FUNDAÇÃO EDSON QUEIROZ
UNIVERSIDADE DE FORTALEZA
ENSINANDO E APRENDENDO

Aula 19
EEPROM e Flash

Microcontroladores PIC18 – Programação em C

MICROCHIP

Prof. Ítalo Jáder Loiola Batista
Universidade de Fortaleza - UNIFOR
Centro de Ciências Tecnológicas - CCT
E-mail: italoloiola@unifor.br

Jan / 2011

Operações na Memória EEPROM Interna

- O PIC18F4520 inclui uma área de 256 bytes de memória EEPROM;
- Pode ser utilizada para salvamento de dados não voláteis, ou seja, dados que não podem ser perdidos após a desenergização do *chip*;
- Essa memória é acessada pelo controlador de memória, com o uso de dois registradores adicionais:
  - EEADR (especifica o endereço de memória a ser lido ou escrito);
  - EEDATA (para leitura ou escrita de dados na EEPROM);

Leitura da EEPROM interna

- O processo utiliza três registradores:
  - EECON1, EEADR e EEDATA;
- Passos para efetuar a leitura de um byte:

1. Configura-se o reg EEADR com o endereço a ser lido (0 a 255);
2. Configura-se o reg EECON1<7,6,4,2> para operação de leitura da EEPROM: bits EEPGD = 0, CFGS = 0, FREE = 0, WREN = 0;
3. Seta o bit RD (EECON1<0>) que provoca o inicio da operação de leitura da EEPROM. Ele é apagado automaticamente após a conclusão dela (leva 4 ciclos de clock);
4. O dado é lido na EEPROM encontra-se disponível no registrador EEDATA e pode ser utilizado na aplicação;

Registradores da EEPROM

R/W-x	R/W-x	U-0	R/W-0	R/W-x	R/W-0	R/S-0	R/S-0
EEPGD	CFGS	—	FREE	WRERR(1)	WREN	WR	RD
bit 7							bit 0

bit 7 **EEPGD:** Flash Program or Data EEPROM Memory Select bit
1 = Access Flash program memory
0 = Access data EEPROM memory

bit 6 **CFGS:** Flash Program/Data EEPROM or Configuration Select bit
1 = Access Configuration registers
0 = Access Flash program or data EEPROM memory

bit 5 **Unimplemented:** Read as '0'

bit 4 **FREE:** Flash Row Erase Enable bit
1 = Erase the program memory row addressed by TBLPTR on the next WR command (cleared by completion of erase operation)
0 = Perform write only

bit 3 **WRERR:** Flash Program/Data EEPROM Error Flag bit(1)
1 = A write operation is prematurely terminated (any Reset during self-timed programming in normal operation, or an improper write attempt)
0 = The write operation completed

bit 2 **WREN:** Flash Program/Data EEPROM Write Enable bit
1 = Allows write cycles to Flash program/data EEPROM
0 = Inhibits write cycles to Flash program/data EEPROM

bit 1 **WR:** Write Control bit
1 = Initiates a data EEPROM erase/write cycle or a program memory erase cycle or write cycle (The operation is self-timed and the bit is cleared by hardware once write is complete. The WR bit can only be set (not cleared) in software.)
0 = Write cycle to the EEPROM is complete

bit 0 **RD:** Read Control bit
1 = Initiates an EEPROM read (Read takes one cycle. RD is cleared in hardware. The RD bit can only be set (not cleared) in software. RD bit cannot be set when EEPGD = 1 or CFGS = 1.)
0 = Does not initiate an EEPROM read

Registradores associados com a EEPROM

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	49
EEADR	EEPROM Address Register								51
EEDATA	EEPROM Data Register								51
EECON2	EEPROM Control Register 2 (not a physical register)								51
EECON1	EEPGD	CFGS	—	FREE	WRERR	WREN	WR	RD	51
IPR2	OSCFIP	CMIP	—	EEIP	BCLIP	HLVDIP	TMR3IP	CCP2IP	52
PIR2	OSCFIF	CMIF	—	EEIF	BCLIF	HLVDIF	TMR3IF	CCP2IF	52
PIE2	OSCFIE	CMIE	—	EEIE	BCLIE	HLVDIE	TMR3IE	CCP2IE	52

