x UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

IF66D - Sistemas Microcontrolados

LISTA DE EXERCÍCIOS 2

RENAN VICENTIN FABRÃO

Prof. André Sanches Fonseca Sobrinho

1/2013

Valor: 1,0 ponto **Formato de entrega: impresso** **Data de entrega: 13/09/13**

1 – **Experimento 7:** utilizando um dos comparadores analógicos com topologia de referência interna e o Timer0, faça com que se a tensão do potenciômetro POT do kit Exsto apresentar uma tensão superior a 2,5V, a lâmpada do kit deverá alternar um segundo acesa e um segundo apagada. Caso contrário, ela deve permanecer apagada. (0,1)

Observações sobre o Kit Exsto:

- Para realizar a interface física entre o potenciômetro (POT) e o pino RA0 do microcontrolador, selecione esta opção através da tecla 5 do dip switch CH1;
- 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;

```
//Configurações comparador analógico
CMCON=0b00010110;
CVRCON=0b10001000;

//Configurações TIMER0
TMR0H=179;
TMR0L=180;
T0CON=0b10000111;
INTCONbits.TMR0IF=0;

//Rotina principal
for(;;){
  //Se >2,5V
  if(CMCONbits.C1OUT){
                T0CON=0b10000111;
  }
  //Se <2,5V
  else {
        PORTCbits.RC1=0;
                T0CON=0b00000111;
  }
  //Se o TIMER contou 1s, executa abaixo
 if(INTCONbits.TMR0IF){
 INTCONbits.TMR0IF=0;
 TMR0H=179;
 TMR0L=180;
        if (PORTCbits.RC1){
                PORTCbits.RC1=0;
        }
        else{
                PORTCbits.RC1=1;
        }
  }
 }
}
```

2 - Projetar um Conversor D/A com rede R/2R, resolução de 3 bits e saída variando entre 0V e -5V. A tensão de referência (VREF) é 5V. (0,05)

C	B	A	Vo(V)
0	0	0	0
0	0	1	-0.625
0	1	0	-1.250
0	1	1	-1.875
1	0	0	-2.500
1	0	1	-3.125
1	1	0	-3.750
1	1	1	-4.375

3 – **Experimento 8:** O kit Exsto possui um sensor de temperatura LM35 que apresenta em sua saída um sinal linear de 10mV/ºC. Faça um programa de controle de temperatura, que mantenha a temperatura lida pelo sensor entre 32 e 40ºC. Para isso faça a leitura do sensor e acione a resistência de aquecimento para aquecer até atingir 40ºC e a ventoinha para resfriar até atingir 32ºC. Apresente no LCD a temperatura e se esta aquecendo ou esfriando. (0,1)

Observações sobre o Kit Exsto:
- Para realizar a interface física entre o sensor de temperatura e o pino RA0 do microcontrolador, selecione esta opção através da tecla 7 do dip switch CH1;
- Para realizar a interface física entre a resistência de aquecimento e o pino RC1 do microcontrolador, selecione esta opção através da tecla 3 do dip switch CH3;.

- Para realizar a interface física entre a ventoinha e o pino RC2 do microcontrolador, selecione esta opção através da tecla 5 do dip switch CH3;
-Tanto a resistência quanto a ventoinha são acionados através da imposição de nível lógico '1';
- O LCD deve ser ativado através da chave 1 do dip switch CH2;
- O ajuste de contraste do LCD e feito pelo trimpot R37.

```
#include<P18F4550.h>
#include<delays.h>
#include"xlcd.h"

// Configurações Gerais
#pragma config   FOSC = HS                      // Fosc = 20MHz          Tcy = 200ns
#pragma config   CPUDIV = OSC1_PLL2          // PLL desligado
#pragma config   WDT = OFF                      // Watchdog desativado
#pragma config   PBADEN = OFF                // Pinos do PORTB começam como digitais
#pragma config   LVP = OFF                      // Desabilita gravação em baixa tensão
#pragma config   DEBUG = ON                   // habilita debug
void main (void){

  int Valor;
  // Configurações ADC
  ADCON0 = 0b00000011;
  ADCON1 = 0b00001110;
  ADCON2 = 0b10100101;

  // Iniciação do LCD
  OpenXLCD( EIGHT_BIT & LINES_5X7 );
  WriteCmdXLCD(0x01);
  Delay10KTCYx(10);

  //Configurações pinos de I/O
  TRISC=0b00000010;
  TRISA=0b00000001;

  // Liga o resistor
  PORTCbits.RC1=1;

  // Desliga o ventilador
  PORTCbits.RC2=0;

  //Rotina principal
  for(;;){
                        while (ADCON0bits.GO_DONE == 1){
                                if (ADRESL >= 82){
                                  PORTCbits.RC2=1;
                                }
                                else if (ADRESL <= 65){
                                  PORTCbits.RC2=0;
                                }
                        }
                        ADCON0bits.GO_DONE =1;
          Valor = ADRESL;
    Valor=Valor*0.488;              // Valor em ºC
          WriteCmdXLCD(0x80);
          putrsXLCD("Temp: ");
    putcXLCD(0x30+(Valor/10));
    putcXLCD(0x30+(Valor%10));
          putrsXLCD(" oC");
    WriteCmdXLCD(0xC0);
    putrsXLCD("COOLER ");
    if(PORTCbits.RC2) putrsXLCD("ON ");
    else putrsXLCD("OFF");
  }
}
```

