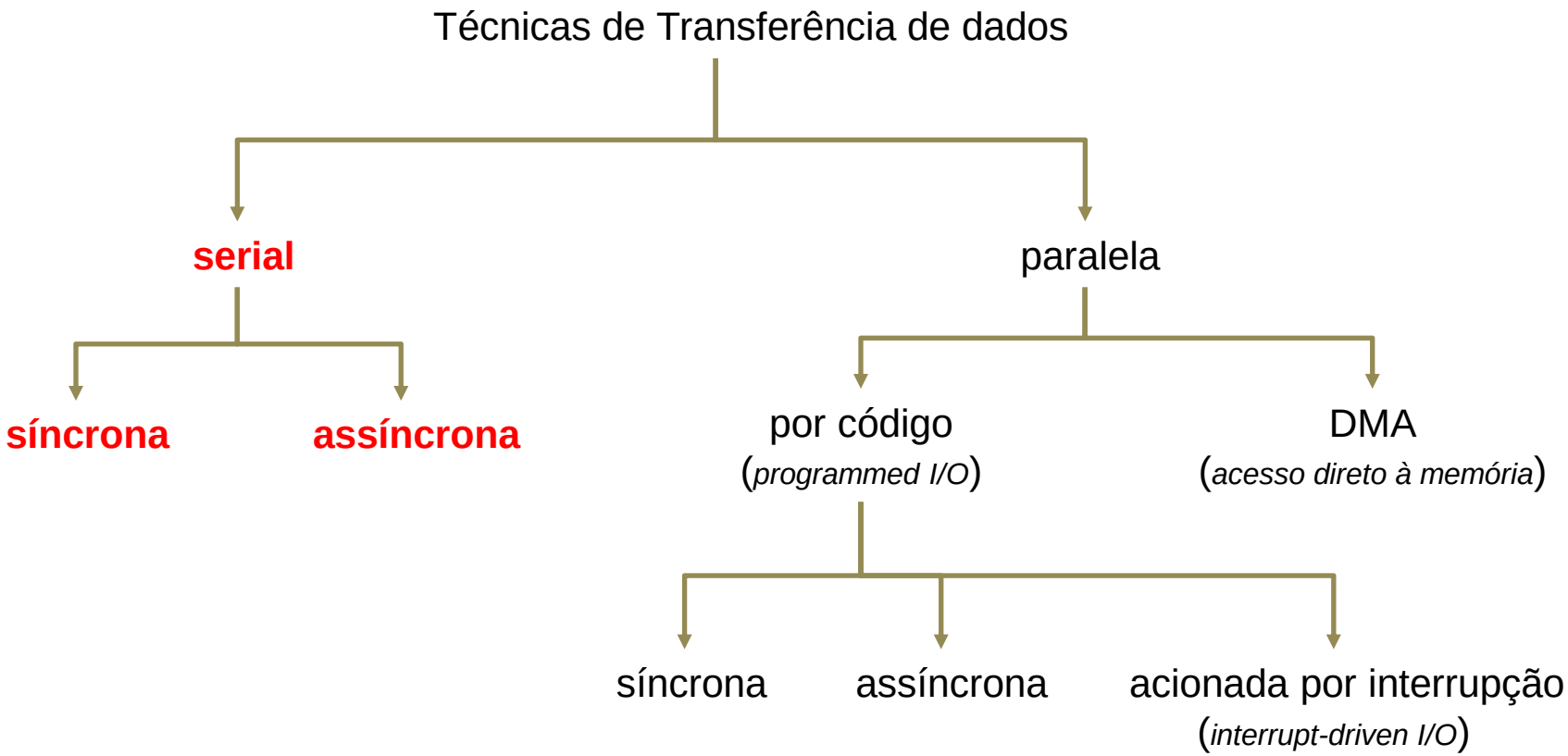
- Núcleo do ARM Cortex – hardware...

(Registradores, ULA, GPIOs, pipeline, barramentos e interconexão com vários periféricos)

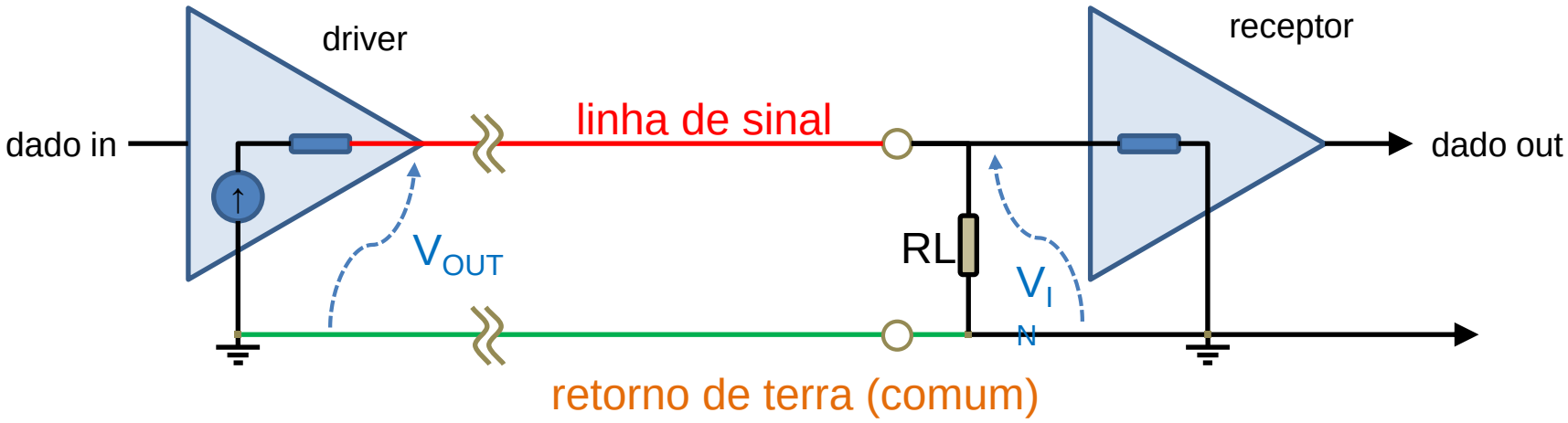
- Conceitos de programação – e.g. superloop, funções, assembly...
- Conceito de Interrupção – programação do NVIC
- Programação de periféricos (GPIO, timers, Systick, PWM...)
- *Comunicação paralela - falamos de barramentos (interfaceamento paralelo)*

- HOJE: comunicação entre sistemas processados (comunicação entre eletrônica embarcada)
- Comunicação serial...



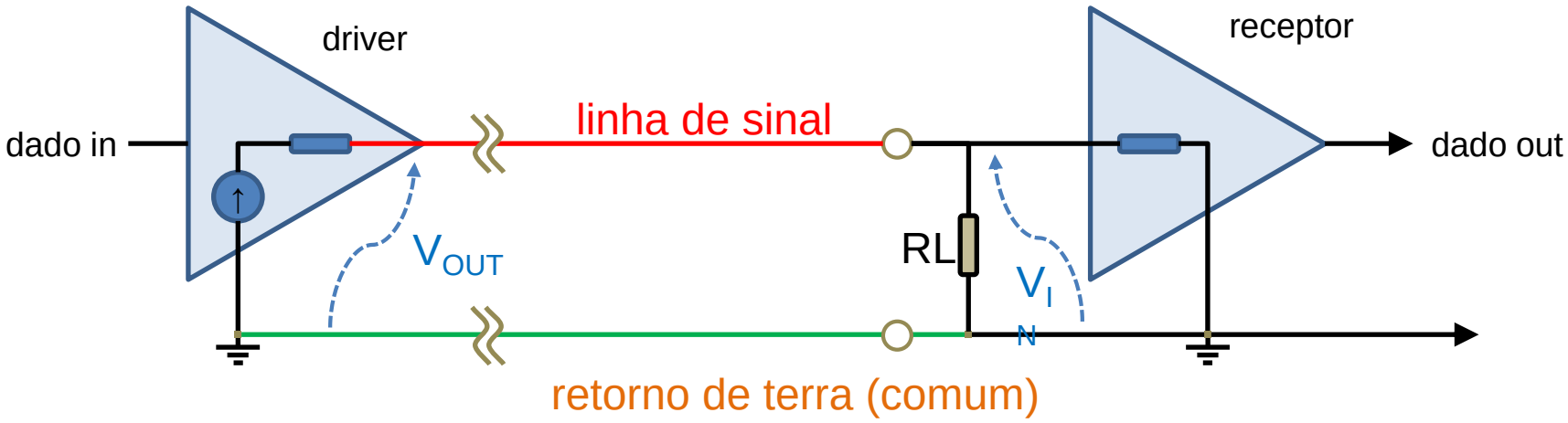
- A ideia de protocolos seriais é enviar dados rapidamente entre dispositivos;
- Alguns protocolos (SPI, I^2C) são projetados para curtas distâncias (dentro de uma caixa), com baixa complexidade e baixo custo;
- Outros são projetados para enviar dados em longas distâncias e alta velocidade (mais caros, alta confiabilidade e alto desempenho).

- **Single-ended** signaling:

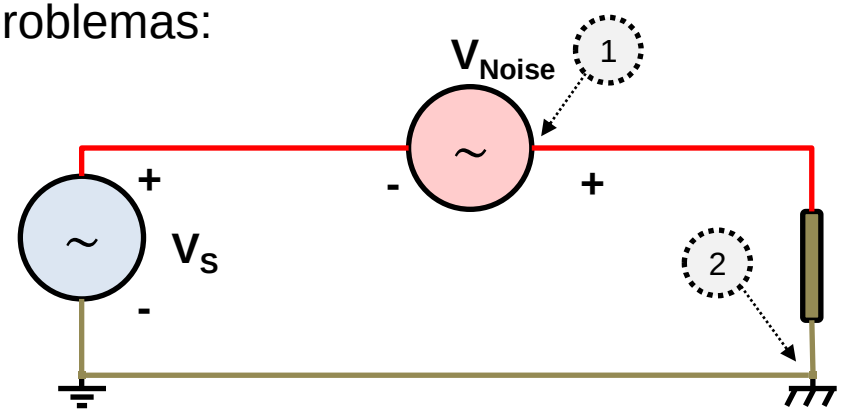


Quando você ver o termo "*Single-Ended*" é porque o sinal de terra é comum entre os dois sistemas eletrônicos.

- **Single-ended** signaling:



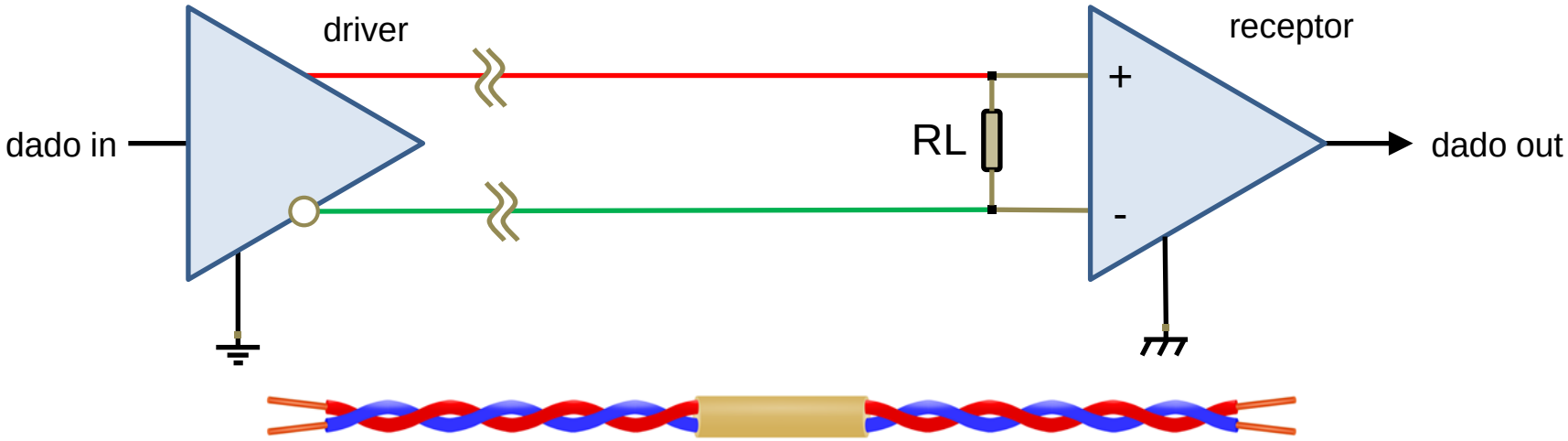
- 2 problemas:



1) ruídos podem ser induzidos na linha de sinal em relação ao terra;

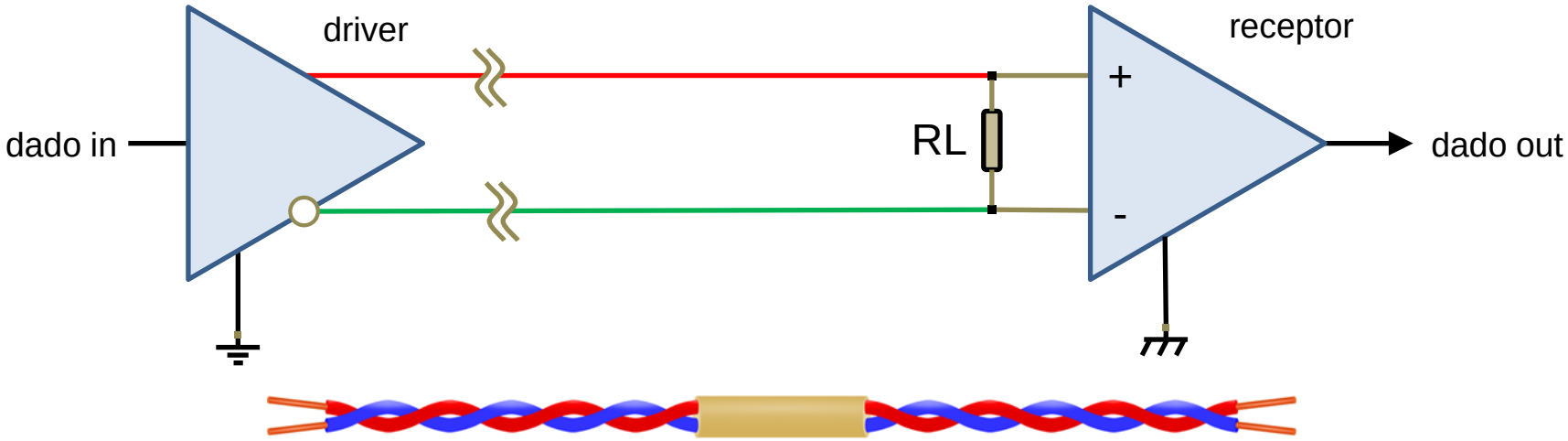
2) terras em diferentes dispositivos podem não ser os mesmos (*distâncias longas introduzem impedâncias na interconexão dos terras*)

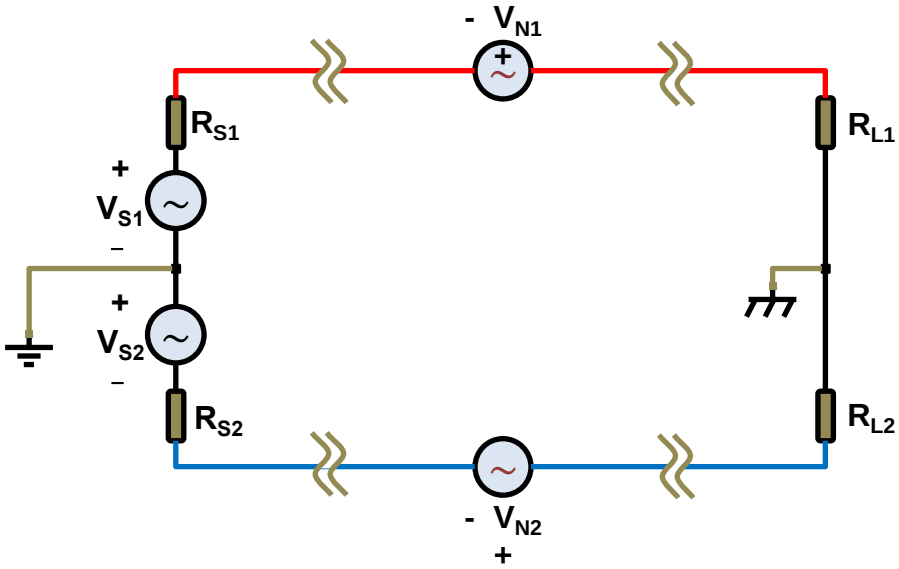
- **Diferential signaling** (balanced sinaling):



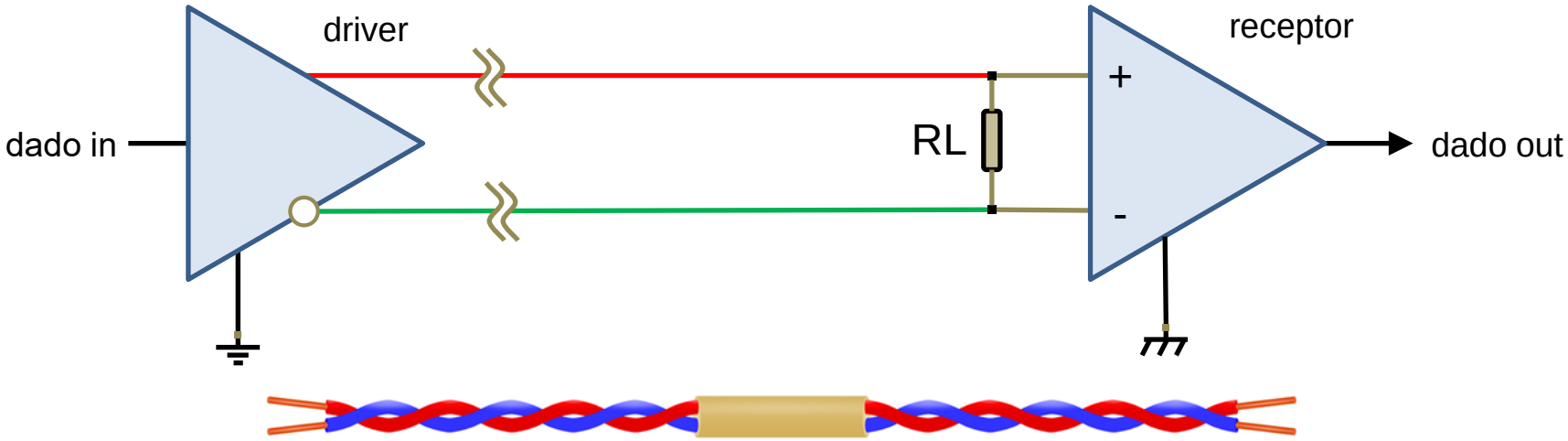
Quando você ver o termo "*Diferential*" é porque o sinal é mandado por dois fios, formando um sinal complementar – ou diferencial.

