UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Relatório do Lab 2

Disciplina de Sistemas Embarcados – Prof. Douglas Renaux

Autores: João Victor Laskoski e Luis Camilo Jussiani Moreira

Versão: 29-Ago-2022

1 Introdução

Este presente relatório trata da prática de laboratório para disciplina. Em que, o experimento é desenvolver um jogo rudimentar, o qual irá detectar o tempo de reação do usuário. Sendo assim, serão abordados conceitos de GPIO, interrupções e temporizador, sendo executados em cima do microcontrolador TIVA TM4C1294XL.

2 Planejamento das fases do processo de desenvolvimento

.Entendimento dos objetivos da prática

.Entendimento dos requisitos funcionais e não funcionais

.Estudo do hardware a ser utilizado, como os botões e LEDs e em quais portas os mesmos estão ligados.

.Estudo do software, principalmente na biblioteca TivaWare, para configuração e execução de todas as partes necessárias para implementação do projeto.

.Estudo de como configurar e utilizar o GPIO através da biblioteca TivaWare.

.Estudo de como configurar e utilizar interrupções através da biblioteca TivaWare.

.Estudo de como configurar e utilizar o temporizador através da biblioteca TivaWare.

.Testes simplórios de GPIO, interrupções e temporizador.

.Execução da prática e testes finais.

3 Definição do problema a ser resolvido

Os problemas para serem tratados na prática estão em indicar o início do jogo ao usuário (através do LED), detectar a reação do usuário (através do *user button*), informar o tempo de reação do usuário (através do terminal) e por fim, detectar tempos limites caso o usuário não tenha a devida reação ao jogo ser iniciado, o qual será feito por intermédio de temporizadores.

4 Especificação da solução

A solução do problema faz uso de conceitos de GPIO, interrupções e de temporizador do microcontrolador TIVA TM4C1294XL.

O uso de GPIOs será feito para realizar a manipulação do LED e da chave de usuário, possibilitando assim a leitura e escrita (no caso do LED) de dados desses periféricos.

O controle do tempo durante o processo será realizado configurando um temporizador. Assim, é possível saber em um determinado momento quanto tempo (período de clocks) se passou desde o início do jogo, além de configurar um tempo máximo de jogo (no caso 3 segundos).

Além disso, o uso de interrupções se dá para identificar o final do jogo, tanto se o jogador pressiona o botão ou se passou 3 segundos do início. Sendo assim, uma interrupção está ligada ao temporizador e outra à chave de usuário.

Todo o uso dos periféricos descritos acima foi feito de modo a cumprir os requisitos funcionais e não funcionais determinados no slide de especificação do laboratório 2.

5 Estudo da plataforma de HW

Na questão dos periféricos utilizados, como o botão e o LED que estão conectados no GPIO, o botão SW1 está conectado ao pino J0 e o LED D1 está conectado ao pino N1 (rever o número).

A lógica para ativação do led é positiva, enquanto para o botão de usuário é a lógica negativa. Para temporização, foi utilizado o temporizador padrão, em que é possível escolher as quantidades de bits para contagem máxima (podendo ser configurado em 16/32/64 bits).

6 Estudo da plataforma de SW

Para garantir que o vetor de exceções esteja na Flash, é necessário fazer a declaração de uma função no arquivo .c/.cpp do projeto a qual será chamada quando ocorrer a interrupção. Por fim, no arquivo start_ewarmup.c, na variável const (quais são armazenadas na memória de código), no parâmetro em que está com o comentado GPIO J, trocar para o nome da função criada anteriormente.

No que diz respeito ao GPIO, foi identificado as funções para habilitar o clock para a porta, como também a função que verifica se a porta está preparada. Além disso, outras funções como configurar o pino como um GPIO para saída e também as funções que escrevem ou fazem a leitura dos pinos de GPIO.

Contudo, para o botão, foi utilizado o driver de botão, tomando os arquivos .c e .h na pasta drivers da biblioteca da TivaWare, em que a configuração do botão se tornou muito mais simplória.

