北京飞锐泰克科技有限公司
Add: 北京市海淀区中关村东路 66 号甲 1 号楼长城大厦 1105 室
Tel:010-59831537/59831538/59831539 Fax:010-59831536
Web:www.free-tech.com.cn

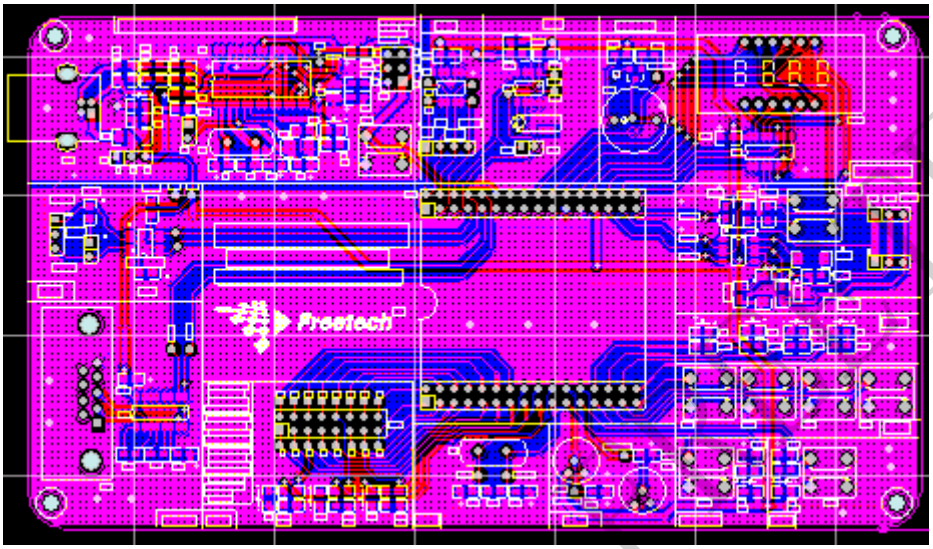
MC9S08DZ 学习板

目录

第 1 章 MC9S08DZ 学习板概述

MC9S08DZ 学习板是基于 MC9S08DZ 系列芯片开发的一款学习板。该学习板硬件资源丰富，布局清晰明了，利用该学习板的资源可熟悉和掌握 MC9S08DZ 系列芯片的功能。重点掌握 CAN 总线和 LIN 总线的通讯协议。

本部分将对 MC9S08DZ 学习板的功能进行简单介绍，以方便用户快速了解该学习板的功能。下图为 MC9S08DZ 学习板实物图。



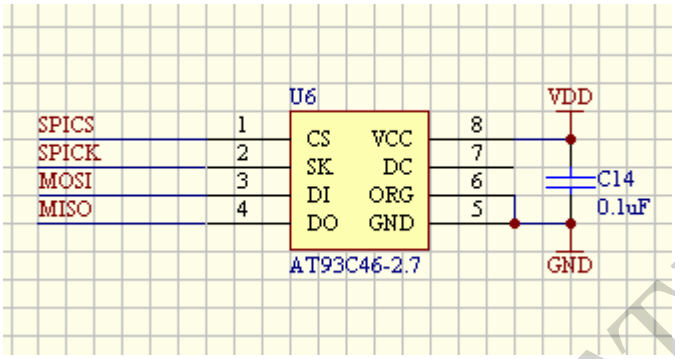
P1-1 MC9S08DZ 学习板主板图

该学习板可分 16 个模块，在学习板上均有标注。下面简单介绍每个功能模块。

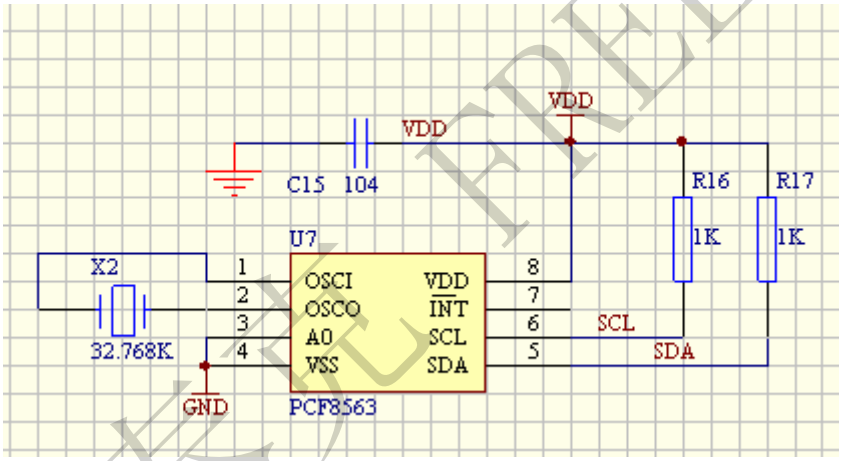
- HCS08 Open Source BDM：该模块功能为程序下载和调试。该模块通过一个 6 芯连接线与 MC9S08DZ 模块相连。同时该模块可独立作为一个程序下载调试器。
- SPI：SPI 接口。通用的 eeprom 93c46。
- IIC：该模块使用 PCF8563 时钟芯片。
- LIN：该模块使用 TJA1020 芯片。与子板配合可调试 LIN 总线。
- SOUND：电平驱动一个蜂鸣器。
- DISPLAY：该模块采用 4 位 LED 数码管，由 74HC164 芯片驱动。
- LED：4 个由单片机 IO 控制的发光二极管。
- KEY：4 个按键，熟悉单片机的口线中断功能。
- IRQ：外部中断按键，熟悉单片机的外部中断功能。
- RESET：复位按键，熟悉单片机的外部复位。
- ADC：两路 ADC 输入，熟悉单片机的 ADC 功能。
- ICS：熟悉单片机的内部时钟发生器，总线时钟。
- TPM：可调试单片机内部定时器相关功能，熟悉单片机的定时器功能。
- UART：可调试单片机的串口。
- CAN：可调试 S08 单片机 CAN2.0 通讯协议功能。
- MC9S08DZ：该板预留晶振电路，调试接口和电源接口。故该板可脱离学习板而独立使用。

