AN511

PLD Replacement

Author: Sumit Mitra
Microchip Technology, Inc.

INTRODUCTION

PIC16C5X microcontrollers are ideal for implementing low cost combinational and sequential logic circuits that traditionally have been implemented using either numerous TTL gates or using programmable logic chips such as PLAs or EPLDs.

The PIC16C5X is a family of high-performance 8-bit microcontrollers from Microchip Technology Inc. It employs a Harvard architecture, (i.e., has a separate data bus [8-bit] and program bus [12-bit]). All instructions are single word and execute in one cycle except for program branches. The instruction cycle time is 200 ns at 20 MHz. PIC16C5X microcontrollers are ideal for PLD-type applications because:

- Very low cost
- Few external components required
- Fully programmable. PIC16C5X Microcontrollers are offered as One Time Programmable (OTP) EPROM devices.
- Available off the shelf from distributors
- Calibration in software for improved measurement accuracy
- Power savings using PIC16C5X's Sleep mode.
- PIC16C5X's output pins have large, current source/sink capability to drive LED's directly.

IMPLEMENTING A PLA

To implement a generic combinational logic function, we can simply emulate an AND-OR PLA in software. This requires the logic outputs to be described as sums of products. To describe our algorithm, we will use a simple 8-input, 8-output PLA with 24 product terms (Figure 1). We will further use the truth table in Figure 2 as the PLA function being implemented. In this example, only four inputs (A3:A0) are used and the other four inputs (A7:A4) are "don't care". On the output side, seven output pins (Y6:Y0) are used and Y7 is unused. To implement this PLA, the logic inputs A0,A1,...,A7 can be connected to PORTB pins RB0,RB1,...,RB7 respectively. The logic outputs Y0,Y1,...,Y7 will appear on PORTC pins RC0,RC1,...,RC7 respectively. PORTB is configured as input and PORTC will be configured as output. To evaluate each product term one XOR (exclusive .OR.) and one .AND. operation will be required. For example, to determine product term P3 = A3.A2.A1.A0, the expression to evaluate is:

(A7:A0 .XOR. `xxxx0011b`) .AND. `00001111b`).

The constant with which .XOR. is performed will be referred to as P3_x in our discussion. P3_x = `xxxx0011b` will ensure that if A3:A0 = `0011b`, then the least significant 4-bits of the result will be `0000b`. The .AND. constant, referred to here as P3_a (Product term 3, .AND. constant) basically eliminates the "don't care" inputs (here A7:A4) by masking them. Therefore, if the result of the .XOR.-.AND. operation is zero then P3 = 1 else P3 = 0. Once the Product terms are evaluated they are stored in four product registers Preg_0 to Preg_3. To determine an output term:

$$Y0 = P0 + P2 + P3 + P5 + P6 + P7 + P8 + P9 + P10 + P12 + P13 + P14$$

we need to evaluate the following expression:

(Preg_a .AND. OR_a0) .OR. (Preg_b .AND. OR_b0) .OR. (Preg_c .AND. OR_c0)

In our case the constant values to implement Y0 are as follows:

OR_a0 =	1	1	1	0	1	1	0	1
	P7	P6	P5		P3	P2		P0
OR_b0 =	1	1	0	1	0	1	1	1
OR_c0 =	0	0	0	0	0	0	0	0

For a larger number of inputs, outputs, or product terms, the evaluation will be more complex but follows the same principle. Appendix A shows the assembly code to implement this 8 input x 8 output x 24 Product PLA. This example optimizes speed as well as program memory requirements. Appendix B shows a slightly different implementation (only the EVAL_Y MACRO is different) that optimizes program memory usage over speed. Table 1 shows the time and resources required to implement different size PLAs.

© 1997 Microchip Technology Inc.

FIGURE 1: A SIMPLE PLA

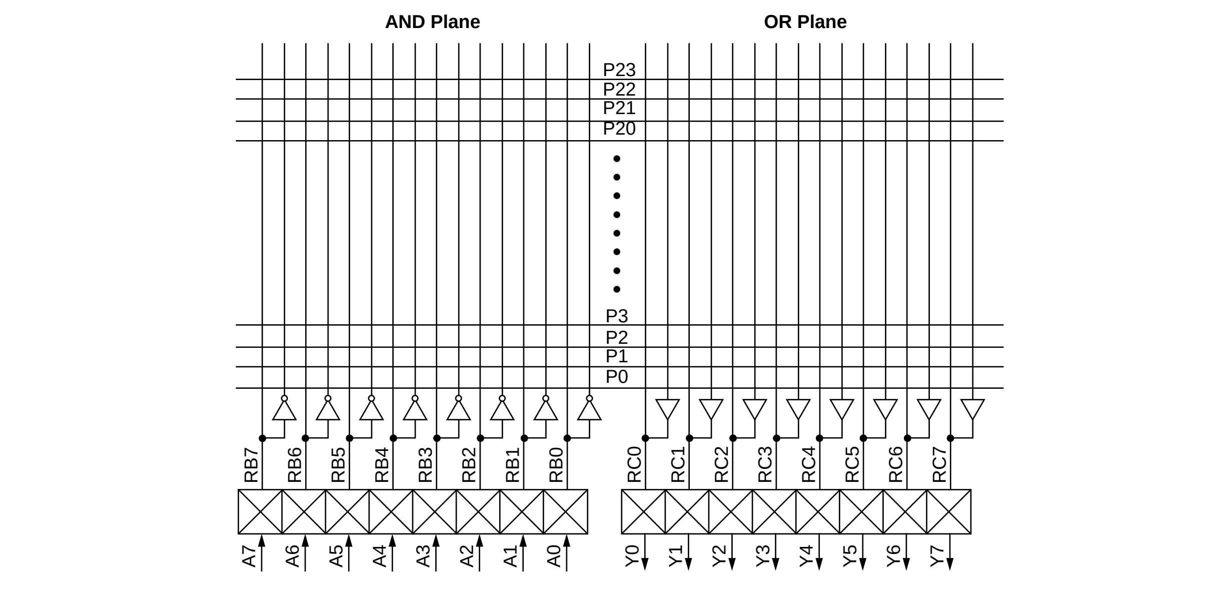
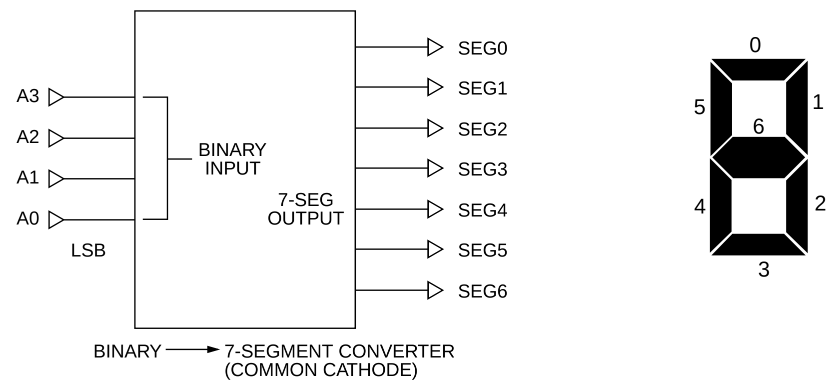


FIGURE 2: BINARY TO 7-SEGMENT CONVERSION EXAMPLE



Truth Table: Product Terms

HEX	A3	A2	A1	A0	Y0	Y1	Y2	Y3	Y4	Y5	Y6	
0	0	0	0	0	1	1	1	1	1	1	0	P0 = A3, A2, A1, A0
1	0	0	0	1	0	1	1	0	0	0	0	P1
2	0	0	1	0	1	1	0	1	1	0	1	P2
3	0	1	1	1	1	1	1	1	0	0	1	P3
4	0	1	0	0	0	1	1	0	0	1	1	P4
5	0	1	0	1	1	0	1	1	0	1	1	P5
6	0	1	1	0	1	0	1	1	1	1	1	P6
7	0	1	1	1	1	1	1	0	0	0	0	P7
8	1	0	0	0	1	1	1	1	1	1	1	P8
9	1	0	0	1	1	1	1	0	0	1	1	P9
A	1	0	1	0	1	1	1	0	1	1	1	P10
B	1	0	1	1	0	0	0	1	1	1	1	P11
C	1	1	0	0	1	0	0	1	1	1	0	P12
D	1	1	0	1	0	1	1	1	1	0	1	P13
E	1	1	1	0	1	0	0	1	1	1	1	P14
F	1	1	1	1	1	0	0	0	1	1	1	P15 = A3, A2, A1, A0

© 1997 Microchip Technology Inc.

SPEED/RESPONSE TIME

The worst case response time of a PLA implemented in this fashion can be calculated as follows. First, we define T_D = time required to execute the PLA program assuming the worst case program branches are taken. Then the maximum propagation delay time from input change to valid output = $2T_D$. This is because if an input changes just after the program reads the input port, its effect will not show up until the program completes the current execution cycle, re-reads the input and recalculates the output. This is shown in Figure 3. There are ways to improve the delay time, such as sample inputs several times throughout the PLA program and if an input change is sensed, return to the beginning rather than execute the rest of the evaluation code.

A Table Lookup Method for Small PLA Implementation

If the number of inputs is small (8 or less) then a simple table lookup method can be used to implement the PLA. This will improve execution time to around 5 µs (@ 8 MHz input clock). The following code implements the Binary Coded Decimal (BCD) to 7-segment conversion (Figure 2) using this technique.

