NEC

应用注释

78K0S/Kx1+

示例程序
(软件 UART 半双工通信)

内部高速振荡时钟及校准

本文件描述了示例程序运行概述，如何使用示例程序，及如何用软件控制设置和使用 UART 半双工通信 。本示例程序中，初始设置完成后，执行校准确定波特率。随后，接收 8 单位数据作为接收试验，并作为发送试验进行发送。

目标设备

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： U18916CA1V0AN00 (第一版)
出版日期： 2008 年 03 月 N

- 本文档所刊登的内容有效期截至 2008 年 03 月。将来可能未经预先通知而更改。在实际进行生产设计时，请参阅各产品最新的数据表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应及其他信息。
- 未经本公司事先书面许可，禁止复制或转载本文件中的内容。否则因本文档所登载内容引发的错误，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作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，音频·视频设备，家电，加工机械以及产业用机器人。

  “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、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所开发或制造，或为日本电气电子株式会社（定义如上）开发或制造的产品。

第一章 概述

在本示例程序提供的示例中，利用软件对通信定时的调整及对端口的控制来进行 UART 半双工通信，且与微控制器是否具有 UART 功能无关，即便提供此功能也不使用（此通信在下文中称为“软件 UART”）。软件 UART 用于与不具备串行接口 UART6 的产品（78K0S/KU1+、78K0S/KY1+）进行串口通信，或者为具备串行接口 UART6 的产品（78K0S/KA1+、78K0S/KB1+）增加串口通信信道的个数。

初始设置完成后进行校准，波特率通过接收等效于 8 位的低电平（80H）确定。之后，接收 8 单位数据作为接收试验，并作为发送试验进行发送。另外，在发送和接收时有一个 LED 灯打开。

1.1 初始设置的主要内容

初始设置的主要内容如下。

- 选择高速内部振荡器作为系统时钟源[注]
- 停止看门狗计时器的运行
- 将 V_{LVI}（低压检测电压）设置为 4.3V±0.2V
- 在 V_{DD}（供电电压）大于等于 V_{LVI} 后，当检测到 V_{DD} 小于 V_{LVI} 时，生成内部复位（LVI 复位）信号
- 将 CPU 时钟频率设置为 8MHz
- 设置 I/O 端口

注 通过选项字节进行设置。

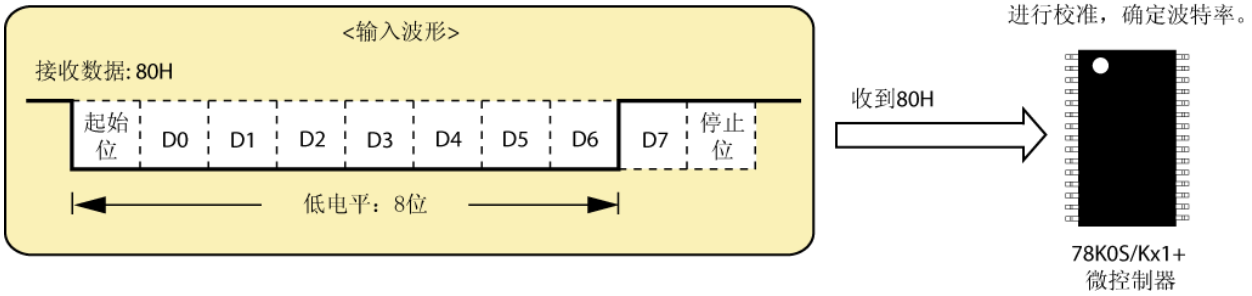
[列] 何为半双工通信？

半双工通信是通信的一种，交替进行接收操作和发送操作。 在本示例程序中使用半双工通信，通过软件启用接收操作和发送操作。

1.2 主循环之后的内容

初始设置完成后进行校准，波特率通过接收等效于 8 位的低电平（80H）确定。

●校准



校准完成后，接收 8 单位数据作为接收试验，并作为发送试验进行发送。另外，在发送和接收时有一个 LED 灯打开。

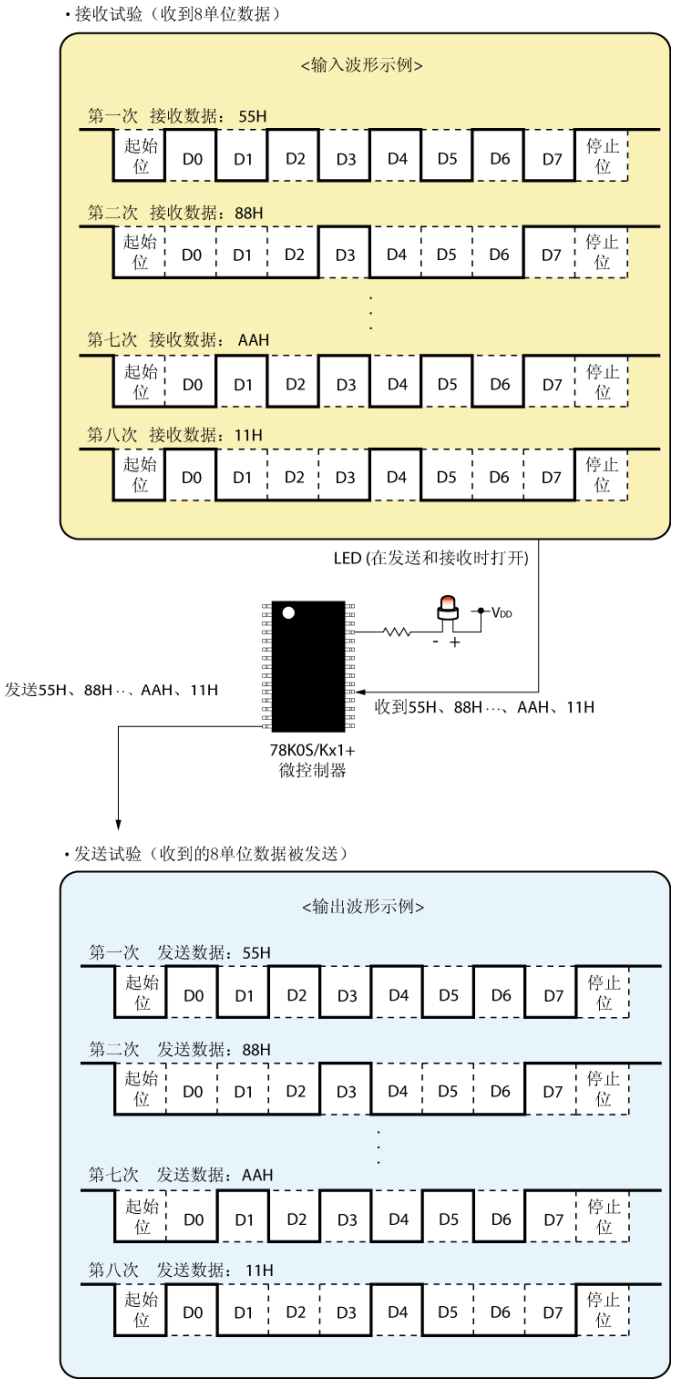
通信协议将设置如下。

- 波特率：4800 至 19200bps 注 1
- 数据字符长度：8 位
- 奇偶校验设定：无校验
- 停止位个数：1 位或 2 位注 2
- 开始位指定：从最低有效位（LSB）开始

注 1 波特率值通过校准确定。在不进行校准时，波特率值可由软件设置。缺省值为 9600bps。

2. 可用软件设置。默认设置为 1 位。

●接收试验→发送试验



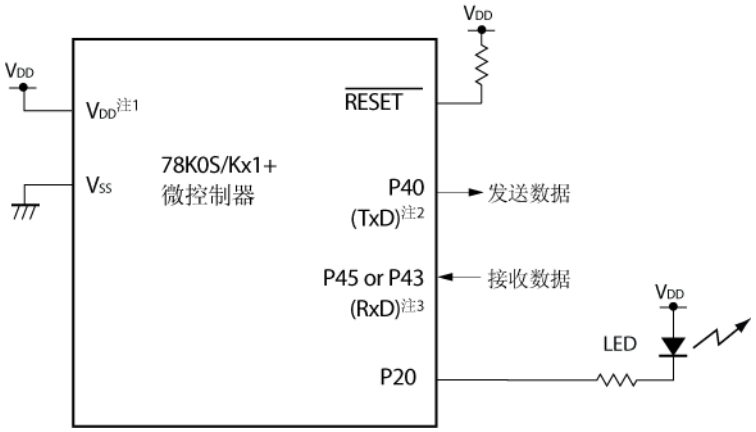
注意事项 关于使用设备的注意事项，请参见各产品（78K0S/KU1+、78K0S/KY1+、78K0S/KA1+、78K0S/KB1+）的用户手册。

第二章 电路图

本章描述本示例程序中所用的电路图和外围硬件。

2.1 电路图

电路图如下所示。



注 **1.** 工作电压范围为 4.5V≤V_{DD}≤5.5V。

2. P40 引脚用于 UART 发送引脚。

3. P45 引脚（78K0S/KA1+和 78K0S/KB1+微控制器）或 P43 引脚（78K0S/KY1+和 78K0S/KU1+微控制器）用作 UART 接收引脚。

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

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

3. 除电路图中所示引脚及 AV_{REF}、AV_{SS} 外，其他所有不用的引脚保留开路（不连接）。

2.2 外围硬件

所用外围硬件如下所示。

●LED

在发送和接收数据时有一个 LED 灯打开。

第三章 软件

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

3.1 文件配置

下表展示所下载的压缩文件的文件配置。

文件名	描述	包含压缩（*.zip）文件		
				
main.asm[注1]	用于微控制器的硬件初始化处理及主处理的源文件	●	●	
op.asm	用于设置选项字节（设置系统时钟源）的汇编器源文件	●	●	
softuart.prw	用于集成开发环境PM+的工作区文件		●	
softuart.prj	用于集成开发环境PM+的项目文件		●	
softuart.pri softuart.prs softuart.prm	用于适用78K0S/Kx1+的系统仿真器SM+的项目文件		●[注2]	
softuart0.pnl	适用78K0S/Kx1+的系统仿真器SM+的I/O面板文件（用于检查外围硬件的工作）		●[注2]	●
softuart0.wvi	用于适用78K0S/Kx1+的系统仿真器SM+的信号数据编辑器文件（用于输入外部信号波形）		●[注2]	●
softuart0.wvo	适用78K0S/Kx1+的系统仿真器SM+的时间图文件（用于检查波形）			●

注 **1.** 软件 UART 示例程序仅有汇编语言版本。

2. 这些文件不包含在 78K0S/KU1+微控制器的文件中。

备注  ： 仅包含源文件。

 ：包含与集成开发环境 PM+和 78K0S/Kx1+系统仿真器 SM+一起使用的文件。

：包含与适用 78K0S/Kx1+的系统仿真器 SM+一起使用的微控制器运行仿真文件。

3.2 所用内部外围功能

本示例程序使用微控制器的下列内部外围功能。

- V_{DD}<V_{LVI}检测： 低压检测器（LVI）
- UART 接收（RxD）： P45 或 P43 注
- UART 发送（TxD）： P40
- LED 输出： P20

