第 11 章 Timer0

目录

本章包括以下一些主要内容：

11.1 简介 11-2
11.2 控制寄存器 11-3
11.3 操作 11-4
11.4 TMR0 中断 11-5
11.5 Timer0 外部时钟的使用 11-6
11.6 TMR0 的预分频器 11-7
11.7 设计技巧 11-10
11.8 相关应用笔记 11-11
11.9 版本历史 11-12

11.1 简介

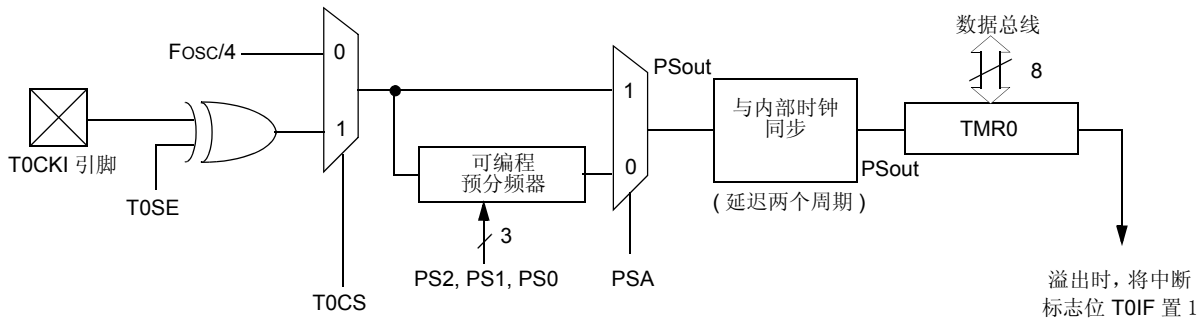
Timer0 模块有以下特性：

- 8 位定时器 / 计数器
- 可读写
- 软件可编程的 8 位预分频器
- 可选择内部或外部时钟信号
- 从 FFh 记数到 00h 时，发生溢出中断
- 外部时钟边沿选择

注：	要使 TMR0 寄存器得到 1:1 的预分频值，可将预分频器分配给看门狗定时器。

图 11-1 是 Timer0 模块的简化框图。

图 11-1：Timer0 结构框图



注 1: T0CS, T0SE, PSA, PS2:PS0 (OPTION_REG<5:0>)。
2: 该预分频器与看门狗定时器共用 (具体框图参见图 11-6)。

© 2004 Microchip Technology Inc.

11.2 控制寄存器

OPTION_REG 寄存器是一个可读写寄存器，它含有各种控制位，用来设置 TMR0/WDT 预分频器、外部 INT 中断、TMR0 和 PORTB 的弱上拉等。

注：	要使 TMR0 寄存器得到 1:1 的预分频值，可将预分频器分配给看门狗定时器。

寄存器 11-1: OPTION_REG 寄存器

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$ (1)	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

bit 7 **$\overline{\text{RBPU}}$ (1):** 弱上拉使能位
1 = 禁止弱上拉
0 = 使能弱上拉

bit 6 **INTEDG**: 中断信号边沿选择位
1 = INT 引脚的上升沿中断
0 = INT 引脚的下降沿中断

bit 5 **T0CS**: TMR0 时钟源选择位
1 = T0CKI 引脚输入时钟
0 = 内部指令周期时钟 (CLKOUT)

bit 4 **T0SE**: TMR0 时钟源边沿选择位
1 = T0CKI 引脚电平由高到低转变时递增
0 = T0CKI 引脚电平由低到高转变时递增

bit 3 **PSA**: 预分频器分配位
1 = 预分频器分配给 WDT
0 = 预分频器分配给 Timer0 模块

bit 2:0 **PS2:PS0**: 预分频器比选择位

选择位的值	TMR0 比率	WDT 比率
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

图注	
R = 可读位	W = 可写位
U = 未用位，读为 ‘0’	- n = 上电复位值

注 1:	有些器件称此位为 $\overline{\text{GPPU}}$。带有 $\overline{\text{RBPU}}$ 位的器件在 PORTB 上有弱上拉，带有 $\overline{\text{GPPU}}$ 位的器件在 GPIO 端口有弱上拉。

11.3 操作

将 T0CS 位 (OPTION<5>) 清零可选择 TMR0 模式。在定时器模式下，Timer0 模块在每个指令周期加 1(不使用预分频器)。TMR0 寄存器被写入时，在接下来的两个指令周期禁止加 1（见图 11-2 和 图 11-3)。用户可通过将校正值写入 TMR0 寄存器避开这一问题。

