Arquitectura de Computadores

LETI, LEE

Projecto de Arquitectura de Computadores

Relógio/Cronómetro/Despertador

2013/2014

1 – Objectivos

Este projecto pretende exercitar as componentes fundamentais de uma cadeira de Arquitectura de Computadores, nomeadamente a programação em linguagem *assembly*, os periféricos e as interrupções.

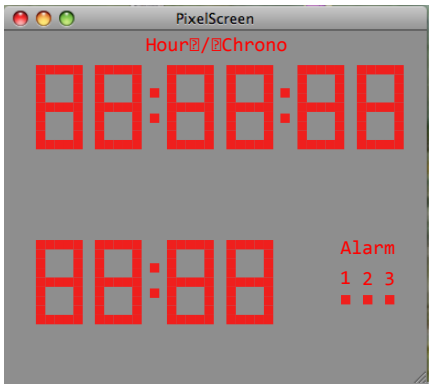
O objectivo deste projecto é realizar um relógio digital com capacidade de cronómetro e despertador. O tempo é mostrado num écran LCD, sendo igualmente possível gerar alarmes de despertar e cronometrar tempo (progressivamente e regressivamente) em simultâneo com as outras funções.

2 – Especificação

O relógio deve ser concretizado sobre uma arquitectura baseada no microprocessador PEPE. A versão final será experimentada sobre a placa PepeOnBoard que emula a arquitectura em que o projecto é desenvolvido.

Existe pelo menos um relógio de tempo-real (RTC) que serve de base de tempo a todo o sistema. É com base neste RTC que se opera a contagem do tempo.

Em modo relógio, o sistema deve apresentar o tempo no formato HH:MM:SS. Esta informação deve ser apresentada no display do sistema com o formato ilustrado na figura seguinte.



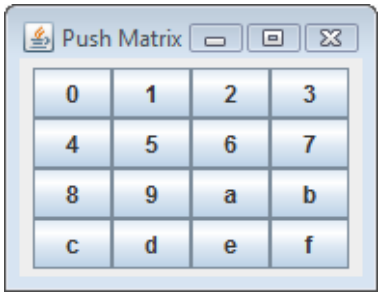
Este display é um LCD com 32 x 32 pixels. Existem 2 campos básicos de representação de tempo neste LCD: um referente ao tempo actual e outro referente ao tempo de alarme. O primeiro serve para mostrar a hora actual (horas, minutos e segundos) quando em modo relógio, ou o tempo passado / em falta, quando em modo cronómetro, exactamente com o mesmo formato do usado para representar o tempo actual; o segundo, destina-se à indicação do tempo de um dos três alarmes activos, à escolha do utilizador através de uma tecla da matriz de botões.

No modo cronómetro, o campo do tempo actual é usado para contagem do tempo decorrido ou em falta para uma dada meta temporal. Quando se passa a este modo de funcionamento, o cronómetro começa a contar o tempo até ser atingido o tempo pré-programado ou decrementa o tempo a partir de um valor pré-programado até chegar a zero, consoante se escolha o modo progressivo ou regressivo. Pode parar-se o cronómetro ou fazer o seu *reset* através de 2 botões de pressão.

Há um display hexadecimal que mostra os décimos de segundo (ou centésimos, em opção) no modo cronómetro.

No modo de acerto de alarme, deve mostrar-se a hora: min programado para esse alarme, acendendo-se um quadrado de *n* pixels por debaixo do número do alarme. Em cada alarme existe um LED que indica se esse alarme está ou não activo.

Os comandos do relógio devem usar uma matriz de botões (ver figura junta). Destes, a atribuição das suas funções é a seguinte:



- Tecla para acerto do tempo actual - f .
- Tecla para programação do cronómetro – c.
- Tecla para programação do alarme – a.
- Tecla para escolha do modo de funcionamento – 0 (pressionando sucessivamente passa ciclicamente do modo relógio para o modo cronómetro).
- Tecla para escolha do modo de cronómetro – 4 (pressionando sucessivamente passa ciclicamente do modo progressivo para o modo recessivo).
- Teclas para escolher o alarme – 1, 2 e 3.

Existem 2 botões de pressão e 2 LEDs que têm as funções auxiliares seguintes:

- botão de pausa / elapsed do cronómetro– permite pausar o cronómetro para se fazer uma leitura enquanto o cronómetro continua a contar em *background.*
- botão de reset do cronómetro –permite parar o cronómetro na 1ª actuação e fazer o seu reset, na 2ª activação.
- Led 1 – indica se o alarme selecionado está ou não activo.

- Led 2 – a piscar, indica se um dos três alarmes disparou.

