Resumos de AC2
RECURSO

Tiago Almeida

July 7, 2024

Contents

1 Introdução

Para o recurso sai toda a matéria dada ao longo da cadeira.

2 Resumo

Começamos por falar um pouco dos basicos de microcontroladores e sistemas embebidos, seguido do funcionamento de perifericos e I/O e como este pode comunicar com o CPU e memoria, onde neste segundo caso falamos de 3 formas diferentes de comunicação que são polling, Interrupção e DMA.
Após esta introdução aprendemos sobre o basico de Timers. Após os Timers falamos de modelos de comunicação em série, onde vimos/relembramos as diferenças entre comunicação em série e em paralelo e depois aprendemos sobre 3 protocolos de comunicação diferentes que foram SPI, CAN e RS-232C. Para terminar a matéria, aprendemos mais a fundo o funcionamento de memorias, mais especificamente, a diferença entre SRAM e DRAM, memoria cache e memoria virtual.

De forma geral podemos dividir a matéria dada da seguinte forma:

- Microcontroladores, Sistemas embebidos e funcionamento de perifericos;
- Timers;
- Comunicação em série e paralelo e protocolos SPI, CAN e RS-232C;
- SRAM, DRAM, memoria cache e memoria virtual;

3 Microcontroladores, Sistemas embebidos e funcionamento de perifericos

Microcontroladores, ao contrario de microprocessadores, tem RAM, ROM e perifericos integrados.

Sistemas embebidos são sistemas computacionais especializados que realizam uma determinada tarefa.
Estes são mais limitados que um sistema computacional de uso geral mas por isso tem um custo menor.

Palavras que é necessário saber o significado:

- **Cross-compiler:** Compila em um dispositivo para funcionar em outro;
- **Bootloader:** Transfere programas para o microcontrolador;
  Executa no arranque do sistema e usa RS-232/USB;
- **Programa-monitor** Transfere programas para o microcontrolador;
  Executa no arranque do sistema e usa RS-232/USB;
  Diferencia-se do Bootloader porque permite ações de debug;
- **In-Circuit Debugger:** Transfere programa para o microcontrolador;
  Podem usar comunicação standard ou proprietaria;
  Ao contrario dos outros dois metodos, este é um dispositivo de hardware e é usado para a transferencia inicial de um Bootloader ou Programa-monitor;

NOTA: Existe um conceito importante sobre a arquitetura do PIC32 que é o BUS Matrix que tenho de ver.

...

4 Timers

Timers são perifericos bastante uteis que podem ser usados para diversos propositos.
Estes podem ser configurados para contar até diferentes valores para terem diferentes frequencias e até mesmo para terem Duty Cycles diferentes (PWM).
Um exemplo de uma utilização dos Timers é no com o *Watchdog Timer*. Este serve para verificar entre intervalos se todas as funções de um sistema computacional estão em ordem.
Por exemplo: Caso o microcontrolador não tiver dado reset ao Watchdog Timer depois de um certo tempo, significa que algo está errado.

5 Comunicação em série e paralelo e protocolos SPI, CAN e RS-232C

A comunicação entre diferentes elementos pode ser feita em série ou em paralelo. Se for em paralelo, isto significa que vários bits podem ser enviados de uma só vez (Várias ligações em paralelo end-to-end), enquanto que se for em série apenas é enviado um bit de cada vez (Apenas uma ligação end-to-end) e a forma como se sabe quando começa e termina a comunicação é com start e stop bits.
As ligações tambem podem ser classificadas como **Simplex**, **Half-Duplex** e **Full-Duplex**:

- Simplex: Comunicação em apenas um sentido;
- Half-Duplex: Comunicação em ambos os sentidos mas apenas em um sentido de cada vez;

- Full-Duplex: Comunicação em ambos os sentido simultaneamente;

Para comunicações entre dispositivos um pouco distantes um do outro, a ligação em paralelo não é ideal.
Assim usa-se por consequencia a ligação em serie. Na cadeira demos 3 exemplos de protocolos de de comunicação em série que foram **SPI**, **CAN** e **RS-232C**

5.1 SPI

O protocolo SPI é o unico dos 3 dados que é sincrono e o relogio é fornecido pelo master que, ao contrario dos slaves, é unico.
É usado mais para curtas distancias quando comparado aos outros 2 protocolos.

A comunicação é full-Duplex e funciona como um ciclo, isto é, a cada tick do relogio, todos os elementos, sejam eles master ou slave, enviam e recebem informação.
Isto acontece porque o master e o slave são shift registers que a cada tick do relogio enviam o ultimo bit e recebem na primeira posição.

