使用说明

78K0S/Kx1+

举例程序(16 位定时器/事件计数器 00)

间隔定时器

本文档描述了举例程序的操作概述及使用方法，以及如何设置和应用 16 位定时器/事件计数器 00 的间隔定时器功能。在该举例程序中，通过使用 16 位定时器/事件计数器 00 的间隔定时器功能使 LED 灯以固定周期闪烁。此外，LED 灯的闪烁周期依照开关输入次数而发生变化。

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 U18887CA1V0AN00 (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 的间隔定时器功能实例。LED 以固定周期闪烁且其闪烁周期依照开关输入次数而发生变化。

1.1 初始设置的主要内容

初始设置的主要内容如下所示：

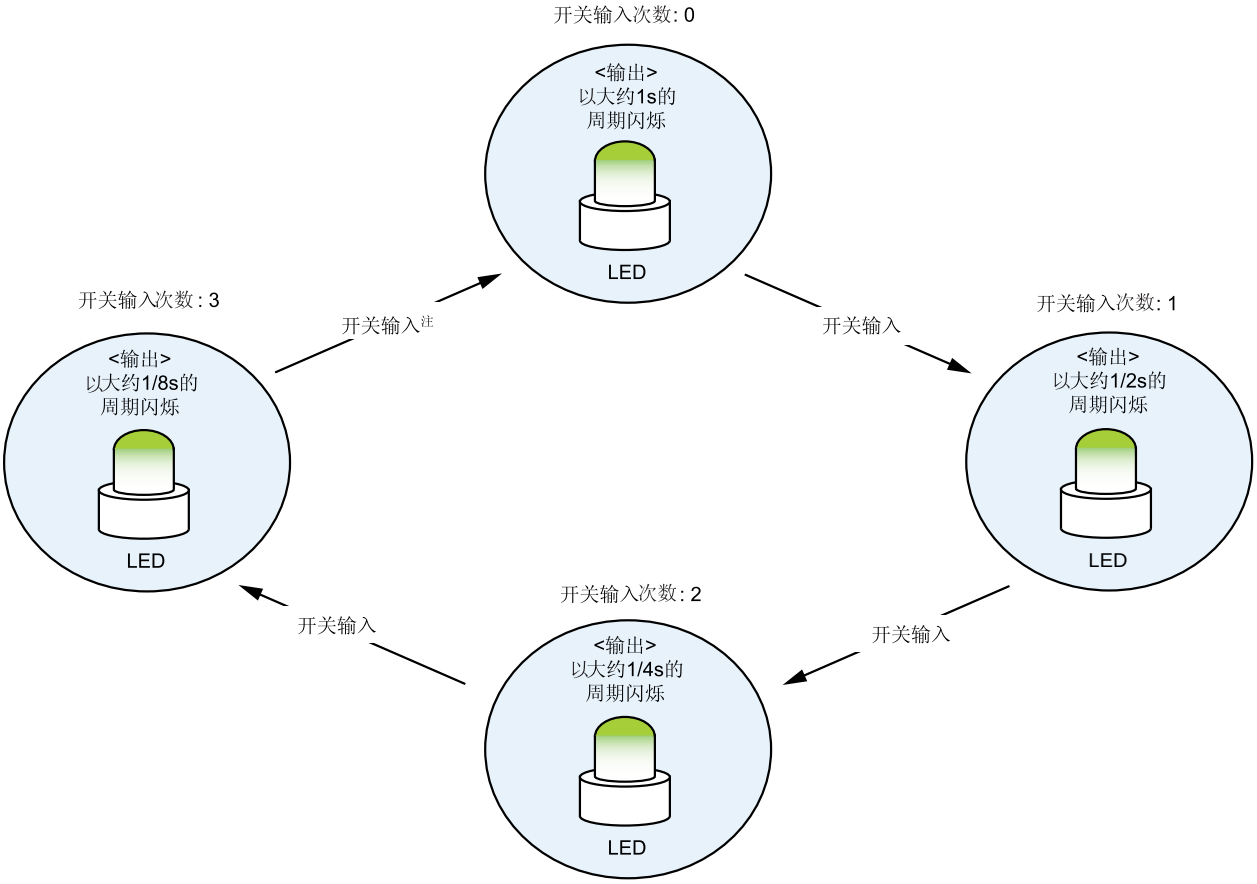
- 选择内部高速振荡器作为系统时钟信号源[注]。
- 停止看门狗定时器运行。
- 将 V_{LVI} (低压检测电压) 设置在 4.3 V ±0.2 V 范围。
- V_{DD}（电源电压）变得高于或等于 V_{LVI} 后，当检测到 V_{DD} 小于 V_{LVI} 时产生内部复位（LVI 复位）信号 。
- 设置 CPU 时钟频率为 8 MHz。
- 设置 I/O 端口。
- 设置 16 位定时器/事件计数器 00。
  - 设置 CR000 为比较寄存器。
  - 设置间隔周期大约为 2 ms (32 μs × 63)。
  - 设置计数时钟为 $f_{XP}/2^8$ (31.25 kHz)。
  - 禁止定时器输出(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 和 INTTM000 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 产生中断(INTTM000)，使 LED 以固定周期闪烁。

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 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 LED 的闪烁周期依照开关输入次数而变化。



注　　第四次开关输入后，闪烁周期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

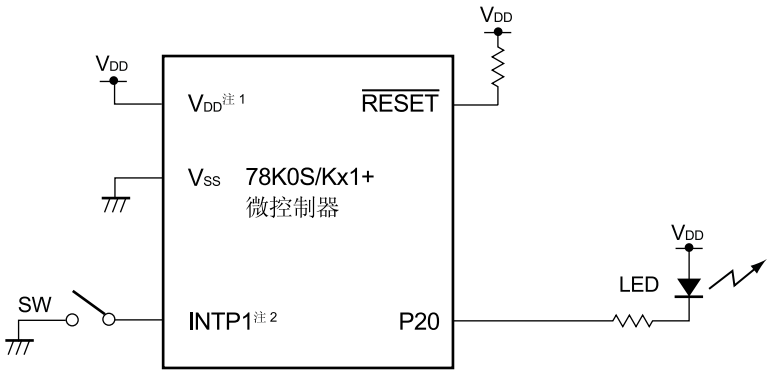
抖动是一种开关按下之后由于机械触点的弹跳而引起电信号瞬时反复接通和断开的现象。

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/P43： 78K0S/KA1+ 和 78K0S/KB1+ 微控制器。
INTP1/P32： 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 灯的输入。

(2)　LED

LED 用作 16 位定时器/事件计数器 00 的间隔定时器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。	●	●	
tm00.prw	集成开发环境PM+的工作空间文件。		●	
tm00.prj	集成开发环境PM+的工程文件。		●	
tm00.pri tm00.prs tm00.prm	78K0S/Kx1+系统仿真器SM+的工程文件。		●注 2	
tm000.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	●
tm000.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：

- 内部定时器功能： 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] (外部中断)
- LED 输出： P20 (输出端口)