4 - No exercício 2 o sensor de temperatura LM35 apresenta em sua saída um sinal linear de 10mV/ºC. Qual o valor apresentado nos registradores ADRESH e ADRESL quando o conversor A/D do PIC18F4550 realiza a aquisição de uma amostra de temperatura de 100ºC? Considere que esteja sendo utilizada a tensão de referência externa onde V_{REF+} = 3.5V e V_{REF-} = 0V. (0,03)

Adotando ADFM = 1; *ADRESL = 36;* *ADRESH = 1;*

5 – No exercício se for utilizada a tensão de referência interna, isto é V_{DD}=5V e V_{SS}=0V, qual seria o valor do registrador ADRESH quando o conversor A/D do PIC18F4550 realiza a aquisição de uma amostra de temperatura entre 0ºC e 100ºC? Justifique sua resposta. (0,03)

Adotando ADFM = 1; O registrador ADRESH sempre é 0, pois o valor encontrado pela conversão nunca irá ultrapassar 205, logo não é necessário a utilização do ADRESH e seu valor sempre é 0.

6 - Qual a principal vantagem do comparador analógico em comparação ao conversor A/D? (0,03)

A principal vantagem em comparação ao conversor A/D é de que o comparador analógico executa a comparação de forma rápida de dois valores analógicos sem que tenha a intervenção do programa principal.