第 2 章 MC9S08AW 学习板结构

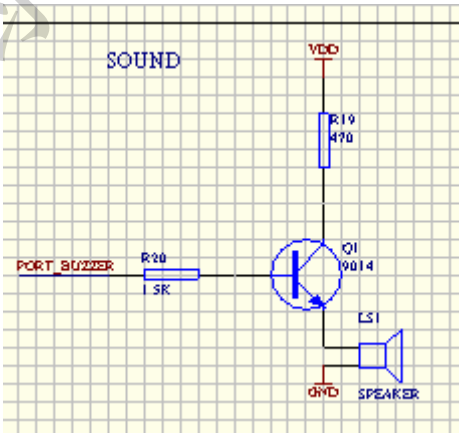
2．1 原理图



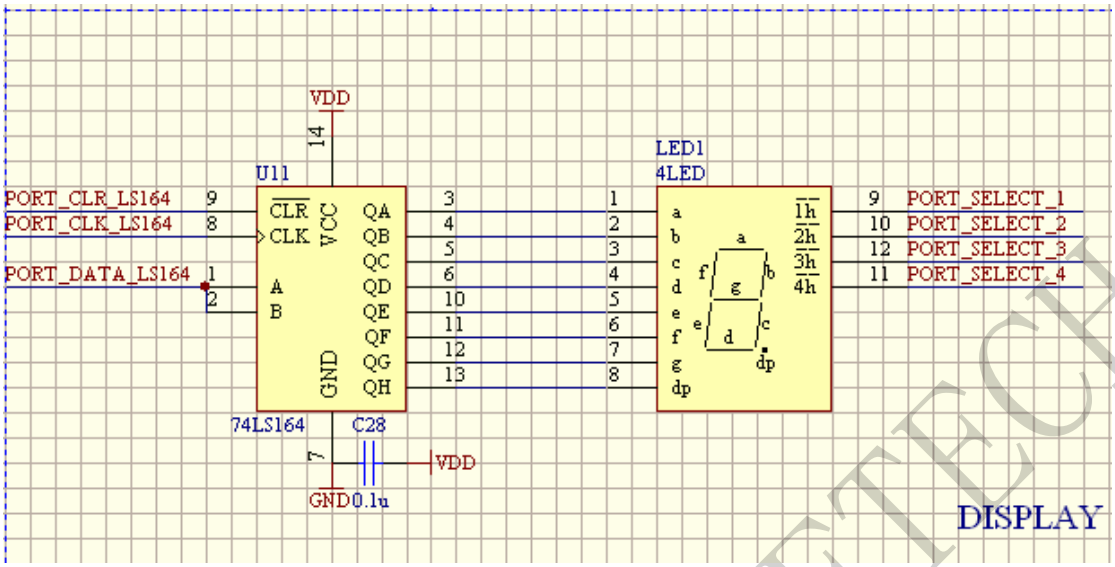
P2-1 SPI 接口原理图



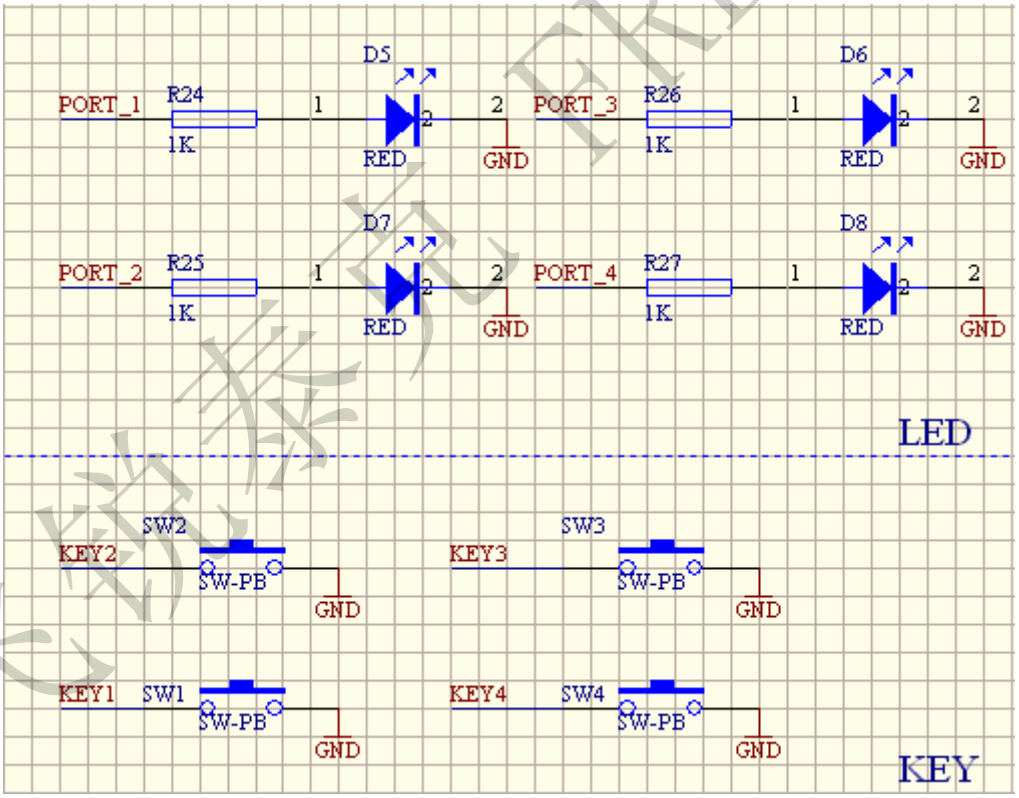
P2-2 IIC 原理图



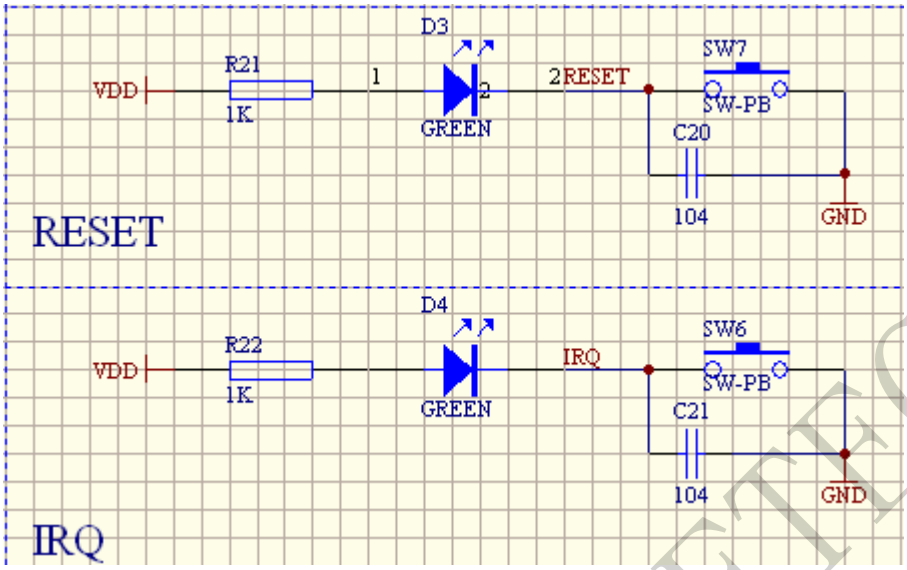
P2-3 蜂咛器驱动原理图



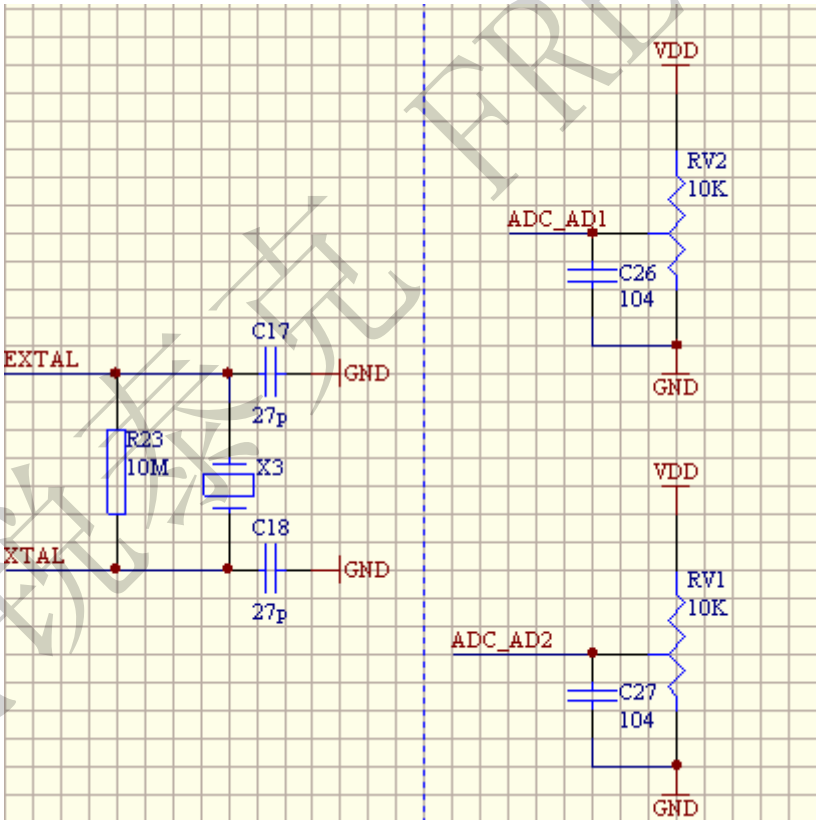
P2-4 4 位数码管原理图



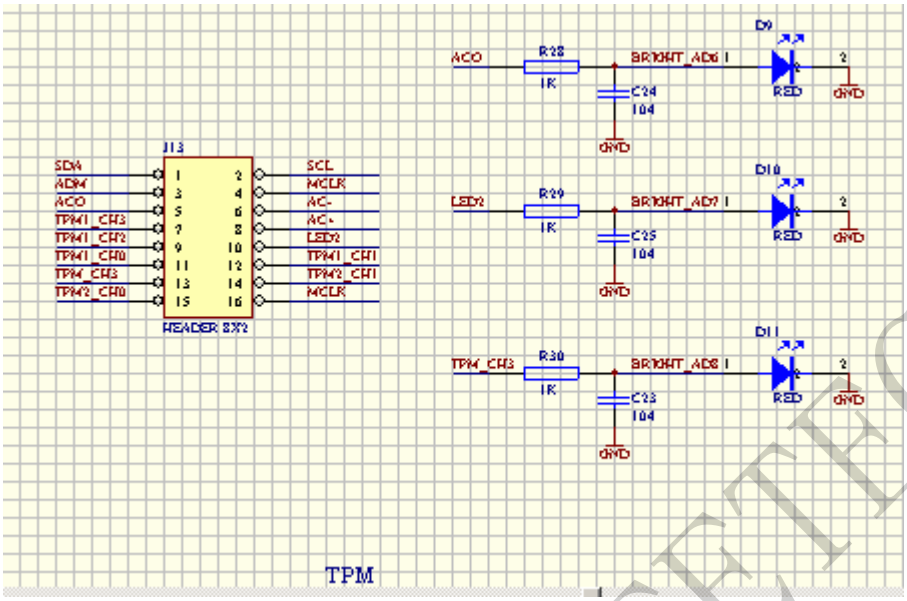
P2-5 LED 与按键原理图



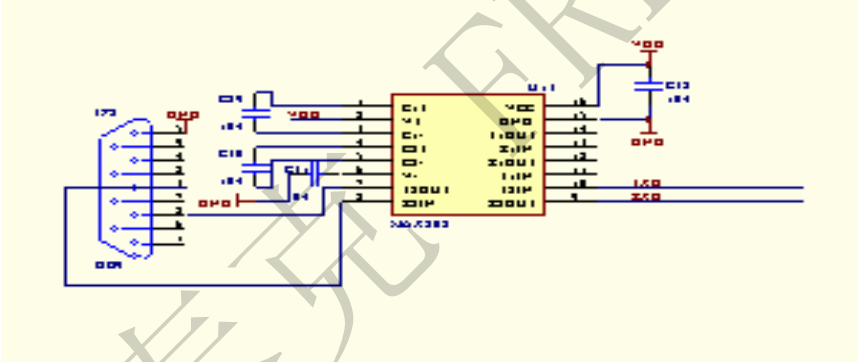
P2-6 外部中断与复位按键原理图



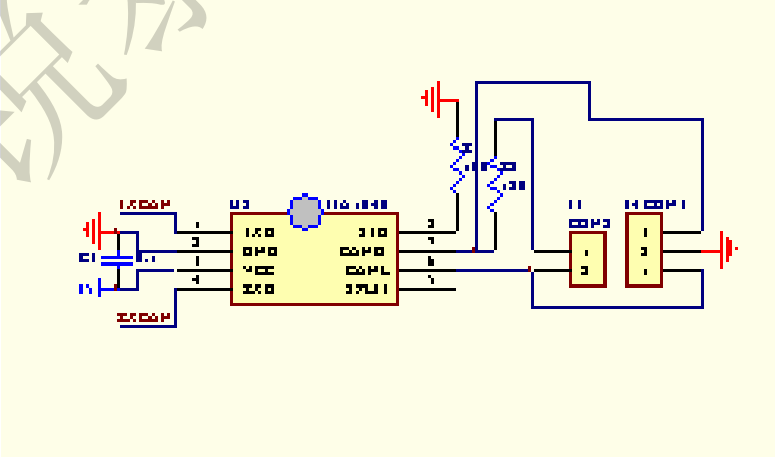
P2-6 晶振和 ADC 原理图



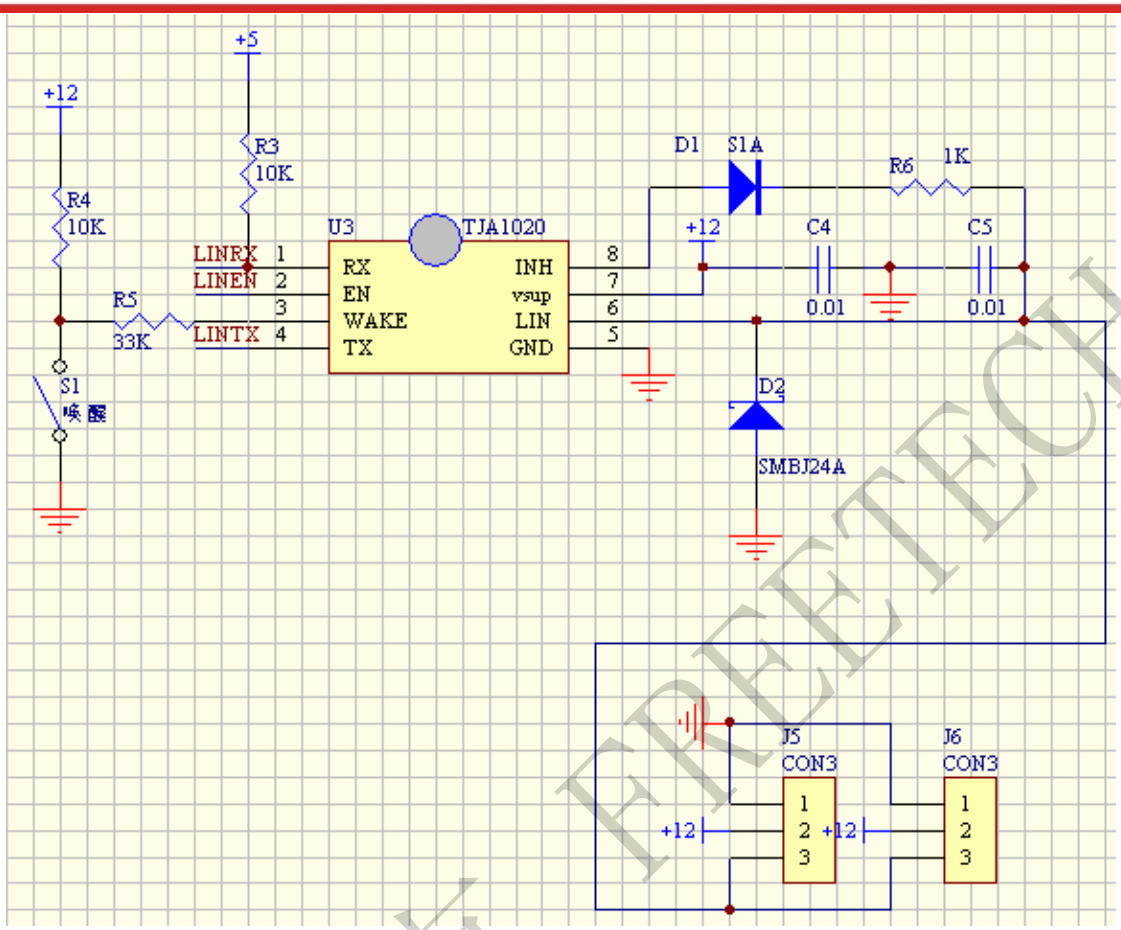
P2-7 TPM 时钟模块原理图



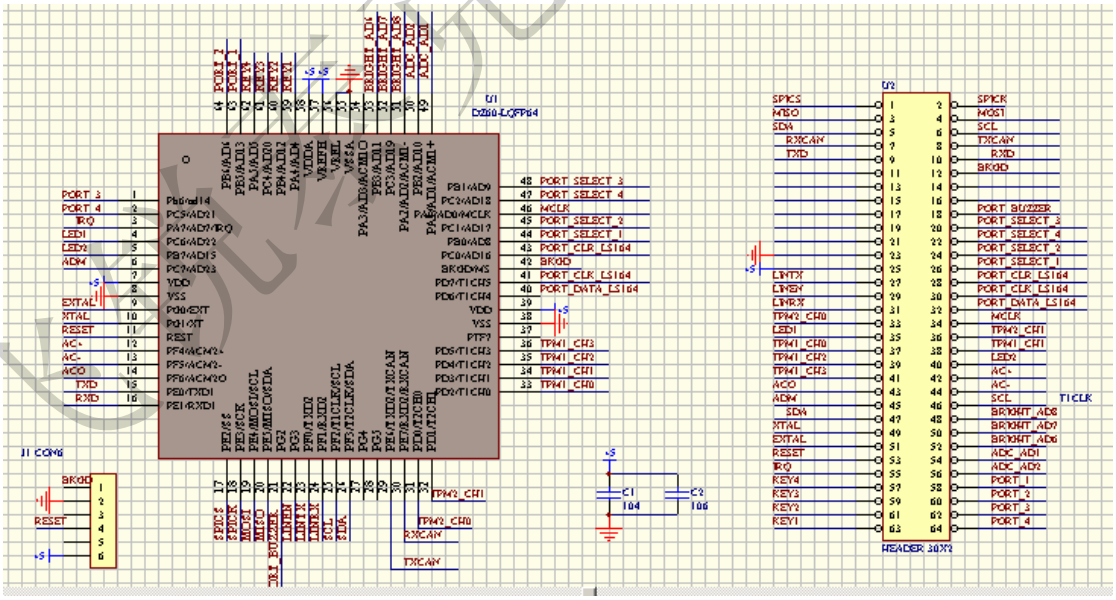