Existem três situações possiveis:
1.Há apenas um slave: a comunicação apenas acontece quando o clock e a sinal SS/ (Slave select com logica negativa) estiverem ambos ligados.
2.Há mais de um slave, e quantidade de sinais SS/ igual ao numero de slaves: a linha MISO dos slaves não selecionados estão em alta impedancia para que a comunicação só seja realizada no slave selecionado.
3.Há mais de um slave mas apenas 1 sinal SS/: Os slaves estão ligados em cascata (Daisy chain) e além de passarem para o proximo tambem guardam os bits até o sinal SS/ ser desativado, sendo que só nesse instante é que é executado o comando recebido.

5.2 CAN

O protocolo CAN é o unico dos 3 que é multi-master, isto é, qualquer elemento pode ser master e iniciar comunicação, é assincrono o que implica que relogio não é transmitido, é Half-duplex e por fim funciona por broadcast, o que significa que o que o master enviar é recebido por todos os slaves, e cabe aos slaves decidir se a informação é util ou não.
É usado em casos onde a segurança é crucial, devido á sua grande capacidade de detetar e evitar erros.

A comunicação é feita por tramas, que são basicamente conjuntos de bits com varios parametros que organizam a informação enviada em grupos.
Um desses grupos é o ID da trama, que serve para diferenciar a prioridade da trama, uma vez que, já que qualquer slave pode virar um master e enviar informação a qualquer momento, tem de haver um mecanismo para escolher qual aceitar primeiro, que no caso do CAN é o ID, onde o master que envia a trama com ID menor tem maior prioridade.

Agora passando para a segurança do CAN, ele tem vários mecanismos para detetar erros, como por exemplo o CRC, que é basicamente um checksum, e o ACK da trama, o tamanho da trama e de cada grupo da mesma e por fim a falta do bit stuffing, um mecanismo do CAN muito importante que faz com que, a cada 5 bit iguais, é adicionado 1 bit extra com o valor contrario, ou seja, se houver 6 bits iguais, isso indica um erro.

Por fim, o ultimo conceito importante dado sobre o CAN é a filtragem de informação pelos slaves, pois uma vez que a comunicação é feita em broadcast, se os slaves tivessem de ler todas as mensagens para saber quais são importantes e quais não são, isso teria um peso computacional muito elevado. Assim, a solução usada é que a informação recebida pelos slaves é posta em um buffer e filtrada com o auxilio de uma **mascara** e um **filtro**.
A forma como funciona é:
Onde a mascara estiver com o valor a 0, é aceito qualquer valor, enquanto que onde estiver a 1, apenas é aceito o valor igual ao que está no filtro.

5.3 RS-232C

O protocolo RS-232C é o mais antigo e por consequencia o mais simples. É assincrono, Full-Duplex e não é nem multi-master nem multi-slave. Na verdade nem ter master nem slave, apenas dois equipamentos que podem comunicar entre si de forma bidirecional ao mesmo tempo, cada um com o seu relogio.
Esta simplicidade tambem vem com alguns problemas, uma vez que tem bastante pouca segurança e pouca imunidade a ruido eletromagnetico.

Quanto á trama, esta é muito mais simples que a do CAN. Apenas apresenta 1 start bit, a secção de dados, o tipo de paridade (N, E ou O) e o stop bit, que pode ser 1 ou 2.

Sobre o RS-232C, o mais complexo e importante de saber é a sincronização, uma vez que, enquanto que no SPI (sincrono) o relogio era enviado pelo master para o slave, e no CAN (assincrono) ninguem usava relogio, no RS-232C (assincrono) cada elemento usa o seu relogio, logo tem de estar sincronizados. A forma como a sincronização é feita é ...

6 SRAM, DRAM, memoria cache e memoria virtual

As memorias são classificadas em termos de tamanho pelo formato (exemplo) 32kx8, onde 32k significa que a memoria apresenta 32 mil palavras e o 8 é o tamanho da palavras, ou seja a palavra tem um tamanho de 8 bits, o que nos leva a concluir que uma memoria que tenha de tamanho 32kx8 tem de tamanho 256k bits.

Existem duas arquiteturas de RAM diferentes: **SRAM** e **DRAM**.
SRAM é mais rapida mas é mais cara por precisar de 6 transistors por celula, enquanto que a DRAM é bastante mais lenta em comparação porque usa condensadores mas é mais barata por celula, o que leva a que seja possivel ter um tamanho maior.

