13
EEPROM data memory

One of the special features of the 16F84, the 16F818 and some other micros is the EEPROM Data Memory. This is a section of Memory not in the usual program memory space. It is a block of data like the user files, but unlike the user files the data in the EEPROM Data Memory is saved when the microcontroller is switched off, i.e. it is non-volatile. Suppose we were counting cars in and out of a car park and we lost the power to our circuit. If we stored the count in EEPROM then we could load our count file with this data and continue without loss of data, when the power returns.

To access the data, i.e. read and write to the EEPROM memory locations, we must of course instruct the microcontroller. There are 64 bytes of EEPROM memory on the 16F84, 128 on the 16F818 and 256 on the 16F819. So we must tell the micro which address we require and if we are reading or writing to it.

When reading we identify the address from 0 to 3Fh (for the 16F84) using the address register EEADR. The data is then available in register EEDATA. When writing to the EEPROM data memory we specify the data in the register EEDATA and the location in the register EEADR.

Two other files are used to enable the process, they are EECON1 and EECON2, two EEPROM control registers.

Register EECON1 and EECON2 have addresses 8 and 9 respectively in Bank1.

The Register EECON1 is shown below in Figure 13.1.

Bit 0, RD is set to a 1 to perform a read. It is cleared by the micro when the read is finished.
Bit 1, WR is set to a 1 to perform a write. It is cleared by the micro when the write is finished.
Bit 2, WREN, WRite ENable a 1 allows the write cycle, a 0 prohibits it.

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit0
EEPGD	-	-	EEIF	WRERR	WREN	WR	RD

Figure 13.1 The EECON1 register

Bit 3, WRERR reads a 1 if a write is not completed, reads a 0 if the write is completed successfully.

Bit 4, EEIF interrupt flag for the EEDATA it is a 1 if the write operation is completed, it reads 0 if it is not completed or not started (for the 16F84). This bit has another purpose for the 16F818. We have not used this bit in this book.

Bit 7, EEPGD, Program/Data EEPROM Select Bit. (Not used on 16F84.) This bit allows either the program memory or the data memory to be selected. 0 selects Data, 1 selects program memory.

Example using the EEPROM

As usual, I think the best way of understanding how this memory works is to look at a simple example.

Suppose we wish to count events, people going into a building, cars going into a carpark etc. So if we loose the power to the circuit the data is still retained. The circuit for this is shown in Figure 13.2.

Switch 1 is used to simulate the counting process and the 8 LEDs on PORTB display the count in binary. (This is a good chance to practice counting in binary.) The switch of course needs de-bouncing.

Remember the idea of this circuit, we are counting events and displaying the count on PORTB. But if we loose power – when the power is re-applied we want to continue the count as if nothing had happened.

So when we switch on we must move the previous EEPROM Data into the COUNT file.

The flowchart is shown below in Figure 13.3.

Just a couple of points before we look at the program:

1. It is a good idea to make sure the EEPROM DATA MEMORY is reset at the very beginning. This can be done by writing 00h to EEPROM DATA address 00h when we blow the program into the chip – this is done with the following lines of code.

```
ORG        2100H
DE         00H
```

   2100H is the address of the first EEPROM data memory file i.e. 00h.