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

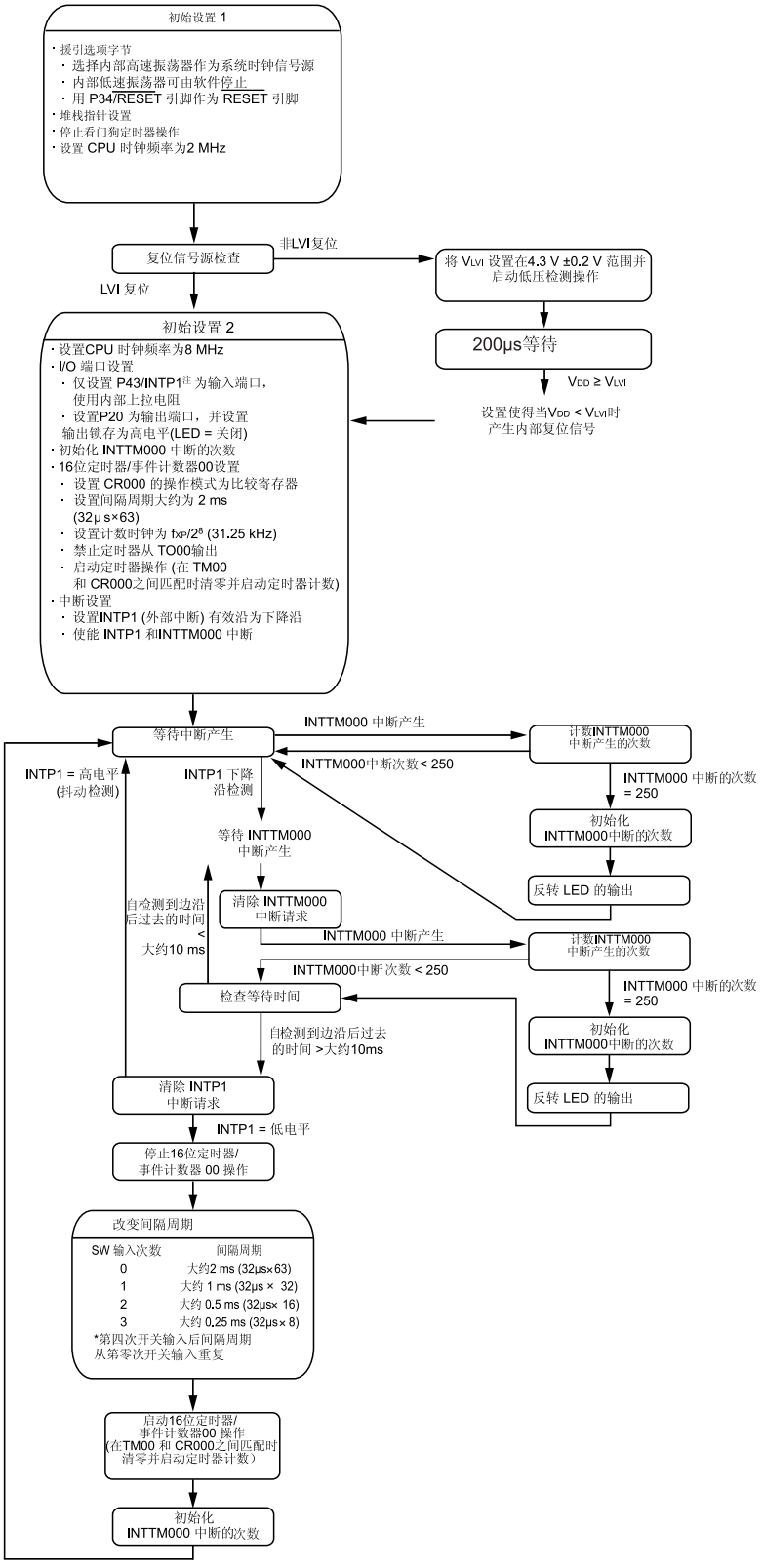
3.3 初始设置和工作概要

在该举例程序中，内部设置包括设置低压检测功能、时钟频率选择、设置 I/O 端口、设置 16 位定时器/事件计数器 00（间隔定时器功能）和中断设置。

初始设置完成后，使用 16 位定时器/事件计数器 00 产生中断(INTTM000)，使 LED 以固定周期闪烁。

当检测到由开关输入产生的 INTP1 引脚信号下降沿时，进行 INTP1 中断服务。自检测到 INTP1 引脚下降沿起 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 LED 闪烁周期依照开关输入次数而发生变化。

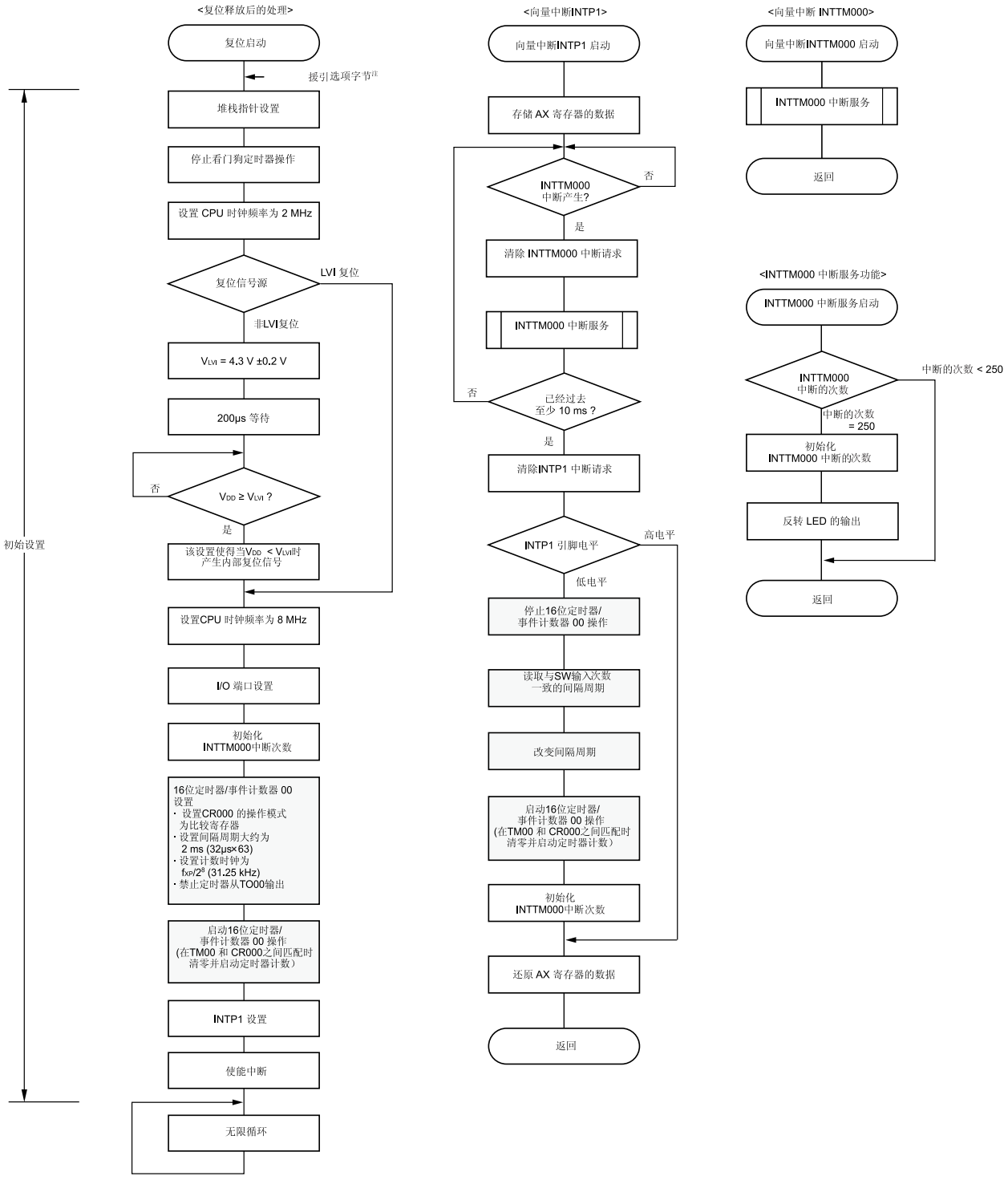
详情如下列状态转换图所示：



注 INTP1/P43： 78K0S/KA1+ 和 78K0S/KB1+ 微控制器。
INTP1/P32： 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 的间隔定时器功能。

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 的间隔定时器功能

当用 16 位定时器/事件计数器 00 作为间隔定时器时，设置以下八类寄存器：

当用 TOC00 寄存器使能定时器输出（TO00 引脚输出）时，通过用间隔定时器功能可输出方波。

- 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)。
- 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)。
- 端口模式寄存器 x (PMx)[注]。
- 端口寄存器 x (Px)[注]。
- 端口模式控制寄存器 x (PMCx)[注]。

注　为了将 TO00 引脚用作定时器输出(用 TO00 引脚输出方波)，设置如下。

当不将 TO00 引脚用作定时器输出时，不需要该设置 。

	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 用作间隔定时器时，基本操作设置步骤举例>

<1> 设置 CRC00 寄存器。

<2> 给 CR000 寄存器设置一个任意值。

<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。

注意事项 当 16 位定时器/事件计数器 00 用作间隔定时器时不使用 CR010 寄存器。

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	作为比较寄存器操作。
1	作为捕获寄存器操作。

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	作为比较寄存器操作。
1	作为捕获寄存器操作。

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功能和比较寄存器功能。

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)。当 TM00 设置为间隔定时器操作方式时，CR000 也可用作保存间隔时间的寄存器。

- 间隔时间= (N + 1)/fsam

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。

2. 如果 CR000 寄存器的新值小于 TM00 计数器值， TM00 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. 若将 CR000 寄存器设置为比较模式，即使输入捕获触发，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。

备注 N： CR000 寄存器的设定值(0001H 至 FFFFH)。

Fsam：TM00 计数器计数时钟频率。

(3) 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 4-3. 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	上升和下降沿

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	上升和下降沿

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}： 供给外围硬件时钟的振荡频率。

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，应注意对 TI0n0 引脚(n = 0， 1)有效沿的检测情况。

<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 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. 使用 TI000 的有效沿为计数时钟时，用 f_{XP} 对其采样以抑制噪声。当采样有效沿且有效电平被检测到两次时才执行捕获操作，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) 。

(4) 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，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。

当不将 TO00 引脚用作定时器输出时，不需要该设置。

图 4-4. 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 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 单次脉冲输出模式操作通常仅用于自由运行模式和由 TI000 引脚有效沿进入清零并启动的模式中。在基于 TM00 和 CR000 匹配而清零并启动的模式中，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 自动被清除，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. 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。

6. 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 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 计数器清除模式、输出时序以及检测溢出 。

图 4-5. 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/事件计数器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匹配时产生 <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，TM00 和CR010匹配或TI000引脚的有效沿	
1	0	0	在TI000引脚有效沿发生清除并启动	–	
1	0	1			
1	1	0	TM00 和CR000匹配发生清除并启动	TM00 和CR000匹配或TM00 和CR010匹配	
1	1	1		TM00 和CR000匹配，TM00 和CR010匹配或TI000引脚的有效沿	

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，为停止操作，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。

[例1]　　设置时钟间隔周期为2.016 ms 且使用16位定时器/事件计数器00作为间隔定时器。

(计数时钟= $f_{XP}/2^8$ (f_{XP} = 8 MHz)，无定时器输出 (TO00引脚输出))

(与该举例源程序内容相同)

(1) 寄存器设置

<1> CRC00

0	0	0	0	0	x	x	0

CR000操作模式的选择

0	作为比较寄存器操作。

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

x	(不使用CR010。)

<2> CR000

设定值(N)： 62

- 计数时钟 fsam = 8 MHz/2^8 = 0.03125 MHz = 31.25 kHz
- 间隔周期 2 ms = (N + 1)/31.25 kHz