P2-7 SCI 原理图



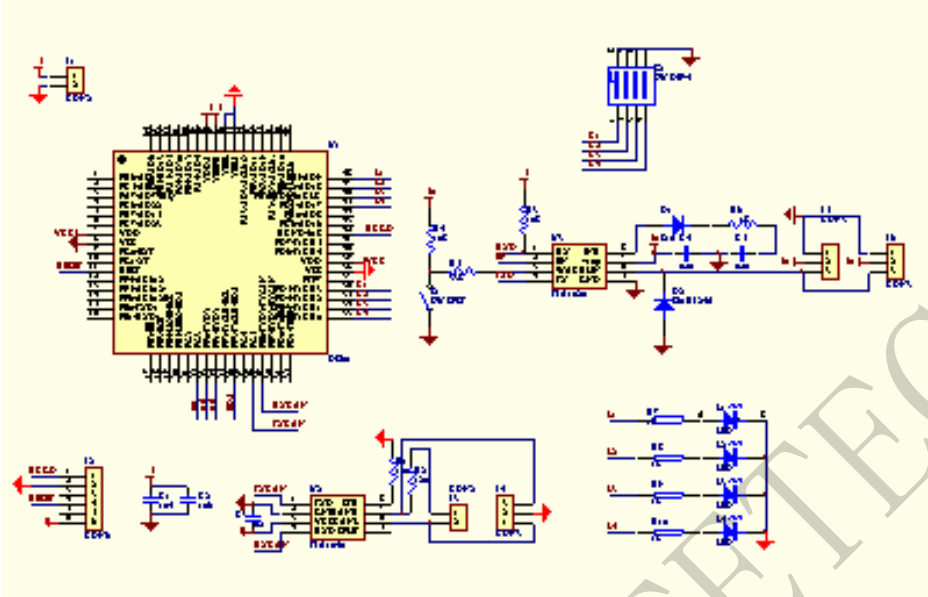
P2-8 CAN 原理图



P2-9 LIN 接口原理图



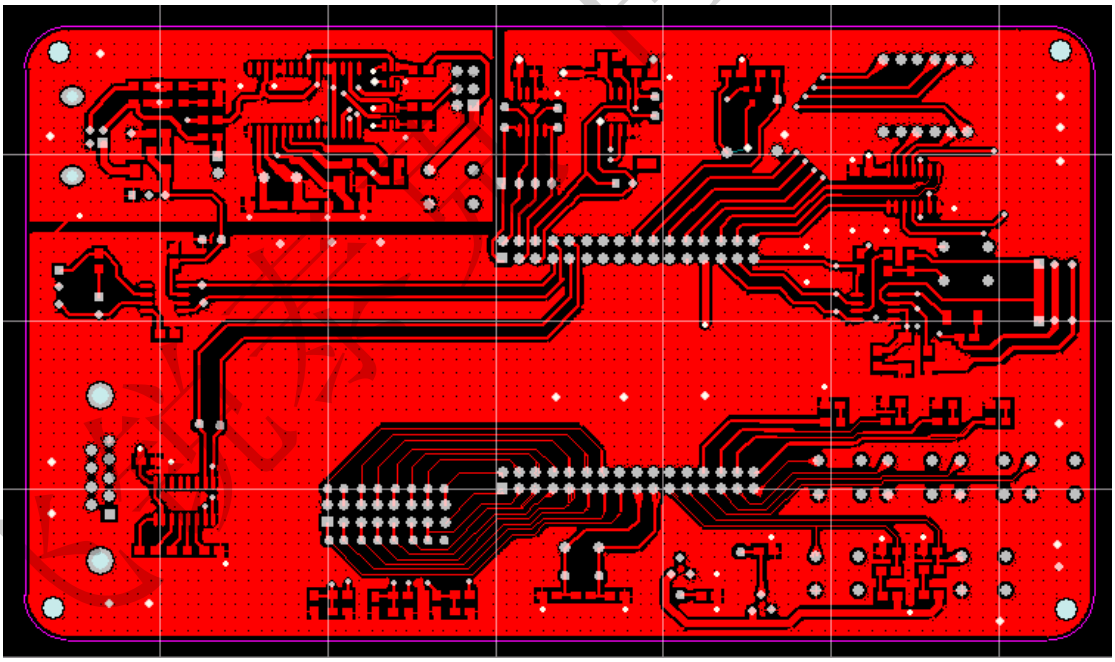
P2-10 DZ60CPU 原理图



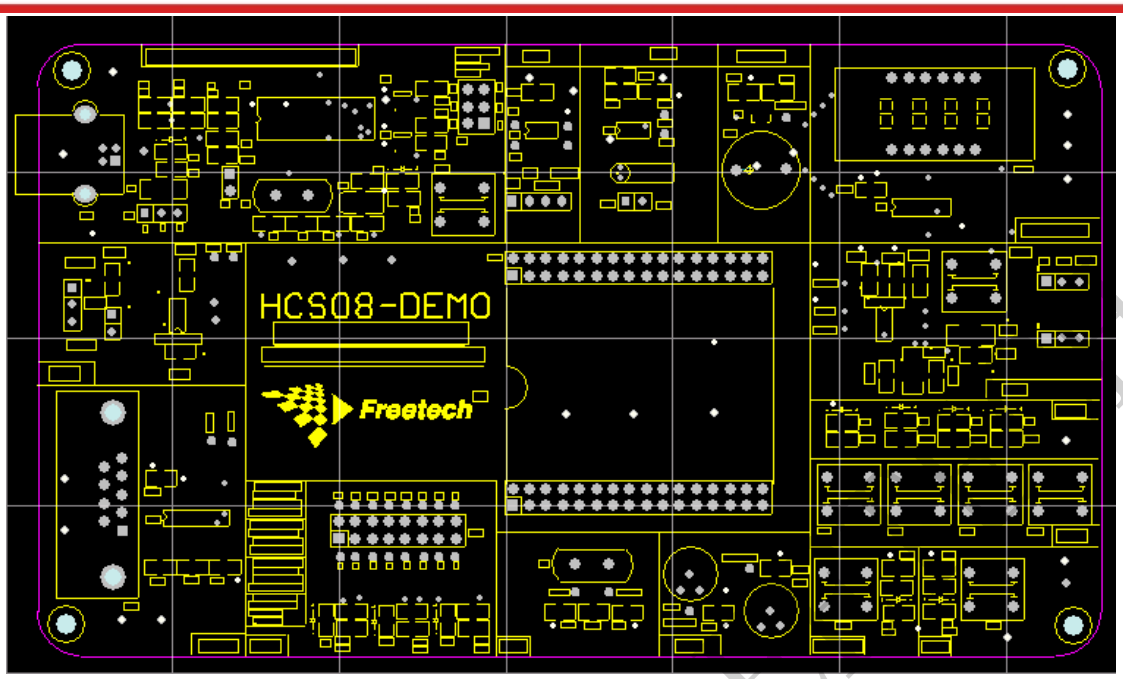
P2-11 目标板原理图

参见 HCS08-DEMO-DEMO 学习板原理图.pdf(HCS08 Open Source BDM 部分原理图被屏蔽)和 MC9S08DZ 原理图.pdf。

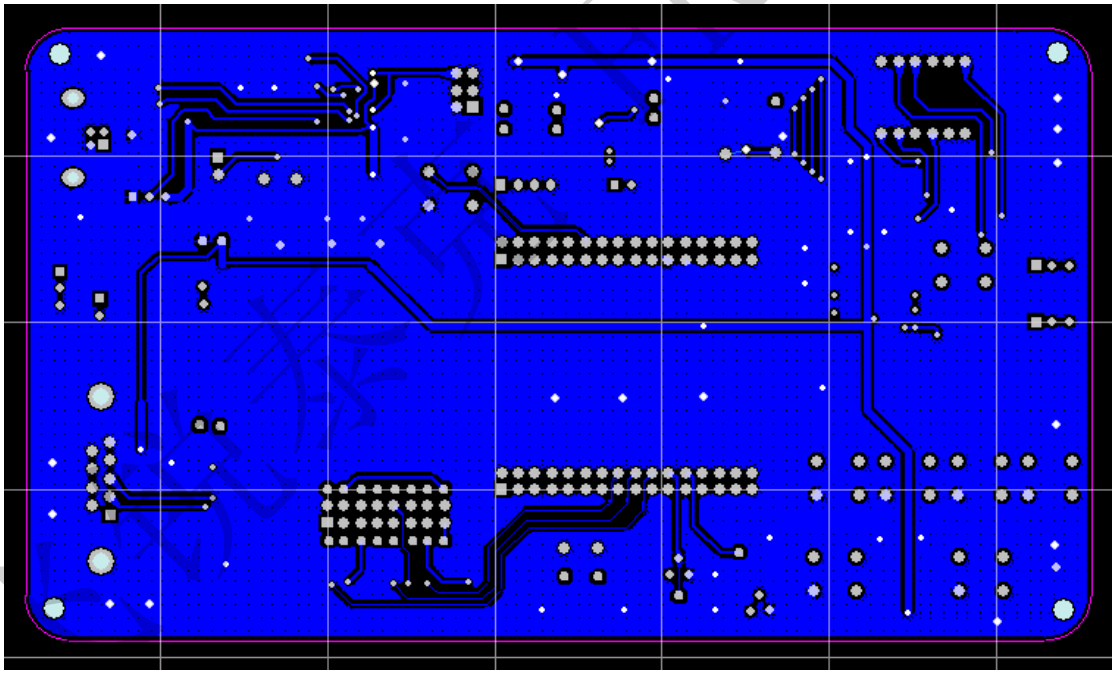
2．2 丝印图



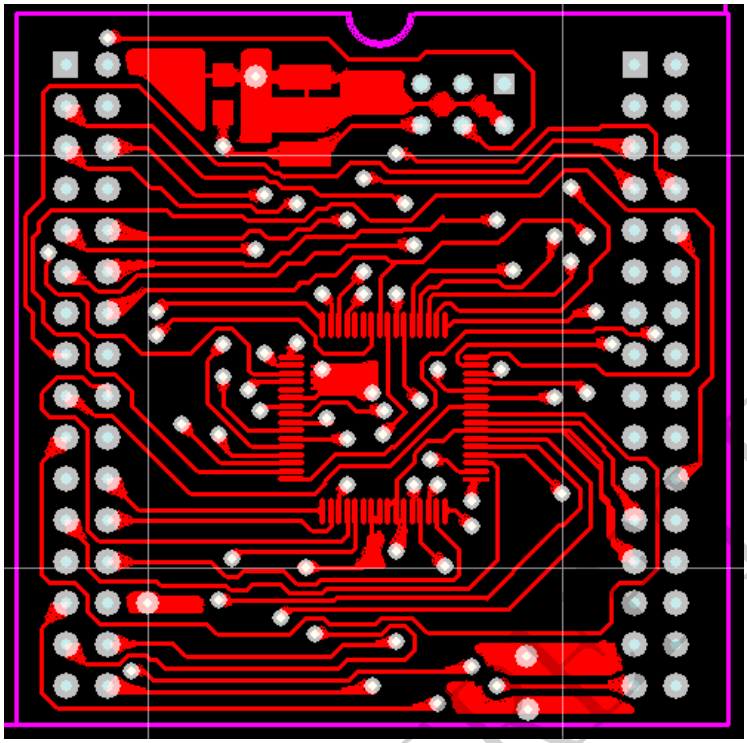
HCS08-DEMO 学习板顶层



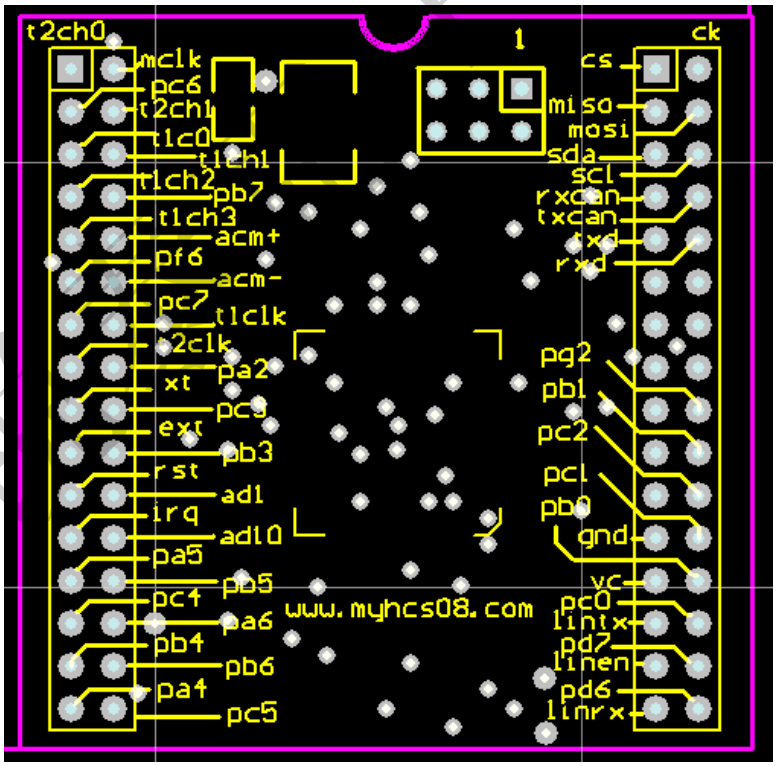
HCS08-DEMO 学习板顶层丝印层

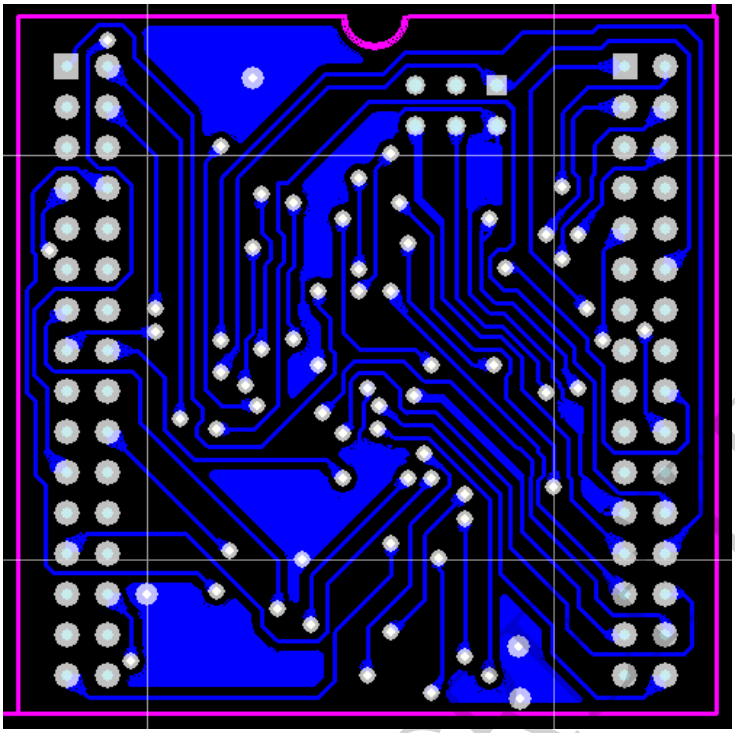


HCS08-DEMO 学习板底层

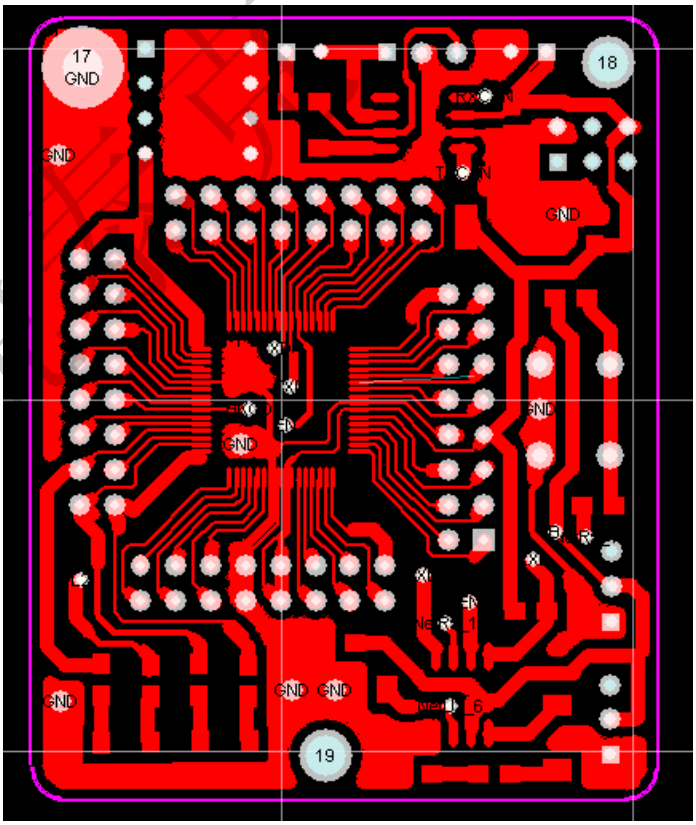


