# Trabalho Prático 2

**Semar Augusto da Cunha Mello Martins**

**1- Introdução:**

O trabalho prático 2 teve o objetivo de apresentar os alunos ao MSP430 e nos ensinar, na prática, como funcionam interrupções. O trabalho foi dividido entre as partes A e B.

A parte A consistia na implementação de dois programas, um que faz o MSP430 acender e apagar o led vermelho e outro que simulava o funcionamento de um semáforo, sendo que a luz amarela era simulada a partir do acendimento dos dois leds.

A parte B é a parte mais interessante do trabalho. Ela se consiste na implementação de um programa que pisca um dos leds a cada 2 segundos e pisca o outro a cada 10 segundos. Além disso, apertar o botão inverte qual led pisca de 2 em 2 segundos com o outro.

**2. Solução Proposta:**

Foram implementados dois programas para a solução da parte A.

**A.1** - Foi usada uma interrupção usando o TimerA, a cada segundo ocorreria a interrupção (8us * 62500 * 2 = 1s) sendo que tinha uma variável global que decidia se o led deveria estar aceso ou apagado.

**A.2 -** Foram usadas duas interrupções, uma do timer e outra do botão. O timer usava uma variável global contadora que decidia quais leds deveriam ficar acesos ou apagados. O botão, ao ser apertado, acelerava a variável contador caso ela estivesse em valores baixos para diminuir o tempo necessário para chegar ao sinal vermelho.

**B -** Para a solução da parte B foi usado o template distribuído aos alunos na versão para linux. Foram usadas duas tarefas somente e duas variáveis globais. Uma contadora para saber quanto tempo havia se passado e outra para decidir qual dos leds estava piscando mais rapidamente.

**3. Funções:**

**A.1 - main -** Seta os pinos para aceitar interrupção, seta as variáveis do timer A para que ele interrompa de 1 em 1segundo.

**- ParteA_1 -** usa da variável global para acender caso ela esteja em 0 e ficar apagada caso esteja em 1 ou 2.

**A.2 - main -** Seta os pinos de interrupção, habilita interrupção do botão e do timer A e acerta as variáveis relativas ao timer A para interromper de 1 em 1 segundo.

- **ParteA_2_timer -** Usa a variável global para acender o verde em tempos menores que 8, ambos os leis, em tempos iguais a 8 e somente o vermelho por tempos entre 8 e 10.

**- ParteA_2_botao -** Caso o timer estiver muito baixo (abaixo de 5), iguala ele a 5 para que em 3 segundo o sinal feche. Caso contrário, não faz nada.

**B** **- task1 -** caso time tenha um valor par, então pisque a luz que deve ser piscada rapidamente (depende do inversor ter sido setado ou não). Caso contrário, desligue os leds caso estejam ligados.

**- task2 -** Caso time seja igual a 9, então acenda a luz que deve ser piscada devagar (também depende do inversor) e caso contrário somente desligue os leds caso estejam ligados.

**- main -** cria a pilha para as duas tasks, seta os bits para os leds poderem ligar, para o botão ser aceito como interrupção, seta os valores das variáveis do timer para que ele interrompa a cada segundo

**- Timer_A** - salva o contexto atual, carrega o ponteiro para a pilha, atualiza qual task será usada, atualiza a variável time, salva o contexto e restaura ele.

**- Botão -** alterna os valores do inversor.

**3. Implementação parte B:**

Para a parte B foi distribuído um template para linux com boa parte do código pronto. Foi necessário completar as macros SAVE_CONTEXT() e RESTORE_CONTEXT() colocando os registradores crescente para SAVE_CONTEXT() e decrescente para RESTORE_CONTEXT() uma vez que a primeira está empilhando e a outra está desempilhando.

Foram criadas duas variáveis, uma chamada "time" e outra chama “inversor”. A variável time foi usada para contar quanto tempo se passou desde o último blink da luz lenta. Ela conta de 0 a 11. O inversor decide qual luz pisca rapidamente, com a luz vermelha sendo a padrão.

Com relação ao escalonador, ficou definido que a task1 rodaria nos segundos ímpares e a task2 rodaria nos segundos pares. O funcionamento do launchpad será da seguinte maneira:

Seja R a luz que pisca rapidamente (a cada 2 segundos), L a luz que pisca lentamente (a cada 10 segundos) e D ambas as luzes desligadas.

case(time) {
0: D
1: R
2: D
3: R
4: D
5: R
6: D
7: R
8: D
9: R
10: L
11: R
}

Como (11+1)mod12 == 0, então recomeçariamos do 0 de novo assim que a luz lenta piscasse.

Foi decidido fazer dessa maneira pois não foi definido se ambas as luzes precisavam piscar juntas no 10º segundo, então foi decidido que nos segundos de valor ímpar, a luz rápida piscaria e no 10º segundo, o segundo de valor 10 no contador, a luz lenta piscaria sozinha.

**Compilação e Execução:**

Para compilar os programas é necessário rodar os comandos

```
msp430-gcc -Os -mmcu=msp430g2553 tp2a1.c -o tp2a1.elf
msp430-gcc -Os -mmcu=msp430g2553 tp2a2.c -o tp2a2.elf
msp430-gcc -Os -mmcu=msp430g2553 tp2b.c -o tp2b.elf
```

E para a execução é necessário usar

```
mspdebug rf2500
prog exemplo.elf // deve ser usado algum dos .elf gerados
run
```

Os programas foram testados em uma máquina virtual usando Manjaro Linux.

**Conclusão:**

A implementação de programas usando o MSP430 é bastante complexa até se entender direito como ele funciona. Após isso, percebe-se que a implementação de programas com tarefa única na verdade são bastante simples. Contudo, quando é necessário o uso de um escalonador, essa tarefa se torna consideravelmente mais complexa e trabalhosa.