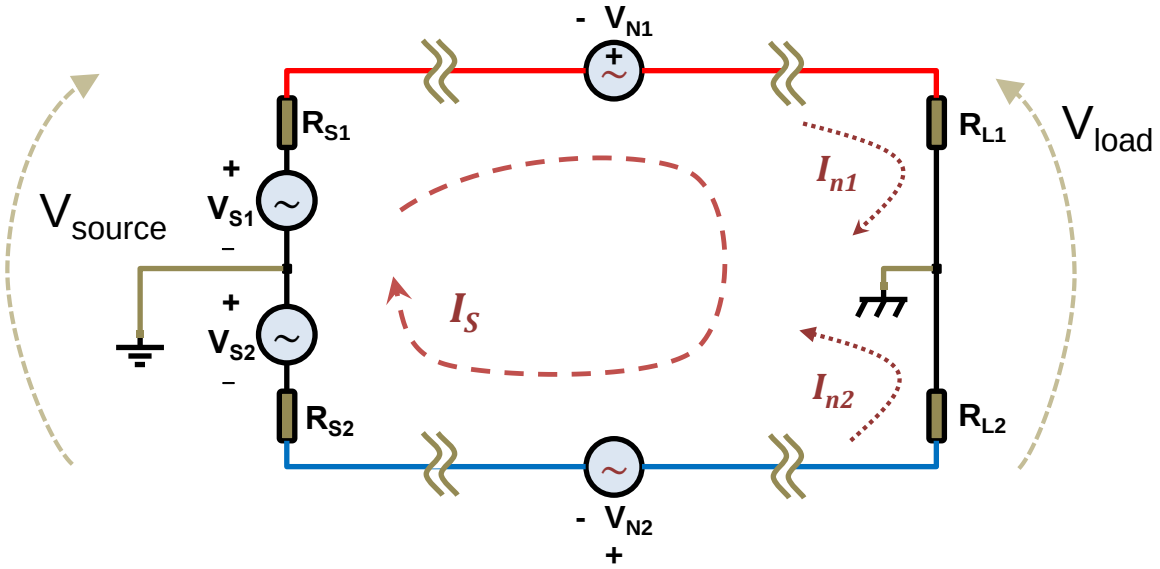
➢ **Diferential signaling** (balanced sinaling):





➢ **Diferential signaling** (balanced sinaling):





$$V_L = I_{n1}R_{L1} - I_{n2}R_{L2} + I_S(R_{l1}+R_{l2})$$

$R_{L1} = R_{L2}$
Usando par trançado:
$V_{N1} = V_{N2}$
portanto:
$I_{n1} = I_{n2}$

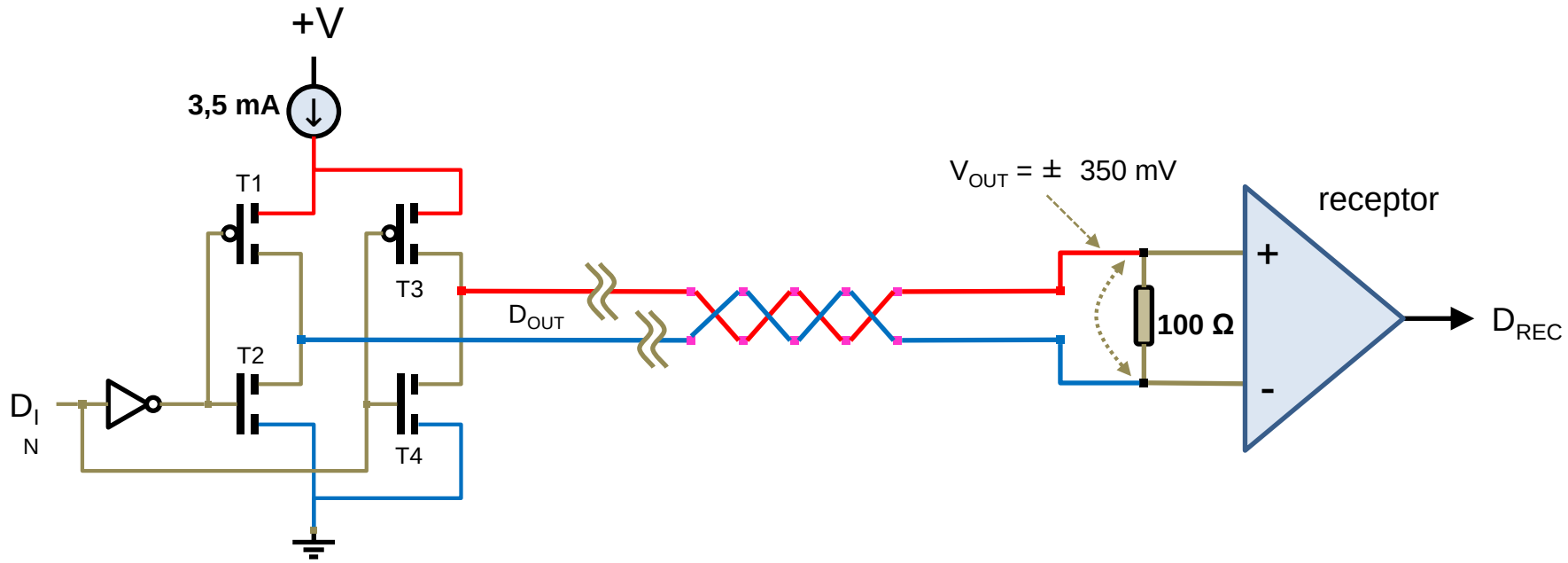
$$V_L = I_S(R_{l1}+R_{l2}) = 2R_L$$

- **Diferential signaling** (balanced sinaling):

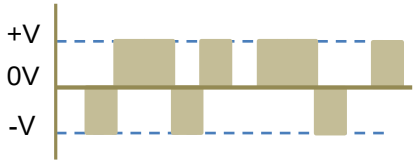
Vários padrões no mercado:

- RS-422 / 485
- USB
- IEEE 1394 (Firewire - High Performance Serial Bus/HPSB)
- Ethernet (IEEE 802.3x // 10B-T +2,5 -2,5 // 100B-T 1V, 0V, -1V // 1000B-T -2, -1, 0 , 1, +2V)

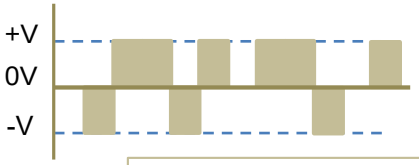
- LVDS *(low voltage differential signal) muito usado para saída de FPGA e de SoC!*



- *Line coding*: técnica para converter ***dados digitais*** em ***sinais digitais***
- Características importantes da codificação de sinais digitais:
  - ✓ Diferença entre data level & signal level (2 níveis ou múltiplos níveis de tensão)



- *Line coding*: técnica para converter ***dados digitais*** em ***sinais digitais***
- Características importantes da codificação de sinais digitais:
  - ✓ Diferença entre data level & signal level (2 níveis ou múltiplos níveis de tensão)

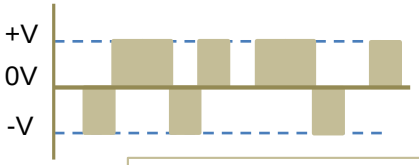


  - ✓ Bandwidth e taxa de transmissão (data rate)
  - ✓ Bit rate x baud rate

$$I = 2.B.log_2 M$$

B = bandwidth
M = num de níveis do sinal
I = quantidade de informação transmitida
Bit rate = num dados / seg
Baude rate = num sinais / seg

- *Line coding*: técnica para converter ***dados digitais*** em ***sinais digitais***
- Características importantes da codificação de sinais digitais:
  - ✓ Diferença entre data level & signal level (2 níveis ou múltiplos níveis de tensão)



  - ✓ Bandwidth e taxa de transmissão (data rate)

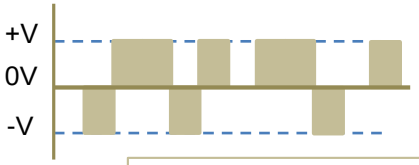
$$I = 2.B.log_2 M$$

B = bandwidth
M = num de níveis do sinal
I = quantidade de informação transmitida
Bit rate = num dados / seg
Baude rate = num sinais / seg

  - ✓ Bit rate x baud rate
  - ✓ Componente contínuo (sucessão de “111111” ou de “000000” inserem componentes DC no sinal)

- *Line coding*: técnica para converter ***dados digitais*** em ***sinais digitais***

- Características importantes da codificação de sinais digitais:
  - ✓ Diferença entre data level & signal level (2 níveis ou múltiplos níveis de tensão)

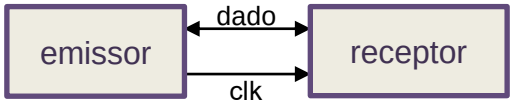


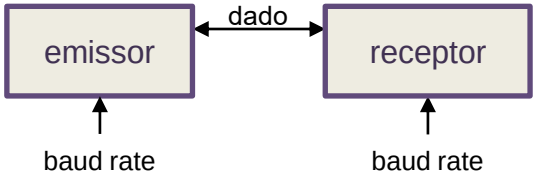
  - ✓ Bandwidth e taxa de transmissão (data rate)

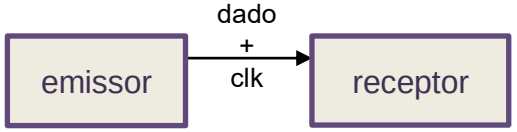
$$I = 2.B.log_2 M$$

B = bandwidth
M = num de níveis do sinal
I = quantidade de informação transmitida
Bit rate = num dados / seg
Baude rate = num sinais / seg

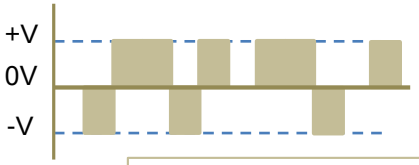
  - ✓ Bit rate x baud rate
  - ✓ Componente contínuo (sucessão de “111111” ou de “000000” inserem componentes DC no sinal)
  - ✓ Sincronização







- *Line coding*: técnica para converter ***dados digitais*** em ***sinais digitais***
- Características importantes da codificação de sinais digitais:
  - ✓ Diferença entre data level & signal level (2 níveis ou múltiplos níveis de tensão)

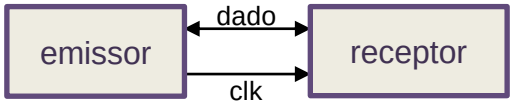


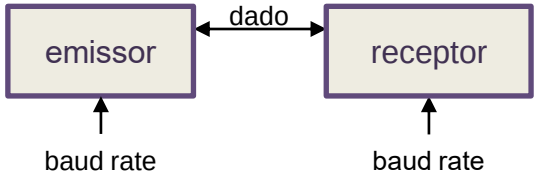
  - ✓ Bandwidth e taxa de transmissão (data rate)

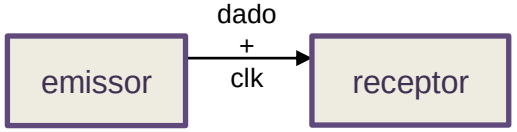
$$I = 2.B.log_2 M$$

B = bandwidth
M = num de níveis do sinal
I = quantidade de informação transmitida
Bit rate = num dados / seg
Baude rate = num sinais / seg

  - ✓ Bit rate x baud rate
  - ✓ Componente contínuo (sucessão de “111111” ou de “000000” inserem componentes DC no sinal)
  - ✓ Sincronização

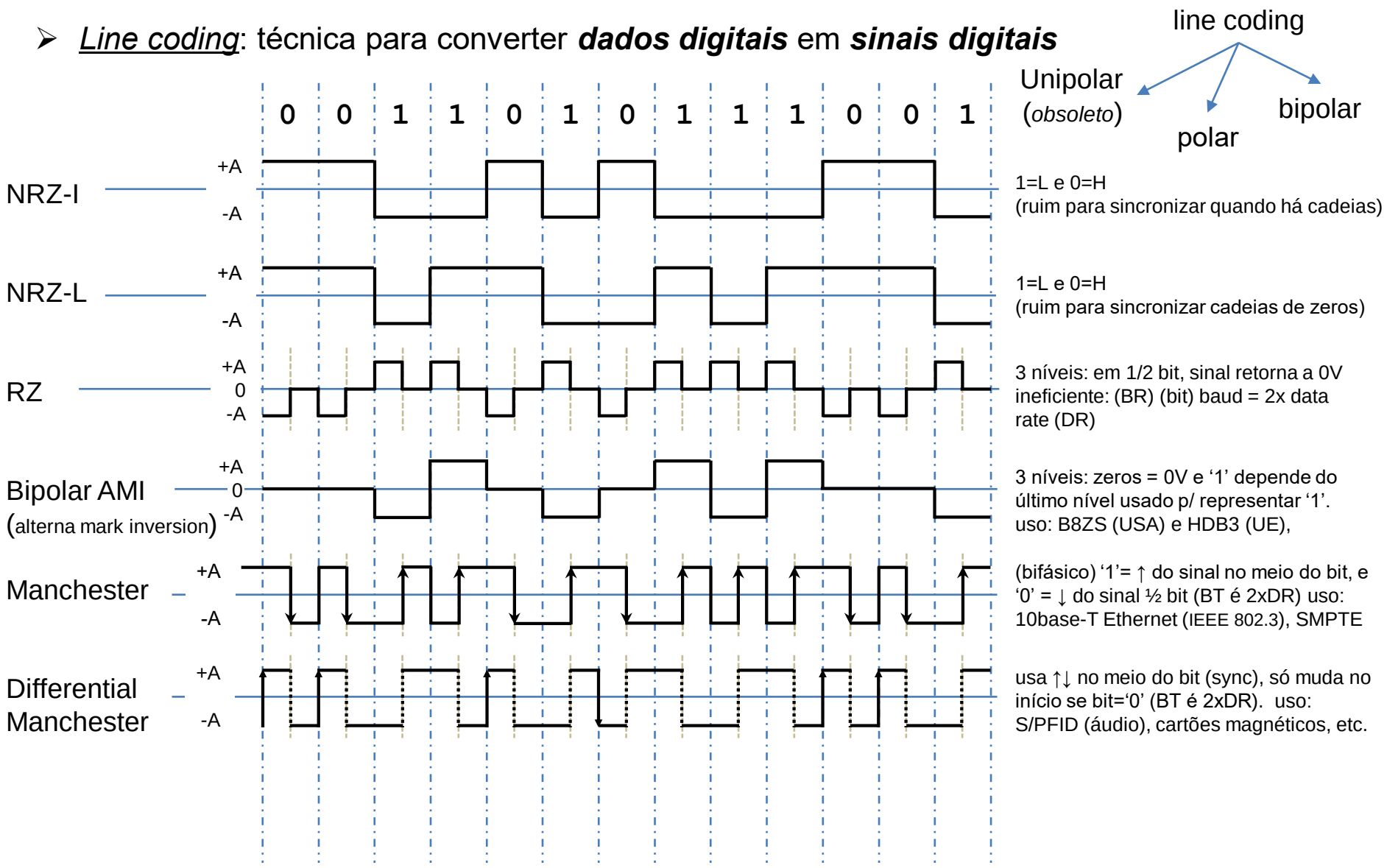






  - ✓ Detecção de erros
  - ✓ Custo de implementação

➢ *Line coding*: técnica para converter ***dados digitais*** em ***sinais digitais***



line coding

- Unipolar (*obsoleto*)
- polar
- bipolar

1=L e 0=H
(ruim para sincronizar quando há cadeias)

1=L e 0=H
(ruim para sincronizar cadeias de zeros)

3 níveis: em 1/2 bit, sinal retorna a 0V
ineficiente: (BR) (bit) baud = 2x data rate (DR)

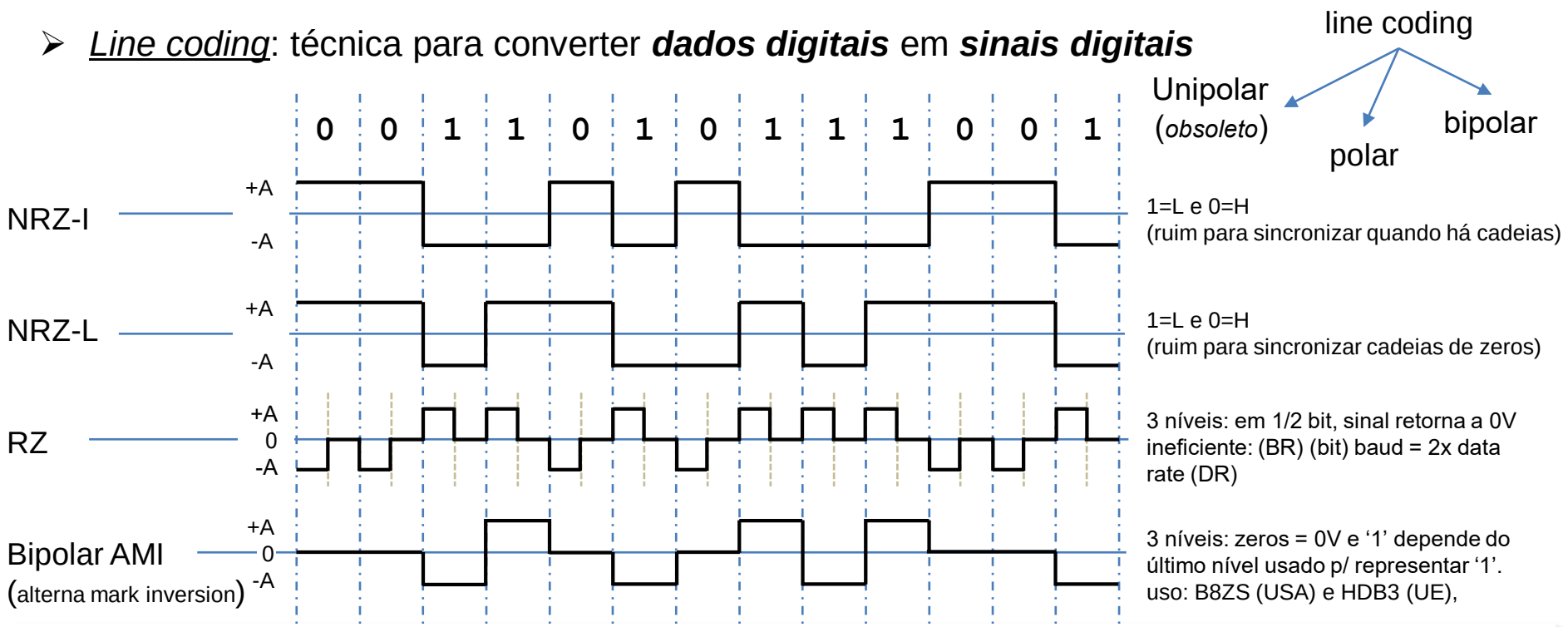
3 níveis: zeros = 0V e '1' depende do último nível usado p/ representar '1'.
uso: B8ZS (USA) e HDB3 (UE),

(bifásico) '1'= ↑ do sinal no meio do bit, e '0' = ↓ do sinal ½ bit (BT é 2xDR) uso: 10base-T Ethernet (IEEE 802.3), SMPTE

usa ↑↓ no meio do bit (sync), só muda no início se bit='0' (BT é 2xDR). uso: S/PFID (áudio), cartões magnéticos, etc.

ver - https://youtu.be/ifgs0uypC78 - comparações de bandas passantes, energia, relação sinal x ruido

➢ *Line coding*: técnica para converter ***dados digitais*** em ***sinais digitais***



line coding

Unipolar (*obsoleto*)

polar

bipolar

1=L e 0=H
(ruim para sincronizar quando há cadeias)

1=L e 0=H
(ruim para sincronizar cadeias de zeros)

3 níveis: em 1/2 bit, sinal retorna a 0V
ineficiente: (BR) (bit) baud = 2x data rate (DR)

3 níveis: zeros = 0V e '1' depende do último nível usado p/ representar '1'.
uso: B8ZS (USA) e HDB3 (UE),

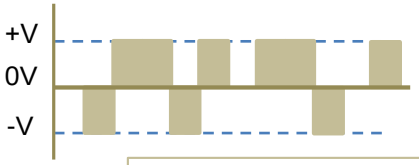
BIPOLAR AMI é mais eficiente porque data-rate = baud rate... Problema é o componente DC gerado por sucessivos zeros.

