Aulas 15 e 16

- A interface I^2C (*Inter-Integrated Circuit*)
- Caraterísticas básicas
- Sinalização
- Endereçamento
- Transferência de dados
- Múltiplos Masters
  - Sincronização dos relógios
  - Arbitragem

José Luís Azevedo, Bernardo Cunha, Tomás O. Silva, P. Bartolomeu

I^2C – Introdução

- Desenvolvido pela Philips Semiconductors (agora NXP Semiconductors)
  - Versão 1 em 1992
  - Atualmente na revisão 7 (Outubro de 2021)
- De acordo com a NXP: "simple bidirectional 2-wire bus for efficient inter-IC control"
  - Requer apenas duas linhas
  - Implementável em hardware e/ou software
  - Desenvolvido inicialmente para controlo de subsistemas em TVs
- Transações "master-slave" com opção "multi-master" (requer arbitragem)
- Taxas de transmissão
  - Standard mode: até 100 kbit/s
  - Fast mode: até 400 kbit/s
  - Fast mode plus: até 1 Mbit/s
  - High Speed: até 3,4 Mbit/s
  - Ultra-fast mode: até 5 Mbit/s

I^2C – Introdução

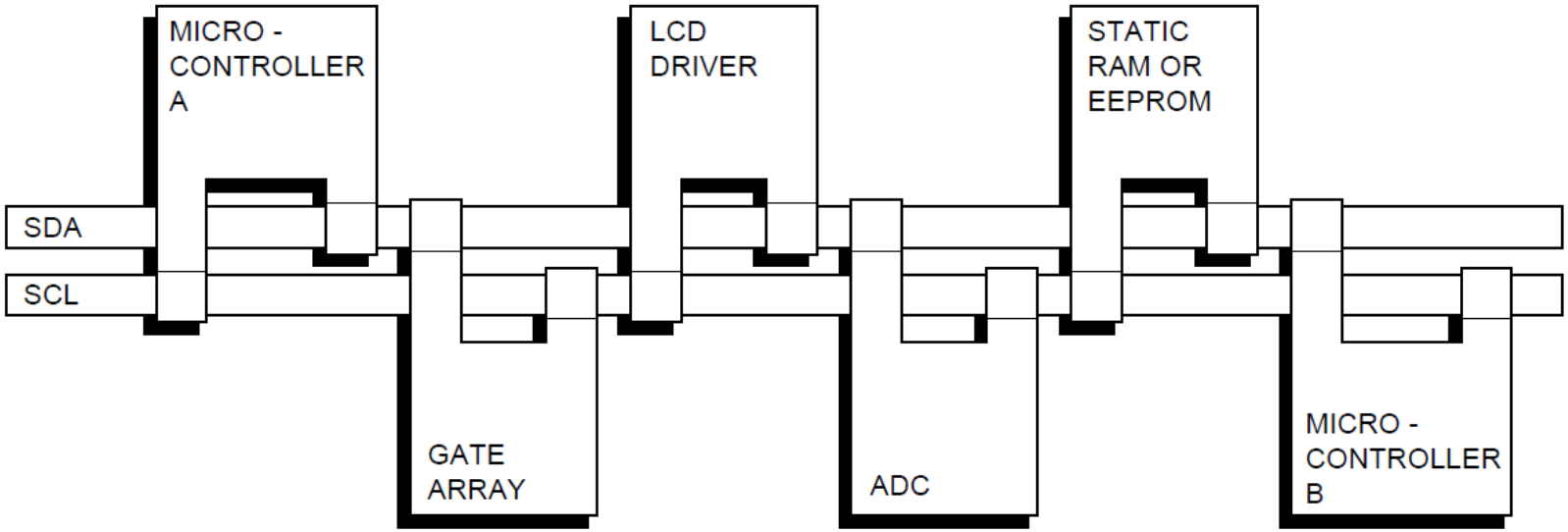
- Dada a sua simplicidade, versatilidade e economia de recursos, o I^2C encontra-se em diversas áreas de aplicação:
  - Sensores, DACs, ADCs
    - sensores de temperatura, sensores de pressão, acelerómetros, giroscópios, …
  - Memória externa em microcontroladores (RAM, EEPROM/FLASH)
    - Armazenamento de dados e/ou configurações
  - Controlo de periféricos e componentes em eletrónica de consumo
    - Ecrãs OLED, LCD, …
  - Monitorização de hardware
    - ex. temperatura de CPUs, controlo da velocidade da ventoinha em motherboards, gestão energética de baterias
  - Interface com relógios de tempo real (RTC)

I^2C – Caraterísticas básicas

- Comunicação série síncrona, bidirecional *half-duplex*, orientada ao byte
- A comunicação segue um modelo *master*/*slave*, no qual o *master* controla a transferência e um *slave* responde
- O *master* pode atuar com transmissor ou como recetor, dependendo do contexto da comunicação
- O barramento de comunicação utiliza apenas duas linhas de sinal:
  - Serial data line (SDA)
  - Serial clock line (SCL)
- Cada dispositivo é endereçado por software através de um identificador único de 7 ou 10 bits, definido pelo fabricante
- No modo "standard" (7 bits), há 128 endereços possíveis, dos quais 16 são reservados e 112 estão disponíveis para uso
- O barramento suporta múltiplos *masters* ("multi-master"); implementa deteção de colisões e arbitragem, garantindo que, se dois ou mais *masters* iniciarem uma transmissão simultaneamente, apenas um continuará sem que haja corrupção dos dados

Exemplo de interligação num barramento I²C

- Barramento a dois fios
  - Serial data line (SDA)
  - Serial clock line (SCL)



De: I²C-bus specification and user manual, Rev. 6 — 4 April 2014

http://www.nxp.com/documents/user_manual/UM10204.pdf

I^2C – terminologia

- **Transmitter** – dispositivo que envia dados para o barramento
- **Receiver** – dispositivo que recebe dados do barramento
- **Master** – dispositivo que inicia e controla a transferência de dados, gerando o sinal de relógio (SCL) e determinando o início e fim da comunicação
- **Slave** – o dispositivo endereçado pelo *master*
- **Multi-master** – permite que múltiplos dispositivos atuem como *masters*, tentando simultaneamente controlar o barramento sem corromper a comunicação em curso
- **Arbitragem** – mecanismo que assegura que, se mais do que um *master* tentar controlar o barramento simultaneamente, apenas um poderá continuar, sem que haja perturbação da comunicação iniciada pelo *master* vencedor
- **Sincronização** – procedimento para sincronizar os sinais de relógio de dois ou mais *masters*, garantindo um funcionamento coordenado do barramento

