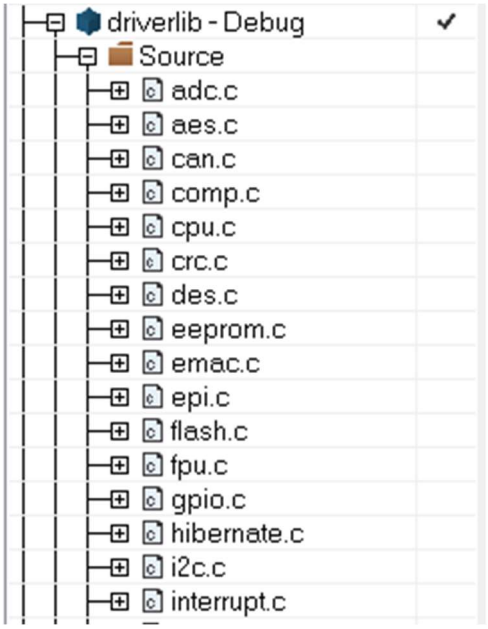
LAB 2 - ELF74 - Sistemas Embarcados

Alunas: Mariana Bittencourt Junghans e Henrique Mazzuchetti

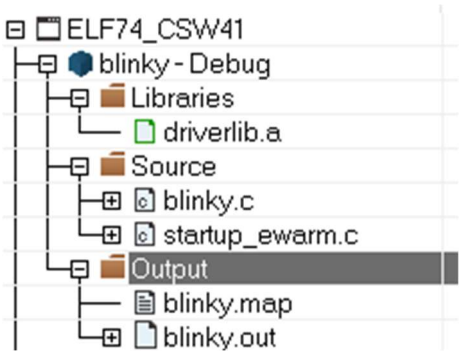
Professor: Eduardo N. dos Santos

1- Estrutura de uma biblioteca (driverlib)



Uma biblioteca é composta de vários arquivos fonte .c, conforme imagem.

2- Estrutura de um executável



Um projeto de um executável (nesse caso, blinky) possui três pastas: Libraries, Source e Output.

Na pasta Libraries, está o arquivo .a, que é uma biblioteca estática criada pelo compilador.

Na pasta Source, estão os arquivos .c, que são os arquivos-fonte. O arquivo blinky.c contém o código que faz o LED piscar, e o arquivo startup_ewarm.c é um código de inicialização do IAR.

Na pasta Output estão os arquivos .map e .out, gerados após a compilação. O arquivo .map serve para realizar o Debug da aplicação, e o arquivo .out é o arquivo executável gerado durante a compilação dos arquivos-fonte.

3- Estrutura do arquivo main.c do projeto blinky

O arquivo main utiliza bibliotecas padrões da linguagem c e também a driverlib para implementar a função de piscar o led da placa TIVA.

Ela inicia configurando o clock que o processador irá utilizar.

Depois habilita as portas que levam ao LED embutido da placa (GPIO N).

Depois define essa porta como saída, e habilita o GPIO como digital.

Então, em um loop infinito, escreve “1” na porta, aguarda um certo tempo (utilizando SysTick) e então escreve “0”, aguardando mais um certo tempo, definido pelo programador.

4- Estrutura do arquivo startup_ewarm.c

Esse arquivo define alguns handlers para quando ocorrerem interrupções ou exceções, como RESET, MPU Fault, Systick, entre outros.

5- Processo de inicialização do projeto

A função ResetISR do arquivo startup_ewarm.c é chamada, para ativar a função de ponto flutuante antes que a função main seja chamada. Isso é necessário pois há a possibilidade da função main utilizar ponto flutuante e para isso essa função deve ter sido habilitada anteriormente.

6- Uso do volatile na variável ui32Loop

A keyword volatile é utilizada para impedir o compilador de realizar a otimização no código, garantindo assim que cada atribuição de variável volátil irá realizar o acesso de memória correspondente.

Essa otimização que é feita quando as variáveis são não-voláteis funciona da seguinte forma: A variável não é relida a cada uso, pois admite-se que não irá mudar muitas vezes durante o código.

Como no exemplo blinky.c o loop que utiliza essa variável faz com que ela mude de valor constantemente, é necessário que ela seja declarada como volátil.

7 e 8 - Utilizar o SysTick para piscar o LED da placa TIVA

(Código modificado no arquivo main.c dentro do workspace ELF74)

Basicamente, utilizou-se as seguintes funções:

```
void LedON(void)
{
   //
   // Turn on the LED.
  //
   GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_0, GPIO_PIN_0);
}

extern void contadorTempo(void)
{
  contadorSysTick = contadorSysTick + 1;
}

void delay(int seconds)

{
  SysTickEnable();

   while (contadorSysTick < seconds)
    {
    }

  SysTickDisable();

  contadorSysTick = 0;
}

void LedOFF(void)
{
  //
   // Turn off the LED.
  //
   GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_0, 0x0);
}
```

A função contadorTempo foi definida no arquivo startup_ewarm.c no vetor de interrupções na posição referente ao SysTick. Assim, quando ocorrer uma interrupção do SysTick, essa função será chamada e incrementará o contadorSysTick.

A função delay() recebe a quantidade de segundos que devem passar naquela função. Utilizando a variável contadorSysTick, faz um loop while até que essa variável atinja o valor desejado. Assim, cria o delay requerido.

A função main:

```
   //
   // Loop forever.
   //
   while(1)
   {

    LedON();

    delay(1);

    LedOFF();

    delay(1);
   }
```

Apenas faz um loop infinito, chamando as funções de ligar e desligar o LED e a função delay.

