* /A13*  A12*  A11*             VMA* PH2*/RESET          ;RAM4,     1800-1FFF
IF (VCC)      IO    =  A15*  A14*  /A13*  A12*  A11*             VMA* PH2*/RESET          ;I/O,      D800-DFFF
IF (VCC)      /VUA  =  ENIN                                                                ;VUA SIGNAL
IF (VCC)      RESET =  S  +  /R* RESET  +  /AR*                  RESET                    ;RESET SIGNAL
IF (VCC)      /EN   =  /PROM1*/PROM2*/RAM1*/RAM2*               /RAM3*/RAM4*/IO* VMA*/RESET :EN=/VUA
```

Function table

```
A15  A14  A13  A12  A11  /S  /R  /AR  /RESET  PH2  VMA          /PROM1  /PROM2  /RAM1  /RAM2  /RAM3
/RAM4  /IO  EN  ENIN  VUA

:ADDR1   S-R          /RE                    PROM     --RAM--          ENABLE
:54321   /S  /R  /AR  SET   PH2   VMA   1   2    1 2 3 4   I/O   OUT   IN   VUA  COMMENT
-------------------------------------------------------------------------------------------
HHHHX    L   H   L    L     L     H     H   H    H H H H   H     H     H    L    RESET (/S=L)
HHHHX    H   L   H    L     L     H     H   H    H H H H   H     H     H    L    AUTO-RESET
HHHHX    H   L   L    H     L     H     H   H    H H H H   H     L     L    H    NO SELECT PH2=L
HHHHX    H   L   L    H     H     L     H   H    H H H H   H     H     H    L    NO SELECT VMA=L
HHHHX    H   L   L    H     H     H     L   H    H H H H   H     H     H    L    SELECT PROM1
HHHLX    H   L   L    H     H     H     H   L    H H H H   H     H     H    L    SELECT PROM2
LLLLL    H   L   L    H     H     H     H   H    L H H H   H     H     H    L    SELECT RAM1
LLLLH    H   L   L    H     H     H     H   H    H L H H   H     H     H    L    SELECT RAM2
LLLHL    H   L   L    H     H     H     H   H    H H L H   H     H     H    L    SELECT RAM3
LLLHH    H   L   L    H     H     H     H   H    H H H L   H     H     H    L    SELECT RAM4
HHLHH    H   L   L    H     H     H     H   H    H H H H   L     H     H    L    SELECT I/O PORT
```

PAL chip hinders software copying

Keith A Miller
Data I/O, Redmond, WA

An inexpensive programmable-array-logic (PAL) device, added to each hardware system you sell, can discourage purchasers from using unauthorized copies of your software: You just add some special code to your software that renders it inoperable unless the PAL, containing an encrypted key, is present. A different PAL program in each system, matched to software sold specifically for that system, prevents moving software to systems that aren't authorized to run it. Deciphering and modifying the software to circumvent the security usually requires more effort than an unauthorized user is willing to expend.

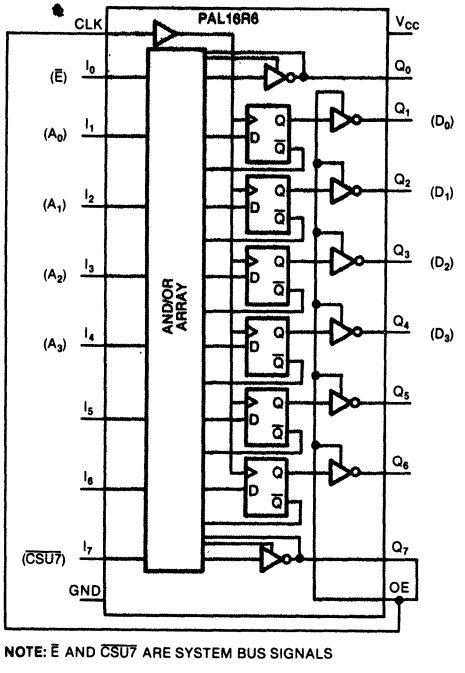


Fig 1—A programmable-array-logic device (PAL) *can store a Boolean-expression key that allows specially prepared software to run on a system containing the device. Systems that don't contain a properly programmed PAL can't run the software.*

INPUT	FUNCTION
00	INITIALIZE (SET OUTPUTS TO ZERO)
01	COMPLEMENT
02	LOGICAL SHIFT RIGHT
03	ARITHMETIC SHIFT LEFT
04	LOGICAL AND WITH INPUT
05	LOGICAL OR WITH INPUT
(a) ALL OTHERS	FORCE ALL OUTPUTS TO ONE

```
/Q1: = /I1 * /I2 * /I3 * /I4
     + I1 * /I2 * /I3 * /I4 * Q1
     + /I1 * I2 * /I3 * /I4 * /Q2
     + I1 * I2 * /I3 * /I4
     + /I1 * /I2 * I3 * /I4

/Q2: = /I1 * /I2 * /I3 * /I4
     + I1 * /I2 * /I3 * /I4 * Q2
     + /I1 * I2 * /I3 * /I4 * /Q3
     + I1 * I2 * /I3 * /I4 * /Q1
     + /I1 * /I2 * I3 * /I4
     + I1 * /I2 * I3 * /I4 * /Q2

/Q3: = /I1 * /I2 * /I3 * /I4
     + I1 * /I2 * /I3 * /I4 * Q3
     + /I1 * I2 * /I3 * /I4 * /Q4
     + I1 * I2 * /I3 * /I4 * /Q2
     + /I1 * /I2 * I3 * /I4 * /Q3

/Q4: = /I1 * /I2 * /I3 * /I4
     + I1 * /I2 * /I3 * /I4 * Q4
     + /I1 * I2 * /I3 * /I4
     + I1 * I2 * /I3 * /I4 * /Q3
     + /I1 * /I2 * I3 * /I4
     + I1 * /I2 * I3 * /I4 * /Q4
```

(b) IF (VCC) /Q7: = /I0 * /I7

Fig 2—By using a logic programmer to store functions (a) in a PAL, *you can create keys for accessing the functions' corresponding sum-of-products expressions* **(b).**

The key contained in the PAL is actually an implementation of a sum-of-products Boolean expression that you choose and program into the PAL. Once programmed, the PAL applies its input signals to the expression and places the results on its output lines. Your specially prepared software need only apply occasional inputs to the PAL and verify that the PAL's output values are the expected ones. You can program the software to halt with an error message if the wrong values appear.

A logic diagram for a PAL (the PAL16R6 from Monolithic Memories Inc) appears in **Fig 1.** This particular device allows you to implement a sum-of-products expression with as many as eight inputs (although only four are used in this example) plus previous-state feedback. You convert the expression to a fuse map, using Monolithic Memories's PALASM program, and then transfer the map into the PAL using a logic programmer such as the Data I/O LogicPak. Blowing the PAL's last fuse (a device that when not blown permits PAL-logic verification) prevents anyone from reading the program.

In the **Fig 1** example, the PAL occupies 16 bytes of address space. The lower four address lines (A_0 through A_3) serve as inputs, and the PAL's outputs appear on the corresponding data lines (D_0 through D_3). A chip-select signal, ORed with system-bus signal $\overline{E}$, clocks the input with its rising edge and enables the output when it goes LOW. Because the signal provides LOW-going pulses, outputs are actually responses to previous inputs.

```
                        ************************************************************
                        *          SOFTWARE LOCK ROUTINE
                        ************************************************************
                        *DESTROYS:A,B,X AND Y REGISTERS
                        *I/O:OUTPUT ADDRESS TO PAL AND INPUT DATA FROM PAL
                        *DESCRIPTION:
                        *THIS ROUTINE SIMPLY DEMONSTRATES THE ACCESS OF THE
                        *SOFTWARE LOCK.  A FIVE VECTOR INPUT BUFFER IS LOADED
                        *AND PRESENTED TO THE LOCK (PAL).  THE RESPONSE IS
                        *VERIFIED AND EITHER "SOFTWARE LOCK INSTALLED PROPERLY"
                        *OR "SOFTWARE PROTECTION WAS VIOLATED" IS OUTPUT TO
                        *THE TERMINAL.  IN A REAL APPLICATION THIS ROUTINE WOULD
                        *BE SPREAD THROUGHOUT THE MAIN SOFTWARE TO PREVENT EASY
                        *DECODING.
                        ************************************************************
                  0FF0  BUFFER  EQU    $0FF0
                  E800  PAL     EQU    $E800
                  F814  MONITOR EQU    $F814
                  F80C  PDATA   EQU    $F80C

0800                            ORG    $0800
0800 108E 0FF0                  LDY    #BUFFER    SET POINTER FOR INPUT VECTORS
0804 8E   E800                  LDX    #PAL       SET POINTER FOR LOCK
0807 CE   088F                  LDU    #OUTPUT    SET POINTER FOR GOOD OUPUT DATA
080A 86   05                    LDA    #$05       LOAD UP THE INPUT VECTOR BUFFER
080C A7   A4                    STA    0,Y
080E 80   02                    SUBA   #$02
0810 A7   21                    STA    1,Y
0812 A7   22                    STA    2,Y
0814 80   02                    SUBA   #$02
0816 A7   23                    STA    3,Y
0818 4C                         INCA
0819 A7   24                    STA    4,Y

081B 5F                         CLRB               INITIALIZE VECTOR COUNTER
081C B7   E800                  STA    PAL         SEND OUT INITIALIZE VECTOR TO LOCK
081F A6   A4          L1        LDA    0,Y         GET INPUT VECTOR
0821 6F   A0                    CLR    0,Y+        CLEAR INPUT VECTOR
0823 30   86                    LEAX   A,X         ADD VECTOR TO LOCK ADDRESS
0825 A6   84                    LDA    0,X         AQUIRE OUTPUT FOR LAST INPUT STIMULUS
0827 84   0F                    ANDA   #$0F        MASK OFF LOWER FOUR BITS
0829 A1   C0                    CMPA   0,U+        COMPARE TO GOOD DATA
082B 26   0A                    BNE    ERROR       NOT EQUAL...PROTECTION VIOLATED
082D 5C                         INCB               BUMP VECTOR COUNTER
082E C1   06                    CMPB   #$06        DONE WITH TEST?
0830 27   0F                    BEQ    EXIT        IF YES...PRINT OKAY MESSAGE
0832 8E   E800                  LDX    #PAL        RESTORE POINTER TO LOCK
0835 20   E8                    BRA    L1          CHECK NEXT INPUT

0837 8E   084B        ERROR     LDX    #ERRORMES   POINT TO ERROR MESSAGE
083A AD   9F F80C               JSR    [PDATA]     OUTPUT MESSAGE
083E 7E   F814                  JMP    MONITOR     BACK TO THE MONITOR

0841 8E   086D        EXIT      LDX    #OKAY       POINT TO OKAY MESSAGE
0844 AD   9F F80C               JSR    [PDATA]     OUTPUT MESSAGE
0848 7E   F814                  JMP    MONITOR     BACK TO THE MONITOR

084B 53 4F 46 54      ERRORMES  FCC    "SOFTWARE PROTECTION WAS VIOLATED "
086C 04                         FCB    $04
086D 53 4F 46 54      OKAY      FCC    "SOFTWARE LOCK INSTALLED PROPERLY "
088E 04                         FCB    $04

088F 00 05 0A 04      OUTPUT    FCB    $00,$05,$0A,$04,$0B,$05 VALID OUTPUT

0 ERROR(S) DETECTED

SYMBOL TABLE:

BUFFER 0FF0   ERROR  0837   ERRORM 084B   EXIT   0841   L1     081F
MONITO F814   OKAY   086D   OUTPUT 088F   PAL    E800   PDATA  F80C
```

Fig 3—This simple routine, *callable from your software, presents a stimulus to the PAL and checks for the expected response. An incorrect response halts the program and generates an error message.*

EXAMPLE	INPUT A_3	A_2	A_1	A_0	OUTPUT D_3	D_2	D_1	D_0
1	0	0	0	0	-	-	-	-
	0	1	0	1	0	0	0	0
	0	0	1	1	0	1	0	1
	0	0	1	1	1	0	1	0
	0	0	0	1	0	1	0	0
	0	0	1	0	1	0	1	1
	-	-	-	-	0	1	0	1
2	1	0	0	0	-	-	-	-
	0	1	0	1	1	1	1	1
	0	0	1	1	1	1	1	1
	0	0	1	1	1	1	1	1
	0	0	0	1	1	1	1	1
	0	0	1	0	0	0	0	0
	-	-	-	-	0	0	0	0
3	0	0	0	0	-	-	-	-
	0	1	0	1	0	0	0	0
	0	1	0	0	0	1	0	1
	0	0	0	1	0	1	0	0
	0	0	1	1	1	0	1	1
	0	0	0	1	0	1	1	0
	-	-	-	-	1	0	0	1
4	0	1	1	1	-	-	-	-
	0	1	0	0	1	1	1	1
	0	0	1	0	0	1	0	0
	0	0	0	1	0	0	1	0
	0	0	1	1	1	1	0	1
	1	0	1	0	1	0	1	0
	-	-	-	-	1	1	1	1
5	0	0	0	0	-	-	-	-
	0	0	0	1	0	0	0	0
	0	1	0	1	1	1	1	1
	0	0	0	1	1	1	1	1
	0	1	0	0	0	0	0	0
	0	0	1	1	0	0	0	0
	-	-	-	-	0	0	0	0

Fig 4—These PAL inputs and outputs *represent stimuli and responses, respectively, for the* ***Fig 2*** *functions.*

Fig 1's example is actually a state-machine implementation that performs functions on the previous outputs. Examples of functions you can choose appear in **Fig 2a,** and their corresponding sum-of-products expressions for programming the PAL appear in **Fig 2b.**

Fig 3 contains a simple example of the software used to access the PAL. This routine loads a buffer of valid PAL addresses (which serve as input vectors), presents vectors to the PAL, verifies correct PAL output and displays a message indicating whether or not a correct PAL is installed in the system. Some example input vectors and the corresponding responses expected from the PAL appear in **Fig 4.**

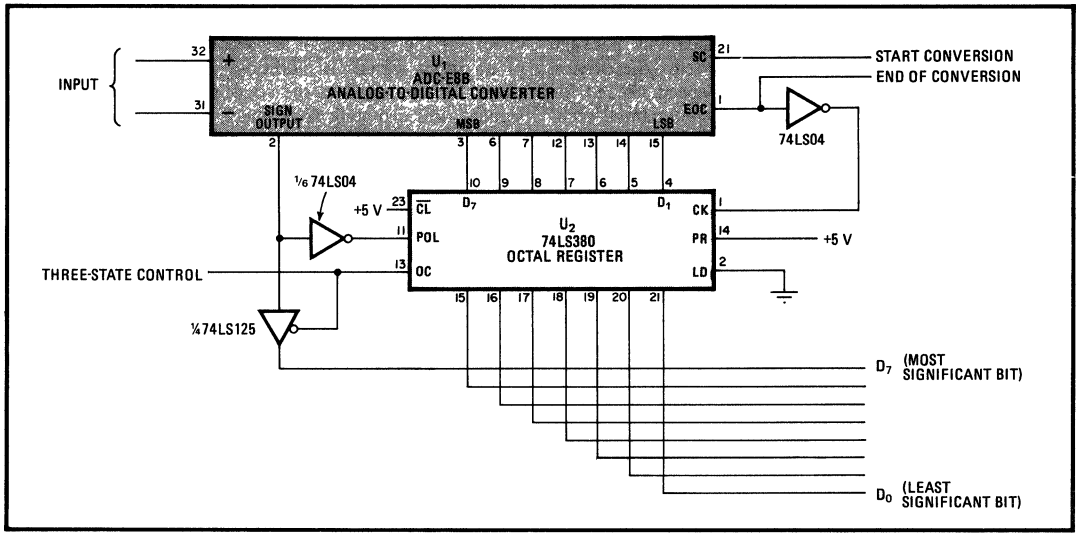
Some additional steps can make this approach even more secure against unauthorized software use; one method involves modifying the PAL input vectors at various points in the software. By making the last modification just before presenting a PAL input and then destroying the vectors immediately after their input, you can make the code quite difficult to break. Using a logic analyzer to break the code, for example, requires triggering within a narrow window to catch the valid input vectors. Having your software make dummy reads from the PAL could add still another dimension of complexity. **EDN**

Octal register links dissimilar a-d converters

by K. Russell Peterman
Radian Corp., Austin, Texas

Many microprocessor-based systems use more than one type of analog-to-digital converter and thus require a circuit to compensate for the differences in output coding. Two of the more popular varieties—complementary-2's complement for successive-approximation converters and sign-magnitude binary for integrating converters—may use this design (see figure). The circuit matches outputs by converting sign-magnitude–binary data into 2's complement with only one multifunction octal register chip and can be applied in all sign-magnitude–binary converters.

When the sign bit is 1, the converter inverts all bits except the sign bit, and when the sign bit is 0, no bits are inverted (see table). The binary code contains two 0s (+ or –). However, the 2's-complement code does have an asymmetry about 0, thus eliminating the double zero. As a result, there is a 1-bit discrepancy between the dc voltages represented by the two codes for input values of 0 or negative. Higher resolution converters may be obtained by adding a second 74LS380 chip to the circuit, and sign-magnitude, binary-coded–decimal converters require BCD-to-binary conversion preceeding the multifunction register. □



Conversion. Using just multifunction octal register U_2, this circuit converts sign-magnitude binary to complementary-2's complement. The circuit inverts all bits except the sign bit when the sign bit is 1 and inverts none when the sign bit is 0.

CONVERTING SIGN-MAGNITUDE BINARY TO 2's COMPLEMENT

Scale	Sign-magnitude binary								Scale	2's complement							
	Sign	Most significant bit						Least significant bit		MSB							LSB
+FS–1 LSB	1	1	1	1	1	1	1	1	+FS–1 LSB	1	1	0	0	0	0	0	0
+1 LSB	1	0	0	0	0	0	0	1	+1 LSB	1	1	1	1	1	1	1	0
+0	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
–0	0	0	0	0	0	0	0	0	–1 LSB	0	0	0	0	0	0	0	0
–1 LSB	0	0	0	0	0	0	0	1	–2 LSB	0	0	0	0	0	0	0	1
–FS+1 LSB	0	1	1	1	1	1	1	1	–FS	0	0	1	1	1	1	1	1

8

SINGLE-CHIP CONTROLLER INCREASES MICROPROCESSOR THROUGHPUT

Design technique uses semicustom logic to minimize hardware in a DMA controller that combines fast response with the potential to service multiple input or output devices and the flexibility to handle many different applications

Alan W. Bentley

Cubic Corporation
9333 Balboa Ave, San Diego, CA 92123

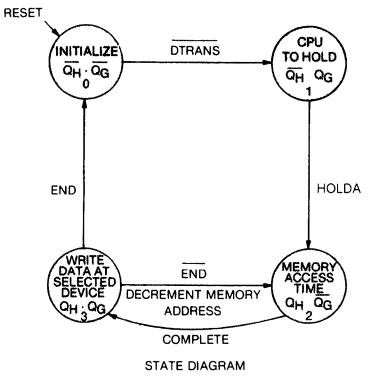
Microprocessors have a broad range of capabilities; yet they are unable to perform specific functions efficiently without hardware augmentation. One such function is transfer of blocks of data to or from processor controlled memory. Under specific conditions, such blocks may be moved efficiently by direct memory access. Data must be sequentially ordered, a starting address must be specified for both the processor associated memory and the external source or destination, and the data block length must be specified. When these three conditions are met, data transfer responsibility passes to a hardware controller, and data moves rapidly because instruction processing is eliminated during the transfer.

Although direct memory access (DMA) efficiently transfers blocks of data between a source and a destination, it normally interfaces memory with only a single external device and is unable to move data between memory and a distributed network. The need to interface a microprocessor with a network of distributed devices led to a generalized concept that can be structured to transfer data rapidly in either direction.

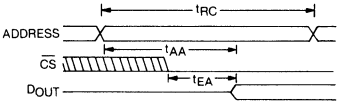
This concept is analogous to the DMA function in that blocks of sequential data are transferred to or from memory. It differs in that, externally, each data word has an associated device location. Implementing the transfer function in terms of this concept increases the data transfer rate while imposing four constraints on the system: the data block length must be constant; data must occupy sequential memory locations; the distributed devices are always serviced in the same sequence; and each implementation is committed to either input or output.

Controller Design Objectives

Conceptually, the controller accepts data transfer responsibility from the microprocessor upon command; steps through the data block, sequentially creating data paths between memory and various external devices; then returns control to the microprocessor. Functionally, upon receipt of the transfer mode command, the controller places the central processing unit (CPU) in the 3-state mode, generates all

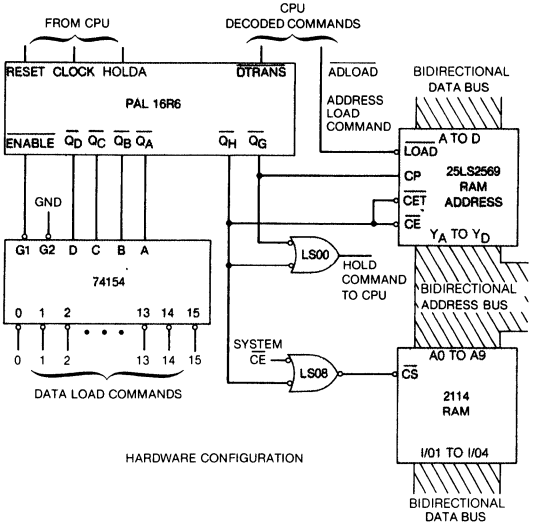


STATE DIAGRAM



t_{RC} = READ CYCLE TIME
t_{AA} = ACCESS TIME - ADDRESS TO D_{OUT}
t_{EA} = ACCESS TIME - CHIP SELECT TO D_{OUT}

RAM TIMING REQUIREMENTS



HARDWARE CONFIGURATION

GENERAL

1 CPU HOLD COMMAND IS Q_G + Q_H (STATE 1, 2, OR 3)

2 STATE 2 IS ENTERED UPON RECEIPT OF HOLDA (CPU IN HOLD, CPU ADDRESS AND DATA BUSES 3-STATE)

3 IN STATES 2 AND 3 RAM ADDRESS COUNTER OUTPUT IS REMOVED FROM 3-STATE MODE AND ITS COUNT CAPABILITY ENABLED THROUGH $\overline{CET}$

1 AT TRANSITION FROM STATE 3 TO STATE 2, BOTH DATA COUNTER AND RAM COUNTER ADDRESS ARE DECREMENTED BY RISING EDGE OF $\overline{Q_G}$

5 UPON COMPLETION OF DATA TRANSFER (END), STATE 0 IS ENTERED FROM STATE 3, DATA COUNTER IS SET AT ITS INITIAL COUNT, ADDRESS COUNTER IS PLACED IN 3-STATE MODE, AND HOLD COMMAND TO CPU IS RELEASED

SPECIFIC (TRANSFER CPU MEMORY TO EXTERNAL DEVICES)

1 RAM IS SELECTED IN STATES 2 AND 3, $\overline{Q_H}$

2 IN STATE 3, ENABLE SELECTS DEMULTIPLEXER, AND CONTENTS OF DATA COUNTER (Q_A THROUGH Q_D) ARE DECODED TO GENERATE DATA LOAD (WRITE ENABLE) PULSE

SEQUENCE OF EVENTS

Fig 1 Transfer from CPU memory to device. PAL implements two categories of Boolean expressions, data counters that control destination selection and state equations that control sequence of events shown in state diagram

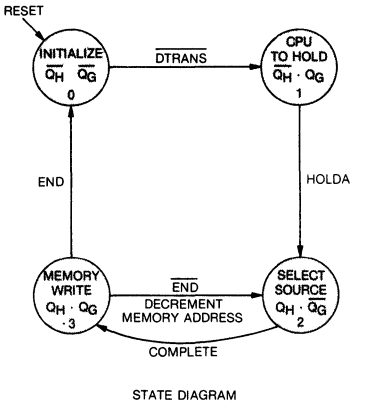
required timing and control signals, sequentially creates the data paths, and transfers data through each path. Upon completion of the data transfer, it relinquishes control to the microprocessor, which then resumes instruction processing.

Increased throughput is the implementation rationale; therefore, the two primary design objectives are to enter and leave the data transfer mode in a minimum number of clock periods and to transfer one data word every two clock periods. This transfer rate allows one clock period to establish each data path, with the write pulse created in the second period to complete the transfer. Secondary design objectives relate to implementation and allow modification to meet varying applications using a minimal amount of additional circuitry.

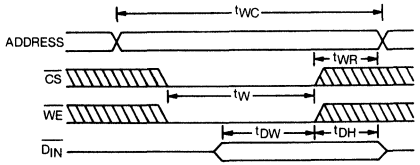
Design approaches that meet these objectives include use of a state sequencer to meet the control transfer and data transmission timing requirements, and use of fusible link, semicustom logic to meet the circuit flexibility and minimization requirements. A simplified microprocessor interface is achieved by incorporating the microprocessor clock and hold capabilities into the sequencer design.

State Sequencer

The state sequencer organizes and concentrates logical functions to realize the high density capabilities of semicustom logic. What determines the number of control flipflops is the number of required states; therefore,



STATE DIAGRAM



t_{WC} = WRITE CYCLE TIME

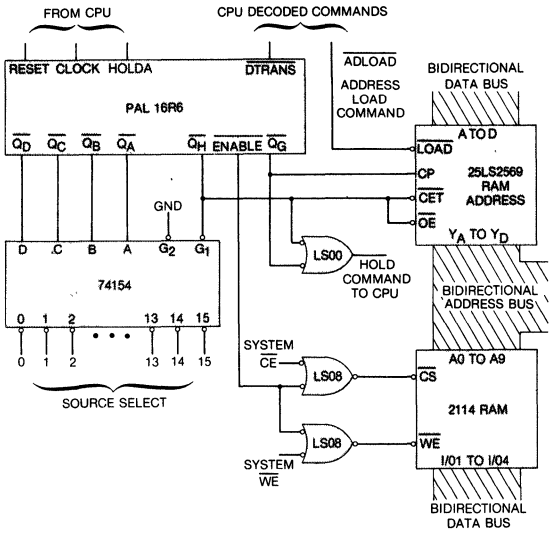
t_W = WRITE TIME

t_{DW} = DATA TO WRITE TIME OVERLAP

t_{DH} = DATA HOLD TIME FROM WRITE TIME

t_{WR} = WRITE RELEASE TIME

RAM TIMING REQUIREMENTS



HARDWARE CONFIGURATION

GENERAL

1 CPU HOLD COMMAND IS Q_G + Q_H (STATE 1, 2, OR 3)

2 STATE 2 IS ENTERED UPON RECEIPT OF HOLDA (CPU IN HOLD, CPU ADDRESS AND DATA BUSES 3-STATE)

3 IN STATES 2 AND 3 RAM ADDRESS COUNTER OUTPUT IS REMOVED FROM 3-STATE MODE AND ITS COUNT CAPABILITY ENABLED THROUGH $\overline{CET}$

4 AT TRANSITION FROM STATE 3 TO STATE 2, BOTH DATA COUNTER AND RAM COUNTER ADDRESS ARE DECREMENTED BY RISING EDGE OF Q_G

5 UPON COMPLETION OF DATA TRANSFER (END), STATE 0 IS ENTERED FROM STATE 3, DATA COUNTER IS SET AT ITS INITIAL COUNT, ADDRESS COUNTER IS PLACED IN 3-STATE MODE, AND HOLD COMMAND TO CPU IS RELEASED

SPECIFIC (TRANSFER EXTERNAL SOURCE TO CPU MEMORY)

1 IN STATES 2 AND 3 ($\overline{Q_H}$) DEMULTIPLEXER IS ENABLED, DECODES DATA COUNTER CONTENTS, AND SELECTS EXTERNAL SOURCE

2 DURING STATE 3 $\overline{ENABLE}$ IS RAM CHIP SELECT AND WRITE ENABLE, WRITING SOURCE DATA WORD IN SELECTED ADDRESS LOCATION

SEQUENCE OF EVENTS

Fig 2 Transfer from device to CPU memory. Similar to Fig 1, with hardware configuration and state diagram modified to transfer data in opposite direction

performing all logic functions in four states reduces the sequencer design requirement to two flipflops and helps meet the secondary design objective. These four states are idle, when the CPU has system control; transitional, while the CPU is entering the hold mode and relinquishing control; and two data transfer, one permitting address and data stabilization, and the other creating the write command (Figs 1 and 2).

In this application, both flipflops are reset in the idle state ($\overline{Q_H}$,$\overline{Q_G}$, state 0) awaiting a software generated command ($\overline{DTRANS}$), which sets the G flipflop, advances the sequencer to state 1 ($\overline{Q_H}$,$\overline{Q_G}$), and initiates a CPU hold command, sustained until the sequencer returns to state 0. The CPU tests its hold command input each machine cycle and, upon sensing a command, enters the hold state, placing its data bus, address bus, and some control lines in the 3-state mode. It remains in this state until the hold command is released by the sequencer.

When it enters the hold state, the CPU issues a hold acknowledge (HOLD). Upon receipt of HOLDA, the

sequencer advances to state 2 (Q_H,$\overline{Q}_C$), which completes transfer of the data handling function from the CPU to the controller. It then toggles between state 2 and state 3 (Q_H,Q_C), processing one data word each cycle until the preestablished number of data words have been transferred (END). Then, the sequencer returns to state 0 and releases the hold command, allowing resumption of instruction processing by the CPU.

While the sequence of states is the same for data transfer in either direction, the events that occur within states 2 and 3 differ. During transfer of data from memory to external devices, state 2 is memory access time and state 3 generates the load command for the specific device. When leaving state 3, the edge of the G flipflop ($\overline{Q}_C$) advances the memory address counter. When data from external devices are read into memory, state 2 time is used to select the external source and stabilize the data. The memory write enable pulse is generated by state 3, and leaving state 3 changes the memory address counter. Transfer states ST2 and ST3 are identified by the output of the H flipflop, which activates the random access memory (RAM) address counter outputs, enables address counting, and, additionally, selects either the RAM or the multiplexer when transferring to or from the external devices, respectively.

Programmed Array Logic

Utilizing semicustom integrated circuitry satisfies the requirement for logic minimization and design flexibility. This approach increases logic density by mitigating two factors that increase random logic package count: function partitioning and pin limitation. To maximize these benefits, the programmable logic chip must contain AND, OR, and flipflop functions, as well as internal feedback. Flipflop functions are required to implement the sequencer and, additionally, a counter that is decremented at the completion of each data transfer (state 3 to state 2 transition). The counter serves a dual purpose: the demultiplexed count selects external devices and, by monitoring the count, the controller determines completion of the data block transfer (END).

Data block length determines the number of counters and demultiplexers required. With three counters and a 3 to 8 demultiplexer (74138), block lengths up to eight may be accommodated. With four counters and a 4 to 16 demultiplexer (74157), the block length may be extended to 16. An increase to six counters will allow block length expansion to 64, with additional demultiplexers. This approach focuses on design for a block length of 16 words so that the flipflops and associated input decoding can be contained on one fusible link, programmable array logic (PAL) integrated circuit.

PAL 16R6 contains six D flipflops (registers) connected to a common clock input. Output Q of each register is available externally through an inverting, 3-state buffer ($\overline{Q}$). Input D of each register is an 8-input OR gate, each input a programmable AND combination of the available terms. Each gate array must be true to set the register on the following clock pulse or false to allow the register to reset. Thus, Boolean expressions state the conditions that cause the registers to become or remain set.

Two additional, non-register, AND-OR gate outputs and the six register outputs have internal feedback paths. With the eight available inputs, they become the 16 terms available when implementing Boolean expressions.

Boolean Expressions

Six counter equations are shown in Fig 3. For a maximum block length of 16, the four least significant counts (A through D) are required; expansion up to length 32 requires equation E; equation F must be implemented for block lengths between 33 and 64. The counter flipflops are held set during states 0 and 1. States 2 and 3 require only one clock period each; hence, from leaving state 1 until completion of data transmission and return to state 0, there is a state transition in every clock cycle.

During this interval, each equation must establish the condition of its counter flipflop for the following state. When entering state 2 from state 1, the counter flipflops are set as established by state 1. Since it is necessary to change the counter on the transition from state 3 to state 2, the counting decision must be made during state 3, and the new count must be established when entering state 2. The counter register content must be maintained during state 2 to prevent change at the transition from state 2 to 3.

Fig 3 also shows the 2-state sequencer equations. Equation G is independent of block length, and equation H

8

$$Q_F = \overline{Q_H} + Q_H \cdot \overline{Q_G} \cdot Q_F + Q_H \cdot Q_G (Q_F \cdot Q_E + Q_F \cdot Q_D + Q_F \cdot Q_C + Q_F \cdot Q_B + Q_F \cdot Q_A)$$

$$Q_E = \overline{Q_H} + Q_H \cdot \overline{Q_G} \cdot Q_E + Q_H \cdot Q_G (Q_E \cdot Q_D + Q_E \cdot Q_C + Q_E \cdot Q_B + Q_E \cdot Q_A + \overline{Q_E} \cdot \overline{Q_D} \cdot \overline{Q_C} \cdot \overline{Q_B} \cdot \overline{Q_A})$$

$$Q_D = \overline{Q_H} + Q_H \cdot \overline{Q_G} \cdot Q_D + Q_H \cdot Q_G (Q_D \cdot Q_C + Q_D \cdot Q_B + Q_D \cdot Q_A + \overline{Q_D} \cdot \overline{Q_C} \cdot \overline{Q_B} \cdot \overline{Q_A})$$

$$Q_C = \overline{Q_H} + Q_H \cdot \overline{Q_G} \cdot Q_C + Q_H \cdot Q_G (Q_C \cdot Q_B + Q_C \cdot Q_A + \overline{Q_C} \cdot \overline{Q_B} \cdot \overline{Q_A})$$

$$Q_B = \overline{Q_H} + Q_H \cdot \overline{Q_G} \cdot Q_B + Q_H \cdot Q_G (Q_B \cdot Q_A + \overline{Q_B} \cdot \overline{Q_A})$$

$$Q_A = \overline{Q_H} + Q_H \cdot \overline{Q_G} \cdot Q_A + Q_H \cdot Q_G \cdot \overline{Q_A}$$

$$Q_G = \overline{\text{RESET}} (\overline{Q_H} \cdot \overline{Q_G} \cdot \overline{\text{DTRANS}} + \overline{Q_H} \cdot Q_G \cdot \overline{\text{HOLDA}} + Q_H \cdot \overline{Q_G})$$

$$Q_H = \overline{\text{RESET}} [\overline{Q_H} \cdot Q_G \cdot \text{HOLDA} + Q_H \cdot \overline{Q_G} + Q_H \cdot Q_G (\overline{\text{END}})]$$

Fig 3 Boolean statements. PAL registers implement data counter and state controller equations. Control equation G is independent of block length. Control equation H must be expanded by expression for $\overline{\text{END}}$ to establish block length

incorporates the block length ($\overline{END}$) decoding. The table, Block Length Control, shows the expressions for $\overline{END}$, when six counters are used, organized by remainder of a modulo 16 number. For shorter block lengths, references to unimplemented counters are ignored. Both control flipflops are reset (expressions false) when the computer issues a RESET command, establishing state 0. This state is maintained until the CPU issues $\overline{DTRANS}$, advancing the sequencer to state 1 by setting G.

A hold command is generated by the sequencer, which waits in state 1 until the CPU responds with HOLDA, verifying that it has stopped instruction processing and that its buses are 3-stated. HOLDA resets G and sets H, advancing the sequencer to state 2. The G flipflop then toggles, causing the sequencer to cycle between states 2 and 3. H remains set during both states but tests the content of the data counter during state 3 to determine if data transmission is complete ($\overline{END}$ expression false). $\overline{END}$ false also causes the H flipflop to reset at the completion of state 3, returning the sequencer to state 0 and releasing the hold command to the CPU.

Block Length Control*

Remainder	$\overline{END}$ Expression
1	$Q_F + Q_E$
2	$Q_F + Q_E + Q_A$
3	$Q_F + Q_E + Q_B$
4	$Q_F + Q_E + Q_B + Q_A$
5	$Q_F + Q_E + Q_C$
6	$Q_F + Q_E + Q_C + Q_A$
7	$Q_F + Q_E + Q_C + Q_B$
8	$Q_F + Q_E + Q_C + Q_B + Q_A$
9	$Q_F + Q_E + Q_D$
10	$Q_F + Q_E + Q_D + Q_A$
11	$Q_F + Q_E + Q_D + Q_B$
12	$Q_F + Q_E + Q_D + Q_B + Q_A$
13	$Q_F + Q_E + Q_D + Q_C$
14	$Q_F + Q_E + Q_D + Q_C + Q_A$
15	$Q_F + Q_E + Q_D + Q_C + Q_B$
0	$Q_F + Q_E + Q_D + Q_C + Q_B + Q_A$

*Required to complete equation H in Fig 3, $\overline{END}$ expressions are listed by modulo 16 remainder of data word count. Counter F is not required for block sizes less than 33. Counter E is not required for block sizes less than 17. Single word transmission is a trivial solution

Implementation

System parameters were developed for use with the model 8085 microprocessor, the 2114 bidirectional RAM, and the 25LS2569 binary up/down counter. Attributes of these components that help meet the design goals are (a) availability of the CPU interface signals reset, clock out, hold, and hold

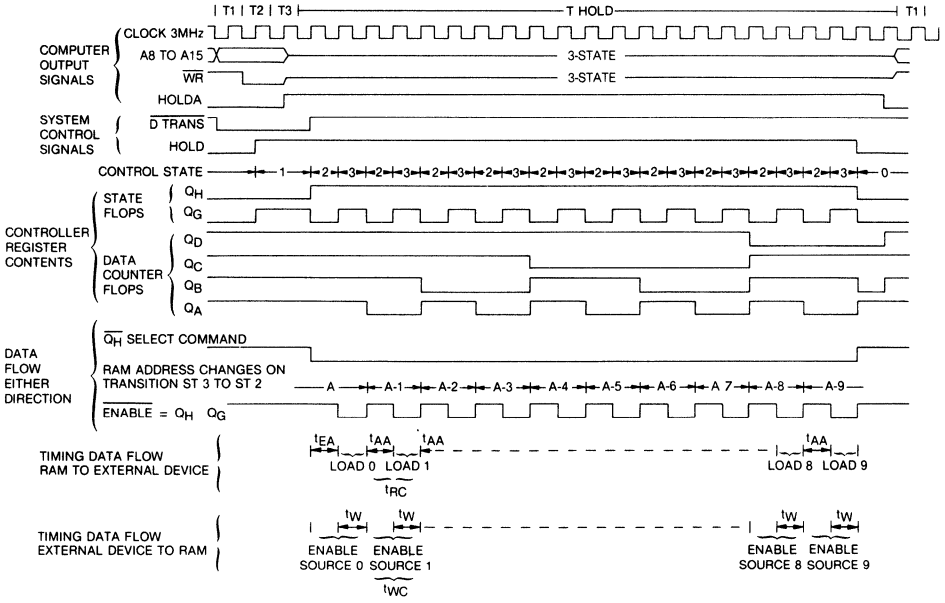


Fig 4 8085 timing and interface signals. Register contents reflect state controller interaction with computer commands. Control states reflect state flipflop outputs. Signals common to transfer in either direction are grouped. Current RAM address starts at A and decreases during successive cycles. Memory timing, shown for data flow both from and to RAM, relates to RAM timing requirements for Fig 1 and Fig 2, respectively

acknowledge; (b) the RAM's high density, 4-bit data word, bidirectional data bus, and simplified dual control line interface; and (c) the counter synchronized load, increment, and decrement controls, and also its 3-state output and cascading capabilities. Additionally, the response and operating speeds of the RAM and counter are compatible with the microprocessor clock.

Figs 1, 2, and 4 show the configuration and timing relationships achieved by using these components. Microprocessor choice establishes the system clock rate and the method of passing control between the microprocessor and the data controller. The method of implementing the data controller, which meets the dual requirements of simplicity and high throughput, restricts RAM selection. In Figs 1 and 2, t_{RC} and t_{WC} cannot exceed two clock periods. When moving data to external devices, the devices must be able to function within the constraints of t_{AA} and t_{EA}. When moving data to the CPU memory, its write requirements, t_W and t_{DW}, must be less than one clock period. Because of the controller structure, t_{DH} and t_{WR} must both be zero, a restriction satisfied by many RAMs currently available.

Adaptation

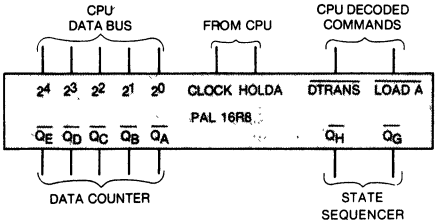
Distributed data port servicing is the primary functional requirement of the data controller, and the specific number of ports is a parameter written into the Boolean expressions implemented by a member of the PAL family. The inherent flexibility arising from implementation of many of the logic functions by easily modified Boolean expressions can be demonstrated by extending the application requirements to include program control of the data block length. Now, the design objectives are to incorporate this feature with minimal hardware reconfiguration and to retain all the operational attributes of the original DMA controller.

Although the state sequencer remains conceptually unchanged, its interaction with the data counter must be modified to implement a variable block length. The original concept has a fixed data block length, with the data counter starting at all 1s and decrementing to a predetermined end count. Its advantages are that the CPU interface is minimized and that all data ports are serviced during each DMA cycle. To implement the variable block length feature, the CPU loads the data counter with the actual count; then during data transfer, the state sequencer decrements the count to 0, the end count that terminates the operation.

It is desirable to implement the state sequencer and data counter on a single programmable chip while maximizing the data counter length. To achieve this, a 16R8 programmable logic chip replaces the 16R6 used previously. This change increases the number of registers by two at the expense of the two gated outputs. The chip configuration now offers eight data inputs; eight D registers, each with an inverting output; a register clock input; and a 3-state control input.

Of the eight registers, two are required for the state sequencer, allowing a maximum block length of 64 in a single-chip implementation. However, three control lines are required: $\overline{HOLDA}$ and $\overline{DTRANS}$, as in the original concept, plus a software generated $\overline{LOADA}$, which transfers the content of the data bus to the data registers. Additionally, the $\overline{LOADA}$ command must be used to initialize the state counter (Fig 5). With this approach, five inputs remain for the data bus, limiting the block size to 32 words.

An alternative configuration using only three data bits allows loading of the six flipflops in two instruction cycles



DATA COUNTERS

$Q_E = \overline{LOAD\ A} \cdot 2^4 + Q_E(\overline{Q_H} + Q_H \cdot \overline{Q_G}) + Q_H \cdot Q_G \cdot Q_E(Q_D + Q_C + Q_B + Q_A)$

$Q_D = \overline{LOAD\ A} \cdot 2^3 + Q_D(\overline{Q_H} + Q_H \cdot \overline{Q_G}) + Q_H \cdot Q_G[Q_D(Q_C + Q_B + Q_A) + \overline{Q_D} \cdot \overline{Q_C} \cdot \overline{Q_B} \cdot \overline{Q_A}]$

$Q_C = \overline{LOAD\ A} \cdot 2^2 + Q_C(\overline{Q_H} + Q_H \cdot \overline{Q_G}) + Q_H \cdot Q_G[Q_C(Q_B + Q_A) + \overline{Q_C} \cdot \overline{Q_B} \cdot \overline{Q_A}]$

$Q_B = \overline{LOAD\ A} \cdot 2^1 + Q_B(\overline{Q_H} + Q_H \cdot \overline{Q_G}) + Q_H \cdot Q_G(Q_B \cdot Q_A + \overline{Q_B} \cdot \overline{Q_A})$

$Q_A = \overline{LOAD\ A} \cdot 2^0 + Q_A(\overline{Q_H} + Q_H \cdot \overline{Q_G}) + Q_H \cdot Q_G \cdot \overline{Q_A}$

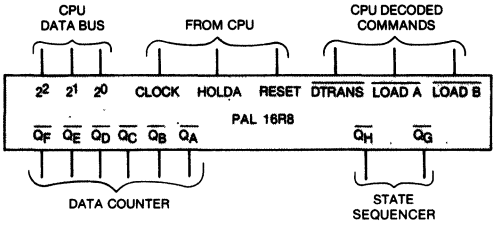
STATE CONTROLLER

$Q_G = LOAD\ A(\overline{Q_H} \cdot \overline{Q_G} \cdot \overline{DTRANS} + \overline{Q_H} \cdot Q_G \cdot \overline{HOLDA} + Q_H \cdot \overline{Q_G})$

$Q_H = LOAD\ A[\overline{Q_H} \cdot Q_G \cdot HOLDA + Q_H \cdot \overline{Q_G} + Q_H \cdot Q_G(Q_E + Q_D + Q_C + Q_B + Q_A)]$

Fig 5 Adding program control over block length. Hardware configuration and Boolean expressions are modified to implement program control of data block length. Data input limitation prevents use of one register, restricting block length to 32 words

8



DATA COUNTERS

$$Q_F = \overline{LOAD\ B} \cdot 2^2 + Q_F(\overline{Q_H} + Q_H \cdot \overline{Q_G}) + Q_H \cdot Q_G \cdot Q_F(Q_E + Q_D + Q_C + Q_B + Q_A)$$

$$Q_E = \overline{LOAD\ B} \cdot 2^1 + Q_E(\overline{Q_H} + Q_H \cdot \overline{Q_G}) + Q_H \cdot Q_G[\ Q_E(Q_D + Q_C + Q_B + Q_A) + \overline{Q_E} \cdot \overline{Q_D} \cdot \overline{Q_C} \cdot \overline{Q_B} \cdot \overline{Q_A}]$$

$$Q_D = \overline{LOAD\ B} \cdot 2^0 + Q_D(\overline{Q_H} + Q_H \cdot \overline{Q_G}) + Q_H \cdot Q_G[Q_D(Q_C + Q_B + Q_A) + \overline{Q_D} \cdot \overline{Q_C} \cdot \overline{Q_B} \cdot \overline{Q_A}]$$

$$Q_C = \overline{LOAD\ A} \cdot 2^2 + Q_C(\overline{Q_H} + Q_H \cdot \overline{Q_G}) + Q_H \cdot Q_G[Q_C(Q_B + Q_A) + \overline{Q_C} \cdot \overline{Q_B} \cdot \overline{Q_A}]$$

$$Q_B = \overline{LOAD\ A} \cdot 2^1 + Q_B(\overline{Q_H} + Q_H \cdot \overline{Q_G}) + Q_H \cdot Q_G(Q_B \cdot Q_A + \overline{Q_B} \cdot \overline{Q_A})$$

$$Q_A = \overline{LOAD\ A} \cdot 2^0 + Q_A(\overline{Q_H} + Q_H \cdot \overline{Q_G}) + Q_H \cdot Q_G \cdot \overline{Q_A}$$

STATE CONTROLLER

$$Q_G = \overline{RESET}(\overline{Q_H} \cdot \overline{Q_G} \cdot \overline{DTRANS} + \overline{Q_H} \cdot Q_G \cdot \overline{HOLDA} + Q_H \cdot \overline{Q_G})$$

$$Q_H = \overline{RESET}[\overline{Q_H} \cdot Q_G \cdot HOLDA + Q_H \cdot \overline{Q_G} + Q_H \cdot Q_G(Q_F + Q_E + Q_D + Q_C + Q_B + Q_A)]$$

Fig 6 Extending block length. Additional modification increases block length to 64 words by using all internal registers. All capabilities of previous configuration have been maintained across both modifications

by implementing a second load command, $\overline{LOADB}$. The DMA controller now handles block lengths as long as 64 data words, and the RESET command can be used to initialize the state counter as in the original concept (Fig 6).

Summary

Generally, a design approach and its means of implementation should be complementary. Specifically, the controller design objectives are met by utilizing a state sequencer, and implementation compatibility is assured by such features of semicustom logic as multiple storage elements with internal feedback paths, dense gating arrays, and interconnection flexibility.

The process of creating a state sequencer to solve a design problem results in a series of Boolean expressions that define the controller capability and the gate array that must be effected. Several partially dedicated gating and register arrays constitute the semicustom logic family. After the most suitable array configuration is selected, the interconnections, as expressed by Boolean statements, are completed to implement the design. This creates a unique logic pattern whose dense gating contributes to chip count minimization. Design variations can be accommodated by restructuring the gating arrays with minimal hardware reconfiguration. Thus, the system can be tailored to the original operating requirements and yet be easily adapted if modifications are desired.

Bibliography

"MCS-85 Users Manual," Intel Corp, Santa Clara, Calif, 1978

J. Nissim, "DMA Controller Capitalizes on Clock Cycles to Bypass CPU," *Computer Design*, Jan 1978, pp 117-124

"Programmable Array Logic Handbook," Monolithic Memories Inc, Sunnyvale, Calif, 1978

Alan W. Bentley is a design specialist, technically responsible for the development of a high speed data processing and display system at Cubic Corp. He is also an instructor in the Engineering Department of San Diego Community College. He has a BS degree in electrical engineering from Tufts University, an MBA from United States International University, and an MA from San Diego State University.

FPLA ARBITER CONCEPT ADAPTS TO APPLICATION NEEDS

Field programmable logic implements efficient, easily customized arbiter whose versatile Boolean statement format meets numerous system requirements

Alan W. Bentley Cubic Corporation
9333 Balboa Ave, San Diego, CA 92123

Today's trends toward shared resources, multiprocessing, and decentralized, bus oriented system organization underscore the demand for arbiters that work independently and issue grants according to predetermined algorithms. Functionally, a requirement for stable processing of system requests is created when several processors use a common device and each requests service asynchronously. It is the arbiter's task to issue grants for sequential access in accordance with a preestablished algorithm. By using a versatile Boolean statement format to express its algorithms, an arbiter can be customized to interface with single-array, multiple-array, and hybrid logic configurations. Designing arbiters to be flexible in implementing algorithms can help to meet the demand for cost-effective data processing systems and bring heretofore expensive, hence scarce, devices into wider use.

Arbiter Interface

From the system's perspective of the interface, when a system is ready for service from the device, it raises a request line. The arbiter responds by issuing a grant to the system that clears it to conduct transactions with the device. Upon completion of service, the system drops its request line. The arbiter, in turn, cancels the grant and is then free to issue a grant to the next pending request.

Stable system requests are necessary because all requests and grants are evaluated each clock period. These requests are generated asynchronously and are synchronized through a set of input latches. If a 2-phase balanced clock is used, with requests synchronized on phase 0 and grants issued or released on phase 1, the maximum time from a system request for service until synchronization of that request is one clock period. Minimum time from completion of the transaction to release of the synchronized request is the system's internal delay, as it drops the request line, plus the setup time of the synchronizing latches. At the completion of a transaction, there is a half-clock period, the time from phase 0 to phase 1, when a unique signal combination exists—a grant issued to a system that does not have a synchronized request. During this half-period, all other synchronized requests are examined and, according to the servicing algorithm, the recipient of the next sequential grant is determined. Then, on phase 1, the existing (unsolicited) grant is canceled and the next grant issued.

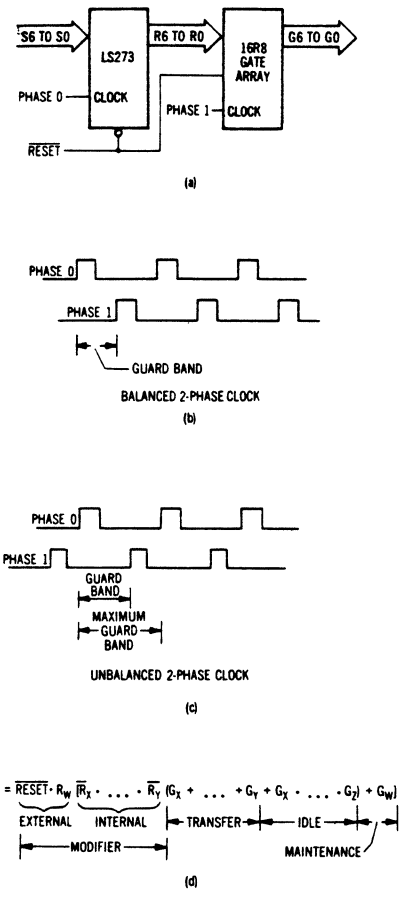


Fig 1 Arbiter circuitry, timing, and grant expression format. In (a), S6 to S0 are system requests. Latch outputs R6 to R0 are synchronized requests that arbiter algorithm converts to system grants, G6 to G0. Unbalanced clock phases increase guard band (b and c), with maximum guard band needing only one phase. Logic equation for algorithm (d) combines four subexpressions

The sum of the requesting system's internal delay in dropping its request line, the synchronizing latch setup time, and one-half the clock period, act as a guard band to ensure separation of systems interacting with the device. Arbiter clock frequency, therefore, is a major factor in determining guard band time. For a given arbiter clock frequency, the guard band can be increased by unbalancing the clock, increasing the time between phase 0 and phase 1 to more than half the clock period. The limiting case occurs when the time between phase 0 and phase 1 is increased until it equals the clock period. This makes phase 0 and phase 1 coincident, requires only a single-phase clock, and maximizes the guard band for a given clock frequency. The limiting case relies on the propagation delay of the latch flipflops and the setup time of the array logic grant flipflops to ensure a delay of one clock period between release of the synchronized request and the associated grant. Start of the guard band signifies completion of the current transaction, and guard band time is used to determine the next grant recipient; then, at the start of the next clock period, the current grant will be canceled, and, if one or more requests are pending, the next grant issued.

Octal D flipflops (LS273) are used to synchronize the input requests, and a programmable gate array (16R8) is used to evaluate the requests and issue grants. In the 2-phase clock, phase 0 synchronizes requests and phase 1 issues grants (Fig 1). Both the octal flipflops and the logic array are 20-pin packages. In addition to the clock and 3-state control inputs, the logic array has 8 data inputs and 8 D flipflop outputs. The Q output of each flipflop is inverted and buffered, and is a 3-state output; the $\overline{Q}$ output is returned to the array internally, where the 8 flipflop settings and the 8 data inputs are complemented and are available in both original and complement form at 8-input OR gates. There are eight OR gates, each forming the D input for one of the eight flipflops.

By programming through fusable links, the set condition of each flipflop is established from the eight data inputs, the current state of the eight flipflops, and the complements of both. Each flipflop's set condition can be represented as a Boolean expression of up to eight ORed statements. Each statement is created by ANDing terms drawn from the data inputs, the flipflop status, and their complements. Limiting parameters of this configuration are the 8 data inputs and the fixed 8-term OR gate at the D input of each flipflop. Boolean expressions implemented through the programmable fuses must be true when the flipflop is to be set or maintained in the set state, and false when the flipflop is to be reset or to remain reset.

Single-Array Configuration

