Resumos de AC2

Teste Teorico 2

Tiago Almeida

June 21, 2024

Contents

1 Introdução

Escrever um pequeno overview da matéria que sai para o teste teórico 2 e o que esperar encontrar neste documento

2 microcontroladores e Sistemas embebidos

3 Noção de Perifericos

3.1 Estrutura

3.2 Descodificação de endereços

3.3 Gerador de sinais de seleção

4 Portos I/O

4.1 Estrutura

4.2 Portos no PIC32

5 Técnicas de transferência de informação entre periféricos e memória

5.1 Programmed I/O (Programada)

5.2 Interrupt Driven I/O (Interrupção)

5.3 DMA (Acesso Direto)

6 Interrupções

6.1 Interrupções no PIC32

7 Timers

7.1 Funcionamento

7.2 Watchdog Timer

7.3 Timers no PIC32

8 Barramentos paralelos e Barramentos série

9 A Interface SPI

SPI é uma interface de **alta velocidade e de curta distância** (dezenas de cm) usada para comunicar com diversos dispositivos diferentes, como por exemplo:

- Sensores de diversos tipos: temperatura, pressão, etc.
- Cartões de memória (MMC / SD)
- Circuitos: memórias, ADCs, DACs, Displays LCD (e.g. telemóveis), comunicação entre corpo de máquinas fotográficas e as lentes, ...
- Comunicação entre microcontroladores

A transferencia de dados em **SPI** é ciclica, isto é, tudo o que é enviado é recebido de volta por quem enviou. Assim, existem 3 tipos de transferência possiveis:

- **Bidirecional**: são transferidos dados válidos em ambos os sentidos (master → slave e slave → master)
- **Master → slave (operação de escrita)**: master transfere dados para o slave, e ignora/descarta os dados recebidos
- **Slave → master (operação de leitura)**: master pretende ler dados do slave; para isso transfere para o slave uma palavra com informação irrelevante (por exemplo 0); o slave ignora/descarta os dados recebidos

9.1 Funcionamento

Pontos chave sobre a arquitetura do **SPI**:

- Apenas tem **1 master** mas pode ter **1 ou mais slaves**
- Relógio gerado e controlado pelo master
- Comunicação síncrona
- Comunicação full-duplex
- Apenas 1 slave pode ser selecionado por vez através do sinal SS
- O master inicia e controla a transferência de dados, com a sinalização:
  - **SCK**: clock
  - **MOSI**: Master Output Slave Input (SDO no master)
  - **MISO**: Master Input Slave Output (SDI no master)
  - **SS**: Slave select
- Quando o master envia um byte, o slave tambem envia um byte de volta ao mesmo tempo. Isso ocorre porque o SPI usa shift registers circulares para transferir dados.
- Dados relevantes e Inúteis:

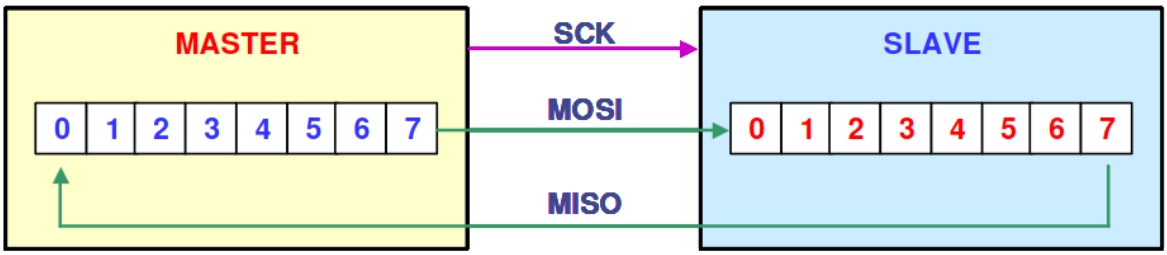

Figure 1: Exemplo do funcionamento do SPI

  - Os dados que o slave envia de volta podem ou não ter significado. Se o slave não tiver dados relevantes para enviar, ele pode enviar bytes preenchidos com zeros, valores padrão ou dados que não são importantes para o contexto da comunicação.

- Protocolo e Significado dos Dados:
  - O significado dos dados trocados entre master e slave depende do protocolo de comunicação específico implementado no software. O master e o slave devem ter um acordo prévio sobre o que os dados representam e como devem ser interpretados.

