# Tópico 9

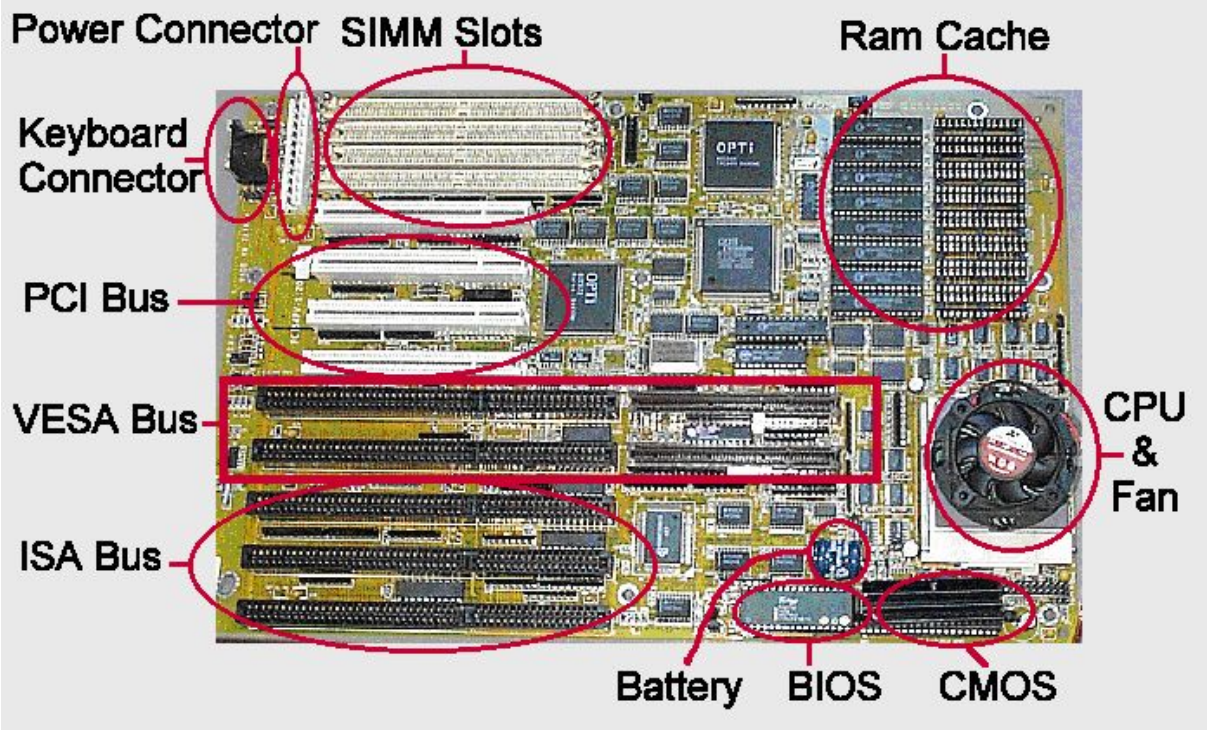
# Barramentos

Autores: José Raimundo de Oliveira e Wu Shin-Ting

DCA - FEEC - Unicamp

Setembro de 2019

Num sistema computacional, **barramentos** são conexões elétricas (fios ou trilhas) que viabilizam transferências dos dados entre os seus componentes. Cada linha do barramento transfere o sinal de um único *bit*. Duas funções vitais são atribuídas a eles no contexto de um sistema computacional: alternativa eficiente para as tradicionais conexões ponto-a-ponto (PTP) e padronização das interfaces entre os módulos e as linhas de conexão. Pois, a ligação por barramento utiliza um único conjunto de fios para interligar diversas unidades de um sistema de computadores. Desta forma, o número de ligações (fios) é independente do número de unidades a ser interligada. A adição de uma nova unidade ao sistema é feita de forma bastante simples, basta *plugar* a nova unidade ao barramento já existente. Figura 9.1 ilustra um circuito impresso com vários barramentos conectando diferentes módulos.



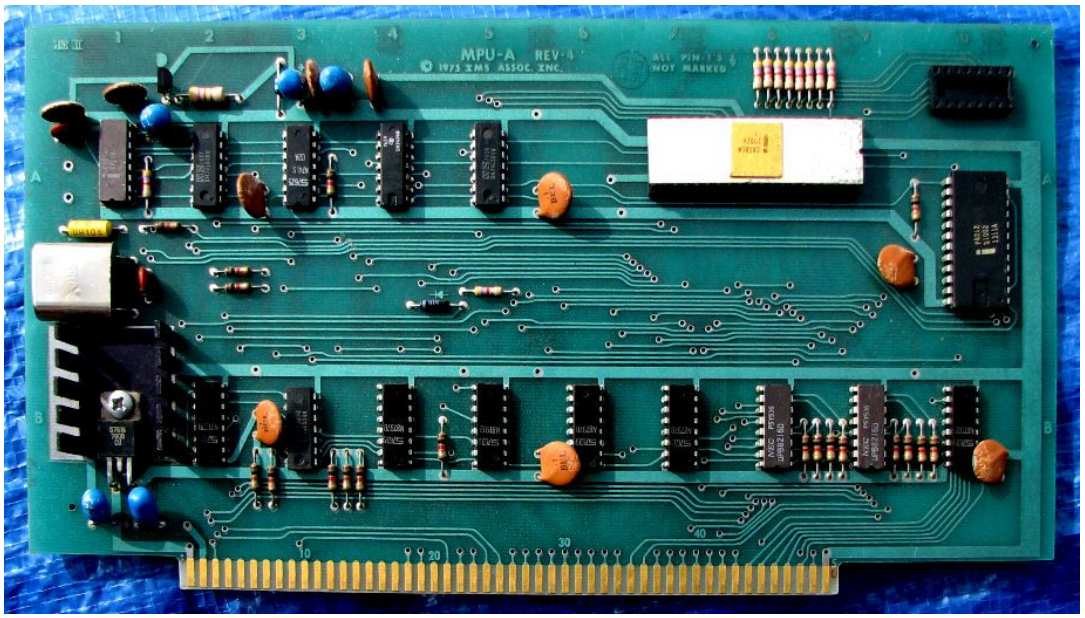



Fig. 9.1: Barramentos num circuito impresso: trilhas de conexões.

Muito mais do que simples fios que ligam os módulos de um sistema computacional, o projeto de um barramento leva em conta as características mecânicas, elétricas e o protocolo de comunicação (Figura 9.2). A especificação mecânica estabelece os aspectos físicos de um barramento, como o tamanho, material e conectores. A especificação elétrica determina tanto o nível de tensão e de corrente dos sinais transportados num barramento quanto as características dos elementos ativos (**acionadores** ou *drives*) e dos elementos passivos (**receptores**) de um barramento. E, tipicamente, o **protocolo** diz respeito ao sequenciamento dos sinais envolvido numa troca de informação. Alguns protocolos, como RS-232 [5], incluem também regras a nível mecânico (características físicas dos conectores), a nível elétrico (características elétricas dos sinais) e a nível temporal (velocidade e sincronização).

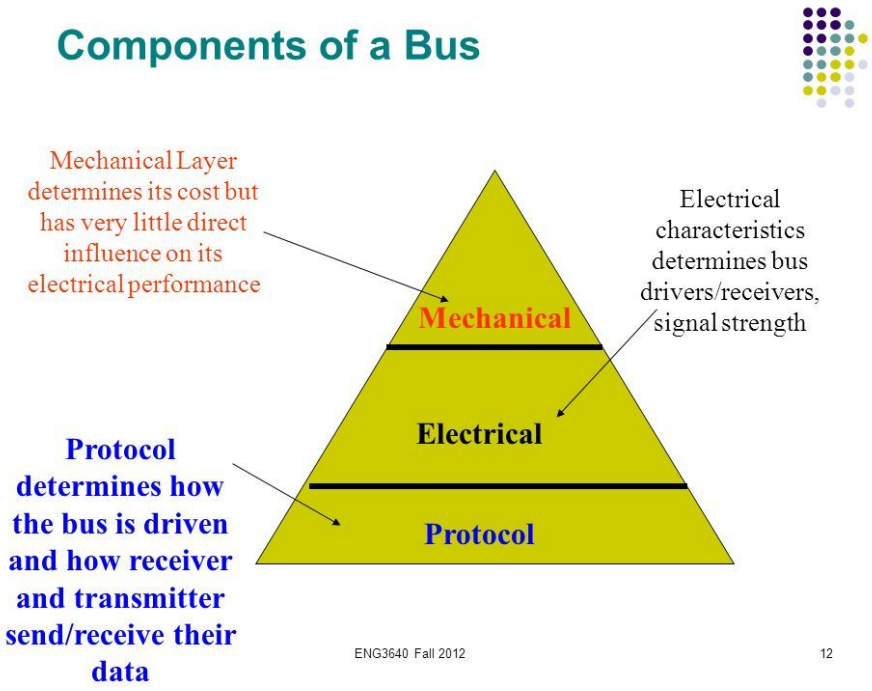



Figura 9.2: Componentes de um barramento (Fonte:[2]).

Muitas vezes os módulos a serem interligados são desenvolvidos por equipes distintas em empresas diferentes, o que impõe um rigor muito grande na caracterização de um barramento. Esta caracterização dos sinais de um novo barramento é feita normalmente através de comissões de representantes de indústrias em entidades como, por exemplo, o ANSI, o EIA (*Electronic Industries Association*), o IEC (*International Electrotechnical Commission*) e o IEEE. Estas comissões definem então um padrão de interligação (um barramento) que passa a ser obedecido pelos fabricantes. Como exemplo de barramentos que surgiram através de comissões de fabricantes temos o padrão de barramento VME (IEC Com. 47b >std. 821) e o padrão FUTURE BUS (projeto IEEE 896) [3].

No entanto, a maioria dos barramentos do mercado surgiram como barramentos próprios de equipamentos específicos [3]. A alta escala de aplicação transformaram-nos em padrões *de facto*. Como exemplo disto, temos o barramento S100, que depois de ser muito utilizado na indústria de microprocessadores de 8 *bits* passou por comissões do IEEE e se transformou no barramento padrão IEEE696. Outro exemplo é o barramento MULTIBUS lançado pela INTEL Co. que se transformou na norma IEEE 796. O exemplo mais conhecido é o barramento do IBM PC, o IOCHANNEL, que foi adotado por diversos fabricantes de placas e de sistemas. Recentemente ele foi padronizado por um conjunto de fabricantes como o padrão ISA (*Industry Standard Architecture*).

Para reduzir a quantidade de linhas num barramento e a quantidade de pinos nos circuitos integrados, é aplicada a **técnica de multiplexação**. Por exemplo, pode-se

multiplexar os sinais de dados com os sinais de endereços se eles tiverem o mesmo tamanho, e adicionar um *bit* de controle para distinguí-los. Ou então, multiplexar os sinais de endereços, dividindo um endereço em duas partes e enviá-las em dois instantes distintos num barramento de metade de tamanho.

Uma das principais evoluções do barramento é o compartilhamento de um conjunto de linhas/trilhas entre os componentes de um sistema digital para comunicação, reduzindo a quantidade de conexões e otimizando assim o uso do espaço físico de um *backplane* (circuito impresso sobre o qual são montados os componentes de um sistema digital). Por outro lado, podendo uma linha do barramento ser compartilhada por diferentes grupos de transmissor-receptores para transferência de dados, levou a novos problemas: **capacidade de acionamento** de um driver, **compatibilização dos sinais** de diferentes tecnologias, **contenção de barramento,** quando há mais de um componente alimentando um mesmo barramento com distintos sinais, e **arbitragem** do barramento, quando há mais de um componente requisitando o uso de um único barramento.

Neste capítulo vamos dar uma visão geral sobre diferentes aspectos sobre um barramento.

# 9.1 Tipos de Barramentos

O termo barramento cobre uma grande gama de *hardware* e *software* envolvidos com a comunicação dos dados entre os dispositivos. Podemos classificá-lo sob vários aspectos. Nesta seção apresentamos algumas formas mais usuais de classificação sob o ponto de vista de comunicação.

Pela largura do barramento, ou pela quantidade de *bits* transferidos em cada instante, distinguem-se barramentos seriais e barramentos paralelos. Quando um barramento reserva somente uma linha para transmissão de dados, dizemos que é um **barramento serial**. Se aumentarmos a quantidade de linhas/fios, aumenta-se a quantidade de *bits* que se pode transferir simultaneamente. Neste caso, o barramento é chamado de **barramento paralelo**.

Pela natureza e pela velocidade dos sinais que circulam em barramentos, os barramentos são classificados em:

- **barramentos internos**: são aqueles que interligam módulos de um sistema computacional dentro de uma mesma placa de circuito impresso (Figura 9.1). São normalmente próprios de cada fabricante do computador e são considerados no projeto os aspectos específicos do processador utilizado. Exemplos de barramentos internos são: (1) trilhas de interligação do microprocessador 8088 aos *chips* de memórias na placa principal do IBMPC;

(2) a conexão entre a CPU e a memória feita pelo **barramento frontal**, em inglês *front-side bus*; e (3) o **barramento por trás**, em inglês *back-side bus*, ou pelos barramentos PCI, SATA, IDE, PATA que conectam as memória de massa internas a um sistema computacional, como as interfaces que vimos na Seção 5.4.

- **barramento de E/S, externos ou de expansão**: são usados, principalmente, na ligação de computadores a seus periféricos. Podem ser aplicados ainda à interligação entre computadores. Exemplos de barramentos padrões de E/S são: (1) GPIB (IEEE 488) *General Purpose Intruments Bus u*sado para a ligação com instrumentos de medição e de acionamento em laboratório, e (2) SCSI *Small Computer System Interface*, comumente usado para a ligação com periféricos do tipo disco, como vimos na Seção 5.4.1.
- **barramento global**: é utilizado principalmente para ligação entre unidades do computador localizadas em placas diferentes. Normalmente, é uma extensão do barramento interno. Exemplos de padrões de barramento global adotados internacionalmente: S100 ou IEEE 696, MULTIBUS ou IEEE 796, e VME, IEEE 1014 ou IEC 821.

Neste capítulo vamos focar nos barramentos internos e externos. No capítulo 13, quando tratamos das redes de computadores, falaremos sobre barramentos globais.

# 9.2 Características Elétricas dos Barramentos

(resumo baseado do cap. 10 de [1])

Essencialmente, a função dos barramentos num sistema digital é transmitir sinais digitais entre os dispositivos digitais conectados. Embora, sob o ponto de vista funcional, os sistemas digitais operem com dois níveis lógicos (0 e 1), sabemos que estes dois níveis lógicos podem ser gerados pelas mais diversas tecnologias como ilustra Figura 9.3. Nesta figura temos um barramento conectando um transmissor, ou um **acionador de barramento**, de tecnologia NMOS e vários receptores de tecnologias NMOS, TTL, Schotky TTL e CMOS. Nesta seção vamos detalhar três aspectos elétricos relevantes no projeto de um sistema de barramento.

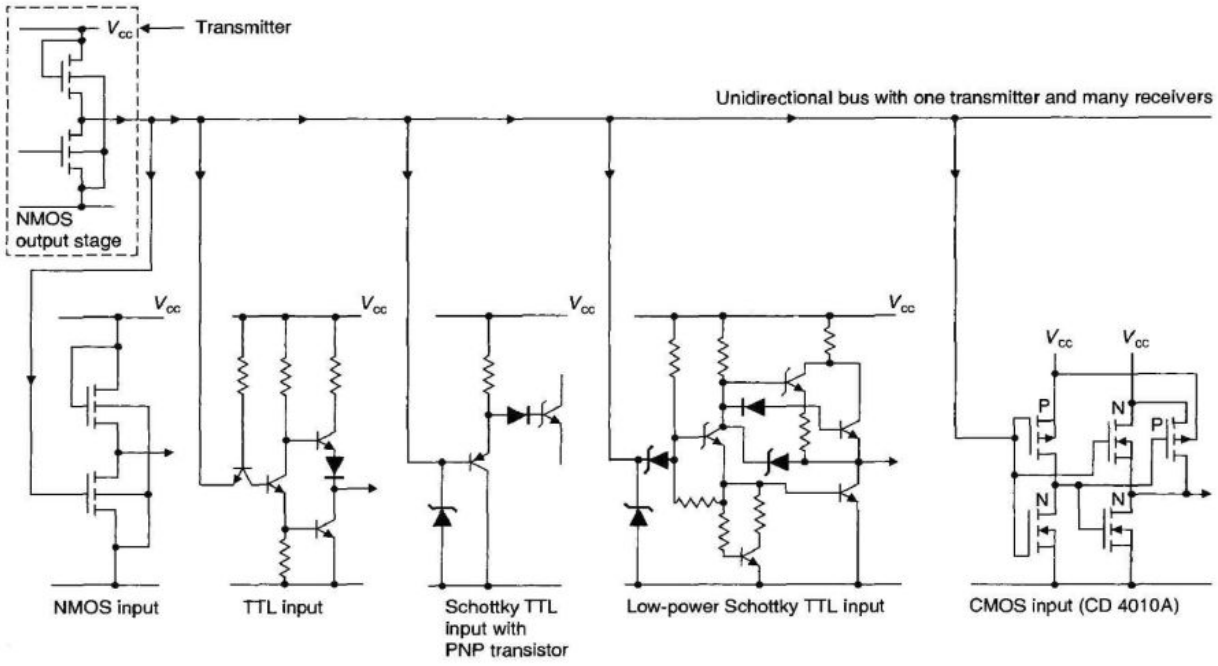



Figura 9.3: Conexão de diferentes tecnologias por um barramento (Fonte: [1]).

## 9.2.1 Receptores

A tabela na Figura 9.4 mostra que os componentes eletrônicos das tecnologias mais usadas apresentam distintas características elétricas.

| | *Logic Family* | | | | | |
|---|---|---|---|---|---|---|
| **Characteristic** | **LS TTL** | **S TTL** | **ALS TTL** | **NMOS** | **CMOS** | **Units** |
| $V_{OL}$ | 0.5 | 0.5 | 0.4 | 0.4 | 0.01 | V |
| $V_{OH}$ | 2.7 | 2.7 | 2.7 | 2.4 | 4.99 | V |
| $V_{IL}$ | 0.8 | 0.8 | 0.8 | 0.8 | 1.5 | V |
| $V_{IH}$ | 2.0 | 2.0 | 2.0 | 2.0 | 3.5 | V |
| $I_{OL}$ | 8 | 20 | 4 | 1.6 | 0.4 | mA |
| $I_{OH}$ | −400 | −1000 | −400 | −200 | −500 | μA |
| $I_{IL}$ | −0.4 | −2.0 | −0.4 | 2.5 μA | 10 pA | mA |
| $I_{IH}$ | 20 μA | 50 μA | 20 μA | 2.5 μA | 10 pA | mA |
| Propagation delay | 9.5 | 3 | 4 | 2.5 | 3.5 | ns |
| Input capacitance | 3.5 | — | — | 10–160 | 5 | pF |

Figura 9.4: Características elétricas dos componentes eletrônicos de diferentes tecnologias (Fonte: [1]).

É, portanto, fundamental elaborar uma estratégia para que os sinais gerados por um dispositivo sejam detectáveis pelos outros com o mesmo nível lógico original. Neste contexto, cabe introduzir o conceito de **imunidade ao ruído em DC**. Esta imunidade corresponde ao montante de ruído que um sistema pode tolerar tendo como as

entradas uma tensão sem exceder $V_{IL}$, correspondente ao nível lógico 0, ou uma tensão maior que $V_{IH}$, correspondente ao nível lógico 1:

**imunidade ao ruído do nível lógico alto** = $V_{OH} - V_{IH}$

**imunidade ao ruído do nível lógico baixo** = $V_{IL} - V_{OL}$

Na Figura 9.5 são mostrados não só os níveis de tensão correspondentes aos níveis lógicos dos sinais, mas também as correntes que circulam entre os componentes em decorrência dos diferenciais de tensão. Assegurar a compatibilidade elétrica entre os dispositivos é garantir que estas características elétricas satisfaçam as restrições impostas pelos fabricantes dos dispositivos.

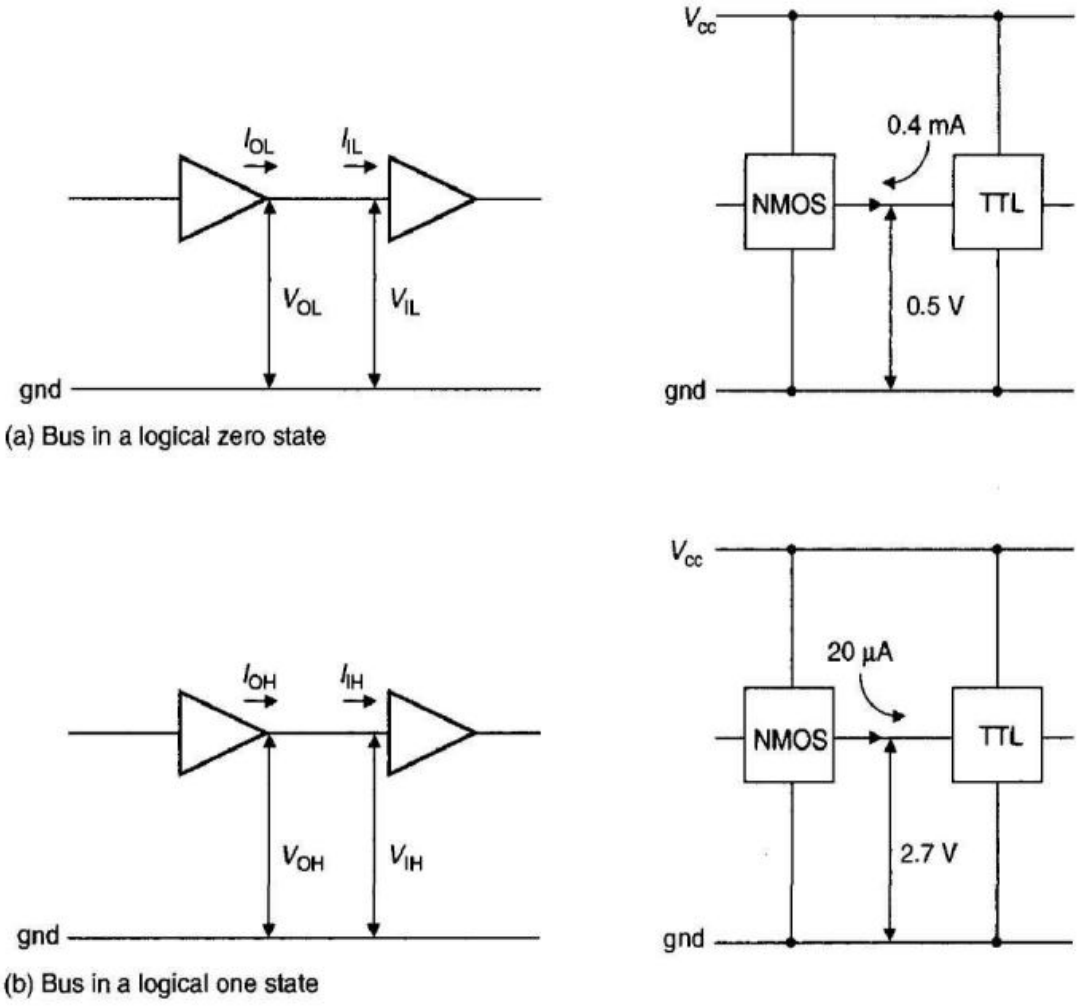



Figura 9.5: Níveis lógicos traduzidos em níveis de tensão e de corrente.

A Figura 9.6 mostra as imunidades ao ruído (no nível lógico 0/no nível lógico 1) entre diferentes combinações das tecnologias mostradas na Figura 9.4. Observe que todos os dispositivos CMOS acionados por dispositivos de outras tecnologias tem uma alta imunidade ao nível lógico 0 (≥ 1.0), mas uma imunidade negativa ao nível lógico 1 (≤ -0.8). Esta imunidade negativa mostra que as outras famílias de

tecnologia não conseguem acionar os dispositivos CMOS no nível lógico 1.

| Output Logic | Input Logic | | | | |
|---|---|---|---|---|---|
| | LS TTL | S TTL | ALS TTL | NMOS | CMOS |
| LS TTL | 0.3/0.7 | 0.3/0.7 | 0.3/0.7 | 0.3/0.7 | 1.0/−0.8 |
| S TTL | 0.3/0.7 | 0.3/0.7 | 0.3/0.7 | 0.3/0.7 | 1.0/−0.8 |
| ALS TTL | 0.4/0.7 | 0.4/0.7 | 0.4/0.7 | 0.4/0.7 | 1.1/−0.8 |
| NMOS | 0.4/0.4 | 0.4/0.4 | 0.4/0.4 | 0.4/0.4 | 1.1/−1.0 |
| CMOS | 0.79/2.99 | 0.79/2.99 | 0.79/2.99 | 0.79/2.99 | 1.45/1.45 |

Note: Each value is presented as "logical 0/logical 1" noise immunity. The effective value is the lower of this pair.

Figura 9.6: Imunidades aos ruídos nas conexões entre diferentes famílias de *chips*.

Um sistema de barramento bem projetado deve conseguir suprir estas diferenças de forma que todos os receptores num barramento interpretem o sinal elétrico transmitido como um mesmo estado lógico, como o circuito de interfaceamento entre um dispositivo TTL e CMOS usando um resistor *pull-up* (Seção 9.2.2.1) na Figura 9.7. Em [9] são mostradas outras soluções equivalentes.

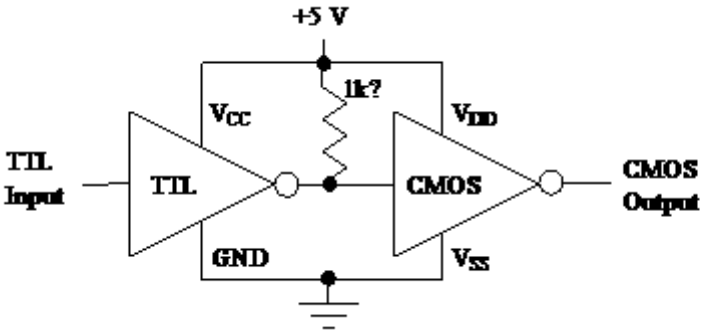
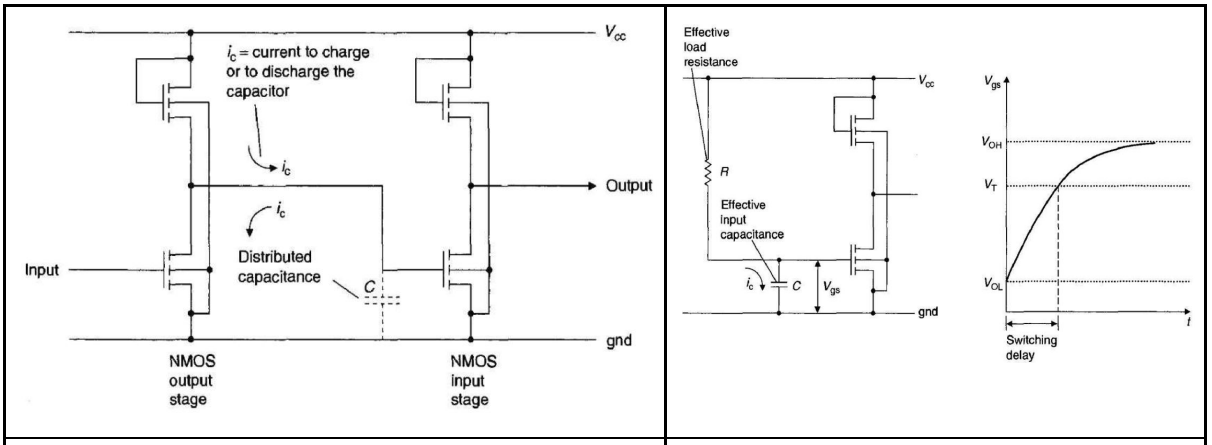



Figura 9.7: Uma solução de interfaceamento entre TTL e CMOS (Fonte: [9]).

As diferenças nas capacitâncias parasitas dos dispositivos eletrônicos podem ocasionar atrasos danosos nas transições (chaveamentos) dos sinais digitais transmitidas por um dispositivo, como ilustra a Figura 9.8.




(a) Capacitâncias parasitas | (b) Atrasos por carga

Figura 9.8: Influência de capacitâncias parasitas no chaveamento dos sinais (Fonte: [1]).

