Lab 4 Issues

Displaying the Frequency and Duty Cycle on the LCD

Initializing the LCD

Constant String (select from among three)
Problem:

Given a 16 bit (two byte) variable DC_HI:DC_LO which contains a (binary) duty cycle value in the range 0 thru 999

Generate the proper display string to show this value on the LCD in the form: xy.z

How do we isolate the digits of the duty cycle??

Consider an example: DC_HI:DC_LO = 765

1) Divide DC_HI:DC_LO by 10:
   765/10 = 76 with a remainder of 5
   So 5 is the rightmost digit.

2) Determine the ASCII code for 5:
The digits 0-9 are represented by the ASCII codes 0x30-0x39 respectively.
   So simply add 0x30:
   5 + 0x30 = 0x35
How do we isolate the digits of the duty cycle??

3) Now place a decimal point (0x2e) to the left of the rightmost ASCII digit.

4) Now divide the quotient from step 1) by 10:
   76/10 = 7 with a remainder of 6
   So the next digit (to the left of the decimal point is 6 and the corresponding
   ASCII code is 6 + 0x30 = 0x36

How do we isolate the digits of the duty cycle??

5. The quotient from Step 4 is the leftmost digit of the duty cycle string
   7 + 0x30 = 0x37
   So the display String is:
   0x37, 0x36, 0x2e, 0x35

Math Subroutines to the Rescue

- Problem:
  - How do we do the division by 10???
  - PIC instruction set does not include a divide instruction
  - There is no obvious way to isolate the digits of the duty cycle without doing a division operation.
  - A lookup table would be very expensive
    - Would require 1000 entries.

- You can make use of a library of prewritten math subroutines
  - These are discussed in Chapter 14 of your text
  - The MATH18 subroutine library has been installed on the machines in the Embedded Systems lab
  - You can download them for installation on your own computer, if desired.
The MATH18 Divide Subroutines

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Max. Cycles</th>
<th>Program Memory</th>
<th>Dividend</th>
<th>Divisor</th>
<th>Quotient</th>
<th>Remainder</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIVIDE18</td>
<td>10</td>
<td>32</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
</tr>
<tr>
<td>DIVIDE18A</td>
<td>10</td>
<td>32</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
</tr>
<tr>
<td>DIVIDE18B</td>
<td>10</td>
<td>32</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
</tr>
<tr>
<td>DIVIDE18C</td>
<td>10</td>
<td>32</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
</tr>
<tr>
<td>DIVIDE18D</td>
<td>10</td>
<td>32</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
</tr>
<tr>
<td>DIVIDE18E</td>
<td>10</td>
<td>32</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
</tr>
<tr>
<td>DIVIDE18F</td>
<td>10</td>
<td>32</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
</tr>
<tr>
<td>DIVIDE18G</td>
<td>10</td>
<td>32</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
</tr>
<tr>
<td>DIVIDE18H</td>
<td>10</td>
<td>32</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
</tr>
<tr>
<td>DIVIDE18I</td>
<td>10</td>
<td>32</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
</tr>
<tr>
<td>DIVIDE18J</td>
<td>10</td>
<td>32</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
</tr>
<tr>
<td>DIVIDE18K</td>
<td>10</td>
<td>32</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
</tr>
<tr>
<td>DIVIDE18L</td>
<td>10</td>
<td>32</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
</tr>
<tr>
<td>DIVIDE18M</td>
<td>10</td>
<td>32</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
</tr>
<tr>
<td>DIVIDE18N</td>
<td>10</td>
<td>32</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
</tr>
<tr>
<td>DIVIDE18O</td>
<td>10</td>
<td>32</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
</tr>
<tr>
<td>DIVIDE18P</td>
<td>10</td>
<td>32</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
</tr>
<tr>
<td>DIVIDE18Q</td>
<td>10</td>
<td>32</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
</tr>
<tr>
<td>DIVIDE18R</td>
<td>10</td>
<td>32</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
</tr>
<tr>
<td>DIVIDE18S</td>
<td>10</td>
<td>32</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
</tr>
<tr>
<td>DIVIDE18T</td>
<td>10</td>
<td>32</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
</tr>
<tr>
<td>DIVIDE18U</td>
<td>10</td>
<td>32</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
</tr>
<tr>
<td>DIVIDE18V</td>
<td>10</td>
<td>32</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
</tr>
<tr>
<td>DIVIDE18W</td>
<td>10</td>
<td>32</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
</tr>
<tr>
<td>DIVIDE18X</td>
<td>10</td>
<td>32</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
<td>A8</td>
</tr>
</tbody>
</table>

