EC45C - Sistemas Microcontrolados
Turmas: C51A/C51B

Prática L5 – Interrupções e display de 7 segmentos.

Objetivos:

- Criar um *firmware* com foco em tratamento de interrupção.
- Simular o *firmware* no Proteus.
- Testar o firmware no kit didático XM118.
- Agrupar os arquivos gerados para envio pelo Moodle.

Interrupção

Interrupção é um dos recursos mais poderosos em aplicações embarcadas. Quase todas as aplicações em tempo real são implementadas usando Interrupções.

Como o próprio nome sugere, interrupções são eventos especiais que requerem atenção imediata, ele interrompe uma tarefa em execução no microcontrolador e passa a executar uma tarefa especial conhecida como rotina de tratamento de interrupção (*Interrupt Service Routine* - ISR) ou *interrupt handler* (manipulador de interrupção), voltando a executar o programa principal ao final da ISR.

Suponha que você está em casa, lavando louça. De repente, o telefone toca. Você para de lavar a louça e atende a chamada. Quando você tiver terminado a conversa, você encerra a chamada e volta a lavar a louça. Este processo é semelhante ao que acontece com um microcontrolador quando ele recebe uma interrupção. Podemos perceber que neste exemplo o programa principal é lavar a louça, o toque do telefone é a interrupção (que solicita que você pare o que está fazendo e atenda à chamada) e a conversa pelo telefone é a ISR.

No microcontrolador a interrupção é uma chamada à ISR, com a característica especial de que essa chamada é realizada via *hardware* e não pelo código do programa. Por exemplo: considerando uma porta de entrada, cada vez que houver uma transição de subida ou de descida no valor lógico lido por esta porta, o PIC18F4550 automaticamente interrompe a execução do programa ativo e gera uma chamada à função situada no endereço 0x0008 ou no endereço 0x0018.

Resumindo, a interrupção executa a seguinte sequência:

- Conclui a execução da instrução de máquina que estiver sendo executada naquele instante;
- Incrementa o contador de programa para apontar para a próxima instrução;
- Salvar esse endereço na pilha;
- Carregar o endereço 0x0008 no contador de programa;
- Executar a rotina de interrupção.

Quando a interrupção é efetuada, o hardware momentaneamente inibe novas interrupções, zerando o bit **GIE** (Global Interrupt Enable) do registrador **INTCON**.

Ao final da rotina de interrupção é executada uma instrução especial, em linguagem de máquina, que retira da pilha o endereço da próxima instrução do programa principal, o transfere para o contador de programa e habilita novas interrupções escrevendo 1 no bit **GIE**. Desta forma, o programa que foi interrompido continua a ser executado exatamente de onde parou.

O PIC18F4550 possui várias fontes de interrupção, mostradas na Figura 1.