Para atenuar o efeito das capacitâncias parasitas inerentes a cada componente, é uma prática comum isolar os dispositivos eletrônicos do barramento através de **transmissores**/**acionadores**, em inglês *drivers*, e **receptores**, em inglês *receivers*, como ilustra a Figura 9.9. A grande vantagem destes circuitos de interface é que as características elétricas do barramento fiquem invariantes em relação às características de cada dispositivo conectado a ele, mesmo que eles introduzam atrasos nos sinais recebidos. Figura 9.8 mostra os diferentes níveis de atraso que os barramentos possam inserir num sinal: atraso no barramento global e atraso no barramento local.

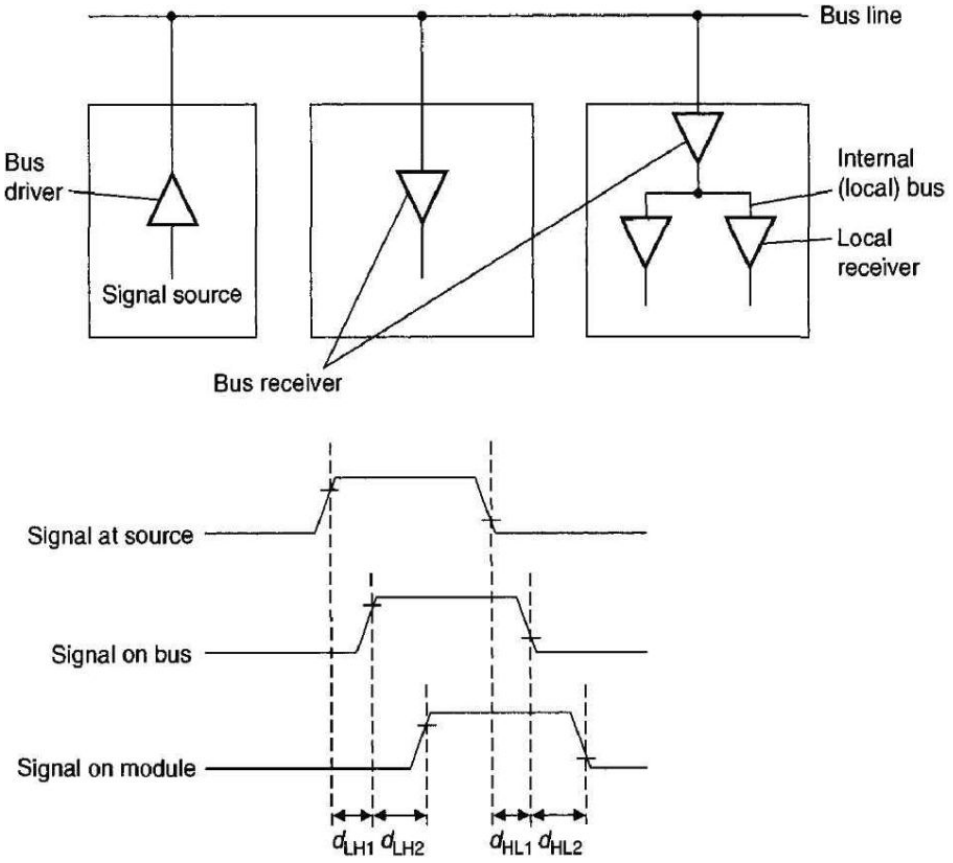



Figura 9.9: Desacoplamento dos dispositivos do barramento através de *drivers* e *receivers* (Fonte: [1]).

## 9.2.2 Transmissores/Acionadores

Para aumentar a versatilidade de um barramento, é interessante que mais de um dispositivo possa, ao invés de um único transmissor como na Figura 9.3, colocar sinais no barramento. Será que podemos simplesmente conectar dois estágios de saída, como os estágios de saída *totem-pole* (Seção 2.3.1) de dois dispositivos mostrados na Figura 9.10? Observe que, se G1 acionar o estado lógico 0 numa linha, drenando a corrente para o terra, e G2 acionar o estado lógico 1, puxando a corrente da fonte $V_{cc}$, simultaneamente, teremos um caminho de baixa impedância entre $V_{cc}$ e o terra, gerando um curto-circuito que pode queimar os componentes. Portanto, é preciso elaborar uma estratégia que evite a **contenção de barramento**, ou seja, o estado do barramento em que mais de um dispositivo ativo tenta colocar dados nele simultaneamente, como veremos na Seção 9.4, e que seja possível assegurar que os transmissores que não estejam habilitados para colocar os dados no barramento não precisam ser fisicamente desconectados.

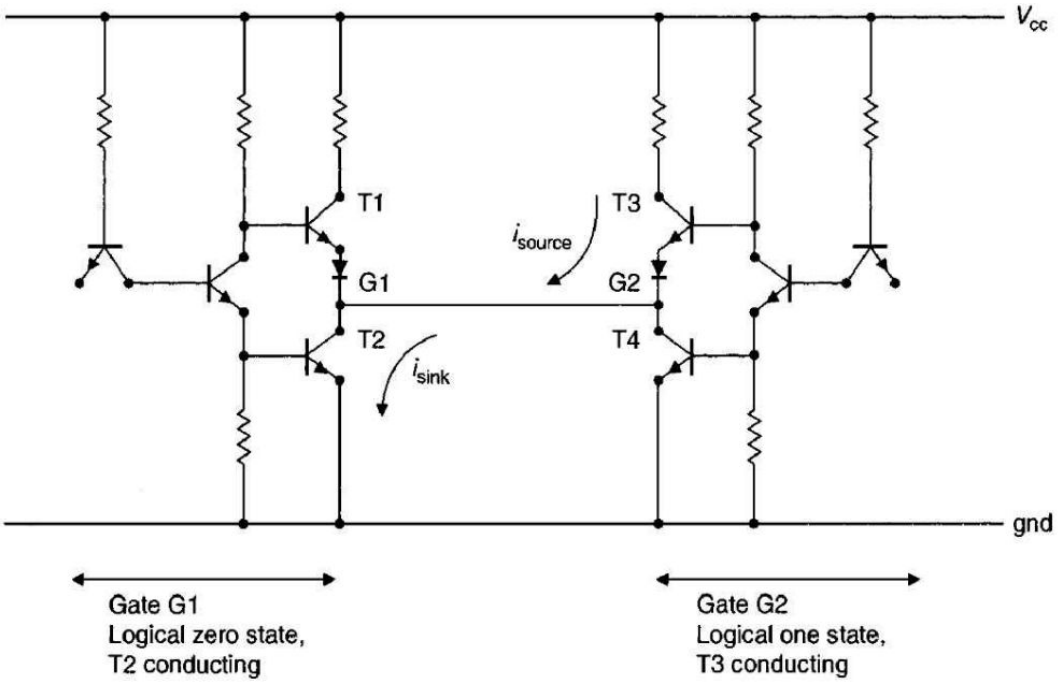



Figura 9.10: Conexão simples de dois estágios de saída (Fonte: [1]).

Duas lógicas de saída foram propostas para melhorar o controle de saída para um barramento: saída coletor aberto, em inglês *open collector*, e saída 3-estados, em inglês *tri-state*.

### 9.2.2.1 Saída Coletor Aberto

Uma saída coletor aberto é uma saída *totem-pole* sem o transistor conectado à fonte, como vimos na Seção 2.3.2. Para atingir o nível lógico 0, este tipo de saída funciona como no *totem-pole*, ou seja, o transistor ligado ao GND fica saturado e drena a corrente para o terra. Mas, para o nível lógico 1, este tipo de saída depende do nível de tensão de um circuito externo, pois o seu único transistor fica cortado e a saída fica em estado indefinido, em inglês ***floating state***.

Na Figura 9.11 o nível de tensão do nível lógico 1 é garantido por um **resistor *pull-up*** R que puxa a tensão do barramento para $V_{cc}$ de forma passiva. Sendo conectados os transmissores através de saídas coletor aberto a um barramento, estes transmissores só conseguem acionar o nível lógico 0. Quando nenhum dos transmissores estiver em operação, as linhas do barramento assumem o estado definido pelo resistor *pull-up* sem que os transmissores sejam fisicamente desconectados do barramento. O fato de que qualquer um dos transmissores consegue acionar o barramento para o estado 0, é comum dizer que os transmissores estão **conectados segundo a lógica OR**, em inglês *wired OR logic*.

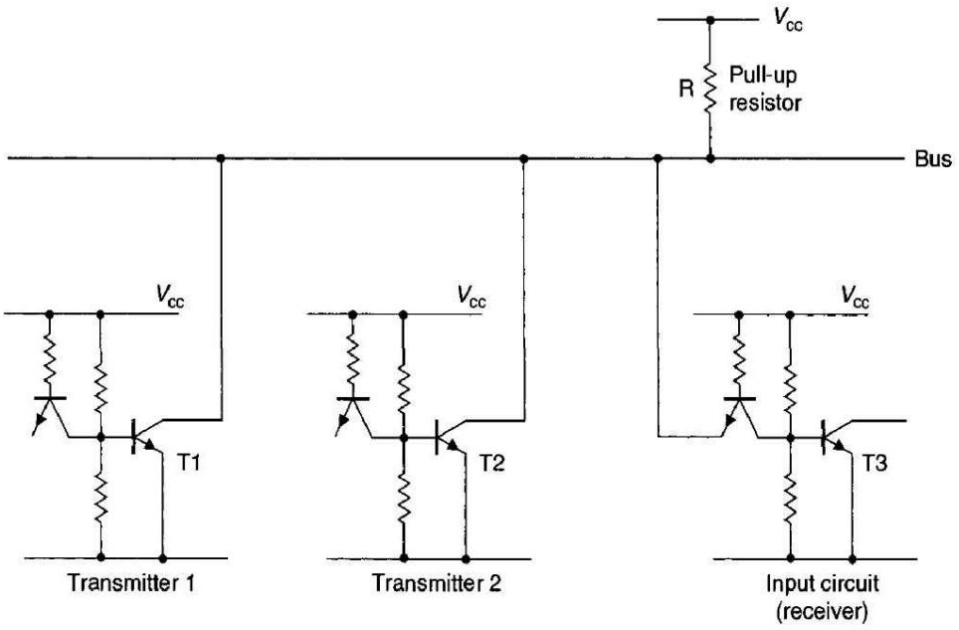



Figura 9.11: Acionamento pela saída coletor aberto (Fonte: [1]).

Como podemos determinar uma resistência apropriada para o resistor *pull-up* no projeto de um sistema de barramento? Veja na Figura 9.12(a) que são essencialmente dois tipos de correntes que fluem pelas portas dos dispositivos quando o barramento estiver no nível lógico 1 ($V_{OH}$). Pelos *n* transmissores desativos entram ínfimas correntes de fuga $I_{fuga}$ e pelos *m* receptores as correntes de entrada em nível alto $I_{IH}$. A resistência máxima do resistor *pull-up* pode ser aproximada por

$$R_{max} = \frac{V_{cc} - V_{OH}}{n \times I_{fuga} + m \times I_{IH}}$$

E a resistência mínima do resistor *pull-up* é requerida quando o barramento está no estado 0 ($V_{OL}$) com um dos transmissores acionando o barramento, drenando uma corrente $I_{OL}$ para o terra, enquanto as correntes de entrada dos receptores passam a ser $I_{IL}$, como mostra a Figura 9.12(b). Algebricamente, isso se traduz em

$$R_{min} = \frac{V_{cc} - V_{OL}}{I_{OL} + (n-1) \times I_{fuga} - m \times I_{IH}} \simeq \frac{V_{cc} - V_{OL}}{I_{OL} - m \times I_{IH}} .$$

Na prática, escolhe-se uma resistência que esteja no intervalo $[R_{min}, R_{max}]$.

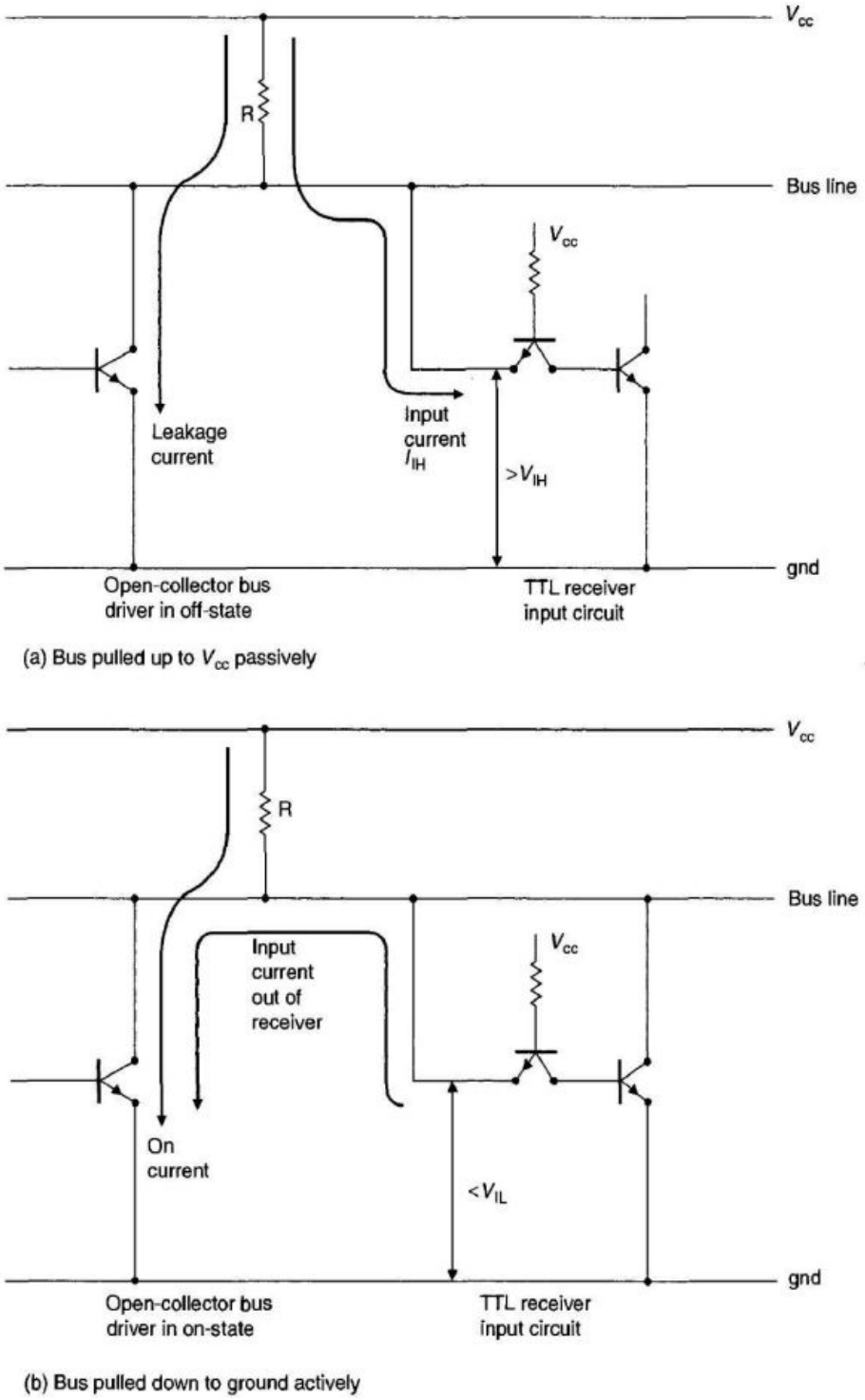



Figura 9.12: Acionamento passivo de barramento (Fonte: [1]).

### 9.2.2.2 Saída Três-Estados

Com a saída coletor aberto, o estado 1 é passivamente definido por um circuito externo. Os transmissores não conseguem acionar ativamente o barramento para o estado 1. Quando isso é necessário, usa-se a **saída três-estados**. Como o nome disse, esta saída pode assumir um dos três estados: 0 quando drena a corrente para o terra, 1 quando puxa a tensão para a de alimentação, e alta-impedância (Z)

suficiente para desconectar eletricamente um dispositivo do barramento. Observe na Figura 9.13 que esta saída tem um pino de controle que pode ser ativo alto ou ativo baixo. E a lógica do sinal de saída pode ser invertida ou não-invertida. Quando o sinal de controle habilita a saída, ela opera como uma saída *totem-pole*, acionando ativamente o estado 0 ou 1 no barramento. E quando o sinal de controle estiver desabilitado, a saída assume alta-impedância (**estado Z**).

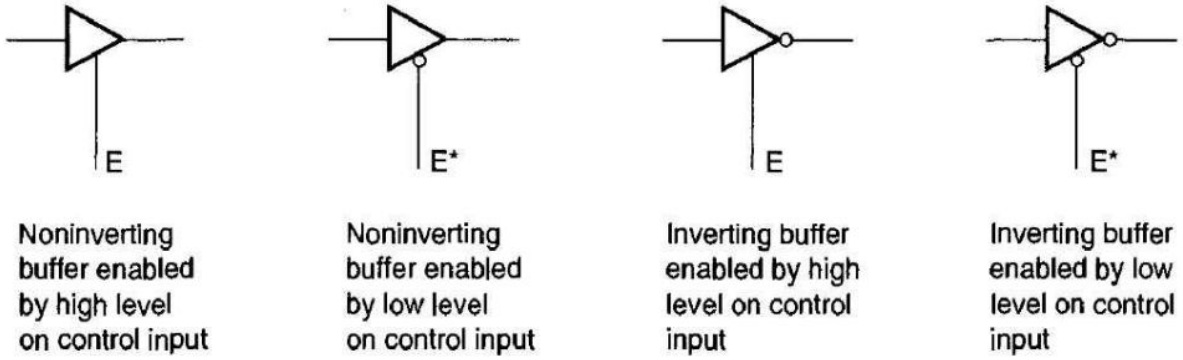



Figura 9.13: Símbolos gráficos de elementos de lógica três-estados (Fonte: [1]).

A saída três-estados é a configuração mais popular utilizada nos transmissores para controlar seus acessos ao barramento, sem que estes tenham que ser desconectados fisicamente quando estiverem inativos. Devido à grande demanda nos projetos baseados em microprocessadores e microcontroladores, há uma grande gama de ofertas de circuitos de interface com o barramento na forma de **acionadores**, em inglês *drivers*, de barramento e **transceptores**, em inglês *transceivers*. Os transceptores são circuitos com ambos o transmissor e o receptor integrados num mesmo *chip*. Eles são capazes de colocar sinais num barramento e de receber dados do barramento. Exemplos de módulos de acionamento são 74LS240 e 74LS244, e de módulos transceptores, 74LS242 e 74LS243.

Figura 9.14 ilustra a aplicação destes dispositivos nos projetos de sistemas computacionais. Códigos com quatro letras foram usados para descrever a função de cada elemento representado graficamente por um triângulo:

- Primeira letra: B = *buffer*[1].
- Segundo letra: A/D = endereços/dados
- Terceira letra: I/O = direção de dados (entrada/saída)
- Quarta letra: C/M = CPU/Memória (localização do *buffer*)

[1] O termo *buffer* tem vários sentidos em *hardware* e *software*. Em *software* ele significa tipicamente um espaço de memória onde os dados são armazenados temporariamente. E em hardware, entende-se como circuitos que interfaceiam dois sub-circuitos de um sistema.

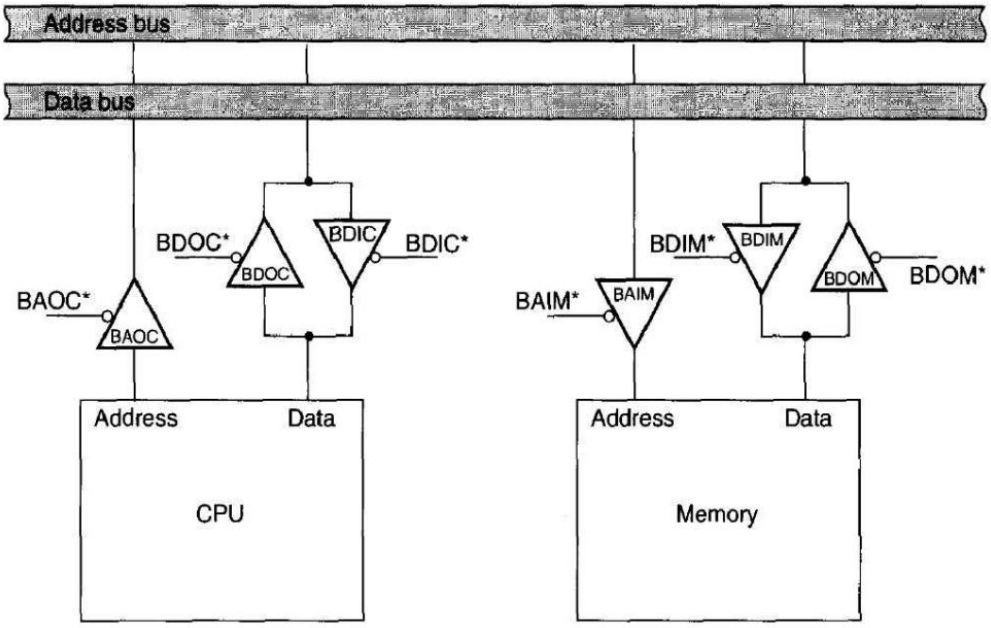



Figura 9.14: Barramento interfaceado pelas saídas três-estados (Fonte: [1]).

## 9.2.3 Transmissão

Mesmo evitando contenção de barramento e garantindo que as imunidades aos ruídos estejam dentro da faixa aceitável, os sinais detectados nos receptores podem estar altamente distorcidos. Nesta seção vamos entender o que pode acontecer com um fluxo de corrente ao longo de um barramento e como podemos evitar tais distorções.

Os barramentos podem ser modelados como linhas de transmissão em que se tem uma fonte (um transmissor) numa ponta, uma carga (receptor) na outra e um condutor (um barramento) conectando as duas pontas com um caminho de retorno pelo terra. No entanto, as capacitâncias, indutâncias e resistências parasitas oferecem inércia à propagação dos sinais ao longo de uma linha de transmissão. Para levar em conta essas características elétricas adicionais, existem dois modelos:

- **circuito passa-baixo** (Figura 9.15), com uma **frequência de corte**[2], em inglês *cutoff*, igual a $f_c = \frac{1}{2\pi RC}$ e o tempo de subida $t_r = 2.2RC$. A frequência de corte de uma placa de circuito impresso com trilhas curtas é tipicamente 1GHz com um tempo de subida menor que 1ns. Portanto, as distorções nos sinais podem ser consideradas desprezíveis.

[2] Frequência de corte é o ponto em que o filtro atenua o sinal para sua metade.

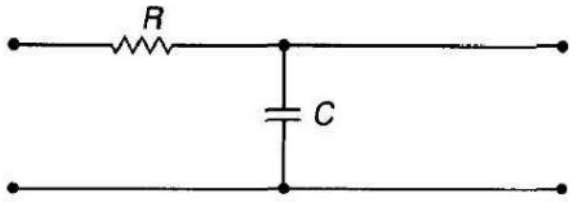



Figura 9.15: Circuito RC como modelo de barramento (Fonte: [1]).

- **modelo de linha de transmissão** (Figura 9.16), com um atraso de propagação de $\sqrt{LC}$ por unidade de comprimento. Tipicamente, aplica-se o modelo de linha de transmissão para análise de um sistema de barramento quando o tempo de subida é menor do que 2 vezes o tempo de propagação. Alguns recomendam o fator 4. O modelo divide uma longa linha em vários circuitos RLC como ilustra Figura 9.16(a). No entanto, como as resistências das trilhas de um circuito impresso são desprezíveis e as resistências às correntes de fuga entre as trilhas são muito grandes, é comum usar o circuito simplificado na Figura 9.16(b) para análise.

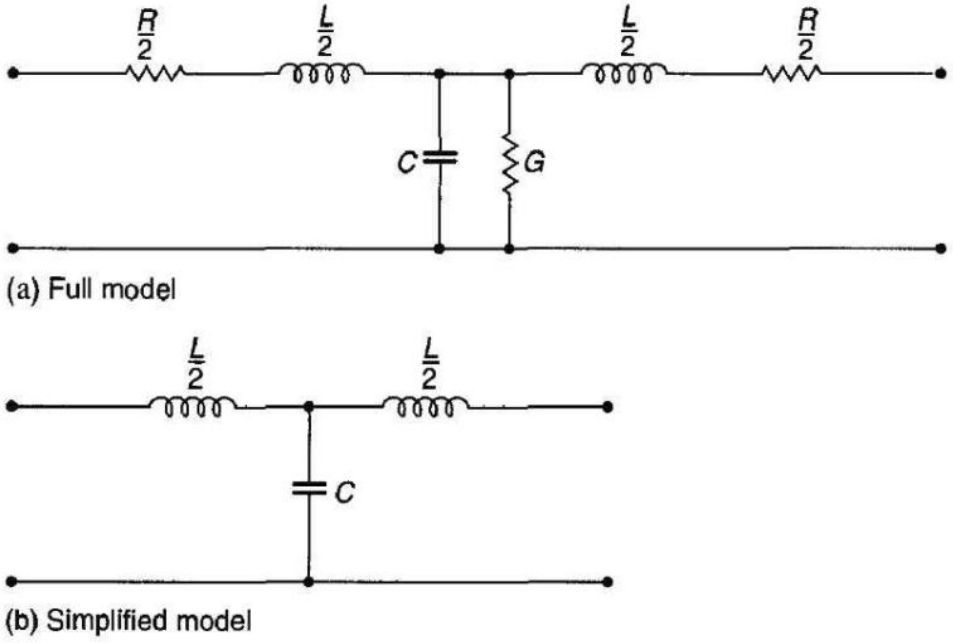



Figura 9.16: Modelo de linha de transmissão como modelo de barramento (Fonte: [1])

Quando um sinal chega ao final de um barramento terminado com a impedância característica[3] $Z_o$ do barramento, o sinal é absorvido na carga e a tensão em todas as linhas do barramento se mantém. Se a terminação for igual a $R_T$, uma parcela de tensão $V_i$ , $V_R$ , será refletida de volta para o barramento:

[3] Impedância característica, ou impedância súbita, de uma linha é a razão de uma tensão elétrica aplicada para a corrente elétrica resultante no ponto em que a tensão foi aplicada. A impedância é dada em ohms e é medida entre os terminais da linha de transmissão na freqüência de trabalho.

$$V_R = V_i \frac{R_T - Z_o}{R_T + Z_o}$$

Figura 9.17 ilustra o fenômeno de multi-reflexões de um pulso ao longo de um barramento com uma impedância característica Zo = 75 Ω, resistência de fonte $R_G$ = 30 Ω e uma resistência de terminação $R_2$ = 100 Ω.

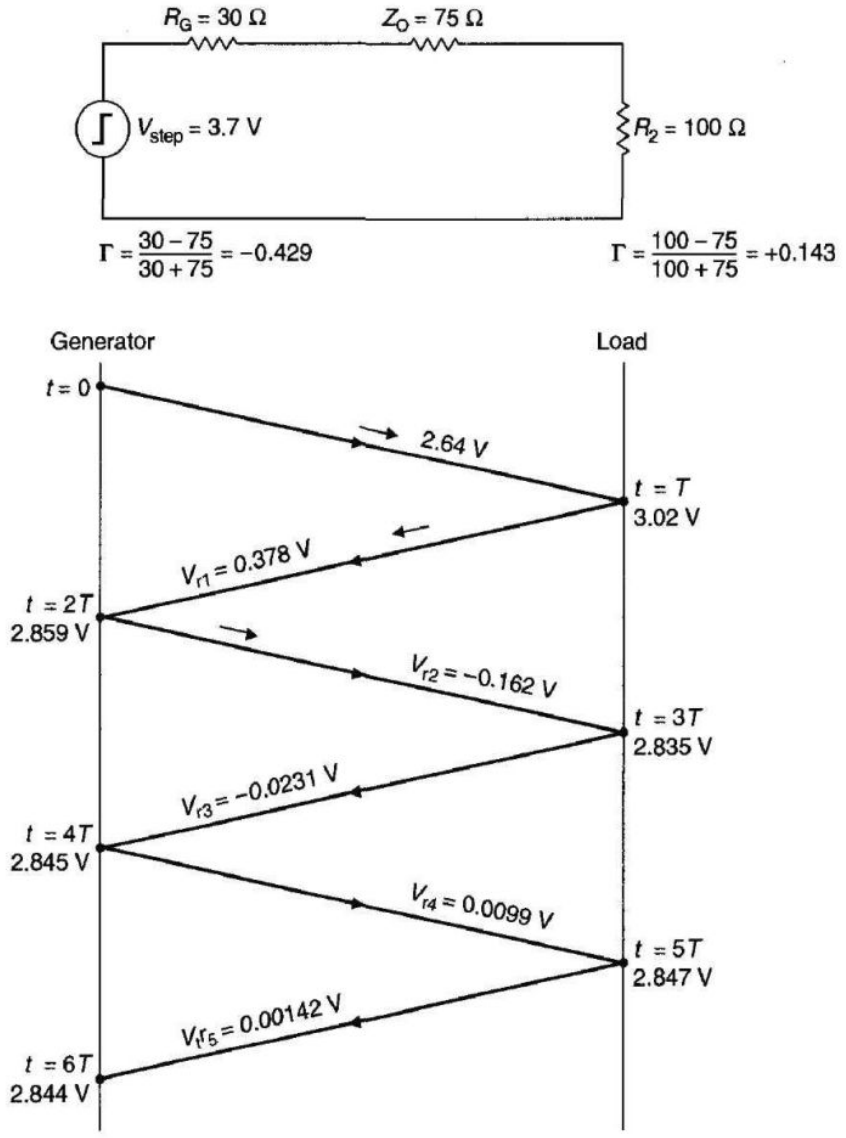



Figura 9.17: Múltiplas reflexões num barramento (Fonte: [1]).

