Universidade Federal do Sul e Sudeste do Pará
Faculdade de Computação e Engenharia Elétrica
Curso de Engenharia da Computação
Iago Costa das Flores

Microprocessadores e Microcontroladores
Experimento 10

Marabá
2021

Iago Costa das Flores

Microprocessadores e Microcontroladores
Experimento 10

Relatório apresentado no curso de Engenharia da Computação, turma de 2018 como obtenção de nota parcial na disciplina de microprocessadores e microcontroladores, ministrada pelo Professor Dr. Elton Alves.

Marabá
2021

UNIFESSPA

Sumário

1 - Introdução **4**

2 - Atividades **4**

2.1 - Desenvolva uma simulação que permita um LED ficar aceso enquanto o TIMER0 ficar rodando. 4

3 - Conclusão **8**

4 - Referências **8**

1 - Introdução

O Trabalho visa apresentar os códigos fontes e resultados de execução da atividade avaliativa a seguir:

> 1 - Desenvolver um código em C, utilizando o software Mplab, para: Inserir um botão, para acionar o semáforo; O semáforo será controlado pelo usuário; Após ser acionado, o sinal passará para a condição fechado (led vermelho) em seguida se iniciará contagem de tempo regressivo de 9 até 0; Em seguida o sinal será aberto.

2 - Atividades

As atividades demonstradas a seguir foram feitas com a ajuda do programa mplab e proteus para escrever e executar os códigos em assembly.

2.1 - Desenvolver um código em C, utilizando o software Mplab, para: Inserir um botão, para acionar o semáforo; O semáforo será controlado pelo usuário; Após ser acionado, o sinal passará para a condição fechado (led vermelho) em seguida se iniciará contagem de tempo regressivo de 9 até 0; Em seguida o sinal será aberto.

Código da atividade 01:

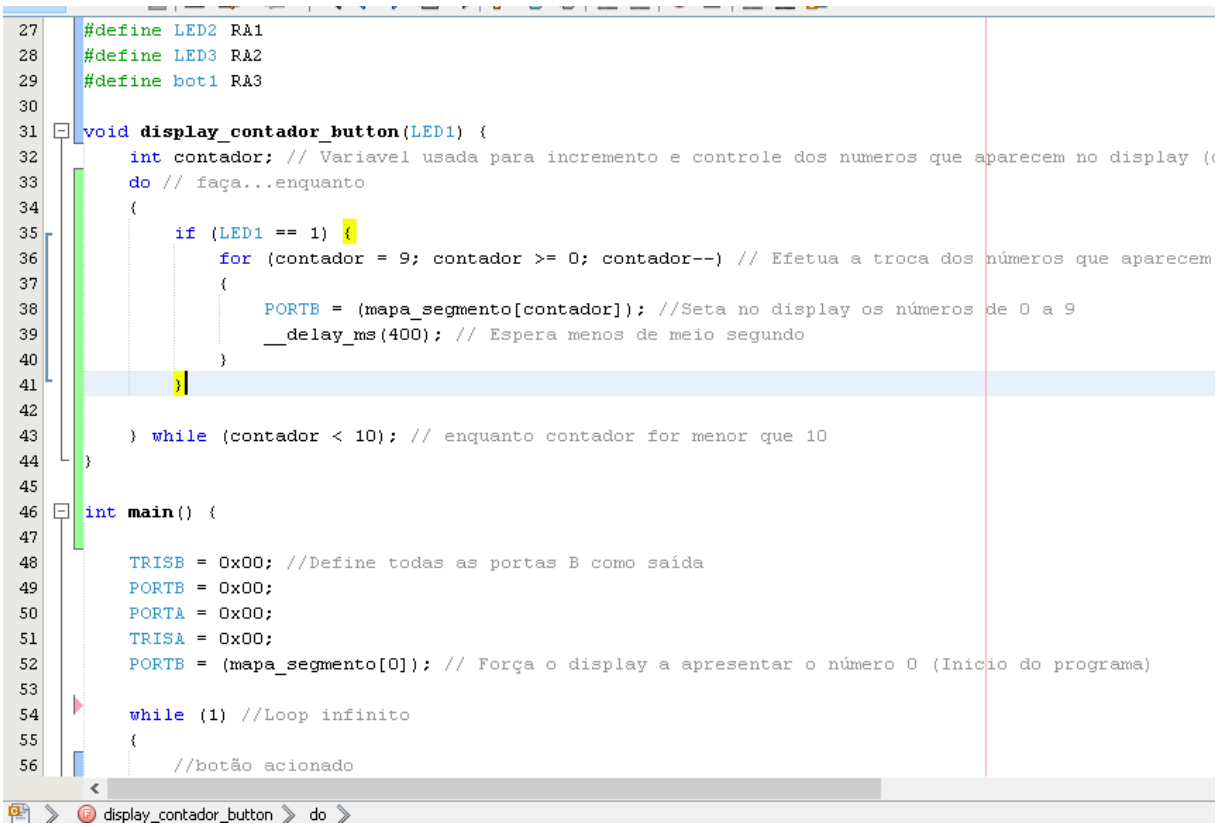
```
//
// * File: semaforo_display.c
// * Author: Iagof
// *
// * Created on July 28, 2021, 11:16 AM
// *

#include<xc.h>

#pragma config FOSC = INTOSCIO // Oscillator Selection bits (HS
oscillator: High-speed crystal/resonator on RA6/OSC2/CLKOUT and
RA7/OSC1/CLKIN)
```

```
#pragma config WDTE = OFF          // Watchdog Timer Enable bit (WDT
disabled)
#pragma config PWRTE = OFF        // Power-up Timer Enable bit (PWRT
disabled)
#pragma config MCLRE = OFF     // RA5/MCLR/VPP Pin Function Select bit
//(RA5/MCLR/VPP pin function is MCLR)
#pragma config BOREN = OFF        // Brown-out Detect Enable bit (BOD
disabled)
#pragma config LVP = OFF         // Low-Voltage Programming Enable bit
//(RB4/PGM pin has digital I/O function, HV on MCLR must be used for
//programming)
#pragma config CPD = OFF         // Data EE Memory Code Protection bit
(Data memory code protection off)
#pragma config CP = OFF

//Define a utilização do clock interno de 4 Mhz
#define _XTAL_FREQ 4000000

unsigned char const mapa_segmento[10] = {0x3F, 0x06, 0x5B, 0x4F, 0x66,
0x6D, 0x7D, 0x07, 0x7F, 0x6F};
//Variaveis que representam os números.

#define LED1 RA0
#define LED2 RA1
#define LED3 RA2
#define bot1 RA3

void display_contador_button(LED1) {
      int contador; // Variavel usada para incremento e controle dos
numeros que aparecem no display (de 0 a 9)
    do // faça...enquanto
    {
        if (LED1 == 1) {
              for (contador = 9; contador >= 0; contador--) // Efetua a
troca dos números que aparecem no display
            {
                 PORTB = (mapa_segmento[contador]); //Seta no display os
números de 0 a 9
                __delay_ms(400); // Espera menos de meio segundo
            }
        }

    } while (contador < 10); // enquanto contador for menor que 10
```

```
}

int main() {

    TRISB = 0x00; //Define todas as portas B como saída
    PORTB = 0x00;
    PORTA = 0x00;
    TRISA = 0x00;
      PORTB = (mapa_segmento[0]); // Força o display a apresentar o
número 0 (Inicio do programa)

    while (1) //Loop infinito
    {
        //botão acionado
        if (bot1 == 0) {
            __delay_ms(100);
            if (bot1 == 0) {
                LED1 = 1;
                LED2 = 0;
                LED3 = 0;
                display_contador_button(LED1);
                //                __delay_ms(3000);
            }
            bot1 = 1;
        } else {
            // display_contador_normal(LED1);
              int contador; // Variavel usada para incremento e controle
dos numeros que aparecem no display (de 0 a 9)
            do // faça...enquanto
            {
                  if (bot1 == 0) // detectar o click no botão e parar o
else para voltar ao if
                    break;
                __delay_ms(500); // Espera 4 segundos
                      PORTB = (mapa_segmento[9]); // Força o display a
apresentar o número 0 (Fim do ciclo e inicio de outro)
                LED1 = 0; // Sinal Vermelho apagado
                LED2 = 0; // Sinal Amarelo apagado
                LED3 = 1; // Sinal Verde do acesso
                __delay_ms(2000); // Espera 2 segundos
                  if (bot1 == 0) // detectar o click no botão e parar o
else para voltar ao if
                    break;
```

```
                LED1 = 0; // Sinal Vermelho apagado
                LED2 = 1; // Sinal Amarelo acesso
                LED3 = 0; // Sinal Verde do apagado
                  if (bot1 == 0) // detectar o click no botão e parar o
else para voltar ao if
                    break;
                __delay_ms(4000); // Espera 4 segundos
                  if (bot1 == 0) // detectar o click no botão e parar o
else para voltar ao if
                    break;
                LED1 = 1; // Sinal Vermelho acesso
                LED2 = 0; // Sinal Amarelo apagado
                LED3 = 0; // Sinal Verde do apagado
                if (LED1 == 1) {
                       for (contador = 9; contador >= 0; contador--) //
Efetua a troca dos números que aparecem no display
                    {
                            PORTB = (mapa_segmento[contador]); //Seta no
display os números de 0 a 9
                                __delay_ms(400); // Espera menos de meio
segundo
                    }
                }
            } while (contador < 10); // enquanto contador for menor que
10
        }
    }
    return 0;
}
```