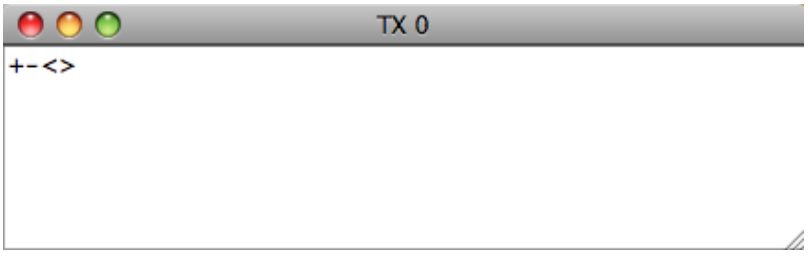
Quando carrega na tecla de acerto do tempo actual, o relógio deixa de contabilizar os impulsos do RTC (ou melhor, ignora-os). Neste modo de acerto, introduza os dígitos correspondentes às horas e aos minutos (segundos a zero), após o que deve carregar na mesma tecla que usou para entrar no modo de acerto para retomar o funcionamento normal.

Igualmente, para a programação do cronómetro deve accionar a tecla de acerto correspondente (ver acima) passando o sistema ao acerto do valor a cronometrar, **mas continuando a contar o tempo actual (o relógio não pára)**. Se carregar na tecla da selecção do modo de cronómetro a contagem aumenta ou diminui consoante o modo escolhido.

Quando carrega na tecla para programar o alarme, deve continuar a mostrar o tempo actual no campo respectivo. Para modificar o valor do alarme deve escolher qual dos três alarmes quer programar (1, 2, ou 3). Em seguida, deve introduzir as horas e os minutos através do teclado, da mesma forma que no acerto da hora ou do tempo a cronometrar. Após a introdução da hora de alarme, se carregar na tecla com o número desse alarme activa-o; se carregar na tecla de programação de alarme, esse alarme fica programado mas não activo.

Pode consultar a hora de cada um dos alarmes através da tecla de programação destes, escolhendo em seguida o seu número e não introduzindo nova programação. Se carregar na tecla de programação de alarme saí desse modo; se carregar sucessivamente no número dos diferentes alarmes pode consultar o que está programado em cada um deles. Para activar um desses alarmes que não esteja activado basta pressionar novamente no seu número e o LED de activação deve acender-se. Atenção que os três alarmes podem estar activos simultaneamente. Quando algum disparar, deve ser piscado o LED de alarme.

Os comandos anteriormente definidos podem também ser dados através de um terminal remoto que usa uma ligação série assíncrona, por intermédio de uma interface série concretizada por uma MUART. O aspecto deste terminal será o seguinte:

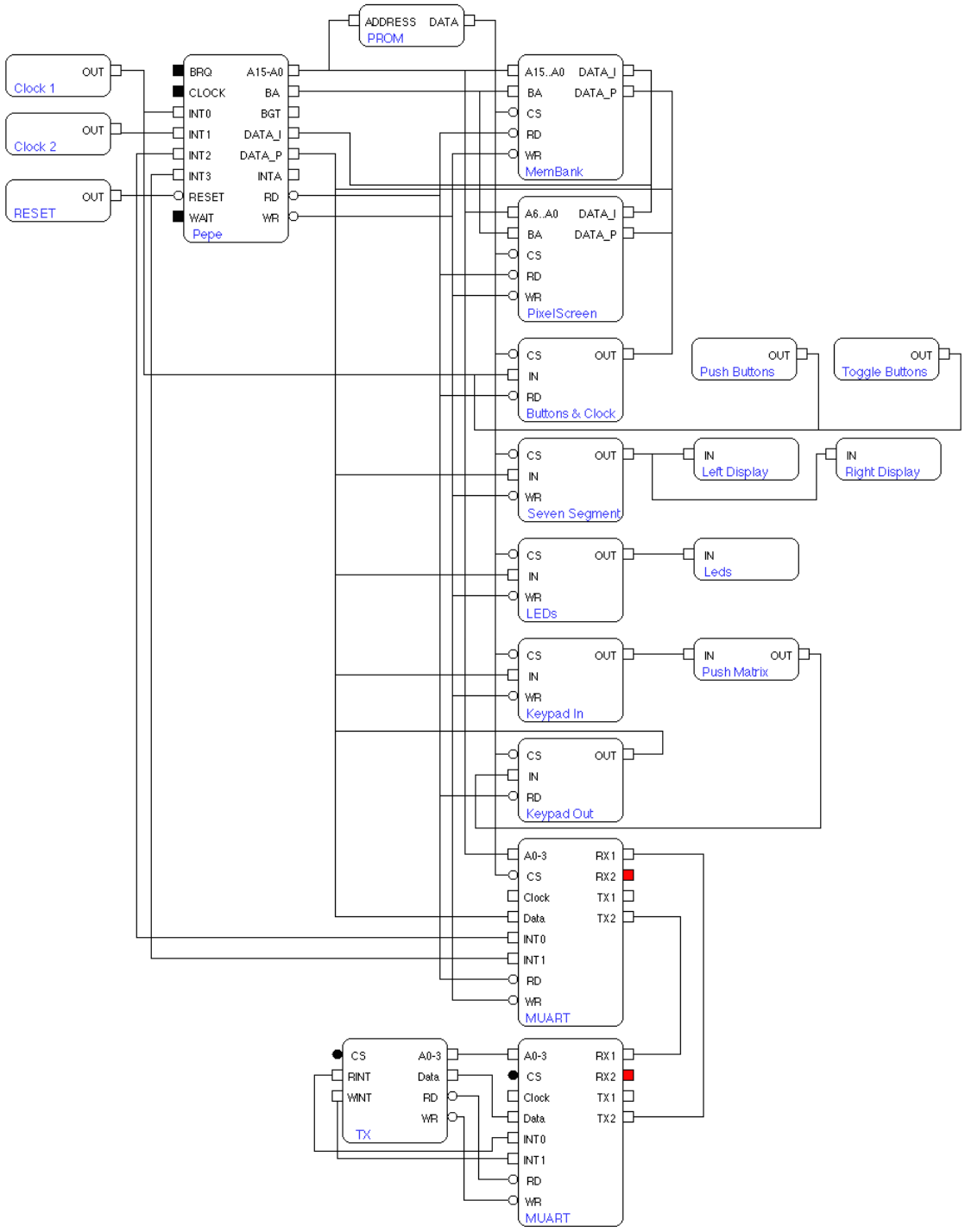


Neste terminal, deve arbitrar as teclas do seu computador (em modo de simulação selecciona a janela do terminal e escreve com o seu teclado) para as várias funções antes definidas para a matriz de botões e botões de pressão. Os alarmes são substituídos por mensagens.

Há liberdade para alterar as especificações do sistema, desde que não se reduza a complexidade de implementação (confirmar ideias com o docente). A criatividade e a demonstração do domínio da tecnologia são sempre valorizadas!

3 – Implementação

A figura seguinte mostra o circuito a usar (fornecido, ficheiro **pepeonboard.cmod**).



Podem observar-se os seguintes módulos, cujo painel de controlo poderá (e deverá, na maior parte dos casos) ser aberto em execução (modo Simulação):

- `Clock1` – Relógio de tempo real, pré-programado para um período de 100 milisegundos, e que deve ser usado como base para a geração do tempo de sistema e para as restantes temporizações envolvidas. Este relógio deve ser usado como interrupção (INT0), mas em versões intermédias pode ser útil lê-lo num ciclo de instruções. Por isso, também liga ao bit 4 do porto de entrada.
- `Clock2` – Relógio de tempo real, pré-programado para um período de 50 milisegundos, e que pode ser usado para aumentar a precisão de contagem do tempo no modo cronómetro em intervalos 5 centésimas de segundo. Este relógio deve ser usado como interrupção (INT1). (opcional)
- Matriz de pixels (`PixelScreen`) – écran de 32 x 32 pixels. É acedido como se fosse uma memória, de 128 bytes (4 bytes em cada linha, 32 linhas). Sugestão: faça rotinas para ligar/desligar o pixel na posição X, Y (tem de determinar o byte e o bit correcto a partir de X e Y).
- `MUART`-1 e 2 para comunicação série com o terminal remoto. Devem ser programadas para permitir a máxima velocidade de comunicação possível entre o sistema e o terminal. `Terminal remoto`, ligado à MUART-2, para controlar o relógio remotamente.
- Display hexadecimal, para os centésimos de segundo. Está ligado aos bits 7-0 do porto `SevenSegment`.
- Teclado (`PushMatrix`), de 4 x 4 botões, com 4 bits ligados ao porto `KeypadIn` e 4 bits ligados ao porto `KeypadOut` (sempre bits 3-0). A detecção de qual botão está carregado é feita por varrimento. O processador tem de escrever apenas um bit a 1 no porto `KeypadIn` (para cada um dos bits 0 a 3), e ler o porto `KeypadOut` em cada caso.
- Botões de pressão, interruptores e LEDs ligados aos portos `Buttons` e `LEDs` para funções auxiliares do sistema

