Discrete Logic Replacement

Control Seven-Segment Displays

Author: *Eduardo H. Sigal*
Buenos Aires Argentina
email: esigal@feedback.net.ar

APPLICATION OPERATION

This application shows a technique for connecting two 7-segment displays directly to the PIC12C50X. For example, a 99-second timer is implemented.

Timer Description

This is a two-digit counter, which is incremented each second. An active low Reset input is available, it is not de-bounced (because it is a Reset input). Holding the Reset input low maintains the counter in '00'.

Display Technique

Common anode displays are used in this circuit.

The display cathodes are driven with four I/O lines (GP0, GP1, GP2 and GP5).The GP4 line is used to select which display is driven, through two transistors connected to each display's anode.

Each digit is driven in two steps, some segments in the first step and the other segments in the second step, depending on the number displayed. With a 60 Hz refresh, two digits driven in two steps need to be updated 240 times per second (I chose 244), that is every 4.1 mS.

A useful feature of PICmicros is the possibility of changing an I/O line from output to input and vice-versa 'on the fly'. This allows for the implementation of input and output circuitry at the same time. In fact, I used this feature to get a third state (hi-Z) at selected I/O pins in selected moments. Combining this with the high sink/source current capability and using some diodes it is possible to select which segments are lit.

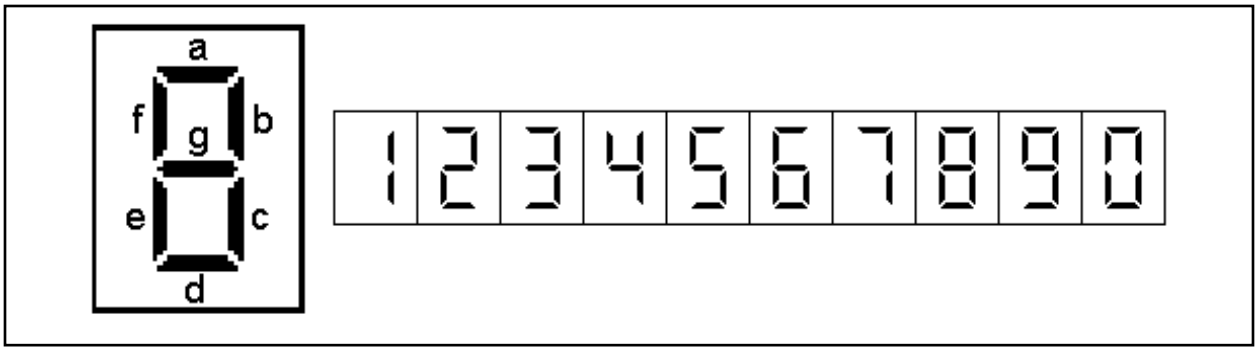
Each pin has two segments connected through two resistors (GP1 has only one). For example, GP5 is connected to segments **a** and **c** (Figure 1) trough two resistors. A diode is also connected to each segment. A '0' in GP5 will turn on both **a** and **c**. If GP2 has a '1' at the same time as the **a** segment, it will not turn on, because it is connected trough a diode to GP2. Therefore, to turn on the **a** segment, GP2 must be held at '0' level (which also turns on segment **d** and **e**) or in hi-Z (segments **d** and **e** off). It works similar with other segments.

Each display is driven in two steps. This is because it is not always possible to turn some segments on and another off at the same time (for example, you cant turn on the **a**, **f** and **g** segments without turning on the **c** segment). In those cases, the segment drive is split into two steps.

There are some additional considerations that simplify (and even make possible) this method, helping to select where the diodes should be placed:

- The segment **e** can turn on only when segment **d** is on.
- The segment **a** never is off when the segment **d** is on.

FIGURE 1: PIN PLACEMENT



With that information from the previous page, the following table can be made: Pin states for numbers display

						STEP1		STEP2							
N	a	b	c	d	e	f	g	GP5	GP2	GP1	GP0	GP5	GP2	GP1	GP0
0	1	1	1	1	1	1	0	0	H	H	1	H	0	0	
1	1	1	0	1	0	H	H	H	0	H					
2	1	1	1	1	1	0	0	0	1	H	1	H	0		
3	1	1			1	1	1	0	1	0	0	0	0	1	1
4	1	1	1	1	0	1	0	0	1	H	H	0			
5	1	1	1	1	1	0	0	1	0	H	H	H	H		
6	1	1	1	1	1	1	0	0	H	0	H	H	H	H	
7	1	1	1	0	H	0	H	H	H	H	H				
8	1	1	1	1	1	1	1	0	0	0	0	H	H	H	H
9	1	1	1	1	1	0	0	H	0	0	H	0	1	H	

Display Selection

The GP4 pin is connected to a PNP and to an NPN transistors, so, '0' in this line activates the display #1 ('units') and a '1' activates the display #2 ('tens').