7 – Explique detalhadamente as funções executadas pelo código abaixo. (0,1)

```
#include<P18F4550.h>                    // Adiciona biblioteca do Microcontrolador
#include"ISR.h"                         // Adiciona biblioteca para Rotina de Interrupção
#pragma config  FOSC=HS                 // Oscilador HS para CPU e USB
#pragma config  CPUDIV=OSC1_PLL2        // Oscilador principal a 48MHz
#pragma config  IESO = OFF              // Desabilita Comunicação entre oscilador Interno e Externo
#pragma config  PWRT = ON               // Microcontrolador opera apenas depois de Estabilizado
#pragma config     WDT = OFF            // Watchdog desativado
#pragma config  BOR = OFF               // Reset por queda de tensão desativado
#pragma config  LVP = OFF               // Desabilita gravação em baixa tensão
#pragma config  CCP2MX = OFF            // Define pino RB3 para CCP

void main (void){
  char CONT=0;                          // Contador iniciando em 0
  TRISBbits.TRISB4=0;                   // Pino RB4 como saída
  TRISBbits.TRISB3=0;                   // Pino RB3 como saída

  CCP2CON=0b00011110;                   // PWM
  CCPR2L=0b00111001;                    // 37%
  T2CON=0b00000111;                     // Timer2          // Prescaler 16, T2 ON, postscaler 1:1, read ON
  PR2=157;                              // Valor de carga de comparação do timer 2

                                        // COMPARADOR ANALOGICO
  CMCON=0b00010110;                     // 4 entra. multiplexiadas p/ 2 comparad. , Comp. não invertidos.
  CVRCON=0b10000100;
  TMR0H=76;
  TMR0L=75;
  T0CON=0b10000111;
  INTCONbits.TMR0IF=0;                  // Desativa interrupção por overflow do timer 0
  INTCONbits.TMR0IE=1;                  // Flag de interrupcão por overflow ON
  RCONbits.IPEN=1;                      // Ativa bit de prioridade para reset do sistema
  INTCONbits.GIEL=1;                    // Habilita bit de habilitação global, para prioridade alta
  INTCONbits.GIEH=1;                    // Habilita bit de habilitação global ON, para prioridade alta
  for(;;){
    if(CMCONbits.C1OUT){                // Caso tenha bits para leitura, entra no laço
      CCP2CONbits.DC2B0=1;              // DC2B0 do registrador CPP2CON recebeu nível logico alto.
      CCPR2L=0b00111001;                //carrega o registrador CCPR2L com o valor 57
    }
    else{
      CPP2CONbits.DC2B0=0;              // DC2B0 do registrador CPP2CON recebeu nível logico baixo.
      CCPR2L=0b00100111;                // Carrega o registrador CCPR2L com o valor 39
   }
  }
}
void HighPriorityISR(void){             // Vetor de alta prioridade da interrupção
  INTCONbits.TMR0IF=0;                  // Flag do timer0 recebe 0
   TMR0H=76;                            // Associa valor 76 a TMR0H
  TMR0L=75;                             // Associa valor 75 a TMR0L
  CONT++;                               // Incrementa CONT
  if(CONT==5){                          // Caso o contador seja igual a 5
    CONT=0;                             // Contador é zerado
    if(PORTBbits.RB4) PORTBbits.RB4=0;  // Se a porta RB4 estiver com nível logico alto, troca pra nível logico baixo else
    else PORTBbits.RB4=1;               // Se não troca para nível logico alto
}
```

PWM=0,4992 ms; Duty cicle= 0,1832 ms
C1 Vin- conectado a RA0 e C2 Vin- conectado a RA1
Timer0: 16 bits, clock interno, pre scale 256, interrupção ativada, tempo de interrupção = 2,35 seg

O timer 0 causa interrupção a cada 2,35 segundos, zera o flag e carrega os valores de de TMR0 E TMROL nos registradores para gerar a nova interrupção a cada 2,35 segundos. A variável cont é incrementada até o valor 5 e quando atinge esse valor é zerado e o valor de RB4 é invertido. Dentro do laço for infinito é verificado constantemente se a saída do comparador C1 é igual a 1. Caso isto seja verdade significa que C1Vin- é maior que C1Vin+ e o valor do período do Duty cicle é alterado para 0,1832 ms, senão é alterado para 0,1248 ms

8 - Cite uma vantagem e uma desvantagem da comunicação paralela em comparação com a comunicação serial. (0,03).
Vantagem: Em comparação a comunicação serial, a comunicação paralela apresenta maior rapidez e interfaces mais simplificadas.
Desvantagem: A comunicação paralela utiliza de muitos pinos e cabos com muitas vias, tendo seu custo proporcional a quantidade de pinos e vias. Também tendo ocasionando ruídos e perda de sincronismo em maiores distancias, logo deve ser aplicado sempre em pequenas distancias.

9 - Cite uma vantagem e uma desvantagem da comunicação serial síncrona em comparação com a comunicação serial assíncrona. (0,03).
Vantagem: A comunicação síncrona apresenta baixo overhead, comparado com a assíncrona. Tornando-a mais rápida.
Desvantagem: Apesar da síncrona ser mais rápida a assíncrona apresenta uma maior facilidade de implementação e menor custo.

10 - Cite duas vantagens do padrão de comunicação RS485 quando comparado com o padrão RS232. (0,03)
Vantagens: O padrão RS485 permite ter maiores distancias alcançadas e a possiblidade de operação em rede com múltiplos usuários

~~11 – **Experimento 9:** Faça um firmware utilizando comunicação serial assíncrona entre o PIC18F4550 e o PC, onde quando o PC enviar o caractere '0', a lâmpada do kit Exsto deverá apagar e o microcontrolador enviará para o PC a mensagem "OFF". Já quando o PC enviar o caractere '1', a lâmpada do kit Exsto deverá acender e o microcontrolador enviará para o PC a mensagem "ON". (0,1)~~

