UNIVERSIDADE FEDERAL DO CEARÁ - CAMPUS SOBRAL

CURSO DE ENGENHARIA DA COMPUTAÇÃO

DISCIPLINA: MICROPROCESSADORES

PROFESSOR: MARCELO SOUZA

VOLTÍMETRO

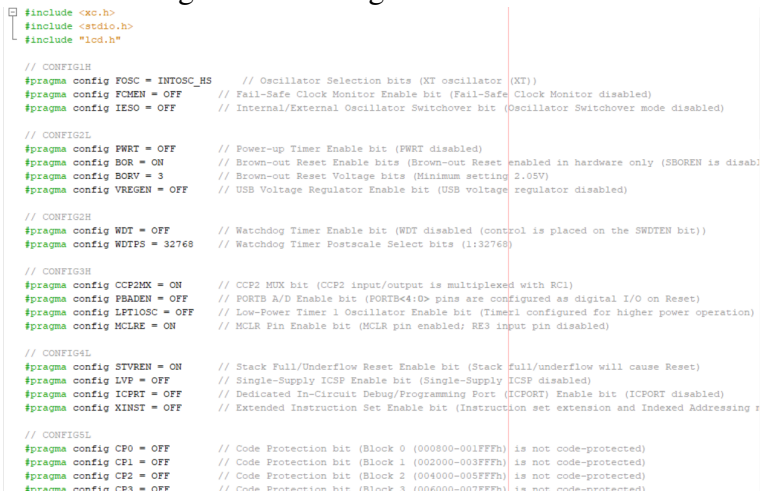
YARA MARIA SANTOS MORAIS - 475867

Sobral - CE

2023

O exercício requer que seja modificado o projeto para receber simultaneamente as voltagens de duas entradas analógicas independentes no display LCD no software Proteus. As configurações iniciais feitas no projeto não foram modificadas, como pode ser observado nas figuras 01 e 02.

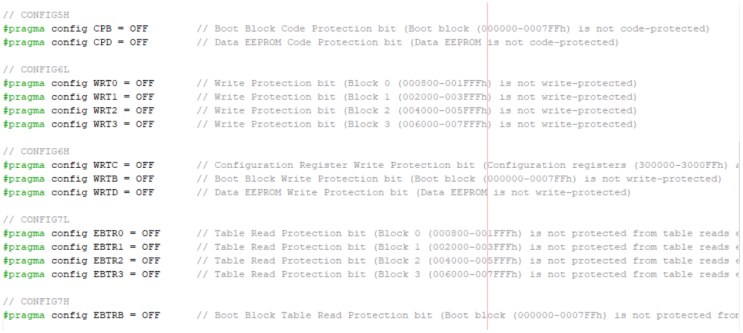
Figura 01 – Código c no MPLABX



```
#include <xc.h>
#include <stdio.h>
#include "lcd.h"

// CONFIG1H
#pragma config FOSC = INTOSC_HS     // Oscillator Selection bits (XT oscillator (XT))
#pragma config FCMEN = OFF       // Fail-Safe Clock Monitor Enable bit (Fail-Safe Clock Monitor disabled)
#pragma config IESO = OFF        // Internal/External Oscillator Switchover bit (Oscillator Switchover mode disabled)

// CONFIG2L
#pragma config PWRT = OFF        // Power-up Timer Enable bit (PWRT disabled)
#pragma config BOR = ON          // Brown-out Reset Enable bits (Brown-out Reset enabled in hardware only (SBOREN is disabl
#pragma config BORV = 3          // Brown-out Reset Voltage bits (Minimum setting 2.05V)
#pragma config VREGEN = OFF      // USB Voltage Regulator Enable bit (USB voltage regulator disabled)

// CONFIG2H
#pragma config WDT = OFF         // Watchdog Timer Enable bit (WDT disabled (control is placed on the SWDTEN bit))
#pragma config WDTPS = 32768     // Watchdog Timer Postscale Select bits (1:32768)

// CONFIG3H
#pragma config CCP2MX = ON       // CCP2 MUX bit (CCP2 input/output is multiplexed with RC1)
#pragma config PBADEN = OFF      // PORTB A/D Enable bit (PORTB<4:0> pins are configured as digital I/O on Reset)
#pragma config LPT1OSC = OFF     // Low-Power Timer 1 Oscillator Enable bit (Timer1 configured for higher power operation)
#pragma config MCLRE = ON        // MCLR Pin Enable bit (MCLR pin enabled; RE3 input pin disabled)

// CONFIG4L
#pragma config STVREN = ON       // Stack Full/Underflow Reset Enable bit (Stack full/underflow will cause Reset)
#pragma config LVP = OFF         // Single-Supply ICSP Enable bit (Single-Supply ICSP disabled)
#pragma config ICPRT = OFF       // Dedicated In-Circuit Debug/Programming Port (ICPORT) Enable bit (ICPORT disabled)
#pragma config XINST = OFF       // Extended Instruction Set Enable bit (Instruction set extension and Indexed Addressing

// CONFIG5L
#pragma config CP0 = OFF         // Code Protection bit (Block 0 (000800-001FFFh) is not code-protected)
#pragma config CP1 = OFF         // Code Protection bit (Block 1 (002000-003FFFh) is not code-protected)
#pragma config CP2 = OFF         // Code Protection bit (Block 2 (004000-005FFFh) is not code-protected)
#pragma config CP3 = OFF         // Code Protection bit (Block 3 (006000-007FFFh) is not code-protected)
```

