Comunicação serial

Introdução.

Durante a graduação, fiz cursos de microprocessadores de 8 bits (Z80, 8085) e microcontroladores (8051). Uma das tarefas obrigatórias era programar em Assembly uma rotina para comunicar dados de um teclado até um motor de passo. Isso deveria ser feito utilizando o 8085 e além do programa precisamos projetar todo o hardware. Como você deve imaginar, não foi uma tarefa fácil. Primeiro, porque Assembly não é uma linguagem muito acessível (embora fosse um curso de Assembly e isso era obrigatório). Segundo, porque era preciso lidar com máscaras de interrupção, sincronizar clock, programar uma porta paralela, lidar com o código ASCII gerado pelo teclado. Até hoje me lembro de ter ficado uma semana extra na faculdade enquanto meus colegas já gozavam das férias, e eu preso dentro de um laboratório com meu caro colega Raul gravando e apagando EPROMs.

O microprocessador era bastante rústico e não possuía UART/USART. Ler um dado do teclado e enviá-lo ao motor de passo era muito complicado. Imagine que a entrada seja um trem de pulsos, seu código deverá reconhecer stop bits, os dados, paridade e ainda deverá estar sincronizado.

Com o padrão atual de microprocessadores para PCs, não é preciso esse trabalho, basta usar os recursos de alto nível. Para WIN32 temos as opções das APIs e Visual Basic. E a melhor delas todas, por ser um meio termo, é a API Java Communications da Sun. A máquina virtual Java faz o trabalho sujo para você. É nessas horas que eu agradeço a Deus por ter criado (ou inspirado os homens a criar) o termo “encapsulamento” e ao James Gosling e companhia pelo Java.

Para testar, você vai precisar conectar os polos de um múltimetro multímetro na saída TX da sua porta (por exemplo) COM2 colocando o polo positivo na saída TX e o negativo no pino GND. Veja, na figura 3 abaixo, a pinagem do conector da porta serial do seu PC (chamada de conector serial DB9). Se você quiser monitorar os dados de saída espete o multímetro entre os terminais 3 (TX- Transmitted data) e 5 (terra). Se você desejar ler dados, conecte o seu gerador de bits entre os pinos 2 (RX- Received Data) e 5 (terra). No mundo real, não utilizamos gerador de pulsos. Abaixo vamos deixar isso mais claro.

Só gostaria de dizer que acredito que no futuro, as portas seriais e os protocolos de comunicação serial serão substituídos pelo padrão USB ("Universal Serial Bus") e também torço e espero que a SUN implemente uma API para que possamos programar as nossas portas USB.

Na verdade, há um link bem interessante que eu encontrei outro dia com detalhes de como ampliar a API de comunicação Java para que ela possa suportar a USB:

http://www.syngress.com/book_catalog/177_lego_JAVA/sample.htm

Aliás, se você puder ler em Inglês, é uma ótima referência para ler sobre a API de comunicação do SUN para Java.

O Padrão EIA-232

Também incorretamente chamado de "RS-232", o padrão EIA-232 específica o meio físico de comunicação serial. Acrescido ao número, há uma letra que indica a revisão atual do padrão. Se não me engano, e posso me enganar, a versão atual é EIA-232-F.

O nível de tensão varia de -12 Volts à -3 Volts para o nível lógico 1 e de +3 Volts à +12 Volts para nível lógico 0.

Uma outra terminologia utilizada abrevia o equipamento que envia os dados, DTE (Data terminal equipament), e o que receve, DCE (Data communication equipament). Um DTE transmite os dados para um DCE (um modem por exemplo). Note que devemos conectar os terminais de transmissão de dados corretamente, de modo que, o pino de envio de dados do DTE seja ligado ao pino de recebimento de dados do DCE e vice versa. Há um limite para o tamanho dos cabos a serem utilizados. Para comunicações de dados em distâncias maiores, é indicado que o padrão EIA 485 seja utilizado.

