应用注释

78K0S/Kx1+

示例程序 (16-位计时器/事件计数器 00)

脉冲宽度测量

本文件内容包含示例程序操作概述，使用方法及怎样设置使用 16-位定时器/事件计数器 00 的脉冲宽度测量功能。在示例程序中，使用 16-位定时器/事件计数器 00 的脉冲宽度测量功能可测量 T1000 引脚输入的信号的脉冲宽度。

目标设备

78K0S/KA1+微处理器
78K0S/KB1+微处理器
78K0S/KU1+微处理器
78K0S/KY1+微处理器

文件号： U18889CA1V0AN00 （第一版）
出版日期： 2008 年 03 月 N

中文版

目录

· 本文档中的信息于 2008 年 3 月开始使用。文档内容可能会不作通知进行修改。实际设计请参阅日电电子最新发布的数据表或数据册等，查看日电电子产品的最新指标。并非所有产品和/或类型在每个国家都能使用。请联系日电电子销售代表，了解可用性信息及其他信息。

· 未经日电电子书面许可，不能以任何形式或方式对本文档的任何部分进行复制或重现。本文档出现的任何错误，日电电子不承担责任。

· 对于在使用本文档列出的日电电子产品时产生的侵犯专利、版权以及其他第三方知识产权的行为，以及对于其他使用这些产品产生的责任，日电电子不承担责任。对于日电电子及其他子公司的任何专利、版权以及其他知识产权，日电电子没有以许可、明示、暗示以及其他任何方式授权。

· 本文档中对电路、软件及其他相关信息的描述旨在说明半导体产品的操作及应用举例。这些电路、软件和信息在客户设备设计中的使用应由客户承担全部责任。如果这些电路、软件和信息导致客户或第三方遭受损失，日电电子不承担责任。

· 日电电子尽力提升日电电子产品的质量、可靠性和安全性，但请客户同意并理解这些产品的瑕疵无法完全消除。为了尽量减少由于日电电子产品导致的财产损失或人身伤害（包括死亡），客户必须在其设计中采取足够的安全措施，如冗余、防火、防故障等特性。

· 日电电子产品分为下列三种质量等级：“标准”、“特别”及“专用”。

“专用”质量等级只适用于基于用户设计的“质量保证项目”的特定应用开发的日电电子产品。日电电子的建议应用由其质量级别决定，如下所示。客户在将日电电子产品用于特别用途之前必须检查各产品的质量等级。

“标准”：计算机、办公设备、通信设备、测试测量设备、音频视频设备、家用电子产品、机械工具、个人电子设备、工业机器人。

“特别”：运输设备（汽车、火车、轮船等）、交通控制系统、抗灾系统、防犯罪系统、安全设备、医疗设备（不专为生命救护而设计）。

“专用”：飞机、航空设备、水下中继器、核反应堆控制系统、生命救护系统、用于生命救护的医疗设备等。

除非在日电电子的数据表或数据册等当中有明确说明，日电电子产品的质量级别均为“标准”。客户如果希望日电电子产品实现日电电子未预定的应用，必须提前联系日电电子的销售代表以确定日电电子愿意支持给定应用。

（注）

（1）本声明中所用的“日电电子”表示日电电子公司，包括其控股的子公司。

（2）“日电电子产品”表示由日电电子（如上所规定）开发或制造的任何产品。

M&E 02 11-1

第一章 概述

本示例程序提供了一个例子，说明 16 位计时器/事件计数器 00 的脉冲宽度测量功能的使用。从 TI000 引脚输入的信号的脉冲宽度测量八次。

1.1 初始设置的主要内容

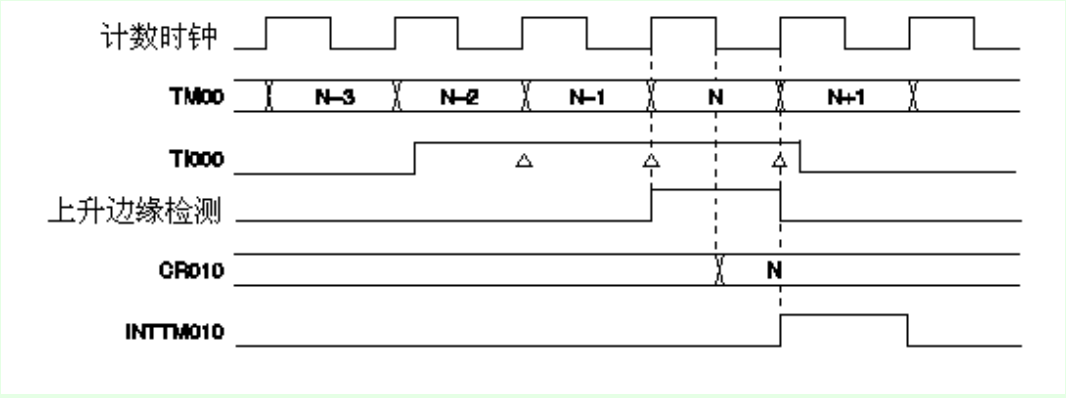
初始设置的主要内容如下。

- 选择高速内部振荡器作为系统时钟源[注]
- 停止看门狗计时器的运行
- 设置 V_{LVI}（低压检测电压）为 4.3V±0.2V
- 当 V_{DD}（供电电压）大于等于 V_{LVI} 后，一旦检测到 V_{DD} 小于 V_{LVI} 就会生成内部复位（LVI 复位）信号
- 设置 CPU 时钟频率为 8MHz
- 设置 I/O 端口
- 设置 16 位计时器/事件计数器 00
  - 分别将 CR000 和 CR010 的工作模式设置为比较寄存器和捕捉寄存器
  - 设置“FFFFH”给 CR000
  - 将 TI000 引脚的有效沿设置为下降及上升沿，将计数时钟设置为 $f_{XP}/2^2$（2MHz）
  - 将工作模式设置为在检测到 TI000 引脚的有效沿时清零并启动
- 启用 INTTM000 和 INTTM010 中断

注 用选项字节进行设置。

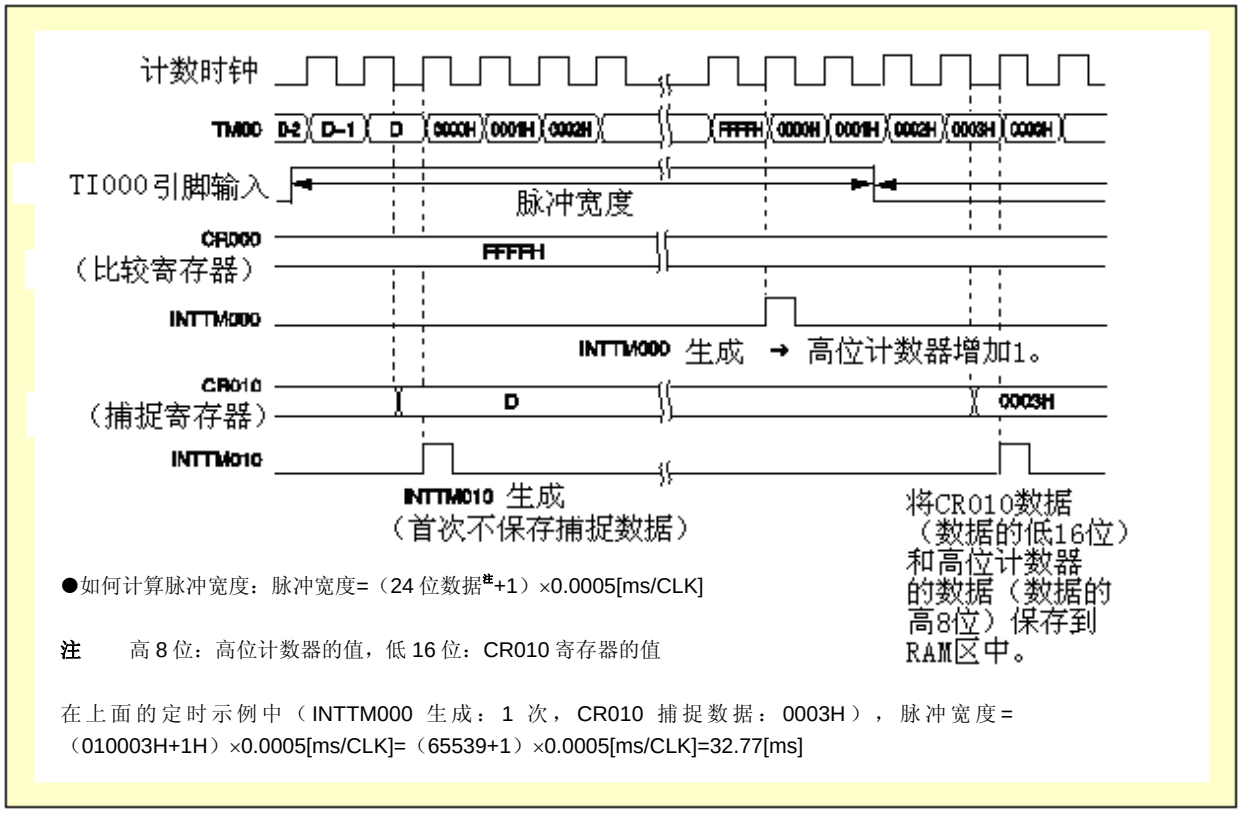
[列]捕捉操作定时

对于 16 位计时器/事件计数器 00，当两次检测到 TI000 引脚或 TI010 引脚的有效电平时进行第一次捕捉操作，以消除短脉冲宽度的噪声。因此，需要输入脉冲长度大于两个计数时钟。下图展示在指定为上升沿时捕捉 CR010 的操作示例。



1.2 主循环后的内容

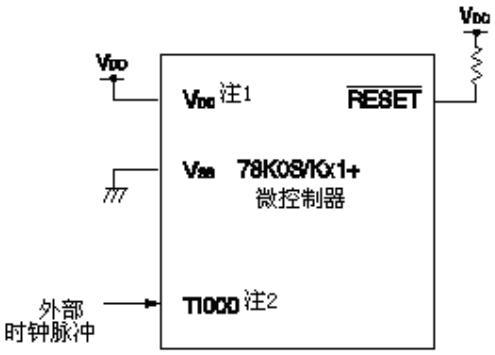
在初始设置完成后，利用 16 位计时器/事件计数器 00 中断（INTTM000 和 INTTM010）的生成，对 TI000 引脚输入信号的脉冲宽度进行八次测量。



注意事项 关于使用设备的注意事项，请参见各产品的用户手册（78K0S/KU1+、78K0S/KY1+、78K0S/KA1+、78K0S/KB1+）。

第二章 电路图

本章描述在本示例程序中所用的电路图。



注 **1.** 使用该电路的电压范围为 4.5V≤V_{DD}≤5.5V。

2. TI000/INTP0/P30： 78K0S/KA1+和 78K0S/KB1+微控制器

TI000/ANI0/TOH1/P20： 78K0S/KY1+和 78K0S/KU1+微控制器

注意事项 1. 将 AV_{REF} 引脚直接连接到 V_{DD}（仅适用于 78K0S/KA1+和 78K0S/KB1+微控制器）。

2. 将 AV_{SS} 引脚直接连接到 GND（仅适用于 78K0S/KB1+微控制器）。

3. 除电路图中的引脚及 AV_{REF}、AV_{SS} 引脚外，其他所有未使用的引脚保留开路（不连接）。

第三章 软件

本章描述所下载的压缩文件的文件配置、所用微控制器的内部外围功能以及示例程序的初始设置及运行概览，并展示流程图。

3.1 文件配置

下表展示所下载的压缩文件的文件配置。

文件名	描述	包含的压缩（*.zip）文件	
			
main.asm （汇编语言版） --- main.c （C语言版）	用于硬件初始化处理及微处理器主处理的源文件	●注	●注
op.asm	用于设置选项字节（设置系统时钟源）的汇编器源文件	●	●
tm00cap.prw	用于集成开发环境PM+的工作区		●
tm00cap.prj	用于集成开发环境PM+的项目文件		●

注 “main.asm”包含在汇编语言版中，“main.c”包含在 C 语言版中。

备注  ： 仅包含源文件。

 ：包含了与集成开发环境 PM+一起使用的文件。

3.2 所用内部外围功能

示例程序使用微处理器的下列内部外围功能。

- 脉冲宽度测量功能： 16 位计时器/事件计数器 00
- V_{DD}<V_{LVI} 检测： 低压检测器（LVI）
- 外部脉冲输入： TI000 [注]

