NEC

使用说明

78K0S/Kx1+

举例程序 (8 位定时器 80)

间隔定时器

本文档介绍了举例程序的操作概要以及使用方法，同时也介绍了如何设置和使用 8 位定时器 80 的间隔定时器功能。此举例程序中，通过使用 8 位定时器 80 的间隔定时器功能，LED 灯以固定周期闪烁，且其闪烁周期根据开关输入次数而改变。

目标器件：

78K0S/KA1+ 微控制器

78K0S/KB1+ 微控制器

目录

文档编号 U18864CA1V0AN00 (第一版)
发布日期 2008 年 3 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

在此举例程序中介绍了一个 8 位定时器 80 的间隔定时器功能应用实例：LED 灯以固定周期闪烁，且其闪烁周期依照开关输入次数的变化而改变。

注意事项 78K0S/KU1+和 78K0S/KY1 微控制器未配置 8 位定时器 80。

1.1 初始设置的主要内容

初始设置的主要内容如下：

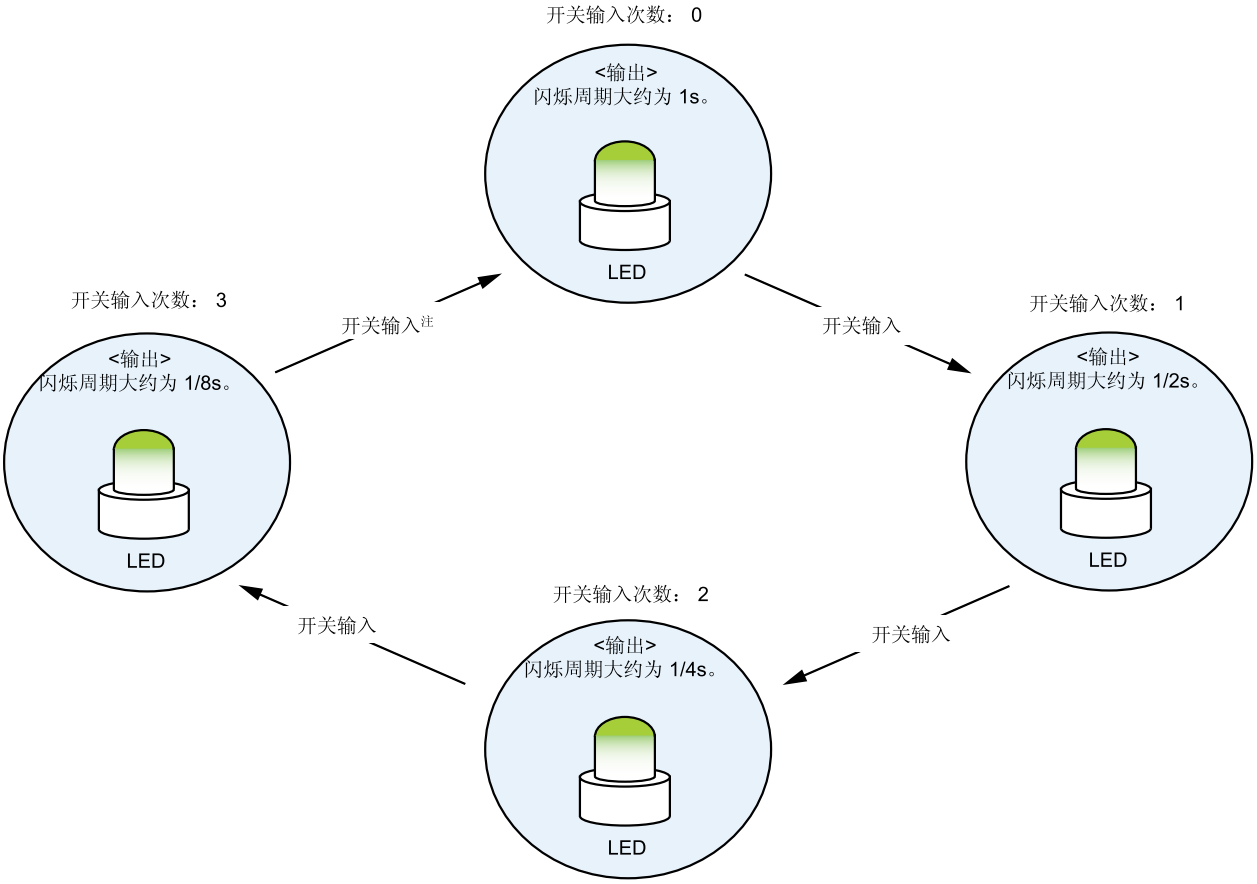
- 选择高速内部振荡器作为系统时钟的信号源[注]。
- 停止看门狗定时器的运行。
- 将 V_{LVI}(低电压检测电压)设置为 4.3 V ±0.2 V。
- V_{DD}(供电电压)高于或等于 V_{LVI} 之后，当检测到 V_{DD} 低于 V_{LVI} 时，产生内部复位(LVI 复位)信号。
- 将 CPU 时钟频率设置为 8 MHz。
- 设置 I/O 端口。
- 设置 8 位定时器 80。
  - 计数时钟设置为 $f_{XP}/2^6$ (125 kHz)。
  - 间隔周期设置为 2 ms (8 μs × 250)。
- 设置 INTP1(外部中断)下降沿有效。
- INTP1 和 INTTM80 中断允许。

注 使用选项字节来设置。

1.2 主循环后的内容

初始设置完成后，利用 8 位定时器 80 产生中断(INTTM80)，使 LED 灯以固定周期闪烁。

当检测到由开关输入产生的 INTP1 引脚的下降沿时，INTP1 服务中断。检测到 INTP1 引脚下降沿 10ms 后，若 INTP1 处于高电平（开关打开），则确认为抖动。而检测到 INTP1 引脚边沿 10ms 后，若 INTP1 为低电平（开关闭合），则此时开关输入有效，LED 灯的闪烁周期依照开关输入有效次数而变。



注 第四次开关输入之后，闪烁周期从第零次开关输入起进行重复。

注意事项 有关使用器件的注意事项，参见各产品 (78K0S/KA1+，78K0S/KB1+)用户手册。

[专栏] 抖动

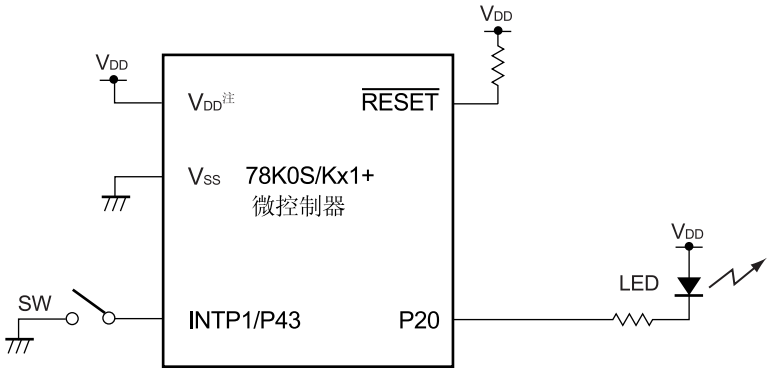
抖动是一种开关按下之后由于机械触点的弹跳而引起电信号瞬时反复接通和断开的现象。

第二章　电路图

本章介绍了该举例程序中所使用的电路图和外围硬件。

2.1　电路图

电路图显示如下：



注　　适用的电压范围：4.5 V ≤ V_{DD} ≤ 5.5 V。

注意事项　1.　直接将 AV_{REF} 引脚连接至 V_{DD}。

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

3.　除电路图中所示引脚及 AV_{REF} 引脚和 AV_{SS} 引脚，所有未用引脚保留为开路状态（未连接）。

2.2　外围硬件

所使用的外围硬件如下所示：

(1)　开关(SW)

此开关用于控制 LED 灯的输入。

(2)　LED

此 LED 用作 8 位定时器 80 的间隔定时器功能和开关输入相应的输出。

第三章　软件

本章介绍了所下载压缩文件的组成、所用微控制器的内部外设功能以及该举例程序中的初始设置和操作概要，并且给出了流程图。

3.1　文件组成

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)	用于微控制器硬件初始化处理和主处理程序的源文件。	●注	●注	
op.asm	用于设置选项字节(设置系统时钟信号源)的汇编程序源文件。	●	●	
tm80.prw	用于集成开发环境PM+的工作空间文件。		●	
tm80.prj	用于集成开发环境PM+的工程文件。		●	
tm80.pri tm80.prs tm80.prm	用于78K0S/Kx1+系统仿真器SM+的工程文件。		●	
tm800.pnl	用于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)。		●	●
tm800.wvo	用于78K0S/Kx1+系统仿真器SM+的时序图文件 (用于检查波形)。			●

注　　汇编语言版本包含“main.asm”文件，而 C 语言版本包含“main.c”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 所使用的内部外设功能

下面显示了在该举例程序中所使用的微控制器的内部外设功能：

- 间隔定时器功能： 8 位定时器 80。
- $V_{DD} < V_{LVI}$ 检测： 低压检测器(LVI)。
- 开关输入： INTP1/P43 (外部中断)。
- LED 输出： P20 (输出端口)。

