Aula 08
Portas de Entrada e Saída – Parte I

(botão, led e display de 7 seg)

Microcontroladores PIC18 – Programação em C

Prof. Ítalo Jáder Loiola Batista

Universidade de Fortaleza - UNIFOR

Centro de Ciências Tecnológicas - CCT

E-mail: italoloiola@unifor.br

Jan / 2011

Portas de E/S

- O PIC18F4520 possui quatro portas de E/S com 8 bits cada e uma com 4 bits, são elas:
  - PORTA (RA7:RA0)
  - PORTB (RB7:RB0)
  - PORTC (RC7:RC0)
  - PORTD (RD7:RD0)
  - PORTE (RE3:RE0)

Portas de E/S

- Cada porta de E/S possui basicamente três registradores SFRs para o seu controle:
  - Registradores TRIS:
    - Controla a direção de cada pino da porta;
  - Registradores PORT:
    - Controla a leitura/escrita em cada pino da porta;
  - Registradores LAT:
    - Funcionam de forma praticamente idêntica ao aos registradores PORT, mas com a diferença de que a leitura de um registrador LAT retorna o último valor escrito nele e não o estado externo do pino;
    - Utilizado nos casos em que a carga externa ligada ao pino possui uma capacitância muito elevada ou velocidades de clock elevadas que causam alterações indesejadas;

Portas de E/S - FSRs

FFFh	TOSU
FFEh	TOSH
FFDh	TOSL
FFCh	STKPTR
FFBh	PCLATU
FFAh	PCLATH
FF9h	PCL
FF8h	TBLPTRU
FF7h	TBLPTRH
FF6h	TBLPTRL
FF5h	TABLAT
FF4h	PRODH
FF3h	PRODL
FF2h	INTCON
FF1h	INTCON2
FF0h	INTCON3
FEFh	INDF0(1)
FEEh	POSTINC0(1)
FEDh	POSTDEC0(1)
FECh	PREINC0(1)
FEBh	PLUSW0(1)
FEAh	FSR0H
FE9h	FSR0L
FE8h	WREG
FE7h	INDF1(1)
FE6h	POSTINC1(1)
FE5h	POSTDEC1(1)
FE4h	PREINC1(1)
FE3h	PLUSW1(1)
FE2h	FSR1H
FE1h	FSR1L
FE0h	BSR

FDFh	INDF2(1)
FDEh	POSTINC2(1)
FDDh	POSTDEC2(1)
FDCh	PREINC2(1)
FDBh	PLUSW2(1)
FDAh	FSR2H
FD9h	FSR2L
FD8h	STATUS
FD7h	TMR0H
FD6h	TMR0L
FD5h	T0CON
FD4h	—(2)
FD3h	OSCCON
FD2h	HLVDCON
FD1h	WDTCON
FD0h	RCON
FCFh	TMR1H
FCEh	TMR1L
FCDh	T1CON
FCCh	TMR2
FCBh	PR2
FCAh	T2CON
FC9h	SSPBUF
FC8h	SSPADD
FC7h	SSPSTAT
FC6h	SSPCON1
FC5h	SSPCON2
FC4h	ADRESH
FC3h	ADRESL
FC2h	ADCON0
FC1h	ADCON1
FC0h	ADCON2

FBFh	CCPR1H
FBEh	CCPR1L
FBDh	CCP1CON
FBCh	CCPR2H
FBBh	CCPR2L
FBAh	CCP2CON
FB9h	—(2)
FB8h	BAUDCON
FB7h	PWM1CON(3)
FB6h	ECCP1AS(3)
FB5h	CVRCON
FB4h	CMCON
FB3h	TMR3H
FB2h	TMR3L
FB1h	T3CON
FB0h	SPBRGH
FAFh	SPBRG
FAEh	RCREG
FADh	TXREG
FACh	TXSTA
FABh	RCSTA
FAAh	—(2)
FA9h	EEADR
FA8h	EEDATA
FA7h	EECON2(1)
FA6h	EECON1
FA5h	—(2)
FA4h	—(2)
FA3h	—(2)
FA2h	IPR2
FA1h	PIR2
FA0h	PIE2

