

PLS159A primer

AN15

Issued June 1988; Revised October 1990

INTRODUCTION

The PLS159A is a programmable logic sequencer which consists of four dedicated inputs, four bidirectional I/O's, eight flip-flops, thirty two 16-input AND gates, twenty 32-input OR gates, and a complement array. Each flip-flop has a bidirectional I/O and may be individually programmed as J-K or D flip-flop, or switch between the two types dynamically. The flip-flops will accept data from the internal logic array or from the bidirectional I/O, or they may be set or reset asynchronously from the AND array. The output polarity of the four bidirectional I/O's are programmable and the direction is controlled by the AND array. Figure 1 is the logic diagram of PLS159A.

PROGRAMMING THE PLS159A

The programming table is shown in Table 1 where there is a place for everything that is

shown in Figure 1. The program table is basically divided into two main sections. The left hand side of the table, section A, represents the input side of the AND gates, while the right hand side, section B, represents the OR gates sections which includes the flip-flops and the combinatorial outputs B(0) to B(3). The flip-flops modes are defined in section C and the output polarities of the combinatorial outputs are defined in section E. The programming symbols are detailed in Figure 2.

As shown in Table 1, the programming table is very similar to a truth table. Each column in section A represents an input to the 32 AND gates, and each row represents an AND gate connecting to 17 inputs. Columns I_0 to I_3 represent the 4 dedicated inputs, I_0 to I_3 . Columns B(I)₀ to B(I)₃ represent the inputs of the 4 bidirectional I/O, B₀ to B₃. Columns Q(P)₀ to Q(P)₇ represent the feedback, F₀ to

F₇, from the flip-flops (the present state). Column "C" represents the complement array.

As shown in Figure 1, the outputs of the AND gates are connected to an array of OR gates which, in turn, are connected to either flip-flops or output circuits. Columns Q(N)₀ to Q(N)₇ represent the next state which the flip-flops will be in. Columns B(O)₀ to B(O)₃ represent the combinatorial outputs B₀ to B₃.