Pelo que foi exposto, a terminação ideal para um barramento é a sua impedância característica. A impedância característica de uma placa de circuito impresso é tipicamente 100 Ω. Se conectarmos um resistor com esta resistência entre o barramento e o terra, ou entre o barramento e a fonte, como terminador da linha, curto-cirtuitaremos o barramento com o terra ou com a alimentação.

Uma solução clássica para terminação de um barramento é conectar um par de resistores ao barramento, um entre o barramento e o terra e outro entre o barramento e a fonte, como mostra a Figura 9.18. Resistores típicos são 330 Ω e

$470\,\Omega$. O circuito equivalente de Thévenin dessa terminação é apresentada também na Figura 9.18.

dois resistores

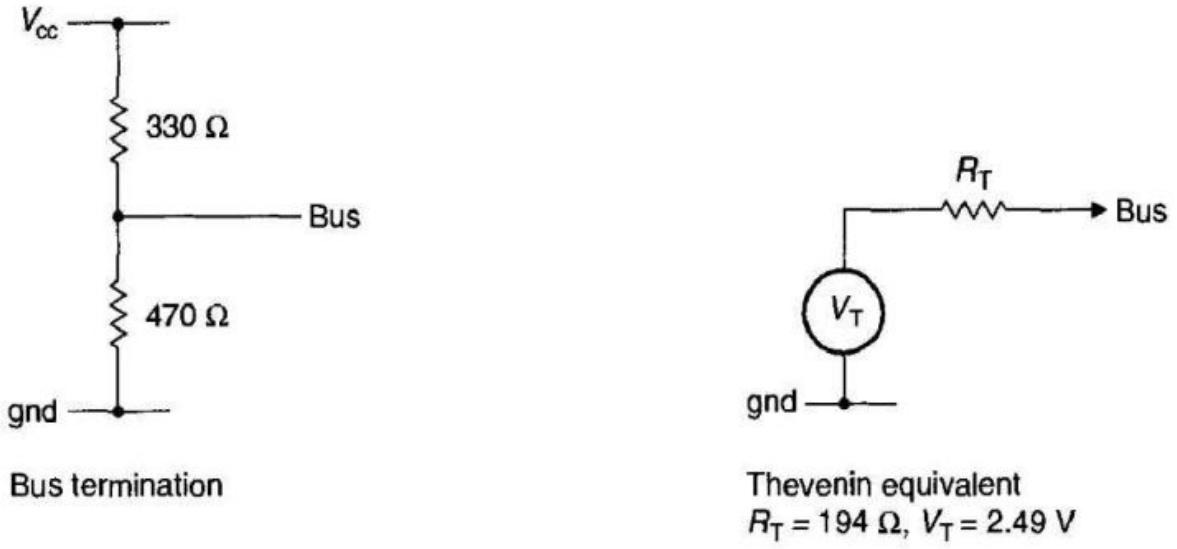
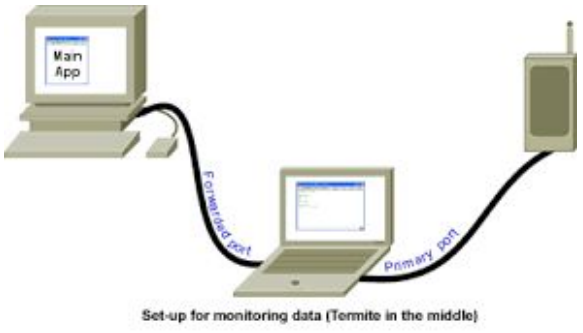



Figura 9.18: Terminação de um barramento (Fonte:[1]).

# 9.3 Topologias

A topologia é a disposição dos vários elementos (*links*, nós, etc.) de um sistema conectado por um conjunto de fios ou cabos. A ligação mais simples é a **ligação ponto a ponto**, ou **ligação dedicada**. Ela estabelece uma ligação única entre dois dispositivos. Pode ser feita de forma serial ou paralela. São exemplos práticos de ligação serial o padrão RS232 [5] entre um computador e um terminal (Figura 9.19) e de ligação paralela o padrão Centronics entre um microcomputador e uma impressora.




(a) Serial


(b) Paralela

Figura 9.19: Ligação ponto a ponto.

Quando um sistema de comunicação envolve mais que dois nós, o barramento é a forma de conexão mais simples. Ele tem a vantagem da simplicidade e extensibilidade. Todos os componentes estão conectados a um único meio. Para que um novo nó possa participar do sistema de comunicação, basta conectá-lo ao meio como na Figura 9.20.

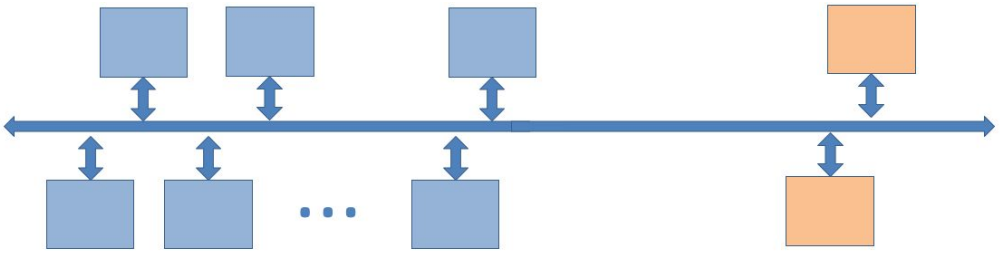


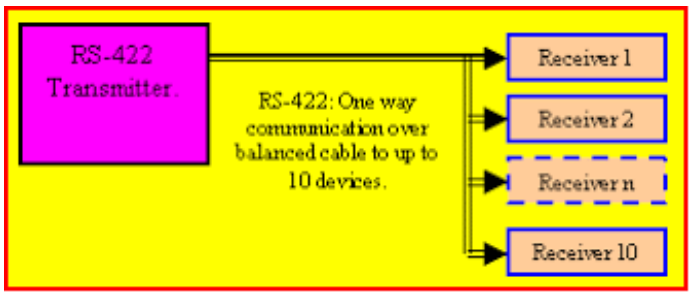
Figura 9.20: Topologia Barramento com expansão.

Nos barramentos modernos, destacam-se três topologias de conexão física: **ligação *multi-dropped***, **ligação *multi-point***, e **ligação em estrela**.

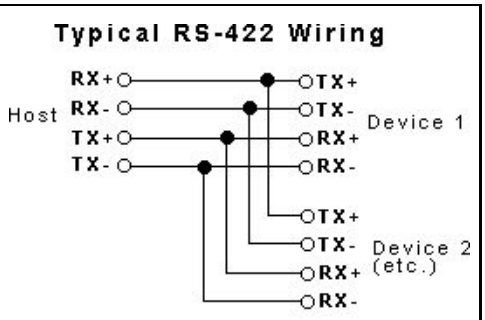
## 9.3.1 Ligação *Multi-dropped*

Na topologia "***Multi-dropped Bus***" (*MDB*), um dispositivo transmissor (mestre) comunica com vários receptores (escravos) unicamente identificáveis através de um barramento comum.

Para se comunicar com um receptor específico, o transmissor coloca no barramento o nome do receptor endereçado. Embora todos os receptores fiquem em escuta, somente o endereçado responde ao transmissor. Figura 9.22 mostra a interface do protocolo EIA-422 entre um transmissor e vários receptores.




(a) RS-422




(b) Conexão física

Figura 9.22: Topologia *multidrop* (Fonte: [6]).

## 9.3.2 Ligação *Multi-point*

Na topologia "***Multi-point Bus***", vários dispositivos comunicam bidirecionalmente com vários outros dispositivos, por exemplo TIA-485 (Figura 9.23). Usando um barramento *Multi-point* é necessário um processo de arbitragem para determinar qual dispositivo "ganha direito" de enviar informações num dado momento.

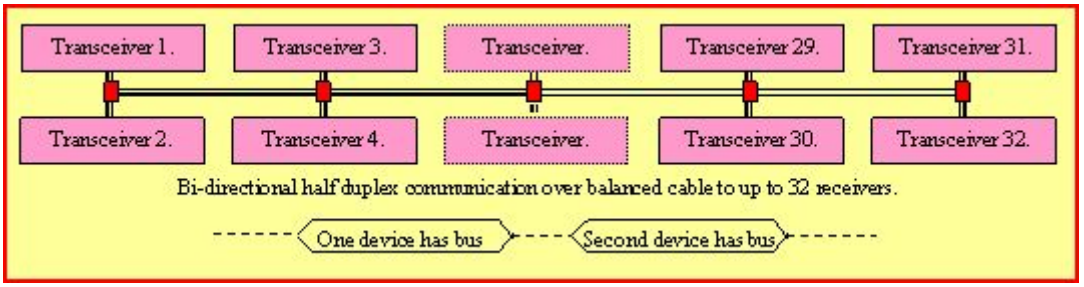



Figura 9.23: Topologia *multi-point* (Fonte: [6]).

## 9.3.3 Ligação em Estrela

Os periféricos mais novos, que são projetados para conectar-se a computadores pessoais, geralmente usam interfaces **USB (*Universal Serial Bus***), padronizadas por um consórcio de fabricantes. O USB 1.0 apareceu em 1996 e suporta uma taxa de dados de 12 Mbits/s. O USB 2.0 apareceu em 2000 e suporta taxas de dados de até 480 Mbits/s. O USB 3.0 apareceu em 2008 e suporta taxas de dados de até 4,8 Gbits/s. O USB é eletricamente mais simples que o RS-232 e usa conectores mais simples e robustos. Mas o padrão USB define muito mais do que o transporte elétrico de *bytes*, e é necessária uma lógica de controle mais complicada para suportá-lo. Os dispositivos periféricos modernos, como impressoras, unidades de disco e dispositivos de áudio e vídeo, incluem microcontroladores, facilitando o suporte ao protocolo USB mais complexo.

A padronização do USB é mantida pelo *USB Implementers Forum*, que conta atualmente (2018) com, aproximadamente, 1060 companhias (http://www.usb.org). O padrão define uma topologia física em estrela hierárquica, em que todos os nós da rede são conectados a um dispositivo central chamado o ***host root hub*** (Figura 9.24). Abaixo deste, podem ser conectados os **dispositivos USB**, que podem ser de dois tipos:

- ***Hubs*** fornecem capacidade adicional de conexão (*fan-out*)
- ***Functions*** fornecem a funcionalidade (por exemplo, armazenamento ou *mouse*)

Até 127 dispositivos podem estar ligados num barramento USB, mas somente um *host.*

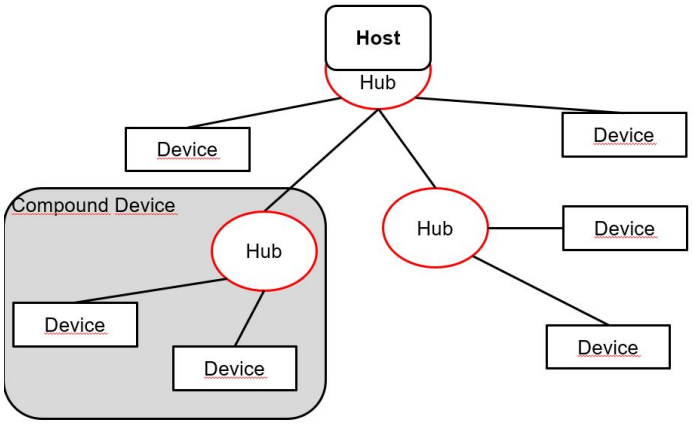



Figura 9.24: Topologia de conexão USB.

A conexão é feita por dois conectores, um macho e outro fêmea. O conector macho, aqui chamado de ***Plug***, normalmente fica no dispositivo, ou vem do dispositivo por cabo. O conector fêmea, aqui chamado de **Receptáculo**, normalmente fica no *host* ou nos *hubs*. O *plug* USB é ***hot-pluggable*** que significa que ele pode ser conectado com o *host* ligado. Quando um dispositivo é inserido no receptáculo, uma mudança na corrente elétrica é detectada pelo *host*. Isto dispara o processo de "*discovery*"[4].

# 9.4 Métodos de Arbitragem

Como já mencionamos, o uso compartilhado de linhas de conexão por uma grande variedade de dispositivos é uma das principais características de um sistema de barramentos. Para evitar disputas por um barramento, é preciso adotar alguma política de arbitragem. **Arbitragem de barramento** significa escolher entre dois ou mais dispositivos que solicitam, simultaneamente, o controle do barramento aquele que vai assumir o controle. Antes de apresentarmos as técnicas mais conhecidas de arbitragem, vamos introduzir os dois conceitos essenciais à arbitragem de barramento: **mestre** e **escravo**.

## 9.4.1 Mestre e Escravo

Um **mestre** é a unidade que controla a transferência num barramento, enquanto um **escravo** é a unidade que participa de uma transferência sob o controle de uma unidade mestre. Como exemplo podemos citar um microprocessador ligado a uma memória. Durante uma operação de leitura, as informações são retiradas da memória a partir de sinais de controle do microprocessador. A memória é uma unidade escravo e o microprocessador uma unidade mestre.

[4] *Discovery process* é o processo que o host de USB usa para detectar o dispositivo que foi conectado e a sua velocidade.

## 9.4.2 Mecanismo para a Mudança do Controlador do Barramento

Quando diversos dispositivos mestres compartilham um mesmo barramento, deve existir um mecanismo segundo o qual um dos dispositivos possa solicitar e obter o controle do barramento. O maior problema na implementação deste mecanismo é a política de escolha em solicitações simultâneas.

### 9.4.2.1 Árbitro Centralizado

Quando o *hardware* usado para passar o controle de um dispositivo para outro estiver concentrado num único local. Este local pode ser um dos dispositivos ligados ao barramento.

#### 9.4.2.1.1 Árbitro Centralizado por *Daisy Chaining*

Esta técnica pode ser introduzida com o auxílio da Figura 9.25. Nela é detalhada a ligação de *n* dispositivos mestres ao circuito de arbitragem de um barramento onde se destaca o circuito **árbitro do barramento**. Este circuito reúne a maior parte da lógica de arbitragem do barramento. O árbitro de barramento utiliza dois sinais de entrada:

- BUStBUSY que indica que o barramento está sendo ocupado por um dispositivo mestre; e
- BUStREQUEST que é usado pelos dispositivos para solicitar o controle do barramento.

O circuito árbitro gera o sinal BUStAVAILABLE para indicar a disponibilidade do barramento. Este sinal é gerado se o barramento não estiver sendo ocupado (BUStBUSY inativo) e houver uma solicitação do barramento (BUStREQUEST ativo).
Com respeito à propagação do sinal BUStAVAILABLE todos os dispositivos mestres formam uma cadeia, em inglês de *Daisy Chain*, de forma que este sinal só é propagado para o dispositivo seguinte se o anterior não estiver solicitando o barramento. Se um dispositivo receber o sinal BUStAVAILABLE e não estiver solicitando o barramento ele propaga este sinal para o próximo dispositivo. Se um dispositivo recebe o sinal BUStAVAILABLE e estiver solicitando o barramento ele impede a propagação do sinal BUStAVAILABLE e ocupa o barramento forçando o sinal BUStBUSY para o seu nível ativo. O dispositivo que ganhou o controle do barramento deve então retirar o seu sinal de BUStREQUEST. O sinal BUStBUSY mantém ativo durante toda a ocupação do barramento. Quando o sinal BUStBUSY é retirado, o sinal BUStAVAILABLE é desativado, se a linha BUStREQUEST for novamente ativada. E o procedimento se repete.

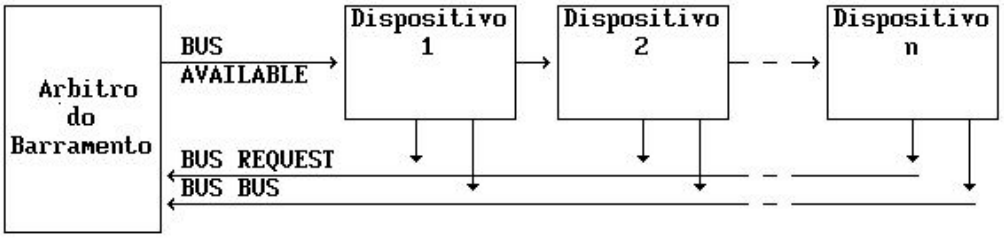



Figura 9.25: Árbitro Centralizado por *Daisy Chaining.*

A principal vantagem deste esquema de arbitragem é a sua simplicidade. Um número muito pequeno de sinais é necessário e é independente do número de dispositivos mestres. E a principal desvantagem é a sua susceptibilidade a falhas. Se um falha ocorrer numa lógica de propagação do sinal BUStAVAILABLE, este sinal pode não atingir os demais dispositivos da cadeia ou, em pior caso, dois dispositivos podem ganhar o controle do barramento.

Outra desvantagem é resultado direto da sua estrutura de prioridade fixada por posição. Se o dispositivo mais próximo fizer solicitações muito freqüentes, os dispositivos mais afastados "nunca" conseguirão controlar o barramento. Como o sinal BUStAVAILABLE deve ser propagado através de diversos dispositivos, o tempo de arbitragem é muito lento e varia de acordo com a configuração. Por último, um cuidado deve ser tomado ao se retirar módulos da cadeia, pois isto pode significar um rompimento da cadeia.

#### 9.4.2.1.2 Árbitro Centralizado por *Polling*

Figura 9.26 mostra a ligação de dois dispositivos mestre num esquema de árbitro centralizado por *Polling.* Neste esquema o dispositivo que requer o barramento força o sinal BUStREQUEST para o seu nível ativo. Recebendo este sinal o árbitro do barramento inicia uma contagem, em binário, nas linhas de *Poll Count.* Todos os dispositivos que estiverem solicitando o barramento comparam o valor nestas linhas com o seu código de *Poll.* Quando nestas linhas estiver um código que coincida com um dos dispositivos solicitantes, este dispositivo força a linha de BUStBUSY indicando que ocupou o barramento.

Este esquema evita os problemas de prioridade fixa do esquema *daisy-chain* (Seção 9.4.2.1.1) e permite um total controle centralizado do estado de ocupação do barramento. A forma de contar pode ser feita de diversas maneiras, cada uma estabelece um tipo de prioridade, por exemplo: contagem crescente e decrescente. Após o atendimento de uma solicitação o contador pode ser zerado ou pode continuar a partir de onde ele estava. Se ele for zerado, a prioridade é estabelecida de acordo com o código de *Poll* dos dispositivos, sendo o de código zero o mais

prioritário. Se a contagem continuar a partir do código do dispositivo atendido, é dada igual oportunidade de ocupação do barramento aos diversos dispositivos independente do seu código. Esta forma caracteriza o que se chama prioridade por ***round-robin***.

A principal desvantagem deste esquema é a limitação de número de dispositivos que podem ser adicionados ao barramento.

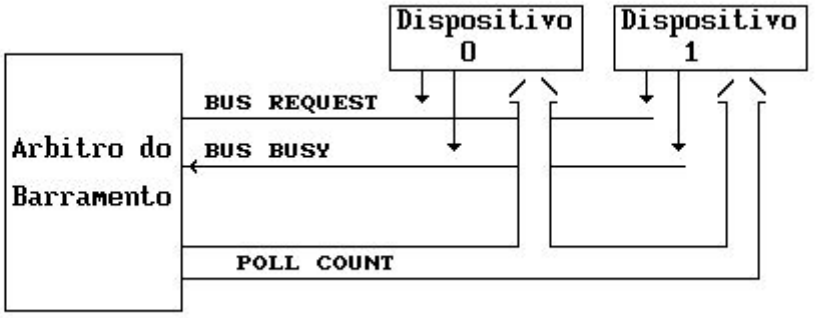



Figura 9.26: Centralizado por *Polling*

#### 9.4.2.1.3 Árbitro Centralizado por Requisições Independentes

Este esquema utiliza um par de linhas para cada dispositivo mestre do barramento, uma linha de solicitação e outra de reconhecimento, como mostra Figura 9.27. Quando um dispositivo precisar do barramento ele envia ao árbitro o seu sinal de BUStREQUEST, o árbitro seleciona o próximo dispositivo a ter a sua solicitação atendida e lhe envia o sinal de BUStGRANTED. O dispositivo atendido retira sua solicitação e ativa o sinal BUStASSIGNED para avisar aos demais dispositivos que o barramento está ocupado. Após completar a sua transferência o dispositivo desativa o sinal BUStASSIGNED e o árbitro remove o sinal BUStGRANTED e seleciona o próximo dispositivo que requer o barramento.

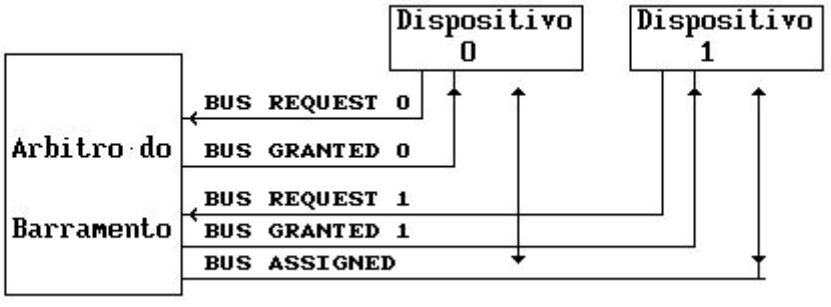



Figura 9.27: Árbitro Centralizado por Requisições Independentes.

Como todas as solicitações estarão presente em paralelo no árbitro do barramento, a resposta a uma solicitação é mais rápida que nos esquemas *daisy-chain* (Seção

9.4.2.1.1) e *polling* (Seção 9.4.2.1.1). No árbitro está localizada toda a política de alocação do barramento, podendo ser implementado um esquema de prioridade pré-especificada, prioridade adaptativa ou mesmo um esquema de *round-robin*.

A maior desvantagem deste tipo de arbitragem é o número limitado de dispositivos que podem ser mestres do barramento. Este número é função do número de pares de linhas de BUStREQUEST e BUStGRANTED. Outra desvantagem é a maior complexidade do circuito de árbitro.

### 9.4.2.2 Árbitro Distribuído.

Dá-se o nome de **árbitro distribuído** quando o *hardware* usado para passar o controle de um dispositivo para outro estiver altamente distribuído pelos dispositivos ligados ao barramento.

#### 9.4.2.2.1 Árbitro Distribuído por Daisy Chaining;

Um método de implementar o árbitro distribuído por *daisy-chaining* é mostrado na Figura 9.28. De acordo com esta topologia o estado do barramento é determinado por transições na linha de *Bus Available* que atravessa, em cadeia, todos os dispositivos do barramento. Quando uma transição ocorrer na linha de *Bus Available* que chega a um dispositivo que está solicitando o barramento, este impede a propagação da transição para os demais dispositivos na cadeia. Assim quando nenhum dispositivo estiver solicitando o barramento, um pulso se desloca na cadeia pelas linhas de *Bus Available*.

Assim, como no árbitro centralizado, o método de *Daisy Chaining* é dependente da posição do dispositivo na cadeia, e se um dispositivo na cadeia falhar, todo o barramento fica comprometido. Um problema que aparece com este método é a susceptibilidade a ruídos, por se tratar de *hardwares* sensíveis a transições de sinal.

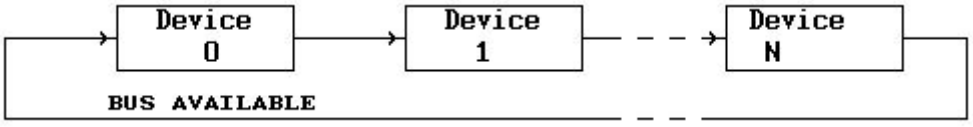



Figura 9.28: Árbitro distribuído por *daisy-chain*.

#### 9.4.2.2.2 Árbitro Distribuído por *Polling*

Este método é descrito com base na Figura 9.29. Um dispositivo que abandona o controle do barramento coloca nas linhas de *Polling Code* o código do próximo dispositivo a assumir o controle do barramento e ativa a linha de *Bus Available*. Podem ocorrer uma das duas situações;

- Primeira hipótese: Se o dispositivo eleito estiver precisando do barramento, ele ativa a linha *Bus Accept*. O primeiro dispositivo, então, retira o código das

linhas de *Polling Code* e desativa a linha de *Bus Available*. O dispositivo eleito desativa a linha de *Bus Accept* e começa a utilizar o barramento.

- Segunda hipótese: se o dispositivo eleito não estiver precisando do barramento, o primeiro dispositivo não recebe a ativação da linha de *Bus Accept* e modifica o código do próximo dispositivo a assumir o barramento de acordo com alguma política de alocação (prioridade ou *round-robin*) e espera de novo.

Este método requer que somente um dispositivo assuma o controle do barramento na iniciação do sistema. O *hardware* de alocação é o mesmo em todos os dispositivos e é igual àquele do árbitro centralizado por *polling*, encarecendo assim a opção por este método. Entretanto a confiabilidade do sistema é mantida se um dispositivo falhar.

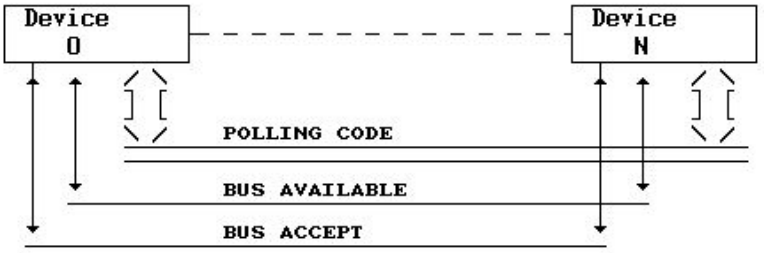



Figura 9.29: Árbitro Distribuído por *Polling*.

#### 9.4.2.2.3 Árbitro Distribuído por Requisições Independentes

Este método requer que cada dispositivo ligado ao barramento possua uma linha de *Bus Request* própria (Figura 9.30). O conjunto de todas as linhas de *Bus Request* é acessível por todos os dispositivos. Quando um dispositivo abandona o barramento, ele desativa a linha de *Bus Assigned*, todos os dispositivos que precisam do barramento analisam o estado das linhas de *Bus Request*. Aquele que se reconhecer como o de mais alta prioridade, assume o controle do barramento, acionando a linha de *Bus Assigned*.

A lógica de prioridade em cada dispositivo é mais simples que no mesmo método centralizado, entretanto o número de ligações e conectores é maior. A diferença de fase no sinal de relógio entre os diversos dispositivos (*clock skew*) limita este método a sistemas pequenos. Este método é muito susceptível a ruídos.

