NEC

使用说明

78K0S/Kx1+

示例程序(16 位定时器/事件计数器 00)

PPG 输出

本文档描述了示例程序的操作概述和如何应用，以及如何设置和应用 16 位定时器/事件计数器 00 的 PPG 输出功能。该示例程序中，利用 16 位定时器/事件计数器 00 的 PPG 输出功能可输出一个任意周期和脉冲宽度的方波。此外，依照开关输入次数，LED 的亮度随 PPG 输出占空比的变化而变化。

目标器件

78K0S/KA1+ 微控制器
78K0S/KB1+ 微控制器
78K0S/KU1+ 微控制器
78K0S/KY1+ 微控制器

目录

文档编号 U18890CA1V0AN00 (第一版)
发布日期 2008 年 03 月 N

于日本印刷

- 本文档信息发布于2008年03月。未来可能未经预先通知而进行更改。在实际进行生产设计时，请参阅各产品最新的数据规格书或数据手册等相关资料，以获取本公司产品的最新规格。并非所有的产品和/或型号都向每个国家供应。请向本公司销售代表查询产品供货及其他信息。
- 未经本公司事先书面许可，禁止采用任何方式复制或转载本文件中的内容。本文件所登载内容的错误，本公司概不负责。
- 本公司对于因使用本文件中列明的本公司产品而引起的，对第三者的专利、版权以及其它知识产权的侵权行为概不负责。本文件登载的内容不应视为本公司对本公司或其他人所有的专利、版权以及其它知识产权做出任何明示或默示的许可及授权。
- 本文件中的电路、软件以及相关信息仅用以说明半导体产品的运作和应用实例。用户如在设备设计中应用本文件中的电路、软件以及相关信息，应自行负责。对于用户或其他人因使用了上述电路、软件以及相关信息而引起的任何损失，本公司概不负责。
- 虽然本公司致力于提高半导体产品的质量及可靠性，但用户应同意并知晓，我们仍然无法完全消除出现产品缺陷的可能。为了最大限度地减少因本公司半导体产品故障而引起的对人身、财产造成损害（包括死亡）的危险，用户务必在其设计中采用必要的安全措施，如冗余度、防火和防故障等安全设计。
- 本公司产品质量分为：“标准等级”、“专业等级”以及“特殊等级”三种质量等级。

  “特殊等级”仅适用于为特定用途而根据用户指定的质量保证程序所开发的日电电子产品。另外，各种日电电子产品的推荐用途取决于其质量等级，详见如下。用户在选用本公司的产品时，请事先确认产品的质量等级。

  “标准等级”： 计算机，办公自动化设备，通信设备，测试和测量设备，视音频设备，家电，加工机械，个人电气设备以及产业用机器人。

  “专业等级”： 运输设备（汽车、火车、船舶等），交通用信号控制设备，防灾装置，防止犯罪装置，各种安全装置以及医疗设备（不包括专门为维持生命而设计的设备）。

  “特殊等级”： 航空器械，宇航设备，海底中继设备，原子能控制系统，为了维持生命的医疗设备和用于维持生命的装置或系统等。

  除在本公司半导体产品的数据表或数据手册等资料中另有特别规定以外，本公司半导体产品的质量等级均为“标准等级”。如果用户希望在本公司设计意图以外使用本公司半导体产品，务必事先与本公司销售代表联系以确认本公司是否同意为该项应用提供支持。

  （注）

  （1）本声明中的“本公司”是指日本电气电子株式会社（NEC Electronics Corporation）及其控股公司。

  （2）本声明中的“本公司产品”是指所有由日本电气电子株式会社或为日本电气电子株式会社（如上定义）开发或制造的产品。

M8E 02.11-1

第一章 概要

该举例程序介绍了使用 16 位定时器/事件计数器 00 的 PPG 输出功能实例。依照开关输入次数控制 PPG 输出占空比以改变 LED 的亮度。

1.1 初始设置的主要内容

初始设置的主要内容如下所示：

- 选择内部高速振荡器作为系统时钟信号源[注]。
- 停止看门狗定时器运行。
- 将 V_{LVI} (低压检测电压) 设置在 4.3 V ±0.2 V 范围。
- V_{DD}（电源电压）变得高于或等于 V_{LVI} 后，当检测到 V_{DD} 低于 V_{LVI} 时产生内部复位（LVI 复位）信号。
- 设置 CPU 时钟频率为 8 MHz 。
- 设置 I/O 端口。
- 设置 16 位定时器/事件计数器 00。
  - 设置 CR000 和 CR010 为比较寄存器 。
  - 设置一个 PPG 输出周期为 200 μs (0.5 μs × 400)。
  - 设置 PPG 输出脉冲宽度为 20 μs (0.5 μs × 40)。
  - 设置计数时钟为 $f_{XP}/2^2$ (2 MHz)。
  - 允许 CR000 和 TM00 或 CR010 和 TM00 匹配引起的定时器输出反相 。
  - 设置定时器输出初始值为 1 (设置(1) 定时器输出 F/F)。
  - 使能定时器输出(TO00 引脚输出)。
  - 设置操作模式为基于 TM00 和 CR000 匹配而清零并启动 。
- 设置 INTP1（外部中断）下降沿有效。
- 使能 INTP1 中断。

注　　用选项字节进行设置。

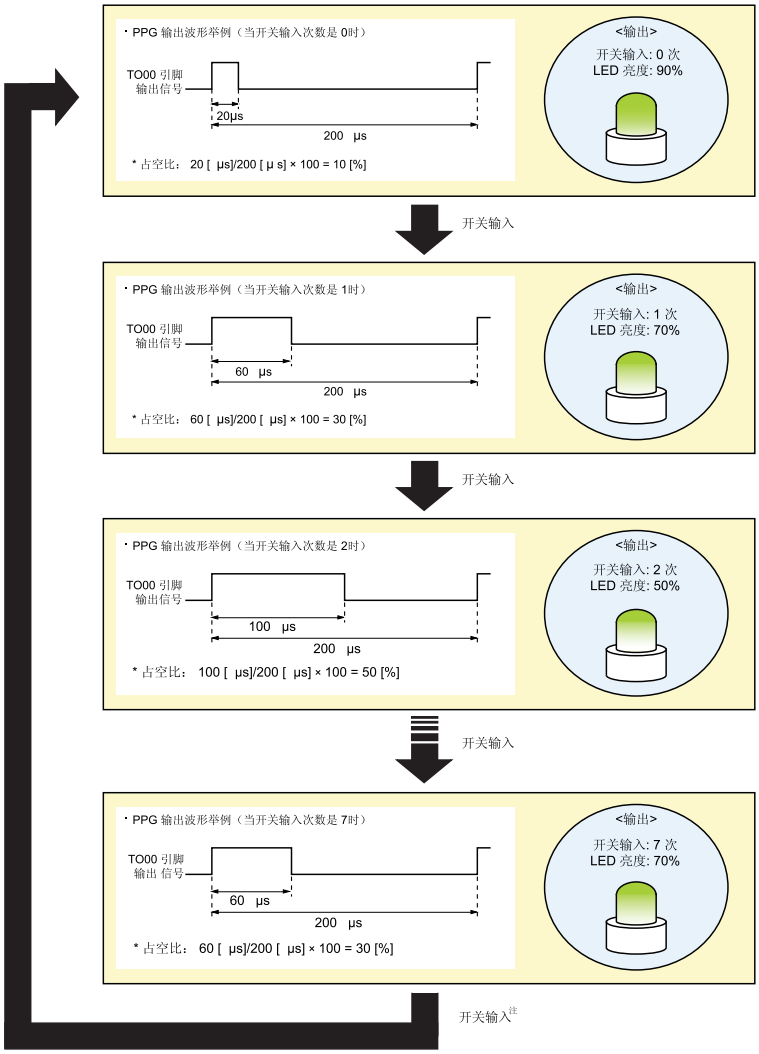
1.2 主循环之后的内容

初始设置完成后，利用 16 位定时器/事件计数器 00 的 PPG 输出使 LED 以固定的亮度发光。

当检测到由开关输入产生的 INTP1 引脚下降沿时，进行 INTP1 中断服务。INTP1 引脚下降沿 10 ms 后，若 INTP1 是高电平（开关关闭），确认为抖动。自检测到边沿 10ms 后，若 INTP1 是低电平（开关开启），则依照开关输入次数，通过改变 PPG 输出占空比来改变 LED 的亮度。

开关输入次数[注]	0 次	1 次	2 次	3 次	4 次	5 次	6 次	7 次
PPG 输出占空比	10%	30%	50%	70%	90%	70%	50%	30%
LED 亮度	90%	70%	50%	30%	10%	30%	50%	70%

该举例程序中，PPG 输出的当前电平设置为高电平，因为当其为低电平时 LED 开启，所以 “LED 亮度= 100 – PPG 输出占空比”。



注 第八次开关输入后 PPG 输出占空比从第零次开关输入重复。

注意事项 关于使用器件时的注意事项，参见各自产品的用户手册(78K0S/KU1+，78K0S/KY1+，78K0S/KA1+，78K0S/KB1+)。

[专栏] 抖动

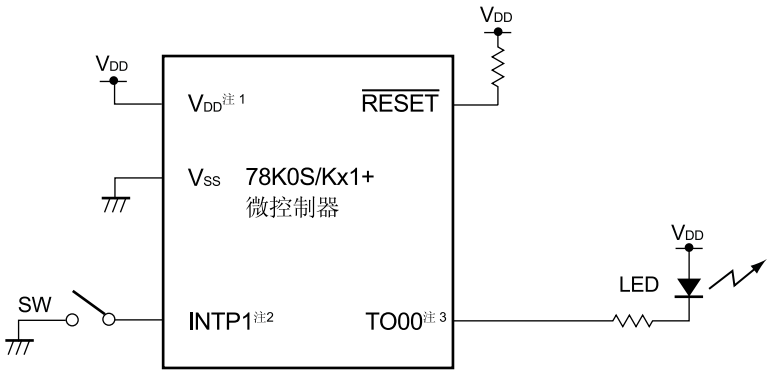
抖动是一种开关按下之后由于机械触点的弹跳而引起电信号瞬时反复接通和断开的现象。

第二章　电路图

本章描述了该举例程序中所使用的电路图和外围硬件。

2.1　电路图

电路图如下所示：



注　**1.** 适用电压范围为 4.5 V ≤ V_{DD} ≤ 5.5 V 。

2. INTP1/TxD6/P43：　78K0S/KA1+ 和 78K0S/KB1+ 微控制器。
INTP1/P32：　78K0S/KY1+ 和 78K0S/KU1+ 微控制器。

3. TO00/TI010/INTP2/P31：　78K0S/KA1+ 和 78K0S/KB1+ 微控制器。
TO00/TI010/INTP0/ANI1/P21：　78K0S/KY1+ 和 78K0S/KU1+ 微控制器。

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

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

3.　除电路图中所示引脚及 AV_{REF} 和 AV_{SS} 引脚外，其他所有未用引脚保留开路状态（未连接）。

2.2　外围硬件

使用的外围硬件如下所示：

(1)　开关(SW)

开关用作控制 LED 亮度的输入。

(2)　LED

LED 用作 16 位定时器/事件计数器 00 的 PPG 输出功能和开关输入相应的显示输出。

第三章　软件

本章描述了所下载的压缩文件组成、所用微控制器的内部外设功能以及举例程序的初始设置和操作概述，并显示了流程图。

3.1　文件的组成

下表显示了所下载压缩文件的组成：

文件名称	说明	包含的压缩文件(*.zip)		
				
main.asm (汇编语言版本) --- main.c (C 语言版本)	有关微控制器硬件初始化处理和主处理程序的源文件。	●注 1	●注 1	
op.asm	设置选项字节(设置系统时钟信号源)的汇编程序源文件。	●	●	
tm00ppg.prw	集成开发环境PM+的工作空间文件。		●	
tm00ppg.prj	集成开发环境PM+的工程文件。		●	
tm00ppg.pri tm00ppg.prs tm00ppg.prm	78K0S/Kx1+系统仿真器SM+的工程文件。		●注 2	
tm00ppg0.pnl	78K0S/Kx1+系统仿真器SM+的I/O面板文件(用于检查外围硬件的工作)。		●注 2	●
tm00ppg0.wvo	78K0S/Kx1+系统仿真器SM+的时序图文件 (用于检查波形)。			●

注　**1.** 汇编语言版本包含“main.asm”文件， C 语言版本包含“main.c”文件。

2. 78K0S/KU1+微控制器的文件中不包含这些文件。

备注　  ：仅包含源文件。

 ：包含用于集成开发环境 PM+和 78K0S/Kx1+系统仿真器 SM+的文件。

：  ：包含用于 78K0S/Kx1+系统仿真器 SM+的微控制器工作仿真文件。

3.2 所用的内部外设功能

该举例程序中，使用了微控制器的下列内部外设功能：

- PPG 输出功能： 16 位定时器/事件计数器 00。
- $V_{DD} < V_{LVI}$ 检测： 低压检测器(LVI)。
- 开关输入： INTP1 注 1 (外部中断)。
- PPG 输出(LED 输出)： TO00 注 2 (定时器输出)。

注 **1.** INTP1/TxD6/P43： 78K0S/KA1+ 和 78K0S/KB1+ 微控制器。
INTP1/P32： 78K0S/KY1+ 和 78K0S/KU1+ 微控制器。
2. TO00/TI010/INTP2/P31： 78K0S/KA1+ 和 78K0S/KB1+ 微控制器。
TO00/TI010/INTP0/ANI1/P21： 78K0S/KY1+ 和 78K0S/KU1+ 微控制器。