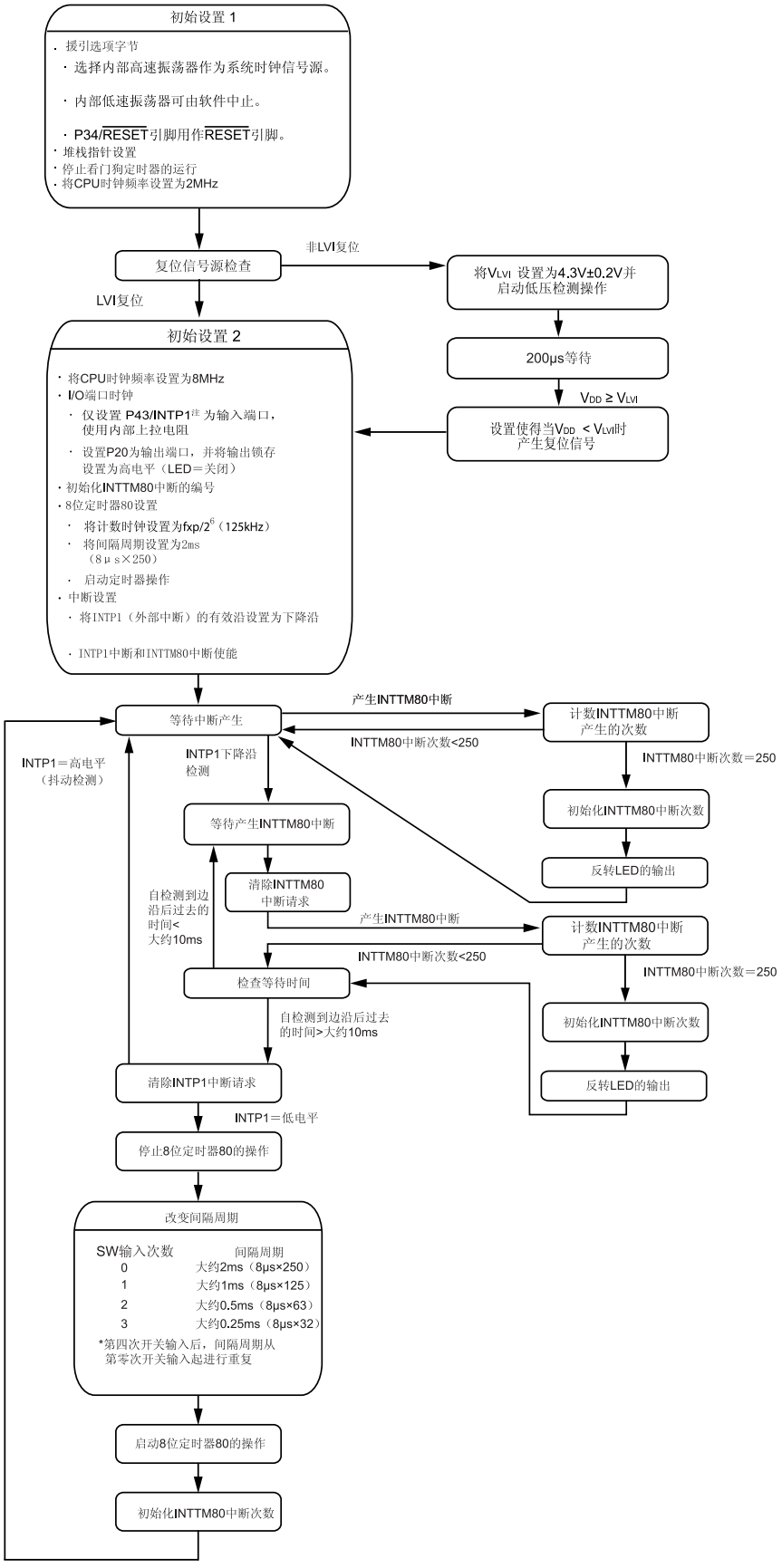
3.3 初始设置和工作概要

在该举例程序中，在初始设置中完成低压检测功能、时钟频率的选择、I/O 端口的设置、8 位定时器 80 的设置以及中断的设置。

初始设置完成后，使用由 8 位定时器 80 产生的中断(INTTM80)使 LED 灯以固定的周期闪烁。

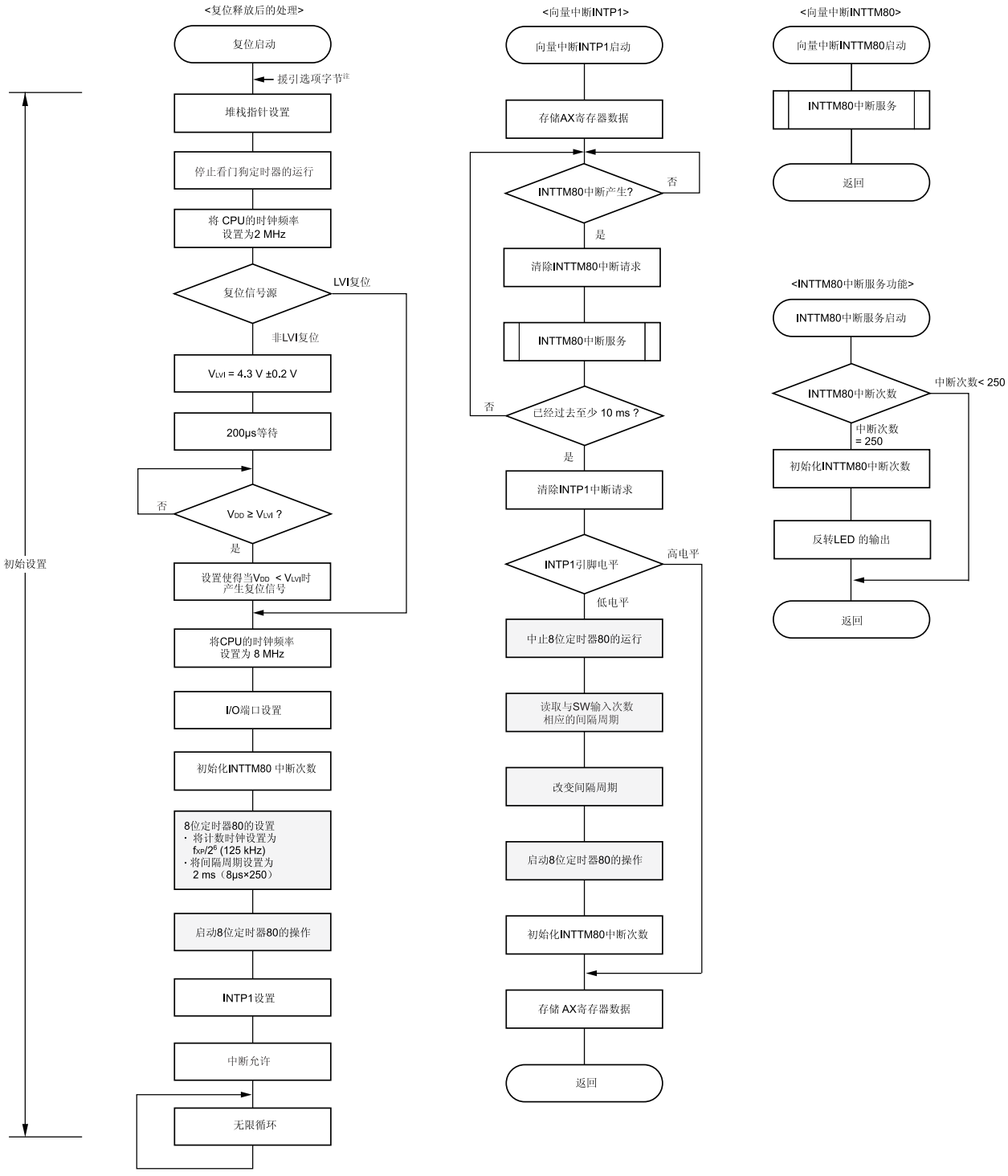
当检测到由开关输入产生的 INTP1 引脚的下降沿时，INTP1 服务中断。检测到 INTP1 引脚下降沿 10ms 后，若 INTP1 为高电平（开关打开），则确认为抖动。而检测到 INTP1 引脚边沿 10ms 后，若 INTP1 为低电平（开关闭合），则此时开关输入有效，LED 灯的闪烁周期依照开关输入有效次数而变。

详情如下列状态转换图所示：



3.4 流程图

本举例程序的流程图如下所示：



注 在复位释放后，微控制器自动援引选项字节。在此举例程序中，援引选项字节设置下面的内容。

- 使用高速内部振荡时钟内部振荡时钟(8 MHz (TYP.))作为系统时钟信号源。
- 能使用软件中止低速内部振荡器。
- 使用 P34/$\overline{\text{RESET}}$ 引脚作为 $\overline{\text{RESET}}$ 引脚。

第四章　设置方法

本章描述了 8 位定时器 80 的间隔定时器功能。

关于其它的初始设置，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**；关于低电压检测(LVI)，参见 **78K0S/Kx1+举例程序（低电压检测）电压低于 2.7V 时的复位使用说明**。

