TÉCNICO LISBOA

Sistemas Computacionais Embebidos

1ª Parte

Sistema de Monitorização e Alarme

Francisco Leal, 72939

Gonçalo Ribeiro, 73294

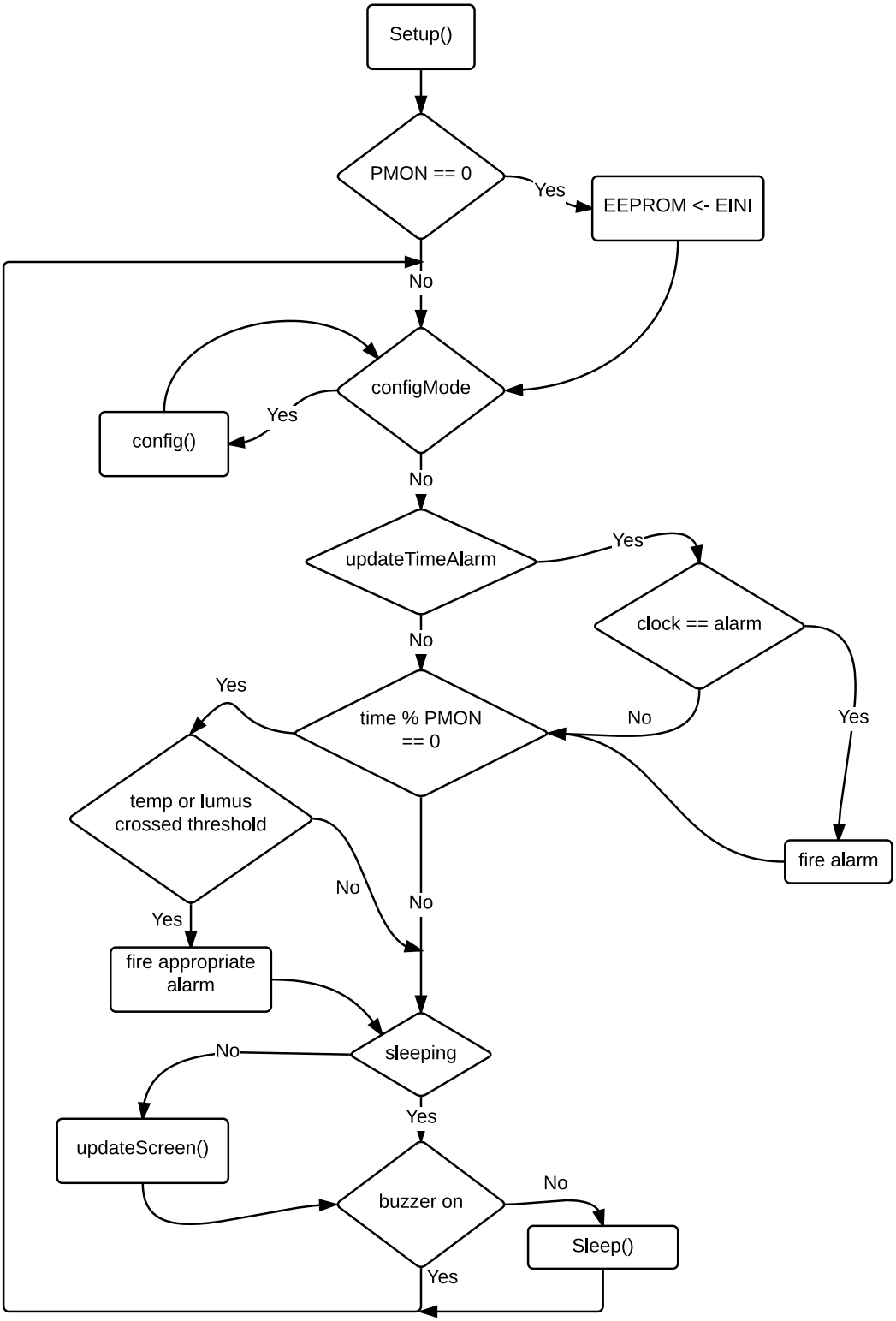
Ricardo Amendoeira, 73373

Docente: Prof. Carlos Almeida

17 de Abril de 2015

Conteúdo

1 Fluxograma



2 Estruturas de Dados

2.1 Buffer Circular

Os registos armazenados estão alinhados a 8 bytes, de modo a que não haja escritas distribuídas por duas páginas. Assim pode-se sempre escrever 8 bytes de uma vez, sem nunca ter que verificar se é necessário continuar a escrever noutra página. Os últimos 8 endereços estão reservados para guardar a informação necessária à manutenção do buffer circular. Para endereçar a EEPROM externa é necessário enviar 2 bytes de endereço através do protocolo I^2C, visto que a EEPROM tem 32K $\times$ 8 bits.