Currents

The current in each segment is limited to approximately 8 mA. That is about 16 mA per pin, leaving a good margin; the anode currents are handled by the transistors.

In fact, high efficiency displays should be used to get brighter digits; I used standard displays that still work fine at these current levels.

Reset input

GP4 is switched as an input to sense the Reset switch. During this very short time, the two transistors connected to it may be working in a nonsaturated mode. This is not a problem as the GP0-GP2 and GP5 are turned to hi-z, thereby, preventing the segments from being driven. Even if those pins were not turned to hi-z, the time is so short that the segments driven during this time are not visible.

Using GP0-GP2 and GP5 as inputs is not possible without turning some segments on (affecting the number displayed).

GP3 is available to use.

Assembler/Compiler version

MPASM V0150, MPLAB V3.22.02

FIGURE 2: FLOW CHART

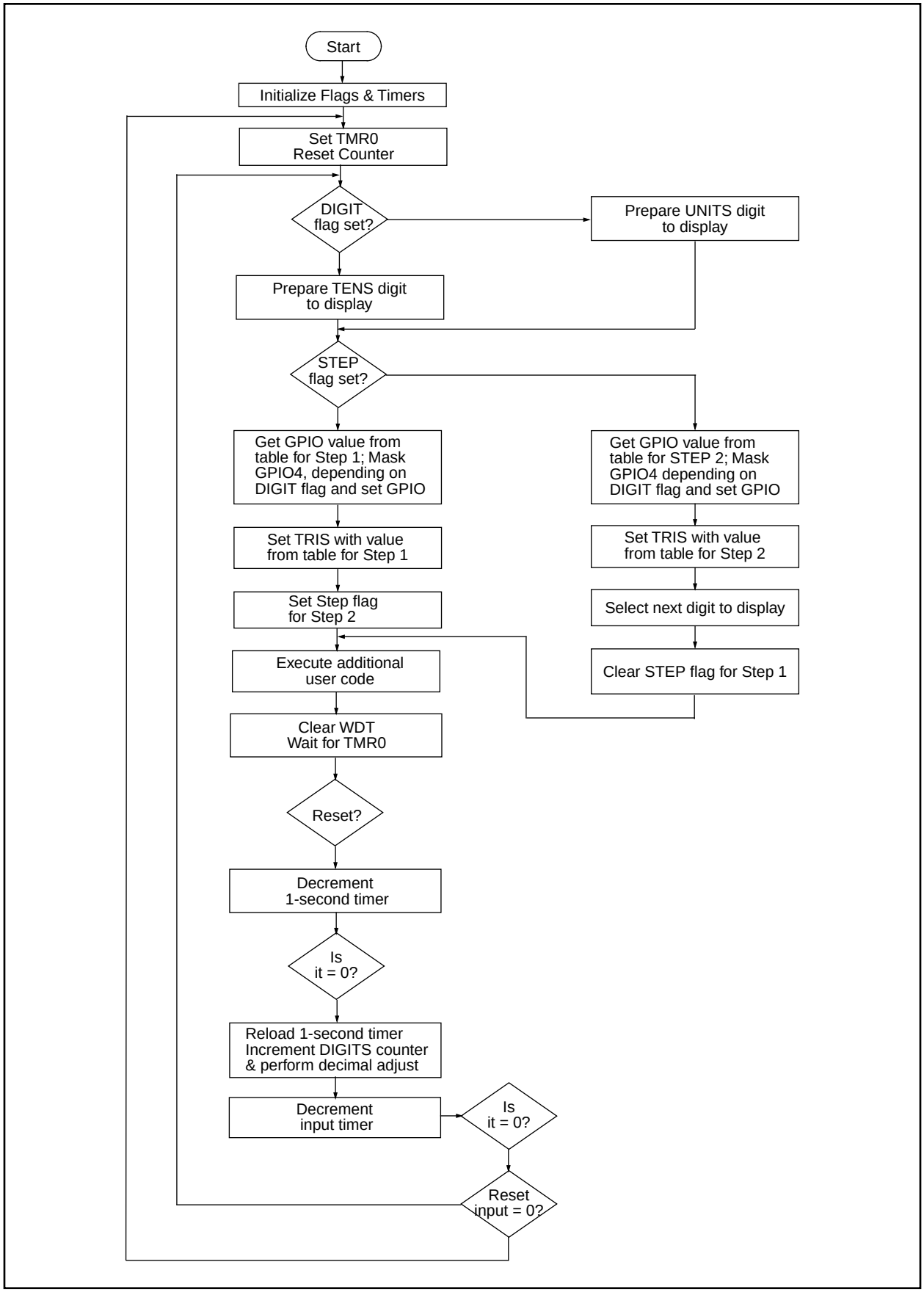
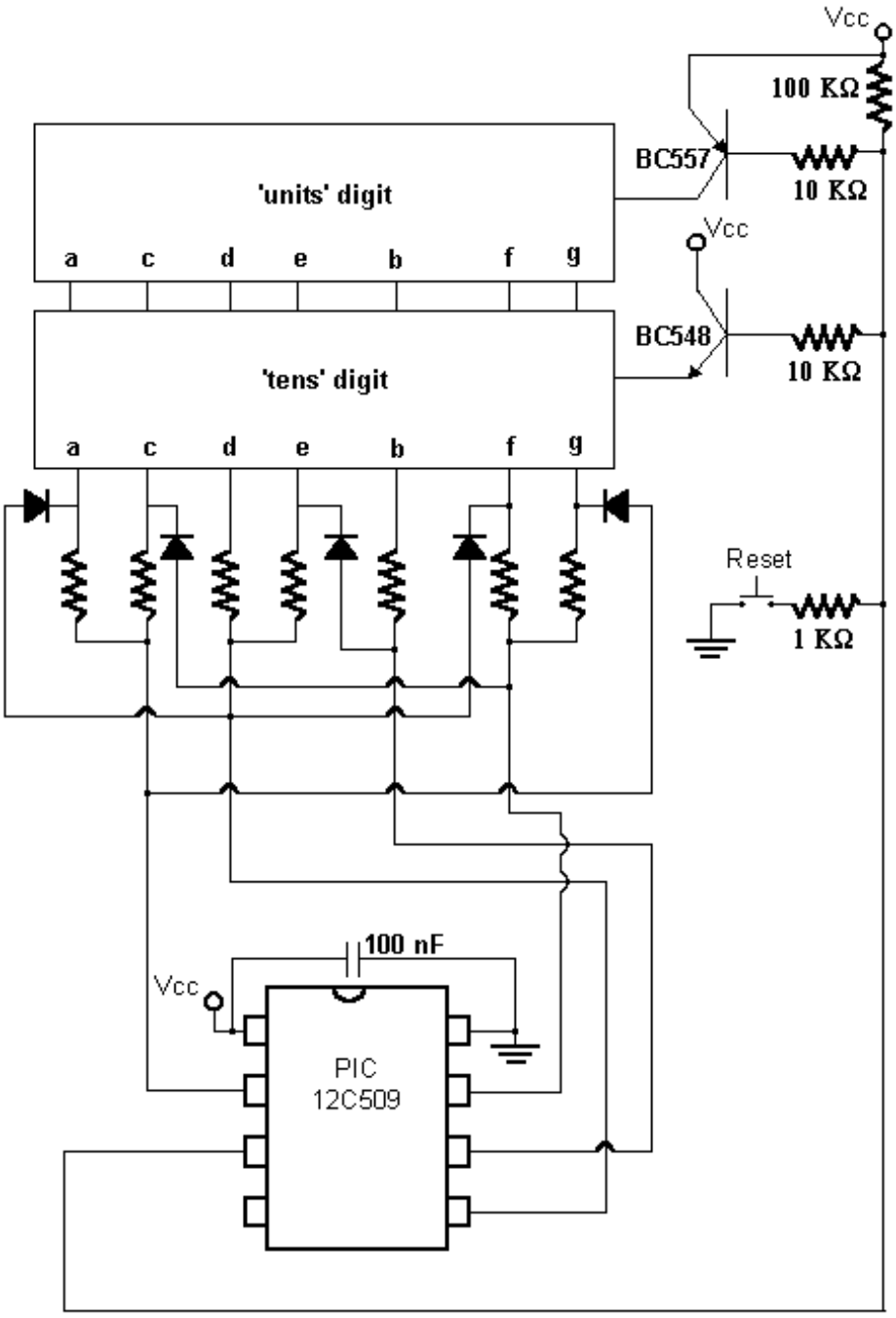