➢ **B8ZS (bipolar com substituição de 8 bits zeros - USA)**
bAMI // substitui 8 zeros seguidos por esquema de violação (tabela) // recuperar clock – requer PLL com capacidade de até 7 ciclos sem dair do sincronismo

➢ **HDB3 (high-density bipolar-3 zeros – EU e Japão)**
bAMI // substitui 4 zeros seguidos por esquema de violação (tabela) // recupera clock – requer PLL com capacidade de 3 ciclos sem sair do sincronismo
Codificação por blocos Block coding – introduzidos para aumentar o desempenho da codificação por linha (line coding)
4B/5B onde 4 bits são codificados no espaço de 5 bits, introduz redundância para conseguir sincronização, corrige erro (de forma limitada)

ver - https://youtu.be/ifgs0uypC78 - comparações de bandas passantes, energia, relação sinal x ruido

- *Line coding*: técnica para converter ***dados digitais*** em ***sinais digitais***
- Características importantes da codificação de sinais digitais:
  - ✓ Diferença entre data level & signal level (2 níveis ou múltiplos níveis de tensão)

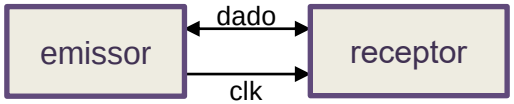


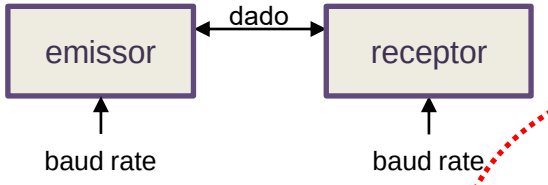
  - ✓ Bandwidth e taxa de transmissão (data rate)

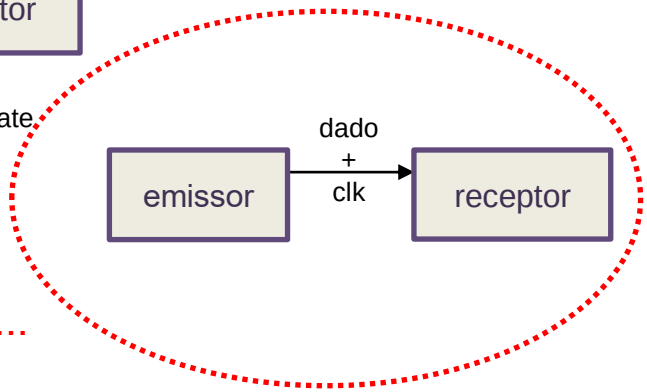
$$I = 2.B.log_2 M$$

B = bandwidth
M = num de níveis do sinal
I = quantidade de informação transmitida
Bit rate = num dados / seg
Baude rate = num sinais / seg

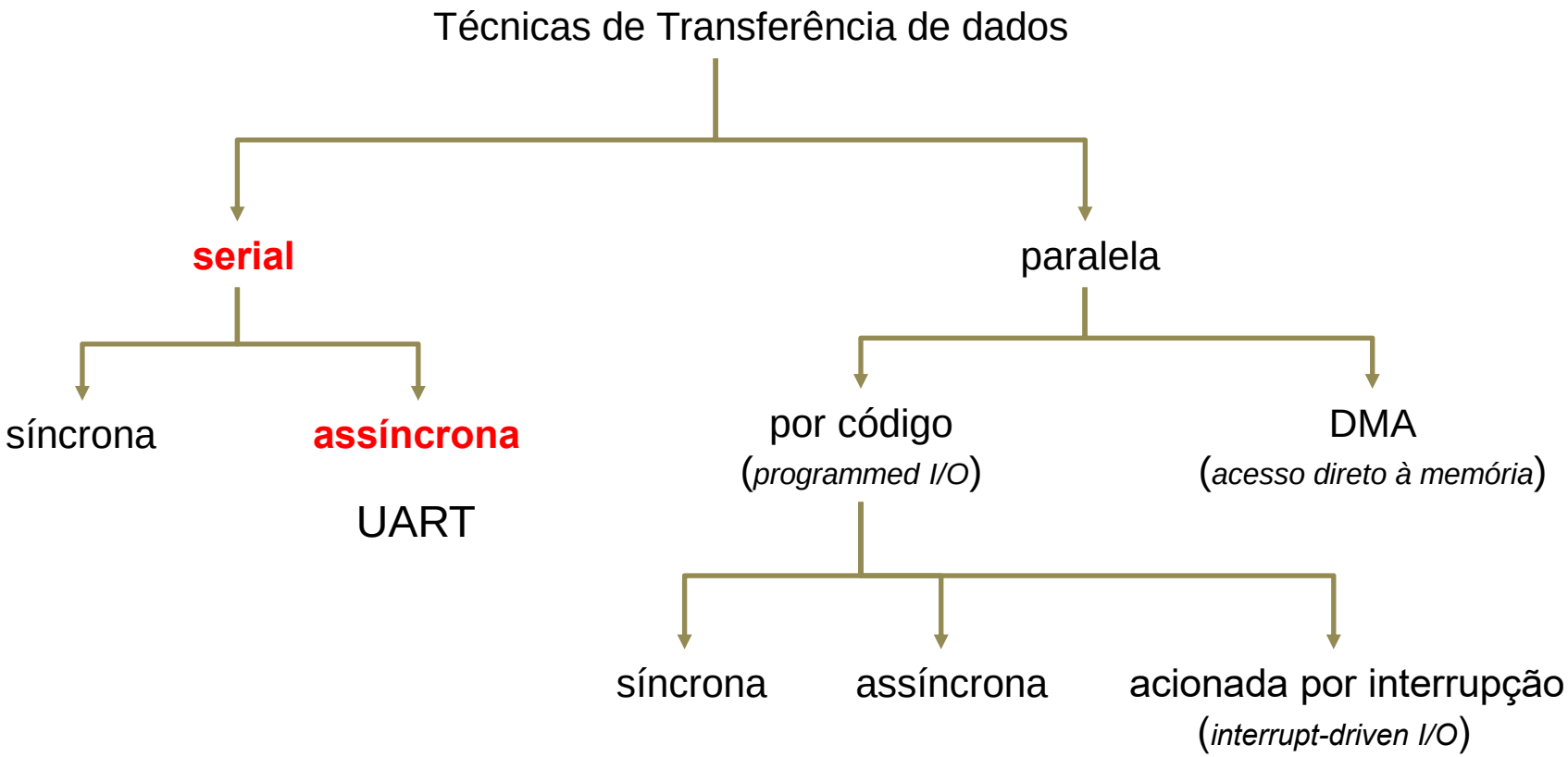
  - ✓ Bit rate x baud rate
  - ✓ Componente contínuo (sucessão de “111111” ou de “000000” inserem componentes DC no sinal)
  - ✓ Sincronização







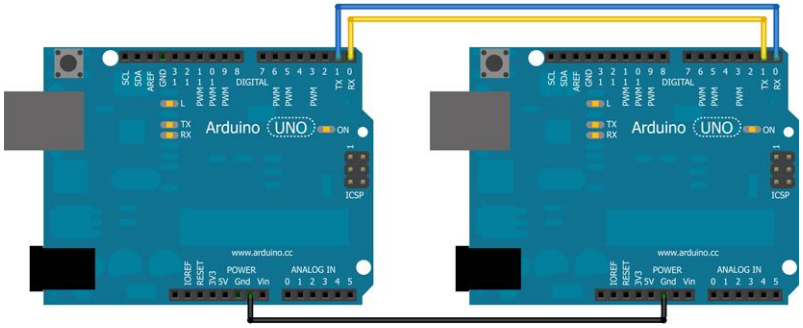
Os slides anteriores se aplicam a esse caso...


Técnicas de Transferência de dados
serial
paralela
síncrona
assíncrona
UART
por código
(programmed I/O)
DMA
(acesso direto à memória)
síncrona
assíncrona
acionada por interrupção
(interrupt-driven I/O)

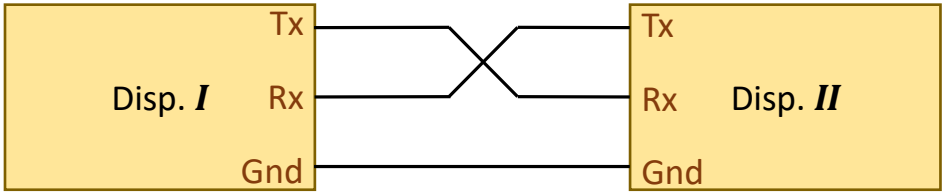
Interface serial UART / USART (overview)

➢ Transmissor e Receptor Assíncrono

Recebe e transmite (bidirecional) dados de forma não sincronizada!

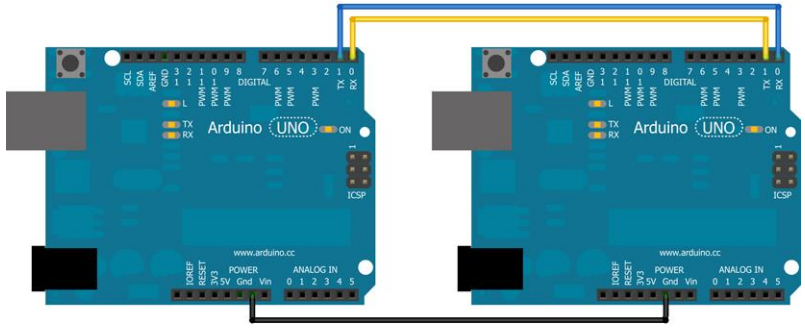


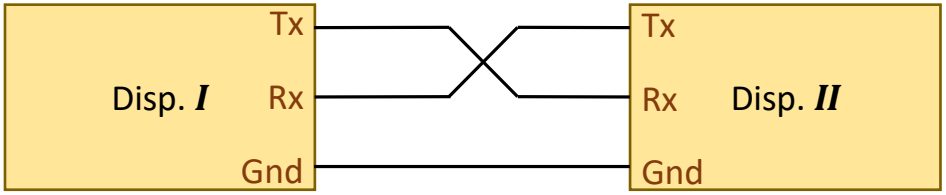
Processador 1 Processador 2

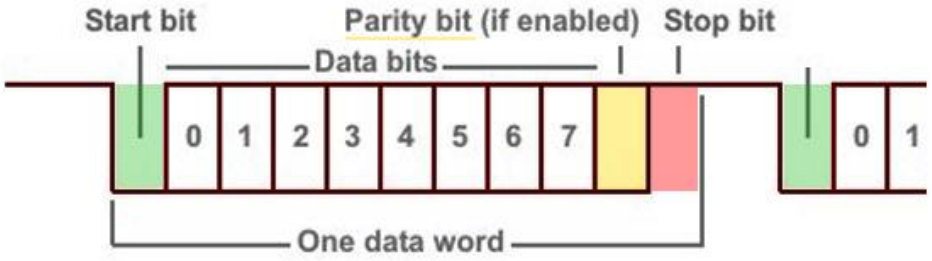


➢ Transmissor e Receptor Assíncrono

Recebe e transmite (bidirecional) dados de forma não sincronizada!



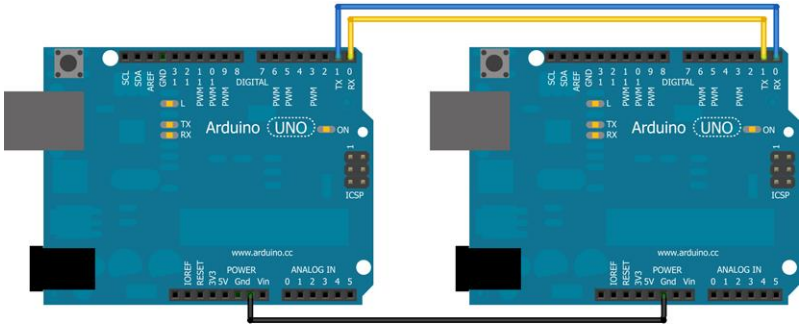




procotolo de transmissão de uma palavra (byte)

➢ Transmissor e Receptor Assíncrono

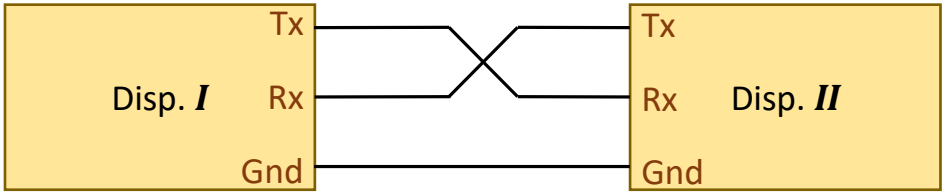
Recebe e transmite (bidirecional) dados de forma não sincronizada!



FUNÇÕES:

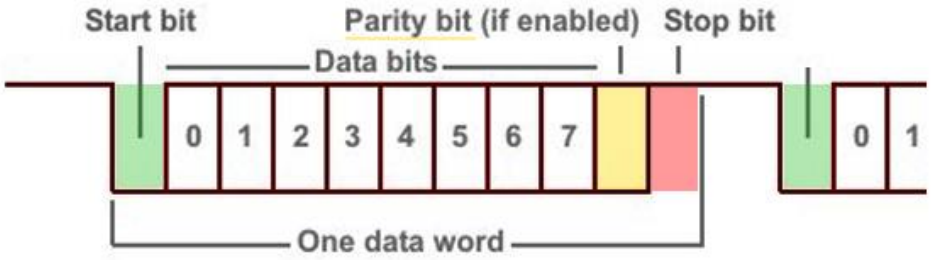
• É carregado com dados *paralelos* (*buffer de transmissão*) e os transmite *serialmente*;

• Recebe dado *serial* (*buffer de recepção*) e converte em *paralelo*;



• Pode usar um bit extra para verificação de erros simples (verificar *paridade*);

• Protocolo simples: um *start* e um *stop bit*;



• Usado p/ transmissão em longa distâncias;

• Taxa de transmissão:
- quantidade de bits/seg (*baud rate*)

(***baud rate*** – refere ao número de mudanças de símbolos - ou sinais - que ocorrem por segundo...
Um símbolo/sinal pode transmitir vários bits; assim, pode-se obter taxas de bits (bps) maiores num certo baud rate.

UART requer um registrador de configuração para guardar:

- taxa de transmissão/recepção;
- tamanho da palavra (7/8 bits)
- paridade (par, impar, none);
- stop bit.

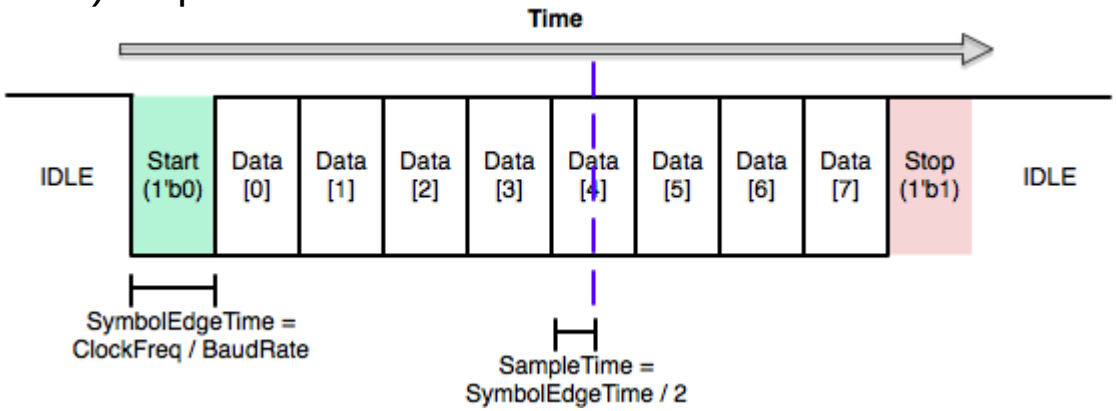
UART requer um temporizador para enviar/receber dados na taxa de dados programada.
Toma-se a amostra (sample) de cada bit na metade da largura de tempo de cada bit.

Ex: 9600 baud rate T ≈ 104 μs
Sample time = ½ do edge de T = ↓ + 52 μs
O clock que alimenta o timer da UART pode causar um pequeno erro (aceitável) no posicionamento da leitura do sample.

Serial Baud Rates Supported By NI-VISA
(National Instruments)

*baixa	*média	**alta
110	9600	
300	14400	128000
600	28800	153600
1200	38400	230400
2400	56000	256000
4800	57600	460800
	115200	921600

*suportado pela maioria das UARTS
** suportado apenas por algumas.



USART é uma interface que é um misto de UART + SPI.

→ Quando o sinal de CLK é enviado, a UART trabalha de modo síncrono e o clock equivale ao mesmo sinal no SPI.

→ Dois sinais adicionais podem estar presentes para que se faça *handshake* por hardware:

→ **RTS**: request to send – indica que a USART está pronta para receber dado (quando está em nível low)
→ **CTS**: clear to send – habilita o recebimento do dados (low) indicando que está pronta para bloqueia a transmissão de dados ao final da transferência atual (quando ativo em high)

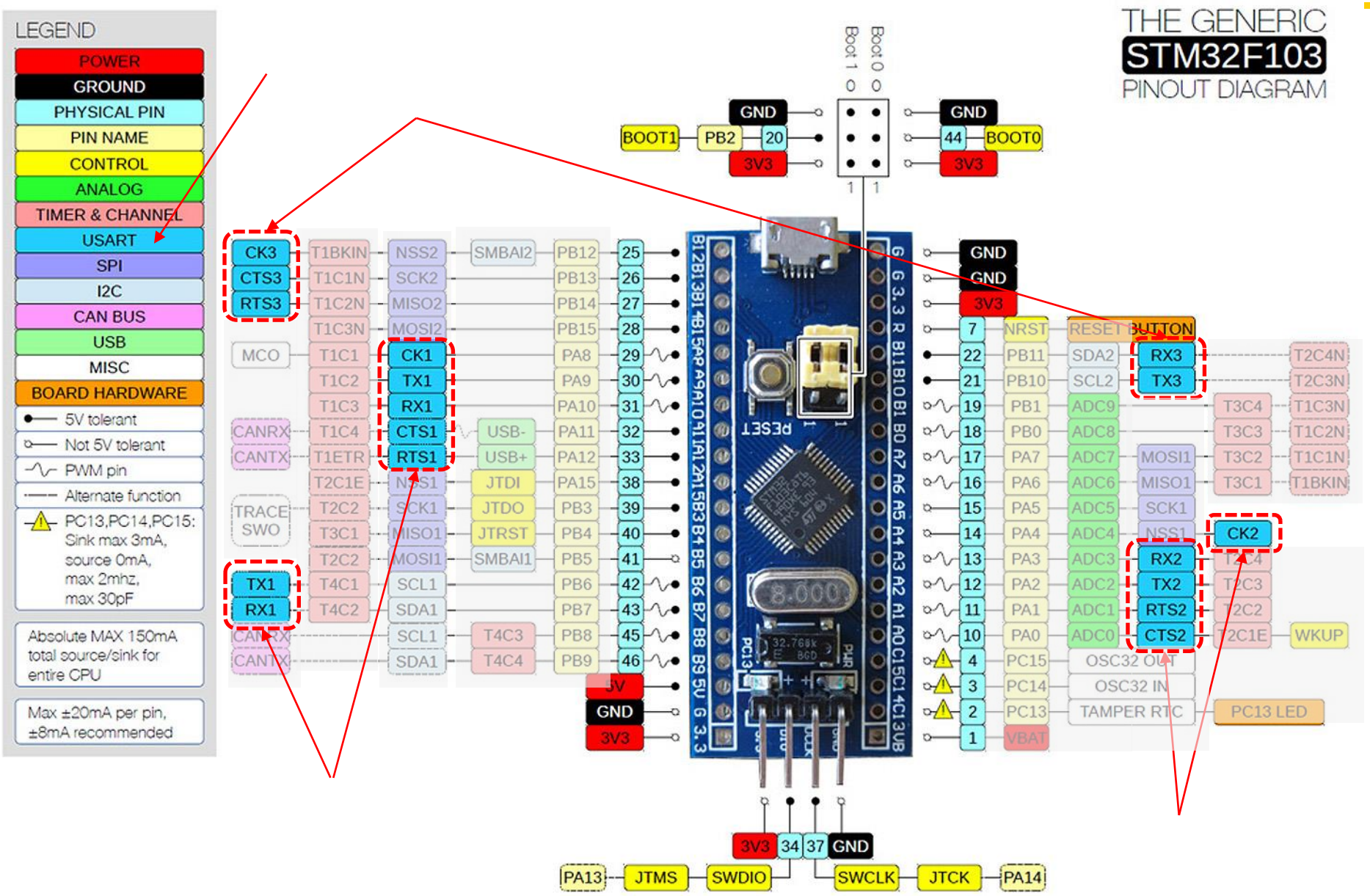
Serial Baud Rates Supported By NI-VISA
(National Instruments)

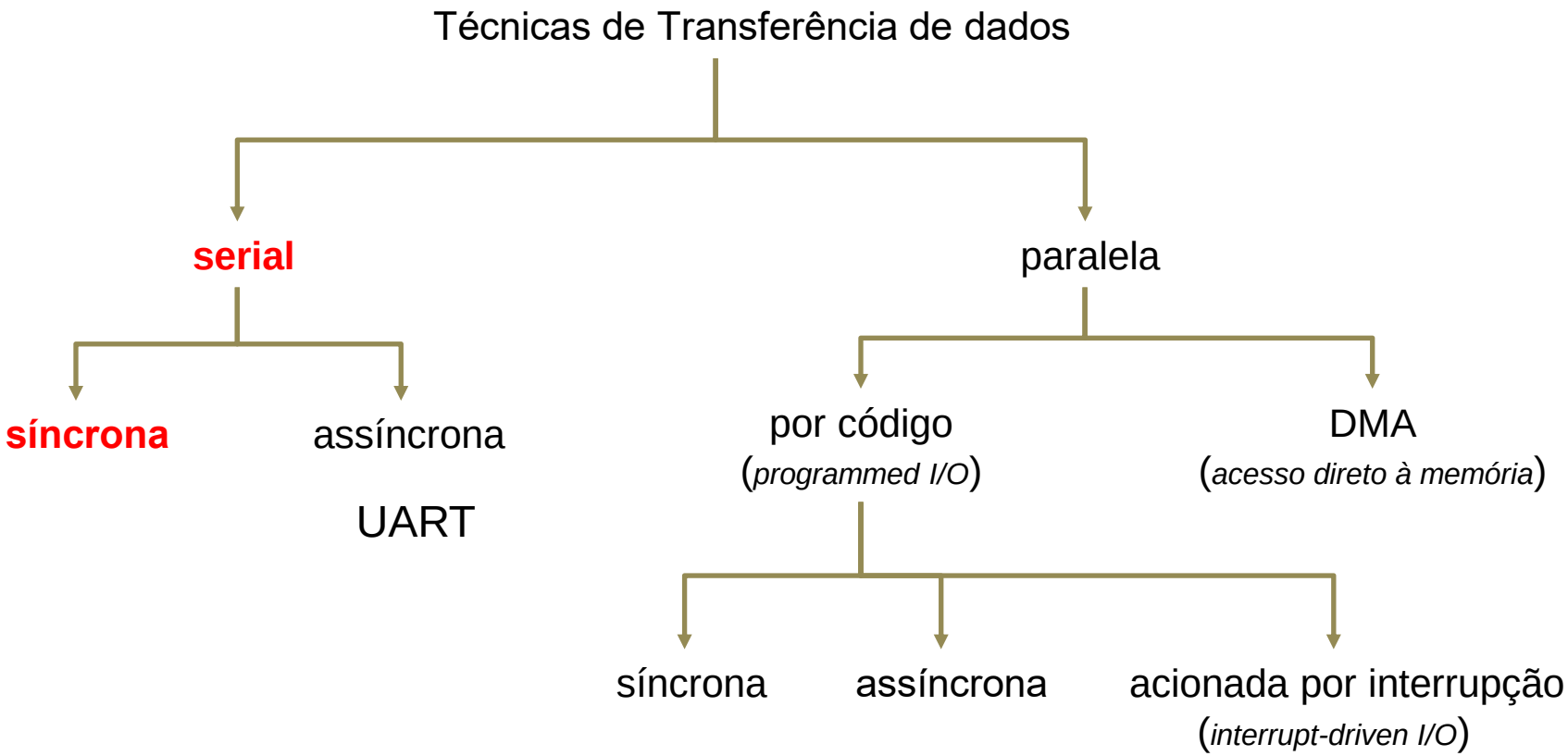
*baixa	*média	**alta
110	9600	
300	14400	128000
600	28800	153600
1200	38400	230400
2400	56000	256000
4800	57600	460800
	115200	921600

*suportado pela maioria das UARTS
** suportado apenas por algumas.