O mapa de endereços (em que os dispositivos podem ser acedidos pelo PEPE) pode ser definido pelos alunos (apenas no simulador). Para arranque do desenvolvimento, apresenta-se a seguinte configuração de endereços, correspondente ao cmod fornecido com este enunciado:

Dispositivo	**Endereços**
RAM (MemBank)	0000H a 7FFFH
Buttons&Clock	8000H a 8FFFH
LEDs	9000H a 9FFFH
Seven Segment Display	A000H a AFFFH
Keypad In/Out	B000H a BFFFH
LCD (PixelScreen)	C000H a C07FH
MUART	D000H a DFFFH

Na discussão do projecto, poderá ser fornecido um outro mapa de endereços que cada grupo deve concretizar através da modificação do cmod fornecido. Para que esta mudança ocorra sem grande impacto no código desenvolvido pelos alunos, devem estes ter a preocupação de definirem, através da directiva EQU, todos os endereços respeitantes aos dispositivos da arquitectura que usam. Evita-se assim uma alteração substancial e penosa do código desenvolvido até aí. Atenção que para correr na placa PepeOnBoard o mapa de endereços fornecido deve ser respeitado.

O programa deve usar programação cooperativa e estar organizado em processos, tais como:

- varrimento do teclado (sugestão: quando um botão é carregado e libertado, altera uma variável em memória correspondente a esse botão. Os restantes processos lêem as variáveis dos botões que lhes interessam)
- interacção com terminal remoto
- gestão de alarmes

Para além dos processos devem ser programadas as seguintes rotinas de serviço de interrupção:

- interrupções de RTC
- interrupção de recepção da MUART;
- interrupção de transmissão da MUART.

Notas:

- A quantidade de informação mínima a escrever no PixelScreen é de um byte. Por isso, para alterar o valor de um pixel, tem primeiro de se ler o byte em que ele está localizado, alterar o bit e escrever de novo no mesmo byte
- **O RTClock (bit 4) e o teclado (bits 3-0) partilham o mesmo porto de entrada**. Por isso, terá de usar uma máscara ou outra forma para isolar os bits que pretender, após ler este porto.
- A leitura e escrita na MUART é feita por interrupção. Quando chega um carácter do terminal remoto, este deve ser colocado numa variável (pela rotina de interrupção) para que o(s) processo(s) que interpretam os comandos saibam o que fazer. Quando se pretende escrever no terminal remoto, a mensagem a escrever deve ser inserida num buffer (enviando o 1º carácter para a MUART) para que a rotina de interrupção de transmissão vá escrevendo os caracteres no registo de dados de transmissão da MUART.
- Na execução sobre a placa PepeOnBoard, é preciso ter em atenção ao "bouncing" associado aos botões de pressão e interruptores da placa.
- As rotinas de interrupção dos RTCs **DEVEM SER MUITO CURTAS** (não fazem mais do que incrementar uma ou um número reduzido de contadores em memória). Soluções em que todo o trabalho é realizado dentro das rotinas de interrupção são considerados **INSATISFATÓRIAS**!

O programa deve estar estruturado de modo a que se quiser escrever caracteres numéricos com formato diferente no LCD não tenha de alterar o código mas apenas uma tabela de dados. Deste modo para desenhar os caracteres numéricos no LCD deve usar tabelas. Os caracteres numéricos podem ser descritos por uma tabela em que o número de elementos é igual ao número de linhas que o caracter tem. Por outro lado apenas se aproveitam os n bits de menor peso, em que n representa a largura do caracter a escrever. Para desenhar os caracteres é só ler esses valores da tabela e escrever os bits que interessam no LCD.

Juntamente com este documento, encontrará um ficheiro excel (**screen.xlsx**), que reproduz os pixels do ecrã. Foi usado para realizar algumas experiências na conceção deste trabalho e poderá também usá-lo para desenhar algumas dos números e traduzi-las para uma tabela. Este aspeto é opcional.

Finalmente:

- Como norma, faça PUSH e POP de todos os registos que use numa rotina e não constituam valores de saída. É muito fácil não reparar que um dado registo é alterado durante um CALL, causando erros que podem ser difíceis de depurar;
- Vá testando todas as rotinas que fizer e quando as alterar. É muito mais difícil descobrir um bug num programa já complexo e ainda não testado;
- Estruture bem o programa, com zona de dados no início e rotinas auxiliares de implementação de cada processo junto a eles;
- Não coloque constantes numéricas (com algumas exceções, como 0 ou 1) pelo meio do código. Defina constantes simbólicas e use-as depois no programa;
- Produza comentários abundantes, não se esquecendo de cabeçalhos para as rotinas com descrição, registos de entrada e de saída.