Any of several servicing algorithms, expressed as Boolean statements, can be implemented with a single logic array that will process service requests for up to seven systems. The first two algorithms represent the organizational extremes. "Priority service" algorithm has a strict priority ranking from R6 to R0: a grant will be issued only if no higher priority system has a pending request. The other extreme is the "polled service" algorithm; here, all systems place equal demand on the device and are serviced through a rotating, or "round-robin," method. In this case, when one transaction has been completed, the following grant is issued in response to the next ranking request (eg, next lower number). Circular continuity is maintained by having R6 follow R0 in the granting sequence. Between the extremes, algorithms 3 and 4 represent hybrid organizations. The "executive and polled service" algorithm allows a single, high priority executive system (R6), with the remaining systems (R5 to R0) equal and polled in a

ALGORITHM 1: PRIORITY SERVICE

$$G6 = \overline{RESET} \cdot R6\,[G5 \cdot \overline{R5} + G4 \cdot \overline{R4} + G3 \cdot \overline{R3} + G2 \cdot \overline{R2} + G1 \cdot \overline{R1} + G0 \cdot \overline{R0} + \overline{G5} \cdot \overline{G4} \cdot \overline{G3} \cdot \overline{G2} \cdot \overline{G1} \cdot \overline{G0} + G6]$$

$$G5 = \overline{RESET} \cdot R5\,[\overline{R6}\,(G6 + G4 \cdot \overline{R4} + G3 \cdot \overline{R3} + G2 \cdot \overline{R2} + G1 \cdot \overline{R1} + G0 \cdot \overline{R0} + \overline{G6} \cdot \overline{G4} \cdot \overline{G3} \cdot \overline{G2} \cdot \overline{G1} \cdot \overline{G0}) + G5]$$

$$G4 = \overline{RESET} \cdot R4\,[\overline{R6} \cdot \overline{R5}\,(G6 + G5 + G3 \cdot \overline{R3} + G2 \cdot \overline{R2} + G1 \cdot \overline{R1} + G0 \cdot \overline{R0} + \overline{G6} \cdot \overline{G5} \cdot \overline{G3} \cdot \overline{G2} \cdot \overline{G1} \cdot \overline{G0}) + G4]$$

$$G3 = \overline{RESET} \cdot R3\,[\overline{R6} \cdot \overline{R5} \cdot \overline{R4}\,(G6 + G5 + G4 + G2 \cdot \overline{R2} + G1 \cdot \overline{R1} + G0 \cdot \overline{R0} + \overline{G6} \cdot \overline{G5} \cdot \overline{G4} \cdot \overline{G2} \cdot \overline{G1} \cdot \overline{G0}) + G3]$$

$$G2 = \overline{RESET} \cdot R2\,[\overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3}\,(G6 + G5 + G4 + G3 + G1 \cdot \overline{R1} + G0 \cdot \overline{R0} + \overline{G6} \cdot \overline{G5} \cdot \overline{G4} \cdot \overline{G3} \cdot \overline{G1} \cdot \overline{G0}) + G2]$$

$$G1 = \overline{RESET} \cdot R1\,[\overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2}\,(G6 + G5 + G4 + G3 + G2 + G0 \cdot \overline{R0} + \overline{G6} \cdot \overline{G5} \cdot \overline{G4} \cdot \overline{G3} \cdot \overline{G2} \cdot \overline{G0}) + G1]$$

$$G0 = \overline{RESET} \cdot R0\,[\overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1}\,(G6 + G5 + G4 + G3 + G2 + G1 + \overline{G6} \cdot \overline{G5} \cdot \overline{G4} \cdot \overline{G3} \cdot \overline{G2} \cdot \overline{G1}) + G0]$$

ALGORITHM 2: POLLED SERVICE

$$G6 = \overline{RESET} \cdot R6\,[\overline{R0}\,(G0 + G1 \cdot \overline{R1} + G2 \cdot \overline{R2} \cdot \overline{R1} + G3 \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} + G4 \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} + G5 \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1}) + \overline{G5} \cdot \overline{G4} \cdot \overline{G3} \cdot \overline{G2} \cdot \overline{G1} \cdot \overline{G0} + G6]$$

$$G5 = \overline{RESET} \cdot R5\,[\overline{R6}\,(G6 + G0 \cdot \overline{R0} + G1 \cdot \overline{R1} \cdot \overline{R0} + G2 \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0} + G3 \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0} + G4 \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0} + \overline{G6} \cdot \overline{G4} \cdot \overline{G3} \cdot \overline{G2} \cdot \overline{G1} \cdot \overline{G0}) + G5]$$

$$G4 = \overline{RESET} \cdot R4\,[\overline{R5}\,(G5 + G6 \cdot \overline{R6} + G0 \cdot \overline{R0} \cdot \overline{R6} + G1 \cdot \overline{R1} \cdot \overline{R0} \cdot \overline{R6} + G2 \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0} \cdot \overline{R6} + G3 \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0} \cdot \overline{R6} + \overline{R6} \cdot \overline{G6} \cdot \overline{G5} \cdot \overline{G3} \cdot \overline{G2} \cdot \overline{G1} \cdot \overline{G0}) + G4]$$

$$G3 = \overline{RESET} \cdot R3\,[\overline{R4}\,(G4 + G5 \cdot \overline{R5} + G6 \cdot \overline{R6} \cdot \overline{R5} + G0 \cdot \overline{R0} \cdot \overline{R6} \cdot \overline{R5} + G1 \cdot \overline{R1} \cdot \overline{R0} \cdot \overline{R6} \cdot \overline{R5} + G2 \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0} \cdot \overline{R6} \cdot \overline{R5} + \overline{R6} \cdot \overline{R5} \cdot \overline{G6} \cdot \overline{G5} \cdot \overline{G4} \cdot \overline{G2} \cdot \overline{G1} \cdot \overline{G0}) + G3]$$

$$?2 = \overline{RESET} \cdot R2\,[\overline{R3}\,(G3 + G4 \cdot \overline{R4} + G5 \cdot \overline{R5} \cdot \overline{R4} + G6 \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} + G0 \cdot \overline{R0} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} + G1 \cdot \overline{R1} \cdot \overline{R0} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} + \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{G6} \cdot \overline{G5} \cdot \overline{G4} \cdot \overline{G3} \cdot \overline{G1} \cdot \overline{G0}) + G2]$$

$$G1 = \overline{RESET} \cdot R1\,[\overline{R2}\,(G2 + G3 \cdot \overline{R3} + G4 \cdot \overline{R4} \cdot \overline{R3} + G5 \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} + G6 \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} + G0 \cdot \overline{R0} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} + \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{G6} \cdot \overline{G5} \cdot \overline{G4} \cdot \overline{G3} \cdot \overline{G2} \cdot \overline{G0}) + G1]$$

$$G0 = \overline{RESET} \cdot R0\,[\overline{R1}\,(G1 + G2 \cdot \overline{R2} + G3 \cdot \overline{R3} \cdot \overline{R2} + G4 \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} + G5 \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} + G6 \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} + \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{G6} \cdot \overline{G5} \cdot \overline{G4} \cdot \overline{G3} \cdot \overline{G2} \cdot \overline{G1}) + G0]$$

ALGORITHM 3: EXECUTIVE AND POLLED SERVICE

$G6$ = SAME AS G6 EQUATION, ALGORITHM 1

$$G5 = \overline{RESET} \cdot R5\,[\overline{R6}\,(G6 + G0 \cdot \overline{R0} + G1 \cdot \overline{R1} \cdot \overline{R0} + G2 \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0} + G3 \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0} + G4 \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0} + \overline{G6} \cdot \overline{G4} \cdot \overline{G3} \cdot \overline{G2} \cdot \overline{G1} \cdot \overline{G0}) + G5]$$

$$G4 = \overline{RESET} \cdot R4\,[\overline{R6} \cdot \overline{R5}\,(G6 + G5 + G0 \cdot \overline{R0} + G1 \cdot \overline{R1} \cdot \overline{R0} + G2 \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0} + G3 \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0} + \overline{G6} \cdot \overline{G5} \cdot \overline{G3} \cdot \overline{G2} \cdot \overline{G1} \cdot \overline{G0}) + G4]$$

$$G3 = \overline{RESET} \cdot R3\,[\overline{R6} \cdot \overline{R4}\,(G6 + \overline{R5} \cdot G5 + G4 + G0 \cdot \overline{R0} \cdot \overline{R5} + G1 \cdot \overline{R1} \cdot \overline{R0} \cdot \overline{R5} + G2 \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0} \cdot \overline{R5} + \overline{R5} \cdot \overline{G6} \cdot \overline{G5} \cdot \overline{G4} \cdot \overline{G2} \cdot \overline{G1} \cdot \overline{G0}) + G3]$$

$$G2 = \overline{RESET} \cdot R2\,[\overline{R6} \cdot \overline{R3}\,(G6 + \overline{R5} \cdot \overline{R4} \cdot G5 + G3 + G4 \cdot \overline{R4} \cdot G5 \cdot \overline{R5} \cdot \overline{R4} + G0 \cdot \overline{R0} \cdot \overline{R5} \cdot \overline{R4} + G1 \cdot \overline{R1} \cdot \overline{R0} \cdot \overline{R5} \cdot \overline{R4} + \overline{R5} \cdot \overline{R4} \cdot \overline{G6} \cdot \overline{G5} \cdot \overline{G4} \cdot \overline{G3} \cdot \overline{G1} \cdot \overline{G0}) + G2]$$

$$G1 = \overline{RESET} \cdot R1\,[\overline{R6} \cdot \overline{R2}\,(G6 + \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot G5 + G2 + G3 \cdot \overline{R3} \cdot G4 \cdot \overline{R4} \cdot \overline{R3} + G5 \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} + G0 \cdot \overline{R0} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} + \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{G6} \cdot \overline{G5} \cdot \overline{G4} \cdot \overline{G3} \cdot \overline{G2} \cdot \overline{G0}) + G1]$$

$$G0 = \overline{RESET} \cdot R0\,[\overline{R6} \cdot \overline{R1}\,(G6 + \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot G1 + G2 \cdot \overline{R2} \cdot G3 \cdot \overline{R3} \cdot \overline{R2} + G4 \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} + G5 \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} + \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{G6} \cdot \overline{G5} \cdot \overline{G4} \cdot \overline{G3} \cdot \overline{G2} \cdot \overline{G1}) + G0]$$

ALGORITHM 4: DECLINING PRIORITY AND POLLED SERVICE

$G6$ = SAME AS G6 EQUATION, ALGORITHM 1

$G5$ = SAME AS G5 EQUATION, ALGORITHM 1

$G4$ = SAME AS G4 EQUATION, ALGORITHM 1

$$G3 = \overline{RESET} \cdot R3\,[\overline{R6} \cdot \overline{R5} \cdot \overline{R4}\,(G6 + G5 + G4 + G0 \cdot \overline{R0} + G1 \cdot \overline{R1} \cdot \overline{R0} + G2 \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0} + \overline{G6} \cdot \overline{G5} \cdot \overline{G4} \cdot \overline{G2} \cdot \overline{G1} \cdot \overline{G0}) + G3]$$

$$G2 = \overline{RESET} \cdot R2\,[\overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3}\,(G6 + G5 + G4 + G3 + G0 \cdot \overline{R0} + G1 \cdot \overline{R1} \cdot \overline{R0} + \overline{G6} \cdot \overline{G5} \cdot \overline{G4} \cdot \overline{G3} \cdot \overline{G1} \cdot \overline{G0}) + G2]$$

$$G1 = \overline{RESET} \cdot R1\,[\overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R2}\,(G6 \cdot \overline{R3} + G5 \cdot \overline{R3} + G4 \cdot \overline{R3} + G3 \cdot \overline{R3} + G2 + G0 \cdot \overline{R0} \cdot \overline{R3} + \overline{R3} \cdot \overline{G6} \cdot \overline{G5} \cdot \overline{G4} \cdot \overline{G3} \cdot \overline{G2} \cdot \overline{G0}) + G1]$$

$$G0 = \overline{RESET} \cdot R0\,[\overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R1}\,(G6 \cdot \overline{R3} \cdot \overline{R2} + G5 \cdot \overline{R3} \cdot \overline{R2} + G4 \cdot \overline{R3} \cdot \overline{R2} + G3 \cdot \overline{R3} \cdot \overline{R2} + G2 \cdot \overline{R2} + G1 + \overline{R3} \cdot \overline{R2} \cdot \overline{G6} \cdot \overline{G5} \cdot \overline{G4} \cdot \overline{G3} \cdot \overline{G2} \cdot \overline{G1}) + G0]$$

circular pattern by decreasing number (R5 follows R0). The other hybrid algorithm, "declining priority and polled service," services three systems in declining priority (R6 to R4), and allocates four systems the lowest priority; R3 to R0 have equal priority and are polled in a circular pattern by decreasing number (R3 follows R0).

In each configuration, the eight inputs consist of a RESET command and seven synchronized requests. Outputs are the seven corresponding grants, each the set output of a flipflop, buffered and inverted. When activated, the RESET command negates all expressions, resetting all grant flipflops. After a synchronized request is received, the appropriate expression becomes true, and the grant flipflop is set at the following clock edge. Upon completion of the transaction, the grant is released one clock period following the clock edge that terminates its synchronized request. This interval, when the grant is true and its synchronized request is false, denotes the completion of the service cycle. If one or more synchronized requests are pending, all expressions are evaluated, and, based on the servicing sequence expressed by the Boolean statements, the next grant is issued. If at the completion of a grant there are no pending requests, then all grant flipflops are reset and the device reverts to the idle state.

When the device is idle, all synchronized requests are scanned each clock period. Detection of a single synchronized request causes a grant to be issued at the following clock edge. If two or more requests are synchronized simultaneously, the expressions must provide a hierarchical method for issuing the initial grant. If the systems are organized from highest to lowest priority, the issuing of an initial grant follows the same seniority pattern. However, if all systems polled are of equal priority, the initial grant hierarchy must be established arbitrarily; in these examples, the grant hierarchy is based on highest to lowest sequence, systems R6 to R0. All these factors are apparent in the Boolean statements that implement the various grant algorithms. Each statement is the D input of a flipflop, and when a statement is true, the corresponding flipflop becomes (or remains) set at the following clock edge, issuing (or maintaining) a grant. As grants must be issued sequentially, the statements must be interlocked to prevent multiple flipflop settings.

The format set forth in the Boolean expressions of algorithms 1 through 4 [compare Fig 1(d)] is used when writing all statements. It consists of four sections: a modifier divided into an external and an internal portion, a transfer expression, an idle expression, and a maintenance expression. The external modifier, which operates on the entire contents of each statement, is the $\overline{\text{RESET}}$ term ANDed with the appropriate synchronized request. This establishes the basic requirement that RESET not be issued—and that the synchronized request be present—to consider whether to issue or to maintain a grant. The internal modifier operates on the transfer and idle expressions, and is the AND of synchronized requests that must be false for a specific grant to be issued. Since in a complete or partial priority structure, the highest numbered system (R6) always takes precedence, the internal modifier for the G6 expression is nonexistent.

The transfer expression identifies the completion of a transaction through the combination of a grant with its corresponding request. In conjunction with the modifiers, the transfer expression evaluates the pending requests to determine the next sequential grant. If one or several simultaneous requests are synchronized when grants are not active, the idle expression, in conjunction with the modifiers, determines which grant will be issued according to the established priority. The maintenance expression, the set output of the grant flipflop, maintains an established grant until its external modifier becomes false.

There are at least two other practical hybrid combinations: two prioritized systems with the remaining five polled, and four prioritized systems with the remaining three polled. Expressions for these two configurations can be derived from the four examples just discussed.

Multiple-Array Configuration

The single-array configuration handles up to seven systems with a broad selection of operation modes and is suitable for a configuration that has a limited number of systems with high rates of interaction with the device. When there is a large number of systems that have low rates of interaction individually, it is necessary to scan blocks of synchronized requests rapidly in order to minimize response time, thereby maximizing device

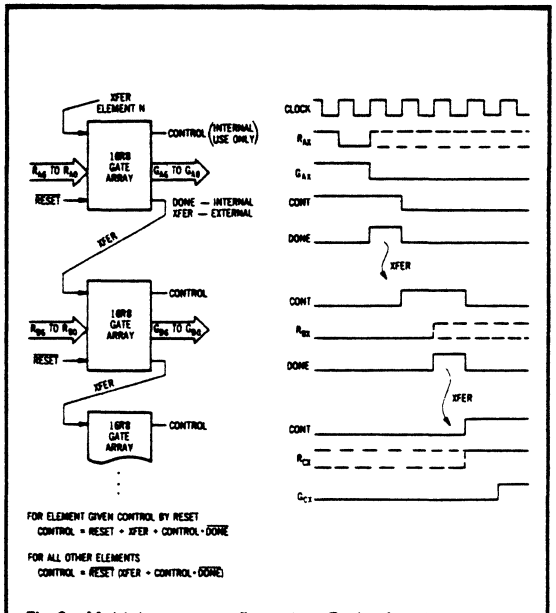


Fig 2 Multiple-array configuration. Each of n elements can service six systems arranged in circular (ie, polled or "round-robin") configuration. RESET issues control to one element. Then, as systems are serviced, DONE and CONTROL commands pass control function among elements as shown

utilization. This is accomplished by using the same 8-register programmable array device, with 6 of the registers issuing grants and the remaining 2 used to transfer the control and done commands. Devices thus organized become elements in a daisy chain configuration that is, in theory, infinitely expandable. Within the daisy chain, the element whose control flipflop is set is empowered to issue grants. Each element, while completing its last transaction, clears the grant flipflop, and, on the same clock edge, sets the done flipflop for one clock cycle. (See Fig 2.) Output of the done flipflop serves as an inhibit command within the element, and externally as a transfer (XFER) command to set the control flipflop of the next element.

Upon receipt of control, if one or more requests are pending, they are evaluated and a grant is issued at the next clock edge; if there are no requests, the clock edge instead sets the done flipflop to again transfer the control capability. Within an element, the same clock edge releases a completed grant and initiates a new grant, as was the case in the single-array configuration; among elements, there is a 2-period delay between succesive grants for sequencing of the done and control flipflops. Therefore, if the device is idle, an element that monitors six system requests is scanned each two clock cycles as the control function is sequenced through the elements.

Since this configuration can be expanded to accommodate servicing of numerous inputs, the elements are polled and equally weighted. However, within each element a protocol for responding to requests must be established; in these examples, priority is from the highest to the lowest number (R5 to R0). Algorithms 5 and 6 are variations of this configuration. In the "multiple-array polling" algorithm, the element retains control and services all requests in a declining polling sequence; control is relinquished only when there are no remaining requests. In the "multiple-array priority" algorithm, requests are serviced in a strict order of declining priority (ie, decreasing request numbers), and control is released in the absence of requests or upon completion of a device service period when no lower numbered systems have pending requests. In an application, these two configurations can be intermixed because the element interfaces are identical. The

ALGORITHM 5: MULTIPLE-ARRAY POLLING

$$G5 = \overline{RESET} \cdot \overline{DONE} \cdot CONT \cdot R5\,[\overline{R0}\,(G0 + G1 \cdot \overline{R1} + G2 \cdot \overline{R2} \cdot \overline{R1} + G3 \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} + G4 \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1}) + \overline{G4} \cdot \overline{G3} \cdot \overline{G2} \cdot \overline{G1} \cdot \overline{G0} + G5]$$

$$G4 = \overline{RESET} \cdot \overline{DONE} \cdot CONT \cdot R4\,[\overline{R5}\,(G5 + G0 \cdot \overline{R0} + G1 \cdot \overline{R1} \cdot \overline{R0} + G2 \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0} + G3 \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0} + \overline{G5} \cdot \overline{G3} \cdot \overline{G2} \cdot \overline{G1} \cdot \overline{G0}) + G4]$$

$$G3 = \overline{RESET} \cdot \overline{DONE} \cdot CONT \cdot R3\,[\overline{R4}\,(G4 + G5 \cdot \overline{R5} + G0 \cdot \overline{R0} \cdot \overline{R5} + G1 \cdot \overline{R1} \cdot \overline{R0} \cdot \overline{R5} + G2 \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0} \cdot \overline{R5} + \overline{R5} \cdot \overline{G5} \cdot \overline{G4} \cdot \overline{G2} \cdot \overline{G1} \cdot \overline{G0}) + G3]$$

$$G2 = \overline{RESET} \cdot \overline{DONE} \cdot CONT \cdot R2\,[\overline{R3}\,(G3 + G4 \cdot \overline{R4} + G5 \cdot \overline{R5} \cdot \overline{R4} + G0 \cdot \overline{R0} \cdot \overline{R5} \cdot \overline{R4} + G1 \cdot \overline{R1} \cdot \overline{R0} \cdot \overline{R5} \cdot \overline{R4} + \overline{R5} \cdot \overline{R4} \cdot \overline{G5} \cdot \overline{G4} \cdot \overline{G3} \cdot \overline{G1} \cdot \overline{G0}) + G2]$$

$$G1 = \overline{RESET} \cdot \overline{DONE} \cdot CONT \cdot R1\,[\overline{R2}\,(G2 + G3 \cdot \overline{R3} + G4 \cdot \overline{R4} \cdot \overline{R3} + G5 \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} + G0 \cdot \overline{R0} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} + \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{G5} \cdot \overline{G4} \cdot \overline{G3} \cdot \overline{G2} \cdot \overline{G0}) + G1]$$

$$G0 = \overline{RESET} \cdot \overline{DONE} \cdot CONT \cdot R0\,[\overline{R1}\,(G1 + G2 \cdot \overline{R2} + G3 \cdot \overline{R3} \cdot \overline{R2} + G4 \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} + G5 \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} + \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{G5} \cdot \overline{G4} \cdot \overline{G3} \cdot \overline{G2} \cdot \overline{G1}) + G0]$$

$$\overline{DONE} = \overline{RESET} \cdot CONT \cdot \overline{DONE} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$$

CONT = CONTROL (SEE FIG 2)

ALGORITHM 6: MULTIPLE-ARRAY PRIORITY

$$G5 = \overline{RESET} \cdot \overline{DONE} \cdot CONT \cdot \overline{G4} \cdot \overline{G3} \cdot \overline{G2} \cdot \overline{G1} \cdot \overline{G0} \cdot R5$$

$$G4 = \overline{RESET} \cdot \overline{DONE} \cdot CONT \cdot \overline{G3} \cdot \overline{G2} \cdot \overline{G1} \cdot \overline{G0} \cdot R4\,[\overline{R5} + G4]$$

$$G3 = \overline{RESET} \cdot \overline{DONE} \cdot CONT \cdot \overline{G2} \cdot \overline{G1} \cdot \overline{G0} \cdot R3\,[\overline{R4}\,(\overline{R5} + G4) + G3]$$

$$G2 = \overline{RESET} \cdot \overline{DONE} \cdot CONT \cdot \overline{G1} \cdot \overline{G0} \cdot R2\,[\overline{R3}\,(\overline{R5} \cdot \overline{R4} + G4 \cdot \overline{R4} + G3) + G2]$$

$$G1 = \overline{RESET} \cdot \overline{DONE} \cdot CONT \cdot \overline{G0} \cdot R1\,[\overline{R2}\,(\overline{R5} \cdot \overline{R4} \cdot \overline{R3} + G4 \cdot \overline{R4} \cdot \overline{R3} + G3 \cdot \overline{R3} + G2) + G1]$$

$$G0 = \overline{RESET} \cdot \overline{DONE} \cdot CONT \cdot R0\,[\overline{R1}\,(\overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} + G4 \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} + G3 \cdot \overline{R3} \cdot \overline{R2} + G2 \cdot \overline{R2} + G1) + G0]$$

$$\overline{DONE} = \overline{RESET} \cdot \overline{DONE} \cdot CONT \cdot \overline{R0}\,[\overline{R1}\,(\overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} + G4 \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} + G3 \cdot \overline{R3} \cdot \overline{R2} + G2 \cdot \overline{R2} + G1) + G0]$$

CONT = CONTROL (SEE FIG 2)

differences are within the elements—ie, the methods used when evaluating requests to issue grants and when generating the transfer command. Since only one element can have a set control flipflop and the associated authority to issue grants, one control flipflop must be set, and all others reset, during initialization. During implementation, the system start of poll (established by RESET) must be selected, and each element's control equation implemented accordingly. (See Fig 2.)

The Boolean statements in algorithms 5 and 6 are configured in the format shown in Fig 1(d). In algorithm 5, the external modifier has two additional terms, CONT and DONE, both establishing that the element has control. The remainder of the terms is similar in content to the terms in algorithm 2, and, with the exception of the idle expression, also similar in concept. Since an element does not retain control in the idle state, this term instead establishes the initial grant upon receipt of the control function. Subsequent grants are issued through the transfer expression, and when all requests are satisfied, the done flipflop is set for one period, initiating a transfer of control to the next element.

Hybrid Configuration

Single- and multiple-array configurations have limitations. The single-array concept services only seven systems, but offers flexibility in configuring the response patterns. Conversely, the multiple-array configuration can handle a large (theoretically unlimited) number of systems, but is not flexible in configuring response patterns because the control function must cycle through elements in a prescribed sequence. By restructuring the control lines, designers can interconnect elements in a hierarchical fashion, both increasing the number of systems serviced and providing service flexibility. An example that consists of two elements, each servicing six systems, is shown in Fig 3. The two elements are labeled "upper" and "lower" to denote their relative priorities. Each element has a control flipflop whose output is an input to the other element.

The control flipflop of the upper element is set whenever it has a synchronized request, and is a DEMAND for control. The control flipflop of the lower element is set when it receives a demand and does not have an active grant; it is a RELEASE of control. Thus, the upper element DEMANDs and receives control when it has a request. When all upper element requests are satisfied, the demand flipflop is reset and control shifts to the lower element by default. The control equations for the upper and lower elements are shown in Fig 3, along with the format for the grant equations and one sample grant equation for each element. Because the equations for the hybrid configuration are similar to those for the single-array configuration, the grant equations are modifications of those shown in algorithms 1 through 4. When in the idle state, the default location for control is in the lower element; therefore, the equation for the upper element does not require an idle section.

Efficient transfer of control and prevention of inadvertent simultaneous grant issuance during the transition sequence are primary considerations in arbiter

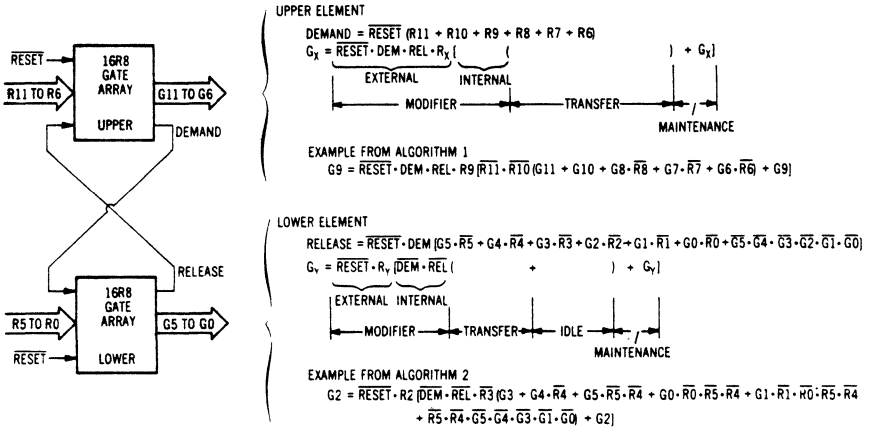


Fig 3 Hybrid configuration. Two elements have hierarchical structure. Upper element demands control when request is pending, thereby preventing lower element from issuing grants and gaining control when lower element becomes idle. Modified grant expression format and typical grant expression appear for each element. Release expression is also shown. Each element offers full flexibility shown in algorithms 1 to 4

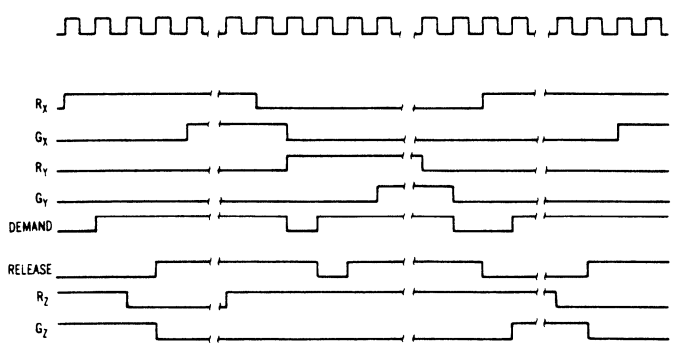


Fig 4 Hybrid configuration timing. Diagram shows timing relationship as control function passes between upper and lower elements. At left, control passes from lower element to upper element upon completion of G_Z. In second sector, upper element releases control and reestablishes demand during same clock period (adverse situation). Lower element with pending request must reinitiate release, causing 3-period delay between grants. In two right sectors, upper element releases control and reestablishes demand one clock period later. Lower element issues grant for its pending request and releases control upon completion of service

design. Control equations and modifier sections of the grant equations, respectively, fulfill these design requirements. Three transitional sequences for hybrid configuration are shown in the timing diagram. (See Fig 4.)

Summary

The increased desirability of sharing resources among several systems has increased the requirements for arbiters that can function independently. Such arbiters enhance the profitability of shared resource data processing systems. A versatile Boolean statement format can implement efficient control protocol and grant algorithms in field programmable logic. The flexibility and logic density of register arrays can facilitate efficient custom arbiter implementation.

Bibliography

K. Søe Højberg, "Queue Handling Arbiter Solves Shared Resource Conflicts," *Computer Design*, Nov 1979, pp 129-135

James Nadir and Bruce McCormick, "Bus Arbiter Streamlines Multiprocessor Design," *Computer Design*, June 1980, pp 103-109

Emil Petriu, "N-Channel Asynchronous Arbiter Resolves Resource Allocation Conflicts," *Computer Design*, Aug 1980, pp 126-132

About the Author:

Alan W. Bentley is a design specialist, responsible for the technical development of a high speed data processing and display system at Cubic Corporation. He is also an instructor in the Engineering Department of San Diego Community College. He has a BS degree in electrical engineering from Tufts University, an MBA from United States International University, and an MA from San Diego State University.

PAL stretches pulses for slower μP peripherals

The versatile programmable array logic (PAL) IC makes a simple one-chip pulse stretcher for microprocessor peripherals that have not caught up with the latest wave of high-speed processors and their shorter cycle times. The pulse stretcher uses up to six decoded function-selection addresses and puts out a synchronous stretched pulse for each address. It works well with an MC6809B; however, nearly all of the present-generation processors should work well with the circuit.

The diagram is a guide to programming the PAL on a conventional PROM programmer. For the PAL16R8 used in this example, the sum of the products is segmented into a fixed array that is stored in a D-type flip-flop on the rising edge of $\overline{E}$, the system clock. The registered outputs are then available as feedback elements to the input sums.

Longer output pulses are generated from the shorter system clock by taking advantage of the synchronous nature of the registers. The board-selection line serves as a master decoded address field for a particular device or series of devices. The six function-selection lines, $\overline{FS}_0$ through $\overline{FS}_5$, constitute the uniquely decoded addresses.

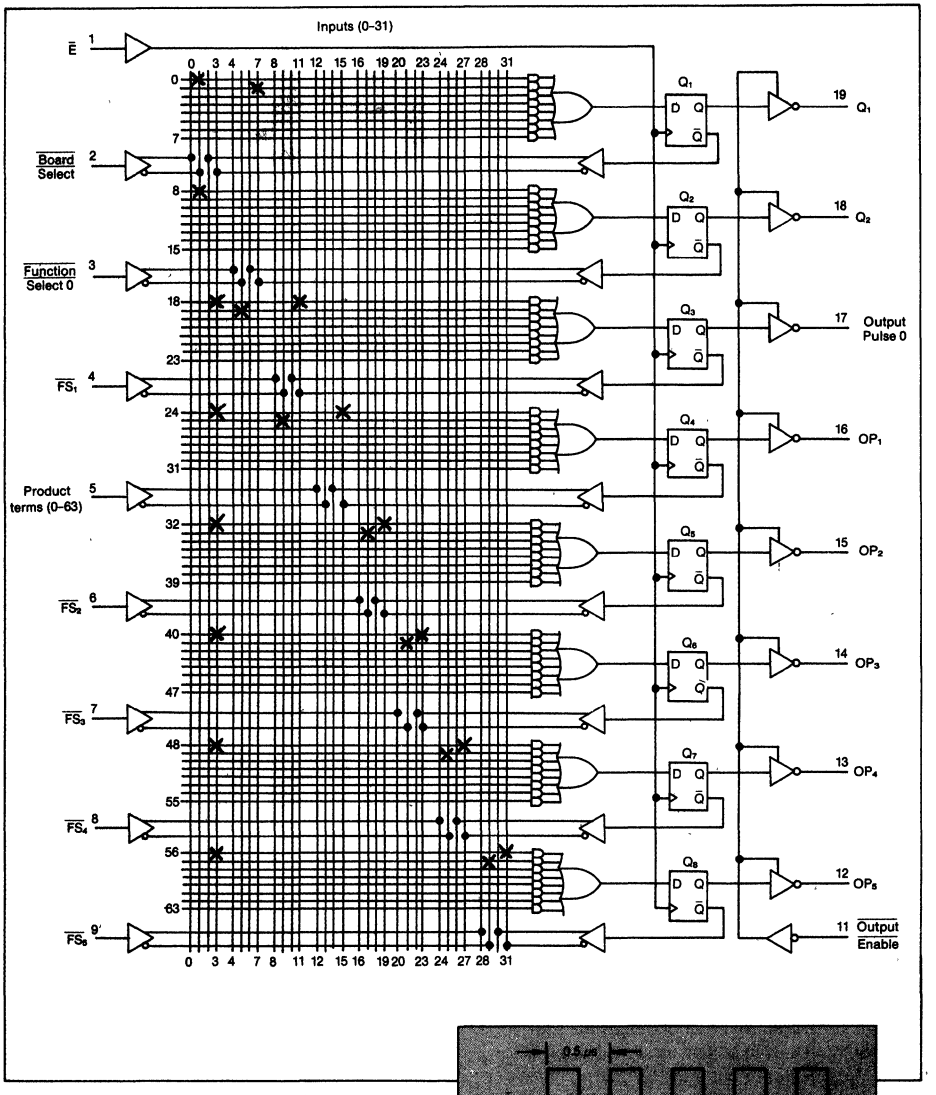
As shown in the timing diagram, the $\overline{E}$ clock synchronizes the system. Because the 6809's valid address period generally occurs during the low state of E, the board selection is conditioned by E. Therefore the rising edge of $\overline{E}$ clocks Q_1 and Q_2 as well as any flip-flop whose function-selection line is active. The next $\overline{E}$ positive edge resets Q_2, but since Q_1's input sum includes Q_2, Q_1 will not reset until the following $\overline{E}$ transition. As a result, Q_1's output is a pulse that is twice as wide as an $\overline{E}$ period.

Additional sections of the PAL can be concatonated to produce increasingly longer output pulses. Alternatively, the remaining six sections can be used as shown to implement individually decoded and stretched output pulses.

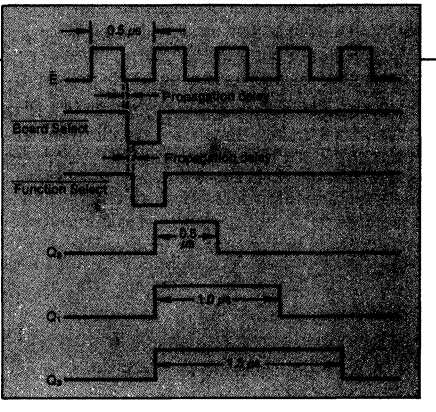
Assuming a 2-MHz processor, six 1.5-μs ($3 \times \overline{E}$ period) pulse stretchers are available in the PAL16R8. If desired, a single pulse can be stretched up to 4 μs in 0.5-μs increments. In that case, the master selection would be the same signal as the function selection. The output pulses would then be synchronous with the $\overline{E}$ clock, as well as adjustable simply by reprogramming the PAL. System timing pulses are then a precision function of the processor's clock and would be synchronized with it. The timing periods can be varied by changing the PAL firmware.

K. Russell Peterman, Senior Engineer, Electronic Design Department, Radian Corp., 8501 Mo-Pac Blvd., Austin, Texas 78766.

See next page for schematic.



The stretched pulses issuing from this programmable array logic IC simplify the work of a high-speed processor that must service slower peripherals. The PAL's fuse links are selectively burned on a covnentional PROM programmer. The × represents the intact fuse that is used to perform the logical AND function.



System Advantages of Programmable Logic

Danesh M. Tavana and Vincent J. Coli
NORTHCON/83

There are many advantages to programmable logic that make these devices an attractive alternative when designing digital systems. In this paper we will highlight these system advantages as well as present the methodology and software used when designing with these devices. The paper will close with a detailed design of a general purpose DMA Controller chip set which illustrates these advantages.

System Advantages

Today's semicustom circuits are commonly implemented using gate arrays, standard cells, and fuse programmable logic. A family of industry standard semicustom devices are not available in the form of fuse programmable logic arrays known as PALs (Programmable Array Logic). Currently, the PAL family includes 29 different generic types and offers a wide spectrum of speed, power, architecture, and package choices. Speed and power tradeoffs of PALs are available to satisfy the high speed bipolar applications as well as the power stingy, lower performance needs. The variety of architectures and industry standard pinouts (JEDEC approved) in 0.3 inch wide (Skinny Dip) packages, as well as second source availability through major integrated circuit manufacturers, make PALs an attractive semicustom solution for many designs.

The features and flexibilities offered to designers by the fuse programmable solution include fast design cycles, flexible PC board routing, large scale integration of standard TTL SSI/MSI functions, and a powerful software package capable of fault detection and functional testing. In today's competitive marketplace the "time to market" plays a critical role in product introduction. The design cycle is minimized when fuse programmable logic arrays are adopted since the PAL can be programmed and tested in a matter of minutes with the proper software and equipment. The first few hundred systems can be implemented with PALs for fast production turnaround and, upon an increased volume demand, the system manufacturer may find it advantageous to switch to HAL (Hard Array Logic), which is a metal mask option of the PAL, but lower in price for larger volumes. If future design alterations are necessary, the internal structure of the PAL can be changed while maintaining identical pinouts, thus being immune to PC board alterations. The flexible pinout configuration is also a tremendously powerful feature that could make the difference between single or multilayered PC boards. PALASM, or the PAL assembler software package, and a PAL or PROM programmer are the only tools needed to get a fully semicustom prototype programmed. The software incorporates a variety of fault and functional testing features to assure proper device operation. The interface to this software and the design methodology for PALs are subjects which need further attention.

Software Is The Key

PALASM, also referred to as a "silicon compiler", is the PAL assembler program. It is a high level language available for a wide range of computers which can translate boolean logic equations into a fuse plot. The fuse plot is made to be compatible with standard PAL or PROM programmers. The basic PAL architecture is a combinatorial matrix of AND and OR terms with the AND terms being fuse programmable. Various other architecture-like arrays of exclusive OR and registered outputs are also available. All of these architectures can be expressed in the form of boolean equation which PALASM will accept and convert to a fuse plot. The organization of the equations which will uniquely specify a PAL is referred to as the "PAL Design Specification". The PAL Design Specification is a formatted data file which must be linked with PALASM and compiled to obtain the fuse plot necessary to burn the PAL. The PAL Design Specification format, as shown in Figure 1, must include the generic part type on the first line and the symbolic pin names starting on the fifth line. The boolean equations followed by a function table (which can logically simulate the part) are next. The function table provides a testing vehicle to assure proper device operation. The next section is the optional device description used for documentation Another feature of PALASM is fault analysis to detect metal line failures and sticky fuses. PALASM is purposely formatted in this manner to force the designer into following a design methodology which simplifies the design cycle. The design methodology concept is an important aspect which is discussed next.