Masters e *Slaves*

- O *master*
  - Controla a linha SCL (Serial Clock) durante a comunicação
  - Inicia e termina a transferência de dados
  - Controla o endereçamento dos outros dispositivos

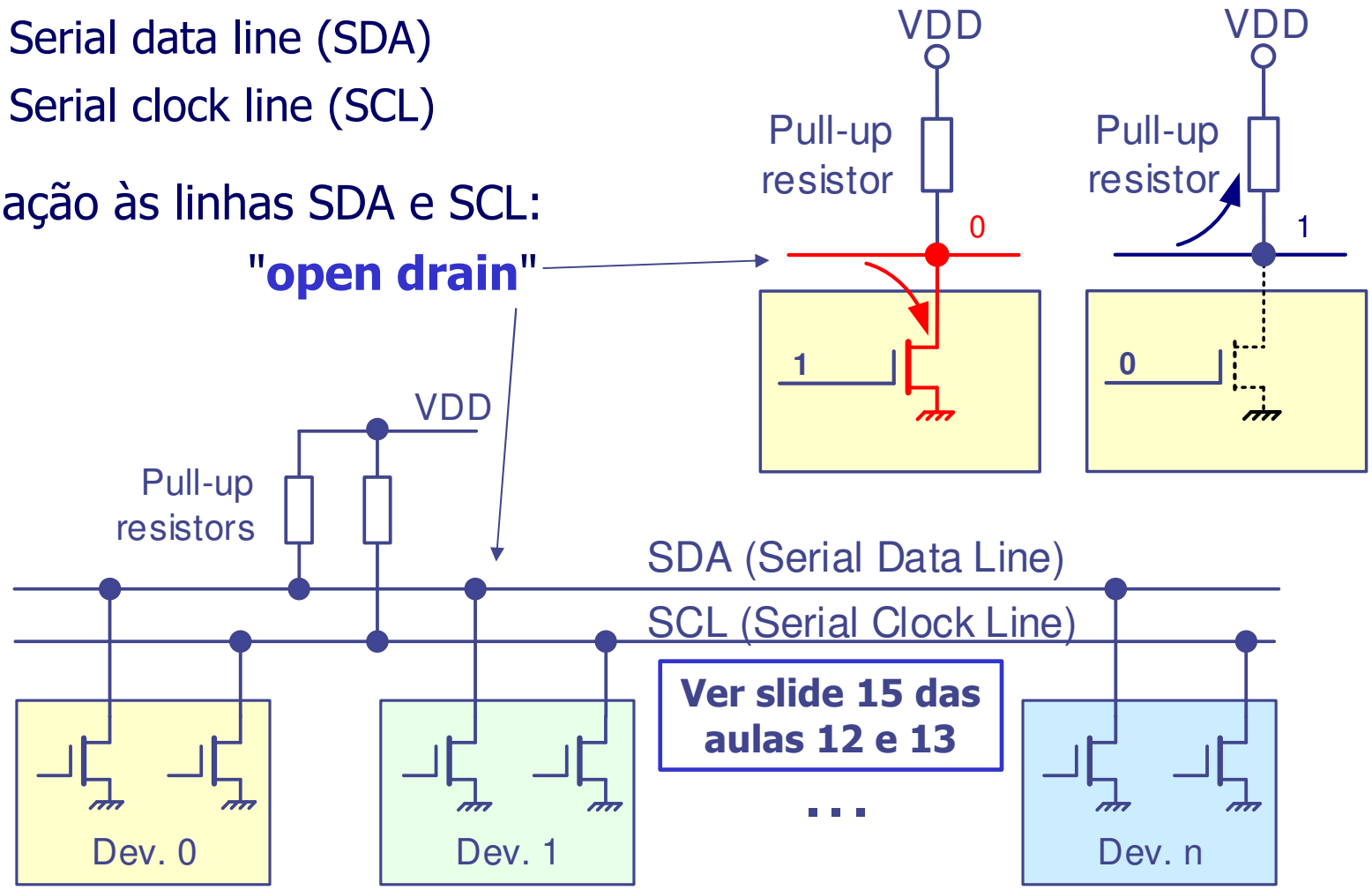
- O *slave*
  - É o dispositivo endereçado pelo *master*
  - Pode condicionar o estado da linha SCL: o *slave* pode gerar uma condição de "clock stretch", mantendo a linha SCL a zero para controlar o ritmo da comunicação e dar mais tempo para processar dados

- Transmissor / Recetor
  - *Master* ou *slave*
  - Um *master* transmissor envia dados para um *slave* recetor (escrita)
  - Um *master* recetor lê dados de um *slave* transmissor (leitura)

I²C – Sinalização

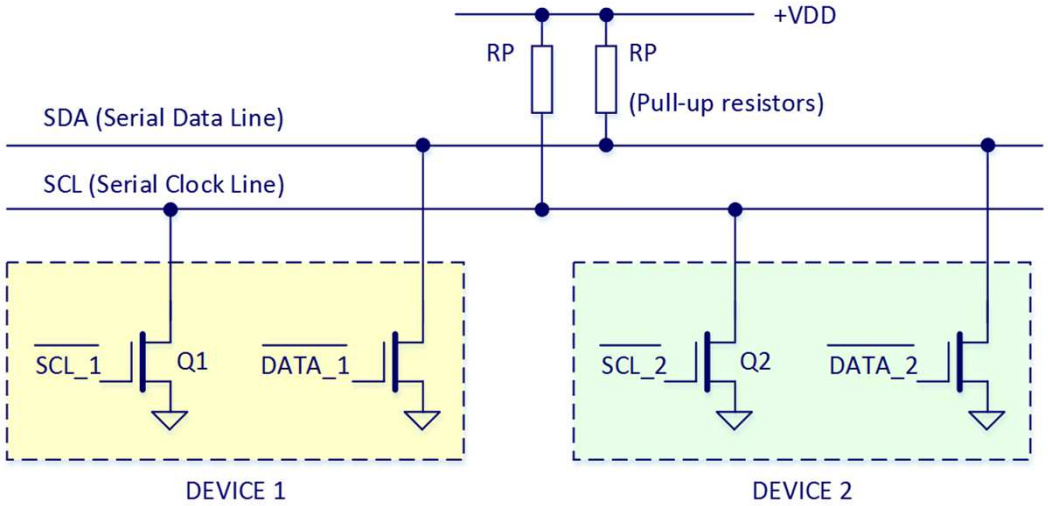
- Barramento a dois fios
  - Serial data line (SDA)
  - Serial clock line (SCL)
- Ligação às linhas SDA e SCL:

