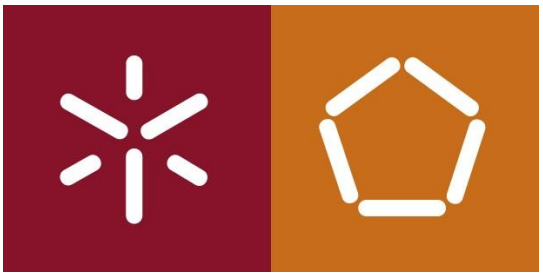


# Arquitetura e Tecnologia de Computadores

**Trabalho Prático 2022/2023**

Licenciatura em Engenharia de Telecomunicações e Informática

Universidade Do Minho


LETI 2º Ano PL1

- Fernando Mendes A101263
- Junlin Lu A101270

# 1.Introdução

Este projeto foi no proposto no âmbito da Unidade Curricular de Arquitetura Tecnologia de Computadores. O objetivo do projeto é criar um sistema de cronómetro (baseado no relógio Seiko) no nosso microcontrolador.

Este Implementação foi feita no microcontrolador C8051F380 e na linguagem C, para a edição do código e ajustar configurações do microcontrolador, utilizamos o IDE" Silicon labs" e "Configuration Wizard2".

# 2.Design

## 2.1 Funcionalidade

Deve ter as seguintes funções e ser mostrado **num LED de 7 segmentos**:

- START
- STOP
- RESTART
- SPLIT
- RESET

O nosso projeto contém todas as funcionalidades requeridas e as mesmas podem ser definidas como:

• Na sequência da implementação do necessário (configuração do timer0, timer1 a ativação das suas respetivas interrupções e a criação de ferramentas auxiliares como funções ) o projeto ficou operacional de modo a incluir as seguintes funcionalidades :

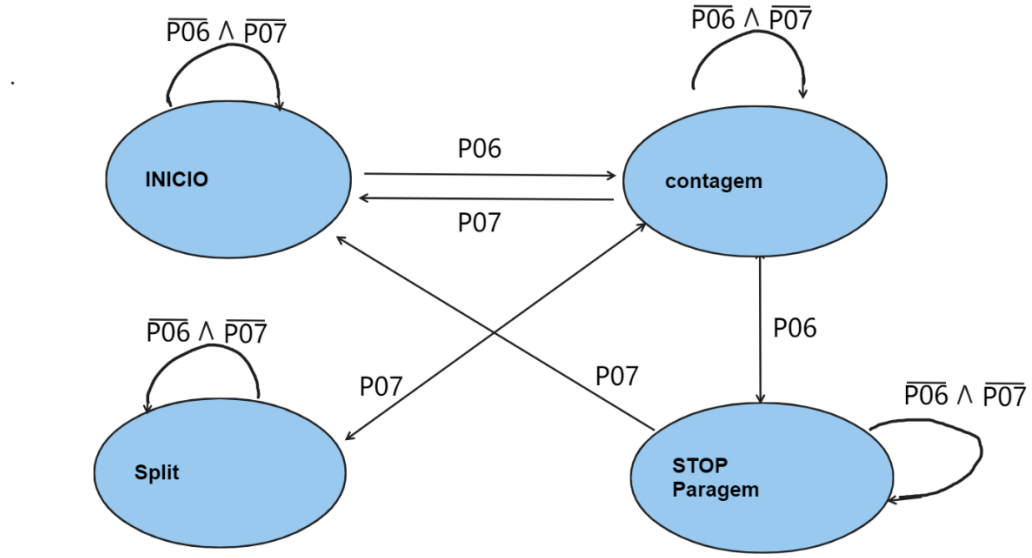
1. Start que corresponde à ativação do timer0 (ao início da contagem) assim que o botão P0_6 fosse premido;

2. Stop que corresponde à paragem da contagem, ou seja, aquando a pressão do botão P0_7 o timer0 é desativado .Para que o número correspondente a esta paragem seja mostrado ativa-se um timer (timer1) e desta forma através da sua interrupção previamente declarada e definida mostra no display de 7 segmentos o valor das dezenas e unidades;

