UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

IF66D - Sistemas Microcontrolados

LISTA DE EXERCÍCIOS 1

Acadêmico: Renan Vicentin Fabrão - 1162853

Prof. André Sanches Fonseca Sobrinho

1/2013

Valor: 1,0 ponto **Formato de entrega: impresso** **Data de entrega: 12/07/2013**

1 – Cite as principais diferenças entre microprocessadores e microcontroladores? (0,02 pontos)

Os microcontroladores são menos poderosos, mais lentos e possuem um espaço de endereçamento menor que os microprocessadores.

Microcontroladores permitem a implementação de sistemas mais compactos.

O conjunto de istruções de um microcontrolador limita-se as instruções mais simples de um microprocessador.

2 – Cite três vantagens da linguagem C sobre a linguagem Assembly. (0,02 pontos)

Não há necessidade de um conhecimento profundo do hardware a ser programado.
Mais fácil compreensão e alteração do código.
Código mais curto e de fácil implementação e possui funções prontas.

3 – Em quais situações é indicada a utilização da linguagem Assembly ao invés da linguagem C? (0,02 pontos)

As situações mais indicadas para a utilização do assembly, são situações de controle crítico, onde a situação exige uma resposta com o mínimo possível de atraso do hardware, por exemplo; acionamento de airbag.

4 - Qual a diferença entre memórias volátil e não volátil? Classifique os três tipos de memória disponíveis no PIC18F4550 nessas duas categorias. (0,02 pontos)

Memórias voláteis são memórias que ao cessar a alimentação elétrica, todos os dados são apagados, resumidamente, guardam dados apenas enquanto há corrente elétrica na memória.

Já Memória não-voláteis tem seus dados preservados ao cessar a passagem da corrente na mesma.

O PIC18F4550 é composto por três tipos diferente de memória;

Memória SRAM: 2MB (volátil);
EEPROM: 256 Bytes (não volátil);
Memória FLASH: 32 KB (não volátil).

5 - Qual o valor máximo que cada pino de I/O do PIC18F4550 pode fornecer de corrente quando configurado como saída e nível lógico '1'? Qual o valor máximo de corrente que pode ser drenado quando configurado como saída e nível lógico '0'? (0,02 pontos)

Cada pino de I/O pode fornecer uma corrente de 25mA.
A máxima corrente drenada por cada pino é de 25mA.

6 - Quando os pinos de I/O do PIC18F4550 são configurados como entrada, qual o consumo de corrente? (0,02 pontos)

Como o pino de entrada não drena corrente, o consumo é de 0mA.

7 – Qual a função da pilha (stack) no microcontrolador? (0,02 pontos)

Armazenar os endereços de retorno quando são utilizadas instruções de desvio de rotinas de chamada.

8 - Cite duas vantagens e duas desvantagens ao se utilizar um cristal como oscilador externo.(0,02)

Vantagens: *oscilador externo é mais preciso e não terá o risco de atrasar o andamento do código.*

Desvantagens: *Custo do componentes necessários para montar um oscilador externo e a perda de um pino de I/O, que será destinado ao oscilador externo.*

9 – Qual a principal vantagem em se utilizar o oscilador interno ao invés do externo? (0,01)

A grande vantagem em se utilizar oscilador interno é o custo.

10 – Descreva o funcionamento do periférico PWRT (Power-up Timer). Caso essa funcionalidade possa ser habilitada/desabilitada, configure-a como habilitada utilizando o registrador associado a esta funcionalidade. (0,02)

PWRT é um temporizador que pode ser ativado para garantir que o microcontrolador comece a operar somente depois que a fonte de alimentação estiver estabilizada. É ativado por meio de um bit de configuração. Ele funciona como uma proteção adicional ao POR.

CONFIG2Lbits.PWRTEN = 0;

11 – Em uma determinada aplicação, o microcontrolador necessita de uma frequência de operação (Fosc) de 4MHz sem que haja necessidade de grande precisão e sem a ocorrência de eventos externos que pudessem afetar a estabilidade do oscilador. Está disponível para ser utilizado nesta aplicação um cristal de 8MHz.

a) Justifique o oscilador a ser escolhido para esta aplicação (cristal, externo ou interno). (0,01)

Oscilador Interno, pois o mesmo opera na frequência de 8MHz e possui divisor de frequência, o que possibilita também trabalhar na frequência de 4MHz.

b) Foi necessário utilizar o periférico PLL? Explique. (0,01)

Não foi necessário, devido ao fato do PLL ser um fator multiplicador de frequência, e foi utilizado uma frequência menor do que a disponibilizada pelo microcontrolador.

