PLD Replacement

INTRODUCTION

The 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.

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

* Very low cost. Extremely cost effective to replace TTL gates or expensive EPLD's.
* Fully programmable. PIC16C5X microcontrollers are offered as One Time Programmable (OTP) EPROM devices.
* Available off the shelf from distributors.
* PC board real estate saving can be substantial when replacing multitude of TTL's or several PLD's with PIC16C5X microcontrollers which are packaged in 18 and 28 pin packages (DIP, PLCC, SOIC).
* Substantial power savings can be attained by using PIC16C5X's SLEEP mode. In this mode typical power consumption of PIC16C5X is less than 1uA.
* PIC16C5X's I/O ports are bidirectional and software configurable as input or output. The user can mix and match number of inputs or outputs as long as the total does not exceed 20 (PIC16C55/57).
* PIC16C5X's output pins have large current source/sink capability. They can directly drive LED's.
* The speed and efficiency of the PIC16C5X allows it to perform other control, timing, and compute functions in addition to implementing a PLA function.

IMPLEMENTING A PLA

To implement a generic combinational logic function, we can simply emulate an AND-OR PLA in software. This will require that the logic outputs be described as sum of products. To describe our algorithm, we will use a simple 8-input, 8-bit 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 port RB pins RB0,RB1,...,RB7 respectively. The logic outputs Y0,Y1,...,Y7 will appear on port RC pins RC0,RC1,...,RC7 respectively. Port RB is configured as input and port RC 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:

(A<7:0> .XOR. XXXX0011B) .AND. 00001111B).

The constant with which XOR is done will be referred to as P3_x in our discussion. P3_x = XXXX0011B will ensure that if A<3:0> = 0011B, 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 A<7:4>) 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 =	11010111							
OR_c0 =	00000000							

For larger number of inputs, outputs or product terms, the evaluation will be more complex but following 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 requirement. Appendix B shows a slightly different implementation (only EVAL_Y MACRO is different) that optimizes program memory usage over speed. Table 1 shows time and resources required to implement different size PLAs.

FIGURE 1 - A SIMPLE PLA

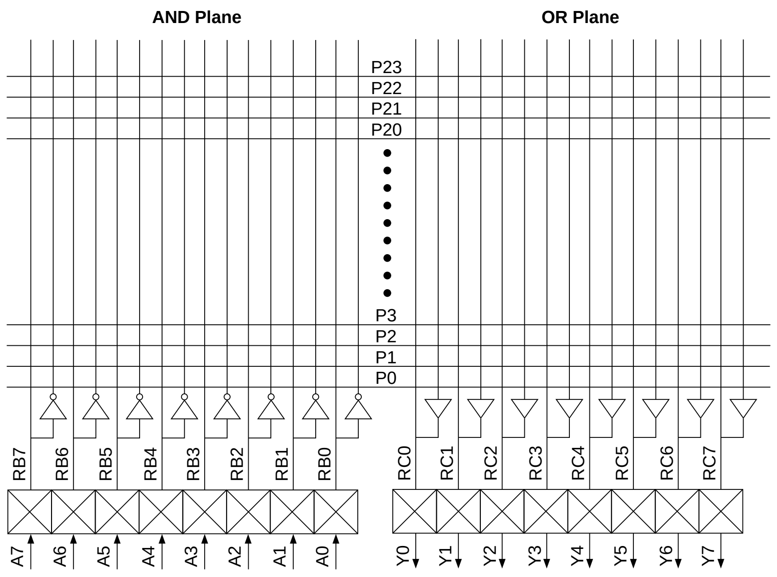
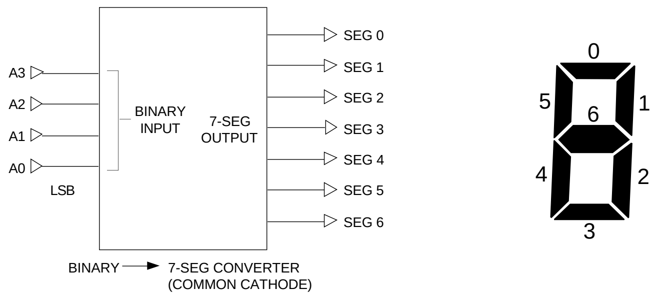


FIGURE 2 - BINARY TO 7-SEGMENT CONVERSION EXAMPLE



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 = $\overline{A3}$, $\overline{A2}$, $\overline{A1}$, $\overline{A0}$
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

SPEED/RESPONSE TIME

The worst case response time of a PLA implemented in this fashion can be calculated as follows. First, we define td = 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 = 2td. This is because if an input changes just after the program reads input port, its effect will not show up until the program completes the current execution cycle, re-reads input and recalculates 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 input change is sensed, return to the beginning rather than execute the rest of the evaluation code.