F9Fh	IPR1
F9Eh	PIR1
F9Dh	PIE1
F9Ch	—(2)
F9Bh	OSCTUNE
F9Ah	—(2)
F99h	—(2)
F98h	—(2)
F97h	—(2)
F96h	TRISE(3)
F95h	TRISD(3)
F94h	TRISC
F93h	TRISB
F92h	TRISA
F91h	—(2)
F90h	—(2)
F8Fh	—(2)
F8Eh	—(2)
F8Dh	LATE(3)
F8Ch	LATD(3)
F8Bh	LATC
F8Ah	LATB
F89h	LATA
F88h	—(2)
F87h	—(2)
F86h	—(2)
F85h	—(2)
F84h	PORTE(3)
F83h	PORTD(3)
F82h	PORTC
F81h	PORTB
F80h	PORTA

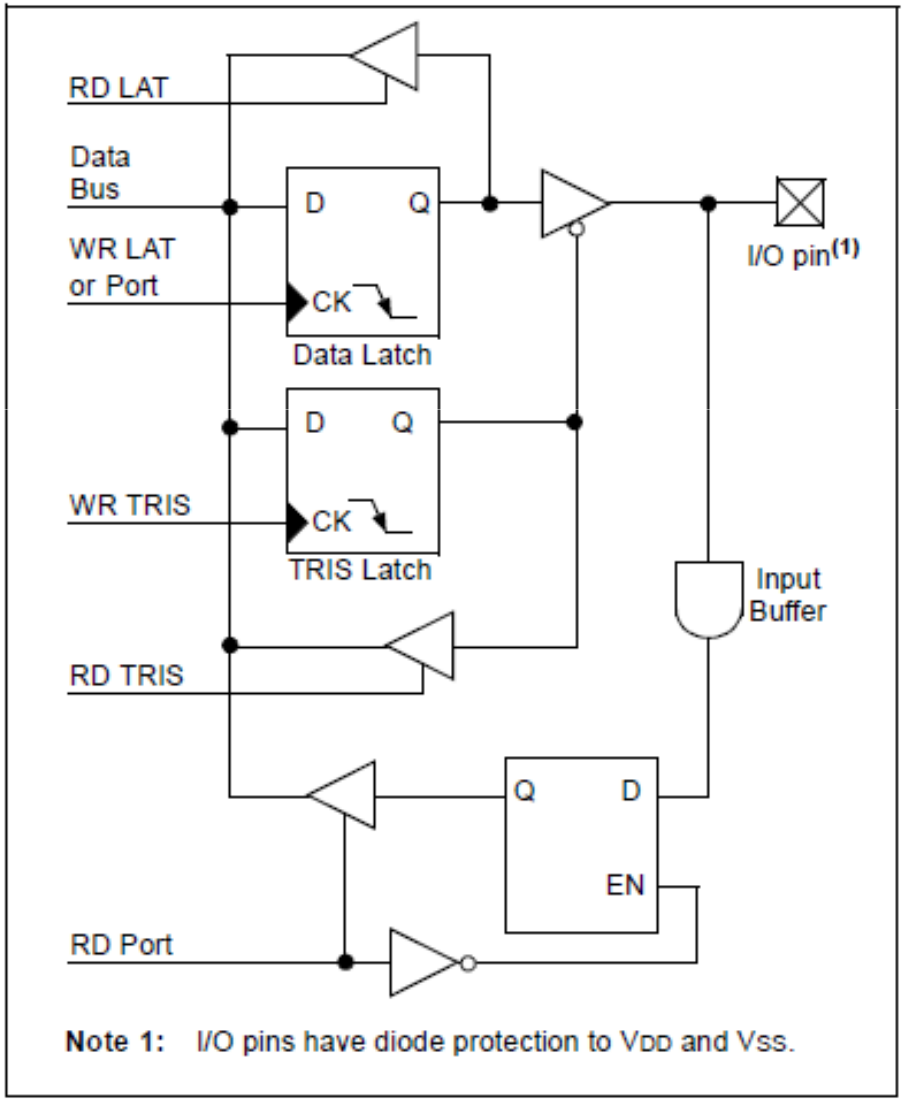
Portas de E/S

- Cada bit de cada PORT está associado ao respectivo pino do PIC;
- Cada pino pode ser configurado como entrada ou saída independente dos demais
- Então, para fazer com que um determinado pino de um dos PORTs funcione como entrada ou saída é necessário configurar os FSRs TRISX (para PORTX)

