WWW.cerne-tec.com.br

Comunicação USB com o PIC

Vitor Amadeu Souza – Parte II

vitor@cerne-tec.com.br

Continuando com o artigo apresentado na edição passada de comunicação USB com o PIC, continuaremos nesta edição o estudo do mikroC e a configuração do mesmo para a correta comunicação do PIC com o PC via USB.

O Ambiente de Programação

Após a instalação do mikroC, execute o mesmo. Teremos a tela da figura 1.

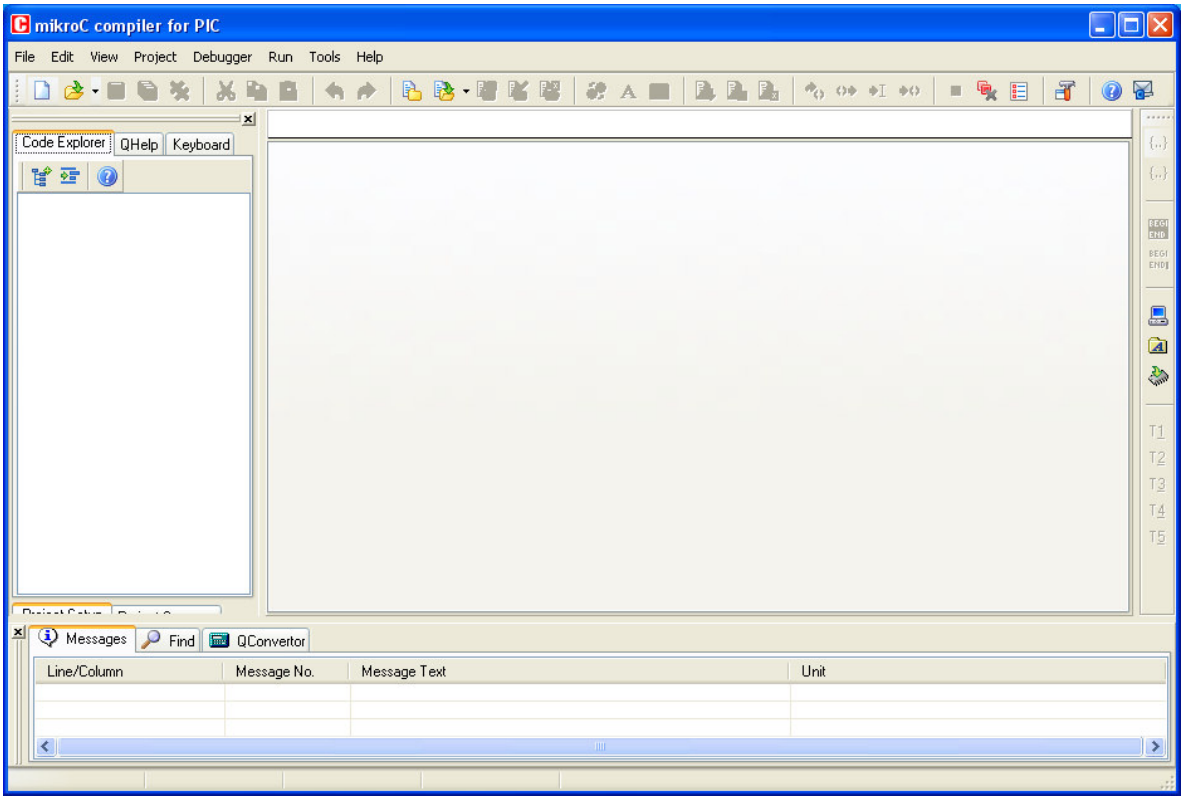

Figura 1 – Tela inicial do mikroC

O mikroC trabalha com o conceito de projeto, ou seja, sempre que quisermos compilar um determinado código, precisaremos criar um projeto. Neste caso, vá no menu Project -> New Project, a tela da figura 2 será apresentada:

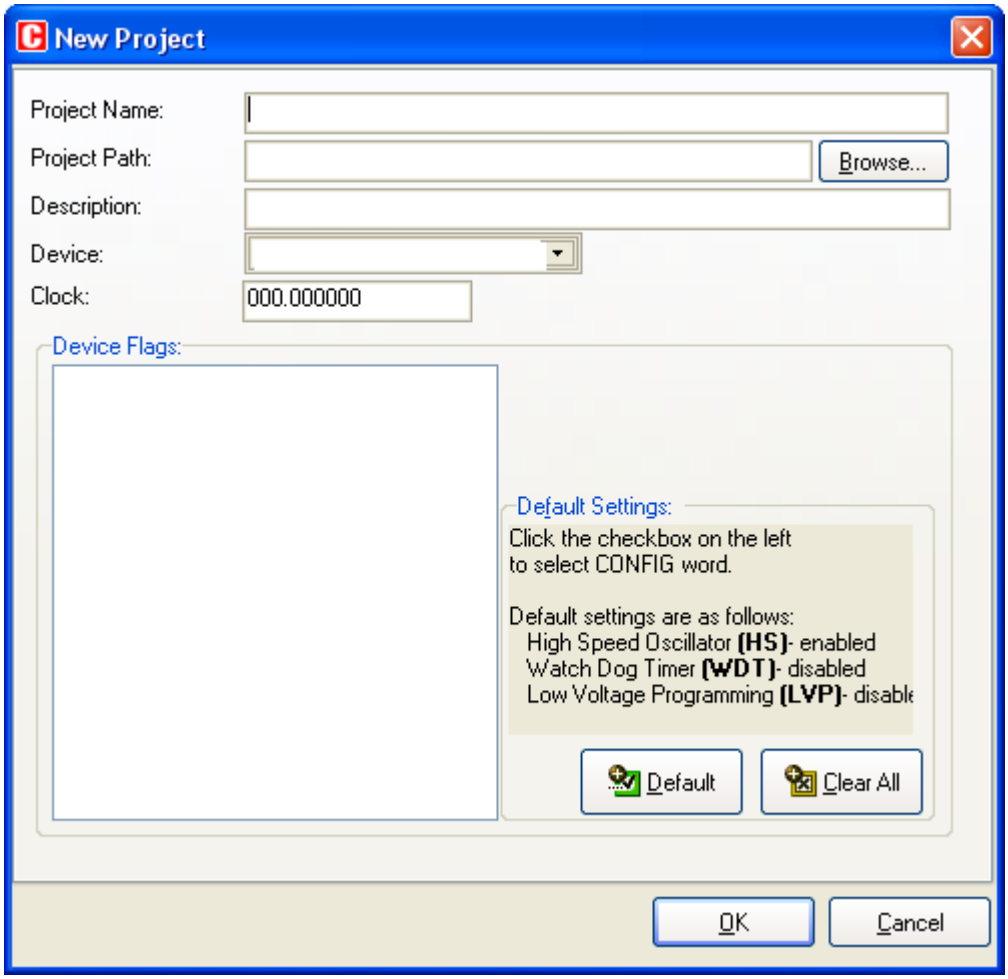

Figura 2 – Criando um projeto no mikroC

No campo Project Name, devemos informar o nome do nosso projeto. Informe neste campo por exemplo, o nome *comunicação_usb* ou o nome que melhor lhe convier. Em seguida, escolha a pasta onde o projeto ficará salvo, clicando no botão Browse do campo Project Path. O campo Descritption é

opcional e serve para detalharmos o que o nosso projeto faz, podendo ficar neste exemplo sem preenchimento. O campo clock serve para configurarmos no compilador qual a velocidade de processamento no qual o PIC está funcionando, No nosso caso, apesar do oscilador ser de 4 MHz, escolha o clock de 48 MHz, pois como a série 18 possui internamente PLLs que permitem aumentar o clock do microcontrolador, neste exemplo iremos utilizar este recurso. No campo Device, escolha o modelo que iremos utilizar neste projeto, neste caso o PIC18F4550. Através do campo Device Flags, podemos ajustar os bits de configuração do PIC. Observe na tabela 1 os campos que devem ficar marcados para o correto funcionamento deste exemplo.

Configuration Bit
PLLDIV_1_1L
CPUDIV_OSC1_PLL2_1L
USBDIV_2_1L
FOSC_XTPLL_XT
VREGEN_ON_2L
WDT_OFF
MCLRE_OFF
LPT1OSC_OFF
PBADEN_OFF
LVP_OFF_4L
ICPRT_OFF_4L
XINST_OFF_4L
DEBUG_OFF_4L

Tabela 1 – Ajuste de Configurations Bits

**** Recomendo a leitura do datasheet do PIC18F4550 na parte dos configurations bits para um entendimento melhor dos mesmos.***

Feito os ajustes apresentados, pressione o botão ok. Agora o mikroC ficará com a tela apresentada na figura 3.