→ N = 2 ms × 31.25 kHz − 1 = 61.5 → 62

<3> PRM00

x	x	x	x	0	0	1	0

计数时钟(fsam) 的选择

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

TI000 引脚有效沿的选择

x	x	(因为不使用TI000引脚所以设置无效。但禁止设置为“1，0”。)

TI010 引脚有效沿的选择

x	x	(因为不使用TI010 引脚所以设置无效。但禁止设置为“1，0”。)

<4>　TOC00

0	0	0	0	0	0	0	0

定时器输出控制

0	禁止输出。

CR000 和TM00匹配时定时器F/F控制

0	禁止反相操作。

定时器输出F/F 状态设置

0	0	没有改变。

CR010和 TM00匹配时定时器F/F 控制

0	禁止反相操作。

单次脉冲输出操作控制

0	连续脉冲输出模式。

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

<5>　TMC00

0	0	0	0	1	1	x	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匹配时产生 <当用作捕获寄存器时> 在TI000引脚或TI010引脚的有效沿产生
1	1	1		TM00和 CR000匹配，TM00 和CR010匹配或TI000 引脚的有效沿	

(2) 举例程序

下例中，在**(1) 寄存器设置**中的“x”设置为“0”。

<1> 汇编语言

```
MOV    CRC00, #00000000B
MOVW   CR000, #62
MOV    PRM00, #00000010B
MOV    TOC00, #00000000B
MOV    TMC00, #00001100B
```

<2> C 语言

```
CRC00 = 0b00000000;
CR000 = 62;
PRM00 = 0b00000010;
TOC00 = 0b00000000;
TMC00 = 0b00001100;
```

[例2]　　设置间隔周期为50 μs且以100 μs (10 kHz)周期从TO00引脚输出方波 。
(计数时钟= $f_{XP}/2^2$ (f_{XP} = 8 MHz)， TO00 输出的初始值为低电平)

(1) 寄存器设置

<1> CRC00

0	0	0	0	0	x	x	0

CR000 操作模式的选择

0	作为比较寄存器操作。

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

x	(不使用CR010。)

<2> CR000

设定值(N)： 99

- 计数时钟 fsam = 2 MHz
- 间隔周期 50 μs = (N + 1)/2 MHz

→ N = 50 μs × 2 MHz − 1 = 99

<3> PRM00

x	x	x	x	0	0	0	1

计数时钟(fsam) 的选择

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

TI000 引脚有效沿的选择

x	x	(因为不使用TI000引脚，所以设置无效。但禁止设置为“1，0”。)

TI010 引脚有效沿的选择

x	x	(因为不使用TI010引脚，所以设置无效。但禁止设置为“1，0”。)

<4> TOC00

0	0	0	0	0	1	1	1

定时器输出控制

1	使能输出。

CR000 和TM00匹配时定时器F/F 控制

1	允许反相操作。

定时器输出F/F 状态设置

0	1	复位(0)定时器输出 F/F。

CR010 和TM00匹配时定时器F/F 控制

0	禁止反相操作。

单次脉冲输出操作控制

0	连续脉冲输出模式。

用软件控制单次脉冲输出触发器

0	不使用单次脉冲输出触发器。

<5>　TMC00

0	0	0	0	1	1	x	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模式	TM00 和CR000匹配或TM00 和CR010匹配	<当用作比较寄存器时> 在TM00 和CR000匹配或TM00 和CR010匹配时产生 <当用作捕获寄存器时> 在TI000引脚或TI010 引脚的有效沿产生
1	1	1		TM00 和CR000匹配，TM00 和CR010匹配或TI000引脚的有效沿	

<6>　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	不需要设置
78K0S/KY1+ 和 78K0S/KU1+ 微控制器	P21 = 0	PM21 = 0	PMC21 = 0

(2) 举例程序

下例中，在**(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, #99
MOV    PRM00, #00000001B
MOV    TOC00, #00000111B
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 = 99;
PRM00 = 0b00000001;
TOC00 = 0b00000111;
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 [例 1] 提到的内容相同)。

(1) 汇编语言

```
XMAIN   CSEG    UNIT
RESET_START：
                •
                •
                •
        MOV     CRC00,  #00000000B      ；用CR000 作为比较寄存器
        MOVW    AX,     #63-1           ；给 AX 寄存器赋值为 LED 闪烁时间基准初始值
        MOVW    CR000,  AX              ；初始化LED闪烁时间基准
        MOV     PRM00,  #00000010B      ；计数时钟= fxp/2^8 = 31.25 kHz
        MOV     TOC00,  #00000000B      ；不执行定时器输出
        MOV     TMC00,  #00001100B      ；启动定时器操作(TM00 和CR000 匹配时清零并启动)

        MOV     INTM0,  #00000000B      ；设置 INTP1 下降沿有效
        MOV     IF0,    #00H            ；预先清除无效的中断请求
        CLR1    PMK1                    ；不屏蔽 INTP1 中断
        CLR1    TMMK000                 ;不屏蔽INTTM000 中断

        EI                              ；使能向量中断

MAIN_LOOP：
        NOP
        BR      $MAIN_LOOP              ；进入 MAIN_LOOP

                •
                •
        MOV     TMC00,  #00000000B      ；停止定时器操作

        MOV     A,      L               ；读取表地址的低 8 位
        ADD     A,      #2              ；表地址加 2
        AND     A,      #00000111B      ；屏蔽除位 0 至位 2 外的其他位
        MOV     L,      A               ；写入表地址的低 8 位
        MOV     A,      [HL]            ；从表中读取比较值的低 8 位
        MOV     X,      A
        MOV     A,      [HL+1]          ；从表中读取比较值的高 8 位
        MOVW    CR000,  AX              ；改变LED闪烁的时间基准

        MOV     TMC00,  #00001100B      ；启动定时器操作(TM00 和CR000 匹配时清零并启动)
                •
                •
                •
INTERRUPT_TM000：
        CALL    !SUB_INTERRUPT_TM000            ；调用 INTTM000 子程序
        RETI                                    ；从中断服务返回
                •
                •
                •
```

设置间隔时间

设置 CR000 操作模式为比较寄存器

设置计数时钟

禁止定时器输出

启动定时器操作

清除 INTTM000 中断请求标志

使能 INTTM000 中断服务

停止定时器操作后设置 CR000 寄存器

启动定时器操作

通过产生 INTTM000 中断启动中断服务

(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 作为比较寄存器*/
        CR000 = 63-1;                      /*  初始化LED闪烁时间基准*/
        PRM00 = 0b00000010;                /*  计数时钟= fxp/2^8 = 31.25 kHz */
        TOC00 = 0b00000000;                /*  不执行定时器输出*/
        TMC00 = 0b00001100;                /*  启动定时器操作(TM00 和CR000 匹配时清零并启动) */

        INTM0 = 0b00000000;                /*  设置 INTP1 有效沿为下降沿*/
        IF0   = 0x00;                      /*  预先清除无效的中断请求*/
        PMK1  = 0;                         /*  不屏蔽 INTP1 中断*/
        TMMK000 = 0;                       /*  不屏蔽INTTM000 中断*/
        return;
}

void main(void){

        EI();                              /*使能间量中断*/

        while (1){
                NOP();
                NOP();
        }
}
                •
                •
                •
        TMC00 = 0b00000000;                /*停止定时器操作*/
        CR000 = g_unCR000data[g_ucSWcnt];
                                           /*依照开关输入次数改变 LED 闪烁时间基准*/
        TMC00 = 0b00001100;                /*启动定时器操作(TM00 和CR000 匹配时清零并启动) */
                •
                •
                •
interrupt void fn_inttm000(){