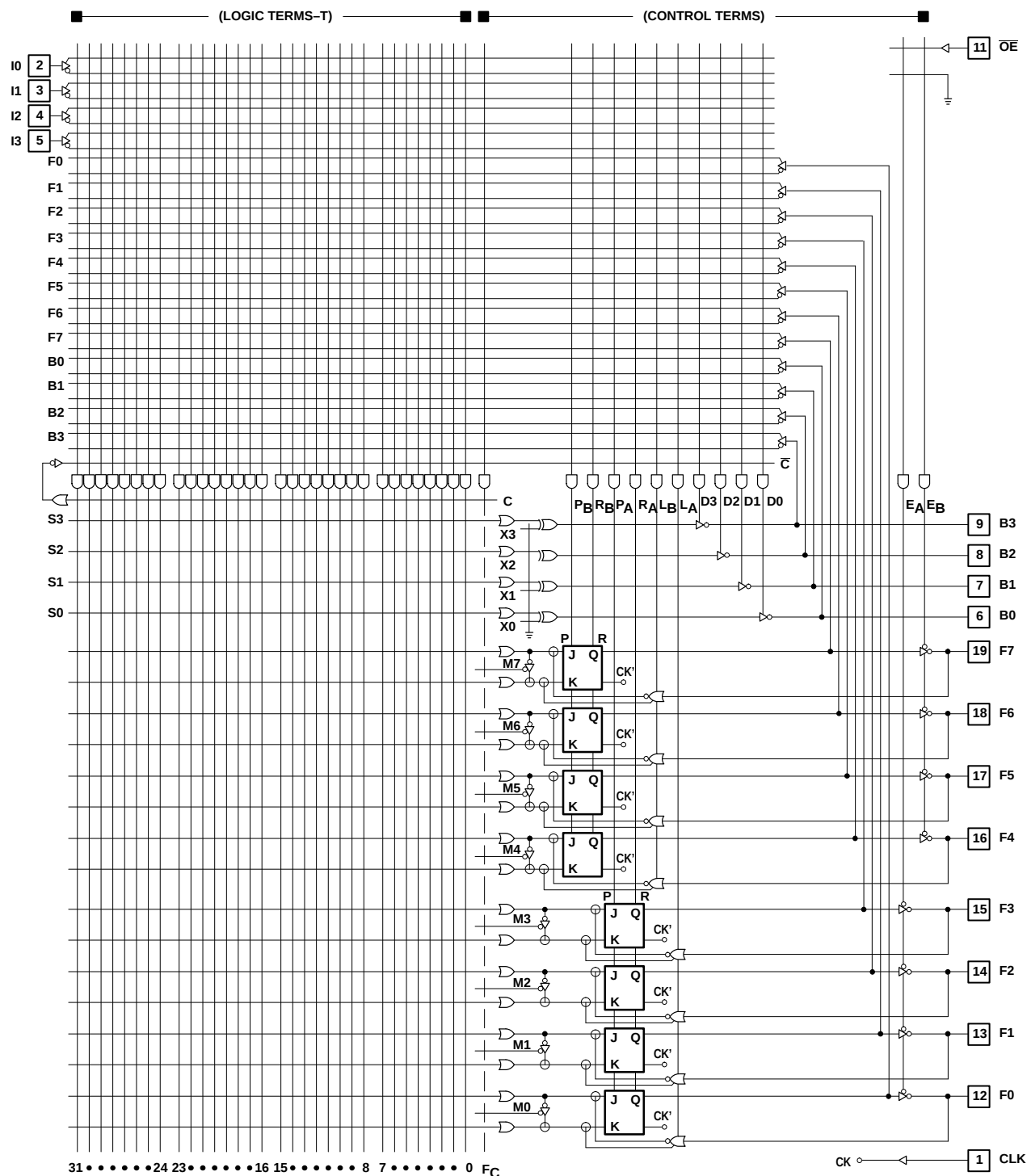
Each row represents an AND gate with 17 inputs each of which may be true and/or complement and is, therefore, a perfect decoder. Referring to the programming symbols in Figure 2, to implement the equation

$$Z = A * B * C * D,$$

all one has to do is to enter one line as shown in Table 2, term-0.

PLS159A primer

AN15

**NOTES:**

1. All OR gate inputs with a blown link float to logic "0".
2. All other gates and control inputs with a blown link float to logic "1".
3. ⊕ denotes WIRE-OR.
4. Programmable connection.