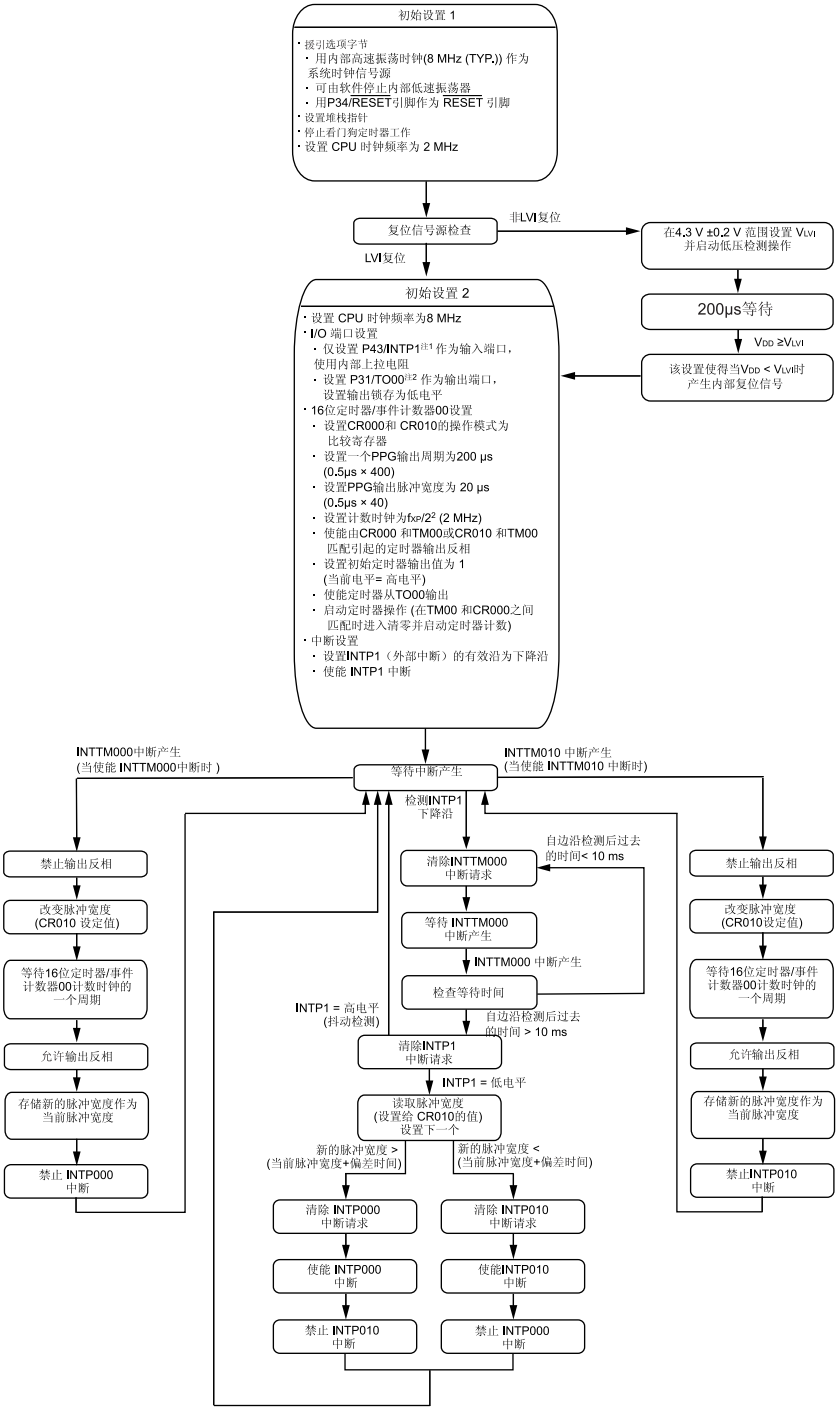
3.3 初始设置和工作概要

在该举例程序中，初始设置包括低压检测功能、选择时钟频率、设置 I/O 端口、设置 16 位定时器/事件计数器 00（PPG 输出功能）和中断设置。

初始设置完成后，利用 16 位定时器/事件计数器 00 的 PPG 输出，使 LED 以固定的亮度发光。

当检测到由开关输入产生的 INTP1 引脚下降沿时，进行 INTP1 中断服务。自检测到 INTP1 引脚下降沿 10ms 后，若 INTP1 检测为高电平（开关关闭），确认为抖动。自检测到边沿 10ms 后，若 INTP1 检测为低电平（开关开启），则依照开关输入次数，改变 PPG 输出占空比来改变 LED 的亮度。

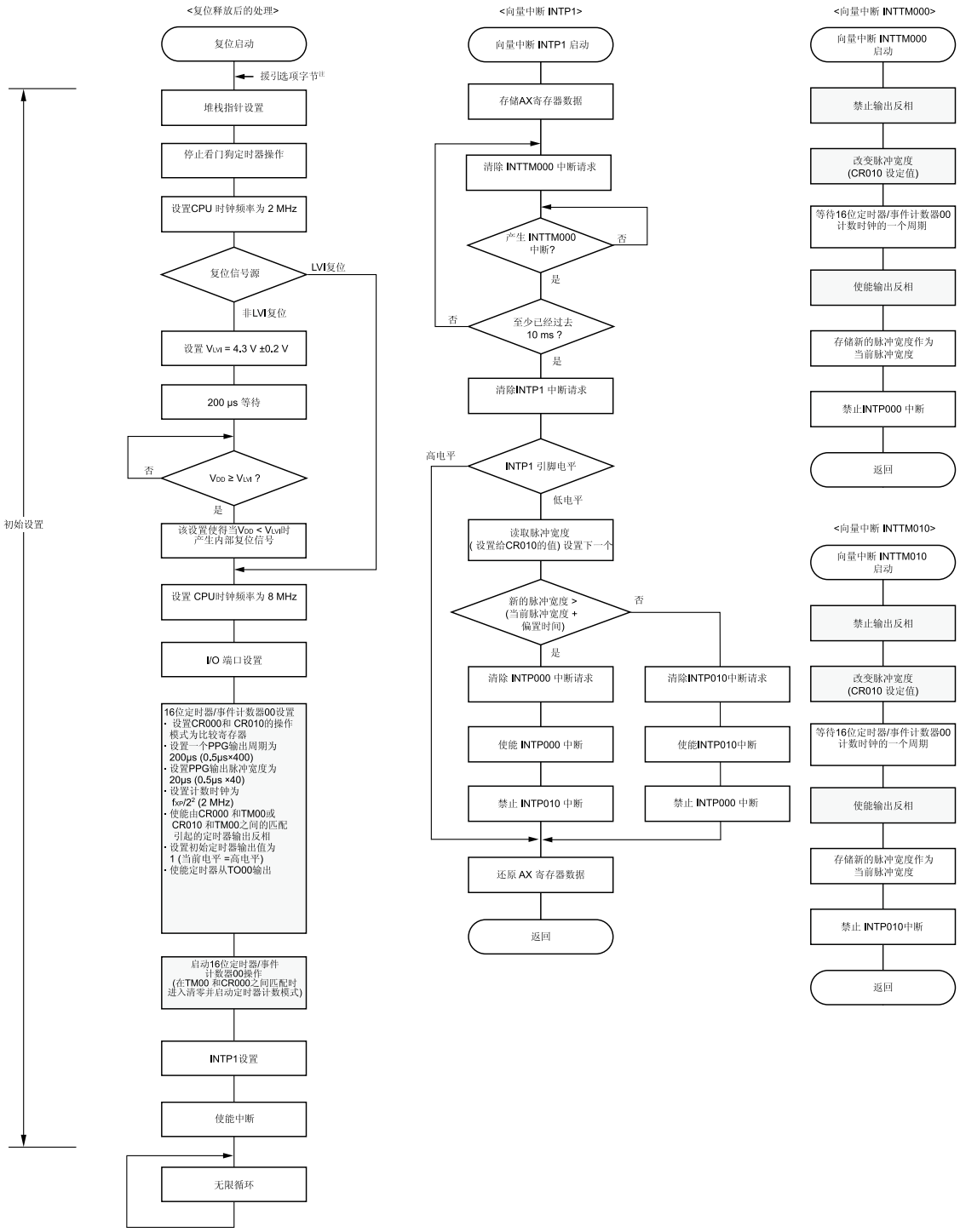
详情如下列状态转换图所示：



注 **1.** INTP1/P43：78K0S/KA1+ 和 78K0S/KB1+ 微控制器。
INTP1/P32：78K0S/KY1+ 和 78K0S/KU1+ 微控制器。
2. TO00/P31： 78K0S/KA1+ 和 78K0S/KB1+ 微控制器。
TO00/P21： 78K0S/KY1+ 和 78K0S/KU1+ 微控制器。

3.4 流程图

举例程序的流程图如下所示：



注 复位解除后，微控制器自动援引选项字节。在该举例程序中，援引选项字节设置以下内容：

- 内部高速振荡时钟(8 MHz (TYP.))用作系统时钟信号源。
- 可由软件停止内部低速振荡器。
- P34/$\overline{\text{RESET}}$ 引脚用作 $\overline{\text{RESET}}$ 引脚。

第四章　设置方法

本章描述了 16 位定时器/事件计数器 00 的 PPG 输出功能。

关于其他的初始设置，参见 **78K0S/Kx1+ 举例程序（初始设置）LED 灯开关控制的使用说明**。关于中断，参见 **78K0S/Kx1+举例程序（中断）由开关输入产生外部中断的使用说明**。关于低压检测（LVI），参见 **78K0S/Kx1+举例程序（低压检测）电压低于 2.7V 时的复位使用说明**。

关于如何设置寄存器，参见各自产品用户手册。（**78K0S/KU1+**，**78K0S/KY1+**，**78K0S/KA1+**， **78K0S/KB1+**）。

关于汇编器指令，参见 **78K/0S 系列指令用户手册**。

4.1　设置 16 位定时器/事件计数器 00 的PPG输出功能

使用 16 位定时器/事件计数器 00 的 PPG 输出功能时，需用以下九类寄存器：

- 捕获/比较控制寄存器 00 (CRC00)。
- 16 位定时器捕获/比较寄存器 000 (CR000)。
- 16 位定时器捕获/比较寄存器 010 (CR010)。
- 预分频模式寄存器 00 (PRM00)。
- 16 位定时器输出控制寄存器 00 (TOC00)。
- 16 位定时器模式控制寄存器 00 (TMC00)。
- 端口寄存器 x (Px)[注]。
- 端口模式寄存器 x (PMx)[注]。
- 端口模式控制寄存器 x (PMCx)[注]。

注　因为 PPG 输出功能将 TO00 引脚用作定时器输出，所以需将 Px、PMx 和 PMCx 寄存器进行如下设置：

	Px 寄存器	PMx 寄存器	PMCx 寄存器
78K0S/KA1+ 和 78K0S/KB1+ 微控制器	P31 = 0	PM31 = 0	不需要设置
78K0S/KY1+ 和 78K0S/KU1+微控制器	P21 = 0	PM21 = 0	PMC21 = 0

<将 16 位定时器/事件计数器 00 用作 PPG 输出时，基本操作设置步骤举例>

<1> 设置 CRC00 寄存器。

<2> 给 CR000 和 CR010 寄存器设置任意值(0000H < CR010 < CR000 ≤ FFFFH)。

<3> 用 PRM00 寄存器设置计数时钟。

<4> 设置 TOC00 寄存器。

<5> 设置 TMC00 寄存器： 开始运行。

注意事项 步骤<1> 至<4>顺序不分先后。

(1) 设置 CRC00 寄存器

该寄存器控制 CR000 和 CR010 寄存器的操作。。

图 4-1. 捕获/比较控制寄存器 00 (CRC00)的格式

CRC00

0	0	0	0	0	CRC002	CRC001	CRC000

CR000操作模式的选择

0	用作比较寄存器操作。
1	用作捕获寄存器操作。

CR000 捕获触发器的选择

0	在TI010引脚的有效沿捕获。
1	在TI000引脚有效沿的反转相位捕获。

CR010操作模式的选择

0	用作比较寄存器操作。
1	用作捕获寄存器操作。

注意事项 1. 设置 CRC00 寄存器之前必须停止定时器的操作。

2. 当使用 TMC00 寄存器选择基于 TM00 和 CR000 匹配而清零并启动的模式时，不要指定 CR000 寄存器作为捕获寄存器。

(2) 设置 CR000 寄存器

该寄存器兼有捕获寄存器和比较寄存器的功能。

图 4-2. 16 位定时器捕获/比较寄存器 000 (CR000)的格式

CR000

CR000 用作比较寄存器

CR000 的设定值与 16 位定时器计数器 00 (TM00) 的计数值比较，若二者匹配则产生中断请求 (INTTM000)。

注意事项 **1.** **基于 TM00 和 CR000 匹配而清零并启动的模式下，给 CR000 寄存器设置一个 0000H 之外的值。在自运行模式或在由 TI000 引脚的有效沿进入的清零并启动模式下，若 CR000 寄存器设置为 0000H，发生溢出后（FFFFH），当寄存器的值由 0000H 变成 0001H 时，产生一个中断请求 (INTTM000) 。**

2. **如果 CR000 寄存器的新值小于 16 位定时器计数器 00(TM00)的值，那么 TM00 寄存器继续计数，直至溢出，然后再从 0 开始重新计数。如果 CR000 寄存器的新值小于原先值，在 CR000 寄存器的值改变之后，定时器必须复位和重新启动。**

3. **不保证 TM00 计数器停止后 CR000 寄存器的值。**

4. **若 CR000 寄存器设置为比较模式，即使输入捕获触发信号，也不会执行捕获操作。**

5. **在 TM00 计数器操作期间改变 CR000 寄存器的设置可能会出错。**

(3) 设置 CR010 寄存器

该寄存器兼有捕获寄存器和比较寄存器的功能。

图 4-3. 16 位定时器捕获/比较寄存器 010(CR010)的格式

CR010

CR010 用作比较寄存器

CR010 的设定值与 16 位定时器计数器 00(TM00)的计数值比较，若二者匹配则产生中断请求 (INTTM010)。

注意事项 **1.** **在自运行模式下或在由 TI000 引脚的有效沿进入的清零并启动模式下，若 CR010 设置为 0000H，发生溢出后(FFFFH)，当寄存器的值由 0000H 变成 0001H 时，产生一个中断请求 (INTTM010)。**

2. **如果 CR010 寄存器的新值小于 TM00 计数器的值，TM00 寄存器继续计数，直至溢出，然后再从 0 开始重新计数。如果 CR010 寄存器的新值小于原先值，在 CR010 寄存器的值改变之后，定时器必须复位和重新启动。**

3. **不保证 TM00 计数器停止后 CR010 寄存器的值。**

4. **若将 CR010 寄存器设置为比较模式，即使输入捕获触发信号，也不会执行捕获操作。**

5. **在 TM00 计数器操作期间改变 CR010 寄存器的设置可能会出错。**

(4) 设置 PRM00 寄存器

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

图 4-4. 预分频模式寄存器 00 (PRM00)的格式

PRM00

ES110	ES100	ES010	ES000	0	0	PRM001	PRM000

计数时钟(fsam) 的选择

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引脚有效沿的选择

0	0	下降沿
0	1	上升沿
1	0	设置禁止
1	1	双边沿

TI010引脚有效沿的选择

0	0	下降沿
0	1	上升沿
1	0	设置禁止
1	1	双边沿

注 **外部时钟脉冲需要长于两个内部时钟(f_{XP})周期。**

备注 f_{XP}：供给外围硬件的时钟振荡频率

注意事项 1. 总是在停止定时器操作后给 PRM00 寄存器设置数据。

2. 当设定 TI000 引脚的有效沿作为计数时钟时，不要设置为由 TI000 引脚的有效沿清零并启动模式，也不要将 TI000 引脚用作捕获触发。

3. 在下列情况中，应对 TI0n0 引脚(n = 0，1)有效沿的检测情况加以注意。

<1> 系统复位后 TI0n0 引脚输入高电平并立即使能 TM00 。

→ 如果指定 TI0n0 引脚为上升沿或双边沿有效，则使能 TM00 后，随即检测到上升沿。

<2> TI0n0 引脚处于高电平时，停止 TM00 操作，且 TI0n0 引脚输入低电平后立即使能 TM00。

→ 如果指定 TI0n0 引脚为下降沿或双边沿有效，则使能 TM00 后随即检测到下降沿。

<3> TI0n0 引脚处于低电平时，停止 TM00 操作，且 TI0n0 引脚输入高电平后，立即使能 TM00。

→ 如果指定 TI0n0 引脚为上升沿或双边沿有效，则使能 TM00 后随即检测到上升沿。

注意事项 4. TI000 引脚的有效沿用作计数时钟时，对其以 f_{XP} 进行采样来抑制噪声。当采样有效沿且两次检测到有效电平时才执行捕获操作，因而抑制了尖脉冲噪声。

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

(5) 设置 TOC00 寄存器

该寄存器控制 16 位定时器/事件计数器 00 输出控制器的工作。其用于设置/复位定时器输出 F/F、允许或禁止输出反相、定时器输出(TO00 引脚输出)、单次脉冲输出操作以及用软件设置单次脉冲输出触发器。

图 4-5. 16位定时器输出控制寄存器 00 (TOC00)的格式

TOC00

0	OSPT00	OSPE00	TOC004	LVS00	LVR00	TOC001	TOE00

定时器输出控制

0	禁止输出(固定输出为0电平)。
1	允许输出。

CR000 和TM00匹配时定时器F/F的 控制

0	禁止反相操作。
1	允许反相操作。

定时器输出F/F 状态设置

0	0	不改变。
0	1	复位(0) 定时器输出F/F。
1	0	设置(1) 定时器输出F/F。
1	1	设置禁止。

CR010 和TM00匹配时定时器F/F的 控制

0	禁止反相操作。
1	允许反相操作。

单次脉冲输出操作控制

0	连续脉冲输出模式
1	单次脉冲输出模式[注]