EXAMPLE 1: LOOKUP TABLE – PLA METHOHD

```
begin   movlw  0ffh       ;
        tris    6         ;Port_b = input
        clrw              ;
        tris    7         ;Port_c = output
pla 88  movf   Port_b,w   ;Read input
        andlw  0fh        ;Mask off bits 7:4
        call   op_tbl     ;
        movwf  Port_c     ;Write output
        goto   pla88      ;
op_tbl  addwf  pc         ;Computed jump for
                          ;table lookup
        retlw  b”00111111”;
        retlw  b”00000110”;
        retlw  b”01011011”;
        retlw  b”01001111”;
        retlw  b”01100110”;
        retlw  b”01101101”;
        retlw  b”01111101”;
        retlw  b”00000111”;
        retlw  b”01111111”;
        retlw  b”01100111”;
        retlw  b”01110111”;
        retlw  b”01111000”;
        retlw  b”00111001”;
        retlw  b”01011110”;
        retlw  b”01111001”;
        retlw  b”01110001”;
```

TABLE 1: EXECUTION TIME AND RESOURCES NECESSARY FOR DIFFERENT SIZE PLAs

Number of Inputs Including F_{DBK}	Number of Outputs Including F_{DBK} and O/E Control	Number of Products	Number of RAM Locations Required N_{RAM}	Number of Memory Locations Required N_{ROM}	Number of Instruction Cycles to Execute PLA N_{CYC}	Real-time @ 20 MHz to Execute PLA	
8	8	24	5	228	228	45.6 µs	Time Efficient
			10	222	352	70.4 µs	Code Efficient
8	8	48	8	447	447	89.8 µs	Time Efficient
			13	384	535	107 µs	Code Efficient
16	16	64	12	1042	1042	208.4 µs	Time Efficient
			22	843	1250	250 µs	Code Efficient
20	24	80	16	1661	1661	372.2 µs	Time Efficient
			40	1462	2221	444.2 µs	Code Efficient

If
N_I = Number of inputs
N_{IW} = Number of input words, $N_{IW} = \frac{N_I}{8}$
N_P = Number of products
N_{PW} = Number of product words, i.e. $N_{PW} = \frac{N_P}{8}$
N_O = Number of output words
N_{OW} = Number of output words, i.e., $N_{OW} = \frac{N_O}{8}$

Then,
N_{RAM} @ $N_{IW} + N_{PW}$: Time Efficient
N_{RAM} @ $N_{IW} + N_{OW} + N_{PW} + 2$: Code Efficient
N_{ROM} @ $8 + N_{PW} + N_{OW} + N_P [2 + 3N_{IW}] + No \bullet 4$: Time Efficient
N_{ROM} @ $17 + N_{PW} + N_{OW} + N_P [2 + 3N_{IW}] + No [N_{PW} + 3]$: Code Efficient
N_{CYC} @ $8 + N_{PW} + N_{OW} + N_P[2 + 3N_{IW}] + No [2N_{PW} \bullet 4]$: Time Efficient
N_{CYC} @ $8 + N_{PW} + N_{OW} + N_P [2 + 3N_{IW}] + 5N [N_{PW} + 1]$: Code Efficient

© 1997 Microchip Technology Inc.

FIGURE 3: PLA PROGRAM FLOW

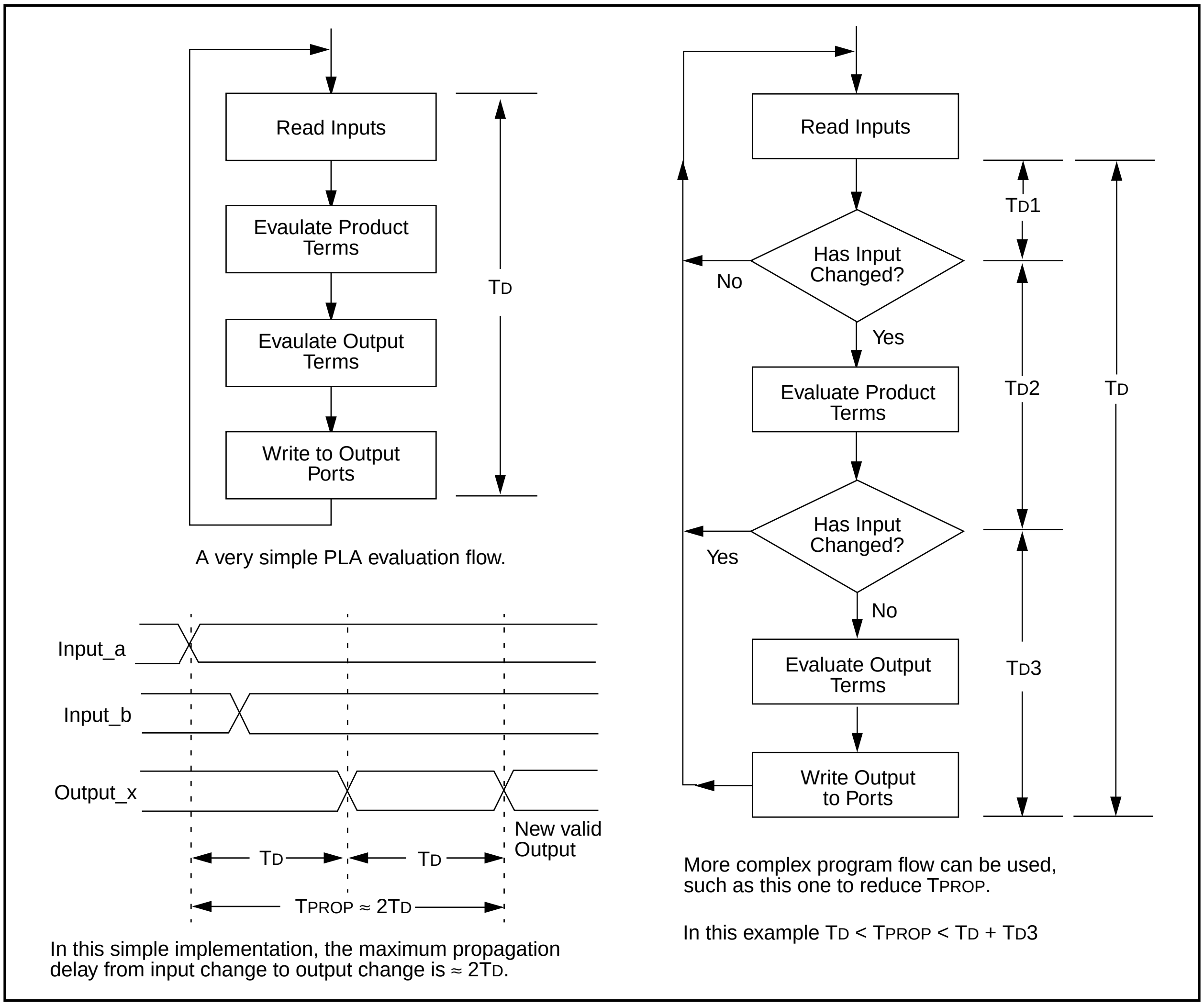
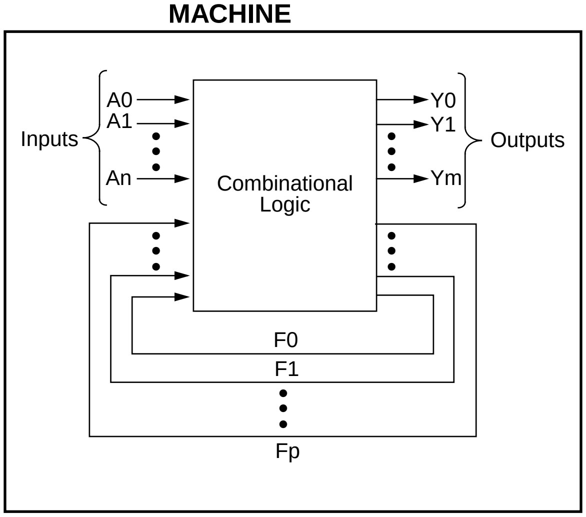


FIGURE 4: ASYNCHRONOUS STATE MACHINE

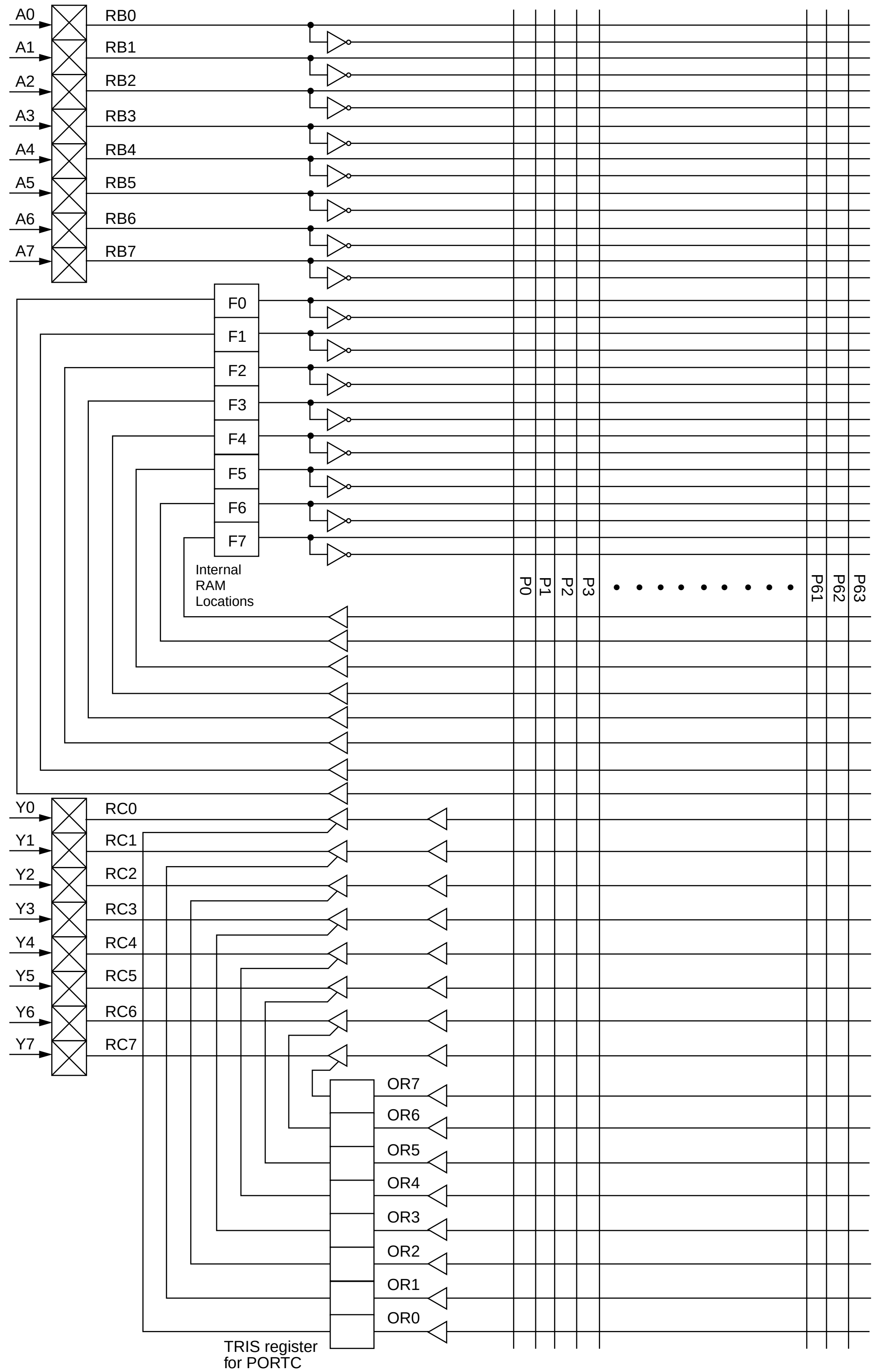


IMPLEMENTING AN ASYNCHRONOUS STATE MACHINE

The concept can be easily extended to implement sequential logic (i.e., a state machine). Figure 4 shows a state machine with n inputs (A0:An), m outputs (Y0:Ym) and p states that feedback as inputs to the PLA (F0:Fp). In a PIC16C5X the states will be stored as bits in RAM location. Input will now mean input from a port as well as from the feedback registers. Figure 5 shows an example PLA with eight inputs A0,A1,..,A7 that are connected to PORTB pins RB0,RB1,..,RB7. This example PLA has a total of 24 outputs of which eight are actual outputs, another eight are output enable control for the outputs and the other eight are feedbacks (or states). The PLA shown here, therefore, in essence implements an asynchronous state machine (i.e., there is no system clock).