Níveis de Tensão

4,5 V < Vdd < 5,5 V			
Entrada	**Min**	**Máx**	**Nível Lógico**
	0 V	0,8 V	0
	2 V	Vdd	1
Saída			
V_{OL} (tensão de saída baixa)	-	0,6 V	0
V_{OH} (tensão de saída alta)	Vdd-0,7 V	-	1

Diagrama Simplificado de um Pino de E/S



Configurando Pino como E/S – Porta A

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PORTA	RA7(1)	RA6(1)	RA5	RA4	RA3	RA2	RA1	RA0
LATA	LATA7(1)	LATA6(1)	PORTA Data Latch Register (Read and Write to Data Latch)					
TRISA	TRISA7(1)	TRISA6(1)	PORTA Data Direction Register					

- Para *n* igual a 0, 1, 2, 3, 4, 5, 6 e 7:
- Bit n (R/W): TRISBn : Configura o pino *RBn* como entrada ou saída
  - 1 = *RBn* configurado como entrada
  - 0 = *RBn* configurado como saída
- Os pinos RA6 e RA7 são multiplexados com o oscilador. Sua configuração como pino de E/S depende do bit de configuração “Oscillator”;

Lendo e Escrevendo no PORTA

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PORTA	RA7(1)	RA6(1)	RA5	RA4	RA3	RA2	RA1	RA0
LATA	LATA7(1)	LATA6(1)	PORTA Data Latch Register (Read and Write to Data Latch)					
TRISA	TRISA7(1)	TRISA6(1)	PORTA Data Direction Register					

- Para n igual a 0, 1, 2, 3, 4, 5, 6 e 7:
- Quando RAn configurado como entrada:
  - Bit n (R): RAn
    - 1 = Nível lógico 1 lido no pino de entrada
    - 0 = Nível lógico 0 lido no pino de entrada
- Quando RAn configurado como saída:
  - Bit n (R/W): RAn
    - 1 = Nível lógico 1 colocado no pino de saída
    - 0 = Nível lógico 0 colocado no pino de saída

Lendo e Escrevendo no PORTA

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PORTA	RA7(1)	RA6(1)	RA5	RA4	RA3	RA2	RA1	RA0
LATA	LATA7(1)	LATA6(1)	PORTA Data Latch Register (Read and Write to Data Latch)					
TRISA	TRISA7(1)	TRISA6(1)	PORTA Data Direction Register					
ADCON1	—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0
CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0
CVRCON	CVREN	CVROE	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0

- Para configurar os bits RA3:RA0 como E/S digital, é preciso setar os quatro bits menos significativos do registrador ADCON1 (SFR do periférico do módulo conversor A/D)
- Os pinos RA5 e RA6 dependem dos bits de configuração “Oscillator” para funcionar como E/S;

Registrador da Porta B

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	52
LATB	PORTB Data Latch Register (Read and Write to Data Latch)								52
TRISB	PORTB Data Direction Register								52
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	49
INTCON2	$\overline{\text{RBPU}}$	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP	49
INTCON3	INT2IP	INT1IP	—	INT2IE	INT1IE	—	INT2IF	INT1IF	49
ADCON1	—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0	51

Registrador da Porta C

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	52
LATC	PORTC Data Latch Register (Read and Write to Data Latch)								52
TRISC	PORTC Data Direction Register								52

Registrador da Porta D

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
PORTD	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	52
LATD	PORTD Data Latch Register (Read and Write to Data Latch)								52
TRISD	PORTD Data Direction Register								52
TRISE(1)	IBF	OBF	IBOV	PSPMODE	—	TRISE2	TRISE1	TRISE0	52
CCP1CON	P1M1(1)	P1M0(1)	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	51