2.2 Tempo

Para o controlo do relógio e do alarme foi definida uma estrutura que contém os segundos, minutos e horas, cada um guardado como um byte. A interrupção do Timer1 actualiza a cada segundo a estrutura que guarda o tempo. Outros eventos que podem ocorrer apenas em tempos múltiplos de um segundo são também sinalizados por esta rotina de interrupção.

2.3 Configuração

No modo configuração é possível ligar e desligar os alarmes individualmente e assim alterar os vários valores que são configuráveis (clock, alarm, limiar de temperatura e de luminosidade), por exemplo, é possível ligar apenas o alarme da temperatura e não ligar os restantes alarmes.

3 Rotinas

3.1 `Config()`

Quando em modo normal, se pressiona o botão S3 uma interrupção é chamada que activa/altera o modo de configuração. Dentro do `config()` entra-se em modo de configuração e o programa mantêm-se nesse modo até que se tenha passado por todos os parâmetros que podem ser configurados – tempo, alarme, limiar de temperatura e luminosidade. O botão S3 permite portanto seleccionar o parâmetro a alterar, enquanto que o botão S2 permite alterar o valor seleccionado.

Foi implementado *debounce* para o botão S2, de forma a que cada vez que se carrega neste botão, o valor que está a ser alterado seja incrementado em apenas uma unidade. Não foi implementado *debounce* para o botão S3 visto que este tem um schmidtt trigger à entrada, o que resolve o *bounce* neste botão.

De modo a minimizar o consumo energético é registado que parâmetros são alterados de modo a apenas actualizar no ecrã e em memória os valores que realmente foram redefinidos pelo utilizador.

3.1.1 Algoritmo de Debouncing

O algoritmo usado para debouncing consiste em encontrar o momento em que o utilizador larga o botão (o flanco descendente). Para isto é usado um registo das últimas 16 leituras do botão S2 (guardado numa variável de 16 bits), shiftando para a esquerda cada vez que

se faz nova leitura e adicionando o valor lido. O flanco descendente é encontrado quando apenas há um registo de botão pressionado e é o registo mais antigo, ou seja, 0x8000.

```
boolean incrementarComS2(* valor)
{
        ultimasLeituras = (ultimasLeituras << 1) | leituraActual();
        if(ultimasLeituras == 0x8000) // flanco descendente
                (*valor)++;
                return True;
        else
                return False;
}
```

3.2 Alarmes

Os alarmes de luminosidade e temperatura têm período de amostragem `PMON`. Se os limiares de temperatura ou de luminosidade forem cruzados (em qualquer sentido), o respectivo alarme dispara. Para fazer a verificação é mantido em memória o valor da amostra anterior. A verificação do alarme do relógio é feita de segundo em segundo. Antes de serem efectuadas comparações, as interrupções são desligadas para garantir a consistência destas.

3.3 Interrupções – Timer1, S3, WREN e LVD

A interrupção do Timer1 é chamada de segundo a segundo e actualiza a hora do relógio. Nesta interrupção também é verificado se já passaram `PMON` segundos desde a última verificação e é preciso voltar a verificar os valores dos vários alarmes.

A rotina de interrupção `S3_isr()` é invocada sempre que se utiliza o botão de pressão S3, que altera o campo selecionado no modo configuração.

A rotina `LVD_isr()` é invocada quando é detectada uma queda na tensão de alimentação. Através de variados testes não foi possível registar na EEPROM interna mais do que um byte pelo que foi tomada a decisão de registar os minutos, as horas e os valores de limiar para os alarmes de temperatura e luminosidade sempre que estes são alterados. Desta forma, apenas temos que registar o valor dos segundos quando a rotina `LVD_isr()` é chamada.

De modo a saber se o que foi registado na EEPROM interna é válido ou não, um valor predefinido de `0xFF` é colocado na posição destinada ao campo dos segundos no início da execução do programa. Se ao iniciar o programa o valor que estiver nesta posição de memória não for um valor válido para o campo dos segundos então é porque a interrupção LVD não teve tempo de escrever na EEPROM interna. Neste caso o relógio é iniciado a `00:00:00`.

Os limiares dos alarmes de temperatura, luminosidade e alarme são inicializados a 0, a não ser que o checksum destes dados esteja correcto. O *checksum* é a soma dos valores anteriormente referidos.

A rotina `EEwriteDisable_isr()` é invocada quando a escrita que está a decorrer para a EEPROM interna acaba. Esta rotina faz *disable* à escrita para a EEPROM interna.