9.2 Arquiteturas de ligação

No caso do SPI, existem 2 arquiteturas de ligação:

- **Slaves independentes**
- **Daisy chain**

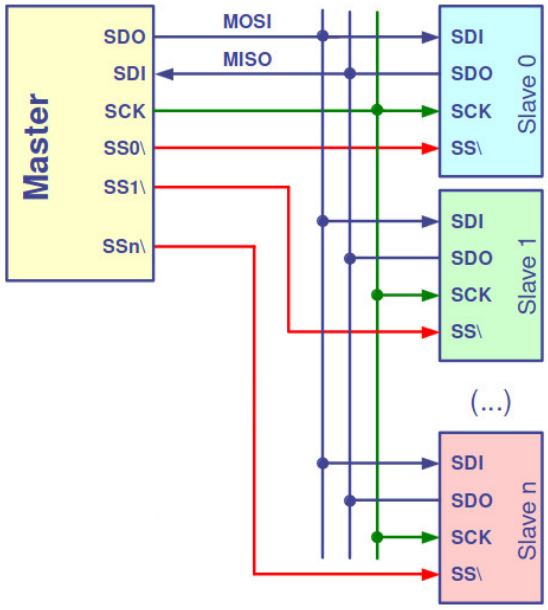

Figure 2: Exemplo da arquitetura 'Slaves independentes' do SPI

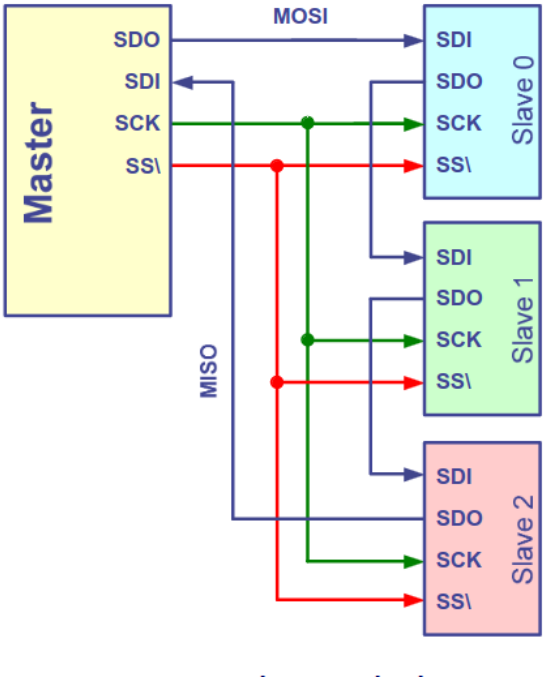

Figure 3: Exemplo da arquitetura 'Daisy chain' do SPI

No primeiro tipo, Slaves independentes, existe no master um sinal de seleção, sinal SS individual para cada salve.

- Apenas um sinal SSx pode estar ligado de cada vez
- O número maximo de slaves está limitado pelo numero de SS
- O microcontrolador pode gerar, através dos seus portos digitais, sinais SSx de forma a ultrapassar a limitação anterior

No segundo tipo, Daisy chain, existe no master um unico sinal hyperref[ss]SS comum para todos os slaves.

- Como podemos ver pelo fio azul do exemplo de daisy chain, a informação tem de passar por todos os slaves até voltar ao master, logo os slaves tem de ter a capacidade de armazenar uma sequência de N bits.
- Enquando o SS estiver ativo, o slave ignora o comando recebido e envia para o slave seguinte ou para o master no caso do ultimo slave.
- O slave apenas executa o comando quando o sinal SS for desativado

9.3 Detalhes adicionais

Algumas notas adicionais sobre **SPI** que podem ou não ser importantes:

- Criado pela empresa Motorola
- Não é exigido precisão no relógio.
  - Permite com que o se possa optar por um oscilador de baixo custo
  - Não é necessário um cristal de quartzo
- Facil de implementar por hardware ou por software
- O SPI funciona sempre em modo 'data exchange', isto é, o processo de comunicação envolve sempre a troca do conteúdo dos shift-registers do master e do slave. Cabe aos dispositivos envolvidos na comunicação usar ou descarta a informação recebida

10 O barramento CAN

CAN é um barramento relativamente rápido, de media distância, **adequado para aplicação de segurança crítica devido á sua elevada robustez.** Alguns dos exemplos de utilização de **CAN** mais comuns são:

- Comunicação entre subsistemas de um automóvel
- Aviónica, Aplicações industriais, Domótica, Robótica
- Equipamentos médicos, ...

