Comunicação Série: *UART, RS232, I2C, SPI e USB usam comunicação série, transmissão bit-a-bit.*

Tipos de transmissão

Síncrona	**Assíncrona**	**Isócrona**
A transmissão dos dados é acompanhada por um relógio (**baudrate**) I2C ou SPI	Não é transmitido nenhum sinal de relógio (RS232 e USB) é enviado um **bit** de **start** e **stop** para sinalizar o princípio e o fim da transmissão.	O tempo entre dados sucessivos é **igual a unidade de tempo básica do sistema**.

Comunicação assíncrona:

É enviado um **bit** para **sinalizar o fim e o início** da transmissão.

Quando a linha não está a transmitir então está a "1" Mark state (se estivesse a 0 estaria desligada).

Transição de 1 para 0 assinala o iníco de transmissão (bit de início de transmissão).

1 start bit seguido de 8 bits mais 1 ou 2 bits para sinalizar o stop.

UART

Uma UART é uma Universal Asynchronous Receiver Transmiter, transmite de forma sequencial. Ao receber junta todos os bits em bytes usando um conversor série-paralelo.

Os sinais da UART são convertidos por dispositivos de interface para as diferentes tecnologias.

Transmissão

O dado a transmitir é carregado no shift-register. A UART gera um sinal de start para começar o envio. Na saída Serial Out o bit seguinte do shift register é colocado na linha. *(LSB)*

É transmitido o bit de paridade e o stop bit.

Recepção

Na recepção, o dispositivo é controlado por um sinal de relógio de frequência multipla da baudrate. A cada impulso de relógio o receptor testa a entrada para saber se existe um start bit (transisão de 1 para 0). Após o start-bit o receptor começa a armazenar no shift register os valores da entrada a cada impulso de relógio. Depois é lido em paralelo, ficando a UART com a flag para ler um novo dado a 1.

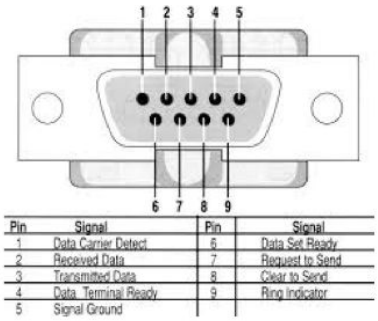
Tipos de comunicação:

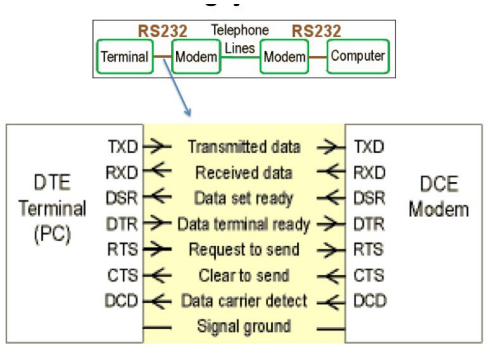
Simplex O dispositivo apenas recebe ou envia	**Half Duplex** O dispositivo recebe e envia alternadamente	**Full duplex** O dispostivo recebe e envia simultâneamente

---

RS232

A RS232 permite comunicação biderecional full-duplex. Usa a mesma estrutura da UART: Start but de 1 para 0; Pode ser usado 1 ou 2 stop bits. A definição de parity bits também é opcional, assim como o baudrate (definido entre 20000 e 19600).





A ligação direta de 2 DTE, sem modems são apenas necessárias 3 linhas e a distância máxima é de 20 metros.

TD	Transmit Data Serial	Data output
TD	Transmit Data	Serial data input
CTS	Clear to Send	O modem está pronto para trocar data
DCD	Data Carrier Detect	Quando o modem deteta data do outro modem no final da linha, ficando assim ativa
DSR	Data Set Ready	Indica que o modem está pronto para estabelecer uma ligação
DTR	Data Terminal Ready	Informa o modem que a UART está pronta para trocar informação
RTS	Request To Send	Informa o modem que a UART está pronta para trocar informação