注 P45：78K0S/KA1+和 78K0S/KB1+微控制器
P43：78K0S/KY1+和 78K0S/KU1+微控制器

3.3 初始设置及运行概览

本示例程序进行的初始设置包括：低压检测功能的设置、时钟频率的选择、I/O 端口的设置、默认波特率值的设置。

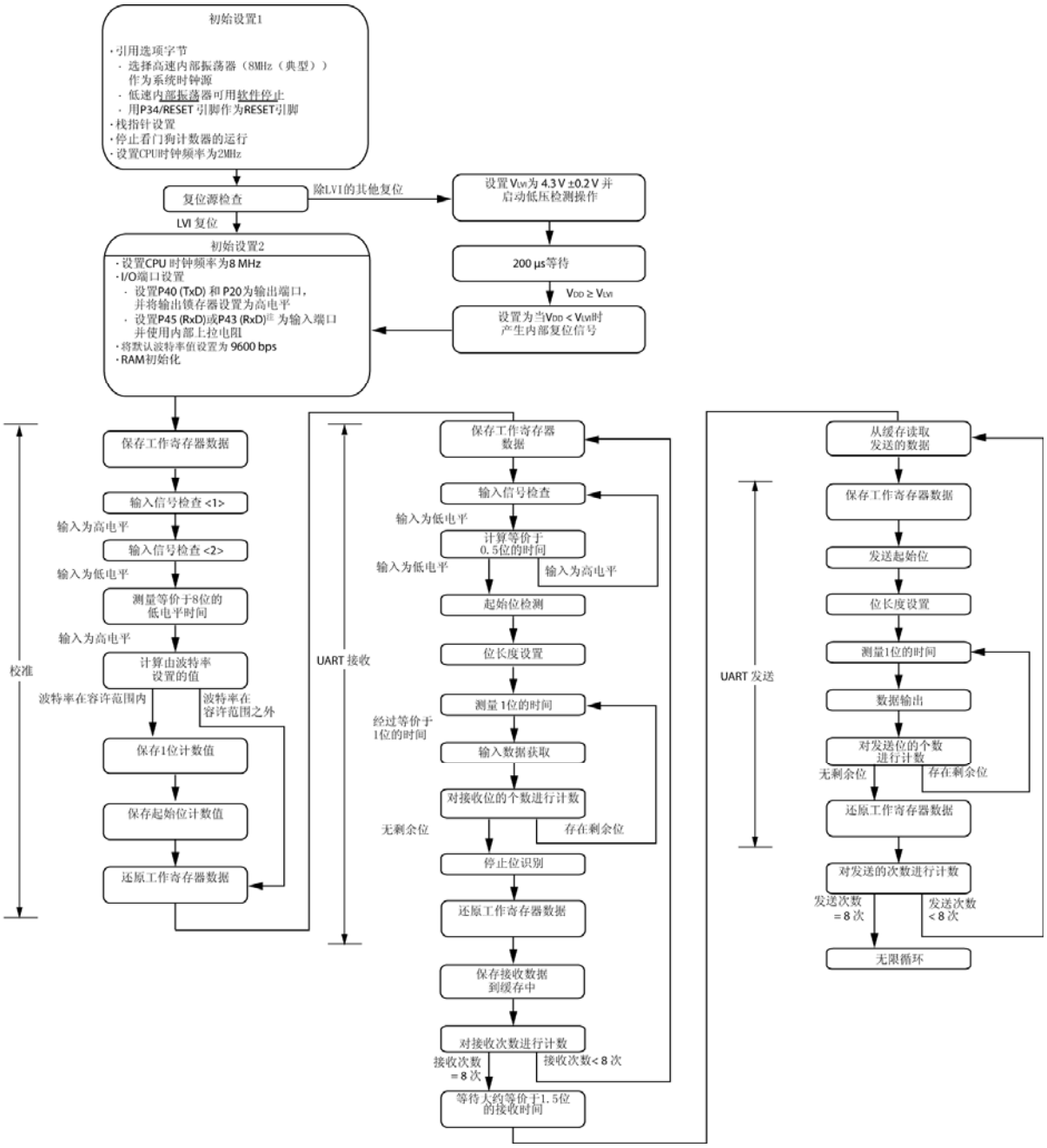
初始设置完成后进行校准，波特率通过接收等效于 8 位的低电平（80H）确定。之后，接收 8 单位数据作为接收试验，并作为发送试验进行发送。另外，在发送和接收时有一个 LED 灯打开。

通信协议将设置如下。

- 波特率：4800 至 19200bps 注 1
- 数据字符长度：8 位
- 奇偶校验设定：无校验
- 停止位个数：1 位或 2 位注 2
- 开始位指定：从最低有效位（LSB）开始

注 **1.** 波特率值通过校准确定。在不进行校准时，波特率值可由软件设置。缺省值为 9600bps。
2. 可用软件设置。默认设置为 1 位。

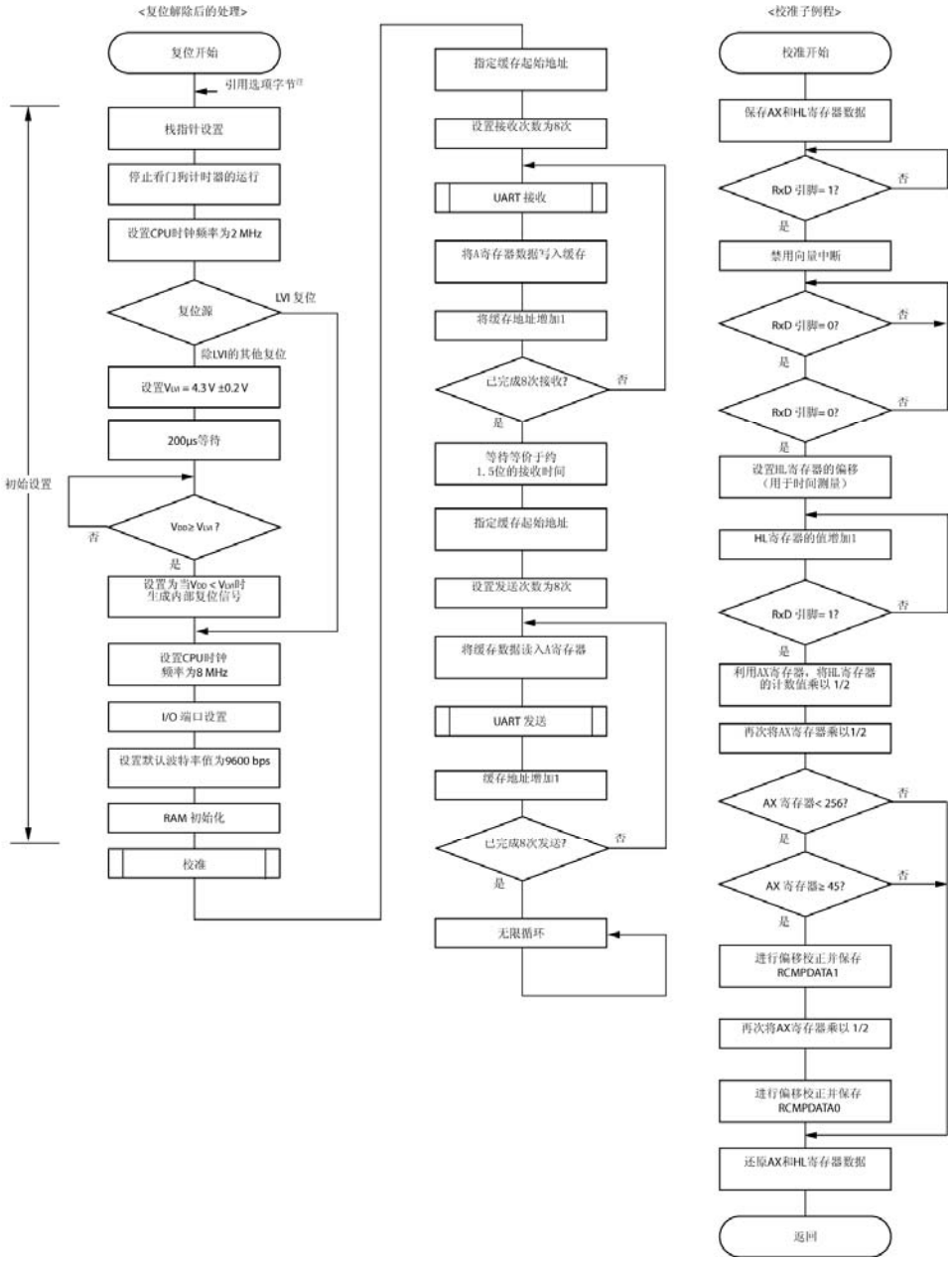
详情在如下所示的状态转换图中描述。



注 P45：78K0S/KA1+和78K0S/KB1+微控制器
P43：78K0S/KY1+和78K0S/KU1+微控制器

3.4 流程图

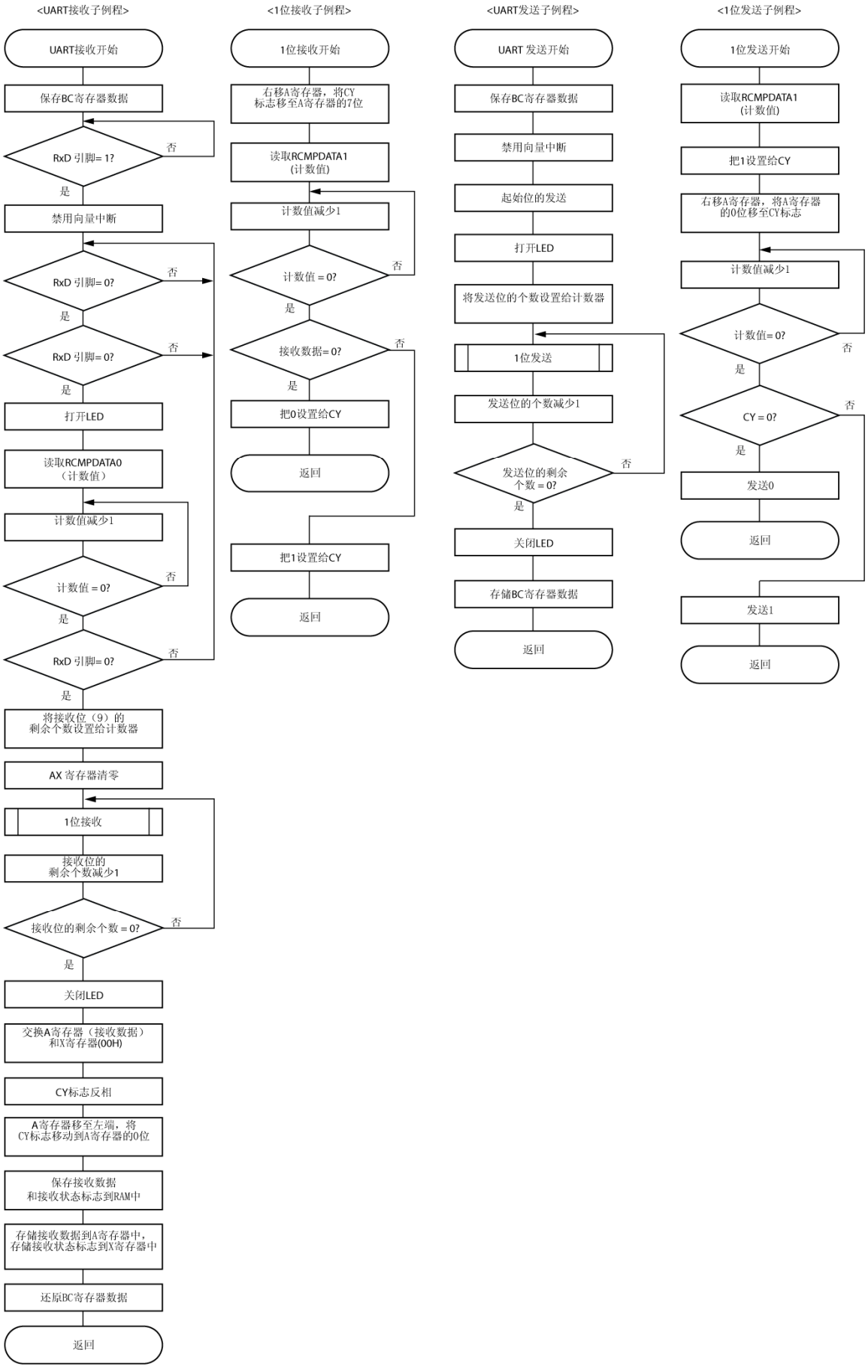
示例程序的流程图如下所示。



注 对选项字节的引用由微控制器在复位解除后自动进行。在本示例程序中，下面的内容通过引用选项字节进行设置。

- 用高速内部时钟（8MHz（典型））作为系统时钟源
- 低速内部振荡器可用软件停止
- 用 P34/$\overline{\text{RESET}}$ 引脚作为 $\overline{\text{RESET}}$ 引脚

备注 <UART 接收子例程><1 位接收子例程><UART 发送子例程><1 位发送子例程>的流程图如下页所示。


<UART接收子例程>
UART接收开始
保存BC寄存器数据
RxD 引脚= 1?
否
是
禁用向量中断
RxD 引脚= 0?
否
是
RxD 引脚= 0?
否
是
打开LED
读取RCMPDATA0
（计数值）
计数值减少1
计数值 = 0?
否
是
RxD 引脚= 0?
否
是
将接收位（9）的
剩余个数设置给计数器
AX 寄存器清零
1位接收
接收位的
剩余个数减少1
接收位的剩余个数 = 0?
否
是
关闭LED
交换A寄存器（接收数据）
和X寄存器(00H)
CY标志反相
A寄存器移至左端，将
CY标志移动到A寄存器的0位
保存接收数据
和接收状态标志到RAM中
存储接收数据到A寄存器中，
存储接收状态标志到X寄存器中
还原BC寄存器数据
返回
<1位接收子例程>
1位接收开始
右移A寄存器，将CY
标志移至A寄存器的7位
读取RCMPDATA1
(计数值)
计数值减少1
计数值 = 0?
否
是
接收数据= 0?
否
是
把0设置给CY
返回
把1设置给CY
返回
<UART发送子例程>
UART 发送开始
保存BC寄存器数据
禁用向量中断
起始位的发送
打开LED
将发送位的个数设置给计数器
1位发送
发送位的个数减少1
发送位的剩余
个数 = 0?
否
是
关闭LED
存储BC寄存器数据
返回
<1位发送子例程>
1位发送开始
读取RCMPDATA1
(计数值)
把1设置给CY
右移A寄存器，将A寄存器
的0位移至CY标志
计数值减少1
计数值= 0?
否
是
CY = 0?
否
是
发送0
返回
发送1
返回

第四章　设置方法

本章描述软件 UART 的初始设置、如何使用校准、UART 接收和 UART 发送子例程，以及子例程的操作概述。

关于其他初始设置，请参见**78K0S/Kx1+示例程序（初始设置）LED发光开关控制的应用注释**。关于低压检测（LVI），请参见**78K0S/Kx1+示例程序（低压检测）检测到小于 2.7V过程中的复位生成的应用注释**。

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

关于汇编器指令，请参见**78K/0S系列指令用户手册**。

4.1　软件UART的初始设置

在使用软件 UART 通信时，应进行下列三项初始设置。

- 软件 UART 通信所使用的端口
- 默认波特率值
- 停止位的个数

4.1.1　端口设置

在本示例程序中，软件 UART 通信所使用的引脚设置如下。

- UART 接收（RxD）：　P45（78K0S/KA1+和 78K0S/KB1+微控制器）
  P43（78K0S/KY1+和 78K0S/KU1+微控制器）
- UART 发送（TxD）：　P40

