Porta USB

Prof. Eduardo Augusto Bezerra <eduardob@inf.pucrs.br>
Faculdade de Informática, PUC-RS

Porto Alegre, Junho de 2006

http://www.inf.pucrs.br/~eduardob/disciplinas/ProgPerif/usb/

- Breve histórico
  - Grandes empresas se unem para definir um novo padrão para interface serial (DEC, IBM, Intel, Microsoft, Compac);
  - Criado o USB-IF (Implementers Forum) para tomar decisões sobre o padrão;
  - **[1995] USB 1.0** – primeiros dispositivos no mercado em 1996;
  - **[1998] USB 1.1** – primeiro padrão USB largamente utilizado;
  - **[2000] USB 2.0** – 40x mais rápido que USB 1.1;
  - **[2001] USB OTG 1.0** (*on-the-go*) – complemento ao USB 2.0 para utilização em dispositivos portáteis;
  - **[2005] USB *Wireless*** - substituto natural do USB com fio – 2.0 provavelmente será o último lançamento de USB com fio;

- USB 1.1 (*FULL-SPEED USB*)
  - 1,5 Mbps e 12 Mbps

- USB 2.0 (*HIGH-SPEED USB*)
  - 480 Mbps (ou 60MBps)
  - 60MBps é uma taxa teórica, pois na prática a taxa máxima medida é de 40MBps, devido a detalhes na implementação do protocolo no momento da formação dos pacotes dos frames USB.

- USB 1.1 vs. USB 2.0
  - USB 2.0 é totalmente compatível com USB 1.1;
  - Mesmos conectores, velocidades diferentes;
  - Tempo para transferir 450MBytes de drive USB para HD
    - USB 1.1 ~ 8 minutos
    - USB 2.0 ~ 30 segundos

- USB OTG 1.0
  - Conector de tamanho reduzido para utilização em dispositivos portáteis;
  - Baixo consumo de energia (*low power*);

    - Capacidade para comunicação com outros periféricos USB, sem a necessidade de um *host*.

- USB *Wireless*
    - Baseado no USB 2.0;
    - 480 Mbps – para comunicação em um raio de 2 metros;
    - 110 Mbps – para comunicação em um raio de 10 metros.

- USB é plug-and-play – não precisa reiniciar o computador;

- O padrão USB, exceto o USB OTH, necessita um *host* (PC) para controlar a rede de escravos (periféricos);

- Existem quatro tipos de conectores: Tipo A (*host*), Tipo B (periférico), Mini A e Mini B (PDA, celular);

- Aspectos positivos:
    - Formato dos conectores padronizado;
    - Características elétricas padronizado;
    - Protocolo padronizado;
    - Excelente documentação – de acesso livre.

- Aspectos negativos:
    - Formato do conector dificulta conexão quando é necessário usar o "tato" – Ex. Uma porta USB localizada atrás de um PC é de difícil conexão.
    - Alguns fabricantes (câmeras, celulares) utilizam conectores Tipo B que não seguem o padrão.

- A complexidade das pinagens (sinais) existente nos padrões para comunicação serial (RS-232C) é resolvida no padrão USB por intermédio de camadas de software. O padrão USB gerencia, em software, toda a complexidade da conexão de multiplos dispositivos com diferentes velocidades.

- USB classifica o hardware serial em dois tipos: *hubs* e funções. Um *hub* USB possui "tomadas" nas quais podem ser conectadas funções. Uma função USB e o dispositivo (periférico) propriamente dito.

- O meio USB funciona como um barramento permitindo a conexão de diversos periféricos a uma mesma porta USB no computador pessoal. Todos os periféricos compartilham exatamente o mesmo sinal.