用软件控制单次脉冲输出触发器

0	不用单次脉冲输出触发器
1	用单次脉冲输出触发器

注 单次脉冲输出模式通常仅用于自由运行模式和由 TI000 引脚有效沿设置的清零并启动模式中。在基于 TM00 和 CR000 匹配而清零并启动模式下，因为不发生溢出，所以不可能有单次脉冲输出。

注意事项 **1. 在设置除 OSPT00 外的其他位前，必须停止定时器操作。**

2. 如果读取 LVS00 和 LVR00 ，读出值为 0。

3. 设置数据后 OSPT00 自动清零，故读出值为 0。

4. 在非单次脉冲输出模式下，不要将 OSPT00 设置为 1 。

5. 连续将 OSPT00 设置为(1)时，写入间隔至少需要用 PRM00 寄存器所选择计数时钟的两个周期。

6. 当 TOE00 为 0 时，用 8 位存储器操作指令同时设置 TOE00、LVS00 和 LVR00。当 TOE00 为 1 时，用 1 位存储器操作指令可设置 LVS00 和 LVR00 。

7. 当 TI010/TO00/Pxx 引脚用作有效沿的输入引脚(TI010)时，则不能将其用作定时器输出引脚(TO00)。当 TI010/TO00/Pxx 引脚用作定时器输出引脚(TO00)时，则不能将其用作有效沿的输入引脚(TI010)。

(6) 设置 TMC00 寄存器

该寄存器设置 16 位定时器/事件计数器 00 的 操作模式、 TM00 计数器清零模式、输出时序以及检测溢出。

图 4-6. 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)的溢出检测

0	未检测到溢出。
1	检测到溢出。

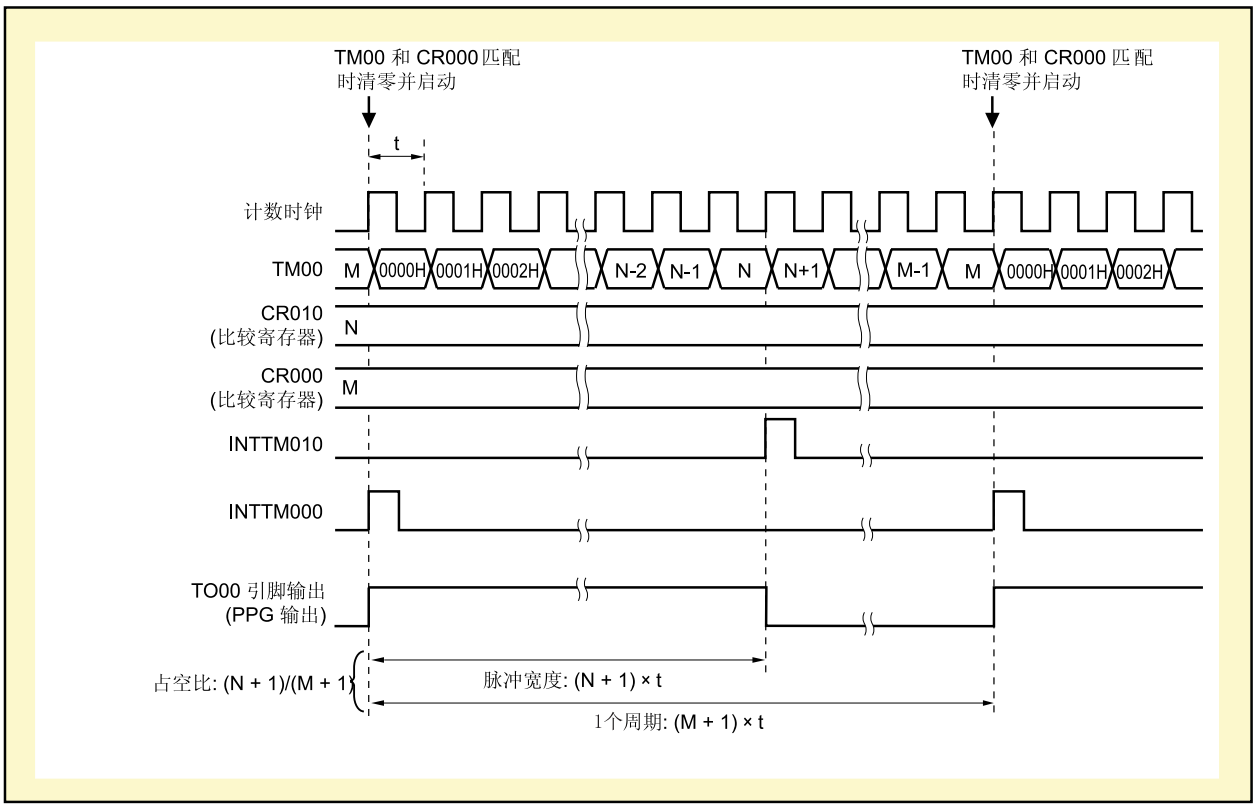
			操作模式和清除模式的选择	TO00 反相时序的选择	中断请求的产生
0	0	0	操作停止(TM00清除为0)。	不改变	不产生
0	0	1			
0	1	0	自运行模式	TM00 和CR000匹配或TM00 和CR010匹配 。	<当用作比较寄存器时> TM00 和CR000匹配或TM00 和CR010匹配的基础上产生。 <当用作捕获寄存器时> 在TI000引脚或TI010引脚的有效沿产生。
0	1	1		TM00 和CR000匹配或TM00 和CR010匹配或TI000引脚有效沿。	
1	0	0	在TI000引脚的有效沿发生清零并启动。	–	
1	0	1			
1	1	0	TM00 和CR000匹配时发生清零并启动。	TM00 和CR000匹配或TM00 和CR010匹配。	
1	1	1		TM00 和CR000匹配或TM00 和CR010匹配或TI000引脚有效沿。	

注意事项 1. 当一个非 0 值和 0（操作停止模式）被分别设置到 TMC002 和 TMC003 时启动 TM00 计数器的操作。若要停止操作，应分别将 TMC002 和 TMC003 设置为 0。

2. 停止定时器操作后写入除 OVF00 标志外的其他位。
3. 当定时器停止时，即使信号输入到 TI000/TI010 引脚也不发生定时器计数和定时器中断。
4. 当 TI000 引脚的有效沿选作计数时钟时，应在设置为停止模式或系统时钟停止模式前停止定时器操作。否则，当系统时钟启动时定时器可能会出错。
5. 停止定时器操作后用 PRM00 寄存器的位 4 和位 5 设置 TI000 引脚的有效沿。
6. 如果在基于 TM00 和 CR000 匹配而清零并启动模式下，或基于 TI000 引脚的有效沿而清零并启动模式下，或选择为自运行模式下，当 CR000 寄存器的设定值是 FFFFH 且 TM00 计数器值从 FFFFH 至 0000H 变化时，OVF00 标志设置为 1。
7. TM00 计数器溢出后，在下一个计数时钟计数之前（TM00 计数器变为 0001H 前），即使清除了 OVF00 标志，TM00 计数器仍被重置并禁止清零。
8. 在计数时钟下降沿执行捕获操作，而中断请求 (INTTM0n0： n = 0， 1) 发生在下一个计数时钟的上升沿。

[举例]　　设置一个PPG输出周期为200 μs 、脉冲宽度为20 μs并执行PPG输出(计数时钟： $f_{XP}/2^2$ (f_{XP} = 8 MHz))
(与该举例源程序内容相同)

图 4-7. PPG 输出时序举例



(1) 寄存器设置

<1> CRC00

0	0	0	0	0	0	x	0

CR000 操作模式的选择

0	用作比较 寄存器操作。

CR000 捕获触发器的选择

x	(因为CR000用作比较寄存器所以设置无效。)

CR010操作模式的选择

0	用作比较寄存器操作。

<2>　CR010

设定值(N)：　39

• 计数时钟 fsam = 8 [MHz]/2^2 = 2 [MHz]

• 脉冲宽度 20 [μs] = (N + 1)/2 [MHz]

→ N = 20 [μs] × 2 [MHz] – 1 = 39

<3>　CR000

设定值(M)：　399

• 计数时钟 fsam = 8 [MHz]/2^2 = 2 [MHz]

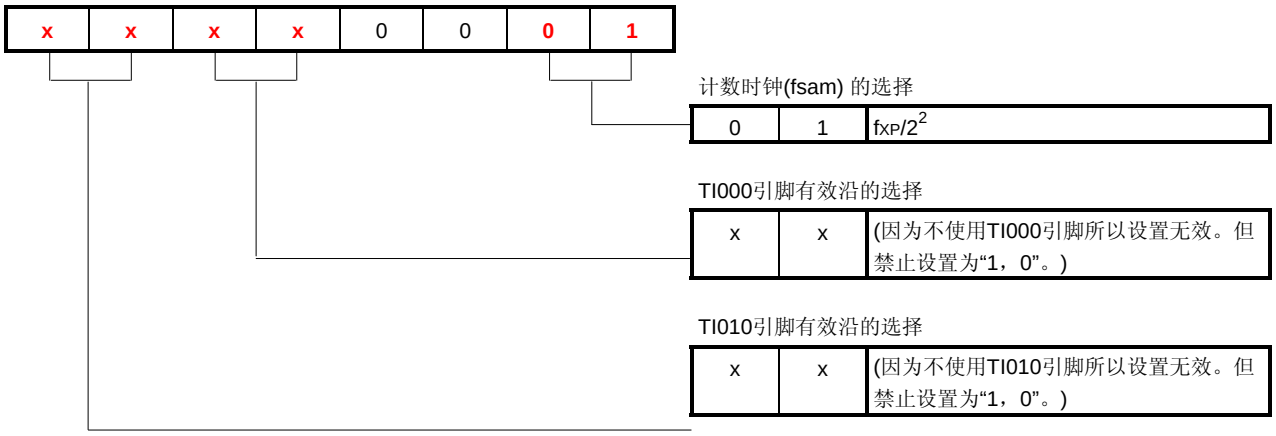
• 1 个周期 200 [μs] = (M + 1)/2 [MHz]

→ M = 200 [μs] × 2 [MHz] – 1 = 399

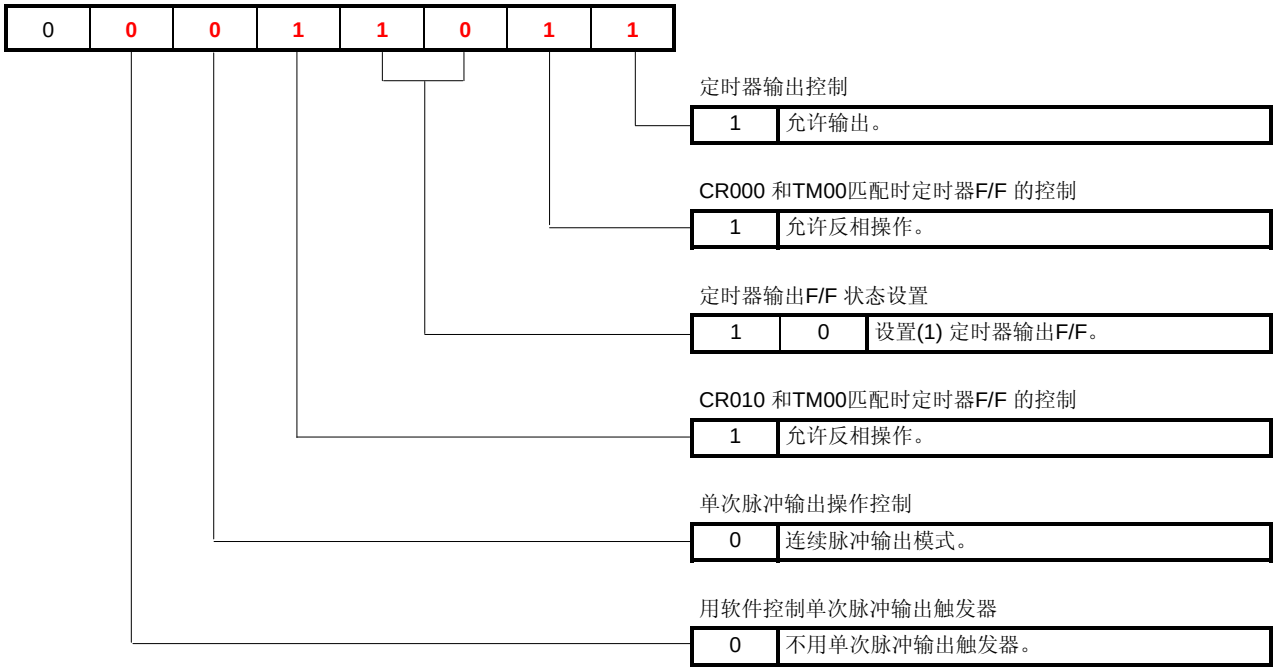
注意事项　在以下范围设置 CR000 和 CR010 的数值：

0000H < CR010 < CR000 ≤ FFFFH

<4>　PRM00



<5>　TOC00



<6>　TMC00

0	0	0	0	1	1	0	0

16位定时器计数器00 (TM00)的溢出检测

0	未检测到溢出。

			操作模式和清除模式的选择	TO00反相时序的选择	中断请求的产生
1	1	0	TM00 和CR000匹配时发生清零并启动	TM00 和CR000匹配或 TM00和CR010匹配	TM00 和CR000匹配或 TM00和CR010匹配时产生

<7>　Px, PMx, PMCx

	Px 寄存器	PMx 寄存器	PMCx 寄存器
78K0S/KA1+和 78K0S/KB1+微控制器	P31 = 0	PM31 = 0	设置禁止
78K0S/KY1+ 和 78K0S/KU1+微控制器	P21 = 0	PM21 = 0	PMC21 = 0

(2)　举例程序

在下面实例中，**(1) 寄存器设置**中的“x”设置为“0”。

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

```
CLR1   P3.1
CLR1   PM3.1
MOV    CRC00, #00000000B
MOVW   CR000, #399
MOVW   CR010, #39
MOV    PRM00, #00000001B
MOV    TOC00, #00011011B
MOV    TMC00, #00001100B
```

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

```
P3.1 = 0;
PM3.1 = 0;
CRC00 = 0b00000000;
CR000 = 399;
CR010 = 39;
PRM00 = 0b00000001;
TOC00 = 0b00011011;
TMC00 = 0b00001100;
```

[举例源程序中的摘录]