在复位解除后的初始设置中，下面三个寄存器的设置如下表所示。

- 端口寄存器：P4
- 端口模式寄存器：PM4
- 上拉电阻选项寄存器：PU4

	P4 寄存器	PM4 寄存器	PU4 寄存器
78K0S/KA1+和 78K0S/KB1+微控制器	P40=1	PM40=0、PM45=1	PU45=1
78K0S/KY1+和 78K0S/KU1+微控制器	P40=1	PM40=0、PM43=1	PU43=1

在源文件中，常用端口寄存器的符号定义如下。

[本示例程序源文件节选（78K0S/KA1+和 78K0S/KB1+微控制器）]

```
PTXD            EQU     P4.0            ; 用于 UART 发送的引脚(TxD 引脚)
PRXD            EQU     P4.5            ; 用于 UART 接收的引脚(RxD 引脚)
```

在软件 UART 通信中，通用 I/O 端口用作为 UART 接收和发送引脚。因此，通过更改这些端口的设置就能够启用指定端口的通信。

4.1.2 通信协议设置

通信协议如下。

- 波特率：4800 至 19200bps（默认为 9600bps）
- 数据字符长度：8 位
- 奇偶校验设定：无校验
- 停止位个数：1 位或 2 位（默认 1 位）
- 开始位指定：从最低有效位（LSB）开始

波特率及停止位个数可通过软件设置。
波特率也可通过校准确定。

（1）设置波特率

下面的两个 RAM 数据用于设置波特率。

- RCMPDATA1：用于 1 位计数
- RCMPDATA0：用于起始位计数

在本示例程序中，这些 RAM 数据的默认值通过初始设置进行如下设置。

[本示例程序源文件节选]

```
MOV     RCMPDATA1, #CB9600  ; 1 位计数定时器默认值(9600 bps)
MOV     RCMPDATA0, #CHB9600 ; 起始位计数定时器默认值 (9600 bps)
```

下面的三个常量通过对符号的定义对波特率进行设置。

波特率	符号	
	RCMPDATA1	RCMPDATA0
4800bps	CB4800	CHB4800
9600bps（默认）	CB9600（默认）	CHB9600（默认）
19200bps	CB19200	CHB19200

当不进行校准时，或在校准过程中发现超出推荐的波特率范围时，将应用通过初始设置设置的 RAM 数据。

（2）设置停止位的个数

在本示例程序中，通过对符号的定义，停止位的个数默认设置为 1。

[本示例程序源文件节选]

```
CSTOPBIT      EQU    1              ; 指定停止位的个数
```

发送过程中的停止位的个数可设置为 2，只要将如上所示的“1”改为“2”即可。
软件 UART 在接收过程中操作时，停止位的个数总是设置为“1”。

4.2 校准

4.2.1 如何使用校准

在本示例程序中，校准处理变为一个子例程。通过如下调用可进行校准。

[调用校准子例程的示例]

```
CALL   !SCALIB
```

该子例程可根据需要进行任意多次校准来校正波特率。

下面的两个设置值通过校准处理存储在 RAM 中。这些 RAM 数据随波特率的不同而改变。

- RCMPDATA1： 用于 1 位计数
- RCMPDATA0： 用于起始位计数

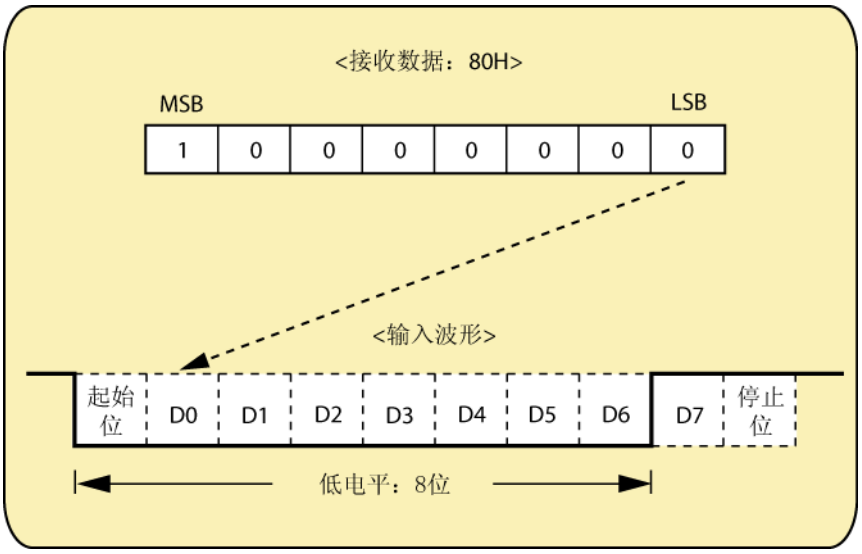
在校准过程中，当波特率明显处于推荐范围之外时，这些 RAM 数据将不会更新，而是保持最近的波特率数据。（在本示例程序中，这些值为初始设置值（当波特率为 9600bps 时的值）。）详情请见“**4.2.2 校准操作概述**”。

4.2.2 校准操作概述

进行校准及确定波特率的方法是，在作为串口通信输入端的引脚（RxD）处接收相当于 8 位的低电平（接收数据：80H）。

在本示例程序中，初始设置完成后立即进行校准。

用于校准检测的 80H 接收数据的波形如下所示。等效于 8 位的低电平的宽度为从字符位的起始位到第 6 位。



在开始测量低电平宽度前，应禁用中断肯定应答，并设置为测量过程中不出现中断服务。

在开始测量低电平宽度时，应两次检查 RxD 引脚处于低电平，从而消除小于约 1.5μs 的噪声。检查后，先将从指令的执行时钟个数中获得的正确值设置给 HL 寄存器，再启动用于测量的 HL 寄存器的计数，从而获得适合通信速度的处理时间。

根据指令的执行时钟的个数，每 16 个时钟后将用于时间测量的 HL 寄存器增加 1。

当检查到 RxD 引脚位于高电平时，对 HL 寄存器的计数终止。

[本示例程序源文件节选（校准）]

```
JCAL000:
        BF      PRXD,  $JCAL000      ;  如果 RxD 引脚为 0，等待直至变为 1
        DI                           ;  启用向量中断
JCAL100:
        BT      PRXD,  $JCAL100      ;10:  等待校准开始
        BT      PRXD,  $JCAL100      ;10:  如果有噪声，返回对校准开始的等待
        NOP                          ;  2:  用于时间调整
        NOP                          ;  2:  用于时间调整
        MOVW    HL,    #CCALOFFSET   ;  6:  时间校正
JCAL200:
        NOP                          ;  2:
        INCW    HL                   ;  4:  时间测量
        BF      PRXD,  $JCAL200      ;10:  等待RxD引脚变为 1
```

检测低电平两次

先禁用中断再启动测量操作

2 + 4 + 10 = 16 时钟

设置校正值

因为 NOP 至 BF 指令将重复至检测到高电平，所以每 16 个时钟将 HL 寄存器增加 1。

HL 寄存器增加 1

等待高电平检测

HI 寄存器计数值与波特率值之间的关系如下所示。

所用时钟频率为 8MHz；因此，8 MHz = 8 × 10^6 Hz。

$$\text{HL 寄存器计数值} = \underbrace{\frac{1}{\text{波特率值}}}_{\text{(a)}} \times \underbrace{8 \times 10^6}_{\text{(b)}} \times \underbrace{8 \times \frac{1}{16}}_{\text{(c)}}$$

根据指令执行时钟的个数，HL 寄存器每 16 个时钟都会递增；因此“等价于 8 位的时钟的个数 × 1/16”成为计数值。

(a)：等价于 1 位的时间
(b)：等价于 1 位的时钟个数
(c)：等价于 8 位的时钟个数

$$\rightarrow \text{波特率值} = \frac{4 \times 10^6}{\text{HL寄存器计数值}}$$

[示例] 当HL寄存器计数值为416时，

$$\text{波特率值} = \frac{4 \times 10^6}{416} \fallingdotseq 9615[\text{bps}]$$

通过 HL 寄存器的计数值的四分之一来确定波特率是否在推荐范围内。此时，根据确认的计数值，波特率的容许范围如下。

$$45 \leq \frac{\text{HL寄存器计数值}}{4} < 256$$

→波特率的容许范围：3907至22222[bps]

（波特率的推荐范围：4800至19200[bps]）

当波特率在容许范围之内时，应确定从 HL 寄存器计数值保存到 RCMPDATA1 和 RCMPDATA0 的值。

备注 当波特率为 9600bps 时，该值默认设置给 RCMPDATA1 和 RCMPDATA0。在不进行校准或者校准结果为波特率在容许范围之外时，默认值将用于 UART 发送和接收。

这是因为在对 HL 寄存器进行计数时所用指令的执行时钟的个数为 16（每 16 个时钟递增 1，对 8 位进行测量）。

<1> 1位等价计数值 = HL寄存器计数值×16×$\frac{1}{8}$ × $\frac{1}{8}$−校正值（RCMPDATA1）

(a)

(b)

对 1 位进行计数的处理时间的校正

=HL寄存器值×$\frac{1}{4}$−校正值

这是因为在对 RCMPDATA1 进行计数时，所用指令的执行时钟个数为 8.（每 8 个时钟，计数值递减 1。）

(a)： 等价于 8 位的时钟个数
(b)： 等价于 1 位的时钟个数

备注 1 位等价计数值（RCMPDATA1）用于在 UART 接收过程中确定 1 位数据的确认定时，以及用于在 UART 发送过程中确定 1 位数据的长度。

<2> 用于起始位检测的计数值（RCMPDATA0）

这是因为在对 HL 寄存器进行计数时所用指令的执行时钟的个数为 16（每 16 个时钟递增 1，对 8 位进行测量）。

对起始位检测的处理时间的校正

= HL寄存器值×16×$\frac{1}{8}$×$\frac{1}{2}$×$\frac{1}{8}$−校正值=HL寄存器值×$\frac{1}{8}$−校正值

(b)

(c)

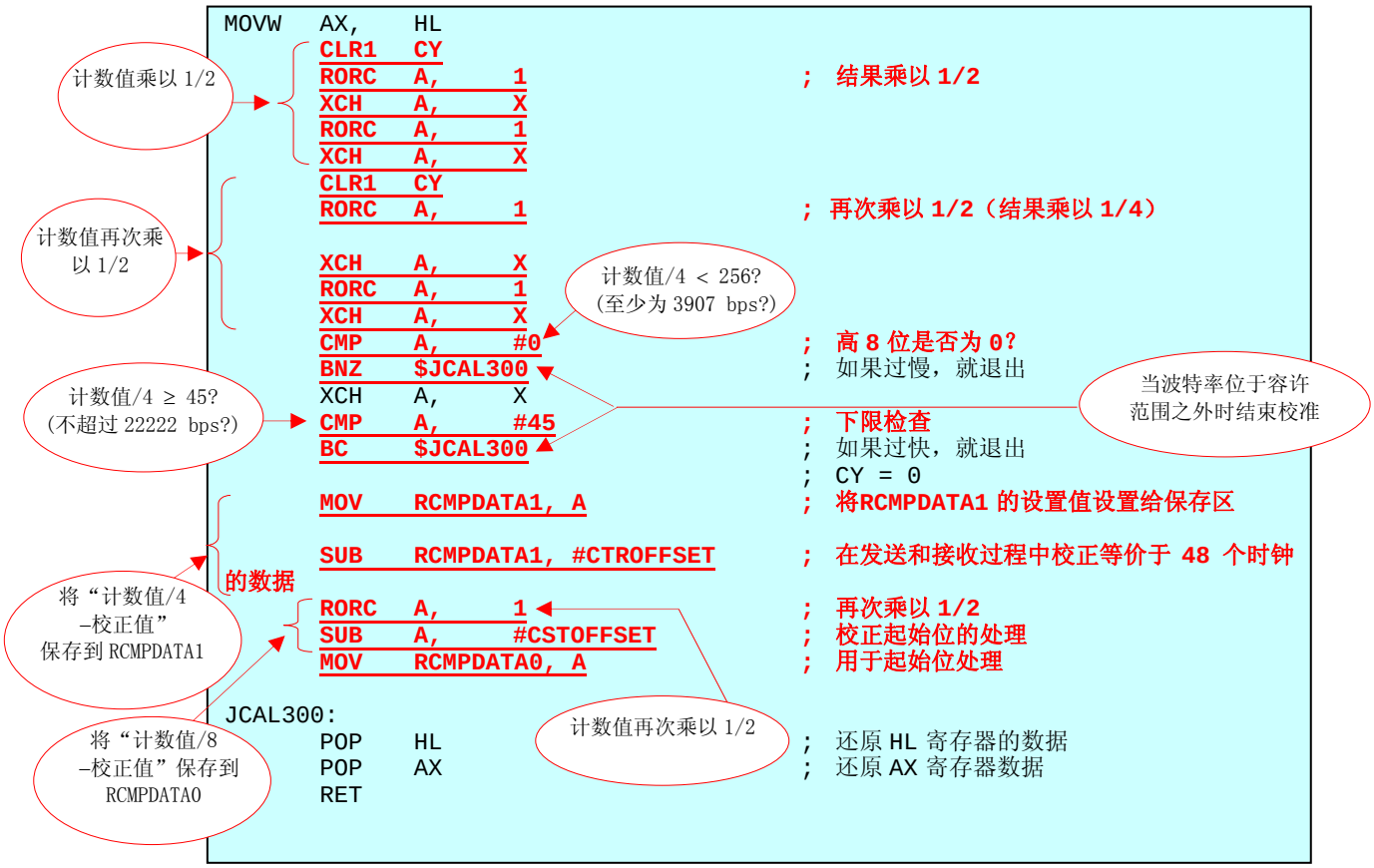
这是因为在对 RCMPDATA0 进行计数时，所用指令的执行时钟个数为 8.（每 8 个时钟，计数值递减 1。）