Do manual da STM32F103 – seção USART :

The following pins are required in *Hardware flow control mode*:
• **CTS:** Clear To Send blocks the data transmission at the end of the current transfer when high
• **RTS:** Request to send indicates that the USART is ready to receive a data (when low).


LEGEND
POWER
GROUND
PHYSICAL PIN
PIN NAME
CONTROL
ANALOG
TIMER & CHANNEL
USART
SPI
I2C
CAN BUS
USB
MISC
BOARD HARDWARE
5V tolerant
Not 5V tolerant
PWM pin
Alternate function
PC13,PC14,PC15: Sink max 3mA, source 0mA, max 2mhz, max 30pF
Absolute MAX 150mA total source/sink for entire CPU
Max ±20mA per pin, ±8mA recommended
THE GENERIC
STM32F103
PINOUT DIAGRAM
Boot 1
Boot 0
BOOT1
PB2
20
GND
3V3
44
BOOT0
CK3
CTS3
RTS3
MCO
CANRX
CANTX
TRACE SWO
TX1
RX1
CK1
TX1
RX1
CTS1
RTS1
USB-
USB+
JTDI
JTDO
JTRST
SMBAI1
SMBAI2
T4C3
T4C4
PB12
PB13
PB14
PB15
PA8
PA9
PA10
PA11
PA12
PA15
PB3
PB4
PB5
PB6
PB7
PB8
PB9
5V
GND
3V3
8.000
32.768k
RESET
PWR
PC13
GND
GND
3V3
NRST
PB11
PB10
PB1
PB0
PA7
PA6
PA5
PA4
PA3
PA2
PA1
PA0
PC15
PC14
PC13
VBAT
RESET BUTTON
SDA2
SCL2
ADC9
ADC8
ADC7
ADC6
ADC5
ADC4
ADC3
ADC2
ADC1
ADC0
OSC32 OUT
OSC32 IN
TAMPER RTC
RX3
TX3
MOSI1
MISO1
SCK1
NSS1
RX2
TX2
RTS2
CTS2
CK2
T3C4
T3C3
T3C2
T3C1
T2C4N
T2C3N
T1C3N
T1C2N
T1C1N
T1BKIN
T2C4
T2C3
T2C2
T2C1E
WKUP
PC13 LED
3V3
34
37
GND
PA13
JTMS
SWDIO
SWCLK
JTCK
PA14


Técnicas de Transferência de dados
serial
paralela
síncrona
assíncrona
UART
por código
(programmed I/O)
DMA
(acesso direto à memória)
síncrona
assíncrona
acionada por interrupção
(interrupt-driven I/O)

Interface serial SPI

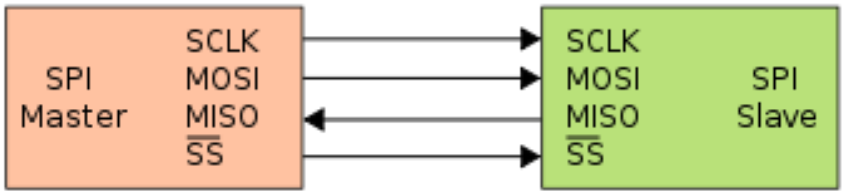
- SPI é uma interface ***single-ended***, ***full duplex***, ***síncrona*** para comunicação de curta distância em eletrônica embarcada (criado pela Motorola final da década de 1980). Usa arquitetura mestre-escravo com um mestre único (alguns escravos)

Mestre:

- ✓ Inicia o contato,
- ✓ Gera o clock da conversação (SCLK),
- ✓ Envia dados através da linha MOSI,
- ✓ Seleciona um dentre os escravos para comunicação (*ativa Service Select - SS_1...SS_4*);

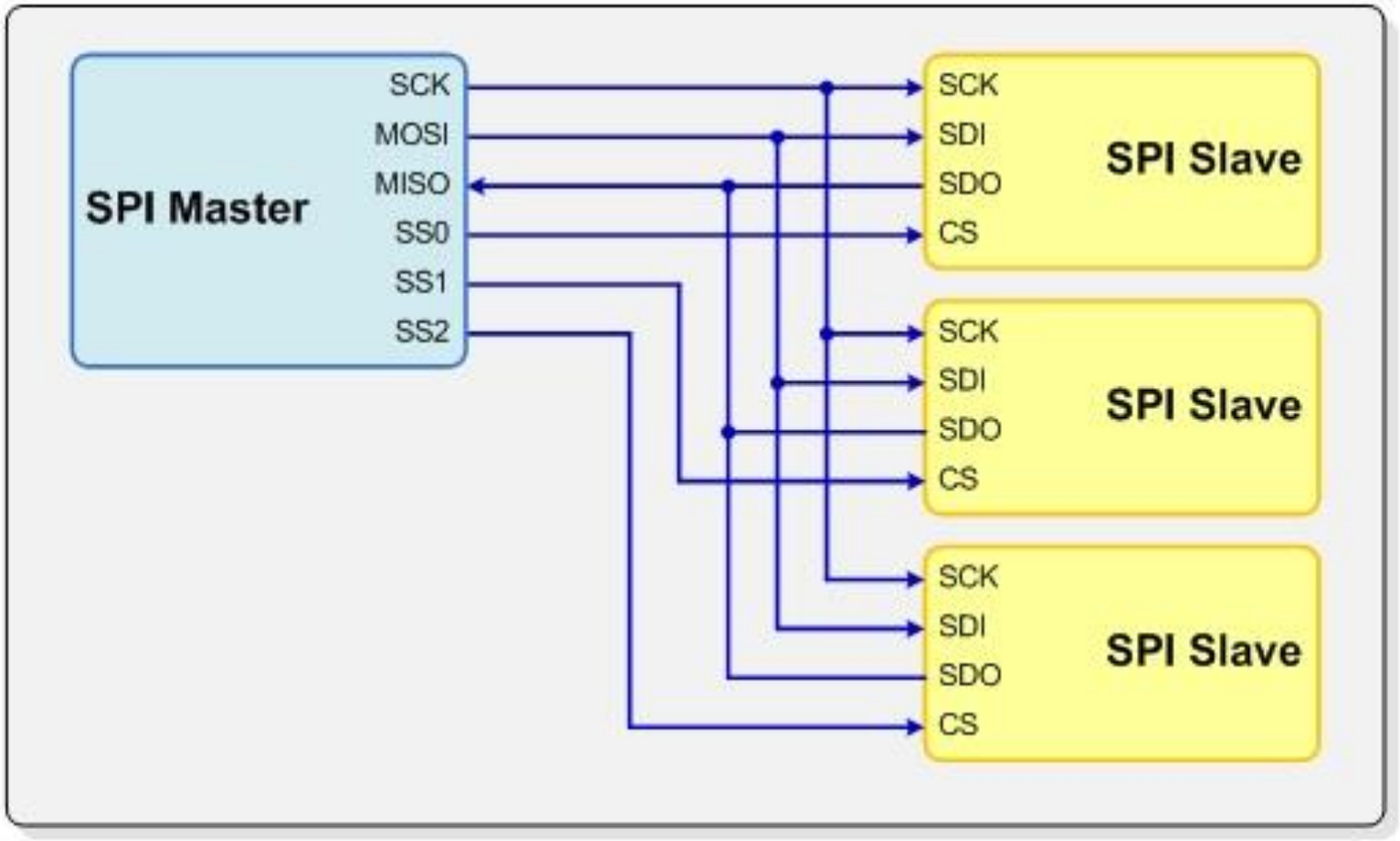
Escravo:

- ✓ Identifica que a operação é com ele (identifica SSx),
- ✓ Recebe comandos do mestre,
- ✓ Envia dados quando solicitado pela linha MISO,
- ✓ Grava dados recebidos do mestre



https://youtu.be/i_TLLACZuRk
https://youtu.be/ifgs0uypC78

- Exemplo de interface com os escravos em paralelo

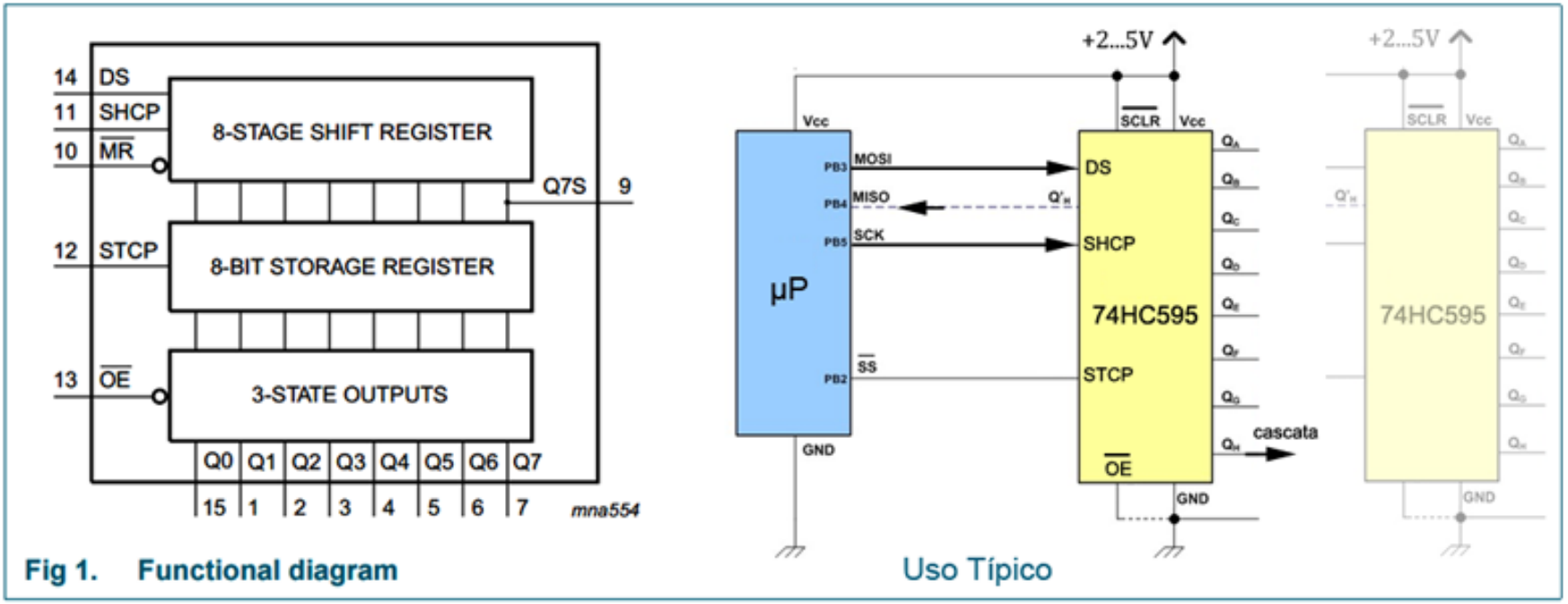


Apenas um sinal SSx pode estar ativo em um determinado momento!
Caso não tenha mais que um escravo, é possível usar apenas os sinais SCK e SDI (*sem retorno de dados do escravo*)

➢ ***SPI slave*** pode ser implementado com 74HC595 – um shift register com saída paralela 3-state.

74HC595; 74HCT595

8-bit serial-in, serial or parallel-out shift register with output latches; 3-state

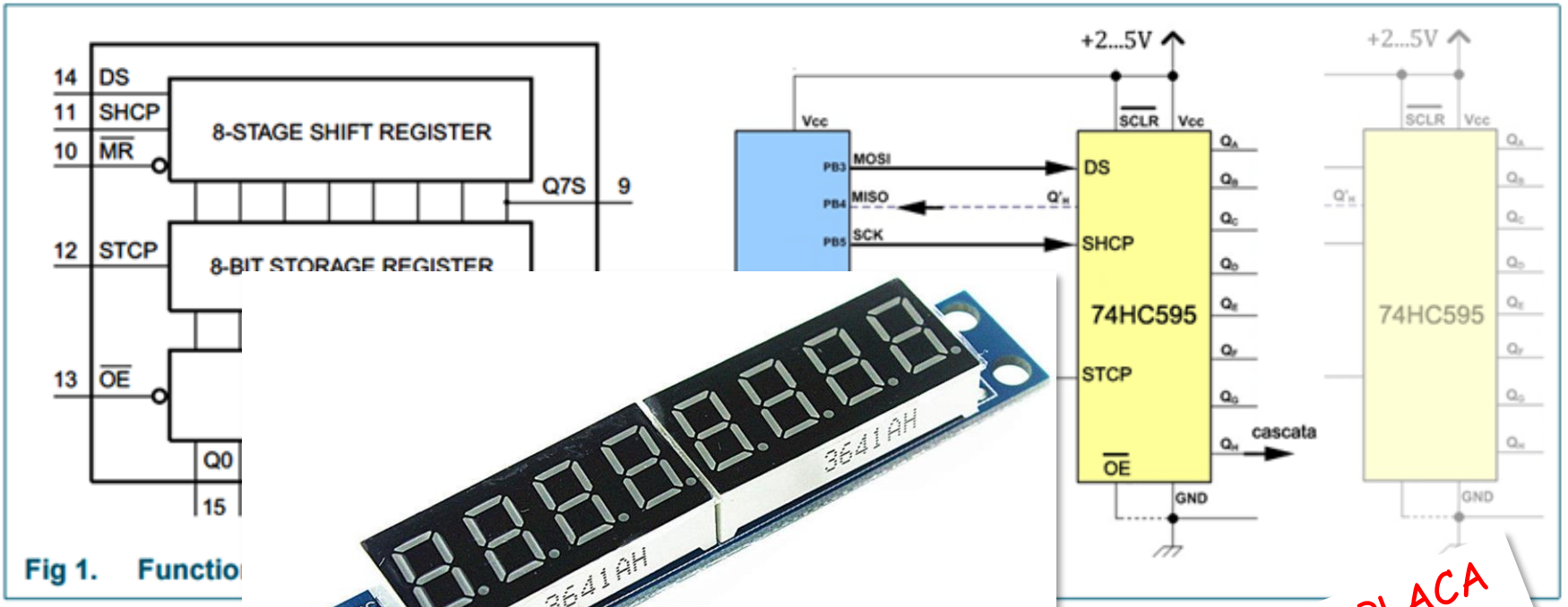


Esta interface tem custo pequeno e serve para vários propósitos (acender LEDs e displays e controles cujos pinos sejam digitais). Expandem sinais de saídas dos μP porque usam saída serial do μP e geram sinais paralelos nas placas.

- SPI slave pode ser implementado com 74HC595 – um shift register com saída paralela 3-state.

74HC595; 74HCT595

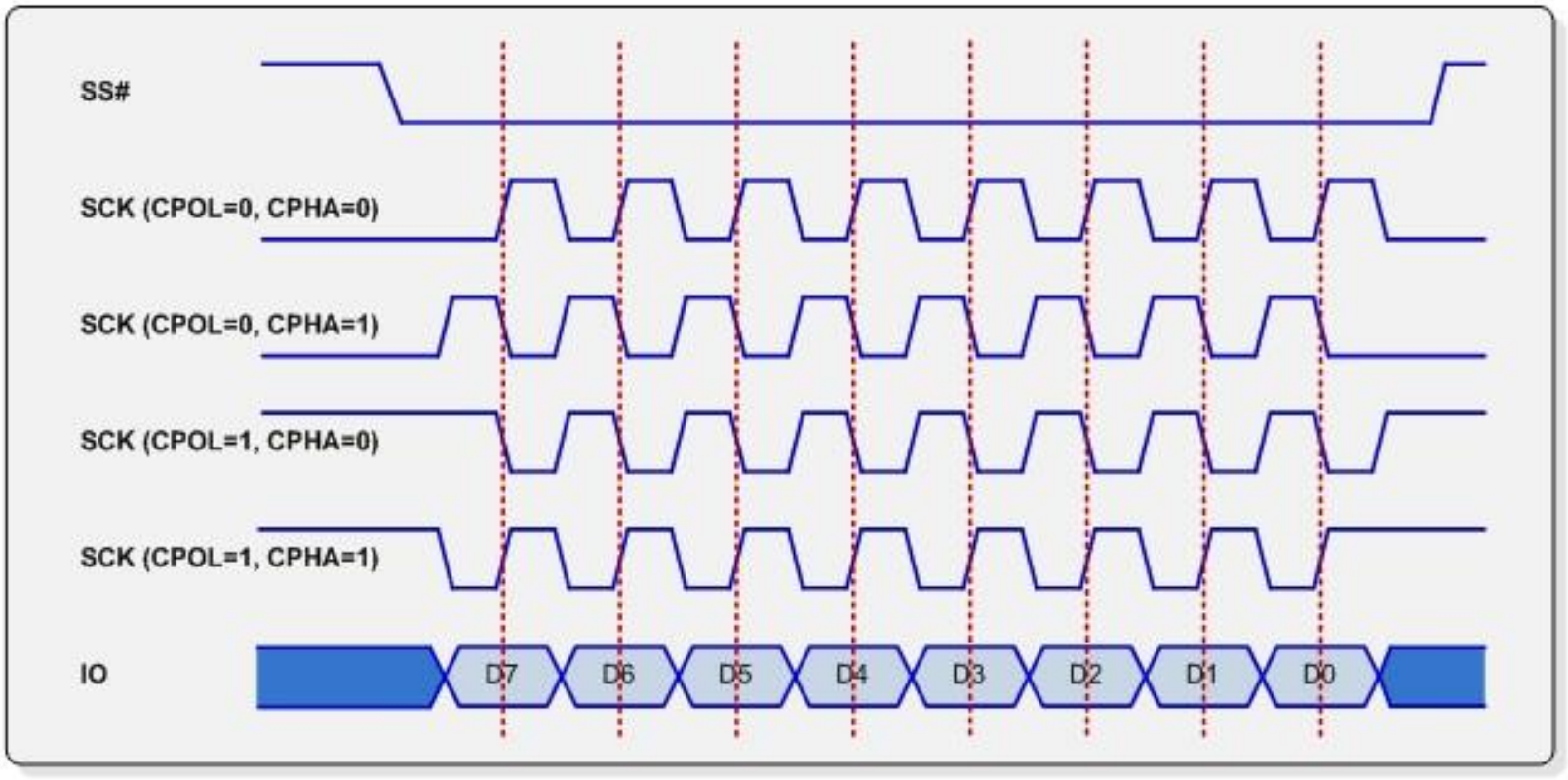
8-bit serial-in, serial or parallel-out shift register with output latches; 3-state



Exemplo de uma interface com HC595 que usamos nas aulas práticas!