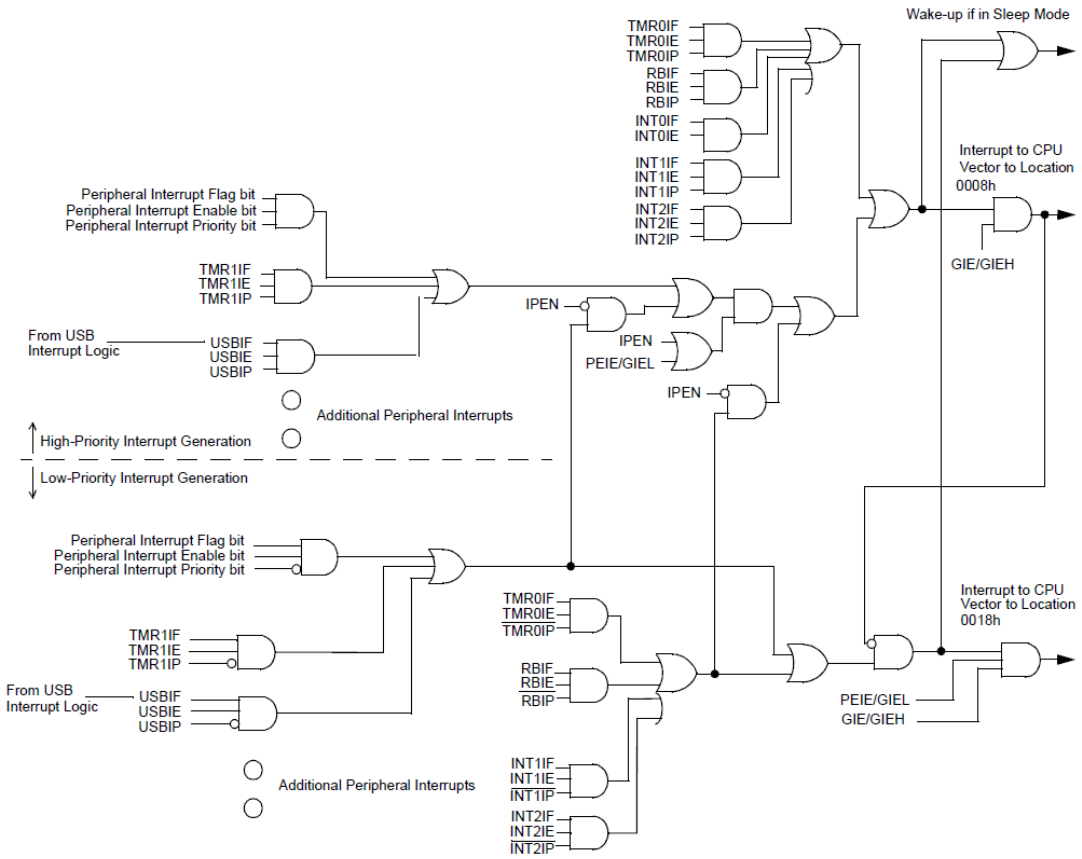


Figura 1: *Lógica de interrupções do* ***PIC18F4550****.*

Cada periférico do microcontrolador tem seu bit de pedido de interrupção (*Interrupt Flag* - IF), seu bit de habilitação de interrupção (*Interrupt Enable* - IE), e seu bit de prioridade da interrupção (*Interrupt Priority* - IP).

O principal registrador que controla as interrupções é o registrador **INTCON**. O bit **GIE**, do registrador **INTCON**, habilita o sistema interrupções. É como se ele fosse uma chave geral das interrupções. Para entender melhor, podemos fazer uma analogia com o circuito elétrico residencial. Para acender uma lâmpada temos que ligar o interruptor que controla esta lâmpada. Mas se a chave geral não estiver ligada, não há corrente elétrica para a lâmpada e ela não irá acender independente do estado do interruptor. No nosso caso, o **INTCONbits.GIE** é o bit equivalente a chave geral enquanto o **INTCONbits.INT0IE** é o bit equivalente ao interruptor que habilita a interrupção externa INT0.

Por padrão as interrupções externas (INT0, INT1 e INT2) são acionadas quando há uma transição de descida, ou seja, quando o nível lógico vai de 1 para 0. Para modificar o tipo da borda da transição da interrupção deve-se modificar o bit **INTEDGx** (onde x é o número da porta) do registrador **INTCON2**.

- **INTCON2bits.INTEDGx** = 0 a interrupção se dá em borda de descida.
- **INTCON2bits.INTEDGx** = 1 a interrupção se dá em borda de subida.

Uma boa prática para o uso de interrupções é escrever a ISR da forma mais enxuta possível. Isto permite que o microcontrolador execute a ISR brevemente e volte a executar a rotina principal de programa afetando minimamente o desempenho do sistema.

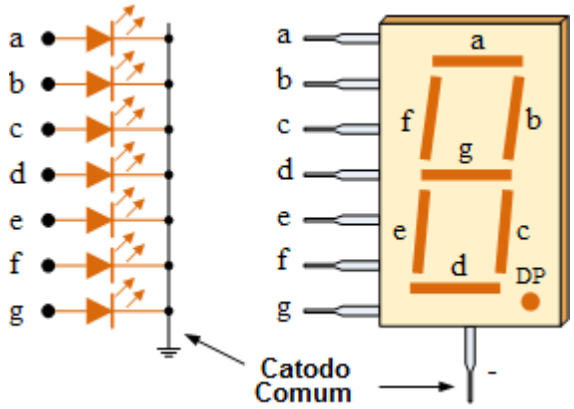
Além do registrador **INTCON**, outros registradores são usados para configurar e controlar as interrupções. Neste ponto, destacamos os registradores **INTCON2 e INTCON3**. Enquanto os bits de habilitação da interrupção e a *flag* de interrupção da porta INT0 estão no registrador **INTCON**, os bits de habilitação da interrupção e a *flag* de interrupção das portas INT1 e INT2 estão no registrador **INTCON3**. Já o registrador **INTCON2** possui os bits que permitem mudar o tipo da transição que ativa a interrupção (bodas de subida ou descida).

Para mais informações ver página 99 do *datasheet* do PIC18F4550.

Display de 7 segmentos.

Os displays de 7 segmentos são displays formados por conjuntos de LEDs dispostos de forma a apresentar a figura de um digito decimal quando ligados ordenadamente.

Para simplificar as conexões, os anodos ou catodos de todos os LEDs são conectados a um terminal comum do display, caracterizando o display como catodo comum ou anodo comum, respectivamente.