A Table Look-Up Method For Small PLA Implementation

If the number of inputs is small (8 or less) then a simple table look-up 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 BCD to 7-segment conversion (Figure 2) using this technique.

```
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 look-up
        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 PLA's

Number of Inputs Including fdbk	Number of Outputs Including fdbk and o/e Control	Number of Products	Number of RAM Locations Required NRAM	Number of Program Memory Locations Required NROM	Number of Instruction Cycle To Execute PLA NCYC	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	89.4 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 Ni = Number of inputs
NIW = Number of input words, NIW = $\left\lceil \frac{Ni}{8} \right\rceil$
NP = Number of products
NPW = Number of product words, i.e. NPW = $\left\lceil \frac{Np}{8} \right\rceil$
NO = Number of outputs
NOW = Number of output words, i.e. NOW = $\left\lceil \frac{No}{8} \right\rceil$

Then, NRAM @ NIW + NOW + N PW : Time efficient
NRAM @ NIW + NOW + 2 N PW + 2: Code efficient
NROM @ 8 + N PW + NOW + N P [2 + 3 NIW] +NO • NPW • 4: Time efficient
NROM @ 17 + N PW + NOW + N P [2 + 3 NIW] +NO [NPW + 3]: Code efficient
NCYC @ 8 + N PW + NOW + N P [2 + 3 NIW] +NO [2 NPW • 4]: Time efficient
NCYC @ 8 + N PW + NOW + N P [2 + 3 NIW] +5 N [NPW + 1]: Code efficient

FIGURE 3 - PLA PROGRAM FLOW

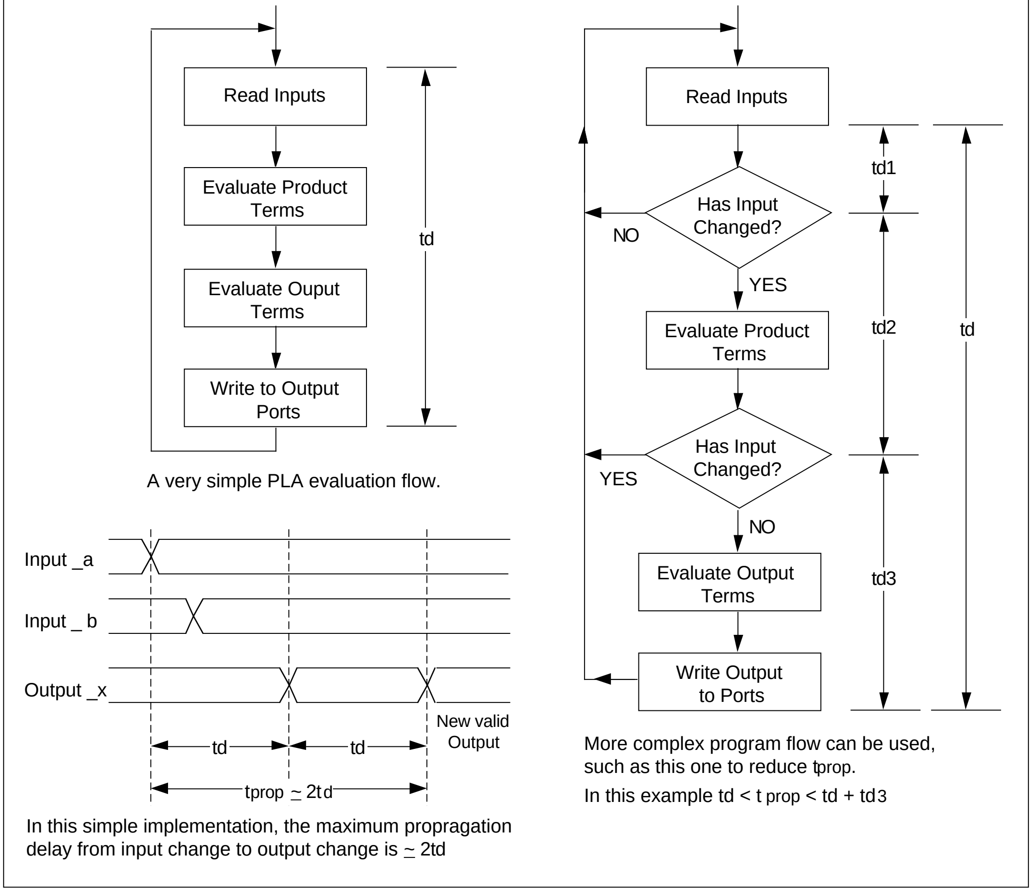
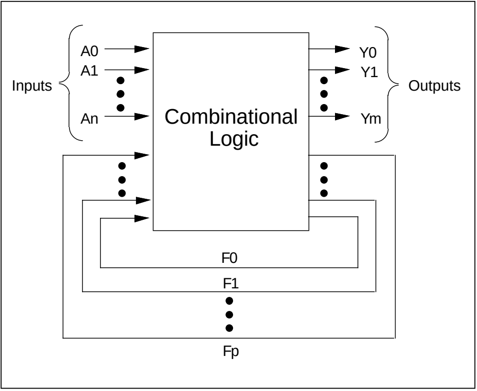


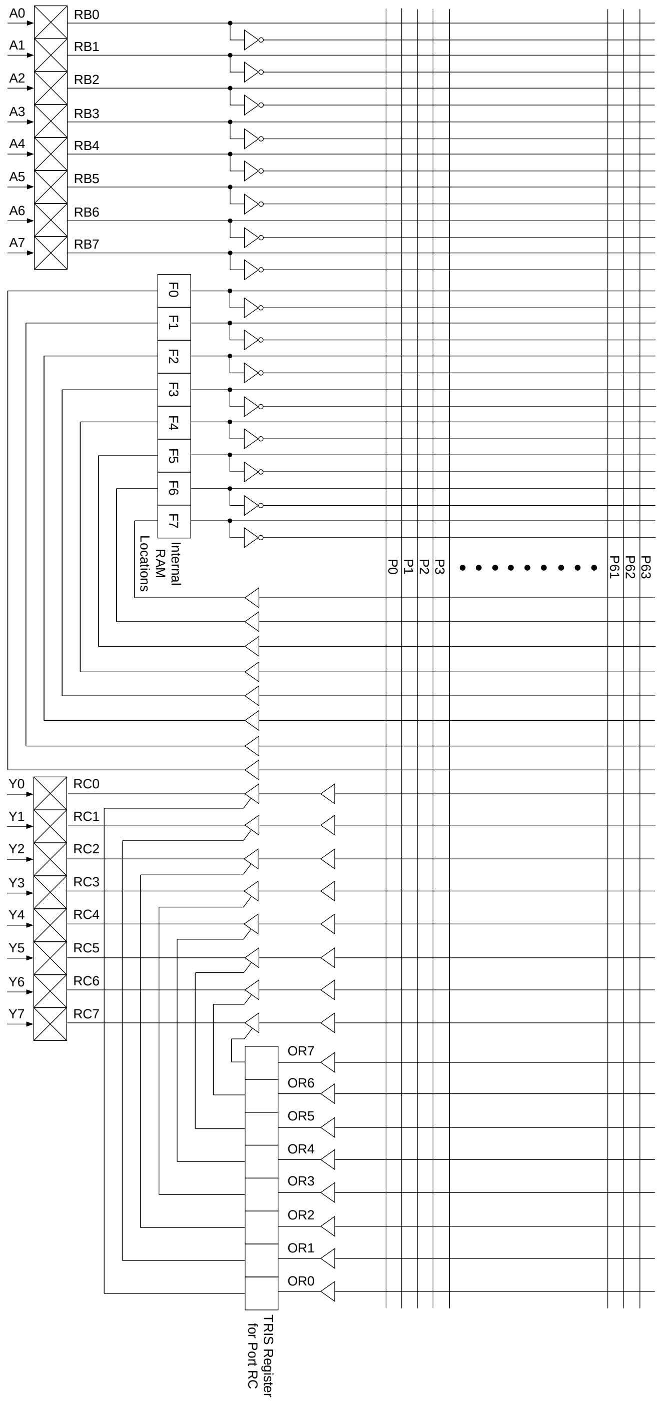
FIGURE 4 - AN ASYNCHRONOUS STATE MACHINE



IMPLEMENTING AN ASYNCHRONOUS STATE MACHINE

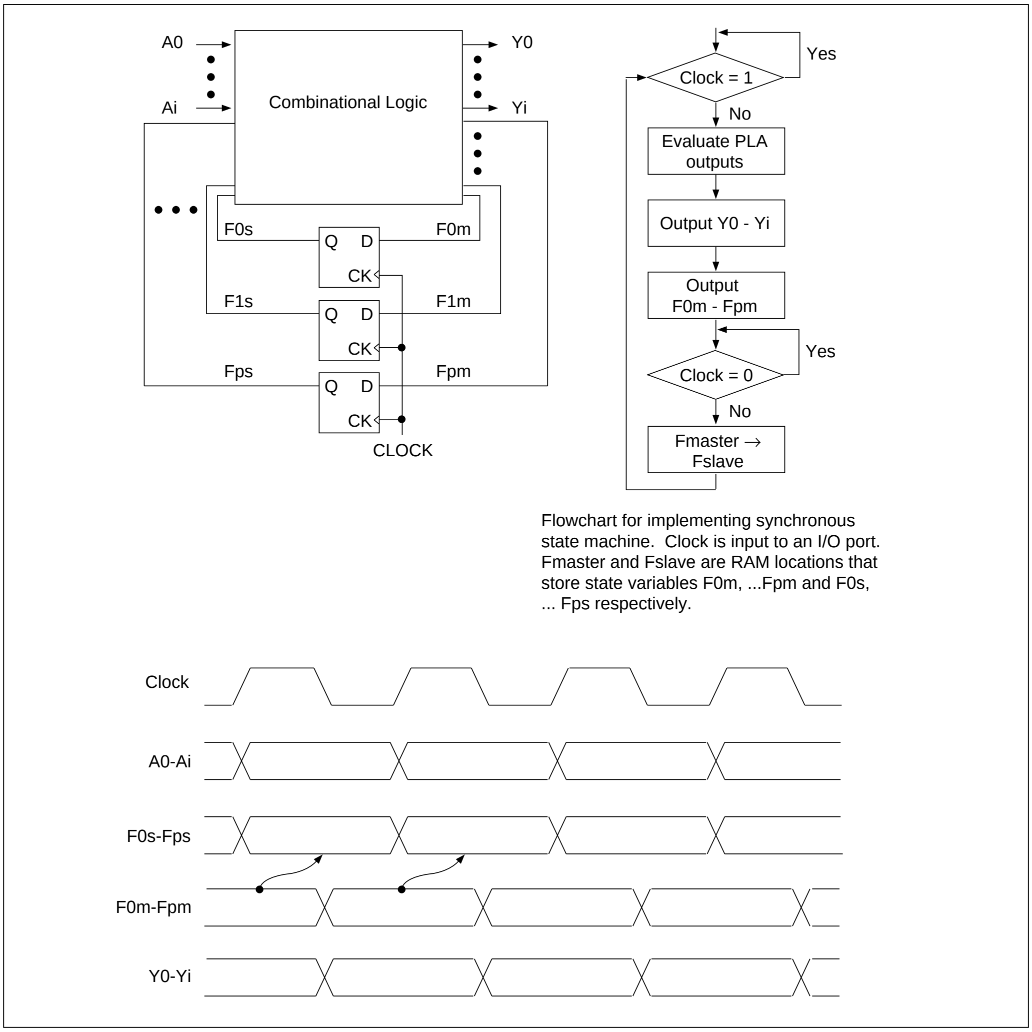
The concept can be easily extended to implement sequential logic i.e. a state machine. Figure 4 shows a state machine with n inputs (A0-An), m outputs (Y0-Ym) and p states that feedback as inputs to the PLA (F0-Fp). In PIC16C5X the states will be stored as bits in RAM location. Input will now mean input 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 port RB 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).

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 bidirectional ports with Tri-State™ control.

FIGURE 6 - A 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 (see Figure 6) usually all inputs are stable at the falling edge (or rising) edge of the system clock. The state machine samples input on the falling edge, 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 quick, low cost solution where system operation speed is not critical.

Author: Sumit Mitra
Logic Products Division

APPENDIX A: PLA IMPLEMENTATION: TIME EFFICIENT APPROACH