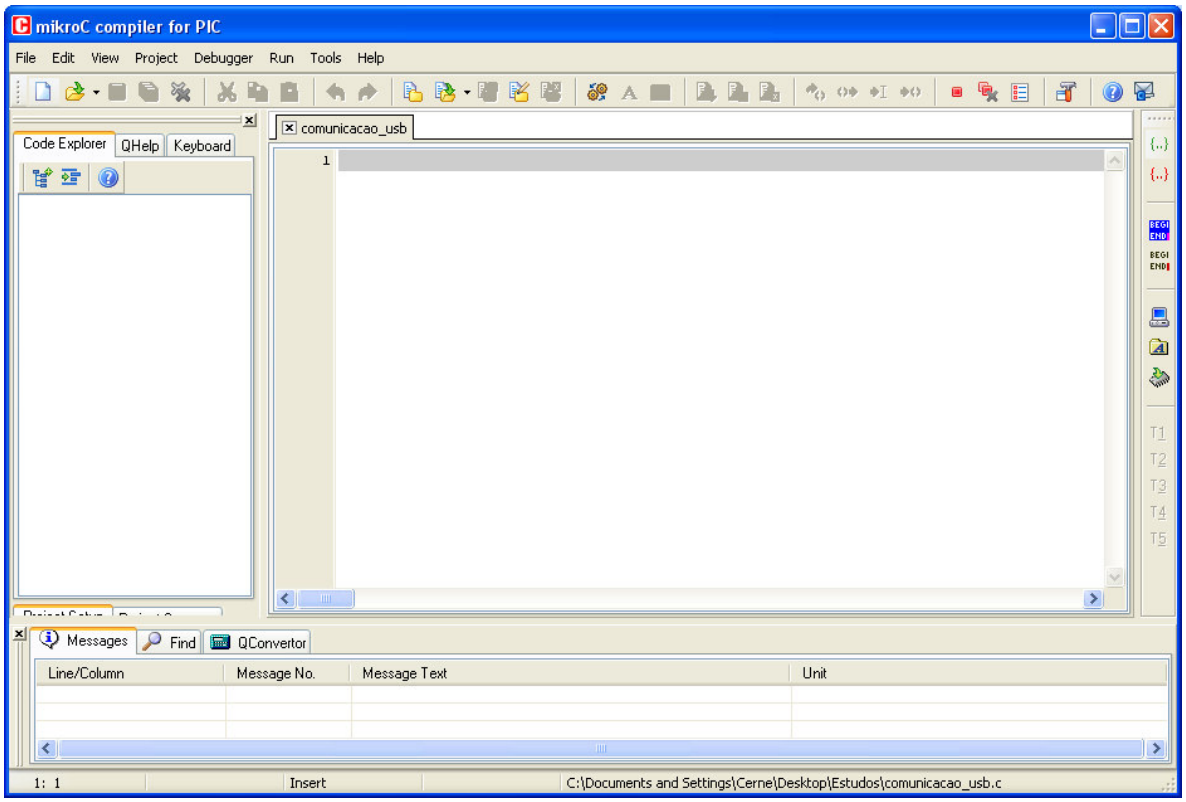

Figura 3 – Tela apresentada pelo mikroC

Agora precisaremos criar um arquivo chamado *descritor.* A função deste arquivo é fazer a identificação do dispositivo USB assim que o mesmo for conectado ao PC. Para acessar este item, vá no menu Tools -> HID Terminal. A tela da figura 4 surgirá.

