Departamento de Engenharia Elétrica
Universidade Federal do Ceará *Campus* Sobral

(SBL0082) Microprocessadores

Prof. Me. Alan Marques da Rocha
Prof. Dr. Marcelo Marques Simões de Souza

3
Prática 03: Cronômetro com Interrupção Externa

1.1 Introdução

Os sistemas embarcados raramente "esperam" sentados: eles *contam tempo*, reagem a eventos e mantêm ritmos. O microcontrolador PIC18F4520 é uma plataforma clássica para atacar problemas de temporização porque combina três ingredientes-chave: (i) fontes de clock configuráveis (oscilador interno, cristal externo HS, PLL), (ii) temporizadores de propósito geral com *prescalers* e **interrupções**, e (iii) periféricos que se beneficiam diretamente de uma base de tempo estável (PWM, comunicação serial). Ao articular esses recursos, o PIC permite ir de "pulsos imprecisos" a bases de tempo de 1 Hz confiáveis (fundamento de cronômetros, relógios, agendadores e protocolos).

Nesta prática, você implementará um cronômetro de segundos `(00-59)` que avança exatamente a cada 1 s. A contagem é exibida em dois dígitos por meio de BCD no `PORTD` (*nibble* alto = dezenas; *nibble* baixo = unidades), solução ideal quando se usa decodificadores BCD→7 segmentos. O controle do usuário é assíncrono: dois botões conectados às **interrupções externas** `INT0`/`INT1` alternam *iniciar/pausar* e *reset*. A precisão temporal não depende de `delays` de software (sensíveis à carga da CPU), mas do **Timer0** operando em 16 bits com **preload** calculado a partir do clock do sistema. Assim, o avanço de um segundo ocorre por **interrupção**, enquanto a CPU permanece livre para outras tarefas.

1.2 Objetivos

- Compreender o uso de interrupções externas (`INT0/INT1`) para interação com botões.
- Projetar a base de tempo de 1 Hz com `Timer0` (16 bits), a partir de um cristal externo (por exemplo, 20 MHz).
- Implementar, em linguagem C (XC8), um cronômetro 00–59 com exibição BCD no `PORTD`.
- Simular e validar o projeto no *software* Proteus, observando o comportamento do contador e das interrupções.

1.3 Lista de Material

- 1 kit da placa UFC PICLAB-4520.
- 1 gravador PICKit2 para programação do microcontrolador.
- Protolab com *display* BCD de 7 segmentos.
- *Jumpers* para ligação dos circuitos.
- *Software* Proteus para simulação e averiguação antes da montagem.

1.4 Procedimento Experimental

Etapa 1: Criação do projeto no MPLAB-X e configuração do dispositivo

1. Abra o MPLAB X e crie um *Standalone Project* para o dispositivo **PIC18F4520**.

2. Selecione o compilador XC8 (C) e crie um arquivo `main.c`.

3. Escreva o código disponível em (https://abre.ai/nYlX) no arquivo criado na etapa anterior. Faça as considerações e observações necessárias. Após a criação, compile o projeto clicando no ícone do martelo ou utilize o atalho F11. Caso o código seja compilado sem erros, deverá aparecer a seguinte mensagem no Output: BUILD SUCCESSFUL (total time: 1s).

Etapa 2: Compilação e geração do arquivo `.hex`

1. No MPLAB X, selecione o projeto e execute em *Build Project* (ou *Clean and Build*).

2. O assembler irá:

   - montar o código Assembly;
   - aplicar os `CONFIG bits`;
   - gerar um arquivo objeto e, ao final, gerar o arquivo `.hex`.

3. Ao término da compilação, localize o arquivo `.hex` na pasta de saída do projeto. Em projetos MPLAB X típicos, esse arquivo aparece em um subdiretório parecido com:

   `dist/default/production/<nome_do_projeto>.production.hex`

4. Esse arquivo `.hex` contém o programa pronto para ser executado pelo PIC18F4520. É esse arquivo que será carregado no Proteus como "firmware" do microcontrolador.

Etapa 3: Montagem do circuito no Proteus

1. Abra o Proteus e crie um novo projeto/**schematic (`.pdsprj`)**

2. Insira os seguintes componentes:

   - PIC18F4520;
   - Um *Crystal* com frequência de 20 MHz;
   - Dois capacitores cerâmios ≈ 22 pF;
   - Resistor de 10kΩ entre o MCLR e VDD (*pull-up* de *reset*);
   - Fonte DC de +5V e referência GND;
   - *Switch* (`SW-SPDT`).