RI	Ring Indicator	Ativa quando deteta um ringing signal

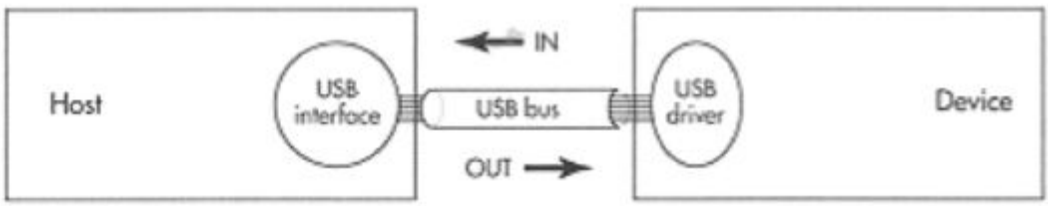
Níveis de tensão: 1 (-3 a -25) e 0 (3 a 25) entre -3 e 3V tem valor indef
Ausência de informação na linha é 1 (-12V) e 0V para corte.
Baud rate: diferente dependendo do sinal transmitido.
Recepção: Frequência de relógio múltipla da bit rate (vários ciclos de relógio durante o tempo correspondente a 1bit) - linha de transmissão amostrada a meio bit.

Códigos de transmissão

RZ Return to Zero	***RZI*** Return to Zero Inverted	***NRZ*** Non-Return to Zero	***NRZI*** Non-return to zero inverted	***Manchester coding***

USB

- No RS232 e UART o utilizador tem de configurar todo o processo. O mesmo já não acontece no USB, que permite a ligação de periféricos a computadores de um modo standard, com um protocolo standard. Os dispositivos interagem com o sistema de operação sem intervenção do utlizador.
- Permite a ligar e retirar dispositivos a qualquer altura sem necessidade de reiniciar o sistema.
- Pode fornecer energia aos periféricos através do mesmo cabo utilizado para a comunicação de dados.
- Permite ligar periféricos com necessidades de taxas de transferências de dados muito mais rápidas.
- Permite ligar periféricos com grandes necessidades de taxas de transferência.



As transferências de dados realizam-se entre o software que está no **Host** (computador) e um **endpoint** específico no dispositivo. Sendo que cada endpoint é uma ligação **simplex** (apenas transferências num só sentido).

Os **endpoints** estão agrupados em **interfaces**. Cada **interface** está **associada a uma única função do dispositivo** (excetuando o endpoint zero usado para a configuração do dispositivo).

Host e Device

Host - é o BUS master. Quando é ligado interrega todos os dispositivos ligados ao BUS e atribui a cada um, um endereço.
Os dispositivos podem ser ligados e desligados sem desligar o host, sem o USB o dispositivo que se ligasse ao sistema teria de obrigar o sistema a reniciar para ser reconhecido. *(Hot-Swappable)*

Tipos de transferências de dados usados

Control transfers	***Bulk* data transfers**	***Interrupt* data transfers**	***Isochronous* data transfers**
São usadas para configurar um dispositivo quando este é ligado ao host.	Transferência de dados em rajada de volumes significativos de dados sem grande exigência temporal. ***Dispositivos que recebem dados em pacote. (impressoras)***	Transferências de poucos dados a serem feitas num dado intervalo de tempo. ***Dispositivos que enviam poucos dados. (teclados)***	(Streaming) transferências com um tempo máximo de latência (os dados fluem entre o dispositivo e o host em tempo real). ***Dispositivos que necessitam de fluxo contínuo de dados. (microfones)***

*Um **pipe** suporta apenas um destes tipos de transferências.*

Camadas do sistema (HOST)

Client S/W	***USB System S/W***	***USB Host Controller***
s/w que executa no host correspondente a um dispositivo USB. Fornecido com o SO ou com o dispositivo USB. ***(manages an interface)***	s/w que suporta USB num dado sistema de operação. ***(manages devices)***	Interface do host com o USB (permite que os dispositivos sejam ligados ao bus) ***(pertence ao USB BUS Interface)***

Interface - *os **endpoints** estão agrupados em interfaces, e cada interface está associada a uma única funão do dispositivo;*

Pipes

Pipe é uma ligação entre um **endpoint** num dispositivo e um software em execução no **host** que se mantem activo.

Stream o USB não impõe um formato aos dados que fluem no pipe. Suporta transferências do tipo bulk, isochronous e interrupt. Os dados fluem de uma extremidade para a outra de modo unidirecional em cada pipe.	***Message*** os dados que circulam no pipe têm um formato importo pelo USB (default control message).

Default control pipe o pipe assoiado aos 2 endpoints zero (IN e OUT) que é criado sempre que um dispositivo é **ligado** e recebeu um **bus reset**. Este **default control pipe** é usado pelo sistema para identificar o dispositivo e configurá-lo.