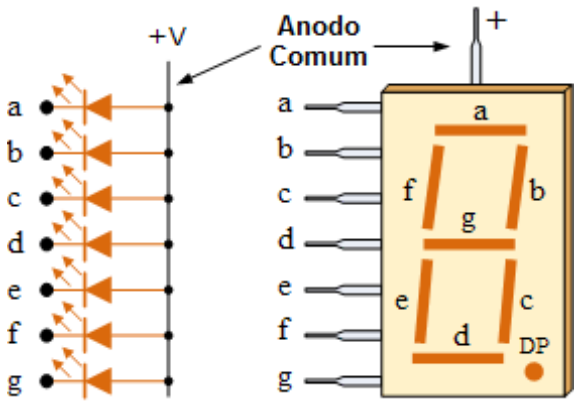


Figura 1a: Display Catodo Comum

Figura 1b: Display Anodo Comum

A exibição dos dígitos é feita ao acender determinados segmentos enquanto outros são mantidos apagados. A figuras 2 apresenta a ligação dos LEDs relacionando a posição de cada um de seus segmentos.

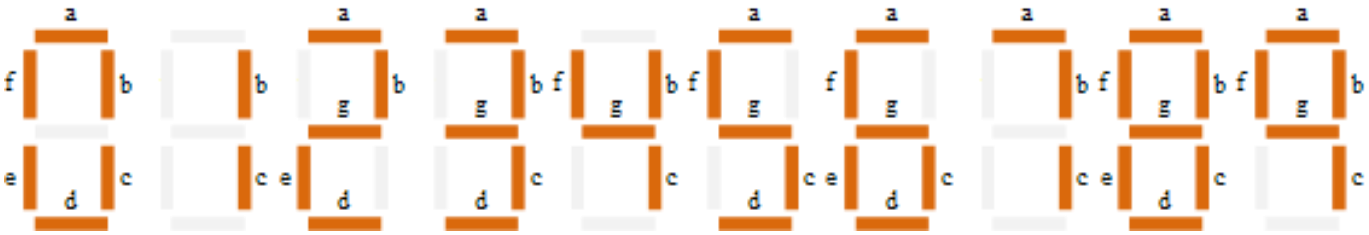


Figura 2: Dígitos criados com segmentos de LEDs.

ATIVIDADES

1- *Download* do circuito de simulação.

Acesse a plataforma Moodle e faça o *download* do circuito L5.pdsprj.
Salve o arquivo no seguinte diretório de trabalho:
{Disciplina}_{Turma}_{Grupo}_{Número do roteiro}
Ex: C:\EC45C_C51A_B1_L5