        fn_subinttm000();                  /INTTM000 中断服务*/

        return;
                •
                •
                •
```

设置间隔时间

设置 CR000 操作模式为比较寄存器

设置计数时钟

禁止定时器输出

启动定时器操作

使能 INTTM000 中断服务

清除 INTTM000 中断请求标志

停止定时器操作后设置 CR000 寄存器

启动定时器操作

通过 INTTM000 中断产生启动中断服务

4.2 设置LED闪烁周期和抖动检测时间

在该举例程序中 LED 闪烁周期和抖动检测时间设置如下：

(1) 设置 LED 闪烁周期

在该举例程序中，16 位定时器/事件计数器 00 中断(INTTM000)每产生 250 次，LED 输出就反相一次。

- 中断周期(间隔时间) = (N + 1)/fsam。
- LED 输出反相周期=中断周期× 中断次数。
- LED 闪烁周期= LED 输出反相周期× 2。

备注 N： 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。

计算举例： 当 CR000 寄存器的设定值为 62 (在以 fsam = 31.25 kHz 操作期间)时，有以下结果：

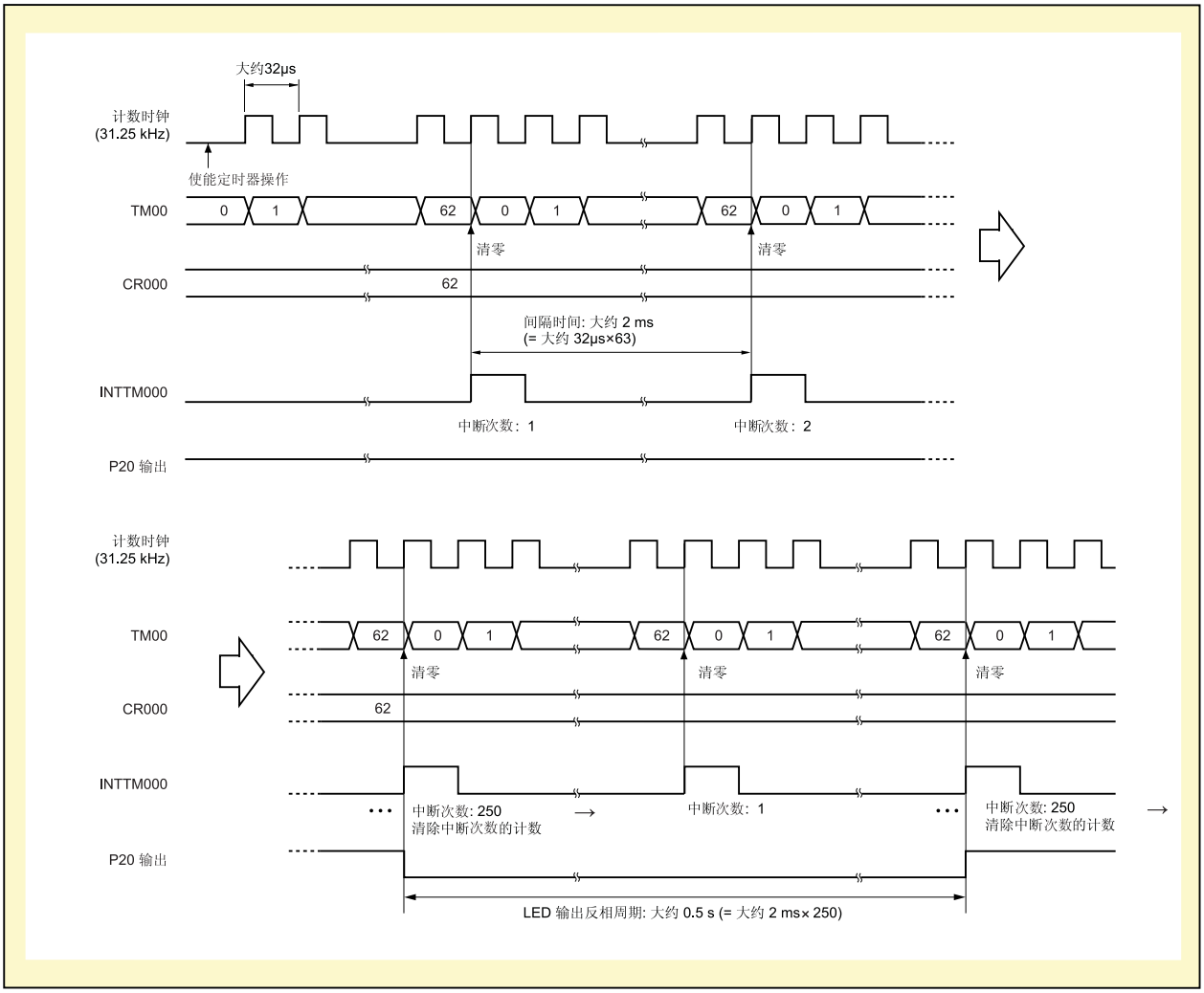
- 中断周期(间隔时间)= (N + 1)/fsam = (62 + 1)/31.25 kHz = 2 ms
- LED 输出反相周期= 中断周期× 中断次数= 2 ms × 250 = 500 ms
- LED 闪烁周期= LED 输出反相周期× 2 = 500 ms × 2 = 1 s

此外， 依照开关输入次数 CR000 寄存器设定值而发生变化，LED 闪烁周期随之改变。

开关输入次数[注]	CR000 寄存器 设定值	中断周期	LED 闪烁周期
0	62	2.016 ms ((62 + 1)/31.25 kHz)	1.008 s (2.016 ms × 250 × 2)
1	31	1.024 ms ((31 + 1)/31.25 kHz)	0.512 s (1.024 ms × 250 × 2)
2	15	0.512 ms ((15 + 1)/31.25 kHz)	0.256 s (0.512 ms × 250 × 2)
3	7	0.256 ms ((7 + 1)/31.25 kHz)	0.128 s (0.256 ms × 250 × 2)

注 第四次开关输入后闪烁周期从第零次开关输入重复。

图 4-6. LED 闪烁周期的时序图举例(当 LED 以大约 1 s 的周期闪烁时)



备注　　当 LED 分别以 1/2 s、 1/4 s 和 1/8 s 周期闪烁时， CR000 寄存器的设定值相应的为 31、 15 和 7。

.

(2) 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。

即使用 INTTM000 中断进行抖动检测，INTTM000 中断也能被连续的计数。因而可以消除由开关输入引起的 LED 闪烁周期的偏差。

- **抖动检测时间**(T_C) = T' + T × (M – 1)

备注　T：　INTTM000 中断周期。

T'：　从 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，

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

↓

抖动检测时间(T_C) > 10 ms

计算举例：　当中断周期(T) 是 2 ms (参见**(1) 设置 LED 闪烁周期**中的计算举例)，INTP1 边沿检测(M)后 INTTM000 中断次数是 6 时

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

$= T' + 2 \text{ ms} \times (6 - 1)$

$= T' + 10$ ms

$0 < T' \leq 2$ ms，因此，

$10 \text{ ms} < T_C \leq 12 \text{ ms}$

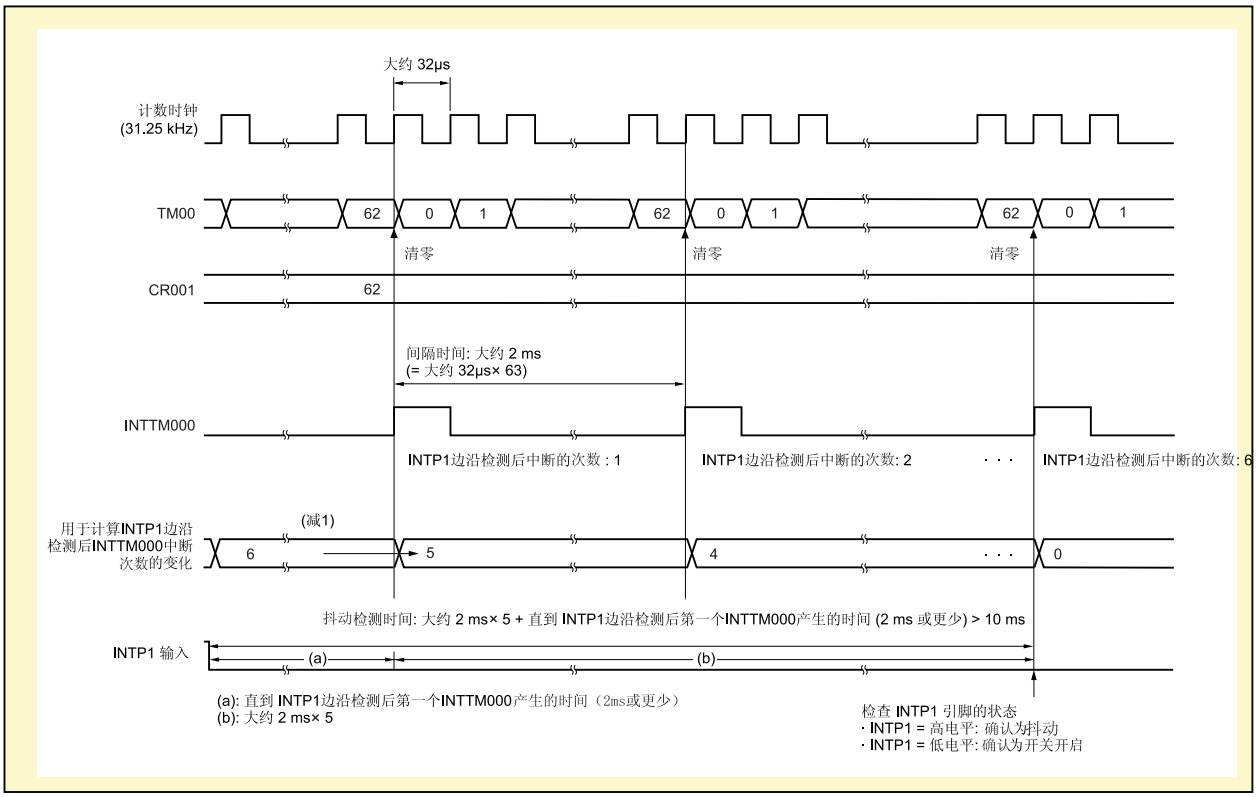
↓

抖动检测时间(T_C) > 10 ms

下表显示了举例程序在开关输入期间的中断周期和 INTP1 边沿检测后 INTTM000 中断的次数之间的对应关系。

LED 闪烁周期	中断周期	INTP1 边沿检测后 INTTM000 中断的次数	抖动检测时间
1.008 s	2.016 ms	6	10.08 ms < $T_C \leq$ 12.096 ms
0.512 s	1.024 ms	11	10.24 ms < $T_C \leq$ 11.264 ms
0.256 s	0.512 ms	21	10.24 ms < $T_C \leq$ 10.752 ms
0.128 s	0.256 ms	41	10.24 ms < $T_C \leq$ 10.496 ms

图 4-7.　　抖动检测的时序图举例(在开关输入期间当 LED 以大约 1 s 的周期闪烁时)



备注　　INTP1 边沿检测后， INTTM000 中断次数的变化取决于在开关输入期间 LED 的闪烁周期。当 LED 分别以 1/2 s、 1/4 s 和 1/8 s 周期闪烁时，中断计数相应为 11、 21 和 41。

使用选择  图标所下载的汇编语言文件（源文件＋工程文件），本章描述了如何使用 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（直至 2007 年 08 月）。因此 78K0S/KU1+微控制器的操作不能由 78K0S/Kx1+的系统仿真器 SM+来检查。

5.1　连编举例程序

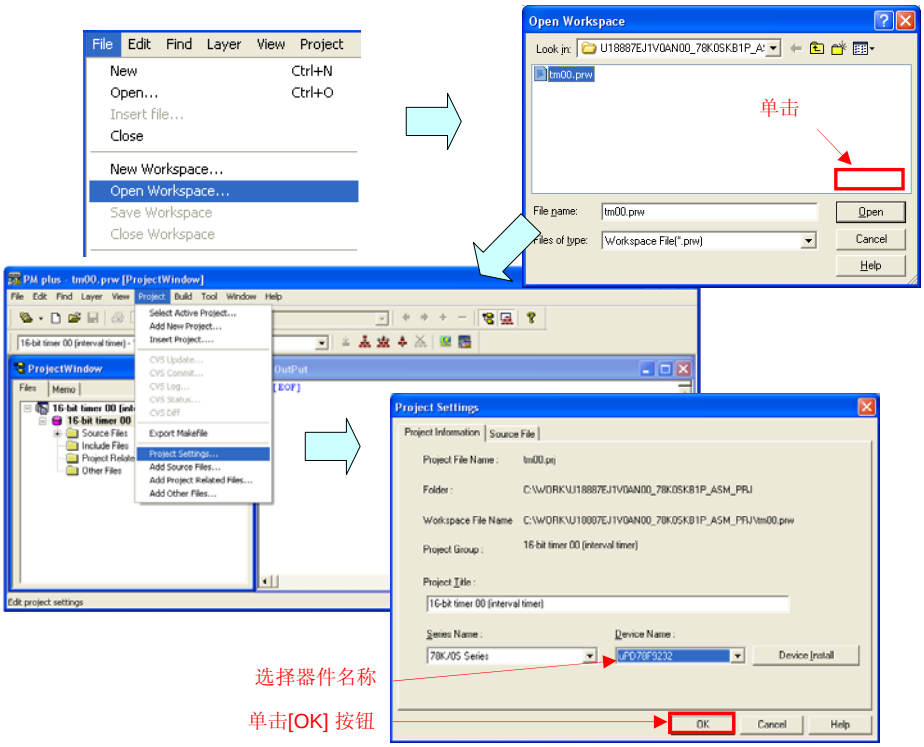
为使用 78K0S/Kx1+的系统仿真器 SM+（以下简称“SM+”）来检查举例程序的操作，必须在连编举例程序之后启动 SM+。本节介绍了操作顺序实例：从用集成开发环境 PM+连编举例程序开始，然后使用选择  图标所下载的汇编语言文件，直到启动 SM+。关于如何编译其他下载的语言程序，参见 **78K0S/Kx1+举例程序启动指南使用说明中的　第三章　注册集成开发环境 PM+工程项目和执行编译。**