(a)： 等价于 8 位的时钟个数
(b)： 等价于 1 位的时钟个数
(c)： 等价于 0.5 位的时钟个数

备注

1. 用于起始位检测的计数值（RCMPDATA0）用来在 UART 接收过程中检测起始位。因为 UART 接收时在 1 位数据的中间识别 0 或 1，所以在确定 RCMPDATA0 时，应使得在 UART 接收引脚（RxD）检测到低电平后 0.5 位时间点处对起始位进行检测。
2. RCMPDATA1 和 RCMPDATA0 二者的校正值是不同的值。

[本示例程序源文件节选（从确认校准结果到保存计数值）]



4.3 UART接收

4.3.1 如何使用UART接收

在本示例程序中，UART 接收处理变为一个子例程。如下所示，利用 UART 接收处理，可进行数据接收处理及保存接收数据和接收状态标志。

[调用 UART 接收子例程及保存数据示例]

```
CALLT   [ZRXDATA]          ; UART 接收子例程调用
MOV     RDATA, A           ; 将接收数据保存到 RDATA
XCH     A,     X           ; 接收状态标志转移到 A 寄存器
MOV     SDATA, A           ; 将接收状态标志保存到 SDATA
```

通过该 UART 接收子例程的调用，接收数据和接收状态标志存入通用寄存器和 RAM 区二者当中，从子例程返回处理。接收数据和接收状态标志、通用寄存器和 RAM 区的对应关系如下所示。

	要存入的通用寄存器	要存入的RAM地址
接收数据	A寄存器	RRXDATA（RRXDATA的低位1字节）
接收状态标志	X寄存器	RRXFLAG（RRXDATA的高1字节）

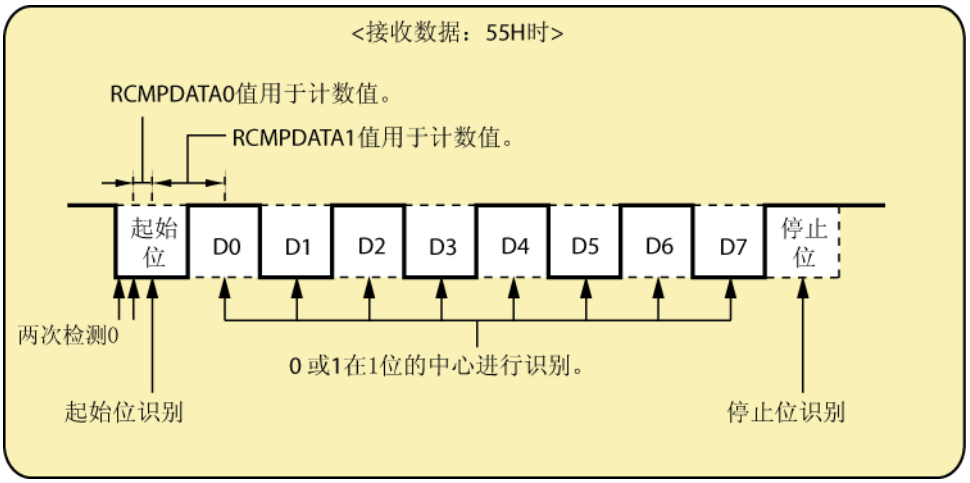
接收状态标志只能识别帧错误（不检测停止位）。

- 接收状态标志=00H：正常接收
- 接收状态标志=01H：帧错误

4.3.2 UART接收的操作概览

在开始数据接收前，应禁用中断肯定应答并设置为在接收过程中不出现中断服务。

在开始数据接收时，应两次检查 UART 接收引脚（RxD 引脚）处于低电平，从而消除小于约 1.5μs 的噪声。检查后，在该位的中间识别起始位。



识别定时通过RCMPDATA0值确定，在识别 0 位之后的各位时利用RCMPDATA1值（等价于 1 位的计数值）确定，从而在各位的中央确认各位的定时；RCMPDATA0值（等价于 0.5 位的计数值）在识别起始位时由校准进行确定。

备注 当波特率为 9600bps 时，该值默认设置给 RCMPDATA1 和 RCMPDATA0。在不进行校准或者校准结果为波特率在容许范围之外时，默认值将用于 UART 发送和接收。

在接收数据时，A 寄存器和 CY 标志用于接收缓冲寄存器。在每次接收 1 位时将接收数据的相同值设置给 CY 标志，并将 A 寄存器右移；这样 CY 标志数据（接收位）将移动到 A 寄存器的 7 位。

在 8 位数据接收结束后，接收数据保存到 A 寄存器和 RRXDATA（RAM 区）中。另外，CY 标志的值反相并保存到 X 寄存器和 RRXFLAG（RAM 区）中作为接收状态标志（0：正常接收；1：帧错误）。

[本示例程序源文件节选（UART 接收<1>）]

```
XCALT  CSEG   CALLT0
ZRXDATA:      DW      SRXDATA         ; UART 接收子例程
                      •
                      •
                      •
MMAINLOOP:
                      •
                      •
                      •
       MOVW   HL,     #RRXDATABUFF    ; 指定缓存起始地址
       MOV    B,      #8              ; 指定接收次数
LMLP100:
       CALLT  [ZRXDATA]               ; 接收数据
       MOV    [HL],   A               ; 数据写入缓存
       INC    L                       ; 缓存地址增加 1
       DBNZ   B,      $LMLP100        ; 重复 8 次
                      •
                      •
SRXDATA:
       PUSH   BC                      ; 将 BC 寄存器数据保存到栈中
JRXD000:
       BF     PRXD,   $JRXD000        ; 如果 RxD 引脚为 0，等待直至变为 1
       DI                             ; 启用向量中断
JRXD100:
       BT     PRXD,   $JRXD100        ;10: 等待起始位检测
       BT     PRXD,   $JRXD100        ;10: 如果有噪声，返回对起始位检测的等待

       CLR1   PLED                    ; 6: 打开 LED（在数据接收过程中）
       NOP                            ; 2: 用于时间调整
       MOV    A,      RCMPDATA0       ; 4: 读取设置值
       MOV    RTIMECNT, A             ; 4: 对位中心进行设置
JRXD200:
       DBNZ   RTIMECNT, $JRXD200      ; 8: 等待起始位中心
       BT     PRXD,   $JRXD100        ;10: 如果未检测到起始位，就返回对检测的等待
       NOP                            ; 2: 用于时间调整
       MOV    B,      #8+1            ; 6: 设置接收位的剩余个数
       MOVW   AX,     #0000H          ; 6: 设置初始数据
JRXD300:
       CALL   !SRXBIT                 ; 6: 接收该位
       DBNZ   B,      $JRXD300        ; 6: 对接收位的个数进行计数
       SET1   PLED                    ; 关闭 LED（数据接收结束）
       XCH    A,      X               ; 将接收数据保存到X寄存器中
       NOT1   CY
       ROLC   A,      1               ; 如果未检测到停止位，就设置 0 至 1 位

       MOVW   RRXDATA, AX             ; 保存接收数据及错误状态
       XCH    A,      X               ; 将接收数据存储到A寄存器中
                                      ; 将错误状态存储到 X 寄存器中
       POP    BC                      ; 还原 BC 寄存器的数据
       RET
                      •
                      •
                      •
```

调用 UART 接收子例程并接收数据

将接收数据存储到缓存中

UART 接收子例程

先禁用中断再启动接收操作

检测低电平两次

读取用于起始位检测的设置值

8 时钟 × RCMPDATA0 值

计数 0.5 位（每 8 个时钟递减 1）

设置接收位的个数

起始位识别

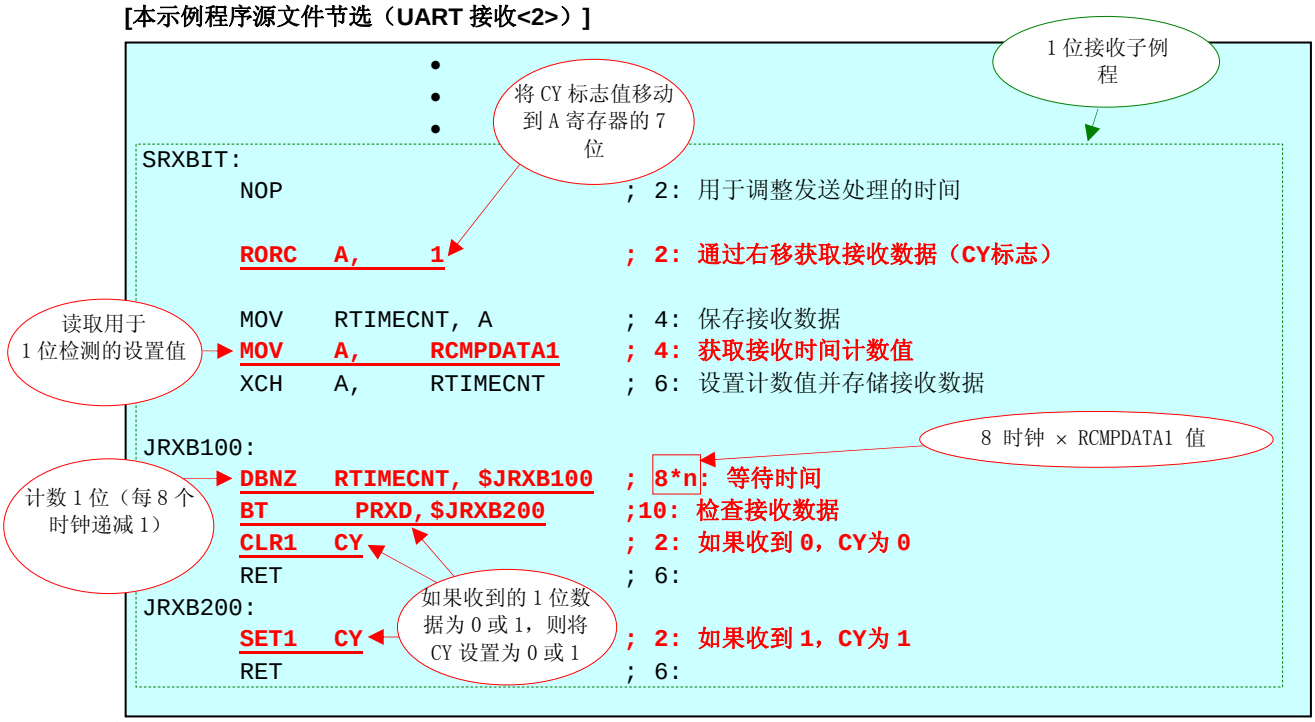
调用 1 位接收子例程并逐位接收

将反相 CY 标志值作为错误状态存储到 X 寄存器中，并将接收数据保存到 A 寄存器中

将接收数据和错误状态保存到 RAM 区中

备注　　1 位接收子例程（SRXBIT）在下页继续描述。

[本示例程序源文件节选（UART 接收<2>）]



备注 该示例程序源文件节选接上页。

4.4 UART发送

4.4.1 如何使用UART发送

在本示例程序中，UART 发送处理变为一个子例程。利用如下所示的 UART 发送处理，可进行数据发送处理。

[调用 UART 发送子例程的示例]

```
MOV    A,     TDATA    ; 将要发送的数据(TDATA)保存到 A 寄存器中
CALLT  [ZTXDATA]       ; 发送数据
```