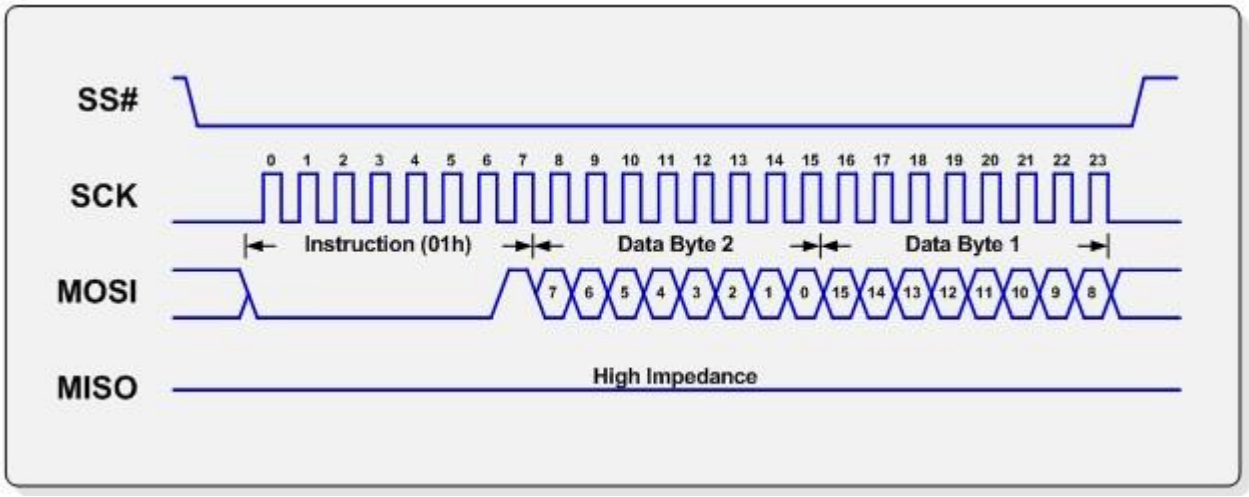
E A NOSSA PLACA UFABC !!!

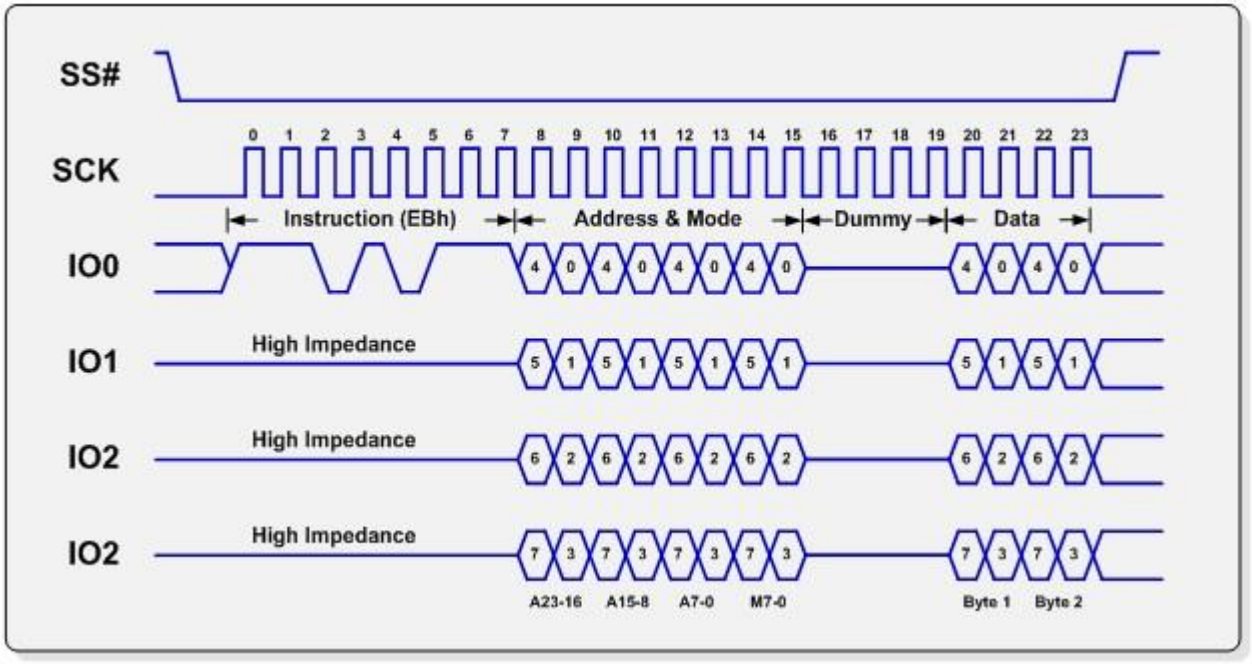
- Como o SPI opera com registradores de deslocamento, é possível operar em 4 modos diferentes, se levarmos em conta a polaridade do clock (CPOL) ou a fase (CPHA) na qual copiamos os dados:



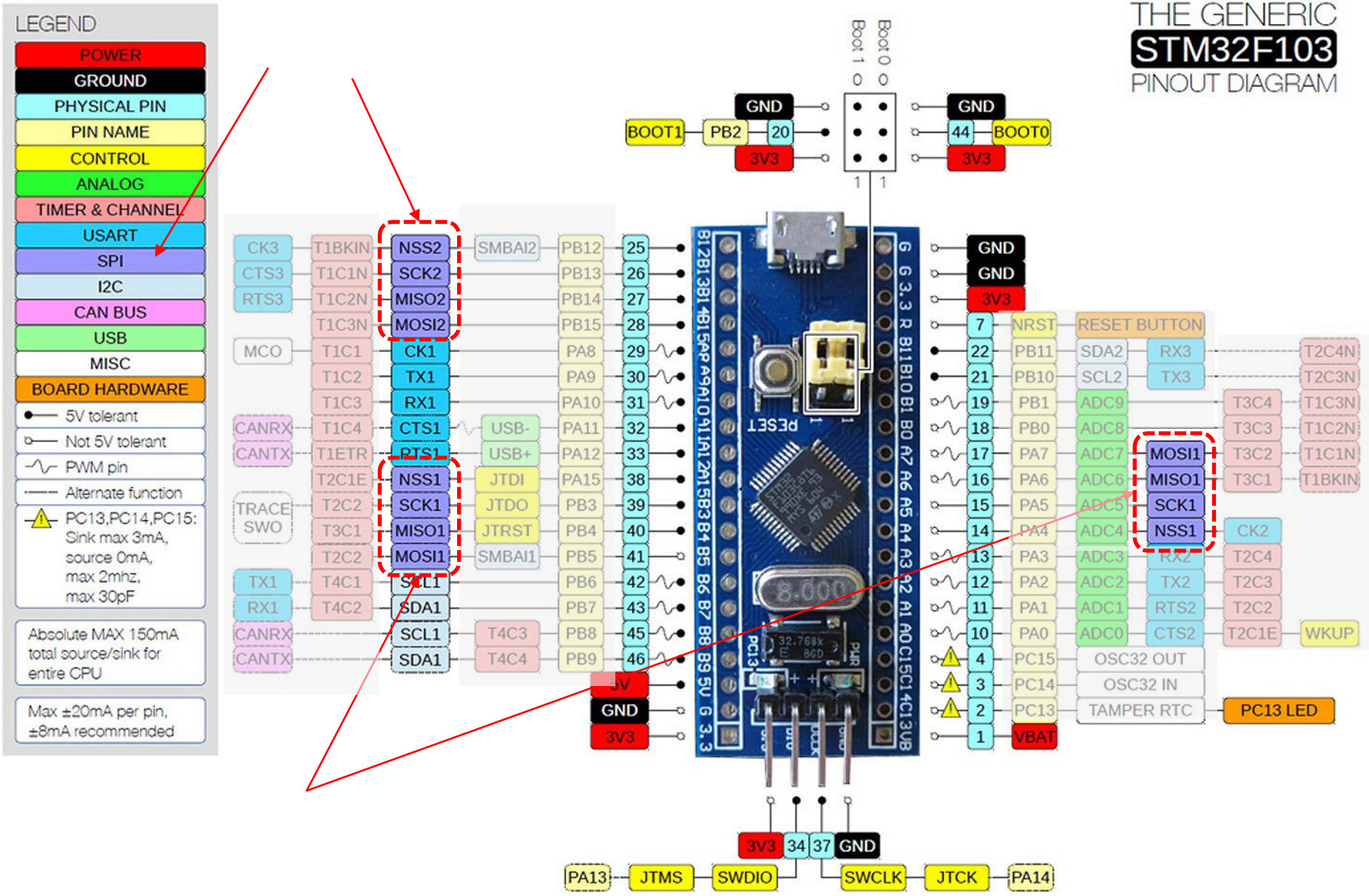
- **modo 0** = clock inicia em '**0**', e dado é copiado na subida do clock (↑ LE) - altera-se o novo dado na descida do clock
- **modo 1** = clock inicia em '**0**', e dado é copiado na descida do clock (↓ LE) - altera-se o novo dado na subida do clock
- **modo 2** = clock inicia em '**1**', e dado é copiado na descida do clock (↓ LE) - altera-se o novo dado na subida do clock
- **modo 3** = clock inicia em '**1**', e dado é copiado na subida do clock (↑ LE) - altera-se o novo dado na descida do clock

- Você tem liberdada para criar seu protocolo – você envia instruções, recebe dados, etc... O *slave* deve estar programado para responder a isso!





- Você pode sofisticar o protocolo e ler/ gravar vários registros dentro do *slave*. Nesse caso, você deve inserir envio de endereços no protocolo.


LEGEND
POWER
GROUND
PHYSICAL PIN
PIN NAME
CONTROL
ANALOG
TIMER & CHANNEL
USART
SPI
I2C
CAN BUS
USB
MISC
BOARD HARDWARE
5V tolerant
Not 5V tolerant
PWM pin
Alternate function
PC13,PC14,PC15: Sink max 3mA, source 0mA, max 2mhz, max 30pF
Absolute MAX 150mA total source/sink for entire CPU
Max ±20mA per pin, ±8mA recommended
THE GENERIC
STM32F103
PINOUT DIAGRAM
Boot 1
Boot 0
GND
BOOT1
PB2
20
3V3
44
BOOT0
CK3
CTS3
RTS3
MCO
CANRX
CANTX
TRACE SWO
TX1
RX1
CANRX
CANTX
T1BKIN
T1C1N
T1C2N
T1C3N
T1C1
T1C2
T1C3
T1C4
T1ETR
T2C1E
T2C2
T3C1
T2C2
T4C1
T4C2
NSS2
SCK2
MISO2
MOSI2
CK1
TX1
RX1
CTS1
RTS1
NSS1
SCK1
MISO1
MOSI1
SCL1
SDA1
SCL1
SDA1
SMBAI2
USB-
USB+
JTDI
JTDO
JTRST
SMBAI1
T4C3
T4C4
PB12
PB13
PB14
PB15
PA8
PA9
PA10
PA11
PA12
PA15
PB3
PB4
PB5
PB6
PB7
PB8
PB9
25
26
27
28
29
30
31
32
33
38
39
40
41
42
43
45
46
5V
GND
3V3
GND
GND
3V3
7
22
21
19
18
17
16
15
14
13
12
11
10
4
3
2
1
NRST
PB11
PB10
PB1
PB0
PA7
PA6
PA5
PA4
PA3
PA2
PA1
PA0
PC15
PC14
PC13
VBAT
RESET BUTTON
SDA2
SCL2
ADC9
ADC8
ADC7
ADC6
ADC5
ADC4
ADC3
ADC2
ADC1
ADC0
OSC32 OUT
OSC32 IN
TAMPER RTC
RX3
TX3
MOSI1
MISO1
SCK1
NSS1
TX2
RTS2
CTS2
T2C4N
T2C3N
T1C3N
T1C2N
T1C1N
T1BKIN
T3C4
T3C3
T3C2
T3C1
CK2
T2C4
T2C3
T2C2
T2C1E
WKUP
PC13 LED
3V3
34
37
GND
PA13
JTMS
SWDIO
SWCLK
JTCK
PA14

Interface serial I2C

- I²C é uma interface ***single-ended***, ***multi-master***, ***multi-slave***, ***half-duplex***, ***síncrona*** para comunicação de curta distância em eletrônica embarcada (*criado pela Philips na década de 1980*).
- Usa 2 linhas bidirecionais *open-drain*, ***SDA*** (*Serial Data Line*) e ***SCL*** (*Serial Clock Line*) + VCC + GND.
- Original = 100 KHz, v.1 = 400 Khz, v.5 (2012) = 5 MHz *Ultra Fast-mode (UFm)*

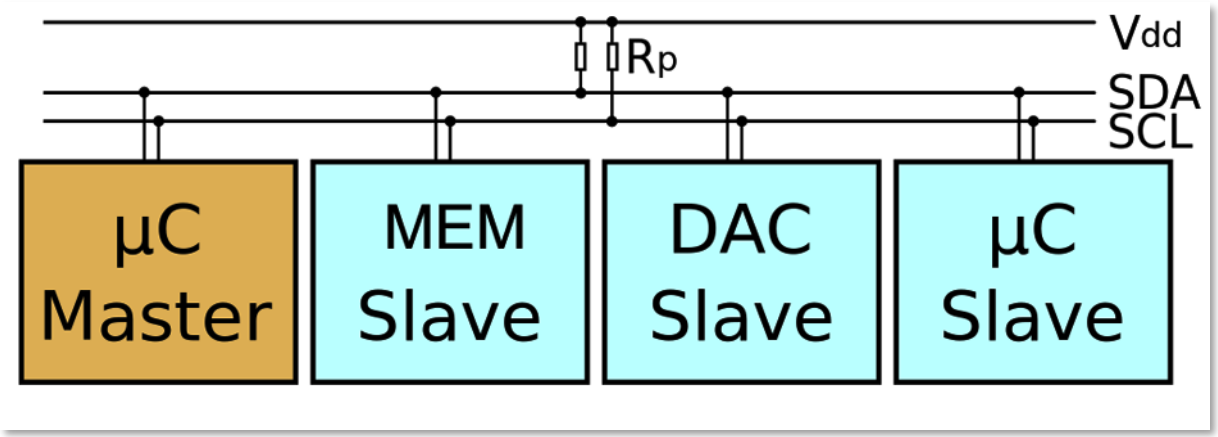
→ Bus derivados: System Management Bus (SMBus or SMB), Power Management Bus (PMBus), I²S (Inter-IC Sound), etc.

Mestre:

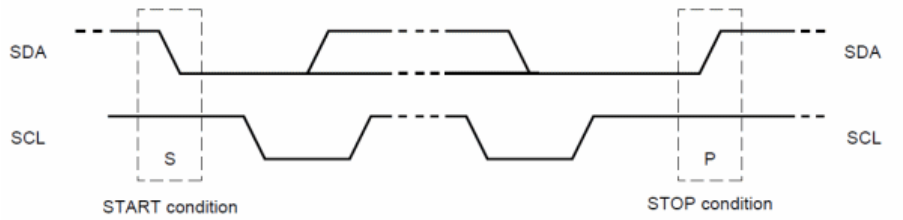
- ✓ Inicia a comunicação: baixa a linha SDA quando SCL ='1', e gera o clock da conversação (SCL),
- ✓ Coloca endereço do escravo (7 ou 10 bits) e 1 bit de comando (**RD /$\overline{\text{WR}}$**) em seguida
- ✓ Avalia ACKNOWLEDGE do escravo
- ✓ Continua gerar clock e enviar/receber dados até o final do frame (pacote de dados programado)
- ✓ Sobe o sinal de clock e em seguida sobe DAS

Escravo:

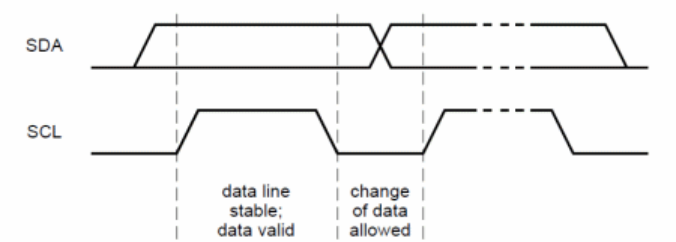
- ✓ Identifica que a operação é com ele (pelo endereço e bit **RD/$\overline{\text{WR}}$**), e gera acknowledge
- ✓ Em seguida, recebe ou envia dados para o mestre a cada pulso de clock,



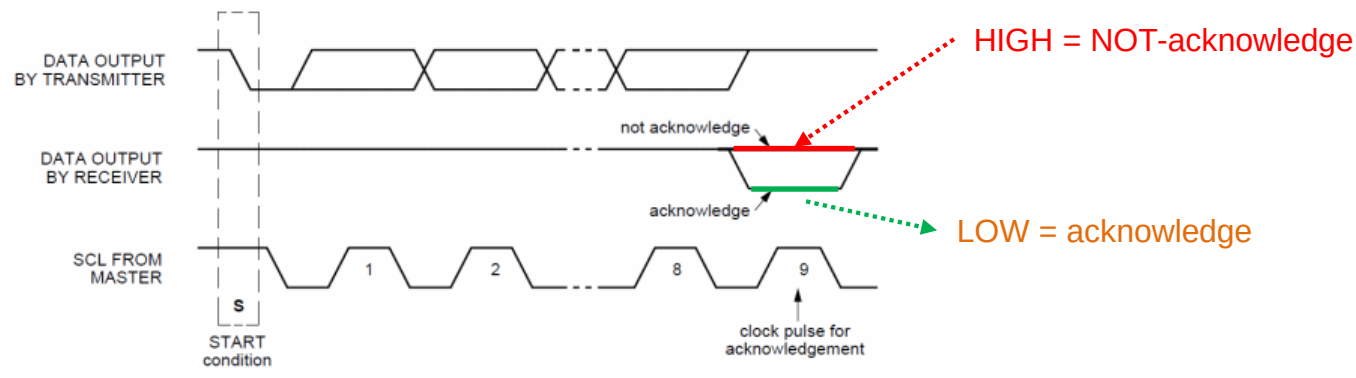
- ***Protocolo***:
- Mestre tenta iniciar uma comunicação baixando SDA enquanto SCL está em alto
- Comunicação será terminada quando SCL for para 1 e em seguida SDA for também para '1'



- SDA terá dado válido quando SCL = '1'. Dado deve mudar quando SCL = low.
- Quando um slave mantém o clock low WAIT states são gerados.