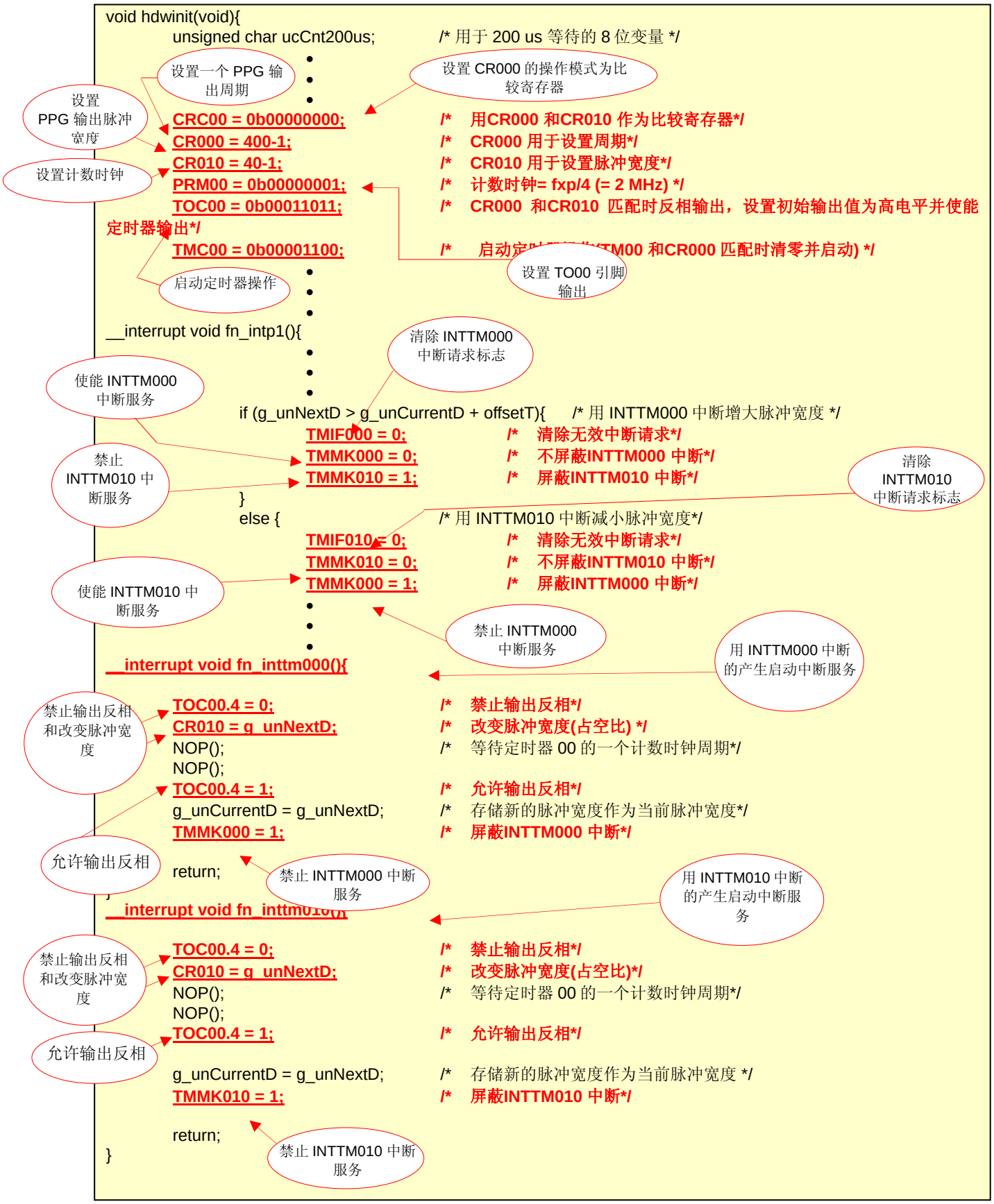
从附录 A 程序清单中摘录与 16 位定时器/事件计数器 00 功能相关的内容，如下所示： (和上面 [举例] 中提到的内容相同)。

(1) 汇编语言

```
XMAIN   CSEG    UNIT
RESET_START:
                •
                •
        MOV     CRC00, #00000000B       ;  CR000 用作比较寄存器
        MOVW    AX,     #400-1
        MOVW    CR000,  AX              ;  CR000 用于设置周期
        MOVW    AX,     #40-1
        MOVW    CR010,  AX              ;  CR010 用于设置脉冲宽度
        MOV     PRM00, #00000001B       ;  计数时钟= fxp/4 (= 2 MHz)
        MOV     TOC00, #00011011B       ;  CR000 和 CR010 匹配时输出反相，设置初始输出值为高电平，并使
能定时器输出
        MOV     TMC00, #00001100B       ;  启动定时器操作(TM00 和 CR000 匹配时清零并启动)
                •
                •
INTERRUPT_P1:
                •
                •
        BNC     $DECDUTY                ;  如果新的脉冲宽度<当前脉冲宽度，则转移
        CLR1    TMIF000                 ;  清除无效的中断请求
        CLR1    TMMK000                 ;  不屏蔽 INTTM000 中断
        SET1    TMMK010                 ;  屏蔽 INTTM010 中断
        BR      $END_INTP1              ;  转移至 END_INTP1
DECDUTY:
        CLR1    TMIF010                 ;  清除无效的中断请求
        CLR1    TMMK010                 ;  不屏蔽 INTTM010 中断
        SET1    TMMK000                 ;  屏蔽 INTTM000 中断
END_INTP1:
        POP     AX                      ;  恢复 AX 寄存器数据
        RETI                            ;  从中断服务返回
                •
                •
INTERRUPT_TM000:
        CLR1    TOC00.4                 ;  禁止输出反相
        MOVW    AX,     NEXTD
        MOVW    CR010,  AX              ;  改变脉冲宽度(占空比)
        NOP                             ;  等待定时器 00 的一个计数时钟周期
        NOP
        SET1    TOC00.4                 ;  允许输出反相
        MOVW    CURRENTD, AX            ;  存储新的脉冲宽度作为当前脉冲宽度
        SET1    TMMK000                 ;  屏蔽 INTTM000 中断
        POP     AX                      ;  恢复 AX 寄存器数据
        RETI                            ;  从中断服务返回
INTERRUPT_TM010:
        PUSH    AX                      ;  将 AX 寄存器数据存入堆栈
        CLR1    TOC00.4                 ;  禁止输出反相
        MOVW    AX,     NEXTD
        MOVW    CR010,  AX              ;  改变脉冲宽度(占空比)
        NOP                             ;  等待定时器 00 的一个计数时钟周期
        NOP
        SET1    TOC00.4                 ;  允许输出反相
        MOVW    CURRENTD, AX            ;  存储新的脉冲宽度作为当前脉冲宽度
        SET1    TMMK010                 ;  屏蔽 INTTM010 中断
        POP     AX                      ;  恢复 AX 寄存器数据
        RETI                            ;  从中断服务返回
```

设置一个 PPG 输出周期
设置 CR000 的操作模式为比较寄存器
设置 PPG 输出脉冲宽度
设置计数时钟
设置 TO00 引脚输出
启动定时器操作
清除 INTTM000 中断请求标志
使能 INTTM000 中断服务
禁止 INTTM010 中断服务
清除 INTTM010 中断请求标志
使能 INTTM010 中断服务
禁止 INTTM000 中断服务
用 INTTM000 中断的产生启动中断服务
禁止输出反相和改变脉冲宽度
允许输出反相
禁止 INTTM000 中断服务
用 INTTM010 中断的产生启动中断服务
禁止输出反相和改变脉冲宽度
允许输出反相
禁止 INTTM010 中断服务

(2) C 语言




```
void hdwinit(void){
        unsigned char ucCnt200us;          /* 用于 200 us 等待的 8 位变量 */
        •
        •
        •
        CRC00 = 0b00000000;                /*  用CR000 和CR010 作为比较寄存器*/
        CR000 = 400-1;                     /*  CR000 用于设置周期*/
        CR010 = 40-1;                      /*  CR010 用于设置脉冲宽度*/
        PRM00 = 0b00000001;                /*  计数时钟= fxp/4 (= 2 MHz) */
        TOC00 = 0b00011011;                /*  CR000  和CR010  匹配时反相输出，设置初始输出值为高电平并使能
定时器输出*/
        TMC00 = 0b00001100;                /*     启动定时器操作(TM00 和CR000 匹配时清零并启动) */
        •
        •
        •
__interrupt void fn_intp1(){
        •
        •
        •
    if (g_unNextD > g_unCurrentD + offsetT){     /* 用 INTTM000 中断增大脉冲宽度 */
        TMIF000 = 0;                       /*  清除无效中断请求*/
        TMMK000 = 0;                       /*  不屏蔽INTTM000 中断*/
        TMMK010 = 1;                       /*  屏蔽INTTM010 中断*/
    }
    else {                                 /* 用 INTTM010 中断减小脉冲宽度*/
        TMIF010 = 0;                       /*  清除无效中断请求*/
        TMMK010 = 0;                       /*  不屏蔽INTTM010 中断*/
        TMMK000 = 1;                       /*  屏蔽INTTM000 中断*/
        •
        •
        •
__interrupt void fn_inttm000(){
        TOC00.4 = 0;                       /*  禁止输出反相*/
        CR010 = g_unNextD;                 /*  改变脉冲宽度(占空比) */
        NOP();                             /*  等待定时器 00 的一个计数时钟周期*/
        NOP();
        TOC00.4 = 1;                       /*  允许输出反相*/
        g_unCurrentD = g_unNextD;          /*  存储新的脉冲宽度作为当前脉冲宽度*/
        TMMK000 = 1;                       /*  屏蔽INTTM000 中断*/

        return;
}
__interrupt void fn_inttm010(){
        TOC00.4 = 0;                       /*  禁止输出反相*/
        CR010 = g_unNextD;                 /*  改变脉冲宽度(占空比)*/
        NOP();                             /*  等待定时器 00 的一个计数时钟周期*/
        NOP();
        TOC00.4 = 1;                       /*  允许输出反相*/

        g_unCurrentD = g_unNextD;          /*  存储新的脉冲宽度作为当前脉冲宽度 */
        TMMK010 = 1;                       /*  屏蔽INTTM010 中断*/

        return;
}
```

4.2 设置和改变PPG输出占空比

4.2.1 设置PPG输出占空比

该举例程序中， 用 16 位定时器/事件计数器 00 的 PPG 输出功能从 TO00 引脚输出一个脉冲。

- 脉冲宽度= (N + 1)/fsam
- 1 个周期= (M + 1)/fsam
- PPG 输出占空比= (N + 1)/(M + 1) × 100

备注 N： CR010 寄存器设定值。
M： CR000 寄存器设定值。
fsam： 16 位定时器/事件计数器 00 的计数时钟频率。
0000H < N < M ≤ FFFFH

计算举例： CR010 寄存器的设定值是 39 且 CR000 寄存器的设定值是 399 (在以 fsam = 2 MHz 操作期间)。
• 脉冲宽度= (N + 1)/fsam = (39 + 1)/2 [MHz] = 20 [μs]
• 1 个周期 = (M + 1)/fsam = (399 + 1)/2 [MHz] = 200 [μs]
• PPG 输出占空比= (N + 1)/(M + 1) × 100 = (39 + 1)/(399 + 1) × 100 = 10 [%]

此外， 依照开关输入次数 CR010 寄存器的设定值而发生变化，PPG 的输出占空比随之改变。

开关输入次数[注]	CR010 寄存器设定值	CR000 寄存器 设定值	PPG输出占空比
0	39	399	10% ((39 + 1)/(399 + 1) × 100)
1	119		30% ((119 + 1)/(399 + 1) × 100)
2	199		50% ((199 + 1)/(399 + 1) × 100)
3	279		70% ((279 + 1)/(399 + 1) × 100)
4	359		90% ((359 + 1)/(399 + 1) × 100)
5	279		70% ((279 + 1)/(399 + 1) × 100)
6	199		50% ((199 + 1)/(399 + 1) × 100)
7	119		30% ((119 + 1)/(399 + 1) × 100)

注 第八次开关输入后 PPG 输出占空比从第零次开关输入重复。

4.2.2 改变CR010 的设定值(PPG输出占空比)

该举例程序中，为了增大或减小 PPG 输出占空比，主要是在 INTTM000 中断期间和在 INTTM010 中断期间分别改变 CR010 寄存器的值。但是考虑到‘从 INTTM010 中断产生，经由改变 CR010 寄存器的值，到允许定时器输出反相’这段时间偏差，需按以下方法执行：

- 当 ‘新的脉冲宽度> 当前脉冲宽度+偏差时间’时： 在 INTTM000 中断服务期间改变寄存器的值。
- 当 ‘新的脉冲宽度<当前脉冲宽度+偏差时间’时： 在INTTM010中断服务期间改变寄存器的值。

如果用上述方法不能改变 CR010 寄存器的值，那么在一个 PPG 输出周期中可能不发生 CR010 和 TM00 匹配，也可能发生 CR010 和 TM00 的多次匹配并且每一次匹配都可能会引起 TO00 引脚输出电平反相。

在 INTTM0x0 (x = 0, 1)中断服务期间用以下步骤改变 CR010 寄存器的值：

<1> 禁止 TM00 和 CR010 匹配时定时器输出反相操作 (TOC004 = 0)。
<2> 重写 CR010。
<3> 等待 TM00 计数时钟的一个周期。
<4> 允许 TM00 和 CR010 匹配时定时器输出反相操作(TOC004 = 1)。

以下是改变 CR010 寄存器值的时序图举例。

图 4-8. “新的脉冲宽度> 当前脉冲宽度+偏差时间”时序图举例

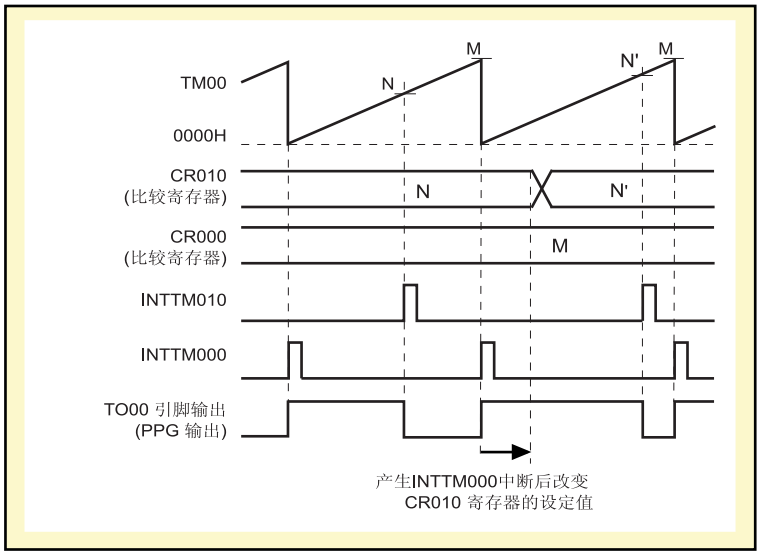
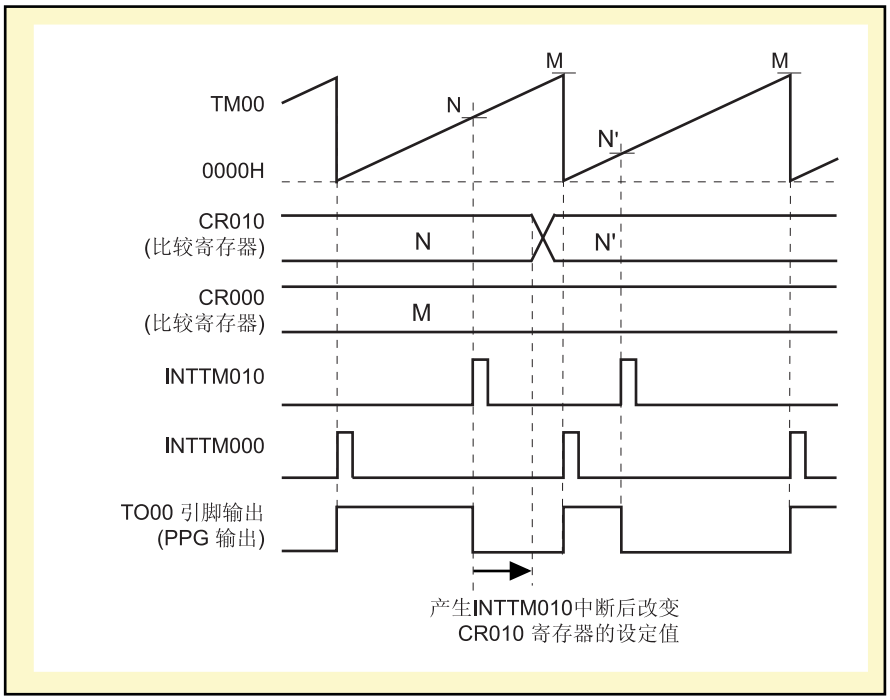