"**open drain**"



I²C – Sinalização

- As linhas de clock e dados de cada dispositivo são "wire ANDed" com os respetivos sinais do barramento



SCL_1	SCL_2	SCL
0 (Q1 ON)	0 (Q2 ON)	0 (**D**)
0 (Q1 ON)	1 (Q2 OFF)	0 (**D**)
1 (Q1 OFF)	0 (Q2 ON)	0 (**D**)
1 (Q1 OFF)	1 (Q2 OFF)	1 (**R**)

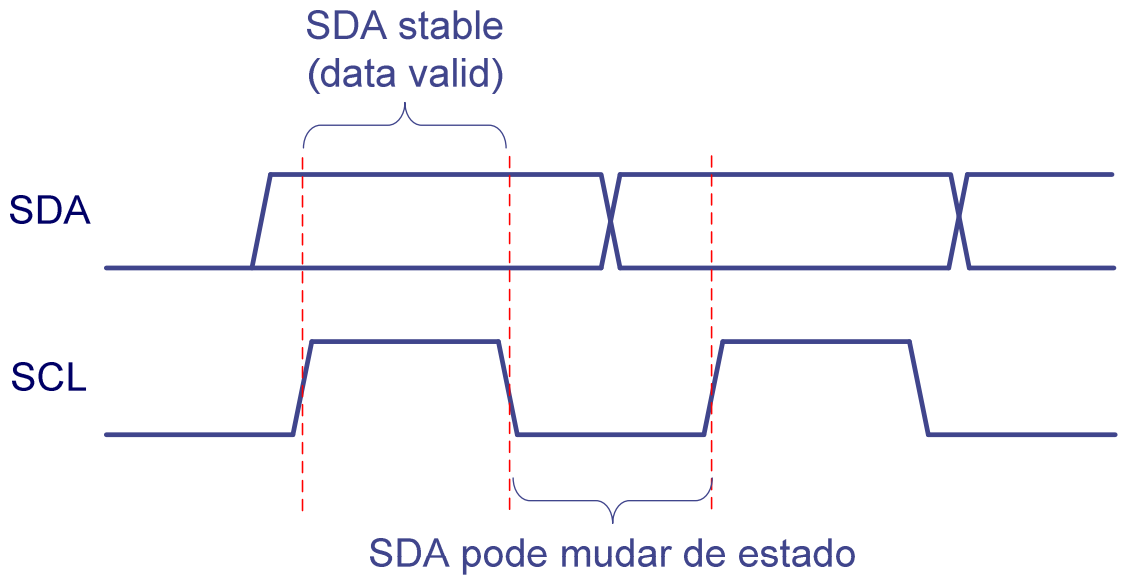
- Na ausência de bit dominante a linha respetiva está no nível lógico 1 (Recessivo), imposto através da resistência de *pull-up* RP
- Este esquema permite usar "bit recessivo" (1) e "bit dominante" (0) para várias sinalizações

Endereçamento

- O primeiro byte transmitido pelo *master* contém:
  - 7 bits: **endereço do *slave***
  - 1 bit: qualificação da operação (RD / WR\)
- Qualificador da operação:
  - **RD/WR\ = 0**: o *master* é o **transmissor** (escreve dados na linha SDA)
  - **RD/WR\ = 1**: o *master* é o **recetor** (lê dados da linha SDA)
- Cada *slave* lê o endereço da linha SDA; se o endereço lido coincide com o seu próprio endereço:
  - comuta para o estado "transmissor" se o bit RD/WR\ for igual a "1"
  - comuta para o estado "recetor" se o bit RD/WR\ for igual a "0"

I^2C – Transferência de bits

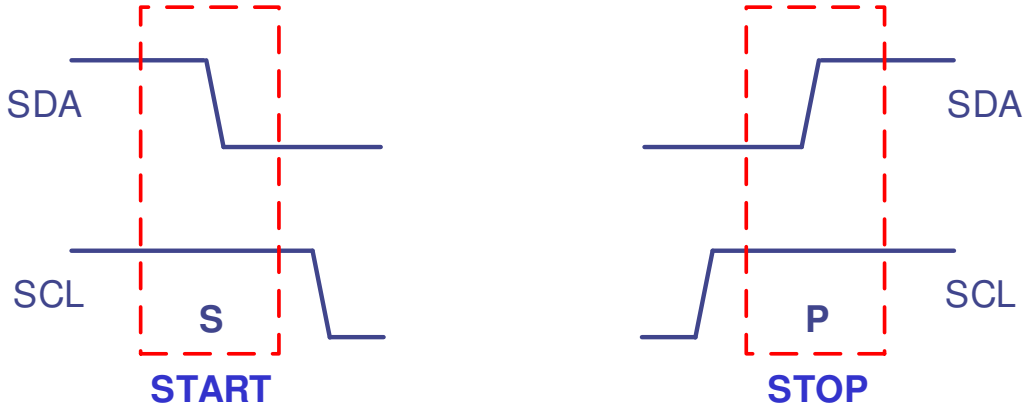
- Um período de relógio por bit de dados
- Dados (SDA) só são alterados quando SCL = 0
- SCL = 1: dados em SDA válidos



- Transições em SDA quando SCL=1 sinalizam "**condições**". As "condições" são sempre geradas pelo *master*

Símbolos (condições)

- As transações são delimitadas por dois símbolos / "condições": **START** e **STOP**, geradas pelo *master*
- As condições START/STOP são sinalizadas por meio de uma sequência que viola as regras normais de transferência de dados