```
MPASM B0.54                                                            PAGE  1



                    ;**********************************************************************
                    ; pla1a.asm :
                    ; This procedure implements a simple AND-OR PLA with:
                    ;
                    ;      8  inputs        := A7 A6 A5 A4 A3 A2 A1 A0
                    ;      24 product terms := P23 P22 ..... P0
                    ;      8  outputs       := Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0
                    ;
                    ; The eight inputs are assumed to be connected to PORT RB such that
                    ; RB0 = A0, RB1 = A1, ... , RB7 = A7.
                    ; The outputs are programmed to appear on port RC such that
                    ; RC0 = Y0, RC1 = Y1, ... , RC7 = Y7.
                    ;
                    ; This implementation optimizes both speed & program memory usage
                    ;
                    ;**********************************************************************
                    ;
                    ; define RAM locations used:
                    ;
                                    LIST    P=16C57
000C                input           equ     d’12'        ;RAM location 12 holds input
000D                Y_reg           equ     d’13'        ;holds output result

000E                Preg_a          equ     d’14'        ;Product terms P0 to P7. Preg_a<0> = P0
000F                Preg_b          equ     d’15'        ;Product terms P8 to P15. Preg_b<0> = P8
0010                Preg_c          equ     d’16'        ;Product terms P16 to P23. Preg_c<0> =
P16
                    ; define some constants and file addresses:
                    ;
0000                bit0            equ     0            ;
0001                bit1            equ     1            ;
0002                bit2            equ     2            ;
0003                bit3            equ     3            ;
0004                bit4            equ     4            ;
0005                bit5            equ     5            ;
0006                bit6            equ     6            ;
0007                bit7            equ     7            ;
                    ;
0003                status          equ     3            ;
0006                port_b          equ     6            ;
0007                port_c          equ     7            ;
                    ;
                    ; define the AND plane programming variables:
                    ;
0000                P0_x            equ     b’00000000'  ;
000F                P0_a            equ     b’00001111'  ;
0001                P1_x            equ     b’00000001'  ;
000F                P1_a            equ     b’00001111'  ;
0002                P2_x            equ     b’00000010'  ;
000F                P2_a            equ     b’00001111'  ;
0003                P3_x            equ     b’00000011'  ;
000F                P3_a            equ     b’00001111'  ;
0004                P4_x            equ     b’00000100'  ;
000F                P4_a            equ     b’00001111'  ;
0005                P5_x            equ     b’00000101'  ;
000F                P5_a            equ     b’00001111'  ;
0006                P6_x            equ     b’00000110'  ;
000F                P6_a            equ     b’00001111'  ;
0007                P7_x            equ     b’00000111'  ;
000F                P7_a            equ     b’00001111'  ;
0008                P8_x            equ     b’00001000'  ;
000F                P8_a            equ     b’00001111'  ;
0009                P9_x            equ     b’00001001'  ;
```

```
000F                P9_a            equ     b'00001111'     ;
O00A                P10_x           equ     b'00001010'     ;
000F                P10_a           equ     b'00001111'     ;
000B                P11_x           equ     b'00001011'     ;
000F                P11_a           equ     b'00001111'     ;
000C                P12_x           equ     b'00001100'     ;
000F                P12_a           equ     b'00001111'     ;
000D                P13_x           equ     b'00001101'     ;
000F                P13_a           equ     b'00001111'     ;
000E                P14_x           equ     b'00001110'     ;
000F                P14_a           equ     b'00001111'     ;
000F                P15_x           equ     b'00001111'     ;
000F                P15_a           equ     b'00001111'     ;
0000                P16_x           equ     b'00000000'     ;
0000                P16_a           equ     b'00000000'     ;
0000                P17_x           equ     b'00000000'     ;
0000                P17_a           equ     b'00000000'     ;
0000                P18_x           equ     b'00000000'     ;
0000                P18_a           equ     b'00000000'     ;
0000                P19_x           equ     b'00000000'     ;
0000                P19_a           equ     b'00000000'     ;
0000                P20_x           equ     b'00000000'     ;
0000                P20_a           equ     b'00000000'     ;
0000                P21_x           equ     b'00000000'     ;
0000                P21_a           equ     b'00000000'     ;
0000                P22_x           equ     b'00000000'     ;
0000                P22_a           equ     b'00000000'     ;
0000                P23_x           equ     b'00000000'     ;
0000                P23_a           equ     b'00000000'     ;

                    ; define OR plane programming variables:
                    x
00ED                OR_a0           equ     b'11101101'     ; for output Y0
00D7                OR_b0           equ     b'11010111'     ;
0000                OR_c0           equ     b'00000000'     ;
009F                OR_a1           equ     b'10011111'     ; for output Y1
0027                OR_b1           equ     b'00100111'     ;
0000                OR_c1           equ     b'00000000'     ;
00FB                OR_a2           equ     b'11111011'     ; for output Y2
002F                OR_b2           equ     b'00101111'     ;
0000                OR_c2           equ     b'00000000'     ;
006D                OR_a3           equ     b'01101101'     ; for output Y3
0079                OR_b3           equ     b'01111001'     ;
0000                OR_c3           equ     b'00000000'     ;
0045                OR_a4           equ     b'01000101'     ; for output Y4
00FD                OR_b4           equ     b'11111101'     ;
0000                OR_c4           equ     b'00000000'     ;
0071                OR_a5           equ     b'01110001'     ; for output Y5
00DF                OR_b5           equ     b'11011111'     ;
0000                OR_c5           equ     b'00000000'     ;
007C                OR_a6           equ     b'01111100'     ; for output Y6
00EF                OR_b6           equ     b'11101111'     ;
0000                OR_c6           equ     b'00000000'     ;
0000                OR_a7           equ     b'00000000'     ; for output Y7
0000                OR_b7           equ     b'00000000'     ;
0000                OR_c7           equ     b'00000000'     ;


                                    org     01ffh           ;
01FF 0A00           begin           goto    main            ;

                                    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

                              EVAL_P  Preg_a,bit0,P0_x,P0_a
000A 020C                            movf    input,W         ;
000B 0F00                            xorlw   P0_x            ;
000C 0E0F                            andlw   P0_a            ;
000D 0643                            btfsc   status,bit2     ; skip if zero bit not set
000E 050E                            bsf     Preg_a,bit0     ; product term = 1