```
PAL16R6 <-- PAL PART NUMBER (MUST START ON LINE 1, COLUMN 1)      PAL DESIGN SPECIFICATION
PART NO XXX <-- USER'S PART NUMBER (LINE 2)                        AUTHOR'S NAME  DATE
DEVICE APPLICATION NAME <-- NAME OF DEVICE (LINE 3)
COMPANY, CITY, STATE <-- USER'S COMPANY NAME, CITY, STATE (LINE 4)
CLK  I0  D0 D1 D2 D3 D4 D5  I1  GND  |  <-- PIN LIST (MUST START ON LINE 5)
/OC RILO Q5 Q4 Q3 Q2 Q1 Q0 LIRO VCC  |      CONSISTS OF 20 (24) SYMBOLIC NAMES
                                          WHICH ARE CONSECUTIVELY ASSIGNED
                                          TO PINS 1 THRU 20 (24)

/Q0  := /I1*/I0*/Q0 + /I1* I0*/Q1 + I1*/I0*/LIRO + I1* I0*/D0
/Q1  := /I1*/I0*/Q1 + /I1* I0*/Q2 + I1*/I0*/Q0 + I1* I0*/D1
/Q2  := /I1*/I0*/Q2 + /I1* I0*/Q3 + I1*/I0*/Q1 + I1* I0*/D2
/Q3  := /I1*/I0*/Q3 + /I1* I0*/Q4 + I1*/I0*/Q2 + I1* I0*/D3
/Q4  := /I1*/I0*/Q4 + /I1* I0*/Q5 + I1*/I0*/Q3 + I1* I0*/D4
/Q5  := /I1*/I0*/Q5 + /I1* I0*/RILO + I1*/I0*/Q4 + I1* I0*/D5
IF(/I1*I0) /LIRO = /Q0              ;LEFT IN RIGHT OUT
IF(I1*/I0) /RILO = /Q5              ;RIGHT IN LEFT OUT
                                                          <-- EQUATIONS
        <-- CONDITIONAL THREE-STATE
FUNCTION TABLE <-- KEYWORD (MUST START IN COLUMN 1)

