# Projeto 3 - Barramento $I^2C$

Tópicos Especiais em Processamento de Sinais

19 de Agosto de 2016

## 1 Contextualização

Uma das tarefas mais comuns em dispositivos embarcados é a comunicação com outros dispositivos e periféricos. Para que esta comunicação seja estabelecida, é necessário que ambos os dispositivos concordem sobre qual protocolo (idioma) irão utilizar. No caso específico de microcontroladores alguns dos protocolos são bastante comuns são RS-232, SPI (*Serial Peripheral Interface*) e I2C (*Inter-Integrated Circuit*). Esses protocolos podem ser usados na comunicação do microcontrolador com periféricos, como chips de conversão digital-analógico (D/A e A/D), LCD, EPROMs, sensores de temperatura. Neste projeto, estudaremos o padrão I2C. Ele se destaca pela sua simplicidade e pelo uso reduzido de fios conectando os diversos pontos da linha. Ele é usado pela Nintendo, por exemplo, para conectar o controle do Wii ao nunchuck.

## 2 Barramento $I^2C$

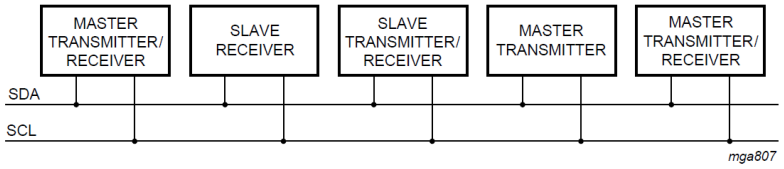


Figura 1: Barramento I2C

O barramento I2C utiliza apenas dois fios para realizar uma comunicação bidirecional entre diversos dispositivos. Na terminologia utilizada, o dispositivo que está gerando uma mensagem é o *transmissor*, enquanto o dispositivo que está recebendo a mensagem é o *receptor*. O dispositivo que controla a

mensagem é o *mestre*, e os dispositivos que são controlados por este são os *escravos*. Os dispositivos mestre e escravo assumem o papel de transmissor e receptor em diferentes etapas da comunicação. O barramento é composto por apenas dois fios: um fio de dados, SDA, e um fio de clock, SCL. Apenas o mestre controla o clock, mesmo quando este estiver recebendo uma mensagem (como veremos abaixo).

## 2.1 Sinais de Controle: START e STOP

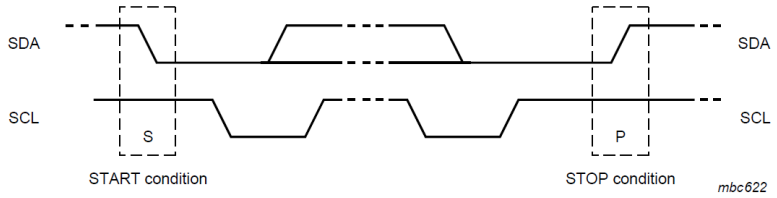


Figura 2: Protocolo I2C - START e STOP

Quando o barramento está ocioso (em *idle*) nenhum dos dispositivos escreve no barramento. Nesse caso SDA e SCL ficam em nível lógico HIGH. Isso é garantido através dos resistor de pull-up.

Para iniciar uma transmissão, o mestre força a descida de SDA (mudança de HIGH para LOW) enquanto o clock está em nível alto. Este sinal é a condição de partida (START), e prepara os dispositivos para a comunicação.

Para finalizar a transmissão, o mestre libera SDA (mudança de LOW para HIGH) quando o clock está em nível HIGH. Este sinal é a condição de parada (STOP) liberando assim o barramento para outras comunicações. Se for necessário re-estabelecer a comunicação, deve-se gerar novamente a condição de partida descrita no parágrafo anterior.

## 2.2 Transferência de um bit

Na transferência de um bit, o dado em SDA deve permanecer estável entre a subida e a descida do clock (SCL). Mudanças em SDA quando SCL está em nível alto são entendidas como sinais de controle (como visto nas condições de START e STOP). Durante a transmissão, o valor de SDA é constante enquanto SCL estiver em nível HIGH e assume o valor do bit a ser transmitido.