12 – **Experimento 10:** faça um firmware para que o PIC18F4550 envie através de comunicação serial para o PC, a cada um segundo, a temperatura detectada pelo conversor A/D e também o status do ventilador (ON ou OFF). O programa também tem que ligar ou desligar o ventilador a qualquer momento caso o PC envie respectivamente os caracteres '1' ou '0'. (0,1)

Observações sobre o Kit Exsto:

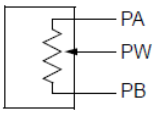
- Para realizar a interface física entre o sensor de temperatura e o pino RA0 do microcontrolador, selecione esta opção através da tecla 7 do dip switch CH1;
- Para realizar a interface física entre a resistência de aquecimento e o pino RC1 do microcontrolador, selecione esta opção através da tecla 3 do dip switch CH3;.
- 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;
- Para realizar a interface física entre a ventoinha e o pino RC2 do microcontrolador, selecione esta opção através da tecla 5 do dip switch CH3;
-Tanto a resistência quanto a ventoinha e a lâmpada são acionados através da imposição de nível lógico '1';
- Para realizar a interface física entre o driver RS232 e os pinos RC6 e RC7 do microcontrolador, selecione estas opções através das teclas 6 e 8 do dip switch CH4.

```
#include<P18F4550.h>
#include<delays.h>
#include"xlcd.h"

// Configurações Gerais
#pragma config   FOSC = HS                  // Fosc = 20MHz          Tcy = 200ns
#pragma config   CPUDIV = OSC1_PLL2         // PLL desligado
#pragma config   WDT = OFF                  // Watchdog desativado
#pragma config   PBADEN = OFF               // Pinos do PORTB começam como digitais
#pragma config   LVP = OFF                  // Desabilita gravação em baixa tensão
#pragma config   DEBUG = ON                 // habilita debug
void main (void){

  // Variáveis
  int Valor;
  int Valor2;
  int var;
  int teste=1;

  // Configurações ADC
  ADCON0 = 0b00000011;
  ADCON1 = 0b00001110;
  ADCON2 = 0b10100101;

  // Configuração da comunicação serial
  TXSTAbits.BRGH=0;
  BAUDCONbits.BRG16=1;
  SPBRG=129;
  TXSTAbits.SYNC=0;
  RCSTAbits.SPEN=1;
  TXSTAbits.TXEN=1;

  //Configurações TIMER0
  TMR0H=179;
  TMR0L=180;
  T0CON=0b10000111;
  INTCONbits.TMR0IF=0;
```

```
// Iniciação do LCD
OpenXLCD( EIGHT_BIT & LINES_5X7 );
WriteCmdXLCD(0x01);
Delay10KTCYx(10);

//Configurações pinos de I/O
TRISC=0b01000010;
TRISA=0b00000001;
// Liga o resistor
PORTCbits.RC1=1;
// Desliga o ventilador
PORTCbits.RC2=0;
//Rotina principal
for(;;){
                        while (ADCON0bits.GO_DONE == 1){
                                if (ADRESL >= 82){
                                  PORTCbits.RC2=1;
                                }
                                else if (ADRESL <= 65){
                                  PORTCbits.RC2=0;
                                }
                        }
                        ADCON0bits.GO_DONE =1;
                //Se o TIMER contou 1s, executa abaixo
                if(INTCONbits.TMR0IF){
                        Valor2 = ADRESL;
                Valor2=Valor2*0.488;
                        var=0;
                        INTCONbits.TMR0IF=0;
                        TMR0H=179;
                        TMR0L=180;
                        while(TXSTAbits.TRMT = 0){}
                        TXREG = (0x30+(Valor2/10));
                        while(TXSTAbits.TRMT = 0){}
                        TXREG = (0x30+(Valor2%10));
                        while(TXSTAbits.TRMT = 0){}
                        TXREG = 10;
                        while(TXSTAbits.TRMT = 0){}
                        TXREG = 13;
                }
        }
}
```

13 - Por que é sempre indicado receber os dados da USART utilizando-se interrupção ao invés da técnica de pooling? (0,02)
Pois na técnica de pooling é aplicado o sistema 'wait for flag' que nada mais é que o programa ficar em loop perguntando ao dispositivo se já recebeu dado ou se já transmitiu o último dado. Técnica que na prática tem um grande custo de processamento. Logo a técnica de interrupção é a maneira mais simples e limpa de fazer o serviço.