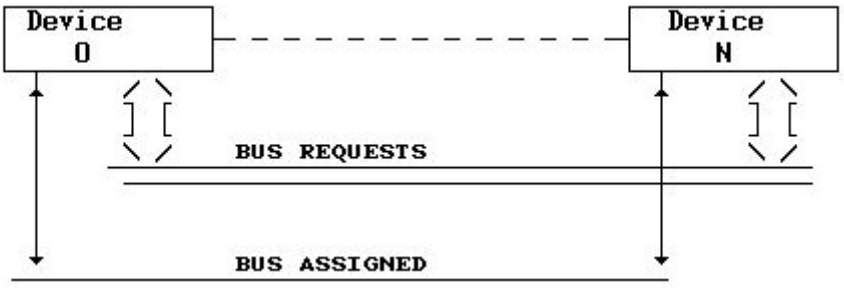


Figura 9.30: Árbitro Distribuído por Requisições Independentes.

# 9.5 Interfaces Paralelas

Nos capítulos 5 e 8 tivemos a oportunidade de analisar as conexões paralelas entre um microprocessador e os módulos de memória e periféricos. Considerávamos que as ligações eram ponto a ponto (Seção 9.3). Vamos ver nesta seção como podemos aplicar os *buffers* mostrados na Seção 9.2.3 para conectar estes dispositivos via barramentos.

Figura 9.32 mostra a conexão dos pinos de endereço de um microprocessador da família 68000 com os pinos de endereço dos módulos de memória através do módulo 74LS244 com 8 *buffers*. Os parâmetros de chaveamento do módulo são sintetizados na Figura 9.31. Eles são codificados em 3 letras:

- Primeira letra: P = propagação
- Segunda letra: L/H/Z = entrada (0/1/alta impedância)
- Terceira letra: L/H/Z = saída (0/1/alta impedância)

$V_{CC}$ = 5 V, $T_A$ = 25°C (see *SN54LS24x and SN74LS24x Devices*)

| PARAMETER | TEST CONDITIONS | | MIN | TYP | MAX | UNIT |
|---|---|---|---|---|---|---|
| $t_{PLH}$ | $R_L$ = 667 Ω, $C_L$ = 45 pF | SNx4LS240 | | 9 | 14 | ns |
| | | SNx4LS241, SNx4LS244 | | 12 | 18 | |
| $t_{PHL}$ | $R_L$ = 667 Ω, $C_L$ = 45 pF | | | 12 | 18 | ns |
| $t_{PZL}$ | $R_L$ = 667 Ω, $C_L$ = 45 pF | | | 20 | 30 | ns |
| $t_{PZH}$ | $R_L$ = 667 Ω, $C_L$ = 45 pF | | | 15 | 23 | ns |
| $t_{PLZ}$ | $R_L$ = 667 Ω, $C_L$ = 5 pF | | | 10 | 20 | ns |
| $t_{PHZ}$ | $R_L$ = 667 Ω, $C_L$ = 5 pF | | | 15 | 25 | ns |

Figura 9:31: Características de chaveamento de SNx4LS24x (Fonte: [7]).

Observe que foram simplesmente adicionados um *buffer* para cada linha tanto na saída da CPU (conexão de um acionador para o barramento de endereços) quanto na entrada dos módulos de memória (conexão do barramento de endereços para um receptor). Como o processador da família 68000 suporta 23 linhas de endereços e cada módulo SNx4LS24x contém 8 *buffers* integrados, 3 módulos foram necessários para a conexão do transmissor (CPU) e 3 para a conexão do receptor (Memória). Ao todo foram usados 6 módulos. Vale observar que, com esta nova conexão, precisamos incluir o tempo de propagação dos sinais nestes *buffers* (máximo 18ns pela Figura 9.31) na nossa análise temporal dos sinais.

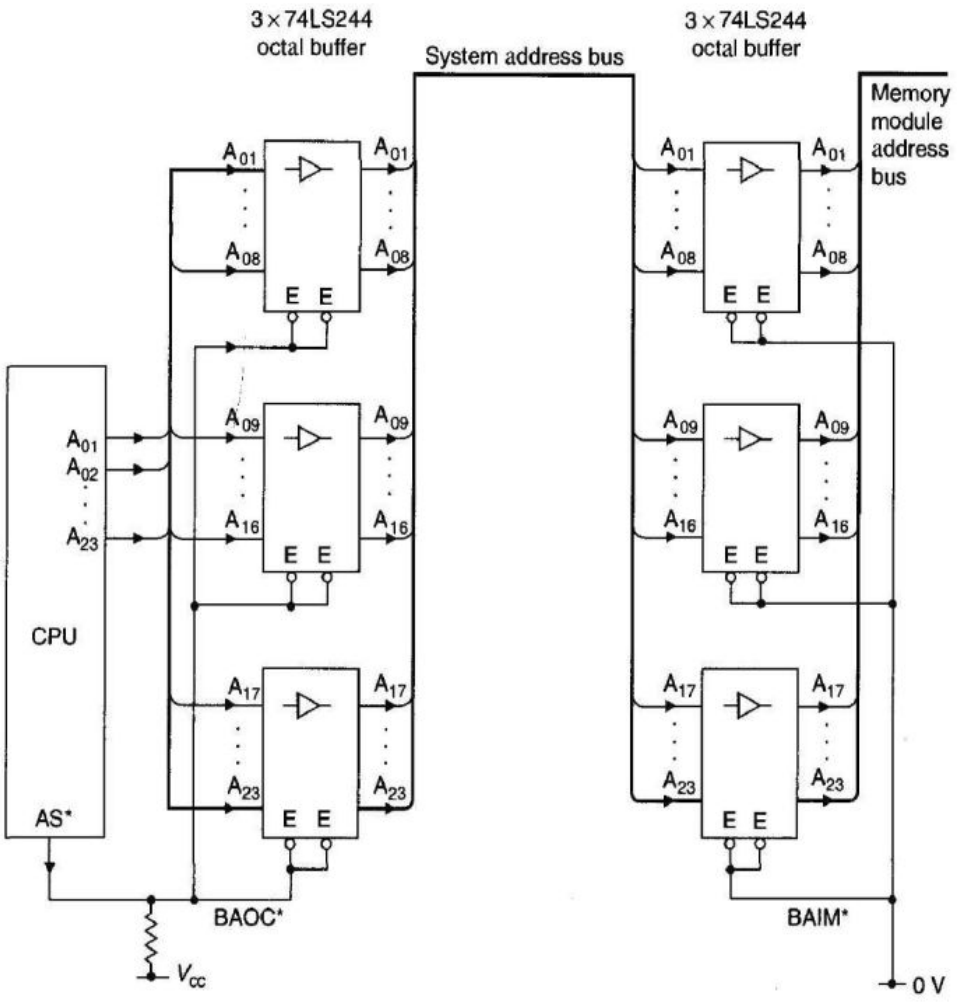



Figura 9.32: Barramento de Endereços com *buffers* (Fonte: [1]).

Na Figura 9.34 apresentamos uma nova versão para as conexões entre os pinos de dados de 68000 e os pinos de dados dos módulos de memória mostradas na Figura 5.50. Nesta nova versão os pinos são conectados, de fato, via um barramento de dados. Note que agora os pinos de dados de 68000 são conectados com o barramento através dos módulos 74LS245 com 8 *buffers* integrados, cujos parâmetros de chaveamento são resumidos na Figura 9.33.

**AC CHARACTERISTICS** ($T_A$ = 25°C, $V_{CC}$ = 5.0 V, $T_{RISE}/T_{FALL} \leq$ 6.0 ns)

| Symbol | Parameter | Limits | | | Unit | Test Conditions |
|---|---|---|---|---|---|---|
| | | Min | Typ | Max | | |
| $t_{PLH}$<br>$t_{PHL}$ | Propagation Delay, Data to Output | | 8.0<br>8.0 | 12<br>12 | ns | $C_L$ = 45 pF,<br>$R_L$ = 667 Ω |
| $t_{PZH}$ | Output Enable Time to HIGH Level | | 25 | 40 | ns | |
| $t_{PZL}$ | Output Enable Time to LOW Level | | 27 | 40 | ns | |
| $t_{PLZ}$ | Output Disable Time from LOW Level | | 15 | 25 | ns | $C_L$ = 5.0 pF,<br>$R_L$ = 667 Ω |
| $t_{PHZ}$ | Output Disable Time from HIGH Level | | 15 | 25 | ns | |

Figura 9.33: Características de chaveamento de SNx4LS24x (Fonte: [8]).

Diferentes dos módulos 74LS244, cujos *buffers* são unidirecionais (transmissores ou receptores), os *buffers* dos módulos 74LS245 são bidirecionais (transceptores), porque o fluxo de dados é bidirecional. Note ainda que, sendo 16 pinos de dados e cada módulo contém 8 *buffers*, 4 módulos 74LS245 foram suficientes para as conexões com o barramento de dados. Atrasos introduzidos por estes buffers (Figura 9.33) devem ser considerados nas análises de compatibilidade temporal.

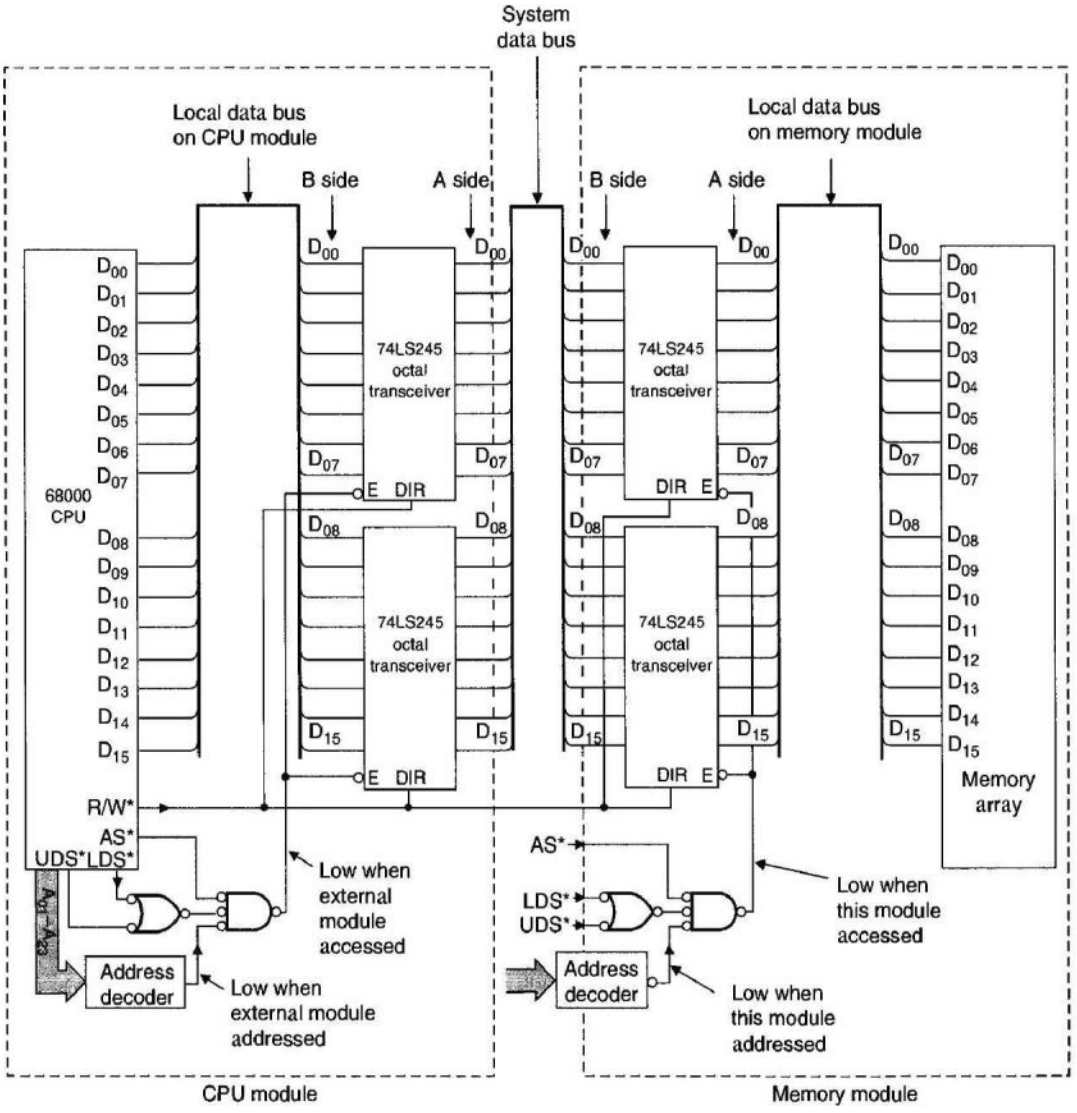



Figura 9.34: Barramento de dados com *buffers* (Fonte: [1]).

# 9.6 Interfaces Seriais

Uma das principais restrições enfrentadas pelos projetistas de processadores embarcados é a necessidade de ter pacotes fisicamente pequenos e baixo consumo de energia. Uma consequência é que o número de pinos no circuito integrado do processador é limitado. Assim, cada pino deve ser usado com eficiência. Além disso, ao conectar os subsistemas, o número de fios precisa ser limitado para manter a quantidade e o custo gerais do produto sob controle. Portanto, os fios também devem ser usados com eficiência. Uma maneira de usar pinos e fios de

maneira eficiente é enviar informações sobre eles em série, como sequências de *bits*, através de barramentos seriais. Os dispositivos são conectados a esses barramentos através de *interfaces* chamadas *seriais*. Vários padrões evoluíram para interfaces seriais, para que dispositivos de diferentes fabricantes possam (geralmente) ser conectados.

Na comunicação serial, os *n* bits da informação são serializados, ou seja, colocados um *bit* após o outro e transmitidos em sequência num único fio, como ilustra a Figura 9.35. Esta economia no número de fios é conveniente para a ligação entre sistemas colocados a longa distância. Com menos fios o número de circuitos de acionamento no transmissor e de filtragem e proteção no receptor é reduzido. Outro aspecto importante da comunicação serial é que ela pode ser feita utilizando-se das linhas e canais de comunicação já existentes (e.g. linhas telefônicas, canais radiofônicos, etc.).

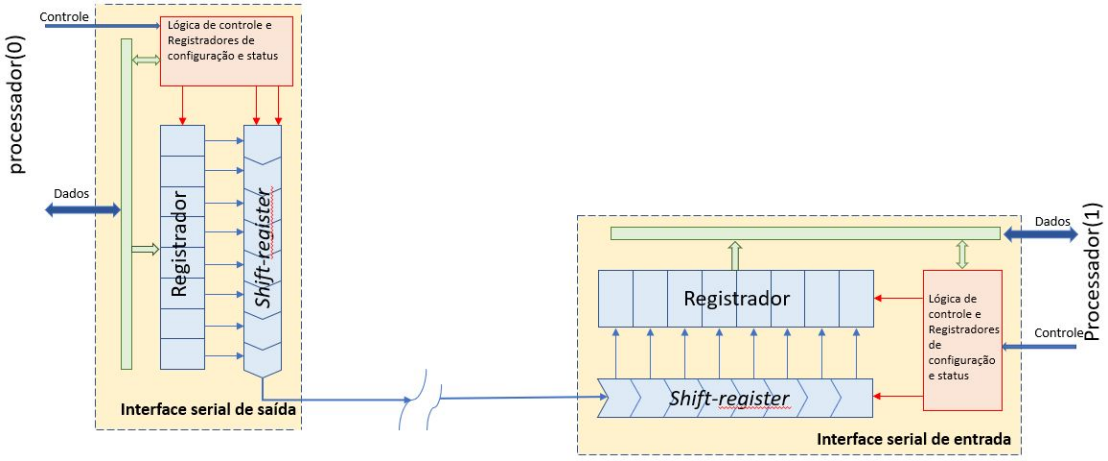



Figura 9.35: Princípio da comunicação serial.

Uma **interface serial de saída** é baseada num conversor paralelo-série (*shift-register*) que recebe os *bits* de dados em paralelo do barramento do computador, os serializa e os envia para o periférico através de um meio de comunicação. Uma **interface serial de entrada** é baseada num conversor série-paralelo que recebe do meio de comunicação os *bits* de dados em sequência até a chegada do último *bit* do dado, quando todos os *bits* são enviados em paralelo para o barramento do computador.

De uma forma geral, todos os circuitos integrados para controle de interfaces seriais possuem internamente as unidades de transmissão e recepção serial. Estes circuitos integrados são chamados de Receptor/Transmissor Assíncrono Universal (**UART**, do inglês: ***Universal Asynchronous Receiver/Transmitter***).

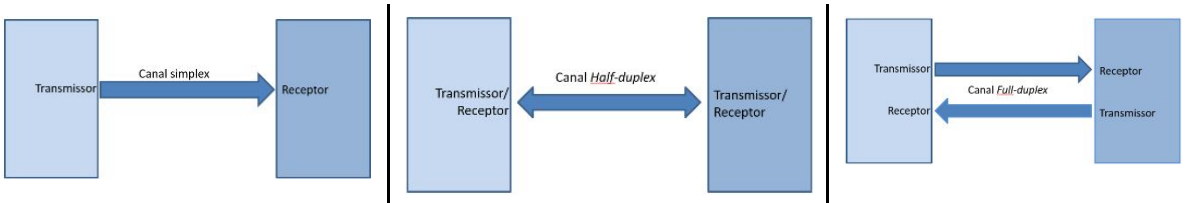
As vantagens econômicas da comunicação serial exigem um maior rigor na definição do protocolo de comunicação comparado com o protocolo da comunicação paralela. Numa ligação serial, para que a comunicação seja feita de forma correta, é necessário que tanto transmissor quanto o receptor estejam de acordo com os seguintes itens:

- **Padrão Elétrico de ligação.** Ou seja, é necessário que o transmissor e o receptor sejam eletricamente compatíveis, seja a nível de tensão ou de corrente;
- **Taxa de Transferência**. É necessário que as unidades de transmissão e recepção trabalhem na mesma base de tempo. De forma que cada *bit* transmitido seja recebido no instante correto;
- **Código da Informação.** É necessário que a codificação da informação seja a mesma nas duas unidades (transmissor e receptor).
- **Modo de Transmissão**. É necessário que cada unidade possa saber exatamente o instante da transferência da informação. Isto depende do modo de transmissão que pode ser síncrono ou assíncrono.

Antes de tratar dos principais tipos de padrões de comunicação serial é preciso introduzir alguns conceitos de comunicação que independem da forma de comunicação. São importantes os conceitos sobre os modos de comunicação, os modos de operação e sobre as topologias de conexão.

## 9.6.1 Modos de Comunicação

A comunicação entre dois sistemas pode ser feita de três modos básicos: *simplex, half-duplex, full-duplex*. Estes modos independem do meio, podendo ser por fio, por rádio frequência, etc.




*(a) simplex* *(b) half-duplex* *(c) full-duplex*

Figura 9.36: Modo de comunicação.

A comunicação *simplex* utiliza um único meio para enviar dados entre dois sistemas (Figura 9.36(a)). Esta é a forma mais simples de comunicação, existe somente um transmissor que envia as informações e um receptor que as recebe. O transmissor não recebe nenhuma informação do receptor sobre as condições dos dados que ele enviou. Justamente por isto, esta forma de comunicação é pouco utilizada em comunicação entre sistemas de computador.

Um sistema de comunicação *half-duplex* é capaz de transmitir em ambas direções (Figura 9.36(b)). Nesta forma de comunicação é utilizado o mesmo meio (por exemplo, um par de fios) para transmitir e receber informações. A principal característica desta forma de comunicação é que o meio só pode enviar informações numa única direção por vez. O tempo necessário para comutar o sentido da comunicação é chamado de *turnaround time*.

Um sistema de comunicação *full-duplex* é capaz de transmitir e receber informações simultaneamente (Figura 9.36(c)). Isto pode ser conseguido utilizando-se dois meios (por exemplo: dois pares de fios), um meio para transmitir e outro para receber. Por isto, é muitas vezes chamado de comunicação a quatro fios. A comunicação *full-duplex* pode ser feita usando somente um meio se o espectro de frequência for dividido em duas faixas não sobreposta de frequência, uma para a transmissão e outra para a recepção.

## 9.6.2 Modos de Operação

Quando a comunicação é feita sobre fios, o modo de operação define como é feita a transmissão dos sinais elétricos, podendo ser:

– Retorno por terra *(**Ground Return**)*, modo aqui tratado por interesse histórico*;*
– Sinalização por terminação única *(**Single-ended signaling**);*
– Sinalização Diferencial/Balanceada *(**Differential/Balanced Signaling**).*

### 9.6.2.1 Retorno por Terra (*Ground Return*)

O modo de operação retorno de aterramento de um único fio (***Single-wire earth return* - SWER**) é uma linha de transmissão de fio único. Hoje em dia, a principal aplicação é no fornecimento de energia elétrica monofásica de uma rede elétrica para áreas remotas a baixo custo. Sua principal característica é que **o solo** (ou às vezes um curso de água) é usado como o caminho de retorno para a corrente, para evitar a necessidade de um segundo fio (ou fio neutro) atuar como um caminho de retorno. No passado foi usado em comunicação, em linhas telegráficas de longo comprimento, mas hoje em dia não é mais! O interesse histórico serve para justificar a adoção do nome "*ground*" para o sinal de referência a outros sinais em circuitos e linhas de comunicação.

### 9.6.2.2 Sinalização por Terminação Única

Sinalização ***single-ended*** é o método mais simples e mais comumente usado para transmitir sinais elétricos através de fios. Um fio transporta uma tensão variável que representa o sinal, enquanto o outro fio é conectado a uma tensão de referência, geralmente chamado de terra (*ground*) (Figura 9.37). A sinalização de terminação

única não tem a capacidade de rejeitar o ruído causado por diferenças no nível de tensão de referência *(ground)* entre os circuitos de transmissão e recepção e por indução aplicada no fio de sinal (interferência eletromagnética).

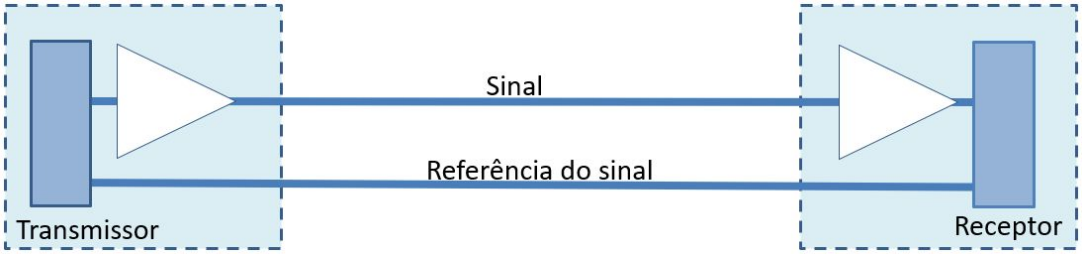



Figura 9.37: Modo de operação *Single Ended*.

### 9.6.2.3 *Differential/Balanced Signaling*

**Sinalização diferencial** é um método para transmitir informações eletronicamente usando dois sinais complementares. A técnica envia o mesmo sinal elétrico num um par diferencial de sinais, cada um em seu próprio condutor. O circuito receptor responde à diferença elétrica entre os dois sinais, ao invés da diferença entre um único fio e o terra. Os pares diferenciais são normalmente encontrados em cabos de par trançado e em conectores (Figura 9.38).

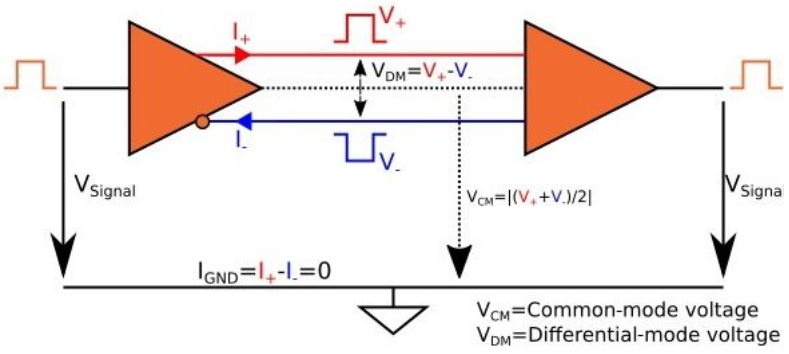



Figura 9.38: Sinalização Diferencial/Balanceada(fonte: allaboutcircuits,2019)

O receptor extrai informações detectando a diferença de potencial entre os sinais invertidos e não invertidos. Os dois sinais de tensão são balanceados, o que significa que eles têm amplitude igual e polaridade oposta em relação a uma tensão em modo comum. As correntes de retorno associadas a essas tensões também são equilibradas e, assim, cancelam-se; por esse motivo, podemos dizer que os sinais diferenciais têm (idealmente) zero corrente fluindo através da conexão de terra.

Com sinalização diferencial, o remetente e o receptor não compartilham necessariamente uma referência comum. No entanto, o uso da sinalização diferencial não significa que diferenças no potencial de aterramento entre remetente e receptor não afetem a operação do circuito.

Tabela 9.1 sintetiza os exemplos de aplicação de sinalização diferencial/balanceada e as respectivas taxas de transferência.

Tabela 9.1: Exemplos de aplicação de sinalização diferencial/balaceada

| | |
|---|---|
| **Serial ATA** | 1.5 Gbit/s |
| **Infiniband** | 2.5 Gbit/s |
| **PCI Express** | 2.5 Gbit/s |
| **Serial ATA Revision 2.0** | 2.4 Gbit/s |
| **Serial ATA Revision 3.0** | 6 Gbit/s |
| **PCI Express 2.0** | 5.0 Gbit/s por linha |
| **10 Gigabit Ethernet** | 10 Gbit/s |

#### 9.5.2.4.1 Comparação entre sinalização terminação única e sinalização diferencial/balanceada

A principal vantagem da sinalização por terminação única sobre sinalização diferencial é que menos fios são necessários para transmitir múltiplos sinais. Para *n* sinais, então existem *n* + *1* fios - um para cada sinal e um para a referência (*ground*). A sinalização diferencial usa pelo menos *2n* fios.

Uma desvantagem da sinalização por terminação única é que as correntes de retorno para todos os sinais usam o mesmo condutor (mesmo se os fios terra separados forem usados, os terras são inevitavelmente conectadas em cada extremidade), e isso às vezes pode causar interferência (*crosstalk*) entre os sinais.

#### 9.5.2.4.2 Benefícios da Sinalização Diferencial

Traduzido de: https://www.allaboutcircuits.com/technical-articles/the-why-and-how-of-differential-signaling/

Existem importantes benefícios da sinalização diferencial que podem mais do que compensar o aumento da contagem de condutores.

**Corrente sem retorno**

Como não tem (idealmente) nenhuma corrente de retorno, a referência de terra se torna menos importante. O potencial de terra pode até ser diferente no transmissor e no receptor ou se mover dentro de um certo intervalo aceitável. No entanto, você precisa ter cuidado porque a sinalização diferencial acoplada a CC (como USB, RS-485, CAN) geralmente requer um potencial de terra compartilhado para garantir que os sinais permaneçam dentro da tensão máxima e mínima permitida da interface em modo comum.

**Resistência à entrada EMI e *cross-talk***

Se *EMI* (interferência eletromagnética) ou *cross-talk* (isto é, *EMI* gerado por sinais próximos) é introduzido de fora dos condutores diferenciais, ele é adicionado igualmente ao sinal invertido e não invertido. O receptor responde à diferença de voltagem entre os dois sinais e não ao terra e, portanto, o circuito do receptor reduzirá bastante a amplitude da interferência ou *cross-talk*. É por isso que os sinais diferenciais são menos sensíveis a *EMI*, *cross-talk* ou qualquer outro ruído associado aos dois sinais do par diferencial.

**Redução de *EMI* de saída e *cross-talk***

Transições rápidas, como as bordas de subida e descida dos sinais digitais, podem gerar quantidades significativas de *EMI*. As sinalizações *single-ended* e diferenciais geram EMI, mas os dois sinais em um par diferencial criarão campos eletromagnéticos que são (idealmente) iguais em magnitude, mas opostos em polaridade. Isso, em conjunto com técnicas que mantêm a proximidade entre os dois condutores (como o uso de cabo de par trançado), garante que as emissões dos dois condutores se cancelem.

**Operação de Baixa Tensão**

Os sinais *single-ended* devem manter uma tensão relativamente alta para garantir uma relação sinal-ruído adequada (*SNR*). As tensões comuns da interface *single-ended* são 3,3 V e 5 V. Devido à sua maior imunidade ao ruído, os sinais diferenciais podem usar tensões mais baixas e ainda manter *SNR* adequado. Além disso, o *SNR* da sinalização diferencial é automaticamente aumentado por um fator de dois em relação a uma implementação de *single-ended* equivalente, porque a faixa dinâmica no receptor diferencial é duas vezes maior que a faixa dinâmica de cada sinal dentro do par diferencial.