图 4-9.　“新的脉冲宽度<当前脉冲宽度+偏差时间”时序图举例



关于在定时器工作期间如何改变比较寄存器以及注意事项的详情，参见各自产品用户手册中的 **6.5 关于 16 位定时器/事件计数器 00 的注意事项(78K0S/KU1+， 78K0S/KY1+， 78K0S/KA1+， 78K0S/KB1+)**。

[举例源程序中的摘录]

从**附录 A　程序清单**中摘录与改变寄存器值相关的内容，如下所示：

(1)　汇编语言

```
OFFSETT     EQU   43/4          ;  用于校准 INTTM010 中断服务的时间
                  •
                  •
                  •
INTERRUPT_P1:
                  •
                  •
                  •
            MOVW  AX,    CURRENTD   ;  读取当前脉冲宽度
            ADDW  AX,    #OFFSETT   ;  加上偏差时间
            XCH   A,     X
            SUB   A,     [HL]       ;  与新的脉冲宽度比较
            XCH   A,     X
            SUBC  A,     [HL+1]
            BNC   $DECDUTY          ;  如果新的脉冲宽度< 当前脉冲宽度，则转移
            CLR1  TMIF000           ;  清除无效的中断请求
            CLR1  TMMK000           ;  不屏蔽 INTTM000 中断
            SET1  TMMK010           ;  屏蔽 INTTM010 中断
            BR    $END_INTP1        ;  转移至 END_INTP1
UTY:
            CLR1  TMIF010           ;  清除无效的中断请求
            CLR1  TMMK010           ;  不屏蔽 INTTM010 中断
            SET1  TMMK000           ;  屏蔽 INTTM000 中断
                  •
                  •
                  •
INTERRUPT_TM000:
            PUSH  AX                ;  将 AX 寄存器数据存入堆栈
            CLR1  TOC00.4           ;  禁止输出反转
            MOVW  AX,    NEXTD
            MOVW  CR010, AX         ;  改变脉冲宽度(占空比)
            NOP                     ;  等待定时器 00 的一个计数时钟周期
            NOP
            SET1  TOC00.4           ;  允许输出反相
            MOVW  CURRENTD, AX      ;  存储新的脉冲宽度作为当前脉冲宽度
            SET1  TMMK000           ;  屏蔽 INTTM000 中断
            POP   AX                ;  恢复 AX 寄存器 数据
            RETI                    ;  从中断服务返回
INTERRUPT_TM010:
            PUSH  AX                ;  将 AX 寄存器数据存入堆栈
            CLR1  TOC00.4           ;  禁止输出反相
            MOVW  AX,    NEXTD
            MOVW  CR010, AX         ;  改变脉冲宽度(占空比)
            NOP                     ;  等待定时器 00 的一个计数时钟周期
            NOP
            SET1  TOC00.4           ;  允许输出反相
            MOVW  CURRENTD, AX      ;  存储新的脉冲宽度作为当前脉冲宽度
            SET1  TMMK010           ;  屏蔽 INTTM010 中断
            POP   AX                ;  恢复 AX 寄存器数据
            RETI                    ;  从中断服务返回
```

将当前脉冲值加偏差时间的和与新的脉冲宽度进行比较

如果“新的脉冲宽度> 当前脉冲值+偏差时间”，则使用 INTTM000 中断

如果“新的脉冲宽度<当前脉冲值+偏差时间”，则使用 INTTM010 中断

在 INTTM000 中断产生期间[注]改变占空比 (CR010)

在 INTTM010 中断产生期间[注]改变占空比 (CR010)

注　该举例程序中，因为在中断服务期间改变 CR010 寄存器的设定值，所以，当改变设定值时不需要禁止或使能所用的中断服务。

(2) C 语言

```
#define offsetT 56/4                    /*  用于校准 INTTM010 中断服务的时间*/
                                        •
                                        •
                                        •
__interrupt void fn_intp1(){
                                        •
                                        •
                                        •
        if (g_unNextD > g_unCurrentD + offsetT){    /* 用 INTTM000 中断增大脉冲宽度 */
                TMIF000 = 0;            /*  清除无效的中断请求*/
                TMMK000 = 0;            /*  不屏蔽 INTTM000 中断 */
                TMMK010 = 1;            /*  屏蔽 INTTM010 中断 */
        }
        else {                          /*  用 INTTM010 中断减小脉冲宽度 */
                TMIF010 = 0;            /*  清除无效的中断请求 */
                TMMK010 = 0;            /*  不屏蔽 INTTM010 中断 */
                TMMK000 = 0;            /*  屏蔽 INTTM000 中断 */
                •
                •
                •
__interrupt void fn_inttm000(){

        TOC00.4 = 0;                    /*  禁止输出反相 */
        CR010 = g_unNextD;              /*  改变脉冲宽度(占空比) */
        NOP();                          /*  等待定时器 00 的一个计数时钟周期*/
        NOP();
        TOC00.4 = 1;                    /*  允许输出反相 */
        g_unCurrentD = g_unNextD;       /*  存储新的脉冲宽度作为当前脉冲宽度*/
        TMMK000 = 1;                    /*  屏蔽 INTTM000 中断 */

        return;
}
__interrupt void fn_inttm010(){

        TOC00.4 = 0;                    /*  禁止输出反相 */
        CR010 = g_unNextD;              /*  改变脉冲宽度(占空比) */
        NOP();                          /*  等待定时器 00 的一个计数时钟周期*/
        NOP();
        TOC00.4 = 1;                    /*  允许输出反相 */
        g_unCurrentD = g_unNextD;       /*  存储新的脉冲宽度作为当前脉冲宽度*/
        TMMK010 = 1;                    /*  屏蔽 INTTM010 中断 */

        return;
```

将当前脉冲值加偏差时间的和与新的脉冲宽度进行比较

如果“新的脉冲宽度>当前脉冲值+偏差时间”，则使用 INTTM000 中断

如果“新的脉冲宽度<当前脉冲值+偏差时间”，则使用 INTTM010 中断

在 INTTM000 中断产生期间[注]改变占空比 (CR010)

在 INTTM010 中断产生期间[注]改变占空比 (CR010)

注　该举例程序中，因为在中断服务期间改变 CR010 寄存器的设定值，所以，当改变设定值时不需要禁止或使能所用的中断服务。

4.3　设置抖动检测时间

在该举例程序中为了处理开关输入（INTP1 中断产生）期间的抖动，16 位定时器/事件计数器 00 中断（INTTM000）的产生已经考虑了消除 10ms 或更短时间的抖动。

- 抖动检测时间(T_C) = T' + T × (M − 1)

备注　T：INTTM000 中断周期
T'：从启动 INTP1 边沿检测开始到 INTP1 边沿检测后第一个 INTTM000 产生的时间($0 < T' \leq T$)
M：INTP1 边沿检测后 INTTM000 中断的次数

当设置 T × (M − 1) = 10 ms 时，

$T_C = T' + 10$ ms

$0 < T' \leq T$，因此，

$10 \text{ ms} < T_C \leq T + 10$ ms

↓

抖动检测时间(T_C) > 10 ms

计算举例：　INTTM000 中断周期(T)是 200 μs (参见 **4.2 设置和改变 PPG 输出占空比**)，INTP1 边沿检测后 INTTM000 中断的次数(M)是 51

$T_C = T' + T \times (M - 1)$

$= T' + 200\ [\mu s] \times (51 - 1)$

$= T' + 10000\ [\mu s]$

$= T' + 10\ [ms]$

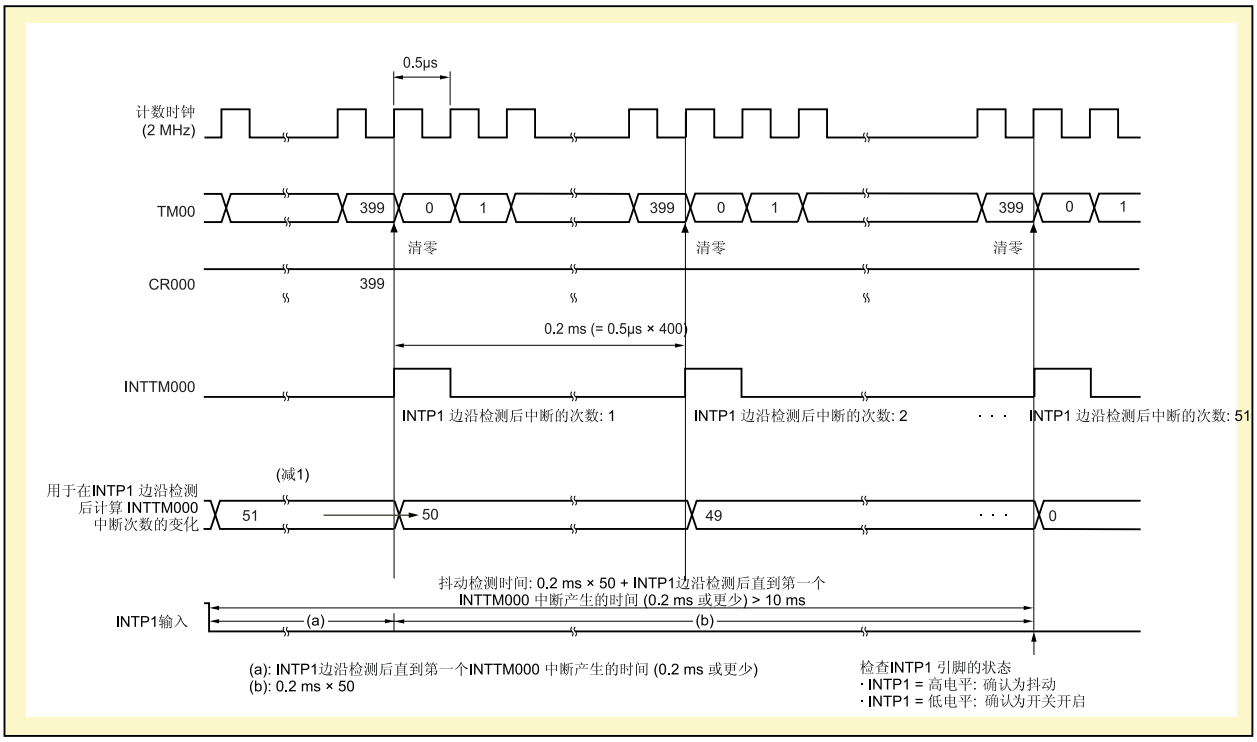
$0 < T' \leq 200\ \mu s$，因此，

$10 \text{ ms} < T_C \leq 10.2\ [ms]$

↓

抖动检测时间(T_C) > 10 ms

图 4-10.　抖动检测的时序图举例



第五章　使用系统仿真器SM+进行操作检验

利用由选择  图标所下载的汇编语言文件（源文件＋工程文件），本章介绍了如何使用 78K0S/Kx1+的系统仿真器 SM+运行举例程序。

注意事项　78K0S/Kx1+的系统仿真器 SM+不支持 78K0S/KU1+微控制器（同 2007 年 09 月）。因此，78K0S/KU1+微控制器的操作不能由 78K0S/Kx1+的系统仿真器 SM+进行检验。

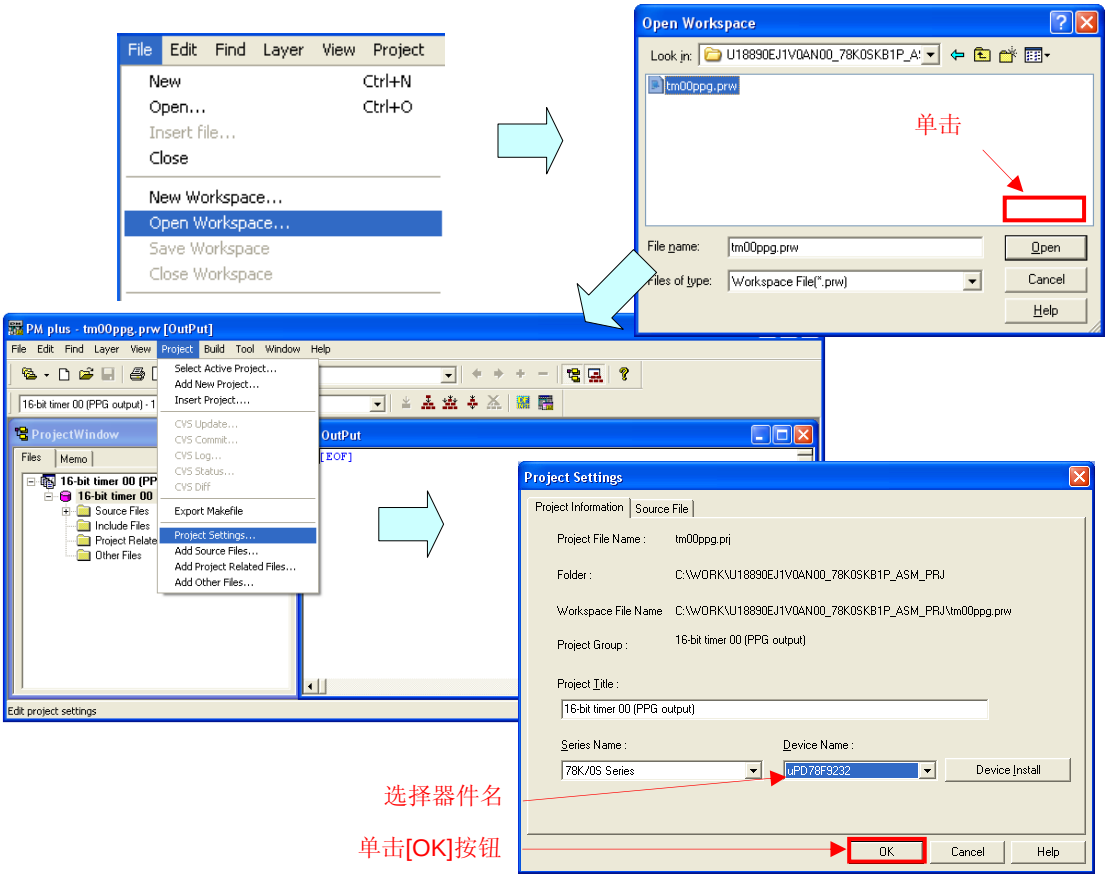
5.1　连编举例程序

为利用 78K0S/Kx1+的系统仿真器 SM+（以下简称“SM+”）来检验举例程序的操作，SM+必须在连编举例程序之后启动。本节介绍了操作顺序实例：从由集成开发环境 PM+连编举例程序开始，然后利用由选择  所下载的汇编语言文件（源文件＋工程文件），直到启动 SM+。关于如何编译其它下载的程序，参见 **78K0S/Kx1+举例程序启动指南用户手册**中的**第三章注册集成开发环境 PM+的工程并执行编译。**