Registrador da Porta E

R-0	R-0	R/W-0	R/W-0	U-0	R/W-1	R/W-1	R/W-1
IBF	OBF	IBOV	PSPMODE	—	TRISE2	TRISE1	TRISE0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **IBF:** Input Buffer Full Status bit
1 = A word has been received and waiting to be read by the CPU
0 = No word has been received

bit 6 **OBF:** Output Buffer Full Status bit
1 = The output buffer still holds a previously written word
0 = The output buffer has been read

bit 5 **IBOV:** Input Buffer Overflow Detect bit (in Microprocessor mode)
1 = A write occurred when a previously input word has not been read (must be cleared in software)
0 = No overflow occurred

bit 4 **PSPMODE:** Parallel Slave Port Mode Select bit
1 = Parallel Slave Port mode
0 = General purpose I/O mode

bit 3 **Unimplemented:** Read as '0'

bit 2 **TRISE2:** RE2 Direction Control bit
1 = Input
0 = Output

bit 1 **TRISE1:** RE1 Direction Control bit
1 = Input
0 = Output

bit 0 **TRISE0:** RE0 Direction Control bit
1 = Input
0 = Output

Registrador da Porta E

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
PORTE	—	—	—	—	RE3(1,2)	RE2	RE1	RE0	52
LATE(2)	—	—	—	—	—	LATE Data Latch Register			52
TRISE	IBF	OBF	IBOV	PSPMODE	—	TRISE2	TRISE1	TRISE0	52
ADCON1	—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0	51

Registrador da Porta E

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
PORTE	—	—	—	—	RE3(1,2)	RE2	RE1	RE0	52
LATE(2)	—	—	—	—	—	LATE Data Latch Register			52
TRISE	IBF	OBF	IBOV	PSPMODE	—	TRISE2	TRISE1	TRISE0	52
ADCON1	—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0	51

Leitura do Estado de um Pino

- Exemplo 1

```
#include <p18f4520.h>      //diretiva de compilação
/*********************************************************************
Esta funcao inicializa os resgistradores SFRs.*/
 void Inic_Regs (void)
{
   TRISA = 0x00;          //PORTA saída
   TRISB = 0x07;          //RB2:RB0 entrada e demais pinos do PORTB saida
   TRISC = 0x00;          //PORTC saida
   TRISD = 0x00;          //PORTD saida
   TRISE = 0x00;          //PORTE saida
   ADCON1 = 0x0F;          //configura PORTA e PORTE como digitais
   PORTA = 0;             //limpa PORTA
   PORTB = 0;             //limpa PORTB
   PORTC = 0;             //limpa PORTC
   PORTD = 0;             //limpa PORTD
   PORTE = 0;             //limpa PORTE
}
//*********************************************************************
void main(void)       //função main
{
   Inic_Regs ();                          //chamada a funcão
   while (1)                             //loop infinito
   {
      if(PORTB & 0x01)PORTD = 0x7F;          //seta pino RD7 se pino RB0 = 0
      else   if(PORTB & 0x02)PORTD = 0xBF;    //se não, limpa pino RD6 se pino RB1 = 0
      else   if(PORTB & 0x04)PORTD = 0xDF;    //se não, limpa pino RD5 se pino RB2 = 0
      else PORTD = 0xFF;                    //se não, seta PORTD se RB2:RB0 = 111
   }
//*********************************************************************
}
```

Código Errado!!!!!

Leitura do Estado de um Pino

- Exemplo 2

```
while (1)                                   //loop infinito
{
   x = PORTB;
   if(x==0x06)PORTD = 0x7F;                 //seta pino RD7 se pino RB0 = 0
   else   if(x==0x05)PORTD = 0xBF;           //se não, limpa pino RD6 se pino RB1 = 0
   else   if(x==0x03)PORTD = 0xDF;           //se não, limpa pino RD5 se pino RB2 = 0
   else PORTD = 0xFF;                          //se não, seta PORTD se RB2:RB0 = 111
}
```