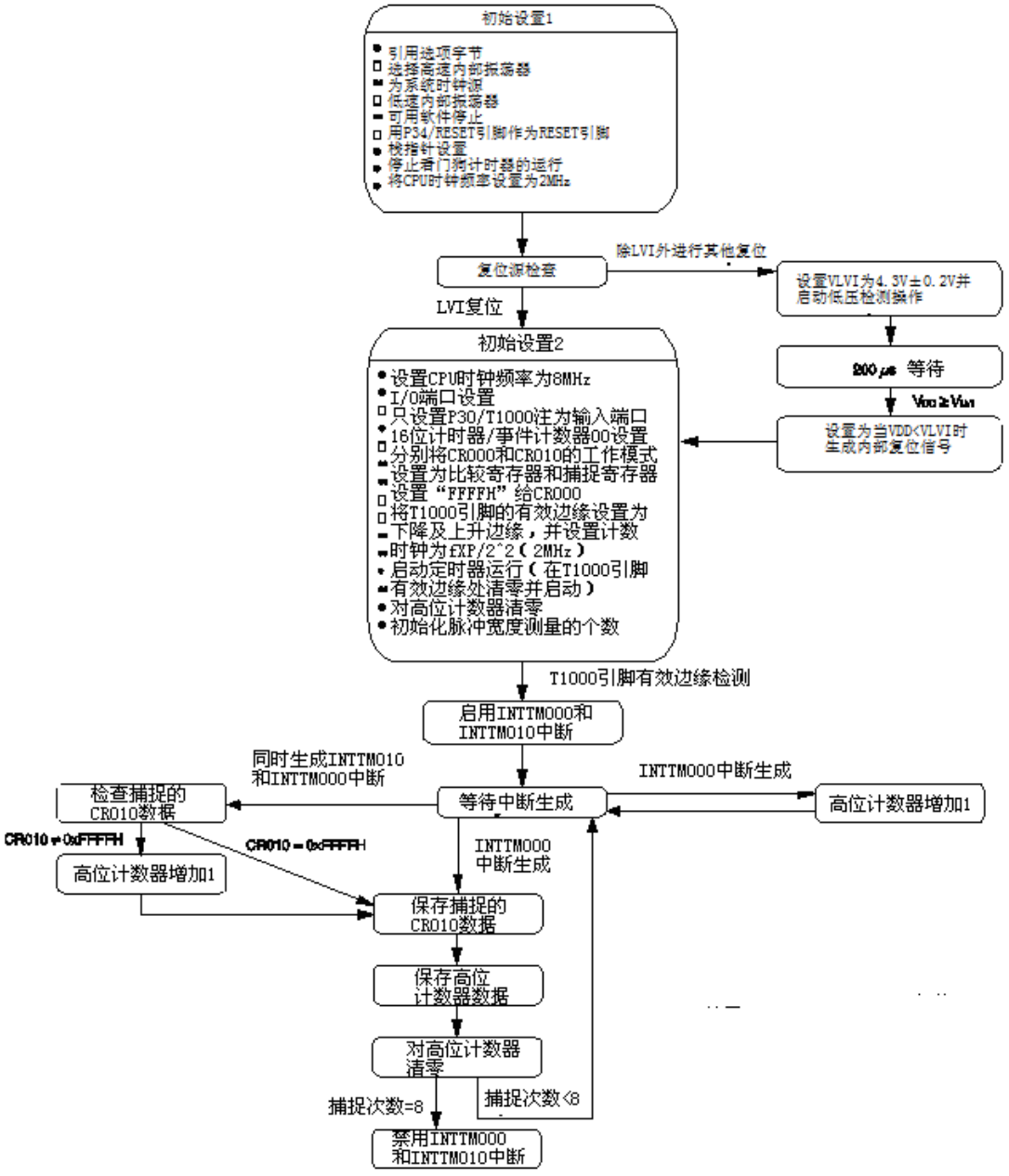
注 TI000/INTP0/P30： 78K0S/KA1+和 78K0S/KB1+微控制器
TI000/ANI0/TOH1/P20： 78K0S/KY1+和 78K0S/KU1+微控制器

3.3 初始设置及操作概览

在本示例程序进行的初始设置中包括了设置低压检测功能、选择时钟频率、设置 I/O 端口、设置 16 位计时器/事件计数器 00（脉冲宽度测量功能）及设置中断。

在初始设置完成后，利用 16 位计时器/事件计数器 00 中断（INTTM000 和 INTTM010）的生成，对 TI000 引脚输入信号的脉冲宽度进行八次测量。

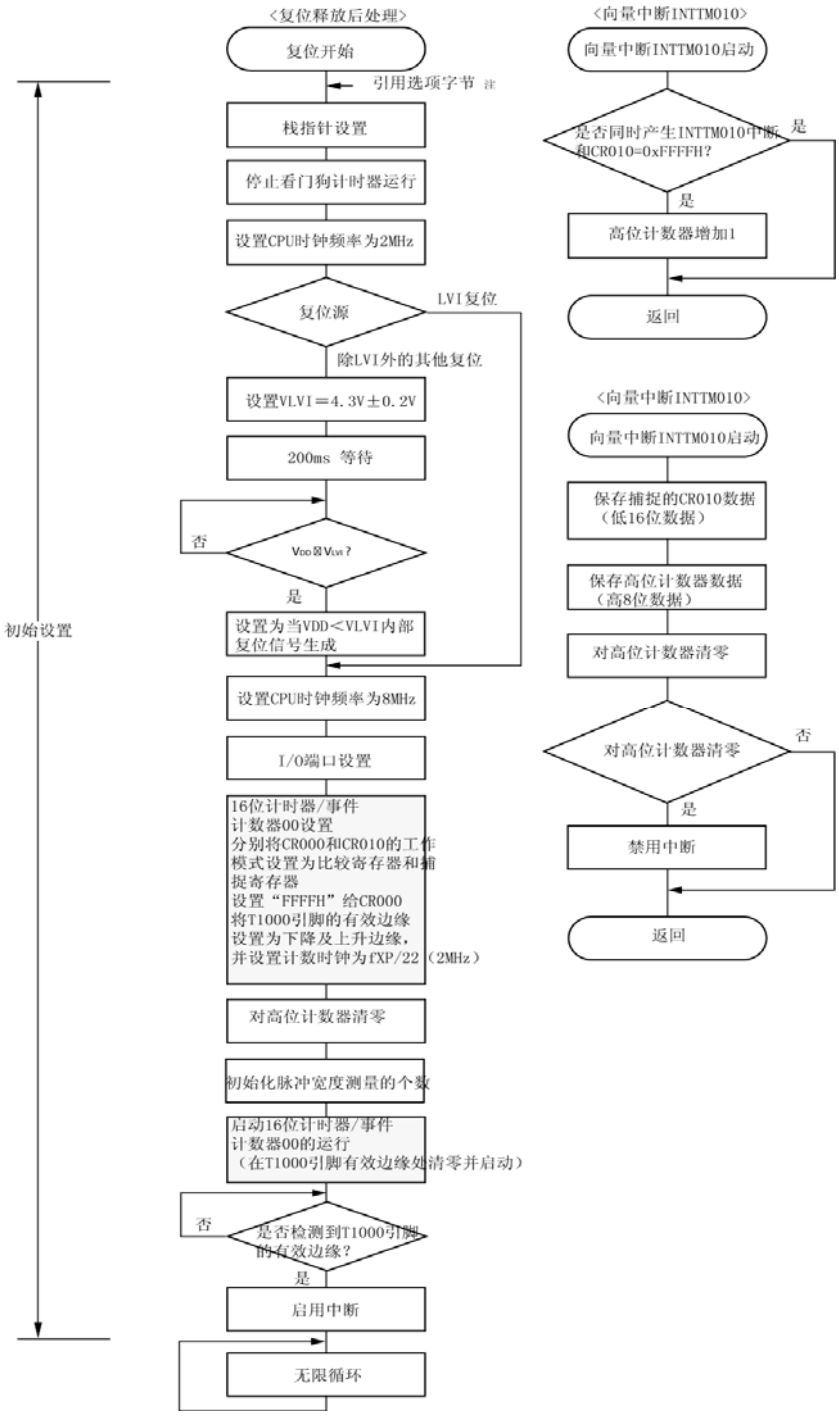
详情在如下所示的状态转变图中描述。



注 TI000/P30：78K0S/KA1+和 78K0S/KB1+微处理器
TI000/P20：78K0S/KY1+和 78K0S/KU1+微处理器

3.4 流程图

示例程序的流程图如下所示。



注 对选项字节的引用由微处理器在复位解除后自动进行。在本示例程序中，通过引用选项字节对下面的内容进行设置。

- 用高速内部时钟（8MHz（典型））做系统时钟源
- 利用软件可停止低速内部振荡器
- 用 P34/$\overline{\text{RESET}}$ 引脚做 $\overline{\text{RESET}}$ 引脚

第四章 设置方法

本章描述 16 位计时器/事件计数器 00 的脉冲宽度测量功能。

关于其他初始设置，请参见**78K0S/Kx1+示例程序（初始设置）LED照明开关控制的应用注释**。关于中断，请参见**78K0S/Kx1+示例程序（中断）由开关输入生成的外部中断的应用注释**。关于低压检查（LVI），请参见**78K0S/Kx1+示例程序（低压检测）检测到小于 2.7V时生成复位的应用注释**。

关于如何设置寄存器，请参见各设备（**78K0S/KU1+**、**78K0S/KY1+**、**78K0S/KA1+**、**78K0S/KB1+**）的用户手册。

关于汇编器指令，请参见**78K/0S系列指令用户手册**。

4.1 设置 16 位计时器/事件计数器 00 的脉冲宽度测量功能

在使用 16 位计时器/事件计数器 00 的脉冲宽度测量功能时，可设置下面的七类寄存器。

- 捕捉/比较控制寄存器 00（CRC00）
- 预换算器模式寄存器 00（PRM00）
- 16 位定时器模式控制寄存器 00（TMC00）
- 16 位定时器捕捉/比较寄存器 000（CR000）
- 16 位定时器捕捉/比较寄存器 010（CR010）
- 端口模式寄存器 x（PMx）注
- 端口模式控制寄存器 x（PMCx）注

注 应如下进行设置，因为脉冲宽度测量功能仅使用 TI000 引脚或 TI000 引脚和 TI010 引脚作为定时器输入。

•TI000 引脚

	PMx 寄存器	PMCx 寄存器
78K0S/KA1+和 78K0S/KB1+微处理器	PM30=1	无需设置
78K0S/KY1+和 78K0S/KU1+微处理器	PM20=1	PMC20=0

•TI010 引脚

	PMx 寄存器	PMCx 寄存器
78K0S/KA1+和 78K0S/KB1+微处理器	PM31=1	无需设置
78K0S/KY1+和 78K0S/KU1+微处理器	PM21=1	PMC21=0

<在使用 16 位计时器/事件计数器 00 进行脉冲宽度测量时的基本设置步骤示例>

<1> 设置 CRC00 寄存器

<2> 用 PRM00 寄存器设置计数时钟

<3> 设置 TMC00 寄存器：开始工作

注意事项 <1>步和<2>步可任意进行。

（1）设置 CRC00 寄存器

该寄存器控制 CR000 和 CR010 寄存器的工作。

图 4-1 捕捉/比较控制寄存器 00（CRC00）的格式

CRC00

0	0	0	0	0	CRC002	CRC001	CRC000

CR000工作模式选择

0	用作比较寄存器
1	用作捕捉寄存器

CR000捕捉触发选择

0	Ti010引脚有效沿捕捉。
1	Ti000引脚[注]有效沿反相时捕捉。

CR010工作模式选择

0	用作比较寄存器
1	用作捕捉寄存器

注 如果 CRC000 为 1，那么当选择下降及上升沿作为 TI000 引脚的有效沿时，CR000 寄存器无法进行捕捉操作。

注意事项 1. 定时器必须在设置 CRC00 寄存器之前停止操作。

2. 如果利用 TMC00 寄存器匹配 TM00 和 CR000 时选择了清零并开始模式（clear&start），就不要将 CR000 寄存器指定为捕捉寄存器。

3. 要确保捕捉操作的进行，捕捉触发所需脉冲的长度应大于预换算器模式寄存器 00（PRM00）所选计数时钟的 2 个周期。

（2）设置 PRM00 寄存器

该寄存器用于设置 TM00 计数器的计数时钟及 TI000、TI010 引脚输入的有效沿。

图 4-2　预换算器模式寄存器 00（PRM00）的格式

PRM00

ES110	ES100	ES010	ES000	0	0	PRM001	PRM000

计数时钟（fsam）选择

0	0	f_{XP}
0	1	$f_{XP}/2^2$
1	0	$f_{XP}/2^8$
1	1	TI000引脚有效沿[注]

TI000引脚有效沿选择

0	0	下降沿
0	1	上升沿
1	0	禁止设置
1	1	上升及下降沿

TI010引脚有效沿选择

0	0	下降沿
0	1	上升沿
1	0	禁止设置
1	1	上升及下降沿

注　外部时钟要求脉冲长度大于内部时钟（f_{XP}）的两个周期。

备注　f_{XP}：供给外围硬件的时钟的振荡频率

注意事项 1. 一定要在停止定时器工作后将数据设置给 PRM00 寄存器。

2. 在将 TI000 引脚的有效沿设置为计数时钟时，在设置清零并开始模式时不要将 TI000 引脚和 TI000 引脚的有效沿设置为捕捉触发。

3. 在下面的例子中，需小心注意的是检测到了 TI0n0 引脚（n=0、1）的有效沿。

<1> 高电平输入 TI0n0 引脚，在系统复位后 TM00 的运行立即启用。

→如果将上升沿或上升及下降沿指定为 TI0n0 引脚的有效沿，则在 TM00 运行启用后立即检测到上升沿。

<2> 当 TI0n0 引脚为高电平时，TM00 停止运行，然后当低电平输入 TI0n0 引脚时被启用。

→如果将下降沿或上升下降沿指定为 TI0n0 引脚的有效沿，则在 TM00 运行启用后立即检测到下降沿。

<3> 当 TI0n0 引脚为低电平时，TM00 停止运行，然后当高电平输入 TI0n0 引脚时被启用。

→如果将上升沿或上升及下降沿指定为 TI0n0 引脚的有效沿，则在 TM00 运行启用后立即检测到上升沿。

注意事项 4. 要用 TI000 的有效沿作为捕捉触发时，会用由预换算器模式寄存器 00（PRM00）选择的计数时钟进行采样以消除噪声。在采样有效沿之前不会进行捕捉操作，有效电平会被检测两次，从而消除较短脉冲宽度的噪声。

5. 当 TI010/TO00/Pxx 引脚用作有效沿的输入引脚（TI010）时，不能用作定时器输出引脚（TO00）。当用作定时器输出引脚（TO00）时，就不能用作有效沿的输入引脚（TI010）。