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

关于汇编程序指令，参见 **78K/0S 系列指令用户手册**。

4.1　设置 8 位定时器 80 的间隔定时器功能

当使用 8 位定时器 80 时，设置下面两种类型寄存器：

- 8 位定时器模式控制寄存器 80(TMC80)。
- 8 位比较寄存器 80(CR80)。

(1) 关于 8 位定时器 80 操作模式的设置

选择 8 位定时器 80 的计数时钟，并通过使用 8 位定时器模式控制寄存器 80(TMC80)来控制其操作。

图 4-1. 8 位定时器模式控制寄存器 80 的格式(TMC80)

TMC80

TCE80	0	0	0	0	TCL801	TCL800	0

计数时钟选择

0	0	$f_{XP}/2^6$
0	1	$f_{XP}/2^8$
1	0	$f_{XP}/2^{10}$
1	1	$f_{XP}/2^{16}$

定时器工作使能

0	停止工作 (将计数器清除为0)。
1	工作使能。

注意事项 TCE80 位为 1 时，禁止设置 TCL801 位和 TCL800 位。

备注 f_{XP}： 提供至外围硬件的时钟振荡频率。

(2) 间隔时间设置

使用 8 位比较寄存器 80(CR80)来设置间隔时间。

- 间隔时间 = (N + 1)/f_{CNT}

备注 N: CR80 设置值(00H 至 FFH)。

f_{CNT}： 8 位定时器 80 的计数时钟频率。

图 4-2. 8 位比较寄存器 80(CR80)的格式

CR80

注意事项 在定时器工作期间，禁止重写 CR80 寄存器的值。

[实例 1]

- 将 8 位定时器 80 的计数时钟设置为 $f_{XP}/2^6$ (f_{XP} = 8 MHz)。
- 将间隔周期设置为 2ms，并启动定时器工作。
  (与此举例程序中的内容相同)。

TMC80

TCE80	0	0	0	0	0	0	0

计数时钟选择

0	0	$f_{XP}/2^6$

定时器工作使能

0	停止工作(将计数器清除为0)。
1	工作使能。

CR80 设置值(N)： 249

- 计数时钟：f_{CNT} = 8 MHz/2^6 = 0.125 MHz = 125 kHz
- 间隔周期：2 ms = (N + 1)/125 kHz

→ N = 2 ms × 125 kHz – 1 = 249

设置 TMC80 为“00000000”且设置 CR80 为“249”后，设置 TCE80 为 1 启动定时器工作。

- 汇编语言

```
MOV    TMC80, #00000000B
MOV    CR80,  #249
SET1   TCE80
```

- C 语言

```
TMC80 = 0b00000000;
CR80 = 249;
TCE80 = 1;
```

[实例 2]

- 将 8 位定时器 80 的计数时钟设置为 $f_{XP}/2^{10}$(f_{XP} = 8 MHz)。
- 将间隔周期设置为 32ms，并启动定时器工作。

TMC80

TCE80	0	0	0	0	1	0	0

计数时钟选择

1	0	$f_{XP}/2^{10}$

定时器工作使能

0	停止工作(将计数器清除为0)。
1	工作使能。

CR80 设置值(N)： 249

- 计数时钟：f_{CNT} = 8 MHz/2^{10} = 0.0078125 MHz = 7.8125 kHz
- 间隔周期：32ms = (N + 1)/7.8125 kHz

→ N = 32 ms × 7.8125 kHz – 1 = 249

设置 TMC80 为“00000100”且设置 CR80 为“249”后，设置 TCE80 为 1 启动定时器工作。

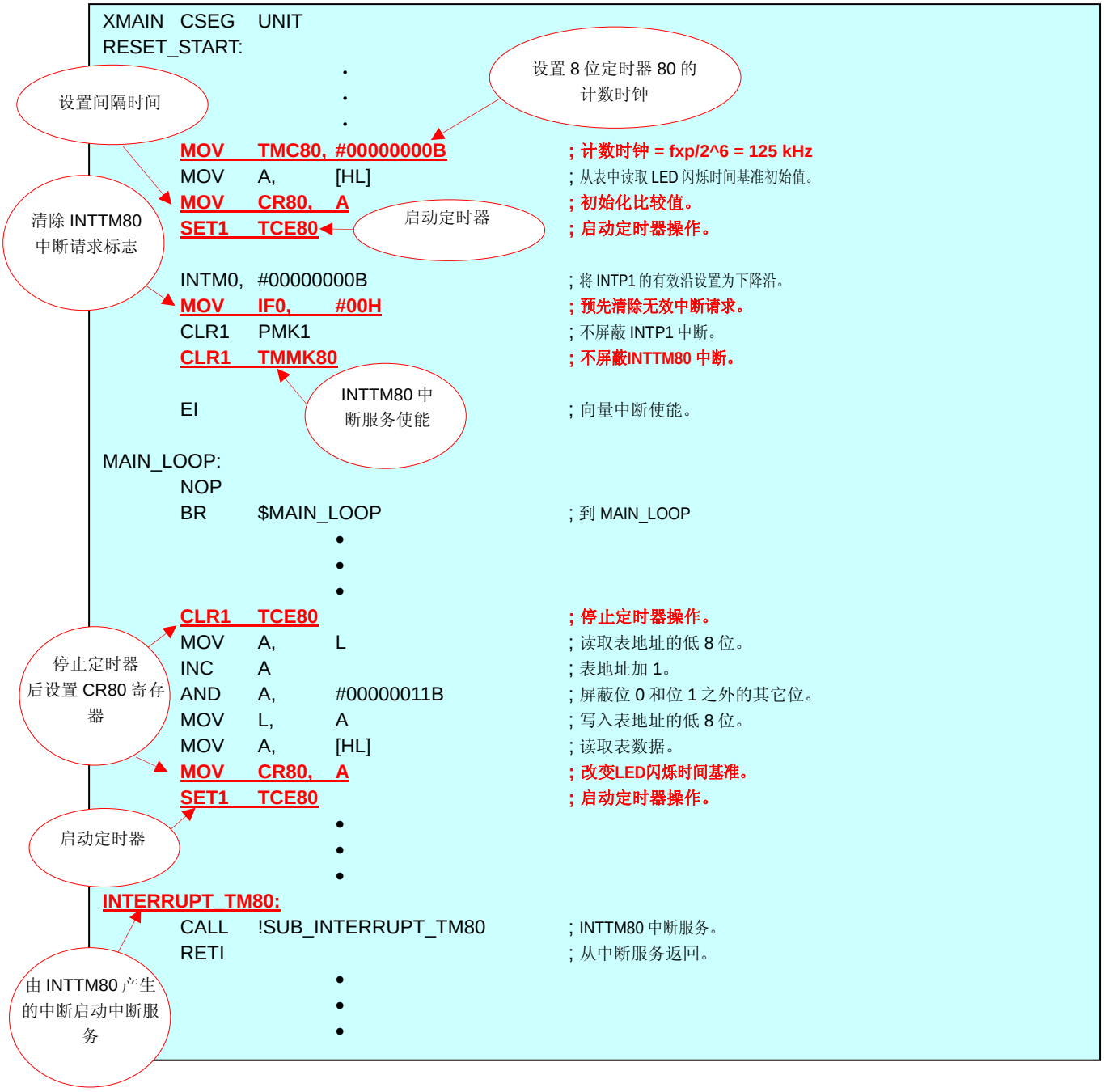
- 汇编语言

```
MOV    TMC80, #00000100B
MOV    CR80,  #249
SET1   TCE80
```

- C 语言

```
TMC80 = 0b00000100;
CR80 = 249;
TCE80 = 1;
```

- 汇编语言程序举例(与上面提到的[实例 1]和举例程序中的内容相同)




```
XMAIN   CSEG    UNIT
RESET_START:
                .
                .
                .
        MOV     TMC80, #00000000B       ; 计数时钟 = fxp/2^6 = 125 kHz
        MOV     A,      [HL]            ; 从表中读取 LED 闪烁时间基准初始值。
        MOV     CR80,   A               ; 初始化比较值。
        SET1    TCE80                   ; 启动定时器操作。

        INTM0,  #00000000B              ; 将 INTP1 的有效沿设置为下降沿。
        MOV     IF0,    #00H            ; 预先清除无效中断请求。
        CLR1    PMK1                    ; 不屏蔽 INTP1 中断。
        CLR1    TMMK80                  ; 不屏蔽INTTM80 中断。

        EI                              ; 向量中断使能。

MAIN_LOOP:
        NOP
        BR      $MAIN_LOOP              ; 到 MAIN_LOOP
                .
                .
                .
        CLR1    TCE80                   ; 停止定时器操作。
        MOV     A,      L               ; 读取表地址的低 8 位。
        INC     A                       ; 表地址加 1。
        AND     A,      #00000011B      ; 屏蔽位 0 和位 1 之外的其它位。
        MOV     L,      A               ; 写入表地址的低 8 位。
        MOV     A,      [HL]            ; 读取表数据。
        MOV     CR80,   A               ; 改变LED闪烁时间基准。
        SET1    TCE80                   ; 启动定时器操作。
                .
                .
                .
INTERRUPT_TM80:
        CALL    !SUB_INTERRUPT_TM80     ; INTTM80 中断服务。
        RETI                            ; 从中断服务返回。
                .
                .
                .
```

