

Aula 15

- A interface I2C
- Características básicas
- Sinalização
- Endereçamento
- Transferência de dados
- Múltiplos Masters
 - Sincronização dos relógios
 - Arbitragem

José Luís Azevedo, Arnaldo Oliveira, Tomás Oliveira e Silva, Nuno Lau

Introdução

- I2C: Inter-Integrated Circuit
- Desenvolvido pela Philips Semiconductors (agora NXP Semiconductors)
 - Versão 1 em 1992
 - Atualmente na revisão 6 (Abril de 2014)
- 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

Introdução

- Dada a sua simplicidade, versatilidade e economia de recursos, o I2C encontra-se em diversos tipos de aplicação, e.g.:
 - Sensores, DACs, ADCs
 - Memória externa em microcontroladores
 - Controlo de subsistemas em eletrónica de consumo
 - e.g. ajuste dos parâmetros de imagem (contraste, brilho e saturação) e som em TVs, monitores, ...
 - Controlo de subsistemas em terminais de telemóvel
 - Monitorização de hardware
 - e.g. temperatura de CPUs e velocidade da ventoinha em motherboards
 - Interface com Real-Time Clocks
 - ...

I2C – terminologia

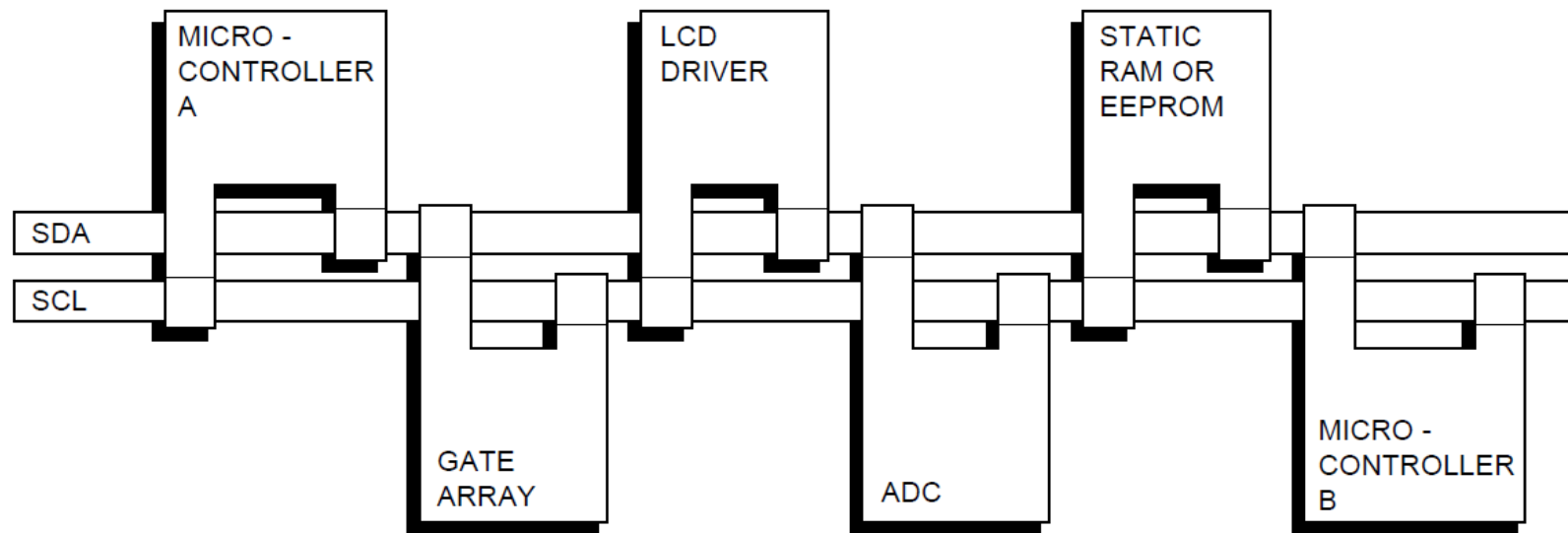
- **Transmitter** – dispositivo que envia dados para o barramento
- **Receiver** – dispositivo que recebe dados do barramento
- **Master** – o dispositivo que inicia a transferência, gera o sinal de relógio e termina a transferência
- **Slave** – o dispositivo endereçado pelo master
- **Multi-master** – mais do que um *master* pode tentar, ao mesmo tempo, controlar o barramento sem corromper a comunicação em curso
- **Arbitragem** – procedimento para assegurar que, se mais do que um *master* tentar, simultaneamente, controlar o barramento, apenas a um é permitido continuar, sem perturbação da comunicação iniciada pelo *master* vencedor
- **Sincronização** – procedimento para sincronização dos sinais de relógio de dois ou mais dispositivos