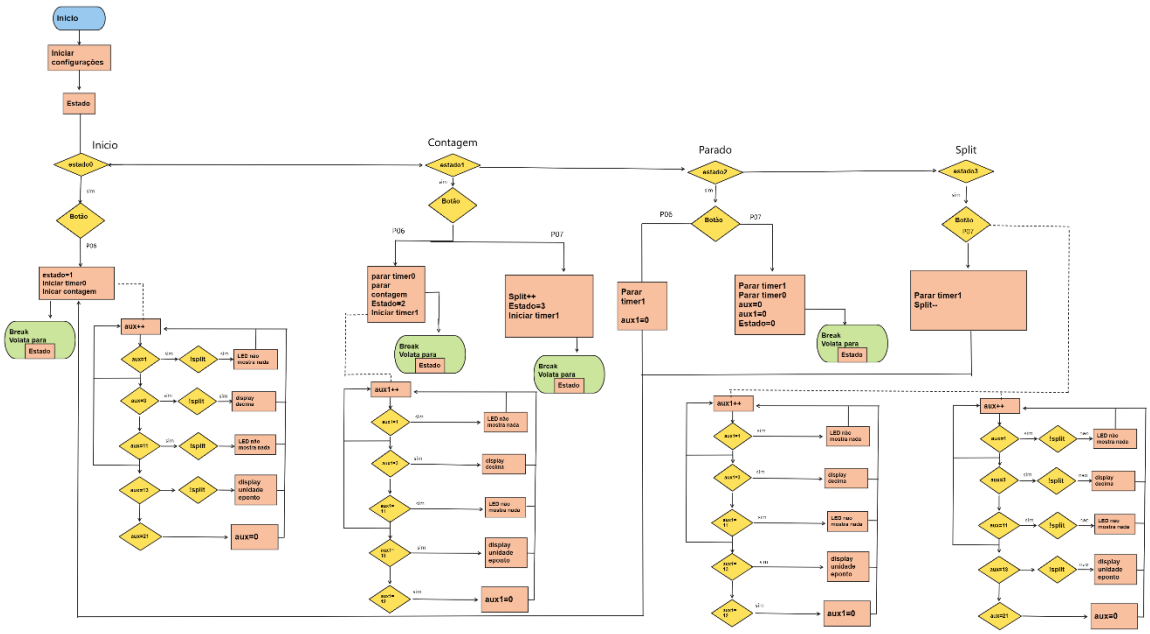
3. Split que estabelece uma divisão entre a contagem e amostragem dos valores , ou seja, aquando a pressão do botão P0_7 em plena contagem, a interrupção correspondente ao timer0 deixa de mostrar no display o valor da contagem, todavia continua realizando a mesma, e recorrendo ao mesmo método usado no STOP, ativa-se o timer1 para que faça a amostragem do número correspondente ao momento do split;

4. Reset que reinicia toda a contagem, ou seja , é responsável por definir todas as variáveis do projeto remitentes à contagem e amostragem para as configurações de inicias, prontas a serem novamente ativadas para novamente realizar a contagem;