将 T0CS 位 (OPTION<5>) 置 1 可选择计数器模式。在计数器模式下， Timer0 可在 T0CKI 引脚的每个上升沿或下降沿递增计数。具体是上升沿还是下降沿由 Timer0 的时钟源边沿选择位 T0SE 位 (OPTION<4>) 决定。将 T0SE 位清零选择上升沿。外部时钟输入的限制在 **11.5 “Timer0 外部时钟的使用 ”** 小节中详细讨论。

预分频器由 Timer0 和看门狗定时器共用，并在分配使用时相互排斥。预分频器的分配在软件中由 PSA 控制位 (OPTION<3>) 控制。将 PSA 清零将预分频器分配给 Timer0。预分频器是不可读写的。当预分频器分配给 Timer0 时，可选择的预分频值有 1:2, 1:4,..., 1:256。预分频器的具体操作参见 **11.6 “TMR0 的预分频器 ”** 小节。

向 TMR0 寄存器写入数据将导致其禁止两个指令周期 (2TCY)。即，在 TMR0 寄存器中写入新的数据后，TMR0 在第三个指令周期后才递增计数 (图 11-2)。当预分频器被分配给 Timer0 模块时，任何向 TMR0 寄存器的写操作都将立即更新 TMR0 寄存器，并将预分频器清零。Timer0（TMR0 和预分频器）的递增计数也将在两个指令周期 (TCY) 中禁止。因此当预分频器配置为 2 时，向 TMR0 寄存器写入后，TMR0 寄存器将在 4 个 Timer0 时钟周期后才开始递增计数 (图 11-3)。此后， TMR0 将在预分频器中配置的时钟周期数之后进行递增计数。

图 11-2：Timer0 时序图 : 内部时钟 / 无预分频器

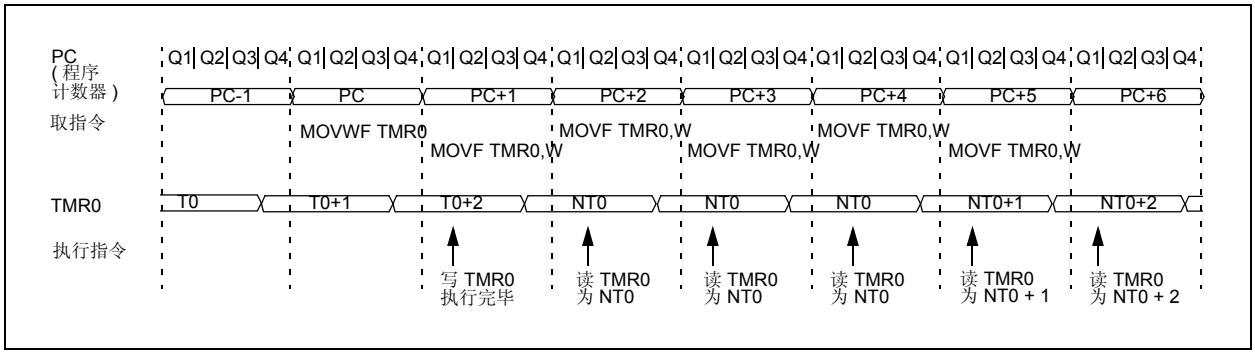
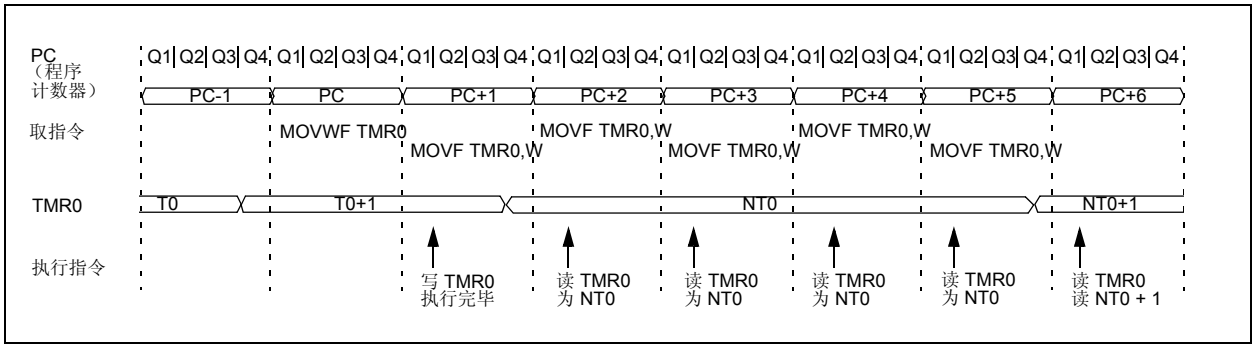


