Sistemas Tolerantes a Falhas

Aulas Práticas

1- Apresentação (1 aula)

Objectivos:

a. Introdução à programação em C para o sistema CANivete
b. Utilização de entradas e saídas digitais

Sumário:

A placa CANivete: constituição, ligação ao PC, carga e iniciação de programas
O compilador C da Keil
Geração de programas elementares:
    Piscar um LED com temporização por ciclo de execução
    Acender um LED comandado por um botão de pressão
        Leitura e actuação simples
        Actuação após leitura filtrada

2- Interrupções e timers (2 aulas)

Objectivos:

a. Programação em C de rotinas de interrupção
b. Leitura do estado de um botão de pressão com interrupção
   Soluções para tolerar *bouncing*
c. Utilização de temporizadores - Geração de interrupções periódicas
d. Utilização do *watchdog timer* para recuperação automática de *crash*

Sumário:

O conceito de interrupção
O controlador de interrupções do 80C592
Repetição do programa que acende o LED comandado por um botão usando uma interrupção
    A interrupção que faz mudar de estado
    Problemas com este método (*bouncing*)
    Utilização de filtragem
Apresentação da unidade de temporização *timer*
Geração de interrupções periódicas
Repetição do programa que faz o LED piscar usando uma interrupção periódica
Programação do *watchdog timer* para recuperação automática de *crashes*

3- Comunicação série (2 aulas)

Objectivos:

a. Comunicação por porta série assíncrona (RS232)
b. Detecção de erros na recepção
c. Tolerância a erros na recepção com CRC e protocolo de confirmação/repetição

Sumário:

Comunicação CANivete → PC
- Utilização do PROCOMM ou equivalente no PC
- CANivete envia uma ou mais *strings* seguidas
- Execução do programa no PC em janela Windows e em modo DOS
- Visualização do fenómeno de *overrun* em Windows

Geração de um programa em C para ler / escrever na porta série do PC
- Utilização deste programa para substituir o PROCOMM

Comunicação com CRC e *acknowledge*
- Realização de um protocolo para envio de uma *string* de cada vez
  - Definição de uma trama
  - Detecção de erros usando CRC
  - Utilização de *acknowledge* para passar à *string* seguinte ou pedir repetição

4- ADC e PWM (1 aula)

Objectivos:

a. Utilização de entradas analógicas
b. Utilização de saídas analógicas com codificação digital PWM
c. Transmissão de valores analógicos por linha série
   Codificação em ASCII e formatação em trama com CRC
d. Limitações impostas pela largura de banda do sistema de comunicação

Sumário:

Apresentação da ADC e da unidade de PWM

Aquisição periódica de um sinal analógico proveniente de um gerador de funções e respectiva escrita numa saída PWM

Envio para o PC do sinal analógico adquirido
- Devidamente formatado numa trama
- Convertido para ASCII e com CRC
- Usando a linha série como no ponto anterior

5- Comunicação por CAN (2 aulas)

Objectivos:

a. Comunicação série sobre a rede CAN
b. Observação de jitter nas transmissões periódicas induzido pela rede

Sumário:

Apresentação da norma CAN

O controlador CAN
- Inicialização
- Envio e recepção de mensagens

Comunicação um para todos (*broadcast*) pela rede CAN, segundo o modelo Produtor-Consumidor.

Aos pares:
- Um CANivete adquire periodicamente um sinal analógico e envia-o pela rede CAN

Outro CANivete lê da rede CAN esse sinal analógico,
coloca-o na unidade PWM e gera um pulso numa saída digital

Envio para o PC, pela linha série, de informação seleccionada sobre os valores recebidos

6- Redundância ao nível físico em CAN (1 aula)

Objectivos:

a. Observação da tolerância do protocolo a erros só numa linha de sinal
b. Observação da retransmissão automática quando os erros afectam ambas as linhas

Sumário:

Construção de um mecanismo para geração controlada de erros:
Geração de erros numa só linha de sinal
Geração de erros em ambas as linhas

Mini-Projecto: Um sistema CAN tolerante a falhas (3 aulas)

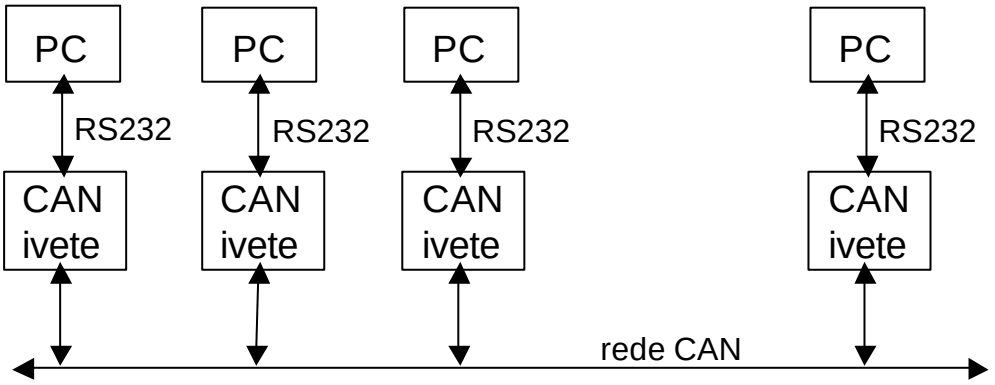
Objectivos:

