Projeto de Sistemas RF

Aula 07 - Transmissão de Dados Através do Protocolo RS232 - Parte I

Apresentação

Até agora, vimos o que é necessário em termos de programação e um pouco de teoria de sistemas de RF. Agora é hora de pôr em prática o que aprendemos e realizar algumas montagens com microcontroladores e comunicá-los. Nesta aula, faremos a comunicação com fio e, posteriormente, faremos a mesma comunicação sem fio. Utilizaremos 2 protocolos, o protocolo RS232 (nesta e na próxima aula) e o protocolo SPI (últimas aulas). O microcontrolador utilizado será o PIC da série 18F.

Objetivos

Ao final desta aula, você deverá ser capaz de:

- Simular a montagem de um transmissor e um receptor (separados) utilizando o microcontrolador.
- Realizar testes na montagem para certificar-se de seu funcionamento.

Parte I (*Software*)

Vamos dividir esta aula em duas partes. Nesta primeira parte, veremos tudo que se refere a *software* e, na segunda, o que se refere a *hardware*. Quando falamos em termos de *software,* estamos querendo dizer o programinha que irá rodar no microcontrolador (no nosso caso o PIC). O que esse programa irá fazer então? Bem, como esse é o nosso primeiro programa, iremos fazer algo muito simples. Iremos apenas acender um LED quando pressionarmos um botão, porém, o botão irá estar em um microcontrolador e o LED em outro.

Basicamente, teremos que implementar dois programas. Um deles será chamado de transmissor e o outro de receptor. Cada um irá implementar comandos diferentes para realizar as tarefas de transmitir e receber dados. Como nosso pequeno sistema é bem simples (acender um

LED), o transmissor irá enviar um byte com algum valor (digamos 85) sempre que o usuário pressionar um botão. O receptor irá ficar em loop esperando chegar um valor e se este valor for 0x55 ele acende um LED.

Você deve estar se perguntando: "Mas esse byte vai ser enviado por onde?". Pois bem, foi por isso que estudamos tantos protocolos seriais. Iremos enviar este byte pela porta serial.

Protocolo

É importante lembrar que, embora simples, este programa que iremos usar possui as características de uma transmissão serial. Neste caso, quando definimos que o byte que iremos transmitir para sinalizar o acendimento do LED é o byte 0x55, estamos definindo nosso próprio protocolo (que irá estar posicionado dentro de outro protocolo, que é o RS232). Ao final da aula, existe uma atividade que irá incrementar este protocolo, fazendo com que existam mais LEDs e mais botões.

Software do Transmissor

Como explicamos anteriormente, este programa inicial é super simples. Ele apenas irá verificar o estado de uma chave e caso esta chave seja pressionada um byte será enviado para a porta serial.

A seguir está a listagem. Iremos comentar passo a passo as partes do código para entendermos o que se passa.

```
1  #include "p18f45k20.h"
2  // "PBADEN = OFF" define que os bits 1, 2, 3 e 4 da porta B
3  // devem ser configurados como pinos I/O digitais.
4  #pragma config PBADEN = OFF
5
6  // esta função serve apenas para organizar o código
7  // ela configura os registradores da comunicação
8  // serial. Para este caso, 1200 bps, 8 bits, sem
9  // bit de paridade e 1 stop bit
10 void configura_serial()
11 {
12     // configura o transmissor
13     TXSTA = 0b00100000;
14     // configura o receptor
15     RCSTA = 0b10010000;
16
```

```
17      // velocidade = Fosc/(64*(SPBRG+1))
18      SPBRG = 12; // 1200 bps
19  }
20
21  void main()
22  {
23      // variável que verificará o estado da chave
24      unsigned char chave;
25
26      // configura a porta B como entrada
27      TRISB = 0xff;
28
29      // chama a função para configurar a serial
30      configura_serial();
31
32      // repete para sempre
33      while(1){
34
35          // ler o estado da chave
36          chave = PORTB;
37
38          // se a chave estiver pressionada
39          if (chave>0) {
40                  // transmite o valor 85 (bits alternados)
41                  TXREG = 0b01010101;
42
43                  // espera 10 ciclos
44                  Delay10TCYx(1);
45
46                  // espera transmitir o byte
47                  while (!(PIR1&0b00010000));
48          }
49      }
50  }
```

```
1  #include "p18f45k20.h"
```