Fonte: Autor

Figura 02 – Código c no MPLABX



```
// CONFIG5H
#pragma config CPB = OFF         // Boot Block Code Protection bit (Boot block (000000-0007FFh) is not code-protected)
#pragma config CPD = OFF         // Data EEPROM Code Protection bit (Data EEPROM is not code-protected)

// CONFIG6L
#pragma config WRT0 = OFF        // Write Protection bit (Block 0 (000800-001FFFh) is not write-protected)
#pragma config WRT1 = OFF        // Write Protection bit (Block 1 (002000-003FFFh) is not write-protected)
#pragma config WRT2 = OFF        // Write Protection bit (Block 2 (004000-005FFFh) is not write-protected)
#pragma config WRT3 = OFF        // Write Protection bit (Block 3 (006000-007FFFh) is not write-protected)

// CONFIG6H
#pragma config WRTC = OFF        // Configuration Register Write Protection bit (Configuration registers (300000-3000FFh)
#pragma config WRTB = OFF        // Boot Block Write Protection bit (Boot block (000000-0007FFh) is not write-protected)
#pragma config WRTD = OFF        // Data EEPROM Write Protection bit (Data EEPROM is not write-protected)

// CONFIG7L
#pragma config EBTR0 = OFF       // Table Read Protection bit (Block 0 (000800-001FFFh) is not protected from table reads
#pragma config EBTR1 = OFF       // Table Read Protection bit (Block 1 (002000-003FFFh) is not protected from table reads
#pragma config EBTR2 = OFF       // Table Read Protection bit (Block 2 (004000-005FFFh) is not protected from table reads
#pragma config EBTR3 = OFF       // Table Read Protection bit (Block 3 (006000-007FFFh) is not protected from table reads

// CONFIG7H
#pragma config EBTRB = OFF       // Boot Block Table Read Protection bit (Boot block (000000-0007FFh) is not protected fro
```

Fonte: Autor

As modificações feitas foram na criação da segunda porta analógica, que se tratando de código foi realizada apenas uma duplicação da inicialização da porta e do valor de tensão, agora o código apresenta VALOR_ADO e VALOR_AD1 para representar e selecionar as duas portas como sendo ANO e AN1 como está sendo mostrado na figura 03.