No quesito das interrupções, após declarar o handler que será chamada para cada interrupção, foi verificado as funções para configuração da própria interrupção, das quais foram necessárias as seguintes funções:

- GPIOIntEnable() : Habilita o GPIO específico, o qual é colocado como parâmetro a porta base e o pino o qual ocorrerá a interrupção.
- GPIOIntTypeSet(): Configuração para dizer se a interrupção ocorrerá em borda de suba/descida ou nível alto/nível baixo
- IntEnable(): Ativação da interrupção sobre a porta base (porta J na prática em questão)
- IntPrioritySet(): É configurado o nível de prioridade da execução da interrupção
- IntMasterEnable(): Habilitação da “chave geral” das interrupções
- GPIOIntRegister(): Indica a rotina de tratamento de interrupção do GPIO em questão.

Para trabalhar com o temporizador, utilizando o temporizador padrão, foi primeiramente identificado as funções para configuração do mesmo, em que foram utilizadas as seguintes funções:

- SysCtlPeripheralEnable(): Habilita o periférico de timer (o qual foi utilizado o TIMER 0.
- SysCtlPeripheralReady(): Função a qual verifica se o TIMER colocado como parâmetro já está preparado.

- TimerConfigure(): Tal função configura como o timer será utilizado, tendo 4 modos. O qual pode ser contado de forma periódica, sem repetição, contador de pulsos, entre outras funcionalidades.
- TimerLoadSet(): Especifica a quantidade da contagem que o timer irá executar
- TimerIntRegister(): Tal função configura qual será a função executada quando o timer finalizar a contagem.
- TimerIntEnable(): Habilita a interrupção no timer desejado.
- TimerValueGet(): retorna a contagem restante para que o dê *timeout*.

Contudo, é possível implementar o temporizador através do SysTick, porém o mesmo é mais limitado no valor para contagem, sendo assim, foram identificadas as seguintes funções úteis para configuração e utilização do contador SysTick:

- SysTickPeriodSet(): Função utilizada para definir o período do contador SysTick
- SysTickEnable(): Função o qual inicia a contagem do contador SysTick
- SysTickValueGet(): Retorna a contagem atual do contador SysTick
- SysTickIntRegister(): Configura qual será a função a ser executada quando a interrupção for ativada.
- SysTickIntEnable(): Habilita a interrupção do contador SysTick

7 Projeto (design) da solução

Para fazer uso dos periféricos citados, as seguintes funções serão utilizadas:

GPIO:

GPIOIntEnable();

GPIOIntTypeSet();

IntEnable();

IntPrioritySet();

IntMasterEnable();

GPIOPinTypeGPIOOutput();

GPIOPinWrite();

Timer:

TimerIntRegister();

TimerIntEnable();

TimerIntClear();

TimerConfigure();

TimerLoadSet();
TimerEnable();

Função Utilizada para obter atraso:

SysCtlDelay();

Funções de inicialização dos periféricos:

SysCtlPeripheralEnable();
SysCtlPeripheralReady();

Assim, ao iniciar o programa, por intermédio da função SysCtlDelay() é possível obter um atraso por volta de 0.5 segundo antes de ligar o LED (por intermédio da função GPIOPinWrite()), respeitando assim o RF1.1. A partir desse momento, o timer é ativado para gerar uma interrupção após 3 segundos. Neste ponto, duas possíveis interrupções desencadeiam o encerramento do programa: Uma configurada pelo GPIO relacionado com a chave de usuário e outra com o encerramento do timer. Ambas imprimem mensagem na tela informando o encerramento do programa, no caso da interrupção ocasionada pela chave de usuário, o tempo e o número de clocks também é incluído.

O diagrama de sequência relativo ao laboratório está ilustrado na Figura 1.