Leitura do Estado de um Pino

- Exemplo 3

```
while (1)                              //loop infinito
{
   if(PORTBbits.RB0==0)PORTD = 0x7F;          //seta pino RD7 se pino RB0 = 0
   else   if(PORTBbits.RB1==0)PORTD = 0xBF;    //se não, limpa pino RD6 se pino RB1 = 0
   else   if(PORTBbits.RB2==0)PORTD = 0xDF;    //se não, limpa pino RD5 se pino RB2 = 0
   else PORTD = 0xFF;                          //se não, seta PORTD se RB2:RB0 = 111
}
```

Leitura do Estado de um Pino

- Exemplo 4

```
while (1)                              //loop infinito
{
   if(PORTBbits.PORTB0==0)PORTD = 0x7F;          //seta pino RD7 se pino RB0 = 0
   else   if(PORTBbits.PORTB1==0)PORTD = 0xBF;    //se não, limpa pino RD6 se pino RB1 = 0
   else   if(PORTBbits.PORTB2==0)PORTD = 0xDF;    //se não, limpa pino RD5 se pino RB2 = 0
   else PORTD = 0xFF;                          //se não, seta PORTD se RB2:RB0 = 111
}
```

Leitura do Estado de um Pino

- Seta o bit 0 do TRISB

```
TRISBbits.TRISB0 = 1; // Configura RB0 como entrada
```

A leitura do estado do pino pode ser feita através da linha de código:

```
if (PORTBbits.PORTB0)
{

}
```

Leitura do Estado de um Pino

- Podemos criar símbolos para simplificar o acesso aos pinos com a diretiva #define:

```
#define   BOTAO    PORTBbits.PORTB0
```

- Seta o bit 0 do TRISB

```
TRISBbits.TRISB0 = 1; // Configura RB0 como entrada
```

A leitura do estado do pino pode ser feita através da linha de código:

```
if (BOTAO)
{

}
```

Escrita em um Pino

Por que o valores 0x7F, 0xBF, 0xDF atribuídos ao Registrador PORTD fazem os respectivos LEDs emitirem?

```
8  #include <p18f4520.h>      //diretiva de compilação
9  //*********************************************************************
10  void Inic_Regs (void);//protótipos de funções
11 //*********************************************************************
12 void main(void)                //função main
13 {
14    Inic_Regs ();              //chamada a funcão
15    while (1)                 //loop infinito
16    {
17       if (PORTBbits.RB0==0)PORTD = 0x7F;          //led3 emite se botão BT1 estiver pressionado
18       else   if (PORTBbits.RB1==0)PORTD = 0xBF;   //se não, led2 emite se botão BT2 estiver pre
19       else   if (PORTBbits.RB2==0)PORTD = 0xDF;   //se não, led1 emite se botão BT3 estiver pre
20       else    PORTD = 0xff;                      //se não, apaga todos os leds se nenhum botão  es
21    }
22 }
23 /*********************************************************************
24 Esta funcao inicializa os resgistradores SFRs.*/
25  void Inic_Regs (void)
26 {
27    TRISA = 0x00;          //PORTA saída
28    TRISB = 0x07;          //pinos RB2:RB0 entrada e demais pinos do PORTB saída
29    TRISC = 0x00;          //PORTC saida
30    TRISD = 0x00;          //PORTD saida
31    TRISE = 0x00;          //PORTE saida
32    ADCON1 = 0x0F;          //configura pinos dos PORTA e PORTE como digitais
33    PORTA = 0;             //limpa PORTA
34    PORTB = 0;             //limpa PORTB
35    PORTC = 0;             //limpa PORTC
36    PORTD = 0xFF;          //apaga LEDs
37    PORTE = 0;             //limpa PORTE
38 //*********************************************************************
39 }
```

Escrita em um Pino

- Seta o bit 0 do TRISB

```
TRISBbits.TRISB0 = 0; // Configura RB0 como saída
```

A escrita do pino pode ser feita através da linha de código:

```
PORTBbits.PORTB0 = 1; // Coloca RB0 em nível lógico alto

PORTBbits.PORTB0 = 0; // Coloca RB0 em nível lógico baixo
```

Escrita em um Pino

- Podemos criar símbolos para simplificar o acesso aos pinos com a diretiva #define:

```
#define   LED1    PORTBbits.PORTB0
```

