```
#include <p18f4550.h>
#include <xc.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

/*Observacão: O PORT DO LCD E CONFIGURADO NO PORTD COMO OS PINOS DE DADOS,(
 ESTA LOCALIZADO PORTE, RS NO PORT RE0 E PINO EN NO PORT RE1 */
/*ESCREVE SEGUNDA LINHA comando_lcd(0xc0);, ESCREVE PRIMEIRA LINHA LINHA co
//%u valor inteiro
// PIC18F4550 Configuration Bit Settings

// CONFIG1L
#pragma config PLLDIV = 1       // PLL Prescaler Selection bits (No prescal

// CONFIG1H
#pragma config FOSC = HS        // Oscillator Selection bits (HS oscillator

// CONFIG2L
#pragma config PWRT = ON        // Power-up Timer Enable bit (PWRT enabled)

// CONFIG2H
#pragma config WDT = OFF        // Watchdog Timer Enable bit (WDT disabled

// CONFIG3H
#pragma config PBADEN = OFF      // PORTB A/D Enable bit (PORTB<4:0> pins a
#pragma config MCLRE = ON        // MCLR Pin Enable bit (MCLR pin enabled;

// CONFIG4L
#pragma config LVP = OFF         // Single-Supply ICSP Enable bit (Single-S

// CONFIG6H
#pragma config WRTC = OFF       // Configuration Register Write Protection
#pragma config WRTB = OFF       // Boot Block Write Protection bit (Boot bl
#pragma config WRTD = OFF       // Data EEPROM Write Protection bit (Data E

//DEFIINES
#define _XTAL_FREQ    20000000
#define LED_TEST      PORTEbits.RE2
#define SINAL_TENSAO   PORTAbits.RA1
#define SINAL_CORRENTE PORTAbits.RA1
//END DEFINE

//PROTOTIPOS FUNCOES
```

```
D4 a D7 no PINO D4 A D7 DO uC, E OS PINOS DE COMANDO

mando_lcd(0x80); printf("LUCAS %f",v); RECEBE TIPO DE VARIAVEL E IMPRIME NO



e (4 MHz oscillator input drives PLL directly))

 (HS))



(control is placed on the SWDTEN bit))

re configured as analog input channels on Reset)
RE3 input pin disabled)

upply ICSP enabled)

bit (Configuration registers (300000-3000FFh) are not write-protected)
ock (000000-0007FFh) is not write-protected)
EPROM is not write-protected)
```

```
LCD*/
```

```
void pic_start();
void lcd_iniciar();                     //PROTOTIPO PARA INICIAR LCD
void comando_lcd(char comandos);        //PROTOTIPO PARA COMANDO DO LCD
void lcd_dado(char dados);              //PROTOTIPO PARA DADOS DO LCD
void putch(char escrita);               //PROTOTIPO PADRÃO C PARA CRIAR MAI
void timer0_config();
float timer0_carga();
//END PROT


void interrupt my_isr_high(void)                // Interrupção de Alta Prio
{
}

void interrupt low_priority my_isr_low(void)    // Interrupção de Baixa Pri
{
}


void main()
{
    pic_start();        // INICIA CONFIG DO PIC
    lcd_iniciar();      // INICIA LCD
    timer0_config();    // INICIA CONFIG DO TIMER0

    float x = 0;        // VARIAVEL GLOBAL

    printf("FATOR POTENCIA:");      //ESCREVE NO LCD



    while(1)
   {

        while(SINAL_TENSAO && SINAL_CORRENTE == 1)
        {
         T0CONbits.TMR0ON = 1;       // TIMER0 LIGADO
        }