A capacidade de transferir dados com sucesso usando tensões de sinal mais baixas traz alguns benefícios importantes:

- Tensões de alimentação mais baixas podem ser usadas.
- Transições de tensão menores
    - reduz *EMI* irradiado,
    - reduz o consumo de energia e
    - permite frequências operacionais mais altas.

**Estado alto ou baixo e tempo preciso**

Você já se perguntou como exatamente decidimos se um sinal está em um estado de lógica alta ou lógica baixa? Em sistemas *single-ended*, deve-se considerar a tensão da fonte de alimentação, as características de limiar dos circuitos do receptor, talvez o valor de uma tensão de referência. E é claro que existem

variações e tolerâncias, que trazem incerteza adicional à questão da lógica alta-ou-lógica-baixa.

Em sinais diferenciais, a determinação do estado lógico é mais direta. Se a tensão do sinal não invertido for maior que a tensão do sinal invertido, você terá uma lógica alta. Se a tensão não invertida for menor que a tensão invertida, você terá uma lógica baixa. E a transição entre os dois estados é o ponto no qual os sinais não invertidos e invertidos se cruzam - isto é, o ponto de cruzamento.

Essa é uma das razões pelas quais é importante combinar os comprimentos dos fios ou traços que transmitem sinais diferenciais: para obter a precisão máxima de tempo, você deseja que o ponto de cruzamento corresponda exatamente à transição lógica, mas quando os dois condutores do par não são iguais em comprimento, a diferença no atraso de propagação fará com que o ponto de cruzamento se desloque.

## 9.6.3 Modos de Transmissão

Segundo a técnica de temporização adotada, é possível classificar os modos de transmissão em: síncrono e assíncrono. Numa **transmissão síncrona**, todas as suas operações sincronizadas por um relógio central, enquanto que numa transmissão assíncrona cada operação possui o seu próprio sinal de indicação de operação.

Um barramento com comunicação síncrona requer menos fios, é mais simples de entender, implementar e testar. Entretanto eles são menos flexíveis que os barramentos assíncronos. Isto se deve ao fato que eles estão condicionados a uma taxa máxima de relógio, ou seja, estão amarrados a uma determinada tecnologia. Assim sendo, os barramentos síncronos não podem tirar vantagem dos ganhos em desempenho das novas tecnologias surgidas após a sua definição. A cada mudança de tecnologia, para se aumentar o desempenho do sistema, é necessário trocar todos os dispositivos da tecnologia anterior. Isto não acontece com os barramentos assíncronos, pois cada operação pode definir a sua própria temporização, de forma que podem conviver num mesmo barramento assíncrono dispositivos de tecnologias diferentes. Esta flexibilidade tem, como custo, uma maior complexidade do barramento.

À medida que os avanços tecnológicos aproximam dos limites físicos de velocidade dos dispositivos, a vantagem da flexibilidade das transmissões assíncronas perde um pouco a sua importância. Pois, numa mesma tecnologia, o desempenho das transmissões síncronas é maior que o desempenho das assíncronas.

### 9.6.3.1 Transmissão Síncrona

Uma transmissão síncrona é caracterizada pela existência de um sistema central de relógio (*clock*) que define intervalos de tempo (*time slots*) de mesmo tamanho para cada operação no barramento. Em geral, estas operações são do tipo: comunicação de dados entre um elemento mestre e outro escravo. Existem duas técnicas básicas para a alocação do intervalo de tempo para as operações, a saber: Intervalo Dedicado e Intervalo não dedicado.

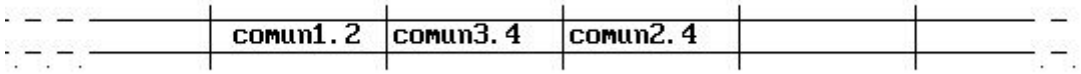



Figura 9.39: Comunicação Síncrona por Intervalo Dedicado.
comun1.2 Intervalo dedicado a comunicação entre os dispositivos 1 e 2
comun3.4 Intervalo dedicado a comunicação entre os dispositivos 3 e 4

Na técnica de Comunicação Síncrona por Intervalo Dedicado o tempo é alocado permanentemente a uma operação, mesmo que esta não seja realizada (Figura 9.39). De acordo com esta técnica os intervalos de tempo são definidos de forma a encobrir a operação mais lenta. Isto compromete o desempenho do sistema, pois as operações mais rápidas não poderão trabalhar a plena velocidade. Outra restrição ao desempenho global do sistema ocorre se as operações que têm intervalos dedicados não forem realizadas.

Na técnica Comunicação Síncrona com Intervalo Não Dedicado, os intervalos de tempo são alocados a uma operação somente se ela for realizada. Isto implica na necessidade de se estabelecer um mecanismo de alocação de intervalos em *hardware* para identificar se as operações vão ocorrer.

### 9.6.3.2 Transmissão Assíncrona

Na técnica de comunicação assíncrona, o instante da transferência não é conhecido *a priori*. Ele é indicado por sinais de controle próprios. O significado de cada um destes sinais e o relacionamento entre eles definem o **protocolo de comunicação**. O número dos sinais num barramento para este controle define o tipo de comunicação assíncrona. Podendo ser Comunicação Assíncrona Controlada por Um Fio (OWC *One Way Controlled*) ou Comunicação Assíncrona Controlada por Dois Fios (*Req/Ack*).

#### 9.6.3.2.1 Transmissão Assíncrona Controlada por Um Fio (*One Way controlled*)

A forma mais simples de comunicação assíncrona utiliza um único fio para controlar a comunicação entre dois dispositivos. Existem duas maneira de se controlar a comunicação com um único fio. Na primeira o circuito fornecedor de dados gera um

sinal de controle para indicar a existência de um dado válido para ser escrito num registrador receptor. Na segunda maneira, o circuito receptor de dados gera um sinal requisitando ao fornecedor o envio de dados.

#### 9.6.3.2.1.1 Transmissão Controlada pelo Transmissor de Dados

Na Figura 9.40 a presença do dado é indicado pelo fornecedor através do sinal DADO VÁLIDO. Enquanto este sinal estiver em nível ATIVO, o receptor pode ter certeza de que o dado está presente nas linhas de dado. Para que esta forma de comunicação funcione é necessário que haja uma rigorosa definição da temporização da comunicação. É importante que o receptor seja sensível à largura do pulso de DADO_VÁLIDO $t_w$ (*pulse width*). Se o sinal DADO_VÁLIDO for usado para gatilhar os dados no receptor, é importante que os dados estejam presentes um intervalo de tempo $t_{su}$ *(set up time*) antes da indicação de DADO_VÁLIDO. permaneça estável no barramento, no mínimo, durante um intervalo de tempo $t_h$ (*hold time*). O intervalo de tempo $t_R$ corresponde ao tempo necessário ao fornecedor dos dados para recarregar os seus registradores de saída de dados. O nome do sinal de controle neste tipo de comunicação pode variar de um barramento para outro, em inglês um nome muito utilizado para este tipo de sinal é ***strobe***.

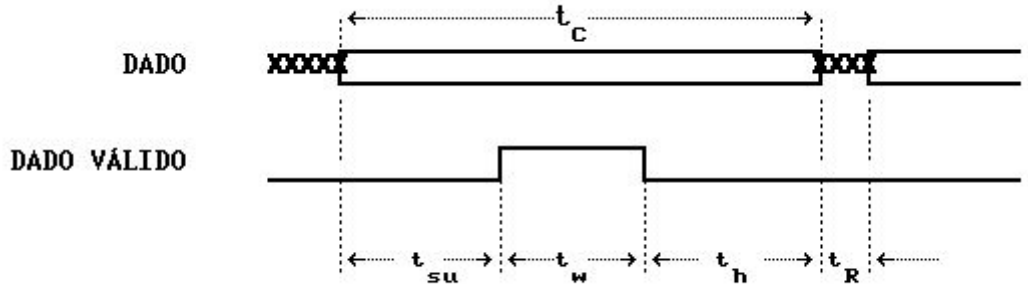



Figura 9.40: Comunicação assíncrona OWC controlada pela fonte dos dados.

As principais desvantagens desta técnica de comunicação são:

- ela é muito sensível a ruídos. O sinal de controle pode sofrer interferência em forma de pulso que pode ser interpretado pelo outro dispositivo como uma indicação de comunicação.
- nada assegura ao fornecedor que o dado que ele enviou foi recebido pelo receptor. É como um *tiro no escuro.*

As principais vantagens desta técnica de comunicação são:

- a simplicidade de implementação, que requer poucos componentes;
- as altas taxas de comunicação que ela permite.

De acordo com a Figura 9.40, um ciclo completo de transferência é dado por

$$t_c = t_{su} + t_w + t_h + t_R$$

O valor de cada um destes tempos dependerá da tecnologia adotada. Para minimizar o problema de interferência no sinal de DADO_VALIDO, $t_w$ pode ser aumentado, entretanto isto compromete a taxa de transferência.

Exemplos típicos de aplicação desta técnica:

- A conexão entre registradores internos às Unidades Processadoras é normalmente realizada por uma ligação barramento controlado por um único fio.
- Num módulo de memória, durante o ciclo de escrita, o barramento de dados comum a todos os *chips* e o sinal de habilitação de escrita tem o comportamento de um barramento controlador por um único fio. Considere como exemplo o ciclo de escrita definido para o *chip* de memória 2114.

#### 9.6.3.2.1.2 Transmissão Controlada pelo Receptor de Dados

Na Figura 9.38, quando um circuito receptor de dados precisar de novas informações, ele gera o sinal REQUER_DADO. O circuito transmissor de dados, recebendo este sinal, coloca os dados no barramento. Para que este tipo de comunicação funcione é necessário que os circuitos receptor e transmissor tenham um completo conhecimento da temporização do outro. Assim, por exemplo, somente após um intervalo de tempo igual a $t_1$ a partir da requisição o transmissor coloca os dados no barramento. Neste intervalo estão considerados o tempo de propagação pela linha REQUER_DADOS (entre o dispositivo receptor e o transmissor de dados) e o tempo necessário ao transmissor de dados preparar a transmissão. O receptor, então, define o instante no qual ele captura os dados do barramento. Este instante (o intervalo de tempo $t_2$ na Figura 9.41) deve considerar o tempo necessário para que os dados possam propagar entre o transmissor e o receptor e o $t_{su}$ do seu dispositivo de recepção . O dado fica no barramento até que uma nova requisição de dados seja fornecida pelo receptor, iniciando assim uma nova comunicação.

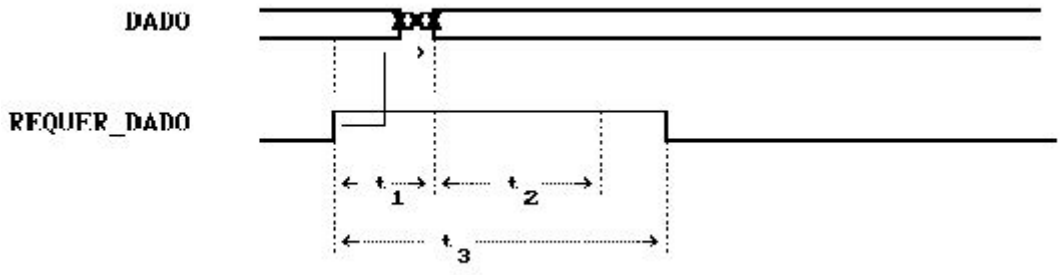



Figura 9.41: Comunicação assíncrona OWC controlada pelo receptor dos dados.

Nesta técnica as principais desvantagens são, como na técnica anterior, a suscetibilidade a ruídos e a incerteza da existência de dados no barramento. Nada

garante que, após ter sido sinalizado uma requisição de dados, $t_1$ segundos os dados estarão no barramento e o que estará sendo capturado depois de $t_2$ segundos é um dado válido. Para a suscetibilidade a ruídos a solução é restringir a largura do pulso do sinal REQUER_DADOS a um valor mínimo, comprometendo assim a taxa de transferência. Se a conexão entre os dispositivos for fisicamente assegurada, esta técnica pode ser utilizada sem incertezas da existência dos dados.

Outra desvantagem é com respeito à taxa de transferência que é normalmente menor que a técnica de controle pela fonte dos dados, o intervalo mínimo de transferência é, pelo menos, t mais longo que aquela técnica.

No ciclo de leitura numa unidade de memória, o barramento de dados tem o comportamento de um barramento controlado por um fio (sinal de habilitação de leitura). Vide, por exemplo, o ciclo de leitura de uma EPROM 27256.

#### 9.6.3.2.2 Transmissão Assíncrona Controlado por Dois Fios (Req/Ack)

Para a ligação entre unidades mais distantes, quando não é possível garantir a existência da conexão, ou ainda, quando não existir nenhum interconhecimento da temporização dos elementos envolvidos na comunicação, além do sinal gerado pelo transmissor indicando a presença dos dados, deve ser usado um sinal a mais, vindo do receptor, que indique a recepção. Esta técnica é chamada de comunicação controlada por dois fios. Como em muitos equipamentos, estes dois fios têm os nomes em inglês: *Request* para o sinal gerado pelo fornecedor e *Acknowledge* para o sinal gerado pelo receptor; esta técnica de comunicação é também conhecida como comunicação REQ/ACK.

Através destes dois fios é estabelecido um protocolo de comunicação (em inglês é também usado o termo *Hand Shaking*). O transmissor indica o envio do dado e o receptor indica a sua recepção. O protocolo mais utilizado em barramento estabelece um intertravamento (em inglês: *interlocking*), ou seja, uma interdependência entre os dois sinais de controle da comunicação.

Para entender melhor a importância deste intertravamento considere a Figura 9.42 onde é ilustrado uma comunicação REQ/ACK sem intertravamento. O transmissor põe o dado no barramento e ativa o sinal DADO_VÁLIDO; o receptor armazena o dado e responde com o sinal DADO_ACEITO. Isto leva ao transmissor a retirar o dado e, se for o caso, enviar um novo dado.

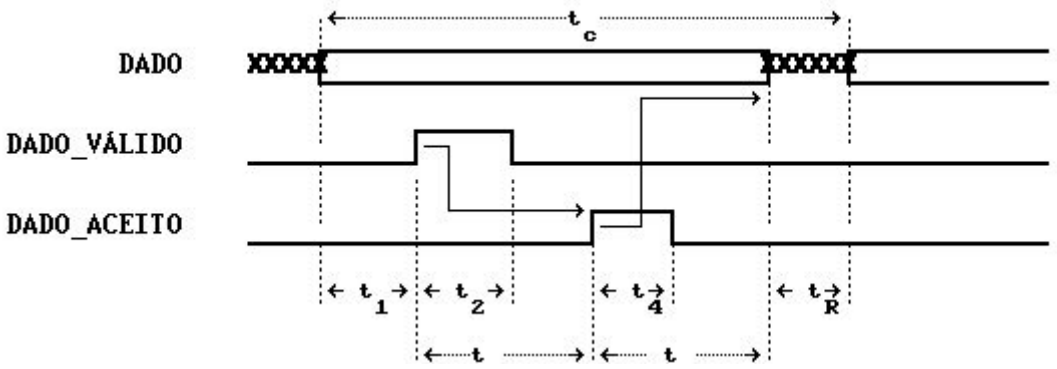



Figura 9.42: Comunicação assíncrona Req/Ack sem intertravamento.

A introdução, por esta técnica, de mais um sinal de controle não só minimiza a incerteza da comunicação, como também permite a operação entre dispositivos de velocidades diferentes. O custo disso é a diminuição da taxa de transferência e a necessidade de um circuito de controle mais complexo, pois devem ser considerados um número maior de atrasos. Outro problema é que com esta técnica passam a existir dois sinais susceptíveis a ruídos.

Para que esta técnica funcione corretamente é necessário que as temporizações sejam muito bem conhecidas, pois valores impróprios de tempo de propagação e de largura dos sinais de controle podem permitir que um outro sinal DADO_VÁLIDO pulse enquanto o sinal DADO_ACEITO ainda estiver ativo em resposta a uma comunicação anterior. Isto pode levar o barramento a ficar **Estrangulado**. Este efeito é melhor explicado com auxílio da Figura 9.43.

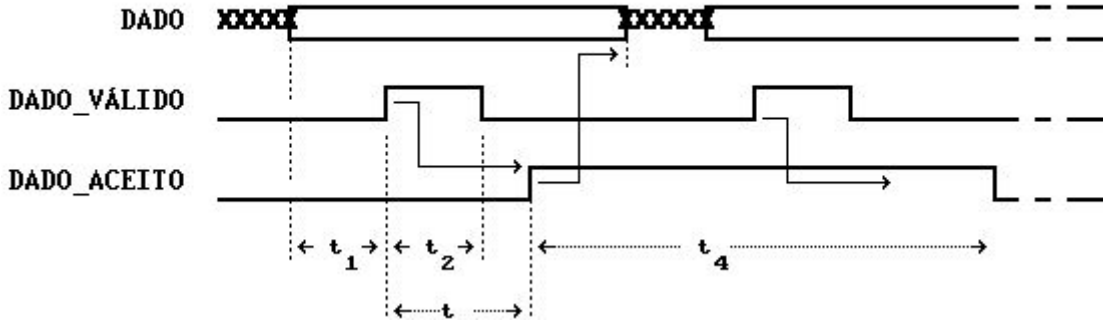



Figura 9.43: Comunicação assíncrona Req/Ack sem intertravamento com estrangulamento devido a erro de temporização

Na Figura 9.44 um transmissor rápido enviou os dados, pulsou o sinal DADO_VÁLIDO e esperou $t_3$ segundos até que o receptor lento respondesse com o sinal DADO_ACEITO. Ao receber este sinal, o transmissor encerra o ciclo de transferência atual e inicia, imediatamente, outro ciclo. Entretanto o vagaroso dispositivo receptor ainda não retirou o sinal de DADO_ACEITO. Como o circuito

receptor ainda está tratando da transferência anterior, ele não percebe o novo pulso do sinal DADO_VÁLIDO e não inicia a geração de um novo pulso de DADO_ACEITO. Assim o barramento fica **Estrangulado**, ou seja, o transmissor esperando o sinal DADO_ACEITO (sem retirar os dados do barramento) e o receptor esperando um novo pulso de DADO_VÁLIDO para iniciar a aceitação.

A eliminação do estrangulamento pode ser conseguida se o sinal DADO_VÁLIDO for mantido alto até a resposta pelo DADO_ACEITO, a retirada do sinal DADO_VÁLIDO provocar a retirada do sinal DADO_ACEITO e um novo DADO_VÁLIDO não puder ser gerado antes da borda de descida do DADO_ACEITO anterior. Os dois sinais de controle estarão, assim, intertravados, ou seja um depende fortemente do estado do outro.

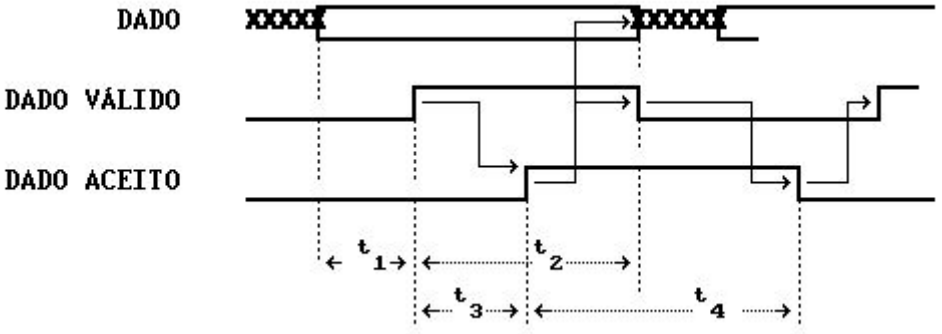



Figura 9.44: Comunicação assíncrona Req/Ack com Intertravamento.

# 9.7 Barramentos Internos

Funcionalmente, um barramento interno pode ser classificado de acordo com o tipo de informação que ele transporta.

- **barramento de dados**: possui, em geral, uma linha (fio) para cada *bit* de dados. Num microcomputador a largura do barramento de dados (*bus width*) é o fator de especificação de barramento mais lembrado. Diz-se por exemplo: "O microcomputador PC tem um barramento de 8 *bits*". Ou então: "O microprocessador 8086 tem um barramento de 16 *bits* e o microprocessador 80386 tem um barramento de 32 *bits*".
- **barramento de endereços**: possui uma linha (fio) para cada *bit* de endereço. A largura do barramento de endereços é importante para determinar a capacidade máxima de posições de memória acessíveis. A quantidade de posições de memória endereçáveis é igual a $2^n$, onde *n* é o número de *bits* do barramento de endereços, como mostra a seguinte tabela:

| *n bits* | $2^n$ palavras | |
|---|---|---|
| 16 | 65 536 | 64 K |
| 20 | 1 048 576 | 1 Mega |

| | | |
|---|---|---|
| 24 | 16 777 216 | 16 Mega |
| 32 | 4 294 967 296 | 4 Giga |

- **barramento de controle**: agrupa todos os sinais necessários ao controle da transferência de informação entre as unidades do sistema.

Vimos na Seção 4.7 que o *chipset*, normalmente encontrado na placa-mãe, é o responsável pelo fluxo de dados entre o processador, memória e periféricos. Embora esses *chipsets* sejam projetados para trabalhar com uma família específica de microprocessadores, eles suportam diferentes padrões de barramento para facilitar a conexão do processador com outros dispositivos. Vimos também que uma divisão tradicional de um *chipset* é em ponte norte, em inglês northbridge, e ponte sul, em inglês southbridge. A **ponte norte** faz a comunicação da CPU com as memórias e periféricos de alta velocidade, enquanto a **ponte sul,** também conhecido como ***hub* de controladores de entrada e saída**, em inglês I/O Controller Hub, estabelece a comunicação de um barramento interno com os periféricos mais lentos (portas paralelas, seriais e USB). O barramento paralelo PCI (*Peripherial Component Interconnect*) do padrão *PCI Local Bus* [11], era a interface entre a ponte norte e a ponte sul. Foi depois substituído pelos barramentos PCI Express, *Peripheral Component Interconnect Express* (PCIe) [11].

**PCI Express**, ou **PCIe**, é um barramento ponto a ponto, onde cada periférico possui um canal exclusivo e bidirecional de comunicação serial com o *chipset*. Está disponível em segmentos de x1 a x32. Estes números têm a ver com o número de vias paralelas, em inglês ***lanes***, utilizadas para a transmissão de dados. Quanto mais vias, maior é a taxa de transmissão e maior é o conector, como mostra a Figura 9.45. A taxa de transferência de PCIe 3.0 x16 é 16.000MB/s, enquanto de x4 é 4 vezes menor (4.000MB/s). A flexibilidade em quantidade de vias permitiu otimizar o compromisso entre o custo e a função do barramento. Para dispositivos que não precisa transferir grande volume de dados como uma placa de rede, PCIe 1x é suficiente. Entre as placas de vídeo *offboard*, o barramento PCIe 16x é o mais comum. Barramentos PCIe foram muito usados na conexão da placa-mãe com as placas de vídeo, os discos rígidos, os SSDs, e *hardware* de *Ethernet* e Wi-Fi.

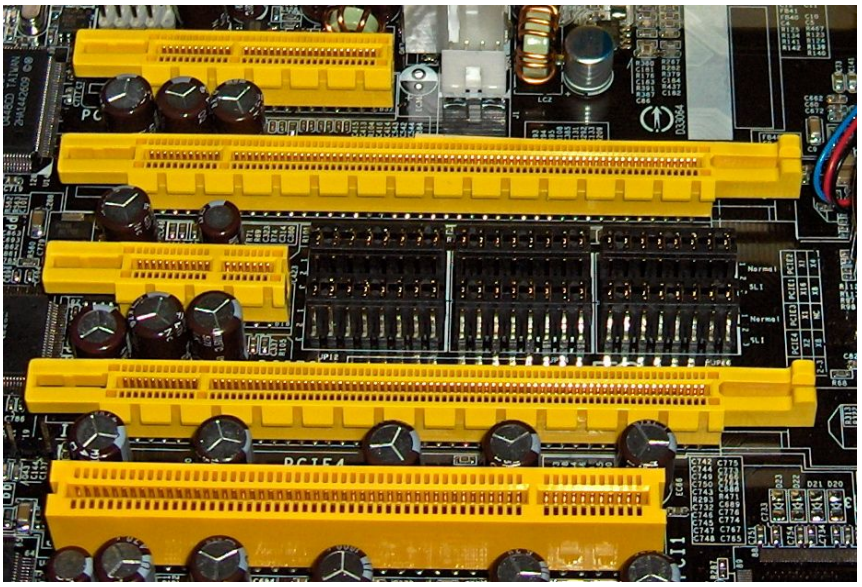


Figura 9.45: De cima para baixo: PCIe 4x, PCIe 16x, PCIe 1x, PCIe 16x e PCI convencional (32 bits, 5V)

Nos microprocessadores modernos os controladores de PCIe e o controlador de vídeo *onboard*[5] já se encontram integrados nos mesmos *chips* do processador. São poucas funções que ficaram fora da CPU. Os fabricantes de *chipsets* decidiram integrar as restantes funções da ponte norte e da ponte sul que não foram incorporadas na CPU numa única pastilha, como o ***Platform Controller Hub*** (*PCH*) *chipset* da Intel mostrado na Figura 9.46. Observe que neste *chipset* as conexões entre a CPU e o *chipset* se dão através de interfaces proprietárias Intel *Flexible Display Interface* (*FDI*) e Intel *Direct Media Interface* (*DMI*) com um desempenho muito melhor. Intel FDI conecta o controlador do processador gráfico com as entradas/saídas de *display* de PCH [13]. E Intel DMI pode ser tem muitas características do barramento PCIe usando múltiplas lanes e sinalização diferencial, separando a ponte norte da ponte sul [14].

[5] As placas de vídeo que vêm embutidas na placa mãe são chamadas de *onboard*, e as que podem ser instaladas posteriormente no PC são conhecidas como *offboard*.

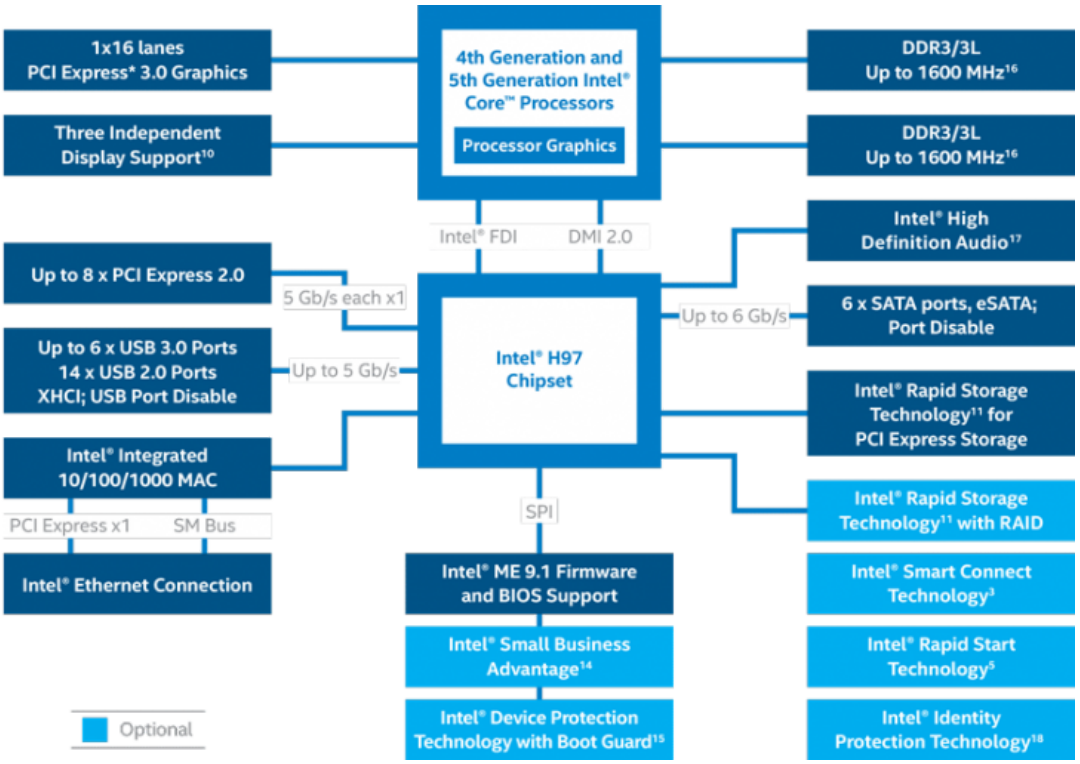



Figura 9.46: PCH chipset Z97 (Fonte: [12]).