（3）设置 TMC00 寄存器

该寄存器设置 16 位计时器/事件计数器 00 的工作模式、TM00 计数器清零模式及输出定时，并对溢出进行检测。

图4-3 16位定时器模式控制寄存器00（TMC00）的格式

TMC00

0	0	0	0	TMC003	TMC002	TMC001	OVF00

16位定时器计数器00（TM00）溢出检测

0	未检测到溢出。
1	检测到溢出。

			工作模式和清零模式选择	TO00反相定时选择	中断请求生成
0	0	0	停止工作（TM00清为0）	无变化	不产生
0	0	1			
0	1	0	自由运行模式	Tm00、CR000匹配或TM00、CR010匹配	<用作比较寄存器> TM00、CR000匹配或TM00、CR010匹配时生成 <用作捕捉寄存器> TI000引脚或TI010引脚的有效沿处生成
0	1	1		Tm00、CR000匹配，TM00、CR010匹配或TI000引脚的有效沿	
1	0	0	Ti000引脚的有效沿处出现清零并启动	–	
1	0	1			
1	1	0	在TM00、CR000匹配时出现清零并启动	Tm00、CR000匹配或TM00、CR010匹配	
1	1	1		Tm00、CR000匹配，TM00、CR010匹配或TI000引脚的有效沿	

注意事项 1. 当非 0 值和 0（停止运行模式）分别设置给 TMC002 和 TMC003 时，TM00 计数器开始工作。要停止工作，应将 TMC002 和 TMC003 分别设置为 0 和 0。
2. 先停止定时器运行，再写入除 OVF00 标志的其他各位。
3. 在定时器停止时，即便有信号输入 TI000/TI010 引脚，也不会出现定时器计数和定时器中断。
4. 除非 TI000 引脚的有效沿选择为计数时钟，否则应先停止定时器工作再设置为 STOP 模式或系统时钟停止模式；否则，当系统时钟启动时定时器可能会发生故障。
5. 应先停止定时器工作再用 PRM00 寄存器的 4、5 位设置 TI000 引脚的有效沿。
6. 如果设置为在 TM00 和 CR000 匹配时或出现 TI000 引脚的有效沿时进入清零并开始模式，或者选择了自由运行模式，则当 CR000 寄存器的设置值为 FFFFH 且 TM00 计数器的值从 FFFFH 变为 0000H 时，OVF00 标志将被设置为 1。
7. 即便在 TM00 计数器溢出后计数到下一个计数时钟之前（TM00 计数器变为 0001H 前）OVF00 标志清零，它也会重新设置且禁止清零。
8. 捕捉操作在计数时钟的下降沿处进行。但是，中断请求（INTTM0n0：n=0、1）出现在下个计数时钟的上升沿处。

（4）设置 CR000 寄存器

该寄存器具有捕捉寄存器和比较寄存器二者的功能。

图 4-4 16 位定时器捕捉/比较寄存器 000（CR000）的格式

CR000

- 用 CR000 作为比较寄存器时

  设置给 CR000 的值不断与 16 位定时器计数器 00（TM00）的计数值进行比较，如果二者匹配就会生成中断请求（INTTM000）。
- 用 CR000 作为捕捉寄存器时

  Ti000 引脚或TI010 引脚的有效沿可选择为捕捉触发。Ti000 和TI010 引脚的有效沿通过**PRM00 寄存器**进行设置。

注意事项 1. 当由于 TM00 和 CR000 匹配而进入清零并开始模式时，应将非 0000H 值设置给 CR000 寄存器。当自由运行模式下 0000H 设置给 CR000，或者由于 TI000 引脚的有效沿而进入清零并开始模式时，在出现溢出（FFFFH）后当 0000H 变为 0001H 时会生成中断请求（INTTM000）。
2. 如果新的 CR000 寄存器值小于 TM00 计数器的值，则 TM00 计数器继续计数，溢出，然后重新从 0 开始计数。因此，如果新 CR000 寄存器值小于原值，在 CR000 寄存器的值改变后定时器必须复位并重新启动。
3. Tm00 计数器停止后 CR000 寄存器的值无法保证。
4. 对于设置为比较模式的 CR000 寄存器，即便输入了捕捉触发也可能不会进行捕捉操作。

注意事项 5. 当 CR000 用作捕捉寄存器时，如果寄存器读取时段与捕捉触发输入相冲突，将优先进行捕捉触发输入，CR000 读取数据变为未定义。如果定时器计数停止和捕捉触发输入相冲突，则捕捉数据变为未定义。
6. 在 TM00 计数器工作过程中改变 CR000 计数器的设置可能会导致故障。

（5）设置 CR010 寄存器

该寄存器具有捕捉寄存器和比较寄存器二者的功能。

图 4-5 16 位定时器捕捉/比较寄存器 010（CR010）的格式

CR010

•用 CR010 作为比较寄存器时

设置给 CR010 的值不断与 16 位定时器计数器 00（TM00）的计数值进行比较，如果二者匹配就会生成中断请求（INTTM010）。

•用 CR010 作为捕捉寄存器时

Ti000 引脚的有效沿可选择为捕捉触发。Ti000 引脚的有效沿通过**PRM00 寄存器**进行设置。

注意事项

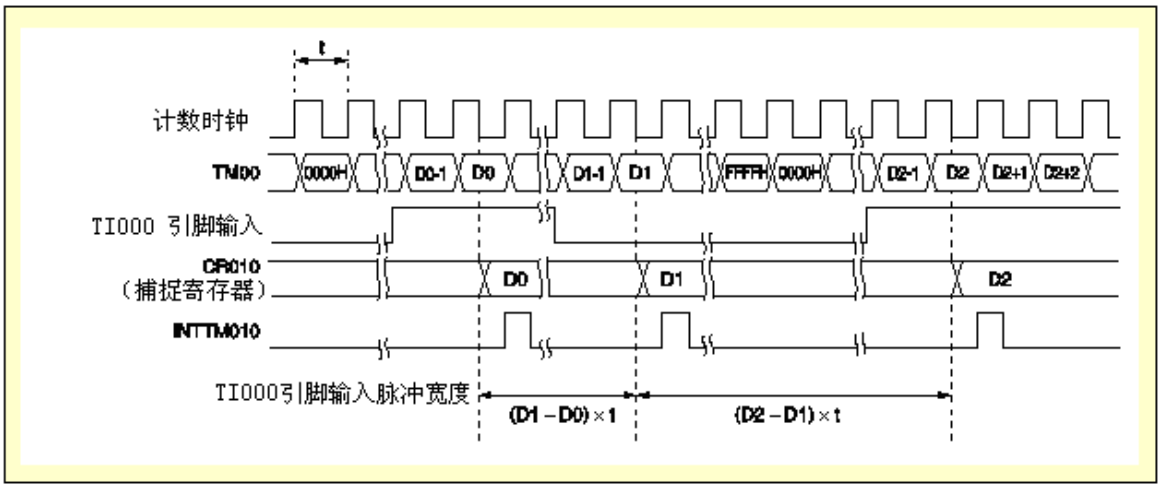
1. **当自由运行模式下 0000H 设置给 CR010，或者由于 TI000 引脚的有效沿而进入清零并开始模式时，在出现溢出（FFFFH）后当 0000H 变为 0001H 时会生成中断请求（（INTTM010）。**
2. **如果新的 CR0 10 寄存器值小于 TM00 计数器的值，则 TM00 计数器继续计数，溢出，然后重新从 0 开始计数。因此，如果新 CR010 寄存器值小于原值，在 CR010 寄存器的值改变后定时器必须复位并重新启动。**
3. **Tm00 计数器停止后 CR010 寄存器的值无法保证。**
4. **对于设置为比较模式的 CR010 寄存器，即便输入了捕捉触发也可能不会进行捕捉操作。**
5. **当 CR010 用作捕捉寄存器时，如果寄存器读取时段与捕捉触发输入相冲突，将优先进行捕捉触发输入，CR010 读取数据变为未定义。如果定时器计数停止和捕捉触发输入相冲突，则捕捉数据变为未定义。**
6. **在 TM00 计数器工作过程中改变 CR010 计数器的设置可能会导致故障。**

[例1] 测量TI000引脚输入信号的脉冲宽度（用CR010寄存器作为捕捉寄存器，自由运行模式）

当TM00计数器工作在自由运行模式下时，测量输入TI000引脚信号的脉冲宽度。当检测到TI000引脚的有效沿时，TM00计数器的计数值被捕捉并放入CR010寄存器中。

注意事项 本操作示例中可测量的脉冲宽度最大为一个定时器计数器周期。

图 4-6 测量 TI000 引脚输入信号的脉冲宽度的定时示例（自由运行模式、指定为双沿）



（1）寄存器设置

<1> CRC00

0	0	0	0	0	1	x	0

CR000工作模式选择

0	用作比较寄存器

CR000捕捉触发选择

x	无效设置

CR010工作模式选择

1	用作捕捉寄存器

<2> PRM00

x	x	0/1	0/1	0	0	0/1	0/1

计数时钟（fsam）选择（禁止设置为“1, 1”。）

0	0	f_{XP}
0	1	$f_{XP}/2^2$
1	0	$f_{XP}/2^8$

TI000引脚有效沿选择（禁止设置为“1, 0”。）

0	0	下降沿
0	1	上升沿
1	1	上升及下降沿

TI010引脚有效沿选择

x	x	无效设置（禁止设置“1, 0”。）

<3> TMC00

0	0	0	0	0	1	x	0

16位定时器计数器00（TM00）溢出检测

0	未检测到溢出。

工作模式和清零模式选择

0	1	0	自由运行模式
0	1	1	

<4> PMx、PMCx

	PMx寄存器	PMCx寄存器
78K0S/KA1+和78K0S/KB1+微处理器	PM30=1	无需设置
78K0S/KY1+和78K0S/KU1+微处理器	PM20=1	PMC20=0

（2）示例程序

在下面的例子中，“**（1）寄存器设置**”中的“x”设置为“0”。此外，TI000 引脚的有效沿设置为双沿且计数时钟设置为 f_{XP}（系统时钟频率）。

<1> 汇编语言（使用 78K0S/KA1+和 78K0S/KB1+微控制器时）

```
SET1   PM3.0
MOV    CRC00, #00000100B
MOV    PRM00, #00110000B
MOV    TMC00, #00000100B
```

<2> C 语言（使用 78K0S/KA1+和 78K0S/KB1+微控制器时）

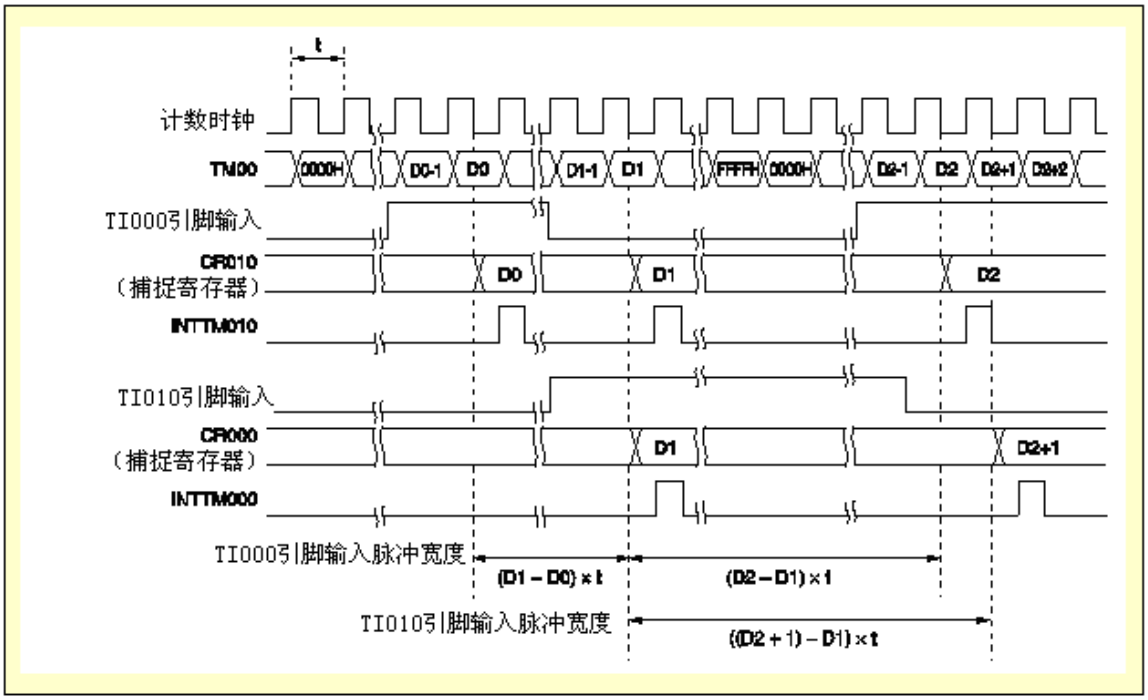
```
PM3.0 = 1;
CRC00 = 0b00000100;
PRM00 = 0b00110000;
TMC00 = 0b00000100;
```

[例2] 测量TI000引脚和TI010引脚的输入信号的脉冲宽度（用CR000寄存器和CR010寄存器作为捕捉寄存器，自由运行模式）

当TM00计数器工作在自由运行模式下时，同时测量输入TI000引脚和TI010引脚的两个信号的脉冲宽度。当检测到TI000引脚的有效沿时，TM00计数器的计数值被捕捉进入CR010寄存器；当检测到TI010引脚的有效沿时，TM00计数器的计数值被捕捉进入CR000寄存器。