14 – **Experimento 11:** o esquemático do potenciômetro digital MC41010 é mostrado na figura abaixo: (0,1)



$$R_{WA}(D_n) = \frac{(R_{AB})(256 - D_n)}{256} + R_W$$

$$R_{WB}(D_n) = \frac{(R_{AB})(D_n)}{256} + R_W$$

Onde

R_{AB} = resistência entre os terminais P_A e P_B . Para este CI, o valor é de 10KΩ.
R_W = resistência do terminal P_W. O valor é 52Ω.
R_{WA} = resistência entre os terminais P_A e P_W
R_{WB} = resistência entre os terminais P_B e P_W
D_n = valor de configuração, o qual varia entre 0 e 255
Ex: Para D_n igual a 192

$$R_{WA}(D_n) = \frac{(R_{AB})(256 - D_n)}{256} + R_W$$
$$R_{WA}(C0h) = \frac{(10k\Omega)(256 - 192)}{256} + 52\Omega$$
$$R_{WA}(C0h) = 2552\Omega$$

$$R_{WB}(D_n) = \frac{(R_{AB})(D_n)}{256} + R_W$$
$$R_{WB}(C0h) = \frac{(10k\Omega)(192)}{256} + 52\Omega$$
$$R_{WB}(C0h) = 7552\Omega$$

Para configurar o potenciômetro com os valores de resistores desejados, deve ser enviado um frame contendo dois bytes através da comunicação SPI, sendo o primeiro o byte o comando 0x11 e o segundo byte o valor do parâmetro D_n.

No kit Exsto o potenciômetro digital MC41010 tem seu terminal P_B ligado em 5V e o terminal P_A ligado em 0V. Desta forma faça um firmware que configure uma tensão entre os terminais P_A e P_W de acordo com o estado da tecla SW10 :

Botão SW10	Valor da tensão
Pressionado	1V
Liberado	2V

Observações sobre o Kit Exsto:

- A entrada CS do potenciômetro digital MC41010 está mapeada no pino RA4 do microcontrolador. Para realizar a interface física, mude o estado da tecla 4 do dip switch CH2;
- O botão SW10 controla o pino de I/O RC0. Se pressionado impõe nível lógico 0.

```
#include<P18F4550.h>
#pragma config   CPUDIV = OSC1_PLL2          // PLL desligado
#pragma config   FOSC = HS                   // Fosc = 20MHz          Tcy = 200ns
#pragma config   WDT = OFF                   // Watchdog desativado
#pragma config   PBADEN = OFF                // Pinos do PORTB começam como digitais
#pragma config   LVP = OFF                   // Desabilita gravação em baixa tensão
#pragma config   DEBUG = ON                  // habilita debug

TRISB=0b00000001;
TRISC=0b00000001;
TRISA=0b00000000;

TRISAbits.RA4=0;
PORTAbits.RA4= 1;

SSPSTATbits.CKE=1;//
SSPSTATbits.SMP=1;
SSPCON1bits.WCOL=0;//
SSPCON1bits.SSPEN=1;
SSPCON1bits.CKP=0; //
SSPCON1bits.SSPM3=0;
SSPCON1bits.SSPM2=0;
SSPCON1bits.SSPM1=0;
SSPCON1bits.SSPM0=1;

void main (void){
  unsigned char valor;
  int i;

for(;;){
          if(PORTBbits.RB0==0){
              PORTAbits.RA4= 0;
      PIR1bits.SSPIF=0;
              SSPBUF=0x11;
              while(PIR1bits.SSPIF){}
      for(i=0;i<4;i++){}
      PIR1bits.SSPIF=0;
              SSPBUF=0xCE;
              while(PIR1bits.SSPIF){}
      for(i=0;i<4;i++){}
              PIR1bits.SSPIF=0;
      PORTAbits.RA4= 1;
  }

  if(PORTBbits.RB0==1){
              PORTAbits.RA4= 0;
```

```
        PIR1bits.SSPIF=0;
                SSPBUF=0x11;
                while(PIR1bits.SSPIF){}
        for(i=0;i<4;i++){}
        PIR1bits.SSPIF=0;
                SSPBUF=0xCE;
                while(PIR1bits.SSPIF){}
        for(i=0;i<4;i++){}
                PIR1bits.SSPIF=0;
        PORTAbits.RA4= 1;
    }
}
```

15 - Qual a função do pino CS em uma comunicação SPI? (0,02)
Em uma rede Master/Slaves cada dispositivo (slave) tem um pino CS ligado a um pino de I/O do microcontrolador master. Se o pino CS de um periférico (slave) estiver habilitado ele se comunica com o microcontrolador (master) enquanto os demais, que não estão habilitados, ignoram a comunicação.