Figure 1. PLS Logic Diagram

PLS159A primer

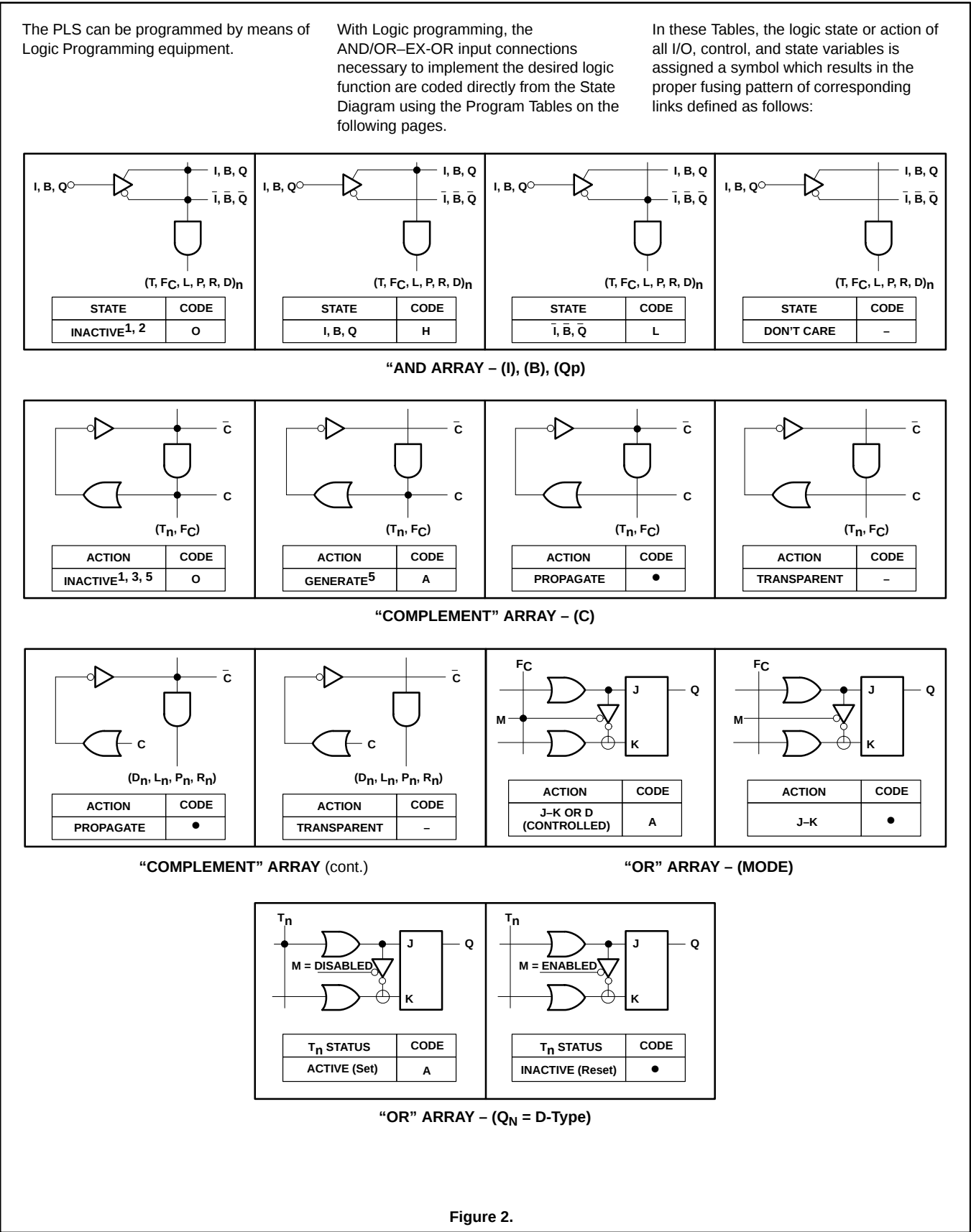
AN15

Table 1. PLS Program Table

[illegible]