图 11-3：Timer0 时序图 : 内部时钟 / 预分频比 1:2

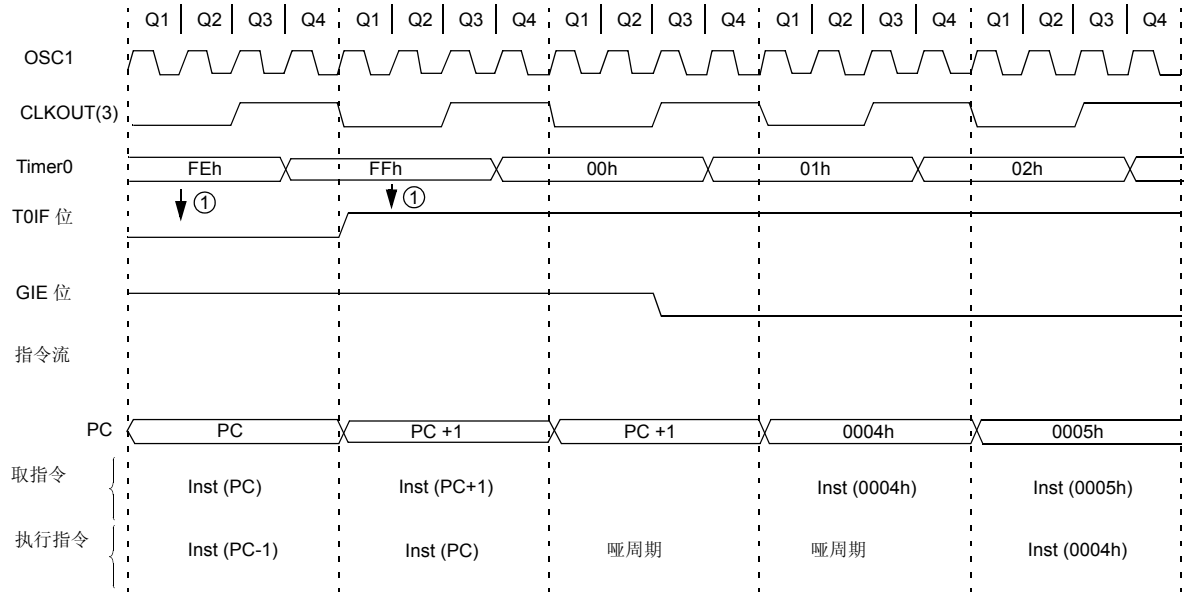


© 2004 Microchip Technology Inc.

11.4 TMR0 中断

当 TMR0 寄存器从 FFh 到 00h 发生计数溢出时，即产生 TMR0 中断。该溢出将 T0IF 位 (INTCON<2>) 置 1。中断请求可以通过清零 T0IE (INTCON<5>) 来屏蔽。在重新允许中断前，必须在 Timer0 中断服务子程序中用软件将 T0IF 位清零。休眠状态时，由于 TMR0 被关闭，所以 TMR0 中断无法唤醒单片机。图 11-4 是 Timer0 的中断时序图。

图 11-4：TMR0 中断时序图



注 1：中断标志位 T0IF 在此被采样 (每个 Q1)。
2：中断延迟时间 = 4TCY，其中 TCY = 一个指令周期。
3：CLKOUT 只在 RC 振荡模式时有效。

11.5 Timer0 外部时钟的使用

当 Timer0 使用外部时钟输入时，必须满足一定要求，详见 **11.5.1 “ 外部时钟的同步。 ”** 这些要求确保外部时钟和内部相位时钟 (Tosc) 保持同步。同步后 Timer0 的实际递增计数也将有一定延时。