关于如何操作 PM+的细节，参见 **PM+ 工程管理用户手册。**

(1) 启动 PM+。

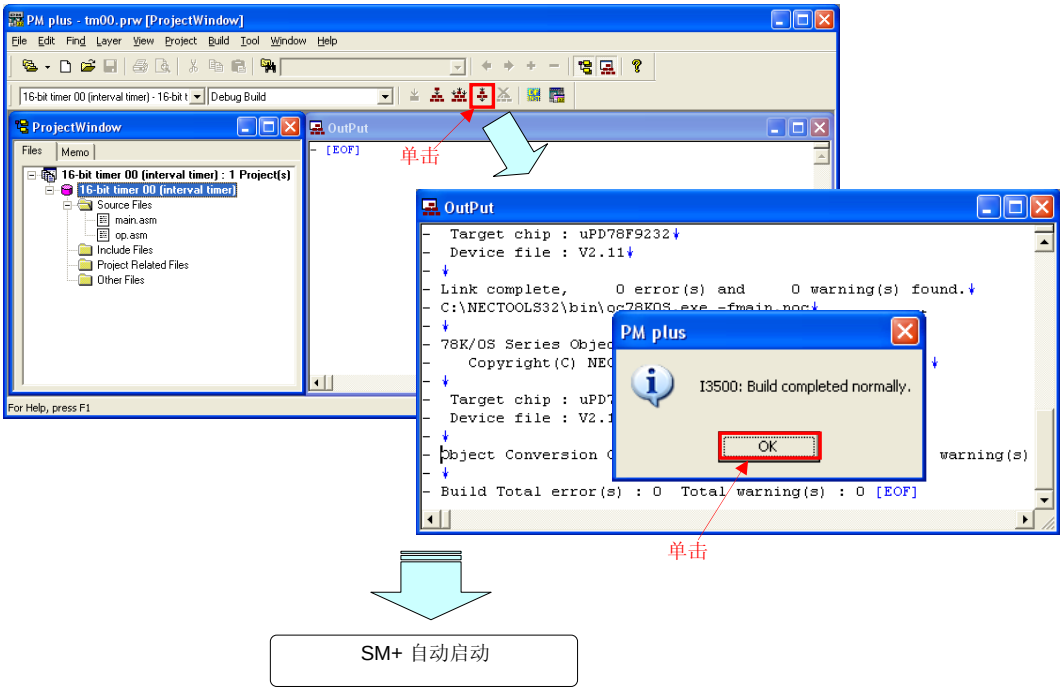
(2) 单击[File]菜单中 [Open Workspace] 选项，选择 “tm00.prw”并单击[Open]按钮。创建一个可自动读取源文件的工作空间。

(3) 选择[Project]菜单中 [Project Settings]选项。 [Project Settings]窗口打开，选择所用的器件名称（缺省选用具有最大容量 ROM 或 RAM 的器件），然后单击[OK]按钮。



(4) 单击  ([Build → Debug]按钮)。当“main.asm”和“op.asm”源文件正常编译完成后， 将显示“I3500：Build completed normally”的信息。

(5) 单击信息框中的[OK]按钮以自动启动 SM+。



5.2 SM+的操作

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

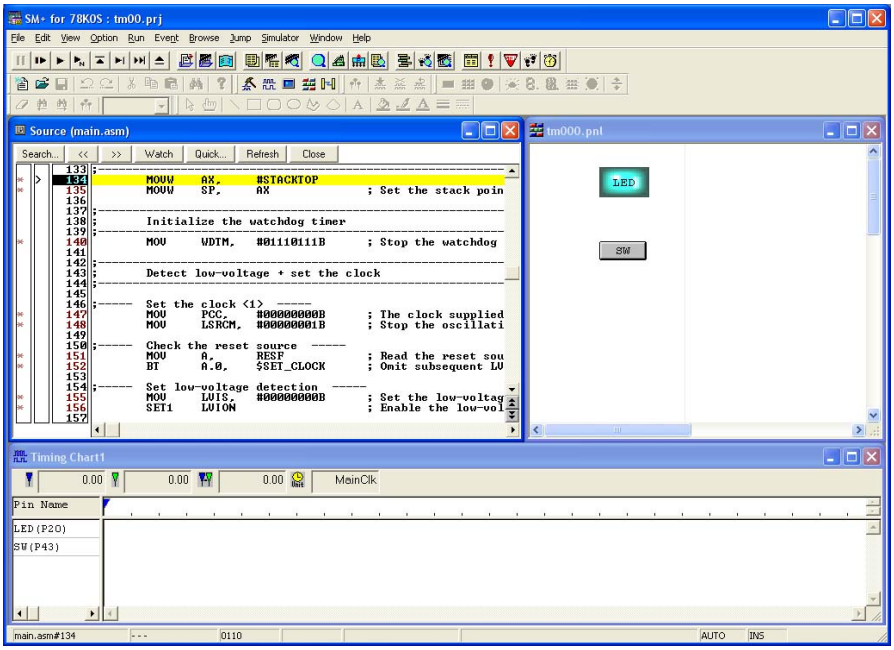
[专栏] 编译链接错误

用 PM+进行编译时，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，根据以下步骤改变编译选项设置。

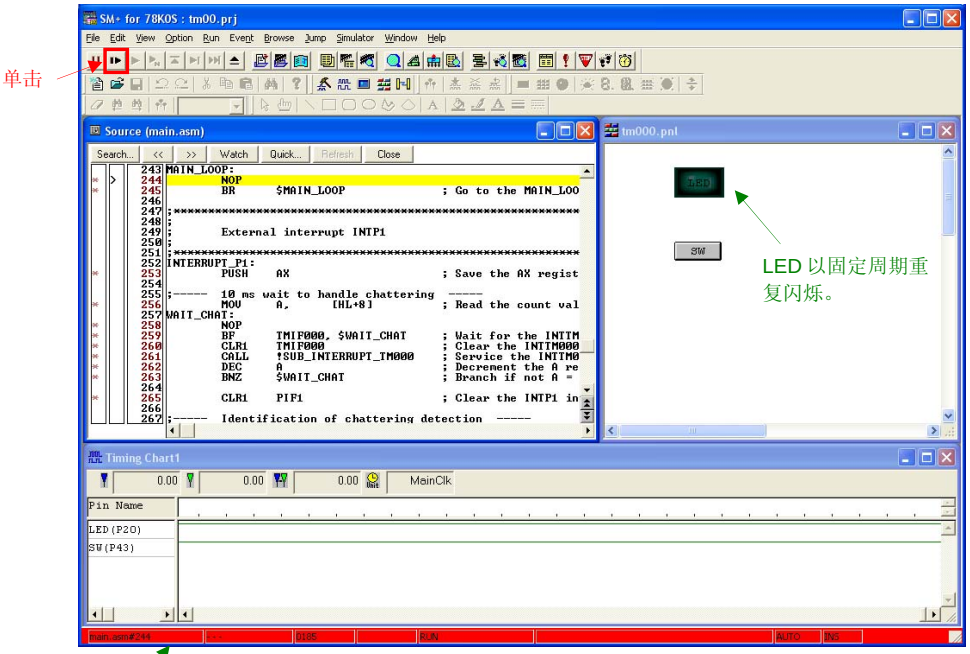
<1> 选择[Tool]菜单中 [Compiler Options]选项。
<2> 显示[Compiler Options]对话框。 选择[Startup Routine]选项。
<3> 取消选择 [Using Fixed Area of Standard Library]复选框。(将其他复选框保持原状。)

当取消[Using Fixed Area of Standard Library]复选框时，只允许使用一个位于 RAM 区域内被保护的 118 字节区域作为固定标准库，但是，禁止使用标准库 (例如 getchar 函数和 malloc 函数)。
在该举例程序中，当使用点击  图标所下载的文件时，[Using Fixed Area of Standard Library] 复选框缺省为不选择。

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

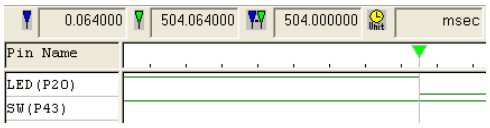
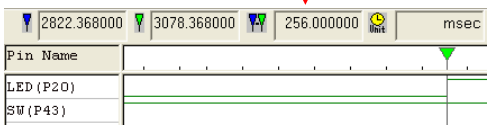
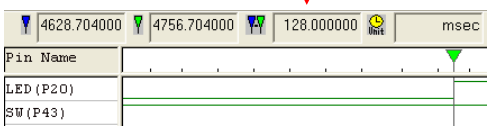
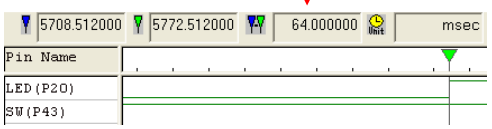


(2) 单击  ([Restart]按钮)。CPU 复位后将执行程序同时显示如下界面：



(3) 程序执行期间，单击 I/O 面板窗口的 [SW]按钮。
根据[SW]按钮输入的次数，在 I/O 面板窗口中观察[LED]灯的闪烁周期，同时观测时序图窗口中波形的变化。

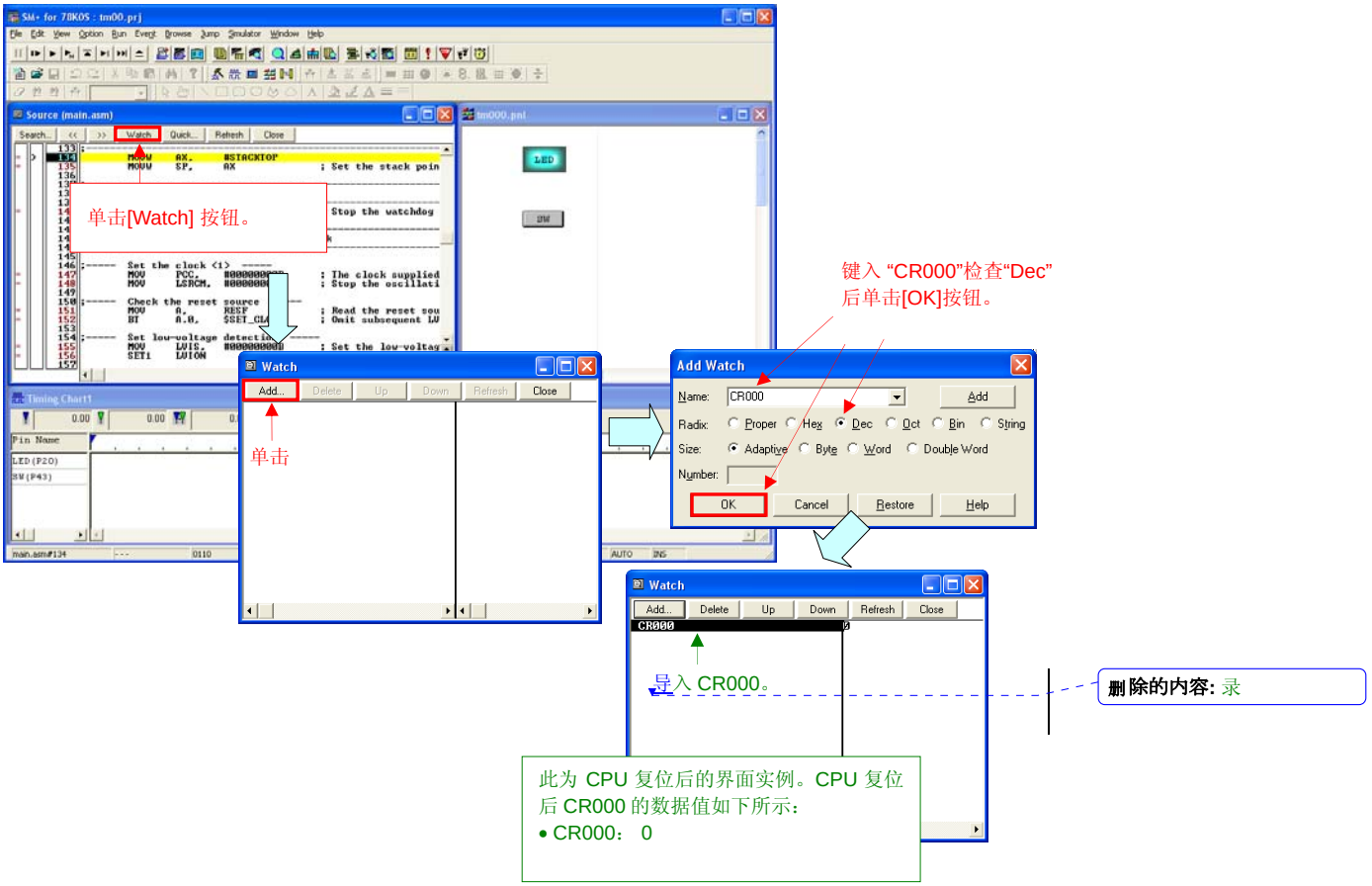
注 1

I/O 面板窗口	时序图窗口
以大约 1s 的周期闪烁[注2]。 LED 不要点击。 SW	LED (P20)的反相周期是 504 ms。 
以大约 1/2s 的周期闪烁[注2]。 LED 点击一次。 SW	LED (P20)的反相周期是 256 ms。 
以大约 1/4s 的周期闪烁[注2]。 LED 点击两次。 SW	LED (P20)的反相周期是 128 ms。 
以大约 1/8s 的周期闪烁[注2]。 LED 点击三次。 SW	LED (P20)的反相周期是 64 ms。 

注　1. 第四次[SW]按钮输入后，闪烁周期从第零次[SW]按钮输入重复。
2. 使用 PC 的操作环境不同，实际闪烁周期可能与此不同。

[补充 1]　　用 SM+的监控功能可检查 CR000 寄存器数据值的变化。

<1>　单击源程序窗口 的[Watch] 按钮以打开[Watch]窗口。
<2>　单击[Add]按钮以打开 [Add Watch] 窗口。(此时，[Watch]窗口仍然处于打开状态。)
<3>　在[Name]域键入“CR000” ，检查 Radix 下的 “Dec”后单击[OK]按钮。于是“CR000”被添加到[Watch]窗口，[Add Watch] 窗口随即关闭。



<4>　执行程序并单击 I/O 面板窗口的[SW]按钮。根据 [SW] 按钮输入的次数，在[Watch]窗口检查 CR000 的数据值变化。

[SW] 按钮输入的次数※	[Watch]窗口的数据值
0	CR000： 62
1	CR000： 31
2	CR000：15
3	CR000：7

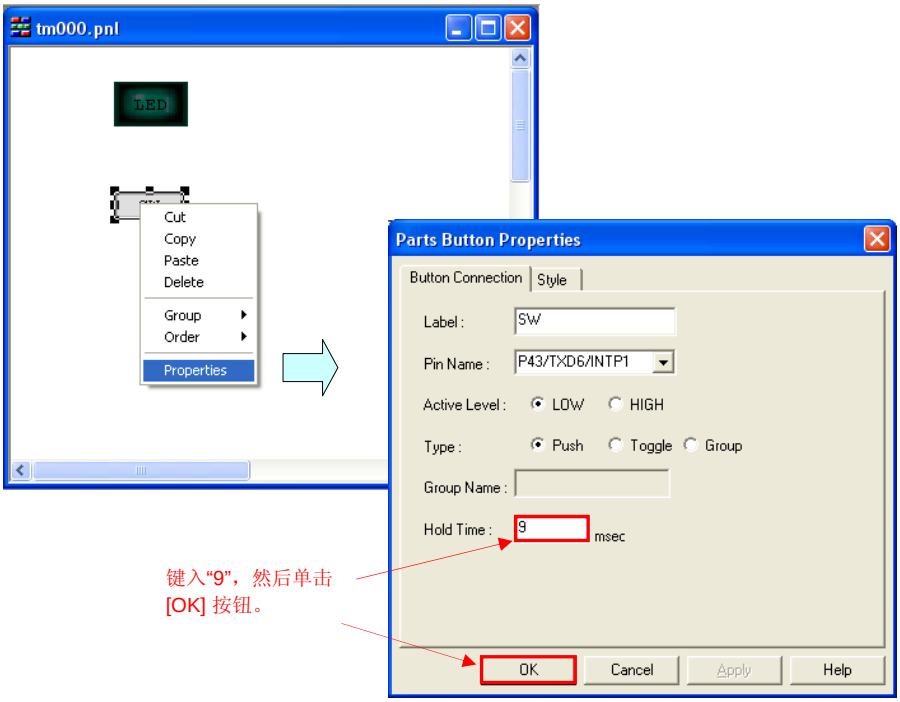
注　　从第四次开关输入后灯亮的方式从第零次开关输入重复。

[补充 2] [SW] 按钮保持时间可设置为小于 10 ms 以检验是否可以检测到抖动。

<1> 在工具栏选择  。

<2> 右击 I/O 面板窗口的[SW] 按钮并选择[Properties]。

<3> 在 Hold Time 键入“9”然后单击[OK] 按钮。



<4> 在工具栏选择  。

<5> 执行程序并单击[SW] 按钮。因为按钮保持时间是 9 ms ，所以即使单击[SW]按钮，也将被视为抖动，因而 LED 闪烁周期不会改变。

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 灯开关控制	PDF
	举例程序（中断）由开关输入产生的外部中断	PDF
	举例程序（低压检测）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）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）PPG（可编程脉冲发生器）输出	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+微控制器源程序作为程序清单实例如下所示：

● main.asm (汇编语言版本)