I2C – Caraterísticas básicas

- Transferência série bidirecional, *half-duplex*, orientada ao byte
- O barramento de comunicação apenas necessita de dois fios:
 - Serial data line (SDA)
 - Serial clock line (SCL)
- Barramento *multi-master* com detecção de colisões e arbitragem, evitando a corrupção de informação se dois ou mais *masters* iniciarem simultaneamente uma transferência
- As transferências envolvem sempre uma relação *master/slave*
- *Master* pode ser transmissor ou recetor (master-transmitter ou master-receiver)
- Cada dispositivo ligado ao barramento é endereçável por software usando um endereço único previamente atribuído
- Endereços de 7 bits ("standard mode") - alguns endereços reservados, 112 disponíveis
- Endereçamento de 10 bits também disponível

Exemplo de interligação num barramento I2C

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

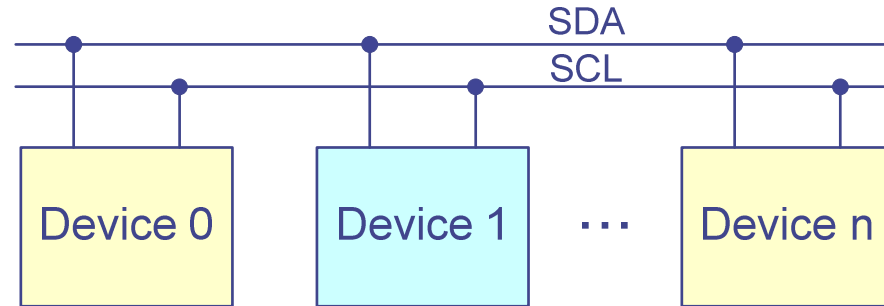


De: I2C-bus specification and user manual, Rev. 6 — 4 April 2014

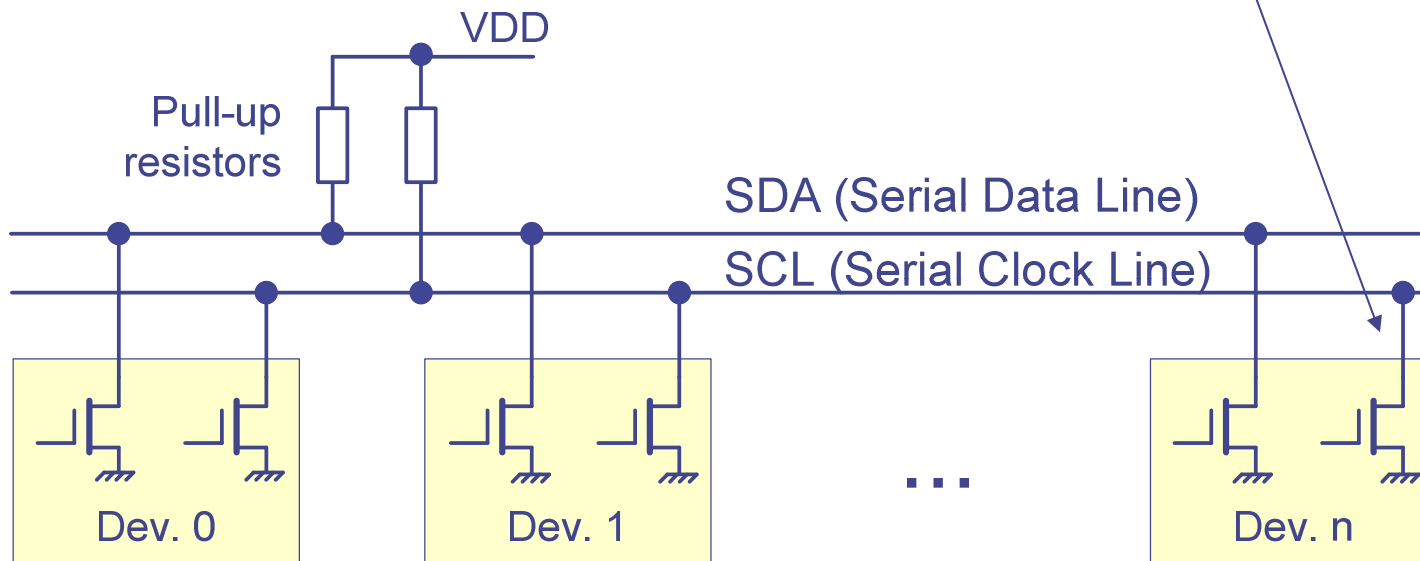
http://www.nxp.com/documents/user_manual/UM10204.pdf

