Bruno Henrique de Carvalho Scaglione - 10335812

PMR 3406
Microprocessadores em Automação e Robótica
Implementação de Teclado Numérico com display 7 segmentos no PIC

Brasil

5 de julho de 2020

1 Resumo

Foi feita uma implementação de teclado/ 4 displays de 7 segmentos , na qual os dígitos teclados pelo administrador são colocados no display mais à esquerda e os outros são 'shiftados' uma posição para direita. Um exemplo seria um consultório médico, que distribui senhas de 1 a 9 para os clientes (durante a pandemia só podem estar 9 clientes por vez na sala). O número do display mais à direita corresponde a próxima senha que será chamada, e quando for chamada, as outras senhas são deslocadas e entra uma nova senha no display.

É importante ressaltar que devido aos 10ms de interrupção impostos, e o tempo que demora uma 'clickada' no simulador, as vezes o número acaba sendo lido 2 vezes seguidas, tem que ser bem sutil a 'clickada'.

2 Código

main.c

```
/*
 * File:   main.c
 * Author: Jun Okamoto Jr.
 *
 * Created on April 18, 2020, 12:58 PM
 */


// PIC16F886 Configuration Bit Settings

// 'C' source line config statements

// CONFIG1
#pragma config FOSC = EC        // Oscillator Selection bits (EC: I/O function on RA6/OSC2/CLKOUT pin, CLKIN on RA7/OSC1/C
#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT disabled and can be enabled by SWDTEN bit of the WDTCON
#pragma config PWRTE = OFF      // Power-up Timer Enable bit (PWRT disabled)
#pragma config MCLRE = ON       // RE3/MCLR pin function select bit (RE3/MCLR pin function is MCLR)
#pragma config CP = OFF         // Code Protection bit (Program memory code protection is disabled)
#pragma config CPD = OFF        // Data Code Protection bit (Data memory code protection is disabled)
#pragma config BOREN = ON       // Brown Out Reset Selection bits (BOR enabled)
#pragma config IESO = ON        // Internal External Switchover bit (Internal/External Switchover mode is enabled)
#pragma config FCMEN = ON       // Fail-Safe Clock Monitor Enabled bit (Fail-Safe Clock Monitor is enabled)
#pragma config LVP = OFF        // Low Voltage Programming Enable bit (RB3 pin has digital I/O, HV on MCLR must be used fo

// CONFIG2
#pragma config BOR4V = BOR40V   // Brown-out Reset Selection bit (Brown-out Reset set to 4.0V)
#pragma config WRT = OFF        // Flash Program Memory Self Write Enable bits (Write protection off)

// #pragma config statements should precede project file includes.
// Use project enums instead of #define for ON and OFF.

#include <xc.h>
#include <stdio.h>
#include <stdlib.h>
#include "always.h"
#include "delay.h"


//Variaveis Globais
//volatile unsigned int ADC_result;
volatile char key;

////////////// Teclado Numerico ///////////////

void keypad_init(void){
    //carrega EEPROM com tabela da verdade
    // lemnrando que  aporta C7 nao é utilizada e sempre vale 0
    // os bits estao na sequencia: C7|  (colunas) -> C6C5C4|C3C2C1C0 <-(linhas)
```

```
    eeprom_write(0x3e, '1'); // 1 colocar (end,valor)
    eeprom_write(0x5e, '2'); // 2 colocar (end,valor)
    eeprom_write(0x6e, '3'); // 3 colocar (end,valor)
    eeprom_write(0x3d, '4'); // 4 colocar (end,valor)
    eeprom_write(0x5d, '5'); // 5 colocar (end,valor)
    eeprom_write(0x6d, '6'); // 6 colocar (end,valor)
    eeprom_write(0x3b, '7'); // 7 colocar (end,valor)
    eeprom_write(0x5b, '8'); // 8 colocar (end,valor)
    eeprom_write(0x6b, '9'); // 9 colocar (end,valor)
    eeprom_write(0x37, '*'); // * colocar (end,valor)
    eeprom_write(0x57, '0'); // 0 colocar (end,valor)
    eeprom_write(0x67, '#'); // # colocar (end,valor)
}


char keypad_read(void){
    // porta C4 sendo ignorada, coloco 1 nela
    TRISC = 0b11110000;
    char var_1 = PORTC;
    TRISC = 0b10001111;
    char var_2 = PORTC;
    char var_3 = (var_1 | var_2);
    return eeprom_read(var_3);
}

void interrupt isr(void) {
    // Função para tratamento de interrupções
    // local variables -> static

    // Tratamento da interrupção do Timer 0
    if (T0IE && T0IF) {
        // Interrupção do Timer 0 aqui
        //ADC_result = adc_read_0(); // valor guardado no ADC
        key = keypad_read(); // tecla do teclado numerico ou 'N' que representa nada teclado
        TMR0 = (0xff - 196 ) ; // valor inicial do Timer 0 - 10ms
        T0IF = 0; // limpa flag de interrupção
    }
  ////////////////////// OUTRA ATIVIDADE ////////////////////////////