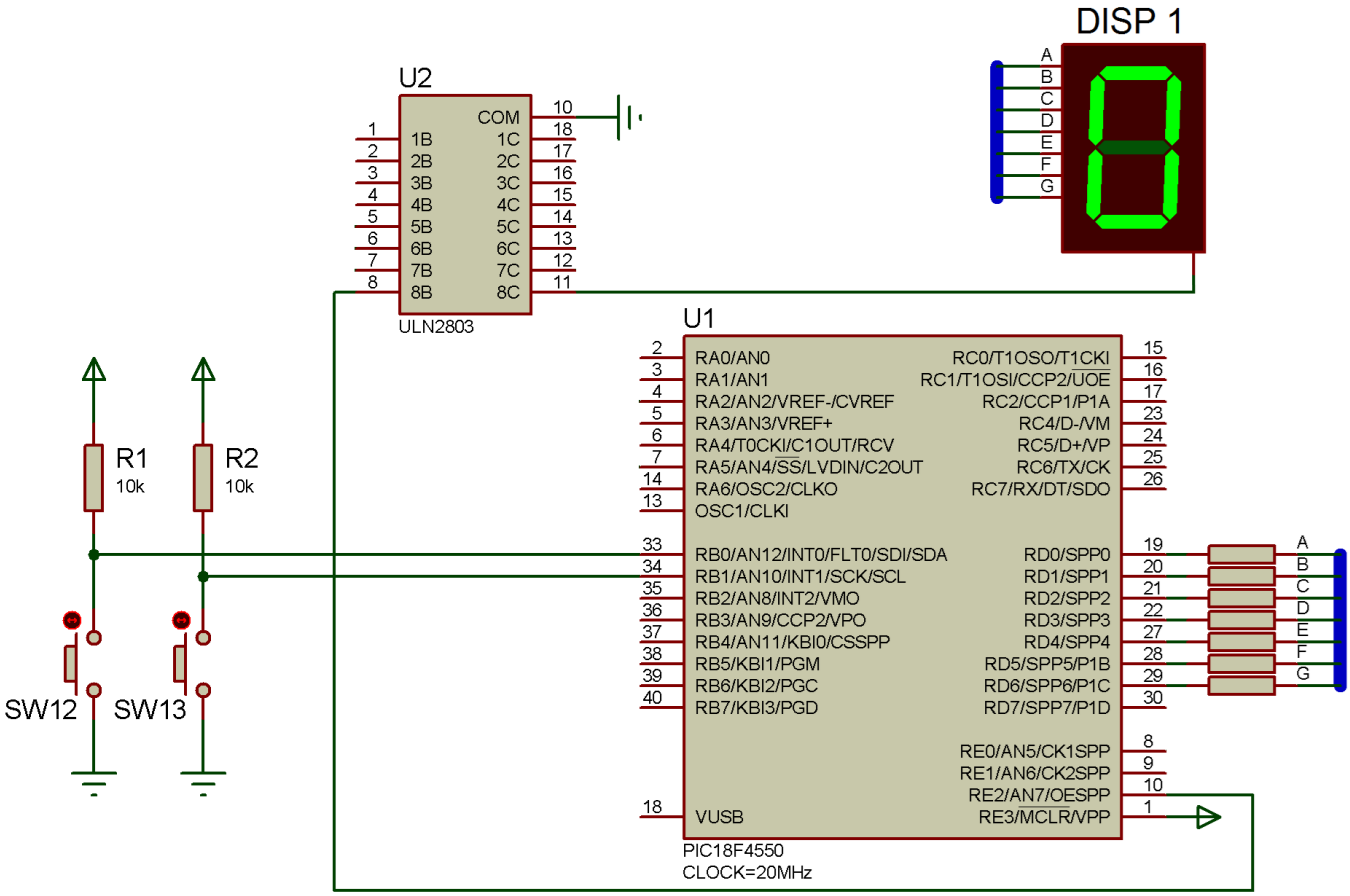


Figura 2: *Circuito da atividade L5.*

2- Desenvolvimento do *firmware*.

Crie um novo projeto no MPLAP X denominado L5 e salve-o no diretório de trabalho criado na atividade anterior.

Desenvolva um *firmware* capaz de acionar os LEDs da seguinte forma:

- Enquanto o microcontrolador estiver processando o loop infinito, o display de 7 segmentos deve apresentar o valor 0.
- Quando ocorrer uma interrupção de baixa prioridade o display de 7 segmentos deve apresentar o valor 1 por 1 segundo e sair da rotina de tratamento de interrupção após este período.
- Quando ocorrer uma interrupção de alta prioridade, display de 7 segmentos deve apresentar o valor 2 durante 1 segundo e sair da rotina de tratamento de interrupção após este período.

A interrupção de baixa prioridade será a interrupção INT1 e a interrupção de alta prioridade será a interrupção INT0.

OBS: É **extremamente** recomendado que não se use rotinas de atraso dentro de uma rotina de tratamento de interrupção. MAS, é aceitável neste estágio do curso.

Utilize o seguinte código para iniciar seu projeto:

```c
#include <xc.h>

// Configurações
#pragma config PLLDIV = 5          // PLL para 20MHz
#pragma config CPUDIV = OSC1_PLL2  // PLL desligado
#pragma config FOSC = HS           // Fosc = 20MHz; Tcy = 200ns
#pragma config WDT = OFF           // Watchdog timer desativado
#pragma config PBADEN = OFF        // Pinos do PORTB começam como digitais
#pragma config LVP = OFF           // Desabilita gravação em baixa tensão
#pragma config DEBUG = ON          // Habilita debug
#pragma config MCLRE = ON          // Habilita MCLR e desabilita RE3 como I/O

#define _XTAL_FREQ 20000000        // uC opera com cristal de 20 MHz

// Funções das ISRs
void interrupt HighPriorityISR(void)
{
}

void interrupt low_priority LowPriorityISR(void)
{
}

// Função Principal
void main(void)
{
}
```

3- Simulação do *firmware* gerado no Proteus.

Simule o *firware* no Proteus e apresente o funcionamento ao professor.

4- Gravação e execução do código de máquina no microcontrolador

Grave o código de máquina no microcontrolador e apresente o funcionamento para o professor.

5- Envio dos resultados para plataforma Moodle.

Compacte o diretório de trabalho com o projeto do *firmware* L5 em um arquivo .zip.

Nomeie o arquivo obedecendo o seguinte formato:
{Disciplina}_{Turma}_{Bancada}_{Número do roteiro}.zip
ex: EC45C_C51A_G1_L5.zip

Envie o arquivo compactado acessando a atividade “L5”.