A SRAM tem uma estrutura 2D e cada linha guarda uma palavra, enquanto que a DRAM tem uma estrutura de matriz.
Como a DRAM usa condensadores para guardar os bits nas celulas, este tem de dar refresh á informação antes de os condensadores perderem o valor, e ela faz isso usando o metodo de RAS Only, que consiste em, de forma periodica, carregar o valor de uma linha inteira da memoria para o buffer e recarregar a linha, para dar refresh ao valor no condensador de cada celula da linha e fazer isso a todas as linhas.
Em sistemas computacionais o ideal é usar as vantagens de cada tipo de memoria diferente e criar uma hierarquia de memorias. Assim a memoria mais lenta mas maior, a DRAM, torna-se a memoria principal enquanto que a memoria mais rapida mas menor, a SRAM, torna-se na Cache.

6.1 Cache

A cache é muito mais rapido que a memoria principal, então, quantas mais vezes se puder usar a cache em vez da memoria principal, mais eficiente temporalmente o sistema será.
A gestão de memoria é feita por niveis. Existem alguns casos possiveis:
1. Acessa-se a cache, se a informação estiver lá, lê-se a informação.
2. Acessa-se a cache, se a informação não estiver lá, acedesse á memoria principal e se a informação estiver lá e atualiza-se a cache.
3. Acessa-se a cache, se a informação não estiver lá, acedesse á memoria principal e se a informação tambem não estiver lá e acede-se ao disco e atualiza-se a memoria principal.
É importante destacar que em termos temporais, o caso 1 é infinitamente mais eficiente que o caso 3.
Caso a cache tenha que ser atualizada com dados da memoria principal, esta guarda de cada vez blocos da memoria principal por 3 metodos possiveis: mapeamento associativo, mapeamento direto e mapeamento parcialmente associativo. O mapeamento associativo é o mais eficiente mas é mais caro por causa da quantidade de comparadores que precisa. O mapeamento direto é o menos eficiente mas mais barato porque só precisa de um comparador e o mapeamento parcialmente associativo é o meio termo entre os dois metodos anteriores.

6.2 Memoria virtual

A memoria virtual veio para resolver alguns problemas de gestão de memoria e funciona criando Page tables para cada processo iniciado que são basicamente arrays que residem dentro da memoria principal que convertem endereços virtuais dados pelo CPU em endereços fisicos, que podem ser da memoria principal ou disco, dependendo do valor do valid bit (1 se for da memoria principal e 0 se for do disco).
Se for necessário aceder o disco, isso tem o nome de page fault e é muito caro em termos temporais.
Pode se modificar as page tables para dar ou tirar permissões de leitura, escrita e execução. Isto serve porque diferentes processos podem ter tradições de endereços virtuais para o mesmo endereço fisico mas podemos querer quem, enquanto que um processo

tenha toda a liberdade para alterar o conteudo desse endereço, o outro apenas possa ler o valor.
A page table reside na memoria principal, e por esse motivo, se tivermos a CPU tiver de aceder á memoria principal sempre, isto faz que o uso da cache não seja possivel em termos de eficiencia temporal, uma vez que aceder primeiro á memoria principal para depois aceder á cache não faz sentido.
Para resolver esse problema foi criado a TLB (translation lookaside buffer), que é uma parte de hardware que reside em uma cache especial na MMU e que transforma endereços virtuais em endereços fisicos da cache e só se der miss é que a page table é acessada.
Algumas notas importantes: A TLB não está ciente de onde os dados fisicamente residem (cache ou memória principal). Ela apenas mapeia endereços virtuais para endereços físicos. O mapeamento, tanto na TLB tanto na page table é feito pelo Sistema Operativo.
De forma a entender como o processo é efetuado as situações são as seguinte (Ordem de eficiencia temporal): **1.** O CPU acede á TLB e dá hit. Acede á cache e dá hit.
2. O CPU acede á TLB e dá hit. Acede á cache e dá miss. Acede á memoria principal e dá hit. Atualiza a Cache.
3. O CPU acede á TLB e dá miss. Acede á page table e dá hit. Acede á memoria principal e da hit. Atualiza a TLB.
4. O CPU acede á TLB e dá miss. Acede á page table e dá hit. Acede á memoria principal e da miss. Acede ao disco e dá hit. Atualiza a memoria principal.
5. O CPU acede á TLB e dá miss. Acede á page table e dá miss. Page Fault. Acede ao disco. Atualiza a page table.