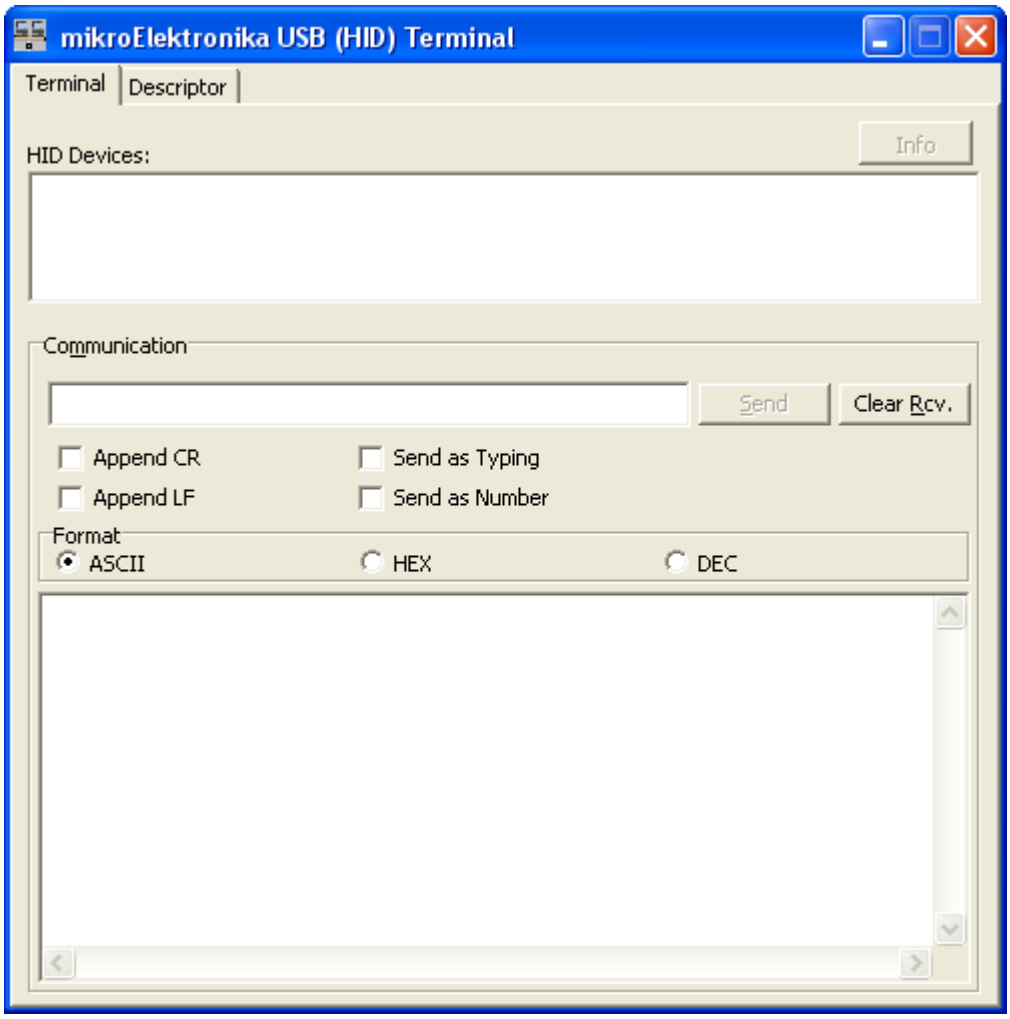

Figura 4 – Configuração do arquivo descritor

Agora clique na aba Descriptor, a tela da figura 5 surgirá.