Message pipes:

1º (host stage): o host envia um pedido ao dispositivo;
2º (data stage): os dados são transferidos no sentido indicado pelo host;
3º (status stage): o estado é transmitido;

Permite a comunicação nos dois sentidos. Um dispositivo só serve um message request a cda message pipe. O S/W USB assegura que não são enviados múltiplos pedidos concorrentes.

Um message pipe requer um único endpoint number no dispositivo em ambas as direções (IN e OUT). O USB não permite que um message pipe esteja associado com endpoint numbers diferentes para cada direção.

IRPs

O **s/w client** solicita a transferência de dados através do **I/O request packets (IRPs)** a um pipe (endpoint to endpoint) e espera. Pode obter dois tipos de resposta: que está ocupado (NAK) ou que a transferência está compelta.

- **Data flow types** as transferências são compostas por 1 ou mais transações. Um IRP corresponde a uma ou mais transferências.
- **Control transfers** Uma control transfer é uma OUT setup transação seguida de multiplas In or Out data transations seguidas de uma com estado contrário.
- **Interrupt transfer** uma interrupt transfer é uma ou mais IN or OUT data transfer.
- **Isochronouns** é uma ou mais transferências IN or OUT data transações.
- **BULK** é uma ou mais transferências IN or OUT data transações.

Transferência de dados

- Os dados são transferidos em pacotes, tanto no caso de stream pipes coo de message pipes.

- A responsabilidade da formatação e interpretação dos dados transportados é da responsabilidade do s/w cliente e da função
- Existem diferentes tipos de transferência para corresponder de modo mais ajustado aos requesitos do s/w do cliente e da função que utilizam no pipe.
- Cada tipo de transferência determina várias características do fluxo de dados:
    - Formato imposto pelo USB
    - Direção da comunicação
    - Tamanho dos pacotes
    - Numero pacote de dados CRC

CRC : usados em todos os campos exeto o do PID nos pacotes token e de dados

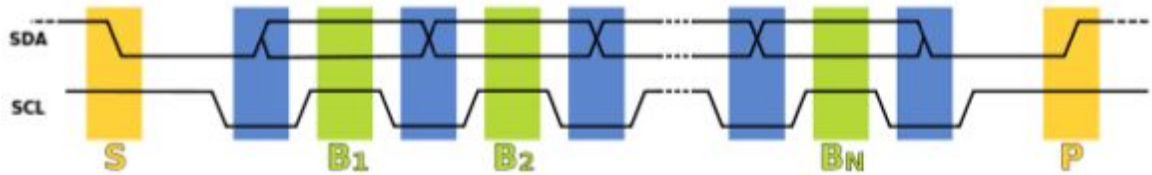
---

I2C

Um bus **I2C** é um **BUS série**, **multi-master** e **half-duplex** (biderecional, serial data line, serial clock line).

Transmiter: dispositivo que **envia** dados para o barramento
Receiver: dispositivo que **recebe** dados do barramento
Master: dispositivo que **inicia** a transferência, **gera** um sinal de relógio e **termina** a transferência. Controla a linha SCL. Controla o endereçamento dos dispositivos.
Slave: endereço endereçado pelo master. Condicionado pelo o estado da linha SCL.

Nos dois modos de operação possíveis (Master transmite - Slave recebe e Slave transmite - Master recebe) o master é o responsável por gerar o sinal de relógio.



É transmitido byte-a-byte, sendo a **transferência iniciada** com um **START bit** que é assinalado com uma mudança na linha **SDA** de **1 -> 0** enquanto **SCL** está **1**.
Cada bit é transmitido em **SDA** com **SCL** a **low** e é **lido** com **SCL** é **high**. O bit presente em SDA tem de se manter válido até que SCL passe de novo a low (SDA transmite o bit com SCL low).
Quando a transferência está compelta, SCL passa a high seguido de SDA a high (stop bits). **MSB first**

Pacotes

START	Slave address	Rd/nWr	ACK	Data	ACK	Data	ACK	STOP
1 bit	7 bits	1 bit	1 bit	8 bits	1 bit	8 bits	1 bit	1 bit