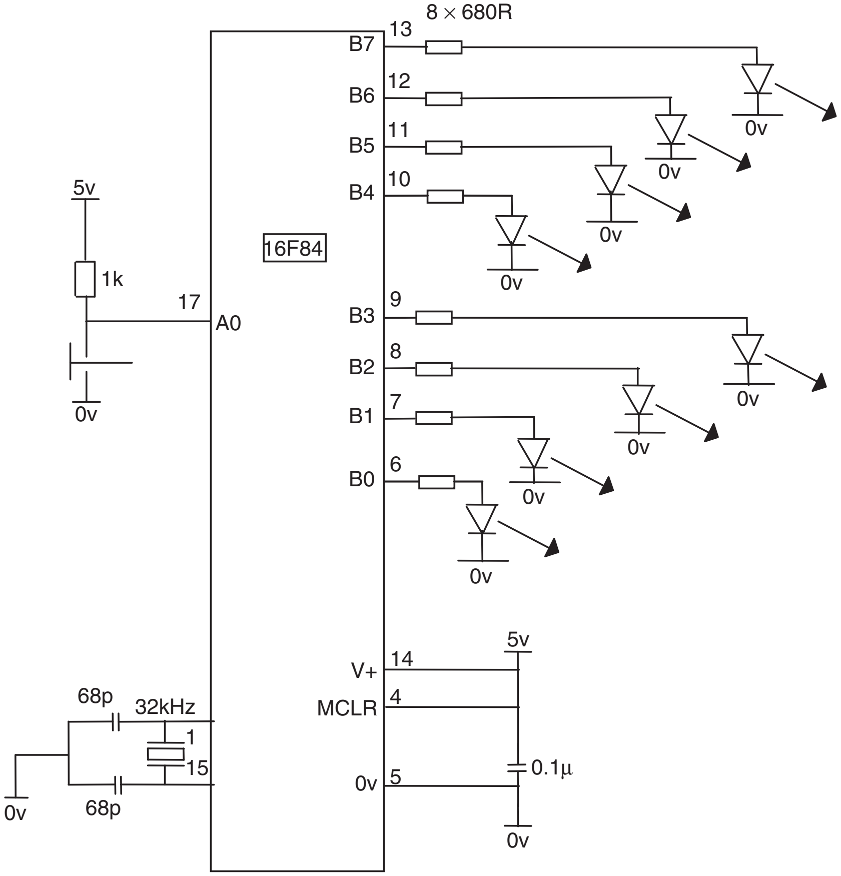


Figure 13.2 Switch press counting circuit

DE is Define EEPROM data memory, so we are initializing it with 00h, and of course 2101H is EEPROM address1 etc.

Data can also be written into the EEPROM using MPLAB, with VIEW, EEPROM and writing the data in the EEPROM box as shown in Figure 13.4.

2. Reading and Writing to EEPROM data is not as straightforward as with user files, you probably suspected that! There is a block of code you need to use – just add it to your program as required.

   When reading EEPROM data at address 0 to the file COUNT then CALL READ. The subroutine written in the header.

   When writing the file COUNT to EEPROM data address 0, CALL WRITE.

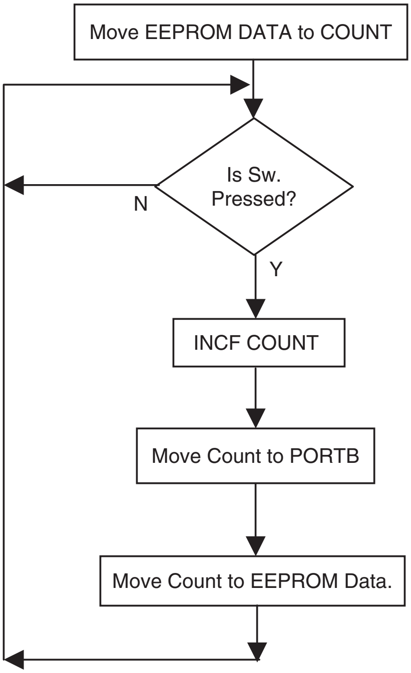
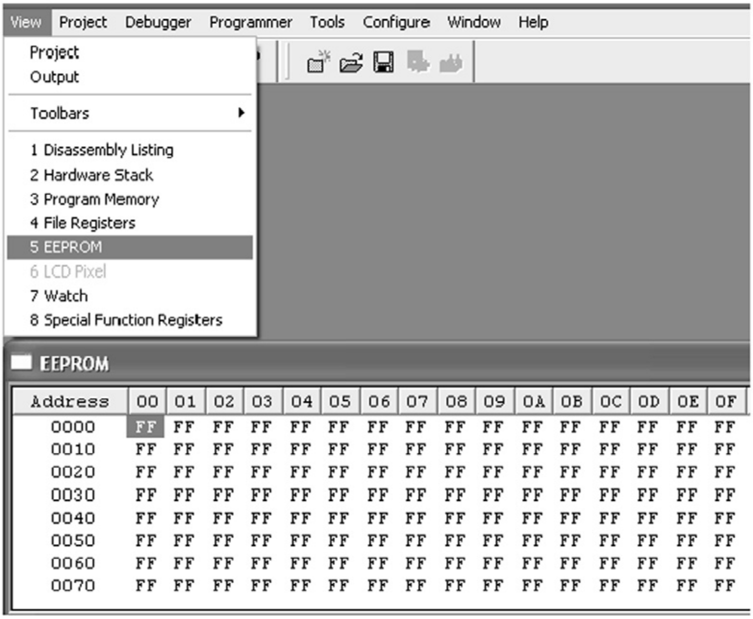


Figure 13.3 The switch press count flowchart



Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
0000	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0010	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0020	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0030	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0040	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0050	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0060	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0070	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF

