Operação com Timer0

Sempre que uma aplicação precisar de um controle preciso de tempo, você precirará utilizar timers. Os timers utilizam o clock interno do microcontrolador para contar tempo. Via software, o programador pode especificar rotinas que dependem deste tempo como:

- Atualização de display
- Captura de dados
- Gerar e capturar pulsos PWM
- Realizar leituras ou acionamentos periódicos
- Execução de tasks de um RTOS

Pelo datasheet, é possível ver que existe mais de um tipo de timer. Geralmente, eles são apresentados como timer0, timer1/3/5 e timer2/4/6. Cada um dos grupos apresentam características diferentes, como: tamanho de buffer; resolução; prescaler; fonte de clock; etc.

O timer0 é ideal para contagens mais prolongadas (alguns segundos). Geralmente os timers atuam na escala de milissegundos ou menos. Para configurar este timer, se faz a configuração do T0CON:

- Bit 7 **TMR0ON** - É o bit que ativa o timer0. Você pode configurar todo o timer sem necessariamente ativá-lo, deixando isso para algum evento específico.
- Bit 6 **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. Se a aplicação precisa de menos que 255 incrementos (independente da fonte), é interessante utilizar este modo 8 bits.
- Bit 5 **T0CS** - O timer0 possui 2 fontes de clock, sendo o ciclo de máquina 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. Desta forma, é possível programar overflows baseado nos pulsos que entram neste pino, podendo ser se um sensor, por exemplo.
- Bit 4 **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.
- Bit 3 **PSA** – Aciona o prescaler. O prescaler é acionado por padrão e parte dividindo a frequência de clock por 2.
- Bit<2:0> **T0PS** – Seleciona o valor do prescaler. Quando maior, mais tempo o timer leva para sofrer o estouro.

Prescaler e tempo de overflow

O prescaler do timer é um recurso encontrado em muitos microcontroladores que permite ajustar a frequência do clock do timer, dividindo-a por um fator específico antes que os ciclos de contagem do timer sejam contados. Isso é útil quando você precisa controlar a resolução do tempo ou reduzir a velocidade de contagem do timer para atender às necessidades específicas da sua aplicação.

Por exemplo, se o prescaler estiver configurado para um fator de divisão de 8 e a frequência do clock do timer for 8 MHz, a cada 8 pulsos de clock, apenas um pulso é contado pelo timer.

Veja o exemplo de alguns valores pré-calculados de timer com um oscilador de 8Mhz, gerando um ciclo de máquina de 0,5 microssegundo, para gerar uma interrupção por 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

A equação matemática para encontrar o tempo exato de overflow é a segunte:

$$Tempo\ de\ Overflow\ =\ Prescaler\ x\ ciclo\ de\ máquina\ x\ buffer$$

Mas desta forma, só é possível encontrar valores fixos de overflow. E se o sistema necessitasse de uma rotina que deve ser executada exatamente a cada 100 microssegundos?

A forma de resolver esta questão é realizando um ajuste fino da contagem através dos próprios buffers (TMR0L e TMR0H). Define-se um tempo de overflow um pouco maior, e diminui pouco a pouco inicializando estes buffers já com algum valor. Existe também uma equação para facilitar este trabalho:

$$TMR0\ =\ Buffer - \frac{(tempo\ desejado)}{(ciclo\ de\ máquina\ x\ Prescaler)}$$

Exemplo – Configurando timer 0 para estouro a cada 100*u*s com ciclo de máquina de 0,5*u*s

Pré-configuração: Modo 8 bits com prescaler 1:2 – 256*u*s (tempo um pouco maior que o desejado)

$$TMR0\ =\ 256 - \frac{100}{(0{,}5\ x\ 2)}$$

Com a equação acima, encontramos o valor de 156. Este valor deverá ser recarregado em TMR0 sempre que ouver um estouro do timer.

Exemplo – Configurando timer 0 para estouro a cada 5s com ciclo de máquina 0,5us

Pré-configuração: Modo 16 bits com prescaler 1:256 – 8388ms (tempo um pouco maior que o desejado)

$$TMR0 \ = \ 65536 - \frac{5000000}{(0{,}5 \ x \ 256)}$$

Se o resultado não for exato (com vírgula), arrendonde. Utilize uma union para dividir o valor nos registradores high e low.

Interrupção por Timer 0

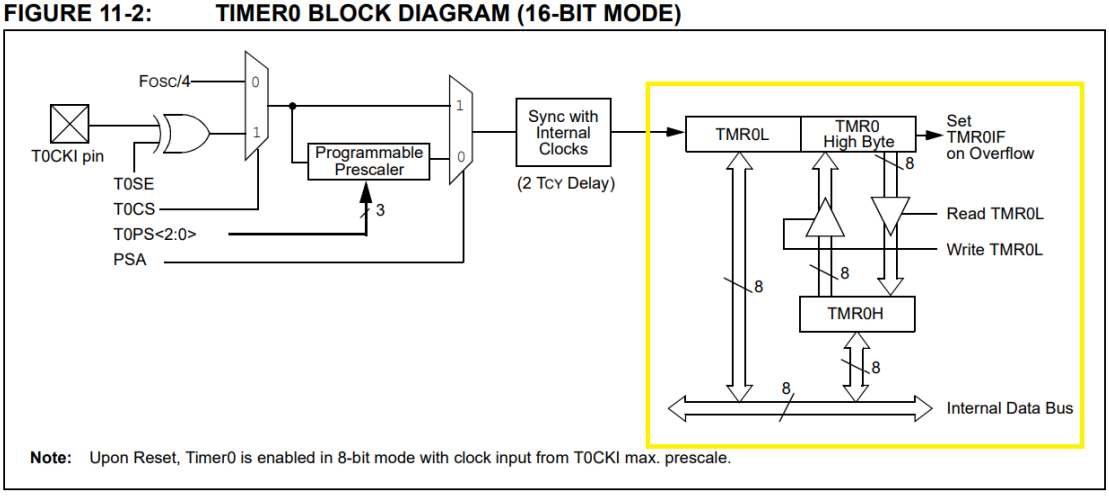
Os bits de interrupção do timer 0 são:

- TMR0IE → Habilita interrupção por timer0
- TMR0IF → Flag de interrupção
- TMR0IP → Bit de prioridade.

Ao habilitar as interrupções globais, o estouro do timer 0 já estará realizando interrupções no software. Não é obrigatório utilizar o timer com interrupção.

O buffer high byte

FIGURE 11-2: TIMER0 BLOCK DIAGRAM (16-BIT MODE)



Note: Upon Reset, Timer0 is enabled in 8-bit mode with clock input from T0CKI max. prescale.

Um processador de 8 bits não tem capacidade de fazer a leitura de uma informação de 16 bits. Para isto, é necessário fazer a leitura de cada metade da informação por vez. O problema é que durante a leitura de um registrador, o outro pode ser alterado pelo funcionamento normal do timer, ocasionando assim um erro de leitura.

O módulo High Byte contorna este erro adicionando um novo 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 à este High Byte.

Quando programa 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.