Timer0

Introdução

Sempre que uma aplicação precisar de um controle preciso de tempo, você precisará utilizar timers.

É muito comum ver a utilização dos timers diretamente empregada com interrupções. Por isso, é importante estar com o conceito de interrupções bem definido. A ideia é utilizar o timer para gerar interrupções em tempos pré-determinados. Isso funciona muito bem na criação e leitura de um PWM (muito conhecido por técnicos em eletrônica), já que você precisa criar pulsos em intervalos de tempo específico.

No PIC18F45k22 existe mais de um tipo de timer. Todos eles atuam como controle de tempo, porém, eles tem configurações diferentes e funcionam de formas diferentes. O timer0 é específico para controle de tempo (temporização) ou para contagem (counter).

O registrador T0CON

11.1 Register Definitions: Timer0 Control

REGISTER 11-1: T0CON: TIMER0 CONTROL REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	TOPS<2:0>		
bit 7							bit 0

Bit TMR0ON

É o bit que ativa o timer0. Você pode configurar todo o timer sem necessariamente ativá-lo, deixando isso para algum evento específico. Acesse-o diretamente com T0CONbits.TMR0ON ativá-lo.

Bit T08BIT

Este bit aciona o modo 8 bits do timer0. Por padrão, o timer0 trabalha no modo 16 bits. Dando acesso aos dois registradores TMR0H e TMR0L. Mas é uma opção sua trabalhar no modo 8 bits. Se sua aplicação precisa de menos que 255 incrementos (independente da fonte), é interessante utilizar este modo 8 bits.

T0CS

O timer0 possui 2 fontes de clock, sendo o ciclo de máquina (0,5 microssegundo) como padrão. Mas ativar este bit coloca o pino T0CKI (pino RA4 do PIC18F45k22) como a fonte de clock, ativando o modo **contador** do timer0. Assim, você pode programar overflows baseado nos pulsos que entram neste pino, podendo ser se um sensor, por exemplo.

T0SE

Este bit funciona somente se o T0CS for setado. Ele define a borda de leitura. Em nível lógico 0, o pulso é contado na subida do sinal. Em nível lógico 1, o pulso é contado na descida do sinal.

PSA

Se o prescaler for habilitado, você estará automaticamente dividindo sua fonte de clock por pelo menos 2. Então, caso queira desativar isso, ative este bit.

T0PS

Define o nível do prescaler. Imagine o prescaler como um redutor de marcha de uma bicicleta, quanto maior for o redutor, mais pedaladas você precisa dar para andar a mesma distância. Funciona da mesma forma com os pulsos de clock.

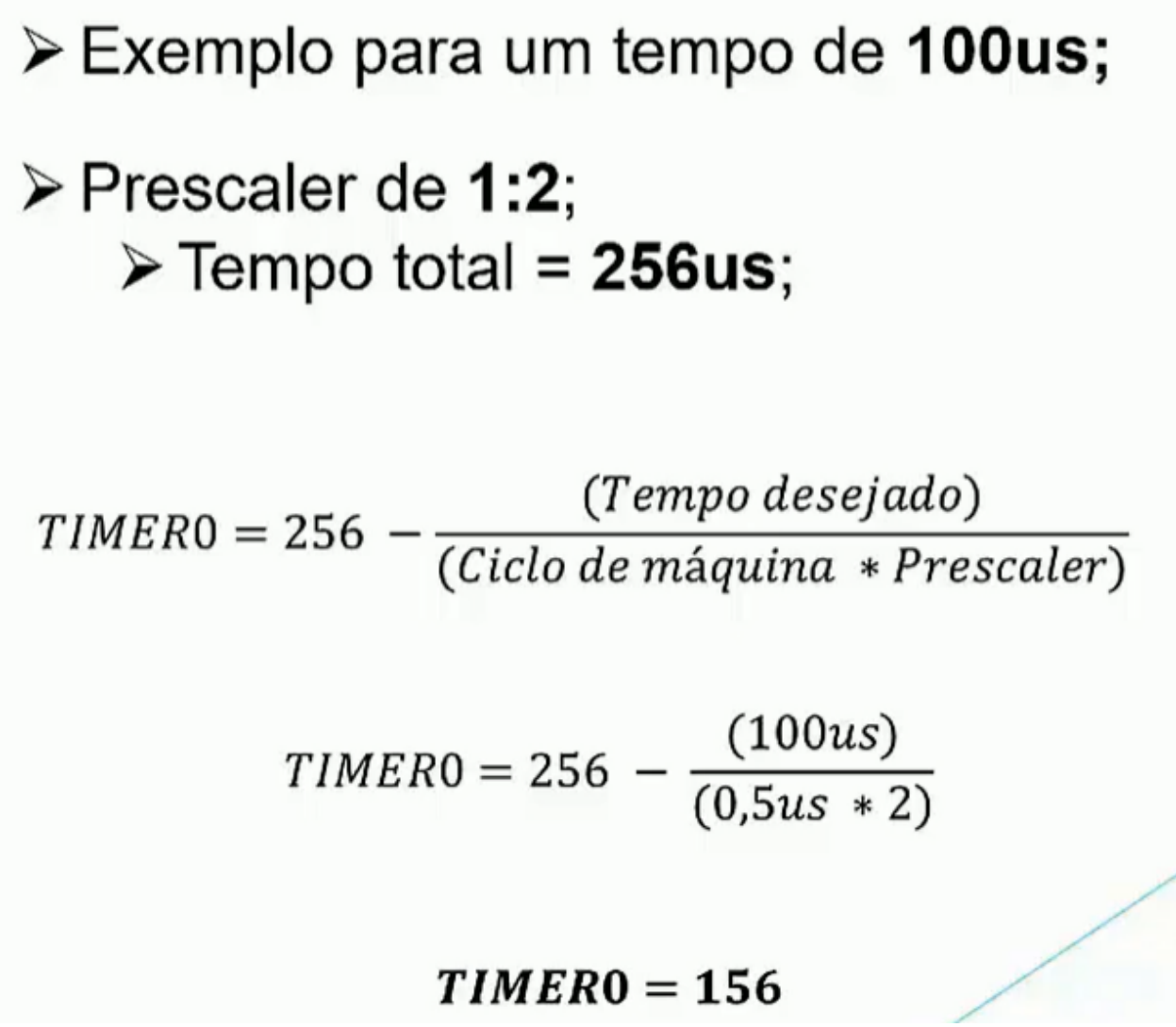
No maior prescaler disponível (256), e no modo 16 bits, o timer0 leva cerca de 8,3 segundos para gerar um overflow. Para definir o prescaler ideal para você, considere que o tempo de overflow deve ser sempre um pouco maior que o tempo que você precisa para sua aplicação.