- Seta o bit 0 do TRISB

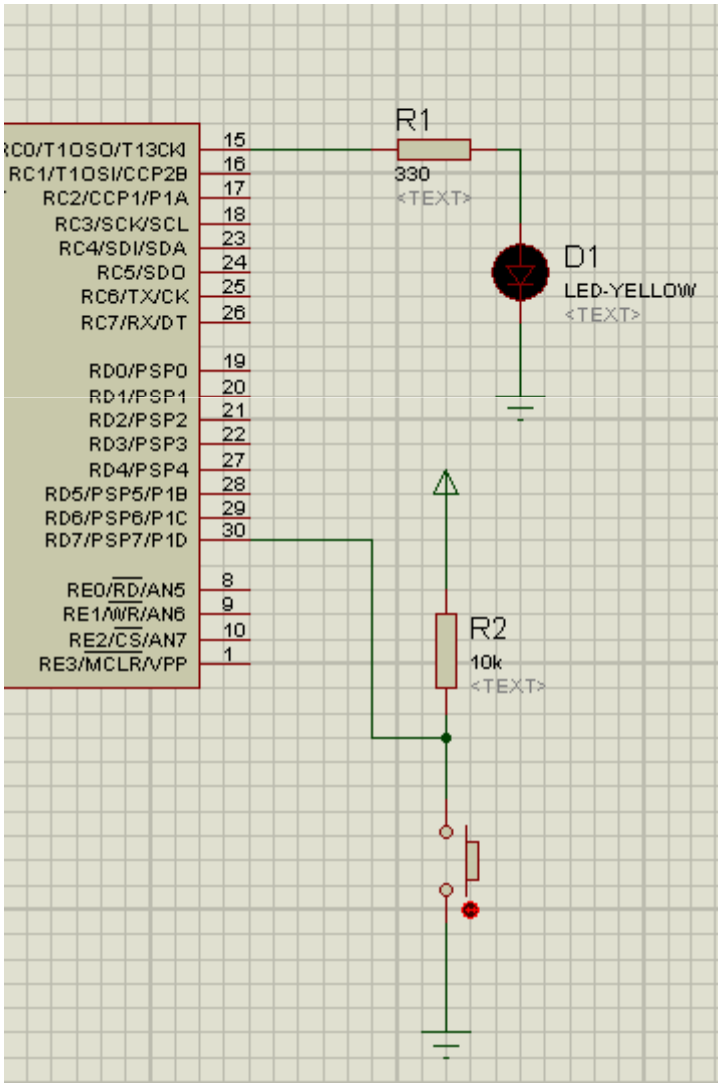
```
TRISBbits.TRISB0 = 0; // Configura RB0 como saída
```

A leitura do estado do pino pode ser feita através da linha de código:

```
LED1 = 1; // Coloca RB0 em nível lógico alto
```

```
LED1 = 0; // Coloca RB0 em nível lógico baixo
```

Acionar LEDs e Ler Botão



Como multiplexar funções de Entrada e Saída num único pino?

```
1  #include <p18f4520.h>
2  #include <stdio.h>
3
4  #pragma config OSC = XT, WDT = OFF, MCLRE = OFF
5  #pragma config DEBUG = OFF, LVP = OFF, PWRT = ON
6
7  #define TECLA_S1   PORTBbits.RB0
8  #define LED_L1   LATBbits.LATB0
9
10 void atraso(void)
11 {
12    unsigned char aux;
13    for(aux=255; aux; aux--);
14 }
15
16 void main(void)
17 {
18    ADCON1 = 0x0F;          // desliga entradas analógicas
19    while(1)
20    {
21       TRISBbits.TRISB0 = 1;      // configura RB0 como entrada
22       if (!TECLA_S1)
23       {                 // se a tecla S1 estiver pressionada
24          TRISBbits.TRISB0 = 0;   // configura RB0 como saída
25          LED_L1 = 1;
26       } else
27       {                 // se a tecla S1 estiver liberada
28          TRISBbits.TRISB0 = 0;   // configura RB0 como saída
29          LED_L1 = 0;
30       }
31       atraso();          // chama a função de atraso
32    }
```