c) Configure os registradores necessários para disponibilizar esta frequência. (0,01)

```
#pragma config FOSC = INTOSC_HS
        OSCCONbits.IRCF0 = 0;
        OSCCONbits.IRCF1 = 1;
        OSCCONbits.IRCF2 = 1;
```

12 - Em uma determinada aplicação, o microcontrolador necessita de uma frequência de operação (Fosc) de 32MHz sem que haja necessidade de grande precisão e sem a ocorrência de eventos externos que pudessem afetar a estabilidade do oscilador. Está disponível para ser utilizado nesta aplicação um cristal de 12MHz.

a) Justifique o oscilador a ser escolhido para esta aplicação (cristal, externo ou interno). (0,01)

Deve utilizar o Cristal.

b) Foi necessário utilizar o periférico PLL? Explique. (0,01)

Sim, pois a frequência desejada é maior que a disponibilizada pelo cristal.

c) Configure os registradores necessários para disponibilizar esta frequência. (0,01)

```
#pragma config FOSC = HSPLL_HS
#pragma config PLLDIV= 3
#pragma config CPUDIV = OSC2_PLL3
```

13 - Em uma determinada aplicação, o microcontrolador necessita de uma frequência de operação (Fosc) de 4MHz precisa e estável frente a eventos externos. Está disponível para ser utilizado nesta aplicação um cristal de 12MHz.

a) Justifique o oscilador a ser escolhido para esta aplicação (cristal, externo ou interno). (0,01)

Cristal, pois necessita estabilidade e frequência precisa.

b) Foi necessário utilizar o periférico PLL? Explique. (0,01)

Não, pois o valor da frequência que será utilizado é menor que a do cristal.

c) Configure os registradores necessários para disponibilizar esta frequência. (0,01)

```
#pragma config FOSC = HS
#pragma config CPUDIV = OSC3_PLL4
```

14 – O que acontece com o microcontrolador quando o mesmo é colocado nos modos “sleep” e “idle”? (0,02)

Quando posto em Sleep, os osciladores do microcontrolador são completamente desligados, os periféricos não recebem sinal da fonte de clock, nem mesmo a CPU. Em Sleep o consumo de energia é mínimo, mas deve se considerar os atrasos gerados ao voltar para a execução normal tendo em consideração que os osciladores estavam completamente desligados.

Já no modo Idle os osciladores não desligam completamente, ainda permanecem em operação, apenas o sinal de clock da CPU é cortado; os periféricos continuam recebendo o sinal da fonte de clock selecionada normalmente. Porém a redução de consumo é menor que a proporcionada pelo modo Sleep. Mas por outro lado o processador volta a operar imediatamente ao retornar ao modo normal, já que os osciladores continuam em operação normalmente. Como os periféricos continuam recebendo sinal de clock, pode-se retornal ao modo normal por interrupções.

15 – Qual a vantagem de se utilizar o “watchdog” em um firmware? Explique seu funcionamento. (0,02)

O watchdog funciona como um sistema de prevenção, para casos quando por qualquer motivo o microcotrolador trava.

Quando habilitado ele deve ser zerado a intervalos regulares menores que seu tempo máximo pela instrução CLRWDT. Se o programa trava e o watchdog não é zerado ocorre um reset.

16 – Em uma determinada aplicação onde ocorre periodicamente instabilidade na fonte de tensão que alimenta um microcontrolador, é necessário que o mesmo somente comece a operar depois 65,5ms depois de pulso POR e resete quando a tensão VDD ficar abaixo de 4,33V. Configure o(s) registrador(es) necessário(s) para implementar essas funcionalidades. (0,02)

#progma config PWRT = ON
#progma config BOR = ON
#progma config BORV = 0

20 - Quando o programa esta sendo executado e ocorre uma interrupção de baixa prioridade, o que ocorre? (0,02)

Quando a interrupção é gerada, o processador salva o seu estado atual e começa a executar o tratamento de interrupção apontado pelo vetor. Sempre que uma interrupção ocorre, o programa guarda o endereço da próxima linha a ser executada na pilha e desvia a execução do programa para o endereço fixo da memória de programação.

21 - Se uma interrupção de alta prioridade ocorre durante a execução de uma interrupção de baixa prioridade, o que ocorre? (0,02)

Ocorre um processo similar ao exercício anterior, a execução da baixa prioridade é interrompida e é guardada sua posição atual na pilha, imediatamente é executada a faixa de código de alta prioridade, após o término da execução da interrupção de alta prioridade é restaurado a posição da pilha e a interrupção de baixa prioridade é continuado.