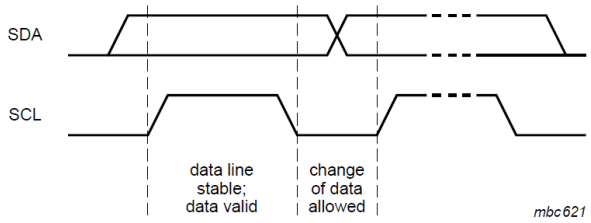


Figura 3: Protocolo I2C - transferencia de 1 bit

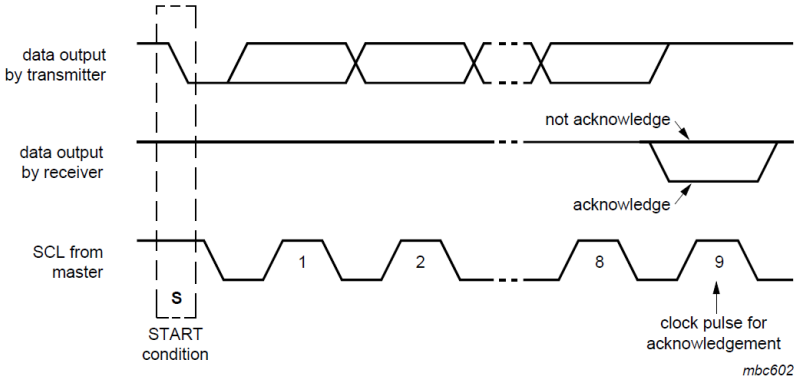


Figura 4: Protocolo I2C - transferencia de 1 byte

## 2.3 Transferência de um Byte

A comunicação no barramento é feita em bytes (8 bits). A cada byte transmitido, um sinal de acknowledge deve ser gerado. Portanto, para cada byte transmitido, 9 sinais (subida e descida) de clock precisam ser gerados. O número de bytes enviados entre um sinal de START e STOP é ilimitado. Na figura acima, é necessário entender que o canal "data output" é o canal SDA, que é comum ao transmissor e ao receptor. Na figura, eles são apresentados em separado para mostrar quem gera os sinais a cada momento. No envio de um byte, o transmissor manda os 8 bits em SDA (em 8 sinais de clock separados), e o receptor gera o sinal de acknowledge, também em SDA, no nono sinal de clock. O receptor gera o sinal 0 para sinalizar o acknowledge ("entendido") e 1 para sinalizar que não entendeu a mensagem. Note que o clock é gerado pelo mestre, enquanto os papéis de transmissor/receptor podem ser feitos tanto pelo mestre quanto pelo escravo.

# 3 Projeto

O projeto consiste em criar um programa em C que decodifique palavras de 8-bits recebidas de maneira serial por um barramento I2C. Simularemos a recepção dos bits usando um arquivo texto onde cada linha representa a amostragem das linhas SDA e SCL.

Considere o sinal da figura 3. As linhas de transmissão SDA e SCL estão amostradas em momentos bem definidos do tempo (marcados pelas linhas verticais). O comando de START é detectado então pela transição entre a segunda e a terceira amostra neste exemplo.

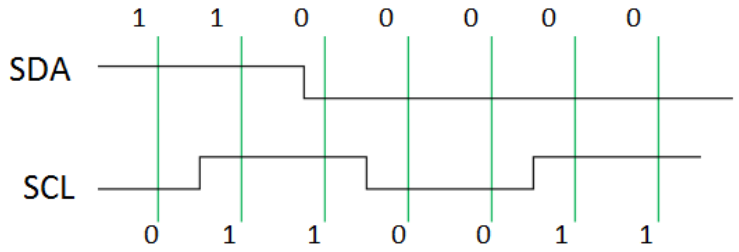


Figura 5: Linhas SDA e SCL amostradas

```
1 # SDA SCL
2 10
3 11
4 01  --> Detecta START
5 00
6 00
7 01  --> Recebe 0.
8 01
9 ...
```

Figura 6: Arquivo : entrada.txt (início)

Este programa deverá ser codificado em C usando um estilo de codificação de máquina de estados. O decodificador deverá apenas receber os valores dos 8 bits transmitidos e não deve se preocupar em gerar/obter o acknowledge (vide figura 7).

Note que o software deve procurar por mudanças nos sinais, e não por valores dos sinais em instantes de tempo específicos (uma vez que o barramento I2C não especifica temporização precisa).

```
20 # SDA SCL
21 ..
22 00
23 10
24 11  --> Ultimo bit transmitido
25 11      (bit 8 = 1)
26 10
27 10
28 11  --> Acknowledge
29 11      (esta batida de clock
30 10      nao interessa)
31 00
32 01
33 11  --> Detecta o STOP
34 11  --> Fim de transmissao
35 ...
```

Figura 7: Arquivo : entrada.txt (fim)

Uma vez terminada a decodificação, o programa deve imprimir no terminal o byte recebido em representação hexadecimal, por exemplo: 0x34 ou 0xFA.