(1) 启动 PM+。

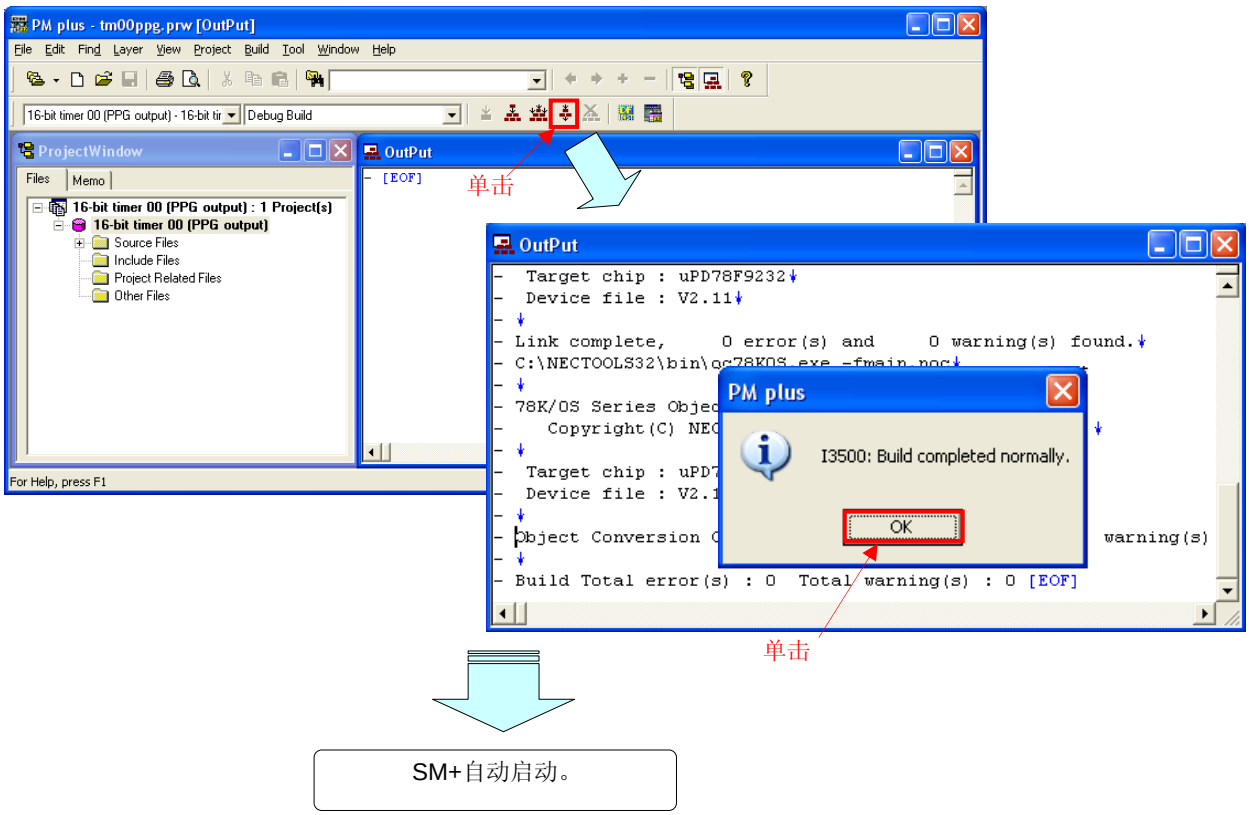
(2) 从[File]菜单中单击[Open Workspace]选项，然后选择“tm00ppg.prw”文件并单击[Open]按钮。将创建一个可自动读取源文件的工作空间。

(3) 从[Project]菜单中选择[Project Settings]选项。打开[Project Settings]对话框后，选择所用的器件名（缺省选择为具备最大容量 ROM 或 RAM 的器件），并单击 [OK]按钮。



(4) 单击  （[Build → Debug]按钮）。当源文件“main.asm”和“op.asm”正常编译时，将会显示“I3500: Build completed normally.”消息。

(5) 单击消息框中的[OK]按钮以启动 SM+。



5.2　SM+的操作

本节介绍了检验 SM+的 I/O 面板操作或时序图窗口操作的实例。
关于如何操作 SM+之详情，参见 **SM+系统仿真器操作用户手册**。

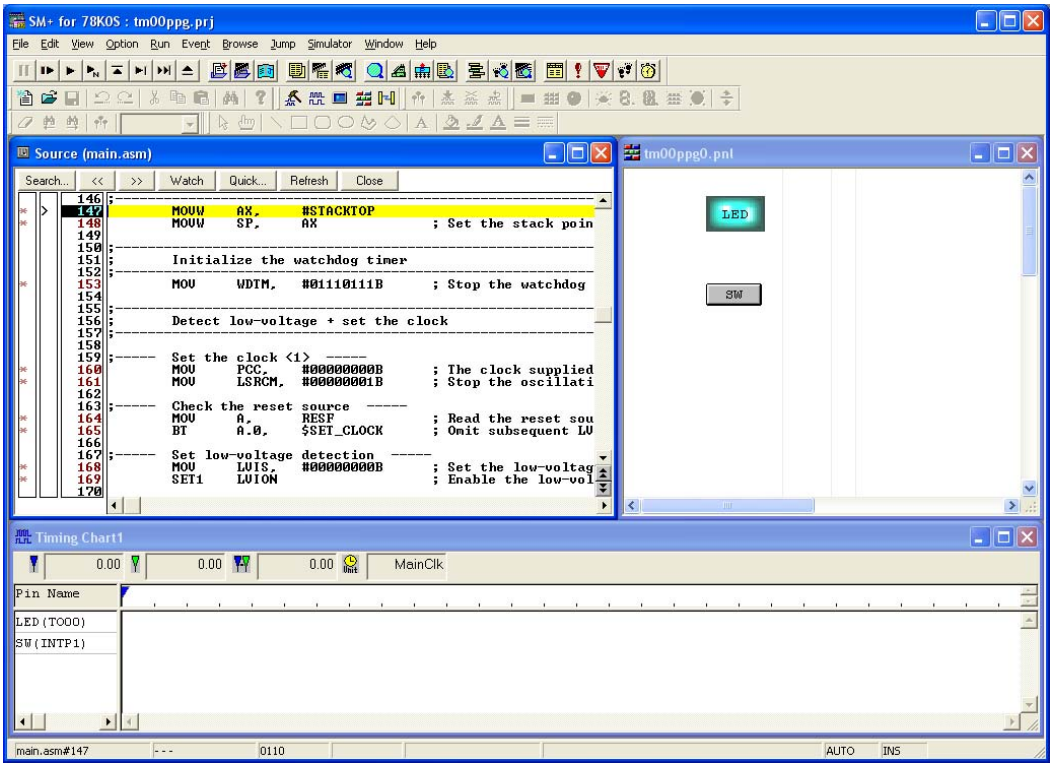
[专栏] 编译 链接错误

使用 PM+进行编译，当显示 “A006 File not found ‘C:\NECTOOLS32\LIB78K0S\s0sl.rel’”或“*** ERROR F206 Segment ‘@@DATA’ can’t allocate to memory - ignored.” 错误信息时，按照下列的步骤改变编译选项设置。

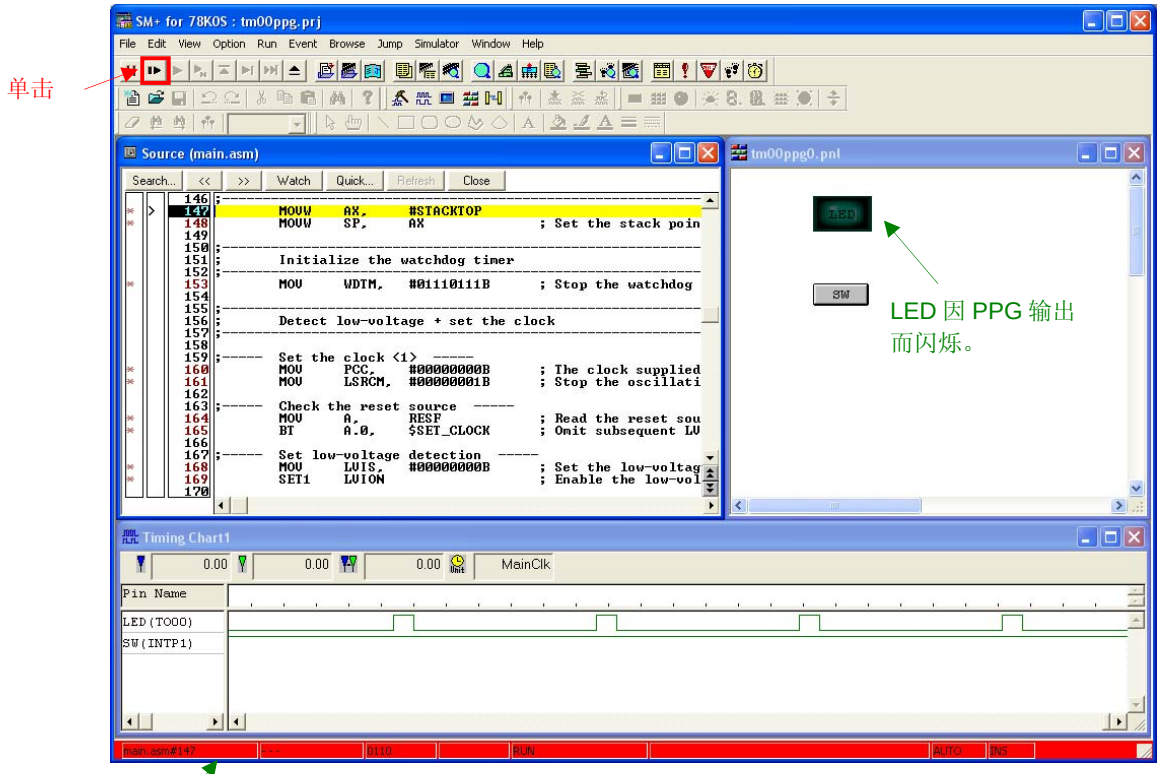
<1>从[Tool]菜单中选择[Compiler Options]选项。
<2>将会显示[Compiler Options]对话框。选择[Startup Routine]选项卡。
<3>取消选择[Using Fixed Area of Standard Library]复选框。（其它复选框不变。）

当取消 [Using Fixed Area of Standard Library]复选框时，允许使用一个位于 RAM 区域内被保护的 118 字节区域作为固定标准库域，但是，禁止使用标准库(例如 getchar 函数和 malloc 函数)。

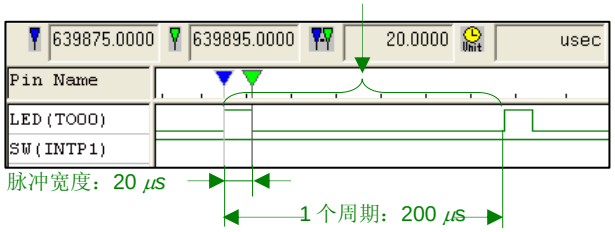
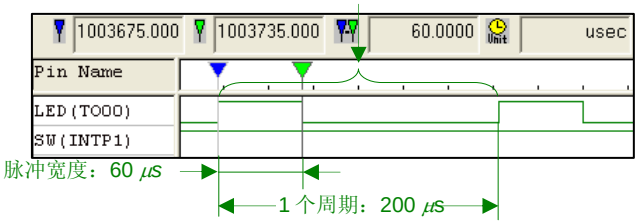
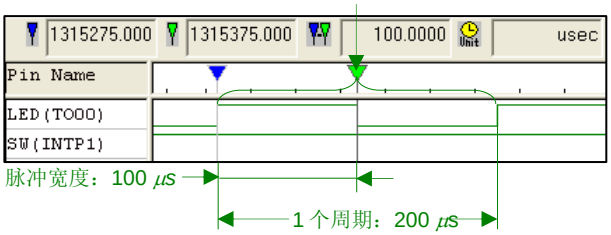
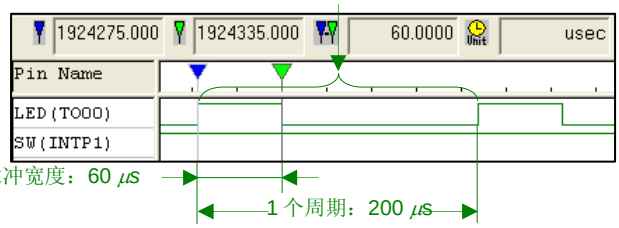
(1)　在 PM+（参见 5.1）中单击[Build → Debug]启动 SM+后，将显示以下界面：　　（此为使用汇编语言源文件时的界面。）



(2) 单击  （[Restart]按钮）。CPU 复位后将执行程序，并显示如下界面：



(3) 程序执行期间，在 I/O 面板窗口中单击[SW]按钮。
根据[SW]按钮输入次数来改变 PPG 输出占空比。
根据[SW]按钮输入次数检验 I/O 面板窗口中[LED]亮度及时序图窗口中方波的变化。

I/O面板窗口	时序图窗口
亮度：90%　LED 不点击。　SW	PPG 输出占空比为 10% 
亮度：70%　LED 点击一次。　SW	PPG 输出占空比为 30% 
亮度：50%　LED 点击两次。　SW	PPG 输出占空比为 50% 
• • •	• • •
亮度：70%　LED 点击 7 次。　SW	PPG 输出占空比为 30% 

注

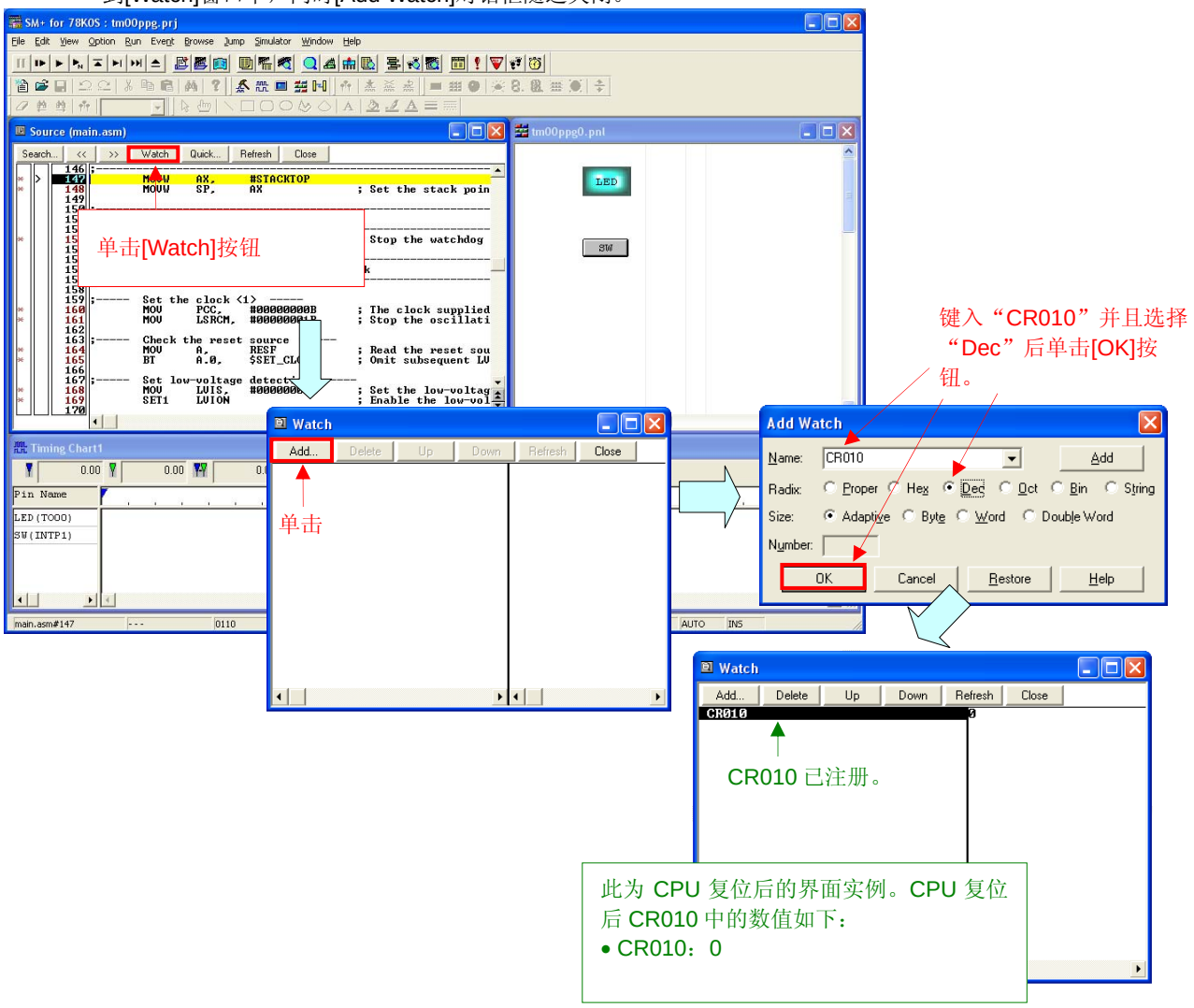
注　　第八次开关输入后，PPG 输出占空比开始从第零次开关输入时进行重复。

[补充 1]　　利用 SM+的监控功能够检验 CR010 寄存器中数值的变化。

<1>　在源窗口中单击[Watch]按钮以打开[Watch]窗口。

<2>　单击[Add]打开[Add Watch]对话框。（此时，[Watch]窗口仍保持打开状态。）

<3>　在[Name]域中键入“CR010”并且在“Radix”中选取“Dec”选项后单击[OK]按钮。“CR010”将注册到[Watch]窗口中，同时[Add Watch]对话框随之关闭。



