**Universidade de São Paulo**

**Escola de Engenharia de São Carlos**

****

***Projeto 2***

***Introdução aos recursos e funcionalidades de***

***Microcontroladores 8 bits com arquitetura Harvard/ RISC***

**SEL0614 - Aplicação de Microprocessadores**

**Prof. Pedro Oliveira**

Bárbara Fernandes Madera, 11915032

Matheus dos Santos Inês, 12546784

Victor Gabriel Miranda Rosa, 11232114

**São Carlos**

**2023**

**Sumário**

[**1. Objetivos**](#_j82qxkrr1r32) **2**

[**2. Explicação Teórica**](#_l7b0fm9zmsws) **2**

**3**[**. Resolução Comentada**](#_pfd3le7ii8qv) **3**

**4**[**. Resposta às perguntas**](#_yic7a5bx83qj) **6**

## **1) Objetivos**

No âmbito da capacitação em sistemas embarcados e microcontroladores, a abordagem compreensiva começa com a revisão crítica dos conceitos associados aos microcontroladores de arquitetura Harvard de 8 bits e ao conjunto de instruções RISC. Em seguida, é crucial praticar a utilização do compilador MikroC PRO for PIC, aprofundando-se na programação elementar de registradores SFR, I/O e periféricos dos microcontroladores PIC18F por meio da linguagem C.

Avançando para a aplicação prática, a execução do projeto no kit EasyPIC v7 proporciona a oportunidade de integrar conhecimentos, explorando aspectos como timer, interrupt, display, I/O, e a manipulação de header and source files, além da implementação de in-system programming. Para fortalecer ainda mais o aprendizado, a condução de simulações utilizando o software SimulIDE oferece um ambiente virtual para validar e testar os projetos antes da implementação física. Este processo abrange, de forma abrangente, conceitos fundamentais como sistemas embarcados, microcontroladores de 8 bits, arquitetura Harvard, set de instruções RISC, linguagem C, e os diversos elementos como timer, interrupt, display, I/O, e práticas como header and source files e in-system programming.

## **2) Explicação Teórica**

Nesta prática, exploramos conceitos fundamentais relacionados a microcontroladores de 8 bits, especificamente utilizando o PIC18F4550, e desenvolvemos um cronômetro digital em linguagem C. Abordamos aspectos cruciais, como timers, interrupções, manipulação de I/O e acionamento de um display de 7 segmentos.

**Configuração dos Principais Blocos:**

**Timers (TMR0):** O temporizador TMR0 foi configurado para gerar as bases de tempo para a contagem do cronômetro. A escolha do TMR0 está relacionada à sua capacidade de operar com interrupções externas, permitindo uma resposta imediata a eventos como pressionar um botão. A frequência do clock do PIC18F foi ajustada para 8 MHz, proporcionando uma base de tempo confiável para o cronômetro.

**Interrupções:** As interrupções foram implementadas para lidar com eventos assíncronos, como pressionar os botões. As interrupções externas foram configuradas nos botões RB0 e RB1, utilizando a mudança de borda como condição de acionamento. Isso garante que a contagem seja iniciada apenas quando um botão for pressionado, evitando-se assim, contagens indesejadas.

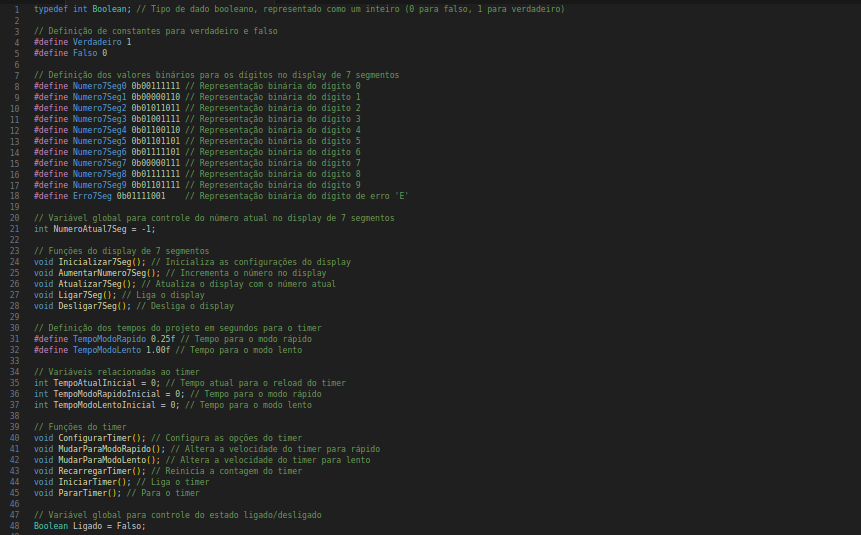
**Acionamento do Display de 7 Segmentos:** O controle do display de 7 segmentos foi realizado através das chaves de SW4. Cada chave é responsável por ligar um dos displays, enquanto a Porta D do PIC18F é usada para controlar os segmentos. A configuração do display como cátodo comum foi considerada, acendendo os segmentos com nível lógico 1.