4.4.2 UART发送的操作概览

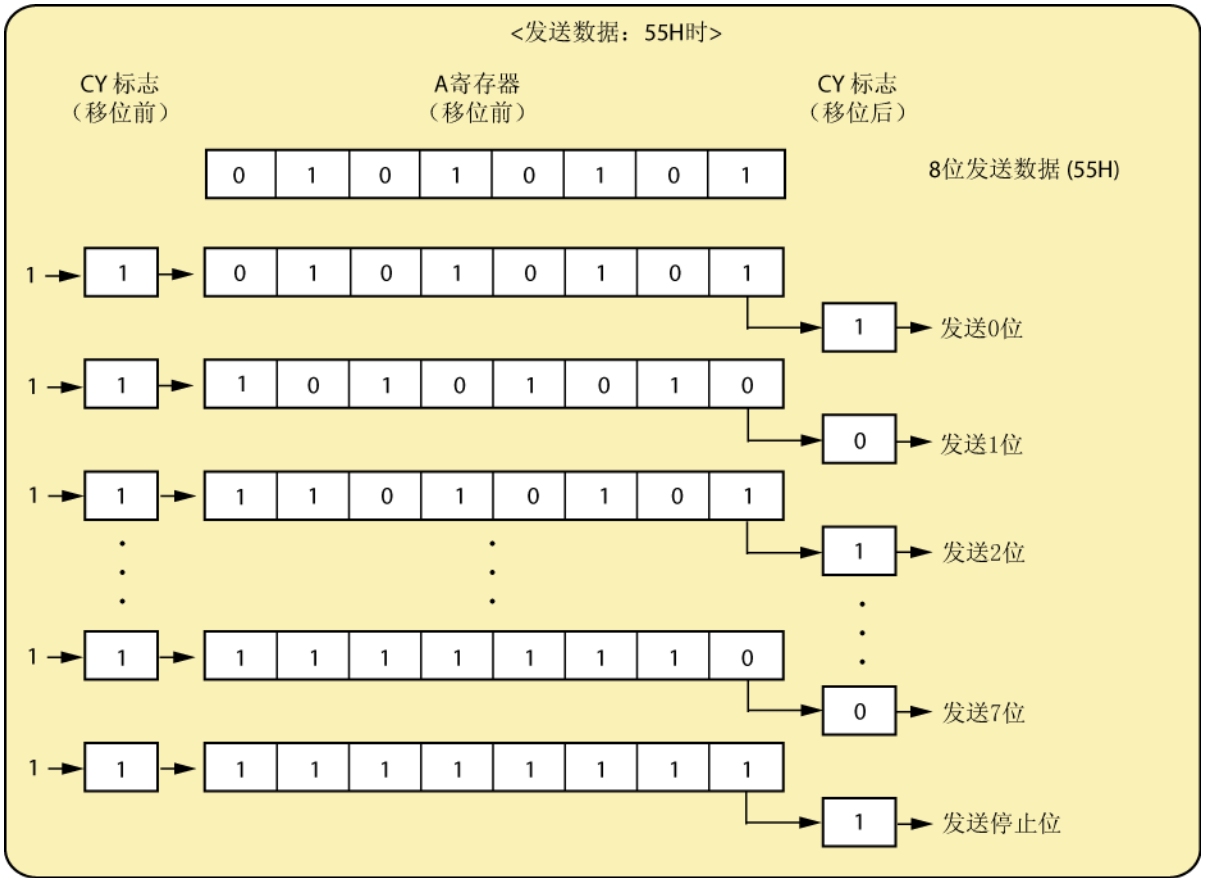
在开始数据发送前，应禁用中断肯定应答并设置为在发送过程中不出现中断服务。

因为发送数据从 A 寄存器中读取，发送数据必须在调用子例程前存储到 A 寄存器中。

在发送数据前，使用通过校准确定的RCMPDATA1值（等价于 1 位的计数值），以确定要发送的 1 位长度。

备注 当波特率为 9600bps 时，该值默认设置给 RCMPDATA1。在不进行校准或者校准结果为波特率在容许范围之外时，默认值将用于 UART 发送和接收。

在发送数据时，A 寄存器和 CY 标志用于发送缓冲寄存器。通过把 1 设置给 CY 标志并右移数据，CY 标志数据将移动到 A 寄存器的 7 位，A 寄存器的 0 位数据（发送位）将移动到 CY 标志，移动后 CY 标志的相同值的各位都将发送。



[本示例程序源文件节选（UART 发送）]

```
XCALT  CSEG   CALLT0
ZTXDATA:      DW      STXDATA         ;  UART 发送子例程
                •
                •
                •
MMAINLOOP:
                •
                •
                •
       MOVW   HL,     #RRXDATABUFF    ;  指定缓存起始地址
       MOV    B,      #8              ;  指定发送次数
LMLP200:
       MOV    A,      [HL]            ;  读取缓存数据
       CALLT  [ZTXDATA]               ;  发送数据
       INC    L                       ;  缓存地址增加 1
       DBNZ   B,      $LMLP200        ;  重复 8 次
                •
                •
                •
STXDATA:
       PUSH   BC                      ;  将 BC 寄存器数据保存到栈中
       DI                             ;  禁用向量中断
       CLR1   PTXD                    ;  6:  发送起始位
       CLR1   PLED                    ;  6:  打开 LED（在数据发送过程中）

       MOV    B,      #1+8+CSTOPBIT   ;  6:  设置发送位的个数
JTXD100:
       CALL   !STXBIT                 ;  6:  发送该位
       DBNZ   B,      $JTXD100        ;  6:  对发送位个数进行计数
       SET1   PLED                    ;  关闭 LED（数据发送结束）
       POP    BC                      ;  还原 BC 寄存器的数据
       RET
                •
                •
                •
STXBIT:
       MOV    RTIMECNT, A             ;  4:  保存发送的数据
       MOV    A,      RCMPDATA1       ;  4:  获取发送时间计数值
       XCH    A,      RTIMECNT        ;  6:  设置计数值并还原发送的数据

       SET1   CY                      ;  2:  输出后将数据设置为 1
       RORC   A,      1               ;  2:  将发送数据右移至CY标志

JTXB100:
       DBNZ   RTIMECNT, $JTXB100      ;  8*n:  等待时间

       BC     $JTXB200                ;  6:  如果CY为 1，就分支执行
       CLR1   PTXD                    ;  6:  发送 0
       RET                            ;  6:
JTXB200:
       SET1   PTXD                    ;  6:  发送 1
       RET                            ;  6:
```

从缓存读取发送数据

调用 UART 发送子例程并发送数据

UART 发送子例程

先禁用中断再启动发送操作

发送起始位

调用 1 位发送子例程并逐位发送

设置发送位的个数

1 位发送子例程

读取用于 1 位发送的设置值

设置 CY 标志为 1

将 A 寄存器的 0 位移动到 CY 标志

计数 1 位（每 8 个时钟递减 1）

8 时钟 × RCMPDATA1 值

如果 CY 为 0，发送 0；如果 CY 为 1，发送 1。

第五章　用系统仿真器SM+进行运行检查

本章描述利用汇编语言文件（源文件+项目文件），示例程序在适用 78K0S/Kx1+的系统仿真器 SM+中如何运行；该汇编语言文件通过选择  图标进行下载。

注意事项　适用 78K0S/Kx1+的系统仿真器 SM+在 78K0S/KU1+微控制器中不支持（至 2007 年 10 月）。因此，78K0S/KU1+微控制器的运行无法由适用 78K0S/Kx1+的系统仿真器 SM+进行检查。

5.1　构建示例程序

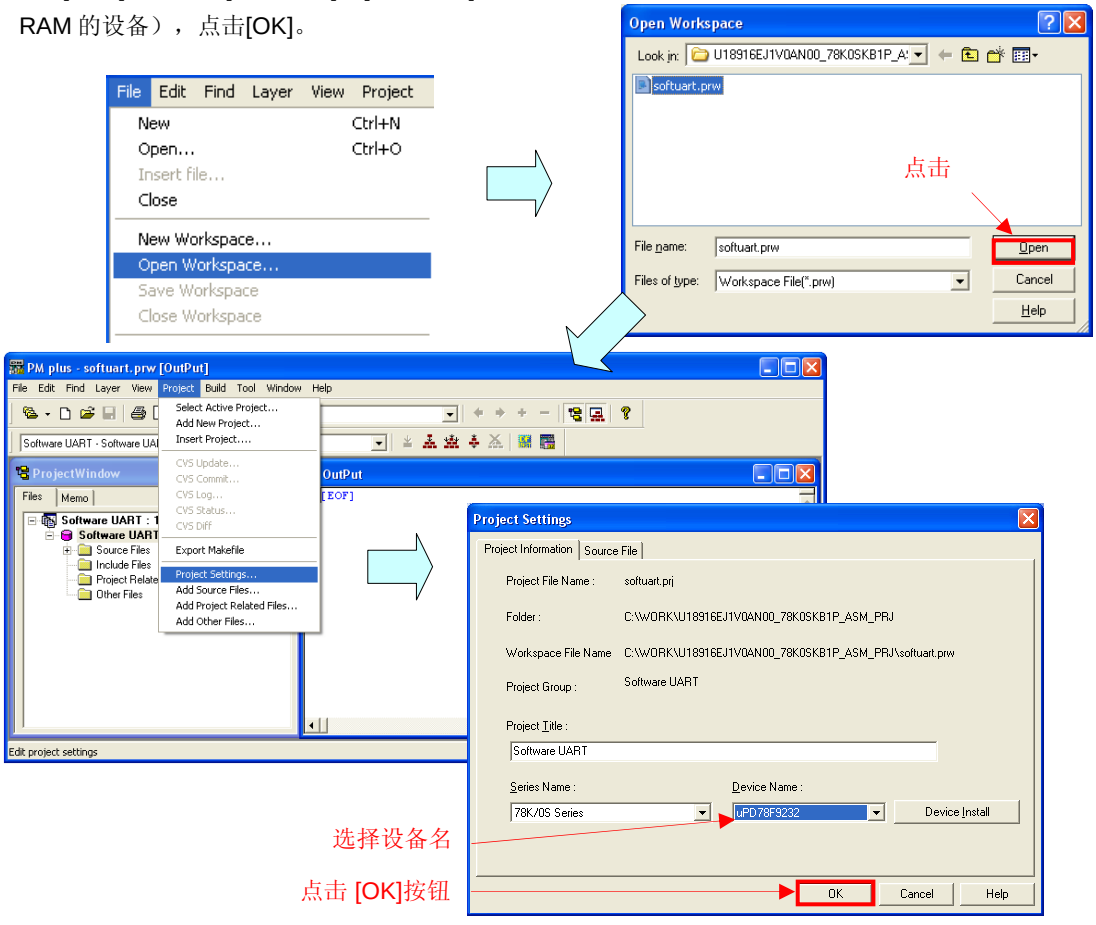
要利用适用 78K0S/Kx1+的系统仿真器SM+（下文称为“SM+”）检查示例程序的运行，在构建示例程序后必须启动SM+。本节描述运行次序的示例，从用集成开发环境PM+构建示例程序到启动SM+；使用  选择下载的汇编语言文件（源文件+项目文件）。关于如何构建其他下载的程序，请参见78K0S/Kx1+示例程序启动向导的应用注释的“**第三章注册集成开发环境PM+项目并执行构建**”。

关于如何操作PM+的详情，请参见PM+项目管理器用户手册。

（1）启动 PM+。

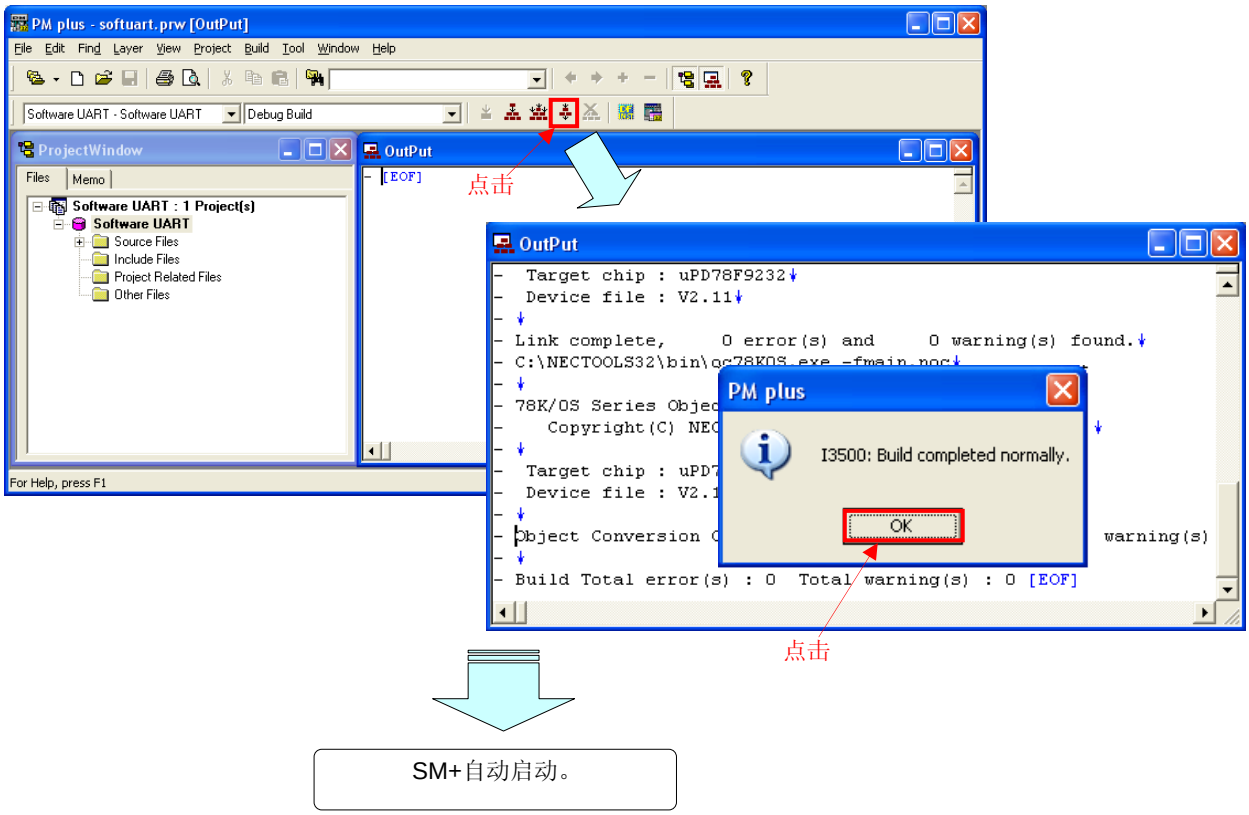
（2）在[文件]菜单点击[打开]再点击[打开工作区]，选择“softuart.prw”。工作区生成，源文件将自动放入该工作区中。