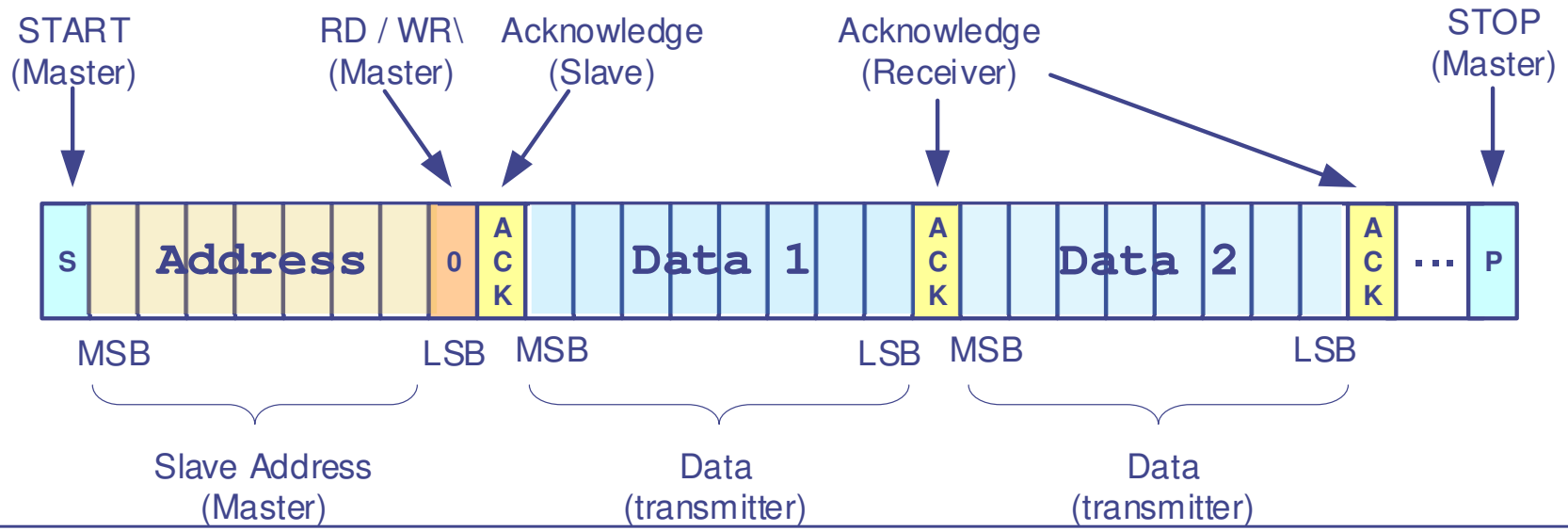
- Condição **START**: Transição de 1→0 em SDA quando SCL = 1
- Condição **STOP**: Transição de 0→1 em SDA quando SCL = 1
- Estado do barramento:
  - **Ocupado**: após um START (S) ou START repetidos (Sr), até ao próximo STOP
  - **Livre**: após um STOP (P), até ao próximo START

Transferência de dados

- A transferência é orientada ao byte (8 bits) sendo transmitido, em primeiro lugar, o bit mais significativo (MSbit)
- No início de uma transferência, o *master*:
  - Envia um **START** (S)
  - De seguida envia o **endereço** do slave (7 bits) e o bit de qualificação da operação (Read / Write\ - RD/WR\)
- Após o 8º bit (o LSbit, correspondente ao bit RD/WR\), o *slave* endereçado gera um *acknowledge* (**ACK**) na linha SDA, **sob a forma de um bit dominante** (0)
- A seguir o transmissor (*master* ou *slave*) envia 1 byte de dados
- Após o 8º bit (o LSbit), o recetor gera um acknowledge (**ACK**) na linha SDA, **sob a forma de um bit dominante** (0)
- Este ciclo de 9 bits repete-se para cada byte de dados que é transferido
- No final da transferência o *master* envia um **STOP** (P)

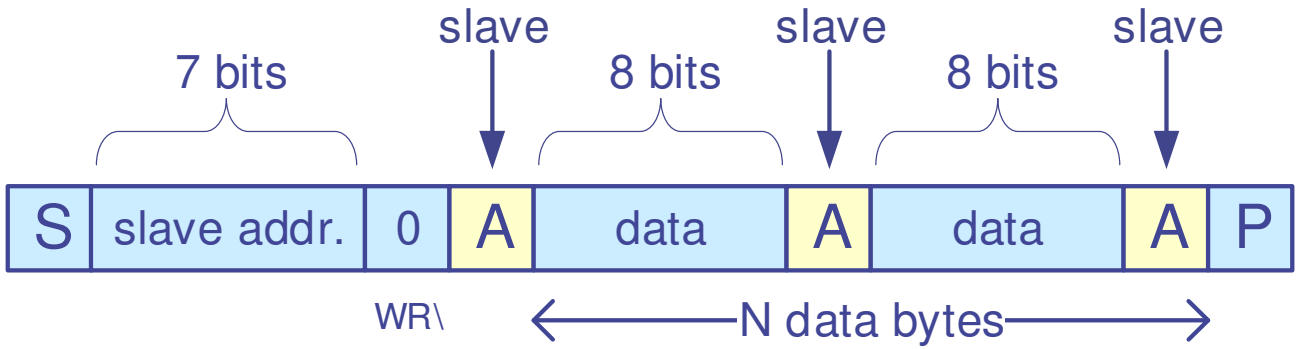
Transferência de dados – escrita

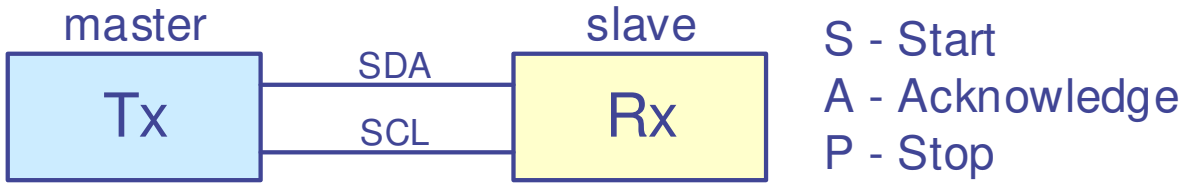
- O *master*:
  - Envia um **START** (S)
  - De seguida envia o **endereço** do slave (7 bits) e o bit de qualificação da operação (Read / Write\ - RD/WR\ = 0)
- O *slave* endereçado faz o **acknowledge** (ACK) na *slot* seguinte
- Nos 8 ciclos de relógio seguintes o *master* envia o byte de dados e no 9º ciclo de relógio avalia o **acknowledge do *slave***. Este ciclo de 9 bits repete-se para cada byte de dados que o *master* pretenda transferir.