I2C – Sinalização

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



- Ligação às linhas SDA e SCL: **"open collector" / "open drain"**

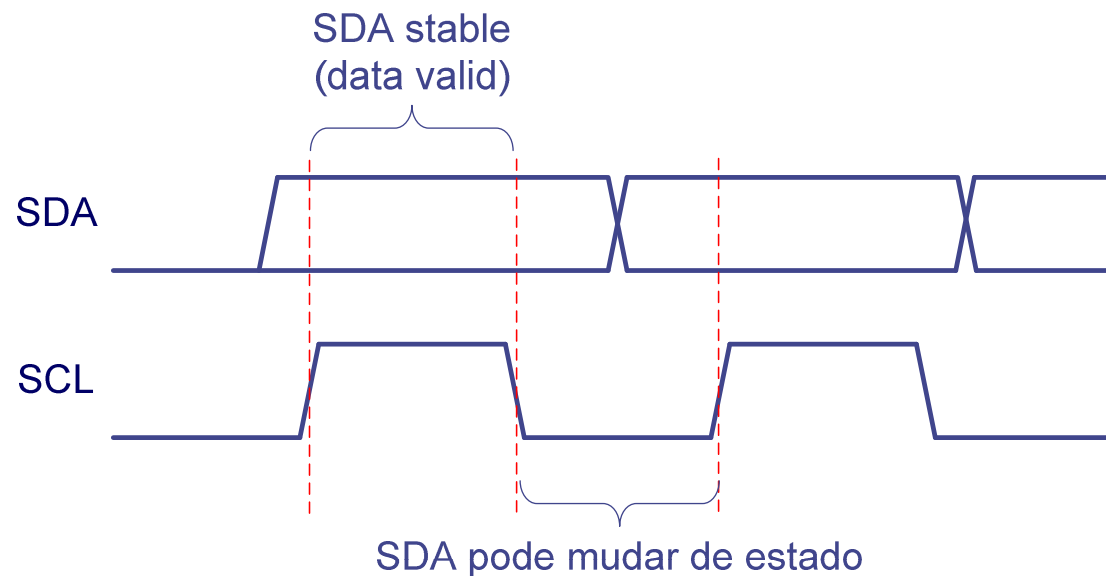


Masters e Slaves

- O *master*
 - Controla a linha SCL (Serial Clock)
 - Inicia e termina a transferência de dados
 - Controla o endereçamento dos outros dispositivos
- O *slave*
 - É o dispositivo endereçado pelo *master*
 - Condiciona o estado da linha SCL
- 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)

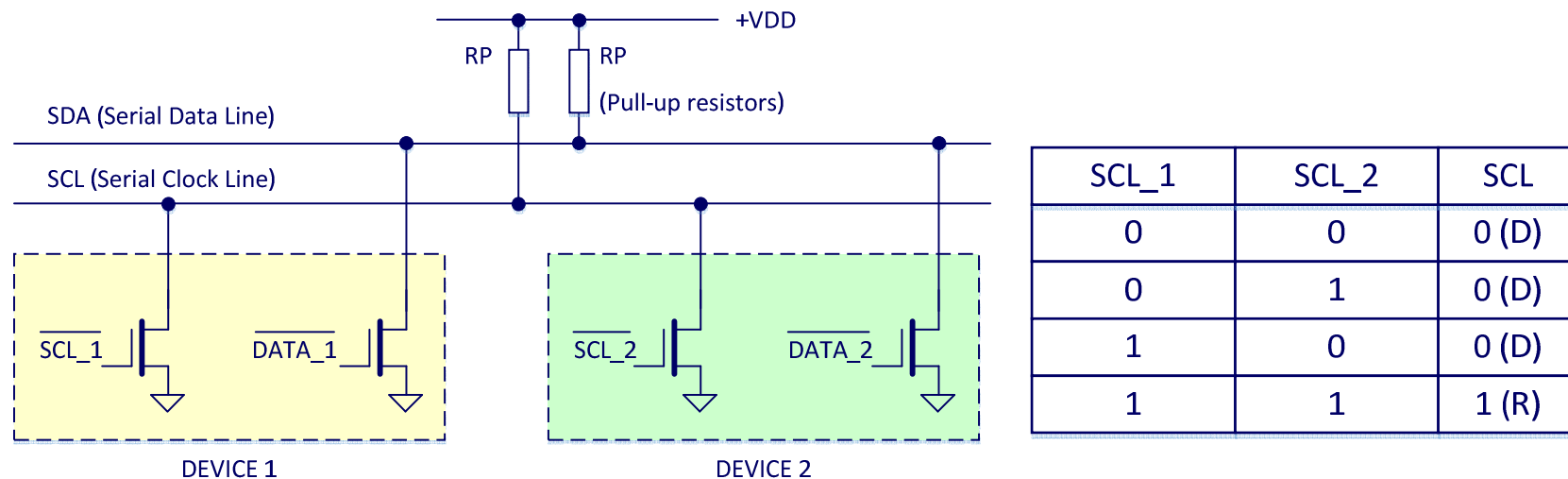
I2C – Sinalização

- Transferência de bits
 - Um período de relógio por bit de dados
 - SCL = '1': dados em SDA válidos
 - Dados (SDA) só são alterados quando SCL = '0'
- Transições em SDA quando SCL='1' sinalizam "condições". As "condições" são sempre geradas pelo *master*



I2C – Sinalização

- As linhas funcionam numa ligação "wired-AND", permitindo usar "bit recessivo" (1) e "bit dominante" (0) para várias sinalizações
- As linhas de clock e dados de cada dispositivo são "wire ANDed" com os respetivos sinais do barramento