```
;  ********************************************************************************
;
;         NEC Electronics     78K0S/KB1+
;
;  ********************************************************************************
;         78K0S/KB1+     举例程序
;  ********************************************************************************
;         16 位定时器 00 (间隔定时器)
;  ********************************************************************************
;  <<历史记录 >>
;         2007.7.--          发布
;  ********************************************************************************
;
;  <<概要 >>
;
;  该举例程序提供了一个使用 16 位定时器 00 的间隔定时器功能的实例。
;  通过 16 位定时器 00 中断反转 P20 引脚的输出使 LED 闪烁。
;  当产生开关输入中断时，通过重写定时器的比较寄存器来改变 LED 闪烁周期。
;
;
;
;    <主要设置内容>
;
;    - 停止看门狗定时器操作。
;    - 将低压检测电压(VLVI)设置在 4.3 V +-0.2 V 范围。
;    - 在 VDD >= VLVI 后当检测到 VDD < VLVI 时，产生内部复位信号（低压检测器）。
;    - 设置 CPU 的时钟频率为 8 MHz。
;    - 设置供给外围硬件的时钟频率为 8 MHz。
;    - 设置外部中断 INTP1 的下降沿有效。
;    - 设置在开关输入期间抖动检测的时间为 10ms。
;    - 将 HL 寄存器用作中断服务（类似于一个全局变量）。
;
;    <16 位定时器 00 的设置>
;    - 操作模式： TM00 和 CR000 匹配时清零并启动定时器计数 。
;    - 不执行定时器输出。
;    - 计数时钟= fxp/2^8 (31.25 kHz)。
;    - 定时器周期的初始值= 大约 2 ms (32[us/clk] x 63[count] = 2.016[ms])。
;
;    <开关输入次数和 LED 闪烁周期>
;
;      +-------------------------+
;      |  SW 输入    | LED 闪烁    |
;      |   (P43)       | 周期(P20)    |
;      |-----------------|-----------------|
```