Projeto – LAB2 – Jogo de tempo

1- Planejamento das fases do processo de desenvolvimento

- Estudo da biblioteca TivaWare, principalmente a API SysTick.
- Configuração do projeto na IDE IAR Workbench (inclusão de driverlib e blinky)
- Implementação do código a partir de exemplos na documentação TivaWare e também do arquivo blinky.c
- Teste na placa TIVA
- Correção de erros se houver
- Mais testes até chegar à solução final
- Gerar a documentação do código

2- Definição do problema a ser resolvido

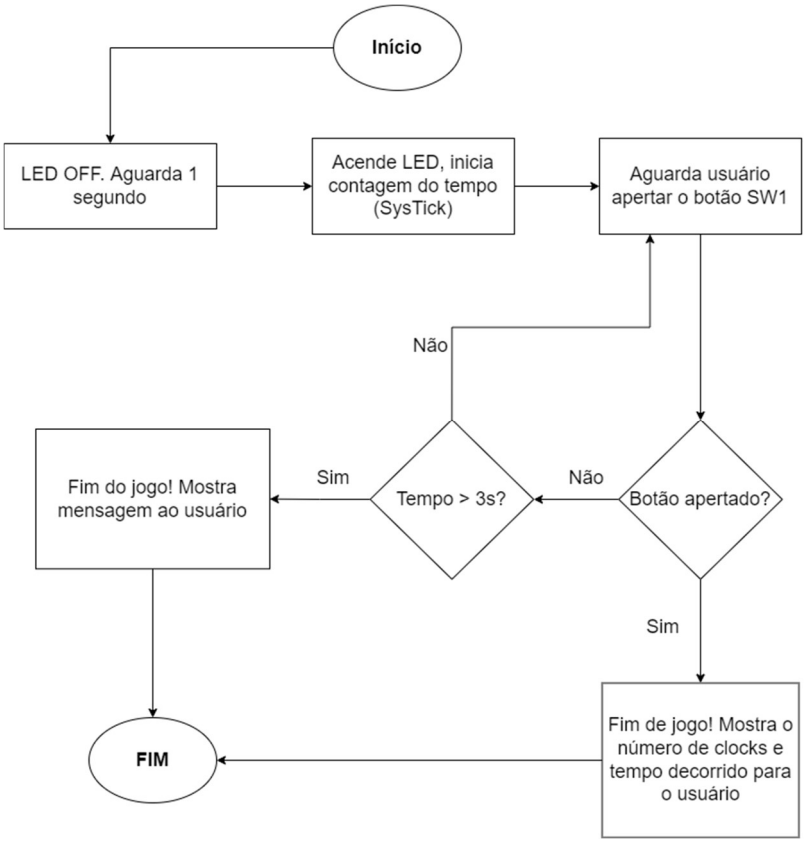
O objetivo da aplicação a ser desenvolvida no Lab2 é medir o tempo de reação do usuário, isto será feito acendendo um LED e medindo o tempo até o usuário pressionar um botão. Pode até ser entendido como um jogo onde o objetivo é responder no menor tempo possível.

3- Especificação da solução

1- O jogo deve ligar o LED D1 para informar o jogador do início da contagem de tempo.
2- O LED deve ser aceso até 1 segundo após o início da operação da placa.
3- O jogo usa o botão SW1 para entrada de dados pelo usuário.
4- O jogo deve apresentar a contagem de tempo no Terminal do IAR indicando o número de clocks entre o LED acender e o botão SW1 ser pressionado e o valor de tempo correspondente em ms.
5- O limite superior de contagem de tempo é o equivalente a 3 segundos.
6- Usar funções da TivaWare para acesso a I/O, SysTick e temporização.
7- A solução deve fazer uso de interrupções, obrigatoriamente de GPIO e opcionalmente do SysTick.
8- O vetor de exceções deve estar em memória Flash e não na RAM.

4- Projeto (design) da solução

Fluxograma:



5- Edição do código da solução

Apesar de ser apenas o segundo laboratório, foi um dos mais desafiadores do semestre.

Isso porque muitos problemas aconteceram e no início não tinha entendido como resolvê-los.

No fim, entretanto, algumas lições foram aprendidas, principalmente em relação a interrupções e às suas respectivas ISRs.

Destaco duas importantes lições: Primeira, não devemos utilizar funções como printf dentro de ISRs. Isso porque essas funções geram interrupções próprias para poder mostrar os valores. O certo é habilitar alguma flag dentro da interrupção e então utilizar essa flag na função principal para mostrar os valores desejados.

Segunda, se uma variável é global e é utilizada tanto na função principal como na ISR, deve ser declarada como volatile.

Como no início não estava fazendo isso, o valor dentro do registrador relacionado à variável não se modificava, o que comprometia o funcionamento do código.

A variável nesse caso era contadorSysTick, e como eu a utilizava como base para calcular outros valores, esses valores se comportavam de maneira estranha e não eram mostrados corretamente.

10 - Teste e depuração

Como mencionado no item anterior, até chegar à solução pretendida demorou algum tempo. Mas ao fim, o teste foi bem-sucedido e, ao iniciar o programa, o LED acende, e, se não há acionamento do botão, após 3 segundos o LED apaga e o jogo acaba, com o jogador saindo “perdedor”, mostrando uma mensagem correspondente no terminal. O jogo reinicia.

Caso o botão seja acionado antes de 3 segundos, o jogador “vence” e uma mensagem diferente é mostrada no terminal. O jogo também reinicia nesse caso.