注意事项 本操作示例中可测量的脉冲宽度最大为一个定时器计数器周期。

图 4-7 测量 TI000 引脚和 TI010 引脚输入信号的脉冲宽度的定时示例（自由运行模式、指定为双沿）



（1）寄存器设置

<1> CRC00

0	0	0	0	0	1	0	1

CR000工作模式选择

1	用作捕捉寄存器

CR000捕捉触发选择

0	Ti010引脚有效沿捕捉。

CR010工作模式选择

1	用作捕捉寄存器

<2> PRM00

0/1	0/1	0/1	0/1	0	0	0/1	0/1

计数时钟（fsam）选择（禁止设置为“1, 1”。）

0	0	f_{XP}
0	1	$f_{XP}/2^2$
1	0	$f_{XP}/2^8$

TI000引脚有效沿选择（禁止设置为“1, 0”。）

0	0	下降沿
0	1	上升沿
1	1	上升及下降沿

TI010引脚有效沿选择（禁止设置为“1, 0”。）

0	0	下降沿
0	1	上升沿
1	1	上升及下降沿

<3> TMC00

0	0	0	0	0	1	x	0

16位定时器计数器00（TM00）溢出检测

0	未检测到溢出。

工作模式和清零模式选择

0	1	0	自由运行模式
0	1	1	

<4> PMx、PMCx

	PMx寄存器	PMCx寄存器
78K0S/KA1+和78K0S/KB1+微处理器	PM30=1、 PM31=1	无需设置
78K0S/KY1+和78K0S/KU1+微处理器	PM20=1、 PM21=1	PMC20=0、 PMC21=0

（2）示例程序

在下面的例子中，“**（1）寄存器设置**”中的“x”设置为“0”。此外，TI000 引脚和 TI010 引脚的有效沿设置为双沿且计数时钟设置为 f_{XP}（系统时钟频率）。

<1> 汇编语言（使用 78K0S/KA1+和 78K0S/KB1+微控制器时）

```
SET1   PM3.0
SET1   PM3.1
MOV    CRC00, #00000101B
MOV    PRM00, #11110000B
MOV    TMC00, #00000100B
```

<2> C 语言（使用 78K0S/KA1+和 78K0S/KB1+微控制器时）

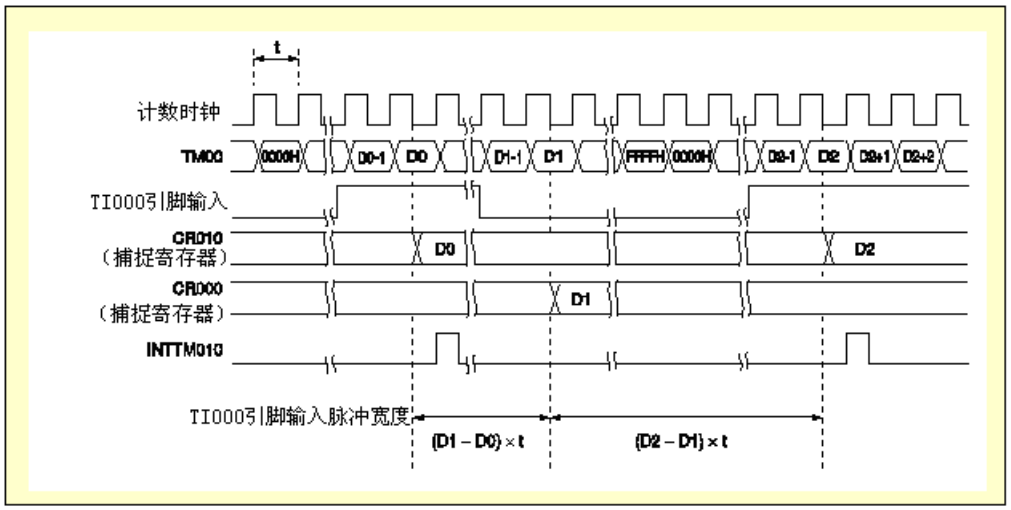
```
PM3.0 = 1;
PM3.1 = 1;
CRC00 = 0b00000101;
PRM00 = 0b11110000;
TMC00 = 0b00000100;
```

[例3] 测量TI000引脚的输入信号的脉冲宽度（用CR000寄存器和CR010寄存器作为捕捉寄存器，自由运行模式）

当TM00计数器工作在自由运行模式下时，测量输入TI000引脚信号的脉冲宽度。当检测到TI000引脚的有效沿时，TM00计数器的计数值被捕捉进入CR010寄存器；当出现与检测到TI000引脚的有效沿反相的信号时，TM00计数器的计数值被捕捉进入CR000寄存器。将TI000引脚的有效沿检测设置为上升沿或下降沿。

注意事项 本操作示例中可测量的脉冲宽度最大为一个定时器计数器周期。

图 4-8 测量 TI000 引脚输入信号的脉冲宽度的定时示例（自由运行模式、指定为上升沿）



（1）寄存器设置

<1> CRC00

0	0	0	0	0	1	1	1

CR000工作模式选择

1	用作捕捉寄存器

CR000捕捉触发选择

1	Ti000引脚有效沿反相时捕捉。注

CR010工作模式选择

1	用作捕捉寄存器

注 如果 CRC000 为 1，那么当选择下降及上升沿作为 TI000 引脚的有效沿时，CR000 寄存器无法进行捕捉操作。如果 CRC001 为 1，则无法由 TI010 引脚的有效沿捕捉进入 CR000 寄存器；但是，因为生成了 INTTM000，不能用 TI010 引脚作为外部中断。

<2> PRM00

x	x	0	0/1	0	0	0/1	0/1

计数时钟（fsam）选择（禁止设置为“1, 1”。）

0	0	f_{XP}
0	1	$f_{XP}/2^2$
1	0	$f_{XP}/2^8$

TI000引脚有效沿选择

0	0	下降沿
0	1	上升沿

TI010引脚有效沿选择

x	x	无效设置（禁止设置“1, 0”。）

<3> TMC00

0	0	0	0	0	1	x	0

16位定时器计数器00（TM00）溢出检测

0	未检测到溢出。

工作模式和清零模式选择

0	1	0	自由运行模式
0	1	1	

<4> PMx、PMCx

	PMx寄存器	PMCx寄存器
78K0S/KA1+和78K0S/KB1+微处理器	PM30=1	无需设置
78K0S/KY1+和78K0S/KU1+微处理器	PM20=1	PMC20=0

（2）示例程序

在下面的例子中，“**（1）寄存器设置**”中的“x”设置为“0”。此外，TI000 引脚的有效沿设置为上升沿且计数时钟设置为 f_{XP}（系统时钟频率）。

<1> 汇编语言（使用 78K0S/KA1+和 78K0S/KB1+微控制器时）

```
SET1   PM3.0
MOV    CRC00, #00000111B
MOV    PRM00, #00010000B
MOV    TMC00, #00000100B
```

<2> C 语言（使用 78K0S/KA1+和 78K0S/KB1+微控制器时）

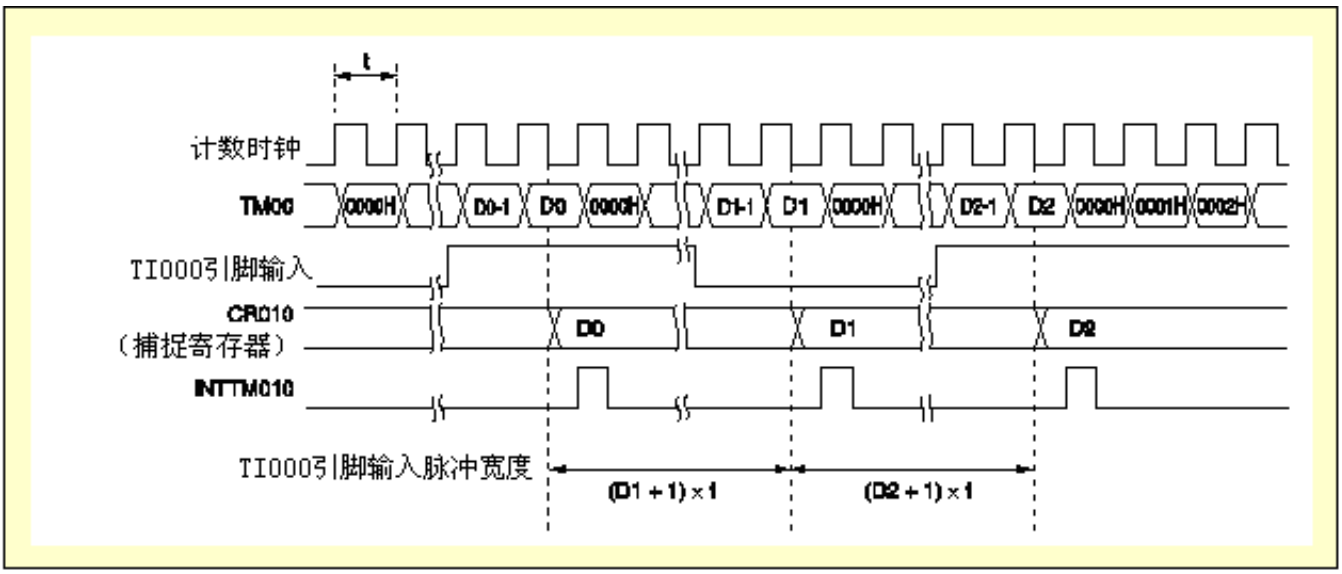
```
PM3.0 = 1;
CRC00 = 0b00000111;
PRM00 = 0b00010000;
TMC00 = 0b00000100;
```

[例4] 测量TI000引脚输入信号的脉冲宽度（用CR000寄存器作为捕捉寄存器，通过输入TI000引脚的有效沿进入清零并开始模式）

通过输入TI000引脚的有效沿进入清零并开始模式时，当TM00计数器工作时，将对TI000引脚输入信号的脉冲宽度进行测量。检测到TI000引脚的有效沿时，TM00计数器的计数值将被捕捉进入CR010寄存器，之后，TM00计数器清零，计数重新开始，对TI000引脚的输入信号的脉冲宽度进行测量。

注意事项 本操作示例中可测量的脉冲宽度最大为一个定时器计数器周期。

图 4-9 测量 TI000 引脚输入信号的脉冲宽度的定时示例（通过 TI000 引脚的有效沿输入进入清零并开始模式，指定为双沿）



（1）寄存器设置

<1> CRC00

0	0	0	0	0	1	x	0

CR000工作模式选择

0	用作比较寄存器

CR000捕捉触发选择

x	无效设置

CR010工作模式选择

1	用作捕捉寄存器

<2> PRM00

x	x	0/1	0/1	0	0	0/1	0/1

计数时钟（fsam）选择（禁止设置为“1, 1”。）

0	0	f_{XP}
0	1	$f_{XP}/2^2$
1	0	$f_{XP}/2^8$

TI000引脚有效沿选择（禁止设置为“1, 0”。）

0	0	下降沿
0	1	上升沿
1	1	上升及下降沿

TI010引脚有效沿选择

x	x	无效设置（禁止设置“1, 0”。）

<3> TMC00

0	0	0	0	1	0	x	0

16位定时器计数器00（TM00）溢出检测

0	未检测到溢出。

工作模式和清零模式选择

1	0	0	Ti000引脚的有效沿处出现清零并启动
1	0	1	

<4> PMx、PMCx

	PMx寄存器	PMCx寄存器
78K0S/KA1+和78K0S/KB1+微处理器	PM30=1	无需设置
78K0S/KY1+和78K0S/KU1+微处理器	PM20=1	PMC20=0

（2）示例程序

在下面的例子中，“**（1）寄存器设置**”中的“x”设置为“0”。此外，TI000 引脚的有效沿设置为双沿且计数时钟设置为 f_{XP}（系统时钟频率）。

<1> 汇编语言（使用 78K0S/KA1+和 78K0S/KB1+微控制器时）

```
SET1   PM3.0
MOV    CRC00, #00000100B
MOV    PRM00, #00110000B
MOV    TMC00, #00001000B
```

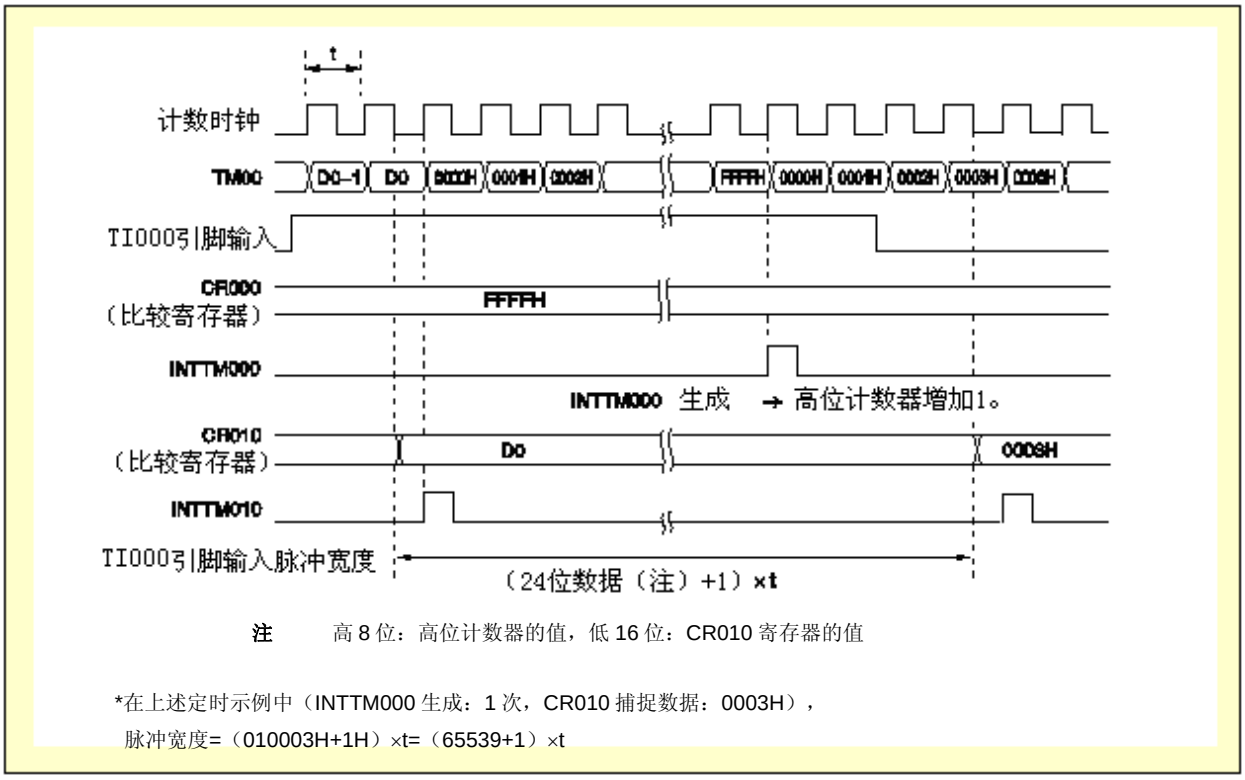
<2> C 语言（使用 78K0S/KA1+和 78K0S/KB1+微控制器时）

```
PM3.0 = 1;
CRC00 = 0b00000100;
PRM00 = 0b00110000;
TMC00 = 0b00001000;
```