MC9S08DZ 学习板顶层

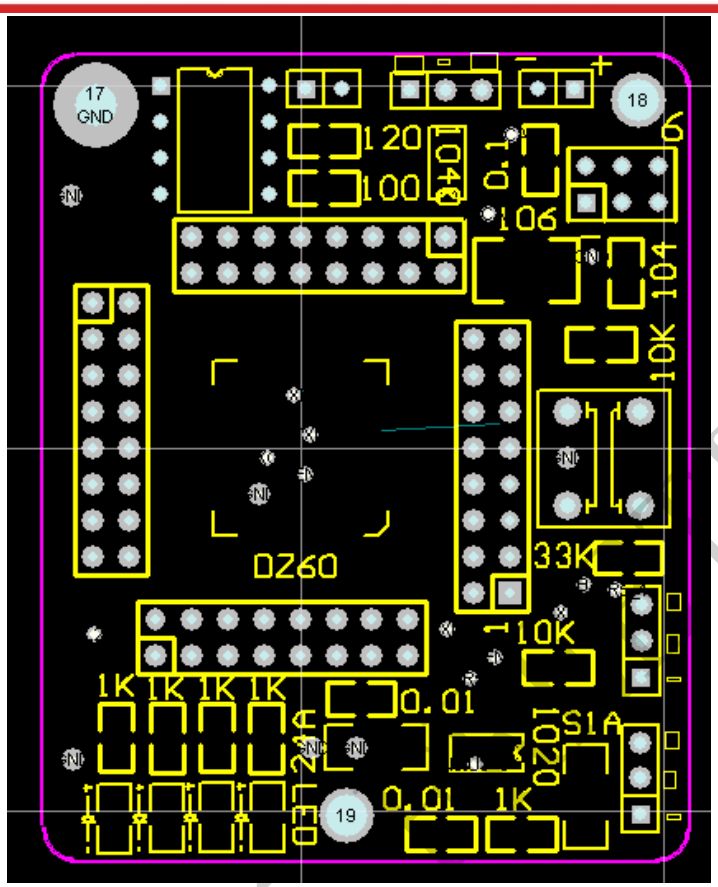


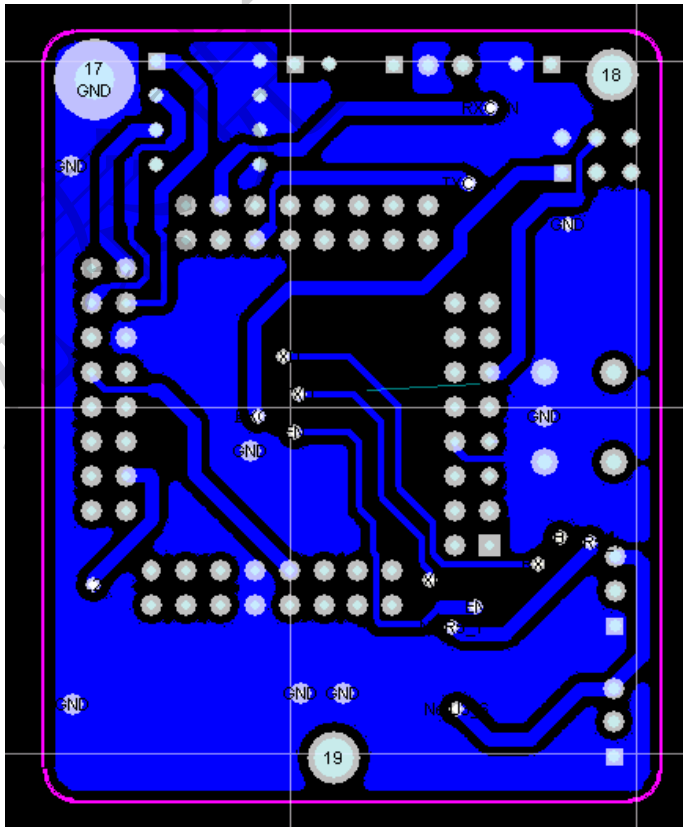


MC9S08DZ 学习板底层



子板的顶层





参见 MC9S08DZ 学习板顶层.pdf；MC9S08DZ 学习板顶层丝印层.pdf；MC9S08DZ 学习板底层.pdf。

2．3 主要器件

参见 HCS08-DEMO 料单.pdf 和 MC9S08DZ 料单.pdf。

2．4 外围接口

MC9S08DZ 学习板主要配用 5 条线缆：USB 数据线连接 PC 的 USB 接口和学习板的 JP1；串口线连接 PC 的串口和学习板的 JP2；6 芯线缆连接 HCS08-DEMO 板的 J3 与 MC9S08DZ 的 J1。3p 的线缆联接子板进行 lin 和 can 的通讯。

2．5 跳线说明

- HCS08 Open Source BDM

1） 正常使用时 J1,J2 的 2，3 短接

2） J3 与 MC9S08DZ 部分的 J1 用线缆连接，要求一一对应

3） SW5 的使用：在调试过程中，当目标 MCU（即 MC9S08DZ 部分的 MCU）进入不了后台调试模式(BDM)时，可使用该按键强制目标 MCU 进入 BDM。