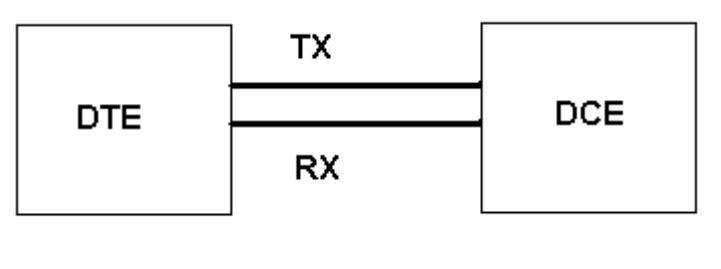

Fig 1- DTE e DCE

O sinal transmitido, no formato serial assíncrono é mostrado abaixo:

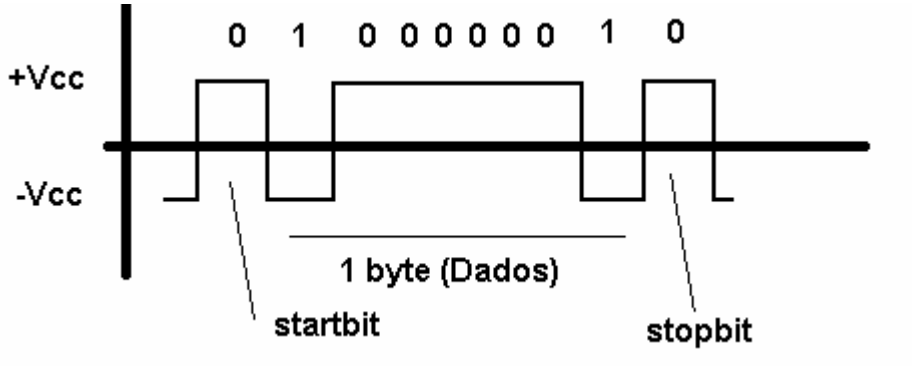

Fig 2- Byte de dados serial

Um bit sinalizador informa ao equipamento que receberá os dados, que uma mensagem será enviada, em seguida, os dados são transmitidos como mostrado acima. Ao final, há um bit que indica o fim da transmissão dos dados.

Outras informações

- Baudrate: é taxa de bits por segundo que é transmitida. Assim, quando falarmos em um baudrate de 9600 bps, queremos dizer 9600 bits por segundo, ou seja, cada bit dura 1/9600 segundos.
- Timeout: é o tempo que a CPU espera para obter uma resposta do DCE. Caso esse tempo seja excedido, a comunicação não é estabelecida.
- Parity : Paridade. Os bits de paridade são bits de controle. Paridade par significa que o total de níveis “1” lógicos transmitidos juntamente com a paridade associada deve ser um número par.

Como meu objetivo é mostrar o uso da API, não vou tentar descrever em maiores detalhes o padrão. Só devo deixar claro, que o programador não precisa converter os dados seriais para paralelo diretamente no código. Isso quem faz é a UART do PC. Ela paraleliza os dados de modo que a CPU possa entender os dados e processá-los. Os DCEs podem possuir drivers 232. Normalmente, são dispositivos cujas CPUs são microcontroladores. Alguns possuem sistemas operacionais proprietários que já suportam até compiladores C++. Novamente, basta ao programador encapsular parte da complexidade e utilizar classes já prontas. Note também que não foi preciso falar em interrupção e outros detalhes mais cabeludos.

Pinagem do conector DB9

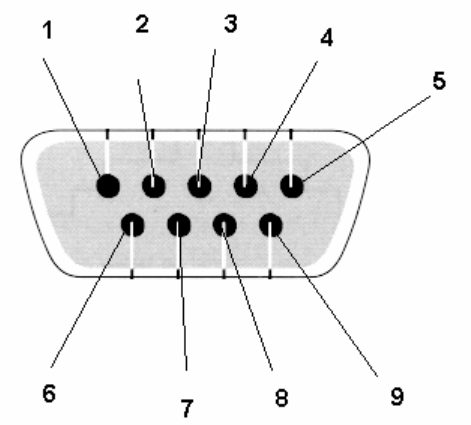

Fig 3-Pinagem do conector DB9

1- DCC (Sinal de controle)
2- Dados recebidos
3- Dados transmitidos
4- DTR (Terminal de dados pronto) Sinal de controle
5- Terra
6- DSR (Sinal de controle)
7- RTS(Requisição para enviar) Sinal de controle
8- CTS (Pronto para enviar) Sinal de controle
9- Indicador de Anel

Em termos práticos, devemos nos preocupar apenas com 1 (TX), 2(RX) e o terra. A tensão lida entre os terminais 3 e 5 fornece o sinal de saída/transmitido. O sinal lido entre 2 e 5 nos fornece o sinal recebido. Os outros pinos nos fornecem sinais de controle (todos acessíveis através da API da SUN é claro) conhecidos como “handshaking”.