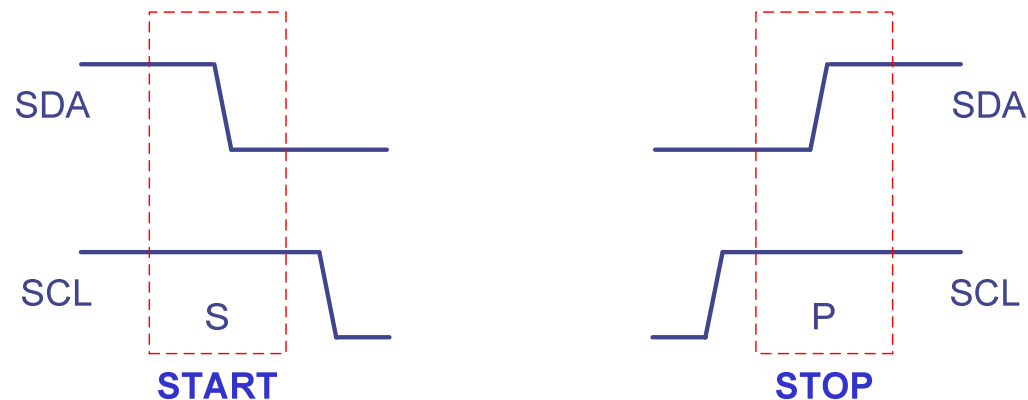
- Na ausência de bit dominante a linha respetiva está no nível lógico '1' (Recessivo), imposto através da resistência de *pull-up* RP

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"

Símbolos (condições)

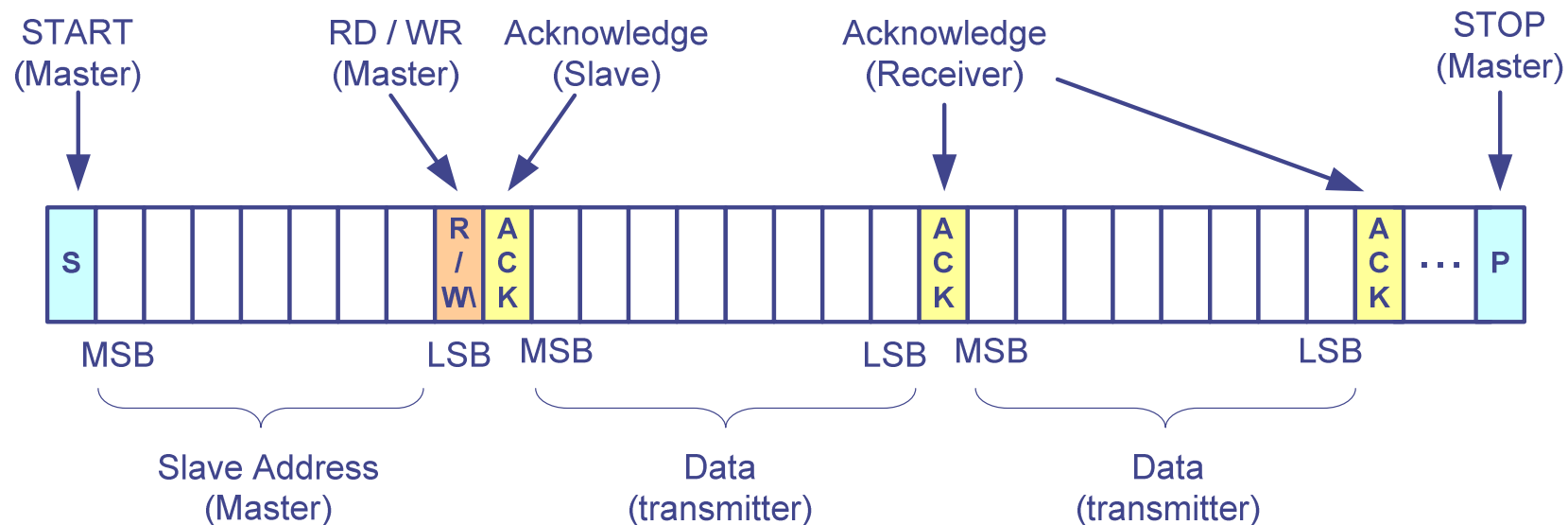
- As transações são delimitadas por dois símbolos / "condições": **START** e **STOP**, geradas pelo *master*
- As condições START/STOP são sinalizadas por meio de uma sequência que viola as regras normais de transferência de dados