24 – Qual a principal vantagem do TIMER2 quando comparado aos outros TIMERS? (0,02)

17 – **Experimento 1**: Implemente um firmware utilizando linguagem Assembler que faz com que os leds D14 e D15 acendam somente quando o botão SW10 for pressionado e os leds D16 e D17 acendam somente quando o botão SW11 estiver liberado. (0,1)

Observações sobre o Kit Exsto:
- Possui um cristal de 20 MHz;
- Os leds D14, D15, D16, D17 e D18 são controlados pelos pinos de I/O RD4,RD3, RD2, RD1 e RD0 respectivamente. Nível lógico 1 apaga os leds e nível lógico 0 acende os mesmos;
- Os botões SW10 e SW11 controlam os pinos de I/O RC0 e RC1 respectivamente. Se pressionados impõem nível lógico 0.

```
#INCLUDE <P18F4550.INC>                ;ARQUIVO PADRÃO MICROCHIP PARA 16F628A

CONFIG WDT=OFF; disable watchdog timer
CONFIG MCLRE = ON; MCLEAR Pin on
CONFIG DEBUG = ON; Enable Debug Mode
CONFIG LVP = OFF; Low-Voltage programming disabled (necessary for debugging)
ORG     0x00            ;ENDEREÇO INICIAL DE PROCESSAMENTO
GOTO    INICIO

INICIO
        MOVLW B'10000000'
        MOVLW B'00000000'
        MOVWF INTCON            ;TODAS AS INTERRUPÇÕES DESLIGADAS
        MOVLW B'00000111'
        MOVWF CMCON             ;DEFINE O MODO DE OPERAÇÃO DO COMPARADOR ANALÓGICO
        MOVLW B'00000000'
        MOVWF   TRISD
        MOVLW B'00000011'
        MOVWF   TRISC

MAIN
        BTFSC PORTC,0    ;TESTAR O BOTÃO SW10
        GOTO  APAGA1     ;SE O BOTÃO ESTIVER LIBERADO, VAI PARA APAGA1
        GOTO  ACENDE1
APAGA1
        BSF PORTD,4
        BSF PORTD,3
        GOTO MAIN2
ACENDE1
        BCF PORTD,4
        BCF PORTD,3
        GOTO MAIN2

MAIN2
        BTFSC PORTC,1    ;TESTAR O BOTÃO SW10
        GOTO  APAGA2     ;SE O BOTÃO ESTIVER LIBERADO, VAI PARA APAGA1
        GOTO  ACENDE2

APAGA2
        BSF PORTD,2
        BSF PORTD,1
        GOTO MAIN

ACENDE2
        BCF PORTD,2
        BCF PORTD,1
        GOTO MAIN

END
```

18 - **Experimento 2**: Implemente um firmware utilizando linguagem C, que faz com que os leds D14 e D15 acendam somente quando o botão SW10 for pressionado e os leds D16 e D17 acendam somente quando o botão SW11 estiver liberado. (0,1)
Piscar o led D18 independentemente do status dos botões. (0,1)

Observações sobre o Kit Exsto:
- Possui um cristal de 20 MHz;
- Os leds D14, D15, D16, D17 e D18 são controlados pelos pinos de I/O RD4,RD3, RD2, RD1 e RD0 respectivamente. Nível lógico 1 apaga os leds e nível lógico 0 acende os mesmos;
- Os botões SW10 e SW11 controlam os pinos de I/O RC0 e RC1 respectivamente. Se pressionados impõem nível lógico 0.

```
#include<P18F4550.h>

        // Configurações
#pragma config  FOSC=HS
#pragma config  CPUDIV=OSC1_PLL2
#pragma config  IESO = OFF
#pragma config  PWRT = ON
#pragma config   WDT = OFF
#pragma config  BOR = OFF
#pragma config  LVP = OFF



// Função Principal
  void main(void){
        int i=0;
        int j =0;

// Configurações para as portas
        TRISC = 0B00000011;
        TRISD = 0B00000000;
        PORTD = 0B11111111;


// Loop Principal
        for(;;){

                 if(!PORTCbits.RC0){
                        PORTDbits.RD4 = 0;
                        PORTDbits.RD3 = 0;
                } else {
                        PORTDbits.RD4 = 1;
                        PORTDbits.RD3 = 1;
                }
                if(!PORTCbits.RC1){
                        PORTDbits.RD2 = 1;
                        PORTDbits.RD1 = 1;
                } else {
                        PORTDbits.RD2 = 0;
                        PORTDbits.RD1 = 0;
                }

                i++;

                if (i==10000) {
                        PORTDbits.RD0 = 0;
                        i=0;
                        for(j=0;j<10000;j++){}
                } else {
                        PORTDbits.RD0 = 1;
                }
        }
}
```