O circuito simulado deverá seguir a montagem apresentada na Figura 1.1, conforme ilustrado a seguir:

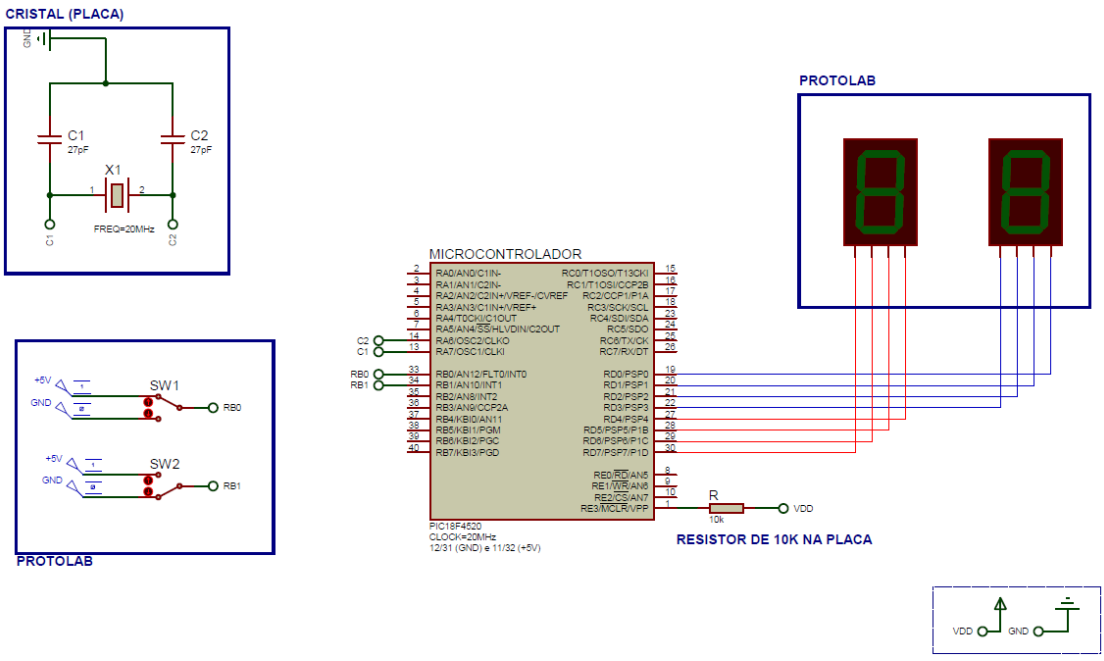

Figura 1.1: Circuito esquemático da prática 03.

Etapa 4: Carregar o firmware `.hex` no PIC dentro do Proteus

1. Após a montagem do circuito, dê um duplo clique no componente PIC18F4520.

2. Na janela de propriedades do PIC, siga os seguintes passos:

   - Em *Program File*, selecione o arquivo `.hex`, conforme ilustrado na Figura 1.2.

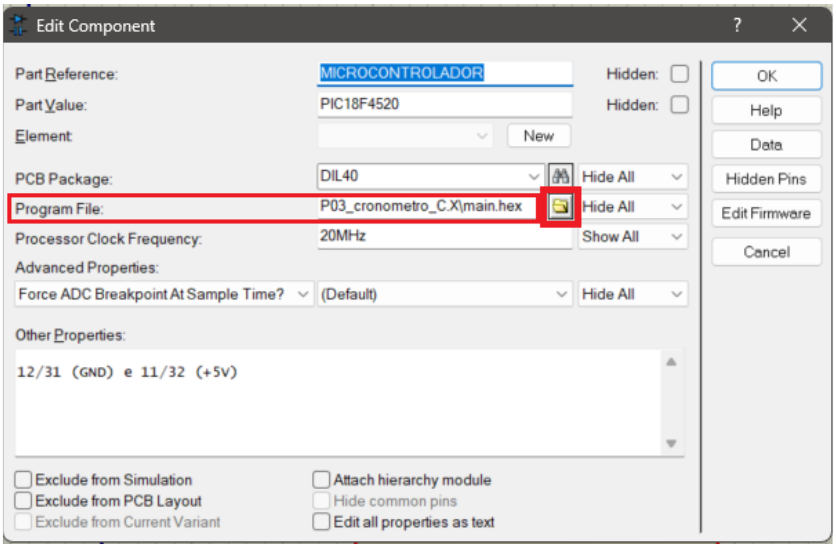

Figura 1.2: Exportação do arquivo `.hex` para o PIC no Proteus.