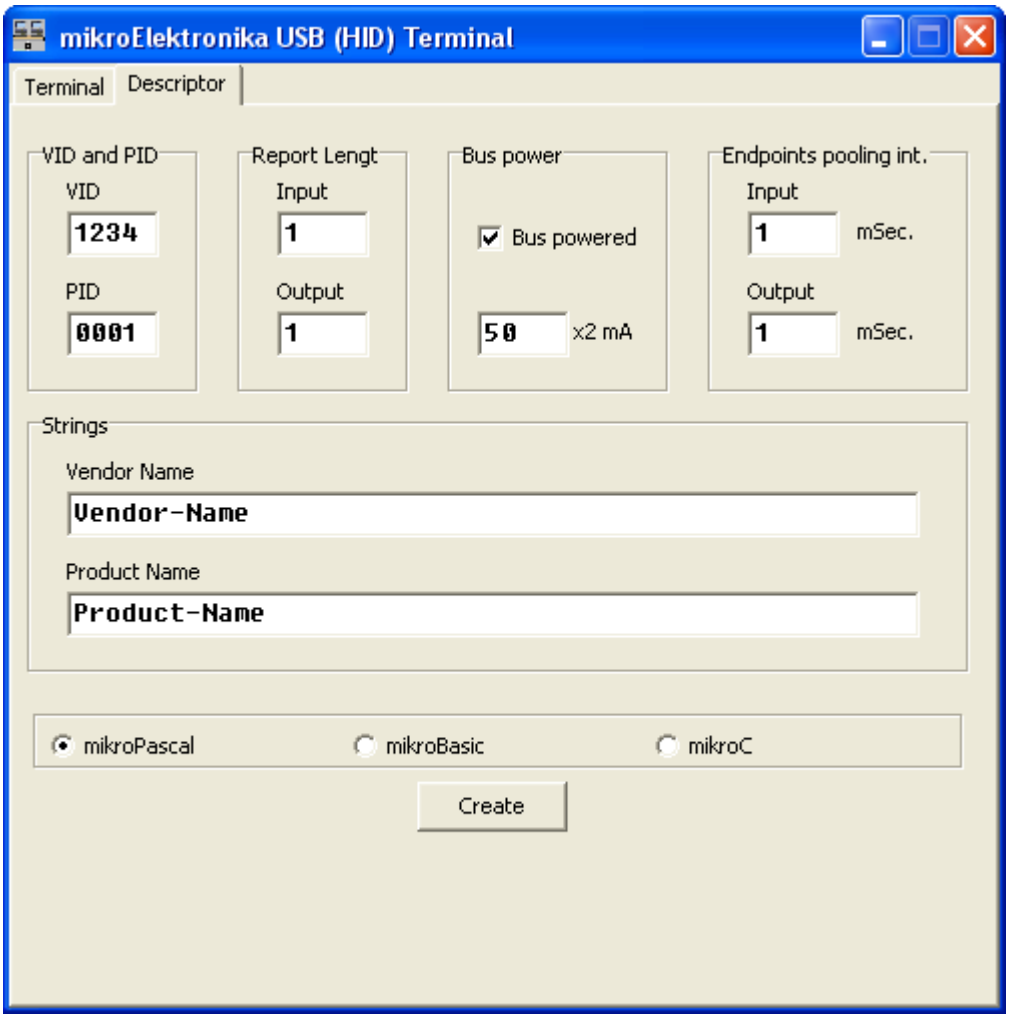

Figura 5 – Abrindo o HID Terminal

Neste arquivo, definimos por exemplo o VID (Vendor ID) e PID (Product ID) do dispositivo. Estes dois parâmetros, são fornecidos diretamente pela USB.org porém iremos utilizar o mesmo que está sendo informado como parâmetro. Outras informações que podemos ajustar é o buffer de entrada e saída, através do campo Report Lenght porém como iremos comunicar a nível de bytes, podemos manter a configuração atual. Além disso, no campo bus power, podemos ajustar se o dispositivo será do tipo bus powered, ou seja alimentado pela própria USB ou self powered, o que quer dizer que está sendo alimentado pela fonte no qual o equipamento está ligado. No nosso caso, a opção bus powered deverá ficar marcado, que por sinal já é o default. Note que no campo abaixo da configuração da alimentação do dispositivo, podemos também configurar a corrente máxima no qual este irá utilizar. Este campo também não precisa ser alterado, podendo manter neste caso com 100 mA que é 50mA x 2. Os campos de Strings de Vendor Name e Product Name servem para que assim que o dispositivo for conectado a porta USB, este nome seja apresentado, podemos também manter a string apresentada. Após este passo, note que existem três ferramentas no qual o HID

Terminal pode configurar para gerar o campo descritor, sendo necessário escolher a opção mikroC. Pronto, agora pressione o botão Create e salve o arquivo com o nome usbdsc.c na pasta onde está o seu projeto. Agora, de volta a janela principal do mikroC, precisamos adicionar este arquivo descritor ao projeto. Para isso, vá no menu Project -> Add to Project. Neste momento, será aberta uma nova janela onde você deverá escolher o arquivo recém criado, neste caso o usbdsc.c na pasta onde você salvou o mesmo.

Para encerrar esta parte de configuração, devemos copiar dois arquivos chamados var.h e definit.h que se encontram na pasta ...\mikroelektronika\mikroC\examples\extra_examples\hid-library para a pasta do seu projeto e adicionar os mesmos, da mesma forma que o usbdsc.c ao projeto. Note que normalmente, o mikroC fica instalado na pasta arquivos de programa do seu PC.

Código

Com o ambiente devidamente configurado, podemos agora nos ater ao código que irá rodar no PIC. Este código está disponível no box1. Vamos agora esmiuçar melhor o seu funcionamento.

```
unsigned char userWR_buffer[64];        //Buffer de transmissão serial
unsigned char userRD_buffer[64];        //Buffer de recepção serial
```

As funções Hid_Write e Hid_Read conseguem comunicar com até 64 bytes. Apesar de utilizarmos somente 1 byte tanto na transmissão como na recepção, precisamos declarar estes dois vetores de forma que todo o byte transmitido ou recebido fique armazenado nestes dois buffers.

```
void main()
{
  char recebe;                          //Variável indicadora de bytes recebidos
  ADCON1 = 0x0F;                        //Configura os pinos de forma digital
  trisa.ra1=0;                          //Configura o pino do led como saída
  trisb.rb0=1;                          //Configura o pino do botão como entrada
```