11.5.1 外部时钟的同步

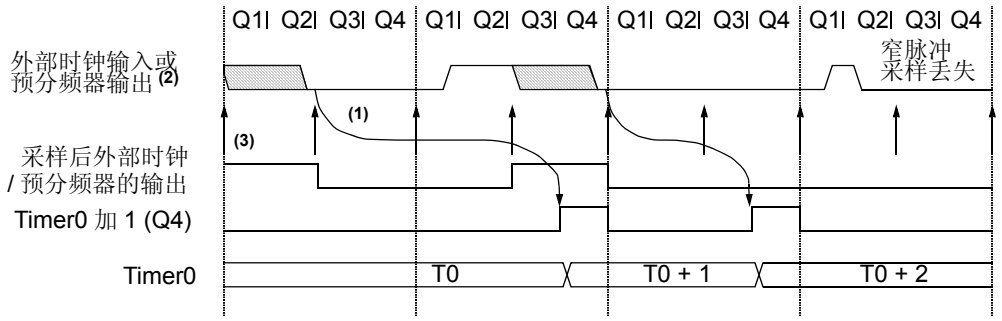
当不使用预分频器时，外部时钟输入与预分频器输出相同。在内部时钟的 Q2 和 Q4 周期对预分频器输出进行采样可实现 T0CKI 与内部时钟的同步 (图 11-5)。因此，要求 T0CKI 的高电平至少保持 2Tosc （加上 20 ns 的一小段 RC 延时）并且其低电平也至少保持 2Tosc（加上 20 ns 的一小段 RC 延时）。具体请参考所用器件的电气特性中的参数 40、41 和 42 。

当使用预分频器时，外部时钟输入被预分频器分频，故预分频器的输出是对称的。为了使外部时钟满足采样要求，必须考虑计数器。因此， T0CKI 的周期必须至少保持 4Tosc（加上 20 ns 的一小段 RC 延时）除以预分频值。对 T0CKI 的高、低电平持续时间的唯一要求是不小于 10 ns 的最小脉宽要求。具体请参考所用器件的电气特性中的参数 40、41 和 42。

11.5.2 TMR0 递增计数的延时

由于预分频器的输出与内部时钟同步，所以从外部时钟沿发生的时间到 Timer0 模块实际递增计数有一小段延时。图 11-5 显示了从外部时钟沿到计时器递增计数的延时。

图 11-5：Timer0 与外部时钟时序图



注 1: 从时钟输入变化到 Timer0 加 1 有 3Tosc 到 7Tosc 的延时。(持续时间 Q=Tosc)。
因此，测量 Timer0 输入的相邻脉冲边沿间隔时，其最大误差为 ±4Tosc。
2: 如果未选择预分频器，则为外部时钟，否则为预分频器输出。
3: 箭头所指为采样时刻。

© 2004 Microchip Technology Inc.

11.6 TMR0 的预分频器

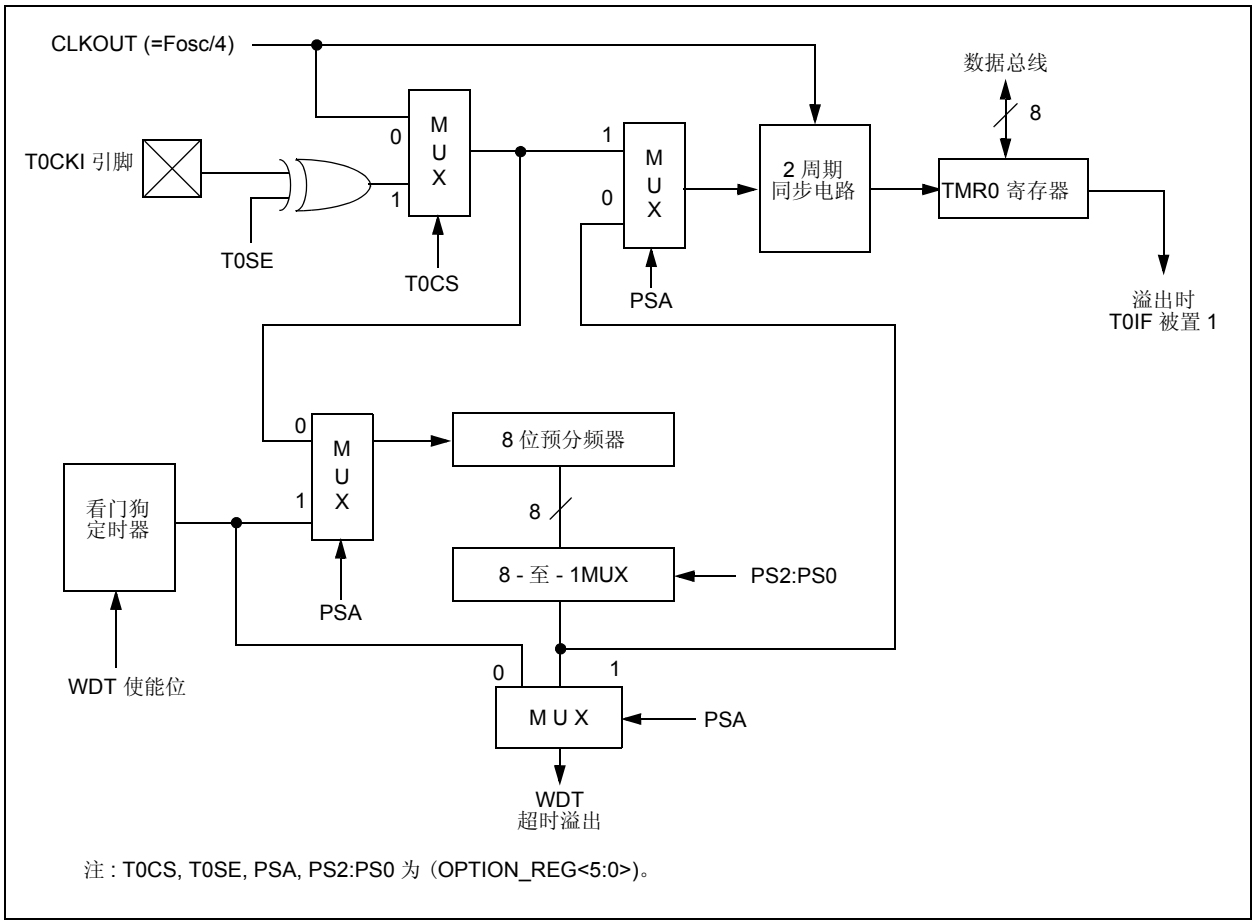
一个 8 位的计数器可作为 Timer0 模块的预分频器，或作为 WDT 的后分频器 (图 11-6)。为简化起见，在 Timer0 说明中称该计数器为 “预分频器”。因此，如果把预分频器分配给 TMR0 就意味着 WDT 无后分频器可用，反之亦然。