© 1997 Microchip Technology Inc.

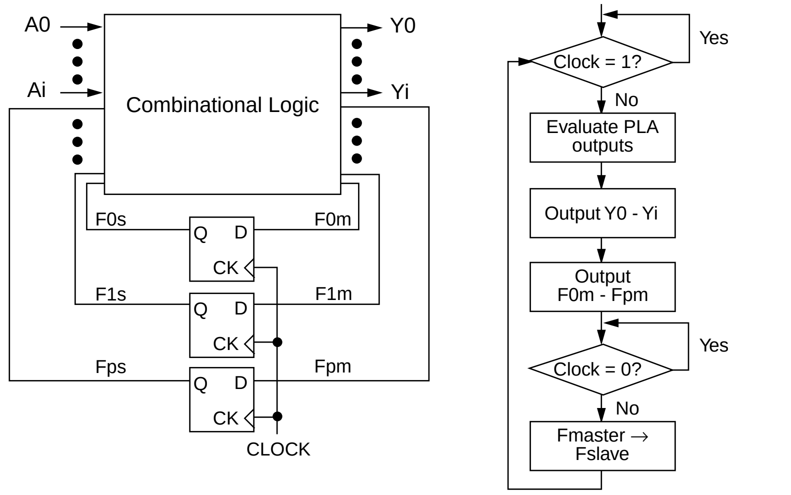
FIGURE 5: EXAMPLE OF A LARGER PLA IMPLEMENTATION



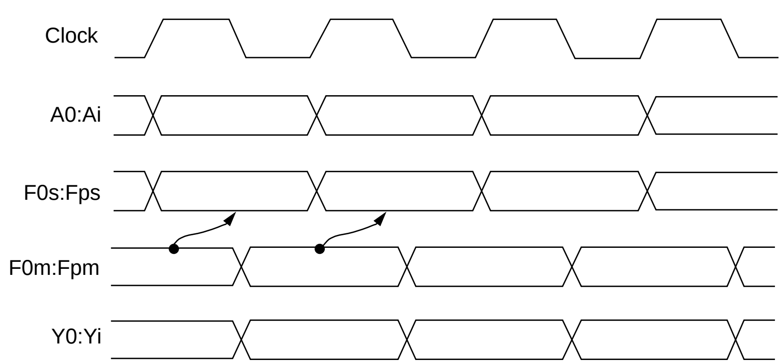
This example shows 16 inputs (including feedback), 64 product terms and 24 outputs including feedback and O/E control. This example demonstrates that O/E control is easily implementable using PIC16C5X's bi-directional ports with Tri-State® control.

Tri-State® is a registered trademark of National Semiconductor Corporation

FIGURE 6: SYNCHRONOUS STATE MACHINE IMPLEMENTATION



Flowchart for implementing synchronous state machine. Clock is input to an I/O port. Fmaster and Fslave are RAM locations that store state variables F0m, ... Fpm and F0s, ... Fps respectively.



IMPLEMENTING A SYNCHRONOUS STATE MACHINE

In a synchronous system (Figure 6) usually all inputs are stable at the falling (or rising) edge of the system clock. The state machine samples inputs on the falling edge, then evaluates state and output information. The state outputs are latched by the rising edge of the clock before feeding them back to the input (so that they are stable at the falling edge of the clock). To implement such a state machine, the system clock will have to be polled by an input pin. When a falling edge is detected, the PLA evaluation procedure will be invoked to compute outputs and write them to output pins. The PLA procedure will also determine the new state variables, F0m, F1m,...,Fpm and store them in RAM. The program will then wait until a rising edge on the clock input is detected and copy the “master” state variables (F0m,..,Fpm) to slave state variables (F0s,F1s,..,Fps). This step emulates the feedback flip-flops.

SUMMARY

In conclusion, the PIC16C5X can implement a generic PLA equation and provide a quick, low cost solution where system operation speed is not critical.

© 1997 Microchip Technology Inc.

Please check the Microchip BBS for the latest version of the source code. Microchip's Worldwide Web Address: www.microchip.com; Bulletin Board Support: MCHIPBBS using CompuServe® (CompuServe membership not required).

APPENDIX A: PLA IMPLEMENTATION: TIME EFFICIENT APPROACH

```
MPASM 01.40 Released            PLA1A.ASM   1-16-1997  12:28:25         PAGE  1


LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                    00001         LIST    P = 16C54, n = 66
                    00002 ;
                    00003 ;*******************************************************************
                    00004 ; pla1a.asm :
                    00005 ; This procedure implements a simple AND-OR PLA with:
                    00006 ;
                    00007 ;      8  inputs        := A7 A6 A5 A4 A3 A2 A1 A0
                    00008 ;      24 product terms := P23 P22 ..... P0
                    00009 ;      8  outputs       := Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0
                    00010 ;
                    00011 ; The eight inputs are assumed to be connected to PORT RB such that
                    00012 ; RB0 = A0, RB1 = A1, ... , RB7 = A7.
                    00013 ; The outputs are programmed to appear on port RC such that
                    00014 ; RC0 = Y0, RC1 = Y1, ... , RC7 = Y7.
                    00015 ;
                    00016 ; This implementation optimizes both speed & program memory usage
                    00017 ;
                    00018 ;       Program:          PLA1A.ASM
                    00019 ;       Revision Date:
                    00020 ;                         1-13-97   Compatibility with MPASMWIN 1.40
                    00021 ;
                    00022 ;********************************************************************
                    00023 ;
                    00024 ; define RAM locations used:
                    00025 ;
  0000000C          00026 input   equ  d'12'   ; RAM location 12 holds input
  0000000D          00027 Y_reg   equ  d'13'   ; holds output result
                    00028
  0000000E          00029 Preg_a  equ  d'14'   ; Product terms P0 to P7. Preg_a<0> = P0
  0000000F          00030 Preg_b  equ  d'15'   ; Product terms P8 to P15. Preg_b<0> = P8
  00000010          00031 Preg_c  equ  d'16'   ; Product terms P16 to P23. Preg_c<0> = P16
                    00032
                    00033 ; define some constants and file addresses:
                    00034 ;
  00000000          00035 bit0    equ  0        ;
  00000001          00036 bit1    equ  1        ;
  00000002          00037 bit2    equ  2        ;
  00000003          00038 bit3    equ  3        ;
  00000004          00039 bit4    equ  4        ;
  00000005          00040 bit5    equ  5        ;
  00000006          00041 bit6    equ  6        ;
  00000007          00042 bit7    equ  7        ;
                    00043 ;
  00000003          00044 status  equ  3        ;
  00000006          00045 port_b  equ  6        ;
  00000007          00046 port_c  equ  7        ;
                    00047 ;
                    00048 ; define the AND plane programming variables:
                    00049 ;
  00000000          00050 P0_x    equ  b'00000000'     ;
  0000000F          00051 P0_a    equ  b'00001111'     ;
  00000001          00052 P1_x    equ  b'00000001'     ;
  0000000F          00053 P1_a    equ  b'00001111'     ;
  00000002          00054 P2_x    equ  b'00000010'     ;
```

```
0000000F          P2_a    equ  b'00001111'     ;
00000003          P3_x    equ  b'00000011'     ;
0000000F          P3_a    equ  b'00001111'     ;
00000004          P4_x    equ  b'00000100'     ;
0000000F          P4_a    equ  b'00001111'     ;
00000005          P5_x    equ  b'00000101'     ;
0000000F          P5_a    equ  b'00001111'     ;
00000006          P6_x    equ  b'00000110'     ;
0000000F          P6_a    equ  b'00001111'     ;
00000007          P7_x    equ  b'00000111'     ;
0000000F          P7_a    equ  b'00001111'     ;
00000008          P8_x    equ  b'00001000'     ;
0000000F          P8_a    equ  b'00001111'     ;
00000009          P9_x    equ  b'00001001'     ;
0000000F          P9_a    equ  b'00001111'     ;
0000000A          P10_x   equ  b'00001010'     ;
0000000F          P10_a   equ  b'00001111'     ;
0000000B          P11_x   equ  b'00001011'     ;
0000000F          P11_a   equ  b'00001111'     ;
0000000C          P12_x   equ  b'00001100'     ;
0000000F          P12_a   equ  b'00001111'     ;
0000000D          P13_x   equ  b'00001101'     ;
0000000F          P13_a   equ  b'00001111'     ;
0000000E          P14_x   equ  b'00001110'     ;
0000000F          P14_a   equ  b'00001111'     ;
0000000F          P15_x   equ  b'00001111'     ;
0000000F          P15_a   equ  b'00001111'     ;
00000000          P16_x   equ  b'00000000'     ;
00000000          P16_a   equ  b'00000000'     ;
00000000          P17_x   equ  b'00000000'     ;
00000000          P17_a   equ  b'00000000'     ;
00000000          P18_x   equ  b'00000000'     ;
00000000          P18_a   equ  b'00000000'     ;
00000000          P19_x   equ  b'00000000'     ;
00000000          P19_a   equ  b'00000000'     ;
00000000          P20_x   equ  b'00000000'     ;
00000000          P20_a   equ  b'00000000'     ;
00000000          P21_x   equ  b'00000000'     ;
00000000          P21_a   equ  b'00000000'     ;
00000000          P22_x   equ  b'00000000'     ;
00000000          P22_a   equ  b'00000000'     ;
00000000          P23_x   equ  b'00000000'     ;
00000000          P23_a   equ  b'00000000'     ;

                  ; define OR plane programming variables:

000000ED          OR_a0   equ  b'11101101'     ; for output Y0
000000D7          OR_b0   equ  b'11010111'     ;
00000000          OR_c0   equ  b'00000000'     ;
0000009F          OR_a1   equ  b'10011111'     ; for output Y1
00000027          OR_b1   equ  b'00100111'     ;
00000000          OR_c1   equ  b'00000000'     ;
000000FB          OR_a2   equ  b'11111011'     ; for output Y2
0000002F          OR_b2   equ  b'00101111'     ;
00000000          OR_c2   equ  b'00000000'     ;
0000006D          OR_a3   equ  b'01101101'     ; for output Y3
00000079          OR_b3   equ  b'01111001'     ;
00000000          OR_c3   equ  b'00000000'     ;
00000045          OR_a4   equ  b'01000101'     ; for output Y4
000000FD          OR_b4   equ  b'11111101'     ;
00000000          OR_c4   equ  b'00000000'     ;
00000071          OR_a5   equ  b'01110001'     ; for output Y5
000000DF          OR_b5   equ  b'11011111'     ;
00000000          OR_c5   equ  b'00000000'     ;
0000007C          OR_a6   equ  b'01111100'     ; for output Y6
000000EF          OR_b6   equ  b'11101111'     ;
```


