Projeto de Sistemas Microcontrolados

Aula 09 - Roteiro Prático nº 2

Apresentação

Nesta aula, você avançará com a programação em C, usando o PICKit 3 em temporizações precisas. Aprenderá também a tratar interrupções no compilador C18 (ou opcionalmente no CCS). Para tal, estudará o Timer 0 e o sistema de interrupção do 18F45k20.

Objetivos

Ao final das atividades previstas para esta aula, você será capaz de:

- Descrever o funcionamento do bloco Timer 0 e compreender como são atendidas interrupções no microcontrolador 18F45k20.
- Escrever, depurar, gravar e testar programa em C que permita o tratamento do Timer 0 e do sistema de interrupções do 18F45k20.

Desenvolvendo atividades e aplicativos em C para o PICKit 3 — Nível Intermediário

Continuando com as atividades práticas, iniciadas no Roteiro Prático n.1, vamos agora para um nível intermediário de programação. Para isso, será necessário que você retome o estudo do PIC 18F45k20, usado no PICkit 3, procurando entender como são tratadas as interrupções de programa, como funciona o Timer 0 e como é feita sua programação para que, a cada estouro de contagem, uma interrupção de programa seja executada. Além disso, observe como o *prescaler* pode ser utilizado para aumentar o tempo de estouro do Timer 0.

Nas atividades práticas desta aula, você terá a oportunidade de alterar o programa desenvolvido na aula 8 para que haja interação entre a sua execução e o ambiente externo. Será proporcionada também a oportunidade de você levar o PIC a executar temporizações precisas, tornando suas tarefas de software compatíveis com comandos de hardware.

Ressalto novamente que as atividades são requeridas em uma sequência natural de implementação e que cada atividade posterior requer o cumprimento correto da anterior e até mesmo das executadas na aula anterior, uma vez que os ambientes de desenvolvimento são os mesmos: o MPLab, o Proteus, o PICkit 3 e, opcionalmente, um dos compiladores C18 ou CCS.

Atividade 01

Estude o funcionamento do temporizador Timer 0 e o tratamento de interrupções do 18F45k20 e, em seguida, defina:

1. Qual ou quais as contagens máximas permitidas pelo Timer 0 do 18F45k20? Como o *Prescaler* pode ser utilizado para aumentar o seu tempo de contagem? Quais os registros envolvidos em sua programação?

2. Como é feito o atendimento a um pedido de interrupção no 18F45k20? Quais os registros com seus respectivos bits que habilitam e permitem o processamento de uma interrupção de Timer 0? E de uma interrupção externa através da entrada INT0?

Interrupções PIC18F45K20: INT0 e TIMER0

Interrupções PIC18F45K20: INT0 - Interrupção externa

As interrupções são utilizadas para executar um determinado trecho de código imediatamente após a ocorrência de um evento especificado. Por exemplo, se um botão for conectado ao pino RB0 e a interrupção externa INT0 estiver habilitada, um determinado trecho de código será executado sempre que o botão for pressionado, independente da sequência de instruções do código principal (*main*).

Para utilizar a interrupção externa INT0 é necessário configurar os bits de alguns registradores, como é indicado no *datasheet* do microcontrolador.

No caso do microcontrolador PIC18F45K20, os registradores INTCON, INTCON2 e RCON precisam ser configurados para utilizar a interrupção INT0. A seguir são descritos alguns bits desses registradores.

- Registrador INTCON:
    - bit 7: GIE - Habilita a interrupção geral.
    - bit 4: INT0IE - Habilita a interrupção externa INT0 no pino RB0.
    - bit 1: INT0IF - Indica a ocorrência da interrupção INT0. Se o bit INT0IF estiver alto (‘1’), ocorreu uma interrupção no pino RB0. Caso contrário, não ocorreu.
- Registrador INTCON2:
    - bit 6: INTEDG0 - Seleciona a borda de detecção do pino RB0. Se o bit INTEDG0 estiver baixo (‘0’), a interrupção será acionada quando houver uma

borda de descida no pino do INT0. Caso contrário, na borda de subida do sinal.