Peatman's LCD Initialization Routine

```
; LCDstr db 0x33,0x32,0x28,0x01,0x0c,0x06,0x00

; Initialize LCD
MOVLF 10,COUNT         ; Wait 0.1 second
REPEAT_
rcall LoopTime ; Call LoopTime 10 times
DEC F, COUNT
UNTIL_ .Z.
bcf PORTE,0            ; RS=0 for command
POINT LCDstr ; Set up table pointer to initialization string
tblrd*                  ; Get first byte from string into TABLAT
REPEAT_
bsf PORTE,1          ; Drive E high
movff TABLAT,PORTD   ; Send upper nibble
bcf PORTE,1          ; Drive E low so LCD will process input
rcall LoopTime ; Wait ten milliseconds
bsf PORTE,1          ; Drive E high
swapf TABLAT,W       ; Swap nibbles
movwf PORTD          ; Send lower nibble
bcf PORTE,1          ; Drive E low so LCD will process input
rcall LoopTime ; Wait ten milliseconds
tblrd+*               ; Increment pointer and get next byte
movf TABLAT,F        ; Is it zero?
UNTIL_ .Z.
return
```
Peatman's LCD Initialization Routine

```
InitLCD
MOVLF  10,COUNT         ;Wait 0.1 second
REPEAT             ;Call LoopTime 10 times
    rcall LoopTime
    decf COUNT,F
UNTIL_  .Z.
bcf PORTE,0            ;RS=0 for command
POINT  LCDstr          ;Set up table pointer to initialization string
tblrd*                  ;Get first byte from string into TABLAT
REPEAT             ;Get first byte from string into TABLAT
    bsf PORTE,1          ;Drive E high
    movff TABLAT,PORTD   ;Send upper nibble
    bcf PORTE,1          ;Drive E low so LCD will process input
    rcall LoopTime       ;Wait ten milliseconds
    bsf PORTE,1          ;Drive E high
    swapf TABLAT,W       ;Swap nibbles
    movwf PORTD          ;Send lower nibble
    rcall LoopTime       ;Wait ten milliseconds
    tblrd+*               ;Increment pointer and get next byte
    movf TABLAT,F        ;Is it zero?
UNTIL_  .Z.
return
```

LCDstr db 0x33,0x32,0x28,0x01,0x0c,0x06,0x00

**Note that this initialization routine will work only if your program employs a 10 msec. LoopTime subroutine.**

LCD Command Execution Times

<table>
<thead>
<tr>
<th>Command</th>
<th>Time (Max)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clear Display</td>
<td>850µs to 1-54ms</td>
</tr>
<tr>
<td>Display &amp; Cursor Home</td>
<td>40µs</td>
</tr>
<tr>
<td>Character Entry Mode</td>
<td>40µs</td>
</tr>
<tr>
<td>Display On/Off &amp; Cursor</td>
<td>40µs</td>
</tr>
<tr>
<td>Flash Cursor Shift</td>
<td>40µs</td>
</tr>
<tr>
<td>Function Set</td>
<td>40µs</td>
</tr>
<tr>
<td>Set COGAM Address</td>
<td>40µs</td>
</tr>
<tr>
<td>Set Display Address</td>
<td>40µs</td>
</tr>
<tr>
<td>Write Data</td>
<td>40µs</td>
</tr>
<tr>
<td>Read Data</td>
<td>40µs</td>
</tr>
<tr>
<td>Read Status</td>
<td>1µs</td>
</tr>
</tbody>
</table>

Some commands take MUCH longer than 40 microsecs. to complete.
Peatman’s LCD Initialization Routine

```assembly
; LCDstr db 0x33,0x32,0x28,0x01,0x0c,0x06,0x00

InitLCD
MOVLF  10,COUNT         ;Wait 0.1 second
REPEAT Rcall LoopTime ;Call LoopTime 10 times
    decf COUNT,F
UNTIL .Z.
bcf PORTE,0            ;RS=0 for command
POINT LCDstr ;Set up table pointer to initialization string
;Get first byte from string into TABLAT
REPEAT
    bsf PORTE,1 ;Drive E high
    movff TABLAT,PORTD ;Send upper nibble
    bcf PORTE,1 ;Drive E low so LCD will process input
    rcall LoopTime ;Wait ten milliseconds
    bsf PORTE,1 ;Drive E high
    swapf TABLAT,W ;Swap nibbles
    movwf PORTD ;Send lower nibble
UNTIL .Z.
return
```

It is probably a better idea to write your own explicit timing routines (similar to the author’s T40 subroutine) to implement these delays.