© 1997 Microchip Technology Inc.

```
  00000000        OR_c6   equ  b'00000000'     ;
  00000000        OR_a7   equ  b'00000000'     ; for output Y7
  00000000        OR_b7   equ  b'00000000'     ;
  00000000        OR_c7   equ  b'00000000'     ;
01FF                      org  01ffh                      ;
01FF 0A00         begin   goto main                       ;
0000                              org     000h            ;
                  ; define macro to evaluate 1 product (AND) term:
                  ;
0000 0902         main    call  pla88                     ;
0001 0A00                 goto  main                      ;
                  ;
                  EVAL_P  MACRO   Preg_x,bit_n,Pn_x,Pn_a
                                  movf    input,W         ;
                                  xorlw   Pn_x            ;
                                  andlw   Pn_a            ;
                                  btfsc   status,bit2     ; skip if zero bit not set
                                  bsf     Preg_x,bit_n    ; product term = 1
                                  ENDM
                  ; define macro to load OR term constants:
                  ;
                  EVAL_Y  MACRO   OR_an,OR_bn,OR_cn,bit_n
                                  LOCAL   SETBIT  ;
                                  movf    Preg_a,W        ;
                                  andlw   OR_an   ;
                                  btfss   status,bit2     ;
                                  goto    SETBIT          ;
                                  movf    Preg_b,W        ;
                                  andlw   OR_bn   ;
                                  btfss   status,bit2     ;
                                  goto    SETBIT  ;
                                  movf    Preg_c,W        ;
                                  andlw   OR_cn   ;
                                  btfss   status,bit2     ;
                  SETBIT  bsf             Y_reg,bit_n     ;
                                  ENDM
                  ; now the  PLA evaluation procedure:
                  ;
0002 0CFF         pla88   movlw   0ffh                    ;
0003 0006                         tris    6               ; port_b = input
0004 0206                         movf    port_b,W        ; read input
0005 002C                         movwf   input           ; store input in a register
0006 006E                         clrf    Preg_a          ; clear Product register a
0007 006F                         clrf    Preg_b          ; clear Product register b
0008 0070                         clrf    Preg_c          ; clear Product register c
0009 006D                         clrf    Y_reg           ; clear output register
                  and_pl  EVAL_P  Preg_a,bit0,P0_x,P0_a
000A 020C             M           movf    input,W         ;
000B 0F00             M           xorlw   P0_x            ;
000C 0E0F             M           andlw   P0_a            ;
000D 0643             M           btfsc   status,bit2     ; skip if zero bit not set
000E 050E             M           bsf     Preg_a,bit0     ; product term = 1
                                  EVAL_P  Preg_a,bit1,P1_x,P1_a
000F 020C             M           movf    input,W         ;
0010 0F01             M           xorlw   P1_x            ;
0011 0E0F             M           andlw   P1_a            ;
```

```
0012 0643           M                btfsc   status,bit2     ; skip if zero bit not set
0013 052E           M                bsf     Preg_a,bit1     ; product term = 1
                00179                EVAL_P  Preg_a,bit2,P2_x,P2_a
0014 020C           M                movf    input,W         ;
0015 0F02           M                xorlw   P2_x            ;
0016 0E0F           M                andlw   P2_a            ;
0017 0643           M                btfsc   status,bit2     ; skip if zero bit not set
0018 054E           M                bsf     Preg_a,bit2     ; product term = 1
                00180                EVAL_P  Preg_a,bit3,P3_x,P3_a
0019 020C           M                movf    input,W         ;
001A 0F03           M                xorlw   P3_x            ;
001B 0E0F           M                andlw   P3_a            ;
001C 0643           M                btfsc   status,bit2     ; skip if zero bit not set
001D 056E           M                bsf     Preg_a,bit3     ; product term = 1
                00181                EVAL_P  Preg_a,bit4,P4_x,P4_a
001E 020C           M                movf    input,W         ;
001F 0F04           M                xorlw   P4_x            ;
0020 0E0F           M                andlw   P4_a            ;
0021 0643           M                btfsc   status,bit2     ; skip if zero bit not set
0022 058E           M                bsf     Preg_a,bit4     ; product term = 1
                00182                EVAL_P  Preg_a,bit5,P5_x,P5_a
0023 020C           M                movf    input,W         ;
0024 0F05           M                xorlw   P5_x            ;
0025 0E0F           M                andlw   P5_a            ;
0026 0643           M                btfsc   status,bit2     ; skip if zero bit not set
0027 05AE           M                bsf     Preg_a,bit5     ; product term = 1
                00183                EVAL_P  Preg_a,bit6,P6_x,P6_a
0028 020C           M                movf    input,W         ;
0029 0F06           M                xorlw   P6_x            ;
002A 0E0F           M                andlw   P6_a            ;
002B 0643           M                btfsc   status,bit2     ; skip if zero bit not set
002C 05CE           M                bsf     Preg_a,bit6     ; product term = 1
                00184                EVAL_P  Preg_a,bit7,P7_x,P7_a
002D 020C           M                movf    input,W         ;
002E 0F07           M                xorlw   P7_x            ;
002F 0E0F           M                andlw   P7_a            ;
0030 0643           M                btfsc   status,bit2     ; skip if zero bit not set
0031 05EE           M                bsf     Preg_a,bit7     ; product term = 1
                00185
                00186                EVAL_P  Preg_b,bit0,P8_x,P8_a
0032 020C           M                movf    input,W         ;
0033 0F08           M                xorlw   P8_x            ;
0034 0E0F           M                andlw   P8_a            ;
0035 0643           M                btfsc   status,bit2     ; skip if zero bit not set
0036 050F           M                bsf     Preg_b,bit0     ; product term = 1
                00187                EVAL_P  Preg_b,bit1,P9_x,P9_a
0037 020C           M                movf    input,W         ;
0038 0F09           M                xorlw   P9_x            ;
0039 0E0F           M                andlw   P9_a            ;
003A 0643           M                btfsc   status,bit2     ; skip if zero bit not set
003B 052F           M                bsf     Preg_b,bit1     ; product term = 1
                00188                EVAL_P  Preg_b,bit2,P10_x,P10_a
003C 020C           M                movf    input,W         ;
003D 0F0A           M                xorlw   P10_x           ;
003E 0E0F           M                andlw   P10_a           ;
003F 0643           M                btfsc   status,bit2     ; skip if zero bit not set
0040 054F           M                bsf     Preg_b,bit2     ; product term = 1
                00189                EVAL_P  Preg_b,bit3,P11_x,P11_a
0041 020C           M                movf    input,W         ;
0042 0F0B           M                xorlw   P11_x           ;
0043 0E0F           M                andlw   P11_a           ;
0044 0643           M                btfsc   status,bit2     ; skip if zero bit not set
0045 056F           M                bsf     Preg_b,bit3     ; product term = 1
                00190                EVAL_P  Preg_b,bit4,P12_x,P12_a
0046 020C           M                movf    input,W         ;
0047 0F0C           M                xorlw   P12_x           ;
```