16 - Cite duas vantagens da comunicação USB sobre a comunicação serial assíncrona padrão RS232. (0,03)
Vantagens: A comunicação USB tem conexão 'Plug&Play' e pode conectar-se a diversos periféricos ao mesmo tempo. Vantagens que não são possíveis no padrão RS232.

17 – **Experimento 12**: faça um programa que execute as mesmas funcionalidades do exercício 15, porém utilizando a comunicação USB para configurar o valor de tensão de acordo com os seguintes caracteres enviados pelo PC para o microcontrolador: (0,1)

Caracter	Valor da tensão
'1'	1V
'2'	2V
'3'	3V
'4'	4V

Observações sobre o Kit Exsto:

- Para utilizar o periférico USB, realiza a interface física entre o pino RA2 e ao senso de tensão do USB (USB_SENSE) através das tecla 2 do dip switch CH2.

```
#if defined(__18CXX)
void main(void)
#else
int main(void)
#endif
{
 //Variáveis
 char numBytesRead;
 char i;
 //********************* Configura a comunicação SPI ***********
   TRISBbits.TRISB0 = 1;
   TRISCbits.TRISC7 = 0;
   TRISBbits.TRISB1 = 0;
   TRISAbits.TRISA4 = 0;

   // Botão SW0 está no RC0
   TRISC = 0b00000001;
   PORTC = 0b00000001;

   // CS do potênciometro está no A4
   TRISA = 0b00000000;
   PORTA = 0b00010000;

   // Configuração do SPI
   SSPSTATbits.CKE = 0;
   SSPSTATbits.SMP = 0;
   SSPCON1 = 0b00100010;

   //Inicializa a comunicação USB
   InitializeSystem();
   #if defined(USB_INTERRUPT)
   USBDeviceAttach();
   #endif
```

```
while(1)
  {
      //Executa as tasks USB
      #if defined(USB_POLLING)
      USBDeviceTasks();
      #endif

            // ************** Verifica se chegaram dados da USB
            numBytesRead = getsUSBUSART(USB_Out_Buffer,64);
            if (numBytesRead > 0){

            // ************** Se chegaram dados do PC iguais a 1,2,3 ou 4, executa abaixo
            if (USB_Out_Buffer[0] == '1' || USB_Out_Buffer[0] == '2' || USB_Out_Buffer[0] == '3' || USB_Out_Buffer[0] == '4'){
            //***********Programar o potenciômetro
                        PORTAbits.RA4 = 0;
                        // ------------------------------PIR1bits.SSPIF = 0;
                        SSPBUF = 0x11;
                        // Aguardar a transmissão
                        while(PIR1bits.SSPIF != 1);
                         // Delay 1 us
                         for(i = 0; i < 4; i++);
                        // ------------------------------PIR1bits.SSPIF = 0;
                        SSPBUF = 255 - (51 *(USB_Out_Buffer[0] - 0x30));
                        // Aguardar a transmissão
                        while(PIR1bits.SSPIF != 1);
                        // Delay 1 us

                        for(i = 0; i < 4; i++);
                        PORTAbits.RA4 = 1;
                        //********** Transmitir OK para o PC
                        while(USBUSARTIsTxTrfReady() == 0);
                        USB_In_Buffer[0] = 'O';
                        USB_In_Buffer[1] = 'K';
                        USB_In_Buffer[2] = ' ';
                        USB_In_Buffer[3] = USB_Out_Buffer[0];
                        USB_In_Buffer[4] = 'V';
                        putUSBUSART(USB_In_Buffer, 5);
                        }
            }
   // Controla os leds USB
   ProcessIO();
  }//end while
}//end main
```