        x = timer0_carga();           // DESLIGA O TIMER0 E FAZ A CONTA PAR

        comando_lcd(0xc0);
        printf("%fms",x);
```

```
S DE UM CARACTER NO LCD




ridade


oridade


















A MS
```

```
   }

}

void lcd_iniciar()
{
    ///////////////////////////////////////////////////////////////////////
    //CONFIG INICIAL
    TRISD = 0x00;               //TODOS PORTD COMO SAIDA
    ///////////////////////////////////////////////////////////////////////
    //INICIALIZACÃO
    __delay_ms(15);        //AGUARDA 15ms PARA INICAR O LCD

    comando_lcd(0x30);
    __delay_ms(5);    //AGUARDA 5ms
    comando_lcd(0x30);
    __delay_us(100);    //AGUARDA 100us
    comando_lcd(0x30);
    __delay_us(40);    //AGUARDA 40us

    comando_lcd(0x30);
    __delay_ms(5);    //AGUARDA 5ms
    comando_lcd(0x30);
    __delay_us(100);    //AGUARDA 100us
    comando_lcd(0x30);
    __delay_us(40);    //AGUARDA 40us

    comando_lcd(0x30);
    __delay_ms(5);    //AGUARDA 5ms
    comando_lcd(0x30);
    __delay_us(100);    //AGUARDA 100us
    comando_lcd(0x30);
    __delay_us(40);    //AGUARDA 40us

    //MODO DE CONFIGURACAO EM 4BITS
    comando_lcd(0x02);  //INICILIZA EM MODO 4BITS
    __delay_us(40);     //AGUARDA 40us
    comando_lcd(0x28);  //INICILIZA EM MODO 4BITS DUAS LINHAS
    __delay_us(40);     //AGUARDA 40us
    comando_lcd(0x01);  //LIMPA O LCD
    __delay_ms(2);    //AGUARDA 2ms
    comando_lcd(0x0C);  //LIGA O LCD SEM O CURSOR
    __delay_us(40);     //AGUARDA 40us
```

```
////


////
```

```
    comando_lcd(0x06);  //DESLOCAMENTO PARA DIREITA
    __delay_us(40);     //AGUARDA 40us
    ////////////////////////////////////////////////////////////////////
}

void comando_lcd(char comandos)
{
    PORTEbits.RE0 = 0;                      //HABILITA LCD PARA COMANDO, PI
    __delay_us(100);                        //AGUARDA 100us
    PORTD = (comandos & 0b11110000);        //PORTD ENVIA O COMANDO PARA LC
    PORTEbits.RE1 = 1;                      //HABILITA PINO ENABLE
    PORTEbits.RE1 = 0;                      //DESABILITA PINO ENABLE
    __delay_us(100);                        //AGUARDA 100us
    PORTD = (comandos << 4)& 0b11110000;    //PORTD ENVIA O COMANDO PARA LC
    PORTEbits.RE1 = 1;                      //HABILITA PINO ENABLE
    PORTEbits.RE1 = 0;                      //DESABILITA PINO ENABLE
    __delay_us(100);                        //AGUARDA 100us
}

void lcd_dado(char dados)
{
    PORTEbits.RE0 = 1;                      //HABILITA LCD PARA DADOS, PINO
    __delay_us(100);                        //AGUARDA 100us
    PORTD = (dados & 0b11110000);           //PORTD ENVIA O CARACTER PARA L
    PORTEbits.RE1 = 1;                      //HABILITA PINO ENABLE
    PORTEbits.RE1 = 0;                      //DESABILITA PINO ENABLE
    __delay_us(100);                        //AGUARDA 100us