# 9.8 Padrões de Comunicação Serial

Para a comunicação série com periféricos existem diversas normas elétricas. Estas normas visam padronizar os níveis elétricos dos sinais a serem transferidos. É necessário colocar entre periférico e a interface um circuito que compatibilize esta conexão.

## 9.8.1 EIA - RS232c / CCITT V24

Um padrão antigo, mas persistente, o RS-232, padronizado pela *Electronics Industries Association* (EIA) e depois adotado pelo CCITT (*Consultative Committee for International Telephony and Telegraphy*)[6] como padrão CCITT V24, foi introduzido pela primeira vez em 1962 para conectar teletipos a modems. A Figura 9.47 ilustra a ligação típica envolvendo a aplicação do padrão EIA RS232, identificando os principais agentes de uma conexão segundo o padrão. O padrão foi definido para normatizar a ligação digital entre um equipamento terminal de dados (**DTE** – ***D**ata **T**erminal **E**quipment*) e um equipamento de comunicação de dados

[6] Baseado em Genebra na Suíça, surgiu em 1947 como um comitê especializado da Organização das Nações Unidas (ONU). Em 1956 esse comitê ganhou status de organização, passando a ser denominado, em francês: *Comité consultatif international téléphonique et télégraphique* (CCITT). Em 1993, passou a ter a designação corrente, *ITU-T* (em inglês: *International Telecommunication Union "ITU" - Telecommunication Standardization Sector, "-T"*) (wiki_itu,2019).

(**DCE** – ***D**ata **C**ommunication **E**quipment*), normalmente um *modem*. O modem converte os sinais digitais para que sejam enviados por meio telefônico.

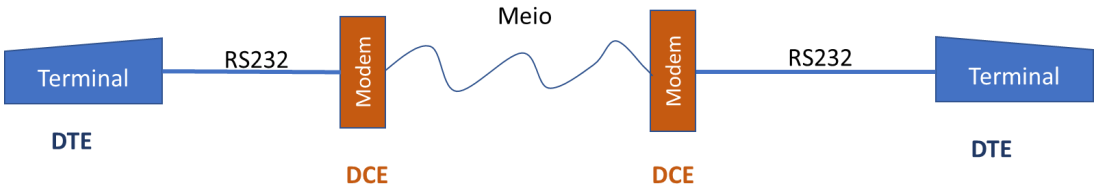



Figura 9.47: Ligação envolvendo o padrão EIA RS232

**O que é um MODEM?** https://www.tldp.org/HOWTO/Modem-HOWTO-4.html

Para enviar dados do computador pela linha telefônica, o equipamento modem pega o sinal digital do computador e o converte em "analógico". Isso é feito criando uma onda senoidal analógica e depois "**modulando**" sobre ela o sinal digital. Como o resultado o sinal modulado representa os dados digitais. No outro extremo da linha telefônica, outro modem "**demodula**" esse sinal e o sinal digital puro é recuperado. Junte as partes "**mod**" e "**dem**" das duas palavras acima e você obterá "modem". Um "modem" é, portanto, um MODulator-DEModulator. Para maiores detalhes sobre os diversos tipos de modulação considere https://www.tldp.org/HOWTO/Modem-HOWTO-21.html#modulate_.

O padrão RS232 define sinais elétricos e tipos de conectores. Ele persiste até hoje por causa de sua simplicidade e por ainda existirem equipamentos industriais antigos que o utilizam. O padrão define como um dispositivo pode transmitir uma palavra para outro dispositivo de forma **assíncrona** (o que significa que os dispositivos não compartilham um sinal de relógio). Um microcontrolador normalmente usa um **UART** para converter o conteúdo de um registro em uma sequência de bits para transmissão através de um link serial RS-232. Para um projetista de sistemas embarcados, uma questão importante a ser considerada é que as interfaces RS-232 podem ser bastante lentas e podem retardar o *software* do aplicativo, se o programador não for muito cuidadoso.

O mecanismo RS-232 é muito simples. O transmissor e o receptor devem, previamente, concordar com alguns valores de parâmetros de configuração da interface. Tanto o transmissor quanto o receptor têm que estar com os mesmos valores dos seguintes parâmetros:

- a **taxa de transmissão**, em bits por segundo, ou *Baud Rate*[7];
- quantidade de ***bits* de dados** transmitidos, podendo ser 7 ou 8 *bits*;

[7] **Baud** deriva do sobrenome de Émile Baudot, francês inventor do código telegráfico Baudot.(wiki_baud, 2019)

- a existência ou não de um **campo de paridade** e o seu tipo, que pode ser: Par ou Ímpar. O campo de paridade, com um *bit*, é definido pela quantidade de *bits* de dados iguais a 1 lógico. Na **Paridade Par**, o número de *bits* de informação (dados e campo de paridade) enviados sempre conterá um número par de 1s lógicos. Se o número de *bits* de dados iguais a 1 for ímpar, será adicionado um *bit* de paridade igual a 1, caso contrário, um *bit* igual a 0 será usado. Na **Paridade ímpar**, definido como na paridade par, só que agora, o número de *bits* iguais a 1 sempre será ímpar. Observe que o sistema de paridade que usa um *bit* para cada palavra de dados não é capaz de encontrar todos os erros. Apenas erros que causam um número ímpar de *bits* serão detectados. O segundo problema é que não há como saber qual *bit* é falso;
- a quantidade de ***Stop bits*****,** que pode ser 1 ou 2. A quantidade de *stop bits* define a separação mínima entre o envio de dois dados consecutivos.

O transmissor inicia a transmissão de um *byte* com um ***Start Bit***, que alerta o receptor de que uma palavra está chegando (Figura 9.48). O transmissor então envia a sequência de *bits* na taxa acordada. O relógio do receptor é reiniciado ao receber o *Start Bit* e espera-se que acompanhe o relógio do transmissor o suficiente para poder amostrar o sinal recebido sequencialmente e recuperar a sequência de bits.

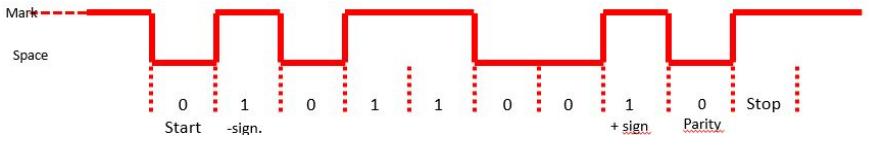



Figura 9.48: Transmissão de um caracter por um canal RS232.

Além das características elétricas dos sinais transmitidos e recebidos (Tabela 9.2), a norma RS 232C define também as características físicas da ligação (cabos e conectores). Ela distribui num conector de 25 pinos, os sinais de transmissão e recepção, o nível de referência (*Ground*), e alguns sinais de controle úteis a ligação com modems (Figura 9.49). Tabela 9.3 detalha as funções de cada pino dos conectores. Figura 9.50 sintetiza os modos mais usuais de conexões.

Tabela 9.2 - Características do padrão TIA/EIA232. Fonte: Linear Technology

| Especificação | | RS232 |
|---|---|---|
| **Modo de Operação** | | Single-Ended |
| **Número de transmissores e receptores permitidos em uma linha** | | 1 Transmissor, 1 Receptor |
| **Comprimento máximo do cabo** | | 50 Feet* (15,24 metros) |
| **Taxa máxima de dados** | | 20kb/s |
| **Máxima voltagem aplicada a saída do transmissor** | | ±25V |
| **Sinal de saída do transmissor** | Mínimo com carga | ±5V |
| | Máximo sem carga | ±15V |
| **Terminação** | | 3kΩ to 7kΩ |
| **Output Slew Rate** | | 30V/µs (Max) |

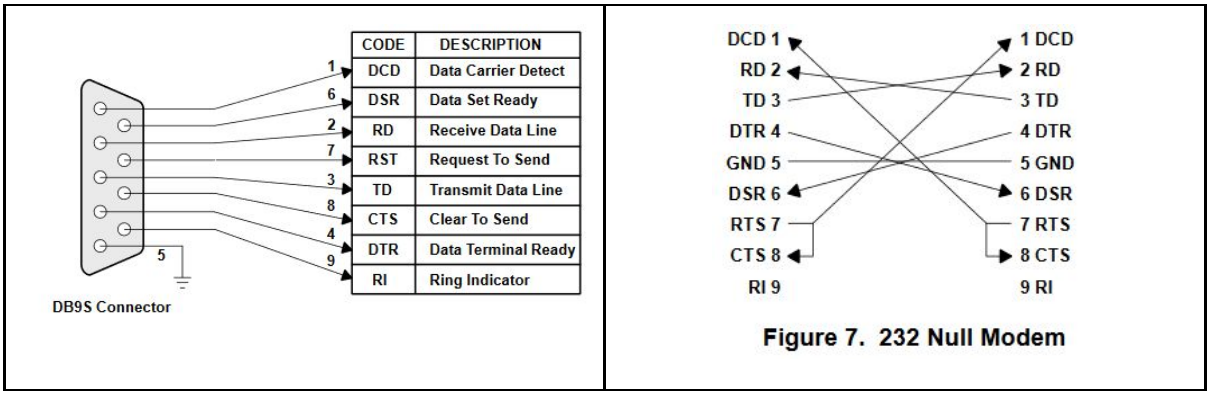
| | |
|---|---|
| **Faixa de voltagem na entrada do receptor** | ±25V (Max) |
| **Sensibilidade na entrada do receptor** | ±3V |
| **Resistência na entrada do receptor** | 3kΩ to 7kΩ |

Conector EIA RS232 do DCE  Conector EIA RS232 do DTE

Figura 9.49: Conectores especificados pela norma EIA RS232 originais DB25..

Tabela 9.3- Pinagem do conector DB25 com sinais EIA S232

| Pino | designação | nome | descrição RS 232 |
|---|---|---|---|
| **1** | AA | Shield | terra de proteção |
| **2** | BA | TxD | dados transmitidos pelo terminal |
| **3** | BB | RxD | dados recebidos do modem |
| **4** | BB | RTS | requisição de envio |
| **5** | CB | CTS | permissão de envio |
| **6** | CC | | dados prontos |
| **7** | ab | ground | terra do sinal |
| **8** | CF | DCD | detector de portadora |
| **9,14** | | Indef. | |
| **15** | DB | | clock do bit transmitido interno |
| **16** | | Indef. | |
| **17** | DD | | clock do bit recebido |
| **18,19** | | Indef. | |
| **20** | CD | | terminal pronto |
| **21** | | Indef. | |
| **22** | CE | | indicador de linha telefônica |
| **23** | | Indef. | |
| **24** | DA | | clock do bit transmitido, externo |
| **25** | | Indef. | |



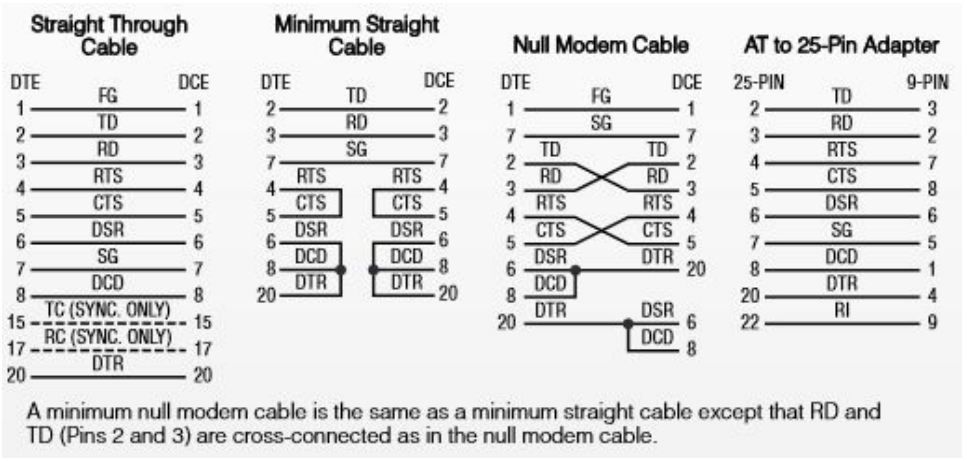
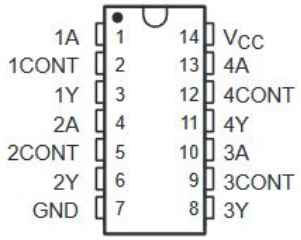



Figura 9.50: Principais característica principais da norma RS 232.

Existem disponíveis comercialmente módulos de circuitos integrados compatíveis com RS232, como módulos receptores MC1489x (Figura 9.50), módulos transmissores MC1488 (Figura 9.51), e módulos com vários receptores e transmissores integrados MAX232 (Figura 9.52).

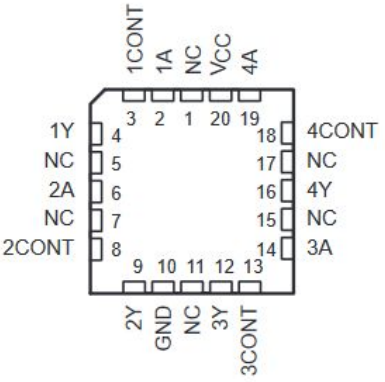
- **Input Resistance . . . 3 kΩ to 7 kΩ**
- **Input Signal Range . . . ±30 V**
- **Operate From Single 5-V Supply**
- **Built-In Input Hysteresis (Double Thresholds)**
- **Response Control that Provides:**
  - **Input Threshold Shifting**
  - **Input Noise Filtering**
- **Meet or Exceed the Requirements of TIA/EIA-232-F and ITU Recommendation V.28**
- **Fully Interchangeable With Motorola™ MC1489 and MC1489A**

**SN55189, SN55189A . . . J OR W PACKAGE**
**MC1489, MC1489A, SN75189, SN75189A**
**D, N, OR NS† PACKAGE**
**(TOP VIEW)**




† The NS package is only available left-end taped and reeled. For SN75189, order SN75189NSR.

**SN55189, SN55189A . . . FK PACKAGE**
**(TOP VIEW)**



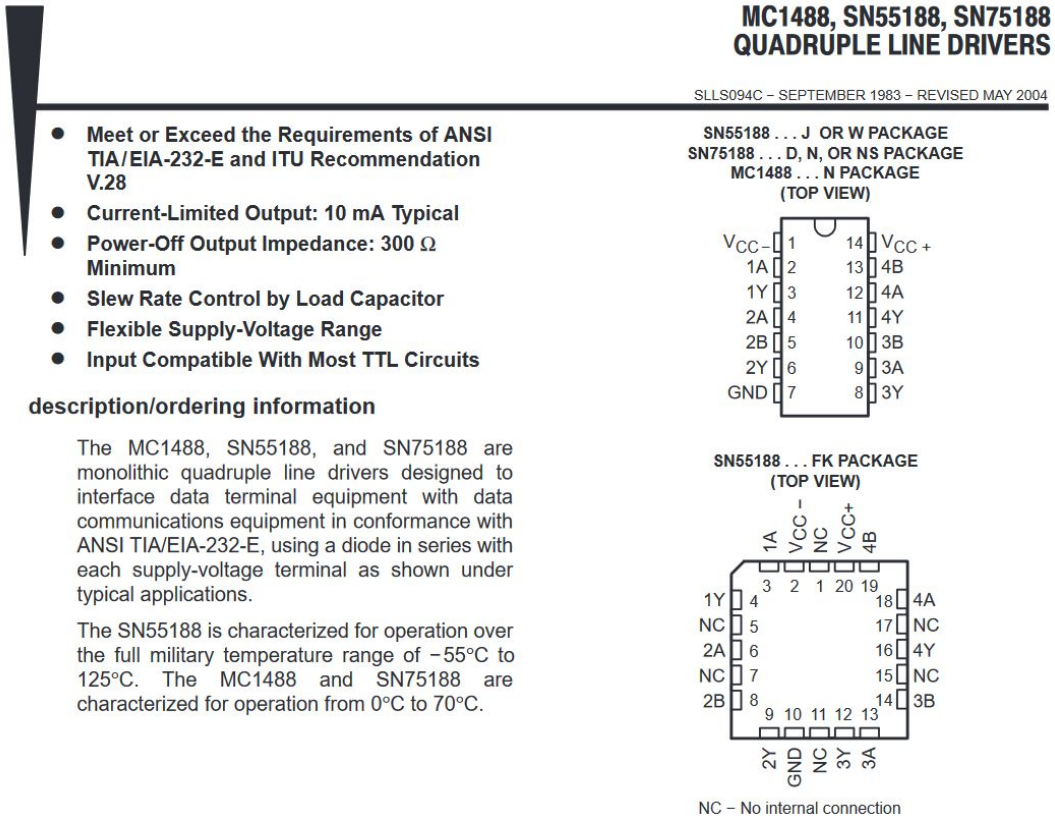

NC – No internal connection

## description

These devices are monolithic low-power Schottky quadruple line receivers designed to satisfy the requirements of the standard interface between data-terminal equipment and data-communication equipment as defined by TIA/EIA-232-F. A separate response-control (CONT) terminal is provided for each receiver. A resistor or a resistor and bias-voltage source can be connected between this terminal and ground to shift the input threshold levels. An external capacitor can be connected between this terminal and ground to provide input noise filtering.

The SN55189 and SN55189A are characterized for operation over the full military temperature range of −55°C to 125°C. The MC1489, MC1489A, SN75189, and SN75189A are characterized for operation from 0°C to 70°C.

Figura 9.50- Exemplo de receptor RS232 – MC1489 (fonte: Datasheet MC1489 1998)



**MC1488, SN55188, SN75188**
**QUADRUPLE LINE DRIVERS**

SLLS094C – SEPTEMBER 1983 – REVISED MAY 2004

- **Meet or Exceed the Requirements of ANSI TIA/EIA-232-E and ITU Recommendation V.28**
- **Current-Limited Output: 10 mA Typical**
- **Power-Off Output Impedance: 300 Ω Minimum**
- **Slew Rate Control by Load Capacitor**
- **Flexible Supply-Voltage Range**
- **Input Compatible With Most TTL Circuits**

**description/ordering information**

The MC1488, SN55188, and SN75188 are monolithic quadruple line drivers designed to interface data terminal equipment with data communications equipment in conformance with ANSI TIA/EIA-232-E, using a diode in series with each supply-voltage terminal as shown under typical applications.

The SN55188 is characterized for operation over the full military temperature range of −55°C to 125°C. The MC1488 and SN75188 are characterized for operation from 0°C to 70°C.

**SN55188 . . . J OR W PACKAGE**
**SN75188 . . . D, N, OR NS PACKAGE**
**MC1488 . . . N PACKAGE**
**(TOP VIEW)**

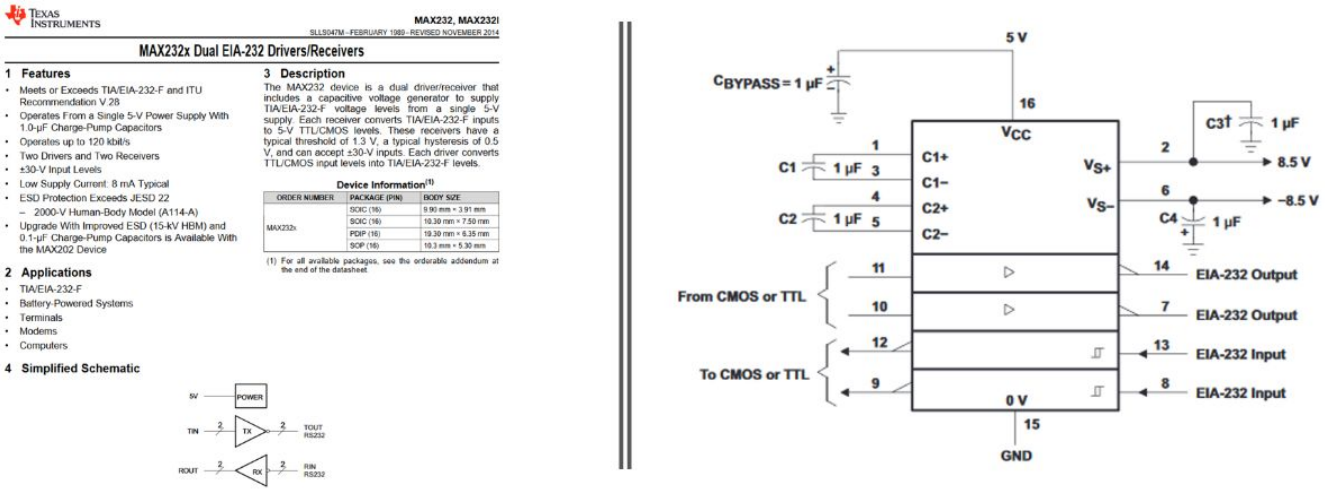


**SN55188 . . . FK PACKAGE**
**(TOP VIEW)**



NC – No internal connection

Figura 9.51: Exemplo de transmissor RS232 (fonte: Datasheet MC1488, 2004)



TEXAS INSTRUMENTS

MAX232, MAX232I

SLLS047M – FEBRUARY 1989 – REVISED NOVEMBER 2014

**MAX232x Dual EIA-232 Drivers/Receivers**

**1 Features**

- Meets or Exceeds TIA/EIA-232-F and ITU Recommendation V.28
- Operates From a Single 5-V Power Supply With 1.0-μF Charge-Pump Capacitors
- Operates up to 120 kbit/s
- Two Drivers and Two Receivers
- ±30-V Input Levels
- Low Supply Current: 8 mA Typical
- ESD Protection Exceeds JESD 22
  - 2000-V Human-Body Model (A114-A)
- Upgrade With Improved ESD (15-kV HBM) and 0.1-μF Charge-Pump Capacitors is Available With the MAX202 Device

**2 Applications**

- TIA/EIA-232-F
- Battery-Powered Systems
- Terminals
- Modems
- Computers

**3 Description**

The MAX232 device is a dual driver/receiver that includes a capacitive voltage generator to supply TIA/EIA-232-F voltage levels from a single 5-V supply. Each receiver converts TIA/EIA-232-F inputs to 5-V TTL/CMOS levels. These receivers have a typical threshold of 1.3 V, a typical hysteresis of 0.5 V, and can accept ±30-V inputs. Each driver converts TTL/CMOS input levels into TIA/EIA-232-F levels.

**Device Information(1)**

| ORDER NUMBER | PACKAGE (PIN) | BODY SIZE |
|---|---|---|
| MAX232x | SOIC (16) | 9.90 mm × 3.91 mm |
| | SOIC (16) | 10.30 mm × 7.50 mm |
| | PDIP (16) | 19.30 mm × 6.35 mm |
| | SOP (16) | 10.3 mm × 5.30 mm |

(1) For all available packages, see the orderable addendum at the end of the datasheet.

**4 Simplified Schematic**





Figura 9.5: Exemplo de chip com dois transmissores e dois receptores RS232 e aplicação típica (fonte: Datasheet MAX232 2014)

## 9.8.2 EIA - RS422 / CCITT V11

O EIA /TIA-422 define uma interface balanceada/diferencial especificando um único transmissor unidirecional com até 32 receptores (Figura 9.54). O TIA-422 suporta circuitos ponto-a-ponto e *multi-dropped*. O padrão EIA/TIA-422 é comumente referenciado como RS422, pois foi este termo que foi definido pelo EIA, quando propôs esta recomendação. O tipo de cabeamento e do conector não são

especificados em RS422. Tabela 9.4 sintetiza as principais características deste padrão.

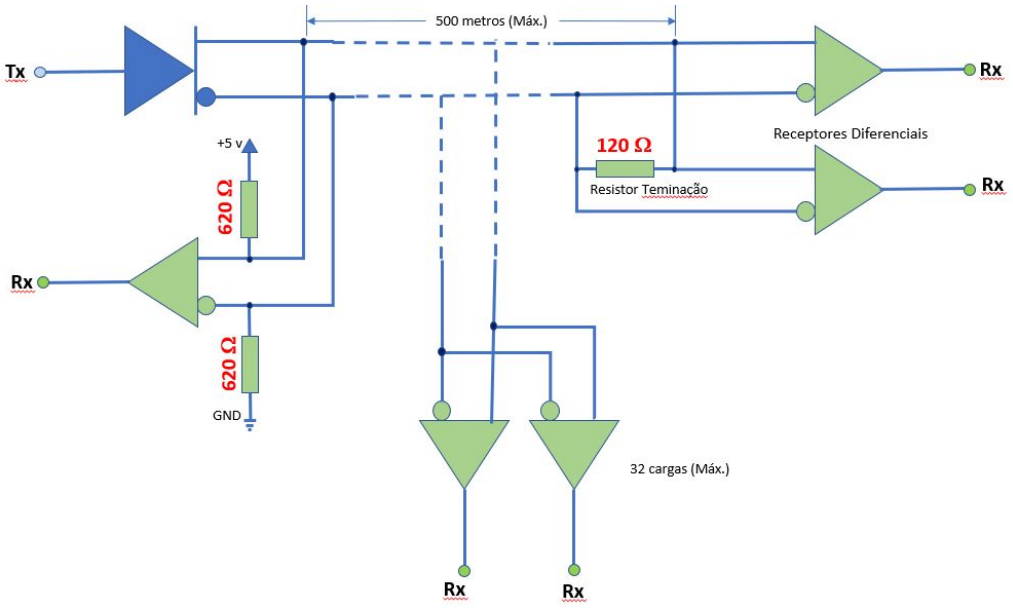



Figura 9.54: Circuito EIA/TIA-422

Tabela 9.4: Resumo da norma EIA/TIA422

| | |
|---|---|
| Norma | TIA/EIA-422 |
| Meio físico | Par trançado (*Twisted Pair*) |
| Topologia de conexão | Ponto-a-ponto, *Multi-dropped* |
| Quantidade máxima de dispositivos | 10 (1 transmissor e 10 receptores) |
| Distância máxima | Soma total de 1500 metros (4,900 ft) |
| Modo de Operação | Differential |
| Máximo *Binary Rate* | 100 kbit/s – 10 Mbit/s |
| Níveis de Voltagem | −6V a +6V (voltagem máxima diferencial) |
| Mark (1) | Voltagens negativas |
| Space (0) | Voltagens positivas |
| Sinais disponíveis | Tx+, Tx-, Rx+, Rx- (*Full Duplex*) |
| Tipos de conectores | Não especificado |

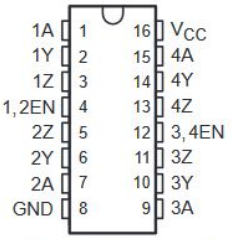
Existem disponíveis comercialmente módulos de circuitos integrados compatíveis com TIA-422, como módulos acionadores MC3487 (Figura 9.55) e módulos transmissores MC3486 (Figura 9.56).

**MC3487**
**QUADRUPLE DIFFERENTIAL LINE DRIVER**

SLLS098C – MAY 1980 – REVISED FEBRUARY 2004

- **Meets or Exceeds Requirements of ANSI TIA/EIA-422-B and ITU Recommendation V.11**
- **3-State, TTL-Compatible Outputs**
- **Fast Transition Times**
- **High-Impedance Inputs**
- **Single 5-V Supply**
- **Power-Up and Power-Down Protection**

D, N, OR NS PACKAGE
(TOP VIEW)




**description/ordering information**

The MC3487 offers four independent differential line drivers designed to meet the specifications of ANSI TIA/EIA-422-B and ITU Recommendation V.11. Each driver has a TTL-compatible input buffered to reduce current and minimize loading.

The driver outputs utilize 3-state circuitry to provide high-impedance states at any pair of differential outputs when the appropriate output enable is at a low logic level. Internal circuitry is provided to ensure the high-impedance state at the differential outputs during power-up and power-down transition times, provided the output enable is low.

The MC3487 is designed for optimum performance when used with the MC3486 quadruple line receiver. It is supplied in a 16-pin dual-in-line package and operates from a single 5-V supply.