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



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

（5）点击消息窗口中的[OK]按钮，自动启动 SM+。

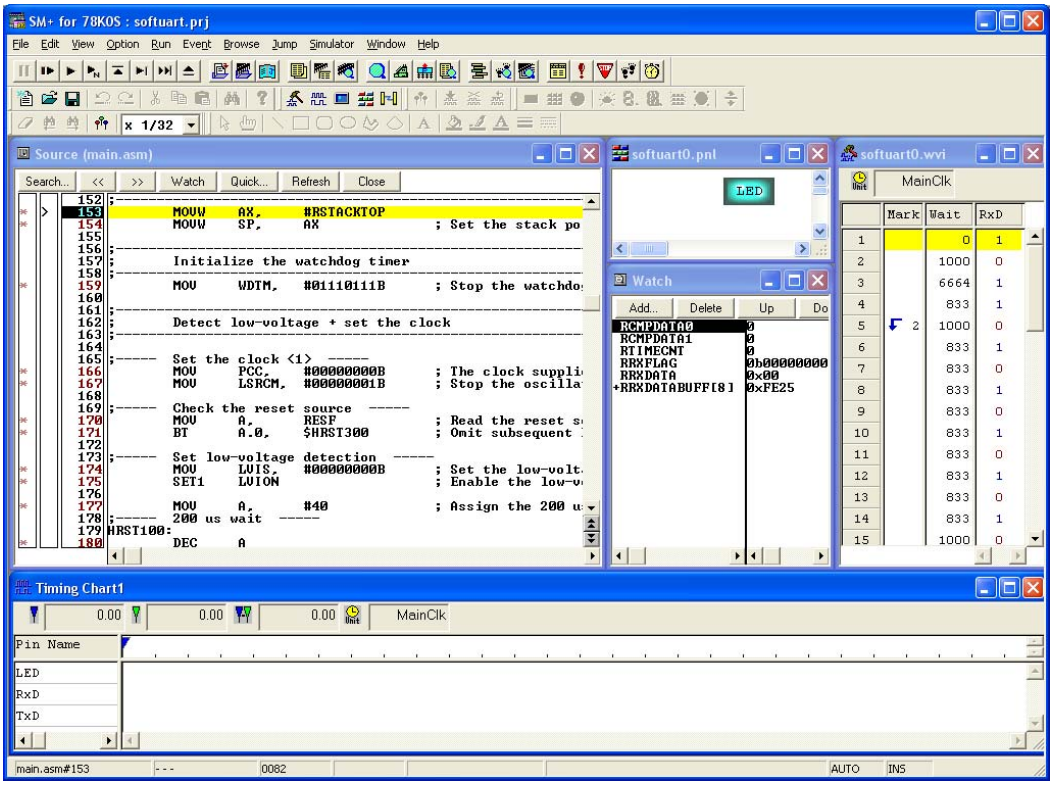


5.2　随SM+运行

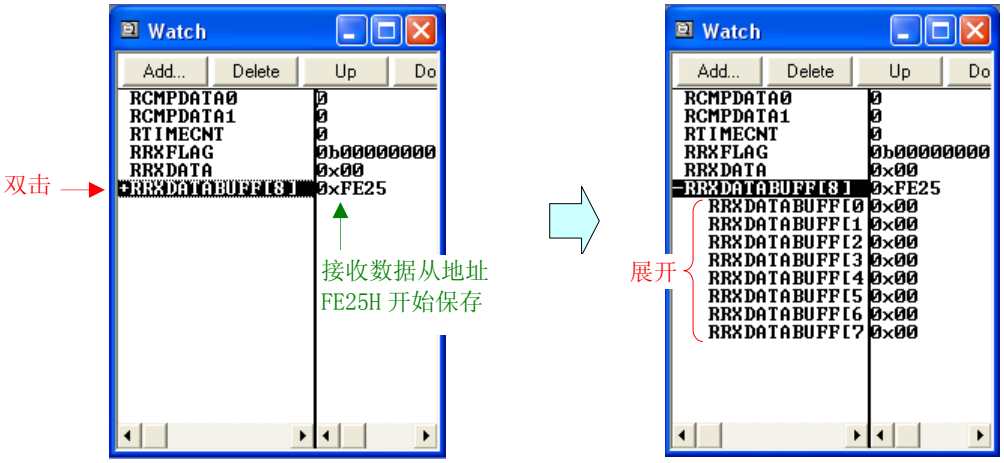
本节描述在 SM+的 I/O 面板窗口或时间图窗口中对运行进行检查的示例。

关于如何操作SM+的详情，请参见**SM+系统仿真器操作用户手册**。

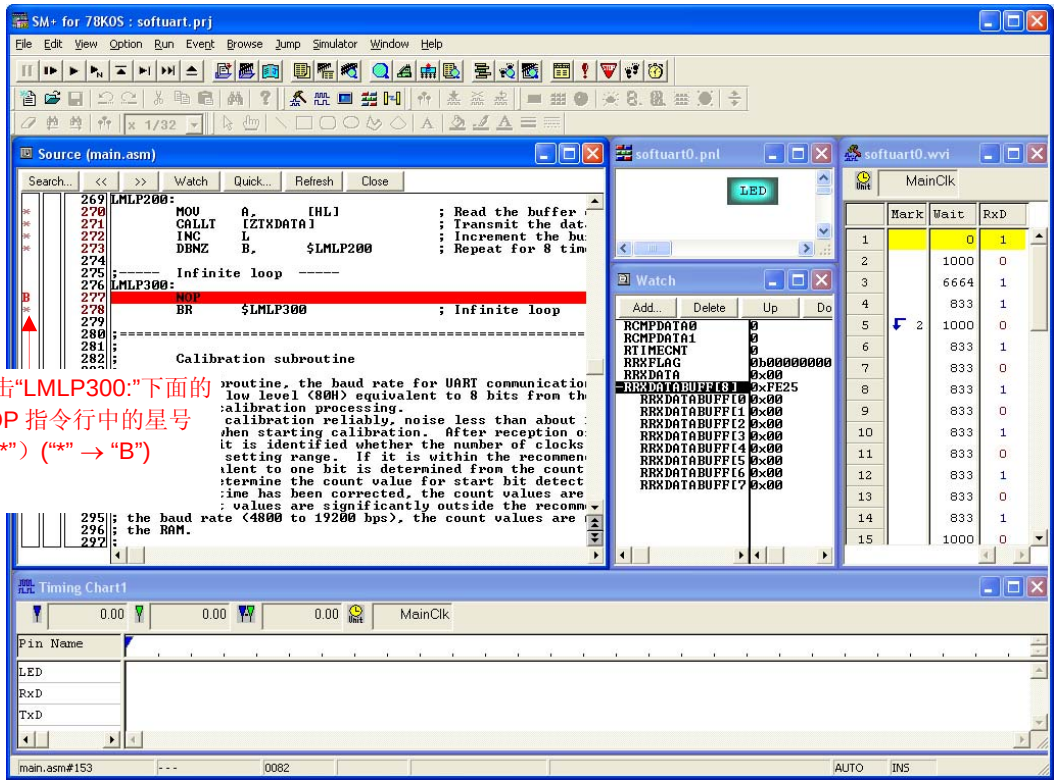
（1）点击PM+中的[构建→调试]启动SM+（参见5.1），屏幕显示如下。



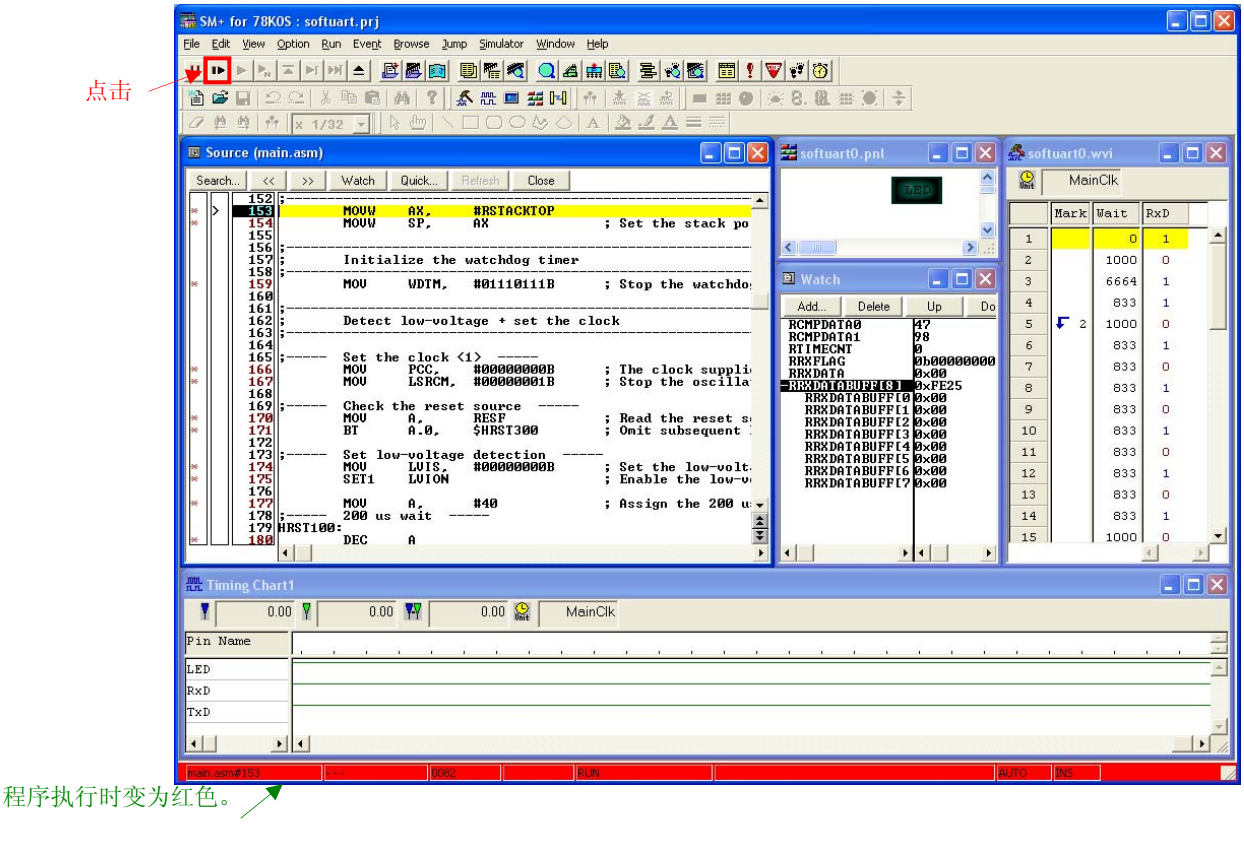
（2）选择“察看”窗口（Watch）并双击“+RRXDATABUFF[8]”，第一个字符从加号（“+”）变为减号（“-”），要保存的接收数据将展开并显示在“–RRXDATABUFF[8]”下方。



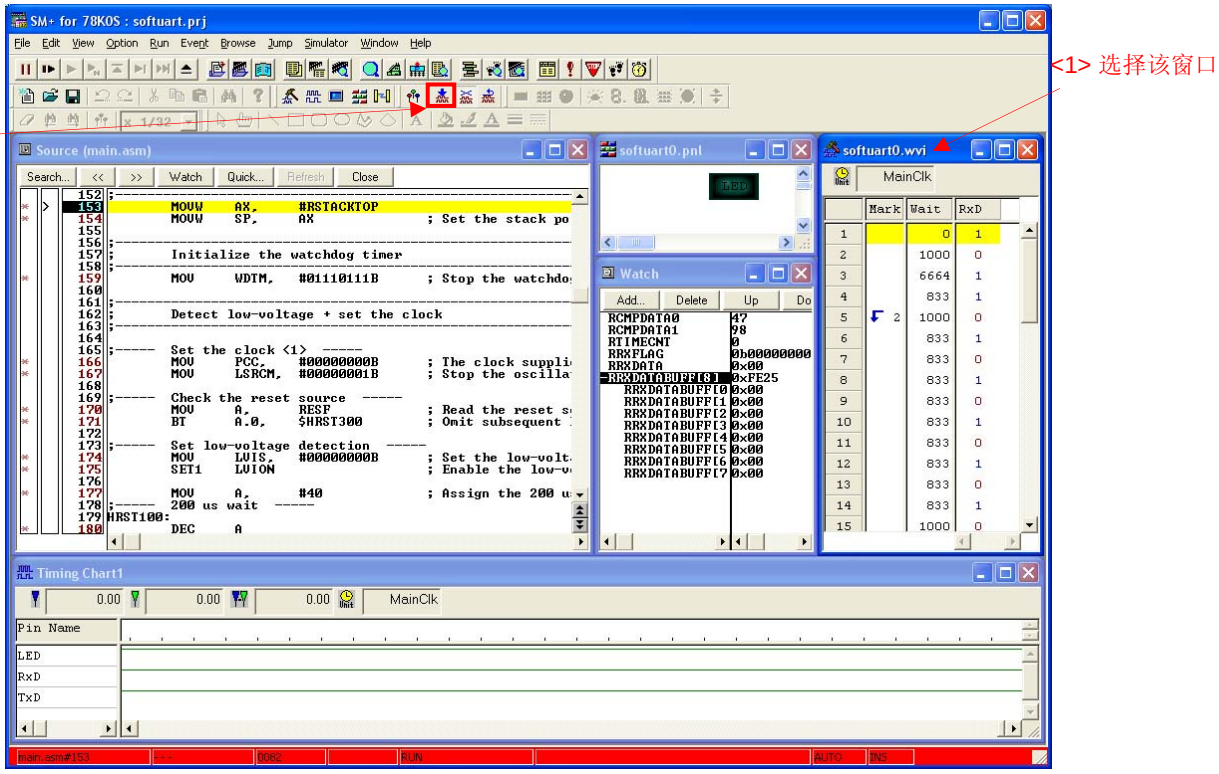
（3）选择源程序正文窗口（Source（main.asm））并在标号“LMLP300：”后为 NOP 指令设置一个断点来在所有处理完成后停止仿真。