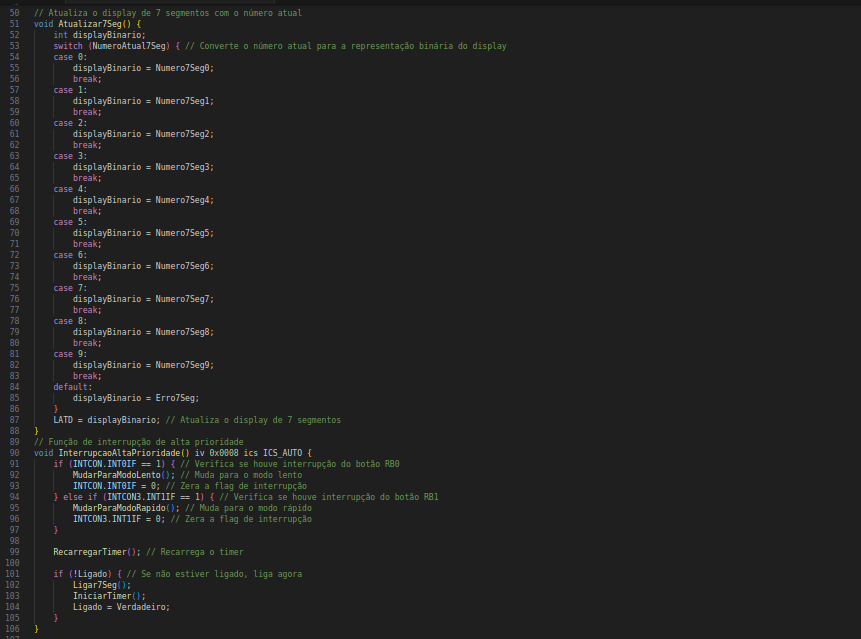
## **3) Implementação**

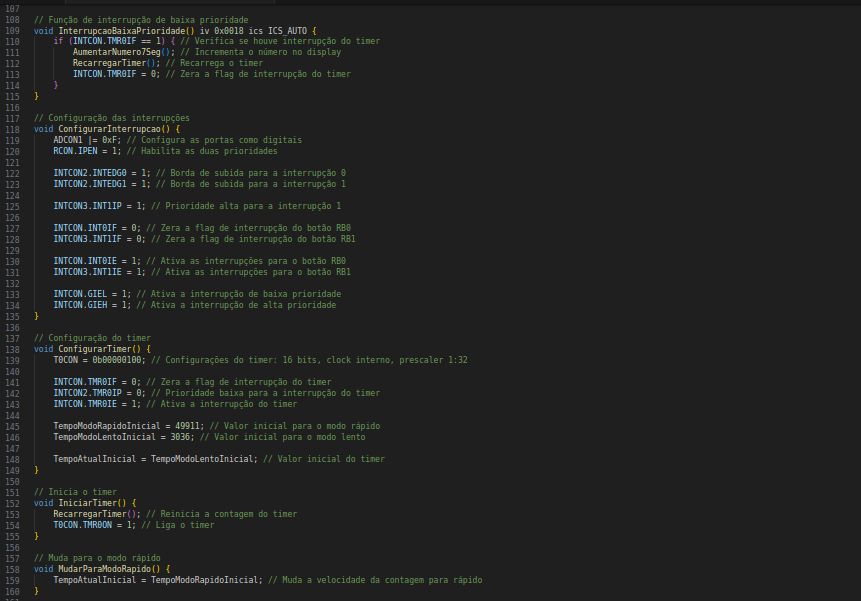
## **3.1) Resolução Comentada**

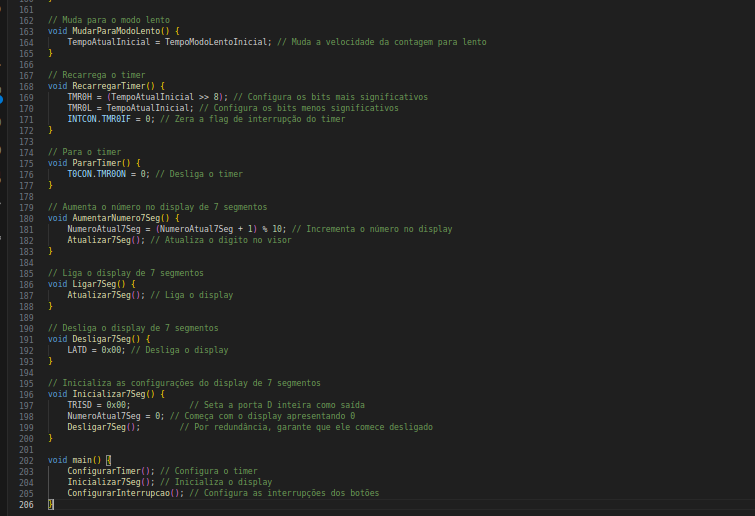
O projeto resultou em um cronômetro digital funcional que atende aos requisitos propostos. Quando o botão RB0 é pressionado, o display conta de 0 a 9 em loop, com um período de 1 segundo. Pressionando o botão RB1, a contagem ocorre com um período reduzido de 0,25 segundos. A contagem inicia apenas quando um dos botões é pressionado, proporcionando uma operação intuitiva.

Neste sentido, o programa escrito define constantes para representações binárias de dígitos no display, funções para inicialização e controle do display, configuração e operação do timer, além de interrupções para manipular botões e alterar a velocidade do timer. A lógica do programa envolve a atualização do display de acordo com um número que é incrementado periodicamente, juntamente com a capacidade de alternar entre modos rápido e lento usando botões físicos. Segue-se abaixo o código comentado implementado:



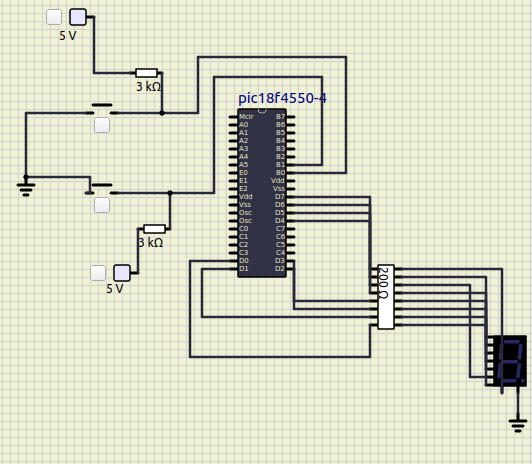






**3.2) Simulação**

Para realizar a simulação, utilizamos o software SimulIDE v0.4.15, onde construímos o circuito seguindo os parâmetros de projeto, incluindo a configuração das portas que se conectam ao display e a associação dos botões às portas RB0 e RB1. É importante ressaltar que os testes foram realizados usando o microcontrolador PIC18F4550, fornecido pela plataforma.

****

*Figura 1: Demonstração do PIC18F4550 na plataforma do SimulIDE.*

## **4) Respostas às Perguntas/Comparação com o Projeto Anterior em Assembly (8051):**

Comparativamente ao projeto anterior em Assembly para o 8051, a implementação em linguagem C para o PIC18F oferece vantagens significativas em termos de facilidade de desenvolvimento e portabilidade. A estrutura modular da linguagem C, juntamente com bibliotecas reutilizáveis, simplifica a manutenção do código. A portabilidade do código para diferentes microcontroladores é uma vantagem importante, haja vista que ela permite adaptações mais rápidas e eficientes para diferentes plataformas.Porém, apresenta desvantagens como menor controle de baixo nível em comparação com o Assembly e possível overhead em termos de tamanho do código e tempo de execução

Em termos de desempenho, embora Assembly permita um controle mais preciso do hardware, a eficiência do código gerado pelo compilador C moderno é notável.Logo, a escolha entre C e Assembly dependerá do contexto específico do projeto, priorizando o equilíbrio entre controle de baixo nível, eficiência de desenvolvimento e desempenho.