Calculando tempos de overflow

- Prescaler 1:2 em 8 bits → 2 x 0,5 x 256 = 256us para overflow
- Prescaler 1:4 em 8 bits → 4 x 0,5 x 256 = 512us para overflow
- Prescaler 1:256 em 8 bits → 256 x 0,5 x 256 = 32,768ms para overflow
- Prescaler 1:2 em 16 bits → 2 x 0,5 x 65536 = 65,646ms para overflow
- Prescaler 1:256 em 16 bits → 256 x 0,5 x 65536 = 8,388s para overflow

Depois de definir um prescaler adequado, calcule o valor de TMR0 ideal para ajustar o tempo exato necessário para sua aplicação através das equações:

Calculando 100us no 8 bits:



Calculando 5 segundos em 16 bits:

- Exemplo para um tempo de **5s**;
- Prescaler **1:256**
  - Tempo total = **8388ms**;

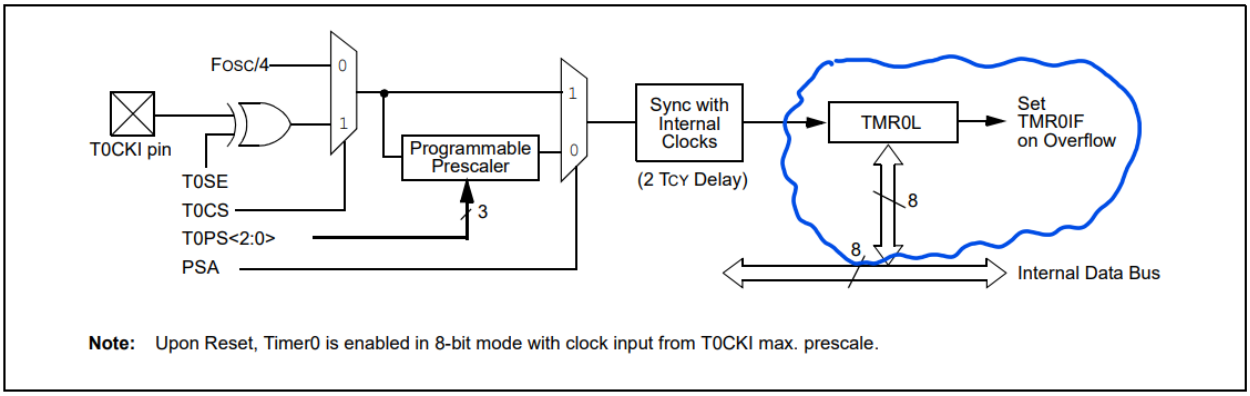
$$TMR0 = 65536 - \left(\frac{Tempo\ desejado}{ciclo\ de\ máquina\ x\ prescaler}\right)$$

$$TMR0 = 65536 - \left(\frac{5}{0{,}5us\ x\ 256}\right)$$

$$TMR0 = 26473{,}5 <=> 26474$$

Registrador TMR0L e bit TMR0IF

TMR0L é um registrador de 8 bits. A função dele é ser uma espécie de variável que incrementa de acordo com os pulsos inseridos no circuito do timer.