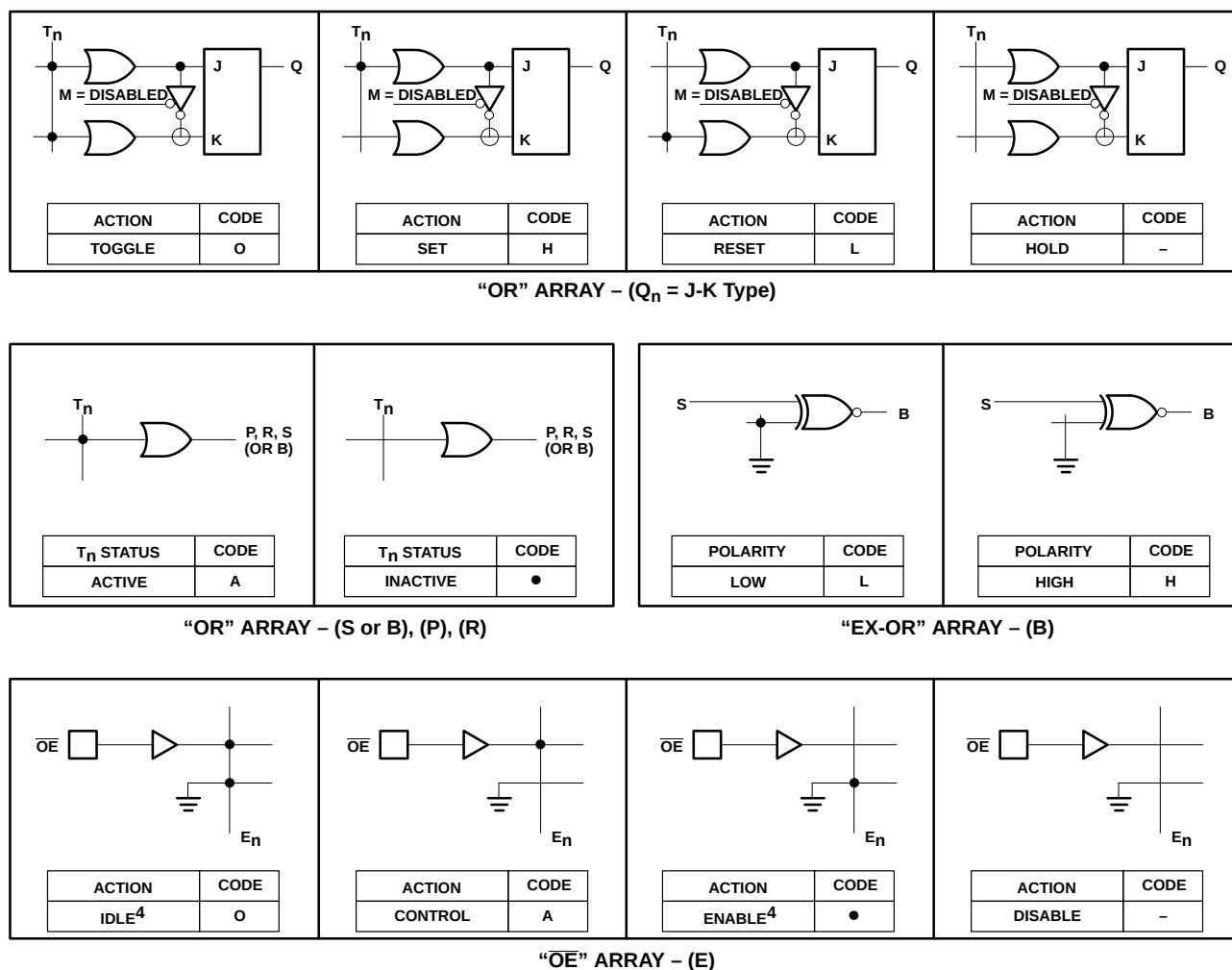
PLS159A primer

AN15



PLS159A primer

AN15



NOTES:

1. This is the initial unprogrammed state of all link pairs. It is normally associated with all unused (inactive) AND gates.
2. Any gate $(T, F_C, L, P, R, D)_n$ will be unconditionally inhibited if any one of the I, B, or Q link pairs is left intact.
3. To prevent oscillations, this state is not allowed for C link pairs coupled to active gates T_n, F_C .
4. $E_n = 0$ and $E_n = \bullet$ are logically equivalent states, since both cause F_n outputs to be unconditionally enabled.
5. These states are not allowed for control gates $(L, P, R, D)_n$ due to their lack of "OR" array links.

Figure 2 (cont.)

PLS159A primer

AN15

Table 2. PLS Program Table

[illegible]

PLS159A primer

AN15

Notice that only I_0 to I_3 on the left hand side and $B(O)_4$ on the right hand side have entries to implement the equation. All unused columns are dashed out or dotted out.