- C 语言程序举例(与上面提到的[实例 1]和举例程序中的内容相同)

```
void hdwinit(void){
        unsigned char ucCnt200us;          /* 用于 200 us 等待的 8 位变量。 */
                    •
                    •
                    •
        TMC80 = 0b00000000;                /* 计数时钟 = fxp/2^6 = 125 kHz */
        CR80  = 250-1;                     /* 初始化LED闪烁时间基准。*/
        TCE80 = 1;                         /* 启动定时器操作。 */

        INTM0 = 0b00000000;                /* 将 INTP1 的有效边沿设置为下降沿。 */
        IF0   = 0x00;                      /* 预先清除无效中断请求。 */
        PMK1  = 0;                         /* 不屏蔽 INTP1 中断。 */
        TMMK80 = 0;                        /* 不屏蔽INTTM80 中断。 */
        return;
}

void main(void){

        EI();                              /* 向量中断使能。 */

        while (1){
                NOP();
                NOP();
        }
}
                    •
                    •
                    •
        TCE80 = 0;                         /* 停止定时器的操作。 */
        CR80 = g_ucCR80data[g_ucSWcnt];
                                           /* 改变 LED 闪烁时间基准，其依照开关输入次数而变化。 */
        TCE80 = 1;                         /*    启动定时器操作。 */
                    •
                    •
                    •
__interrupt void fn_inttm80(){

        fn_subinttm80();                   /* INTTM80 中断服务。 */

        return;
                    •
                    •
                    •
```

设置间隔时间

设置 8 位定时器 80 的计数时钟

清除 INTTM80 中断请求标志

启动定时器

INTTM80 中断服务允许

停止定时器操作后设置 CR80 寄存器

启动定时器

通过 INTTM80 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

在此举例程序中 LED 闪烁周期和抖动检测时间的设置如下：

(1) LED 闪烁周期的设置

在此举例程序中，8 位定时器 80 (INTTM80)每产生 250 次中断 LED 输出反转一次。

- 中断周期(间隔时间) = (N + 1)/f_{CNT}
- LED 输出反转周期= 中断周期× 中断次数
- LED 闪烁周期 = LED 输出反转周期 × 2

备注 N: CR80 寄存器设置值。

f_{CNT}: 8 位定时器 80 的计数时钟频率。

计算举例： 当 CR80 寄存器设置值为 249 时(当工作于 f_{CNT} = 125kHz 期间时)，结果如下所示：

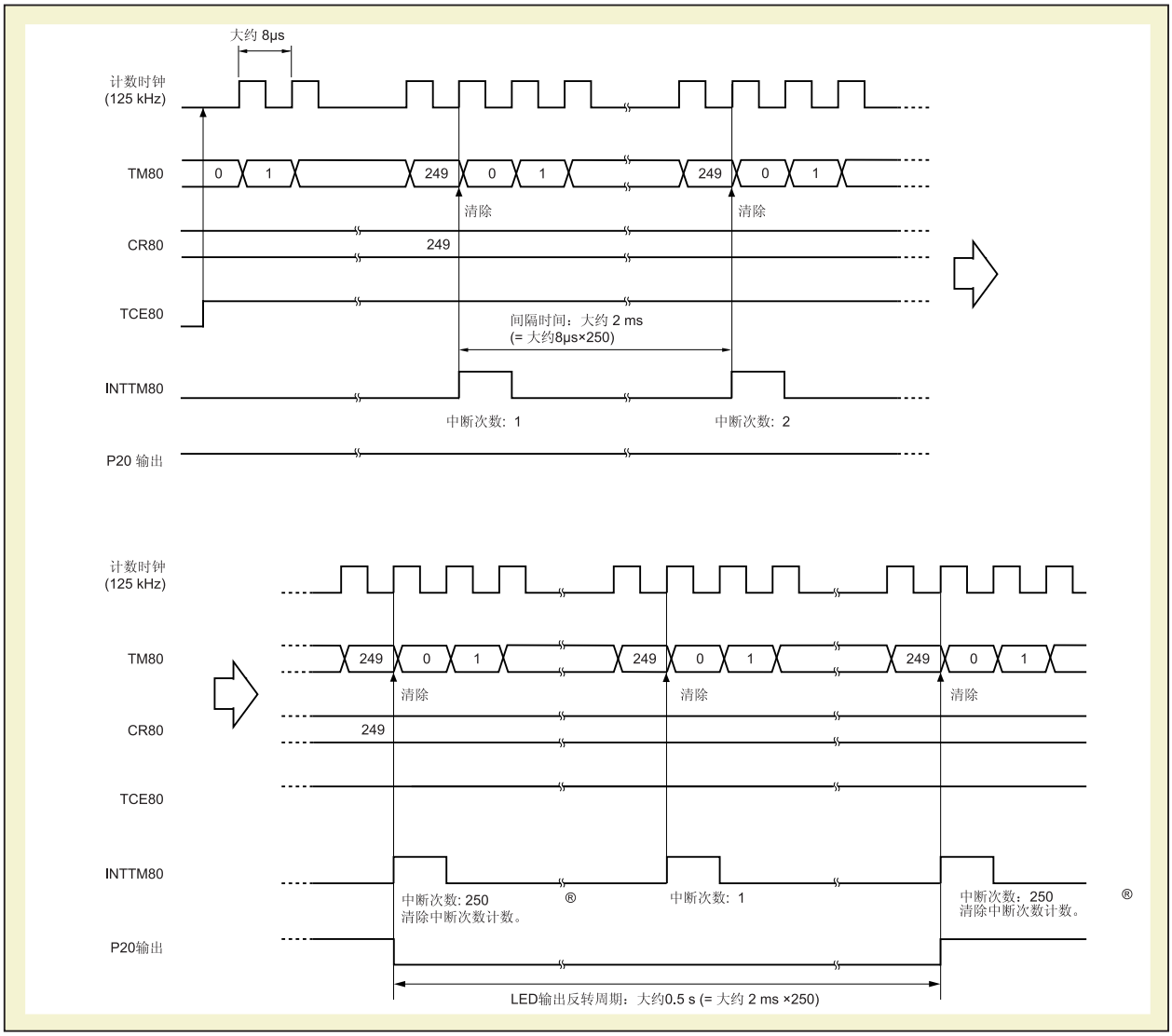
- 中断周期(间隔时间) = (N + 1)/f_{CNT} = (249 + 1)/125 kHz = 2 ms
- LED 输出反转周期 = 中断周期 × 中断次数 = 2 ms × 250 = 500 ms
- LED 闪烁周期 = LED 输出反转周期 × 2 = 500 ms × 2 = 1 s

此外，CR80 寄存器设置值依照开关输入次数的变化而改变，而且 LED 闪烁周期也随之改变。

开关输入次数[注]	CR80 寄存器设置值	中断周期	LED 闪烁周期
0	249	大约 2 ms ((249 + 1)/125 kHz)	大约 1 s (大约 2 ms × 250 × 2)
1	124	大约 1 ms ((124 + 1)/125 kHz)	大约 0.5 s (大约 1 ms × 250 × 2)
2	62	大约 0.504 ms ((62 + 1)/125 kHz)	大约 0.252 s (大约 504 μs × 250 × 2)
3	31	大约 0.256 ms ((31 + 1)/125 kHz)	大约 0.128 s (大约 256 μs × 250 × 2)

注 第四次开关输入之后，闪烁周期从零次状态开始重复。

图 4-3. LED 闪烁周期时序图实例(LED 闪烁周期大约为 1s)



备注　　当 LED 的闪烁周期为 1/2s，1/4s 和 1/8s 时，CR80 寄存器设置值相应为 124，64 和 31。

(2) 抖动检测时间的设置

在此举例程序中为了处理在开关输入（INTP1 中断产生）期间的抖动，8 位定时器 80 中断(INTTM80)产生已考虑了消除 10ms 或更短时间的抖动。

因为即使在使用 INTTM80 中断检测抖动期间， INTTM80 中断也能连续计数，故能够抑制由开关输入引起的 LED 闪烁周期的偏差。

• 抖动检测时间(T_C) = T' + T × (M – 1)

备注　T:　INTTM80 中断周期。

T':　从开始检测到 INTP1 边沿起直到产生第一个 INTTM80 的时间($0 < T' \leq T$)。

M:　INTP1 边沿检测后 INTTM80 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，

$10\ ms < T_C \leq T + 10\ ms$

↓

抖动检测时间(T_C) > 10 ms

计算举例:　当中断周期(T)为 2 ms (参见**(1) 设置 LED 闪烁周期**的计算举例)，并且在 INTP1 边沿检测(M)后 INTTM80 中断次数为 6 时：