Agora entramos na função principal de programa do programa em C, neste caso a função void (principal). Note que logo destarte, é declarada uma variável do tipo char chamada *recebe*. Esta variável será utilizada logo a frente no programa, para indicar se um dado foi recebido ou não do PC. Inicialmente, quando o microcontrolador é energizado, os pinos do PORTA e PORTE ficam configurados como entradas analógicas. No nosso caso, em que a porta será utilizada da forma digital, devemos desligar este recurso e isso é feito atribuindo

ao registrador ADCON1 o valor 0x0F. De acordo com o esquema elétrico, no pino RA1 está conectado um led, enquanto no pino RB0 o botão. Configurando os registradores TRISB e TRISA do PIC, estamos ajustando as direções no qual os mesmos irão funcionar.

```
HID_Enable(&userRD_buffer, &userWR_buffer);
                                        //Inicializa a USB
```

Neste ponto, é feita a inicialização da USB no PIC. Note que os dois vetores que foram inicializados no início do programa, são informados para a função HID_Enable. Todo o byte recebido ou transmitido pela USB, serão feitos através destes dois registradores.

```
while (1)
  {
        if (portb.rb0==0)
           {
              userWR_buffer[0]='1';         //Carrega byte a ser enviado para o PC
              while (!HID_Write(&userWR_buffer, 1)) ;
                                           //Envia e aguarda finalizar o envio do
                                           //byte pela USB
           }
        else
           {
              userWR_buffer[0]='0';         //Carrega byte a ser enviado para o PC
              while (!HID_Write(&userWR_buffer, 1)) ;
                                            //Envia e aguarda finalizar o envio
                                            //do byte pela USB
```

Dando prosseguimento a análise do código, entramos no loop while em que constante os estados do botão são verificados e enviado o byte referente se o mesmo está pressionado ou não. Se o botão estiver pressionado (`if (portb.rb0==0))` é primeiramente carregado o byte 'A' no vetor usbWR_Buffer[0] que é o byte que será enviado para o PC e logo em seguida o mesmo é escrito através da função HID_Write, que transmite 1 byte do vetor usbWR_Buffer[0].

```
          recebe=hid_read();                          //Verifica se há algum byte no
                                                      //buffer de recepção

         if (recebe!=0)                               //Há algum byte para receber?
          {
               if(userRD_buffer[0]=='A')              //É o 'A'?
                         porta.ra1=1;                 //Sim, então liga o led
               if(userRD_buffer[0]=='B')              //É o 'B'?
                         porta.ra1=0;                 //Sim, então desliga o led
          }

                 Delay_ms(1000);                      //Delay para atualização
```

Para saber se um determinado byte foi recebido, a variável *recebe* recebe o valor proveniente da função hid_read(). Caso a mesma retorne 0, indica que nenhum byte foi recebido e caso contrário, há a existência de algum dado no buffer de recepção. Note que logo em seguida, o conteúdo do buffer de recepção é checado e caso haja o caracter 'A', o led é aceso e caso contrário, é verificado se é o 'B' sendo neste caso apagado. O delay_ms(1000) foi colocado apenas para que possamos verificar com mais clareza os dados recebidos pelo PIC, não sendo obrigatório o seu uso.

```
void interrupt(void)
{
  HID_InterruptProc();                        //Tratamento da interrupção de USB

}
```

Este bloco de interrupção foi criado para que toda a transação que ocorrer na USB, seja tratada pela mesma. Observe que a função `HID_InterruptProc()` também já é disponibilizada pelo próprio mikroC.

Agora compile e grave este programa no microcontrolador, de forma que possamos continuar o nosso estudo.

```
unsigned char userWR_buffer[64];        //Buffer de transmissão serial
unsigned char userRD_buffer[64];        //Buffer de recepção serial

void main()
{
  char recebe;

  ADCON1 = 0x0F;                        //Configura os pinos de forma digital

  trisa.ra1=0;                          //Configura o pino do led como saída
  trisb.rb0=1;                          //Configura o pino do botão como entrada

  HID_Enable(&userRD_buffer, &userWR_buffer);
                                        //Inicializa a USB


  while (1)
  {
        if (portb.rb0==0)
           {
              userWR_buffer[0]='1';              //Carrega byte a ser enviado para o PC
              while (!HID_Write(&userWR_buffer, 1)) ;
                                                 //Envia e aguarda finalizar o envio do
                                                 //byte pela USB
           }
        else
           {
              userWR_buffer[0]='0';              //Carrega byte a ser enviado para o PC
              while (!HID_Write(&userWR_buffer, 1)) ;
                                                  //Envia e aguarda finalizar o envio
                                                  //do byte pela USB
           }


           recebe=hid_read();                       //Verifica se há algum byte no
                                                    //buffer de recepção

          if (recebe!=0)                            //Há algum byte para receber?
           {
                if(userRD_buffer[0]=='A')           //É o 'A'?
                          porta.ra1=1;              //Sim, então liga o led
                if(userRD_buffer[0]=='B')           //É o 'B'?
                          porta.ra1=0;              //Sim, então desliga o led
           }

             Delay_ms(1000);                        //Delay para atualização


  }
}


void interrupt(void)
{
  HID_InterruptProc();                               //Tratamento da interrupção de USB

`
```

Box 1 – Código Fonte

Conectando ao PC

Assim que o microcontrolador for conectado ao PC, irá aparecer uma mensagem informativa do Windows, conforme a figura 6.