[例5] 测量的TI000引脚输入信号的脉冲宽度大于TM00计数器的周期（用CR010寄存器作为捕捉寄存器、CR000寄存器作为比较寄存器，通过输入TI000引脚的有效沿进入清零并开始模式）
（与示例程序源内容相同）

通过输入TI000引脚的有效沿进入清零并开始模式时，当TM00计数器工作时，将对TI000引脚输入信号的脉冲宽度进行测量。
当CR000寄存器的工作模式设置为比较寄存器且CR000设置为FFFFH时，CR000寄存器和TM00计数器的值进行匹配，当TM00计数器从FFFFH变为0000H时将生成INTTM000中断。通过利用该中断作为溢出中断并在生成INTTM000中断时对高位进行向上计数，可测量大于TM00计数器周期的脉冲的宽度。
检测到TI000引脚的有效沿时，TM00计数器的计数值将被捕捉进入CR010寄存器，之后，TM00计数器清零，计数重新开始，对TI000引脚的输入信号的脉冲宽度进行测量。

图 4-10 测量 TI000 引脚输入信号的脉冲宽度的定时示例，该脉冲的宽度大于 TM00 的计数周期（通过 TI000 引脚的有效沿输入进入清零并开始模式，指定为双沿）



注 高 8 位：高位计数器的值，低 16 位：CR010 寄存器的值

*在上述定时示例中（INTTM000 生成：1 次，CR010 捕捉数据：0003H），
脉冲宽度=（010003H+1H）×t=（65539+1）×t

（1）寄存器设置

<1> CRC00

0	0	0	0	0	1	x	0

CR000工作模式选择

0	用作比较寄存器

CR000捕捉触发选择

x	无效设置

CR010工作模式选择

1	用作捕捉寄存器

<2> PRM00

x	x	0/1	0/1	0	0	0/1	0/1

计数时钟（fsam）选择（禁止设置为“1, 1”。）

0	0	f_{XP}
0	1	$f_{XP}/2^2$
1	0	$f_{XP}/2^8$

TI000引脚有效沿选择（禁止设置为“1, 0”。）

0	0	下降沿
0	1	上升沿
1	1	上升及下降沿

TI010引脚有效沿选择

x	x	无效设置（禁止设置“1, 0”。）

<3> TMC00

0	0	0	0	1	0	x	0

16位定时器计数器00（TM00）溢出检测

0	未检测到溢出。

工作模式和清零模式选择

1	0	0	在TI000引脚的有效沿出现清零并启动
1	0	1	

<4> PMx、PMCx

	PMx寄存器	PMCx寄存器
78K0S/KA1+和78K0S/KB1+微处理器	PM30=1	无需设置
78K0S/KY1+和78K0S/KU1+微处理器	PM20=1	PMC20=0

<5> CR000：FFFFH

（2）示例程序

在下面的例子中，“**（1）寄存器设置**”中的“x”设置为“0”。此外，TI000 引脚的有效沿设置为双沿且计数时钟（fsam）设置为 $f_{XP}/2^2$。

<1> 汇编语言（使用 78K0S/KA1+和 78K0S/KB1+微控制器时）

```
SET1   PM3.0
MOV    CRC00, #00000100B
MOVW   AX, #0FFFFH
MOVW   CR000, AX
MOV    PRM00, #00110001B
MOV    TMC00, #00001000B
```

<2> C 语言（使用 78K0S/KA1+和 78K0S/KB1+微控制器时）

```
PM3.0 = 1;
CRC00 = 0b00000100;
CR000 = 0xFFFF;
PRM00 = 0b00110001;
TMC00 = 0b00001000;
```

[该示例程序源文件节选]

如下所示为附件A程序列表节选，与 16 位计时器/事件计数器 00 功能相关（与上面提到的[例 5]内容相同）。

（1）汇编语言

```
XMAIN  CSEG   UNIT
RESET_START:
                    •
                    •
                    •
        MOV     CRC00, #00000100B   ; 用 CR010 作为捕捉寄存器
        MOVW    AX,    #0FFFFH
        MOVW    CR000, AX           ; 用 CR000 进行溢出检测
        MOV     PRM00, #00110001B   ; 指定 TI000 引脚的有效沿为双沿，设置计数时钟为 fxp/4
        MOV     TOC00, #00000000B   ; 不进行定时器输出

        MOV     MK0,   #0FFH        ; 首先屏蔽所有的中断
        MOV     IF0,   #00H         ; 事先清除无效中断请求
                    •
                    •
                    •
        MOV     TMC00, #00001000B   ; 启动定时器的工作（通过 TI000 引脚的有效沿清零并启动）
WAITCAP:
        NOP
        BF      TMIF010, $WAITCAP   ; 等待第一次捕捉的完成

        MOV     IF0,   #00H         ; 中断请求标志清零
        CLR1    TMMK000             ; INTTM000 中断去屏蔽
        CLR1    TMMK010             ; INTTM010 中断去屏蔽

        EI;                         ; 启用向量中断

MAIN_LOOP:
        NOP
        BR      $MAIN_LOOP          ; 转到 MAIN_LOOP

INTERRUPT_TM000:
        PUSH    AX                  ; 将 AX 寄存器的数据保存到
                    •
                    •
                    •
        INC     HIGHCOUNT           ; 高位计数增加 1
                    •
                    •
                    •
INTERRUPT_TM010:
        PUSH    AX                  ; 将 AX 寄存器的数据保存到栈中
        MOVW    AX,    CR010        ; 读取捕捉的数据
                    •
                    •
                    •
        MOV     A,     HIGHCOUNT    ; 读取高位数据
                    •
                    •
                    •
end
```

设置 CR010 的工作模式为捕捉寄存器，设置 CR000 的工作模式为比较寄存器

设置 FFFFH 给 CR000

设置计数时钟和 TI000 引脚的有效沿

启动定时器的运行

INTTM000 和 INTTM010 中断请求标志清零

启用 INTTM000 中断服务

启用 INTTM010 中断服务

由 INTTM000 中断生成启动中断服务

高位计数器增加 1

由 INTTM010 中断生成启动中断服务

读取 CR010 寄存器的捕捉数据

读取高位计数器

（2） C 语言

```
void hdwinit(void){
        unsigned char ucCnt200us;          /* 用于 200 us 等待的 8 位变量*/
                        •
                        •
                        •
        CRC00 = 0b00000100;                /* 用于捕捉寄存器 */
        CR000 = 0xFFFF;                    /* CR000 用于溢出检测 */
        PRM00 = 0b00110001;                /* 指定 TI000 引脚的有效沿为双沿，设置计数时钟为 fxp/4
*/
        TOC00 = 0b00000000;                /* 不进行定时器输出 */

        MK0   = 0xFF;                      /* 首先屏蔽所有中断 */
        IF0   = 0x00;                      /* 清除无效中断请求 */

        return;
}

void main(void){
        g_ucHighCount = 0;                 /* 高位寄存器清零 */
        g_ucTimes = 8;                     /* 初始化测量次数 */
        TMC00 = 0b00001000;                /* 启动定时器的运行（在 TI000 引脚的有效沿处清零并启动
*/

        while(TMIF010==0){                 /* 等待第一次捕捉的完成 */
                NOP();
        }

        IF0 = 0x00;                        /* 中断请求标志清零 */
        TMMK000 = 0;                       /* INTTM000 中断去屏蔽 */
        TMMK010 = 0;                       /* INTTM010 中断去屏蔽 */

        EI();                              /* 启用向量中断 */

        while(1){                          /* 无限循环 */
                NOP();
                NOP();
        }
}

__interrupt void fn_inttm000(){
        if (!(TMIF010 && (CR010 == 0xFFFF))){   /* 如果中断同时生成且 CR010 == 0xFFFF，就不计
数 */
                g_ucHighCount++;           /* 高位计数增加 1 */
        }

        return;
}
__interrupt void fn_inttm010(){
        g_unLowLength[8 - g_ucTimes] = CR010;   /* 读取捕捉的数据 */
        g_unHighLength[8 - g_ucTimes] = g_ucHighCount; /* 读取高位数据 */
                        •
                        •
                        •
        return;
}
```

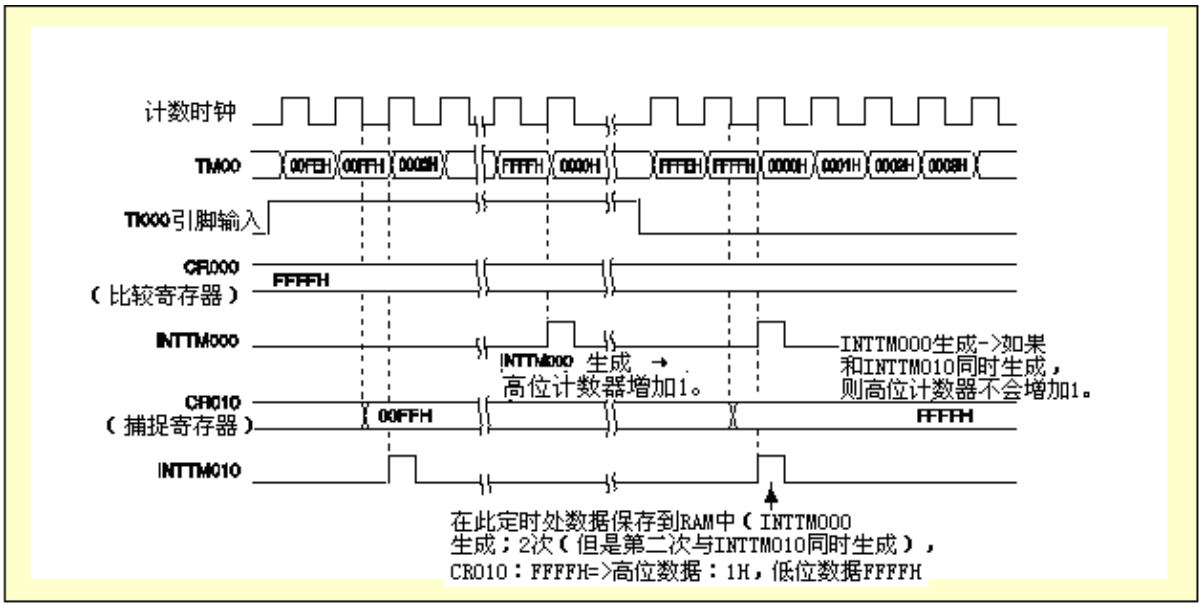
4.2 当INTTM000 和INTTM010 中断生成相冲突时的定时（测量脉冲宽度大于TM00 计数器的周期）

在本示例程序中，CR010 和CR000 寄存器的工作模式分别设置为捕捉寄存器和比较寄存器，CR000 寄存器设置为FFFFH。设置为由TI000 引脚的有效沿进入清零并开始模式（见上面的[例 5]）。

这样的设置使得 CR000 寄存器和 TM00 计数器的值相匹配，且当 TM00 计数器从 FFFFH 变为 0000H 时生成INTTM000 中断。利用该中断作为溢出中断并在 INTTM000 中断生成时对高位进行向上计数，可以测量大于 TM00 计数器周期的脉冲宽度。

当 INTTM000 中断与 INTTM010 中断同时生成时，INTTM000 中断服务优先。这时，CR010 寄存器的捕捉数据变为FFFFH，高位不需要向上计数。因此，高位不会向上计数，且当 INTTM000 中断与 INTTM010 中断同时生成时，高位数据和 CR010 寄存器的捕捉数据（FFFFH）保存到 RAM 区中。