© 1997 Microchip Technology Inc.

```
0048 0E0F              M         andlw   P12_a           ;
0049 0643              M         btfsc   status,bit2     ; skip if zero bit not set
004A 058F              M         bsf     Preg_b,bit4     ; product term = 1
                   00191         EVAL_P  Preg_b,bit5,P13_x,P13_a
004B 020C              M         movf    input,W         ;
004C 0F0D              M         xorlw   P13_x           ;
004D 0E0F              M         andlw   P13_a           ;
004E 0643              M         btfsc   status,bit2     ; skip if zero bit not set
004F 05AF              M         bsf     Preg_b,bit5     ; product term = 1
                   00192         EVAL_P  Preg_b,bit6,P14_x,P14_a
0050 020C              M         movf    input,W         ;
0051 0F0E              M         xorlw   P14_x           ;
0052 0E0F              M         andlw   P14_a           ;
0053 0643              M         btfsc   status,bit2     ; skip if zero bit not set
0054 05CF              M         bsf     Preg_b,bit6     ; product term = 1
                   00193         EVAL_P  Preg_b,bit7,P15_x,P15_a
0055 020C              M         movf    input,W         ;
0056 0F0F              M         xorlw   P15_x           ;
0057 0E0F              M         andlw   P15_a           ;
0058 0643              M         btfsc   status,bit2     ; skip if zero bit not set
0059 05EF              M         bsf     Preg_b,bit7     ; product term = 1
                   00194
                   00195         EVAL_P  Preg_c,bit0,P16_x,P16_a
005A 020C              M         movf    input,W         ;
005B 0F00              M         xorlw   P16_x           ;
005C 0E00              M         andlw   P16_a           ;
005D 0643              M         btfsc   status,bit2     ; skip if zero bit not set
005E 0510              M         bsf     Preg_c,bit0     ; product term = 1
                   00196         EVAL_P  Preg_c,bit1,P17_x,P17_a
005F 020C              M         movf    input,W         ;
0060 0F00              M         xorlw   P17_x           ;
0061 0E00              M         andlw   P17_a           ;
0062 0643              M         btfsc   status,bit2     ; skip if zero bit not set
0063 0530              M         bsf     Preg_c,bit1     ; product term = 1
                   00197         EVAL_P  Preg_c,bit2,P18_x,P18_a
0064 020C              M         movf    input,W         ;
0065 0F00              M         xorlw   P18_x           ;
0066 0E00              M         andlw   P18_a           ;
0067 0643              M         btfsc   status,bit2     ; skip if zero bit not set
0068 0550              M         bsf     Preg_c,bit2     ; product term = 1
                   00198         EVAL_P  Preg_c,bit3,P19_x,P19_a
0069 020C              M         movf    input,W         ;
006A 0F00              M         xorlw   P19_x           ;
006B 0E00              M         andlw   P19_a           ;
006C 0643              M         btfsc   status,bit2     ; skip if zero bit not set
006D 0570              M         bsf     Preg_c,bit3     ; product term = 1
                   00199         EVAL_P  Preg_c,bit4,P20_x,P20_a
006E 020C              M         movf    input,W         ;
006F 0F00              M         xorlw   P20_x           ;
0070 0E00              M         andlw   P20_a           ;
0071 0643              M         btfsc   status,bit2     ; skip if zero bit not set
0072 0590              M         bsf     Preg_c,bit4     ; product term = 1
                   00200         EVAL_P  Preg_c,bit5,P21_x,P21_a
0073 020C              M         movf    input,W         ;
0074 0F00              M         xorlw   P21_x           ;
0075 0E00              M         andlw   P21_a           ;
0076 0643              M         btfsc   status,bit2     ; skip if zero bit not set
0077 05B0              M         bsf     Preg_c,bit5     ; product term = 1
                   00201         EVAL_P  Preg_c,bit6,P22_x,P22_a
0078 020C              M         movf    input,W         ;
0079 0F00              M         xorlw   P22_x           ;
007A 0E00              M         andlw   P22_a           ;
007B 0643              M         btfsc   status,bit2     ; skip if zero bit not set
007C 05D0              M         bsf     Preg_c,bit6     ; product term = 1
                   00202         EVAL_P  Preg_c,bit7,P23_x,P23_a
007D 020C              M         movf    input,W         ;
```

```
007E 0F00              M                  xorlw   P23_x          ;
007F 0E00              M                  andlw   P23_a          ;
0080 0643              M                  btfsc   status,bit2    ; skip if zero bit not set
0081 05F0              M                  bsf     Preg_c,bit7    ; product term = 1
                   00203
                   00204 or_pl            EVAL_Y  OR_a0,OR_b0,OR_c0,bit0
  0000                 M                  LOCAL   SETBIT         ;
0082 020E              M                  movf    Preg_a,W       ;
0083 0EED              M                  andlw   OR_a0          ;
0084 0743              M                  btfss   status,bit2    ;
0085 0A8D              M                  goto    SETBIT         ;
                       M
0086 020F              M                  movf    Preg_b,W       ;
0087 0ED7              M                  andlw   OR_b0   ;
0088 0743              M                  btfss   status,bit2    ;
0089 0A8D              M                  goto    SETBIT         ;
                       M
008A 0210              M                  movf    Preg_c,W       ;
008B 0E00              M                  andlw   OR_c0          ;
008C 0743              M                  btfss   status,bit2    ;
008D 050D              M SETBIT           bsf     Y_reg,bit0     ;
                   00205                  EVAL_Y  OR_a1,OR_b1,OR_c1,bit1
  0000                 M                  LOCAL   SETBIT         ;
008E 020E              M                  movf    Preg_a,W       ;
008F 0E9F              M                  andlw   OR_a1          ;
0090 0743              M                  btfss   status,bit2    ;
0091 0A99              M                  goto    SETBIT         ;
                       M
0092 020F              M                  movf    Preg_b,W       ;
0093 0E27              M                  andlw   OR_b1          ;
0094 0743              M                  btfss   status,bit2    ;
0095 0A99              M                  goto    SETBIT         ;
                       M
0096 0210              M                  movf    Preg_c,W       ;
0097 0E00              M                  andlw   OR_c1          ;
0098 0743              M                  btfss   status,bit2    ;
0099 052D              M SETBIT           bsf     Y_reg,bit1     ;
                   00206                  EVAL_Y  OR_a2,OR_b2,OR_c2,bit2
  0000                 M                  LOCAL   SETBIT         ;
009A 020E              M                  movf    Preg_a,W       ;
009B 0EFB              M                  andlw   OR_a2          ;
009C 0743              M                  btfss   status,bit2    ;
009D 0AA5              M                  goto    SETBIT         ;
                       M
009E 020F              M                  movf    Preg_b,W       ;
009F 0E2F              M                  andlw   OR_b2          ;
00A0 0743              M                  btfss   status,bit2    ;
00A1 0AA5              M                  goto    SETBIT  ;
                       M
00A2 0210              M                  movf    Preg_c,W       ;
00A3 0E00              M                  andlw   OR_c2          ;
00A4 0743              M                  btfss   status,bit2    ;
00A5 054D              M SETBIT           bsf     Y_reg,bit2     ;
                   00207                  EVAL_Y  OR_a3,OR_b3,OR_c3,bit3
  0000                 M                  LOCAL   SETBIT         ;
00A6 020E              M                  movf    Preg_a,W       ;
00A7 0E6D              M                  andlw   OR_a3          ;
00A8 0743              M                  btfss   status,bit2    ;
00A9 0AB1              M                  goto    SETBIT         ;
                       M
00AA 020F              M                  movf    Preg_b,W       ;
00AB 0E79              M                  andlw   OR_b3          ;
00AC 0743              M                  btfss   status,bit2    ;
00AD 0AB1              M                  goto    SETBIT         ;
                       M
00AE 0210              M                  movf    Preg_c,W       ;
```

 © 1997 Microchip Technology Inc.

```
00AF 0E00              M                 andlw   OR_c3           ;
00B0 0743              M                 btfss   status,bit2     ;
00B1 056D              M SETBIT          bsf     Y_reg,bit3      ;
                   00208                 EVAL_Y  OR_a4,OR_b4,OR_c4,bit4
  0000                 M                 LOCAL   SETBIT          ;
00B2 020E              M                 movf    Preg_a,W        ;
00B3 0E45              M                 andlw   OR_a4           ;
00B4 0743              M                 btfss   status,bit2     ;
00B5 0ABD              M                 goto    SETBIT          ;
                       M
00B6 020F              M                 movf    Preg_b,W        ;
00B7 0EFD              M                 andlw   OR_b4           ;
00B8 0743              M                 btfss   status,bit2     ;
00B9 0ABD              M                 goto    SETBIT          ;
                       M
00BA 0210              M                 movf    Preg_c,W        ;
00BB 0E00              M                 andlw   OR_c4           ;
00BC 0743              M                 btfss   status,bit2     ;
00BD 058D              M SETBIT          bsf     Y_reg,bit4      ;
                   00209                 EVAL_Y  OR_a5,OR_b5,OR_c5,bit5
  0000                 M                 LOCAL   SETBIT          ;
00BE 020E              M                 movf    Preg_a,W        ;
00BF 0E71              M                 andlw   OR_a5           ;
00C0 0743              M                 btfss   status,bit2     ;
00C1 0AC9              M                 goto    SETBIT          ;
                       M
Preg_b,W        ;
00C3 0EDF              M                 andlw   OR_b5           ;
00C4 0743              M                 btfss   status,bit2     ;
00C5 0AC9              M                 goto    SETBIT          ;
                       M
00C6 0210              M                 movf    Preg_c,W        ;
00C7 0E00              M                 andlw   OR_c5           ;
00C8 0743              M                 btfss   status,bit2     ;
00C9 05AD              M SETBIT          bsf     Y_reg,bit5      ;
                   00210                 EVAL_Y  OR_a6,OR_b6,OR_c6,bit6
  0000                 M                 LOCAL   SETBIT          ;
00CA 020E              M                 movf    Preg_a,W        ;
00CB 0E7C              M                 andlw   OR_a6           ;
00CC 0743              M                 btfss   status,bit2     ;
00CD 0AD5              M                 goto    SETBIT          ;
                       M
00CE 020F              M                 movf    Preg_b,W        ;
00CF 0EEF              M                 andlw   OR_b6           ;
00D0 0743              M                 btfss   status,bit2     ;
00D1 0AD5              M                 goto    SETBIT          ;
                       M
00D2 0210              M                 movf    Preg_c,W        ;
00D3 0E00              M                 andlw   OR_c6           ;
00D4 0743              M                 btfss   status,bit2     ;
00D5 05CD              M SETBIT  bsf             Y_reg,bit6      ;
                   00211                 EVAL_Y  OR_a7,OR_b7,OR_c7,bit7
  0000                 M                 LOCAL   SETBIT          ;
00D6 020E              M                 movf    Preg_a,W        ;
00D7 0E00              M                 andlw   OR_a7           ;
00D8 0743              M                 btfss   status,bit2     ;
00D9 0AE1              M                 goto    SETBIT          ;
                       M
00DA 020F              M                 movf    Preg_b,W        ;
00DB 0E00              M                 andlw   OR_b7           ;
00DC 0743              M                 btfss   status,bit2     ;
00DD 0AE1              M                 goto    SETBIT          ;
                       M
00DE 0210              M                 movf    Preg_c,W        ;
00DF 0E00              M                 andlw   OR_c7           ;
00E0 0743              M                 btfss   status,bit2     ;
```

```
00E1 05ED                M SETBIT  bsf             Y_reg,bit7      ;
                    00212
                    00213 ; Y_reg now contains 8 output values:
00E2 0040           00214 wr_out  clrw                             ;
00E3 0007           00215                 tris    7                ; port_c = output
00E4 020D           00216                 movf    Y_reg,W          ;
00E5 0027           00217                 movwf   port_c           ; Y_reg -> port_c
00E6 0800           00218                 retlw   0                ;
                    00219
00E7 0000           00220 ZZZ             nop
                    00221
                    00222         END
```

```
MEMORY USAGE MAP ('X' = Used,  '-' = Unused)

0000 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0080 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
00C0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXX-------- ----------------
01C0 : ---------------- ---------------- ---------------- ---------------X

All other memory blocks unused.

Program Memory Words Used:   233
Program Memory Words Free:   279


Errors   :     0
Warnings :     0 reported,     0 suppressed
Messages :     0 reported,     0 suppressed
```


© 1997 Microchip Technology Inc.

Please check the Microchip BBS for the latest version of the source code. Microchip's Worldwide Web Address: www.microchip.com; Bulletin Board Support: MCHIPBBS using CompuServe® (CompuServe membership not required).