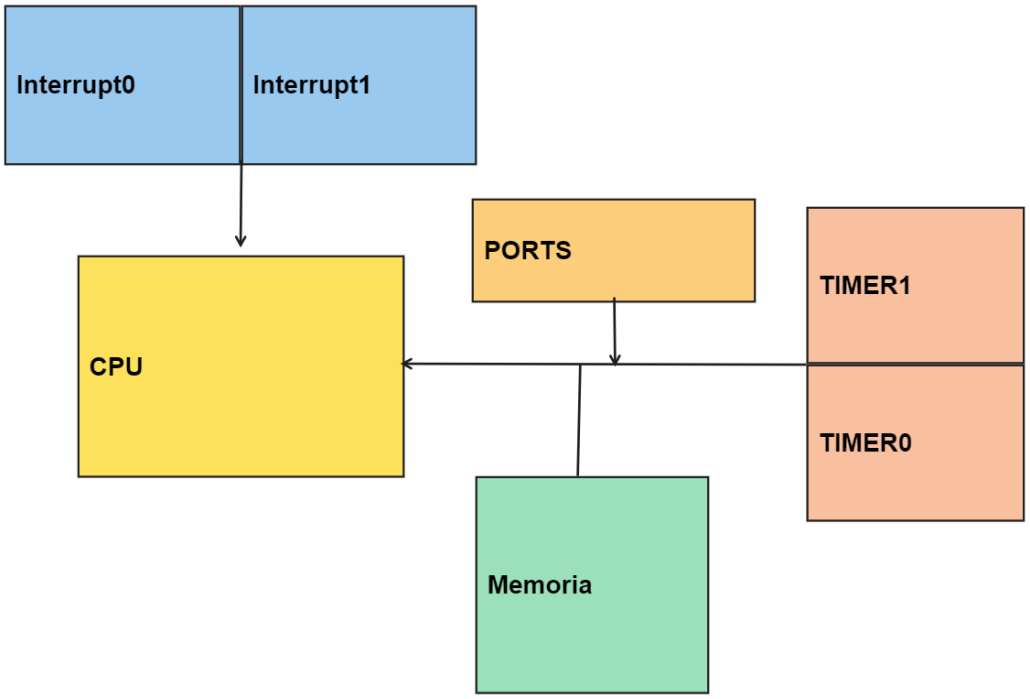
## 2.2 Máquina de estado




## 2.3 Fluxograma



## 2.2 Diagrama de blocos



# 3.Implementação

## 3.1 Configuração

```
#define Inicio 0
#define Contagem 1
#define Parado 2
#define Split 3
```

1. As diretivas de pré-processamento #define são utilizadas para criar uma constante de substituição de texto. Assim pode tornar o código mais legível, mais fácil de entender e vão ao encontro dos estados previamente definidos.

```
const char sequencia[] = {0xc0, 0xf9, 0xa4, 0xb0, 0x99, 0x92, 0x82, 0xf8, 0x80,
0x90};
volatile unsigned int d = 0, u = 1,U,D, aux = 0,aux1=0,estado,split =0;
```

2. Um Array constante de char que armazena os códigos de 7 segmentos para os dígitos de 0 a 9. “const” é útil para direcionar as variáveis para memoria de apena leitura, ou seja ,usada para variáveis e não vão ser alteradas.
3. “volatile” é usado para que o compilador não otimize o acesso à variável de forma a supor que seu valor é constante durante a execução do código
4. Criamos as funções para configurar os timers e interruptores

```
5. void iniciar_timer(void)
6. {
7.     CLKSEL = 0x03; //clock da sistema
8.     CKCON = 0x02;   //  f=1MHZ   T=1ms
9.     TMOD = 0x11;  //timer0 time1 16bit sem autoreload
10.    TR0 = 0;
11.    TR1 = 0;
12.}
```

A variável CLKSEL é utilizada para selecionar a fonte de clock para o sistema. O valor 0x03 indica que a fonte de clock é o oscilador externo.

A variável CKCON é utilizada para configurar o divisor de clock. O valor 0x02 indica que o clock é dividido por 2, o que significa que a frequência do clock será de 1 MHz.

A variável TMOD é utilizada para configurar o modo de operação do timer. O valor 0x11 indica que o timer 0 e o timer 1 estão configurados no modo 1 (modo de contagem de 16 bits sem autoreload).

As variáveis TR0 e TR1 são utilizadas para habilitar ou desabilitar o timer 0 e o timer 1,

**respetivamente. O valor 0 indica que o timer está desativa, enquanto o valor 1 indica que o timer está ativa.**

```
void iniciar_timer0(void)
{
    TR0=1;
    TL0 = -50000;
    TH0 = (-50000 >> 8);
}

void iniciar_timer1(){

    TR1=1;
    TL1 = -50000;
    TH1 = (-50000 >> 8);
}
```

As funções iniciar_timer0 e iniciar_timer1 são duas funções que habilitam os timers 0 e 1, respetivamente, e configuram os valores iniciais de contagem dos timers. para tentar aproximar 1 secundo efetuamos cálculos seguindo:
2^16=65536 máximo conta 65536 values, se frequência do sistema clock é 1MHZ então período(T)=1 µs.
-50000 é o valor inicial do timer, neste caso corresponde 0.05s(50000*1 µs).Quando contagem até transbordo(0.05s),para inicializa, tem que fazer um Shift de 8bit de High bit para low bit.

```
void iniciar_interrupcoes()
{
    IE = 0x8A;
}
```

Função interrupções para timer0 e timer1 IE=0x8A que ativa Interrupções do time0 e 1;

# 3.2 Sistema

- Optamos sempre por usar *switch* para uma maior eficiência e organização.

- O método de usarmos um timer para a contagem apenas, e a forma como foi implementado permite que o código se adapte a alterações instantâneas e acessíveis como por exemplo para implementar uma contagem com limites superiores às dezenas. Desta forma permite demonstrar o reqeusito do projeto, realizar um cronómetro e adaptá-lo facilmente a novas condições de tempo!

# 4 Conclusão

O nosso projeto foi implementado de forma a pôr em prática as aprendizagens correspondentes a configuração de timers, ao uso de interrupções e todo o conhecimento ganoh sobre o microcontrolador, os seus periféricos e registos. Desta forma o objetivo de criarmos um cronometro foi concluído sublinhando todas estas aprendizagens.

O seu funcionamento depende da configuração dos timers, realizada na função declarada no código respetiva (*iniciar_timer() )* e do uso de funções, como a função *sart()* que inicia o timer e a contagem respetivamente, a função *stop()* que é responsável por parar a contagem e mostrar o seu valor no display, a função *reset()* que usada apenas enquanto a contagem estiver parada inicializa todas as variáveis de modo a prontificar para uma nova contagem (voltar ao início), *split()* que permite separar um valor da constante contagem sobrepondo esse valor à amostragem da mesma.

Desta forma garantimos o sucesso do projeto de maneira a implementarmos os nossos conhecimentos anteriormente enunciados.

# 5 Anexos

## Código do projeto