Essa linha inclui no programa o arquivo com as funções que o PIC possui e define as variáveis que serão nomes dos registradores. Observe que ela tem o nome exato do PIC que iremos usar, pois os registradores e funções variam de PIC para PIC.

```
1  void configura_serial()
2  {
3      // configura o transmissor
4      TXSTA = 0b00100000;
5      // configura o receptor
6      RCSTA = 0b10010000;
7
8      // velocidade = Fosc/(64*(SPBRG+1))
```

```
 9     SPBRG = 12; // 1200 bps
10 }
```

Nesse block, estamos definindo uma função que configura os parâmetros da porta serial. A primeira coisa a fazer é configurar a transmissão (ver datasheet do PIC) no registrador TXSTA. É aí que definimos que não vai haver bit de paridade e que teremos apenas um stop bit. Em uma transmissão serial, temos que configurar também o registrador de recepção (embora não seja utilizado aqui) e isso é feito no registrador RCSTA. Após isso, temos que ajustar o divisor de clock que fará com que a taxa de transmissão seja a que desejamos. Isso é feito no registrador SPBRG.

Se Liga!

Para calcular o valor a ser colocado no registrador SPBRG, temos que ter a frequência de clock do PIC e a taxa de transferência desejada. Dessa forma, fazemos o seguinte:

$$SPBRG = \frac{F_{clock}}{64 \cdot BPS} - 1$$

No caso do nosso exemplo, como queremos 1200bps e iremos usar um cristal de 1MHz, temos:

$$SPBRG = \frac{1000000}{64 \cdot 1200} - 1 = 12$$

```
 1 void main()
 2 {
 3     // variável que verificará o estado da chave
 4     unsigned char chave;
 5 
 6     // configura a porta B como saída
 7     TRISB = 0xff;
 8 
 9     // chama a função para configurar a serial
10     configura_serial();
```

Aqui começa a execução. Definimos uma variável para receber o estado da chave (que iremos pressionar para enviar o byte), definimos a porta que a chave está ligada (no caso a porta B do PIC) como entrada e, finalmente, chamamos a função que configura a porta serial (que comentamos há pouco).

```
1  // repete para sempre
2      while(1){
3
4          // ler o estado da chave
5          chave = PORTB;;
```

Aqui é o laço principal do programa. Ele repetirá para sempre e a primeira coisa que faz é ler o estado da porta B (e, consequentemente, da chave).

```
1  // se a chave estiver pressionada
2          if (chave>0) {
3              // transmite o valor 85 (bits alternados)
4              TXREG = 0b01010101;
5
6              // espera 10 ciclos
7              Delay10TCYx(1);
8
9              // espera transmitir o byte
10             while (!(PIR1&0b00010000));
11         }
12     }
13 }
```

Caso a chave seja pressionada, um valor que não é zero (e que depende de qual bit a chave esteja conectada) irá aparecer na variável chave. Então, nós transmitimos o byte (bastando escrever no registrador TXREG) e depois esperamos a transmissão terminar (testando o 5º bit do registrador PIR1).

Software do receptor

Como no transmissor, iremos colocar o código aqui e comentar. Basicamente, o que o receptor faz é (em um laço) esperar chegar algum valor na porta serial e, caso chegue, testa se é o valor que estamos esperando. Se for, acende o LED.

```
#include "p18f45k20.h"

// esta função serve apenas para organizar o código
// ela configura os registradores da comunicação
// serial. Para este caso, 1200 bps, 8 bits, sem
// bit de paridade e 1 stop bit
void configura_serial()
{
    // configura o transmissor
    TXSTA = 0b00100000;
    // configura o receptor
    RCSTA = 0b10010000;

    // velocidade = Fosc/(64*(SPBRG+1))
    SPBRG = 12; // 1200 bps
}



void main (void)
{
    // variável que armazenará o valor recebido da serial
    unsigned char valor_recebido;

    // chama a função para configurar a serial
    configura_serial();

    // configura a porta B como saída
    TRISB = 0;

    // repete para sempre
    while(1){

        // prepara para receber
        RCSTA = 0b10010000;
        // espera chegar algum dado na serial
        while ((PIR1&32)==0);

            // ler o valor recebido
            valor_recebido = RCREG;

            // reseta a serial para uma nova recepção
            RCSTA = 0b10000000;

            // compara para ver se é o valor correto
            // e então acende o LED
            if (valor_recebido == 85)
                PORTB = 0xff;

            // espera alguns instantes
            Delay10KTCYx(10);

            // apaga o LED
```

```
54              PORTB=0;
55      }
56  }
```