APPENDIX B: PLA IMPLEMENTATION: CODE EFFICIENT APPROACH

```
MPASM 01.40 Released            PLA1B.ASM   1-16-1997  12:29:06        PAGE  1

LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                    00001         LIST    P = 16C54, n = 66
                    00002 ;
                    00003 ;*******************************************************************
                    00004 ; pla1b.asm :
                    00005 ; This procedure implements a simple AND-OR PLA with:
                    00006 ;
                    00007 ;      8  inputs        := A7 A6 A5 A4 A3 A2 A1 A0
                    00008 ;      24 product terms := P23 P22 ..... P0
                    00009 ;      8  outputs       := Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0
                    00010 ;
                    00011 ; The eight inputs are assumed to be connected to PORT RB such that
                    00012 ; RB0 = A0, RB1 = A1, ... , RB7 = A7.
                    00013 ; The outputs are programmed to appear on port RC such that
                    00014 ; RC0 = Y0, RC1 = Y1, ... , RC7 = Y7.
                    00015 ;
                    00016 ; This implementation optimizes program memory usage over
                    00017 ; speed
                    00018 ;
                    00019 ;       Program:          PLA1B.ASM
                    00020 ;       Revision Date:
                    00021 ;                         1-13-97      Compatibility with MPASMWIN 1.40
                    00022 ;
                    00023 ;*******************************************************************
                    00024 ;
                    00025 ; define RAM locations used:
                    00026 ;
  0000000C          00027 input   equ     d'12'   ; RAM location 12 holds input
  0000000D          00028 Y_reg   equ     d'13'   ; holds output result
                    00029
  0000000E          00030 Preg_a  equ     d'14'   ; Product terms P0 to P7. Preg_a<0> = P0
  0000000F          00031 Preg_b  equ     d'15'   ; Product terms P8 to P15. Preg_b<0> = P8
  00000010          00032 Preg_c  equ     d'16'   ; Product terms P16 to P23. Preg_c<0> = P16
                    00033
  00000012          00034 Pn_x    equ     d'18'   ;
  00000013          00035 Pn_a    equ     d'19'   ;
  00000014          00036 OR_a    equ     d'20'   ;
  00000015          00037 OR_b    equ     d'21'   ;
  00000016          00038 OR_c    equ     d'22'   ;
                    00039
                    00040 ; define some constants and file addresses:
                    00041 ;
  00000000          00042 bit0    equ     0       ;
  00000001          00043 bit1    equ     1       ;
  00000002          00044 bit2    equ     2       ;
  00000003          00045 bit3    equ     3       ;
  00000004          00046 bit4    equ     4       ;
  00000005          00047 bit5    equ     5       ;
  00000006          00048 bit6    equ     6       ;
  00000007          00049 bit7    equ     7       ;
                    00050 ;
  00000003          00051 status  equ           ;
  00000006          00052 port_b  equ     6       ;
  00000007          00053 port_c  equ     7       ;
                    00054 ;
                    00055 ; define the AND plane programming variables:
```

© 1997 Microchip Technology Inc.

```
                  ;
00000000          P0_x    equ     b'00000000'     ;
0000000F          P0_a    equ     b'00001111'     ;
00000001          P1_x    equ     b'00000001'     ;
0000000F          P1_a    equ     b'00001111'     ;
00000002          P2_x    equ     b'00000010'     ;
0000000F          P2_a    equ     b'00001111'     ;
00000003          P3_x    equ     b'00000011'     ;
0000000F          P3_a    equ     b'00001111'     ;
00000004          P4_x    equ     b'00000100'     ;
0000000F          P4_a    equ     b'00001111'     ;
00000005          P5_x    equ     b'00000101'     ;
0000000F          P5_a    equ     b'00001111'     ;
00000006          P6_x    equ     b'00000110'     ;
0000000F          P6_a    equ     b'00001111'     ;
00000007          P7_x    equ     b'00000111'     ;
0000000F          P7_a    equ     b'00001111'     ;
00000008          P8_x    equ     b'00001000'     ;
0000000F          P8_a    equ     b'00001111'     ;
00000009          P9_x    equ     b'00001001'     ;
0000000F          P9_a    equ     b'00001111'     ;
0000000A          P10_x   equ     b'00001010'     ;
0000000F          P10_a   equ     b'00001111'     ;
0000000B          P11_x   equ     b'00001011'     ;
0000000F          P11_a   equ     b'00001111'     ;
0000000C          P12_x   equ     b'00001100'     ;
0000000F          P12_a   equ     b'00001111'     ;
0000000D          P13_x   equ     b'00001101'     ;
0000000F          P13_a   equ     b'00001111'     ;
0000000E          P14_x   equ     b'00001110'     ;
0000000F          P14_a   equ     b'00001111'     ;
0000000F          P15_x   equ     b'00001111'     ;
0000000F          P15_a   equ     b'00001111'     ;
00000000          P16_x   equ     b'00000000'     ;
00000000          P16_a   equ     b'00000000'     ;
00000000          P17_x   equ     b'00000000'     ;
00000000          P17_a   equ     b'00000000'     ;
00000000          P18_x   equ     b'00000000'     ;
00000000          P18_a   equ     b'00000000'     ;
00000000          P19_x   equ     b'00000000'     ;
00000000          P19_a   equ     b'00000000'     ;
00000000          P20_x   equ     b'00000000'     ;
00000000          P20_a   equ     b'00000000'     ;
00000000          P21_x   equ     b'00000000'     ;
00000000          P21_a   equ     b'00000000'     ;
00000000          P22_x   equ     b'00000000'     ;
00000000          P22_a   equ     b'00000000'     ;
00000000          P23_x   equ     b'00000000'     ;
00000000          P23_a   equ     b'00000000'     ;

                  ; define OR plane programming variables:

000000ED          OR_a0   equ     b'11101101'     ; for output Y0
000000D7          OR_b0   equ     b'11010111'     ;
00000000          OR_c0   equ     b'00000000'     ;
0000009F          OR_a1   equ     b'10011111'     ; for output Y1
00000027          OR_b1   equ     b'00100111'     ;
00000000          OR_c1   equ     b'00000000'     ;
000000FB          OR_a2   equ     b'11111011'     ; for output Y2
0000002F          OR_b2   equ     b'00101111'     ;
00000000          OR_c2   equ     b'00000000'     ;
0000006D          OR_a3   equ     b'01101101'     ; for output Y3
00000079          OR_b3   equ     b'01111001'     ;
00000000          OR_c3   equ     b'00000000'     ;
00000045          OR_a4   equ     b'01000101'     ; for output Y4
000000FD          OR_b4   equ     b'11111101'     ;
```

 © 1997 Microchip Technology Inc.

```
  00000000          00122 OR_c4   equ     b'00000000'             ;
  00000071          00123 OR_a5   equ     b'01110001'             ; for output Y5
  000000DF          00124 OR_b5   equ     b'11011111'             ;
  00000000          00125 OR_c5   equ     b'00000000'             ;
  0000007C          00126 OR_a6   equ     b'01111100'             ; for output Y6
  000000EF          00127 OR_b6   equ     b'11101111'             ;
  00000000          00128 OR_c6   equ     b'00000000'             ;
  00000000          00129 OR_a7   equ     b'00000000'             ; for output Y7
  00000000          00130 OR_b7   equ     b'00000000'             ;
  00000000          00131 OR_c7   equ     b'00000000'             ;
                    00132
                    00133
01FF                00134                 org     01ffh           ;
01FF 0A00           00135 begin   goto    main                    ;
                    00136
0000                00137                 org     000h            ;
                    00138 ; define macro to evaluate 1 product (AND) term:
                    00139 ;
0000 090B           00140 main            call    pla88           ;
0001 0A00           00141                 goto    main            ;
                    00142 ;
                    00143
                    00144 EVAL_P  MACRO   Preg_x,bit_n,Pn_x,Pn_a
                    00145                 movf    input,W         ;
                    00146                 xorlw   Pn_x            ;
                    00147                 andlw   Pn_a            ;
                    00148                 btfsc   status,bit2     ; skip if zero bit not set
                    00149                 bsf     Preg_x,bit_n    ; product term = 1
                    00150                 ENDM
                    00151
                    00152
                    00153 ; define macro to load OR term constants:
                    00154 ;
                    00155 EVAL_Y  MACRO   OR_an,OR_bn,OR_cn,bit_n
                    00156                 movlw   OR_an   ; load constants
                    00157                 movwf   OR_a            ;
                    00158                 movlw   OR_bn           ;
                    00159                 movwf   OR_b            ;
                    00160                 movlw   OR_cn           ;
                    00161                 movwf   OR_c            ;
                    00162                 call    EVAL1           ;
                    00163                 btfss   status,bit2     ;
                    00164                 bsf     Y_reg,bit_n     ;
                    00165                 ENDM
                    00166
                    00167 ; define procedure to evaluate 1 output (OR) term:
                    00168 ;
0002 020E           00169 EVAL1   movf            Preg_a,W        ;
0003 0174           00170                 andwf   OR_a,1          ;
                    00171
0004 020F           00172                 movf    Preg_b,W        ;
0005 0175           00173                 andwf   OR_b,1          ;
                    00174
0006 0210           00175                 movf    Preg_c,W        ;
0007 0156           00176                 andwf   OR_c,W          ;
                    00177
0008 0114           00178                 iorwf   OR_a,W          ;
0009 0115           00179                 iorwf   OR_b,W          ;
000A 0800           00180                 retlw   0               ;  W = 1 implies Yn = 1
                    00181
                    00182 ; now the  PLA evaluation procedure:
                    00183 ;
000B 0CFF           00184 pla88   movlw   0ffh                    ;
000C 0006           00185                 tris    6               ; port_b = input
000D 0206           00186                 movf    port_b,W        ; read input
000E 002C           00187                 movwf   input           ; store input in a register
```

```
000F 006E          00188                clrf    Preg_a          ; clear Product register a
0010 006F          00189                clrf    Preg_b          ; clear Product register b
0011 0070          00190                clrf    Preg_c          ; clear Product register c
0012 006D          00191                clrf    Y_reg           ; clear output register
                   00192
                   00193 and_pl  EVAL_P  Preg_a,bit0,P0_x,P0_a
0013 020C              M                movf    input,W         ;
0014 0F00              M                xorlw   P0_x            ;
0015 0E0F              M                andlw   P0_a            ;
0016 0643              M                btfsc   status,bit2     ; skip if zero bit not set
0017 050E              M                bsf     Preg_a,bit0     ; product term = 1
                   00194                EVAL_P  Preg_a,bit1,P1_x,P1_a
0018 020C              M                movf    input,W         ;
0019 0F01              M                xorlw   P1_x            ;
001A 0E0F              M                andlw   P1_a            ;
001B 0643              M                btfsc   status,bit2     ; skip if zero bit not set
001C 052E              M                bsf     Preg_a,bit1     ; product term = 1
                   00195                EVAL_P  Preg_a,bit2,P2_x,P2_a
001D 020C              M                movf    input,W         ;
001E 0F02              M                xorlw   P2_x            ;
001F 0E0F              M                andlw   P2_a            ;
0020 0643              M                btfsc   status,bit2     ; skip if zero bit not set
0021 054E              M                bsf     Preg_a,bit2     ; product term = 1
                   00196                EVAL_P  Preg_a,bit3,P3_x,P3_a
0022 020C              M                movf    input,W         ;
0023 0F03              M                xorlw   P3_x            ;
0024 0E0F              M                andlw   P3_a            ;
0025 0643              M                btfsc   status,bit2     ; skip if zero bit not set
0026 056E              M                bsf     Preg_a,bit3     ; product term = 1
                   00197                EVAL_P  Preg_a,bit4,P4_x,P4_a
0027 020C              M                movf    input,W         ;
0028 0F04              M                xorlw   P4_x            ;
0029 0E0F              M                andlw   P4_a            ;
002A 0643              M                btfsc   status,bit2     ; skip if zero bit not set
002B 058E              M                bsf     Preg_a,bit4     ; product term = 1
                   00198                EVAL_P  Preg_a,bit5,P5_x,P5_a
002C 020C              M                movf    input,W         ;
002D 0F05              M                xorlw   P5_x            ;
002E 0E0F              M                andlw   P5_a            ;
002F 0643              M                btfsc   status,bit2     ; skip if zero bit not set
0030 05AE              M                bsf     Preg_a,bit5     ; product term = 1
                   00199                EVAL_P  Preg_a,bit6,P6_x,P6_a
0031 020C              M                movf    input,W         ;
0032 0F06              M                xorlw   P6_x            ;
0033 0E0F              M                andlw   P6_a            ;
0034 0643              M                btfsc   status,bit2     ; skip if zero bit not set
0035 05CE              M                bsf     Preg_a,bit6     ; product term = 1
                   00200                EVAL_P  Preg_a,bit7,P7_x,P7_a
0036 020C              M                movf    input,W         ;
0037 0F07              M                xorlw   P7_x            ;
0038 0E0F              M                andlw   P7_a            ;
0039 0643              M                btfsc   status,bit2     ; skip if zero bit not set
003A 05EE              M                bsf     Preg_a,bit7     ; product term = 1
                   00201
                   00202                EVAL_P  Preg_b,bit0,P8_x,P8_a
003B 020C              M                movf    input,W         ;
003C 0F08              M                xorlw   P8_x            ;
003D 0E0F              M                andlw   P8_a            ;
003E 0643              M                btfsc   status,bit2     ; skip if zero bit not set
003F 050F              M                bsf     Preg_b,bit0     ; product term = 1
                   00203                EVAL_P  Preg_b,bit1,P9_x,P9_a
0040 020C              M                movf    input,W         ;
0041 0F09              M                xorlw   P9_x            ;
0042 0E0F              M                andlw   P9_a            ;
0043 0643              M                btfsc   status,bit2     ; skip if zero bit not set
0044 052F              M                bsf     Preg_b,bit1     ; product term = 1
```