                              EVAL_P  Preg_a,bit1,P1_x,P1_a
000F 020C                            movf    input,W         ;
0010 0F01                            xorlw   P1_x            ;
0011 0E0F                            andlw   P1_a            ;
0012 0643                            btfsc   status,bit2     ; skip if zero bit not set
0013 052E                            bsf     Preg_a,bit1     ; product term = 1

                              EVAL_P  Preg_a,bit2,P2_x,P2_a
0014 020C                            movf    input,W         ;
0015 0F02                            xorlw   P2_x            ;
0016 0E0F                            andlw   P2_a            ;
0017 0643                            btfsc   status,bit2     ; skip if zero bit not set
0018 054E                            bsf     Preg_a,bit2     ; product term = 1

                              EVAL_P  Preg_a,bit3,P3_x,P3_a
0019 020C                            movf    input,W         ;
001A 0F03                            xorlw   P3_x            ;
001B 0E0F                            andlw   P3_a            ;
001C 0643                            btfsc   status,bit2     ; skip if zero bit not set
001D 056E                            bsf     Preg_a,bit3     ; product term = 1
```

```
                              EVAL_P  Preg_a,bit4,P4_x,P4_a
001E 020C                            movf    input,W         ;
001F 0F04                            xorlw   P4_x            ;
0020 0E0F                            andlw   P4_a            ;
0021 0643                            btfsc   status,bit2     ; skip if zero bit not set
0022 058E                            bsf     Preg_a,bit4     ; product term = 1

                              EVAL_P  Preg_a,bit5,P5_x,P5_a
0023 020C                            movf    input,W         ;
0024 0F05                            xorlw   P5_x            ;
0025 0E0F                            andlw   P5_a            ;
0026 0643                            btfsc   status,bit2     ; skip if zero bit not set
0027 05AE                            bsf     Preg_a,bit5     ; product term = 1

                              EVAL_P  Preg_a,bit6,P6_x,P6_a
0028 020C                            movf    input,W         ;
0029 0F06                            xorlw   P6_x            ;
002A 0E0F                            andlw   P6_a            ;
002B 0643                            btfsc   status,bit2     ; skip if zero bit not set
002C 05CE                            bsf     Preg_a,bit6     ; product term = 1

                              EVAL_P  Preg_a,bit7,P7_x,P7_a
002D 020C                            movf    input,W         ;
002E 0F07                            xorlw   P7_x            ;
002F 0E0F                            andlw   P7_a            ;
0030 0643                            btfsc   status,bit2     ; skip if zero bit not set
0031 05EE                            bsf     Preg_a,bit7     ; product term = 1


                              EVAL_P  Preg_b,bit0,P8_x,P8_a
0032 020C                            movf    input,W         ;
0033 0F08                            xorlw   P8_x            ;
0034 0E0F                            andlw   P8_a            ;
0035 0643                            btfsc   status,bit2     ; skip if zero bit not set
0036 050F                            bsf     Preg_b,bit0     ; product term = 1

                              EVAL_P  Preg_b,bit1,P9_x,P9_a
0037 020C                            movf    input,W         ;
0038 0F09                            xorlw   P9_x            ;
0039 0E0F                            andlw   P9_a            ;
003A 0643                            btfsc   status,bit2     ; skip if zero bit not set
003B 052F                            bsf     Preg_b,bit1     ; product term = 1

                              EVAL_P  Preg_b,bit2,P10_x,P10_a
003C 020C                            movf    input,W          ;
003D 0F0A                            xorlw   P10_x            ;
003E 0E0F                            andlw   P10_a            ;
003F 0643                            btfsc   status,bit2      ; skip if zero bit not set
0040 054F                            bsf     Preg_b,bit2      ; product term = 1

                              EVAL_P  Preg_b,bit3,P11_x,P11_a
0041 020C                            movf    input,W          ;
0042 0F0B                            xorlw   P11_x            ;
0043 0E0F                            andlw   P11_a            ;
0044 0643                            btfsc   status,bit2      ; skip if zero bit not set
0045 056F                            bsf     Preg_b,bit3      ; product term = 1

                              EVAL_P  Preg_b,bit4,P12_x,P12_a
0046 020C                            movf    input,W          ;
0047 0F0C                            xorlw   P12_x            ;
0048 0E0F                            andlw   P12_a            ;
0049 0643                            btfsc   status,bit2      ; skip if zero bit not set
004A 058F                            bsf     Preg_b,bit4      ; product term = 1

                              EVAL_P  Preg_b,bit5,P13_x,P13_a
004B 020C                            movf    input,W          ;
004C 0F0D                            xorlw   P13_x            ;
004D 0E0F                            andlw   P13_a            ;
004E 0643                            btfsc   status,bit2      ; skip if zero bit not set
004F 05AF                            bsf     Preg_b,bit5      ; product term = 1
```

```
                              EVAL_P  Preg_b,bit6,P14_x,P14_a
0050 020C                            movf    input,W          ;
0051 0F0E                            xorlw   P14_x            ;
0052 0E0F                            andlw   P14_a            ;
0053 0643                            btfsc   status,bit2      ; skip if zero bit not set
0054 05CF                            bsf     Preg_b,bit6      ; product term = 1

                              EVAL_P  Preg_b,bit7,P15_x,P15_a
0055 020C                            movf    input,W          ;
0056 0F0F                            xorlw   P15_x            ;
0057 0E0F                            andlw   P15_a            ;
0058 0643                            btfsc   status,bit2      ; skip if zero bit not set
0059 05EF                            bsf     Preg_b,bit7      ; product term = 1


                              EVAL_P  Preg_c,bit0,P16_x,P16_a
005A 020C                            movf    input,W          ;
005B 0F00                            xorlw   P16_x            ;
005C 0E00                            andlw   P16_a            ;
005D 0643                            btfsc   status,bit2      ; skip if zero bit not set
005E 0510                            bsf     Preg_c,bit0      ; product term = 1

                              EVAL_P  Preg_c,bit1,P17_x,P17_a
005F 020C                            movf    input,W          ;
0060 0F00                            xorlw   P17_x            ;
0061 0E00                            andlw   P17_a            ;
0062 0643                            btfsc   status,bit2      ; skip if zero bit not set
0063 0530                            bsf     Preg_c,bit1      ; product term = 1

                              EVAL_P  Preg_c,bit2,P18_x,P18_a
0064 020C                            movf    input,W          ;
0065 0F00                            xorlw   P18_x            ;
0066 0E00                            andlw   P18_a            ;
0067 0643                            btfsc   status,bit2      ; skip if zero bit not set
0068 0550                            bsf     Preg_c,bit2      ; product term = 1

                              EVAL_P  Preg_c,bit3,P19_x,P19_a
0069 020C                            movf    input,W          ;
006A 0F00                            xorlw   P19_x            ;
006B 0E00                            andlw   P19_a            ;
006C 0643                            btfsc   status,bit2      ; skip if zero bit not set
006D 0570                            bsf     Preg_c,bit3      ; product term = 1

                              EVAL_P  Preg_c,bit4,P20_x,P20_a
006E 020C                            movf    input,W          ;
006F 0F00                            xorlw   P20_x            ;
0070 0E00                            andlw   P20_a            ;
0071 0643                            btfsc   status,bit2      ; skip if zero bit not set
0072 0590                            bsf     Preg_c,bit4      ; product term = 1

                              EVAL_P  Preg_c,bit5,P21_x,P21_a
0073 020C                            movf    input,W          ;
0074 0F00                            xorlw   P21_x            ;
0075 0E00                            andlw   P21_a            ;
0076 0643                            btfsc   status,bit2      ; skip if zero bit not set
0077 05B0                            bsf     Preg_c,bit5      ; product term = 1