I1 I0 D5 D4 D3 D2 D1 D0 CLK /OC RILO LIRO Q5 Q4 Q3 Q2 Q1 Q0 <-- FUNCTION TABLE PIN LIST

;        D IN                          Q OUT                  |  OPTIONAL COMMENTS
;INST    D5---D0     CLK  /OC  RILO  LIRO  Q5---Q0   COMMENTS  |
-------------------------------------------------------------------------
  HH    LLLLLL       C    L    Z     Z     LLLLLL   LOAD ZEROS
  LL    XXXXXX       C    L    Z     Z     LLLLLL   HOLD
  HL    XXXXXX       C    L    L     H     LLLLLH   SHIFT LEFT IN A H
  HL    XXXXXX       C    L    L     L     LLLLHL   SHIFT LEFT IN A L
  HL    XXXXXX       C    L    L     L     LLLHLL   SHIFT LEFT IN A L
  HL    XXXXXX       C    L    L     L     LLHLLL   SHIFT LEFT IN A L
  HL    XXXXXX       C    L    L     L     LHLLLL   SHIFT LEFT IN A L
  HL    XXXXXX       C    L    H     L     HLLLLL   SHIFT LEFT IN A L
  HL    XXXXXX       C    L    L     L     LLLLLL   SHIFT LEFT IN A L
  HH    HHHHHH       C    L    Z     Z     HHHHHH   LOAD ONES
  LL    XXXXXX       C    L    Z     Z     HHHHHH   HOLD
  LL    XXXXXX       X    H    Z     Z     ZZZZZZ   TEST HI-Z
-------------------------------------------------------------------------
                                                          <-- DASHED LINE
                                                              (LENGTH OPTIONAL)
                                                          <-- FUNCTION TABLE VECTORS
                                                              ONE VECTOR PER
                                                              LINE FOLLOWED BY
                                                              OPTIONAL COMMENTS.
                                                              THE STATES (H, L, X, C, Z)
                                                              CORRESPOND TO THE
                                                              ORDER AND POLARITY
                                                              OF THE FUNCTION
                                                              TABLE PINLIST.
                                                          <-- DASHED LINE
                                                              (LENGTH OPTIONAL)

DESCRIPTION <-- KEYWORD, OPTIONAL IF FOLLOWING FUNCTION TABLE (MUST START COLUMN 1)
THIS PARAGRAPH DESCRIBES THE OPERATION AND APPLICATION OF THE DEVICE.
```

Figure 1

The Methodology

The design methodology is a key element in the total approach to fuse programmable logic arrays. The methodology suggests a simple and structured means of creating the prototype device. The basic procedures are: DESIGN, SIMULATE, BUILD, and TEST. These procedures are enforced when using PALASM in compiling boolean logic equations into silicon.

Most of today's systems, whether a board in a microcomputer, or a dedicated controller board, or a telecommunication's hardware design, contain many discrete SSI/MSI implementations. The philosophy behind semicustom logic arrays is to integrate all the discrete functions into one piece of silicon, thus, saving board space, gaining speed, and lowering manufacturing cost. The difficulty arises because a simple and cheap interface has not yet been available between the logic designer's implementation and the semicustom solution until now. All that is needed to fully define a semicustom PAL design is a set of boolean equations. Generally these equations must be expressed in the sum of products form with other possible variations existing depending on the particular internal architecture of the PAL used. The acceptable syntax and operators used in defining the boolean equations takes little effort to learn and is used in the data file (PAL Design Specifications) to obtain the fuse plot.

There are provisions made for a function table within the PAL Design Specifications to exercise the functionality of the device from the boolean equations. The function table is used by PALASM to simulate the table's vectors which result in a pass/fail response. This feature gives the designer an opportunity to debug his design before building it. Once all errors are detected and corrected the output of the simulated function table is a set of test vectors which are used to test the device. Thus, through PALASM the user can confidently design, simulate, and test his circuit.

To actually build the part, standard programmers can be used to accept the fuse pattern generated by PALASM. The programmer will selectively blow the required fuses and then verify the fuse array.

A Typical PAL Application

To demonstrate the design methodology and the interface to PALASM let's design a general purpose DMA controller using PALs. This controller is capable of high speed block transfers of data between the peripheral device and the main mamory. The block diagram of this system is shown in Figure 2.

The DMA controller is a five chip solution capable of high speed block transfer of data. Data from a peripheral device is read or written to memory in 64K blocks starting at a preprogrammed location. The starting address, the block length, and the instruction command are microprogrammed into the PALs prior to any transfers. Upon a DMA request from the peripheral, the controller makes a request to CPU for bus mastership. When the bus is released by the CPU by three-stating it's address, data, and control signals, the PAL chip sets gain control of the bus and execute the instruction command. At the end of instruction execution, a terminal count signal relinquishes the bus mastership to the CPU.

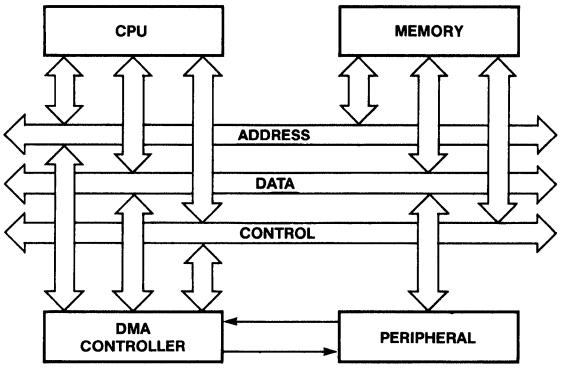


Figure 2. Block Diagram

The first step in defining this circuit is to specify the necessary input and output signals of the PALs. As shown in Figure 3 the four address counter and length counter PALs (U1-U4) are programmable, cascadable, up and down counters respectively, while the fifth PAL (U5) creates the necessary interface signals to any typical CPU environment The most suited generic PALs for implementing the counters and the control circuitry are PAL part types 20X8 and 20X10 respectively. The PAL Design Specification is included in the appendix. Upon successful execution of PALASM and it's data file, the fuse plot is generated and can be down-loaded to a standard programmer for burning the fuses in the PALs. The DMA operation is summarized here:

- CPU loads the lower byte of starting address in U1.
- CPU loads the upper byte of starting address in U2.
- CPU loads the lower byte of block length in U3.
- CPU loads the upper byte of block length in U4.
- CPU loads the instruction command in U5.
- The bus is controlled by the CPU and the DMA's bus interfaces are three-stated in normal mode. Upon assertion of DMAREQ the controller PAL (U5) will request the bus from the CPU.
- Upon granting the bus, the DMA becomes the bus master and the CPU's bus interfaces are three-stated. The DMACK output from the controller PAL is used to enable the DMA's address and control signals unto the bus when BUSACK is received from the CPU.
- The block transfer read or write instruction is executed starting at the specified address. Upon completion a terminal count signal which is the carry out of U4 will flag end of process.
- The peripheral must respond to the terminal count signal by removing the DMA request, at which point the CPU will relinquish bus mastership.

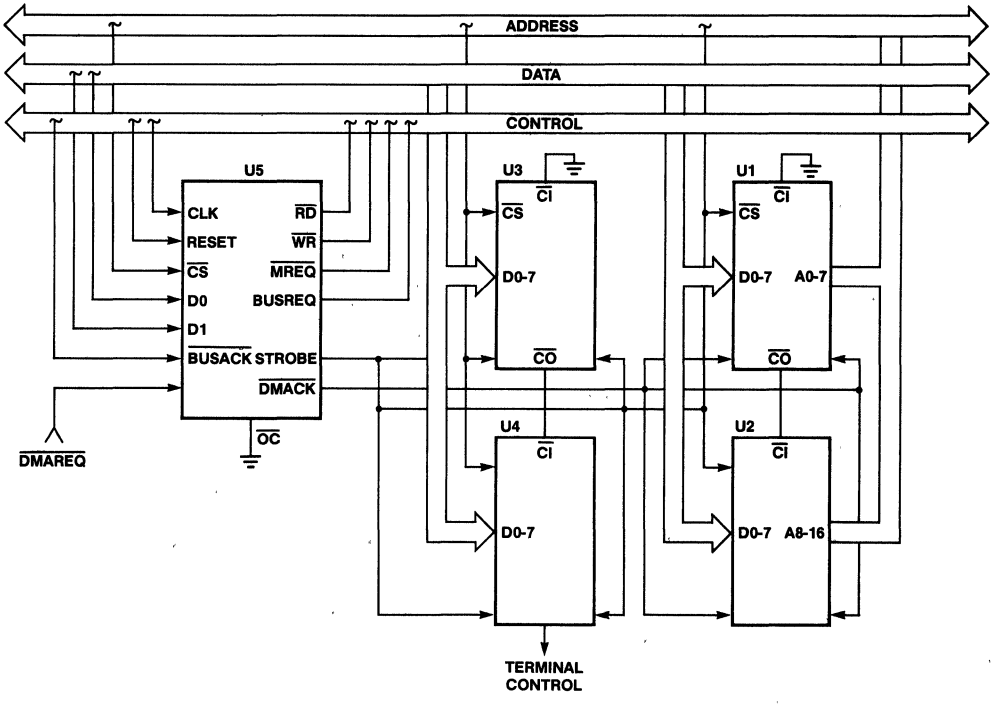


Figure 3. DMA Controller Logic Diagram

Conclusion

The semicustom fuse programmable logic devices are a cheap and easy alternative available to the system designer. The many features and benefits offered by programmable logic give the users extreme flexibility, yet, a structured and easy to learn methodology when designing a circuit. The "silicon compiler" software (PALASM), which is also capable of functional testing and fault detection, is the necessary tool to convert logic equations into a fuse plot.

References

(All references available from Monolithic Memories)

1. J. Birkner, V. Coli, *"PAL Programmable Array Logic Handbook"*, Monolithic Memories Inc., 1981.
2. *"PMSI PAL Medium Scale Integration"*, Monolithic Memories Inc., 1982.
3. A. Rappaport, *"Field Programmable Logic Arrays Bridge The Standard-IC/ Gate-Array GAP"*, EDN, January 6, 1983, pp. 59-74.
4. Peg Killmon, Joseph Aseo, *"Processor Technology"*, Computer Design December 1982, pp. 101-133.

```
PAL20X8                                          PAL DESIGN SPECIFICATIONS
DMA1.DAT                                         DANESH TAVANA
DMA ADDRESS COUNTER
MMI SUNNYVALE
   CLK /CI D0 D1 D2 D3 D4 D5 D6 D7 STROBE GND
/DMACK /CS A7 A6 A5 A4 A3 A2 A1 A0    /CO VCC


IF(VCC)  CO  =  A7* A6* A5* A4* A3 *A2 *A1 *A0          ;CARRY OUT TERM

        /A0 :=  CS*/D0                                  ;LOAD
             + /CS*/A0                                  ;HOLD
            :+:/CS* STROBE* CI                          ;INCREMENT

        /A1 :=  CS*/D1                                  ;LOAD
             + /CS*/A1                                  ;HOLD
            :+:/CS* STROBE* CI*A0                       ;INCREMENT

        /A2 :=  CS*/D2                                  ;LOAD
             + /CS*/A2                                  ;HOLD
            :+:/CS* STROBE* CI*A0*A1                    ;INCREMENT

        /A3 :=  CS*/D3                                  ;LOAD
             + /CS*/A3                                  ;HOLD
            :+:/CS* STROBE* CI*A0*A1*A2                 ;INCREMENT

        /A4 :=  CS*/D4                                  ;LOAD
             + /CS*/A4                                  ;HOLD
            :+:/CS* STROBE* CI*A0*A1*A2*A3              ;INCREMENT

        /A5 :=  CS*/D5                                  ;LOAD
             + /CS*/A5                                  ;HOLD
            :+:/CS* STROBE* CI*A0*A1*A2*A3*A4           ;INCREMENT

        /A6 :=  CS*/D6                                  ;LOAD
             + /CS*/A6                                  ;HOLD
            :+:/CS* STROBE* CI*A0*A1*A2*A3*A4*A5        ;INCREMENT

        /A7 :=  CS*/D7                                  ;LOAD
             + /CS*/A7                                  ;HOLD
            :+:/CS* STROBE* CI*A0*A1*A2*A3*A4*A5*A6     ;INCREMENT
```

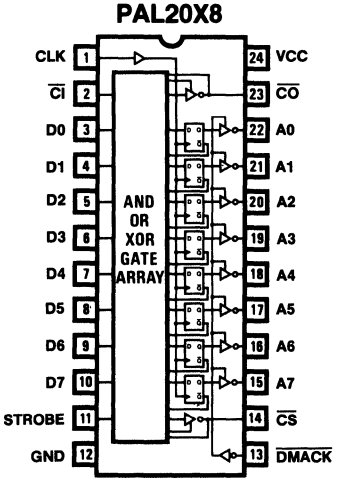
FUNCTION TABLE

CLK STROBE /CS /DMACK D7 D6 D5 D4 D3 D2 D1 D0
/CI A7 A6 A5 A4 A3 A2 A1 A0 /CO

; S /																				
; T D																				
; R M																				
;C O / A									/									/		
;L B C C	D	D	D	D	D	D	D	D	C	A	A	A	A	A	A	A	A	C		
;K E S K	7	6	5	4	3	2	1	0	I	7	6	5	4	3	2	1	0	O	DESCRIPTION	
C X L L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	H	LOAD ADDRESS	
C L H L	X	X	X	X	X	X	X	X	L	L	L	L	L	L	L	L	L	H	COUNT DISABLED BY STROBE	
C L H L	X	X	X	X	X	X	X	X	L	L	L	L	L	L	L	L	L	H	COUNT DISABLED BY STROBE	
C H H L	X	X	X	X	X	X	X	X	H	L	L	L	L	L	L	L	L	H	COUNT DISABLED BY CI	
C H H L	X	X	X	X	X	X	X	X	H	L	L	L	L	L	L	L	L	H	COUNT DISABLED BY CI	
C H H L	X	X	X	X	X	X	X	X	L	L	L	L	L	L	L	L	H	H	COUNT UP	
C H H L	X	X	X	X	X	X	X	X	L	L	L	L	L	L	L	H	L	H	COUNT UP	
C H H L	X	X	X	X	X	X	X	X	L	L	L	L	L	L	L	H	H	H	COUNT UP	
C H H L	X	X	X	X	X	X	X	X	L	L	L	L	L	L	H	L	L	H	COUNT UP	
C H H L	X	X	X	X	X	X	X	X	L	L	L	L	L	L	H	L	H	H	COUNT UP	
C X L L	H	H	H	H	H	H	H	H	X	H	H	H	H	H	H	H	H	L	LOAD ADDRESS/ TRIGGER CO	
C X X H	X	X	X	X	X	X	X	X	X	Z	Z	Z	Z	Z	Z	Z	Z	X	HI-Z	

DESCRIPTION

THIS PAL DESIGN SPECIFICATION IS AN ADDRESS COUNTER.



DMA ADDRESS COUNTER