- 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, até ao próximo START

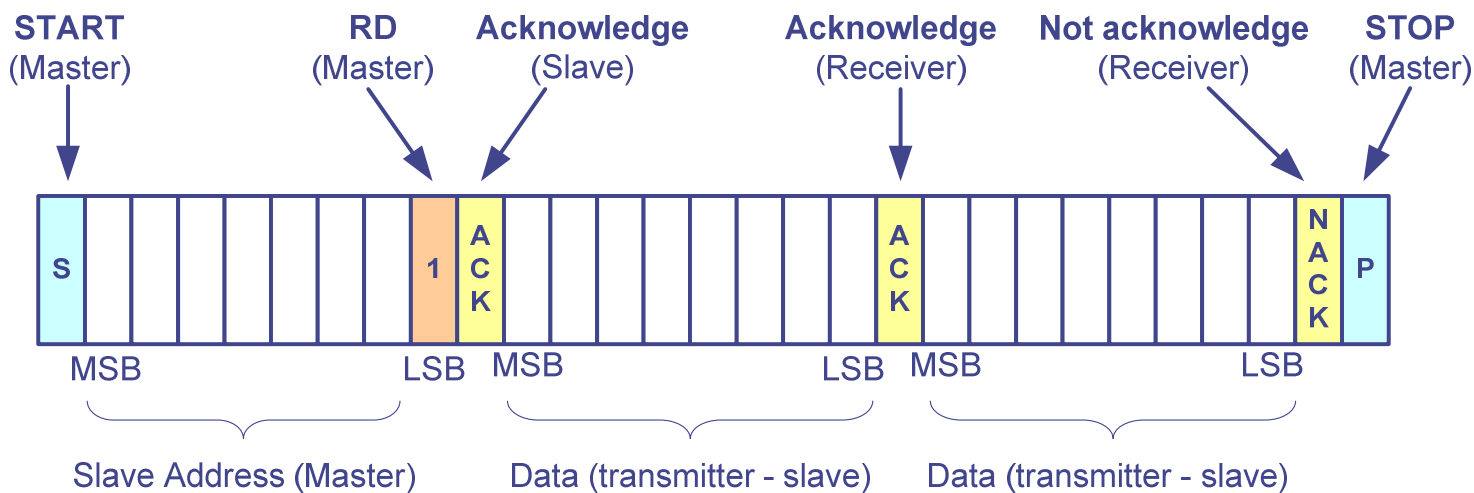
Transferência de dados

- O *master*.
 - Envia um **START** (S)
 - De seguida envia o **endereço** do slave (7 bits) e o bit de qualificação da operação (Read / Write\ - R/W\)
- O *slave* endereçado faz o **acknowledge** (ACK) na *slot* seguinte
- De seguida o transmissor (*master* ou *slave*) envia o byte de dados aguardando um **acknowledge do recetor** no final. Este ciclo de 9 bits repete-se para cada byte de dados que se pretenda transferir.



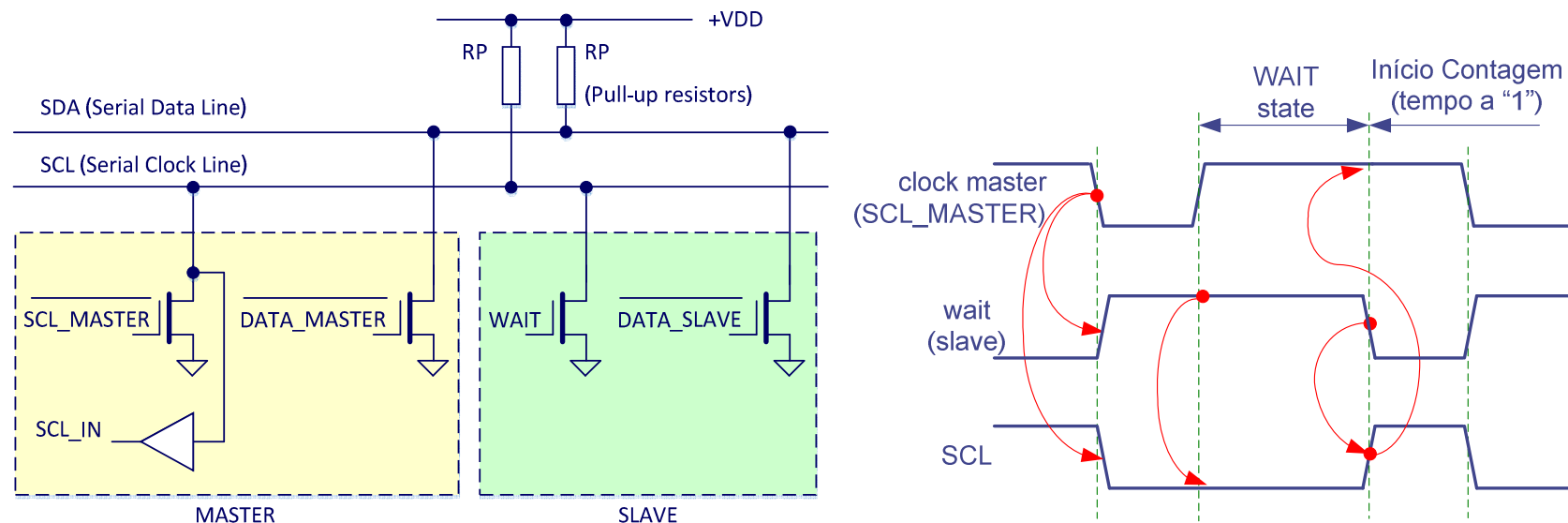
Transferência de dados

- A transferência é orientada ao byte (8 bits) sendo transmitido, em primeiro lugar, o bit mais significativo (MSbit)
- Após o 8º bit (o LSbit), o recetor gera um acknowledge, na linha SDA, sob a forma de um bit dominante (0)
- Na situação em que o *master* é recetor gera:
 - **acknowledge** sinalizando dessa forma o *slave* de que vai continuar a ler dados, ou
 - **not acknowledge** sinalizando o *slave* de que o byte recebido constitui o fim da transferência. Neste caso o *master* envia logo de seguida um STOP (P)
- Exemplo (**master recetor**):