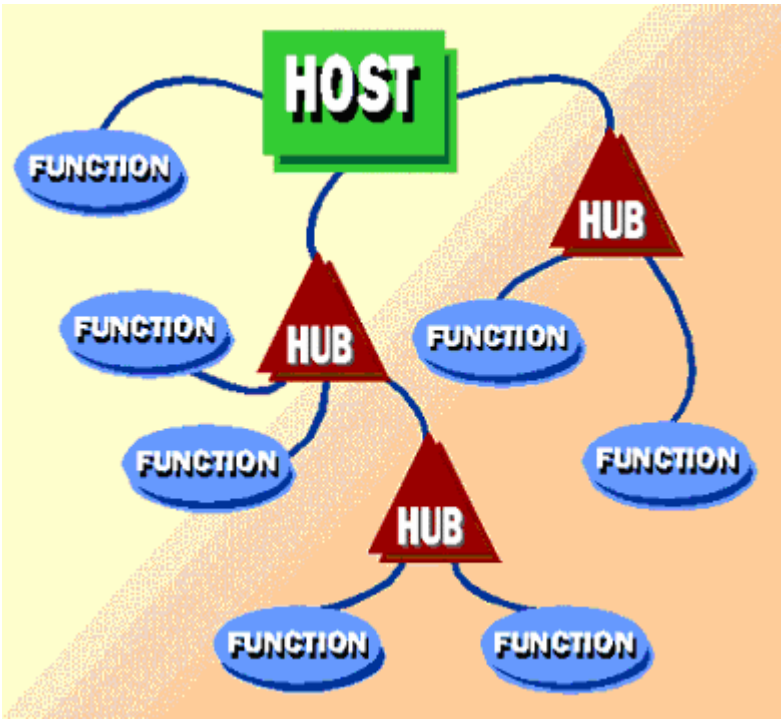
- Topologia estrela – *hubs* atuam como portas para conexão de outros dispositivos USB. Apenas um dispositivo precisa estar conectado no *host*. Uma porta USB pode suportar até 127 dispositivos. **Limite: hub com 7 portas – após esse limite a transferência fica lenta com grande perda de pacotes.**

- A informação trafega no barramento na forma de pacotes, e todas as funções (periféricos) recebem todos os pacotes. O computador pessoal acessa funções individuais por meio da inclusão de endereços nos pacotes. Apenas a função endereçada utiliza o pacote enviado.

- O projeto do padrão USB permite a utilização de *hubs* de forma hierárquica, com *hubs* conectados a *hubs* que são conectados a *hubs* e assim por diante, funcionando como uma estrutura de arvore, como mostrado na figura a seguir.

- O computador pessoal é o *hub* base para o sistema USB e é denominado *host*. O software/hardware no computador pessoal que controla o *hub* e todo o sistema USB e denominado "controlador do barramento". Cada sistema USB possui apenas um controlador de barramento.

- A parte mais complexa do sistema é o software que implementa o protocolo USB. Esse software se encarrega de gerenciar toda a árvore de *hubs* e funções, que é construída de forma simples bastando utilizar as regras para conexão ilustradas na figura anterior. Não existe um limite teórico para o numero de *hubs* a serem utilizados, porem existe um numero máximo de 127 funções a serem utilizadas em um sistema (árvore) USB. Esse limite é imposto pelos 7 bits utilizados no endereçamento das funções (um endereço e reservado).



- Outro limite é o comprimento de no máximo 5 metros que um cabo USB pode ter para conectar uma função a um barramento. Porem, como *hubs* podem fortalecer o sinal, um sistema USB pode se prolongar por grandes distâncias ao se utilizar diversos *hubs*.

- Como parte do processo *plug-and-play*, o controlador USB realiza uma caça a dispositivos na inicialização do computador pessoal. Cada dispositivo USB conectado ao barramento e interrogado e um mapa e construido localizando cada dispositivo por *hub* e endereço da porta. Essa informação se torna parte do endereçamento.

- Devido ao uso desse sistema de endereçamento, cada dispositivo USB precisa ter um software basico para entender o protocolo. No computador pessoal, cada função precisa ter um driver, normalmente em software, responsavel por gerar os comandos ou pacotes de dados para o dispositivo associado.

- Um driver USB funciona como um provedor de servicos, fornecendo o canal (*pipe*) para roteamento dos dados para as diversas funções. Consequentemente, para cada dispositivo USB adicionado ao barramento e necessario a instalação do driver (software).

- Os cabos USB possuem em uma ponta conectores do tipo A e na outra ponta conectores do tipo B. Isso evita ligações incorretas, uma vez que todas as portas A são saidas e todas as portas B são entradas.

- Os cabos possuem 4 fios. Um para alimentação (5 Volts), um terra (GND), e dois para os dados. O fio de alimentação permite o fornecimento de tensão para o dispositivo periférico. Os dois fios de dados são trancados e conduzem os dados na forma de um sinal digital diferencial, ou seja, possuem um sinal de magnitude igual, porem de polaridade invertida. Isso e utilizado de forma que quando os dois sinais são subtraidos o resultado cancela qualquer ruido que venha a aparecer nos fios.

- Os fios de dados são verde (sinal positivo D+ na figura abaixo) e branco (sinal negativo D- na figura abaixo). A alimentação e o fio vermelho, e o terra o fio preto.