Transferência de dados – escrita

- Operação de **Escrita** (*master* é transmissor, *slave* é recetor)

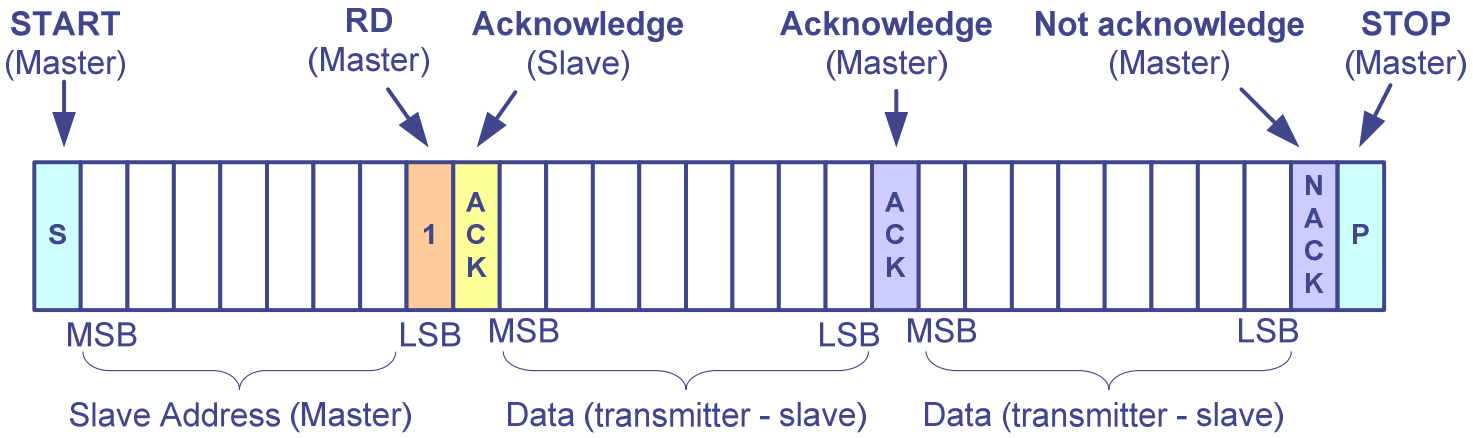




S - Start
A - Acknowledge
P - Stop

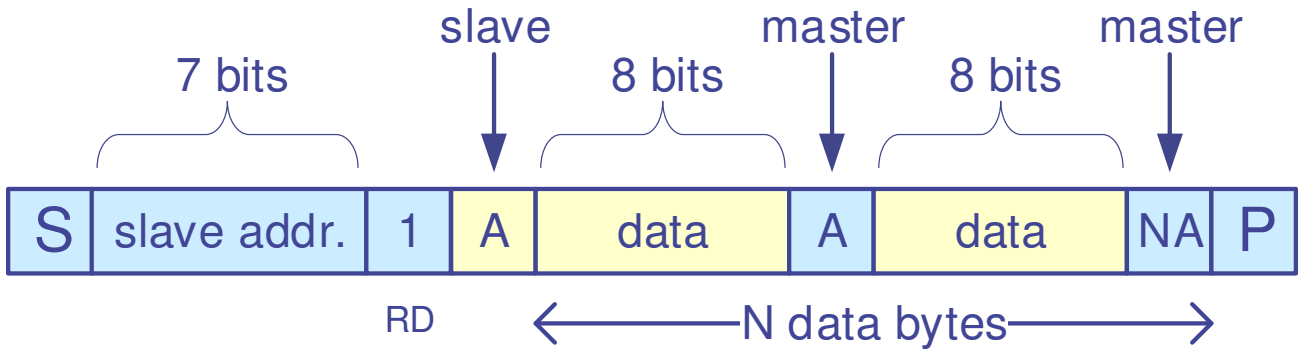
Transferência de dados – leitura

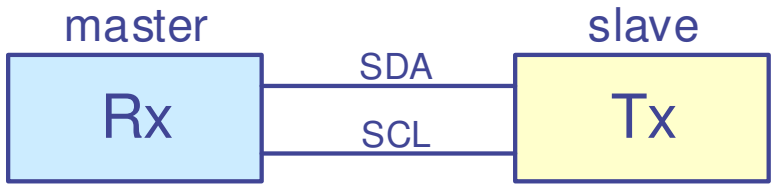
- O *master*:
  - Envia um **START** (S)
  - De seguida envia o **endereço** do slave (7 bits) e o bit de qualificação da operação (Read / Write\ - RD/WR\ = 1)
- O *slave* endereçado faz o **acknowledge** (ACK) na *slot* seguinte
- O *slave* envia 8 bits de dados
- No final da sequência de 8 bits, o *master* (que é recetor) gera:
  - um **acknowledge,** sinalizando que vai continuar a ler dados, ou
  - um **not acknowledge** informando o *slave* de que o byte recebido constitui o fim da transferência. Neste caso o *master* envia logo de seguida um STOP (P)



Transferência de dados – leitura

- Operação de **Leitura** (*master* é recetor, *slave* é transmissor)