（4）点击  （[重新启动]按钮）。在 CPU 复位后，程序开始执行，屏幕显示如下。

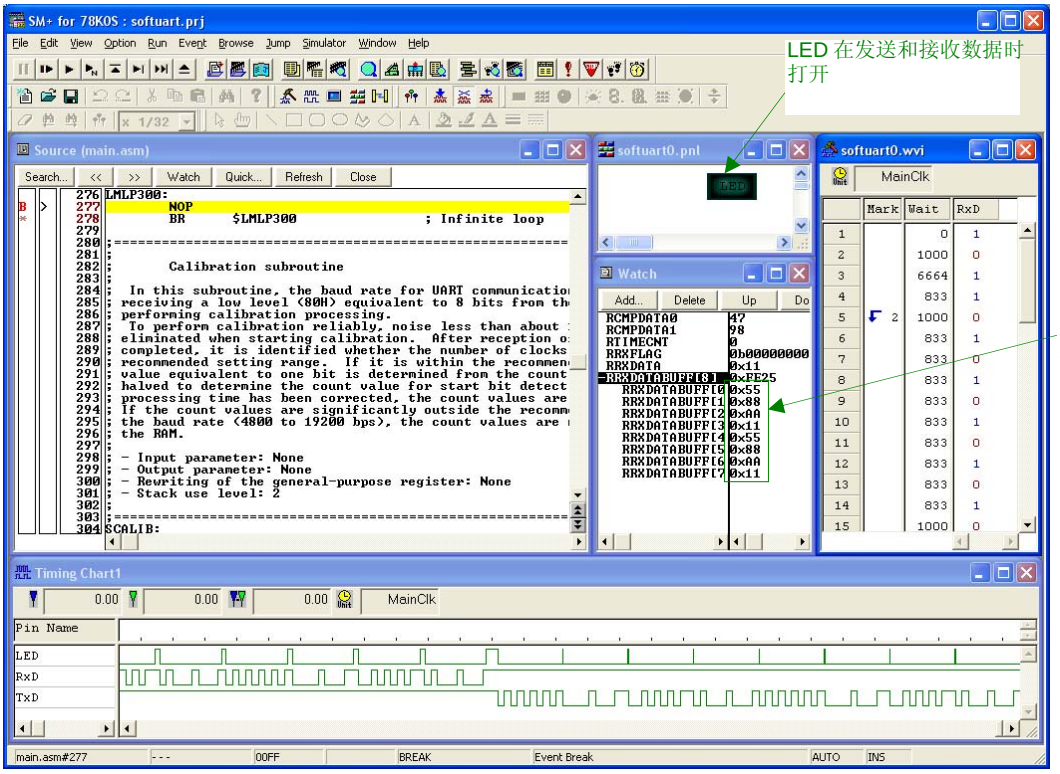


（5）选择信号数据编辑器窗口（softuart0.wvi），点击  （信号输入开始按钮）。



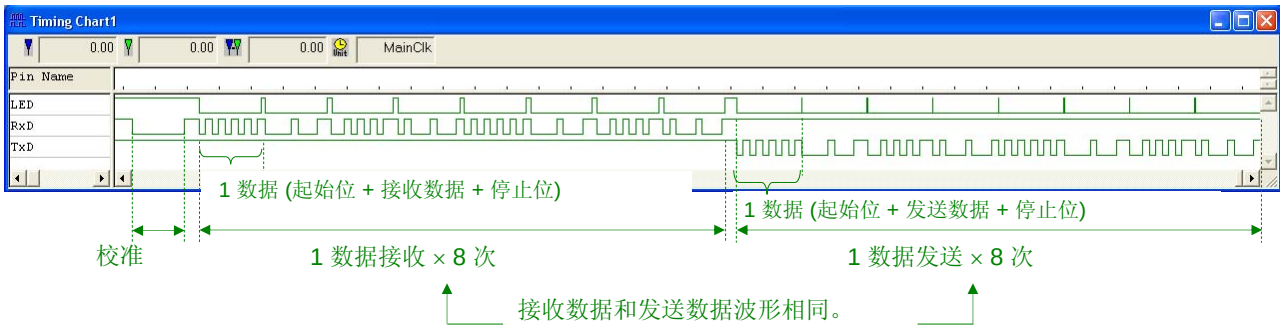
信号输入

（6）数据发送和接收将通过软件 UART 进行仿真，在所有处理完成后停止。

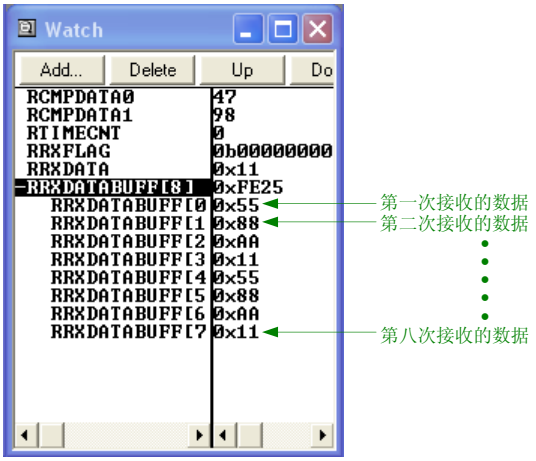


（7）I/O 波形将显示在时间图窗口中，接收的数据将显示在察看窗口中，如下所示。

●时间图窗口



● 察看窗口



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 系列说明书用户手册		PDF
RA78K0S 汇编包用户手册	语言	PDF
	操作	PDF
CC78K0S C 编译器用户手册	语言	PDF
	操作	PDF
PM+ 项目管理器用户手册		PDF
SM+ 系统仿真器操作用户手册		PDF
78K0S/KA1+ 简体闪存写入手册 MINICUBE2 信息		PDF
78K0S/Kx1+ 应用注释	示例程序启动指南	PDF
	示例程序 (初始设置) LED 照明开关控制	PDF
	示例程序 (低压检测) 检测电压低于2.7V时产生复位	PDF