FIGURE 3: SCHEMATIC



Displays: KNIGHTBRIGHT SA56-11HDB (common anode)
All diodes 1N4148
All R = 470 ohm, unless otherwise specified

APPENDIX A: SOURCE CODE

```
LIST   P=12C509
             TITLE  '7-SEGMENTS TIMER'
#include <p12C509.inc>
;*****************************************************************
;    Program:       7SEGTMR.ASM
;
;    Revision date:
;                     8-23-97
;*****************************************************************
;
 __CONFIG b'000000001110'
;
;Timers:
;TMR0 is connected to the Prescaler which is set to 32. It is used
;for display timing. Each of the two digits is updated 62.5 times
;per second in two steps, that means, it completes 250 cycles per
;second. It is initialized to 125.
;There are two software timers based on TMR0. One is 'keytimer' allows
;the reset input to be read every 64 ms, it is initialized to 16.
;The other is 'tim' which is the timer used to measure 1 second. It is
;initialized to 250. It is decremented every TMR0 timeout, thus
;250 * 125 * 32 = 1000000; with a 4 Mhz clock it is a 1 second cycle.
;
; Timers constants
DISPLAYTIMER            equ     83h             ; Initialization value for TMR0
KEYCOUNT_BIT_VALUE      equ     4               ; Init. for 'keytimer'. Bit 4 is set
                                                ; ==> count to 16 ==> Read input every 64 mS
SEC_TIMER               equ     0FAh            ; Init. value for 'tim'
;
; GPIO inputs pin assignment
RESET_IN                equ     4               ; GPIO4 ==> Reset counter
;
; flags1 bit assignment
STEP                    equ     0               ; 0: STEP1 is displayed, 1: STEP2 is displayed
DIGIT                   equ     7               ; 0: 'unites' digit, 1: 'tens' digit
;
; RAM assignation
;digit1 and digit2 MUST be in these locations (see step2)
digits                  equ     07h             ; Memory location of two digit BCD counter
flags1                  equ     08h             ; Flags
tim                     equ     09h             ; 1 second timer
keytimer                equ     0Ah             ; Input timer
temp                    equ     0Bh             ; Temporal use register
;
;Program
             org    03FFh
             MOVLW 020h
      org    0000h
start MOVWF OSCCAL                              ; Set osc. compensation w/ original value
             CLRF flags1                        ; Initialize flags
             CLRF keytimer
             BSF keytimer, KEYCOUNT_BIT_VALUE   ; Initialize input timer

             MOVLW b'11010100'                 ; Prescaler 1/32 for Timer0, no pull-ups
             OPTION
counter_reset
             CLRF digits                        ; Clear counter
             MOVLW SEC_TIMER                    ; Initialize 1 second timer
             MOVWF tim
             MOVLW DISPLAYTIMER                 ; Set timer
             MOVWF TMR0
             GOTO digitsel
;
; One second timer routine and increment counter
```

```
;
main
             DECFSZ tim, F                          ; Decrement 'one second' timer
             GOTO readinput
reload
             MOVLW SEC_TIMER                        ; Reload  'one second' timer
             MOVWF tim
;
incda        MOVLW 07h                              ; Routine for counter increment and
             ADDWF digits, F                        ; decimal adjust
             BTFSC STATUS, DC
             GOTO test_c
             MOVLW 06h
             SUBWF digits, F
test_c  MOVLW 60h
             ADDWF digits, F
             BTFSS STATUS, C
             SUBWF digits, F
;
;Routine for RESET input reading
;
readinput
             DECFSZ keytimer, F                     ; Decrement input counter
             GOTO digitsel
             BSF keytimer, KEYCOUNT_BIT_VALUE       ; Reload input counter
             MOVLW b'00111111'                      ; Set ALL pins as input
             TRIS GPIO
             NOP                                    ; Discharge pins
;
; There is no need to debounce this input because it is a RESET input
;
res_test
             BTFSS GPIO, RESET_IN
             GOTO counter_reset                     ; Input low ==> RESET
;
; Routine for display driving