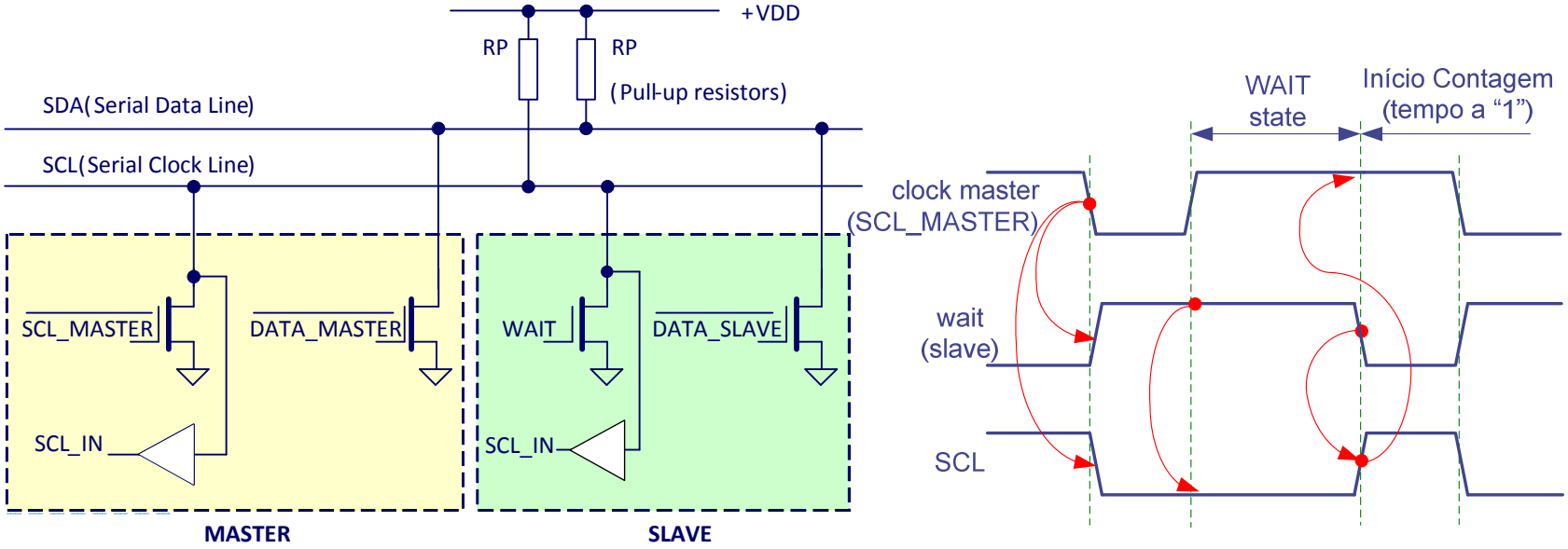


S - Start
A – Acknowledge
NA - Not Acknowledge
P - Stop

- O "not acknowledge" (NA) enviado pelo *master* sinaliza o *slave* do fim da transferência

Transferência de dados

- O *slave* pode forçar o alargamento da transferência mantendo a zero (bit dominante) a linha SCL; esta técnica designa-se por "clock stretching"
- O sinal "wait" (do *slave*) condiciona o estado da linhal SCL do barramento: enquanto "wait" estiver a 1, a linha SCL está forçada a nível lógico 0 (dominante)



- *master* fica em "wait state" enquanto **SCL_MASTER ≠ SCL_IN**

Transferência de dados - exemplo

- Exemplo de uma sequência de transferência:

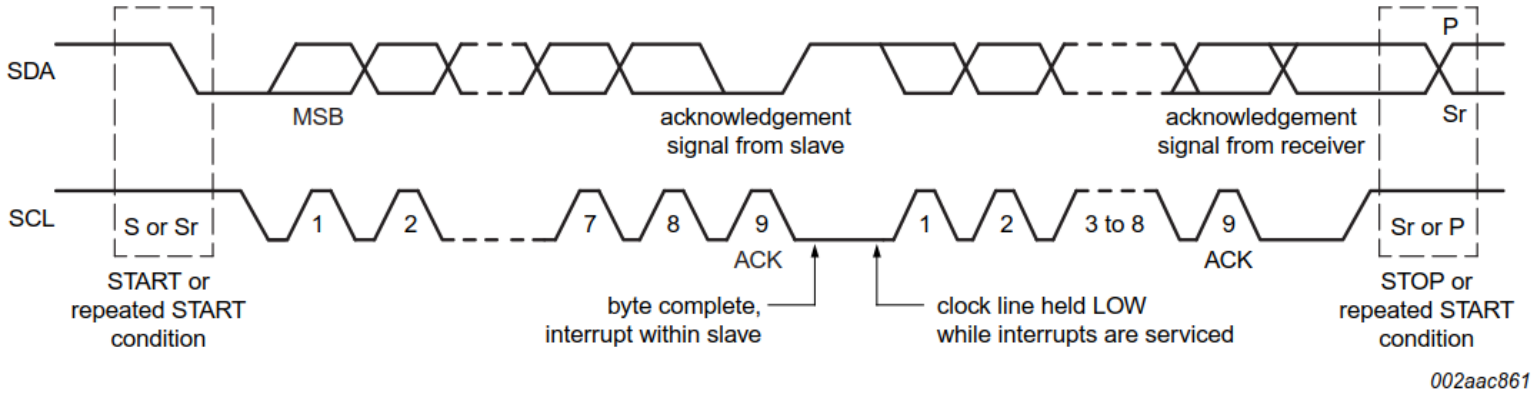


Fig 6. Data transfer on the I²C-bus

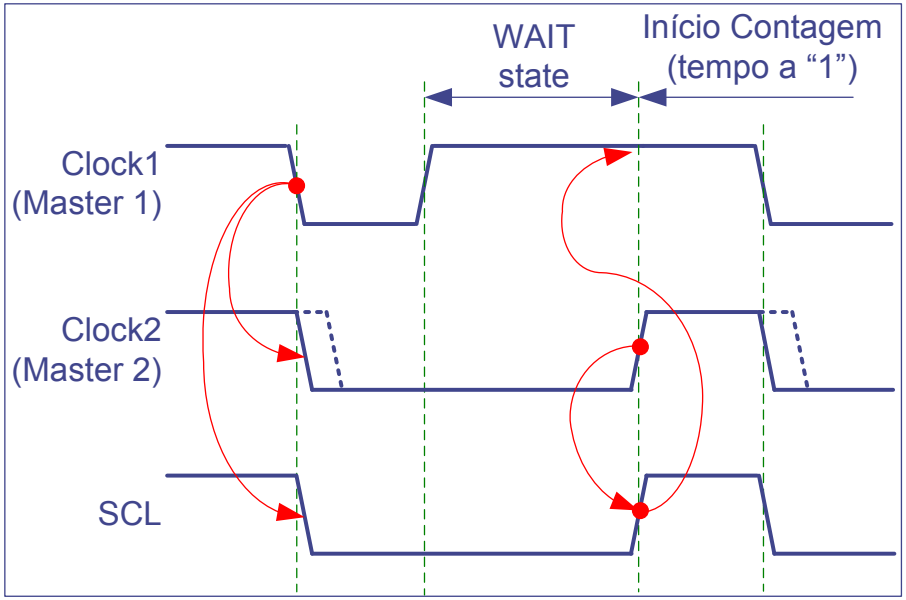
De: I²C-bus specification and user manual, Rev. 6 — 4 April 2014