图 4-11 当 INTTM000 和 INTTM010 中断冲突时的定时示例（由 TI000 引脚有效沿输入进入清零并开始模式，指定为双沿）



第五章 用设备进行运行检查

本章描述利用下载的示例程序从构建到用设备进行运行检查的流程。

5.1 构建示例程序

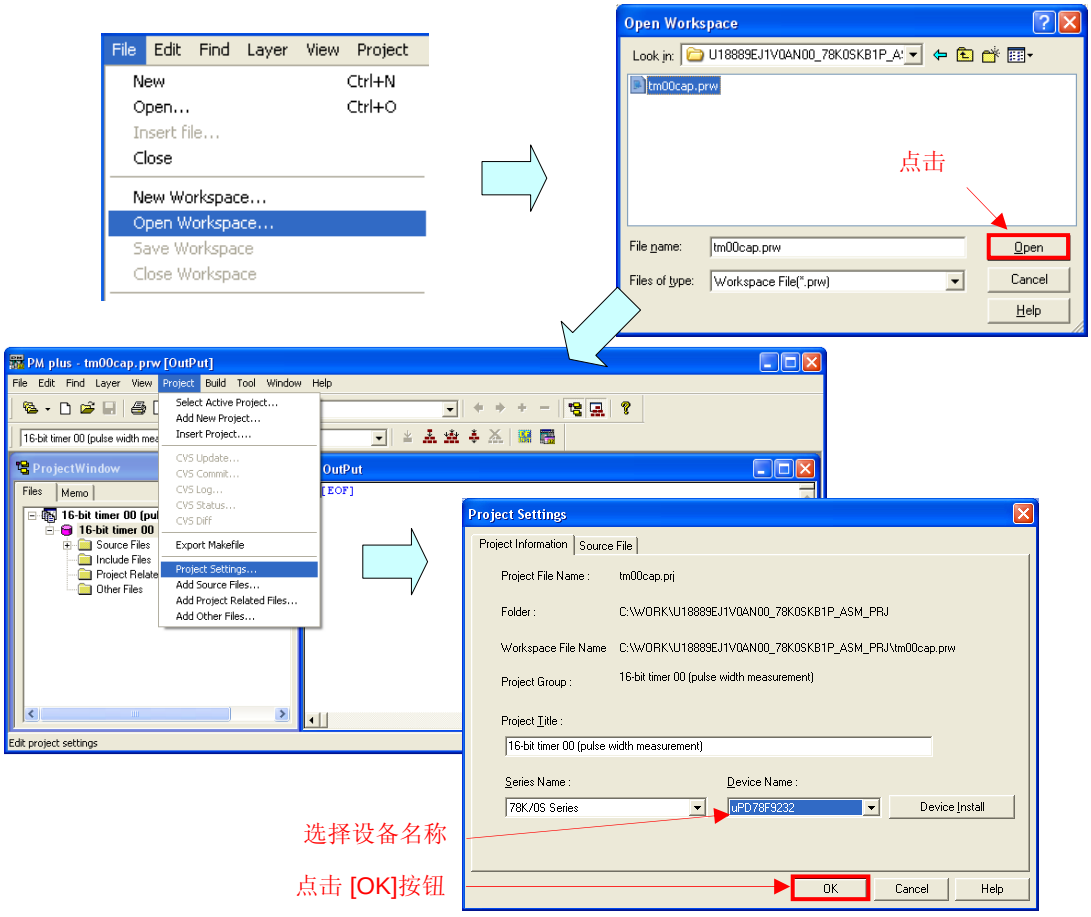
点击  图标下载示例程序（源文件+项目文件），本节描述如何用该程序构建各示例程序。关于如何构建其他下载的程序，请参见**78K0S/Kx1+示例程序启动向导的应用注释**的“**第三章注册集成开发环境PM+项目并执行构建**”。

关于如何操作PM+的详情，请参见**PM+项目管理器用户手册**。

（1）启动 PM+。

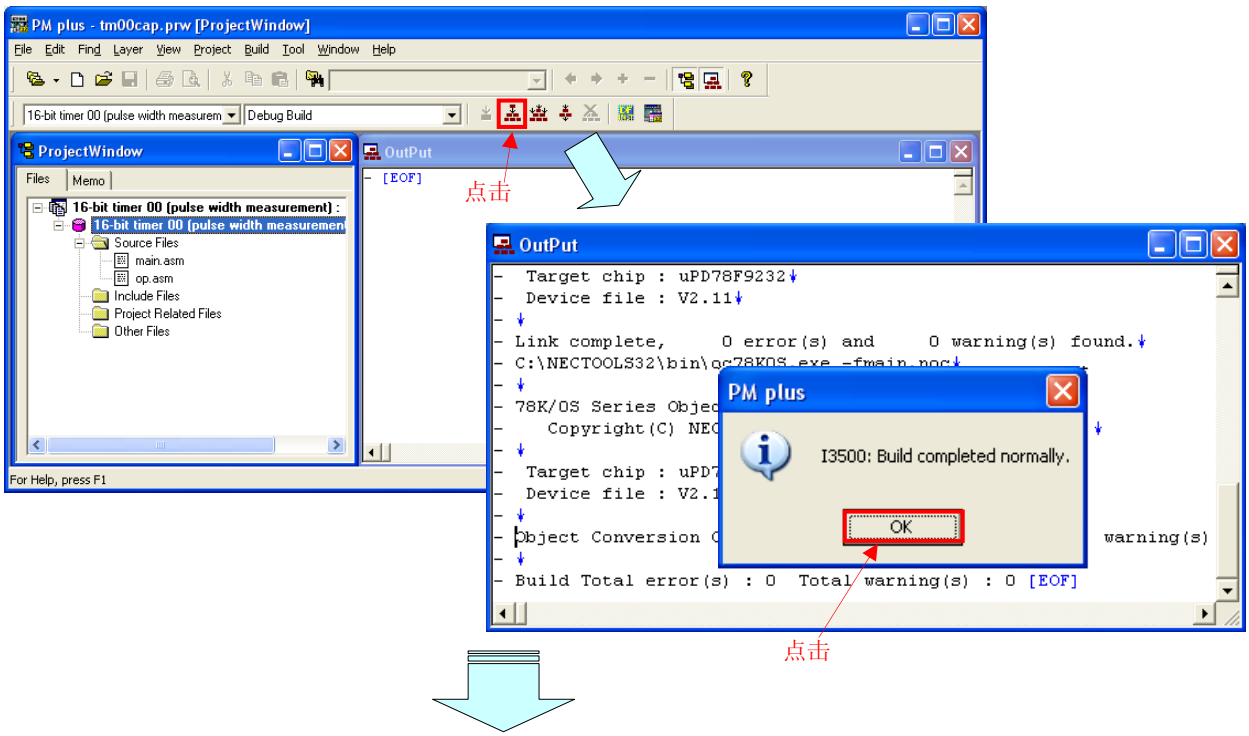
（2）在[文件]菜单中点击[打开]，点击[打开工作区]，选择“tm00cap.prw”。工作区生成，源文件将自动放入该工作区中。

（3）从[项目]菜单选择[项目设置]。[项目设置]窗口打开后，选择要用的设备的名称（默认选择具有最大 ROM 或 RAM 的设备），点击[OK]。



（4）点击[构建]按钮）。当“main.asm”和“op.asm”源文件正常构建时，将显示消息“I3500：Build completed normally.（构建正常完成）”。

（5）在消息窗口中点击[OK]按钮。将创建用于闪存写入的 HEX 文件。



生成用于闪存写入的 HEX 文件。 → 见 **5.2**.

[列] 构建错误

在用 PM+进行构建时，如果显示错误消息“A006File not found‘C：\NECTOOLS32\LIB78K0S\s0sl.rel’（文件未找到）”或“***ERROR F206Segment‘@@DATA’can’t allocate to memory-ignored.（片段‘@@DATA’无法放入存储器中——忽略）”，应按照下面的步骤改变编译器的选项设置。

<1>从[工具]菜单选择[编译器选项]。
<2>显示[编译器选项]对话框。选择[启动例程]标签。
<3>不选[使用标准库的固定区域]复选框。（其他复选框不动。）

当[使用标准库的固定区域]复选框不选时，将保留 118 字节的 RAM 区作为固定标准库区供使用；但是，标准库（如 getchar 函数和 malloc 函数）会被禁用。

点击 图标下载的文件在本示例程序中使用时，默认不选[使用标准库的固定区域]复选框。

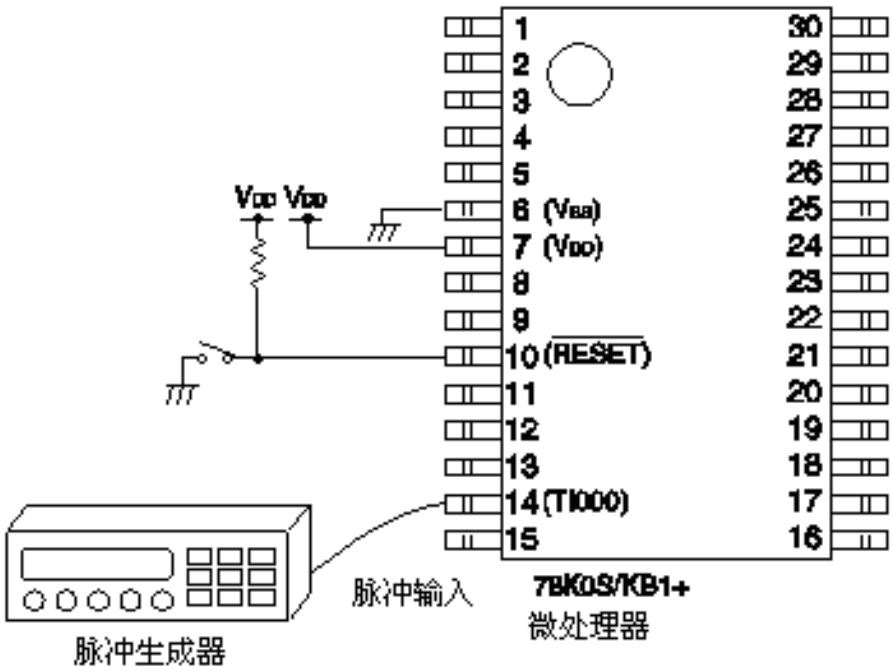
5.2　带设备运行

本节描述用设备进行运行检查的示例。

由执行构建生成的 HEX 文件可写入设备的闪存。

关于如何写入设备的闪存，请参见“**78K0S/Kx1+简化闪存写入手册信息**”（准备中）。

关于如何连接设备及所用外围硬件（脉冲发生器）的示例如下所示。



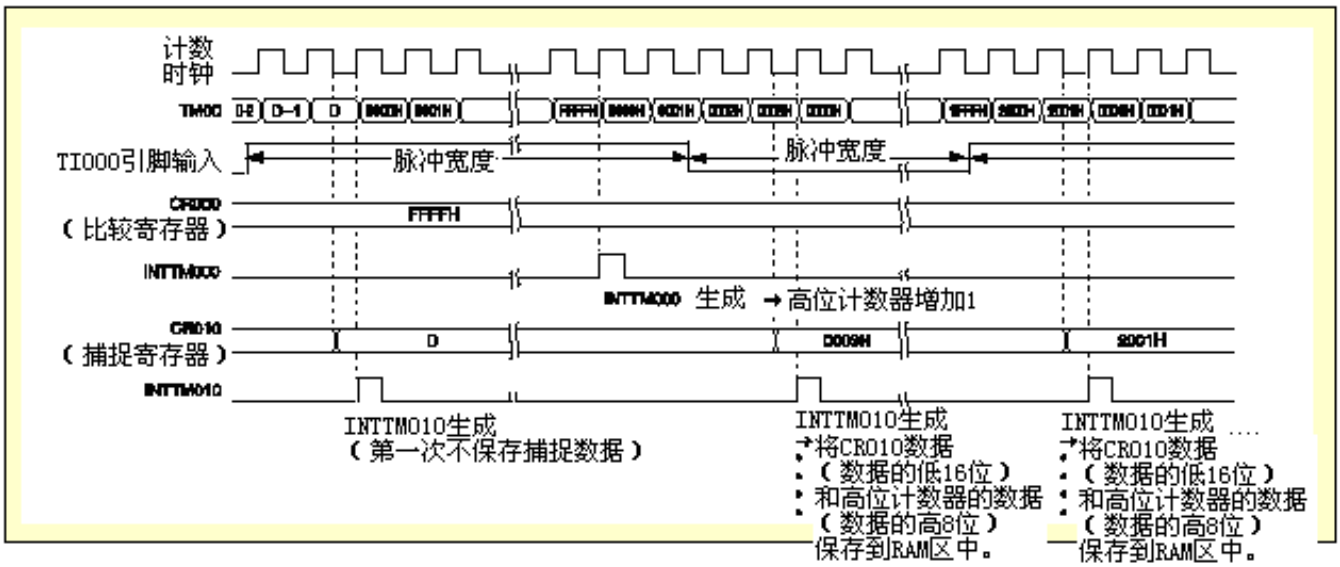
写入该示例程序设备的操作示例如下所示。

（1）在检测到第一个有效沿时的操作

当脉冲输入 TI000 引脚时，第一个有效沿成为触发，启动脉冲宽度测量。

（2）在检测到第二个及以后的有效沿时的操作

因为 TI000 引脚的有效沿检测为下降及上升沿，所以高电平脉冲宽度和低电平脉冲宽度交替捕捉，捕捉数据存储在 RAM 中。



<计算本示例程序的脉冲宽度表达式>

脉冲宽度=（24 位数据[注]+1）×0.0005[ms/CLK]

注　　高 8 位：高位计数器值，低 16 位：CR010 寄存器值

上述操作示例中的脉冲宽度<1>和脉冲宽度<2>如下。

- 脉冲宽度<1>（INTTM000 生成：1 次，CR010 捕捉数据：0003H）
  =（010003H+1H）×0.0005[ms/CLK]=（65539+1）×0.0005[ms/CLK]=32.77[ms]）
- 脉冲宽度<2>（INTTM000 生成：0 次，CR010 捕捉数据：2001H）
  =（002001H+1H）×0.0005[ms/CLK]=（8193+1）×0.0005[ms/CLK]=4.097[ms]）