As linhas do include e da função *configura_serial* são as mesmas do transmissor, por isso não iremos comentar.

```
1  void main (void)
2  {
3     // variável que armazenará o valor recebido da serial
4       unsigned char valor_recebido;
5
6     // chama a função para configurar a serial
7     configura_serial();
8
9     // configura a porta B como saída
10    TRISB = 0;
```

Aí se inicia a execução do programa. Definimos uma variável para guardar o valor recebido pela serial e configuramos a porta. É necessário também configurar a porta B como saída (pois o LED irá conectar-se a ela).

```
1  // repete para sempre
2       while(1){
3
4           // prepara para receber
5           RCSTA = 0b10010000;
6           // espera chegar algum dado na serial
7           while ((PIR1&32)==0);
```

Aqui o laço começa. Precisamos iniciar a recepção colocando o valor de configuração no registrador RCSTA (o mesmo que configuramos na função de configurar a porta). Depois disso prendemos o programa em um laço que só sai quando o 6ª bit do registrador PIR1 é ligado (indicando que chegou um byte na serial).

```
// ler o valor recebido
        valor_recebido = RCREG;

        // reseta a serial para uma nova recepção
        RCSTA = 0b10000000;
```

Para ler o byte que chegou, simplesmente, lemos o valor do registrador RCREG. Em seguida, inicializamos o receptor novamente (para que fique pronto novamente para ler outro byte).

```
// compara para ver se é o valor correto
        // e então acende o LED
        if (valor_recebido == 85)
            PORTB = 0xff;

        // espera alguns instantes
        Delay10KTCYx(10);

        // apaga o LED
        PORTB=0;
    }
}
```

Finalmente, testamos se o valor recebido foi o esperado (85) e, se for, ligamos os bits da porta B (acendendo o LED). Após isso, esperamos um tempinho (para o usuário ver que o LED acendeu) e depois o apagamos. O laço volta e todo o processo se repete.

Se Liga!

A forma de escrever um byte, usada ainda há pouco (0x55), refere-se a um valor HEXADECIMAL (neste caso o valor 85). Mas porque usamos esse valor?

Bem, poderia ser qualquer valor, mas esse aí em binário é 01010101, o que fica fácil de gerar com um gerador de sinais e testar o receptor sem o transmissor!

Parte II (*hardware*)

Nesta parte de *hardware*, iremos descrever a montagem do sistema em si, suas conexões elétricas, ligações com o microcontrolador etc. Como descrevemos no início, deverão existir duas montagens, uma para o transmissor e outra para o receptor. Em cada uma verificaremos o seu funcionamento apenas observando e gerando os bits de uma transmissão. Nas próximas aulas, faremos a conexão de ambos.

Transmissor

Na **Figura 1**, podemos ver o circuito utilizado como transmissor. Basicamente, temos o microcontrolador (e seu circuito de oscilador e reset) e as ligações para uma chave e um osciloscópio.

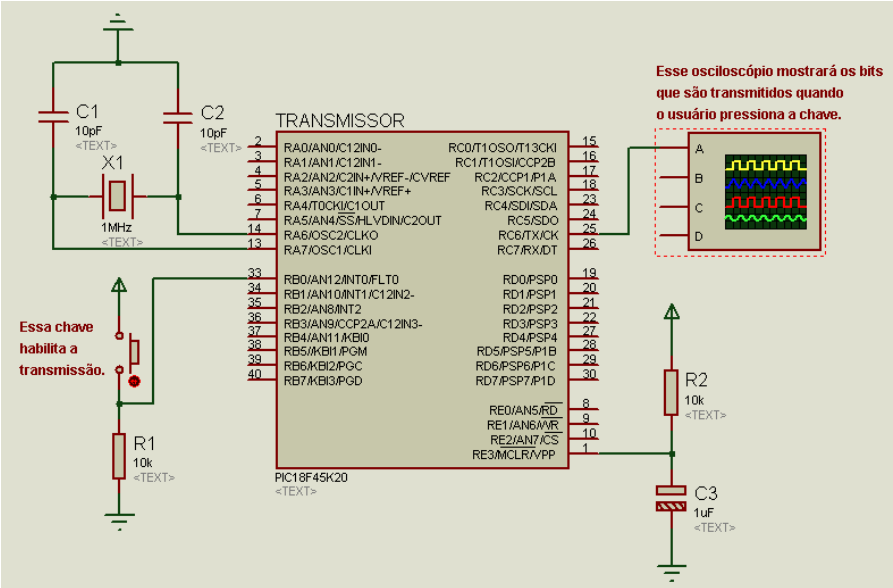

Figura 1 - Circuito do transmissor usado para testar o *software* de transmissão.
Fonte: Autor