To implement the equation

$$Y = A * B * /C,$$

enter one line as shown in Table 2, term-2 where the entry "H" represents the non-inverting input buffer while the entry "L" represents the inverting buffer. To have the AND gate to be unconditionally "High", dash out all the inputs of that particular AND gate as shown in Table 2, term-4. The virgin condition of the device, as shipped from the factory, has all connections intact, which means that the inverting and the non-inverting buffers of the same inputs are connected together. Such connection will cause the AND gate to be unconditionally "Low" as shown in Table 2, terms 6 and 7. The unconditional High and Low states are normally useful only internally and seldom brought out to the output pins.

To implement the equation

$$W = A * /B + C * /D,$$

enter one line for $A * /B$ and another line for $C * /D$ as shown in Table 2, terms 9 and 10. Use one line to AND something together; use different lines to OR something together — one line per item to be OR'ed.

All the pins which are labelled B's are bidirectional I/O pins. Their input buffers are represented by the B(I) columns on the left hand side of the programming table. An "H" entry represents the non-inverting buffer and an "L" entry represents the inverting buffer. Their output buffers are represented by the B(O) columns on the right hand side of the table. An "A" entry means that the output is active (connected to the AND gates); a "."

entry means that the output is inactive (not connected). The outputs may be programmed to be inverting or noninverting. The polarity of each output is determined by its exclusive OR gate (Figure 1 and Figure 2). To have a non-inverting output, enter an "H" in the section labelled "POLARITY" (Table 1, Section E). To have an inverting output, enter an "L". For example, Table 3, terms -0 and -2 implement the equation

$$Z = /(A * B) \text{ and } Y = A * B$$

respectively. The above two equations may also be implemented by term-4 which uses the same AND gate to drive two OR gates.

Besides being able to have programmable Active-High or Active-Low output, the programmable output polarity feature also allows the user to minimize his AND term utilization by converting his logic equation into other forms such as conversion by De Morgan Theorem.

For example, the equation

$$X = A + B + C + D$$

takes four AND terms to implement as shown in Table 3, terms 6 to 9. By using De Morgan Theorem, the same equation is changed to

$$/W = /A * /B * /C * /D$$

The result is as shown in term 11 — a saving of three AND terms. The output buffers are disabled in their virgin states so that they all behave as inputs. The buffers are enabled or disabled by their corresponding Control AND terms D_0 to D_3 (see Figure 1). The Control AND terms are represented in the programming table on the last four rows on the left hand side. Dashing out all the inputs will cause the output buffer to be unconditionally enabled, whereas a "0" (zero)

will cause the buffer to be unconditionally disabled. The buffers may also be controlled by a logical condition, e.g. $A * /B * /C$, etc.

There are eight flip-flops on the chip each of which may be programmed as a J/K or a D flip-flop, or they may be programmed to switch dynamically. As shown in Figure 1, each flip-flop is a J/K to begin with. A 3-State inverter is connected in between the J and K inputs of each flip-flop, which when enabled by the AND gate F_C , will cause the flip-flop to function as a D flip-flop. The inverters are enabled by F_C through fuses M_0 to M_7 . A "." in the F/F Mode entry of the programming table means that particular fuse is to be disconnected and that particular flip-flop is to be J/K. An "A" entry will leave the M fuses intact, which allow the flip-flop to be D or J/K as controlled by the output of F_C (see Figure 2, "OR" ARRAY — (MODE)). The inputs to the flip-flops are represented by the programming table as the next state, $Q(N)_0$ to $Q(N)_7$ since their inputs are from the OR array. The outputs of these registers are connected to their respective 3-State inverting output buffers, four of which are controlled by EA and the other four by EB. A "." in EA will enable outputs F_0 to F_3 , whereas a "-" will disable them. An "A" will allow the output buffers to be controlled by /OE, pin 11. Table 4, terms 0, 1 and 3 represent the following equations

$$Q_0: J = A * C + /B * /E \quad \text{eq. 1}$$

$$Q_0: K = A * /C \quad \text{eq. 2}$$

Notice that the J input in equation 1 is represented by the "H" entry in terms-0 and 1, column $Q(N)_0$ while the K input in equation 2 is represented by the "L" entry in term-3, column $Q(N)_0$. An undefined input, J or K, is considered "Low".