（3）第八次脉冲宽度测量后的操作

当八个脉冲宽度值存入 RAM 时，不再进行脉冲宽度测量。

第六章 相关文件

文件名称		日语/英语
78K0S/KU1+ 用户手册		PDF
78K0S/KY1+用户手册		PDF
78K0S/KA1+用户手册		PDF
78K0S/KB1+用户手册		PDF
78K/0S系列指令用户手册		PDF
RA78K0S汇编程序包用户手册	语言	PDF
	操作	PDF
CC78K0S C编译器用户手册	语言	PDF
	操作	PDF
PM+项目管理器用户手册		PDF
SM+系统模拟器操作用户手册		PDF
78K0S/KA1+ 简化闪存写入手册 MINICUBE2 信息		PDF
78K0S/Kx1+ 应用注释	示例程序启动向导	PDF
	示例程序（初始设置）LED 照明开关控制	PDF
	示例程序（中断）开关输入引起的外部中断	PDF
	示例程序（低压检测）电压低于2.7V检测时复位发生。	PDF
	示例程序（16-位 计时器/事件计数器00）内部计时器	PDF
	示例程序（16-位计时器/事件计数器00）外部事件计数器	PDF
	示例程序（16-位计时器/事件计数器00）PPG输出	PDF
	示例程序（16-位计时器/事件计数器00）一步脉冲输出	PDF

附件A 程序列表

作为程序列表示例，78K0S/KB1+微处理器的源程序如下所示。

● main.asm (汇编语言版)

```
;*******************************************************************************
;
;       日电电子     78K0S/KB1+
;
;*******************************************************************************
;       78K0S/KB1+        示例程序
;*******************************************************************************
;16 位定时器 00（脉冲宽度测量）
;*******************************************************************************
;<<历史>>
;       2007.7.--             发布
;*******************************************************************************
;
;<<概述>>
;
; 本示例程序提供使用 16 位定时器 00 脉冲宽度测量功能的例子。当计数时钟设置为 fxp/4
;（= 2 MHz）且检测到 TI000 引脚输入信号的上升及下降沿时的定时器计数值被捕捉进入
; CRC010 寄存器中。从捕捉值可知输入 TI000 引脚信号的宽度或时间间隔。通过设置 CR000
; 寄存器为 0xFFFF 并生成溢出中断，可测量长度至少为定时器周期的脉冲的宽度或时间间隔。
;
;
; <主要设置内容>
;
;     -停止看门狗计时器的运行
;     -将低压检查电压（VLVI）设置为 4.3 V +-0.2 V
;     -在 VDD >= VLVI 后当 VDD < VLVI 时生成内部复位信号（低压检测器）
;     -设置 CPU 时钟为 8MHz
;     -设置供给外围硬件的时钟为 8MHz
;     -将 DE 和 HL 寄存器用于中断服务（与全局变量类似）
;
;
; <16 位定时器 00 设置>
```

```
;     -工作模式：在 TI000 引脚的有效沿处清零并启动定时器计数
;     -用 CR000 作为溢出比较寄存器
; - 用 CR010 作为捕捉寄存器
; - 通过检测 TI000 引脚的双沿捕捉脉冲宽度
; - 计数时钟 = fxp/4
; - 不进行定时器输出
;
;
; <计算脉冲宽度>
; 要捕捉的数据长度为 24 位，由 CR010 寄存器的 16 位数据和高 8 位数据组合而成。各数据
; 保存在 RAM 区中，如下表所示。
; 脉冲宽度的计算来自于 24 位数据，如下所示。
;
;   24 位数据 x 0.0005 [ms/clk] = 脉冲宽度 [ms]
;
; # 这里，1 [clk]为定时器 00 计数时钟（2MHz）。
;
; +-----------------------------------------------------------------+
; |    地 址    | 数据长度 |          数据类型          |
; |-----------------------------------------------------------------|
; | LOWLENGTH    | 16 位    |    脉冲宽度低位数据（第一次）       |
; | LOWLENGTH + 2 | 16 位    |    脉冲宽度低位数据（第二次）       |
; | LOWLENGTH + 4 | 16 位    |    脉冲宽度低位数据（第三次）       |
; | LOWLENGTH + 6 | 16 位    |    脉冲宽度低位数据（第四次）       |
; | LOWLENGTH + 8 | 16 位    |    脉冲宽度低位数据（第五次）       |
; | LOWLENGTH +10 | 16 位    |    脉冲宽度低位数据（第六次）       |
; | LOWLENGTH +12 | 16 位    |            脉冲宽度低位数据（第七次）       |
; | LOWLENGTH +14 | 16 位    |            脉冲宽度低位数据（第八次）       |
; |-----------------------------------------------------------------|
; | HIGHLENGTH   |  8 位    |     脉冲宽度高位数据（第一次）     |
; | HIGHLENGTH +1 |  8 位    |     脉冲宽度高位数据（第二次）     |
; | HIGHLENGTH +2 |  8 位    |     脉冲宽度高位数据（第三次）     |
; | HIGHLENGTH +3 |  8 位    |     脉冲宽度高位数据（第四次）     |
; | HIGHLENGTH +4 |  8 位    |     脉冲宽度高位数据（第五次）     |
; | HIGHLENGTH +5 |  8 位    |     脉冲宽度高位数据（第六次）     |
; | HIGHLENGTH +6 |  8 位    |     脉冲宽度高位数据（第七次）     |
; | HIGHLENGTH +7 |  8 位    |     脉冲宽度高位数据（第八次）     |
; +-----------------------------------------------------------------+
;
;
;<<I/O 端口设置>>
;
; 输入：P30
; 输出：P00-P03， P20-P23， P31-P33， P40-P47， P120-P123， P130
; # 所有未使用的端口设置为输出模式。
;
;*******************************************************************************
```

```
;=======================================================================
;
;       向量表
;
;=======================================================================
XVCTCSEG    AT      0000H
        DW      RESET_START             ;(00)   RESET
        DW      RESET_START             ;(02)   --
        DW      RESET_START             ;(04)   --
        DW      RESET_START             ;(06)   INTLVI
        DW      RESET_START             ;(08)   INTP0
        DW      RESET_START             ;(0A)   INTP1
        DW      RESET_START             ;(0C)   INTTMH1
        DW      INTERRUPT_TM000         ;(0E)   INTTM000
        DW      INTERRUPT_TM010         ;(10)   INTTM010
        DW      RESET_START             ;(12)   INTAD
        DW      RESET_START             ;(14)   --
        DW      RESET_START             ;(16)   INTP2
        DW      RESET_START             ;(18)   INTP3
        DW      RESET_START             ;(1A)   INTTM80
        DW      RESET_START             ;(1C)   INTSRE6
        DW      RESET_START             ;(1E)   INTSR6
        DW      RESET_START             ;(20)   INTST6

;=======================================================================
;
;       定义 RAM
;
;=======================================================================
XRAM            DSEG    SADDRP
LOWLENGTH:DS    16              ; 脉冲宽度低 16 位存储表
HIGHLENGTH:     DS      8       ; 脉冲宽度高 8 位存储表
HIGHCOUNT: DS   1               ; 用于对脉冲宽度高位进行计数
TIMES:          DS      1       ; 用于对测量次数进行计数

;=======================================================================
;
;       定义存储器栈区
;
;=======================================================================
XSTKDSEG    AT      0FEE0H
STACKEND:
        DS      20H             ; 存储器栈区 = 32 字节
STACKTOP:                       ; 存储器栈区起始地址 = FF00H

;************************************************************************
```

```
;
;       Reset 后的初始化
;
;************************************************************************
XMAIN           CSEG    UNIT
RESET_START:
;------------------------------------------------------------------------
;       初始化栈指针
;------------------------------------------------------------------------
        MOVW    AX，    #STACKTOP
        MOVW    SP，    AX                  ; 设置栈指针

;------------------------------------------------------------------------
;       初始化看门狗计时器
;------------------------------------------------------------------------
        MOV     WDTM，          #01110111B      ; 停止看门狗计时器的运行

;------------------------------------------------------------------------
;       检测低压+设置时钟
;------------------------------------------------------------------------

;-----  设置时钟<1>  -----
        MOV     PCC，   #00000000B      ; 供给 CPU 的时钟(fcpu)= fxp（= fx/4 = 2 MHz）
        MOV     LSRCM，         #00000001B      ; 停止低速内部振荡器的振荡

;-----  检查复位源  -----
        MOV     A，     RESF            ; 读取复位源
        BT      A.0，   $SET_CLOCK      ; 在 LVI 复位时，省略后续 LVI 相关处理，转到 SET_CLOCK

;-----  设置低压检测  -----
        MOV     LVIS，  #00000000B      ; 将低压检查电压（VLVI）设置为 4.3 V +-0.2 V
        SET1    LVION                   ; 启用低压检测器的运行

        MOV     A，     #40             ; 分配 200us 等待计数值
;-----  200 us 等待  -----
WAIT_200US:
        DEC     A
        BNZ     $WAIT_200US             ; 0.5[us/clk] x 10[clk] x 40[计数] = 200[us]

;-----  VDD >= VLVI 等待处理  -----
WAIT_LVI:
        NOP
        BT      LVIF，  $WAIT_LVI       ; 如果 VDD < VLVI，分支执行

        SET1    LVIMD                   ; 设置为当 VDD < VLVI 时生成内部复位信号

;-----  设置时钟<2> -----
SET_CLOCK:
```

```
        MOV     PPCC，  #00000000B      ; 设置供给外围硬件的时钟(fxp) = fx (= 8 MHz)
                                        ;  ->供给 CPU 的时钟(fcpu)= fxp = 8 MHz

;---------------------------------------------------------------------
;       初始化端口 0
;---------------------------------------------------------------------
        MOV     P0，    #00000000B      ; 将 P00-P03 的输出锁存器设置为低
        MOV     PM0，   #11110000B      ; 设置 P00-P03 为输出模式

;---------------------------------------------------------------------
;       初始化端口 2
;---------------------------------------------------------------------
        MOV     P2，    #00000000B      ; 将 P20-P23 的输出锁存器设置为低
        MOV     PM2，   #11110000B      ; 设置 P20-P23 为输出模式

;---------------------------------------------------------------------
;       初始化端口 3
;---------------------------------------------------------------------
        MOV     P3，    #00000000B      ; 将 P30-P33 的输出锁存器设置为低
        MOV     PM3，   #11110001B      ; 设置 P31-P33 为输出模式、P30/TI000 为输入模式

;---------------------------------------------------------------------
;       初始化端口 4
;---------------------------------------------------------------------
        MOV     P4，    #00000000B      ; 将 P40-P47 的输出锁存器设置为低
        MOV     PM4，   #00000000B      ; 设置 P40-P47 为输出模式

;---------------------------------------------------------------------
;       初始化端口 12
;---------------------------------------------------------------------
        MOV     P12，   #00000000B      ; 将 P120-P123 的输出锁存器设置为低
        MOV     PM12，  #11110000B      ; 设置 P120-P123 为输出模式

;---------------------------------------------------------------------
;       初始化端口 13
;---------------------------------------------------------------------
        MOV     P13，   #00000001B      ; 将 P130 的输出锁存器设置为高
```

```
;--------------------------------------------------------------------------
;       设置 16 位定时器 00
;--------------------------------------------------------------------------
        MOV     CRC00，         #00000100B      ; 用 CR010 作为捕捉寄存器
        MOVW    AX，    #0FFFFH
        MOVW    CR000，         AX              ; 用 CR000 进行溢出检测
        MOV     PRM00，         #00110001B      ; 将 TI000 引脚的有效沿指定为双沿，设置计数时钟为 fxp/4
        MOV     TOC00，         #00000000B      ; 不进行定时器输出

;--------------------------------------------------------------------------
;       设置中断
;--------------------------------------------------------------------------
        MOV     MK0，   #0FFH           ; 首先屏蔽所有中断
        MOV     IF0，   #00H            ; 事先清除无效中断请求

;**************************************************************************
;
;       主循环
;
;**************************************************************************
        MOVW    HL，    #LOWLENGTH      ; 初始化存储低 16 位的表格的地址
        MOVW    DE，    #HIGHLENGTH     ; 初始化存储高 8 位的表格的地址
        MOV     HIGHCOUNT， #0          ; 高位计数器清零
        MOV     TIMES，         #8      ; 初始化保存的结果个数

        MOV     TMC00，         #00001000B      ; 启动定时器的运行（在 TI000 引脚的有效沿处清零并启动）
WAITCAP:
        NOP
        BF      TMIF010， $WAITCAP      ; 等待第一次捕捉的完成

        MOV     IF0，   #00H            ; 中断请求标志清零
        CLR1    TMMK000                 ; INTTM000 中断去屏蔽
        CLR1    TMMK010                 ; Inttm000 中断去屏蔽

        EI                              ; 启用向量中断

MAIN_LOOP:
        NOP
        BR      $MAIN_LOOP              ; 转到 MAIN_LOOP
```

```
;*******************************************************************************
;
;       中断 INTTM000
;
;*******************************************************************************
INTERRUPT_TM000：
        PUSH    AX                      ; 将 AX 寄存器数据保存到栈中

        BF      TMIF010， $INCHIGH      ; 如果未与 INTTM000 中断同时生成，就分支执行
        MOVW    AX，    CR010           ; 读取捕捉值
        CMPW    AX，    #0FFFFH
        BZ      $ENDINTTM000 ; 如果中断同时生成且如果 CR010==0xFFFF，就不计数

INCHIGH:
        INC     HIGHCOUNT               ; 将高位计数增加 1

ENDINTTM000:
        POP     AX                      ; 还原 AX 寄存器数据
        RETI                            ; 从中断服务返回

;*******************************************************************************
;
;       中断 INTTM010
;
;*******************************************************************************
INTERRUPT_TM010：
        PUSH    AX                      ; 将 AX 寄存器数据保存到栈中

        MOVW    AX，    CR010           ; 读取捕捉值
        MOV     [HL+1]，A       ; 保存到存储低 16 位的表格中
        XCH     A，     X
        MOV     [HL]，  A               ; 保存到存储低 16 位的表格中
        MOV     A，     HIGHCOUNT       ; 读取高位
        MOV     [DE]，  A               ; 保存到存储高 8 位的表格中

        INC     L                       ; 存储低 16 位的表格的地址增加 2
        INC     L
        INC     E                       ; 存储高 8 位的表格的地址增加 1
        MOV     HIGHCOUNT， #0          ; 高位清零

        DBNZ    TIMES，         $ENDINTTM000 ；如果测量次数小于 8 就分支执行
        SET1    TMMK000                 ; 屏蔽 INTTM000 中断
        SET1    TMMK010                 ; 屏蔽 INTTM010 中断
```

```
ENDINTTM010:
        POP     AX                      ; 还原 AX 寄存器数据
        RETI                            ; 从中断服务返回

end
```