<4>　执行程序并单击 I/O 面板窗口中的[SW]按钮。根据[SW]按钮输入次数，检验[Watch]窗口中 CR010 的数值变动。

[SW]按钮输入次数[注]	[Watch]窗口中的数值
0	CR010：39
1	CR010：119
2	CR010：199
3	CR010：279
4	CR010：359
5	CR010：279
6	CR010：199
7	CR010：119

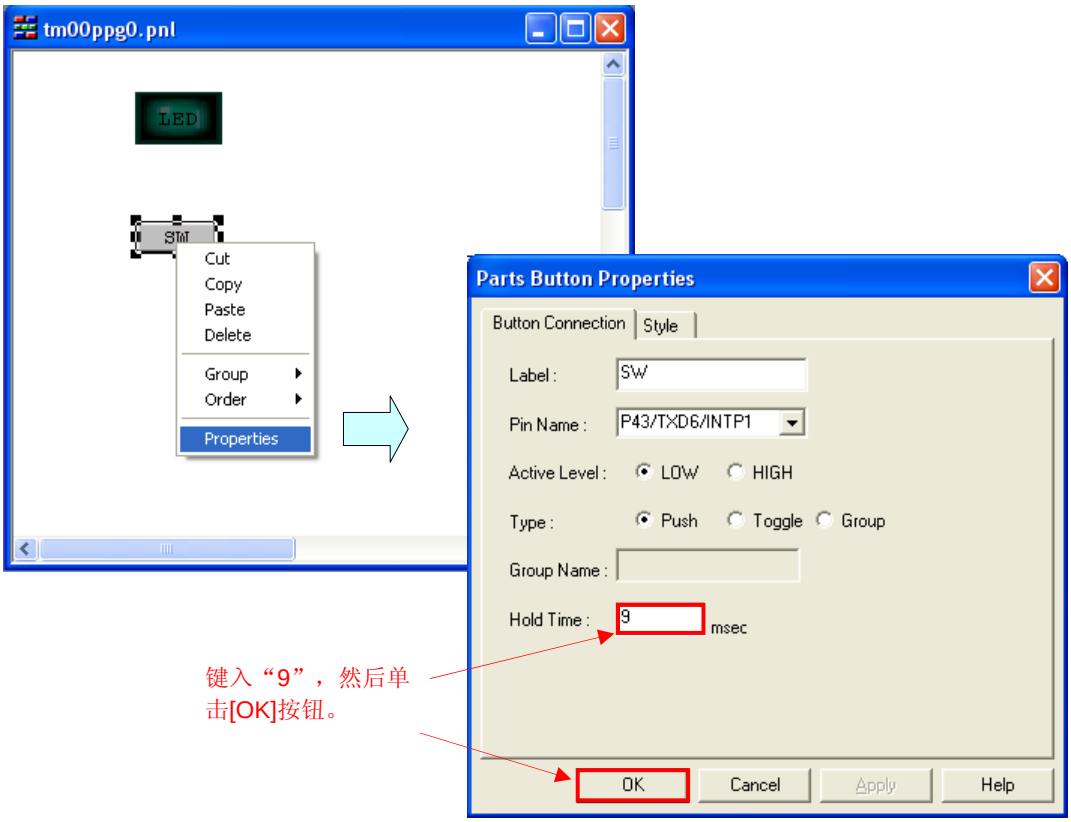
注　　第八次开关输入后，PPG输出占空比开始从第零次开关输入时进行重复。

[补充 2]　　设置[SW]按钮的保持时间少于 10 ms 以检验是否可以检测到抖动。

<1>　在工具栏中选择  。

<2>　在 I/O 面板窗口中右击[SW]按钮，并选择[Properties]选项。

<3>　至“Hold Time”域中键入“9”并单击[OK]按钮。



<4>　在工具栏中选择  。

<5>　执行程序并单击[SW]按钮。因为按钮的保持时间为 9 ms，所以即使单击[SW]按钮，也将视本次操作为抖动且 PPG 输出占空比不会发生变化。

第六章　相关文档

文档名称		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汇编语言程序包用户手册	语言	PDF
	操作	PDF
CC78K0S C编译器用户手册	语言	PDF
	操作	PDF
PM+工程管理器用户手册		PDF
SM+系统仿真器操作用户手册		PDF
78K0S/KA1+简化的Flash写入手册 MINICUBE2 信息		PDF
78K0S/Kx1+ 使用说明	举例程序启动指南	PDF
	举例程序(初始设置) LED灯的开关控制	PDF
	举例程序(中断)由开关输入所产生的外部中断	PDF
	举例程序(低电压检测)电压低于2.7 V时的复位	PDF
	举例程序(16位定时器/事件计数器00)间隔定时器	PDF
	举例程序(16位定时器/事件计数器00)外部事件计数器	PDF
	举例程序(16位定时器/事件计数器00)脉冲宽度测量	PDF
	举例程序(16位定时器/事件计数器00)单次脉冲输出	PDF

附录A 程序清单

78K0S/KB1+微控制器的源程序作为一个程序实例显示如下：

● main.asm（汇编语言版本）

```
;*******************************************************************************
;
;    NEC Electronics    78K0S/KB1+
;
;*******************************************************************************
;    78K0S/KB1+    举例程序
;*******************************************************************************
;    16 位定时器 00（PPG 输出）
;*******************************************************************************
;<<历史记录>>
;    2007.7.--          发布
;*******************************************************************************
;
;<<概要>>
;
;本举例程序提供了一个 16 位定时器 00 的 PPG 输出功能应用实例。
;对于每次开关输入，执行以 200 us 为周期的 PPG 输出并且按照 20%
;的步长改变占空比宽度。
;
;
;  <主要设置内容>
;
;  - 停止看门狗定时器的操作。
;  - 设置低电压检测电压（VLVI）为 4.3 V +-0.2 V。
;  - VDD >= VLVI 后，当 VDD < VLVI 时，
     产生内部复位信号（低电压检测器）。
;  - 将 CPU 的时钟设为 8 MHz。
;  - 将提供至外围硬件的时钟设置为 8 MHz。
;  - 将外部中断 INTP1 的有效沿设置为下降沿。
;  - 设置开关输入期间的抖动检测时间为 10 ms。
;  - 将 HL 寄存器用于中断（类似于全局变量）。
;
;
;  <16 位定时器 00 的设置>
;  - 操作模式：TM00 和 CR000 匹配时清零并启动的模式。
;  - 计数时钟 = fxp/4（2 MHz）。
;  - 将 CR000 和 CR010 用作比较寄存器。
;  - 将 CR000 的周期初始化为 200 us。
;  - 将 CR010 的脉冲宽度初始化为 20 us。
;  - CR000 和 CR010 匹配时，输出反转（PPG 输出）。
;  - 将初始输出值（=有效电平）设置为高电平。
```

```
;   - 设置 P31 的输出锁存为低电平（为了使用 TO00 输出）。
;   - 将 P31 设置为输出模式（为了使用 TO00 输出）。
;
;
;   <开关输入次数和 PPG 输出占空比>
;
;    +-------------------------------------+
;    | SW 输入 | PPG 输出 |    LED    |
;    |         |   占空比   | 亮度 |
;    |-------------------------------------|
;    |  0 次数 |   10%    |   90%    |
;    |  1 次数  |   30%    |   70%    |
;    |  2 次数 |   50%    |   50%    |
;    |  3 次数 |   70%    |   30%    |
;    |  4 次数 |   90%    |   10%    |
;    |  5 次数 |   70%    |   30%    |
;    |  6 次数 |   50%    |   50%    |
;    |  7 次数 |   30%    |   70%    |
;    +-------------------------------------+
;    # 第八次开关输入后，PPG 输出占空比开始从第零次开关输入时的状态进行重复。
;
;
; <<I/O 端口设置>>
;
;   输入：P43。
;   输出：P00-P03、P20-P23、P30-P33、P40-P42、P44-P47、P120-P123、P130。
;   # 所有未用端口设置为输出模式。
;
; ************************************************************************


; ========================================================================
;
;     定义标号
;
; ========================================================================
OFFSETT     EQU     43/4                    ；校准 INTTM010 中断服务时间。

; ========================================================================
;
;     向量表
;
; ========================================================================
XVCT        CSEG   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      INTERRUPT_P1        ；（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 ROM 数据表
；
；=============================================================================
XROM        CSEG  AT      0100H
PWIDTH：    DW    40-1    ；20 us 的脉冲宽度（占空比为 10%）。
        DW      40*3-1          ；60 us 的脉冲宽度（占空比为 30%）。
        DW      40*5-1          ；100 us 的脉冲宽度（占空比为 50%）。
        DW      40*7-1          ；140 us 的脉冲宽度（占空比为 70%）。
        DW      40*9-1          ；180 us 的脉冲宽度（占空比为 90%）。
        DW      40*7-1          ；140 us 的脉冲宽度（占空比为 70%）。
        DW      40*5-1          ；100 us 的脉冲宽度（占空比为 50%）。
        DW      40*3-1          ；60 us 的脉冲宽度（占空比为 30%）。

；=============================================================================
；
；    定义 RAM
；
；=============================================================================
XRAM        DSEG  SADDRP
NEXTD：             DS      2       ；用于存储新的脉冲宽度（占空比）。
CURRENTD：          DS      2       ；用于存储当前脉冲宽度（占空比）。

；=============================================================================
；
；    定义内存堆栈区
；
；=============================================================================
XSTK        DSEG  AT      0FEE0H
STACKEND：
        DS      20H                 ；内存堆栈区 = 32 个字节。
STACKTOP：                          ；内存堆栈区的起始地址 = FF00H。
```

```
;*******************************************************************************
;
;    RESET 后的初始化
;
;*******************************************************************************
XMAIN       CSEG  UNIT
RESET_START：
;-------------------------------------------------------------------------------
;    初始化堆栈指针
;-------------------------------------------------------------------------------
    MOVW AX，    #STACKTOP
    MOVW SP，    AX              ；设置堆栈指针。

;-------------------------------------------------------------------------------
;    初始化看门狗定时器
;-------------------------------------------------------------------------------
    MOV   WDTM，       #01110111B    ；停止看门狗定时器的操作。

;-------------------------------------------------------------------------------
;    检测低电压 + 设置时钟
;-------------------------------------------------------------------------------

;----- 设置时钟 <1> -----
    MOV   PCC，  #00000000B    ；提供至 CPU 的时钟（fcpu） = fxp（= fx/4 = 2 MHz）。
    MOV   LSRCM，      #00000001B    ；停止内部低速振荡器的振荡。

;----- 检查复位信号源 -----
    MOV   A，    RESF           ；读取复位信号源。
    BT    A.0，  $SET_CLOCK     ；复位期间，省略后续 LVI 相关处理，并转到 SET_CLOCK。

;----- 设置低电压检测 -----
    MOV   LVIS，  #00000000B    ；将低电压检测电平（VLVI）设置为 4.3 V +-0.2 V。
    SET1  LVION                 ；低电压检测器操作允许。

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

;----- VDD >= VLVI 时的等待处理 -----
WAIT_LVI：
    NOP
    BT    LVIF，  $WAIT_LVI     ；如果 VDD < VLVI，则进行转移。

    SET1  LVIMD                 ；如此设置以便当 VDD < VLVI 时，产生内部复位信号。

;----- 设置时钟<2> -----
```

```
SET_CLOCK：
    MOV   PPCC，        #00000000B    ；提供至外围硬件的时钟（fxp） = fx（= 8 MHz）
                                      ；  -> 提供至 CPU 的时钟（fcpu） = fxp = 8 MHz。

；-------------------------------------------------------------------------
；   初始化端口 0
；-------------------------------------------------------------------------
    MOV   P0，    #00000000B    ；设置 P00-P03 的输出锁存为低电平。
    MOV   PM0，   #11110000B    ；设置 P00-P03 为输出模式。

；-------------------------------------------------------------------------
；   初始化端口 2
；-------------------------------------------------------------------------
    MOV   P2，    #00000000B    ；设置 P20-P23 的输出锁存为低电平。
    MOV   PM2，   #11110000B    ；设置 P20-P23 为输出模式。

；-------------------------------------------------------------------------
；   初始化端口 3
；-------------------------------------------------------------------------
    MOV   P3，    #00000000B    ；设置 P30-P33 的输出锁存为低电平。
    MOV   PM3，   #11110000B    ；设置 P30-P33 为输出模式。

；-------------------------------------------------------------------------
；   初始化端口 4
；-------------------------------------------------------------------------
    MOV   P4，    #00000000B    ；设置 P40-P47 的输出锁存为低电平。
    MOV   PU4，   #00001000B    ；将片内上拉电阻连接至 P43。
    MOV   PM4，   #00001000B    ；将 P40-P42 和 P44-P47 设置为输出模式，P43 为输入模式。

；-------------------------------------------------------------------------
；   初始化端口 12
；-------------------------------------------------------------------------
    MOV   P12，   #00000000B    ；设置的输出锁存为低电平。
    MOV   PM12，        #11110000B    ；设置 P120-P123 为输出模式。

；-------------------------------------------------------------------------
；   初始化端口 13
；-------------------------------------------------------------------------
    MOV   P13，   #00000001B    设置 P130 的输出锁存为高电平。

；-------------------------------------------------------------------------
；   设置 16 位定时器 00
；-------------------------------------------------------------------------
    MOV   CRC00，       #00000000B    ；将 CR000 和 CR010 用作比较寄存器。
    MOVW AX，    #400-1
    MOVW CR000，        AX            ；用 CR000 来设置周期。
    MOVW AX，    #40-1
    MOVW CR010，        AX            ；用 CR010 来设置脉冲宽度。
```

```
        MOV    PRM00，        #00000001B    ；计数时钟 = fxp/4（= 2 MHz）
        MOV    TOC00，        #00011011B    ；CR000 和 CR010 匹配时输出反转，将初始输出值设置为高电平，允许定时器输出。
        MOV    TMC00，        #00001100B    ；启动定时器的操作。（TM00 和 CR000 匹配时清除并启动）

；-------------------------------------------------------------------------
；    初始化 RAM 和通用寄存器
；-------------------------------------------------------------------------
        MOVW CURRENTD， AX                ；保存初始脉冲宽度值作为当前脉冲宽度。
        MOVW HL，    #PWIDTH              ；指定脉冲宽度的表地址。

；-------------------------------------------------------------------------
；    设置中断
；-------------------------------------------------------------------------
        MOV    INTM0，        #00000000B    ；设置 INTP1 下降沿有效。
        MOV    IF0，    #00H              ；预先清除无效中断请求。
        CLR1   PMK1                       ；不屏蔽 INTP1 中断。

        EI                                ；允许向量表中断。

；*************************************************************************
；
；    主循环
；
；*************************************************************************
MAIN_LOOP：
        NOP
        BR     $MAIN_LOOP                 ；转到 MAIN_LOOP。

；*************************************************************************
；
；    外部中断 INTP1
；
；*************************************************************************
INTERRUPT_P1：
        PUSH   AX                         ；将 AX 寄存器的值压入堆栈。

；----- 等待 10 ms 以处理抖动 -----
        MOV    A，        #50+1           ；200 us x 50 = 10 ms。
WAIT_CHAT：
        CLR1   TMIF000                    ；清除 INTTM000 中断请求标志。
WAIT_INT：
        BF     TMIF000，  $WAIT_INT；等待 INTTM000 中断。
        DEC    A                          ；A 寄存器的值减 1。
        BNZ    $WAIT_CHAT                 ；如果 A 不为 0，则进行转移。

        CLR1   PIF1                       ；清除 INTP1 中断请求。
```

```
; ----- 抖动检测的识别 -----
    BT      P4.3，  $END_INTP1   ；如果没有开关输入，则进行转移。

; ----- 读取新的脉冲宽度 -----
    MOV     A，     L            ；读取表地址。
    ADD     A，     #2           ；表地址加 2。
    AND     A，     #00001111B   ；屏蔽除位 0 至位 3 之外的其它位。
    MOV     L，     A
    MOV     A，     [HL]         ；读取脉冲宽度的低 8 位。
    XCH     A，     X
    MOV     A，     [HL+1]       ；读取脉冲宽度的高 8 位。
    MOVW NEXTD，        AX       ；保存新的脉冲宽度。

; ----- 与当前脉冲宽度作比较 -----
    MOVW AX，   CURRENTD     ；读取当前脉冲宽度。
    ADDW AX，   #OFFSETT     ；加上偏移时间。
    XCH     A，     X
    SUB     A，     [HL]         ；与新脉冲宽度作比较。
    XCH     A，     X
    SUBC    A，     [HL+1]

; ----- 声明改变脉冲宽度的中断服务 -----
    BNC     $DECDUTY             ；如果新脉宽<当前脉宽，则进行转移。
    CLR1    TMIF000              ；清除无效中断请求。
    CLR1    TMMK000              ；不屏蔽 INTTM000 中断。
    SET1    TMMK010              ；屏蔽 INTTM010 中断。
    BR      $END_INTP1           ；转到 END_INTP1。
DECDUTY：
    CLR1    TMIF010              ；清除无效中断请求。
    CLR1    TMMK010              ；不屏蔽 INTTM010 中断。
    SET1    TMMK000              ；屏蔽 INTTM000 中断。

END_INTP1：
    POP     AX              ；恢复 AX 寄存器的值。
    RETI                    ；从中断服务程序中返回。

; ************************************************************************
;
;   INTTM000 中断
;
; ************************************************************************
INTERRUPT_TM000：
    PUSH    AX              ；将 AX 寄存器的值压入堆栈。

    CLR1    TOC00.4         ；禁止输出反转。
    MOVW AX，   NEXTD
    MOVW CR010，    AX      ；改变脉冲宽度（占空比）
```

```
        NOP                     ；等待定时器 00 一个时钟周期的时间。
        NOP
        SET1    TOC00.4         ；输出反转允许。

        MOVW CURRENTD，AX；将新的脉冲宽度保存为当前脉宽。
        SET1    TMMK000         ；屏蔽 INTTM000 中断。

        POP     AX              ；恢复 AX 寄存器的值。
        RETI                    ；从中断服务程序中返回。

；*****************************************************************************
；
；    INTTM010 中断
；
；*****************************************************************************
INTERRUPT_TM010：
        PUSH    AX              ；将 AX 寄存器的值压入堆栈。

        CLR1    TOC00.4         ；输出反转禁止。
        MOVW AX，    NEXTD
        MOVW CR010，AX     ；改变脉冲宽度（占空比）。
        NOP                     ；等待定时器 00 一个时钟周期的时间。
        NOP
        SET1    TOC00.4         ；输出反转允许。

        MOVW CURRENTD，AX；将新的脉冲宽度保存作为当前脉宽。
        SET1    TMMK010         ；屏蔽 INTTM010 中断。

        POP     AX              ；恢复 AX 寄存器的值。
        RETI                    ；从中断服务程序中返回。

end
```