19 – **Experimento 3:** Realize a mesma implementação do exercício 18 com as modificações (0,1):

a) Verifique se os botões SW10 e SW11 foram pressionados utilizando interrupção externa.
Obs: o botão SW10 deve ter prioridade sobre o botão SW11.
b) O led D18 deverá parar de piscar se qualquer um dos botões estiver pressionado.

```
#pragma config FOSC = HS
#pragma config CPUDIV = OSC2_PLL3
#pragma config PWRT = ON
#pragma config WDT = OFF
#pragma config BOR = OFF

void main(){
    int i=0;

    //configurar as portas
    TRISD = 0x00;

    //configura interrupção
    INTCONbits.INT0IE = 1;
    INTCONbits.INT0IF = 0;
    INTCON2bits.INTEDG0 = 0;

    INTCON3bits.INT1IE = 1;
    INTCON3bits.INT1IF = 0;
    INTCON3bits.INT1IP = 0;
    INTCON2bits.INTEDG1 = 0;

    for(;;){
       PORTDbits.RD4 = 1;
       PORTDbits.RD3 = 1;
       PORTDbits.RD2 = 0;
       PORTDbits.RD1 = 0;

        i++;
        if(i==10000){
           if(PORTDbits.RD0)
              PORTDbits.RD0=0;
           else(!PORTDbits.RD0)
              PORTDbits.RD0=1;

           i=0;
        }
    }
}

void HighPriority (void){
    INTCONbits.INT0IF = 0;
    PORTDbits.RD4 = 0; //acende com nível 0 e apaga com 1
    PORTDbits.RD3 = 0;
    while(!PORTBbits.RB0){}

}

void LowPriority (void){
    INTCON3bits.INT1IF = 0;
    PORTDbits.RD2 = 1; //acende com nível 0 e apaga com 1
    PORTDbits.RD1 = 1;
    while(!PORTBbits.RB0){}
}
```

22 – **Experimento 4:** implemente um firmware que execute as seguintes tarefas (0,1):

a) Utilizando interrupções e o Timer0 faça com que o led D18 do Kit Exsto alterne um segundo aceso e um segundo apagado;

b) Faça com que o led D17 acenda quando o botão SW10 estiver pressionado e apague quando o mesmo estiver liberado;

c) Enquanto o botão SW11 estiver pressionado, o led D18 não deverá piscar.

```
#include<P18F4550.H>
#include"ISR.h"

// Configurações
#pragma config  FOSC=HS
#pragma config  CPUDIV=OSC1_PLL2
#pragma config  IESO = OFF
#pragma config  PWRT = ON
#pragma config   WDT = OFF
#pragma config  BOR = OFF
#pragma config  LVP = OFF
#pragma config  PBADEN = OFF

void main (void){
        //Declaração de variáveis
        unsigned int i;

        //Configuração dos portais
        TRISD = 0x00;               // PORTD é saída

        //Configuração das interrupções
        INTCONbits.INT0IF=0;
        INTCONbits.INT0IE=1;    //INT0

        //Habilita timer 0
        INTCONbits.TMR0IE = 1;
        INTCONbits.TMR0IF = 0;
        INTCON2bits.TMR0IP = 1;

        //Configuração timer 0
        T0CON = 0B10000111;
        TMR0H = 179;
        TMR0L = 181;

        //Configuração botão SW10 - RC0
        TRISCbits.TRISC0 = 1;
        INTCON3bits.INT1P=1;

        //Configuração botão SW11 - RC1
        TRISCbits.TRISC0 = 1;
        INTCON3bits.INT1P=1;

        INTCONbits.INT0IF=0;
        INTCON2bits.INTEDG0=1;  //(SUBIDA)
        INTCONbits.INT0IE=1;

        RCONbits.IPEN=1;
        INTCONbits.GIEL=1;
        INTCONbits.GIEH=1;
```

```
   //Rotina Principal
        for(;;){
                if(PORTCbits.RC0)PORTDbits.RD1=1;
                else PORTDbits.RD1=0;

                if(!PORTCbits.RC1) T0CONbits.TMR0ON = 0;

                T0CONbits.TMR0ON = 1;
        }
}


void HighPriorityISR(void){
        if(INTCONbits.TMR0IF){
                TMR0H = 179;
                TMR0L = 181;
                INTCONbits.TMR0IF=0;

                if(PORTDbits.RD0)
                        PORTDbits.RD0=0;
                else
                        PORTDbits.RD0=1;
        }
}

void LowPriorityISR (void){

}
```