PLS159A primer

AN15

Table 3. PLS Program Table

[illegible]

PLS159A primer

AN15

Table 4. PLS Program Table

CODE NO.										F/F MODE							REMARKS	EB				EA				POLARITY				REMARKS				
T E R M	AND																	REMARKS	(OR)															
	C	I				B(I)				Q(P)									Q(N)								B(O)							
3		2	1	0	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	3	2	1	0						
0	—	H	—	H	—	—	—	—	—	—	—	—	—	—	—	—	A*C	—	—	—	—	—	—	—	H					J = A*C+/B*/E				
1	—	—	L	—	L	—	—	—	—	—	—	—	—	—	—	—	/B*/E	—	—	—	—	—	—	—	H									
2																																		
3	—	H	—	L	—	—	—	—	—	—	—	—	—	—	—	—	A*/C	—	—	—	—	—	—	—	L					K = A*/C				
4																																		
5	—	L	L	L	—	—	—	—	—	—	—	—	—	—	—	—	/A*/B*/C	•	•	•	•	•	•	A	•					D = /A*/B*/C+E				
6	—	—	—	—	H	—	—	—	—	—	—	—	—	—	—	—	E	•	•	•	•	•	•	A	•									
7																																		
8	—	L	L	L	—	—	—	—	—	—	—	—	—	—	—	—	/A*/B*/C	—	—	—	—	—	—	H	—					D = /A*/B*/C+E				
9	—	—	—	—	H	—	—	—	—	—	—	—	—	—	—	—	E	—	—	—	—	—	—	H	—									
10																																		
11	—	—	—	—	—	—	—	—	—	—	—	H	—	—	—	—		—	—	O	—	—	—	—						T = I				
12	—	—	—	—	—	—	—	—	—	—	—	H	—	—	—	—		—	O	—	—	—	—							T = Q ₅				
13	—	—	—	—	—	—	—	—	—	—	H	H	—	—	—	—		O	—	—	—	—	—	—						T = Q ₅ *Q ₆				
14																																		
15																																		
16																																		
17																																		
18																																		
19																																		
20																																		
21																																		
22																																		
23																																		
24																																		
25																																		
26																																		
27																																		
28																																		
29																																		
30																																		
31																																		
FC	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—																		
P _B																																		
R _B																																		
L _B																																		
P _A																																		
R _A																																		

PLS159A

PLS159A primer

AN15

A D flip-flop may be implemented by first entering an "A" in F/F MODE. Then enter "0" in the row F_C , which will unconditionally enable the 3-State inverter between the J and K inputs. The following logic equation may be implemented as shown in Table 4, term 5

$$Q_1: D = \overline{A} * \overline{B} * C + E.$$

Notice that the entries in term 5, columns $Q(N)_{0 \text{ to } 7}$ are "A" and "." instead of "H" and "L" as in the case of J/K flip-flops. The entry "A" will cause the fuse connecting to the "K" input to be disconnected and the "J" fuse to be intact. Whereas the entry "." will cause both fuses to be disconnected. This feature enables the user to quickly recognize the mode in which the flip-flops are operating without having to go through the control terms. Some commercially available device programmers in the market may not have the software capability to implement this feature, in which case an "H" and a "—" may be used in place of "A" and "." respectively as shown in Table 4, terms 8 and 9.

Of course, the term F_C may have inputs instead of zeros and dashes, in which case the flip-flop modes are controlled dynamically.

When both the J and K inputs are "1's", the flip-flop will toggle. A simple 3-bit counter may be implemented using only AND terms as shown in Table 4 terms 11, 12 and 13. The logic equations for the three flip-flops are as the following:

$$Q_5: \quad T=1; \quad (Q_5 \text{ toggles unconditionally})$$

$$Q_6: \quad T=Q_5; \quad (Q_6 \text{ toggles when } Q_5=1)$$

$$Q_7: \quad T=Q_5 * Q_6; \quad (Q_7 \text{ toggles when } Q_5 * Q_6=1)$$

The above equations represent an octal up-counter. However, since the outputs of the flip-flops are inverted, the counting sequence of the outputs is that of a down-counter.