- Registrador RCON:
    - bit 7: IPEN - Habilita ou desabilita os níveis de prioridade nas interrupções. Se IPEN estiver baixo (‘0’), todas as interrupções estão no mesmo nível de prioridade.

O quadro abaixo apresenta um código exemplo que pode ser utilizado como base para configurar o pino RB0 para detectar interrupção externa INT0.

```
TRISB |= 0b00000001;          // Configurar o bit RB0 como entrada.

INTCONbits.INT0IE = 1;        // Habilitar interrupção externa INT0
INTCONbits.INT0IF = 0;        // Limpar flag da interrupção INT0
INTCON2bits.INTEDG0 = 0;    // Habilitar interrupção INT0 na borda de descida
RCONbits.IPEN = 0;             // Desligar todas as prioridades na interrupção.
INTCONbits.GIE = 1;            // Habilitar interrupção geral.

```

Além disso, é necessário limpar o bit de configuração PBADEN (PBADEN == ‘0’). Se o bit PBADEN estiver alto (‘1’), todos os bits da porta RB são configurados para receber sinais análogicos.

Quando a interrupção externa é detectada o microcontrolador executa as instruções indicadas no vetor de tratamento de interrupções. Para tratar as interrupções, independente do tipo de interrupção, o compilador XC8 da Microchip utilizar a instrução “**__interrupt()**”.

O trecho de código abaixo apresenta a forma de definir a função que trata as interrupções utilizando o compilador XC8.

```
void __interrupt() nomeFuncao(){
    ...
}

```

O desenvolvedor tem a permissão de escolher o nome da função (nomeFuncao). Dentro da função, é necessário verificar as *flag* de interrupção para identificar que tipo de interrupção foi acionada (interrupção externa, interrupção por temporizador,

etc).

```
void __interrupt() nomeFuncao(){
    if (INTCONbits.INT0IF == 1) {
        PORTA |= 0b00001000;
        INTCONbits.INT0IF = 0;     \\ Limpar a flag para detectar uma nova interrupção
    }
}

```

No código mostrado no quadro acima, sempre que houver uma interrupção externa INT0 a instrução “PORTA |= 0b00001000” será executada e o bit INT0IF será limpado (‘0’) para possibilitar a detecção de uma nova interrupção.

Temporizadores PIC18F45K20: TIMER0

Configurar um temporizador é configurar uma interrupção. Os temporizadores são interrupções que são detectadas quando ocorre um estouro na contagem do tempo. Sempre que a contador alcançar o tempo predeterminado uma interrupção é gerada.

Para utilizar a interrupção o temporizador TIMER0 é necessário configurar os bits de alguns registradores, conforme indica o *datasheet* do microcontrolador.

No caso do microcontrolador PIC18F45K20, os registradores T0CON, INTCON, TMR0L e TMR0H são utilizados para configurar o temporizador TIMER0. A seguir são descritos alguns bits desses registradores.

O quadro abaixo apresenta um código exemplo que pode ser utilizado como base para configurar o TIMER0.

```
OSCCONbits.IRCF = 0b101; // Oscilador interno 4MHz

T0CONbits.TMR0ON = 0; // Parar o contador TIMER0
T0CONbits.T08BIT = 0; // Selecionar o modo 16-bits
T0CONbits.T0CS = 0; // Selecionar o clock interno
T0CONbits.PSA = 0; // Selecionar o prescaler
T0CONbits.T0PS = 0; // Prescaler: dividir 1:2
INTCONbits.TMR0IE = 1; // Habilitar o TIMER0
INTCONbits.TMR0IF = 0; // Limpar a flag do TIMER0
T0CONbits.TMR0ON = 1; // Iniciar o TIMER0

INTCONbits.GIE = 1; // Habilitar a interrupção geral
INTCONbits.PEIE = 1; // Habilitar a interrupção de periferica; TIMER0 = Periférico.