● main.c (C 语言版)

```
/*********************************************************************************

        日电电子 78K0S/KB1+

*********************************************************************************
        78K0S/KB1+      示例程序
*********************************************************************************
16 位定时器 00（脉冲宽度测量）
*********************************************************************************
<<历史>>
        2007.7.--           发布
*********************************************************************************

<<概述>>

本示例程序提供使用 16 位定时器 00 脉冲宽度测量功能的例子。当计数时钟设置为 fxp/4（= 2 MHz）且检测到 TI000
引脚输入信号的上升及下降沿时的定时器计数值被捕捉进入 CRC010 寄存器中。从捕捉值可知输入 TI000 引脚信号的
宽度或时间间隔。通过设置 CR000 寄存器为 0xFFFF 并生成溢出中断，可测量长度至少为定时器周期的脉冲的宽度或
时间间隔。


 <主要设置内容>

- 声明由中断运行的函数：INTTM000 -> fn_inttm000()
- 声明由中断运行的函数：INTTM010 -> fn_inttm010()
- 停止看门狗计时器的运行
- 将低压检查电压（VLVI）设置为 4.3 V +-0.2 V
- 在 VDD >= VLVI 后当 VDD < VLVI 时生成内部复位信号（低压检测器）
- 设置 CPU 时钟为 8MHz
- 设置供给外围硬件的时钟为 8MHz


 <16 位定时器 00 设置>

- 工作模式：在 TI000 引脚的有效沿处清零并启动定时器计数
- 用 CR000 作为溢出比较寄存器
- 用 CR010 作为捕捉寄存器
- 通过检测 TI000 引脚的双沿捕捉脉冲宽度
- 计数时钟 = fxp/4
- 不进行定时器输出


 <计算脉冲宽度>

 要捕捉的数据长度为 24 位，由 CR010 寄存器的 16 位数据和高 8 位数据组合而成。
 各数据保存在 RAM 区中，如下表所示。
 脉冲宽度的计算来自于 24 位数据，如下所示。

   24 位数据 x 0.0005 [ms/clk] = 脉冲宽度 [ms]

 # 这里，1 [clk]为定时器 00 计数时钟（2MHz）。

 +-------------------------------------------------+
 |      变量名     |数据长度|     数据类型          |
 |-------------------------------------------------|
 | g_unLowLength[0] | 16 位 | 脉冲宽度低位数据（第一次） |
 | g_unLowLength[1] | 16 位 | 脉冲宽度低位数据（第二次） |
 | g_unLowLength[2] | 16 位 | 脉冲宽度低位数据（第三次） |
```

```
 | g_unLowLength[3] | 16 位 | 脉冲宽度低位数据（第四次） |
 | g_unLowLength[4] | 16 位 | 脉冲宽度低位数据（第五次） |
 | g_unLowLength[5] | 16 位 | 脉冲宽度低位数据（第六次） |
 | g_unLowLength[6] | 16 位 | 脉冲宽度低位数据（第七次） |
 | g_unLowLength[7] | 16 位 | 脉冲宽度低位数据（第八次） |
 |-------------------------------------------------|
 | g_unHighLength[0] | 8 位 | 脉冲宽度高位数据（第一次） |
 | g_unHighLength[1] | 8 位 | 脉冲宽度高位数据（第二次） |
 | g_unHighLength[2] | 8 位 | 脉冲宽度高位数据（第三次） |
 | g_unHighLength[3] | 8 位 | 脉冲宽度高位数据（第四次） |
 | g_unHighLength[4] | 8 位 | 脉冲宽度高位数据（第五次） |
 | g_unHighLength[5] | 8 位 | 脉冲宽度高位数据（第六次） |
 | g_unHighLength[6] | 8 位 | 脉冲宽度高位数据（第七次） |
 | g_unHighLength[7] | 8 位 | 脉冲宽度高位数据（第八次） |
 +-------------------------------------------------+

<<I/O 端口设置>>

 输入：P30
 输出：P00-P03， P20-P23， P31-P33， P40-P47， P120-P123， P130
 # 所有未使用的端口设置为输出模式。

*****************************************************************************/


/*============================================================================

        预处理指令 (#pragma)

============================================================================*/
#pragmaSFR                                  /* 可在 C 源程序级描述 SFR 名称 */
#pragmaEI                                   /* 可在 C 源程序级描述 EI 指令 */
#pragmaNOP                                  /* 可在 C 源程序级描述 NOP 指令 */
#pragma interrupt INTTM000 fn_inttm000      /* 中断函数声明：INTTM000 */
#pragma interrupt INTTM010 fn_inttm010      /* 中断函数声明：INTTM010 */

/*============================================================================

        定义全局变量

============================================================================*/
sreg static unsigned int g_unLowLength[8];   /* 用于存储脉冲宽度低 16 位的 16 位变量 */
sreg static unsigned char g_unHighLength[8];          /* 用于存储脉冲宽度高位的 8 位变量 */
sreg static unsigned char g_ucHighCount;    /* 用于对脉冲宽度高位进行计数的 8 位变量 */
sreg static unsigned char g_ucTimes;        /* 用于对测量次数进行计数的 8 位变量 */

/*****************************************************************************

        Reset 后的初始化

*****************************************************************************/
void hdwinit(void){
        unsigned char ucCnt200us;           /* 用于 200us 等待的 8 位变量 */

/*---------------------------------------------------------------------
        初始化看门狗计时器 + 检测低压 + 设置时钟
---------------------------------------------------------------------*/
        /* 初始化看门狗计时器 */
        WDTM  = 0b01110111;                 /* 停止看门狗计时器的运行 */
```

```
        /* 设置时钟<1> */
        PCC   = 0b00000000;             /* 供给 CPU 的时钟(fcpu)= fxp（= fx/4 = 2 MHz） */
        LSRCM = 0b00000001;             /* 停止低速内部振荡器的振荡*/

        /* 检查复位源 */
        if (!(RESF & 0b00000001)){      /* 在 LVI 复位时，省略后续 LVI 相关处理 */

                /* 设置低压检测 */
                LVIS  = 0b00000000;     /* 将低压检查电压（VLVI）设置为 4.3 V +-0.2 V */
                LVION = 1;              /* 启用低压检测器的运行 */

                for (ucCnt200us = 0; ucCnt200us < 9; ucCnt200us++){         /* 等待 200us 左右 */
                        NOP();
                }

                while (LVIF){           /* 等待 VDD >= VLVI */
                        NOP();
                }

                LVIMD = 1;                      /* 设置为当 VDD < VLVI 时生成内部复位信号 */
        }

        /* 设置时钟<2> */
        PPCC  = 0b00000000;             /* 设置供给外围硬件的时钟(fxp) = fx (= 8 MHz)
                                                -> 供给 CPU 的时钟(fcpu)= fxp = 8 MHz */

/*------------------------------------------------------------------------
        初始化端口 0
------------------------------------------------------------------------*/
        P0    = 0b00000000;             /* 将 P00-P03 的输出锁存器设置为低 */
        PM0   = 0b11110000;             /* 设置 P00-P03 为输出模式 */

/*------------------------------------------------------------------------
        初始化端口 2
------------------------------------------------------------------------*/
        P2    = 0b00000000;             /* 将 PP20-P23 的输出锁存器设置为低 */
        PM2   = 0b11110000;             /* 设置 P20-P23 为输出模式 */

/*------------------------------------------------------------------------
        初始化端口 3
------------------------------------------------------------------------*/
        P3    = 0b00000000;             /* 将 P30-P33 的输出锁存器设置为低 */
        PM3   = 0b11110001;             /* 设置 P31-P33 为输出模式、P30/TI000 为输入模式 */

/*------------------------------------------------------------------------
        初始化端口 4
------------------------------------------------------------------------*/
        P4    = 0b00000000;             /* 将 P40-P47 的输出锁存器设置为低 */
        PM4   = 0b00000000;             /* 设置 P40-P47 为输出模式 */

/*------------------------------------------------------------------------
        初始化端口 12
------------------------------------------------------------------------*/
        P12   = 0b00000000;             /* 将 P120-P123 的输出锁存器设置为低 */
        PM12  = 0b11110000;             /* 设置 P120-P123 为输出模式 */

/*------------------------------------------------------------------------
        初始化端口 13
------------------------------------------------------------------------*/
        P13   = 0b00000001;             /* 将 P130 的输出锁存器设置为高 */
```

```
/*---------------------------------------------------------------------------
        设置 16 位定时器 00
---------------------------------------------------------------------------*/
        CRC00 = 0b00000100;             /* Cr010 用作捕捉寄存器 */
        CR000 = 0xFFFF;                         /* Cr000 用作溢出检测 */
        PRM00 = 0b00110001;             /* 指定 TI000 引脚的有效沿为双沿，设置计数时钟为 fxp/4 */
        TOC00 = 0b00000000;             /* 不进行定时器输出 */

/*---------------------------------------------------------------------------
        设置中断
---------------------------------------------------------------------------*/
        MK0   = 0xFF;                   /* 首先屏蔽所有中断 */
        IF0   = 0x00;                   /* 无效中断请求清零 */

        return;
}

/*****************************************************************************

        主循环

*****************************************************************************/
void main(void){
        g_ucHighCount = 0;              /* 高位计数器清零 */
        g_ucTimes = 8;                  /* 初始化测量次数 */
        TMC00 = 0b00001000;             /* 启动定时器的运行（在 TI000 引脚的有效沿处清零并启动） */

        while(TMIF010==0){              /* 等待第一次捕捉的完成 */
                NOP();
        }

        IF0 = 0x00;                             /* 中断请求标志清零 */
        TMMK000 = 0;                    /* Inttm000 中断去屏蔽 */
        TMMK010 = 0;                    /* Inttm010 中断去屏蔽 */

        EI();                                   /* 启用向量中断 */

        while(1){                       /* 无限循环 */
                NOP();
                NOP();
        }
}

/*****************************************************************************

        中断 INTTM000

*****************************************************************************/
__interrupt void fn_inttm000(){

        if (!(TMIF010 && (CR010 == 0xFFFF))){       /* 如果中断同时生成且如果 CR010==0xFFFF，就不计数 */
                g_ucHighCount++;        /* 将高位计数增加 1 */
        }

        return;
}

/*****************************************************************************

        中断 INTTM010

*****************************************************************************/
```

```
__interrupt void fn_inttm010(){

        g_unLowLength[8 - g_ucTimes] = CR010;     /* 读取捕捉值 */
        g_unHighLength[8 - g_ucTimes] = g_ucHighCount;     /* 读取高位 */
        g_ucHighCount = 0;                        /* 高位清零 */
        g_ucTimes--;                              /* 测量次数减少 1 */

        if(g_ucTimes == 0){          /* 当第八次捕捉完成时 */
                TMMK000 = 1;                      /* 屏蔽 INTTM000 中断 */
                TMMK010 = 1;                      /* 屏蔽 INTTM010 中断 */
        }

        return;
}
```

● op.asm（汇编语言和 C 语言版共用）

```
;=========================================================================
;
;          选项字节
;
;=========================================================================
OPBT      CSEG                AT        0080H
      DB      10011100B ; 选项字节区
;                           ||||
;                           |||+-----------              低速内部振荡器可用软件停止
;                           |++------------              高速内部时钟(8 MHz)选择为系统时钟源
;                +--------------        P34/RESET 用作 RESET 引脚

      DB                  11111111B               ; 保护字节区（用于自编程模式）
;                           ||||||||
;                           ++++++++-----------       所有模块均可写入或删除

结束
```

附件B 修订记录

版本	出版日期	页码	修订
第一版	2008年2月	–	–

详细信息请联系：

中国区

MCU 技术支持热线：
电话：+86-400-700-0606 (普通话)
服务时间：9:00-12:00，13:00-17:00 （不含法定节假日）

网址：
http://www.cn.necel.com/ （中文）
http://www.necel.com/ （英文）

[北京]

日电电子（中国）有限公司
中国北京市海淀区知春路 27 号
量子芯座 7，8，9，15 层
电话：（+86）10-8235-1155
传真：（+86）10-8235-7679

[上海]

日电电子（中国）有限公司上海分公司
中国上海市浦东新区银城中路 200 号
中银大厦 2409-2412 和 2509-2510 室
电话：（+86）21-5888-5400
传真：（+86）21-5888-5230

上海恩益禧电子国际贸易有限公司
中国上海市浦东新区银城中路 200 号
中银大厦 2511-2512 室
电话：（+86）21-5888-5400
传真：（+86）21-5888-5230

[深圳]

日电电子（中国）有限公司深圳分公司
深圳市福田区益田路卓越时代广场大厦 39 楼
3901，3902，3909 室
电话：（+86）755-8282-9800
传真：（+86）755-8282-9899

[香港]

香港日电电子有限公司
香港九龙旺角太子道西 193 号新世纪广场
第 2 座 16 楼 1601-1613 室
电话： （+852）2886-9318
传真： （+852）2886-9022
2886-9044

[成都]

日电电子（中国）有限公司成都分公司
成都市二环路南三段 15 号天华大厦 7 楼 703 室
电话：(+86)28-8512-5224
传真：(+86)28-8512-5334