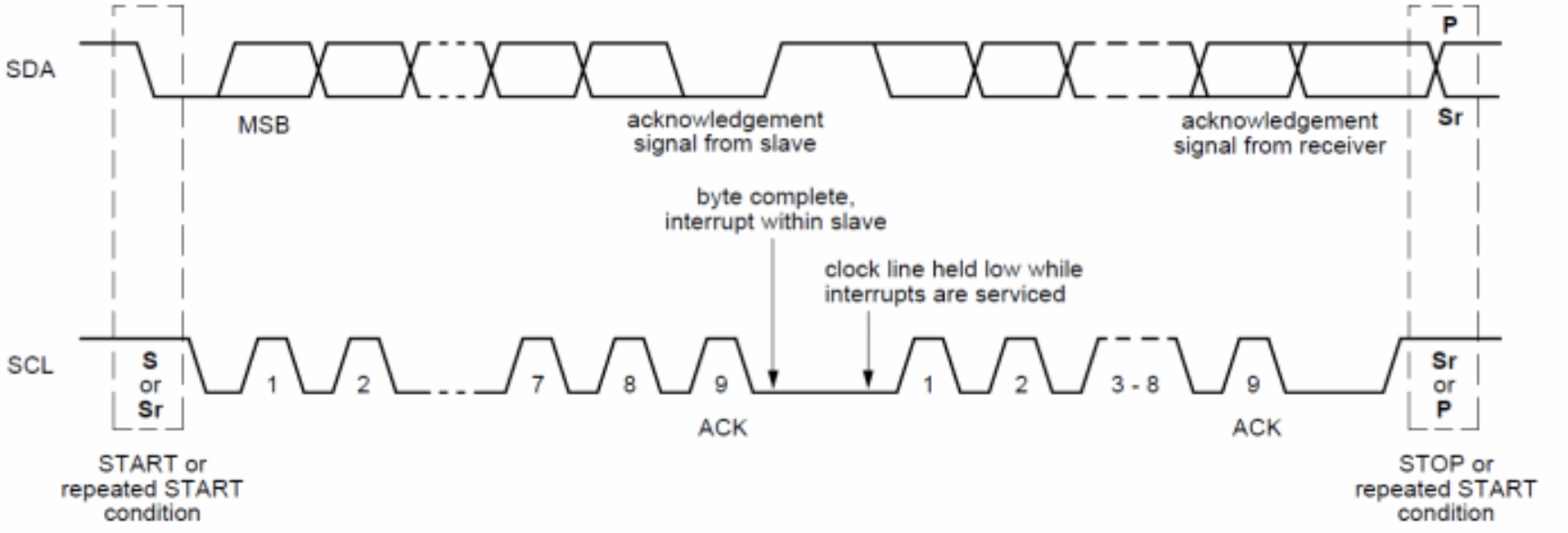


- Um bit acknowledge controlado pelo escravo informar reconhecimento do início da comunicação



http://i2c.info/i2c-bus-specification

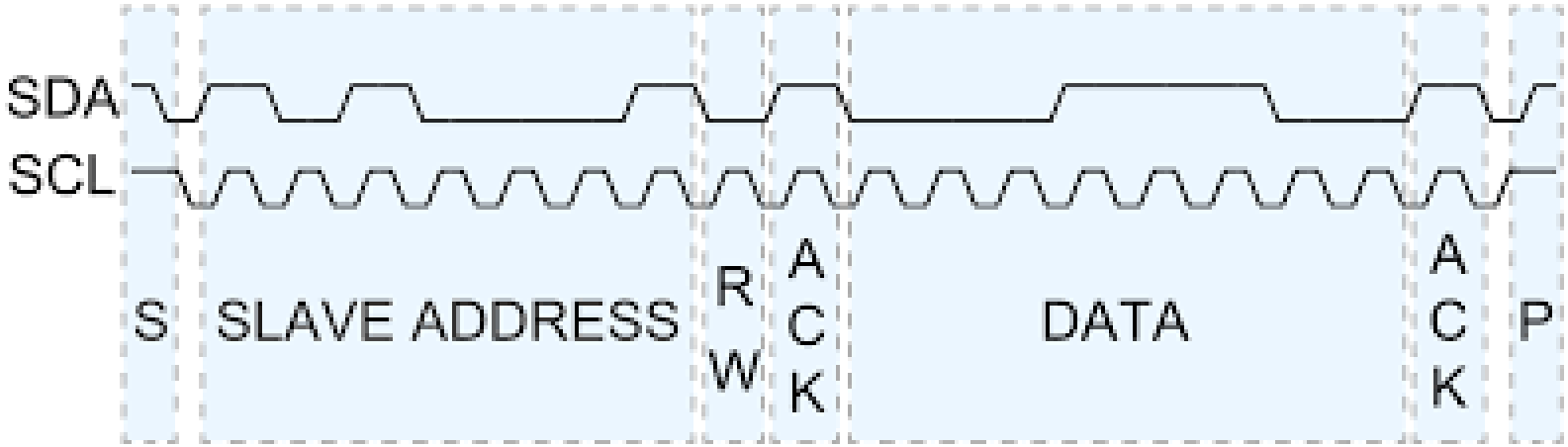
- ***Protocolo***:
- Mestre inicia comunicação, envia 7 bits de endereço + 1 bit comando RD/$\overline{WR}$, (todos os slaves devem atentar para o endereço);
- O bit mais significativo é enviado primeiro;
- No 9º pulso de clock o receptor deve enviar acknowledge – na versão 10 bits de endereço isso ocorre no 12º clock;
- Depois do acknowledge, o mestre continua gerando clock para receber ou enviar dados para o escravo;
- Assim que gera acknowledge o escravo pode segurar o sinal de clock baixo para gerar wait states (ex: gerar/atender interrupções);
- Durante a transferência do primeiro byte o mestre é o transmissor e o slave (endereçado) é o receptor;
- Quem transmite o próximo byte depende do bit de comando **RD /$\overline{WR}$** que foi enviado depois do endereço;

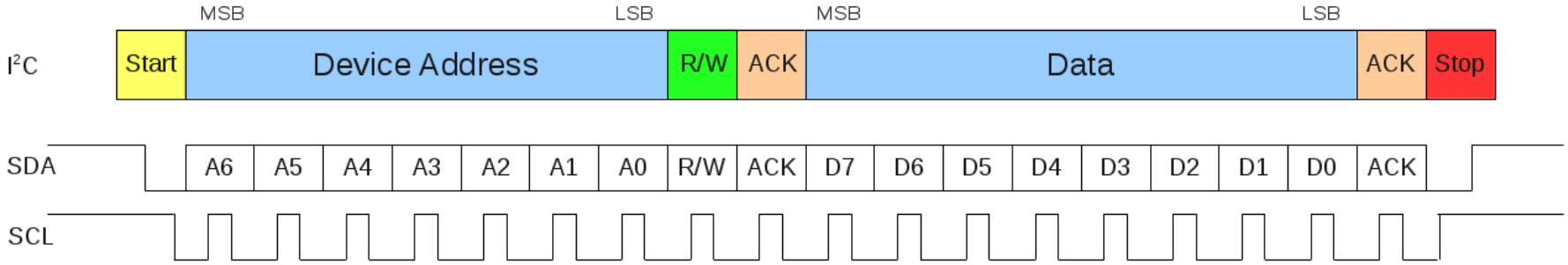


- TRANSMISSÃO DE DADOS: o transmissor [escravo (**RD**) / mestre ($\overline{\mathbf{WR}}$)] pulsa SDA para transmitir os dados (MSB 1º.)
- Durante o 9º pulso de clock o transmissor libera SDA para que o receptor indique data acknowledge
- Escravo deve liberar SDA depois do bit acknowledge;

- ***Protocolo*** (***data frame***):

- Um ***frame*** demarca 1 start bit, 1 endereço, 1 comando, 1 acknowledge endereço, envio ou recebimento de um byte, um acknowledge data, e (1 novo start ou 1 stop bit). Ao final, um byte foi enviado ou lido de um periférico.

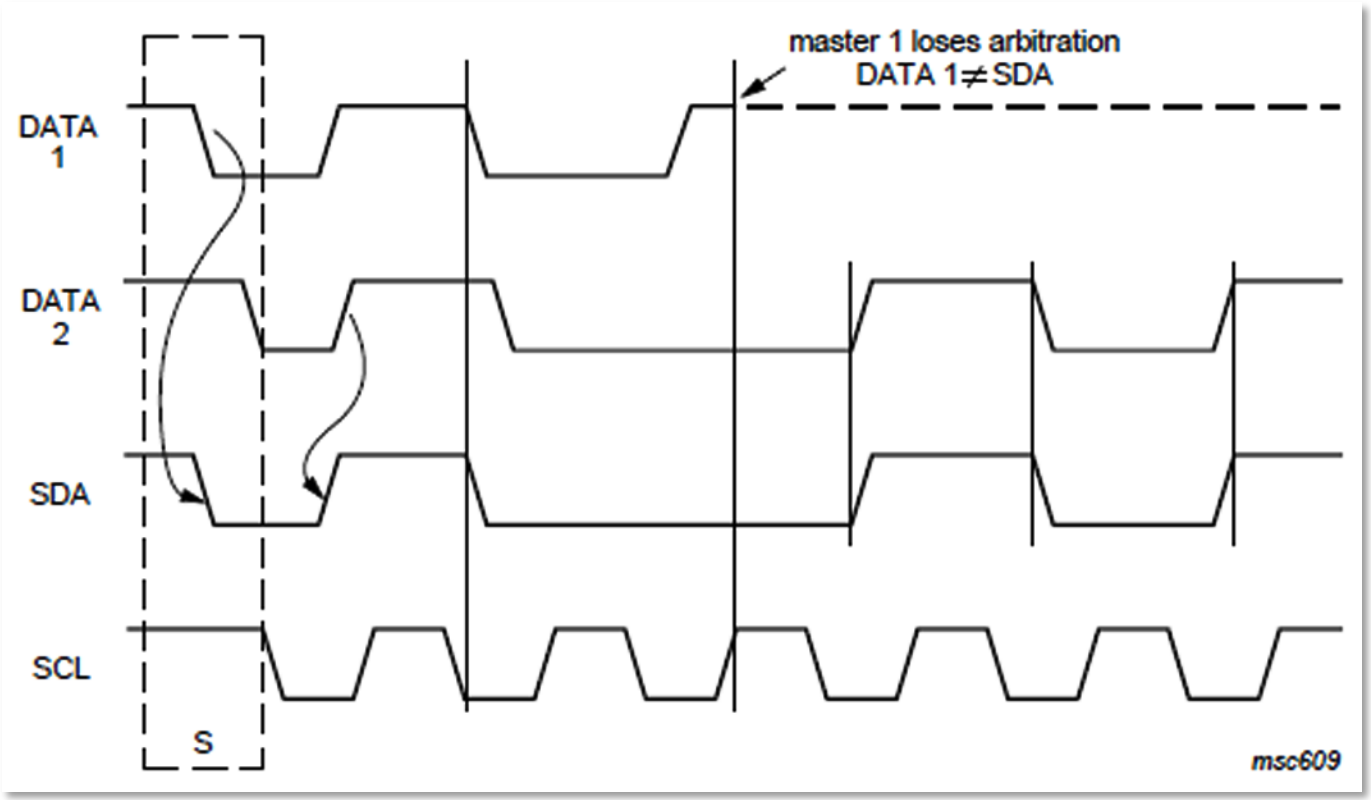




- ***Protocolo*** (***ARBITRAGEM***):
- Como vários dispositivos I²C podem ser mestres no mesmo barramento, pode ocorrer conflito (dois tentando ser mestre)...
- Neste caso há necessidade de arbitragem

(1) Antes de tentar iniciar comunicação o dispositivo verifica as linhas (ambas high)

(2) Assim que detecta que enviou um valor '1' e a linha não foi para '1' é porque está havendo conflito – o dispositivo abandona a tentativa de comunicação e tenta mais tarde



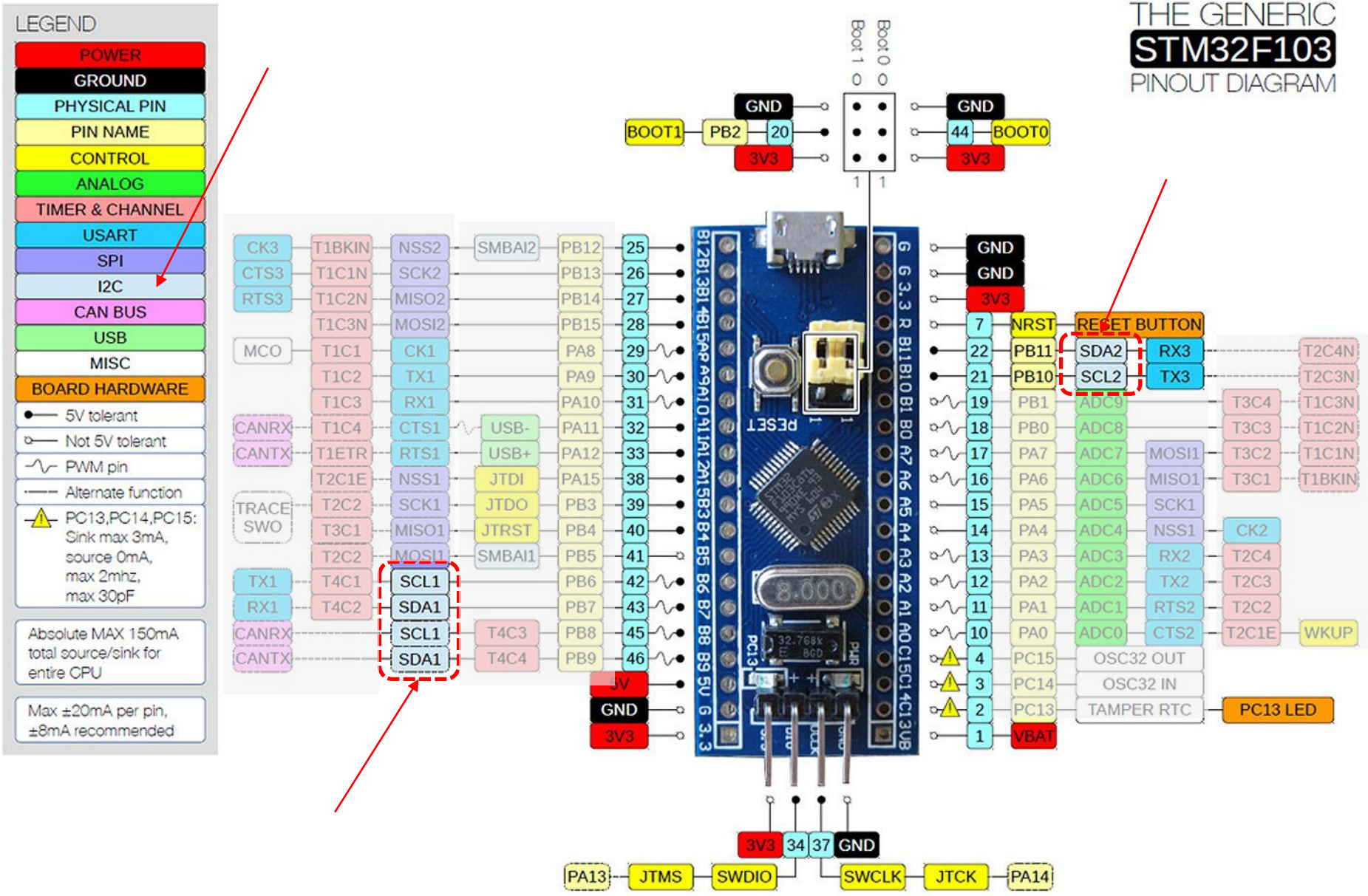
Ver manual do I²C em http://i2c.info/i2c-bus-specification

Vantagens:

- ✓ Bom para comunicar com dispositivos on board (on PCB) que não são acessados constantemente
- ✓ Devido ao esquema de endereços é fácil interconectar múltiplos dispositivos
- ✓ Custo e a complexidade não aumentam com o número de dispositivos
- ✓ Novas versões estão razoavelmente rápidas
- ✓ Requerem menos pinos dos μP ou SoC,
- ✓ Dispositivo pode assumir o papel de mestre – qualquer dispositivo pode se tornar mestre

Desvantagens:

- ✓ Escravos precisam de um endereço específico e único
- ✓ Gasta ao menos 8 bits para enviar endereço e mais dois bits para acknowledge (baixo throughput)
- ✓ Comparado a outros esquemas de comunicação (SPI) os componentes de software são mais complexos (porque tem todo um protocolo estipulado para a comunicação)


LEGEND
POWER
GROUND
PHYSICAL PIN
PIN NAME
CONTROL
ANALOG
TIMER & CHANNEL
USART
SPI
I2C
CAN BUS
USB
MISC
BOARD HARDWARE
5V tolerant
Not 5V tolerant
PWM pin
Alternate function
PC13,PC14,PC15: Sink max 3mA, source 0mA, max 2mhz, max 30pF
Absolute MAX 150mA total source/sink for entire CPU
Max ±20mA per pin, ±8mA recommended
THE GENERIC
STM32F103
PINOUT DIAGRAM
Boot 1
Boot 0
GND
BOOT1
PB2
20
3V3
GND
44
BOOT0
3V3
CK3
CTS3
RTS3
MCO
CANRX
CANTX
TRACE SWO
TX1
RX1
CANRX
CANTX
T1BKIN
T1C1N
T1C2N
T1C3N
T1C1
T1C2
T1C3
T1C4
T1ETR
T2C1E
T2C2
T3C1
T2C2
T4C1
T4C2
NSS2
SCK2
MISO2
MOSI2
CK1
TX1
RX1
CTS1
RTS1
NSS1
SCK1
MISO1
MOSI1
SCL1
SDA1
SCL1
SDA1
SMBAI2
USB-
USB+
JTDI
JTDO
JTRST
SMBAI1
T4C3
T4C4
PB12
PB13
PB14
PB15
PA8
PA9
PA10
PA11
PA12
PA15
PB3
PB4
PB5
PB6
PB7
PB8
PB9
25
26
27
28
29
30
31
32
33
38
39
40
41
42
43
45
46
5V
GND
3V3
GND
GND
3V3
7
22
21
19
18
17
16
15
14
13
12
11
10
4
3
2
1
NRST
PB11
PB10
PB1
PB0
PA7
PA6
PA5
PA4
PA3
PA2
PA1
PA0
PC15
PC14
PC13
VBAT
RESET BUTTON
SDA2
SCL2
RX3
TX3
ADC9
ADC8
ADC7
ADC6
ADC5
ADC4
ADC3
ADC2
ADC1
ADC0
MOSI1
MISO1
SCK1
NSS1
RX2
TX2
RTS2
CTS2
OSC32 OUT
OSC32 IN
TAMPER RTC
T2C4N
T2C3N
T3C4
T3C3
T3C2
T3C1
T1C4N
T1C3N
T1C2N
T1C1N
T1BKIN
CK2
T2C4
T2C3
T2C2
T2C1E
WKUP
PC13 LED
3V3
34
37
GND
PA13
JTMS
SWDIO
SWCLK
JTCK
PA14

- Existem CIs que fazem bridge entre barramento SPI e I²C

(embora a maioria dos μC e SoC atualmente dêem suporte para os dois barramentos)

4. Block diagram

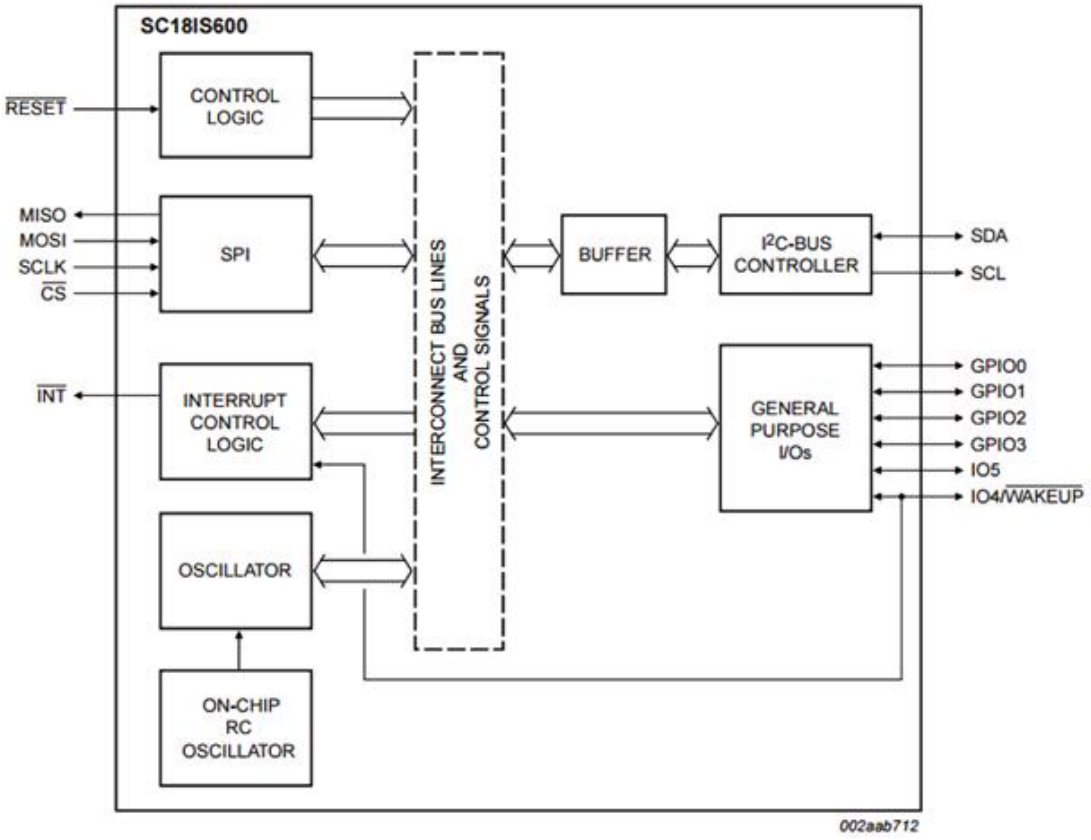


Fig 1. Block diagram of SC18IS600

SPI vs I²C

Vantagens:

- ✓ comunicação Full duplex (comunicação em paralelo)
- ✓ Menos energia que I²C
- ✓ Não necessita de arbitragem
- ✓ Throughput maior que I²C ou SMBus
- ✓ Escravos usam os clock do mestre
- ✓ Tamanho da mensagem é arbitrário,
- ✓ Escravos não precisam de um endereço específico e único
- ✓ O conteúdo e o propósito também é flexível
- ✓ Sinal é unidirecional
- ✓ Hardware interface bastante simples