Múltiplos *masters*

- Dois (ou mais) *masters* podem iniciar uma transmissão simultânea num barramento livre (isto é, após um STOP)
- É necessário um método para decidir qual dos *masters* toma o controlo do barramento e completa a transmissão – este processo é conhecido como **arbitragem de acesso ao barramento**
- O *master* que perde o processo de arbitragem retira-se e só tenta novo acesso ao barramento na próxima situação de "barramento livre" (após um STOP)
- Na gestão do acesso ao barramento é necessário:
  1. garantir que os relógios dos *masters* estão **sincronizados**
  2. um processo que defina qual o *master* que ganha o acesso ao barramento, i.e., que controla a linha SDA (**arbitragem**)
- A sincronização de relógios e a arbitragem são baseados na técnica bit dominante/bit recessivo:
  - Nível lógico "0" – bit dominante
  - Nível lógico "1" – bit recessivo (é anulado por um bit dominante)

Sincronização dos relógios dos masters (linha SCL)

- Quando a linha SCL passa de 1 -> 0 todos os *masters* colocam a 0 os seus relógios
- Os *masters* mantêm o seu relógio a 0 até o seu tempo a 0 ter chegado ao fim
- Quando um *master* termina a contagem do tempo a 0 do seu relógio liberta a linha SCL (permite que esta passe a 1)

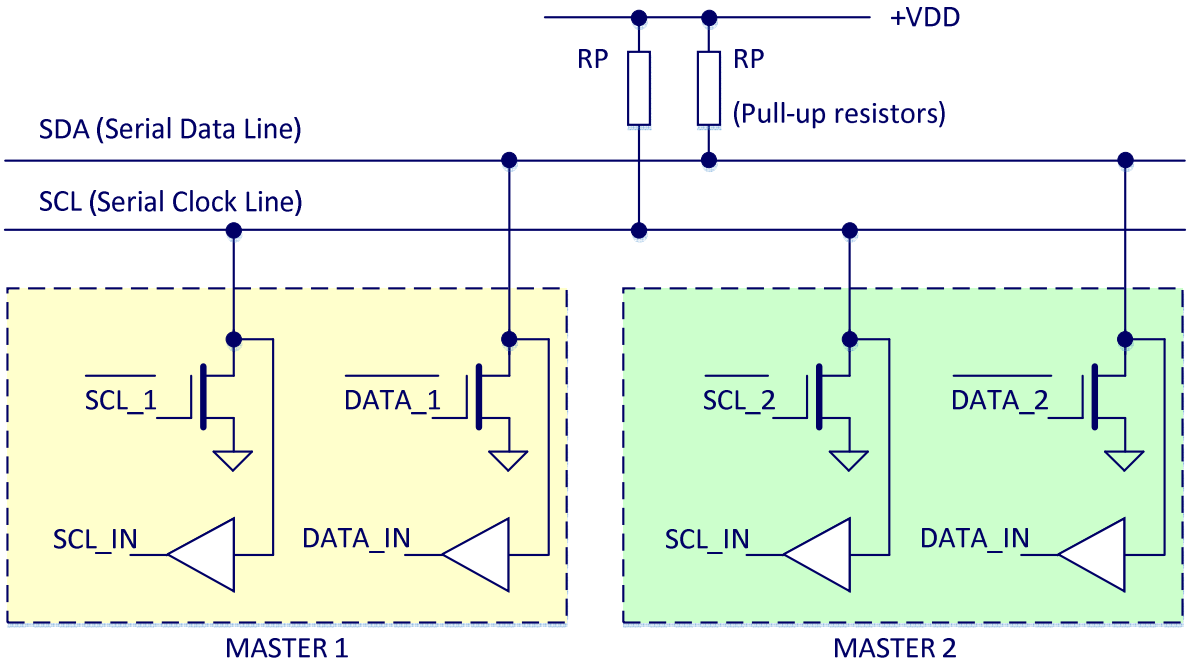


- Se SCL se mantém a 0 comuta para um estado de "wait", ficando a aguardar que a linha SCL passe a 1
- Logo que SCL passe a 1 inicia a contagem do tempo a 1 do seu relógio
- O primeiro *master* a terminar o seu tempo a 1 força a linha SCL a 0
- O sinal SCL fica sincronizado com um $\mathbf{t_{low}}$ determinado pelo *master* com maior $\mathbf{t_{low}}$ e um $\mathbf{t_{high}}$ imposto pelo master com menor $\mathbf{t_{high}}$

Arbitragem (linha SDA)

- Quando o barramento está livre ("idle") dois ou mais *masters* podem iniciar uma transferência; todos geram START resultando numa condição de START válida no barramento
  - Recordando: **arbitragem** é o mecanismo que assegura que, se mais do que um *master* tentar controlar o barramento simultaneamente, apenas um poderá continuar, sem que haja perturbação da comunicação iniciada pelo *master* vencedor
- A arbitragem é feita por bit dominante / bit recessivo e **processa-se bit a bit**
- Por cada novo bit enviado, quando a linha SCL está a 1 cada *master* lê a linha SDA e verifica se o seu valor coincide com o que enviou:
  - O processo de arbitragem é perdido por um *master* quando lê da linha nível lógico 0 (dominante) tendo enviado nível lógico 1 (recessivo)
- O *master* que perde o processo de arbitragem
  - Retira-se, libertando a linha SDA (comuta de imediato para modo *slave*)
  - Tenta de novo quando o barramento passar ao estado "idle" (espera o aparecimento de uma condição STOP)

Arbitragem (linha SDA)