附录 A 程序列表

作为程序列表示例， 78K0S/KB1+微控制器的源程序如下所示。

● main.asm

```
;*******************************************************************************
;
;       日电电子    78K0S/KB1+
;
;*******************************************************************************
;       78K0S/KB1+    示例程序
;*******************************************************************************
;       软件 UART
;*******************************************************************************
;<<记录>>
;       2007.9.--       发布
;*******************************************************************************
;
;<<概述>>
;
;本示例展示了利用软件控制 UART 通信示例。初始设置完成后进行校准，波特率通过接收等效于 8 位
;的低电平（80H）确定。之后，接收 8 单位数据作为接收试验，并作为发送试验进行发送。接收数据后，
;观察接收引脚的输入电平，利用低电平检测作为触发器开始校准和数据接收。波特率推荐设置范围为
; 4800 至 19200bps，当不执行校验时，设默认波特率为 9600bps。另外，在发送和接收时 LED 灯打开。
;
;
; <主要设置内容>
;
; - 停止看门狗定时器的运行
; - 将低压检测电压 (VLVI) 设置为 4.3V±0.2V
; - 在 VDD 大于等于 VLVI 后，当检测到 VDD 小于 VLVI 时，生成内部复位（LVI 复位）信号
; - 将 CPU 时钟频率设置为 8MHz
;
;
; <串口通信协议>
;
; - 波特率:                    4800 至 19200 bps (默认为 9600 bps)
; - 数据字符长度:      8 位
; - 奇偶校验设定:      无校验
; - 停止位个数: 1 位或 2 位 (默认 1 位)
; - 开始位设定: 从最低有效位（LSB）开始
;
;
; <关于接收错误>
;
; - 仅检测帧错误。
; - 检测不到奇偶校验错误和超限错误。
;
```

```
;
;<<I/O 端口设置>>
;
; 输入: P45
; 输出: P00-P03, P20-P23, P30-P33, P40-P44, P46, P47, P120-P123, P130
;  # 所有不使用的端口设为输出模式。
;
;************************************************************************


;========================================================================
;
;       定义符号
;
;========================================================================
PTXD            EQU   P4.0              ; UART 发送引脚(TxD 引脚)
PRXD            EQU   P4.5              ; UART 接收引脚(RxD 引脚)
PLED            EQU   P2.0              ; LED 显示发送和接收状态的引脚
CSTOPBIT        EQU   1                 ; 指定停止位个数

CCALOFFSET      EQU   (10+20-13)/16     ; 在开始校验时，校正 17 个时钟
CTROFFSET       EQU   (6+18+18+6)/8     ; 发送和接收期间，校正 48 个时钟
CSTOFFSET       EQU   (10+26+5)/8       ; 检测开始位时，校正 41 个时钟

CB4800                EQU   202                                 ; 1 位计数值为 4800 波特
CHB4800               EQU   (CB4800+CTROFFSET)/2-CSTOFFSET; 开始为计数值为 4800 波特
CB9600                EQU   98                                  ; 1 位计数值为 9600 波特
CHB9600               EQU   (CB9600+CTROFFSET)/2-CSTOFFSET;开始位计数值为 9600 波特
CB19200               EQU   46                                  ; 1 位计数值为 19200 波特
CHB19200        EQU   (CB19200+CTROFFSET)/2-CSTOFFSET       ; 开始位计数值为 19200 波特

;========================================================================
;
;       向量表
;
;========================================================================
XVCT   CSEG  AT     0000H
             DW     IRESET              ;(00)   RESET
             DW     IRESET              ;(02)   --
             DW     IRESET              ;(04)   --
             DW     IRESET              ;(06)   INTLVI
             DW     IRESET              ;(08)   INTP0
             DW     IRESET              ;(0A)   INTP1
             DW     IRESET              ;(0C)   INTTMH1
             DW     IRESET              ;(0E)   INTTM000
             DW     IRESET              ;(10)   INTTM010
             DW     IRESET              ;(12)   INTAD
             DW     IRESET              ;(14)   --
```

```
                DW      IRESET                  ;(16)   INTP2
                DW      IRESET                  ;(18)   INTP3
                DW      IRESET                  ;(1A)   INTTM80
                DW      IRESET                  ;(1C)   INTSRE6
                DW      IRESET                  ;(1E)   INTSR6
                DW      IRESET                  ;(20)   INTST6

;=============================================================================
;
;       CALLT 表
;
; 常用调用的子例程指令码可以利用 1 字节调用指令的 CALLT 指令进行缩短。
;
;=============================================================================
XCALT CSEG  CALLT0
ZRXDATA:    DW      SRXDATA             ; UART 接收子例程
ZTXDATA:    DW      STXDATA             ; UART 发送子例程

;=============================================================================
;
;       定义 RAM
;
;=============================================================================
DRAM  DSEG  SADDRP
RRXDATA:    DS      1               ; 接收数据(与接收状态成对)
RRXFLAG:    DS      1               ; 接收状态标志
                                    ; (帧错误如果 0 位为 1)
        DSEG  SADDR
RCMPDATA0:  DS      1               ; 开始位计数
RCMPDATA1:  DS      1               ; 1 位间隔计数
RTIMECNT:   DS      1               ; 实际计数
RRXDATABUFF:        DS      8       ; 发送和接收测试的数据缓存

;=============================================================================
;
;       定义存储体区
;
;=============================================================================
DSTK  DSEG  AT      0FEE0H
RSTACKEND:  DS      20H             ; 存储体区= 32 字节
RSTACKTOP:                          ; 存储体区开始地址 = FF00H

;*****************************************************************************
;
;       复位后初始化
;
;*****************************************************************************
XMAIN CSEG  UNIT
IRESET:
;-----------------------------------------------------------------------------
;       初始化栈指针
;-----------------------------------------------------------------------------
        MOVW AX,    #RSTACKTOP
        MOVW SP,    AX              ;设置栈指针
;-----------------------------------------------------------------------------
;       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 (fcpu) = fxp (= fx/4 = 2 MHz)
        MOV     LSRCM,          #00000001B    ; 低速内部振荡器停止振荡

;-----  检查复位源  -----
        MOV     A,      RESF            ; 读出复位源
        BT      A.0,    $HRST300        ;LVI 复位期间，省略随后的与 LVI 有关的处理并进入
SET_CLOCK
;-----  设置低压检测  -----
        MOV     LVIS,   #00000000B    ; 设置低压检测电平 (VLVI) 为 4.3 V +-0.2 V
        SET1    LVION                 ; 启用低压检测器操作

        MOV     A,      #40           ; 指定 200 us 等待计数值
;-----  200 us 等待  -----
HRST100:
        DEC     A
        BNZ     $HRST100              ; 0.5[us/clk] x 10[clk] x 40[计数] = 200[us]

;-----  VDD >= VLVI 等待处理  -----
HRST200:
        NOP
        BT      LVIF,   $HRST200      ; 如 VDD < VLVI，就分支执行

        SET1    LVIMD                 ; 设置以当 VDD < VLVI 时，生成内部复位信号
;-----  设置时钟 <2>  -----
HRST300:
        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 P00-P03 输出锁存器为低
        MOV     PM0,    #11110000B    ; 设 P00-P03 为输出模式

;-------------------------------------------------------------------------
;       初始化端口 2
;-------------------------------------------------------------------------
        MOV     P2,     #00000001B    ; 设 P21-P23 输出锁存器为高 (关掉 LED)
        MOV     PM2,    #11110000B    ; 设 P20-P23 为输出模式

;-------------------------------------------------------------------------
;       初始化端口 3
```

```
;-------------------------------------------------------------------------
        MOV    P3,     #00000000B    ; 设 P30-P33 输出锁存器为低
        MOV    PM3,    #11110000B    ; 设 P30-P33 为输出模式

;-------------------------------------------------------------------------
;       初始化端口 4
;-------------------------------------------------------------------------
        MOV    P4,     #00000001B    ; 设置 P41-P47 输出锁存器为高,设置 P40 为高（为串口发送而
设）
        MOV    PU4,    #00100000B    ; 连接片上上拉电阻至 P45
        MOV    PM4,    #00100000B    ; 设 P40-P44、P46、P47 为输出模式，P45 为输入模式 (串口接
收)

;-------------------------------------------------------------------------
;       初始化端口 12
;-------------------------------------------------------------------------
        MOV    P12,    #00000000B    ; 设 P120-P123 输出锁存器为低
        MOV    PM12,   #11110000B    ; 设 P120-P123 为输出模式

;-------------------------------------------------------------------------
;       初始化端口 13
;-------------------------------------------------------------------------
        MOV    P13,    #00000001B    ; 设置 P130 输出锁存器为高

;-------------------------------------------------------------------------
;       初始化 RAM
;-------------------------------------------------------------------------
        MOV    RCMPDATA1, #CB9600          ; 1 位计数定时器默认值 (9600 bps)
        MOV    RCMPDATA0, #CHB9600         ; 开始位计数定时器默认值  (9600 bps)
        MOVW AX,      #0000H
        MOVW RRXDATA, AX             ; 初始化接收数据及接收状态

;*************************************************************************
;
;       主循环
;
;*************************************************************************
MMAINLOOP:

;-----  校准  -----
        CALL   !SCALIB               ; 校准处理 (等待 80H 接收)

;-----  接收测试  -----
        MOVW HL,      #RRXDATABUFF; 指定缓存开始地址
        MOV    B,     #8            ; 指定接收个数
LMLP100:
        CALLT  [ZRXDATA]             ; 接收数据
        MOV    [HL],  A              ; 将数据写入缓存
        INC    L                     ; 缓存地址增加 1
        DBNZ   B,     $LMLP100       ; 重复 8 次

;-----  等待其他方通信处理  -----
        MOV    A,     RCMPDATA1      ; 读出 1 位间隔计数数据
LMLP150:
```

```
        NOP                             ; 通过利用处理时间设置等待时间直到发送完成
        NOP                             ; 考虑启用其他方通信接收
        DEC     A
        BNZ     $LMLP150

;----- 发送测试 -----
        MOVW HL,        #RRXDATABUFF; 指定缓存开始地址
        MOV     B,      #8              ; 指定发送个数
LMLP200:
        MOV     A,      [HL]            ; 读出缓存数据
        CALLT   [ZTXDATA]               ; 发送数据
        INC     L                       ; 缓存地址增加 1
        DBNZ    B,      $LMLP200        ; 重复 8 次

;-----  无限循环  -----
LMLP300:
        NOP
        BR      $LMLP300                ; 无限循环

;===============================================================================
;
;       校准子例程
;
;  在本子例程中, 通过接收 RxD 引脚等于 8 位的低电平(80H)确定 UART 通信波特率，然后执行校准
;处理。
;为校准可靠，开始校准时，消除噪音小于约 1.5 us 的校准。 完成接收低电平之后，定义是否时钟个数
;在推荐设定范围内。如果在推荐范围内，开始位检测计数值一半的计数值确定等于 1 位计数值。校正各
;处理时间后，存储计数值至 RAM 区。如果波特率计数值大大超出推荐范围(4800 至 19200 bps)，计
;数值不存储至 RAM 区。
;
; - 输入参数: 无
; - 输出参数: 无
; - 通用寄存器的重写:无
; - 栈使用电平: 2
;
;===============================================================================
SCALIB:
        PUSH    AX                      ; 保存 AX 寄存器数据至栈
        PUSH    HL                      ; 保存 HL 寄存器数据至栈

;-----  开始校准前处理  -----
JCAL000:
        BF      PRXD,  $JCAL000         ; 如果 RxD 引脚为 0,等直到变为 1
        DI                              ; 禁用向量中断
;-----  校准处理  -----
JCAL100:
        BT      PRXD,  $JCAL100         ;10: 等待校准开始
        BT      PRXD,  $JCAL100         ;10: 如出现噪音，返回等待校准开始

        NOP                             ; 2: 时间调整
```

```
        NOP                             ; 2: 时间调整
        MOVW HL,    #CCALOFFSET; 6: 时间校正
JCAL200:
        NOP                             ; 2:
        INCW   HL                       ; 4: 时间测量
        BF     PRXD,  $JCAL200          ;10: 等待 RxD 引脚变为 1

;-----  定义校验结果  -----
        MOVW AX,    HL
        CLR1   CY
        RORC   A,      1                ; 将结果乘以 1/2
        XCH    A,      X
        RORC   A,      1
        XCH    A,      X
        CLR1   CY
        RORC   A,      1                ; 再乘以 1/2 (将结果乘以 1/4)
        XCH    A,      X
        RORC   A,      1
        XCH    A,      X
        CMP    A,      #0               ; 高 8 位为 0?
        BNZ    $JCAL300                 ; 如太慢，退出
        XCH    A,      X
        CMP    A,      #45              ; 低限检查
        BC     $JCAL300                 ; 如太快，退出
                                        ; CY = 0
;-----  保存至定时器计数寄存器  -----
        MOV    RCMPDATA1, A             ; 设置 RCMPDATA1 设置值为保存区
        SUB    RCMPDATA1, #CTROFFSET    ; 在发送和接收期间，校正等效于 48 个时钟数据
        RORC   A,      1                ; 再乘以 1/2
        SUB    A,      #CSTOFFSET       ; 校正开始位处理
        MOV    RCMPDATA0, A             ; 开始位处理

JCAL300:
        POP    HL                       ; 恢复 HL 寄存器数据
        POP    AX                       ; 恢复 AX 寄存器数据
        RET

;===============================================================================
;
;       UART 接收子例程
;
; 在本子例程中，执行等效于 1 个字符的数据接收处理
; 为进行可靠的数据接收，当开始接收时，消除小于 1.5 us 噪音。开始位检测后，在存储的 1 位数据
; 中心定义 0 或 1。1 位接收子例程用来定义和存储 1 位数据，当停止位检测完成后，数据存储至
; RRXDATA(2 字节)。同时，接收数据存储至低 1 字节(RRXDATA)，接收状态标志存储至高 1 字节
; (RRXFLAG)。另外，接收数据通过可与 RAM 和 AX 寄存器同时执行，因为接收数据存储至 A 寄存器，接
; 收状态标志存储至 X 寄存器，处理从子例程返回 。
;
```

```
; - 输入参数:无
; - 输出参数: A 寄存器(接收数据)，X 寄存器(接收状态标志)
; - 通用寄存器的重写: AX 寄存器
; - 栈使用电平: 2
;
;=========================================================================
SRXDATA:
        PUSH   BC                       ; 保存 BC 寄存器数据至栈

;-----  开始接收之前处理  -----
JRXD000:
        BF     PRXD,  $JRXD000          ; 如果 RxD 引脚为 0，等待直到变为 1
        DI                              ; 禁用向量中断


;-----  开始位检测处理 -----
JRXD100:
        BT     PRXD,  $JRXD100          ;10: 等待开始位检测
        BT     PRXD,  $JRXD100          ;10: 如果出现噪音，返回等待位开始位检测

        CLR1   PLED                     ; 6: 打开 LED (数据接收期间)
        NOP                             ; 2: 时间调整
        MOV    A,     RCMPDATA0         ; 4: 读出设置值
        MOV    RTIMECNT, A              ; 4: 设立位中心
JRXD200:
        DBNZ   RTIMECNT, $JRXD200; 8: 等待开始为中心
        BT     PRXD,  $JRXD100          ;10: 如未检测到起始位，返回等待检测

        NOP                             ; 2: 时间调整
        MOV    B,     #8+1              ; 6: 设置接收位的剩余个数
        MOVW AX,      #0000H; 6: 设置初始数据

;-----  数据接收处理 -----
JRXD300:
        CALL   !SRXBIT                  ; 6: 接收位
        DBNZ   B,     $JRXD300          ; 6: 计算接收位个数

        SET1   PLED                     ; 关闭 LED (数据接收结束)

;-----  接收数据保存处理 -----
        XCH    A,     X                 ; 保存接收数据至 X 寄存器
        NOT1   CY
        ROLC   A,     1                 ; 如未检测停止位，设 0 位为 1
        MOVW RRXDATA, AX                ; 保存接收数据和错误状态
        XCH    A,     X                 ; 存储接收数据至 A 寄存器
                                        ; 存储错误状态至 X 寄存器
        POP    BC                       ; 恢复 BC 寄存器数据
        RET

;-------------------------------------------------------------------------
;       1 位接收子例程
;
; - 输入参数: A 寄存器（接收数据), CY 标志(接收位)
```

```
; - 输出参数: A 寄存器(接收数据), CY 标志 (接收位)
; - 通用寄存器重写: A 寄存器
; - 栈使用电平: 0
;
;-------------------------------------------------------------------------
SRXBIT:
        NOP                             ; 2: 用发送处理调整时间
        RORC  A,        1               ; 2: 通过右移获取接收数据 (CY 标志)
        MOV   RTIMECNT, A               ; 4: 保存接收数据
        MOV   A,        RCMPDATA1       ; 4: 获取接收时间计数值
        XCH   A,        RTIMECNT        ; 6: 设置该计数值及恢复该接收数据
JRXB100:
        DBNZ  RTIMECNT, $JRXB100 ; 8*n:等待时间

        BT    PRXD,   $JRXB200          ;10: 检查接收数据
        CLR1  CY                        ; 2: 如果接收 0，CY 为 0
        RET                             ; 6:
JRXB200:
        SET1  CY                        ; 2: 如果接收 1，CY 为 1
        RET                             ; 6:

;=========================================================================
;
;       UART 发送子例程
;
; 在该子例程中, 执行等效于一个字符的数据发送
; 将发送的数据存储至 A 寄存器，且如下列所示调用该子例程
; 1 位发送子例程用来发送该数据，且当完成了停止位发送时，从本子例程返回处理 。
;
; 程序示例:
;       MOV   A,        #54H    ;存储 54H 至 A 寄存器
;       CALLT [ZTXDATA]         ; 调用 UART 发送子例程
;
; - 输入参数: A 寄存器 (发送数据)
; - 输出参数:无
; - 通用寄存器重写: A 寄存器
; - 栈使用电平: 2
;
;=========================================================================
STXDATA:
        PUSH  BC                        ; 保存 BC 寄存器数据至栈
;-----  开始发送前处理   -----
        DI                              ; 禁用向量中断

;-----  开始位发送处理   -----
        CLR1  PTXD                      ; 6: 发送开始位
        CLR1  PLED                      ; 6: 打开 LED (数据发送期间)
        MOV   B,        #1+8+CSTOPBIT       ; 6: 设置发送位数

;----- 数据发送处理  -----
JTXD100:
        CALL  !STXBIT                   ; 6: 发送位
        DBNZ  B,        $JTXD100        ; 6: 计算发送位个数
```

```
        SET1    PLED                    ; 关闭 LED (数据发送结束)

        POP     BC                      ; 恢复 BC 寄存器数据
        RET
;---------------------------------------------------------------------------
;       1 位发送子例程
;
; - 输入参数: A 寄存器 (发送数据)
; - 输出参数: A 寄存器(发送数据)
; - 通用寄存器重写: A 寄存器
; - 栈使用电平: 0
;
;---------------------------------------------------------------------------
STXBIT:
        MOV     RTIMECNT, A             ; 4: 保存发送数据
        MOV     A,      RCMPDATA1       ; 4: 获取发送时间计数值
        XCH     A,      RTIMECNT        ; 6: 设置计数值及恢复发送数据
        SET1    CY                      ; 2: 输出后设数据为 1
        RORC    A,      1               ; 2: 右移发送数据为 CY 标志
JTXB100:
        DBNZ    RTIMECNT, $JTXB100; 8*n: 等待时间

        BC      $JTXB200                ; 6:如 CY 为 1，就分支执行
        CLR1    PTXD            ; 6: 发送 0
        RET                             ; 6:
JTXB200:
        SET1    PTXD            ; 6: 发送 1
        RET                             ; 6:
end
```

● op.asm

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

      DB     11111111B     ; 保护字节区 (自编程模式)
;           ||||||||
;           ++++++++-----------所有的块可以被写或擦除
```

结束

附录 B 修订记录

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