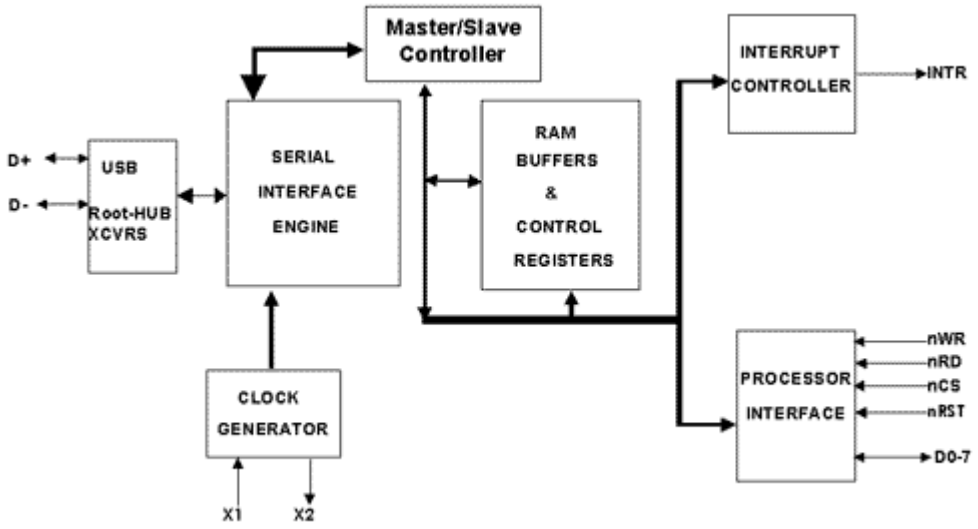


Figure 4-1. SL811HS USB Host/Slave Controller Functional Block Diagram

- Como o forte do padrão USB e o protocolo implementado em software, e como chips decodificadores USB (ex. Cypress SL811HS) são de preco bastante acessivel podendo ser facilmente adicionados ao dispositivo periférico a ser conectado ao barramento, a codificação dos sinais nos fios nao e apresentada aqui. O mais importante e um bom entendimento do protocolo USB de forma a facilitar a escrita do driver para o novo dispositivo.

- **O protocolo USB é baseado em pacotes**. Todas as mensagens trocadas necessitam de três pacotes. Uma troca de mensagens inicia com o *host* enviando um **pacote *tocken***. O pacote *tocken* possui o endereço do dispositivo desejado e tambem informação de controle descrevendo a natureza da mensagem. Dependendo da natureza da operação, o *host* ou o dispositivo envia o **pacote de**

dados. Apesar desse nome, o pacote de dados poderá estar vazio, sem nenhuma imformação. A troca de dados finaliza com o recebimento de um **pacote ACK** que informa o recebimento do pacote de dados. Um quarto tipo de pacote, denominado **pacote Especial**, e utilizado para funções adicionais.

- Todos os pacotes iniciam com dois componentes de um byte cada: um campo de sincronismo (*Sync*) e uma identificação.

- O campo de sincronismo gera uma rajada de bits no barramento USB fazendo com que todos os dispositivos conectados resetem seus relogios e sincronizem com o *host*. Esse campo aparece no barramento codificado como três pulsos on/off seguidos por uma marca de largura de dois bits.

- O byte identificador do pacote contem quatro bits que definem a natureza do pacote, e quatro bits utilizados para confirmar a validade dos quatro primeiros bits. Os quatro bits de verificação são o complemento de um dos quatro primeiros bits. Os quatro bits de identificação possibilitam a definição de 16 tipos diferentes de pacotes. Os dois bits mais significativos especificam um entre quatro tipos de pacotes. Os dois bits menos significativos sub-dividem a categoria do pacote. A tabela a seguir lista o pacote de identificação dos quatro tipos basicos de pacotes USB.

Byte identificador	Tipo do pacote
XX00XX11	Especial
XX01XX10	Token
XX10XX01	Handshake (ACK)
XX11XX00	Dados

- Apenas o *host* envia **pacotes do tipo token**. Esses pacotes possuem quatro bytes, e se divide em cinco partes, conforme mostrado na figura a seguir.

- Os dois primeiros bytes seguem o padrão de todos os pacotes USB. O primeiro byte e um campo de sincronismo que marca o inicio dos bits do token. O segundo byte e a identificação do pacote (PID).

- O PID define quatro tipos de pacotes token: pacote de saida que envia dados do *host* para o dispositivo; pacote de entrada que recebe dados no *host* proveniente de um dispositivo; pacote de configuração (*setup*) que endereça um dispositivo especifico; e um pacote de inicio de frame, que ajuda na sincronização do sistema. A tabela a seguir mostra PIDs e respectivos tipos de pacote token.