    PORTD = (dados << 4)& 0b11110000;       //PORTD ENVIA O CARACTER PARA L
    PORTEbits.RE1 = 1;                      //HABILITA PINO ENABLE
    PORTEbits.RE1 = 0;                      //DESABILITA PINO ENABLE
    __delay_us(100);                        //AGUARDA 100us
}

void putch(char escrita)
{
    lcd_dado(escrita);
}

void pic_start()
{
    TRISEbits.TRISE0 = 0;       //TODOS PORTE COMO SAIDA, EXCETO MCLR PARA
    TRISEbits.TRISE1 = 0;       //TODOS PORTE COMO SAIDA, EXCETO MCLR PARA
    TRISEbits.TRISE2 = 0;       //TODOS PORTE COMO SAIDA, EXCETO MCLR PARA
    PORTEbits.RE0 = 0;          //PORTE EM ZERO
```

```
////



NO RS

D, NIBLE MAIS SIGNIFICATIVO


D, NIBLE MENOS SIGNIFICATIVO









 RS

CD, NIBLE MAIS SIGNIFICATIVO


CD, NIBLE MENOS SIGNIFICATIVO


O RESET
O RESET
O RESET
```

```
    PORTEbits.RE1 = 0;          //PORTE EM ZERO
    PORTEbits.RE2 = 0;          //PORTE EM ZERO
    TRISAbits.TRISA1 = 1;       //RA1 COMO ENTRADA
    TRISAbits.TRISA2 = 1;       //RA2 COMO ENTRADA
    PORTAbits.RA1 = 0;          //PORTA EM ZERO
    PORTAbits.RA1 = 0;          //PORTA EM ZERO

    ADCON1bits.PCFG0 = 1;       //CONFIGURANDO TODAS PORTAS DO uC COMO DIGI
    ADCON1bits.PCFG1 = 1;       //CONFIGURANDO TODAS PORTAS DO uC COMO DIGI
    ADCON1bits.PCFG2 = 1;       //CONFIGURANDO TODAS PORTAS DO uC COMO DIGI
    ADCON1bits.PCFG3 = 1;       //CONFIGURANDO TODAS PORTAS DO uC COMO DIGI

}

void timer0_config()
{
INTCONbits.GIE = 0;    // DESABILITA TODAS INTERRUPCOES MASCARADAS
INTCONbits.GIEH = 0;   // DESABILITA TODAS INTERRUPCOES DE ALTA PRIORIDADE
INTCONbits.PEIE = 0;   // DESABILITA TODAS INTERRUPCOES PERIFERICAS
INTCONbits.GIEL = 0;   // DESABILITA TODAS INTERRUPCOES DE BAIXA PRIORIDADE
INTCONbits.TMR0IE = 0; // BIT DE OVERFLOW DO TIMER0 DESLIGADO
INTCONbits.TMR0IF = 0; // BIT TIMER0 LIMPO

//REGISTRADOR T0CON

T0CONbits.TMR0ON = 0; // TIMER0 DESLIGADO
T0CONbits.T08BIT = 1;  // TIMER0 CONFIGURADO COMO 8BIT
T0CONbits.T0CS = 0;     // TIMER0 CONFIGURADO COMO TEMPORIZADOR
T0CONbits.PSA = 0;      // O PRESCALER FOI ASSOCIADO AO TIMER0

T0CONbits.T0PS0 = 1; // PRESCALER DO TIMER 1:256
T0CONbits.T0PS1 = 1;
T0CONbits.T0PS2 = 1;

// CARGA DO TIMER0

TMR0L = 0x00;
}

float timer0_carga()
{
    float result;

     T0CONbits.TMR0ON = 0;       // TIMER0 DESLIGADO
```

```
TAIS
TAIS
TAIS
TAIS
```

```
     INTCONbits.TMR0IF = 0;        // LIMPA FLAG DO TIMER0


      result = TMR0L;              // ABSORVE A CARGA DO TIMER0 NA VARIAVEL

      result = result*13.1072;     // CONVERSÃO DO RESULTADO DO TIMER0 PARA

      result = result/256.0;       // CONVERSÃO DO RESULTADO DO TIMER0 PARA

      TMR0L = 0x00;                // LIMPA A CARGA DO TIMER0


    return result;
}
```

```
RESULT

MILISEGUNDOS PARA CRYSTAL 20MHz 13ms

MILISEGUNDOS PARA CRYSTAL 20MHz 13ms
```