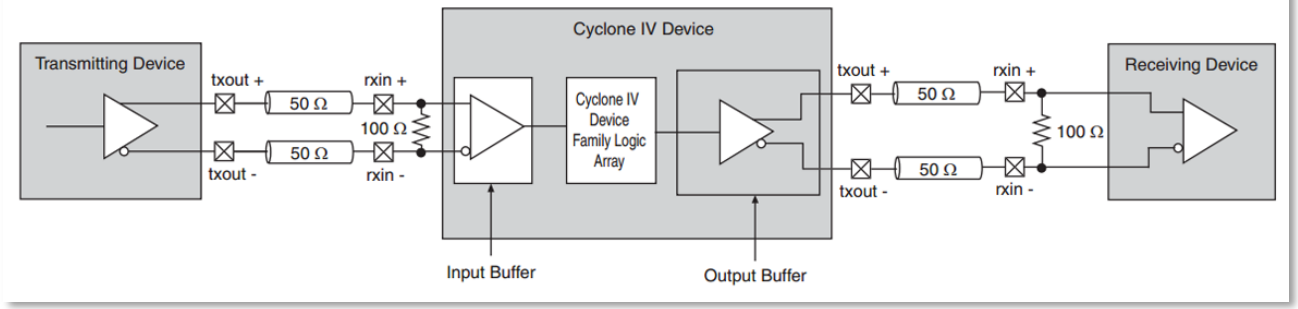
Desvantagens:

- ✓ Requer mais pinos do CI do que I²C
- ✓ Escravo não tem controle sobre o hardware ou o fluxo de informação
- ✓ Não há confirmação de recebimento (*acknowledge*) por parte do escravo
- ✓ Suporta apenas um mestre
- ✓ Não tem um padrão formalizado

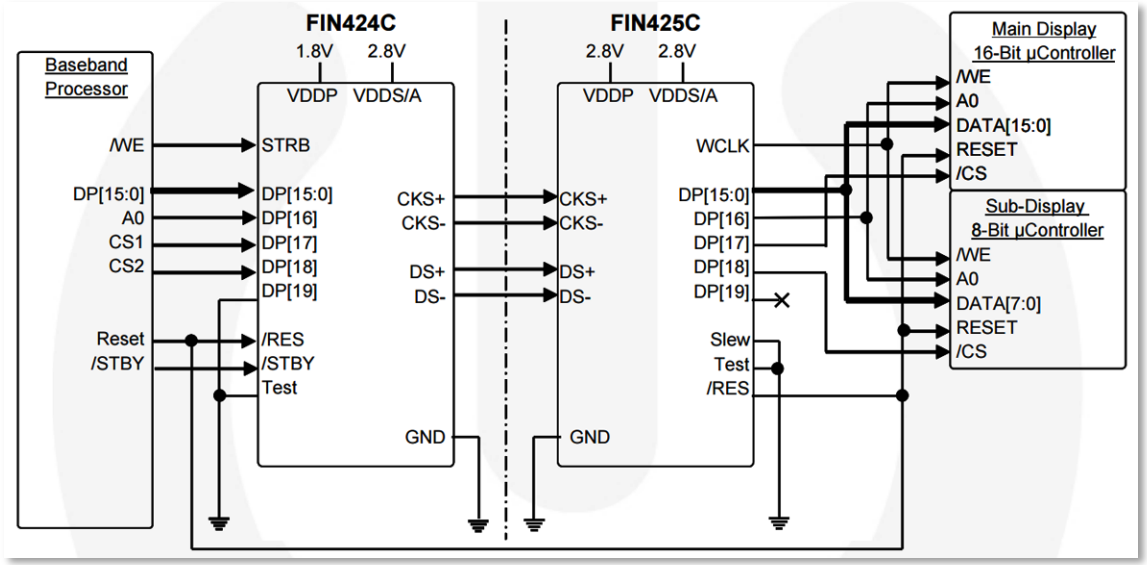
Interface serial USB (overview)

➢ Sinais diferenciais – na FPGA e em CIs dedicados

Fig 6–12. Cyclone IV Devices LVDS Interface with True Output Buffer on the Right I/O Banks

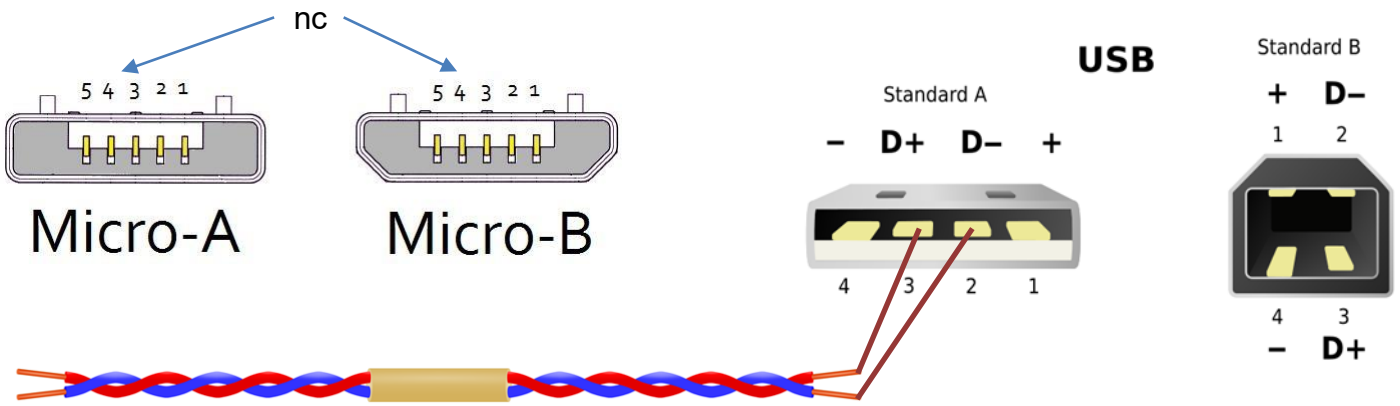


As FPGAs modernas têm uma série de pares de pinos que podem ser programados para funcionarem como sinais diferenciais e gerar LVDS (**Low-Voltage Differential Signaling**) ou ainda SSTL (**Stub Series Terminated Logic** - usado para comunicação com memória DDR) e HSTL (**High-speed transceiver logic** - um padrão independente para sinalização entre circuitos integrados).



- USB (*Universal Serial Bus - 1994*) é um protocolo serial assíncrono e um link físico (*protocolo de rede*)
- Transmite dados diferencialmente em um único par trançado de fios
- Outro par provê energia para o periférico em cascata (*downstream*)
- Protocolo centrado em Personal Computers (PC)
- Cada dispositivo USB é um sistema embarcado...
- Se você criar um sistema embarcado com capacidade/habilidade para se comunicar com um PC hoster, você possivelmente deve optar por USB.

- Muito altas velocidades (◇ ***3.0*** *= 5 Gb/s;* ◇ ***2.0*** *= 480 Mb/s;* ◇ ***1.0*** *= 12 Mb/s e 1,5 Mb/s*)

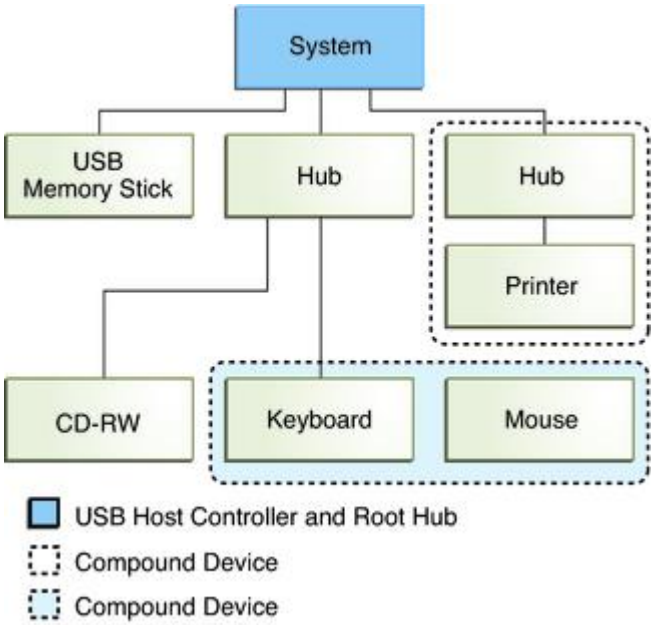


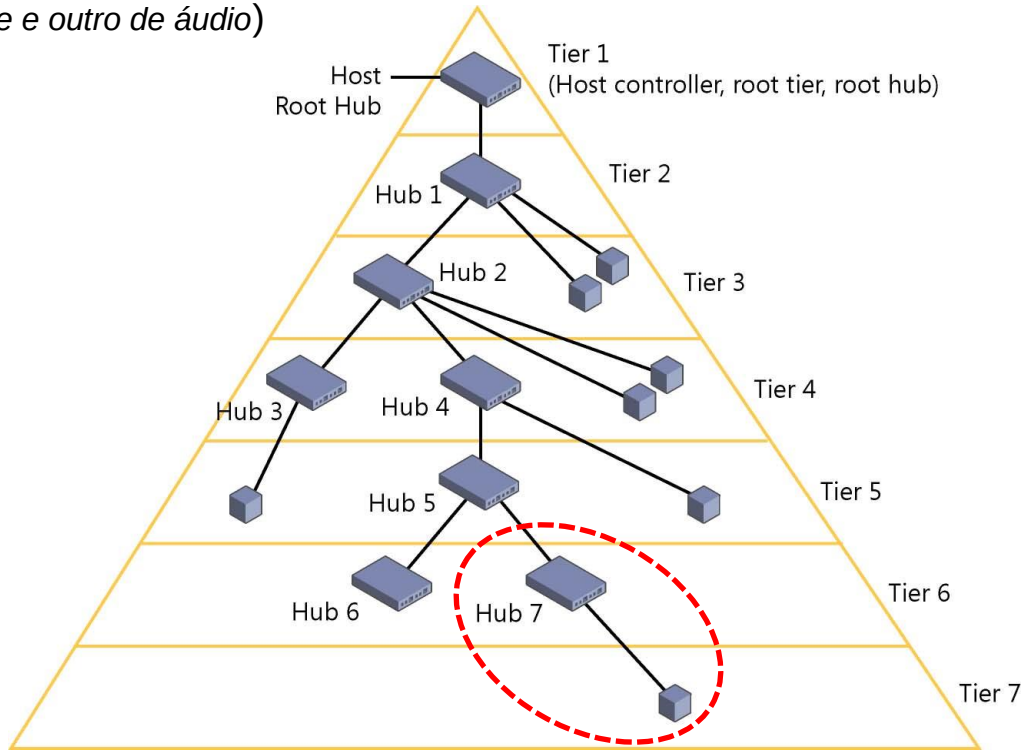
Arquitetura:

- Tipicamente, cada periférico USB é um slave, respondendo comandos de um host
- Quando o periférico é inserido no barramento o host se comunica com ele para descobrir sua "identidade" e encontrar um driver que deve ser usado (*processo chamado enumeration*)
- USB HUBS são bridges (*e são eles próprios USB devices*) que detectam mudanças na topologia quando se **insere** ou se **retira** um dispositivo na rede.

Especificações reconhecem dois tipos de periféricos:

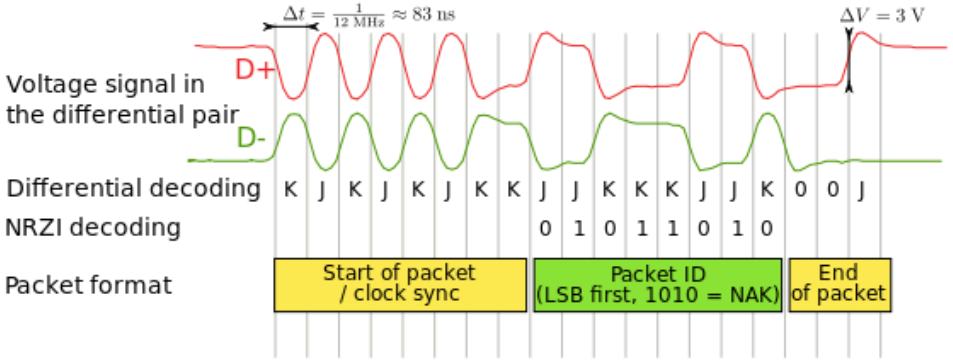
- Stand-alone (*periféricos de função única, como teclado, mouse, etc*)
- Dispositivos Compostos (*periféricos que possuem mais de um periférico compartilhando a porta USB, como uma câmera de vídeo que possui um processador de víde e outro de áudio*)





Operação:

- Protocolo bastante complexo –
- Sinal utiliza encoding/decoding NRZ-I e forte esquema de correção de erros



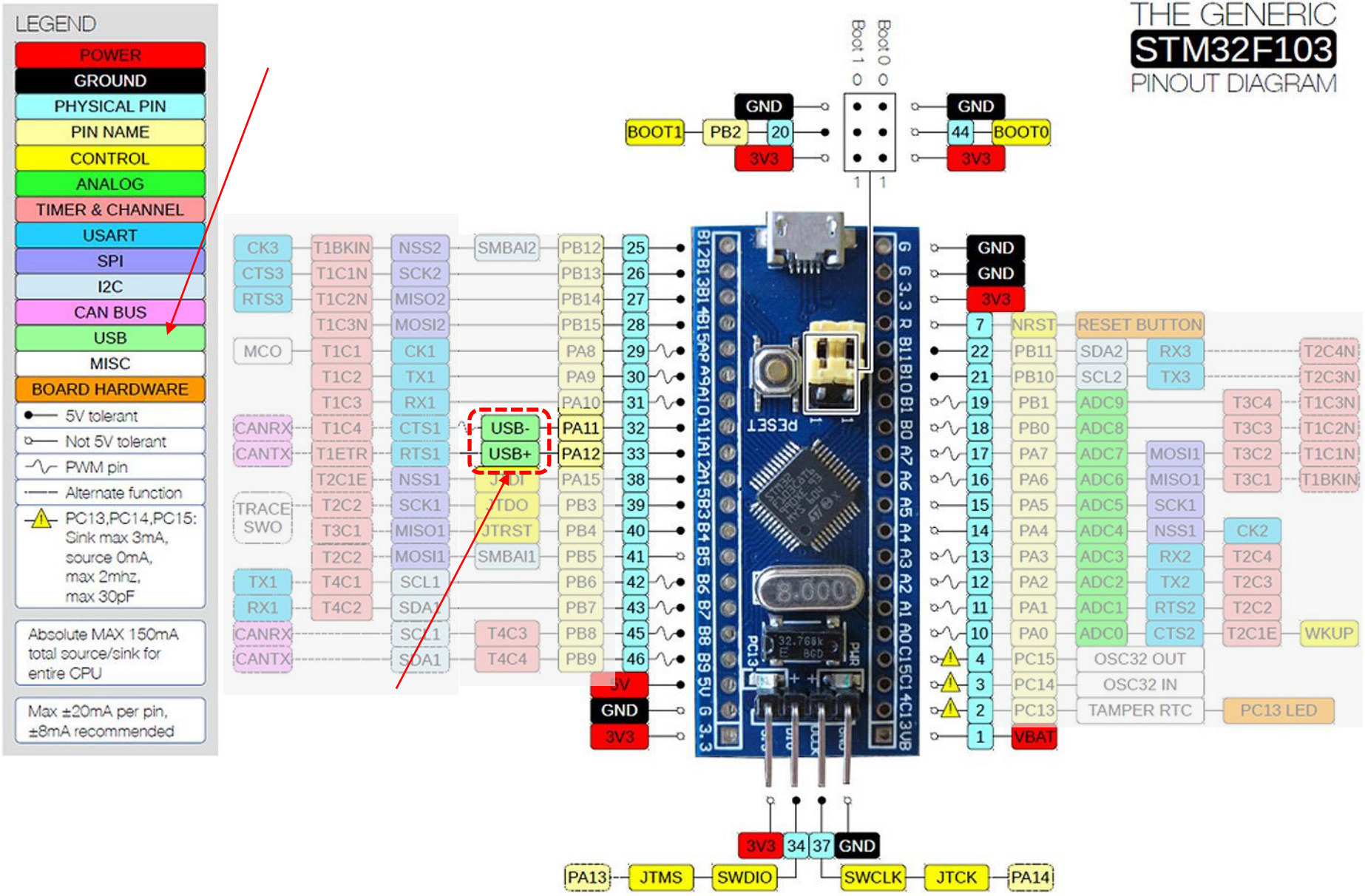
- 4 tipos de transferência de dados:
  - ✓ Control
  - ✓ Isochronous
  - ✓ Bulk
  - ✓ Interrupt

- O protocolo prevê 5 tipos de packets:
  - ✓ Handshake packets
  - ✓ Token packets
  - ✓ Data packets
  - ✓ PRE packets
  - ✓ Start of Frame packets

Token Packet

Sync	PID	ADDR	ENDP	CRC5	EOP
	8 bits	7 bits	4 bits	5 bits	

Usado para os pacotes de setup, out, in e PING. Esse pacote é sempre o 1º em uma transação, identificando o dispositivo alvo e o propósito da transação.


LEGEND
POWER
GROUND
PHYSICAL PIN
PIN NAME
CONTROL
ANALOG
TIMER & CHANNEL
USART
SPI
I2C
CAN BUS
USB
MISC
BOARD HARDWARE
5V tolerant
Not 5V tolerant
PWM pin
Alternate function
PC13,PC14,PC15: Sink max 3mA, source 0mA, max 2mhz, max 30pF
Absolute MAX 150mA total source/sink for entire CPU
Max ±20mA per pin, ±8mA recommended
THE GENERIC
STM32F103
PINOUT DIAGRAM
Boot 1
Boot 0
BOOT1
PB2
20
GND
3V3
44
BOOT0
USB-
USB+
PA11
PA12
RESET BUTTON
NRST
8.000
32.768k
PC13
PWR
CK3
CTS3
RTS3
MCO
CANRX
CANTX
TRACE SWO
TX1
RX1
T1BKIN
T1C1N
T1C2N
T1C3N
T1C1
T1C2
T1C3
T1C4
T1ETR
T2C1E
T2C2
T3C1
T2C2
T4C1
T4C2
NSS2
SCK2
MISO2
MOSI2
CK1
TX1
RX1
CTS1
RTS1
NSS1
SCK1
MISO1
MOSI1
SCL1
SDA1
SCL1
SDA1
SMBAI2
JTDI
JTDO
JTRST
SMBAI1
T4C3
T4C4
PB12
PB13
PB14
PB15
PA8
PA9
PA10
PB3
PB4
PB5
PB6
PB7
PB8
PB9
25
26
27
28
29
30
31
32
33
38
39
40
41
42
43
45
46
3V3
GND
3V3
GND
GND
3V3
7
22
21
19
18
17
16
15
14
13
12
11
10
4
3
2
1
VBAT
PB11
PB10
PB1
PB0
PA7
PA6
PA5
PA4
PA3
PA2
PA1
PA0
PC15
PC14
PC13
SDA2
SCL2
ADC9
ADC8
ADC7
ADC6
ADC5
ADC4
ADC3
ADC2
ADC1
ADC0
OSC32 OUT
OSC32 IN
TAMPER RTC
RX3
TX3
MOSI1
MISO1
SCK1
NSS1
RX2
TX2
RTS2
CTS2
T3C4
T3C3
T3C2
T3C1
CK2
T2C4
T2C3
T2C2
T2C1E
T2C4N
T2C3N
T1C3N
T1C2N
T1C1N
T1BKIN
WKUP
PC13 LED
3V3
34
37
GND
PA13
JTMS
SWDIO
SWCLK
JTCK
PA14

Interface rede CAN (overview)

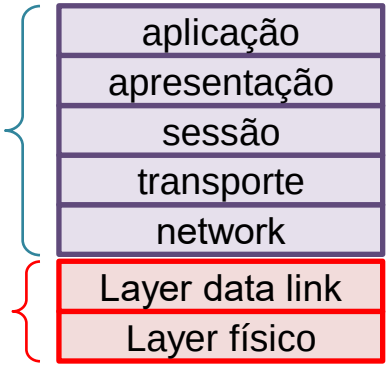
- CAN (*Controller Area Network – 1983/87 – desenvolvido pela Bosh*) é um barramento e um protocolo com múltiplos mestres, sinais diferenciais, múltiplos dispositivos, que provê um meio de comunicação entre sensores, atuadores e controladores.
- Usado inicialmente em veículos, hoje é usado em equipamentos hospitalares, elevadores, aviões, maquinas industriais e agrícolas, veículos e equipamentos militares, etc.
- Trata-se de uma forma barata de instanciar uma rede confiável, robusta e bastante testada.
- Satisfaz os requerimentos de tempo real de muitas aplicações.

- CAN 2.0 A (*formato padrão, identificador de 11 bits*) e CAN 2.0 B (*formato extendido – 29 bits*)
- 2 padrões internacionais (*ISO11898-2 high speed 1 Mb/s e ISO11898-3 low speed 125 Kb/s*)

- CAN DEFINE 2 LAYERS MAIS BÁSICOS:

Vários padrões industriais definem os layers superiores da rede: DeviceNET, CANOpen, CANAerospace, ISO1192, etc.