● main.c（C 语言版本）

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

      NEC Electronics     78K0S/KB1+

*********************************************************************************
      78K0S/KB1+     举例程序
*********************************************************************************
      16 位定时器 00（PPG 输出）
*********************************************************************************
<<历史记录>>
      2007.7.--          发布
*********************************************************************************

<<概要>>

本举例程序提供了一个 16 位定时器 00 的 PPG 输出功能应用实例。
对于每次开关输入，以 200 us 的周期执行 PPG 输出并且
按照 20%的步长值改变占空比宽度。


 <主要设置内容>

 -声明一个由中断运行的函数：INTP1 -> fn_intp1（）。
 -声明一个由中断运行的函数：INTTM000 -> fn_inttm000（）。
 -声明一个由中断运行的函数：INTTM010 -> fn_inttm010（）。
 - 停止看门狗定时器的操作。
 - 设置低电压检测电压（VLVI）为 4.3 V +-0.2 V。
 - VDD >= VLVI 后，当 VDD < VLVI 时，
    产生内部复位信号（低电压检测器）。
 - 设置 CPU 时钟为 8 MHz。
 - 将提供至外围硬件的时钟设置为 8 MHz。
 - 设置外部中断 INTP1 的有效沿为下降沿。
 - 设置开关输入期间的抖动检测时间为 10 ms。

 <16 位定时器 00 的设置>

 - 操作模式：TM00 和 CR000 匹配时清除并启动定时器计数的模式。
 - 计数时钟 = fxp/4（2 MHz）。
 - 将 CR000 和 CR010 用作比较寄存器。
 - 将 CR000 的周期初始化为 200 us。
 - 将 CR010 的脉冲宽度初始化为 20 us。
 - CR000 和 CR010 匹配时，输出反转（PPG 输出）。
 - 设置初始输出值（= 有效电平）为高电平。
 - 设置 P31 的输出锁存为低电平（为使用 TO00 输出）
 - 将 P31 设置为输出模式（为使用 TO00 输出）。
```

<开关输入次数和 PPG 输出占空比>

SW 输入	PPG 输出	LED 亮度
	占空比	
0 次数	10%	90%
1 次数	30%	70%
2 次数	50%	50%
3 次数	70%	30%
4 次数	90%	10%
5 次数	70%	30%
6 次数	50%	50%
7 次数	30%	70%

#第八次开关输入后，PPG 输出占空比开始从第零次开关输入时的状态进行重复。

<<I/O 端口设置>>

输入： P43
输出： P00-P03， P20-P23， P30-P33， P40-P42， P44-P47， P120-P123， P130
所有未用端口设置为输出模式。

**********/

```
/*============================================================================

    预处理指示符（#pragma）

============================================================================*/
#pragma     SFR                                 /* SFR 名称能够在 C 语言层面上进行描述 */
#pragma     EI                                  /* EI 指令能够在 C 语言层面上进行描述 */
#pragma     DI                                  /* DI 指令能够在 C 语言层面上进行描述 */
#pragma     NOP                                 /* NOP 指令能够在 C 语言层面上进行描述 */
#pragma interrupt INTP1 fn_intp1      /* 中断函数声明：INTP1 */
#pragma interrupt INTTM000 fn_inttm000      /*中断函数声明：INTTM000 */
#pragma interrupt INTTM010 fn_inttm010      /*中断函数声明：INTTM010 */

#define offsetT 56/4                        /* 校准 INTTM010 中断服务时间 */



/*============================================================================
```

```
        定义全局变量和常量表

    ================================================================================*/
    sreg unsigned char g_ucSWcnt = 0；  /* 计算开关输入次数的 8 位变量 */
    sreg unsigned int g_unNextD = 40-1；          /* 保存新脉宽的 16 位变量（占空比） */
    sreg unsigned int g_unCurrentD = 40-1；       /* 保存当前脉宽的 16 位变量（占空比） */
    const unsigned int g_unOutData[8] = {40-1，40*3-1，40*5-1，40*7-1，40*9-1，40*7-1，40*5-1，
40*3-1}；  /* 输出占空比表 */

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

        RESET 后的初始化

    *****************************************************************************/
    void hdwinit（void）{
        unsigned char ucCnt200us；      /* 200 us 等待时间的 8 位变量 */

    /*-------------------------------------------------------------------------
        初始化看门狗定时器 + 检测低电压 + 设置时钟
    -------------------------------------------------------------------------*/
        /* 初始化看门狗定时器 */
        WDTM  = 0b01110111；          /* 停止看门狗定时器的操作 */

        /* 设置时钟<1> */
        PCC   = 0b00000000；          /* 提供至 CPU 的时钟（fcpu） = fxp（= fx/4 = 2 MHz） */
        LSRCM = 0b00000001；                  /* 停止内部低速振荡器的振荡 */

        /* 检查复位信号源 */
        if（!（RESF & 0b00000001））{         /* LVI 复位期间，省略后续 LVI 相关处理 */

            /* 设置低电压检测 */
            LVIS  = 0b00000000；  /* 设置低电压检测电平（VLVI）为 4.3 V +-0.2 V */
            LVION = 1；           /* 低电压检测器操作允许 */

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

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

            LVIMD = 1；                   /* 如此设置以便当 VDD < VLVI 时，产生内部复位信号 */
        }


        /* 设置时钟<2> */
```

```
    PPCC  = 0b00000000;         /* 提供至外围硬件的时钟（fxp） = fx（= 8 MHz）
                                   -> 提供至 CPU 的时钟（fcpu） = fxp = 8 MHz */

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

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

/*---------------------------------------------------------------------------
    Initialize the port 3
---------------------------------------------------------------------------*/
    P3    = 0b00000000;         /* 设置 P30-P33 的输出锁存为低电平 */
    PM3   = 0b11110000;         /* 将 P30-P33 设置为输出模式 */

/*---------------------------------------------------------------------------
    初始化端口 4
---------------------------------------------------------------------------*/
    P4    = 0b00000000;         /* 设置 P40-P47 的输出锁存为低电平 */
    PU4   = 0b00001000;         /* 将片内上拉电阻连接至 P43 */
    PM4   = 0b00001000;         /* 将 P40-P42 和 P44-P47 设置为输出模式，P43 为输入模式 */

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

/*---------------------------------------------------------------------------
    初始化端口 13
---------------------------------------------------------------------------*/
    P13   = 0b00000001;         /* 设置 P130 的输出锁存为高电平 */




/*---------------------------------------------------------------------------
```

```
        设置 16 位定时器 00
    ------------------------------------------------------------------------*/
        CRC00 = 0b00000000;                 /* 将 CR000 和 CR010 用作比较寄存器 */
        CR000 = 400-1;                      /* 用 CR000 来设置周期 */
        CR010 = 40-1;                       /* CR010 用来设置脉冲宽度 */
        PRM00 = 0b00000001;                 /* 计数时钟 = fxp/4（= 2 MHz） */
        TOC00 = 0b00011011;             /* CR000 和 CR010 匹配时输出反转，设置初始化输出值为高电
平，允许定时器输出 */
        TMC00 = 0b00001100;                 /* 启动定时器的操作（TM00 和 CR000 匹配时清除并启
动） */

    /*------------------------------------------------------------------------
        设置中断
    ------------------------------------------------------------------------*/
        INTM0 = 0b00000000;             /* 设置 INTP1 下降沿有效 */
        IF0 = 0x00;                         /* 清除无效中断请求 */
        PMK1 = 0;                           /* 不屏蔽 INTP1 中断 */

        return;
    }

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

        Main loop

    ************************************************************************/
    void main（void）{

        EI（）;                                     /* 向量表中断允许 */

        while（1）{
                NOP（）;
        }
    }
```

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

    INTP1 中断

*****************************************************************************/
__interrupt void fn_intp1（）{
    unsigned char ucChat；          /* 消除抖动的 8 位变量 */

    for（ucChat = 0； ucChat <50+1； ucChat++）{/*循环等待 10 ms */
        TMIF000 = 0；           /* 清除 INTTM000 中断标志 */
        while（!TMIF000）{          /* 等待 INTTM000 中断 */
            NOP（）；
        }
    }

    PIF1 = 0；                          /* 清除 INTP1 中断请求 */

    if（!P4.3）{            /* 如果 SW 闭合达 10 ms 或更久，则执行处理 */
        g_ucSWcnt += 1；            /* 开关输入次数加 1 */
        g_ucSWcnt &= 0b00000111；   /* 屏蔽除位 0 至位 2 之外的其它位 */

        g_unNextD = g_unOutData[g_ucSWcnt]；        /* 保存新脉宽 */

        if（g_unNextD > g_unCurrentD + offsetT）{   /* 利用 INTTM000 中断增加脉宽 */
            TMIF000 = 0；           /* 清除无效中断请求 */
            TMMK000 = 0；               /* 不屏蔽 INTTM000 中断 */
            TMMK010 = 1；               /* 屏蔽 INTTM010 中断 */
        }
        else {                  /* 利用 INTTM010 中断减少脉宽值 */
            TMIF010 = 0；           /* 清除无效中断请求 */
            TMMK010 = 0；               /* 不屏蔽 INTTM010 中断 */
            TMMK000 = 1；               /* 屏蔽 INTTM000 中断 */
        }
    }

    return；
}

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

    INTTM000 中断

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

    TOC00.4 = 0；               /* 输出反转允许 */
    CR010 = g_unNextD；         /* 改变脉冲宽度（占空比） */
    NOP（）；                       /* 等待定时器 00 一个时钟周期的时间 */
```

```
    NOP（）；
    TOC00.4 = 1；                    /* 输出反转允许 */

    g_unCurrentD = g_unNextD；    /* 将新脉宽保存为当前脉宽 */
    TMMK000 = 1；                        /* 屏蔽 INTTM000 中断 */

    return；
}



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

    INTTM010 中断

***************************************************************************/
__interrupt void fn_inttm010（）{

    TOC00.4 = 0；                    /* 输出反转禁止 */
    CR010 = g_unNextD；              /* 改变脉冲宽度（占空比） */
    NOP（）；                            /* 等待定时器 00 一个时钟周期的时间 */
    NOP（）；
    TOC00.4 = 1；                    /* 输出反转允许 */

    g_unCurrentD = g_unNextD；    /* 将新脉宽保存为当前脉宽 */
    TMMK010 = 1；                        /* 屏蔽 INTTM010 中断 */

    return；
}
```

● op.asm（汇编语言版和 C 语言版通用）

```
; ===========================================================================
;
;    选项字节
;
; ===========================================================================
OPBTCSEG AT     0080H
    DB          10011100B           ；选项字节区域。
;               ||||
;           |||+----------  内部低速振荡器可由软件停止。
;           |++----------  内部高速振荡时钟（8 MHz）选择为系统时钟信号源。
;           +----------  P34/RESET 引脚用作 RESET 引脚。

    DB      11111111B   ;           保护字节区域（用于自编程模式）。
;           ||||||||
;           ++++++++----------       所有模块都可以写入或擦除。

end
```

附录B 版本修订历史

版本	发布日期	页码	修订
第一版	2008年02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