方法：断开 USB 数据线，按下 SW5(不松开)，然后把 USB 数据线连接到 JP1，几秒种后松开 SW5。

- TPM

通过跳线可完成目标 MCU 的定时器 1，定时器 2，定时器 3 的计数功能，通道的输入捕捉功能，通道的比较输出功能和通道的 PWM 功能。

- MC9S08DZ

该模块可脱离学习板使用，有自己的外部晶振电路。

J2 为该板的电源接口：2.7V~5.5V。

J1 与 HCS08 Open Source BDM 处的 J3 一一连接。

- LIN

该模块可以实现 LIN 的多机通讯，接口定义，vc(12v)，LIN，GND。接口电平为显性和隐性。

- CAN

该模块可以实现 CAN 的多机通讯，接口定义，CANL，GND，CANH。接口电平为差分信号，显性和隐性。

第 3 章 CodeWarrior 快速入门

（一）安装CodeWarrior 软件

SYSTEM REQUIREMENTS	
Hardware	PC with 1 GHz Intel® Pentium®-compatible processor 512 MB of RAM (1 GB recommended) CD-ROM drive Depending on host-target connection: Parallel Port, 9-pin Serial Port, or USB Port
Operating System	Microsoft® Windows® 2000 or Windows® XP
Disk Space	2 GB total 1.7 GB on Windows system disk

1）运行CD 中的CW08_V6_0.EXE 文件。

2）按照提示完成安装。 该免费软件在使用上有一定限制。当使用汇编语言开发应用程序时，对源代码数量是没有限制的；当使用C 语言在HC(S)08/RS08系列单片机上开发应用程序时，对源代码有32K 的限制；当使用C 语言在ColdFire V1系列单片机上开发应用程序时，对源代码有64K 的限制。

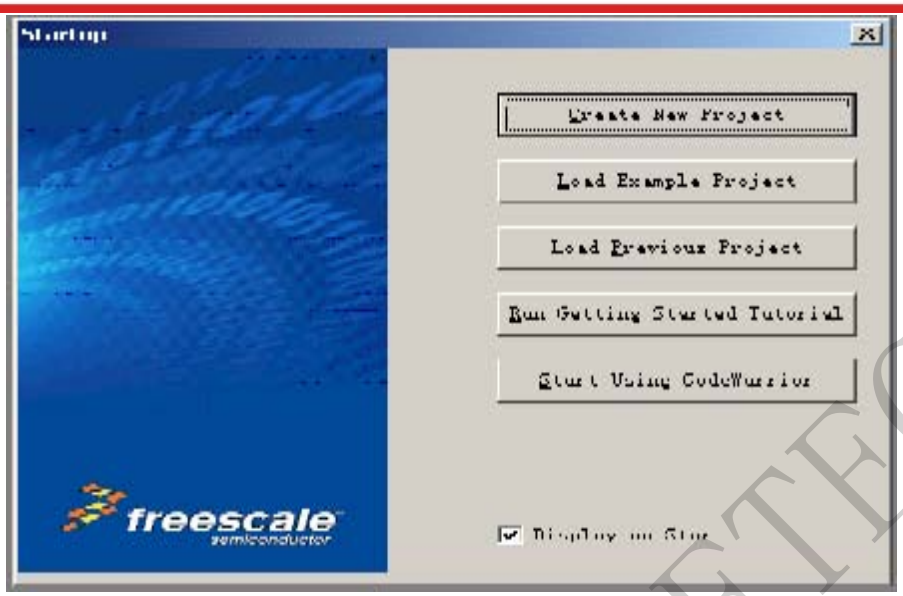
（二）、安装Open Source BDM驱动

将学习板与PC用USB数据线连接起来，此时PC会检测到一个USB设备，并提示安装该设备驱动，用户可按提示完整驱动安装。该USB设备驱动程序见“OpenSourceBDM_S08_驱动”。

（三）、创建一个新工程

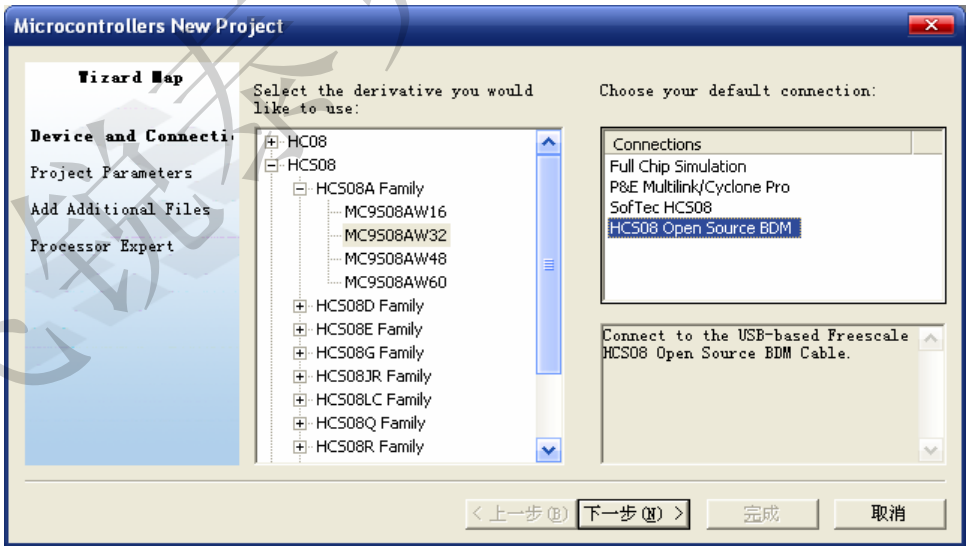
1) 开始>程序>Freescale CodeWarrior>CW for Microcontrollers V6.0选择CodeWarrior IDE。IDE 开始运行，出现Startup 窗口

- Create New Project ： 创建一个新工程
- Load Example Project ： 加载一个示例工程
- Load Previous Project ： 加载以前打开过的工程
- Run Getting started Tutorial ： 运行CodeWarrior 软件帮助文档
- Start Using CodeWarrior ： 返回CodeWarrior 主窗口



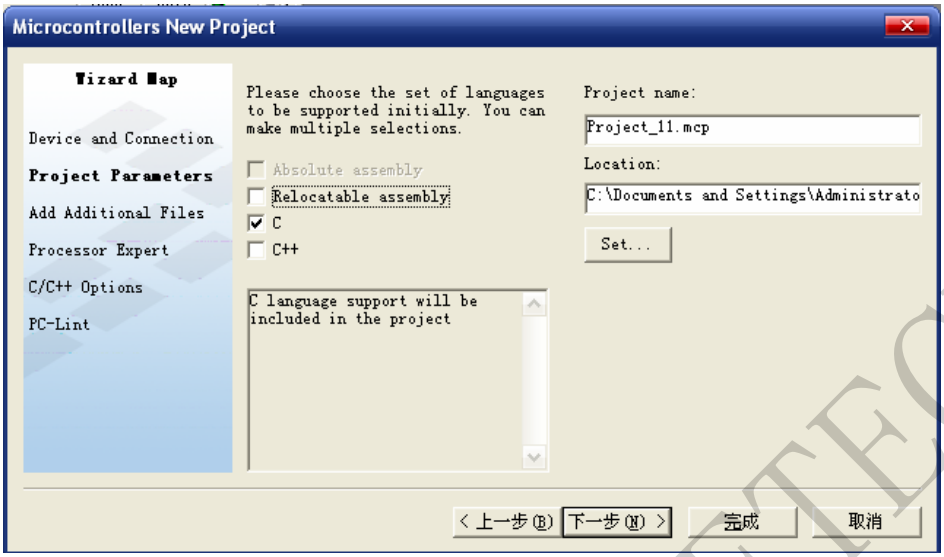
Startup 窗口

2）选择Create New Project ，出现Microcontrollers New Project 窗口，见新建工程向导1 。
Select the derivative you would like to use 对话框中选择所使用的单片机型号
Choose your default connection 对话框选择工程所使用的开发工具，选择每一种开发工具，在下面的显示栏中显示其相关信息。
Full Chip Simulation ：软件仿真
P&E Multilink/Cyclone Pro：P&E 公司设计的开发工具
SofTec HCS08：SofTec 公司设计的开发工具
HCS08 Open Source BDM：USB 型开源BDM 开发工具



新建工程向导1

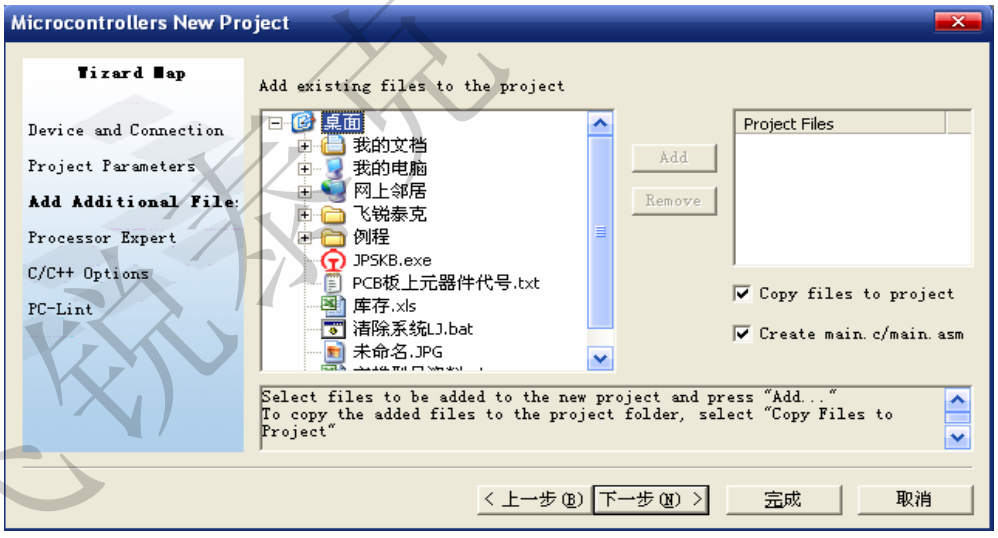
3）展开HCS08 下的HCS08D Family，选择MC9S08DZ64；“Choose your default connection ”对话框中选择HCS08 Open Source BDM；单击“下一步”出现新建工程向导2 窗口。在该对话框可以选择开发语言，这里我们选择C 语言开发应用程序。



新建工程向导2

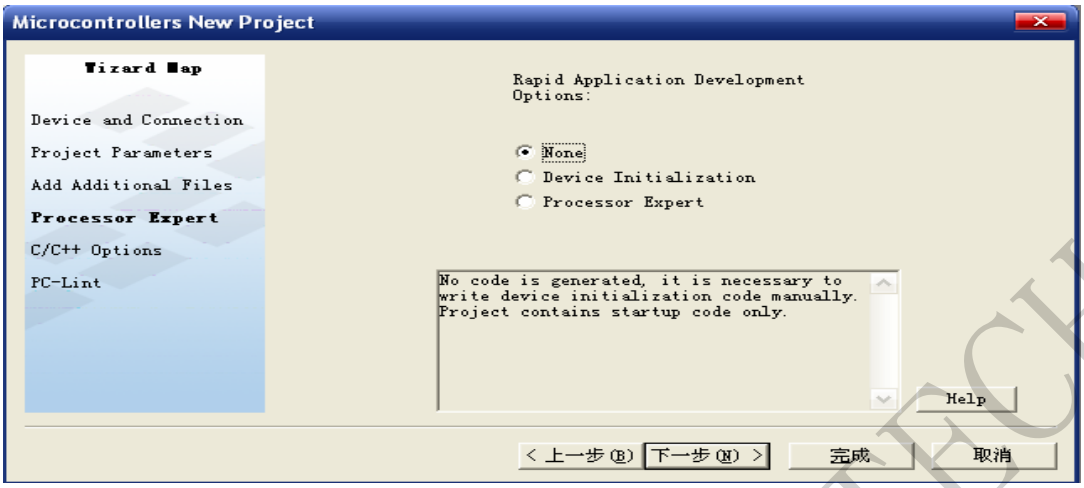
在Project name 对话框中输入新建工程名称；点击Location 对话框下的Set 按钮可以设置工程保存位置。(注：此时可以点击“完成”按钮，以下设置由CodeWarrior 自动完成。）