```
 1 C000000000XX00LLLLLLLLH1
 2 C0XXXXXXXX0X01LLLLLLLLH1
 3 C0XXXXXXXX0X01LLLLLLLLH1
 4 C1XXXXXXXX1X01LLLLLLLLH1
 5 C1XXXXXXXX1X01LLLLLLLLH1
 6 C0XXXXXXXX1X01LLLLLLLHH1
 7 C0XXXXXXXX1X01LLLLLLHLH1
 8 C0XXXXXXXX1X01LLLLLLHHH1
 9 C0XXXXXXXX1X01LLLLLHLLH1
10 C0XXXXXXXX1X01LLLLLHLHH1
11 CX111111111XX00HHHHHHHHL1
12 CXXXXXXXXXXX1XZZZZZZZZZX1
```

PASS SIMULATION

DMA ADDRESS COUNTER

```
              11 1111 1111 2222 2222 2233 3333 3333
   0123 4567 8901 2345 6789 0123 4567 8901 2345 6789

 0 ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
 1 ---- --X- --X- --X- --X- --X- --X- --X- --X- ---- A7*A6*A5*A4*A3*A2*A1*A0

 8 ---- -X-- ---- ---- ---- ---- ---- ---- ---- ---X CS*/D0
 9 ---- ---X ---- ---- ---- ---- ---- ---- ---- --X- /CS*/A0
10 -X-- ---- ---- ---- ---- ---- ---- ---- ---- X-X- /CS*STROBE*CI

16 ---- ---- -X-- ---- ---- ---- ---- ---- ---- ---X CS*/D1
17 ---- ---- ---X ---- ---- ---- ---- ---- ---- --X- /CS*/A1
18 -X-- --X- ---- ---- ---- ---- ---- ---- ---- X-X- /CS*STROBE*CI*A0

24 ---- ---- ---- -X-- ---- ---- ---- ---- ---- ---X CS*/D2
25 ---- ---- ---- ---X ---- ---- ---- ---- ---- --X- /CS*/A2
26 -X-- --X- --X- ---- ---- ---- ---- ---- ---- X-X- /CS*STROBE*CI*A0*A1

32 ---- ---- ---- ---- -X-- ---- ---- ---- ---- ---X CS*/D3
33 ---- ---- ---- ---- ---X ---- ---- ---- ---- --X- /CS*/A3
34 -X-- --X- --X- --X- ---- ---- ---- ---- ---- X-X- /CS*STROBE*CI*A0*A1*A2

40 ---- ---- ---- ---- ---- -X-- ---- ---- ---- ---X CS*/D4
41 ---- ---- ---- ---- ---- ---X ---- ---- ---- --X- /CS*/A4
42 -X-- --X- --X- --X- --X- ---- ---- ---- ---- X-X- /CS*STROBE*CI*A0*A1*A2*-

48 ---- ---- ---- ---- ---- ---- -X-- ---- ---- ---X CS*/D5
49 ---- ---- ---- ---- ---- ---- ---X ---- ---- --X- /CS*/A5
50 -X-- --X- --X- --X- --X- --X- ---- ---- ---- X-X- /CS*STROBE*CI*A0*A1*A2*-

56 ---- ---- ---- ---- ---- ---- ---- -X-- ---- ---X CS*/D6
57 ---- ---- ---- ---- ---- ---- ---- ---X ---- --X- /CS*/A6
58 -X-- --X- --X- --X- --X- --X- --X- ---- ---- X-X- /CS*STROBE*CI*A0*A1*A2*-

64 ---- ---- ---- ---- ---- ---- ---- ---- -X-- ---X CS*/D7
65 ---- ---- ---- ---- ---- ---- ---- ---- ---X --X- /CS*/A7
66 -X-- --X- --X- --X- --X- --X- --X- --X- ---- X-X- /CS*STROBE*CI*A0*A1*A2*-
```

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

NUMBER OF FUSES BLOW = 948

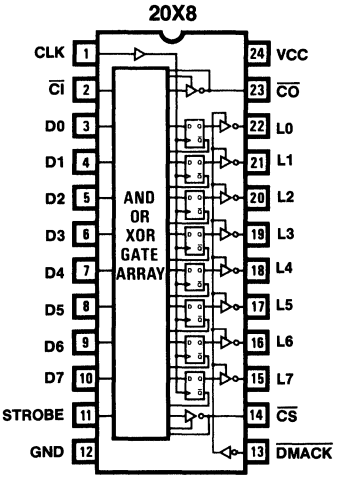
```
PAL20X8                                                PAL DESIGN SPECIFICATIONS
DMA2.DAT                                               DANESH TAVANA
DMA LENGTH COUNTER
MMI SUNNYVALE
   CLK /CI D0 D1 D2 D3 D4 D5 D6 D7 STROBE GND
/DMACK /CS L7 L6 L5 L4 L3 L2 L1 L0    /CO VCC


IF(VCC)  CO  = /L7*/L6*/L5*/L4*/L3*/L2*/L1*/L0                ;CARRY OUT TERM

        /L0 :=  CS*/D0                                        ;LOAD
             + /CS*/L0                                        ;HOLD
            :+:/CS* STROBE* CI                                ;INCREMENT

        /L1 :=  CS*/D1                                        ;LOAD
             + /CS*/L1                                        ;HOLD
            :+:/CS* STROBE* CI*/L0                            ;INCREMENT

        /L2 :=  CS*/D2                                        ;LOAD
             + /CS*/L2                                        ;HOLD
            :+:/CS* STROBE* CI*/L0*/L1                        ;INCREMENT

        /L3 :=  CS*/D3                                        ;LOAD
             + /CS*/L3                                        ;HOLD
            :+:/CS* STROBE* CI*/L0*/L1*/L2                    ;INCREMENT

        /L4 :=  CS*/D4                                        ;LOAD
             + /CS*/L4                                        ;HOLD
            :+:/CS* STROBE* CI*/L0*/L1*/L2*/L3                ;INCREMENT

        /L5 :=  CS*/D5                                        ;LOAD
             + /CS*/L5                                        ;HOLD
            :+:/CS* STROBE* CI*/L0*/L1*/L2*/L3*/L4            ;INCREMENT

        /L6 :=  CS*/D6                                        ;LOAD
             + /CS*/L6                                        ;HOLD
            :+:/CS* STROBE* CI*/L0*/L1*/L2*/L3*/L4*/L5        ;INCREMENT

        /L7 :=  CS*/D7                                        ;LOAD
             + /CS*/L7                                        ;HOLD
            :+:/CS* STROBE* CI*/L0*/L1*/L2*/L3*/L4*/L5*/L6    ;INCREMENT
```

FUNCTION TABLE

```
CLK STROBE /CS /DMACK D7 D6 D5 D4 D3 D2 D1 D0
/CI L7 L6 L5 L4 L3 L2 L1 L0 /CO

;  S    /
;  T    D
;  R    M
;C O  / A                    /                 /
;L B  C C  D D D D D D D D  C  L L L L L L L L  C
;K E  S K  7 6 5 4 3 2 1 0  I  7 6 5 4 3 2 1 0  O   DESCRIPTION
-------------------------------------------------------------------------------
 C X  L L  H H H H H H H H  L  H H H H H H H H  H   LOAD LENGTH
 C L  H L  X X X X X X X X  L  H H H H H H H H  H   COUNT DISABLED BY STROBE
 C L  H L  X X X X X X X X  L  H H H H H H H H  H   COUNT DISABLED BY STROBE
 C H  H L  X X X X X X X X  H  H H H H H H H H  H   COUNT DISABLED BY CI
 C H  H L  X X X X X X X X  H  H H H H H H H H  H   COUNT DISABLED BY CI
 C H  H L  X X X X X X X X  L  H H H H H H H H  L   COUNT DOWN
 C H  H L  X X X X X X X X  L  H H H H H H H L  H   COUNT DOWN
 C H  H L  X X X X X X X X  L  H H H H H H L L  H   COUNT DOWN
 C H  H L  X X X X X X X X  L  H H H H H L H H  H   COUNT DOWN
 C H  H L  X X X X X X X X  L  H H H H H L H L  H   COUNT DOWN
 C X  L L  L L L L L L L L  X  L L L L L L L L  L   LOAD ADDRESS/ TRIGGER CO
 C X  X H  X X X X X X X X  X  Z Z Z Z Z Z Z Z  X   HI-Z
-------------------------------------------------------------------------------
```

DESCRIPTION

THIS PAL DESIGN SPECIFICATION IS THE LENGTH COUNTER.



DMA LENGTH COUNTER

```
  1 C011111111XX00HHHHHHHHH1
  2 C0XXXXXXXX0X01HHHHHHHHH1
  3 C0XXXXXXXX0X01HHHHHHHHH1
  4 C1XXXXXXXX1X01HHHHHHHHH1
  5 C1XXXXXXXX1X01HHHHHHHHH1
  6 C0XXXXXXXX1X01HHHHHHHLH1
  7 C0XXXXXXXX1X01HHHHHHLHH1
  8 C0XXXXXXXX1X01HHHHHHLLH1
  9 C0XXXXXXXX1X01HHHHHLHHH1
 10 C0XXXXXXXX1X01HHHHHLHLH1
 11 CX00000000XX00LLLLLLLLL1
 12 CXXXXXXXXXX1XZZZZZZZZZX1
```

PASS SIMULATION

```
DMA LENGTH COUNTER

                11 1111 1111 2222 2222 2233 3333 3333
   0123 4567 8901 2345 6789 0123 4567 8901 2345 6789

 0 ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
 1 ---- ---X ---X ---X ---X ---X ---X ---X ---X ---- /L7*/L6*/L5*/L4*/L3*/L2-

 8 ---- -X-- ---- ---- ---- ---- ---- ---- ---- ---X CS*/D0
 9 ---- ---X ---- ---- ---- ---- ---- ---- ---- --X- /CS*/L0
10 -X-- ---- ---- ---- ---- ---- ---- ---- ---- X-X- /CS*STROBE*CI

16 ---- ---- -X-- ---- ---- ---- ---- ---- ---- ---X CS*/D1
17 ---- ---- ---X ---- ---- ---- ---- ---- ---- --X- /CS*/L1
18 -X-- ---X ---- ---- ---- ---- ---- ---- ---- X-X- /CS*STROBE*CI*/L0

24 ---- ---- ---- -X-- ---- ---- ---- ---- ---- ---X CS*/D2
25 ---- ---- ---- ---X ---- ---- ---- ---- ---- --X- /CS*/L2
26 -X-- ---X ---X ---- ---- ---- ---- ---- ---- X-X- /CS*STROBE*CI*/L0*/L1

32 ---- ---- ---- ---- -X-- ---- ---- ---- ---- ---X CS*/D3
33 ---- ---- ---- ---- ---X ---- ---- ---- ---- --X- /CS*/L3
34 -X-- ---X ---X ---X ---- ---- ---- ---- ---- X-X- /CS*STROBE*CI*/L0*/L1*/-

40 ---- ---- ---- ---- ---- -X-- ---- ---- ---- ---X CS*/D4
41 ---- ---- ---- ---- ---- ---X ---- ---- ---- --X- /CS*/L4
42 -X-- ---X ---X ---X ---X ---- ---- ---- ---- X-X- /CS*STROBE*CI*/L0*/L1*/-

48 ---- ---- ---- ---- ---- ---- -X-- ---- ---- ---X CS*/D5
49 ---- ---- ---- ---- ---- ---- ---X ---- ---- --X- /CS*/L5
50 -X-- ---X ---X ---X ---X ---X ---- ---- ---- X-X- /CS*STROBE*CI*/L0*/L1*/-

56 ---- ---- ---- ---- ---- ---- ---- -X-- ---- ---X CS*/D6
57 ---- ---- ---- ---- ---- ---- ---- ---X ---- --X- /CS*/L6
58 -X-- ---X ---X ---X ---X ---X ---X ---- ---- X-X- /CS*STROBE*CI*/L0*/L1*/-

64 ---- ---- ---- ---- ---- ---- ---- ---- -X-- ---X CS*/D7
65 ---- ---- ---- ---- ---- ---- ---- ---- ---X --X- /CS*/L7
66 -X-- ---X ---X ---X ---X ---X ---X ---X ---- X-X- /CS*STROBE*CI*/L0*/L1*/-


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

NUMBER OF FUSES BLOW =  948
```

```
PAL20X10                                          PAL DESIGN SPECIFICATIONS
DMA3.DAT                                          DANESH TAVANA
DMA CONTROLLER PAL
MMI SUNYVALE
CLK /RESET     /CS     D0 D1 NC NC NC  NC /DMAREQ /BUSACK GND
/OC /BUSREQ /DMACK STROBE T1 T0 I1 I0 /WR     /RD   /MREQ VCC

    MREQ := /RESET* MREQ                      ;HOLD
          + /RESET* MREQ                      ;HOLD
         :+:/RESET* DMACK*/T1*/T0             ;SET ON FIRST CLOCK IF DMACK
          + /RESET* DMACK* T1* T0             ;RESET ON FOURTH CLOCK IF DMACK

      RD := /RESET* RD                        ;HOLD
          + /RESET* RD                        ;HOLD
         :+:/RESET* MREQ*/T1* T0* I1* I0      ;SET ON SECOND CLOCK IF READ INST.
          + /RESET* MREQ* T1* T0* I1* I0      ;RESET ON FOURTH CLOCK

      WR := /RESET* WR                        ;HOLD
          + /RESET* WR                        ;HOLD
         :+:/RESET* MREQ* T1*/T0*/I1*/I0      ;SET ON THIRD CLOCK IF WRITE INST.
          + /RESET* MREQ* T1* T0*/I1*/I0      ;RESET ON THIRD CLOCK

     /I0 :=  CS*/D0                           ;LOAD DATA IF CHIP SELECTED
          +  CS*/D0                           ;LOAD DATA IF CHIP SELECTED
         :+:/CS*/I0                           ;HOLD INSTRUCTION OTHERWISE

     /I1 :=  CS*/D1                           ;LOAD DATA IF CHIP SELECTED
          +  CS*/D1                           ;LOAD DATA IF CHIP SELECTED
         :+:/CS*/I1                           ;HOLD INSTRUCTION OTHERWISE

     /T0 := /RESET*/T0                        ;STATE COUNTER
          +  RESET                            ;RESET
         :+:/RESET* DMACK                     ;START COUNTER IF DMACK

     /T1 := /RESET*/T1                        ;STATE COUNTER
          +  RESET                            ;RESET
         :+:/RESET* DMACK* T0                 ;START COUNTER IF DMACK

 /STROBE := /RESET*/STROBE                    ;HOLD
          + /RESET*/STROBE                    ;HOLD
         :+:/RESET* DMACK*/T1*/T0             ;SET ON FIRST CLOCK IF DMACK
          + /RESET* DMACK* T1* T0             ;RESET ON FOURTH CLOCK

   DMACK := /RESET* DMACK                     ;HOLD
          + /RESET* DMACK                     ;HOLD
         :+:/RESET*/DMACK * BUSACK            ;SET IF BUS IS ACKNOWLEDGED
          + /RESET*/DMAREQ* BUSREQ            ;RESET IF NO DMA REQUEST

  BUSREQ := /RESET* BUSREQ                    ;HOLD
          + /RESET* BUSREQ                    ;HOLD
         :+:/RESET* DMAREQ*/BUSREQ            ;SET IF THERE IS DMA REQUEST
          + /RESET*/DMAREQ* BUSREQ            ;RESET IF NO DMA REQUEST
```

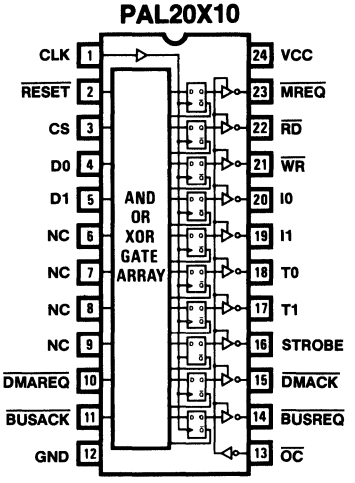
FUNCTION TABLE

CLK /RESET /CS D1 D0 I1 I0 T1 T0 STROBE
/MREQ /RD /WR /DMAREQ /BUSREQ /BUSACK /DMACK

;																
;	/							S				/	/	B	/	
;	R							T	/			M	U	U	D	
;	E							R	M			A	S	S	M	
;C	S	/						O	R	/	/	R	R	A	A	
;L	E	C	D	D	I	I	T	T	B	E	R	W	E	E	C	C
;K	T	S	1	0	1	0	1	0	E	Q	D	R	Q	Q	K	K
C	L	L	L	L	L	L	L	L	H	H	H	H	X	H	X	H
C	H	H	X	X	L	L	L	L	H	H	H	H	H	H	H	H
C	H	H	X	X	L	L	L	L	H	H	H	H	H	H	H	H
C	H	H	X	X	L	L	L	L	H	H	H	H	H	H	H	H
C	H	H	X	X	L	L	L	L	H	H	H	H	L	L	H	H
C	H	H	X	X	L	L	L	L	H	H	H	H	L	L	L	L
C	H	H	X	X	L	L	L	H	L	L	H	H	L	L	L	L
C	H	H	X	X	L	L	H	L	L	L	H	H	L	L	L	L
C	H	H	X	X	L	L	H	H	L	L	H	L	L	L	L	L
C	H	H	X	X	L	L	L	L	H	H	H	H	H	H	X	H
C	H	L	H	H	H	H	L	L	H	H	H	H	H	H	H	H
C	H	H	H	H	H	H	L	L	H	H	H	H	H	H	H	H
C	H	H	X	X	H	H	L	L	H	H	H	H	L	L	H	H
C	H	H	X	X	H	H	L	L	H	H	H	H	L	L	L	L
C	H	H	X	X	H	H	L	H	L	L	H	H	L	L	L	L
C	H	H	X	X	H	H	H	L	L	L	L	H	L	L	L	L
C	H	H	X	X	H	H	H	H	L	L	L	H	L	L	L	L
C	H	H	X	X	H	H	L	L	H	H	H	H	L	L	L	L
C	H	H	X	X	H	H	L	H	L	L	H	H	L	L	L	L
C	H	H	X	X	H	H	H	L	L	L	L	H	L	L	L	L
C	H	H	X	X	H	H	H	H	L	L	L	H	L	L	L	L
C	H	H	X	X	H	H	L	L	H	H	H	H	H	H	X	H

DESCRIPTION

THIS PAL DESIGN SPECIFICATION IS THE DMA CONTROLLER PAL.



DMA CONTROLLER PAL

```
 1 C0000LLLLXXXLHHHLLLLHHH1
 2 C11XXLLLL11XLHHHLLLLHHH1
 3 C11XXLLLL11XLHHHLLLLHHH1
 4 C11XXLLLL11XLHHHLLLLHHH1
 5 C11XXLLLL01XLLHHLLLLHHH1
 6 C11XXLLLL00XLLLHLLLLHHH1
 7 C11XXLLLL00XLLLLLHLLHHL1
 8 C11XXLLLL00XLLLLHLLLHHL1
 9 C11XXLLLL00XLLLLHHLLLHL1
10 C11XXLLLL1XXLHHHLLLLHHH1
11 C1011LLLL11XLHHHLLHHHHH1
12 C1111LLLL11XLHHHLLHHHHH1
13 C11XXLLLL01XLLHHLLHHHHH1
14 C11XXLLLL00XLLLHLLHHHHH1
15 C11XXLLLL00XLLLLLHHHHHL1
16 C11XXLLLL00XLLLLHLHHHLL1
17 C11XXLLLL00XLLLLHHHHHLL1
18 C11XXLLLL00XLLLHLLHHHHH1
19 C11XXLLLL00XLLLLLHHHHHL1
20 C11XXLLLL00XLLLLHLHHHLL1
21 C11XXLLLL00XLLLLHHHHHLL1
22 C11XXLLLL1XXLHHHLLHHHHH1
```

PASS SIMULATION

8

```
DMA CONTROLLER PAL

                 11 1111 1111 2222 2222 2233 3333 3333
   0123 4567 8901 2345 6789 0123 4567 8901 2345 6789

 0 X--X ---- ---- ---- ---- ---- ---- ---- ---- ---- /RESET*MREQ
 1 X--X ---- ---- ---- ---- ---- ---- ---- ---- ---- /RESET*MREQ
 2 X--- ---- ---- ---- ---- ---X ---X ---- ---X ---- /RESET*DMACK*/T1*/T0
 3 X--- ---- ---- ---- ---- --X- --X- ---- ---X ---- /RESET*DMACK*T1*T0