                              EVAL_P  Preg_c,bit6,P22_x,P22_a
0078 020C                            movf    input,W          ;
0079 0F00                            xorlw   P22_x            ;
007A 0E00                            andlw   P22_a            ;
007B 0643                            btfsc   status,bit2      ; skip if zero bit not set
007C 05D0                            bsf     Preg_c,bit6      ; product term = 1

                              EVAL_P  Preg_c,bit7,P23_x,P23_a
007D 020C                            movf    input,W          ;
007E 0F00                            xorlw   P23_x            ;
007F 0E00                            andlw   P23_a            ;
0080 0643                            btfsc   status,bit2      ; skip if zero bit not set
```

```
0081 05F0                        bsf     Preg_c,bit7     ; product term = 1
                    or_pl     EVAL_Y  OR_a0,OR_b0,OR_c0,bit0
                               LOCAL   SETBIT            ;
0082 020E                        movf    Preg_a,W        ;
0083 0EED                        andlw   OR_a0           ;
0084 0743                        btfss   status,bit2     ;
0085 0A8D                        goto    SETBIT          ;

0086 020F                        movf    Preg_b,W        ;
0087 0ED7                        andlw   OR_b0           ;
0088 0743                        btfss   status,bit2     ;
0089 0A8D                        goto    SETBIT          ;

008A 0210                        movf    Preg_c,W        ;
008B 0E00                        andlw   OR_c0           ;
008C 0743                        btfss   status,bit2     ;
008D 050D            SETBIT      bsf     Y_reg,bit0      ;

                          EVAL_Y  OR_a1,OR_b1,OR_c1,bit1
                               LOCAL   SETBIT            ;
008E 020E                        movf    Preg_a,W        ;
008F 0E9F                        andlw   OR_a1           ;
0090 0743                        btfss   status,bit2     ;
0091 0A99                        goto    SETBIT          ;

0092 020F                        movf    Preg_b,W        ;
0093 0E27                        andlw   OR_b1           ;
0094 0743                        btfss   status,bit2     ;
0095 0A99                        goto    SETBIT          ;

0096 0210                        movf    Preg_c,W        ;
0097 0E00                        andlw   OR_c1           ;
0098 0743                        btfss   status,bit2     ;
0099 052D            SETBIT      bsf     Y_reg,bit1      ;

                          EVAL_Y  OR_a2,OR_b2,OR_c2,bit2
                               LOCAL   SETBIT            ;
009A 020E                        movf    Preg_a,W        ;
009B 0EFB                        andlw   OR_a2           ;
009C 0743                        btfss   status,bit2     ;
009D 0AA5                        goto    SETBIT          ;

009E 020F                        movf    Preg_b,W        ;
009F 0E2F                        andlw   OR_b2           ;
00A0 0743                        btfss   status,bit2     ;
00A1 0AA5                        goto    SETBIT          ;

00A2 0210                        movf    Preg_c,W        ;
00A3 0E00                        andlw   OR_c2           ;
00A4 0743                        btfss   status,bit2     ;
00A5 054D            SETBIT      bsf     Y_reg,bit2      ;

                          EVAL_Y  OR_a3,OR_b3,OR_c3,bit3
                               LOCAL   SETBIT            ;
00A6 020E                        movf    Preg_a,W        ;
00A7 0E6D                        andlw   OR_a3           ;
00A8 0743                        btfss   status,bit2     ;
00A9 0AB1                        goto    SETBIT          ;

00AA 020F                        movf    Preg_b,W        ;
00AB 0E79                        andlw   OR_b3           ;
00AC 0743                        btfss   status,bit2     ;
00AD 0AB1                        goto    SETBIT          ;

00AE 0210                        movf    Preg_c,W        ;
00AF 0E00                        andlw   OR_c3           ;
00B0 0743                        btfss   status,bit2     ;
00B1 056D            SETBIT      bsf     Y_reg,bit3      ;
```

```
                              EVAL_Y  OR_a4,OR_b4,OR_c4,bit4
                                   LOCAL   SETBIT            ;
00B2 020E                            movf    Preg_a,W        ;
00B3 0E45                            andlw   OR_a4           ;
00B4 0743                            btfss   status,bit2     ;
00B5 0ABD                            goto    SETBIT          ;

00B6 020F                            movf    Preg_b,W        ;
00B7 0EFD                            andlw   OR_b4           ;
00B8 0743                            btfss   status,bit2     ;
00B9 0ABD                            goto    SETBIT          ;

00BA 0210                            movf    Preg_c,W        ;
00BB 0E00                            andlw   OR_c4           ;
00BC 0743                            btfss   status,bit2     ;
00BD 058D             SETBIT         bsf     Y_reg,bit4      ;

                              EVAL_Y  OR_a5,OR_b5,OR_c5,bit5
                                   LOCAL   SETBIT            ;
00BE 020E                            movf    Preg_a,W        ;
00BF 0E71                            andlw   OR_a5           ;
00C0 0743                            btfss   status,bit2     ;
00C1 0AC9                            goto    SETBIT          ;

00C2 020F                            movf    Preg_b,W        ;
00C3 0EDF                            andlw   OR_b5           ;
00C4 0743                            btfss   status,bit2     ;
00C5 0AC9                            goto    SETBIT          ;

00C6 0210                            movf    Preg_c,W        ;
00C7 0E00                            andlw   OR_c5           ;
00C8 0743                            btfss   status,bit2     ;
00C9 05AD             SETBIT         bsf     Y_reg,bit5      ;

                              EVAL_Y  OR_a6,OR_b6,OR_c6,bit6
                                   LOCAL   SETBIT            ;
00CA 020E                            movf    Preg_a,W        ;
00CB 0E7C                            andlw   OR_a6           ;
00CC 0743                            btfss   status,bit2     ;
00CD 0AD5                            goto    SETBIT          ;

00CE 020F                            movf    Preg_b,W        ;
00CF 0EEF                            andlw   OR_b6           ;
00D0 0743                            btfss   status,bit2     ;
00D1 0AD5                            goto    SETBIT          ;

00D2 0210                            movf    Preg_c,W        ;
00D3 0E00                            andlw   OR_c6           ;
00D4 0743                            btfss   status,bit2     ;
00D5 05CD             SETBIT         bsf     Y_reg,bit6      ;

                              EVAL_Y  OR_a7,OR_b7,OR_c7,bit7
                                   LOCAL   SETBIT            ;
00D6 020E                            movf    Preg_a,W        ;
00D7 0E00                            andlw   OR_a7           ;
00D8 0743                            btfss   status,bit2     ;
00D9 0AE1                            goto    SETBIT          ;

00DA 020F                            movf    Preg_b,W        ;
00DB 0E00                            andlw   OR_b7           ;
00DC 0743                            btfss   status,bit2     ;
00DD 0AE1                            goto    SETBIT          ;

00DE 0210                            movf    Preg_c,W        ;
00DF 0E00                            andlw   OR_c7           ;
00E0 0743                            btfss   status,bit2     ;
00E1 05ED             SETBIT         bsf     Y_reg,bit7      ;
```

```
                    ; Y_reg now contains 8 output values:
00E2 0040            wr_out  clrw                            ;
00E3 0007                            tris    7               ; port_c = output
00E4 020D                            movf    Y_reg,W         ;
00E5 0027                            movwf   port_c          ; Y_reg -> port_c
00E6 0800                            retlw   0               ;