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

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

$= T' + 10$ ms

$0 < T' \leq 2$ ms，因此，

$10\ ms < T_C \leq 12\ ms$

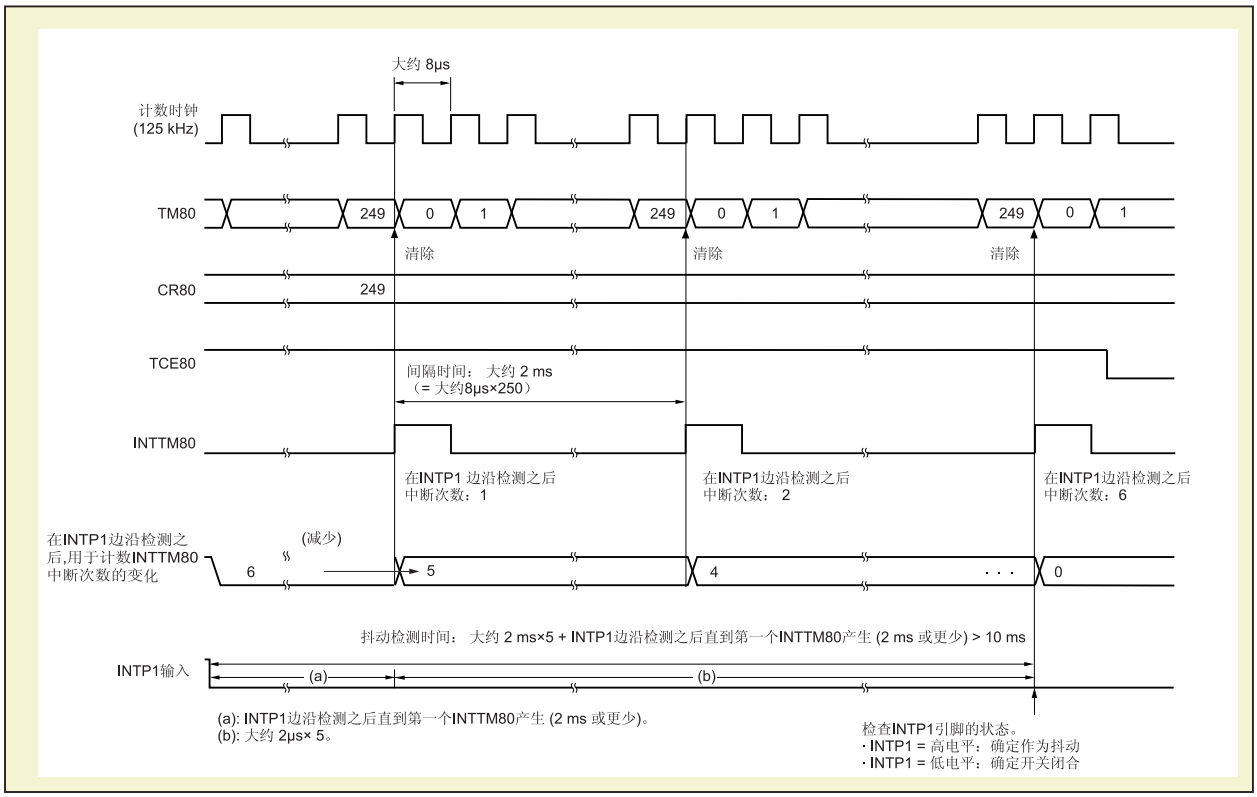
↓

抖动检测时间(T_C) > 10 ms

下表显示了举例程序中开关输入期间中断周期和 INTP1 边沿检测之后 INTTM80 中断次数之间的对应关系：

LED 闪烁周期	中断周期	INTP1 边沿检测后 INTTM80 中断次数	抖动检测时间
大约 1 s	大约 2 ms	6	$10\ ms < T_C \leq 12\ ms$
大约 0.5 s	大约 1 ms	11	$10\ ms < T_C \leq 11\ ms$
大约 0.252 s	大约 0.504 ms	21	$10.08\ ms < T_C \leq 10.584\ ms$
大约 0.128 s	大约 0.256 ms	41	$10.24\ ms < T_C \leq 10.496\ ms$

图 4-4.　　抖动检测的时序图实例(开关输入期间，LED 闪烁周期大约为 1s)



备注　　INTP1 边沿检测后，INTTM80 中断次数计数的变化，取决于开关输入期间 LED 的闪烁周期。当 LED 闪烁周期分别为 1/2 s、1/4 s 和 1/8 s 时，中断计数相应为 11，21 和 41。

第五章　使用系统仿真器SM+进行操作检验

使用由选择图标  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。

5.1　连编举例程序

为了使用 78K0S/Kx1+的系统仿真器 SM+（以下简称“SM+”）来检验举例程序的操作，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：从由集成开发环境 PM+连编举例程序开始，然后使用由选择  图标所下载的汇编语言文件(源文件+工程文件)，直到启动 SM+。

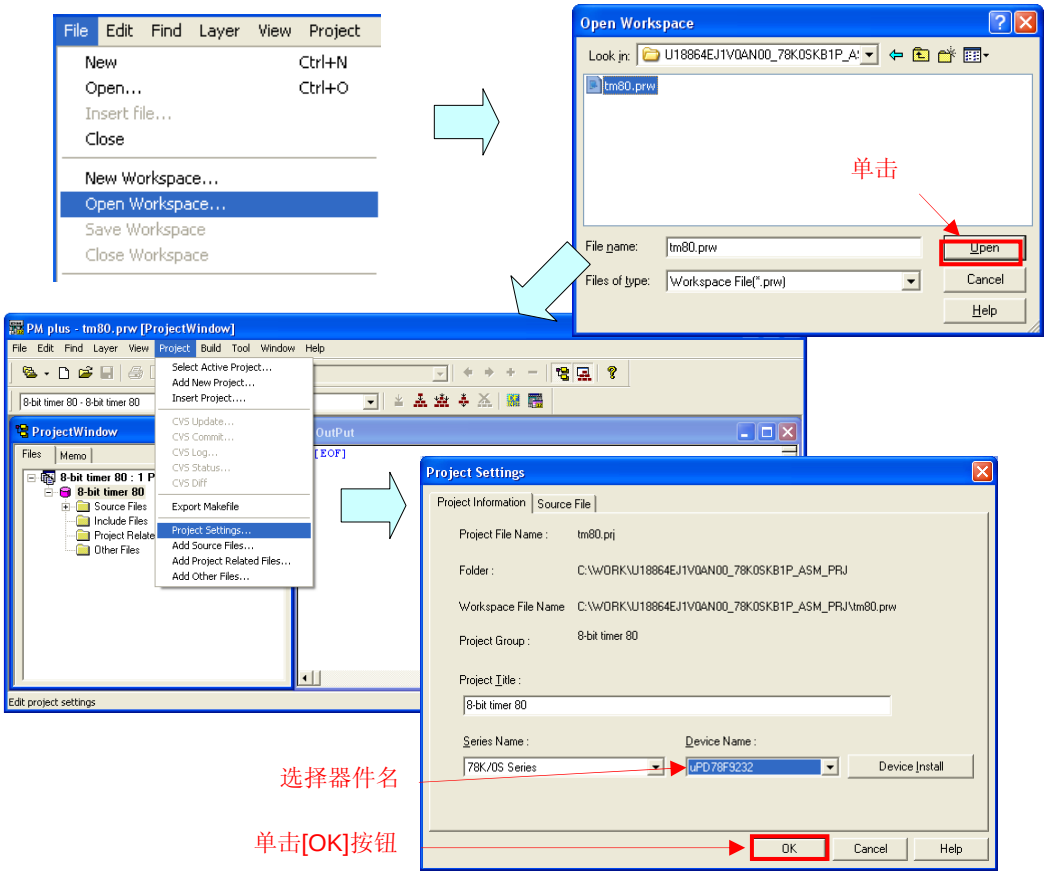
关于如何编译其它下载程序，参见 **78K0S/Kx1+举例程序启动指南使用说明**中的**第三章 注册集成开发环境 PM+工程项目和执行编译。**