3. Confirme se o PIC está realmente alimentado: VDD ligado a +5 V e VSS ao GND. Se o nó VDD aparecer "cinza" durante a simulação, significa que a fonte de +5 V ainda não está realmente inserida no circuito ou conectada corretamente.

Etapa 5: Execução da simulação

1. Para realizar a simulação, clique em *Run* no Proteus (canto inferior esquerdo).

2. Se todas as conexções e o arquivo `.hex` estiverem corretos, o programa será executado, conforme ilustrado na Figura 1.3.

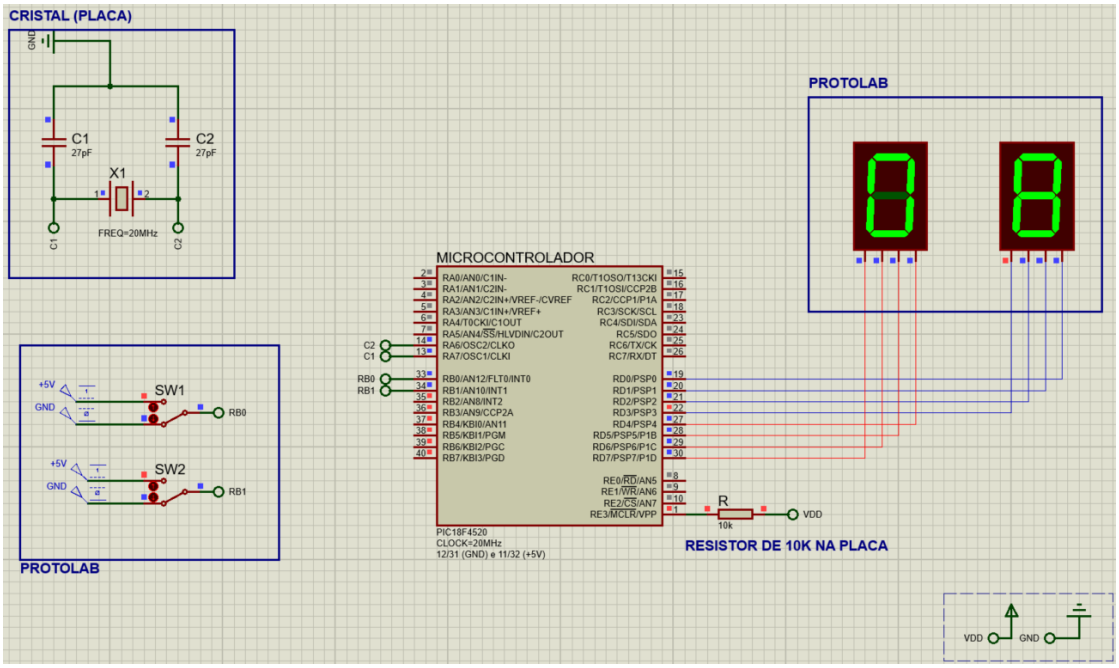

Figura 1.3: Circuito da prática 03 simulado no Proteus.

3. Utilize o *Switch* `SW1` para iniciar a contagem e `SW2` para encerrar. O cronômetro deverá iniciar em 0 e seguir até 59, reiniciando a contagem.

Etapa 6: Montagem do circuito físico

1. Monte o circuito com o auxílio da protolab, replicando o esquemático anteriormente validado em simulação no Proteus.

2. Ao se utilizar a placa UFC PICLAB-4520 não se faz necessário realizar a montagem do MCLR ligado ao VDD por meio de um resistor de 10kΩ.

3. Também não se faz necessário a montagem do cristal externo com os capacitores de desacoplamento ao terra.

4. Verifique as conexões e a continuidade dos *jumpers* e observe o que acontece com os *displays* BCD. Registre o resultado para a inclusão no relatório.

1.5 Questionário

1. Explique por que *delays* por *software* não garantem uma base de tempo estável e como o uso de `Timer0` com `preload` corrige esse problema.

2. Com Fosc = 20 MHz, calcule o *preload* do `Timer0` (16 bits, prescaler 1:256) para obter 1 Hz. Mostre as etapas do cálculo e compare com o valor usado no código.

3. Se o cristal for alterado para 4 MHz, qual deve ser o *novo preload* para manter 1 Hz? Apresente a dedução.

4. Modifique o projeto para `MM:SS` (quatro dígitos). Especifique a lógica de *overflow* (00:00 $\rightarrow$ 59:59 $\rightarrow$ 00:00) e como distribuir a saída (BCD ou segmentos) para os quatro dígitos. Realize essa etapa na linguagem C. (O código em C e a simulação do Proteus devem ser encaminhadas junto com o relatório).