 8 X--- ---X ---- ---- ---- ---- ---- ---- ---- ---- /RESET*RD
 9 X--- ---X ---- ---- ---- ---- ---- ---- ---- ---- /RESET*RD
10 X--X ---- ---- --X- --X- --X- ---X ---- ---- ---- /RESET*MREQ*/T1*T0*I1*I0
11 X--X ---- ---- --X- --X- --X- --X- ---- ---- ---- /RESET*MREQ*T1*T0*I1*I0

16 X--- ---- ---X ---- ---- ---- ---- ---- ---- ---- /RESET*WR
17 X--- ---- ---X ---- ---- ---- ---- ---- ---- ---- /RESET*WR
18 X--X ---- ---- ---X ---X ---X --X- ---- ---- ---- /RESET*MREQ*T1*/T0*/I1*-
19 X--X ---- ---- ---X ---X --X- --X- ---- ---- ---- /RESET*MREQ*T1*T0*/I1*/-

24 ---- -X-- -X-- ---- ---- ---- ---- ---- ---- ---- CS*/D0
25 ---- -X-- -X-- ---- ---- ---- ---- ---- ---- ---- CS*/D0
26 ---- X--- ---- ---X ---- ---- ---- ---- ---- ---- /CS*/I0

32 ---- -X-- ---- -X-- ---- ---- ---- ---- ---- ---- CS*/D1
33 ---- -X-- ---- -X-- ---- ---- ---- ---- ---- ---- CS*/D1
34 ---- X--- ---- ---- ---X ---- ---- ---- ---- ---- /CS*/I1

40 X--- ---- ---- ---- ---- ---X ---- ---- ---- ---- /RESET*/T0
41 -X-- ---- ---- ---- ---- ---- ---- ---- ---- ---- RESET
42 X--- ---- ---- ---- ---- ---- ---- ---- ---X ---- /RESET*DMACK

48 X--- ---- ---- ---- ---- ---- ---X ---- ---- ---- /RESET*/T1
49 -X-- ---- ---- ---- ---- ---- ---- ---- ---- ---- RESET
50 X--- ---- ---- ---- ---- --X- ---- ---- ---X ---- /RESET*DMACK*T0

56 X--- ---- ---- ---- ---- ---- ---- ---X ---- ---- /RESET*/STROBE
57 X--- ---- ---- ---- ---- ---- ---- ---X ---- ---- /RESET*/STROBE
58 X--- ---- ---- ---- ---- ---X ---X ---- ---X ---- /RESET*DMACK*/T1*/T0
59 X--- ---- ---- ---- ---- --X- --X- ---- ---X ---- /RESET*DMACK*T1*T0

64 X--- ---- ---- ---- ---- ---- ---- ---- ---X ---- /RESET*DMACK
65 X--- ---- ---- ---- ---- ---- ---- ---- ---X ---- /RESET*DMACK
66 X--- ---- ---- ---- ---- ---- ---- ---- --X- -X-- /RESET*/DMACK*BUSACK
67 X--- ---- ---- ---- ---- ---- ---- ---- X--- ---X /RESET*/DMAREQ*BUSREQ

72 X--- ---- ---- ---- ---- ---- ---- ---- ---- ---X /RESET*BUSREQ
73 X--- ---- ---- ---- ---- ---- ---- ---- ---- ---X /RESET*BUSREQ
74 X--- ---- ---- ---- ---- ---- ---- ---- -X-- --X- /RESET*DMAREQ*/BUSREQ
75 X--- ---- ---- ---- ---- ---- ---- ---- X--- ---X /RESET*/DMAREQ*BUSREQ


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

NUMBER OF FUSES BLOW = 1341
```

Testing Your PAL Devices

Manouchehr Vafai

Introduction

The advantage of Programmable Array Logic (PAL®) circuits as a basic building block of digital system is now well established.

PAL circuits are a unified group of devices which combine programmable flexibility with high speed and extensive selection of interface options.

The architecture of PAL circuits consists of programmable-AND-OR gate arrays, output-registers and I/O feedback as shown in Figure 1.

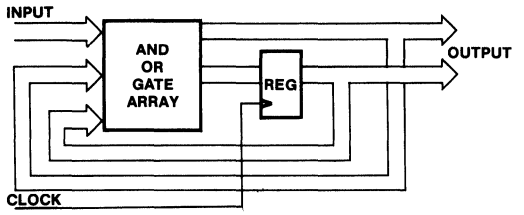


Figure 1. PAL Circuit Architecture

The increased system speed, reduced chip count and availability of a CAD tool called PALASM® software should leave no doubt for design engineers that they have made a right choice in choosing PAL circuits.

The HAL circuit family is the masked program version of a PAL circuit. HAL® circuits will provide the users a cost-effective solution for large quantities and is unique in that it is a gate array with a programmable prototype.

The following steps are required when designing with PAL circuits.

- Familiarity with Demorgan's law.
- Familiarity with the Karnaugh maps.
- Ability to express logic equation in Sum-of-Product form.
- Ability to write simple seed vector for function table.
- Familiarity with different PAL circuits.

Programming PAL Circuits

PAL circuits will be programmed using PALASM software.

PALASM software is the CAD tool developed by Monolithic Memories to facilitate the process of programming. PALASM software is a Fortran IV program which assembles and simulates PAL circuits design specifications. It generates PAL circuit fuse patterns in formats compatible with PAL circuits or PROM programmers.

Besides generating PAL circuit fuse pattern in different programming formats, PALASM software does the following:

- Assembles PAL circuit design specification and reports error messages.
- Simulates the Function Table.
- Tests each product term for Stuck at zero (SA0) and stuck at one (SA1) faults.

The purpose of writing vectors is to prove that a device is capable of performing it's function before it is put in a system. PALASM software will exercise the vectors and will report any discrepancy. Writing vectors will raise confidence that a device will function properly at least in the design level. The simulator also transfers the function table vectors to a set of universal test vectors which may be used for functional testing after the device is programmed.

When a new system is transferred to production, the system designer hands over the responsibility for the system to the test engineering department, who now determines how and what test should be performed to ensure proper operation of the system. At this point the system designer transmits the necessary information for understanding the system operation. Unfortunately, much information is lost at this point. Test engineers usually have a hard time understanding how the system works with insufficient information. It is the design engineer who best knows the operation of his PAL circuit design, and it is the design engineer who can quickly specify a few seed vectors to give the test a starting point in solving the future problem.

Design for Testability

In short the only way to control a digital circuit is to apply a known value to it's input. Fault simulation has been the best technique of yielding a quantitive measure of test effectiveness. Fault simulation will test stuck-at-0 (SA0) and stuck-at-1 (SA1) of input and output lines. By generating test vectors that will test for each product term for (SA0) and (SA1) faults, then by observing the corresponding output and comparing it with the fault-free output, one can conclude whether a fault can be detected or not.

Consider the following example:

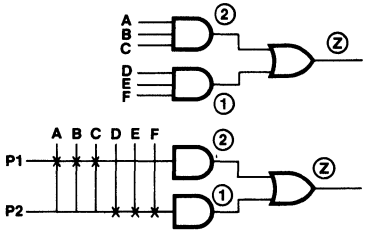


Figure 2. Logic Diagram and It's PAL Circuit Implementation

A	B	C	D	E	F	Z	
1	1	1	0	X	X	1	(vector-1)

The (vector-1) selects a product term P1. Under a fault-free condition, the output (2) will be high (we can observe this); however, under a fault condition the output will be low. In other words, one can conclude that either product term (P1) is (SA0) or outputs Z (Figure 2.) are (SA0).

Now consider vector 2.

A	B	C	D	E	F	G	
0	0	0	0	0	0	0	(vector-2)

As it can be seen that both of the product terms are low, if the observed output is high, one can conclude that either product terms or outputs are Stuck-at-one.

Fault simulation grading is used by Monolithic Memories to evaluate candidates design for transfering from a PAL circuit to a HAL circuit.

In designing with PAL circuits, four different cases should be considered.

1. A purely combinational circuit where output is function of input.
2. A purely combinational circuit where output is function of input and feedback from output.
3. A purely sequential logic where output is function of input and feedback from output.
4. A combinational-sequential logic where output is function of input, feedback from combinational output and feedback from sequential output.

In cases 1 and 2 we can define a structured way of writing function table. Cases 3 and 4, on the other hand, because of dependency of the device on the previous state of the device, impose a relatively more sophisticated scheme of testing strategy.

In the following examples the various techniques which might be helpful in testability of PALs, will be discussed.

Example 1: Glitch-free and Testable

Suppose we want to implement (EQ-1) using any of the combinational PAL.

$$F = X^*A + \overline{X}^*B \qquad \text{(eq-1)}$$

The K-MAP and logic diagram are shown in (Figure 3.)

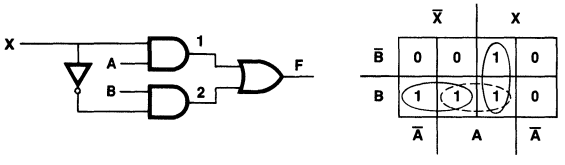


Figure 3. Logic Diagram and It's K-Map

The above logic is testable because we have full control over each node for (SA0) and (SA1) test.

The implementation using PAL circuits is as follows:

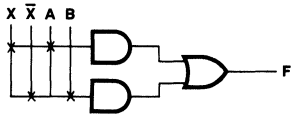


Figure 4. PAL Circuit Implementation of the Logic

Ideally the output should always be high if both inputs are high. The circuit is not glitch-free, the output might momentarily drop to low if we change the state of X, due to propagation delay between X and $\overline{X}$.

The problem will be solved by including a redundant (AB) term to (eq-2).

The equation will look like this.

$$F = XA + \overline{X}B + AB \qquad \text{(eq-2)}$$

The output is glitch-free, but untestable!

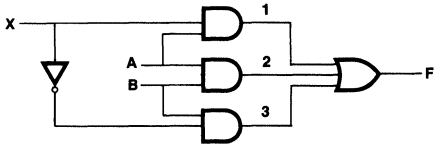


Figure 5. A Glitch-free Circuit

Node (2) is not Observable for (SA0). One can not force node (2) to one and keep node (1) and (3) in the low state. So the redundant product term is untestable.

This circuit can be made testable by the addition of control signal (Y) as follows:

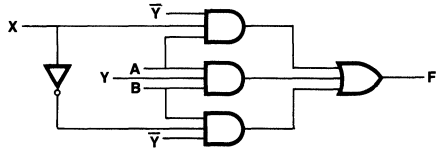


Figure 6. Glitch-free and Testable Logic

Now the logic is glitch-free and testable.

Example 2: Untestable Logic — A Simple Example

The logic F := $\overline{F}$ is untestable

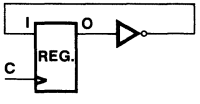


Figure 7. Implementation of F := $\overline{F}$

The initial state of the oscillator is unknown; this system can be made testable as follows:

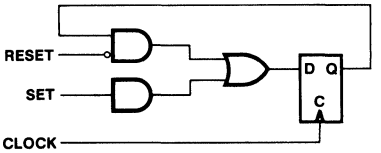


Figure 8. Implementation of F := $\overline{F}$ with SET and RESET

It has been done by addition of two control signals (RESET and SET) and one extra product term.

Illegal States

Upon power-up the initial state of output registers are unknown; this might force the device into one of the "illegal states".

The design engineer should be worried about the illegal state at design time. For example let's look at modulo-6 state machine

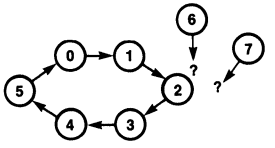


Figure 9. State Transition Diagram for a Modulo-6 Counter

The design engineer might ignore the other possible state (6, 7) but his ignorance might be costly at test time. If upon power-up the machine starts at either of (6) or (7) state, there is no way to control the state-machine. The best solution is to force both of the illegal states into one of the known states.

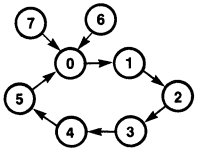


Figure 10. A Modulo 5 Counter with No Illegal State

Example 3: Design "pitfall" Case One

Consider the implementation of the following example

Q1 := I1 * Q1

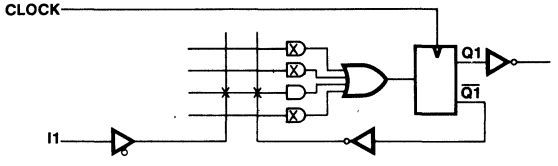


Figure 11. Implementation of Q1 := I1 * Q1 Using a PAL

If Q1 falls to zero, it will stay there forever. The logic needs a control signal for output reset.

Example 4: Design "Pitfall" Case Two

Consider the implementation of the following equation:

$\overline{Q0} = \overline{A}*\overline{Q1}*Q0 + A*Q1*\overline{Q0} + \overline{Q0}$

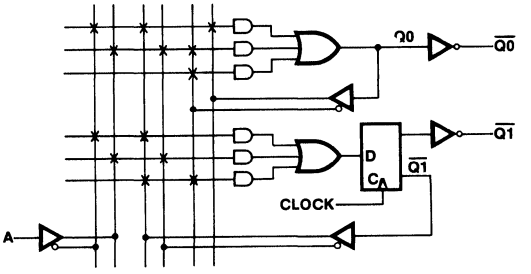


Figure 12. Implementation of $\overline{Q0} = \overline{A}*\overline{Q1}*Q0 + A*Q1*\overline{Q0} + \overline{Q0}$

If $\overline{Q0}$ goes to one it will stay there forever, the logic needs a control signal to clear it's output.

Hard Array Logic (HAL)

The HAL is the Hard Array version of PAL.

HAL's are the best choice for designs that are firm and volumes are large enough to justify the initial cost. Besides having boolean equation in PAL DESIGN SPECIFICATION format the user should provide the following.

1: A FUNCTION TABLE which gives enough information about the operation of the device. Normally this FUNCTION TABLE shall test a minimum of 50% "Stuck at fault" grading using PALASM or TEGAS fault grading test.

2: The FUNCTION TABLE shall be constructed such that the device may be initilized to a known state within a specified number of steps (or clocks).

The HAL CIRCUIT SPECIFICATION is the input file used with PALASM software for the HAL's. The input format as shown in example 5 is as follow:

- Line 1 HAL circuit part number
- Line 2 user's part number followed by originator's name and the date
- Line 3 device application name
- Line 4 user's company name, city, state
- Line 5 pin list which is a sequence of symbolic names separated by one or more spaces. All pins including VCC and GND must be named
- Line M the logic equation which are used to generate metal masks from the provided equations

```
HAL12H6                                  HAL DESIGN SPECIFICATION
EXAMPLE 5                                 MANO  VAFAI  04/20/83
BASIC GATES
MMI SANTA CLARA, CALIFORNIA
C D F G M N P Q I GND J K L R O H E B A VCC

B   = /A                ;       INVERTER GATE (EQ - 1)
                        ;     ONE PRODUCT TERM: #1 (/A)

E   = C*D               ;         AND GATE  (EQ - 2)
                        ;      ONE PRODUCT TERM: #1 (C*D)

H   = F + G             ;          OR GATE (EQ - 3)
                        ;       TWO PRODUCT TERMS: #1 (F), #2 (G)

L   = /I + /J + /K      ;          NAND GATE (EQ -4)
                        ; THREE PRODUCT TERMS: #1(/I), #2(/J), #3(/K)

O   = /M*/N             ;          NOR GATE  (EQ-5)
                        ;    ONE PRODUCT TERM: #1 (/M*/N)

R   = P*/Q + /P*Q       ;          XOR GATE  (EQ-6)
                        ;    TWO PRODUCT TERMS: #1(P*/Q), #2(/P*Q)

FUNCTION TABLE

A B C D E F G H I J K L M N O P Q R

;AB CDE FGH IJKL MNO PQR     OPTIONAL COMMENTS FIELD
-------------------------------------------------------
 LH XXX XXX XXXX XXX XXX     (EQ-1,PT-1)        SA0 TEST
 HL XXX XXX XXXX XXX XXX     (EQ-1,PT-1)        SA1 TEST
 XX HHH XXX XXXX XXX XXX     (EQ-2,PT-1)        SA0 TEST
 XX LLL XXX XXXX XXX XXX     (EQ-2,PT-1)        SA1 TEST
 XX XXX HLH XXXX XXX XXX     (EQ-3,PT-1)        SA0 TEST
 XX XXX LHH XXXX XXX XXX     (EQ-3,PT-2)        SA0 TEST
 XX XXX LLL XXXX XXX XXX     (EQ-3,PT-1,2)      SA1 TEST
 XX XXX XXX LHHH XXX XXX     (EQ-4,PT-1)        SA0 TEST
 XX XXX XXX HLHH XXX XXX     (EQ-4,PT-2)        SA0 TEST
 XX XXX XXX HHLH XXX XXX     (EQ-4,PT-3)        SA0 TEST
 XX XXX XXX HHHL XXX XXX     (EQ-4,PT-1,2,3)    SA1 TEST
 XX XXX XXX XXXX LLH XXX     (EQ-5,PT-1)        SA0 TEST
 XX XXX XXX XXXX HHL XXX     (EQ-5,PT-1,2)      SA1 TEST
 XX XXX XXX XXXX XXX HLH     (EQ-6,PT-1)        SA0 TEST
-------------------------------------------------------

DESCRIPTION
THE MAIN PURPOSE OF THIS EXAMPLE IS TO FAMILIARIZE
THE USER WITH WHAT WE MEAN BY "FUNCTION TABLE",
PRODUCT TERM(PT) COVERAGE, STUCK-AT-0(SA0) AND
 STUCK-AT-ONE (SA1) TESTS.
```

LINE 1
LINE 2
LINE 3
LINE 4
LINE 5
LINE M
LINE N

EXAMPLE 5

- Line N the function table which begins with the key word "FUNCTION TABLE." It's followed by a pin list which may be in a different order and polarity from the pin list in line 5. VCC and GND cannot be listed. The pin list is followed by dashed line; e.g.;_ _ _ _ which in turn is followed by a list of vectors, one vector per line. One state must be specified for each pin name and optionally separated by spaces. A vector is a sequence of states listed in the same order as the pin list and followed by an optional comment.

The allowable states are H (HIGH LEVEL), L (LOW LEVEL), X (IRRELEVENT), C (TRANSITION FROM HIGH TO LOW OR CLOCK PULSE) and Z (HIGH IMPEDENCE). After preparing the PAL DESIGN SPECIFICATION in the above format, PALASM software can be used to simulate and perform fault testing.

```
BASIC GATES

  1 XXXXXXXXXXXXXXXXH01
  2 XXXXXXXXXXXXXXXXL11
  3 11XXXXXXXXXXXXXHXX1
  4 00XXXXXXXXXXXXXLXX1
  5 XX10XXXXXXXXXXXHXX1
  6 XX01XXXXXXXXXXXHXX1
  7 XX00XXXXXXXXXXLXXX1
  8 XXXXXXXX0X11HXXXXX1
  9 XXXXXXXX1X01HXXXXX1
 10 XXXXXXXX1X10HXXXXX1
 11 XXXXXXXX1X11LXXXXX1
 12 XXXX00XXXXXXXHXXXX1
 13 XXXX11XXXXXXXLXXXX1
 14 XXXXXX10XXXXXHXXXX1

PASS SIMULATION
 PRODUCT:   1 OF EQUATION.  6 UNTESTED(SA1)FAULT
 PRODUCT:   2 OF EQUATION.  6 UNTESTED(SA1)FAULT
 PRODUCT:   2 OF EQUATION.  6 UNTESTED(SA0)FAULT

NUMBER OF STUCK AT ONE (SA1)  FAULTS ARE =  8

NUMBER OF STUCK AT ZERO (SA0) FAULTS ARE =  9

PRODUCT  TERM   COVERAGE                  = 85%
```

$P * \overline{Q}$

$R = P.\overline{Q} + \overline{P}.Q$ EQ - 6

$\overline{P} * Q$

FAULT-TESTING

The following information is reported to the user

— Total number of SA1 Faults. (8 in example 5)

— Total number of detected SA0 faults. (9 in example 5)

— $\dfrac{\text{SA1 faults + SA0 faults}}{\text{2 * total number of product terms}}$

$* 100\ \%\quad (\dfrac{8+9}{2*10} \quad *100\% = 85\%$ ex-5)

— One vector may detect more than one SA0 OR SA1 FAULTS (vector # 11 in example 5)

— The user is reported with a message which tells him the product term for which it was not tested. (PRODUCT TERM 1 & 2-EQ 6, in example 5)

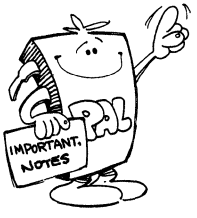
The following vectors can be added to the function table in example 5 in order to achieve 100% fault coverage.