4）单击“下一步”出现新建工程向导3，在该步骤中，可以向工程中添加或移除文件 。



新建工程向导3

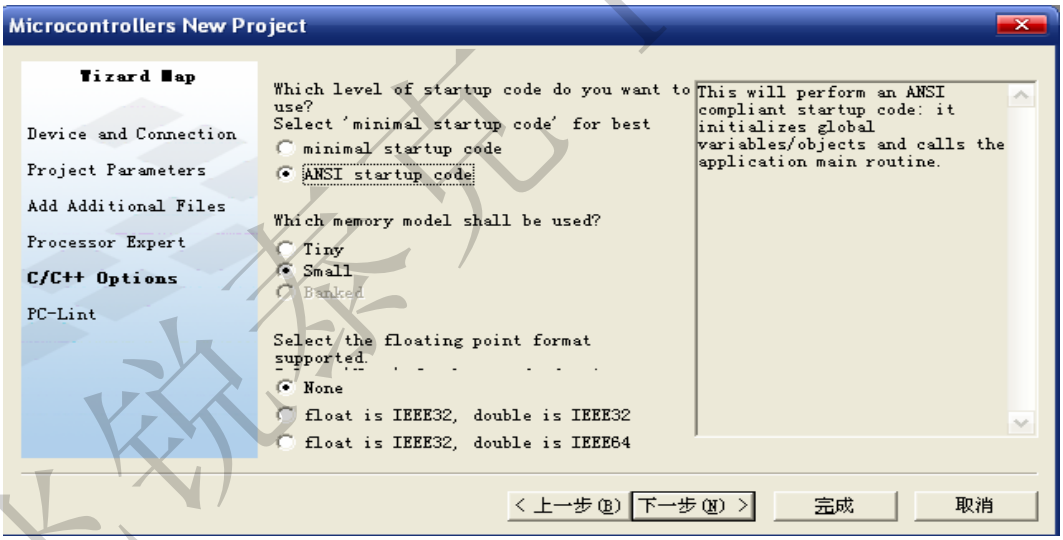
5）单击“下一步”出现新建工程向导4，用户可以选择“Device Initialization” 或“Processor Expert” 由软件自动完成中断向量，外围模块等初始化工作。这里我们选择“None” 。



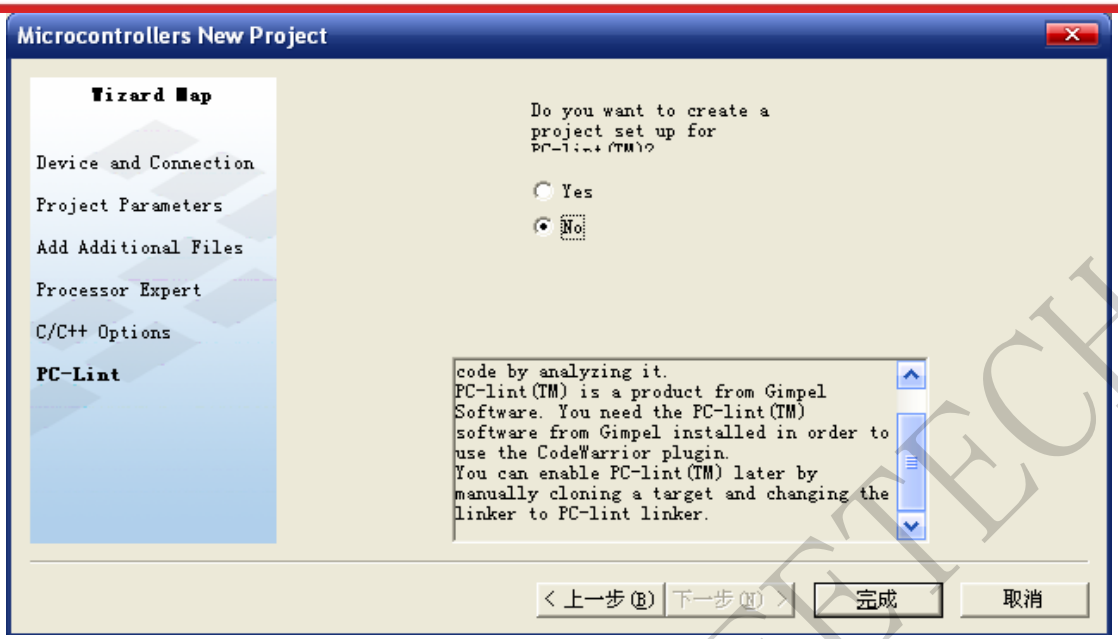
新建工程向导4

6）单击“下一步”出现新建工程向导5，选择“ANSI startup code”，“Small”，“None” ，单击下一步出现新建工程向导6。

在该对话框下可以选择startup 代码类型，建议选择minimal startup code ； 设置RAM存储模式，建议选择Tiny ； 设置是否支持浮点格式，建议选择None。详细介绍请参照参考资料文件夹下的“CW6.0开发环境介绍”。

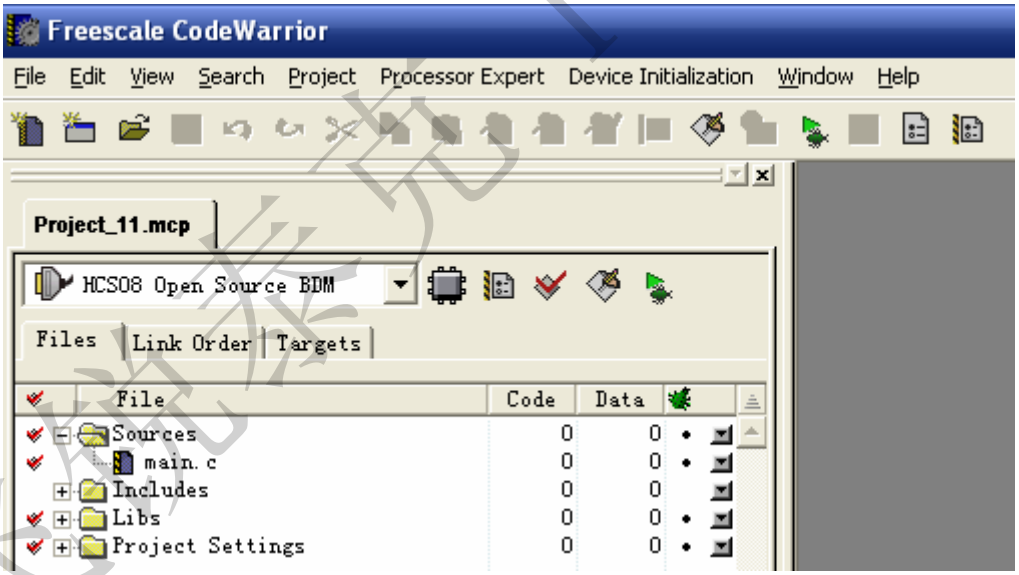


新建工程向导5



新建工程向导6

7）单击“完成”。IDE会自动生成工程文件。双击“main.c”文件可对其进行修改。其他文件功能介绍请参考参考资料文件夹下的“CW6.0开发环境介绍”。



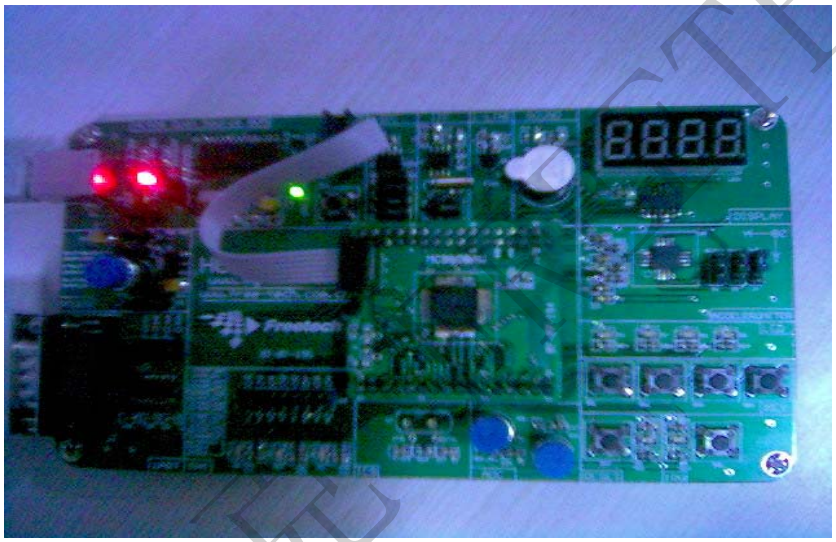
CW6.0编辑窗口

（四）、调试新建工程

1）连接好硬件环境。USB数据线将学习板和PC的USB口连接起来。PC会把学习板上的HCS08 Open Source BDM部分作为一个USB设备，在PC的硬件设备中可以看到该设备－LibUSB-Win32 Devices。

设备管理器窗口

此时学习板上HCS08 Open Source BDM部分的红色LED-D2亮指示USB可靠连接；绿色LED-D1亮指示PC与HCS08 Open Source BDM已建立连接。

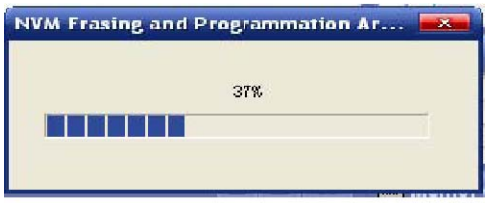


硬件环境（更换图片）

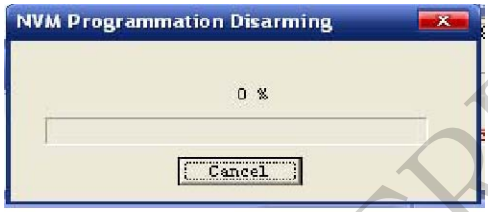
2）CodeWarrior IDE 调试过程：编译—>汇编—>连接—>程序下载—>调试 。