00E7 0000            ZZZ             nop

                                   END


Errors   :    0
Warnings :    0
```

APPENDIX B: PLA IMPLEMENTATION: CODE EFFICIENT APPROACH

```
MPASM B0.54                                                          PAGE  1



                    ;**********************************************************************
                    ; pla1b.asm :
                    ; This procedure implements a simple AND-OR PLA with:
                    ;
                    ;      8  inputs        := A7 A6 A5 A4 A3 A2 A1 A0
                    ;      24 product terms := P23 P22 ..... P0
                    ;      8  outputs       := Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0
                    ;
                    ; The eight inputs are assumed to be connected to PORT RB such that
                    ; RB0 = A0, RB1 = A1, ... , RB7 = A7.
                    ; The outputs are programmed to appear on port RC such that
                    ; RC0 = Y0, RC1 = Y1, ... , RC7 = Y7.
                    ;
                    ; This implementation optimizes program memory usage over
                    ; speed
                    ;**********************************************************************
                    ;
                    ; define RAM locations used:
                    ;
                                   LIST    P=16C57
000C                input           equ     d″12"     ; RAM location 12 holds input
000D                Y_reg           equ     d″13"     ; holds output result

000E                Preg_a          equ     d″14"     ; Product terms P0 to P7. Preg_a<0> = P0
000F                Preg_b          equ     d″15"     ; Product terms P8 to P15. Preg_b<0> = P8
0010                Preg_c          equ     d″16"     ; Product terms P16 to P23. Preg_c<0> = P16
0012                Pn_x            equ     d″18"     ;
0013                Pn_a            equ     d″19"     ;
0014                OR_a            equ     d″20"     ;
0015                OR_b            equ     d″21"     ;
0016                OR_c            equ     d″22"     ;

                    ; define some constants and file addresses:
                    ;
0000                bit0            equ     0          ;
0001                bit1            equ     1          ;
0002                bit2            equ     2          ;
0003                bit3            equ     3          ;
0004                bit4            equ     4          ;
0005                bit5            equ     5          ;
0006                bit6            equ     6          ;
0007                bit7            equ     7          ;
                    ;
0003                status          equ     3          ;
0006                port_b          equ     6          ;
0007                port_c          equ     7          ;
```

```
                    ;
                    ; define the AND plane programming variables:
                    ;
0000                P0_x            equ     b”00000000"     ;
000F                P0_a            equ     b”00001111"     ;
0001                P1_x            equ     b”00000001"     ;
000F                P1_a            equ     b”00001111"     ;
0002                P2_x            equ     b”00000010"     ;
000F                P2_a            equ     b”00001111"     ;
0003                P3_x            equ     b”00000011"     ;
000F                P3_a            equ     b”00001111"     ;
0004                P4_x            equ     b”00000100"     ;
000F                P4_a            equ     b”00001111"     ;
0005                P5_x            equ     b”00000101"     ;
000F                P5_a            equ     b”00001111"     ;
0006                P6_x            equ     b”00000110"     ;
000F                P6_a            equ     b”00001111"     ;
0007                P7_x            equ     b”00000111"     ;
000F                P7_a            equ     b”00001111"     ;
0008                P8_x            equ     b”00001000"     ;
000F                P8_a            equ     b”00001111"     ;
0009                P9_x            equ     b”00001001"     ;
000F                P9_a            equ     b”00001111"     ;
000A                P10_x           equ     b”00001010"     ;
000F                P10_a           equ     b”00001111"     ;
000B                P11_x           equ     b”00001011"     ;
000F                P11_a           equ     b”00001111"     ;
000C                P12_x           equ     b”00001100"     ;
000F                P12_a           equ     b”00001111"     ;
000D                P13_x           equ     b”00001101"     ;
000F                P13_a           equ     b”00001111"     ;
000E                P14_x           equ     b”00001110"     ;
000F                P14_a           equ     b”00001111"     ;
000F                P15_x           equ     b”00001111"     ;
000F                P15_a           equ     b”00001111"     ;
0000                P16_x           equ     b”00000000"     ;
0000                P16_a           equ     b”00000000"     ;
0000                P17_x           equ     b”00000000"     ;
0000                P17_a           equ     b”00000000"     ;
0000                P18_x           equ     b”00000000"     ;
0000                P18_a           equ     b”00000000"     ;
0000                P19_x           equ     b”00000000"     ;
0000                P19_a           equ     b”00000000"     ;
0000                P20_x           equ     b”00000000"     ;
0000                P20_a           equ     b”00000000"     ;
0000                P21_x           equ     b”00000000"     ;
0000                P21_a           equ     b”00000000"     ;
0000                P22_x           equ     b”00000000"     ;
0000                P22_a           equ     b”00000000"     ;
0000                P23_x           equ     b”00000000"     ;
0000                P23_a           equ     b”00000000"     ;

                    ; define OR plane programming variables:

00ED                OR_a0           equ     b”11101101"     ; for output Y0
00D7                OR_b0           equ     b”11010111"     ;
0000                OR_c0           equ     b”00000000"     ;
009F                OR_a1           equ     b”10011111"     ; for output Y1
0027                OR_b1           equ     b”00100111"     ;
0000                OR_c1           equ     b”00000000"     ;
00FB                OR_a2           equ     b”11111011"     ; for output Y2
002F                OR_b2           equ     b”00101111"     ;
0000                OR_c2           equ     b”00000000"     ;
006D                OR_a3           equ     b”01101101"     ; for output Y3
0079                OR_b3           equ     b”01111001"     ;
0000                OR_c3           equ     b”00000000"     ;
0045                OR_a4           equ     b”01000101"     ; for output Y4
00FD                OR_b4           equ     b”11111101"     ;
0000                OR_c4           equ     b”00000000"     ;
```

```
0071                OR_a5           equ     b”01110001"     ; for output Y5
00DF                OR_b5           equ     b”11011111"     ;
0000                OR_c5           equ     b”00000000"     ;
007C                OR_a6           equ     b”01111100"     ; for output Y6
00EF                OR_b6           equ     b”11101111"     ;
0000                OR_c6           equ     b”00000000"     ;
0000                OR_a7           equ     b”00000000"     ; for output Y7
0000                OR_b7           equ     b”00000000"     ;
0000                OR_c7           equ     b”00000000"     ;


                                    org     01ffh           ;
01FF 0A00           begin           goto    main            ;

                                    org     000h            ;
                    ; define macro to evaluate 1 product (AND) term:
0000 090B           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
                              movlw   OR_an                 ; load constants
                              movwf   OR_a                  ;
                              movlw   OR_bn                 ;
                              movwf   OR_b                  ;
                              movlw   OR_cn                 ;
                              movwf   OR_c                  ;
                              call    EVAL1                 ;
                              btfss   status,bit2           ;
                              bsf     Y_reg,bit_n           ;
                              ENDM
                    ; define procedure to evaluate 1 output (OR) term:
                    ;
0002 020E           EVAL1           movf    Preg_a,W        ;
0003 0174                           andwf   OR_a,1          ;

0004 020F                           movf    Preg_b,W        ;
0005 0175                           andwf   OR_b,1          ;

0006 0210                           movf    Preg_c,W        ;
0007 0156                           andwf   OR_c,W          ;

0008 0114                           iorwf   OR_a,W          ;
0009 0115                           iorwf   OR_b,W          ;
000A 0800                           retlw   0               ;  W = 1 implies Yn = 1

                    ; now the  PLA evaluation procedure:
                    ;
000B 0CFF           pla88           movlw   0ffh            ;
000C 0006                           tris    6               ; port_b = input
000D 0206                           movf    port_b,W        ; read input
000E 002C                           movwf   input           ; store input in a register
000F 006E                           clrf    Preg_a          ; clear Product register a
0010 006F                           clrf    Preg_b          ; clear Product register b
0011 0070                           clrf    Preg_c          ; clear Product register c
0012 006D                           clrf    Y_reg           ; clear output register
```

```
                    and_pl     EVAL_P  Preg_a,bit0,P0_x,P0_a
0013 020C                            movf    input,W        ;
0014 0F00                            xorlw   P0_x           ;
0015 0E0F                            andlw   P0_a           ;
0016 0643                            btfsc   status,bit2    ; skip if zero bit not set
0017 050E                            bsf     Preg_a,bit0    ; product term = 1