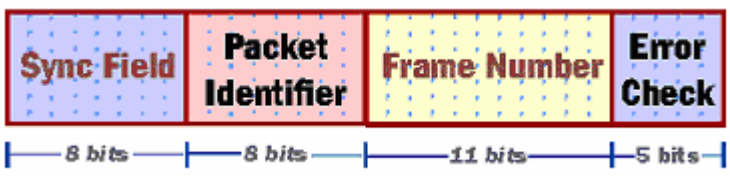
Byte de identificação de pacote (PID)	Tipo de pacote token
00011110	Saida
01011010	Inicio de Frame (SOF)
10010110	Entrada
11010010	Configuração (setup)

- Para pacotes token “Entrada”, “Saida” e “Setup”, os sete bits que seguem o PID representam o campo de endereço, que identifica o dispositivo para o qual o *host* deseja comandar ou enviar dados. Quatro bits adicionais fornecem um codigo *Endpoint*. Um *Endpoint* e uma seção endereçavel individualmente de uma função USB, que possibilita que projetistas de hardware possam separar um dispositivo fisico em diversas unidades logicas. Por exemplo, um teclado com um mouse embutido pode ter um endereço geral para atuar como um dispositivo USB único. Atribuindo *Endpoints* individuais ao teclado e ao mouse, possibilita que os projetistas possam endereçar individualmente cada componente do teclado.

- Pacotes token do tipo inicio de frame (SOF) diferem de outros pacotes USB uma vez que eles são do tipo *broadcast*. Todos os dispositivos do sistema recebem e decodificam esses pacotes, mas nao retornam um ACK referente a eles. Os 11 bits que deveriam ser os campos de endereço e *Endpoint* indicam um numero do frame.

- O *host* envia um pacote token do tipo SOF a cada milisegundo definindo o incio do frame USB denominado *one-millisecond*.

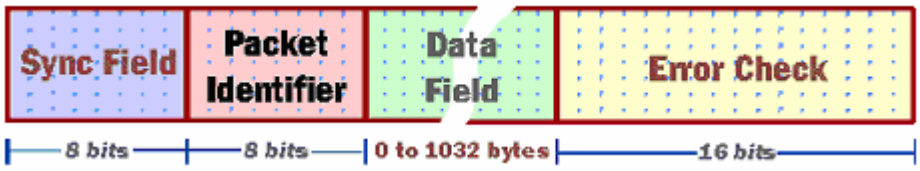
- O *host* atribui numeros de frames de forma incremental, iniciando com zero e adicionando um a cada frame sucessivo. Quando essa contagem atinge o valor máximo de 3072 (11 bits), a mesma e iniciada novamente em zero. A figura a seguir mostra a representação grafica de um pacote token do tipo SOF.



- Todos os pacotes token possuem no final cinco bits de codigo CRC (Cyclic Redundacy Check). O CRC fornece uma forma de verificar a integridade do campo de endereçamento e *Endpoint*. O CRC nao cobre o PID, que possui o sua propria correção de erro embutida.

- **Os pacotes de dados** são os responsaveis pela transmissão da informação em sistemas USB. Um pacote de dados, assim como todos os demais pacotes USB, inicia com um campo de sincronismo de um byte, seguido por um pacote de identificação.

- O dado propriamente dito segue em uma sequencia de tamanho entre 0 e 1,023 bytes. Um campo de CRC de dois bytes e utilizado para verificar a integridade do campo de dados. O campo PID possui um mecanismo proprio para verificação da sua integridade. A figura a seguir mostra a representação grafica de um pacote de dados USB.



- O campo PID define dois tipos de pacotes de dados, Dados 0 e Dados 1. Funcionalmente, contudo, os dois tipos de dados e consequentemente o PID, formam um sistema de verificação de erros adicional entre o transmissor e o receptor. O transmissor altera entre Dados 0 e Dados 1 para indicar que recebeu um ACK valido referente ao pacote de dados anterior. A tabela a seguir lista esses tipos de pacotes de dados.

Identificação do pacote	Tipo do pacote de dados
00110011	Dados 0
10110010	Dados 1

- Por exemplo, o transmissor envia um pacote de dados do tipo Dados 0. Apos o receptor ter decodificado esse pacote com sucesso, ele envia um sinal de confirmação para o transmissor na forma de um pacote de ACK. Se o transmissor receber e decodificar o pacote de ACK com sucesso, o proximo pacote de dados que ele enviara será Dados 1. A partir dessa troca no tipo do pacote de dados, o receptor sabera que o seu ACK foi recebido com sucesso.

- **Pacotes ACK** possuem um tamanho de dois bytes, consistindo de um campo de sincronismo e um pacote de identificação. A figura a seguir mostra um pacote de ACK do padrão USB, e a tabela seguinte lista as três formas desse tipo de pacote.

Byte de identificação do pacote	Tipo de ACK
00101101	ACK
01011010	NAK
11100001	STALL