注：	预分频器要么分配给 Timer0 模块，要么分配给看门狗定时器。

PSA 和 PS2:PS0 位 (OPTION<3:0>) 决定预分频器的分配和分频比。

当预分频器分配给 Timer0 模块时，对 TMR0 寄存器执行的所有指令写入操作（如 `CLRF TMR0`, `MOVWF TMR0`, `BSF TMR0,x`.... 等) 都将对预分频器清零。当预分频器分配给 WDT 时，执行 `CLRWDT` 指令将同时将预分频器和 WDT 清零。预分频器是不可读写的。

图 11-6：Timer0/WDT 预分频器框图



11.6.1 预分频器分配的切换

预分频器的分配完全由软件控制，即，它可在程序执行期间 “随时” 被改变。

> **注：** 为避免器件意外复位，当把预分频器从 Timer0 分配给 WDT 时，必须执行下列指令序列 (见例 11-1) 。即使 WDT 被禁止也要执行该指令序列。

在例 11-1 中，如果所需的最终预分频值不是 1:1，就不需要对 OPTION_REG 进行初始修改。如果所需的最终预分频值是 1:1，就需先设置一个非 1:1 的临时预分频值，并将所需的最终预分频值在最后一次修改 OPTION_REG 时予以设置。

例 11-1: 改变预分频器 (Timer0→WDT)

如果所需的最终预分频值不是 1:1，则无需第 2 行和第 3 行。如果所需的最终预分频值是 1:1，则在第 2 行和第 3 行先设定临时预分频值，而所需的最终预分频值在第 10 行和第 11 行设置。

```
1)  BSF    STATUS, RP0   ;Bank1
2)  MOVLW  b'xx0x0xxx'   ;Select clock source and prescale value of
3)  MOVWF  OPTION_REG    ;other than 1:1
4)  BCF    STATUS, RP0   ;Bank0
5)  CLRF   TMR0          ;Clear TMR0 and prescaler
6)  BSF    STATUS, RP1   ;Bank1
7)  MOVLW  b'xxxx1xxx'   ;Select WDT, do not change prescale value
8)  MOVWF  OPTION_REG    ;
9)  CLRWDT               ;Clears WDT and prescaler
10) MOVLW  b'xxxx1xxx'   ;Select new prescale value and WDT
11) MOVWF  OPTION_REG    ;
12) BCF    STATUS, RP0   ;Bank0
```

要将预分频器从 WDT 分配给 Timer0 模块，使用例 11-2 所示的指令序列。

例 11-2: 改变预分频器 (WDT→Timer0)

```
CLRWDT                  ; Clear WDT and prescaler
BSF      STATUS, RP0    ; Bank1
MOVLW    b'xxxx0xxx'    ; Select TMR0, new prescale
MOVWF    OPTION_REG     ; value and clock source
BCF      STATUS, RP0    ; Bank0
```


© 2004 Microchip Technology Inc.