Transferência de dados

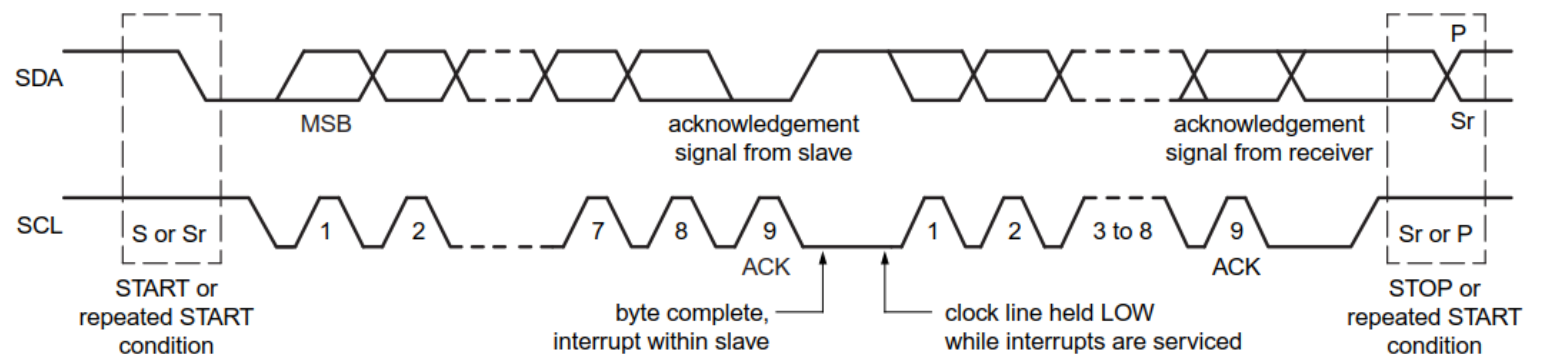
- O *slave* pode forçar o alargamento da transferência mantendo a zero (bit dominante) a linha SCL; esta técnica designa-se por "clock stretching"
- O sinal "wait" (do *slave*) condiciona o estado da linha SCL do barramento: enquanto "wait" estiver a '1', a linha SCL está forçada a nível lógico '0' (dominante)



- *master* fica em "wait state" enquanto $SCL_MASTER \neq SCL_IN$

Transferência de dados

- Exemplo de uma sequência de transferência:



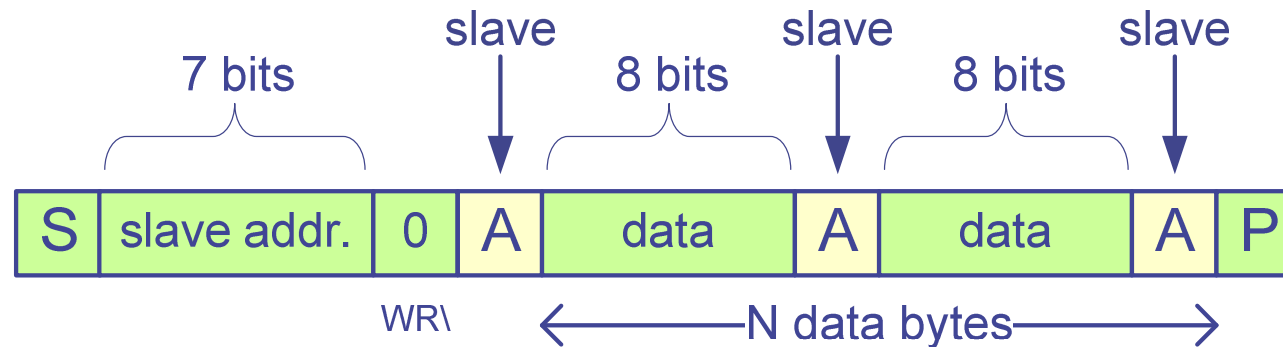
002aac861

Fig 6. Data transfer on the I²C-bus

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

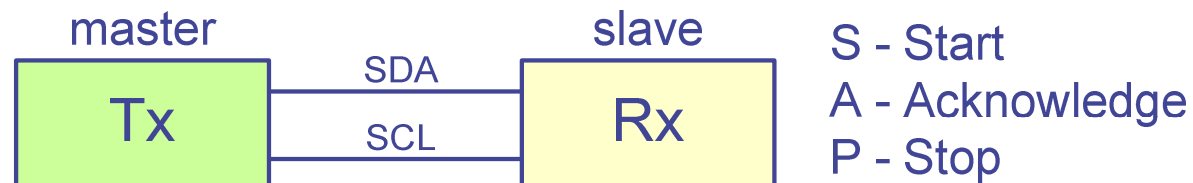
Transferência de dados – escrita

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



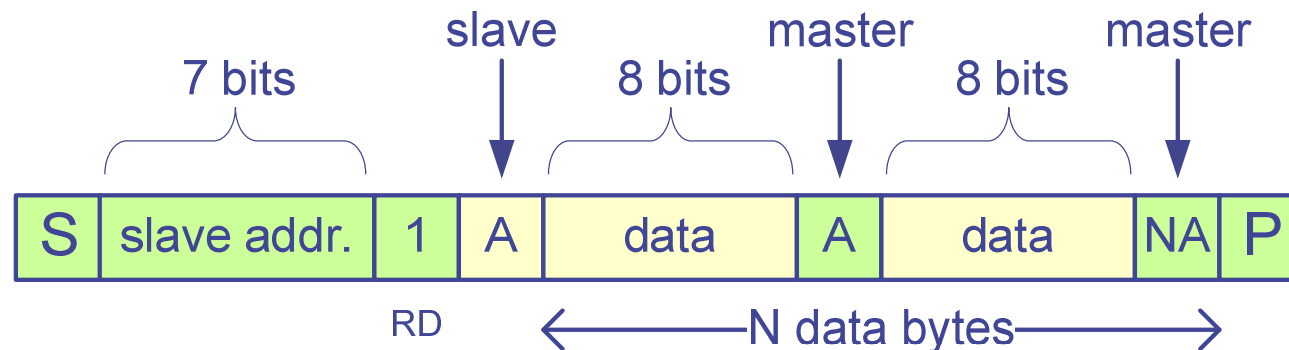
 Informação produzida pelo Master

 Informação produzida pelo Slave



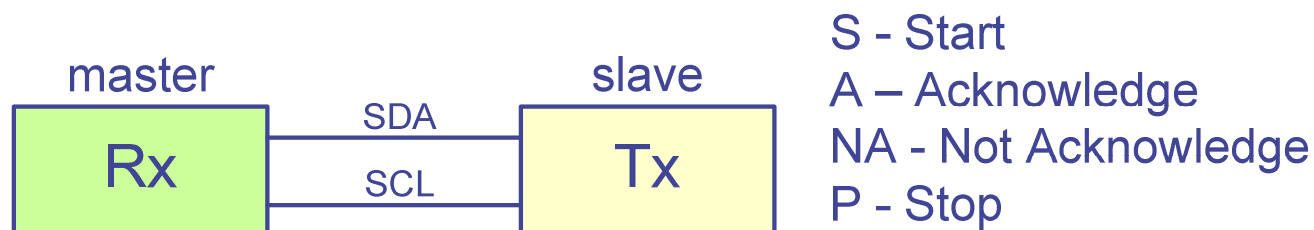
Transferência de dados – leitura

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



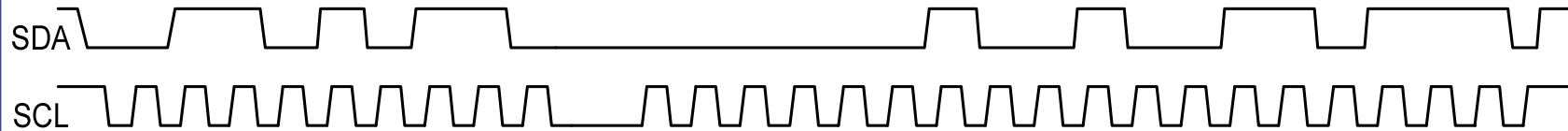
Informação produzida pelo Master

Informação produzida pelo Slave



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

Exercício



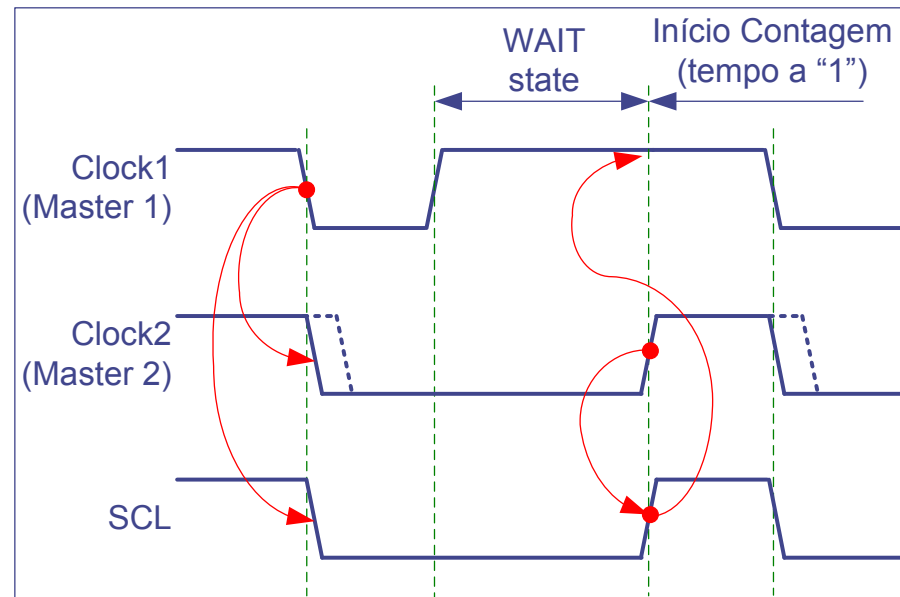
- Considere o diagrama temporal acima representado. Admita que representa a comunicação entre um *master* (μC) e um *slave* (ADC de 10 bits).
1. Qual o endereço do elemento *slave* (ADC)?
 2. Estamos perante uma operação de escrita ou de leitura?
 3. Quantos ACKs são gerados pelo *slave*?
 4. Quantos ACKs são gerados pelo *master*?
 5. Quantos NACKs são gerados? Por quem?
 6. Qual o valor (expresso em hexadecimal) 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?

Múltiplos *masters*