;
digitsel
             MOVF digits, W
             BTFSC flags1, DIGIT       ; 'units' digit?, skip if yes
             SWAPF digits, W           ; 'tens' digit
             ANDLW b'00001111'         ; mask digit not displayed
             MOVWF temp                ; store digit to display in a temporal register
stepsel
             BTFSC flags1, STEP  ; Select step of digit to display
             GOTO step2
;
step1 CALL led_o1                      ; Get output port STEP1 value
             BTFSS flags1, DIGIT       ; If displaying 'tens' skip
             ANDLW b'11101111'         ; Mask to select 'units' digit
             MOVWF GPIO                ; Output to port
             CALL led_iosel1           ; Get STEP1 i/o port direction selection
             TRIS GPIO                 ; Set i/o port direction
             BSF flags1, STEP          ; Change flags to STEP2
             GOTO add_code
;
step2        CALL led_o2               ; Get output port STEP2 value
             BTFSS flags1, DIGIT       ; If displaying 'tens' skip
             ANDLW b'11101111'         ; Mask to select 'units' digit
             MOVWF GPIO                ; Output to port
             CALL led_iosel2           ; Get STEP2 i/o port direction selection
             TRIS GPIO                 ; Set i/o port direction
             BCF flags1, STEP          ; Change flags to STEP1
;Select the next digit
             MOVLW b'10000000'         ; Complemet bit 7 (DIGIT flag)
             ADDWF flags1, F           ; to select next digit to display
```

```
;
;
add_code                                ;
                                        ; Place additional code here
                                        ;
wait          CLRWDT                    ;
              MOVF TMR0, W              ; Wait for timer
              BTFSS STATUS, Z
       GOTO wait
              MOVLW DISPLAYTIMER        ; Set timer
              MOVWF TMR0
              GOTO main
;
;
;
;
;STEP1: Table for ports output values selection
led_o1        MOVF temp, W
              ADDWF PCL, F
              RETLW b'00010000'   ; #0
              RETLW b'00010100'   ; #1
              RETLW b'00010001'   ; #2
              RETLW b'00010100'   ; #3
              RETLW b'00010100'   ; #4
              RETLW b'00010010'   ; #5
              RETLW b'00010000'   ; #6
              RETLW b'00010000'   ; #7
              RETLW b'00010000'   ; #8
              RETLW b'00010000'   ; #9
;
;
;STEP1: Table for ports I/O selection (Unused pins are set as inputs to
;get Hi-Z)
led_iosel1
              MOVF temp, W
              ADDWF PCL, F        ; Jump to decoding location
              RETLW b'00001011'   ; #0
              RETLW b'00001001'   ; #1
              RETLW b'00001000'   ; #2
              RETLW b'00001000'   ; #3
              RETLW b'00001000'   ; #4
              RETLW b'00001000'   ; #5
              RETLW b'00001010'   ; #6
              RETLW b'00001101'   ; #7
              RETLW b'00001000'   ; #8
;             RETLW b'00001000'   ; #9
              RETLW b'00001100'   ; #9
;
;
;STEP2: Table for ports output values selection
led_o2        MOVF temp, W
              ADDWF PCL, F
              RETLW b'00110000'   ; #0
              RETLW b'00010000'   ; #1
              RETLW b'00010100'   ; #2
              RETLW b'00010011'   ; #3
              RETLW b'00110000'   ; #4
              RETLW b'00010000'   ; #5
              RETLW b'00010000'   ; #6
              RETLW b'00010000'   ; #7
              RETLW b'00010000'   ; #8
              RETLW b'00010010'   ; #9
;
;
;STEP2: Table for ports I/O selection (Unused pins are set as inputs to
;get Hi-Z)
```

```
led_iosel2
             MOVF temp, W
             ADDWF PCL, F        ; Jump to decoding location
             RETLW b'00001100'   ; #0
             RETLW b'00101111'   ; #1
             RETLW b'00101010'   ; #2
             RETLW b'00001000'   ; #3
             RETLW b'00001110'   ; #4
             RETLW b'00101111'   ; #5
             RETLW b'00101111'   ; #6
             RETLW b'00101111'   ; #7
             RETLW b'00101111'   ; #8
             RETLW b'00101001'   ; #9
;
     END
```