Alguns fatores que lhe dão essa robustez são:

- Tolerância a interferência eletromagnética
- Capacidade de detetar diferentes tipos de erros
- Baixa probabilidade de não deteção de um erro de transmissão (4.7×10^{-11})

10.1 Funcionamento

Pontos chave sobre a arquitetura do **CAN**:

- É multi-master e todos os nós ligados ao barramento **CAN** são masters e podem produzir informação e iniciar uma transmissão.
- Transmissão sempre em 'broadcast', ou seja, quando algum nó enviar informação todos os outros recebem
- Comunicação Half-Duplex
- Transmissão orientada ao bit
- Comunicação assíncrona e não há transmissão de relogio como em ISP.
  O transmissor e o recetor têm relógios locais independentes
- De forma a organizar a informação a ser transmitida dentro do barramento, cada mensagem enviada tem um ID associado que determina a sua prioridade
- Na transmissão de bits, a cada 5 bits iguais é adicionado 1 bit extra de polaridade oposta. **NÃO ENTENDI BEM O MOTIVO**
- Na transmissão, o CAN Transceiver do nó transforma o nivel lógico do bit em duas tensões e coloca nas linhas CAN_H e CAN_L
- Na receção, o CAN Transceiver do nó a receber transforma a diferença de tensões nas linhas em um nivel lógico novamente e envia para o CAN Controller.

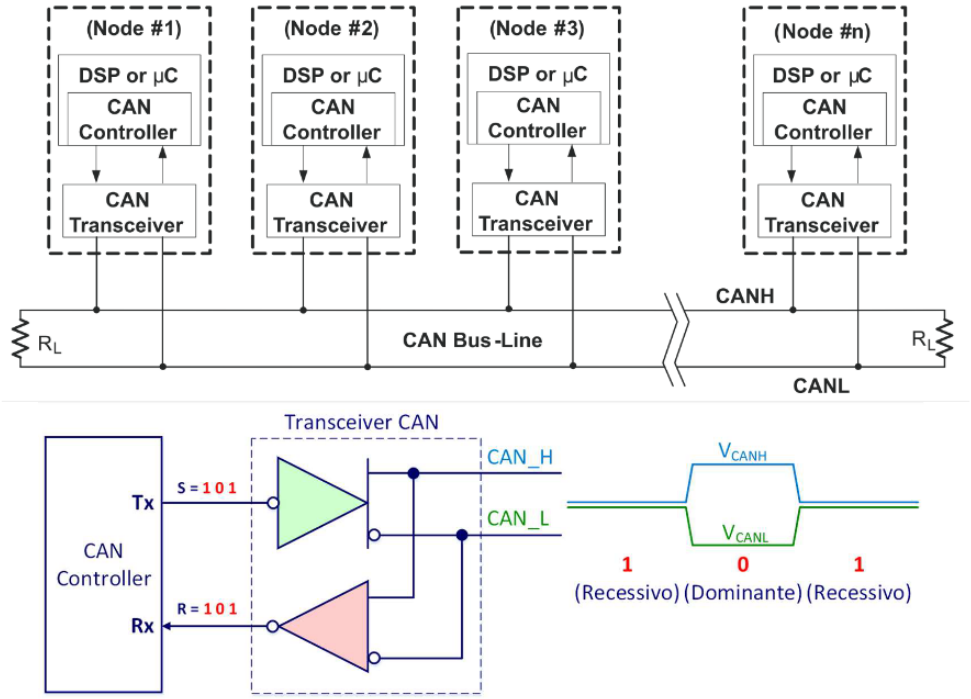

Figure 4: Exemplo do funcionamento do CAN

10.2 Formato das tramas do CAN

O CAN envia os dados em formato de tramas, que são conjuntos de bits que carregam informação de controlo e dados.
Revisão de cada parte da trama:

1. **SOF** (Start of Frame)
   - Consiste de apenas 1 bit
   - Bit dominante ('0') indica o início da trama
   - Usado para sincronizar os nós recetores
2. **Arbitration Field**
   - Consiste de 12 bits no total
   - Os primeiros 11 bits indicam o ID da mensagem. IDs mais baixo tem maior prioridade.
   - O ultimo bit é o bit RTR (Remote Transmission Request). Contém o bit dominante ('0') quando a trama apenas serve para enviar dados, e o bit recessivo ('1') quando a trama pretende pedir informação de outros nós. O segundo caso é menos comum.
3. **Control Field**

- Consiste de 6 bits no total
- O primeiro bit, chamado de IDE (identifier extension) serve para identificar o versão da trama CAN que se está a usar
- O segundo bit é reservado e não tem significado
- Os restantes 4 bits, chamam-se DLC (DLC3-DLC0) e indicam o numero de bytes que serão transmitidos na trama. Podem ir de 0 a 8 bytes.

4. **Data Field**
   - Pode ter desde 0 até 8 bytes de tamanho.
   - Contem a informação a ser transmitida
5. **CRC Field** (Cyclic Redundancy Check)
   - Consiste de 16 bits no total
   - Serve para deteção de erros
   - Funciona como o checksum: Transmissor e recetor ambos calculam o valor CRC com base nos dados enviados. Se o valor calculado pelo recetor não for igual ao que se encontra no CRC Field (valor calculado pelo transmissor), isso indica que houve um erro.
6. **ACK Field**
   - Consiste de 2 bits no total
   - Primeiro bit é o ACK Slot: Durante a transmissão da mensagem, o transmissor envia este bit como recessivo (1). Se pelo menos um nó na rede receber a mensagem corretamente, ele sobrepõe o bit recessivo com um bit dominante (0) no ACK Slot.
   - O segundo bit é o ACK Delimiter: É um bit recessivo ('1') e serve apenas para separa o ACK do proximo campo da trama
7. **EOF** (End of Frame)
   - Consiste em 7 bits no total
   - Todos os bits são recessivos ('1') e indicam o fim da trama
8. **IFS** (Interframe/Intermission)
   - Consiste em, no minimo, 3 bits no total, mas podem ser mais
   - Todos os bits são recessivos ('1') e servem como um delay no bus para dar tempo aos nós de processar a mensagem da trama antes da proxima ser transmitida

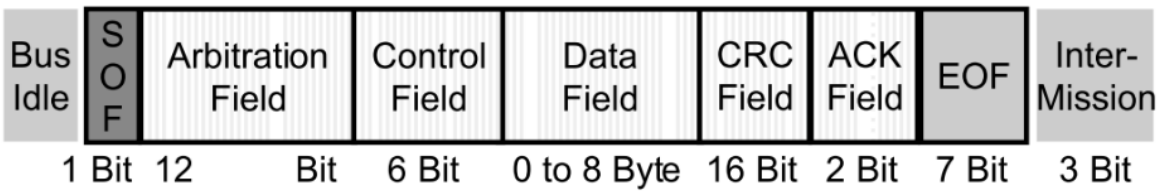

Figure 5: Exemplo da trama do CAN

10.3 Motivos da segurança adicional do CAN

...

11 A interface RS-232C

A interface RS-232C é uma interface muito antiga e, por consequencia, simples, o que leva a que seja muito popular em microcontroladores.

11.1 Funcionamento

Pontos chave sobre a arquitetura do **RS-232C**:

- Comunicação assíncrona
- Comunicação Full-duplex
- Transmissão orientada ao byte
- Consiste apenas de 2 linhas de sinalização, uma de de transmissão e uma de recessão, e uma linha de Ground (voltagem = 0V)
- O valor enviado é codificado como a diferença entre a tensão enviada no barramento Tx e o GND.Se o valor for **negativo**, o valor digital enviado é 1, e se for **positivo**, o valor enviado é 0.
- A descodificação é feito nos drivers de linha
- A versão default desta interface apenas suporta comunicação ponto-a-ponto