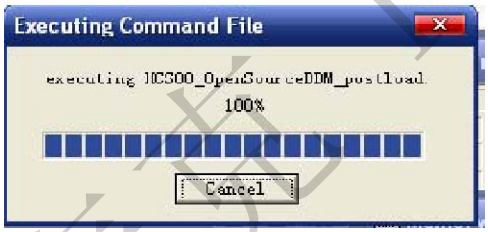
单击 按钮会依次出现如下对话框。

擦除提示对话框（单击“OK”）

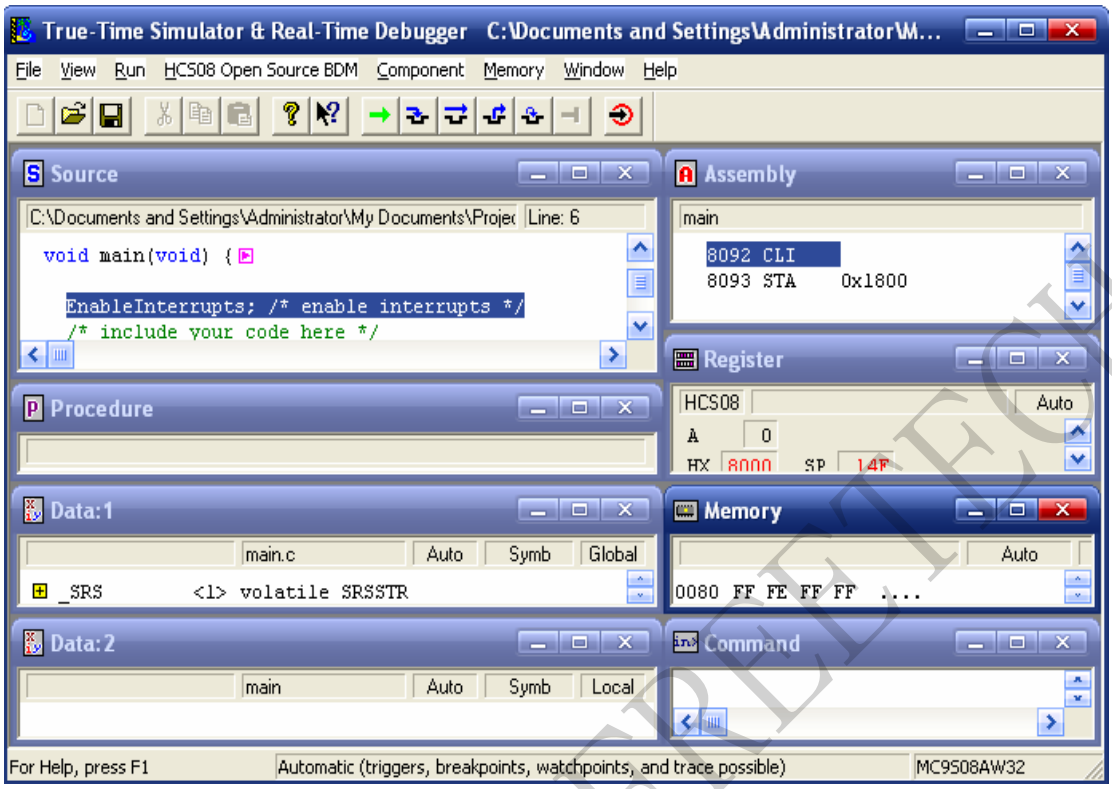


代码下载对话框





执行BDM 命令



调试环境窗口

3）熟悉调试窗口。调试窗口是一个用户可自定义的调试环境。

其中：

Source：源文件窗口，在该串口单击鼠标右键可以添加/取消断点，添加/取消 触发点等操作，支持最多3个断点。

Assembly ：汇编窗口，该窗口为Source 中C 语言程序对应的汇编程序，在该 窗口单击鼠标右键可以添加/取消断点，添加/取消触点等操作。

Register ： 寄存器窗口，显示CPU 各寄存器的内容，双击可修改寄存器的值。

Memory ： 存储器窗口，显示整个存储器的内容，双击某个存储单元可修改该存储单元的内容。 在该窗口单击鼠标右键输入地址，可查看该地址的存放的数据。

Command ： 命令窗口，显示整个调试系统的通信命令。

Procedure： 过程窗口，在该窗口显示整个工程所使用到的函数。

Data:1，Data:2 ：数据窗口，该窗口显示各种变量的存储地址，内容。在窗口 单击右键，选择“Add Expression…”可以增加要显示的变量。

单击该按钮，程序将开始运行。如有断点，程序运行到断点处停止。快捷键F5。

程序停止状态下，单击该按钮，程序single step 方式运行，会停止在子函数内部。快捷键F11 。

程序停止状态下，单击该按钮，程序step over 方式运行，不会停止在子函数内部。快捷键F10。

当single step 运行方式下，进入某一个子函数内部后，单击该按钮，程序执行完该子函数，停止在紧随该子函数的外部的下一条语句。快捷键shift+F11 。

程序停止状态下，单击该按钮，程序assembly step 方式运行，该方式与single step 方式相似，但是以汇编语句为一单步。捷键ctrl+F11 。

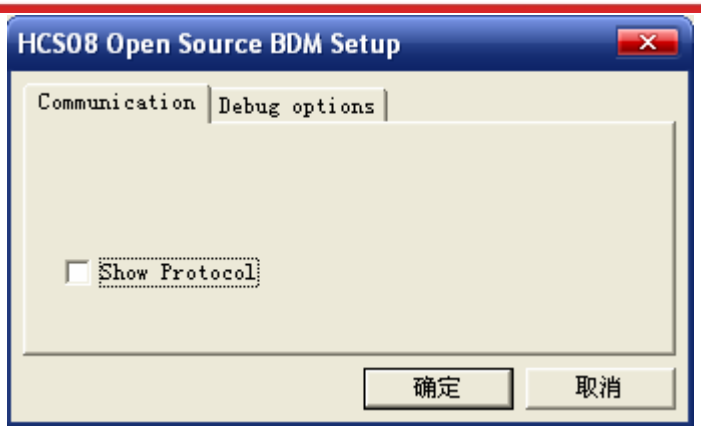
当程序运行时，单击该按钮，程序当停止。

单击该按钮，程序将复位。

（五）、常见问题及技巧

1）IDE 提供了很完善的帮助文档，用户可以根据遇到的问题在帮助文档中搜索相关问题。

2）在使用Open Source BDM 调试工具时，出现以下对话框，说明调试工具与目标板未通信上，应检查硬件连接是否正确；



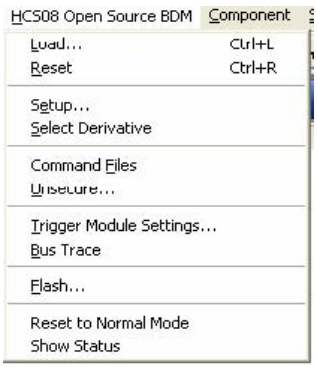
断开USB数据下，按下HCS08 Open Source BDM部分的按键SW5，然后连接好USB数据线，几秒种后松开按键。

3） 在使用Open Source BDM 调试工具调试时，程序处于运行状态时，不要单击 复位按钮。如果希望复位程序，应先按 停止按钮，然后按 复位按钮。

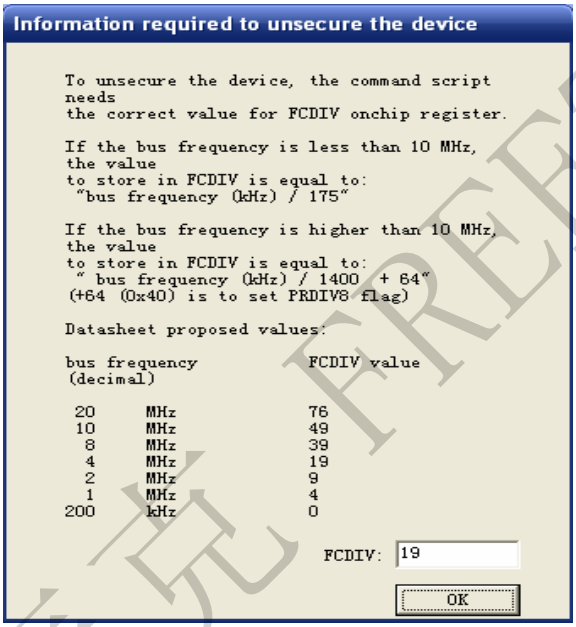
4） 半导体8 位单片机的复位向量为0xFFFE~0xFFFF，该两个字节放置的是单片机复位后要跳转到的程序地址。单片机复位后执行0xE07B 开始的程 序，也就是start08.c 程序。当单击 按钮进入调试窗口时，start08.c 会被自动执行，程序指向main() 函数的入口。在调试窗口中，单击 复位按钮，程序会真正指向复位向量保存的跳转地址。

5） 当用户在开发过程中，有意或无意的将flash 进行了加密，那么用户在下次调试过程中会出现如下错误窗口：

解决方法：单击确定，进入调试窗口；在调试窗口下，单击菜单项HCS08 Open Source BDM ，然后选择Unsecure… ，即可对加密的单片机进行解密。



选择Unsecure



单击“ok”开始解密（擦除芯片）

第4章 实验例程

实验一 复位及看门狗

1、实验功能介绍

1、1 特征

- 多种复位源，系统配置灵活，可靠
  - 上电检测
  - 低电压检测
  - 复位引脚上施加复位信号
  - 看门狗复位
  - 非法操作码复位
  - 后台调试复位
- 复位状态寄存器(SRS)记录最近一次发生的复位源
- 每个片上外围功能模块有独立的中断向量

1、2 MCU复位

复位MCU操作为用户提供一种预知的设置状态。复位期间，大多数控制和状态寄存器被强制复位到初始值，复位向量装载到程序计数器中。片上外围模块禁止，I/O初始化为通用的高阻抗输入，且内部上拉取消。中断禁止。用户程序初始化程序可以初始化堆栈指针SP和系统控制配置。

为了与老型号芯片兼容注意：SP复位之后为0x00FF。

7个复位源：

- 上电复位（POR）
- 低电压复位（LVD）
- 看门狗复位（COP）
- 非法操作码复位
- 后台调试强制复位
- 复位引脚
- 时钟发生器锁定失败和时钟丢失复位

除了后台调试强制复位，其他每个复位源与复位状态寄存器相关联。MCU进入复位状态，内部时钟发生器(ICG)模块打开，工作于自时钟模式，时钟频率为f_{self_reset}，复位引脚被MCU拉低，持续34个内部总线时钟周期，然后该引脚被释放且内部上拉。再经过38个周期，采样该引脚，决定是否是该引脚引发的复位。也就是说用户通过复位引脚复位单片机，那么将复位引脚拉低，并且低电平维持时间至少（34+38）个总线周期。

1、3 看门狗（COP）

当用户程序“跑飞”，看门狗产生一个强制系统复位。为了阻止看门狗复位，用户程序

必须定期的复位COP计数器。

任何复位之后，SOPT寄存器中的COPE位＝1，使能看门狗复位。如果用户程序禁止看门狗复位功能，通过清除COPE位可以关闭看门狗。向SRS寄存器写任何值，COP计数器会复位。但该写操作并不会影响只读寄存器SRS内容。

用户程序可以使用COPE，COPCLKS和COPT的复位默认值，建议用户程序复位初始化过程中写一次SOPT寄存器，可以避免在用户程序运行过程中意外修改它们。而最初的对SOPT写操作将复位COP计数器。

写SRS寄存器操作（清看门狗操作）不能放在中断服务程序（ISR）中，这是因为尽管主程序已经“跑飞”，但ISR仍会定期执行，从而失去看门狗监测的功能。

当MCU处于后台调试模式下，COP计数器不会递增。

当MCU处于处于STOP模式，看门狗时钟源选择的是总线时钟的情况下，COP计数器会停止计数。一旦MCU脱离后台调试模式，COP计数器继续递增式计数。

看门狗时间间隔可选择总线时钟的2的13次方或2的18次方个周期。

1、4 相关寄存器

1、4、1 系统复位状态寄存器：SRS($1800)

该寄存器的 7 个只读位用于表明最近一次发生复位的复位源。当调试主机强制 MCU 复位，那么 SRS 不会置位。写该寄存器操作会清除看门狗计数器，而不会影响该寄存器的内容。

	7	6	5	4	3	2	1	0
R	POR	PIN	COP	ILOP	ILAD	LOC	LVD	0
W	Writing 0x55, 0xAA to SRS address clears COP watchdog timer.							
POR:	1	0	0	0	0	0	1	0
LVD:	u	0	0	0	0	0	1	0
Any other reset:	0	Note(1)	Note(1)	Note(1)	Note(1)	0	0	0

1 Any of these reset sources that are active at the time of reset entry will cause the corresponding bit(s) to be set; bits corresponding to sources that are not active at the time of reset entry will be cleared.