Figure 13.4 Writing EEPROM data

EEPROM program code

The complete program EEDATAWR.ASM is shown below:

```
;EEDATAWR.ASM          This program will count and display switch
;                      presses.
;                      The count is saved when the power is removed
;                      and continues when the
;                      power is re-applied.


TMR0        EQU        1             ;TMR0 is FILE 1.
PORTA       EQU        5             ;PORTA is FILE 5.
PORTB       EQU        6             ;PORTB is FILE 6.
TRISA       EQU        85H           ;TRISA (the PORTA I/O selection)
TRISB       EQU        86H           ;TRISB (the PORTB I/O selection)
OPTION_R    EQU        81H           ;the OPTION register is file 81H
STATUS      EQU        3             ;STATUS is FILE 3.
ZEROBIT     EQU        2             ;ZEROBIT is Bit 2.
COUNT       EQU        0CH           ;USER RAM LOCATION.
EEADR       EQU        9             ;EEPROM address register
EEDATA      EQU        8             ;EEPROM data register
EECON1      EQU        8             ;EEPROM control register1
EECON2      EQU        9             ;EEPROM control register2
RD          EQU        0             ;read bit in EECON1
WR          EQU        1             ;Write bit in EECON1
WREN        EQU        2             ;Write enable bit in EECON1

;*******************************************************

            LIST       P=16F84       ;We are using the 16F84.
            ORG        2100H         ;ADDRESS EEADR 0
            DE         00H           ;put 00H in EEADR 0
            ORG        0             ;0 is the start address.
            GOTO       START         ;goto start!

;*******************************************************
;Configuration Bits

__CONFIG H'3FF0'    ;selects LP oscillator, WDT off, PUT on,
                    ;Code Protection disabled.

;**************************************************
```

```
;SUBROUTINE SECTION.

;0.1 SECOND DELAY
DELAYP1 CLRF     TMR0            ;Start TMR0
LOOPA   MOVF     TMR0,W          ;Read TMR0 into W
        SUBLW    .3              ;TIME - W
        BTFSS    STATUS,ZEROBIT  ;CHECK TIME-W=0
        GOTO     LOOPA
        RETLW    0               ;Return after TMR0=3

;Put EEDATA 0 into COUNT
READ    MOVLW    0               ;read EEDATA from EEADR
                                 0 into W
        MOVWF    EEADR
        BSF      STATUS,5        ;BANK1
        BSF      EECON1,RD
        BCF      STATUS,5        ;BANK0
        MOVF     EEDATA,W
        MOVWF    COUNT
        RETLW    0

;WRITE COUNT INTO EEDATA 0
WRITE   BSF      STATUS,5        ;BANK1
        BSF      EECON1,WREN     ;set WRITE ENABLE
        BCF      STATUS,5        ;BANK0
        MOVF     COUNT,W         ;move COUNT to EEDATA
        MOVWF    EEDATA
        MOVLW    0               ;set EEADR 0 to receive

EEDATA
        MOVWF    EEADR
        BSF      STATUS,5        ;BANK1
        MOVLW    55H             ;55 and AA initiates write cycle
        MOVWF    EECON2
        MOVLW    0AAH
        MOVWF    EECON2
        BSF      EECON1,WR       ;WRITE data to EEADR 0

WRDONE  BTFSC    EECON1,WR
        GOTO     WRDONE          ;wait for write cycle to complete
```

```
            BCF       EECON1,WREN
            BCF       STATUS,5          ;BANK0
            RETLW     0

;*********************************************************
;CONFIGURATION SECTION.

START       BSF       STATUS,5    ;Turn to BANK1
            MOVLW     B'00011111' ;5 bits of PORTA are I/Ps.
            MOVWF     TRISA
            MOVLW     0
            MOVWF     TRISB       ;PORTB IS OUTPUT
            MOVLW     B'00000111'
            MOVWF     OPTION_R    ;PRESCALER is /256
            BCF       STATUS,5    ;Return to BANK0
            CLRF      PORTA       ;Clears PORTA
            CLRF      PORTB       ;Clears PORTB
            CLRF      COUNT

;*********************************************************
;Program starts now.

            CALL      READ        ;read EEPROM data into COUNT
            MOVF      COUNT,W
            MOVWF     PORTB       ;Display previous COUNT (if any)
PRESS       BTFSC     PORTA,0     ;wait for switch press
            GOTO      PRESS
            CALL      DELAYP1     ;antibounce
RELEASE     BTFSS     PORTA,0     ;wait for switch release
            GOTO      RELEASE
            CALL      DELAYP1     ;antibounce

            INCF      COUNT       ;add 1 to COUNT
            MOVF      COUNT,W     ;put COUNT into W
            MOVWF     PORTB       ;move W (COUNT) to PORTB to
                                  display
            CALL      WRITE       ;write COUNT to EEPROM
                                  address 0
            GOTO      PRESS       ;return and wait for press
END
```

Microchip are continually expanding their range of microcontrollers and a new series of flash micros have been introduced, namely the 16F87X series which include 8k of program memory, 368 bytes of user RAM, 256 bytes of EEPROM data memory and an 8 channel 10 bit A/D converter. So now analogue measurements can be stored and saved in EEPROM Data!