                              EVAL_P  Preg_a,bit1,P1_x,P1_a
0018 020C                            movf    input,W        ;
0019 0F01                            xorlw   P1_x           ;
001A 0E0F                            andlw   P1_a           ;
001B 0643                            btfsc   status,bit2    ; skip if zero bit not set
001C 052E                            bsf     Preg_a,bit1    ; product term = 1

                              EVAL_P  Preg_a,bit2,P2_x,P2_a
001D 020C                            movf    input,W        ;
001E 0F02                            xorlw   P2_x           ;
001F 0E0F                            andlw   P2_a           ;
0020 0643                            btfsc   status,bit2    ; skip if zero bit not set
0021 054E                            bsf     Preg_a,bit2    ; product term = 1

                              EVAL_P  Preg_a,bit3,P3_x,P3_a
0022 020C                            movf    input,W        ;
0023 0F03                            xorlw   P3_x           ;
0024 0E0F                            andlw   P3_a           ;
0025 0643                            btfsc   status,bit2    ; skip if zero bit not set
0026 056E                            bsf     Preg_a,bit3    ; product term = 1

                              EVAL_P  Preg_a,bit4,P4_x,P4_a
0027 020C                            movf    input,W        ;
0028 0F04                            xorlw   P4_x           ;
0029 0E0F                            andlw   P4_a           ;
002A 0643                            btfsc   status,bit2    ; skip if zero bit not set
002B 058E                            bsf     Preg_a,bit4    ; product term = 1

                              EVAL_P  Preg_a,bit5,P5_x,P5_a
002C 020C                            movf    input,W        ;
002D 0F05                            xorlw   P5_x           ;
002E 0E0F                            andlw   P5_a           ;
002F 0643                            btfsc   status,bit2    ; skip if zero bit not set
0030 05AE                            bsf     Preg_a,bit5    ; product term = 1

                              EVAL_P  Preg_a,bit6,P6_x,P6_a
0031 020C                            movf    input,W        ;
0032 0F06                            xorlw   P6_x           ;
0033 0E0F                            andlw   P6_a           ;
0034 0643                            btfsc   status,bit2    ; skip if zero bit not set
0035 05CE                            bsf     Preg_a,bit6    ; product term = 1

                              EVAL_P  Preg_a,bit7,P7_x,P7_a
0036 020C                            movf    input,W        ;
0037 0F07                            xorlw   P7_x           ;
0038 0E0F                            andlw   P7_a           ;
0039 0643                            btfsc   status,bit2    ; skip if zero bit not set
003A 05EE                            bsf     Preg_a,bit7    ; product term = 1


                              EVAL_P  Preg_b,bit0,P8_x,P8_a
003B 020C                            movf    input,W        ;
003C 0F08                            xorlw   P8_x           ;
003D 0E0F                            andlw   P8_a           ;
003E 0643                            btfsc   status,bit2    ; skip if zero bit not set
003F 050F                            bsf     Preg_b,bit0    ; product term = 1

                              EVAL_P  Preg_b,bit1,P9_x,P9_a
0040 020C                            movf    input,W        ;
0041 0F09                            xorlw   P9_x           ;
0042 0E0F                            andlw   P9_a           ;
0043 0643                            btfsc   status,bit2    ; skip if zero bit not set
0044 052F                            bsf     Preg_b,bit1    ; product term = 1
```

```
                              EVAL_P  Preg_b,bit2,P10_x,P10_a
0045 020C                            movf    input,W        ;
0046 0F0A                            xorlw   P10_x          ;
0047 0E0F                            andlw   P10_a          ;
0048 0643                            btfsc   status,bit2    ; skip if zero bit not set
0049 054F                            bsf     Preg_b,bit2    ; product term = 1

                              EVAL_P  Preg_b,bit3,P11_x,P11_a
004A 020C                            movf    input,W        ;
004B 0F0B                            xorlw   P11_x          ;
004C 0E0F                            andlw   P11_a          ;
004D 0643                            btfsc   status,bit2    ; skip if zero bit not set
004E 056F                            bsf     Preg_b,bit3    ; product term = 1

                              EVAL_P  Preg_b,bit4,P12_x,P12_a
004F 020C                            movf    input,W        ;
0050 0F0C                            xorlw   P12_x          ;
0051 0E0F                            andlw   P12_a          ;
0052 0643                            btfsc   status,bit2    ; skip if zero bit not set
0053 058F                            bsf     Preg_b,bit4    ; product term = 1
                              EVAL_P  Preg_b,bit5,P13_x,P13_a
0054 020C                            movf    input,W        ;
0055 0F0D                            xorlw   P13_x          ;
0056 0E0F                            andlw   P13_a          ;
0057 0643                            btfsc   status,bit2    ; skip if zero bit not set
0058 05AF                            bsf     Preg_b,bit5    ; product term = 1

                              EVAL_P  Preg_b,bit6,P14_x,P14_a
0059 020C                            movf    input,W        ;
005A 0F0E                            xorlw   P14_x          ;
005B 0E0F                            andlw   P14_a          ;
005C 0643                            btfsc   status,bit2    ; skip if zero bit not set
005D 05CF                            bsf     Preg_b,bit6    ; product term = 1

                              EVAL_P  Preg_b,bit7,P15_x,P15_a
005E 020C                            movf    input,W        ;
005F 0F0F                            xorlw   P15_x          ;
0060 0E0F                            andlw   P15_a          ;
0061 0643                            btfsc   status,bit2    ; skip if zero bit not set
0062 05EF                            bsf     Preg_b,bit7    ; product term = 1


                              EVAL_P  Preg_c,bit0,P16_x,P16_a
0063 020C                            movf    input,W        ;
0064 0F00                            xorlw   P16_x          ;
0065 0E00                            andlw   P16_a          ;
0066 0643                            btfsc   status,bit2    ; skip if zero bit not set
0067 0510                            bsf     Preg_c,bit0    ; product term = 1

                              EVAL_P  Preg_c,bit1,P17_x,P17_a
0068 020C                            movf    input,W        ;
0069 0F00                            xorlw   P17_x          ;
006A 0E00                            andlw   P17_a          ;
006B 0643                            btfsc   status,bit2    ; skip if zero bit not set
006C 0530                            bsf     Preg_c,bit1    ; product term = 1

                              EVAL_P  Preg_c,bit2,P18_x,P18_a
006D 020C                            movf    input,W        ;
006E 0F00                            xorlw   P18_x          ;
006F 0E00                            andlw   P18_a          ;
0070 0643                            btfsc   status,bit2    ; skip if zero bit not set
0071 0550                            bsf     Preg_c,bit2    ; product term = 1

                              EVAL_P  Preg_c,bit3,P19_x,P19_a
0072 020C                            movf    input,W        ;
0073 0F00                            xorlw   P19_x          ;
0074 0E00                            andlw   P19_a          ;
0075 0643                            btfsc   status,bit2    ; skip if zero bit not set
0076 0570                            bsf     Preg_c,bit3    ; product term = 1
```

```
                              EVAL_P  Preg_c,bit4,P20_x,P20_a
0077 020C                            movf    input,W        ;
0078 0F00                            xorlw   P20_x          ;
0079 0E00                            andlw   P20_a          ;
007A 0643                            btfsc   status,bit2    ; skip if zero bit not set
007B 0590                            bsf     Preg_c,bit4    ; product term = 1

                              EVAL_P  Preg_c,bit5,P21_x,P21_a
007C 020C                            movf    input,W        ;
007D 0F00                            xorlw   P21_x          ;
007E 0E00                            andlw   P21_a          ;
007F 0643                            btfsc   status,bit2    ; skip if zero bit not set
0080 05B0                            bsf     Preg_c,bit5    ; product term = 1

                              EVAL_P  Preg_c,bit6,P22_x,P22_a
0081 020C                            movf    input,W        ;
0082 0F00                            xorlw   P22_x          ;
0083 0E00                            andlw   P22_a          ;
0084 0643                            btfsc   status,bit2    ; skip if zero bit not set
0085 05D0                            bsf     Preg_c,bit6    ; product term = 1