23 – **Experimento 5:** implemente um firmware que execute as seguintes tarefas (0,1):

a) Utilizando interrupções e o Timer0, faça com que o led D18 do Kit Exsto alterne um segundo aceso e três segundos apagados.

b) Utilizando interrupções e o Timer2, faça com que o led D17 alterne um segundo aceso e um segundo apagado.

c) Enquanto o botão SW11 estiver pressionado, nenhum led deverá piscar.

```
#include<P18F4550.H>
#include"ISR.h"

// Configurações
#pragma config  FOSC=HS
#pragma config  CPUDIV=OSC1_PLL2
#pragma config  IESO = OFF
#pragma config  PWRT = ON
#pragma config   WDT = OFF
#pragma config  BOR = OFF
#pragma config  LVP = OFF
#pragma config  PBADEN = OFF


void main (void){

        //Declaração de variáveis
        int i = 0;
        int j = 0;

        //Configuração dos portais
        TRISD = 0x00;               // PORTD é saída
        TRISC = 0B00000011;

        TRISBbits.TRISB0=1;     //SW10 Alta prioridade
        TRISBbits.TRISB1=1;     //SW11 Baixa prioridade

        //Configuração das interrupções
        INTCONbits.INT0IF=0;
        INTCON2bits.INTEDG0=0;  //(DESCIDA)
        INTCONbits.INT0IE=1;    //INT0

        INTCON3bits.INT1IF=0;
        INTCON2bits.INTEDG1=0;  //(SUBIDA)
        INTCON3bits.INT1IE=1;    //INT0
        INTCON3bits.INT1IP=0;

        RCONbits.IPEN=1;
        INTCONbits.GIEL=1;
        INTCONbits.GIEH=1;

        PORTD = 0B11111111;
```

```
	//Rotina Principal
	for(;;){
		if(!PORTCbits.RC0){
		} else {
			PORTDbits.RD4 = 1;
			PORTDbits.RD3 = 1;
		}

		if(!PORTCbits.RC1){
		} else {
			PORTDbits.RD2 = 0;
			PORTDbits.RD1 = 0;
		}

		 i++;

		 if (i==10000) {
			PORTDbits.RD0 = 0;
			for(j=0;j<10000;j++){}
		} else {
			PORTDbits.RD0 = 1;
		}
	}
}

void HighPriorityISR(void){
	INTCONbits.INT0IF=0;
	PORTDbits.RD4=0;
	PORTDbits.RD3=0;
}

void LowPriorityISR (void){
	INTCON3bits.INT1IF=0;
	PORTDbits.RD2=1;
	PORTDbits.RD1=1;
}
```

25 – **Experimento 6:** utilizando o módulo CCP2, faça um firmware onde se o botão SW10 estiver pressionado, a lâmpada do kit deverá se acender com uma frequência de 2KHz e duty-cycle de 75%. Caso contrário, o duty-cycle deverá ser modificado para 25%. (0,1)

Observações sobre o Kit Exsto:

- O botão SW10 controla o pino de I/O RC0. Se pressionado impõe nível lógico 0;
- Para realizar a interface física entre a lâmpada e o pino RC1 do microcontrolador, selecione esta opção através da tecla 4 do dip switch CH3.

```
#include<P18F4550.H>

// Configurações
#pragma config  FOSC=HS
#pragma config  CPUDIV=OSC1_PLL2
#pragma config  IESO = OFF
#pragma config  PWRT = ON
#pragma config   WDT = OFF
#pragma config  BOR = OFF
#pragma config  LVP = OFF
#pragma config  PBADEN = OFF
#pragma config   CCP2MX = ON

//Declaração de variáveis
unsigned int i=1;

void main (void){

        //Configuração dos portais
        TRISD = 0x00;               // PORTD é saída

        //Habilita timer 2
        T2CON = 0B00000111;
        PR2 = 156;

        //Configuração botão SW10 - RC0
        TRISCbits.TRISC0 = 1;

        TRISCbits.TRISC1 = 0;

        //Configuração CCP2 - 75%
        CCP2CON = 0B00011111;
        CCPR2L = 0B01110101;


   //Rotina Principal
        for(;;){
                if((!PORTCbits.RC0)&&(i==0)){
                        CCP2CON = 0B00011111;
                        CCPR2L = 0B01110101;
                        i=1;
                }
                else if((PORTCbits.RC0)&&(i==1)){
                        CCP2CON = 0B00001111;
                        CCPR2L = 0B00100111;
                        i=0;
                }
        }
}
```