- Dois (ou mais) *masters* podem iniciar uma transmissão num barramento livre (isto é, após um STOP) ao mesmo tempo
- Tem de estar previsto um método para decidir qual dos *masters* toma o controlo do barramento e completa a transmissão – 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"
- 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')

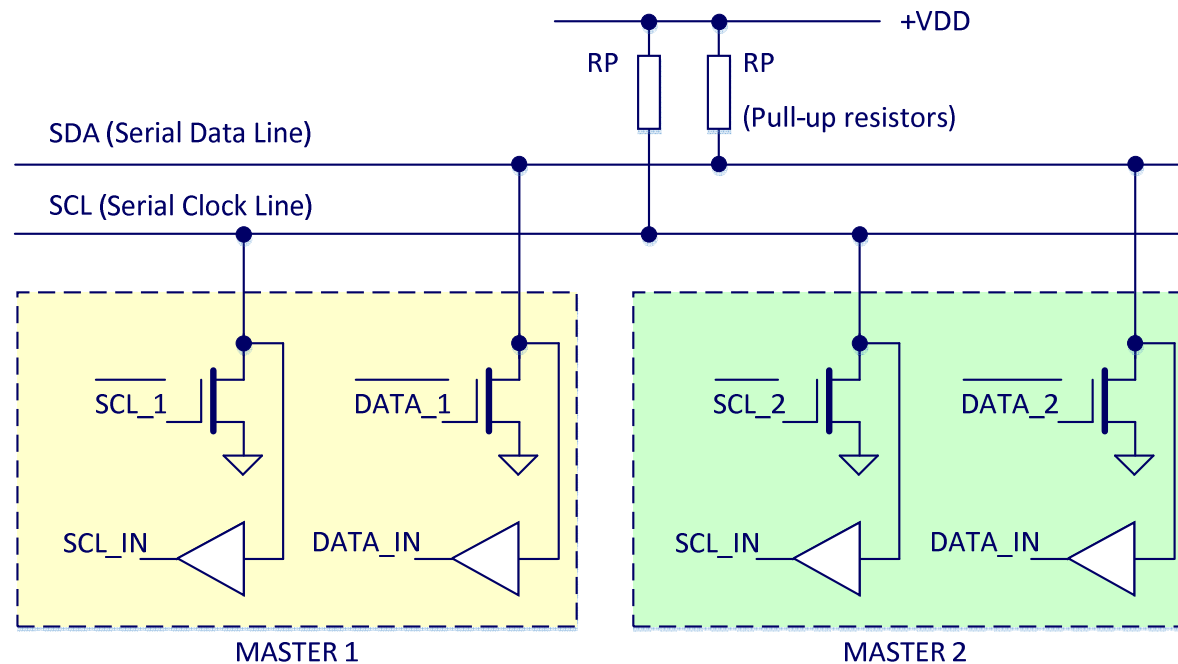


- Se SCL se mantém a '0' comuta para um estado de "wait", ficando a aguardar que a linha SCL passe a '1'
- Logo que SCL passe a '1' inicia a contagem do tempo a '1' do seu relógio
- O primeiro *master* a terminar o seu tempo a '1' força a linha SCL a '0'
- O sinal SCL fica sincronizado com um t_{low} determinado pelo *master* com maior t_{low} e um t_{high} imposto pelo master com menor 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
- **Arbitragem** é o procedimento que assegura que, se mais do que um master tentar, simultaneamente, controlar o barramento, apenas a um é permitido continuar, sem 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)
0	1	0 (D)
1	0	0 (D)
1	1	1 (R)

Arbitragem continua

Master 2 perde arbitragem

Master 1 perde arbitragem

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