The flip-flops may be asynchronously set and reset by the Control AND terms PA/PB and RA/RB respectively. As shown in Figure 1, PA and RA controls flip-flops F_0 to F_3 , while PB and RB control F_4 to F_7 .

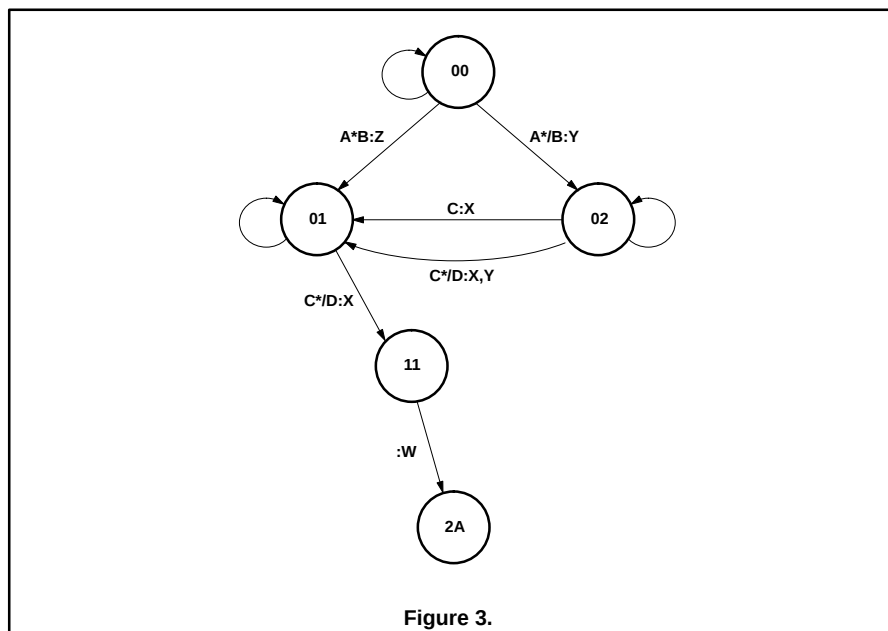


Figure 3.

In order to save the number of input pins, the eight flip-flops may be synchronously loaded directly from their own output pins. To use this feature, EA and/or EB must be programmed "A" or "—" so that the output buffers may be disabled before loading. As shown in Figure 1, every flip-flop has an OR/NOR gate the input of which is directly connected to the output pin and the outputs of the OR/NOR are connected to the K and J inputs respectively. This OR/NOR gate inverts the input and feeds it to the flip-flop in a "wire-OR" fashion. Therefore, when loading data directly into the flip-flops from the output pins, caution must be exercised to insure that the inputs from the OR array does not interfere with the data being loaded. For example, if the data being loaded is a "1" on the output pin, the J input will be a "0" and the K input will be a "1". If, at the same time, a "1"

is present at the J-input from the OR array, the flip-flop will see "1's" in both J and K inputs. It will toggle as a result. The OR/NOR gates are enabled by the Control AND terms LA and LB. LA controls flip-flops F_0 to F_3 and LB controls F_4 to F_7 .

All Control AND terms function and are programmed in the same manner as the other AND terms. The only difference is that the Control AND terms are not connected to the OR array.

The outputs of the flip-flops may be fed back into the AND array as the present state, $Q(P)$. The output of the AND array into the OR array and the inputs to the flip-flops is the next state, $Q(N)$. As an example, Figure 3 is a state machine implemented in a PLS159A as shown in Table 5, terms 0 to 6.

PLS159A primer

AN15

Table 5. PLS Program Table

[illegible]