CAN define os dois layers mais básicos da rede

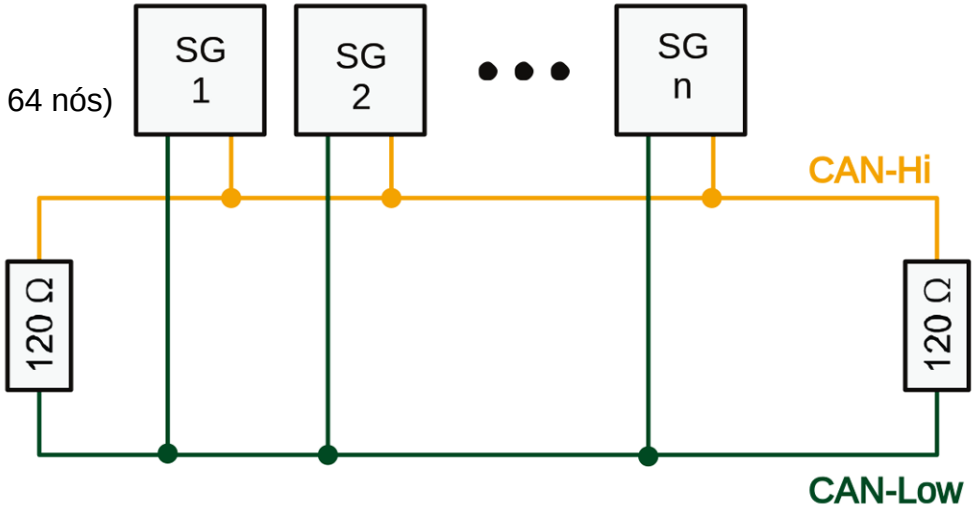


- CAN usa o princípio de BROADCAST – enviar mensagens para todos os “nós”.
- Cada nó pode enviar uma mensagem broadcast
- Toda mensagem tem um campo de ID que identifica a ***fonte*** ou o ***conteúdo*** da mensagem
- Cada nó (receptor) decide se vai ignorar ou processar a uma certa mensagem.
- Comprimento do barramento:
  - ✓ **1 Mb/s – até 40 ms**
  - ✓ **500 Kb/s – até 100 ms**
  - ✓ **250 Kb/s – até 200 ms**
  - ✓ **125 Kb/s – até 500 ms**

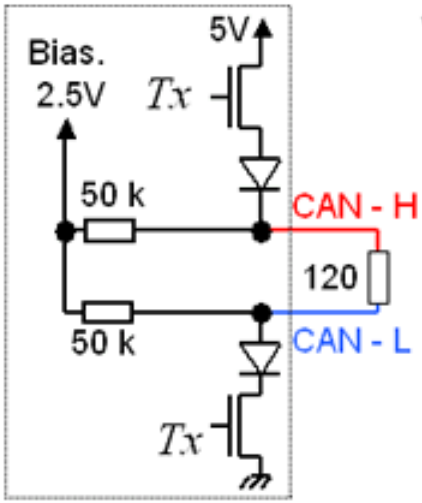
- CAN usa o princípio de BROADCAST – enviar mensagens para todos os “nós”.
- Cada nó pode enviar uma mensagem broadcast
- Toda mensagem tem um campo de ID que identifica a ***fonte*** ou o ***conteúdo*** da mensagem
- Cada nó (receptor) decide se vai ignorar ou processar a uma certa mensagem.
- Comprimento do barramento:
  - ✓ **1 Mb/s – até 40 ms**
  - ✓ **500 Kb/s – até 100 ms**
  - ✓ **250 Kb/s – até 200 ms**
  - ✓ **125 Kb/s – até 500 ms**

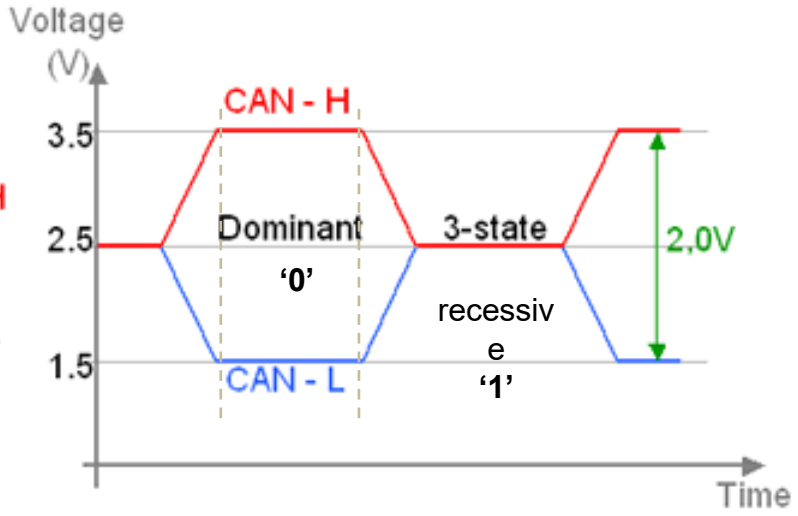
- Topologia do barramento:

(max num nós definido p/ características elétricas ≈ 64 nós)

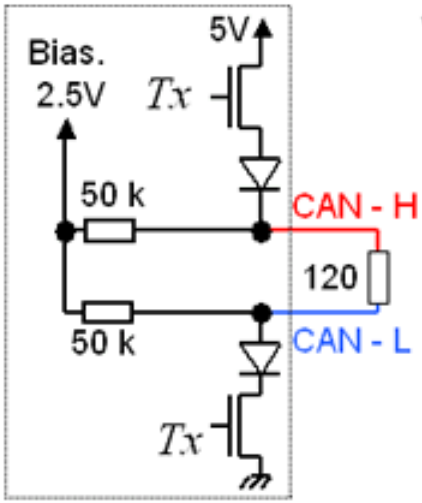


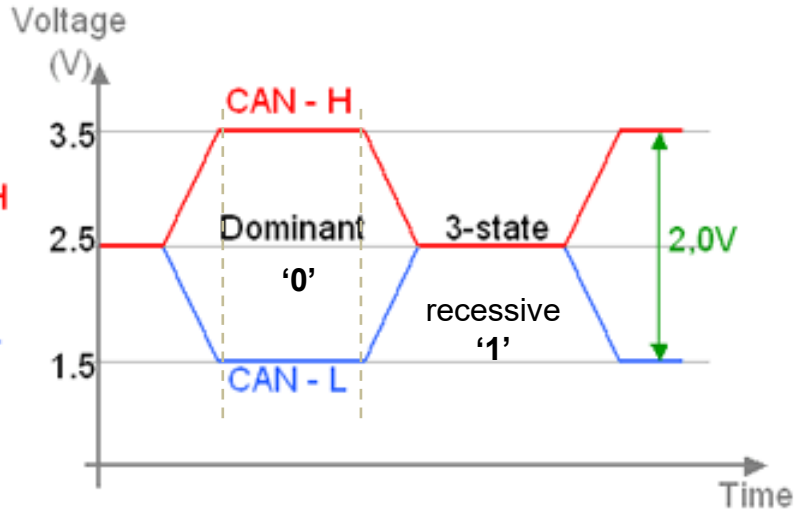
- Níveis dos sinais diferenciais:



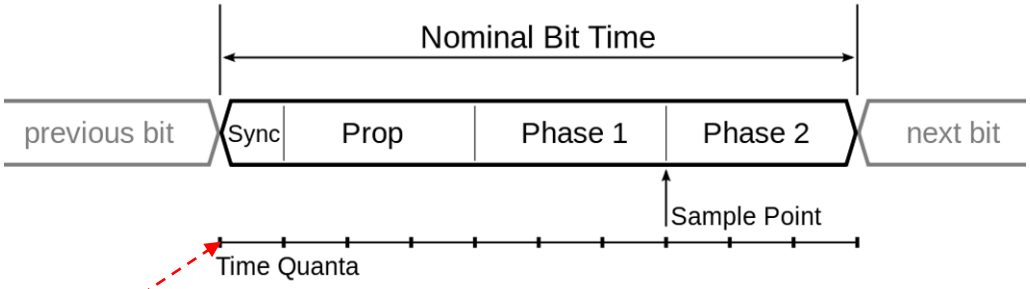


- Níveis dos sinais diferenciais:





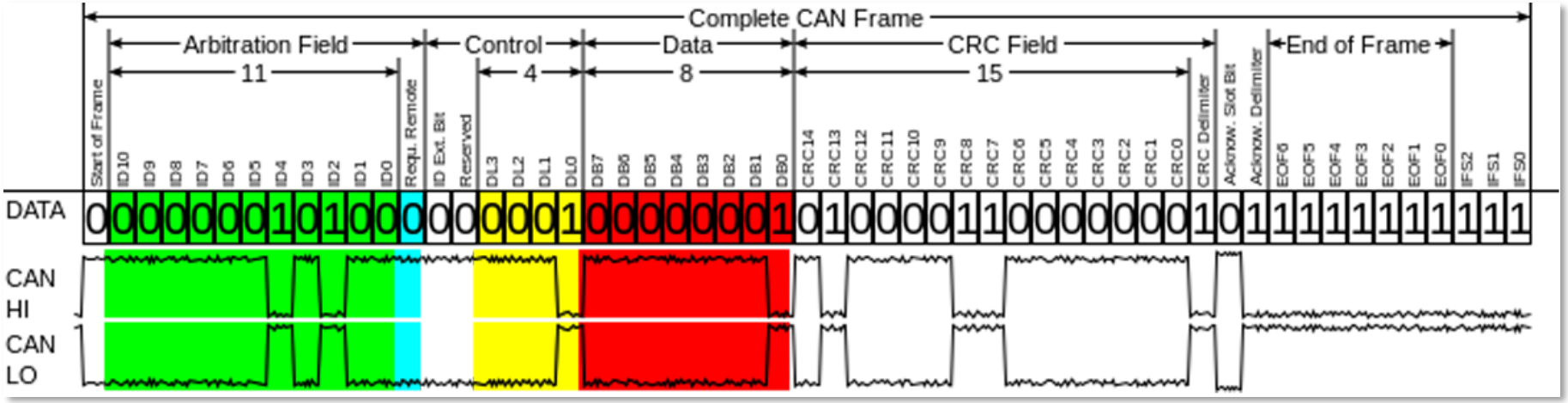
- CAN tem apenas 2 fios, e a sincronização é crítica. Cada nó usa lógica local p/ sincronizar com o barramento a cada transição de *recessiva* para *dominante* (de 3-state para '0')
- Não mais que *cinco bits consecutivos do mesmo valor* são enviados (erro de *stuff*) p/ sincronização.



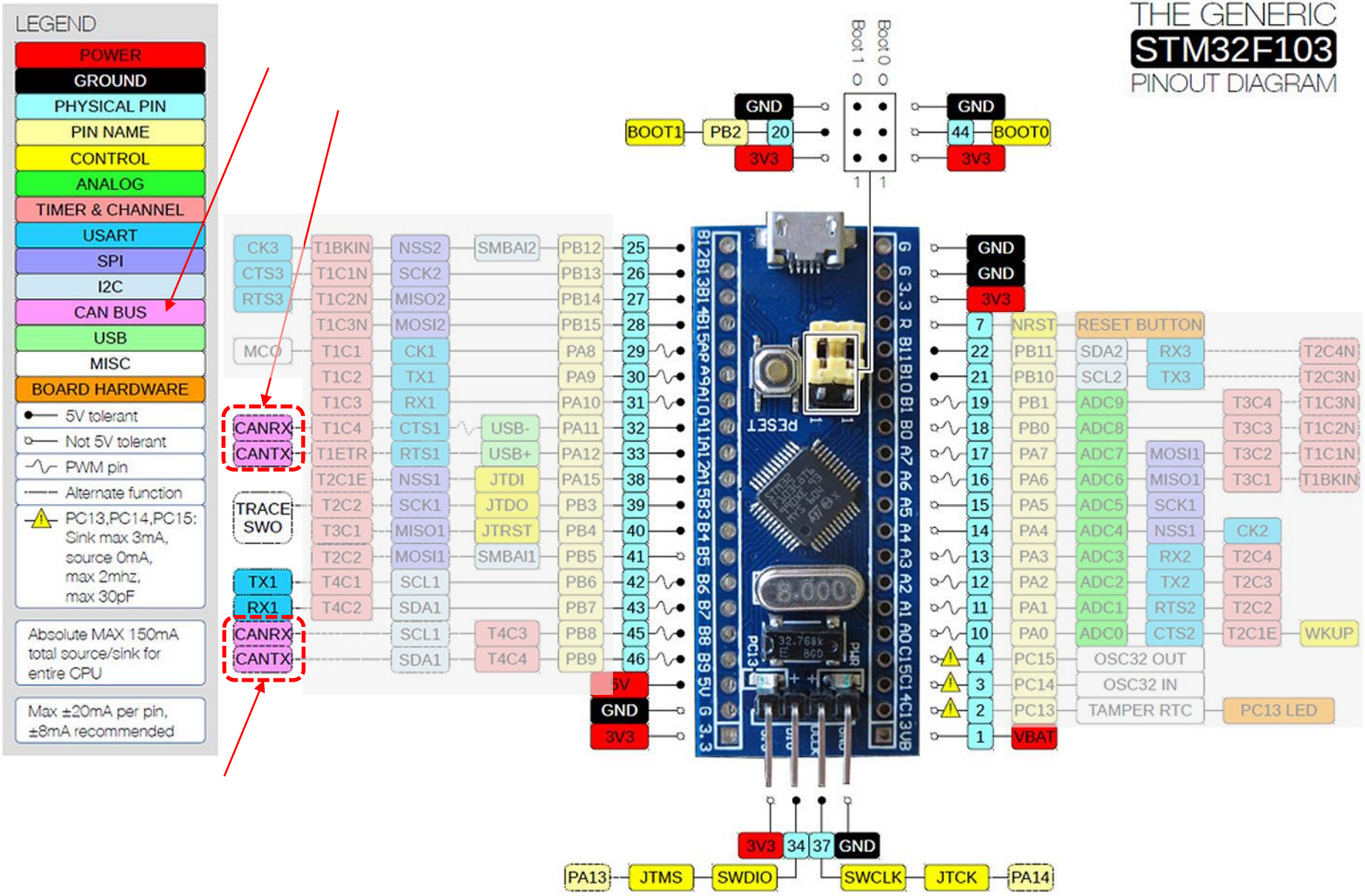
clock local (do nó) que provê N 'quanta' time para resincronizar cada bit
(deve ser algum múltiplo do bit-rate)

- Como tem apenas dois fios, CAN tem que ter um ***protocolo sofisticado*** e ***arbitragem***...
- Além disso, as capacidades de detecção de erro são apuradas (*bit error, acknlowledge error, Form error, CRC errors, Stuff errors*);
- 4 tipos de mensagens podem ser circular no bus:
  - ✓ ***Data Frame*** (usado para transmitir dados)
  - ✓ ***Remote Frame*** (usado para requisitar transmissão de dados a um nó remoto)
  - ✓ ***Error Frame*** (enviado por um nó que detecta um erro de transmissão ou protocolo)
  - ✓ ***Overload Frame*** (enviado por um nó para requisitar um atraso na transmissão)

- Formato básico de um frame CAN:



Field name	Length (bits)	Purpose
Start-of-frame	1	Denotes the start of frame transmission
Identifier (green)	11	A (unique) identifier which also represents the message priority
Remote transmission request (RTR) (blue)	1	Must be dominant (0) for data frames and recessive (1) for remote request frames (see Remote Frame, below)
Identifier extension bit (IDE)	1	Must be dominant (0) for base frame format with 11-bit identifiers
Reserved bit (r0)	1	Reserved bit. Must be dominant (0), but accepted as either dominant or recessive.
Data length code (DLC) (yellow)	4	Number of bytes of data (0–8 bytes)[a]
Data field (red)	0–64 (0-8 bytes)	Data to be transmitted (length in bytes dictated by DLC field)
CRC	15	Cyclic redundancy check
CRC delimiter	1	Must be recessive (1)
ACK slot	1	Transmitter sends recessive (1) and any receiver can assert a dominant (0)
ACK delimiter	1	Must be recessive (1)
End-of-frame (EOF)	7	Must be recessive (1)


LEGEND
POWER
GROUND
PHYSICAL PIN
PIN NAME
CONTROL
ANALOG
TIMER & CHANNEL
USART
SPI
I2C
CAN BUS
USB
MISC
BOARD HARDWARE
5V tolerant
Not 5V tolerant
PWM pin
Alternate function
PC13,PC14,PC15: Sink max 3mA, source 0mA, max 2mhz, max 30pF
Absolute MAX 150mA total source/sink for entire CPU
Max ±20mA per pin, ±8mA recommended
THE GENERIC
STM32F103
PINOUT DIAGRAM
Boot 1
Boot 0
GND
BOOT1
PB2
20
3V3
GND
44
BOOT0
3V3
CANRX
CANTX
TRACE SWO
TX1
RX1
CANRX
CANTX
CK3
CTS3
RTS3
MCO
T1BKIN
T1C1N
T1C2N
T1C3N
T1C1
T1C2
T1C3
T1C4
T1ETR
T2C1E
T2C2
T3C1
T2C2
T4C1
T4C2
NSS2
SCK2
MISO2
MOSI2
CK1
TX1
RX1
CTS1
RTS1
NSS1
SCK1
MISO1
MOSI1
SCL1
SDA1
SCL1
SDA1
SMBAI2
USB-
USB+
JTDI
JTDO
JTRST
SMBAI1
T4C3
T4C4
PB12
PB13
PB14
PB15
PA8
PA9
PA10
PA11
PA12
PA15
PB3
PB4
PB5
PB6
PB7
PB8
PB9
5V
GND
3V3
25
26
27
28
29
30
31
32
33
38
39
40
41
42
43
45
46
B12 B13 B14 B15 A8 A9 A10 A11 A12 A15 B3 B4 B5 B6 B7 B8 B9 5V G 3.3
G G 3.3 R B11 B10 B1 B0 A7 A6 A5 A4 A3 A2 A1 A0 C15 C14 C13 VB
PC13
PWR
GND
GND
3V3
7
22
21
19
18
17
16
15
14
13
12
11
10
4
3
2
1
NRST
PB11
PB10
PB1
PB0
PA7
PA6
PA5
PA4
PA3
PA2
PA1
PA0
PC15
PC14
PC13
VBAT
RESET BUTTON
SDA2
SCL2
ADC9
ADC8
ADC7
ADC6
ADC5
ADC4
ADC3
ADC2
ADC1
ADC0
OSC32 OUT
OSC32 IN
TAMPER RTC
RX3
TX3
MOSI1
MISO1
SCK1
NSS1
RX2
TX2
RTS2
CTS2
T3C4
T3C3
T3C2
T3C1
CK2
T2C4
T2C3
T2C2
T2C1E
PC13 LED
T2C4N
T2C3N
T1C3N
T1C2N
T1C1N
T1BKIN
WKUP
3V3
34
37
GND
PA13
JTMS
SWDIO
SWCLK
JTCK
PA14

- Há várias formas de comunicação sem fio (***wireless*** - que fogem ao escopo dessa disciplina)
- **Bluetooth** - especificação de rede sem fio de âmbito pessoal (*Wireless personal area networks* – PANs); provê uma maneira de conectar e trocar informações entre dispositivos como notebooks, computadores, impressoras, celulares, câmeras digitais, etc., através de uma frequência de rádio de curto alcance globalmente licenciada e segura.
- **Infrared Data Association** (**IrDA**) - padrões de comunicação entre equipamentos de comunicação wireless que usam infravermelho, permite conexão de dispositivos sem fio ao microcomputador (ou equipamento com tecnologia apropriada), tais como impressoras, telefones celulares, notebooks e PDAs.
- ***Wi-Fi*** - marca de rede local sem fios (WLAN) referente ao padrão IEEE 802.11. Trata-se de uma rede que segue protocolo de redes de computadores. opera em faixas de frequências que não necessitam de licença para instalação e/ou operação.
- **Zigbee** - padrão de rede sem fio para arquitetura em malha de baixo custo, baixa potência, para a comunicação sem fio entre dispositivos eletrônicos que não requerem altas taxas de transmissão de dados. Podem ser ligadas em malha (ver Zigbee Mesh) e cada nó pode retransmitir mensagens. Existem três tipos diferentes de dispositivos ZigBee: ***ZigBee coordenador*** (ZC - o dispositivo mais completo), ***ZigBee Router*** (ZR – pode funcionar como um roteador intermediário, ***ZigBee dispositivo final*** (ZED – com funcionalidade apenas para falar com o nó pai, seja um coordenador ou um roteador).

Obrigado...

Até a próxima aula.