© 1997 Microchip Technology Inc.

```
                    00204              EVAL_P  Preg_b,bit2,P10_x,P10_a
0045 020C               M              movf    input,W        ;
0046 0F0A               M              xorlw   P10_x          ;
0047 0E0F               M              andlw   P10_a          ;
0048 0643               M              btfsc   status,bit2    ; skip if zero bit not set
0049 054F               M              bsf     Preg_b,bit2    ; product term = 1
                    00205              EVAL_P  Preg_b,bit3,P11_x,P11_a
004A 020C               M              movf    input,W        ;
004B 0F0B               M              xorlw   P11_x          ;
004C 0E0F               M              andlw   P11_a          ;
004D 0643               M              btfsc   status,bit2    ; skip if zero bit not set
004E 056F               M              bsf     Preg_b,bit3    ; product term = 1
                    00206              EVAL_P  Preg_b,bit4,P12_x,P12_a
004F 020C               M              movf    input,W        ;
0050 0F0C               M              xorlw   P12_x          ;
0051 0E0F               M              andlw   P12_a          ;
0052 0643               M              btfsc   status,bit2    ; skip if zero bit not set
0053 058F               M              bsf     Preg_b,bit4    ; product term = 1
                    00207              EVAL_P  Preg_b,bit5,P13_x,P13_a
0054 020C               M              movf    input,W        ;
0055 0F0D               M              xorlw   P13_x          ;
0056 0E0F               M              andlw   P13_a          ;
0057 0643               M              btfsc   status,bit2    ; skip if zero bit not set
0058 05AF               M              bsf     Preg_b,bit5    ; product term = 1
                    00208              EVAL_P  Preg_b,bit6,P14_x,P14_a
0059 020C               M              movf    input,W        ;
005A 0F0E               M              xorlw   P14_x          ;
005B 0E0F               M              andlw   P14_a          ;
005C 0643               M              btfsc   status,bit2    ; skip if zero bit not set
005D 05CF               M              bsf     Preg_b,bit6    ; product term = 1
                    00209              EVAL_P  Preg_b,bit7,P15_x,P15_a
005E 020C               M              movf    input,W        ;
005F 0F0F               M              xorlw   P15_x          ;
0060 0E0F               M              andlw   P15_a          ;
0061 0643               M              btfsc   status,bit2    ; skip if zero bit not set
0062 05EF               M              bsf     Preg_b,bit7    ; product term = 1
                    00210
                    00211              EVAL_P  Preg_c,bit0,P16_x,P16_a
0063 020C               M              movf    input,W        ;
0064 0F00               M              xorlw   P16_x          ;
0065 0E00               M              andlw   P16_a          ;
0066 0643               M              btfsc   status,bit2    ; skip if zero bit not set
0067 0510               M              bsf     Preg_c,bit0    ; product term = 1
                    00212              EVAL_P  Preg_c,bit1,P17_x,P17_a
0068 020C               M              movf    input,W        ;
0069 0F00               M              xorlw   P17_x          ;
006A 0E00               M              andlw   P17_a          ;
006B 0643               M              btfsc   status,bit2    ; skip if zero bit not set
006C 0530               M              bsf     Preg_c,bit1    ; product term = 1
                    00213              EVAL_P  Preg_c,bit2,P18_x,P18_a
006D 020C               M              movf    input,W        ;
006E 0F00               M              xorlw   P18_x          ;
006F 0E00               M              andlw   P18_a          ;
0070 0643               M              btfsc   status,bit2    ; skip if zero bit not set
0071 0550               M              bsf     Preg_c,bit2    ; product term = 1
                    00214              EVAL_P  Preg_c,bit3,P19_x,P19_a
0072 020C               M              movf    input,W        ;
0073 0F00               M              xorlw   P19_x          ;
0074 0E00               M              andlw   P19_a          ;
0075 0643               M              btfsc   status,bit2    ; skip if zero bit not set
0076 0570               M              bsf     Preg_c,bit3    ; product term = 1
                    00215              EVAL_P  Preg_c,bit4,P20_x,P20_a
0077 020C               M              movf    input,W        ;
0078 0F00               M              xorlw   P20_x          ;
0079 0E00               M              andlw   P20_a          ;
007A 0643               M              btfsc   status,bit2    ; skip if zero bit not set
```

© 1997 Microchip Technology Inc.

```
007B 0590            M                bsf     Preg_c,bit4     ; product term = 1
                 00216                EVAL_P  Preg_c,bit5,P21_x,P21_a
007C 020C            M                movf    input,W         ;
007D 0F00            M                xorlw   P21_x           ;
007E 0E00            M                andlw   P21_a           ;
007F 0643            M                btfsc   status,bit2     ; skip if zero bit not set
0080 05B0            M                bsf     Preg_c,bit5     ; product term = 1
                 00217                EVAL_P  Preg_c,bit6,P22_x,P22_a
0081 020C            M                movf    input,W         ;
0082 0F00            M                xorlw   P22_x           ;
0083 0E00            M                andlw   P22_a           ;
0084 0643            M                btfsc   status,bit2     ; skip if zero bit not set
0085 05D0            M                bsf     Preg_c,bit6     ; product term = 1
                 00218                EVAL_P  Preg_c,bit7,P23_x,P23_a
0086 020C            M                movf    input,W         ;
0087 0F00            M                xorlw   P23_x           ;
0088 0E00            M                andlw   P23_a           ;
0089 0643            M                btfsc   status,bit2     ; skip if zero bit not set
008A 05F0            M                bsf     Preg_c,bit7     ; product term = 1
                 00219
                 00220 or_pl          EVAL_Y   OR_a0,OR_b0,OR_c0,bit0
008B 0CED            M                movlw   OR_a0           ; load constants
008C 0034            M                movwf   OR_a            ;
008D 0CD7            M                movlw   OR_b0           ;
008E 0035            M                movwf   OR_b            ;
008F 0C00            M                movlw   OR_c0           ;
0090 0036            M                movwf   OR_c            ;
0091 0902            M                call    EVAL1           ;
0092 0743            M                btfss   status,bit2     ;
0093 050D            M                bsf     Y_reg,bit0      ;
                 00221                EVAL_Y  OR_a1,OR_b1,OR_c1,bit1
0094 0C9F            M                movlw   OR_a1           ; load constants
0095 0034            M                movwf   OR_a            ;
0096 0C27            M                movlw   OR_b1           ;
0097 0035            M                movwf   OR_b            ;
0098 0C00            M                movlw   OR_c1           ;
0099 0036            M                movwf   OR_c            ;
009A 0902            M                call    EVAL1           ;
009B 0743            M                btfss   status,bit2     ;
009C 052D            M                bsf     Y_reg,bit1      ;
                 00222                EVAL_Y  OR_a2,OR_b2,OR_c2,bit2
009D 0CFB            M                movlw   OR_a2           ; load constants
009E 0034            M                movwf   OR_a            ;
009F 0C2F            M                movlw   OR_b2           ;
00A0 0035            M                movwf   OR_b            ;
00A1 0C00            M                movlw   OR_c2           ;
00A2 0036            M                movwf   OR_c            ;
00A3 0902            M                call    EVAL1           ;
00A4 0743            M                btfss   status,bit2     ;
00A5 054D            M                bsf     Y_reg,bit2      ;
                 00223                EVAL_Y  OR_a3,OR_b3,OR_c3,bit3
00A6 0C6D            M                movlw   OR_a3           ; load constants
00A7 0034            M                movwf   OR_a            ;
00A8 0C79            M                movlw   OR_b3           ;
00A9 0035            M                movwf   OR_b            ;
00AA 0C00            M                movlw   OR_c3           ;
00AB 0036            M                movwf   OR_c            ;
00AC 0902            M                call    EVAL1           ;
00AD 0743            M                btfss   status,bit2     ;
00AE 056D            M                bsf     Y_reg,bit3      ;
                 00224                EVAL_Y  OR_a4,OR_b4,OR_c4,bit4
00AF 0C45            M                movlw   OR_a4           ; load constants
00B0 0034            M                movwf   OR_a            ;
00B1 0CFD            M                movlw   OR_b4           ;
00B2 0035            M                movwf   OR_b            ;
00B3 0C00            M                movlw   OR_c4           ;
```