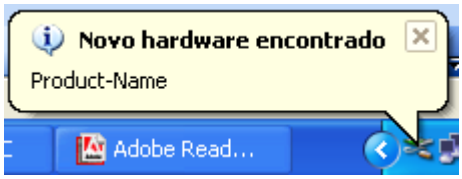

Figura 6 - Detectando o dispositivo USB

Neste instante, irá aparecer o assistente de instalação do Windows, conforme a figura 7.

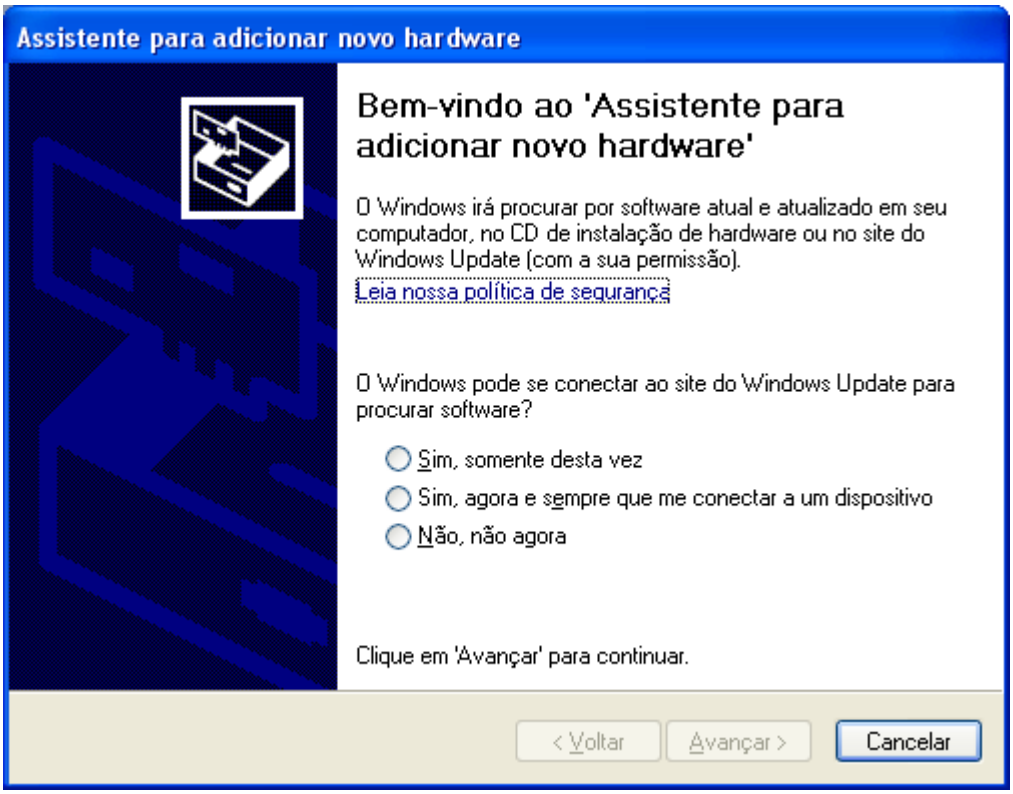

Figura 7 – Assistente de instalação do Windows

Escolha a opção Não, não agora e pressione Avançar. Na próxima tela, escolha a opção *Instalar o Software Automaticamente (recomendável)* e pressione ok. Em seguida, o Windows pode informar que o dispositivo não está registrado no logotipo do mesmo. Ignore esta mensagem e continue pressionando Continuar. O resultado deverá ser o apresentado na figura 8.

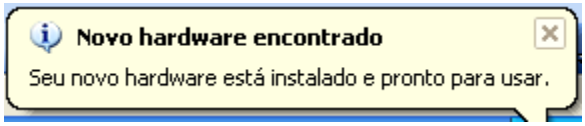

Figura 8 – Dispositivo reconhecido e pronto para funcionar

Agora iremos utilizar a ferramenta HID Terminal que fica no próprio mikroC para comunicar com o PIC. Note que neste ponto, outras aplicações como o Delphi, Visual Basic ou Java já poderiam comunicar com o PIC, desde que estes tivessem os devidos componentes de acesso a USB. Para abrir o HID Terminal, vá em Tools -> HID Terminal. Note que assim que você abrir este software, o mesmo já irá apresentar os dados recebidos da USB, conforme apresentado na figura9.