                              EVAL_P  Preg_c,bit7,P23_x,P23_a
0086 020C                            movf    input,W        ;
0087 0F00                            xorlw   P23_x          ;
0088 0E00                            andlw   P23_a          ;
0089 0643                            btfsc   status,bit2    ; skip if zero bit not set
008A 05F0                            bsf     Preg_c,bit7    ; product term = 1


                    or_pl      EVAL_Y  OR_a0,OR_b0,OR_c0,bit0
008B 0CED                            movlw   OR_a0          ; load constants
008C 0034                            movwf   OR_a           ;
008D 0CD7                            movlw   OR_b0          ;
008E 0035                            movwf   OR_b           ;
008F 0C00                            movlw   OR_c0          ;
0090 0036                            movwf   OR_c           ;
0091 0902                            call    EVAL1          ;
0092 0743                            btfss   status,bit2    ;
0093 050D                            bsf     Y_reg,bit0     ;

                              EVAL_Y  OR_a1,OR_b1,OR_c1,bit1
0094 0C9F                            movlw   OR_a1          ; load constants
0095 0034                            movwf   OR_a           ;
0096 0C27                            movlw   OR_b1          ;
0097 0035                            movwf   OR_b           ;
0098 0C00                            movlw   OR_c1          ;
0099 0036                            movwf   OR_c           ;
009A 0902                            call    EVAL1          ;
009B 0743                            btfss   status,bit2    ;
009C 052D                            bsf     Y_reg,bit1     ;

                              EVAL_Y  OR_a2,OR_b2,OR_c2,bit2
009D 0CFB                            movlw   OR_a2          ; load constants
009E 0034                            movwf   OR_a           ;
009F 0C2F                            movlw   OR_b2          ;
00A0 0035                            movwf   OR_b           ;
00A1 0C00                            movlw   OR_c2          ;
00A2 0036                            movwf   OR_c           ;
00A3 0902                            call    EVAL1          ;
00A4 0743                            btfss   status,bit2    ;
```

```
00A5 054D                          bsf     Y_reg,bit2      ;

                             EVAL_Y  OR_a3,OR_b3,OR_c3,bit3
00A6 0C6D                          movlw   OR_a3           ; load constants
00A7 0034                          movwf   OR_a            ;
00A8 0C79                          movlw   OR_b3           ;
00A9 0035                          movwf   OR_b            ;
00AA 0C00                          movlw   OR_c3           ;
00AB 0036                          movwf   OR_c            ;
00AC 0902                          call    EVAL1           ;
00AD 0743                          btfss   status,bit2     ;
00AE 056D                          bsf     Y_reg,bit3      ;

                             EVAL_Y  OR_a4,OR_b4,OR_c4,bit4
00AF 0C45                          movlw   OR_a4           ; load constants
00B0 0034                          movwf   OR_a            ;
00B1 0CFD                          movlw   OR_b4           ;
00B2 0035                          movwf   OR_b            ;
00B3 0C00                          movlw   OR_c4           ;
00B4 0036                          movwf   OR_c            ;
00B5 0902                          call    EVAL1           ;
00B6 0743                          btfss   status,bit2     ;
00B7 058D                          bsf     Y_reg,bit4      ;

                             EVAL_Y  OR_a5,OR_b5,OR_c5,bit5
00B8 0C71                          movlw   OR_a5           ; load constants
00B9 0034                          movwf   OR_a            ;
00BA 0CDF                          movlw   OR_b5           ;
00BB 0035                          movwf   OR_b            ;
00BC 0C00                          movlw   OR_c5           ;
00BD 0036                          movwf   OR_c            ;
00BE 0902                          call    EVAL1           ;
00BF 0743                          btfss   status,bit2     ;
00C0 05AD                          bsf     Y_reg,bit5      ;

                             EVAL_Y  OR_a6,OR_b6,OR_c6,bit6
00C1 0C7C                          movlw   OR_a6           ; load constants
00C2 0034                          movwf   OR_a            ;
00C3 0CEF                          movlw   OR_b6           ;
00C4 0035                          movwf   OR_b            ;
00C5 0C00                          movlw   OR_c6           ;
00C6 0036                          movwf   OR_c            ;
00C7 0902                          call    EVAL1           ;
00C8 0743                          btfss   status,bit2     ;
00C9 05CD                          bsf     Y_reg,bit6      ;

                             EVAL_Y  OR_a7,OR_b7,OR_c7,bit7
00CA 0C00                          movlw   OR_a7           ; load constants
00CB 0034                          movwf   OR_a            ;
00CC 0C00                          movlw   OR_b7           ;
00CD 0035                          movwf   OR_b            ;
00CE 0C00                          movlw   OR_c7           ;
00CF 0036                          movwf   OR_c            ;
00D0 0902                          call    EVAL1           ;
00D1 0743                          btfss   status,bit2     ;
00D2 05ED                          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
Errors   :    0
Warnings :    0
```

WORLDWIDE SALES & SERVICE

AMERICAS

Corporate Office
Microchip Technology Inc.
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 602 786-7200 Fax: 602 786-7277
Technical Support: 602 786-7627
Web: http://www.mchip.com/microhip

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

Boston
Microchip Technology Inc.
5 Mount Royal Avenue
Marlborough, MA 01752
Tel: 508 480-9990 Fax: 508 480-8575

Chicago
Microchip Technology Inc.
333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 708 285-0071 Fax: 708 285-0075

Dallas
Microchip Technology Inc.
14651 Dallas Parkway, Suite 816
Dallas, TX 75240-8809
Tel: 214 991-7177 Fax: 214 991-8588

Dayton
Microchip Technology Inc.
35 Rockridge Road
Englewood, OH 45322
Tel: 513 832-2543 Fax: 513 832-2841

Los Angeles
Microchip Technology Inc.
18201 Von Karman, Suite 455
Irvine, CA 92715
Tel: 714 263-1888 Fax: 714 263-1338

New York
Microchip Technology Inc.
150 Motor Parkway, Suite 416
Hauppauge, NY 11788
Tel: 516 273-5305 Fax: 516 273-5335

AMERICAS (continued)

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

ASIA/PACIFIC

Hong Kong
Microchip Technology
Unit No. 3002-3004, Tower 1
Metroplaza
223 Hing Fong Road
Kwai Fong, N.T. Hong Kong
Tel: 852 2 401 1200 Fax: 852 2 401 3431

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

Singapore
Microchip Technology
200 Middle Road
#10-03 Prime Centre
Singapore 188980
Tel: 65 334 8870 Fax: 65 334 8850

Taiwan
Microchip Technology
10F-1C 207
Tung Hua North Road
Taipei, Taiwan, ROC
Tel: 886 2 717 7175 Fax: 886 2 545 0139

EUROPE

United Kingdom
Arizona Microchip Technology Ltd.
Unit 6, The Courtyard
Meadow Bank, Furlong Road
Bourne End, Buckinghamshire SL8 5AJ
Tel: 44 0 1628 851077 Fax: 44 0 1628 850259

France
Arizona Microchip Technology SARL
2 Rue du Buisson aux Fraises
91300 Massy - France
Tel: 33 1 69 53 63 20 Fax: 33 1 69 30 90 79

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

Italy
Arizona Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Pegaso Ingresso No. 2
Via Paracelso 23, 20041
Agrate Brianza (MI) Italy
Tel: 39 039 689 9939 Fax: 39 039 689 9883

JAPAN

Microchip Technology Intl. Inc.
Benex S-1 6F
3-18-20, Shin Yokohama
Kohoku-Ku, Yokohama
Kanagawa 222 Japan
Tel: 81 45 471 6166 Fax: 81 45 471 6122

9/22/95