Leitura da EEPROM interna

```
1  #include <p18f4520.h>
2  #include <stdio.h>
3  #include "pic_simb.h"
4
5  #pragma config OSC = XT, WDT = OFF, MCLRE = ON
6  #pragma config DEBUG = OFF, LVP = OFF, PWRT = ON, BOREN = OFF
7
8  #pragma romdata meus_dados = 0xF00000
9  rom unsigned char tabela[5] =
10 {0x00, 0x01, 0x02, 0x03, 0x04};
11 rom unsigned char string[] = {"Teste"};
12 volatile char temp;
13
14 // Lê um byte do endereço especificado da EEPROM interna
15 unsigned char eeprom_int_read_byte(unsigned char addr)
16 {
17    EEADR = addr;    // EEADR recebe o endereço passado para a função
18    EECON1 = bRD;    // seta o bit de leitura em EECON1
19    return (EEDATA);    // retorna o conteúdo lido (de EEDATA)
20 }
21 void main(void)
22 {
23    temp = eeprom_int_read_byte(0);    // lê o conteúdo do endereço 0 da EEPROM (=
24    temp = eeprom_int_read_byte(5);    // lê o conteúdo do endereço 5 da EEPROM (=
25    while(1);       // loop
26 }
27
```

Escrita na EEPROM interna

- O processo utiliza três registradores:
  - EECON1, EEADR e EEDATA;
- Passos para efetuar a leitura de um byte:

1. Configura-se o reg EEADR com o endereço EEPROm que será reprogramado (0 a 255);
2. Configura-se o reg EECON1<7,6,4,2> para operação de escrita da EEPROM: bits EEPGD = 0, CFGS = 0;
3. Habilita-se a operação de escrita, bit WREN = 1 (EECON1<2>)
4. Escreve-se o dado no registrador EEDATA.
5. Em seguida é necessário desabilitar as interrupções, pois o processo seguinte não deve ser executado fora de seqüência, apaga-se o bit GIE (INTCON<7>);

Escrita na EEPROM interna

- cont.

6. Escreve-se a senha de segurança no registrador EECON2: 0x55 e em seguida 0xAA;
7. Seta-se o bit WR no registrador EECON1, isso provoca o início da operação de escrita (leva aproximadamente 4 ms para ser completado);
8. Reabilitam-se as interrupções (caso estiverem habilitadas antes do início da operação);
9. Após completada a operação de escrita, o bit WR é automaticamente apagado e o bit EEIF é setado, podendo gerar uma interrupção caso esteja habilitada;

Obs.: Em casos em que se necessita escrever múltiplos bytes na EEPROM, é vantajoso utilizar a interrupção da EEPROM;

Escrita na EEPROM interna

```
1  #include <p18f4520.h>
2  #include <stdio.h>
3  #include "pic_simb.h"
4  #pragma config OSC = XT, WDT = OFF, MCLRE = ON
5  #pragma config DEBUG = OFF, LVP = OFF, PWRT = ON, BOREN = OFF
6  volatile unsigned char eeprom_int_num_byte_write;
7  volatile unsigned char *eeprom_wr_data_ptr;
8  unsigned char teste[5]={0x12,0x34,0x56,0x78,0x9A};
9  #pragma interrupt EEPROM_ISR
10 void EEPROM_ISR(void)
11 {
12    PIR2bits.EEIF = 0;   // apaga o flag de interrupção
13    // retorna se não houverem mais bytes para escrever
14    if (!eeprom_int_num_byte_write) return;
15    EEDATA = *eeprom_wr_data_ptr++;   // novo dado a ser escrito
16    EECON2 = 0x55;      // senha de escrita na EEPROM
17    EECON2 = 0xAA;
18    EECON1bits.WR = 1;   // seta WR e inicia a escrita na EEPROM
19    EEADR++;      // incrementa o endereço da EEPROM
20    eeprom_int_num_byte_write--;      // decrementa o contador de bytes
21 }
22 #pragma code isr_baixa = 0x0008
23 void ISR_baixa_prioridade(void)
24 {
25    if (PIR2bits.EEIF) _asm BRA EEPROM_ISR _endasm
26 }
27 #pragma code
```

Escrita na EEPROM interna

```
28  // Escreve "num" bytes apontados por "*data" na EEPROM a partir do endereço espec
29  // cado por "addr"
30  void eeprom_int_byte_write(char addr, char *data, char num)
31  {
32     while (EECON1bits.WR);    // se uma escrita estiver em andamento, aguarda
33     EEADR = addr;       // configura o endereço inicial da escrita
34     eeprom_wr_data_ptr = data;    // configura o ponteiro de dados
35     eeprom_int_num_byte_write = num;    // número de bytes a serem escritos
36     EECON1 = bWREN;          // configura EECON1 para escrita na EEPROM
37     PIR2bits.EEIF = 1;       // seta flag de interrupção da EEPROM
38     PIE2bits.EEIE = 1;       // habilita a interrupção da EEPROM
39  }
40  void main(void)
41  {
42     INTCON = bGIE | bPEIE;    // habilita as interrupções
43     eeprom_int_byte_write(0,teste,5);    // escreve 5 bytes do array teste na EEPRC
44     while(1);       // loop
45  }
```

Próxima Aula

Aula 20

Exemplos de Projetos