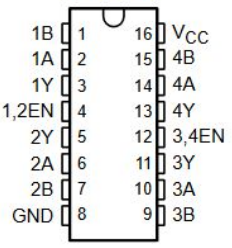
Figura 9.55: Circuito integrado acionador de linha TIA-422 - MC3487 (fonte: Datasheet3487,2004)

**MC3486**
**QUADRUPLE DIFFERENTIAL LINE RECEIVER WITH 3-STATE OUTPUTS**

SLLS097C – JUNE 1980 – REVISED FEBRUARY 2002

- **Meets or Exceeds the Requirements of ANSI Standards EIA/TIA-422-B and EIA/TIA-423-B and ITU Recommendations V.10 and V.11**
- **3-State, TTL-Compatible Outputs**
- **Fast Transition Times**
- **Operates From Single 5-V Supply**
- **Designed to Be Interchangeable With Motorola™ MC3486**

D, N, OR NS PACKAGE
(TOP VIEW)




**description**

The MC3486 is a monolithic quadruple differential line receiver designed to meet the specifications of ANSI Standards TIA/EIA-422-B and TIA/EIA-423-B and ITU Recommendations V.10 and V.11. The MC3486 offers four independent differential-input line receivers that have TTL-compatible outputs. The outputs utilize 3-state circuitry to provide a high-impedance state at any output when the appropriate output enable is at a low logic level.

The MC3486 is designed for optimum performance when used with the MC3487 quadruple differential line driver. It is supplied in a 16-pin package and operates from a single 5-V supply.

The MC3486 is characterized for operation from 0°C to 70°C.

Figura 9.56: Circuito integrado receptor de linha TIA-422 - MC3486 (fonte: Datasheet3486,2004)

## 9.8.3 EIA - RS423 / CCITT V11

A interface segundo o padrão EIA/TIA-423 é desbalanceada, ou seja, *single-ended* (Figura 9.57). É *multi-dropped*, especifica um único driver unidirecional com vários receptores (até 10). Especifica as características elétricas do circuito de interface

digital de tensão desbalanceada. O padrão EIA-423 não define conector nem cabo. Mas suas características elétricas são usadas no EIA-449 e no EIA530, ambos definem o cabeamento e a pinagem para formar uma interface completa. As suas características são resumidas na Tabela 9.5.

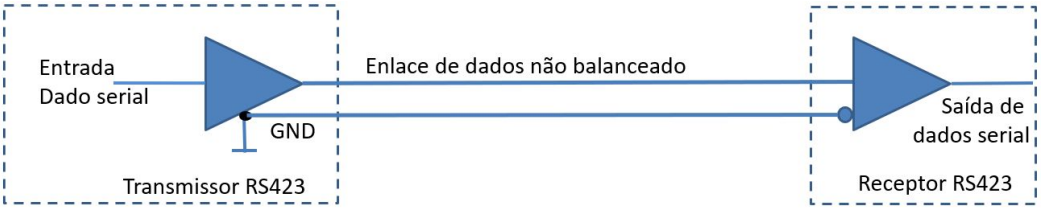



Figura 9.57: Conexão EAI RS423

Tabela 9.5: Resumo de características do RS423 (Fonte: wiki423, 2019)

| RS423 | |
|---|---|
| Standard | EIA RS-423 |
| Physical Media | Group of copper cables |
| Network Topology | Point-to-point, Multi-dropped |
| Maximum Devices | 10 (1 driver & 10 receivers) |
| Maximum Distance | 1200 metres (4000 feet) |
| Mode of Operation | Single-ended (unbalanced) |
| Maximum Baud Rate | Up to 100kbit/s |
| Voltage Levels | -6V to +6V (maximum) |
| Mark(1) | -4V to -6V |
| Space(0) | +4V to +6V |
| Available Signals | Tx, Rx, GND |

## 9.8.4 TIA - RS485

TIA-485 é um padrão para interconexão balanceada/diferencial. O padrão define somente a camada física, ou seja, as características elétricas e temporais dos sinais da interconexão (Figura 9.58). O protocolo e a aplicação da sinalização não são definidos.

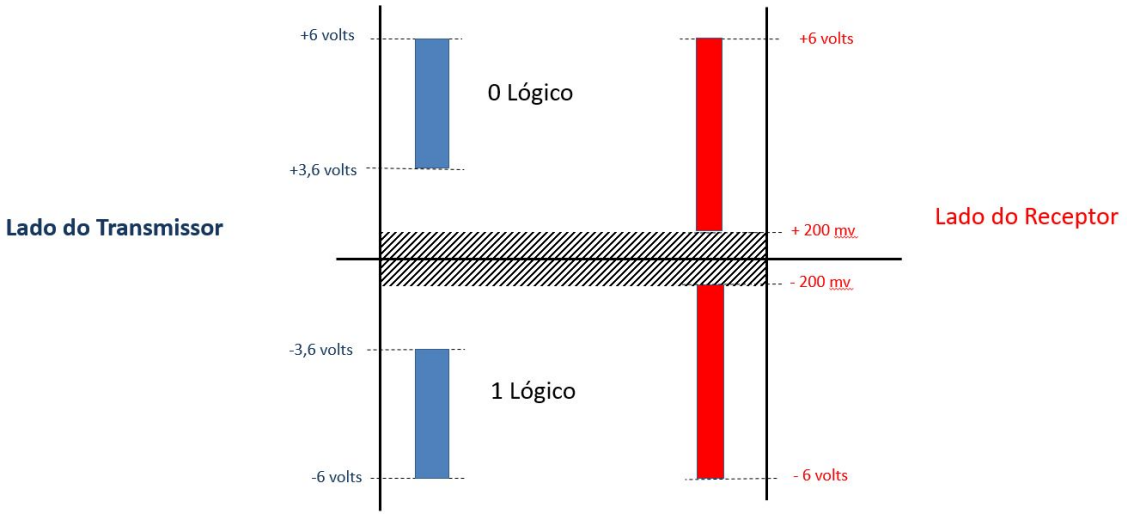



Figura 9.58: Níveis elétricos dos sinais TIA485.

O TIA-485 é comumente chamado de padrão RS485. O termo RS485 foi usado quando da sua definição pela EIA (*Electronics Industries Association*). Hoje em dia o termo está em desuso. O TIA-485 especifica a transmissão de dados bidirecional e *half-duplex*. Até 32 transmissores e 32 receptores podem ser interconectados em qualquer combinação, incluindo um transmissor e vários receptores (*multi-drop*) ou um receptor e vários transmissores. Diversas especificações de sistemas de controle industrial usam o TIA-485 como padrão da camada física (elétrica), incluindo o MODBus e Profibus. Outras especificações de aplicação como, em entretenimento (DMX512, AES3) e em automação residencial usam o TIA 485 como padrão elétrico de interconexão (wiki485, 2019).

A Tabela 9.6 resume as características principais da norma TIA-485. Em contraste com o RS-422, que possui um único circuito de acionamento que não pode ser desligado, os *drivers* RS-485 usam uma lógica de *tri-state*, permitindo que os transmissores individuais possam ser desativados. Isso permite que o TIA-485 implemente topologias de barramento linear usando apenas dois fios.

Os dispositivos TIA-485 podem ser usados em circuitos 422, mas os transmissores TIA-422 não podem ser usados em circuitos TIA-485.

Tabela 9.6: Resumo das características da norma TIA485

| | |
|---|---|
| Norma | ANSI/TIA/EIA-485-A-1998<br>Aprovado em: 3 de março de 1998<br>Reafirmado: 28 de março de 2003 |
| Meio físico | Cabo balanceado de interconexão |
| Topologia de rede | Ponto-a-ponto, *multi-dropped*, *multi-point* |
| Quantidade máxima de dispositivos | Ao menos 32 cargas |
| Máxima distância | Não especificada |
| Máximo *Baud Rate* | 10 Mbps - 64Mbps |
| Modos de operação | Diferentes níveis no receptor:<br>1 binário (OFF) (Voa–Vob < −200 mV)<br>0 binário (ON) (Voa–Vob > +200 mV) |
| Sinais disponíveis | A, B, C |
| Tipos de conector | Não especificado |

## 9.8.5 JTAG

Outra interface serial amplamente implementada em processadores embarcados é conhecida como JTAG (*Joint Test Action Group*), ou mais formalmente como a porta de acesso de teste padrão IEEE 1149.1 e arquitetura "*boundary-scan"*. Essa interface surgiu em meados da década de 1980 para resolver o problema de que os empacotamentos de circuitos integrados e a tecnologia da placa de circuito impresso haviam evoluído a tal ponto que testar circuitos usando pontas de prova elétricas se tornou difícil ou impossível. Pontos no circuito que precisavam ser acessados tornaram-se inacessíveis às pontas de prova.

A porta de acesso JTAG serve para várias funções. Uma mesma porta pode incluir qualquer uma ou todos as seguintes funções:

- lógica de teste que permite testar conexões entre dispositivos sem pontas de prova externas;
- lógica de programação nos CPLDs e FPGAs para programação desses dispositivos já instaladas em suas placas; e
- lógica de depuração em microprocessadores e microcontroladores usados para depuração de *software*, ou para testar conexões com dispositivos periféricos embutidos, ou para programar o flash integrado num microcontrolador.

A noção de uma **varredura de limite**, em inglês *boundary-scan*, permite que o estado de um ponto lógico de um circuito (o que tradicionalmente seriam pinos acessíveis às pontas de prova) seja lido ou gravado serialmente através de pinos acessíveis. Hoje, as portas JTAG são amplamente usadas para fornecer uma interface de depuração para processadores embutidos, permitindo que um ambiente de depuração hospedado em PC examine e controle o estado interno de um processador embutido. A porta JTAG é usada, por exemplo, para ler o estado dos registros do processador, definir pontos de interrupção em um programa e executar uma única etapa do programa. Uma variante mais recente é o *Serial Wire Debug* (SWD), que fornece funcionalidade semelhante com menos pinos.

O padrão JTAG é extensível para incluir, por exemplo, instruções definidas pelo usuário e registradores de dados definidos pelo usuário. Isto permitiu que fosse estendido por muitos fabricantes de *chips* semicondutores com variantes especializadas para fornecer recursos específicos deles.

# 9.9 Padrões *De Facto* de Comunicação Serial

Alguns padrões de conexão foram definidos por único fabricante, mas se mostraram muito interessante e eficazes que acabaram sendo adotados pelo mercado, tornando de fato um padrão.

## 9.9.1 I2C - *Inter Integrated Circuit*

O barramento I2C foi inventado pela Philips (NXP) em 1982. É um padrão mundial *de facto* que hoje em dia é implementado em, aproximadamente, 1000 diferentes CI's fabricados por mais de 50 companhias. Adicionalmente, o barramento I2C inspirou várias arquiteturas de controle como *System Management Bus* (SMBus), *Power Management Bus* (PMBus), *Intelligent Platform Management Interface* (IPMI), *Display Data Channel* (DDC) e *Advanced Telecom Computing Architecture* (ATCA). O barramento I2C está presente em muitos sistemas embarcados.

O barramento I2C tem somente dois fios bidirecionais em o*pen collector,* uma linha de dados (SDA) e uma linha de relógio (SCL). A Figura 9.59 ilustra um barramento I2C com um *Master* que pode enviar dados para vários *Slaves.* Cada *slave* tem seu próprio endereço, configurado pelas entradas A0-A2 e previamente conhecido pelo *master.* Na medida que o *master* precisa enviar uma informação para um *slave*, ele endereça o seu alvo, envia os dados e termina a transferência.

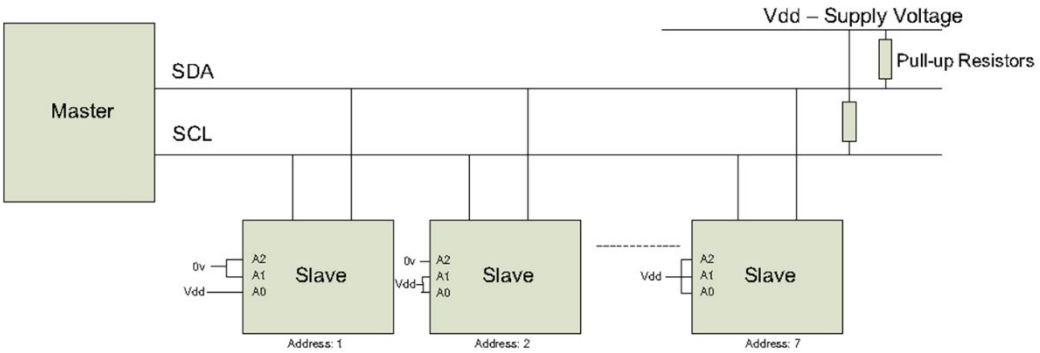



Figura 9.59: Exemplo de circuito I2C

### 9.9.1.1 O que faz o padrão interessante para o projetista?

Os CIs compatíveis com o barramento I2C permitem que um projeto de sistema progrida rapidamente diretamente do diagrama de blocos funcional para um protótipo. Além disso, como eles 'conectam' diretamente no barramento I2C sem nenhuma interface externa adicional, eles permitem que um sistema de protótipo seja modificado ou atualizado simplesmente ‘ligando' ou 'soltando' ICs do, ou para o, barramento.

Abaixo alguns recursos dos CIs compatíveis com o barramento I2C que são particularmente atraentes para os projetistas:

- Os blocos funcionais no diagrama de blocos correspondem aos CIs reais; os projetos avançam rapidamente do diagrama de blocos para o esquema final.
- Não há necessidade de projetar interfaces de barramento porque a interface de barramento I2C já está integrada no *chip*.
- O protocolo integrado de endereçamento e transferência de dados permite que os sistemas sejam completamente definidos por *software*.
- Os mesmos tipos de IC costumam ser usados em muitas aplicações diferentes.
- O tempo de projeto reduz à medida que os projetistas se familiarizam rapidamente com os blocos funcionais frequentemente representados pelos CI’s compatíveis com barramento I2C.
- Os ICs podem ser adicionados ou removidos de um sistema sem afetar outros circuitos no barramento.
- O diagnóstico e a depuração de falhas são simples.
- O tempo de desenvolvimento de *software* pode ser reduzido montando uma biblioteca de módulos de *software* reutilizáveis.

Além dessas vantagens, os CIs CMOS compatíveis com o barramento I2C oferecem aos projetistas recursos especiais que são particularmente atraentes para equipamentos portáteis e sistemas alimentados por bateria. Todos eles têm:

- consumo de corrente extremamente baixo,

- imunidade a ruídos altos,
- ampla faixa de tensão de alimentação, e
- ampla faixa de temperatura de operação.

O barramento I2C é popular porque é simples de usar, podendo haver mais de um mestre, apenas a velocidade superior do barramento é definida e apenas dois fios com resistores *pull-up* são necessários para conectar um número quase ilimitado de dispositivos I2C. O I2C pode usar microcontroladores ainda mais lentos com pinos GPIO, pois eles só precisam gerar condições corretas de Partida e Parada, além de funções para ler e escrever um *byte*.

Cada dispositivo escravo possui um endereço exclusivo. A transferência de e para o dispositivo mestre é serial e é dividida em pacotes de 8 *bits*. Todos esses requisitos simples simplificam a implementação da interface I2C, mesmo com microcontroladores baratos que não possuem controlador de *hardware* I2C especial. Você só precisa de 2 pinos de E / S livres e poucas rotinas i2C simples para enviar e receber comandos.

As especificações I2C iniciais definiam a frequência máxima do *clock* em 100 kHz. Mais tarde, isso foi aumentado para 400 kHz no ***Fast Mode***. Há também um ***High Speed Mode*** que pode ir até 3,4 MHz e também um ***Ultra-Fast Mode*** de 5 MHz.

### 9.9.1.2 Endereços I2C

(https://i2c.info/)

A comunicação I2C básica usa transferências de 8 *bits* (*byte*). Cada dispositivo escravo I2C possui um endereço de 7 *bits* que precisa ser exclusivo num barramento. Alguns dispositivos têm endereço I2C fixo, enquanto outros têm poucas entradas de endereço que determinam os bits menos significativos do endereço I2C. Isso torna muito fácil ter todos os dispositivos I2C no barramento com endereço I2C exclusivo. Também existem dispositivos com endereço de 10 *bits*.

No endereçamento de 7 *bits*, os *bits* 7 a 1 representam o endereço, enquanto o *bit* 0 é usado para sinalizar a operação (leitura ou escrita) no dispositivo. Se o *bit* 0 (no *byte* do endereço) estiver definido como 1, o dispositivo mestre lerá o dispositivo I2C escravo. O dispositivo mestre não precisa de endereço, pois é ele que gera o relógio (via SCL) e endereça individualmente os dispositivos escravos I2C.

### 9.9.1.3 Protocolo I2C

No estado normal, ambas as linhas (SCL e SDA) estão em nível alto. A comunicação é iniciada pelo dispositivo mestre. Os dados são transferidos em mensagens. As mensagens são divididas em quadros de dados (Figura 9.60). Cada mensagem possui um quadro de endereços que contém o endereço binário do escravo e um ou mais quadros de dados que contêm os dados que estão sendo transmitidos. A mensagem também inclui condições de início e parada, *bits* de leitura/gravação e *bits* ACK/NACK entre cada quadro de dados.

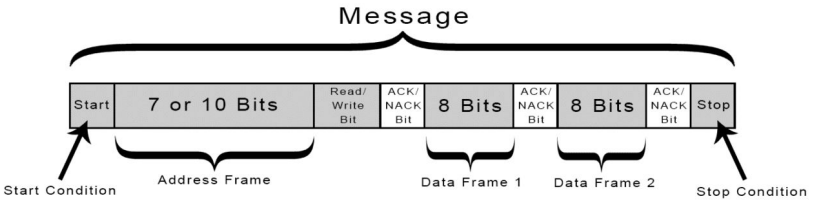



Figura 9.60: Uma mensagem e seus quadros num barramento I2C (fonte: i2c.info,2019) ).

Nas Figuras 9.61 e 9.62 são sintetizadas, respectivamente, as características elétricas e temporais co protocolo I2C.

| Symbol | Parameter | Conditions | Standard-mode | | Fast-mode | | Fast-mode Plus | | Unit |
|---|---|---|---|---|---|---|---|---|---|
| | | | Min | Max | Min | Max | Min | Max | |
| $V_{IL}$ | LOW-level input voltage[1] | | –0.5 | $0.3V_{DD}$ | –0.5 | $0.3V_{DD}$ | –0.5 | $0.3V_{DD}$ | V |
| $V_{IH}$ | HIGH-level input voltage[1] | | $0.7V_{DD}$ | [2] | $0.7V_{DD}$ | [2] | $0.7V_{DD}$[1] | [2] | V |
| $V_{hys}$ | hysteresis of Schmitt trigger inputs | | - | - | $0.05V_{DD}$ | - | $0.05V_{DD}$ | - | V |
| $V_{OL1}$ | LOW-level output voltage 1 | (open-drain or open-collector) at 3 mA sink current; $V_{DD} > 2$ V | 0 | 0.4 | 0 | 0.4 | 0 | 0.4 | V |
| $V_{OL2}$ | LOW-level output voltage 2 | (open-drain or open-collector) at 2 mA sink current[3]; $V_{DD} \leq 2$ V | - | - | 0 | $0.2V_{DD}$ | 0 | $0.2V_{DD}$ | V |
| $I_{OL}$ | LOW-level output current | $V_{OL} = 0.4$ V | 3 | - | 3 | - | 20 | - | mA |
| | | $V_{OL} = 0.6$ V[4] | - | - | 6 | - | - | - | mA |
| $t_{of}$ | output fall time from $V_{IHmin}$ to $V_{ILmax}$ | | - | 250[5] | 20 × ($V_{DD}$ / 5.5 V)[6] | 250[5] | 20 × ($V_{DD}$ / 5.5 V)[6] | 120[7] | ns |
| $t_{SP}$ | pulse width of spikes that must be suppressed by the input filter | | - | - | 0 | 50[8] | 0 | 50[8] | ns |
| $I_i$ | input current each I/O pin | $0.1V_{DD} < V_I < 0.9V_{DDmax}$ | –10 | +10 | –10[9] | +10[9] | –10[9] | +10[9] | µA |
| $C_i$ | capacitance for each I/O pin[10] | | - | 10 | - | 10 | - | 10 | pF |

[1] Some legacy Standard-mode devices had fixed input levels of $V_{IL}$ = 1.5 V and $V_{IH}$ = 3.0 V. Refer to component data sheets.

[2] Maximum $V_{IH} = V_{DD(max)} + 0.5$ V or 5.5 V, which ever is lower. See component data sheets.

[3] The same resistor value to drive 3 mA at 3.0 V $V_{DD}$ provides the same RC time constant when using <2 V $V_{DD}$ with a smaller current draw.

[4] In order to drive full bus load at 400 kHz, 6 mA $I_{OL}$ is required at 0.6 V $V_{OL}$. Parts not meeting this specification can still function, but not at 400 kHz and 400 pF.

[5] The maximum $t_f$ for the SDA and SCL bus lines quoted in Table 10 (300 ns) is longer than the specified maximum $t_{of}$ for the output stages (250 ns). This allows series protection resistors ($R_s$) to be connected between the SDA/SCL pins and the SDA/SCL bus lines as shown in Figure 45 without exceeding the maximum specified $t_f$.

[6] Necessary to be backwards compatible with Fast-mode.

[7] In Fast-mode Plus, fall time is specified the same for both output stage and bus timing. If series resistors are used, designers should allow for this when considering bus timing.

[8] Input filters on the SDA and SCL inputs suppress noise spikes of less than 50 ns.

[9] If $V_{DD}$ is switched off, I/O pins of Fast-mode and Fast-mode Plus devices must not obstruct the SDA and SCL lines.

[10] Special purpose devices such as multiplexers and switches may exceed this capacitance because they connect multiple paths together.

Figura 9.61: Características Elétricas dos sinais SDA e SCL (I2C) (fonte: NXP, 2014)

| Symbol | Parameter | Conditions | Standard-mode | | Fast-mode | | Fast-mode Plus | | Unit |
|---|---|---|---|---|---|---|---|---|---|
| | | | Min | Max | Min | Max | Min | Max | |
| $f_{SCL}$ | SCL clock frequency | | 0 | 100 | 0 | 400 | 0 | 1000 | kHz |
| $t_{HD;STA}$ | hold time (repeated) START condition | After this period, the first clock pulse is generated. | 4.0 | - | 0.6 | - | 0.26 | - | µs |
| $t_{LOW}$ | LOW period of the SCL clock | | 4.7 | - | 1.3 | - | 0.5 | - | µs |
| $t_{HIGH}$ | HIGH period of the SCL clock | | 4.0 | - | 0.6 | - | 0.26 | - | µs |
| $t_{SU;STA}$ | set-up time for a repeated START condition | | 4.7 | - | 0.6 | - | 0.26 | - | µs |
| $t_{HD;DAT}$ | data hold time[2] | CBUS compatible masters (see Remark in Section 4.1) | 5.0 | - | - | - | - | - | µs |
| | | $I^2C$-bus devices | 0[3] | -[4] | 0[3] | -[4] | 0 | - | µs |
| $t_{SU;DAT}$ | data set-up time | | 250 | - | 100[5] | - | 50 | - | ns |
| $t_r$ | rise time of both SDA and SCL signals | | - | 1000 | 20 | 300 | - | 120 | ns |
| $t_f$ | fall time of both SDA and SCL signals[3][6][7][8] | | - | 300 | $20 \times (V_{DD} / 5.5\ V)$ | 300 | $20 \times (V_{DD} / 5.5\ V)$[9] | 120[8] | ns |
| $t_{SU;STO}$ | set-up time for STOP condition | | 4.0 | - | 0.6 | - | 0.26 | - | µs |
| $t_{BUF}$ | bus free time between a STOP and START condition | | 4.7 | - | 1.3 | - | 0.5 | - | µs |
| $C_b$ | capacitive load for each bus line[10] | | - | 400 | - | 400 | - | 550 | pF |
| $t_{VD;DAT}$ | data valid time[11] | | - | 3.45[4] | - | 0.9[4] | - | 0.45[4] | µs |
| $t_{VD;ACK}$ | data valid acknowledge time[12] | | - | 3.45[4] | - | 0.9[4] | - | 0.45[4] | µs |
| $V_{nL}$ | noise margin at the LOW level | for each connected device (including hysteresis) | $0.1V_{DD}$ | - | $0.1V_{DD}$ | - | $0.1V_{DD}$ | - | V |
| $V_{nH}$ | noise margin at the HIGH level | for each connected device (including hysteresis) | $0.2V_{DD}$ | - | $0.2V_{DD}$ | - | $0.2V_{DD}$ | - | V |

[1] All values referred to $V_{IH(min)}$ ($0.3V_{DD}$) and $V_{IL(max)}$ ($0.7V_{DD}$) levels (see Table 9).

[2] $t_{HD;DAT}$ is the data hold time that is measured from the falling edge of SCL, applies to data in transmission and the acknowledge.

[3] A device must internally provide a hold time of at least 300 ns for the SDA signal (with respect to the $V_{IH(min)}$ of the SCL signal) to bridge the undefined region of the falling edge of SCL.

[4] The maximum $t_{HD;DAT}$ could be 3.45 µs and 0.9 µs for Standard-mode and Fast-mode, but must be less than the maximum of $t_{VD;DAT}$ or $t_{VD;ACK}$ by a transition time. This maximum must only be met if the device does not stretch the LOW period ($t_{LOW}$) of the SCL signal. If the clock stretches the SCL, the data must be valid by the set-up time before it releases the clock.

Figura 9.62: Características Temporais de SDA e SCL (I2C) (fonte: NXP, 2014)

### 9.9.1.4 Passos da transmissão de dados no I2C

http://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/

1. O mestre envia uma condição de *Start* para todos os escravos conectados. Esta condição é sinalizada comutando a linha de SDA do nível alto para baixo antes de comutar o SCL de alto para baixo.
2. O mestre envia aos escravos conectados, os 7 ou 10 *bits* de endereço do escravo como o qual ele quer se comunicar, junto com o *bit* de leitura/escrita.
3. Cada escravo compara o endereço enviado pelo mestre com seu próprio endereço. Se o endereço for o seu, o escravo retorna um *bit* de ACK fazendo que a linha DAS fique em nível baixo por um *bit*. Se o endereço recebido não coincidir com o seu endereço, o escravo deixa a linha SDA em nível alto.
4. O mestre envia ou recebe o quadro de dados.
5. Após cada quadro de dados tiver sido transferido, o dispositivo que o receber retorna um *bit* de ACK para quem o enviou para reconhecer a recepção com sucesso do quadro.
6. Para parar a transmissão de dados, o mestre envia uma condição de parada para o escravo, fazendo o SCL alto antes de comutar o SDA para alto.

## 9.9.2 SMB - System Management Bus

O *System Management Bus* é um barramento de dois fios usado para comunicações *host*-dispositivo e dispositivo-dispositivo (http://smbus.org/specs/) (Figura 9.63). Foi originalmente inventado pela Intel, mas seus princípios são semelhantes aos definidos pelo barramento I2C. Um resumo comparaativo entre I2C e SMBus é apresentado na Tabela 9.7.

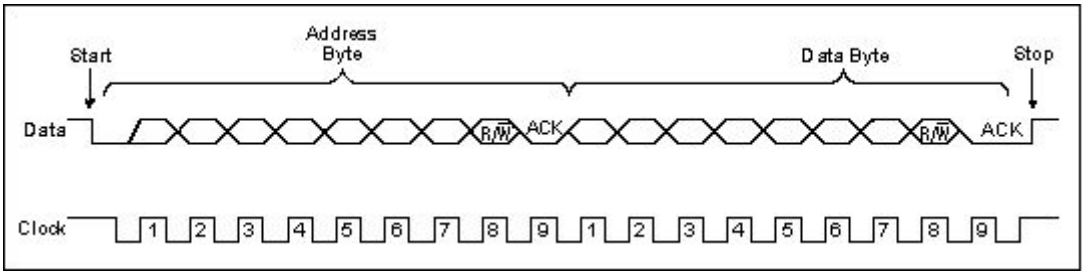



Figura 9.63: Uma típica comunicação no barramento SMBus, mostrando as condições de Start e de Stop (Fonte: Maxim, 2000)

Tabela 9.7: Resumo comparativo entre I2C e SMBus (fonte: Maxim, 2000)

| | I²C | SMBus |
|---|---|---|
| **Timeout** | No | Yes |
| **Minimum Clock Speed** | DC | 10kHz |
| **Maximum Clock Speed** | 100kHz (400kHz and 2MHz also available) | 100kHz |
| **VHIGH** | 0.7 × VDD, 3.0V Fixed | 2.1V |
| **VLOW** | 0.3 × VDD, 1.5V Fixed | 0.8V |
| **Max I** | 3mA | 350µA |
| **Clock Nomenclature** | SCL | SMBCLK |
| **Data Nomenclature** | SDA | SMBDAT |
| **General Call** | Yes | Yes |
| **Alert#** | No | Yes |

SMB está presente nas modernas placas-mãe dos PCs onde é usado para controlar o ligar/desligar da fonte de alimentação, os sensores de temperatura, o controle do ventilador, para ler a configuração dos DIMMs de memória e similares. Dispositivos Intel, como Intel Atom Processor E6xx Series, têm integrado, internamente, o controlador *host* SMBus, que atua como mestre e se comunica com dispositivos escravos na placa-mãe. A especificação SMBus 2.0 permite uma velocidade de barramento para um SMBus entre 10 Khz e 100 KHz (Barry & Crowley, 2012).