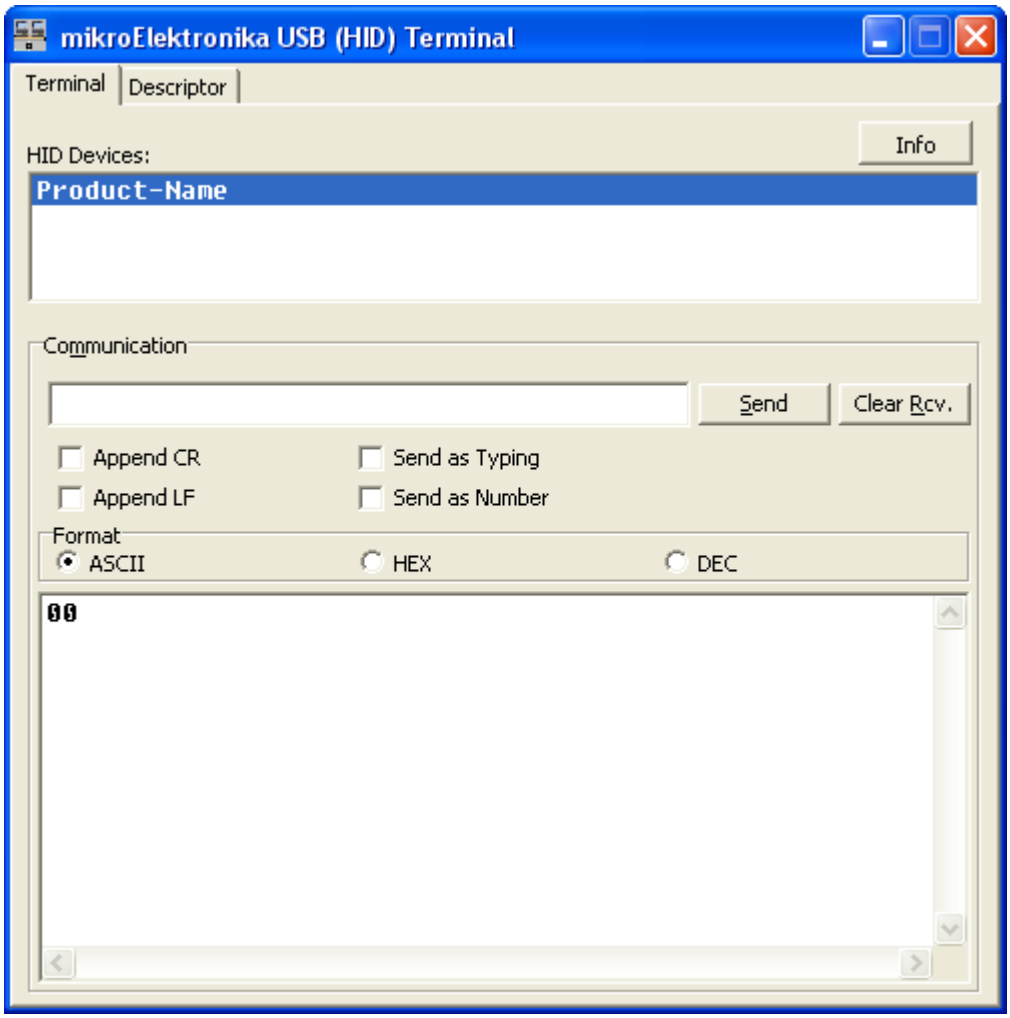

Figura 9– Recebendo dados da USB

Observe que o 0 está sendo apresentado em função do botão estar solto. Conforme o mesmo for pressionado, o ‘1’ será enviado.

Agora para testar o acesso as saídas do PIC via USB, na caixa communication, coloque o caracter ‘A’ e logo em seguida, pressione o botão Send. Observe que neste instante, o led irá acender. A mesma idéia é válida para apagar, bastando neste caso enviar o caracter ‘B’.

Conclusão

Vimos nestes dois artigos como implementar a comunicação USB com o PIC. Este tema se torna importante nos dias atuais, em que cada vez mais a porta RS232 entra em desuso, e atualizar os atuais projetos com a USB se torna tão necessário.