Ligação de hardware externos

Como já disse, dispositivos seriais, usam drivers 232 prontos. Mas, para efeito de testes basta um trem de pulsos. A UART converte os dados seriais para dados paralelos, e esses dados são lidos pela CPU do PC. Não há muito com que se preocupar. Agradeça ao encapsulamento.

Enviamos um trem de pulsos digitais para saída e este sinal é ligado a base do tiristor que é disparado. Da mesma forma, podemos gerar um PWM para acionar um motor. Será preciso apenas um driver de potência que aumenta a tensão para os valores necessários para acionar o motor. De modo semelhante, podemos medir uma temperatura utilizando um resistor que varia com a temperatura. A tensão no resistor é convertida para valores digitais, por um conversor A/D e em seguida este valor é serializado em um trem de pulsos e lido na porta serial.

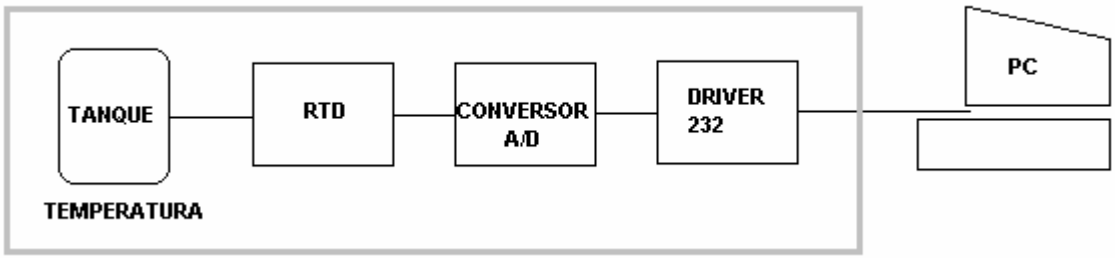

Fig 4-Lendo uma variável através da porta serial

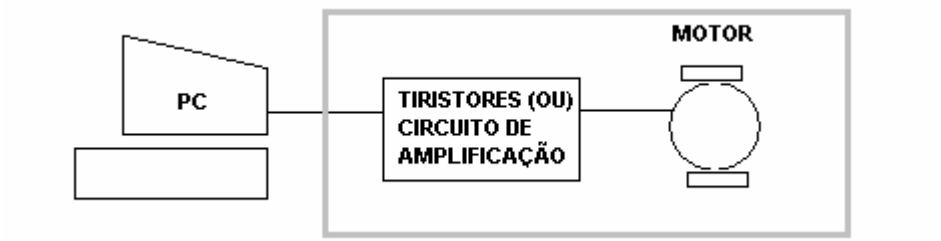

Fig 5- Acionando um motor através da serial

Para maiores detalhes sobre o padrão EIA-232 veja as referências ao final, mas um site bem completo é:

http://www.arcelect.com/rs232.htm

A API de Comunicação da SUN

A SUN fornece como download gratuito a API de comunicação serial e paralela na URL:

http://java.sun.com/products/javacomm/index.jsp

Basta baixar a API e realizar os procedimentos de instalação. Após baixar a API, descompactá-la, você terá:

1. Copiar o arquivo win32com.dll para o diretório C:\JavaSDK\BIN (isto é, o diretório onde o J2SDK foi instalado no seu PC).
2. Copiar o arquivo comm.jar para o diretório C:\JavaSDK\BIN\LIB.
3. Copiar o arquivo javax.comm.properties para o diretório C:\JavaSDK\BIN\LIB.

Em seguida configure o CLASSPATH para que ele reconheça o arquivo comm.jar.

Referências

Sun Communications Api

- http://java.sun.com/developer/Books/javaprogramming/cookbook/
"Java Cook Book", Capítulo 11, disponível para download gratuito .

- http://www.syngress.com/book_catalog/177_lego_JAVA/sample.htm
Outro sample gratuito.

- http://community.borland.com/article/0%2C1410%2C31915%2C00.html
Artigo sobre o uso da API de comunicação elaborado por Rick Proctor da Borland.

- http://java.sun.com/products/javacomm/index.jsp
Site oficial da SUN

Comunicação Serial EIA 232

- http://www.arcelect.com/rs232.htm
Uma descrição bem completa do padrão serial

- http://www.idc-online.com/html/pocket_guides_free_download.html
IDC. Eles têm alguns tutoriais gratuitos em pdf muito bons sobre conversores A/D, D/A, automação industrial e comunicações (serial, TCP-IP).

- http://users.telenet.be/educypedia/
Vários links interessantes. Aliás, milhares. Não somente relacionados à comunicação serial.