a. Construção de um sistema com replicação de nodos produtores de informação
b. Utilização da informação replicada em nodos consumidores
    i. Como *backup*
    ii. Com votação de maioria
    iii. Com troca de mensagens para precaver contra falhas bizantinas

Sumário:

Cada grupo terá que construir uma parte do sistema distribuído (a especificar mais tarde):

- Replicação de um nodo produtor de informação:
    - as réplicas deverão detectar a falha de transmissão do nodo principal
    - deverão, ainda, eleger o nodo de *backup* que substituirá o nodo principal
    - a réplica eleita deverá substituir o nodo principal enquanto este estiver inactivo
- Utilização, num nodo consumidor, de informação replicada (transmitida por vários produtores simultaneamente)
    - utilização de um segundo valor por o principal estar corrompido
    - realização de um processo de votação e utilização do valor mais votado
- Construção de um sistema de partilha de informação com comunicação cliente-servidor (envio para outros nodos dos valores recebidos do nodo produtor)
    - verificação da coerência dos valores enviados pelo produtor
    - detecção e tolerância a falhas bizantinas



Outros aspectos pertinentes:

Bibliografia de apoio:

- *CANivete, Manual do utilizador*. MICRO-I/O Serviços de Electrónica Lda. (http://www.microio.pt/produtos/canivete/canivete.htm)
- Manual do 80C592 (http://sweet.ua.pt/~lda/stf/P8XC592_3.pdf) mais informação em: http://www-us.semiconductors.com/pip/P80C592FFA/00
- Manual do µVision2-IDE da Keil Software (http://www.keil.com/)
- Paret, D., *Le bus CAN*, Dunot, Paris, 1996.
- Lawrenz, W., *CAN system engineering: from theory to practical applications*. Springer-Verlag, New York, 1997.

Controlador 80C592:

É um micro-controlador de 8 bits, fabricado pela Philips, baseado no micro-controlador 8051 ao qual foi anexado um controlador de rede CAN que se encontra integrado no mesmo circuito. Para além do controlador CAN possui, como é habitual nos micro-controladores, memória interna que está dividida em memória de programa e de dados, vários portos de entras/saídas digitais, uma UART para comunicação série assíncrona RS232, uma unidade de temporização com 3 timers e 1 *watchdog*, 15 entradas de interrupção, 8 entradas analógicas e 2 saídas PWM.
É de realçar que este tipo de controlador usa uma arquitectura *Harvard*, i.e. com barramentos e memória de dados e de programa separados.

Compilador C da Keil, µVision2:

É um compilador integrado num ambiente gráfico de desenvolvimento (IDE), muito versátil e fácil de utilizar. Como permite gerar programas para muitas variantes do micro-controlador 8051, é necessário começar por especificar que o controlador que se vai usar é o 80C592. Para iniciar o ambiente de desenvolvimento deve-se executar o comando UV2 dentro da pasta Keil\UV2.
O compilador obedece à norma ANSI C com algumas extensões para facilitar a utilização de pormenores arquitecturais do 8051.

Geração e execução de programas na placa CANivete:

Para executar um programa escrito em C na placa CANivete, são necessários os seguintes passos:

1- **editar** o programa, e.g. no editor integrado
2- **compilar** o programa e corrigir eventuais erros
3- gerar o ficheiro executável para o CANivete em formato **Intel HEX**
4- ligar a alimentação da placa CANivete
5- ligar a porta série do PC de desenvolvimento à da placa CANivete
6- verificar que a placa CANivete está em **modo *download*** (*reset* 0)
7- executar, numa janela de DOS no PC de desenvolvimento, o comando:
   Pcload <nome_ficheiro> 0 1
   que carrega o ficheiro executável em formato Intel HEX no CANivete 0 através da COM 1.
8- **iniciar a execução** do programa na placa CANivete premindo o botão de *reset* 1.

Cuidados a ter na execução dos trabalhos:

Dada a fragilidade das placas, deve-se usar o máximo cuidado na sua manipulação, particularmente na colocação da ficha série e na colocação dos fios na régua de contactos.
Ninguém deve ligar o circuito sem o mesmo ter sido revisto pelo professor.
A nota final desta disciplina só será dada depois da devolução do material em boas condições.