  // Tratamento da interrupção do Port B
  //if (RBIE && RBIF) {

    //char portB = PORTB; // leitura do port B limpa interrupção

    //io_sw_read(portB);      // Necessário para usar a chave
    //debug_led_toggle(1);    // exemplo de uso do LED de debug

    //RBIF = 0; // limpa o flag de interrupção para poder atender nova
  //}

  // Tratamento de outras interrupções aqui

 ////////////////////// OUTRA ATIVIDADE ////////////////////////////
}
```

```c
void t0_init(void) {
    // Inicialização do Timer 0 aqui

    TOCS = 0; // clock interno FOSC/4
    PSA = 0; // prescalar p/timer0

    OPTION_REGbits.TOCS = 0; // usa clock interno FOSC /4
    OPTION_REGbits.PSA = 0; // prescaler é para o Timer 0 e não para o WDT
    OPTION_REGbits.PS = 7 ; // ajusta o Prescaler do Timer 0 (divide por 256)
    TMR0 = (0xff - 196 ) ; // valor inicial do Timer 0 a cada 10ms

    // Interrupções
    T0IE = 1; // habilita a interrupção do Timer 0

}


// conexao dos pinos:

// RB6 -> a
// RB5 -> b
// RB4 -> c
// RB3 -> d
// RB2 -> e
// RB1 -> f
// RB0 -> g


// Digito :

//     aaaaaaa
//     f     b
//     f     b
//     f ggg b
//     e     c
//     e     c
//     ddddddc

// pinos de chaveamento

// RC3 -> primeiro digito
// RC2 -> segundo digito
// RC1 -> terceiro digito
// RC0 -> quarto digito

// digitos -> PORTB

// 0 -> 01111110
// 1 -> 00110000
// 2 -> 01101101
// 3 -> 01111001
// 4 -> 00110011
// 5 -> 01011011
// 6 -> 01011111
// 7 -> 01110000
// 8 -> 01111111
```

```
// 9 -> 01110011

// Projeto para 20mA nos fios dos leds aproximadamente ( 20mA no maximo)

// Salvando na EEPROM
// deslocado para a direita , pois o primeiro endereço tava dando problema
__EEPROM_DATA(255, 0b01111110, 0b00110000, 0b01101101, 0b01111001, 0b00110011, 0b01011011, 0b01011111);  // addr: 0x00 ...
__EEPROM_DATA(0b01110000, 0b01111111, 0b01110011, 255, 255, 255, 255, 255);  // addr: 0x08 ... 0x0F


void display( short num) {
    PORTB = eeprom_read(num + 1); // esta deslocado pois o primeiro enderenco da eeprom nao estava funcionando
}

void main(void) {
    // Programa Principal

    //variáveis locais
    int key_int_1 = 0;
    int key_int_2 = 0;
    int key_int_3 = 0;
    int key_int_4 = 0;
    char current_key;

    // Inicializações

    //adc_init_0();        // inicializa conversor A/D
    t0_init();           // inicializa Timer 0
    //io_init();         // inicializa chave, LED e Buzzer
    //lcd_init();          // inicializa LCD
    //debug_init();        // inicializa LEDs para debug
    ei();                // macro do XC8, equivale a GIE = 1, habilita interrupções
    keypad_init();       // incicializa keypad

    // configuracao das portas

    ANS13, ANS11, ANS9, ANS8, ANS10, ANS12 = 0; // saídas digitais dos pinos do PORTB usados
    TRISB  &= 10000000; // configura os pinos RB6 à RB0 como saída
    PORTB  = 0; // leds desligados

    TRISA  &= 11110000;  // pinos como saída
    ANS0, ANS1, ANS2, ANS3 = 0; // digital
    PORTA = 0; // mosfets cortados


    while (1) {

        current_key = key;
        int value = (int)current_key - 48;
        if (current_key == '*' || current_key == '#' ) {
            // zera o display
            // pelo enunciado nao entendi a diferenca entre * e # ento implemenntei igual
            key_int_1 = 0;
            key_int_2 = 0;
            key_int_3 = 0;
            key_int_4 = 0;
```

```
        }
        else if((value > -1 && value < 10)){ // 0-9
            // coloca sempre o numero na primeira posicao do display
            // como se fosse um display de senha de um consultorio por ex
            // as senhas mais velhas vao ficando a direita, e apos a ultima senha da direita
            // ser chamada ela sai do display, todas as senhas sao deslocadas para direita
            // e entra outra senha no display
            key_int_4 = key_int_3 ;
            key_int_3 = key_int_2 ;
            key_int_2 = key_int_1 ;
            key_int_1 = current_key - 48;
        }

        // primeiro digito
        RA0 = 0;
        RA3 = 1;

        display(key_int_1);

        delay_ms(10);
        // segundo digito
        RA3 = 0;
        RA2 = 1;

        display(key_int_2);

        delay_ms(10);
        // terceiro digito
        RA2 = 0;
        RA1 = 1;

        display(key_int_3);

        delay_ms(10);
        // quarto digito
        RA1 = 0;
        RA0 = 1;

        display(key_int_4);

        delay_ms(10);

    }
}
```