Jogo do Reflexo

Universidade Federal de Itajubá - UNIFEI

Gabriel Akio Vieira Onohara

2019009078 - Turma 03

Introdução

Este é documento para o projeto final da matéria ECOP14 - Laboratório de Programação Embarcada. Neste documento será desenvolvido toda a experiência do aluno durante o projeto, dificuldades e curiosidades sobre o microcontrolador PIC18F4520 da Microchip Technology e a ultilização do simulador PICSimLab do Luis Claudio Gambôa Lopes.

Objetivos

- Aplicar os conhecimentos da aula teórica de programação embarcada, utilizando o simulador PICSimLab na linguagem C.
- Criar uma aplicação que testava os periféricos da placa como os displays de 7 segmentos, display LCD, leds, as teclas entre outros.
- Explicar todo o código e o processo de criação do projeto mostrando as dificuldades e as suas respectivas soluções.

Ressalvas

Outra parte da nota final foi a publicação de um vídeo mostrando o funcionamento do código criado e a publicação no Linkedin com os respectivos links de cada material utilizado e dos envolvidos com a matéria e o projeto.

Link:

Desenvolvimento

A ideia foi do desenvolvimento é explicar código responsável por cada funcionamento

1 - Mensagens no display LCD


```
void print1(){
    lcd_init(); //Config LCD
    lcd_cmd(L_CLR);
    lcd_cmd(L_L2);
    lcd_str("   Loading...");
    atraso_ms(1000);

    lcd_cmd(L_CLR);
    lcd_cmd(L_L1);
    lcd_str("   Bem vindo!");
    lcd_cmd(L_L2);
    lcd_str("vc esta jogando:");
    lcd_cmd(L_L3);
    lcd_str("JOGO DO REFLEXO");
    atraso_ms(2200);

    lcd_cmd(L_CLR);
    lcd_cmd(L_L1);
    lcd_str("   Pressione");
    lcd_cmd(L_L2);
    lcd_str("    a tecla");
    lcd_cmd(L_L3);
    lcd_str(" correspondente");
    lcd_cmd(L_L4);
    lcd_str("  ao LED aceso");
    atraso_ms(2200);

    lcd_cmd(L_CLR);
    lcd_cmd(L_L1);
    lcd_str("    ATENCAO!");
    lcd_cmd(L_L2);
    lcd_str("     o jogo");
    lcd_cmd(L_L3);
    lcd_str("  Comecara em:");
    lcd_cmd(L_L4);
    lcd_str("       *****");
    atraso_ms(2000);


}
```

Esse é o código responsável por exibir as mensagens no display LCD, as funções no LCD foram disponibilizadas pelo professor, a não ser a “atraso_ms” que foi criado durante uma das aulas, que basicamente são três estruturas de repetição, que se aproximam de 1 milissegundo, o parâmetro de de entrada é a quantidade desejada em ms.

2 - Contagens no display de 7-Seg


```
for (aux = 0; aux < 5; aux++) { //exibe os segundos no display de 7 segmentos
    PORTD = values7seg2[cont2];
    BitSet(PORTA, 5);
    atraso_ms(10);
    BitClr(PORTA, 5);

    PORTD = values7seg2[cont1];
    BitSet(PORTA, 4);
    atraso_ms(10);
    BitClr(PORTA, 4);
}
```

Esse código é responsável pela contagem no display de 7-seg, as primeiras quatro linhas dentro do for é responsável por mostrar o número no Display “DIS3” e as outras quatro no Display “DIS4”, os valores no vetor “value7seg2” são os valores em hexadecimal de 0 a 5 para o primeiro dígito e de 0 a 9 para o segundo dígito. Como o intervalo desejado era de aproximadamente 1,5 segundos por contagem as funções de “atraso_ms()” permitem que seja mostrado o primeiro dígito por 10 ms e o segundo por 10 ms, fazendo isso 5 vezes no “for” da variável “aux”.

3 - Interação com o teclado e LEDs

PORTB é o que mostra o LED ligado e qual deverá ser a tecla clicada.

```
    valorLed = rand() % 7 + 1;
    TRISB = 0x00;   //config da porta B como Saída
    PORTB = 0b00000000;    //apaga os LEDs no PORTB
    atraso_ms(100);    //tempo que os LEDs ficam apagados
    PORTB = valuesLED[valorLed]; //porta B recebe valor do LED aleatório
    atraso_ms(300);    //tempo que o LED fica aceso
    PORTB = 0b00000000;    //apaga os LEDs no PORTB
    TRISB = 0xF8;   //config a porta B como entrada
    TRISD = 0x0F;   //config a porta D como entrada

    valorTeclado = leTeclado(); //Lê o teclado

    if (7 - valorTeclado == valorLed) { //compara o valor recebido do teclado com o valor do LED
        if (pontuacaoUn < '9'){
            pontuacaoUn++;   //incrementa o contador de pontos da unidade
        }
        else{
            pontuacaoUn = '0';
            pontuacaoDz++;
        }
    }
    TRISD = 0x00; //config porta D como Saída
}
```

Explicando o código linha por linha:

Na primeira linha a variável de controle recebe um valor aleatório de 1 a 7 que será responsável por controlar o valor no vetor “ValuesLED” que atribui os valores em hexadecimal para o PORTB ligando o LED.

Depois o PORTB é zerado e é configurado o PORTD para entrada no objetivo de atribuir na variável o “valorTeclado” a tecla que foi pressionada.

Finalmente os valores são comparados e caso sejam iguais a pontuação é incrementada(de 1 até 9 para unidade e depois caso seja >9 a dezena é incrementada para exibir no final do jogo). Observação: a comparação da tecla com o valor do LED está na sintaxe “7 - valorTeclado == valorLed", o valor de 7 é porque os números para leitura do teclado estão em ordem contrária a dos LEDs, ou seja o mais significativo vira o menos significativo apenas na comparação.

A próxima imagem contém a função de leitura do teclado.

Essa é a função de leitura do teclado, lembrando que interação é feita por uma matriz no qual o teclado comporta, e é verificado o valor do PORTD para retornar o valor na função “BitTst”, tanto a função teclado como “BitTst” foram fornecidas pelo professor Otávio sendo que houve algumas pequenas alterações na do teclado para ficar mais condizente com nosso projeto.

```
unsigned char tc_tecla() {
    unsigned char i;
    unsigned char ret = 0;
    unsigned char aux = PORTB;

    for (i = 0; i < 3; i++) {
        PORTB = ~colunaTeclado[i];
        atraso_ms(atrasoMin);
        if (!BitTst(PORTD, 3)) { //TC1
            atraso_ms(atrasoMax);
            if (!BitTst(PORTD, 3)) {
                while (!BitTst(PORTD, 3));
                ret = 1 + i;
                break;
            }
        };
        if (!BitTst(PORTD, 2)) {
            atraso_ms(atrasoMax);
            if (!BitTst(PORTD, 2)) {
                while (!BitTst(PORTD, 2));
                ret = 4 + i;
                break;
            }
        };
        if (!BitTst(PORTD, 1)) {
            atraso_ms(atrasoMax);
            if (!BitTst(PORTD, 1)) {
                while (!BitTst(PORTD, 1));
                ret = 7 + i;
                break;
            }
        };
        if (!BitTst(PORTD, 0)) {
            atraso_ms(atrasoMax);
            if (!BitTst(PORTD, 0)) {
                while (!BitTst(PORTD, 0));
                ret = 10 + i;
                break;
            }
        };
        PORTB = 0x00;
    };

    if (!ret)ret = 255;
    if (ret == 11)ret = 0;
    PORTB = aux;
    return ret;
```