Duas características importantes do SMBus, que o distingue do I2C são: a existência da condição de ***Timeout***, que estabelece um limite de tempo para a resposta a um quadro e um sinal adicional de ***Alert#***. A linha de Alert# funciona

como uma linha de interrupção. Após receber um sinal de Alert#, o mestre deve enviar uma resposta ao alerta (*Alert Response*) para o endereço 0001 100. O disposto escravo que lançou o alerta deve, então, colocar no barramento seu endereço.

### 9.9.3 SPI - *Serial Peripheral Interface*

O ***Serial Peripheral Interface*** (SPI) é um barramento de quatro fios inventado pela Motorola. É composto por uma linha de *clock* (**SCLK**), uma linha de saída do mestre/entrada do escravo (**MOSI** – ***M**aster **O**utput-**S**lave **I**nput*), uma linha de entrada do mestre/saída do escravo (**MISO** – ***M**aster **I**nput-**S**lave **O**utput*) e uma linha de seleção de dispositivo (**SS** – ***S**lave **S**elect*). Figura 9.64 ilustra uma conexão típica do protocolo SPI.

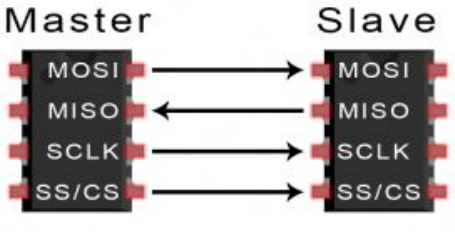



Figura 9.64: Conexão Típica SPI

A faixa de velocidade do barramento é muito maior do que a encontrada no I2C ou no SMBus; velocidades de até 80 MHz não são incomuns. Existem variantes que fornecem vários bits para a transferência (até 4). Esses bits de dados adicionais aumentam o desempenho.

O barramento SPI é usado, por exemplo, para conectar memória *flash* serial que fornece o código de inicialização inicial (***boot***) para plataformas Intel, como mostra a Figura 9.65. A taxa de transferência efetiva de uma interface periférica serial de 80 MHz e 4 pinos para um dispositivo flash NOR é de aproximadamente 40 megabytes por segundo. Isso é mais rápido que muitas interfaces paralelas para dispositivos flash NOR mais antigos.

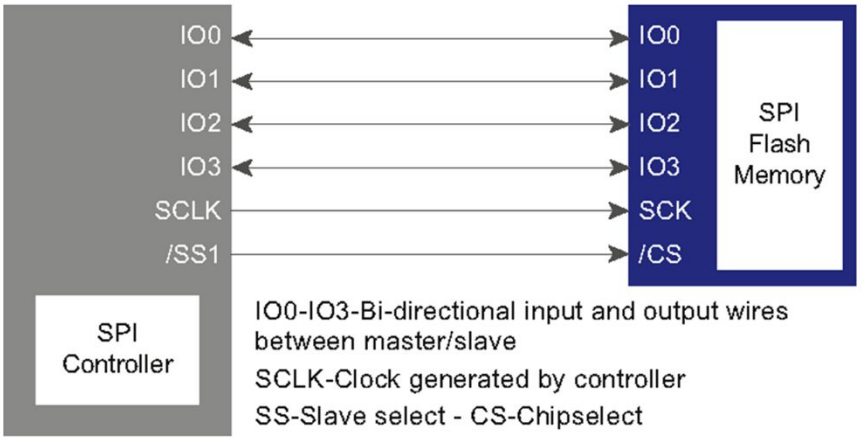



Figura 9.65: Conexão de uma memória compatível com SPI.

# 9.10 USB – Universal Serial Bus

**Porta Serial**, em inglês *Universal Serial Bus* (**USB**), é um padrão de indústria que estabelece especificações para cabos, conectores, e protocolos de comunicação para conexão, comunicação e provimento de energia entre computadores pessoais e seus dispositivos periféricos.

Conforme vimos na Seção 9.3.3, o padrão define uma topologia física em estrela hierárquica, em que todos os nós da rede são conectados a um dispositivo central chamado o ***host root hub***. Um dispositivo é endereçado por número atribuído pelo *host* na etapa de conexão, chamada de **Enumeração**. Os mecanismos usados para a comunicação Universal Serial Bus (USB) são *buffers* de dados chamados ***Endpoints***. Os *endpoints* existem no dispositivo e geralmente são implementados nos registros de sua CPU ou uma memória de porta dupla.

Os *endpoints* são unidirecionais, ou seja, cada *endpoints* é projetado para transmitir dados em uma direção. Os *endpoints* OUT enviam dados do host para o dispositivo. Os *endpoints* IN recebem dados do dispositivo destinado ao host. Os *endpoints* existem em pares numerados e cada par contém um terminal OUT e IN. O *endpoint* 0 (EP0IN e EP0OUT) é reservado em todos os dispositivos para fins de controle. Os *endpoints* de controle são usados durante a enumeração, fornecendo ao *host* a capacidade de ler as tabelas do descritor de dispositivos e enviar comandos de controle ao dispositivo durante a execução normal do aplicativo. *Endpoint* de controle suporta comandos do tipo: *set_address, set_configuration, get_descriptor.*

No descritor, estão informações padrões como: *Vendor identification*, *Device class*, *Power management capabilities*, *Endpoint description* com informações de

configurações. No Linux, o comando lsusb lista todos os dispositivos USB conectados ao host (o computador). Por exemplo, o comando de linha:

lsusb -t

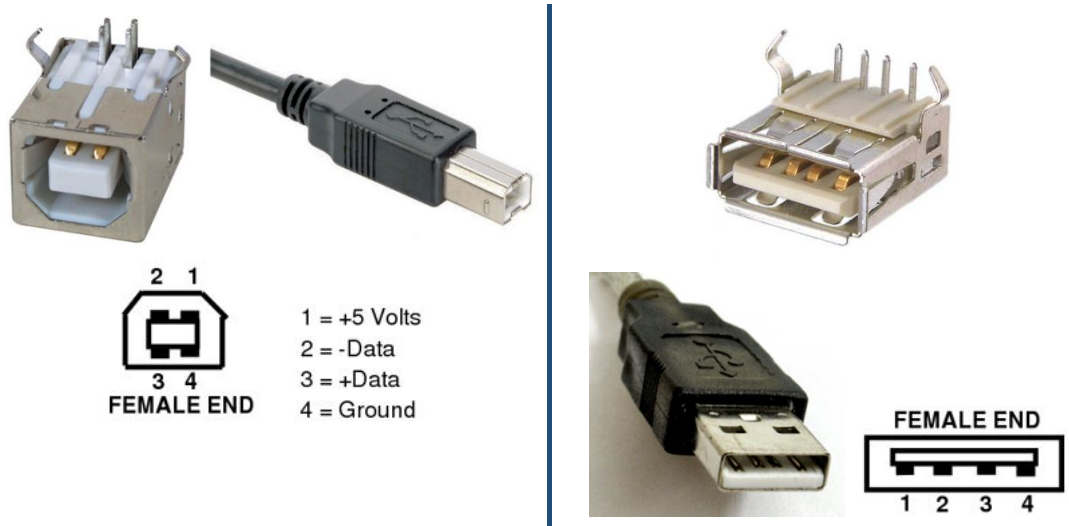
irá mostrar:

```
/:  Bus 02.Port 1: Dev 1, Class=root_hub, Driver=ehci_hcd/3p, 480M
    |__ Port 1: Dev 2, If 0, Class=hub, Driver=hub/8p, 480M
        |__ Port 6: Dev 4, If 0, Class=hub, Driver=hub/3p, 12M
            |__ Port 1: Dev 5, If 0, Class=HID, Driver=usbhid, 12M
            |__ Port 2: Dev 6, If 0, Class=HID, Driver=, 12M
            |__ Port 3: Dev 7, If 0, Class='bInterfaceClass 0xe0 not yet handled',
Driver=btusb, 12M
            |__ Port 3: Dev 7, If 1, Class='bInterfaceClass 0xe0 not yet handled',
Driver=btusb, 12M
            |__ Port 3: Dev 7, If 2, Class=vend., Driver=, 12M
            |__ Port 3: Dev 7, If 3, Class=app., Driver=, 12M
/:  Bus 01.Port 1: Dev 1, Class=root_hub, Driver=ehci_hcd/3p, 480M
    |__ Port 1: Dev 2, If 0, Class=hub, Driver=hub/6p, 480M
        |__ Port 4: Dev 3, If 0, Class='bInterfaceClass 0x0e not yet handled',
Driver=uvcvideo, 480M
        |__ Port 4: Dev 3, If 1, Class='bInterfaceClass 0x0e not yet handled',
Driver=uvcvideo, 480M
```

Existem 3 gerações de especificações USB: USB 1.0, USB 2.0 com múltiplas atualizações e USB 3.0.

## 9.10.1 USB 2.0 - Característica da comunicação

No padrão USB 2.0 cada conexão física é feita por 4 fios, sendo 2 de alimentação (+5volts e terra) e 2 fios de sinal diferencial (D+ e D-) (Figura 9.66). O receptáculo fornece uma tensão de +5 volts limitado a entre 100mA e 500mA.




*Conexão tipo B (receptáculo e plug)* *Conexão tipo A (receptáculo e plug)*

Figura 9.66: Conectores USB 2.0

Os dados são enviados usando uma codificação NRZI e uma sequência especial de sincronismo é enviada no início de todo pacote para permitir ao receptor recuperar o *clock* (Figura 9.65). **NRZI (*Non Return to Zero Inverted*)** é um método de codificar um sinal binário num sinal elétrico para transmissão num meio físico. Nele o nível lógico 0 é sinalizado pela transição do sinal e o nível lógico 1 pela ausência de transição do sinal (Figura 9.68). Para prevenir queda do sincronismo é necessário a inserção de um *bit* 0 após 6 *bits* iguais a 1 consecutivos, técnica conhecida como ***Bit Stuffing***.

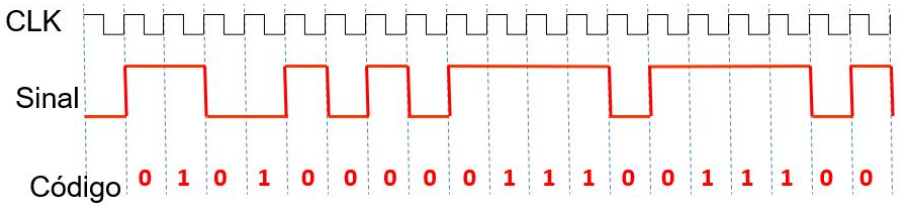



Figura 9.67: Exemplo do envio de um código segundo a codificação NRZI.

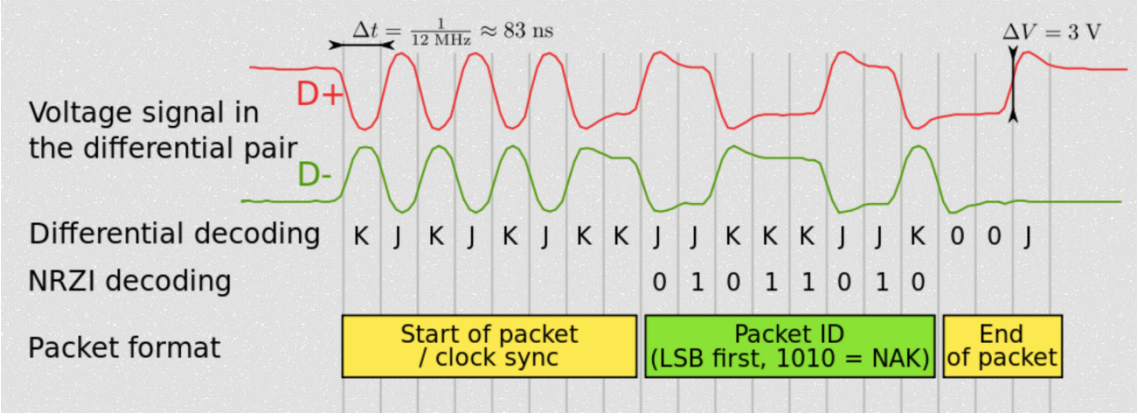



Figura 9.68: Sinalização USB (Fonte: wiki_usb,2019).

A comunicação *half-duplex* é feita pelos sinais, D+ e D- que formam um par trançado com impedância de 90Ω ± 15%. Entre estes dois sinais os níveis lógicos são representados da seguinte forma, uma tensão de 0.0 até 0.3volts representa o nível lógico 0 e uma tensão de 2.8 até 3.6 volts representa o nível 1. No *host* tem um resistor *pull-down* de 15kΩ. Se não tiver um dispositivo conectado, isto leva ao chamado estado "*single-ended zero*" (SE0). No dispositivo USB tem um resistor de *pull-up* de 1,5kΩ. Quando conectado, ele força as linhas de dados a um estado chamado "J" (Figura 9.68). Figura 9.69 apresenta o fluxo de controle do protocolo de transferência de dados do padrão USB 2.0.

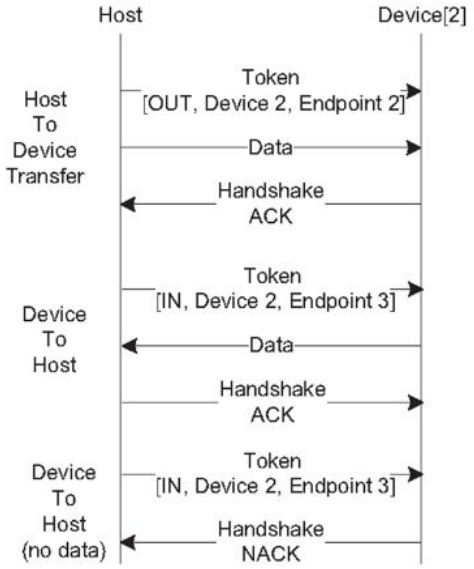



Figura 9.69: Protocolo de transferência de dados.

## 9.10.2 USB 2.0 – Tipos de transferência

Referem-se ao modo de comunicação usado entre o *host* e os pontos finais de um dispositivo. O tipo de transferência determina a frequência e a duração das transações usadas para se comunicar com o dispositivo final. Os tipos de transferência também podem atribuir uma CRC (código de verificação de erro por redundância cíclica) a ser verificada com cada pacote transmitido. Os tipos de transferência são definidos pelo dispositivo e comunicados ao *host* durante o processo de enumeração na etapa de conexão do dispositivo (microchip_usb, 2019).

Distinguem-se os seguintes tipos de transferência:

***Control transfer*** do *host* para o dispositivo informando algum comando de controle ou de configuração.

***Bulk Data Transfer***, (transferência em massa) que são pacotes curtos com CRC. As transferências em massa não são agendadas, elas são executadas quando há

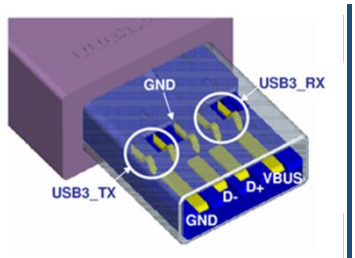
largura de banda disponível no quadro. Várias transferências em massa podem ser executadas em um único quadro se houver largura de banda disponível. São usadas por exemplo em transferência com dispositivos de armazenamento e impressoras.

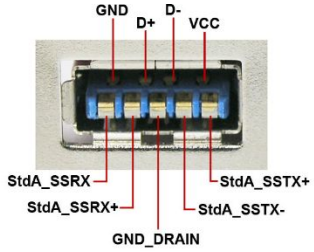
***Interrupt Data Transfer*** que são pacotes curtos com CRC, agendados em intervalos periódicos fixos, por exemplo, a cada 10 quadros. As transferências de interrupção sempre ocorrerão, independentemente de haver ou não dados a serem transferidos.

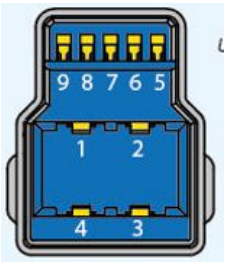
***Isochronous Data Transfer*** que são pacotes mais longos sem CRC. Transferências isócronas são agendadas em períodos fixos. São usados, por exemplo, para comunicação de áudio/vídeo ou emulação de portas seriais.

## 9.10.3 USB 3.0

A versão USB 3.0 aumentou a taxa de transferência para 4,8 Gbps e introduziu novos sinais nos conectores. Novas linhas de dados passam a existir, sendo mais dois pares de linhas diferenciais de dados, **SSRX+/SSRX-** para recepção pelo *host* e **SSTX+/SSTX-** para a transmissão do *host* para o dispositivo. Estes dois pares permitem a transferência *Full-duplex.* O par de sinais D+ e D- das versões anteriores permanecem por compatibilidade. Nesta nova versão novos conectores são definidos e para identificá-los é padronizado que tenham **cor azul**. A Figura 9.70 exibe todos os conectores e suas pinagens compatíveis a norma USB 3.0.

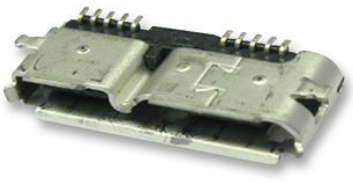



USB type B Pinout, Cable Assembly

| Pin | Signal Name | Description |
|---|---|---|
| 1 | VBUS | Red |
| 2 | D- | White |
| 3 | D+ | Green |
| 4 | GND | Black |
| 5 | StdA_SSTX- | Blue |
| 6 | StdA_SSTX+ | Yellow |
| 7 | GND_DRAIN | GROUND |
| 8 | StdA_SSRX- | Purple |
| 9 | StdA_SSRX+ | Orange |
| Shell | Shield | Connector Shell |




Micro-B USB 3.0 plug

1. Power (VBUS)
2. USB 2.0 differential pair (D−)
3. USB 2.0 differential pair (D+)
4. USB OTG ID for identifying lines
5. GND
6. USB 3.0 signal transmission line (−)
7. USB 3.0 signal transmission line (+)
8. GND
9. USB 3.0 signal receiving line (−)
10. USB 3.0 signal receiving line (+)

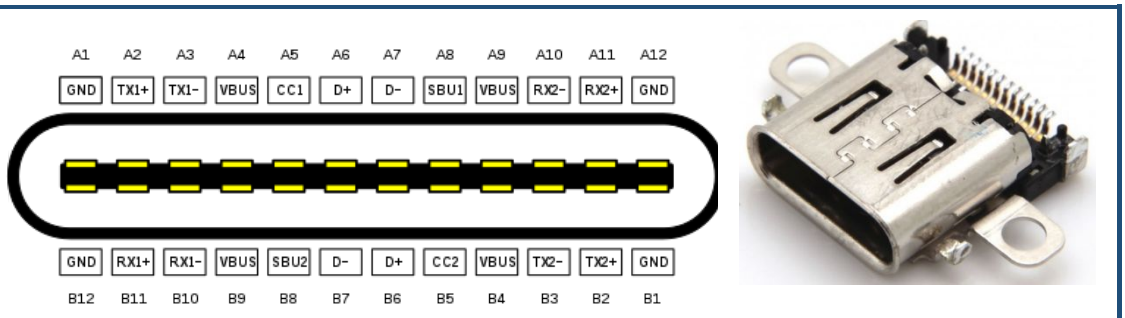
source : wikipedia.org

| Plug e receptáculo USB 3.0 tipo A | Plug e receptáculo USB 3.0 tipo B | Micro plug e micro receptáculo USB 3.0 |
|---|---|---|

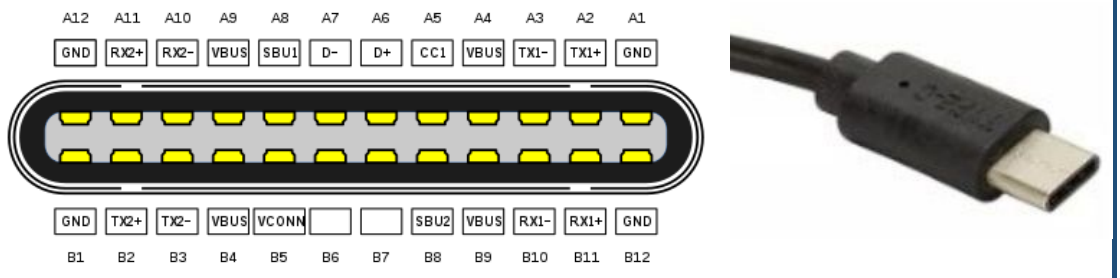
Figura 9.70: Conectores USB 3.0

## 9.10.4 Conector USB tipo C

Visando simplificar e reduzir o número de conectores, a partir dos sinais do USB 3.0, o fórum do USB propôs um terceiro tipo de conexão, chamado tipo C (Figura 9.71). Este conector permite o aumento do desempenho, suportando velocidades até 10 Gbps, permite liberar mais potência para alimentação, permitindo fornecer até 100 W. Além de ser mais conveniente para o usuário, permitindo que o encaixe possa ser feito de qualquer forma (direto ou invertido).




Pinagem e o receptáculo USB-C. Vista Frontal




Pinagem e o plug USB-C. Vista Frontal

Figura 9.71: Receptáculo e plug USB tipo C

## 9.10.5 Programação USB

Para a programação de aplicações usando a interface USB podem ser usadas várias bibliotecas de programas. Por exemplo, no **ambiente Windows** têm-se a biblioteca **libusb-win32**, disponível em https://sourceforge.net/p/libusb-win32/wiki/Home/ (acessado em setembro de 2019).

Para a programação em **ambiente Arduino** está disponível em *github*, por exemplo, o trabalho de Oleg Mazurov (felis), disponível em https://github.com/felis/USB_Host_Shield_2.0 (acessado em setembro 2019).

Para a programação em **ambiente Linux** estão disponíveis diversas bibliotecas dentro das várias distribuições. Por exemplo, para a distribuição Ubuntu pode ser instalada a biblioteca **libusb** através de "sudo apt-get install libusb-1.0-0-dev".

## 9.10.6 Resumo das versões USB

A Tabela 9.8 resume as diversas versões do padrão USB desde o seu lançamento até a mais recente. A Tabela 9.9 apresenta a capacidade de fornecimento de potência dos cabos de conexão USB usados para, por exemplo, carregamento de dispositivos.

Tabela 9.8: Resumo das versões

| Nome | Data | Taxa de transferência máxima | Nota |
|---|---|---|---|
| USB 0.8 | 12/1994 | | Prerelease |
| USB 0.9 | 04/1995 | | Prerelease |
| USB 0.99 | 08/1995 | | Prerelease |
| USB 1.0-RC | 11/1995 | | Release Candidate |
| USB 1.0 | 01/1996 | Full Speed (12 Mbit/s) | |
| USB 1.1 | 08/1998 | Full Speed (12 Mbit/s)[I] | |
| USB 2.0 | 04/2000 | High Speed (480 Mbit/s) | |
| USB 3.0 | 11/2008 | SuperSpeed (5 Gbit/s) | Também referenciado como USB 3.1 Gen 1 e USB 3.2 Gen 1x1 |
| USB 3.1 | 07/2013 | SuperSpeed+ (10 Gbit/s) | Também referenciado como USB 3.1 Gen 2 e USB 3.2 Gen 2x1 |
| USB 3.2 | 09/2017 | SuperSpeed+ (20 Gbit/s) | Inclue novos USB 3.2 Gen 1x2 e USB 3.2 Gen 2x2 multi-link modes |
| USB 4.0 | 07/2019 | 40Gbits/s | Compatível com Thunderbolt 3 |

Tabela 9.9: Resumo de potência máxima dos cabos

| Nome | Data | Potência Máx. | Nota |
|---|---|---|---|
| USB Battery Charging 1.0 | 2007-03-08 | 5 V, 1.5 A | |
| USB Battery Charging 1.1 | 2009-04-15 | | |
| USB Battery Charging 1.2 | 2010-12-07 | 5 V, 5 A | |
| USB Power Delivery revision 1.0 (version 1.0) | 2012-07-05 | 20 V, 5 A | Uso do protocol FSK sobre o bus power ($V_{BUS}$) |
| USB Power Delivery revision 1.0 (version 1.3) | 2014-03-11 | | |
| USB Power Delivery revision 2.0 (version 1.0) | 2014-08-11 | 20 V, 5 A | Uso do protocolo BMC sobre o canal de comunicação (CC) nos cabos tipo-C. |
| USB Power Delivery revision 2.0 (version 1.1) | 2015-05-07 | 20 V, 5 A | |
| USB Power Delivery revision 2.0 (version 1.2) | 2016-03-25 | 20 V, 5 A | |
| USB Power Delivery revision 3.0 (version 1.1) | 2017-01-12 | 20 V, 5 A | |
| USB Type-C 1.0 | 2014-08-11 | 5 V, 3 A | Nova especificação de conector e cabo |
| USB Type-C 1.1 | 2015-04-03 | 5 V, 3 A | |

# 9.11 Referências


[1] Alan Clements. Microprocessor Systems Design: 68000 Hardware, Software, and Interfacing. ISBN 0-534-94822-7

[2] Microcomputer Interfacing. https://slideplayer.com/slide/6953893/.

[3] Gustavson,David B. "*Computer Buses – A Tutorial*" IEEE Micro Agosto 1984.

[4] Thurber, K.J. *et al.* "*A Systematic Approach to the Design of Digital Busing Structure*" AFIPS Conf. Proc. Vol. 41, Part II 1972.

[5] Wikipedia. RS-232. https://en.wikipedia.org/wiki/RS-232

[6] https://sites.google.com/site/johnkneenmicrocontrollers/serialcoms.

[7] Texas Instruments. SNx4LS2x Datasheet.
http://www.ti.com/lit/ds/symlink/sn54ls240-sp.pdf
[8] Motorola. SN74LS245 Datasheet. http://ecee.colorado.edu/~mcclurel/sn74ls245rev5.pdf
[9] National Instruments. Interfacing TTL and CMOS Circuits.
https://knowledge.ni.com/KnowledgeArticleDetails?id=kA00Z0000019MNOSA2&l=pt-BR
[10] Wikipedia. Southbridge. https://pt.wikipedia.org/wiki/Southbridge
[11] Wikipedia. PCI Express.
https://en.wikipedia.org/wiki/PCI_Express
[12] Hilbert Hagedoom. Core i7 4790K Processor Review - PCH - The Platform Controller Hub. https://www.guru3d.com/articles-pages/core-i7-4790k-processor-review,7.html
[13] Wikipedia. Flexible Display Interface.
https://en.wikipedia.org/wiki/Flexible_Display_Interface
[14] Wikipedia. Direct Media Interface. https://en.wikipedia.org/wiki/Direct_Media_Interface

**(Barry & Crowley, 2012).** Peter Barry, Patrick Crowley. Modern Embedded Computing – Designing Connected, Pervasive, Media-Rich Systems. Morgan Kaufmann Elsevier. 225 Wyman Street, Waltham, MA 02451, USA. 2012. ISBN: 978-0-12-391490-3;

**(Maxim, 2000)** Maxim Integrated Products, Inc. APPLICATION NOTE 476 – Comparing the I²C Bus to the SMBus. 2000. Disponível em: https://www.maximintegrated.com/en/app-notes/index.mvp/id/476. Acessado em setembro, 2019;

**(Microchip_usb, 2019)** USB Transfer Types. Disponível em: https://microchipdeveloper.com/usb:transfer. Acessado em setembro de 2019;

(**SMB, 2018**). System Management Interface Forum, Inc. System Management Bus (SMBus) Specification. 2018;

(**TI, 2016**). Texas Instruments Incorporated. SMBus Made Simple Application Report SLUA475. November 2016;