- **Start bit**: transição **1 -> 0** em **SDA** com **SCL** a **1**
- Master **endereça** um slave para transmitir dados (**endereço de 7bits**)
- Master transmite ou recebe (**Rd/nWr**)
- **Data:** Word (8bits) MSB first
  - A cada bit **transmitido** tem de se manter estável enquanto **SCL** estiver **1**
  - Só pode mudar quando **SCL** estiver a **0**
  - Nº de bits transmitidos não tem restrições, seria transmitido um bbyte data seguido de um bit ACK.
- Master lê os dados do slave após o endereçamento
- **Acknowledge:** É feito no 9º impulso de relógico com SCL a 1.
- **Stop bit**: transição de **0 -> 1** em **SDA** com **SCL** a 1
- Cada byte (palavra data de 8 bits) que é transmitido tem de ser acknowledged pelo receptor.

Arbitragem

- Os **2 masters** podem iniciar uma transferência se o **bus** estiver **livre**
  - Podem ambos gerar **START** resultando no bus uma condição válida de **START**
- A arbitragem processa-se **bit a bit**. Durante cada bit, quando **SCL** estiver a **1** cada master verifica se o valor de **SDA** coincide com o que enviou.
- A 1º vez que tentar enviar um bit high e verificar que SDA está a 0, quer dizer que outro master enviou informação, então este master perdeu a arbitragem e desliga o seu driver da linha.
- Arbitragem feita na linha **SDA (a que envia os dados)**

Vantagens

No I2C o hardware é standard e simples assim como o protocologo de comunicação. É fácil remover e adicionar dispositivos. Só duas vias de comunicação.

SPI (Serial Peripheral Interface)

Full-duplex biderecional, comunicação feita em **4 linhas**. Tem apenas um **único BUS master**. Arquetectura master-slave com ligação **ponto-a-ponto**.

Funciona como data-exchange. Para cada bit que é enviado para o receptor também é recebido um bit no master. Isto é, ao fim de N ciclos de relógio o transmissor enviou uma palavra de N bits e recebeu outra com a mesma dimensão.

Comunicação síncrona

Relógio é gerado pelo master que o disponibiliza para todos os slaves. Não é exigida precisão ao relógio e os bits vão sendo transferidos a cada transição de relógio. Isto permite utilizar um oscilador de baixo custo no master.

Shift-registers. MOSI (Master Out Slave In). MISO (Master In Slave Out).

Configuração: baudrate, forma de comunicação flanco ascendente ou flanco descendente.

Modos (pdf)

Modo 0: novo dado na frente descendente (1->0) do relógio. amostragem na frente ascendente (0->1) do relógio

Modo 1: novo dado na frente ascendente (0->1) do relógio. amostragem na frente descendente (1->0) do relógio

...

Tipos de transferências

- **Biderecional:** os dados são transferidos em ambos os sentidos : Master -> Slave e Slave -> Master
- **Master -> Slave** *(para operação de escrita)* o master transfere os dados pretendidos para o slave e ignora/descarta os dados recebidos
- **Slave -> Master** *(para operação de leitura)* o master desencadeira a transferência quando pretende ler dados dos slave. O slave ignora todos os dados recebidos.

Controller Area Network

Vantagens do CAN

- Standard bem definido

- Existência de muitos produtos e ferramentas CAN no mercado
- Protocolo implementado em hardware
- O tratamento de erros e transmissão são bem definidos e têm velocidades elevadas
    - CRC para detetar erros
    - mecanismo para evitar que um nó avariado bloqueie o sistema
- Muito usado na indústria e em veículos de tranposrte
- A melhor relação preço/ performance
- Meio de transmissão simples
    - Twisted pair
    - ou um fio único

Características

- Bus multi-master, assíncrono
- Inexistência de endereçamento de nós
- Broadcasting das mensagens (todos recebem as mensagens)
- É uma rede fechada, não é preciso logins ou interface com o user

CANopen

Características

- CANopen é um subconjunto de CAN Application Layer
- Tem auto-configuração de rede
- Acesso fácil a todos os parâmetros dos dispositivos
- Sincronização dos dispositivos
- Transferência de dados cíclica e desencadeada por eventos
- Estabelecimento de leitura inputs, outputs ou parâmetros síncrona

Protocol

- Cada nó é recetor e transmissor
- Cada mensagem é transmitida a todos os dispositivos ligados ao bus (broadcast)
- Todos os nós lêem a mensagem, e decidem se é relevante para eles
- Todos os nós verificam a inexistência de erros na recepção
- Os nós fazem o acknowledge da recepção