Resistores de pull-up - FSRs

FFFh	TOSU	FDFh	INDF2(1)	FBFh	CCPR1H	F9Fh	IPR1
FFEh	TOSH	FDEh	POSTINC2(1)	FBEh	CCPR1L	F9Eh	PIR1
FFDh	TOSL	FDDh	POSTDEC2(1)	FBDh	CCP1CON	F9Dh	PIE1
FFCh	STKPTR	FDCh	PREINC2(1)	FBCh	CCPR2H	F9Ch	—(2)
FFBh	PCLATU	FDBh	PLUSW2(1)	FBBh	CCPR2L	F9Bh	OSCTUNE
FFAh	PCLATH	FDAh	FSR2H	FBAh	CCP2CON	F9Ah	—(2)
FF9h	PCL	FD9h	FSR2L	FB9h	—(2)	F99h	—(2)
FF8h	TBLPTRU	FD8h	STATUS	FB8h	BAUDCON	F98h	—(2)
FF7h	TBLPTRH	FD7h	TMR0H	FB7h	PWM1CON(3)	F97h	—(2)
FF6h	TBLPTRL	FD6h	TMR0L	FB6h	ECCP1AS(3)	F96h	TRISE(3)
FF5h	TABLAT	FD5h	T0CON	FB5h	CVRCON	F95h	TRISD(3)
FF4h	PRODH	FD4h	—(2)	FB4h	CMCON	F94h	TRISC
FF3h	PRODL	FD3h	OSCCON	FB3h	TMR3H	F93h	TRISB
FF2h	INTCON	FD2h	HLVDCON	FB2h	TMR3L	F92h	TRISA
FF1h	INTCON2	FD1h	WDTCON	FB1h	T3CON	F91h	—(2)
FF0h	INTCON3	FD0h	RCON	FB0h	SPBRGH	F90h	—(2)
FEFh	INDF0(1)	FCFh	TMR1H	FAFh	SPBRG	F8Fh	—(2)
FEEh	POSTINC0(1)	FCEh	TMR1L	FAEh	RCREG	F8Eh	—(2)
FEDh	POSTDEC0(1)	FCDh	T1CON	FADh	TXREG	F8Dh	LATE(3)
FECh	PREINC0(1)	FCCh	TMR2	FACh	TXSTA	F8Ch	LATD(3)
FEBh	PLUSW0(1)	FCBh	PR2	FABh	RCSTA	F8Bh	LATC
FEAh	FSR0H	FCAh	T2CON	FAAh	—(2)	F8Ah	LATB
FE9h	FSR0L	FC9h	SSPBUF	FA9h	EEADR	F89h	LATA
FE8h	WREG	FC8h	SSPADD	FA8h	EEDATA	F88h	—(2)
FE7h	INDF1(1)	FC7h	SSPSTAT	FA7h	EECON2(1)	F87h	—(2)
FE6h	POSTINC1(1)	FC6h	SSPCON1	FA6h	EECON1	F86h	—(2)
FE5h	POSTDEC1(1)	FC5h	SSPCON2	FA5h	—(2)	F85h	—(2)
FE4h	PREINC1(1)	FC4h	ADRESH	FA4h	—(2)	F84h	PORTE(3)
FE3h	PLUSW1(1)	FC3h	ADRESL	FA3h	—(2)	F83h	PORTD(3)
FE2h	FSR1H	FC2h	ADCON0	FA2h	IPR2	F82h	PORTC
FE1h	FSR1L	FC1h	ADCON1	FA1h	PIR2	F81h	PORTB
FE0h	BSR	FC0h	ADCON2	FA0h	PIE2	F80h	PORTA

Resistores de pull-up

R/W-1	R/W-1	R/W-1	R/W-1	U-0	R/W-1	U-0	R/W-1
$\overline{\text{RBPU}}$	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7

$\overline{\text{RBPU}}$: PORTB Pull-up Enable bit

1 = All PORTB pull-ups are disabled

0 = PORTB pull-ups are enabled by individual port latch values

Próxima Aula

Aula 09

Introdução a Linguagem C para PIC – Parte II