Figura 03 – Código c no MPLABX expecificando as duas portas analógicas

```
unsigned int Valor_AD0(){
    unsigned int C;
    ADCON0bits.CHS = 0b0000; //Seleciona AN0 como a primeira entrada analogica
    ADCON0bits.GO_DONE = 1; // Inicia conversão

    while(!ADCON0bits.GO_DONE); // Aguarda término
    C = ADRES; // Pega resultado
    return C;
}

float Tensao0(){
    float V = (float)Valor_AD0();
    V = V*5/1023;
    return V;}

unsigned int Valor_AD1(){
    unsigned int C;
    ADCON0bits.CHS = 0b0001; //Seleciona AN1 como a segunda entrada analogica
    ADCON0bits.GO_DONE = 1; // Inicia conversão

    while(!ADCON0bits.GO_DONE); // Aguarda término
    C = ADRES; // Pega resultado
    return C;
}

float Tensao1(){
    float V = (float)Valor_AD1();
    V = V*5/1023;
    return V;}

// para escrever caracteres no lcd com printf()
void putch(char data){
  escreve_lcd(data);
}
```

Fonte: Autor

Após a definição das portas e dos valores de tensões, na main foi duplicada a função ”caracter_inicio(1,1)”para ”caracter_inicio(2,1)”, especificando nesta segunda que a voltagem da porta analógica 2 será mostrada na linha 2 no lcd, abaixo da voltagem 1 e um printf chamando a função do cálculo da tensão.

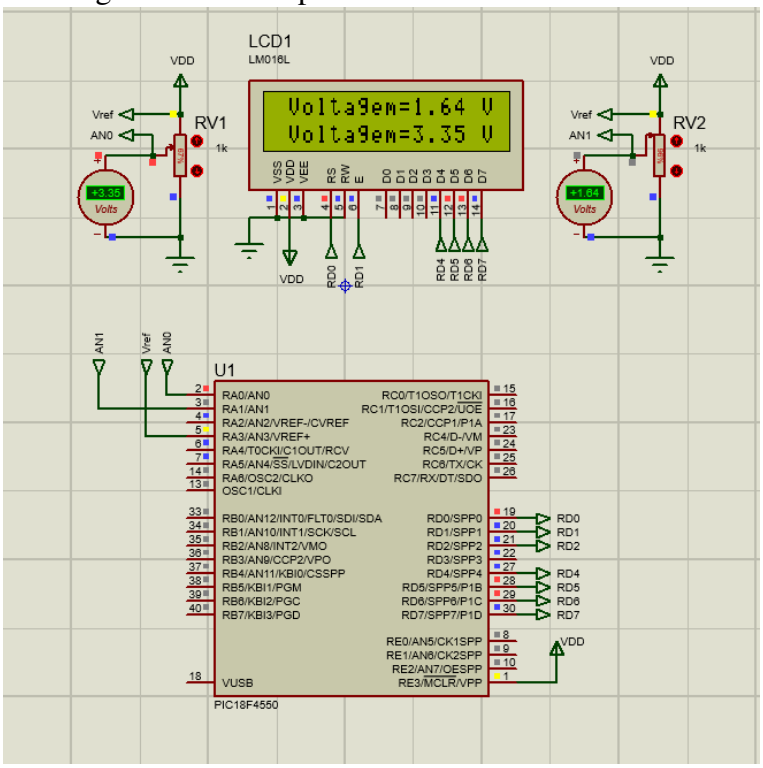
Figura 04 – Código c no MPLABX - Main

```
void main(void){
 OSCCON = 0b01100000;
 PORTD = 0;
 TRISD = 0x00;
 inicializa_lcd();
 // Inicialização do PORTA (RA0/AN0 como entrada)
 PORTA = 0;
 TRISA = 0x01;
 //Inicializações para conversor A/D
 ADCON0 = 0x01;
 ADCON1 = 0x0E;
 ADCON2 = 0x89;

 while (1) {
    caracter_inicio(1,1); //Valor da tensão 1 no lcd vai ficar na primeira linha
    printf("Voltagem=%.2f V",Tensao0()); //Valor da tensão 1
    caracter_inicio(2,1); //Valor da tensão 2 no lcd vai ficar na primeira linha
    printf("Voltagem=%.2f V",Tensao1()); //Valor da tensão 2
    __delay_ms(50);
 }



}
```

