EE461 Microprocessor-based Digital Design

EEPROM

1. Gives values for the typical time $t_{Typical}$ and the maximum time $t_{Maximum}$ for erasing or writing one byte of the EEPROM.

   SOLUTION

   Parameter number D133 on page 179 of the data sheet gives the value

   $$t_{Typical} = T_{PEW,Typical} = 4 \text{ ms}$$

   and

   $$t_{Maximum} = T_{PEW,Maximum} = 8 \text{ ms.}$$

2. Assuming that the time to *initiate* a write to the EEPROM is negligible compared to the time the hardware requires to *accomplish* the write, calculate the typical time $t_{Write,Typical}$ and the maximum time $t_{Write,Maximum}$ it would take to fill all of the PIC16F874's data EEPROM.

   SOLUTION

   There are 128 bytes of Data EEPROM in the PIC16F874. The typical total time $t_{Typical}$ for a write would be

   $$\begin{aligned} t_{Typical} &= 128\ (4 \text{ ms}) \\ &= 512 \text{ ms} \end{aligned}$$

   and the maximum total time $t_{Maximum}$ for a write would be

   $$\begin{aligned} t_{Maximum} &= 128\ (8 \text{ ms}) \\ &= 1.024\text{s.} \end{aligned}$$

3. Example 3-1 in the data sheet gives a program fragment that does not change the EEPROM address: this would have to be done elsewhere in the program.

   Rewrite the example program fragment, embedding it in a loop that will cause all of the EEPROM's locations to be read into the `W` register just once. (Ignore the fact that you would need to transfer the contents from `W` to some other location in order for this to be a useful exercise.)

The basic structure of your program fragment should be

Loop initialization
Repeated loop execution.

On a PIC16F874 with oscillator frequency f_{OSC} = 10 MHz, explain how long would it take to complete your program fragment.

Calculate the ratio of the time $t_{Write,Typical}$ you calculated in question ?? for *writing* all of the EEPROM locations to the time your modified program takes to *read* all of the EEPROM locations. Comment on the significance of this ratio.

SOLUTION

```
 1  Next_Addr      equ 0x120 ; Next EEPROM address,
 2                            ; stored in Bank 2
 3 . . .
 4        ; Start at EEPROM address 0
 5        clrf     Next_Addr
 6        bsf      STATUS,RP1
 7        bcf      STATUS,RP0          ; Bank 2
 8  loop:
 9        movf     Next_Addr ,W        ; Data Memory
10        movwf    EEADR               ; Address to read
11        bsf      STATUS,RP0          ; Bank 3
12        bcf      EECON1,EEPGD        ; Point to Data memory
13        bsf      EECON1,RD           ; EE Read
14        bcf      STATUS,RP0          ; Bank 2
15        movf     EEDATA,W            ; W= EEDATA
16        incf     Next_Addr ,F        ; Next EEPROM address
17        ; Keep address in valid range (Z <= 0x7F)
18        movlw    B'01111111'
19        andwf    Next_Addr ,F
20        btfss    STATUS,Z
21        goto     loop
```

Prior to the loop there is a single cycle for line 1.

For the first 127 iterations of the loop, every one of the 12 instructions in the loop is executed and consumes a cycle, except that the `goto` instruction in line 18 takes two cycles, making 13 cycles per iteration. For the final iteration, the value of `Next Addr` rolls over from 127 to 0, setting the `Z` bit in the `STATUS` register. This time the `btfss` instruction in line 20 takes two cycles but the `goto` instruction does not execute this time, so the loop consumes just 12 cycles.

The total time T consumed, therefore, is

$$T = ((3 + 127 \times 13 + 12)\ \text{instructions})\ 1\ \frac{\text{instruction}}{1\ \text{instruction}} \left(\frac{4\ T_{OSC}}{4T_{OSC}} \right)$$

$$= \frac{1666}{10\text{MHz}} \times 4$$

$$= 666.4\ \mu\text{s}.$$

The ratio of the two times is

$$\frac{\text{typical write time}}{\text{read time}} = \frac{512\ \text{ms}}{666.4\ \mu\text{s}}$$

$$= 768.3.$$

It is a characteristic of a RUM that this ratio is much greater than 1. This ordinarily makes it infeasible to use an EEPRUM as if it were RAM. The only exception to this is when the extra time for each write operation is of no consequence. For example, in a data logging application it might be perfectly acceptable to store readings in data EEPRUM or in program Flash EEPRUM if the slow write time were irrelevant. This might be the case in storing temperature readings, for instance, since they tend to change but slowly. As long as the time to store them in the EEPRUM is less than the time between samples, storing them there would be quite feasible.