关于如何操作 PM+的详细情况，参见 **PM+工程管理用户手册。**

(1) 启动 PM+。

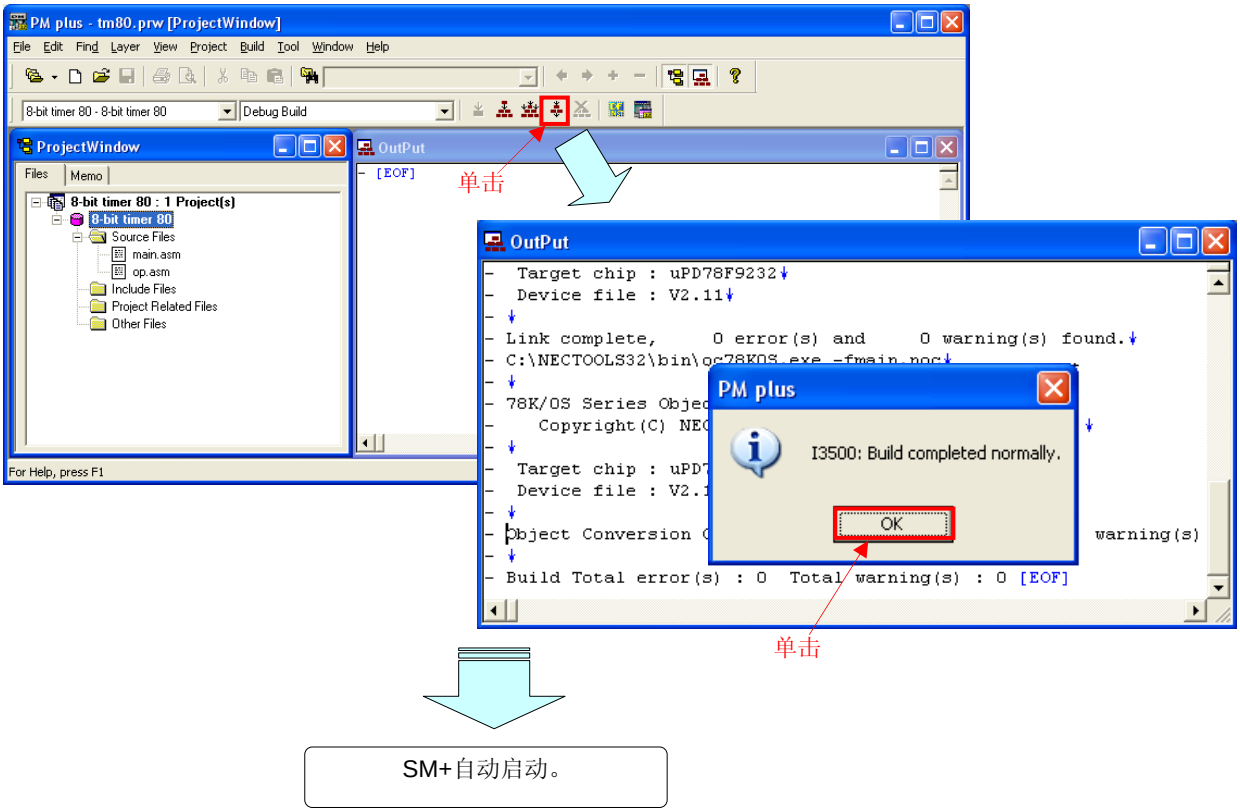
(2) 单击[File]菜单中的[Open Workspace]选项，选择“tm80.prw”并单击 [Open]按钮，创建一个可自动读取源文件的工作站。

(3) 选择[Project]菜单中的[Project Settings]选项。打开[Project Settings]选项窗口，选择所用的器件名（默认选用 ROM 或 RAM 最大容量的器件），然后单击[OK]按钮。



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

(5) 单击信息窗口中的[OK]按钮，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+的详细情况，参见 **SM+系统仿真器操作用户手册**。

[专栏]编译错误

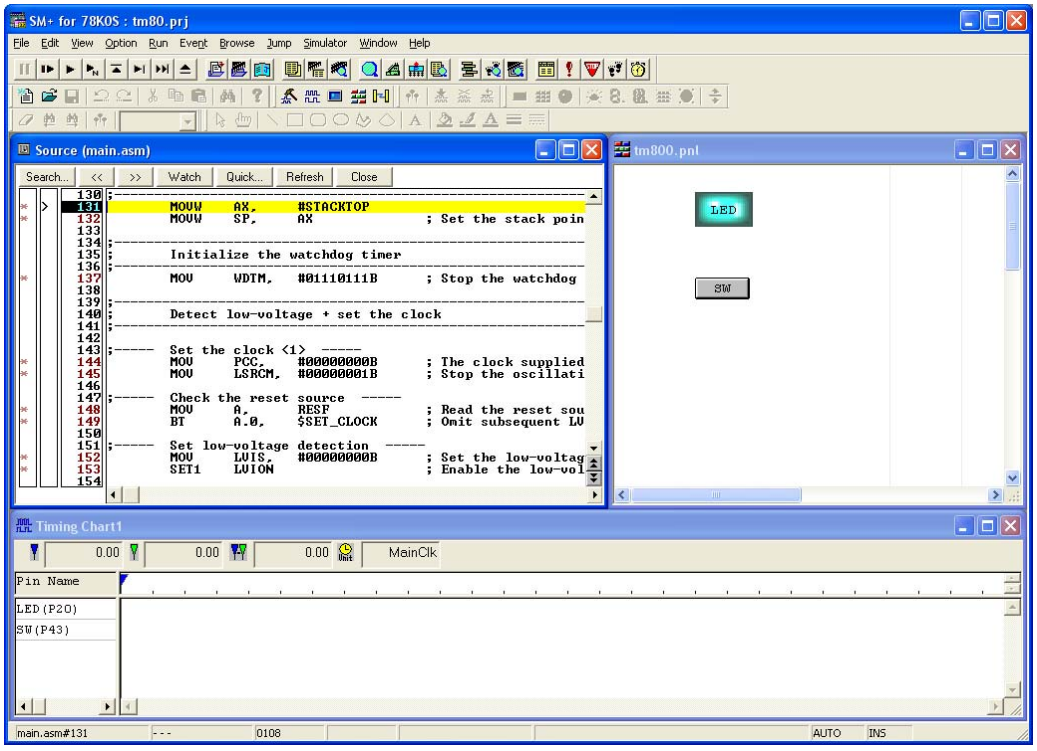
使用 PM+进行编译，当显示 “A006 File not found ‘C:\NECTOOLS32\LIB78K0S\s0sl.rel’”或“*** ERROR F206 Segment ‘@@DATA’ can’t allocate to memory - ignored.” 错误信息时，按照下面的步骤改变编译选项设置。

<1> 从[Tool]菜单中选择[Compiler Options]。
<2> 将会显示[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。(其它复选框不变。)

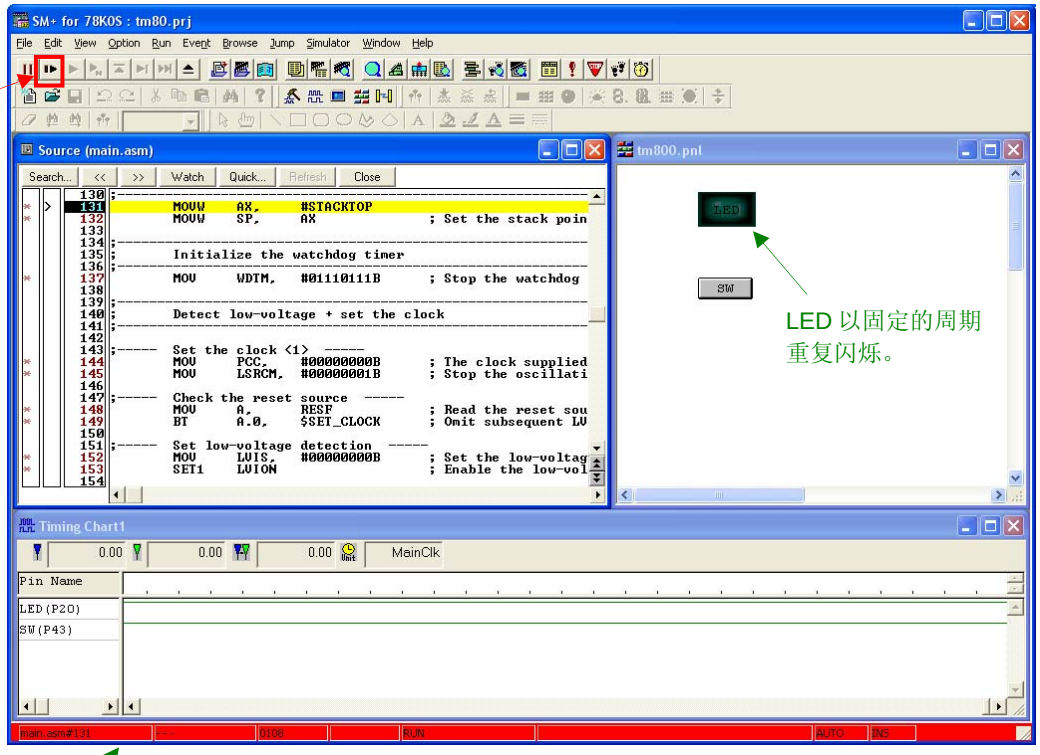
当取消 [Using Fixed Area of Standard Library]复选框时，只允许使用一个位于 RAM 区域内被保护的 118 字节区域作为固定标准库域，但是，禁止使用标准库(例如 getchar 函数和 malloc 函数)。

在此举例程序中，当使用由单击  图标下载的文件时，[Using Fixed Area of Standard Library]复选框缺省为不选择。

(1)　在 PM+窗口中单击[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。）

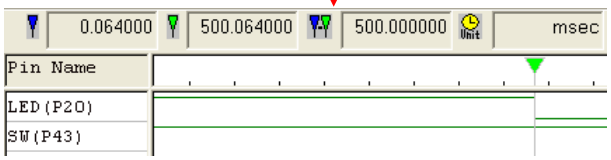
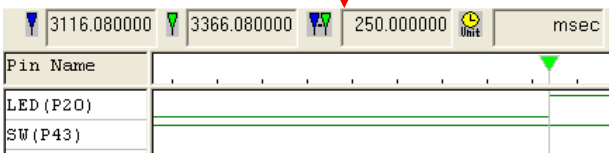
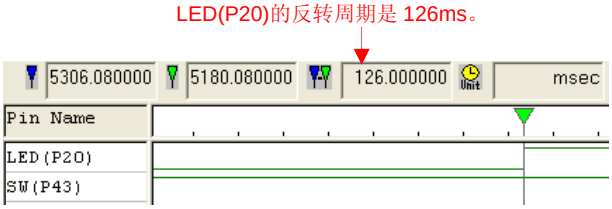
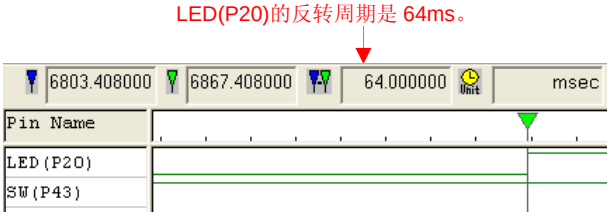