```
;    |  0 次        |   1 秒       |
;    |  1 次        |  1/2 秒      |
;    |  2 次        |   1/4 秒     |
;    |  3 次        |   1/8 秒     |
;    +--------------------------+
;    # 第四次开关输入后闪烁周期从第零次开关输入重复。
;
;
; <<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。
;
; ***************************************************************************
; ===========================================================================
;
;       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    RESET_START         ; (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
; -----  设置定时器 00 周期  -----
      DW    63-1                ;  2 ms 间隔比较值
      DW    32-1                ;  1 ms 间隔比较值
      DW    16-1                ;  0.5 ms 间隔比较值
      DW    8-1                 ;  0.25 ms 间隔比较值
; -----  处理抖动  -----
      DW    5+1                 ;  处理抖动的计数值 (2 ms 间隔)
      DW    10+1                ;  处理抖动的计数值 (1 ms 间隔)
```

```
        DW      20+1                    ;  处理抖动的计数值 (0.5 ms 间隔)
        DW      40+1                    ;  处理抖动计的数值 (0.25 ms 间隔)

;  ================================================================================
;
;       定义 RAM
;
;  ================================================================================
XRAM  DSEG  SADDR
CNT_TM000:  DS      1               ;  计数 INTTM000 中断

;  ================================================================================
;
;       定义内存堆栈区
;
;  ================================================================================
XSTK  DSEG  AT      0FEE0H
STACKEND:
        DS      20H                 ;  内存堆栈区= 32 字节
STACKTOP:                           ;  内存堆栈区的开始地址= FF00H

;  ********************************************************************************
;
;       复位后的初始化
;
;  ********************************************************************************
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    ;  停止看门狗定时器操作

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

;  -----  设置时钟<1>  -----
        MOV   PCC,  #00000000B    ;  提供给 CPU 的时钟频率(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  ;  忽略在 LVI 复位期间后续的 LVI 相关处理并进入 SET_CLOCK

;  -----  设置低压检测  -----
        MOV   LVIS,  #00000000B    ;  将低压检测电平(VLVI)设置在 4.3 V +-0.2 V 范围
```

```
        SET1    LVION                   ；  使能低压检测器操作

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

；----- VDD >= VLVI 等待处理 -----
WAIT_LVI：
        NOP
        BT      LVIF，  $WAIT_LVI       ；  如果 VDD < VLVI，则转移

        SET1    LVIMD                   ；  设置使得当 VDD < VLVI 时产生内部复位信号

；----- 设置时钟<2> -----
SET_CLOCK：
        MOV     PPCC，          #00000000B      ；提供给外围硬件的时钟 (fxp) = fx (= 8 MHz)
                                        ；  -> 提供给 CPU 的时钟(fcpu) = fxp = 8 MHz

；---------------------------------------------------------------------------
；       初始化端口 0
；---------------------------------------------------------------------------
        MOV     P0，    #00000000B      ；  设置 P00-P03 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，    #00000001B      ；  设置 P21-P23 输出锁存为低电平，P20 为高电平（关闭 LED）
        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 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 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      ；  设置 P40-P42 和 P44-P47 为输出模式 ，P43 为输入模式

；---------------------------------------------------------------------------
；       初始化端口 12
；---------------------------------------------------------------------------
        MOV     P12，   #00000000B      ；  设置 P120-P123 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 输出锁存为高电平
; ---------------------------------------------------------------------------
;       初始化通用寄存器和 RAM
; ---------------------------------------------------------------------------
        MOV    CNT_TM000，  #250            ； 初始化 INTTM000 中断的次数
        MOVW HL，    #0100H          ；指定表地址到 HL 寄存器 (用于 INTP1 中断)

; ---------------------------------------------------------------------------
;       设置 16 位定时器 00
; ---------------------------------------------------------------------------
        MOV    CRC00，       #00000000B    ； 用 CR000 作为比较寄存器
        MOVW AX，    #63-1           ； 给 AX 寄存器赋值为 LED 闪烁时间基准初始值
        MOVW CR000，       AX            ； 初始化 LED 闪烁时间基准
        MOV    PRM00，       #00000010B    ； 计数时钟= fxp/2^8 = 31.25 kHz
        MOV    TOC00，       #00000000B    ； 不执行定时器输出
        MOV    TMC00，       #00001100B    ； 启动定时器操作(TM00 和 CR000 匹配时清零并启动)

; ---------------------------------------------------------------------------
;       设置中断
; ---------------------------------------------------------------------------
        MOV    INTM0，       #00000000B    ； 设置 INTP1 的有效沿为下降沿
        MOV    IF0，   #00H                ； 预先清除无效的中断请求
        CLR1   PMK1                        ； 不屏蔽 INTP1 中断
        CLR1   TMMK000                     ； 不屏蔽 INTTM000 中断

        EI                                 ； 使能向量中断

; ***************************************************************************
;
;       主循环
;
; ***************************************************************************
MAIN_LOOP：
        NOP
        BR     $MAIN_LOOP                  ； 进入 MAIN_LOOP

; ***************************************************************************
;
;       外部中断 INTP1
;
; ***************************************************************************
INTERRUPT_P1：
        PUSH   AX                          ； 将 AX 寄存器数据存入堆栈

; ----- 等待 10 ms 以处理抖动 -----
        MOV    A，     [HL+8]          ； 读取与定时器 000 周期相应的计数值
WAIT_CHAT：
        NOP
        BF     TMIF000，  $WAIT_CHAT       ； 等待 INTTM000 中断
        CLR1   TMIF000                     ； 清除 INTTM000 中断请求标志

        CALL   !SUB_INTERRUPT_TM000        ； 调用 INTTM000 中断
```

```
        DEC     A                       ； A 寄存器减 1
        BNZ     $WAIT_CHAT              ； 如果不是 A = 0，则转移

        CLR1    PIF1                    ； 清除 INTP1 中断请求

； ----- 抖动检测的确认 -----
        BT      P4.3， $END_INTP1       ； 如果没有开关输入，则转移

； ----- 改变 TM00 间隔周期 -----
        MOV     TMC00，         #00000000B      ； 停止定时器操作

        MOV     A，     L               ； 读取表地址的低 8 位
        ADD     A，     #2              ； 表地址加 2
        AND     A，     #00000111B      ； 屏蔽除位 0 至位 2 外的其他位
        MOV     L，     A               ； 写入表地址的低 8 位
        MOV     A，     [HL]            ； 从表中读取比较值的低 8 位
        MOV     X，     A
        MOV     A，     [HL+1]  ； 从表中读取比较值的高 8 位
        MOVW CR000，            AX      ； 改变 LED 闪烁的时间基准