O que esse circuito faz é rodar o programinha do transmissor que, ao se pressionar a chave, envia o valor 85 para porta serial. Esta, por sua vez, está conectada a um osciloscópio. Dessa maneira, o que deve acontecer é que o osciloscópio vai mostrar uma forma de onda que será composta por pulsos de 0V e 5V alternados. Os pulsos correspondem aos valores 0 e 1 enviados para a serial. Como o valor 85 em binário é 10101010, o que devemos ver é um sinal quadrado de pulsos, confirmando que o PIC está, realmente, enviando o valor.

Receptor

A **Figura 2** mostra o circuito do receptor. Como no transmissor, temos o PIC e seu circuito de reset e oscilador. Agora, no lugar da chave temos um LED e no lugar do osciloscópio temos um gerador de sinais.

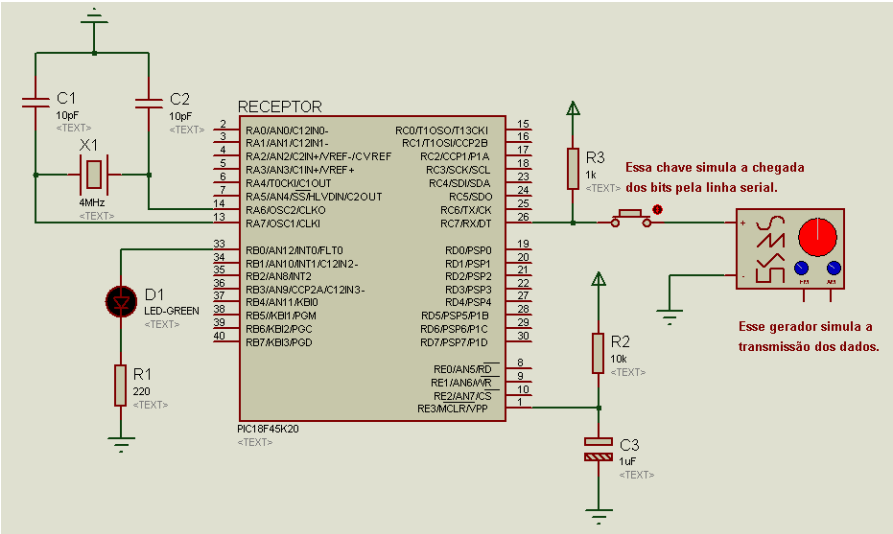

Figura 2 - Circuito do receptor usado para testar o *software* de recepção.
Fonte: Autor

Neste circuito, o gerador de sinais é configurado para gerar um sinal de 600Hz para simular uma sequência de bits (que estão a 1200bps, porque o dobro?). O PIC, então, receberá esses valores e acenderá o LED. Observe que o LED só acende quando pressionamos a chave, pois ela simula o fato de o transmissor estar ou não transmitindo algo.

Atividade 01

1. Modifique o valor 85 para outro valor e veja a diferença nos circuitos tanto do transmissor quanto do receptor. Comente seus resultados.

Resumo

Nesta aula, vimos como implementar um sistema bem simples para transmitir um sinal de um microcontrolador para outro. O sinal enviado foi apenas um byte, que sinalizaria para o receptor o acendimento de um LED. Vimos também que a transmissão se deu utilizando a porta serial do computador. Neste primeiro exemplo, a transmissão se deu com fio. O próximo passo será realizar a transmissão sem fios.

Autoavaliação

1. Qual a função do programa do transmissor usado nesta aula?
2. Qual a função do programa do receptor usado nesta aula?
3. É, realmente, necessário usar 1 byte no sistema desta aula? Por quê?
4. Por não foi preciso utilizar o MAX232 (descrito nas aulas anteriores) na comunicação serial?
5. Qual seria o valor do registrador de divisão de clock caso queiramos uma transmissão a 25000bps usando um cristal de 16MHz?

Referências

LUZ, Carlos Eduardo Sandrini. **PROGRAMANDO MICROCONTROLADORES PIC EM C**. 1. ed. São Paulo: Ensino Profissional, 2011.