(2) 单击  ([Restart]按钮)。CPU 复位后执行程序，同时显示如下界面：



(3) 程序执行期间，在 I/O 面板窗口中单击[SW]按钮。依据[SW]按钮输入次数，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 注 2 为周期闪烁。 LED 不要单击。 SW	LED(P20)的反转周期是 500 ms。 
以 1/2s 注 2 为周期闪烁。 LED 点击一次。 SW	LED(P20)的反转周期是 250 ms。 
以大约 1/4s 注 2 为周期闪烁。 LED 点击两次。 SW	LED(P20)的反转周期是 126ms。 
以大约 1/8s 注 为周期闪烁。 LED 点击三次。 SW	LED(P20)的反转周期是 64ms。 

注　**1.** 第四次[SW]按钮输入后，闪烁周期从第零次[SW]按钮输入起进行重复。

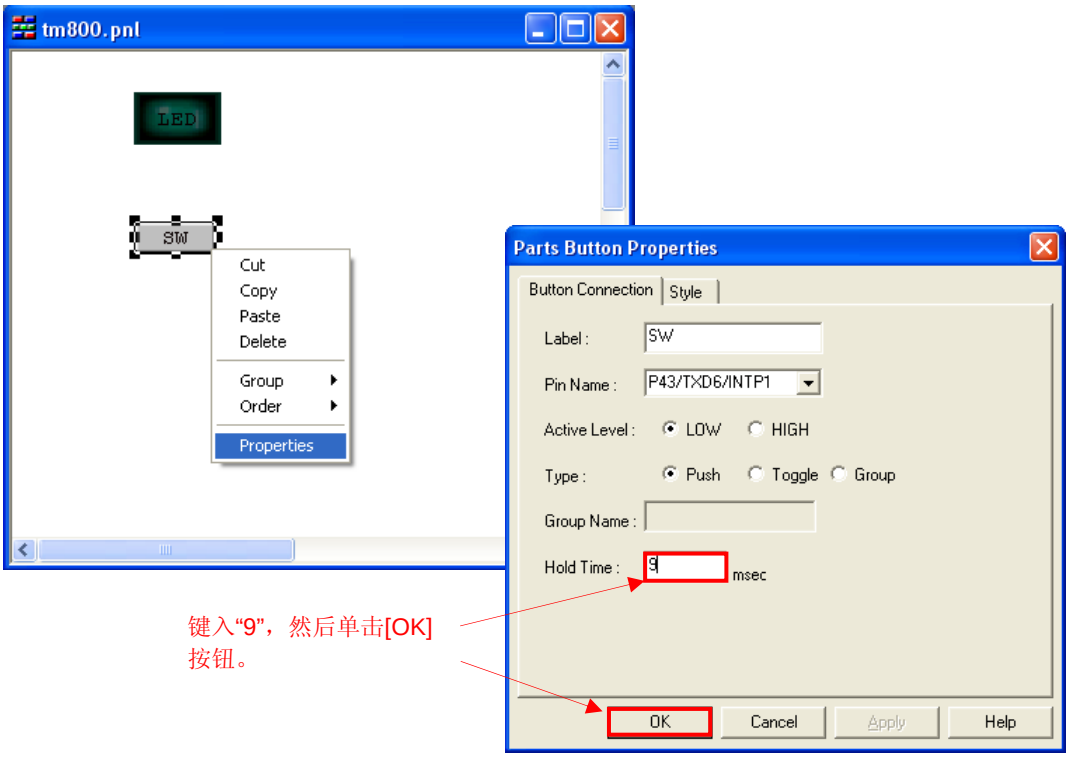
2. 使用 PC 的操作环境不同，实际闪烁周期可能与此不同。

[补充]　　[SW]按钮的保持时间能被设置为低于 10ms，以检验是否可以检测到抖动。

<1> 在工具栏上选择  。

<2> 在面板窗口右击[SW]按钮并选择[Properties]。

<3> 保持时间键入“9”并单击[OK]按钮。



<4> 在工具栏上选择  。

<5> 执行程序并单击[SW]按钮。因为按钮保持时间为 9ms，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/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（初始化设置）LED灯开关控制	PDF
	举例程序由开关输入产生外部中断	PDF
	举例程序（低电压检测）电压低于2.7V时的复位	PDF