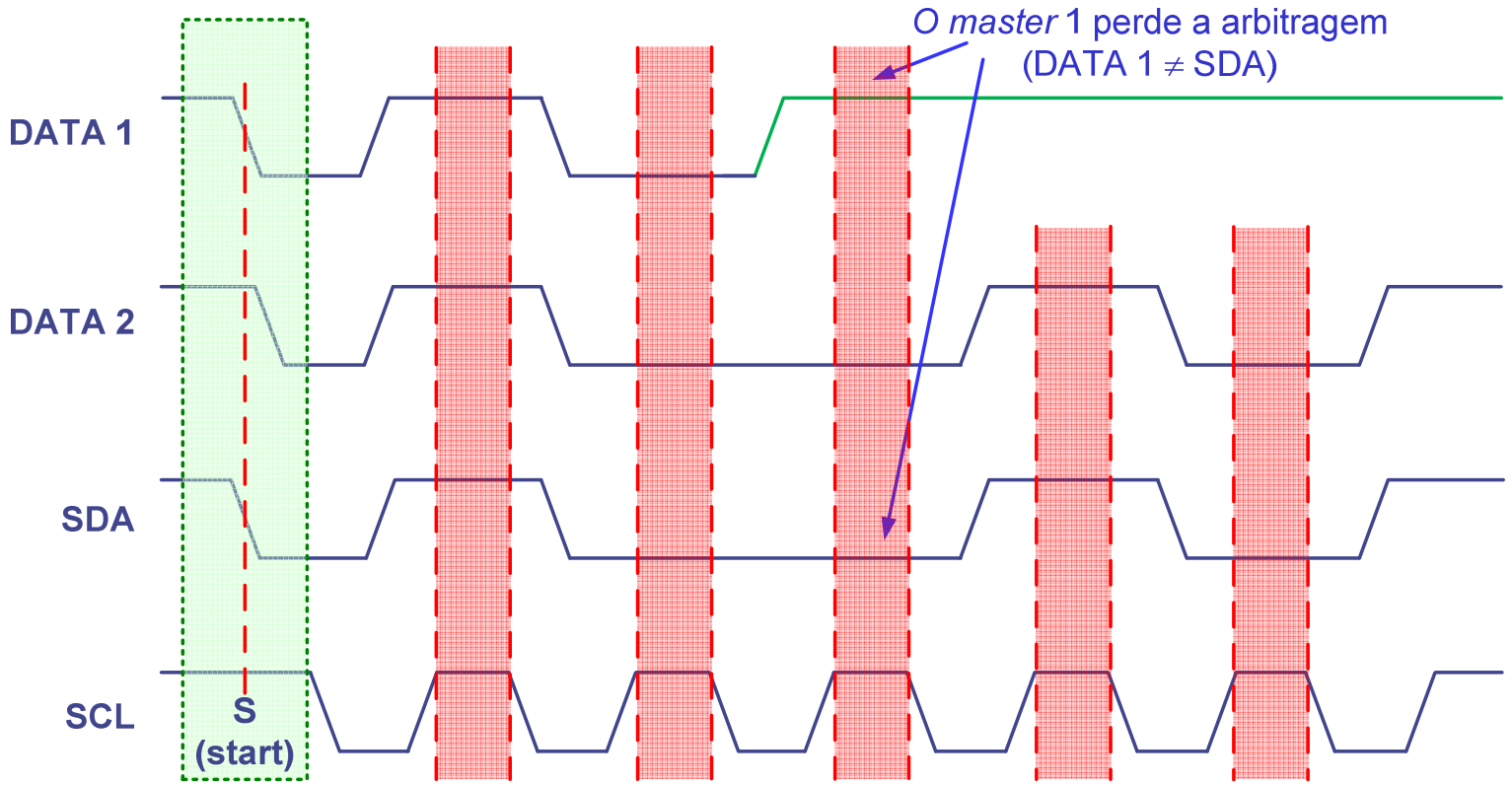
DATA_1	DATA_2	SDA	
0	0	0 (D)	Arbitragem continua
0	**1**	**0 (D)**	Master 2 perde arbitragem
1	**0**	**0 (D)**	Master 1 perde arbitragem
1	1	1 (R)	Arbitragem continua

Arbitragem (linha SDA) – Exemplo

- Exemplo:

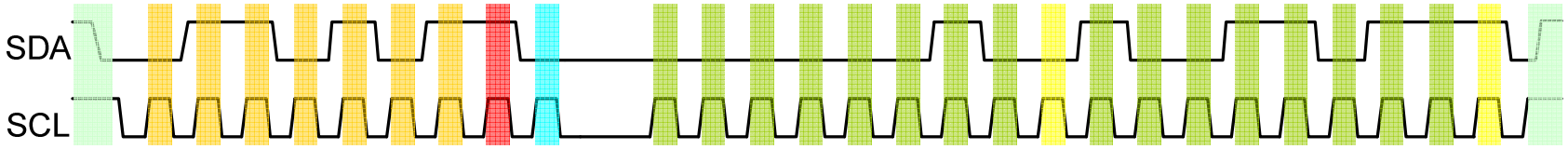
  1) o *master* 1 e o *master* 2 iniciam uma transmissão (quando o barramento passa a "idle")

  2) o *master* 1 perde a arbitragem na transmissão do 3º bit



Exercício

S



- Considere o diagrama temporal acima representado. Admita que representa a comunicação entre um *master* (μC) e um *slave* (ADC de 10 bits).

1. Qual o endereço do elemento *slave* (ADC)?
2. Estamos perante uma operação de escrita ou de leitura?
3. Quantos ACKs são gerados pelo *slave?*
4. Quantos ACKs são gerados pelo *master*?
5. Quantos NACKs são gerados? Por quem?
6. Qual o valor (expresso em hexadecimal) que foi fornecido pela ADC ao μC, sabendo que este começa sempre pelo MSBit?
7. Quantas situações de *clock stretch* são gerados nesta transação? Por quem?
8. Supondo que a frequência do relógio é de 1MHz e que o *stretch* corresponde a dois ciclos de relógio, qual a duração total da transação?

Anexo

- Endereços reservados.

Two groups of eight addresses (0000 XXX and 1111 XXX) are reserved for the purposes shown in Table 3.

Table 3. Reserved addresses

X = don't care; 1 = HIGH; 0 = LOW.

Slave address	R/$\overline{W}$ bit	Description
0000 000	0	general call address[1]
0000 000	1	START byte[2]
0000 001	X	CBUS address[3]
0000 010	X	reserved for different bus format[4]
0000 011	X	reserved for future purposes
0000 1XX	X	Hs-mode master code
1111 1XX	1	device ID
1111 0XX	X	10-bit slave addressing

[1] The general call address is used for several functions including software reset.

[2] No device is allowed to acknowledge at the reception of the START byte.

[3] The CBUS address has been reserved to enable the inter-mixing of CBUS compatible and I²C-bus compatible devices in the same system. I²C-bus compatible devices are not allowed to respond on reception of this address.

[4] The address reserved for a different bus format is included to enable I²C and other protocols to be mixed. Only I²C-bus compatible devices that can work with such formats and protocols are allowed to respond to this address.

De: I²C-bus specification and user manual, Rev. 6 — 4 April 2014