```
AB CDE FGH IJKL MNO PQR        COMMENTS (EXAMPLE 5)
XX XXX XXX XXXX XXX LHH        (EQ-6,PT2)     SA0 TEST
XX XXX XXX XXXX XXX HHL        (EQ-6,PT-1,2)  SA1 TEST
```

PALASM software has tested the above function table for example 5, the result is as follows:

```
BASIC GATES
  1 XXXXXXXXXXXXXXXXH01
  2 XXXXXXXXXXXXXXXXL11
  3 11XXXXXXXXXXXXXHXX1
  4 00XXXXXXXXXXXXXLXX1
  5 XX10XXXXXXXXXXXHXX1
  6 XX01XXXXXXXXXXXHXX1
  7 XX00XXXXXXXXXXLXXX1
  8 XXXXXXXX0X11HXXXXX1
  9 XXXXXXXX1X01HXXXXX1
 10 XXXXXXXX1X10HXXXXX1
 11 XXXXXXXX1X11LXXXXX1
 12 XXXX00XXXXXXXHXXXX1
 13 XXXX11XXXXXXXLXXXX1
 14 XXXXXX10XXXXXHXXXX1
 15 XXXXXX01XXXXXHXXXX1
 16 XXXXXX11XXXXXLXXXX1
PASS SIMULATION
NUMBER OF STUCK AT ONE (SA1)  FAULTS ARE = 10
NUMBER OF STUCK AT ZERO (SA0) FAULTS ARE = 10
PRODUCT  TERM   COVERAGE                  =100%
```

8


IMPORTANT
NOTES

California

Brea

Mike Vogel, FAE	(714) 556-1216

Cupertino

George Anderl	(408) 996-1477
Dan Jesswein, FAE	(408) 996-2014
Gerry Smith	(408) 225-2391

Pasadena

Wayne Caraway	(714) 556-1216

Irvine

Bernard Brafman	(714) 556-1216

Colorado

Wheatridge

Roger Rios	(303) 423-1020

Georgia

Norcross

Tom Lewis	(404) 447-4119

Illinois

Lombard

Dick Jones	(312) 932-1940

Naperville

Sal Graziano	(312) 961-9200

Worth

Bill Karkula, FAE	(312) 961-9200

Massachusetts

Framingham

Jack Abbott	(617) 875-7373
Russ French	(617) 875-7373
Mike Volpigno, FAE	(617) 875-7373

Minnesota

Edina

Alex Sherbanenko	(612) 922-2260

New Jersey

Sussex

Bill Bartley	(201) 875-9430

Ohio

Dayton

Mike Wier	(513) 439-0470

Texas

Garland

Bob Rainwater	(214) 233-5833

Dallas

Brad Mitchell	(214) 233-5833

U.S.A.

Alabama

Huntsville
Hall-Mark Electronics (205) 837-8700

Arizona

Phoenix
Kierulff Electronics (602) 243-4101

Tempe
Marshall Electronics Group (602) 968-6181
Bell Industries (602) 966-7800
Anthem Electronics (602) 244-0900

Tucson
Kierulff Electronics (602) 624-9986

California

Canoga Park
Marshall Electronics Group (213) 999-5001

El Monte
Marshall Electronics Group (213) 686-0141

Irvine
Marshall Electronics Group (714) 556-6400

Los Angeles
Kierulff Electronics (213) 725-0325

Newport Beach
Arrow Electronics (714) 851-8961

Palo Alto
Kierulff Electronics (415) 968-6292

San Diego
Anthem Electronics (619) 279-5200
Kierulff Electronics (619) 278-2112
Arrow Electronics (619) 565-4800

San Jose
Anthem Electronics (408) 946-8000

Sunnyvale
Arrow Electronics (408) 745-6600

Tustin
Anthem Electronics (714) 730-8000
Image Electronics (714) 730-0303
Kierulff Electronics (714) 731-5711

Chatsworth
Anthem Electronics (213) 700-1000
Arrow Electronics (213) 701-7500

Colorado

Aurora
Arrow Electronics (303) 696-1111

Englewood
Anthem Electronics (303) 790-4500
Kierulff Electronics (303) 790-4444

Wheatridge
Bell Industries (303) 424-1985

Connecticut

S. Norwalk
Bond Electronics (203) 852-1001

Wallingford
Arrow Electronics (203) 265-7741
Marshall Electronics Group (203) 265-3822
Kierulff Electronics (203) 265-1115

Florida

Fort Lauderdale
Arrow Electronics (305) 776-7790
Hall-Mark Electronics (305) 971-9280

Orlando
Hall-Mark Electronics (305) 855-4020

Palm Bay
Arrow Electronics (305) 725-1480

St. Petersburg
Kierulff Electronics (813) 576-1966

Georgia

Norcross
Arrow Electronics (404) 449-8252
Hall-Mark Electronics (404) 447-8000
Kierulff Electronics (404) 447-5252

Illinois

Bensenville
Hall-Mark Electronics (312) 860-3800

Elk Grove Village
Kierulff Electronics (312) 640-0200

Schaumburg
Arrow Electronics (312) 397-3440

Indiana

Indianapolis
Advent Electronics (317) 872-4910
Arrow Electronics (317) 243-9353

Iowa

Cedar Rapids
Advent Electronics (319) 363-0221
Arrow Electronics (319) 395-7300

Kansas

Lenexa
Hall-Mark Electronics (913) 888-4747

Maryland

Baltimore
Arrow Electronics (301) 247-5200
Hall-Mark Electronics (301) 796-9300
Kierulff Electronics (301) 247-5020

Gaithersburg
Pioneer Washington (301) 948-0710

Massachusetts

Billerica
Kierulff Electronics (617) 667-8331

Burlington
Lionex (617) 272-9400

Woburn
Arrow Electronics (617) 933-8130

Michigan

Ann Arbor
Arrow Electronics (313) 971-8220

Grand Rapids
RS Electronics (616) 241-3483
Arrow Electronics (616) 243-0912

Kalamazoo
RS Electronics (616) 381-5470

Livonia
RS Electronics (313) 525-1155

Minnesota

Bloomington
Hall-Mark Electronics (612) 941-7500

Edina
Arrow Electronics (612) 830-1800
Kierulff Electronics (612) 835-4388

Missouri

Maryland Heights
Hall-Mark Electronics (314) 291-5350

St. Louis
Arrow Electronics (314) 567-6888

New Hampshire

Manchester
Arrow Electronics (603) 668-6968

New Jersey

Fairfield
Arrow Electronics (201) 575-5300
Kierulff Electronics (201) 575-6750
Lionex (201) 227-7960

Mt. Laurel
Marshall Electronics Group (215) 627-1920

Moorestown
Arrow Electronics (609) 235-1900

Cherry Hill
Hall-Mark Electronics (609) 424-0800

Nutley
Vantage Electronics (201) 667-1840

New Mexico

Albuquerque
Bell Industries (505) 292-2700
Arrow Electronics (505) 243-4566

New York

Buffalo
Summit Distributors (716) 884-3450

E. Syracuse
Add Electronics (315) 437-0300

Endwell
Marshall Electronics Group (607) 754-1570

Farmingdale
Arrow Electronics (516) 694-6800

Rochester
Arrow Electronics (716) 275-0300
Summit Distributors (716) 334-8110

Hauppauge
Arrow Electronics (516) 231-1000
Current Components (516) 273-2600
Lionex (516) 273-1660

Liverpool
Arrow Electronics (315) 652-1000

North Carolina

Raleigh
Hall-Mark Electronics (919) 872-0712
Resco Raleigh (919) 781-5700

Winston/Salem
Arrow Electronics (919) 725-8711

Ohio
Solon
Arrow Electronics (216) 248-3990
Centerville
Arrow Electronics (513) 435-5563
Cleveland
Hall-Mark Electronics (216) 473-2907
Dayton
Marshall Electronics Group (513) 236-8088
Westerville
Hall-Mark Electronics (614) 891-4555

Oklahoma
Tulsa
Hall-Mark Electronics (918) 835-8458
Quality Components (918) 664-8812
Radio, Inc. (918) 587-9123

Oregon
Portland
Kierulff Electronics (503) 641-9150

Pennsylvania
Horsham
Pioneer/Delaware Valley (215) 674-4000
Monroeville
Arrow Electronics (412) 856-7000

Texas
Addison
Quality Components (214) 387-4949
Austin
Hall-Mark Electronics (512) 258-8848
Quality Components (512) 835-0220
Dallas
Arrow Electronics (214) 386-7500
Hall-Mark Electronics (214) 234-7300
Houston
Arrow Electronics (713) 530-4700
Hall-Mark Electronics (713) 781-6100
Quality Components (713) 772-7100

Utah
Salt Lake City
Bell Industries (801) 972-6969
Kierulff Electronics (801) 973-6913

Washington
Bellevue
Almac Electronics Corporation (206) 643-9992
Arrow Electronics (206) 643-4800
Redmond
Anthem Electronics (206) 881-0850
Tukwila
Kierulff Electronics (206) 575-4420

Wisconsin
Oak Creek
Arrow Electronics (414) 764-6600
Hall-Mark Electronics (414) 761-3000
Waukesha
Kierulff Electronics (414) 784-8160

CANADA

Alberta
Calgary
Zentronics Limited (403) 230-1422

British Columbia
Richmond
Zentronics Limited (604) 273-5575
Vancouver
RAE Electronics (604) 291-8866

Manitoba
Winnipeg
Zentronics Limited (204) 775-8661

Ontario
Brampton
Zentronics Limited (416) 451-9600
Nepean
Zentronics Limited (613) 226-8840
Toronto
Future Electronics (416) 663-5563
Waterloo
Zentronics Limited (519) 884-5700

Quebec
Montreal
Future Electronics (514) 694-7710
Zentronics Limited (514) 735-5361

U.S.A.

Alabama
Huntsville
REP, Inc. (205) 881-9270

Arizona
Scottsdale
Summit Sales (602) 998-4850

California
Auburn
Thresum Associates (916) 269-1543
Cupertino
Thresum Associates (408) 996-9889
Fountain Valley
Bager Electronics (714) 957-3367
San Diego
Littlefield & Smith (619) 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
Palm Bay
Dyne-A-Mark (305) 727-0192

Georgia
Tucker
REP, Inc. (404) 938-4358

Illinois
Rolling Meadows
Sumer (312) 991-8500

Indiana
Indianapolis
Leslie M. DeVoe Co. (317) 842-3245

Iowa
Cedar Rapids
S & O Sales (319) 393-1845

Kansas
Olathe
Rush and West (913) 764-2700

Maryland
Baltimore
Monolithic Sales (301) 296-2444

Massachusetts
Westwood
Comp Rep Associates (617) 329-3454

Michigan
Grosse Point
Greiner Associates (313) 499-0188

Minnesota
Edina
Technical Sales, Inc. (612) 941-9790

Missouri
Ballwin
Rush and West (314) 394-7271

New Jersey
Teaneck
R.T. Reid Associates (201) 692-0200

New Mexico
Albuquerque
BFA Corporation (505) 292-1212

New York
East Rochester
Tri-Tech Electronics, Incorporated (716) 385-6500
Endwell
Tri-Tech Electronics, Incorporated (607) 754-1094
Fayetteville
Tri-Tech Electronics, Incorporated (315) 446-2881
Fishkill
Tri-Tech Electronics, Incorporated (914) 897-5611

North Carolina
Raleigh
REP, Inc. (919) 851-3007

Ohio
Cincinnati
Makin Associates (513) 871-2424
Columbus
Makin Associates (614) 459-2423
Kent
Makin Associates (216) 921-0080

Oklahoma
Tulsa
West Associates (918) 492-0390

Oregon
Portland
Northwest Marketing (503) 297-2581

Pennsylvania
Glenside
CMS Marketing (215) 885-5106

Puerto Rico
Mayaguez
Comp Rep Associates (809) 832-9529

Tennessee
Jefferson City
REP, Inc. (615) 475-4105

Texas
Austin
West Associates (512) 454-3681
Dallas
West Associates (214) 248-7060
Houston
West Associates (713) 777-4108

Utah
Salt Lake City
Waugaman Associates (801) 261-0802

Washington
Bellevue
Northwest Marketing (206) 455-5846

Wisconsin
Brookfield
Sumer (414) 784-6641

CANADA

Ontario
Brampton
Cantec (416) 791-5922
Ottawa
Cantec (613) 725-3704
Waterloo
Cantec (519) 744-6341

Quebec
Dollard Des Ormeaux
Cantec (514) 683-6131

AUSTRIA

Ing. Ernst Steiner
Hummelgasse 14
A 1130 Wien
Phone: 22-8274740
Telex: 135026

AUSTRALIA

R & D Electronics Pty Ltd.
257 Burwood Hwy.
Burwood, Vic. 3125
Phone: 3-288-8232
Telex: AA33288

R & D Electronics Pty Ltd.
133 Alexander St.
Crows Nest—2065
Phone: 2-439-5488
Telex: AA25468

BELGIUM

D & D Electronics
VII Olympiadelaan 93
2020 Antwerp
Phone: 3-8277934
Telex: 73121

DENMARK

C-88 APS
Kokkedal Industripark 42A
DK-2980 Kokkedal
Phone: 244888
Telex: 41198

ENGLAND

Monolithic Memories Ltd.
Lynwood House
1 Camp Road
Farnborough
Hampshire
GU14 6EN
Phone: 9-011-44-252-511099
Telex: 858051 MONO UK G

Memory Devices Ltd.
5th Floor Hagley House
Hagley Road
Edgbaston
Birmingham B16 8QG
Phone: 021-455-9395
Telex: 339752 Analog G

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

Macro Marketing Ltd.
396 Bath Road
Slough, Berkshire
Phone: 628663011
Telex: 847083

Microlog Ltd.
First Floor, Elizabeth House
Duke Street
Woking, Surrey GU21 5BA
Phone: (04862) 66771
Telex: 859219 (ULOG G)

FINLAND

Findip-Havulinna
Instrumentarium Oy
P.O. Box 357
00101 Helsinki 10
Finland

FRANCE

Monolithic Memories France S.A.R.L.
Silic 463
94613 Rungis Cedex
Phone: 1-6874500
Telex: 202146
FAX: 6860818

Alfatronic S.A.R.L.
La Tour d'Asnieres 4
Avenue Laurent Cely
F 92606 Asnieres
Phone: 1-7914444
Telex: 612790

Bellion Electronique
Z.I. Kerscao/Brest
B.P. 16-29219 Le Relecq-Kerhuon
Phone: 9-(98)-28-03-03
Telex: 940930

Composants S.A.
Avenue Gustave Eiffel
B.P. 81-33605 Pessac Cedex
Phone: 9-(56)-36-40-40
Telex: 550696

Datadis S.A.
10-12 Rue Emile Landrin
92100 Boulogne
Phone: 9-1-6056000
Telex: 20195

Dimel
Le Marino
Ave. Claude Farrere
83000 Toulon
Phone: 94/414963
Telex: 490093

Generim
24 Avenue De la Hoville Blanche
B.P. 1-38170 Seyssinet
Phone: 1-(76)-49-14-49
Telex: 320000

Generim S.A.R.L.
Zone d'Activities de Courtaboeuf
Avenue de la Baltique
P.O. Box 88
91943 Les Ulis Cedex
Phone: 1-9077878
Telex: 691700

GERMANY

Monolithic Memories, GmbH
Mauerkircherstr 4
8000 Munich 80
Phone: 89-984961
Telex: 524385
Fax: 89-983162

GERMANY (continued)

Astronic GmbH
Winzerstrasse 47D
8000 Munich 40
Phone: 304011
Telex: 5216187

Dr. Dohrenberg Vertriess GmbH
Bayreuther Strabe 3
1000 Berlin 30
Phone: 030-2138043-45
Telex: 0184860

Electronic 2000 Vertriebs GmbH
Neumarkter Strasse 75
8000 Munich 80
Phone: 89-434061
Telex: 522561

Nordelektronik GmbH KG
Harksheiderweg 238-240
2085 Quickborn
Phone: 04106-4031
Telex: 214299

Positron Bauelemente Vertriebs GmbH
Benzstrasse 1
Postfach 1140
7016 Gerlingen-Stuttgart
Phone: 07 156-23051
Telex: 7245266

INDIA

Kryonix
Kowdiar
Trivandrum
PIN 695 003
India
Phone: 63805
Telex: 884-307

Micro Aids International
790 Lucerne Dr.
Suite 40
Sunnyvale, CA 94086

ISRAEL

Telsys Ltd.
12 Kehilat Venetsia St.
Tel Aviv
Phone: (3)494891-5
Telex: 032392

ITALY

Comprel S.R.L.
Viale Romagna 1
20092 Cinisello Balsamo/Milano
Phone: 2-6120641
Telex: 332484

JAPAN

Internix Inc.
Shinjuku Hamad Bldg.
7-4-7 Nishi-Shinjuku
Shinjuku-Ku
Tokyo 160
Phone: (03) 369-1101
Telex: J26733

JAPAN, Cont'd.

K. Tokiwa & co.
Asahi-Seimei-Omori Bldg.
1-1-10 Omori-Kita
Shinagawa-Ku
Tokyo 143
Phone: (03) 766-6701
Telex: 246-6821
Fax: (03) 766-1300

Synerdyne Inc.
Ishibashi Bldg.
1-20-2 Dogenzaka
Shibuya-Ku
Tokyo 150
Phone: (03) 461-9311
Telex: J32457

KOREA

Duck Woo Trading Company
K.P.O. Box 570
Seoul, Korea
Phone: 725-1330
Telex: MOCNDM K23231

NETHERLANDS

Manudax Nederland BV
5473 ZG Meerstraat 7
Heeswijk (N.B.) Holland
Phone: 04139-2901
Telex: 50175

NORWAY

Henaco A/S
P.O. Box 126 Kaldbakken
Trondheimsveien 436 Ammerud
Oslo 9
Phone: 2-162110
Telex: 76716

SINGAPORE

Dynamar International Ltd.
Unit 05-11,
12 Lo Rong Bakar Batu
Kolam Ayer Industrial Estate
Singapore 1334
Phone: 7476188
Telex: RS26283

SOUTH AFRICA

Promilect Pty Ltd.
P.O. Box 56310
Pinegowrie 2123
Phone: 789-1400
Telex: 424822

SOUTH AMERICA

Intectra
2629 Terminal Blvd
Mountain View, CA 94043
Phone: (415) 967-8818
Telex: 910-345545

SPAIN

Sagitron
C/Castello, 25, 2
Madrid 1
Phone: (1)4026085
Telex: 43819

SWEDEN

Naxab
Box 4115
S 17104 Solna
Phone: 8-985140
Telex: 17912

SWITZERLAND

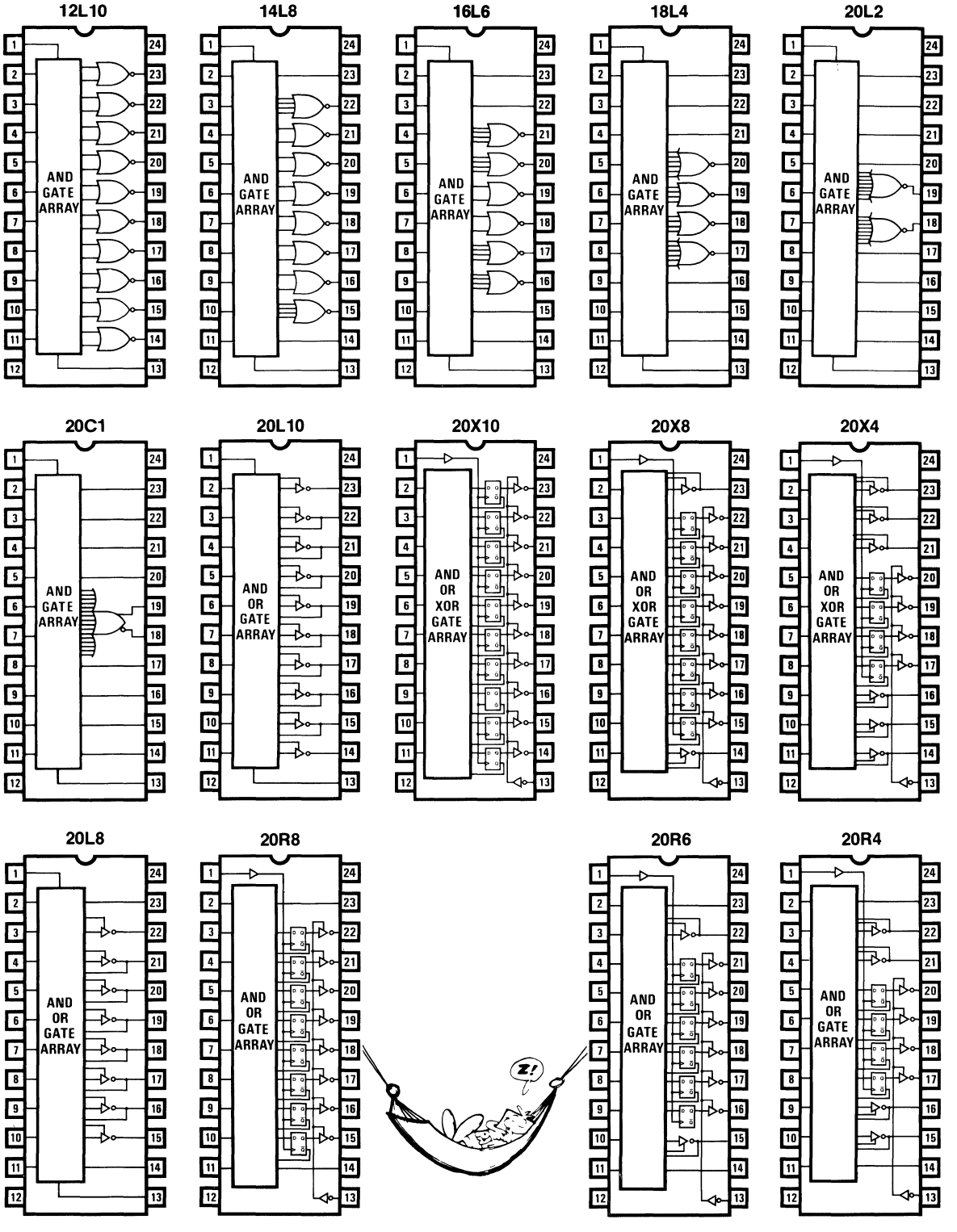
Industrade AG
Gemsenstrasse 2
CH 8021 Zurich
Phone: 01-3632230
Telex: 56788

TAIWAN

Multitech International Corp.
2nd Floor
977 Min Shen East Road
Taipei, Taiwan R.O.C.
Phone: (2)7691225
Telex: 23756 or 19162


IMPORTANT NOTES

24-Pin PAL Logic Symbols



Corporate Headquarters

AMERICAS
Monolithic Memories
2175 Mission College Blvd.
Santa Clara, CA 95050
Phone (408) 970-9700
Telex (910) 338-2374
Telex (910) 338-2376

FRANCE
Monolithic Memories France S.A.R.L.
Silic 463
F 94613 Rungis Cedex
France
Phone 1-6874500
Telex 202146
Fax 1-6876825

JAPAN
Monolithic Memories Japan KK
1-21-14 Nakamachi
Machida-Shi
Tokyo 194
Japan
Phone 427-29-3401
Telex 2872-220
Fax 427-29-5589

UNITED KINGDOM
Monolithic Memories, Ltd.
Lynwood House
1 Camp Road
Farnborough, Hampshire
England GU146EN
Phone 9-011-44-252-511099
Telex 858051 MONO UKG
Fax (0252) 43724

GERMANY
Monolithic Memories, GmbH
Mauerkircherstr 4
D 8000 Munich 80
West Germany
Phone 89-984961
Telex 524385
Fax 89-983162

Monolithic Memories MMI

Printed in U.S.A. 1983