 © 1997 Microchip Technology Inc.

```
00B4 0036              M                movwf   OR_c            ;
00B5 0902              M                call    EVAL1           ;
00B6 0743              M                btfss   status,bit2     ;
00B7 058D              M                bsf     Y_reg,bit4      ;
                                        EVAL_Y  OR_a5,OR_b5,OR_c5,bit5
00B8 0C71              M                movlw   OR_a5           ; load constants
00B9 0034              M                movwf   OR_a            ;
00BA 0CDF              M                movlw   OR_b5           ;
00BB 0035              M                movwf   OR_b            ;
00BC 0C00              M                movlw   OR_c5           ;
00BD 0036              M                movwf   OR_c            ;
00BE 0902              M                call    EVAL1           ;
00BF 0743              M                btfss   status,bit2     ;
00C0 05AD              M                bsf     Y_reg,bit5      ;
                                        EVAL_Y  OR_a6,OR_b6,OR_c6,bit6
00C1 0C7C              M                movlw   OR_a6           ; load constants
00C2 0034              M                movwf   OR_a            ;
00C3 0CEF              M                movlw   OR_b6           ;
00C4 0035              M                movwf   OR_b            ;
00C5 0C00              M                movlw   OR_c6           ;
00C6 0036              M                movwf   OR_c            ;
00C7 0902              M                call    EVAL1           ;
00C8 0743              M                btfss   status,bit2     ;
00C9 05CD              M                bsf     Y_reg,bit6      ;
                                        EVAL_Y  OR_a7,OR_b7,OR_c7,bit7
00CA 0C00              M                movlw   OR_a7           ; load constants
00CB 0034              M                movwf   OR_a            ;
00CC 0C00              M                movlw   OR_b7           ;
00CD 0035              M                movwf   OR_b            ;
00CE 0C00              M                movlw   OR_c7           ;
00CF 0036              M                movwf   OR_c            ;
00D0 0902              M                call    EVAL1           ;
00D1 0743              M                btfss   status,bit2     ;
00D2 05ED              M                bsf     Y_reg,bit7      ;

                          ; Y_reg now contains 8 output values:
00D3 0040                 wr_out  clrw                          ;
00D4 0007                               tris    7               ; port_c = output
00D5 020D                               movf    Y_reg,W         ;
00D6 0027                               movwf   port_c          ; Y_reg -> port_c
00D7 0800                               retlw   0               ;

00D8 0000                 ZZZ           nop

                                  END

MEMORY USAGE MAP ('X' = Used,  '-' = Unused)

0000 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0080 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
00C0 : XXXXXXXXXXXXXXXX XXXXXXXXX------- ---------------- ----------------
01C0 : ---------------- ---------------- ---------------- ---------------X

All other memory blocks unused.

Program Memory Words Used:   218
Program Memory Words Free:   294


Errors   :     0
Warnings :     0 reported,     0 suppressed
Messages :     0 reported,     0 suppressed
```

Note the following details of the code protection feature on PICmicro® MCUs.

- The PICmicro family meets the specifications contained in the Microchip Data Sheet.
- Microchip believes that its family of PICmicro microcontrollers is one of the most secure products of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the PICmicro microcontroller in a manner outside the operating specifications contained in the data sheet. The person doing so may be engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable”.
- Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our product.

If you have any further questions about this matter, please contact the local sales office nearest to you.

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip’s products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.

Trademarks

The Microchip name and logo, the Microchip logo, FilterLab, KEELOQ, microID, MPLAB, PIC, PICmicro, PICMASTER, PICSTART, PRO MATE, SEEVAL and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

dsPIC, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, microPort, Migratable Memory, MPASM, MPLIB, MPLINK, MPSIM, MXDEV, PICC, PICDEM, PICDEM.net, rfPIC, Select Mode and Total Endurance are trademarks of Microchip Technology Incorporated in the U.S.A.

Serialized Quick Turn Programming (SQTP) is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

© 2002, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

Printed on recycled paper.

DNV Certification, Inc. USA
ANSI-RAB QMS ACCREDITED
DNV
DNV MSC The Netherlands Accredited by the RvA
ISO 9001 / QS-9000 REGISTERED FIRM

Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999. The Company’s quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs and microperipheral products. In addition, Microchip’s quality system for the design and manufacture of development systems is ISO 9001 certified.

WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200 Fax: 480-792-7277
Technical Support: 480-792-7627
Web Address: http://www.microchip.com

Rocky Mountain
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7966 Fax: 480-792-7456

Atlanta
500 Sugar Mill Road, Suite 200B
Atlanta, GA 30350
Tel: 770-640-0034 Fax: 770-640-0307

Boston
2 Lan Drive, Suite 120
Westford, MA 01886
Tel: 978-692-3848 Fax: 978-692-3821

Chicago
333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 630-285-0071 Fax: 630-285-0075

Dallas
4570 Westgrove Drive, Suite 160
Addison, TX 75001
Tel: 972-818-7423 Fax: 972-818-2924

Detroit
Tri-Atria Office Building
32255 Northwestern Highway, Suite 190
Farmington Hills, MI 48334
Tel: 248-538-2250 Fax: 248-538-2260

Kokomo
2767 S. Albright Road
Kokomo, Indiana 46902
Tel: 765-864-8360 Fax: 765-864-8387

Los Angeles
18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 949-263-1888 Fax: 949-263-1338

New York
150 Motor Parkway, Suite 202
Hauppauge, NY 11788
Tel: 631-273-5305 Fax: 631-273-5335

San Jose
Microchip Technology Inc.
2107 North First Street, Suite 590
San Jose, CA 95131
Tel: 408-436-7950 Fax: 408-436-7955

Toronto
6285 Northam Drive, Suite 108
Mississauga, Ontario L4V 1X5, Canada
Tel: 905-673-0699 Fax: 905-673-6509

ASIA/PACIFIC

Australia
Microchip Technology Australia Pty Ltd
Suite 22, 41 Rawson Street
Epping 2121, NSW
Australia
Tel: 61-2-9868-6733 Fax: 61-2-9868-6755

China - Beijing
Microchip Technology Consulting (Shanghai) Co., Ltd., Beijing Liaison Office
Unit 915
Bei Hai Wan Tai Bldg.
No. 6 Chaoyangmen Beidajie
Beijing, 100027, No. China
Tel: 86-10-85282100 Fax: 86-10-85282104

China - Chengdu
Microchip Technology Consulting (Shanghai) Co., Ltd., Chengdu Liaison Office
Rm. 2401, 24th Floor,
Ming Xing Financial Tower
No. 88 TIDU Street
Chengdu 610016, China
Tel: 86-28-6766200 Fax: 86-28-6766599

China - Fuzhou
Microchip Technology Consulting (Shanghai) Co., Ltd., Fuzhou Liaison Office
Unit 28F, World Trade Plaza
No. 71 Wusi Road
Fuzhou 350001, China
Tel: 86-591-7503506 Fax: 86-591-7503521

China - Shanghai
Microchip Technology Consulting (Shanghai) Co., Ltd.
Room 701, Bldg. B
Far East International Plaza
No. 317 Xian Xia Road
Shanghai, 200051
Tel: 86-21-6275-5700 Fax: 86-21-6275-5060

China - Shenzhen
Microchip Technology Consulting (Shanghai) Co., Ltd., Shenzhen Liaison Office
Rm. 1315, 13/F, Shenzhen Kerry Centre,
Renminnan Lu
Shenzhen 518001, China
Tel: 86-755-2350361 Fax: 86-755-2366086

Hong Kong
Microchip Technology Hongkong Ltd.
Unit 901-6, Tower 2, Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2401-1200 Fax: 852-2401-3431

India
Microchip Technology Inc.
India Liaison Office
Divyasree Chambers
1 Floor, Wing A (A3/A4)
No. 11, O'Shaugnessey Road
Bangalore, 560 025, India
Tel: 91-80-2290061 Fax: 91-80-2290062

Japan
Microchip Technology Japan K.K.
Benex S-1 6F
3-18-20, Shinyokohama
Kohoku-Ku, Yokohama-shi
Kanagawa, 222-0033, Japan
Tel: 81-45-471- 6166 Fax: 81-45-471-6122

Korea
Microchip Technology Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea 135-882
Tel: 82-2-554-7200 Fax: 82-2-558-5934

Singapore
Microchip Technology Singapore Pte Ltd.
200 Middle Road
#07-02 Prime Centre
Singapore, 188980
Tel: 65-6334-8870 Fax: 65-6334-8850

Taiwan
Microchip Technology Taiwan
11F-3, No. 207
Tung Hua North Road
Taipei, 105, Taiwan
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

EUROPE

Denmark
Microchip Technology Nordic ApS
Regus Business Centre
Lautrup hoj 1-3
Ballerup DK-2750 Denmark
Tel: 45 4420 9895 Fax: 45 4420 9910

France
Microchip Technology SARL
Parc d'Activite du Moulin de Massy
43 Rue du Saule Trapu
Batiment A - ler Etage
91300 Massy, France
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

Germany
Microchip Technology GmbH
Gustav-Heinemann Ring 125
D-81739 Munich, Germany
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

Italy
Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Taurus 1 V. Le Colleoni 1
20041 Agrate Brianza
Milan, Italy
Tel: 39-039-65791-1 Fax: 39-039-6899883

United Kingdom
Arizona Microchip Technology Ltd.
505 Eskdale Road
Winnersh Triangle
Wokingham
Berkshire, England RG41 5TU
Tel: 44 118 921 5869 Fax: 44-118 921-5820

03/01/02