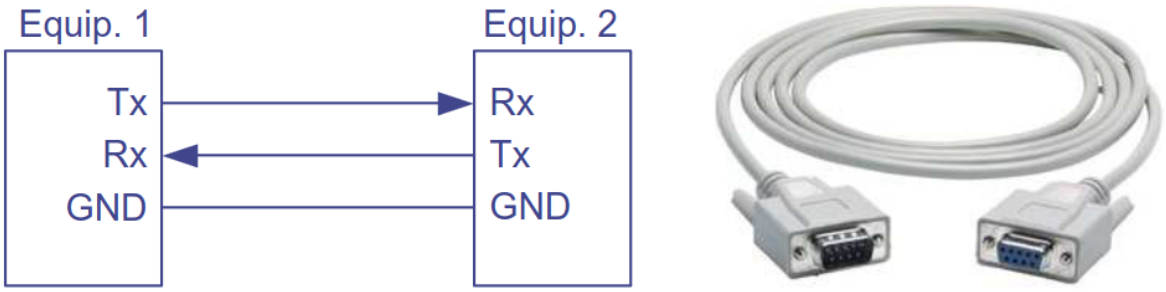

Figure 6: Exemplo do funcionamento do RS-232C

11.2 Formato das tramas do RS-232C

O RS-232C, assim como o CAN, envia informação no formato de tramas. No entanto, tambem por ser um protocolo mais antigo é uma trama muito mais simples em comparação. Revisão de cada parte da trama:

1. **Start bit**
   - Apenas um bit com valor lógico 0 que indica o inicio da trama
   - Tem o uso adicional de servir como ponto de sincronização para o receptor
2. **Data bits**
   - Consiste de 5 a 9 bits
   - Envia os dados do menos significativo para o mais significativo
     Exemplo: Data = 0x0F (0000 1111); Trama: $D_0 = 1$, $D_1 = 1$, $D_2 = 1$, ..., $D_7 = 0$
3. **Parity bit** (opcional)
   - Por explicar ...
4. **Stop bit**
   - Consiste de 1 ou 2 bits
   - Indica o fim da trama e tem a funcionalidade adicional de servir como um 'delay' entre tramas para dar tempo para processar dados

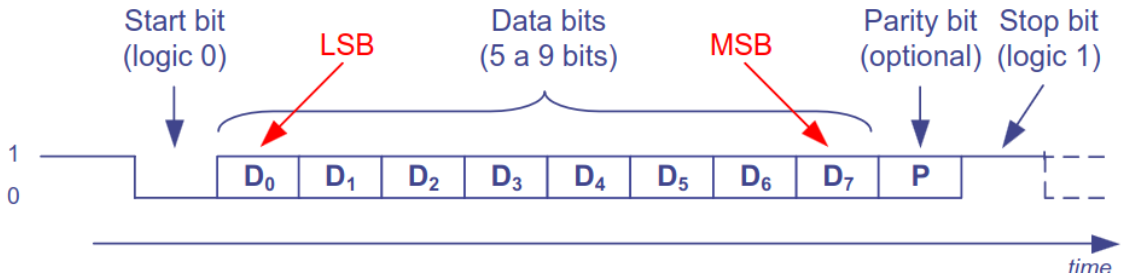

Figure 7: Exemplo da Trama do RS-232C

11.3 Sincronização de relógio

Apesar da comunicação ser assincrona, porque não há transmissão de relógio, cada dispositivo envolvido (transmissor e receptor) usam o seu proprio relógio, e por isso é necessário que estejam sincronizados. Para relembrar o funcionamento da sincronização reler páginas 9 a 15 da aula teorica 16.

11.4 Máximo desvio de frequência

...

12 Device Drivers

13 Tecnologias de memória

13.1 SRAM

13.2 DRAM

14 Memória Cache

15 Memória Virtual

16 Conclusão

Algumas conclusões e considerações que se deve ter após ter acabado o estudo

17 Glossário

Aqui está a secção de glossário. Cada termo usado repetidamente no documento está listado aqui com sua definição.

- **SPI**: Serial Peripheral Interface
- **CAN**: Controller Area Network
- **Simplex**: comunicação apenas num sentido (TX -¿ RX); usada, por exemplo, em telemetria, para leitura remota de sensores
- **Half-Duplex**: comunicação nos dois sentidos, mas apenas um de cada vez (é usada uma só linha)
- **Full-Duplex**: Comunicação simultânea nos dois sentidos (são usadas duas linhas)
- **SS**: Slave select. Sinal usado pelo master para selecionar o slave. Por vezes tambem é usado CS (chip select) como selecionador de slave.
- **SCK**: Clock. Relógio gerado pelo master que sincroniza a transmissão/receção de dados
- **MOSI**: Master Output Slave Input (SDO no master). Linha do master para envio de dados para o slave
- **MISO**: Master Input Slave Output (SDI no master). Linha do slave para enviar dados para o master