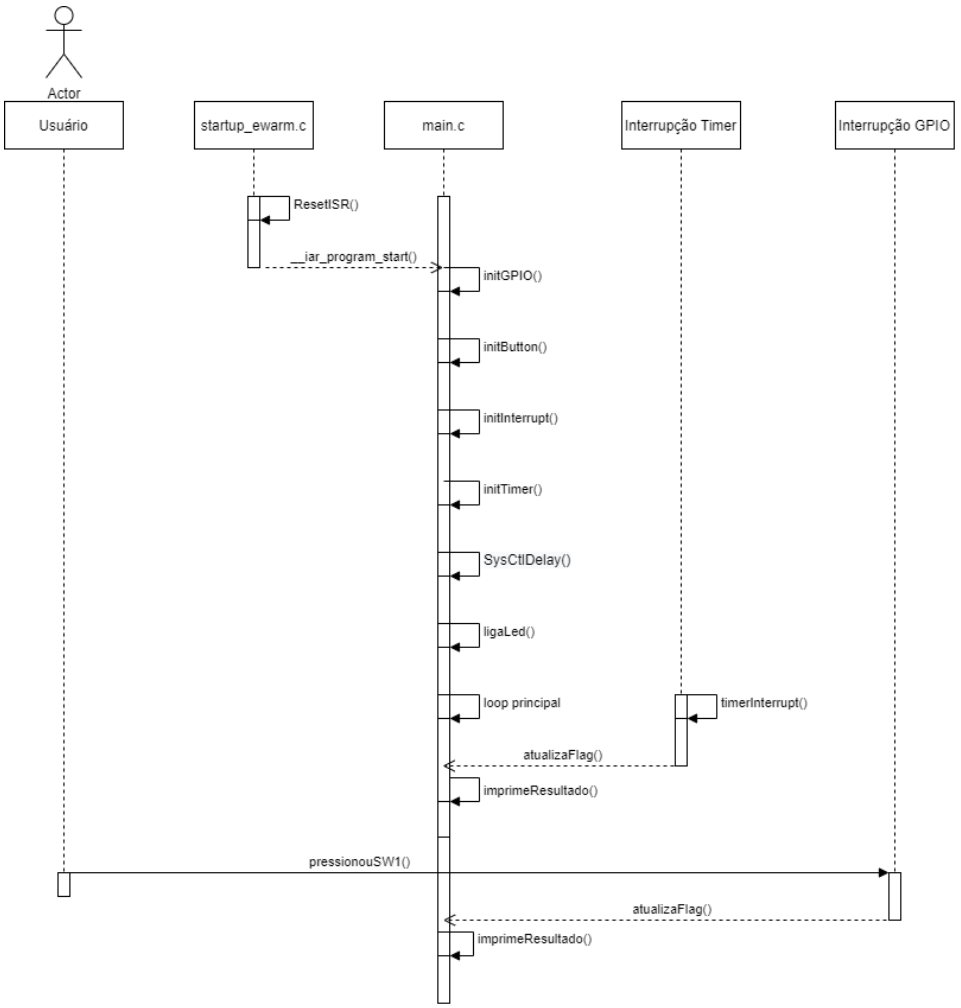

Figura 1: Diagrama de sequência.

8 Configuração do projeto na IDE (IAR).

Não foi necessário mudanças nas configurações com os padrões já existentes para execução deste laboratório.

9 Teste e depuração.

Basicamente, dois tipos de testes podem ser feitos: 1- esperar o término do programa por intermédio da interrupção gerada com o timer, podendo cronometrar o tempo para certificar se o tempo do timer está coerente com 3 segundos; 2- Antes da interrupção do timer ocorrer, pressionar a chave de usuário e certificar que o programa é encerrado imprimindo corretamente as informações na tela.

A frequência do clock pode ser verificada por intermédio da função do timer TimerValueGet(), que retorna o número de clocks contados pelo timer no momento. Assim, no momento da chamada da rotina de tratamento de interrupção do timer, o valor deve ser de 3*frequencia_clock. Assim, basta imprimir o resultado e certifica-lo.