Na figura 01 é possível parte do código fonte da atividade 01.



```
#define LED2 RA1
#define LED3 RA2
#define bot1 RA3

void display_contador_button(LED1) {
    int contador; // Variavel usada para incremento e controle dos numeros que aparecem no display (
    do // faça...enquanto
    {
        if (LED1 == 1) {
            for (contador = 9; contador >= 0; contador--) // Efetua a troca dos números que aparecem
            {
                PORTB = (mapa_segmento[contador]); //Seta no display os números de 0 a 9
                __delay_ms(400); // Espera menos de meio segundo
            }
        }

    } while (contador < 10); // enquanto contador for menor que 10
}

int main() {

    TRISB = 0x00; //Define todas as portas B como saída
    PORTB = 0x00;
    PORTA = 0x00;
    TRISA = 0x00;
    PORTB = (mapa_segmento[0]); // Força o display a apresentar o número 0 (Inicio do programa)

    while (1) //Loop infinito
    {
        //botão acionado
```

display_contador_button > do

Figura 01: Código fonte da atividade 01

Na figura 02, temos a execução da atividade 01, com o esquema montado com o PIC16F628A.

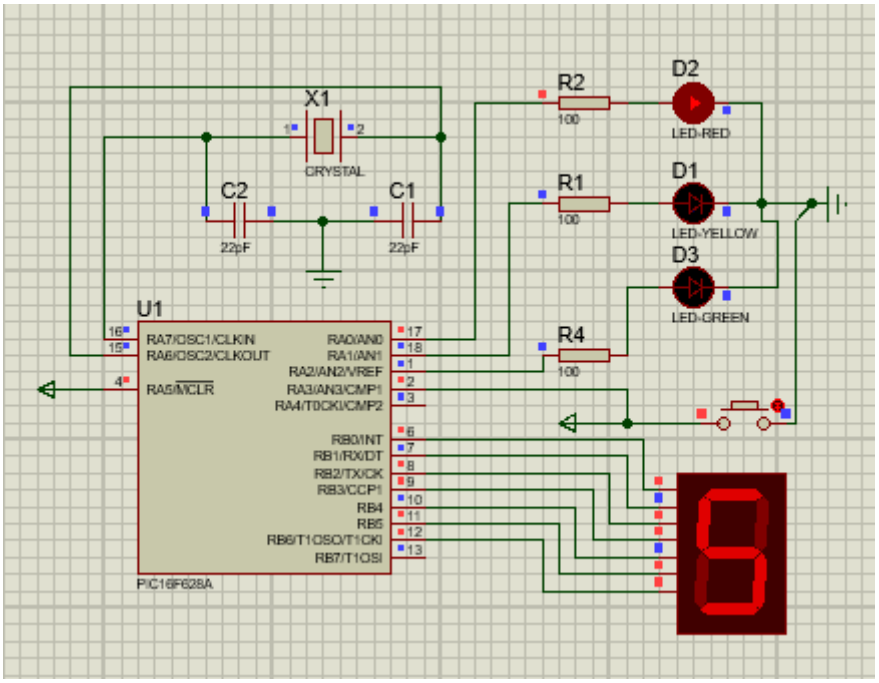


Figura 02: Execução da atividade 01

3 - Conclusão

Foram demonstradas as impressões do código e execução dos mesmos através das imagens apresentadas com suas devidas explicações. Os códigos fontes estão comentados e a atividade foi feita conforme o solicitado no comando do trabalho.

4 - Referências

José, D. S. **Desbravando o PIC - Ampliado e atualizado para PIC16F628A.** 7ª ed. Ed.Érica, 2003.