```

Atenção

Acesse o *datasheet* do microcontrolador PIC18F45K20 para entender melhor cada bit de cada registrador utilizado.

A definição da função "**__interrupt()**" também pode ser utilizada para detectar o estouro do temporizador TIMER0. É necessário somente verificar o bit de detecção da interrupção TIMER0, bit TMR0IF do registrador INTCON.

```
void __interrupt() nomeFuncao(){
   if (INTCONbits.TMR0IF == 1){

        ...

        INTCONbits.TMR0IF = 0;     \\ Limpar a flag
    }
}

```

Como saber o tempo de contagem do temporizador TIMER0? **Veja o passo a passo a seguir!**

Vamos utilizar o trecho de código apresentado anteriormente.

1. Frequência do oscilador.

   Fclk = 4MHz;

2. Frequência do oscilador após uso do prescaler. Prescaler 1:2 (bits T0PS = 0).

   Fclk = 4MHz / 2 = 2MHz;

3. Período do cada instrução de contagem do TMR0 (Timer0).

   Ttmr = 4 / Fclk = 4 / 2MHz = 2 µs.

4. Período de estouro do temporizador configurado no modo 16-bits (0 a 65536).

   T = (Ttmr) * 65536 = 2µs * 65536 = 131,072 ms.

Assim, diante das configurações apresentadas anteriormente, o temporizador gera uma interrupção a cada 131,072 ms.

Atenção

Supondo que foram realizadas as configurações do TIMER0 descritas anteriormente e que o programador precisa executar um trecho de código a cada 525ms, uma solução é executar o código após 4 estouros do TMR0.

(4 * 131,072 ms) é aproximadamente 525 ms.

Ou seja, após quatro detecções de interrupção, através do bit INTCONbits.TMR0IF, o trecho de código deve ser executado.

Atividade 02

1. Altere o programa escrito no Roteiro Prático n.1 – Atividade 3 de modo que ele:

   - Só inicie acendimentos de Leds após a chave Sw1 ser pressionada;
   - Após a chave Sw1 ser pressionada, aguarde 3 s para acender o LED0;

- Após o acendimento do LED0, aguarde 2 s para que o LED2 comece a piscar;
- Após o LED2 começar a piscar, aguarde mais 2 segundos para começar a rotacionar o acendimento dos Leds LED4, LED5, LED6 e LED7, de tal forma que apenas um led permaneça aceso por exatos 0,6 segundos.

Atividade 03

---

1. Compile, usando o C18 (ou opcionalmente o CCS), o programa escrito na Atividade 2. Armazene o executável no PIC do esquemático desenhado no *Proteus* e simule sua execução.

Atividade 04

---

1. Grave o executável do programa escrito na Atividade 3 no PIC do *Demo Board* do PICKit 3 e teste o seu funcionamento.

Resumo

Nesta aula prática, você estudou o bloco Timer 0 e o sistema de interrupções do microcontrolador 18F45k20. Evoluiu no uso do PICKit 3 e na programação da linguagem C, usando funções e diretivas do compilador C18 (ou opcionalmente do CCS).

Autoavaliação

1. Descreva o funcionamento do Timer0 do 18F45k20?
2. Quais os registros usados para tratamento de interrupção quando ocorre um estouro no Timer0?
3. Relacione as diretivas utilizadas durante esta aula prática.
4. Relacione as instruções utilizadas durante esta aula prática.
5. Explique os procedimentos usados nesta aula prática para definir temporizações precisas usando o Timer0.

Referências

PEREIRA, Fábio. **Microcontroladores PIC:** Programação em C. Fábio Pereira. São Paulo: Érica, 2005.

_____. **Microcontroladores PIC 18 Detalhado:** Hardware e Software. São Paulo: Érica, 2010.

SOUZA, David J.; LAVINIA, Nicolas C. **Conectando o PIC:** Explorando recursos avançados. São Paulo: Érica, 2003.