No momento que o valor deste registrador ultrapassar seu limite de 8 bits (overflow), o bit TMR0IF será setado (nível lógico 1).

Overflow

Overflow (estouro da capacidade). O TIMER0 8 bits tem capacidade de incrementar de 0 à 255. Ao ocorrer mais 1 incremento, o timer volta a ser 0 (TMR0 = 0) e o interruptor vai para 1 (TMR0IF = 1), acontecerá também o overflow. O tempo deste estouro vai depender do ciclo de máquina e do prescaler. Quanto atingir 255 +1 ocorrerá o prescaler.

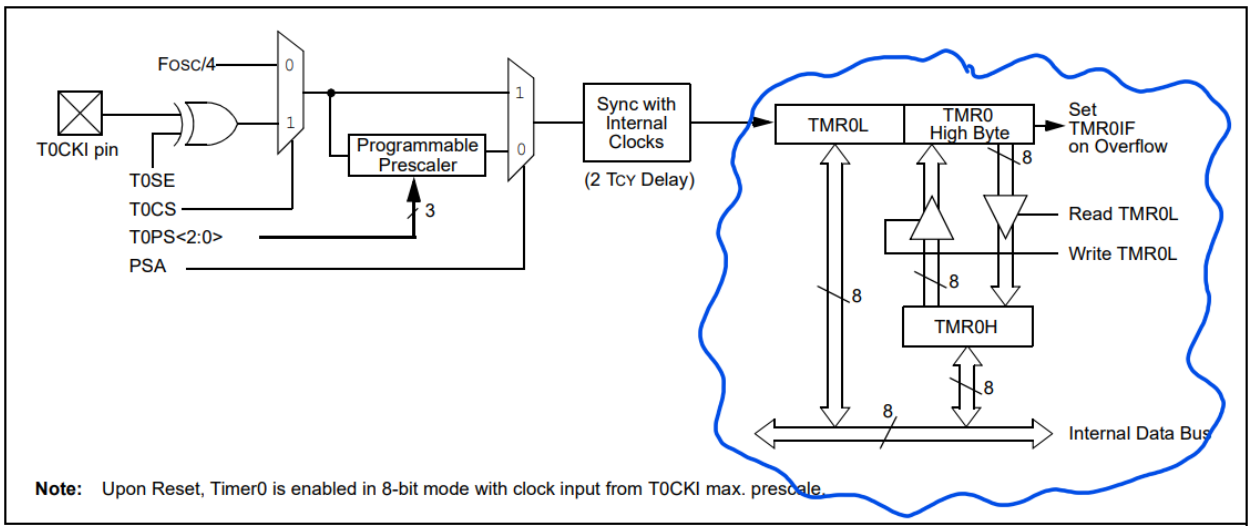
No modo 16 bits, a quantidade de incrementos é elevada a 65535. Aumentando também o range de tempo necessário para sofrer o overflow.

Interrupção por Timer0

Além de ativar a interrupção global e interrupção por periférico no registrador INTCON, também é necessário ativar a interrupção por timer0. Os bits são:

- TMR0IE → Habilita interrupção por timer0
- TMR0IF → Flag de interrupção
- TMR0IP → Bit de prioridade. Setar este bit torna esta interrupção de alta prioridade

O buffer high byte



Durante o processo, se eu tenho um processador de 8 bits e eu preciso fazer a leitura de uma informação de 16 bits (que é o caso que estamos analisando), é necessário fazer a leitura de metade do dado por vez. Durante o tempo de ler TMR0L, pode

ocorrer mais incrementos antes que eu possa ler a parte alta, causando um erro de leitura. Para resolver este problema, foi implementado o barramento HighByte.

O módulo High Byte se trata de um buffer. O TMR0H não está diretamente conectado ao barramento dos timers, que seria esta linha conectada ao Clock Sync. Ao invés disso, ele é conectado ao buffer High Byte.

Quando o programador precisa ler ou escrever valores de TMR0L e TMR0H, algumas particularidades devem ser levadas em conta:

- Ao **escrever** na parte baixa (TMR0L), automaticamente o valor presente em TRM0H é carregado no buffer HighByte. Isto ocorre dentro de 1 ciclo de máquina.
- Ao **ler** a parte baixa (TMR0L) acontece o contrário. A informação presente no buffer é carregada no TMR0H. Isto ocorre dentro de 1 ciclo de máquina.

Então a leitura ou a escrita do TMR0L é o gatilho que provoca a atualização do valor de TMR0H de acordo com o valor presente no buffer.

Portando, para escrever nestes registradores é necessário começar pelo TMR0H. Para ler, é necessário começar pelo TMR0L.