附录A 程序清单

78K0S/KB1+微控制器源程序作为程序实例如下所示：

● main.asm (汇编语言版本)

```
;  ********************************************************************************
;
;         NEC Electronics     78K0S/KB1+
;
;  ********************************************************************************
;         78K0S/KB1+     举例程序
;  ********************************************************************************
;         8 位定时器 80
;  ********************************************************************************
;  <<历史记录>>
;         2007.7.--          发布
;  ********************************************************************************
;
;  <<概要>>
;
;  本举例程序提供了一个使用 8 位定时器 80 间隔定时器功能的实例。
;  通过使用 8 位定时器 80 中断反转 P20 引脚的输出使得 LED 灯闪烁。
;  当产生开关输入中断时，LED 闪烁周期由重写定时器的比较寄存器来改变。
;
;
;    <主要设置内容>
;
;    - 停止看门狗定时器的操作。
;    - 将低电压检测电压(VLVI)设置为 4.3 V +-0.2 V。
;    - VDD>=VLVI 之后，当 VDD<VLVI 时，产生内部复位信号(低电压检测器)。
;    - 将 CPU 时钟设置为 8MHz。
;    - 将提供至外围硬件的时钟设置为 8MHz。
;    - 设置外部中断 INTP1 下降沿有效。
;    - 在开关输入期间，设置抖动检测时间为 10ms。
;
;
;    <8 位定时器 80 设置>
;    - 计数时钟 = fxp/2^6 (125 kHz)
;    - 定时器周期的初始值 = 2 ms (8[us/clk] x 250[计数值] = 2[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    |
;    +-------------------------+
;    # 闪烁周期在第四次开关输入后，从第零次开关输入起进行重复。
;
;
; <<I/O 端口设置>>
;
;  输入: P43
;  输出: P00-P03、P20-P23、P30-P33、P40-P42、P44-P47、P120-P123、P130。
;  #将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    RESET_START       ; (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    INTERRUPT_TM80    ; (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 80 周期  -----
      DB    250-1             ;  2ms 间隔比较值。
      DB    125-1             ;  1ms 间隔比较值。
      DB    63-1              ;  0.5ms 间隔比较值。
      DB    32-1              ;  0.25ms 间隔比较值。
; -----  处理抖动  -----
      DB    5+1               ;  处理抖动的计数值(2ms 间隔)。
```

```
        DB      10+1            ;  处理抖动的计数值(1ms 间隔)。
        DB      20+1            ;  处理抖动的计数值(0.5ms 间隔)。
        DB      40+1            ;  处理抖动的计数值(0.25ms 间隔)。


;  ================================================================================
;
;       定义 RAM
;
;  ================================================================================
XRAM  DSEG  SADDR
CNT_TM80:   DS      1               ;  计数 INTTM80 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:                           ;  内存堆栈区的起始地址=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。

;  -----  检查复位源信号  -----
        MOV   A,      RESF          ;  读取复位源信号。
        BT    A.0,    $SET_CLOCK    ;  在 LVI 复位期间忽略后续 LVI 相关处理并进入 SET_CLOCK。
```

```
; ----- 设置低电压检测 -----
        MOV    LVIS,   #00000000B    ;  设置低电压检测电平(VLVI)为 4.3 V +-0.2 V。

        SET1   LVION                 ;  低电压监测器操作使能。

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

; ----- VDD >= VLVI 等待处理 -----
WAIT_LVI:
        NOP
        BT     LVIF,   $WAIT_LVI     ;  如果 VDD < VLVI，跳转。
        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    ;  将 P00-P03 的输出锁存设置为低电平。
        MOV    PM0,    #11110000B    ;  将 P00-P03 设置为输出模式。

; ---------------------------------------------------------------------------
;       初始化端口 2
; ---------------------------------------------------------------------------
        MOV    P2,     #00000001B    ;  将 P21-P23 的输出锁存设置为低电平，将 P20 的输出锁存设置
为高电平(关闭 LED)。
        MOV    PM2,    #11110000B    ;  将 P20-P23 设置为输出模式。

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

; ---------------------------------------------------------------------------
;       初始化端口 4
; ---------------------------------------------------------------------------
        MOV    P4,     #00000000B    ;  将 P40-P47 的输出锁存设置为低电平。
        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，将 P43 设置为输入
模式。

; ---------------------------------------------------------------------------
;       初始化端口 12
; ---------------------------------------------------------------------------
        MOV    P12,    #00000000B    ;  将 P120-P123 的输出锁存设置为低电平。
```

```
        MOV    PM12,  #11110000B    ;  将 P120-P123 设置为输出模式。

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

; ---------------------------------------------------------------------------
;       初始化通用寄存器和 RAM
; ---------------------------------------------------------------------------
        MOV    CNT_TM80, #250       ;  初始化 INTTM80 中断次数。
        MOVW HL,      #0100H;  指定表地址到 HL (由于 INTP1 中断)。

; ---------------------------------------------------------------------------
;       设置 8 位定时器 80
; ---------------------------------------------------------------------------
        MOV    TMC80,#00000000B     ;  计数时钟 = fxp/2^6 = 125 kHz
        MOV    A,     [HL]          ;  从表中读取 LED 闪烁的时间基准初始值。

        MOV    CR80,  A             ;  初始化比较值。
        SET1   TCE80                ;  启动定时器操作。

; ---------------------------------------------------------------------------
;       设置中断
; ---------------------------------------------------------------------------
        MOV    INTM0, #00000000B    ;  将 INTP1 的有效边沿设置为下降沿。
        MOV    IF0,   #00H          ;  预先清除无效中断请求。
        CLR1   PMK1                 ;  不屏蔽中断 INTP1。
        CLR1   TMMK80               ;  不屏蔽中断 INTTM80。

        EI                          ;  向量中断允许。

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

; ***************************************************************************
;
;       外部中断 INTP1
;
; ***************************************************************************
INTERRUPT_P1:
        PUSH   AX                   ;  将 AX 寄存器的数据保存到堆栈。

; -----  等待 10 ms 处理抖动-----
        MOV    A,     [HL+4]  ;  读取与定时器 80 周期相应的计数值。
WAIT_CHAT:
        NOP
```

```
        BF      TMIF80,         $WAIT_CHAT  ;  等待 INTTM80 中断
        CLR1    TMIF80          ;  清除 INTTM80 中断请求标志。
        CALL    !SUB_INTERRUPT_TM80     ;  调用 INTTM80 中断。
        DEC     A                       ;  A 寄存器减 1
        BNZ     $WAIT_CHAT              ;  如果非 A = 0，跳转。

        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，跳转。

;  -----  改变 TM80 间隔周期  -----
        CLR1    TCE80                   ;  停止定时器操作。
        MOV     A,      L               ;  读取表地址的低 8 位。
        INC     A                       ;  表地址加 1。
        AND     A,      #00000011B      ;  屏蔽非位 0 和非位 1。
        MOV     L,      A               ;  写入表地址的低 8 位。
        MOV     A,      [HL]            ;  读取表数据。
        MOV     CR80,   A               ;  改变 LED 闪烁的时间基准。

        SET1    TCE80                   ;  启动定时器操作。

        MOV     CNT_TM80, #250          ;  初始化 INTTM80 中断次数。

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

;  ****************************************************************************
;
;       中断 INTTM80
;
;  ****************************************************************************
INTERRUPT_TM80:
        CALL    !SUB_INTERRUPT_TM80     ;  调用 INTTM80 中断。
        RETI                            ;  从中断服务返回。

;  ----------------------------------------------------------------------------
;       测量 INTTM80 中断次数的子程序
;  ----------------------------------------------------------------------------
SUB_INTERRUPT_TM80:
        DBNZ    CNT_TM80, $END_INTTM80  ;  如果 INTTM80 中断次数 < 250，跳转。
        MOV     CNT_TM80, #250          ;  初始化 INTTM80 中断次数。

        XOR     P2,     #00000001B      ;  LED 输出反转。

END_INTTM80:
        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
*********************************************************************************
        8 位定时器 80
*********************************************************************************
<<历史记录>>
        2007.7.--          发布
*********************************************************************************

<<概要>>

本举例程序提供了一个使用 8 位定时器 80 间隔定时器功能的实例。
通过使用 8 位定时器 80 中断反转 P20 引脚的输出使得 LED 灯闪烁。
当产生开关输入中断时，LED 闪烁周期由重写定时器的比较寄存器来改变。

 <主要设置内容 >

 - 声明由中断运行的函数: INTP1 -> fn_intp1()
 - 声明由中断运行的函数: INTTM80 -> fn_inttm80()
 - 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 之后，当 VDD < VLVI 时，产生内部复位信号(低电压检测器)。
 - 将 CPU 时钟设置为 8MHz 。
 - 将提供至外围硬件的时钟设置为 8MHz。
 - 将外部中断 INTP1 的有效边沿设置为下降沿。
 - 在开关输入期间，将抖动检测时间设置为 10ms 。


 <8 位定时器 80 的设置>
 - 计数时钟 = fxp/2^6 (125 kHz)
 - 定时器周期的初始值 = 2 ms (8[us/clk] x 250[计数值] = 2[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         |
   +-------------------------+
   # 闪烁周期在第四次开关输入之后从第零次输入起进行重复。


<<I/O 端口设置 >>
```

```
  输入: P43
  输出: P00-P03、P20-P23、P30-P33、P40-P42、P44-P47、P120-P123、P130。
  # 所有未使用端口都设置为输出模式。

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


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

        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 INTTM80 fn_inttm80  /* 中断函数声明:INTTM80 */

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

        声明函数原型

============================================================================*/
void fn_subinttm80();                   /* INTTM80 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_ucTM80cnt = 0;   /* 用于计数 INTTM80 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 char g_ucCR80data[4] = {250-1,125-1,63-1,32-1};          /* 用于 LED 闪烁时间基准的 8 位
常量表。*/

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

        复位后初始化

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

/*---------------------------------------------------------------------------
        初始化看门狗定时器 + 低电压检测 + 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。 */

    /* 检查复位源信号 */
    if (!(RESF & 0b00000001)){      /* 在 LVI 复位期间，忽略后续 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;                       /* 低电压检测器操作使能。 */

        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;          /* 将 P00-P03 的输出锁存设置为低电平。 */
    PM0   = 0b11110000;          /* 将 P00-P03 设置为输出模式。 */

/*---------------------------------------------------------------------------
    初始化端口 2
---------------------------------------------------------------------------*/
    P2    = 0b00000001;          /* 将 P21-P23 的输出锁存设置为低电平，将 P20 的输出锁存设置
为高(LED 关闭)。 */

    PM2   = 0b11110000;          /* 将 P20-P23 设置为输出模式。*/

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

/*---------------------------------------------------------------------------
    初始化端口 4
---------------------------------------------------------------------------*/
    P4    = 0b00000000;          /* 将 P40-P47 的输出锁存设置为低电平。 */
    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，将 P43 设置为输入模
式。 */

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

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

/*---------------------------------------------------------------------------
        设置 8 位定时器 80
---------------------------------------------------------------------------*/
        TMC80 = 0b00000000;                     /* 计数时钟 = fxp/2^6 = 125 kHz */
        CR80  = 250-1;                          /* 初始化 LED 闪烁时间基准。 */
        TCE80 = 1;                              /* 启动定时器操作。 */

/*---------------------------------------------------------------------------
        设置中断
---------------------------------------------------------------------------*/
        INTM0 = 0b00000000;             /* 设置 INTP1 为下降沿有效。 */
        IF0   = 0x00;                   /* 预先清除无效中断请求。 */
        PMK1  = 0;                              /* 不屏蔽 INTP1 中断。 */
        TMMK80 = 0;                             /* 不屏蔽 INTTM80 中断。*/

        return;
}

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

        主循环

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

        EI();                                   /* 向量中断使能。 */

        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 (用于消除抖动) 。*/
                while (!TMIF80){        /* 等待 INTTM80 中断请求 。*/
                        NOP()；
                }

                TMIF80 = 0；                    /* 清除 INTTM80 中断请求标志。 */
                fn_subinttm80()；               /* INTTM80 中断服务 */
        }

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

        if (!P4.3){                     /* 如果 SW 闭合 10ms 或更长时间执行处理。*/
                g_ucSWcnt = (g_ucSWcnt + 1) & 0b00000011；  /* 开关输入次数加 1。 */

                TCE80 = 0；                     /* 停止定时器操作。 */
                CR80 = g_ucCR80data[g_ucSWcnt]；    /* 改变 LED 闪烁时间基准，其依照开关输入次数的变化而改变。*/
                TCE80 = 1；                     /* 启动定时器操作。 */

                g_ucTM80cnt = 0；               /* 清除 INTTM80 中断次数。 */
        }

        return；
}

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

        中断 INTTM80

***************************************************************************/
__interrupt void fn_inttm80(){

        fn_subinttm80()；               /* INTTM80 中断服务 */

        return；
}

/*--------------------------------------------------------------------------
        测量 INTTM80 中断次数的子程序
--------------------------------------------------------------------------*/
void fn_subinttm80(){

        if (++g_ucTM80cnt == 250){      /* 当 INTTM80 中断次数为 250 时处理。 */
                g_ucTM80cnt = 0；       /* 清除 INTTM80 中断次数。 */
                P2 ^= 0b00000001；      /* LED 输出反转。 */
        }

        return；
}
```

● op.asm (汇编语言和 C 语言版本通用)

```
; ===========================================================================
;
;       选项字节
;
; ===========================================================================
OPBT            CSEG   AT       0080H
     DB     10011100B         ;  选项字节区域
;              ||||
;              |||+----------- 内部低速振荡器能由软件停止。
;              |++------------ 内部高速振荡器时钟(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)。
;            ||||||||
;            ++++++++----------- 所有块都能写入或擦除。

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