11.6.2 初始化

由于 Timer0 有一个软件可编程的时钟源，下面给出 2 个例子说明 Timer0 与不同时钟源工作的初始化。例 11-3 是内部时钟源 (定时器模式) 的初始化，例 11-4 是外部时钟源 (计数器模式) 的初始化。

例 11-3: Timer0 的初始化 (内部时钟源)

```
        CLRF    TMR0          ; Clear Timer0 register
        CLRF    INTCON        ; Disable interrupts and clear T0IF
        BSF     STATUS, RP0   ; Bank1
        MOVLW   0xC3          ; PortB pull-ups are disabled,
        MOVWF   OPTION_REG    ;   Interrupt on rising edge of RB0
                              ;   Timer0 increment from internal clock
                              ;   with a prescaler of 1:16.
        BCF     STATUS, RP0   ; Bank0
;**     BSF     INTCON, T0IE  ; Enable TMR0 interrupt
;**     BSF     INTCON, GIE   ; Enable all interrupts
;
; The TMR0 interrupt is disabled, do polling on the overflow bit
;
T0_OVFL_WAIT
        BTFSS   INTCON, T0IF
        GOTO    T0_OVFL_WAIT
; Timer has overflowed
```

例 11-4: Timer0 的初始化 (外部时钟源)

```
    CLRF    TMR0          ; Clear Timer0 register
    CLRF    INTCON        ; Disable interrupts and clear T0IF
    BSF     STATUS, RP0   ; Bank1
    MOVLW   0x37          ; PortB pull-ups are enabled,
    MOVWF   OPTION_REG    ;   Interrupt on falling edge of RB0
                          ;   Timer0 increment from external clock
                          ;   on the high-to-low transition of T0CKI
                          ;   with a prescaler of 1:256.
    BCF     STATUS, RP0   ; Bank0
;** BSF     INTCON, T0IE  ; Enable TMR0 interrupt
;** BSF     INTCON, GIE   ; Enable all interrupts
;
; The TMR0 interrupt is disabled, do polling on the overflow bit
;
T0_OVFL_WAIT
    BTFSS   INTCON, T0IF
    GOTO    T0_OVFL_WAIT
; Timer has overflowed
```

11.7 设计技巧

问 1: *在我的计数器 / 时钟应用中，为什么时钟会丢失时间或不准确？*

答 1:

如果您通过轮询 TMR0 来查看它是否计满回零，可以执行：

```
wait   MOVF    TMR0,W     ; read the timer into W
       BTFSS   STATUS,Z   ; see if it was zero, if so,
                          ;    break from loop
       GOTO     wait      ; if not zero yet, keep waiting
```

丢失时钟周期的两种可能的情况为：

1. 当 TMR0 的递增计数是来自内部指令时钟，或与内部时钟几乎同样快的外部时钟源时，那么在执行两周期的 GOTO 指令时，可能会发生溢出，使您的时钟周期丢失。在这种情况下，应将 TMR0 进行预分频。
   或者您也可做一个测试，通过检测小于标称值的值来查看 TMR0 是否计满回零：

   ```
   Wait   movlw  3
          subwf  TMR0,W
          btfsc  STATUS,C
          goto   Wait
   ```

2. 当向 TMR0 写入数据时，会丢失两个时钟周期。通常您希望对一个特定的时间段计数，如十进制数 100。此时您会将 156 写入 TMR0(256–100 = 156)。然而，由于向 TMR0 写入数据时会丢失两个指令周期（用于内部逻辑电路的同步），因此实际上应向定时器写入 158。

© 2004 Microchip Technology Inc.

11.8 相关应用笔记

本部分列出了与本章内容相关的应用笔记。这些应用笔记并非都是专门针对中档单片机系列而写的 (即有些针对低档系列，有些针对高档系列)，但其概念是相近的，通过适当修改并受一定限制即可使用。目前与配置字相关的应用笔记有：

标题	应用笔记 #
Frequency Counter Using PIC16C5X	AN592
A Clock Design using the PIC16C54 for LED Display and Switch Inputs	AN590

11.9 版本历史

版本 A

这是描述 Timer0 模块的初始发行版。

© 2004 Microchip Technology Inc.