Fonte: Autor

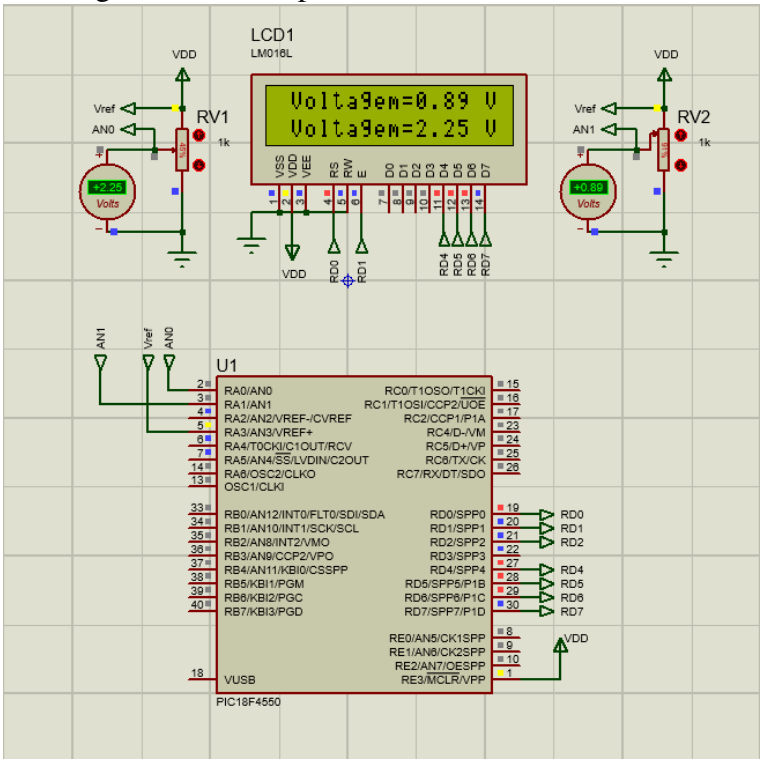
Compilando o programa no MPLABX e copiando o caminho do arquivo para o software Proteus obteve os seguintes valores de tensões vistas na figura 05, 06 e 07.

Figura 05 – Exemplo 1 de resultado das tensões



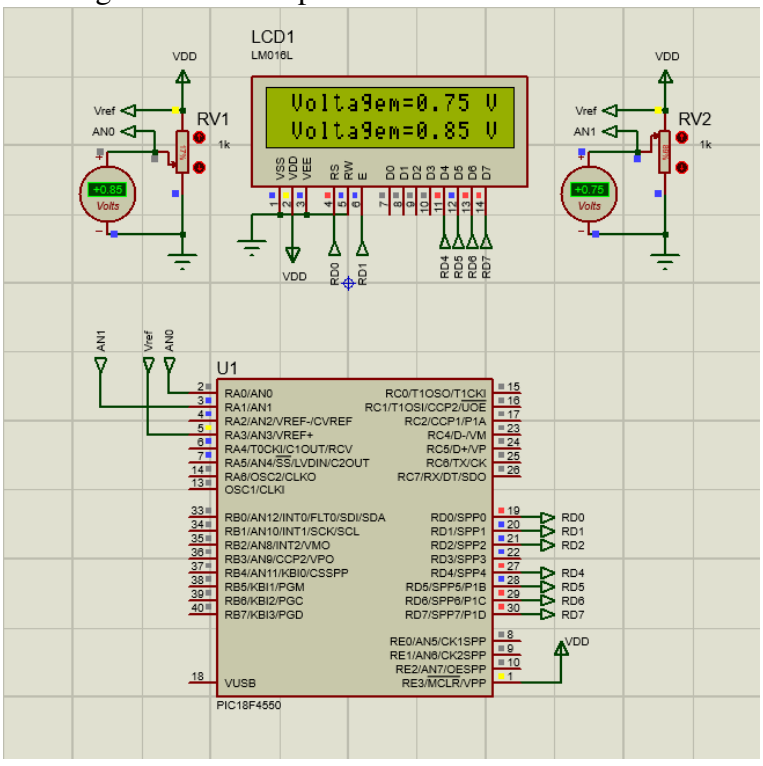
Fonte: Autor

Figura 06 – Exemplo 2 de resultado das tensões



Fonte: Autor

Figura 07 – Exemplo 3 de resultado das tensões



Fonte: Autor