        MOV     TMC00，         #00001100B      ； 启动定时器操作(TM00 和 CR000 匹配时清零并启动)

        MOV     CNT_TM000， #250        ； 初始化 INTTM000 中断的次数

END_INTP1：
        POP     AX                      ； 恢复 AX 寄存器数据
        RETI                            ； 从中断服务返回

； ****************************************************************************
；
；       中断 INTTM000
；
； ****************************************************************************
INTERRUPT_TM000：
        CALL    !SUB_INTERRUPT_TM000    ； 调用 INTTM000 子程序
        RETI                            ； 从中断服务返回

； ----------------------------------------------------------------------------
；       测量 INTTM000 中断次数的子程序
； ----------------------------------------------------------------------------
SUB_INTERRUPT_TM000：
        DBNZ    CNT_TM000， $END_INTTM000 ； 如果 INTTM000 中断次数< 250，则转移
        MOV     CNT_TM000， #250        ； 初始化 INTTM000 中断次数

        XOR     P2，    #00000001B      ； 反转 LED 输出

END_INTTM000：
        RET                             ； 从子程序返回

end
```

● main.c (C 语言版本)

```
/*********************************************************************************
        NEC Electronics     78K0S/KB1+
*********************************************************************************
        78K0S/KB1+     举例程序
*********************************************************************************
        16 位定时器 00 (间隔定时器)
*********************************************************************************
<<历史记录 >>
        2007.7.--          发布
*********************************************************************************

<<概要 >>

该举例程序提供了一个使用 16 位定时器 00 的间隔定时器功能的实例。通过用 16 位定时器 00 中断反转
P20 引脚输出使 LED 闪烁。当产生开关输入中断时，通过重写定时器的比较寄存器来改变 LED 闪烁周期。


 <主要的设置内容>

 - 声明由中断运行的函数： INTP1 -> fn_intp1()
 - 声明由中断运行的函数：INTTM000 -> fn_inttm000()
 - 停止看门狗定时器操作
 - 将低压检测电压(VLVI)设置在 4.3 V +-0.2 V 范围
 - 在 VDD >= VLVI 后当检测到 VDD < VLVI 时产生内部复位信号（低压检测器）
 - 设置 CPU 的时钟为 8 MHz
 - 设置供给外围硬件的时钟为 8 MHz
 - 设置外部中断 INTP1 的下降沿有效
 - 设置在开关输入期间抖动检测时间为 10ms


 <16 位定时器 00 的设置>
 - 操作模式： TM00 和 CR000 匹配时清零并启动定时器计数
 - 不执行定时器输出
 - 计数时钟= fxp/2^8 (31.25 kHz)
 - 定时器周期的初始值= 大约 2 ms (32[us/clk] x 63[count] = 2.016[ms])


 <开关输入次数和 LED 闪烁周期>

  +--------------------------+
  | SW 输入     | LED 闪烁|
  |   (P43)        | 周期(P20) |
  |----------------|-------------|
  |   0 次          |    1 秒      |
  |   1 次          |  1/2 秒     |
  |   2 次          |  1/4 秒     |
  |   3 次          |  1/8 秒     |
  +--------------------------+
```

```
  # 在第四次开关输入后闪烁周期从第零次开关输入重复。


<<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。

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


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

        预处理指令(#pragma)

=========================================================================*/
#pragma         SFR                             /* SFR 名称可在 C 源程序层面上描述*/
#pragma         EI                              /* EI 指令可在 C 源程序层面上描述*/
#pragma         NOP                             /* NOP 指令可在 C 源程序层面上描述*/
#pragma interrupt INTP1 fn_intp1        /* 中断函数声明： INTP1 */
#pragma interrupt INTTM000 fn_inttm000          /*中断函数声明： INTTM000 */

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


        声明函数原型

=========================================================================*/
void fn_subinttm000();                  /* INTTM000 中断子程序*/

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

        定义全局变量

=========================================================================*/
sreg unsigned char g_ucSWcnt = 0;       /* 用于计数开关输入次数的 8 位变量*/
sreg unsigned char g_ucTM000cnt = 0;            /* 用于计数 INTTM000 中断次数的 8 位变量*/
const unsigned char g_ucChat[4] = {5+1，10+1，20+1，40+1};              /* 用于消除抖动的 8 位常量表*/
const unsigned int g_unCR000data[4] = {63-1，32-1，16-1，8-1};          /* 用于 LED 闪烁时间基准的 16
位常量表*/

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

        复位后的初始化


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

/*-------------------------------------------------------------------------
        初始化看门狗定时器+检测低压+设置时钟
-------------------------------------------------------------------------*/
```

```
    /*初始化看门狗定时器*/
    WDTM  = 0b01110111；                    /* 停止看门狗定时器操作 */


    /* 设置时钟<1> */
    PCC   = 0b00000000；            /* 供给 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)){      /* 忽略在 LVI 复位期间后续的 LVI 相关处理*/

        /* 设置低压检测*/
        LVIS  = 0b00000000；    /* 将低压检测电平(VLVI)设置在 4.3 V +-0.2 V 范围 */
        LVION = 1；             /* 使能低压检测操作*/

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

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

        LVIMD = 1；                     /* 设置使得当 VDD < VLVI 时产生内部复位信号 */
    }

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


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


/*---------------------------------------------------------------------------
    初始化端口 2
---------------------------------------------------------------------------*/
    P2    = 0b00000001；            /* 设置 P21-P23 输出锁存为低电平，P20 为高电平 (关闭 LED) */
    PM2   = 0b11110000；            /* 设置 P20-P23 为输出模式*/


/*---------------------------------------------------------------------------
    初始化端口 3
---------------------------------------------------------------------------*/
    P3    = 0b00000000；            /* 设置 P30-P33 输出锁存为低电平*/
    PM3   = 0b11110000；            /* 设置 P30-P33 为输出模式*/


/*---------------------------------------------------------------------------
    初始化端口 4
```

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


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

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

/*----------------------------------------------------------------------------
        设置 16 位定时器 00
----------------------------------------------------------------------------*/
        CRC00 = 0b00000000;                     /* 用 CR000 作为比较寄存器*/
        CR000 = 63-1;                           /* 初始化 LED 闪烁时间基准*/
        PRM00 = 0b00000010;                     /* 计数时钟= fxp/2^8 = 31.25 kHz */
        TOC00 = 0b00000000;                     /* 不执行定时器输出*/
        TMC00 = 0b00001100;                     /* 启动定时器操作(TM00 和 CR000 匹配时清零并启动) */

/*----------------------------------------------------------------------------
        设置中断
----------------------------------------------------------------------------*/
        INTM0 = 0b00000000;             /* 设置 INTP1 有效沿为下降沿*/
        IF0   = 0x00;                   /* 预先清除无效的中断请求*/
        PMK1  = 0;                              /*不屏蔽 INTP1 中断 */
        TMMK000 = 0;                            /*不屏蔽 INTTM000 中断 */

        return;
}

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

        主循环

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

        EI();                                   /* 使能向量中断*/

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

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

        外部中断 INTP1
```

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

        for (ucChat = g_ucChat[g_ucSWcnt] ； ucChat > 0 ； ucChat--){ /* 等待大约 10 ms (以消除抖动)
*/
                while (!TMIF000){       /* 等待 INTTM000 中断请求 */
                        NOP();
                }

                TMIF000 = 0;            /* 清除 INTTM000 中断请求标志*/
                fn_subinttm000();       /*服务 INTTM000 中断*/
        }

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

        if (!P4.3){                     /* 如果 SW 开启 10ms 或更长时间则执行处理 */
                g_ucSWcnt = (g_ucSWcnt + 1) & 0b00000011；/* 更新开关输入次数*/

                TMC00 = 0b00000000；            /* 停止定时器操作*/
                CR000 = g_unCR000data[g_ucSWcnt]；          /*依照开关输入次数改变 LED 闪烁时间基
准*/
                TMC00 = 0b00001100；            /* 启动定时器操作(TM00 和 CR000 匹配时清零并启动) */

                g_ucTM000cnt = 0；      /* 清除 INTTM000 中断的次数*/
        }

        return；
}

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

        中断 INTTM000

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

        fn_subinttm000()；              /* 服务 INTTM000 中断*/

        return；
}

/*---------------------------------------------------------------------------
        测量 INTTM000 中断次数的子程序
---------------------------------------------------------------------------*/
void fn_subinttm000(){

        if (++g_ucTM000cnt == 250){     /* 当 INTTM000 中断次数是 250 次时执行处理*/
                g_ucTM000cnt = 0；      /* 清除 INTTM000 中断次数*/
                P2 ^= 0b00000001；      /* 反转 LED 输出*/
        }

        return；
}
```

● op.asm (汇编语言版本和 C 语言版本通用)

```
;  ===============================================================================
;
;         选项字节
;
;  ===============================================================================
OPBT   CSEG   AT       0080H
       DB      10011100B       ;  选项字节区域
;                 ||||
;                 |||+---------- 可用软件停止内部低速振荡器
;                 |++----------- 内部高速振荡时钟 (8 MHz)选作系统时钟信号源
;                 +------------- P34/RESET 引脚用作 RESET 引脚

       DB      11111111B       ;  保护字节区域(用于自编程模式)
;               ||||||||
;               ++++++++----------所有块都能被写入或擦除

end
```

附录 B 版本修订历史

版本	出版日期	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