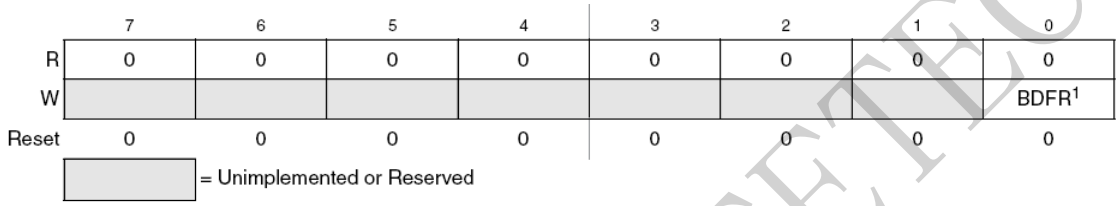
Figure 5-3. System Reset Status (SRS)

7 POR	上电复位 0：不是 POR 引起复位 1：POR 导致复位
6 PIN	外部复位引脚 0：不是外部复位引脚引起复位 1：外部复位引脚导致复位
5 COP	看门狗复位 0：不是看门狗引起复位 1：看门狗引起复位
4 ILOP	非法操作码复位 0：不是非法操作码引起复位 1：非法操作码导致复位
3 ILAD	非法地址 0：不是非法地址引起中断

	1：非法地址导致中断
2 IOC	外部时钟发生器模块复位 0：不是外部时钟发生器引起复位 1：外部时钟发生器导致复位
1 LVD	低电压检测复位 0：不是低电压检测模块或上电复位引起复位 1：低电压检测或上电导致复位

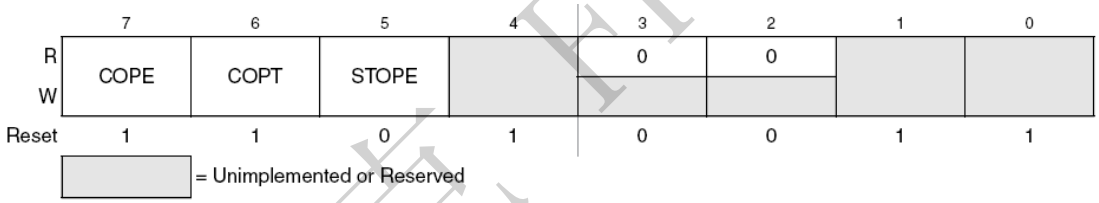
1、4、2 系统后台调试强制复位寄存器：SBDFR（$1801）

该寄存器只包含一个只写位。像 WRITE_BYTE 后台调试指令必须用于写 SBDFR 寄存器。



0 BDFR	后台调试强制复位：通过后台调试命令写 1，强制 MCU 复位。 该位不能被用户程序写。

1、4、3 系统选项寄存器：SOPT（$1802）



该寄存器在任何时候均可以读取。复位之后，只允许写一次。用户在复位之后初始化过程中，应写一次该寄存器，尽管写入的值与该寄存器复位之后相同。

7 COPE	看门狗使能位 0：关闭看门狗；1：看门狗使能
6 COPT	看门狗溢出时间 0：较短的溢出时间间隔（2 的 13 次方个总线周期）；1：较长的溢出时间间隔（2 的 18 次方个总线周期）
5 STOPE	STOP 模式使能：如果 STOPE 设置为 0，而执行 STOP 指令，那么会产生一个非法操作码复位。 0：禁止进入 STOP 模式；1：允许进入 STOP 模式

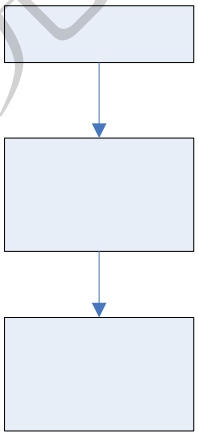
Control Bits		Clock Source	COP Window[1] Opens (COPW = 1)	COP Overflow Count
COPCLKS	COPT[1:0]			
N/A	0:0	N/A	N/A	COP is disabled
0	0:1	1 kHz	N/A	2^5 cycles (32 ms[2])
0	1:0	1 kHz	N/A	2^8 cycles (256 ms[1])
0	1:1	1 kHz	N/A	2^{10} cycles (1.024 s[1])
1	0:1	Bus	6144 cycles	2^{13} cycles
1	1:0	Bus	49,152 cycles	2^{16} cycles
1	1:1	Bus	196,608 cycles	2^{18} cycles

[1] Windowed COP operation requires the user to clear the COP timer in the last 25% of the selected timeout period. This column displays the minimum number of clock counts required before the COP timer can be reset when in windowed COP mode

	7	6	5	4	3	2	1	0
R	COPCLKS[1]	COPW[1]	0	ADHTS	0	MCSEL		
W								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

2、流程图

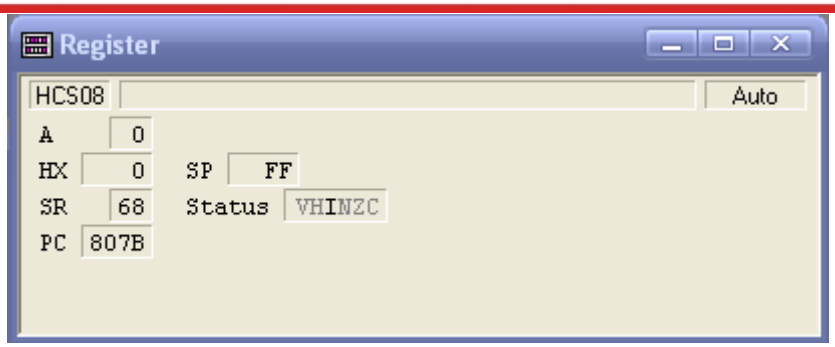


3、实验现象

3、1 打开源代码文件下的复位实验下的复位实验.mcp。

3、2 单击  进入调试环境窗口。进入调试窗口后，程序停止在“EnableInterrupts; /* enable interrupts */”语句。该过程其实已经执行完start08.c程序。

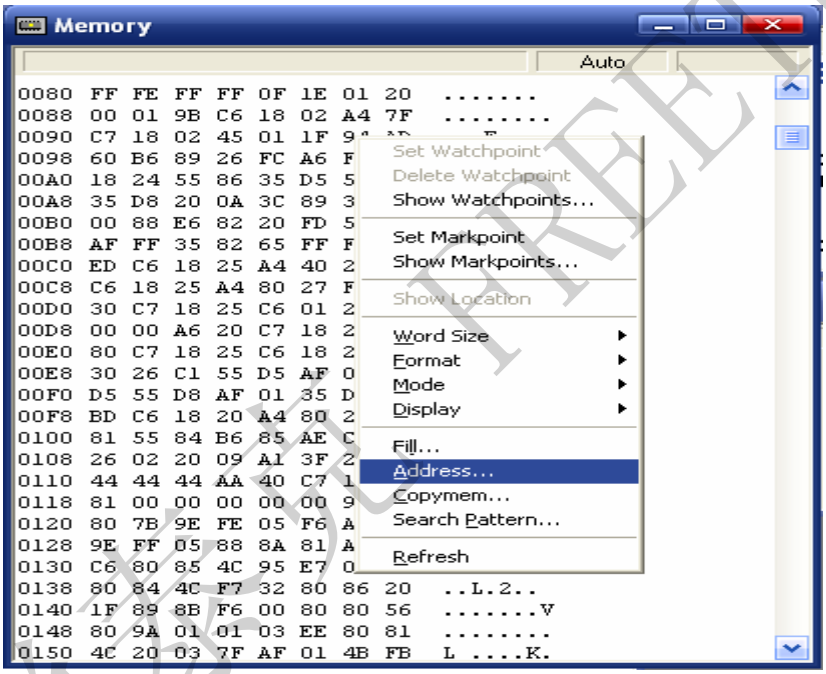
3、3 在调试窗口中单击强制复位按钮  ，观测 Register 窗口：



寄存器窗口

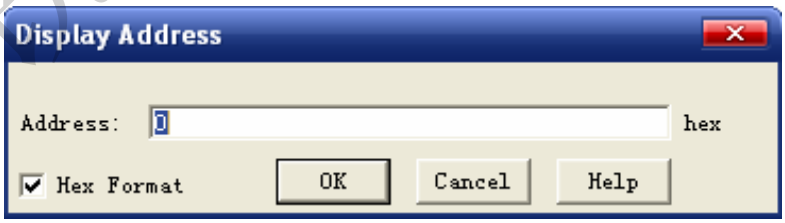
从上面的窗口我们可以看出 MCU 复位之后，SP 的值为 0XFF，PC 指向复位向量$807B，CPU 的条件寄存器中的 I 位置位（禁止全局中断）。

在 Memory 窗口单击右键，出现如下对话框：



Memory 窗口

选择 Address 出现如下对话框：



在上面的地址中输入“1800”，观测记录复位源寄存器 SRS 的值。此时 SRS 的值为 0x00，这是因为我们刚才的操作是执行的后台调试强制复位操作，该操作会将该寄存器清零。

3、4 观测复位按键的作用

按一下学习板上的复位按键 SW7，我们会观测到 SRS 寄存器的值变为 0x40，表明 MCU 复位源为复位引脚触发的复位。同时 MCU 已经运行起来了。

4、程序代码

参见源代码下的“复位实验”。

实验二 外部中断实验

1、实验功能介绍

1、1 中断介绍

中断是指保存当前CPU状态和寄存器，执行中断服务程序，中断服务程序执行完毕后恢复CPU状态，MCU从被中断打断的地方继续执行。IRQ引脚的边沿事件或计数器溢出事件等产生硬件中断。用户应用程序执行特定指令可以产生软件中断（SWI），后台调试模块在一定条件下也可以产生SWI。

当某一中断事件发生，与该事件相对应的只读标志将被置位。如果该中断未使能，那么CPU不会对该中断作出反映。CCR寄存器中的全局中断屏蔽位I在复位之后为1，屏蔽掉所有中断事件。在用户程序完成堆栈指针和其他设置之后，清除I，允许CPU响应中断事件。

当CPU接受到一个有效的中断请求，在响应之前，它会执行完当前运行的指令，然后逐周期执行以下过程：

- 保存 CPU 寄存器到堆栈中
- 设置 CCR 寄存器中的 I=1，禁止全局中断
- 获取当前所有挂起的待处理中断事件中最高优先级中断向量
- 预取中断向量的地址排列到指令队列中

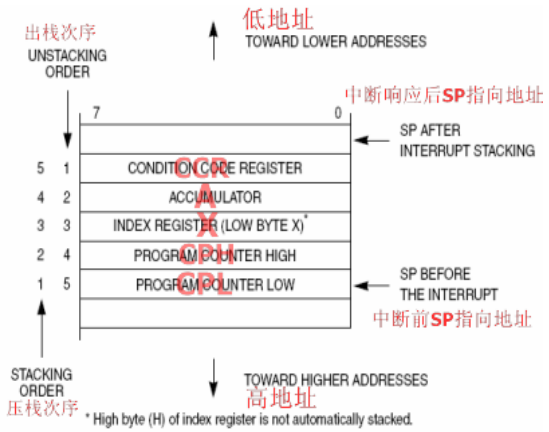
当CPU响应中断后，I自动置为1，避免中断嵌套。通常，ISR执行完毕后，I恢复为0（CCR从堆栈弹出）。如果在ISR中，清除I（即设置I=0），那么CPU有可能在当前中断事件未完成之前响应其他中断事件，即中断潜套。中断潜套可能对用户程序调试带来麻烦。

当ISR执行到RTI指令，CCR，A，X，PC从堆栈恢复。

注意：为兼容MC68HC08，H寄存器不会自动压栈和出栈。用户程序完成对H的压栈出栈。

当然，用户可在自己的中断服务程序一开始将 I 清零，从而使能全局中断。那么当前运行的中断服务程序可以被其他中断打断，CPU 运行新的中断服务程序，从而形成中断嵌套。用户在使用中断嵌套时会对调试带来不便，故建议避免使用中断嵌套。

在中断服务程序的开始，用户程序应清掉产生该中断的标志位，其目的是同一中断源产生另外一个中断事件能够被记录，并在完成当前中断服务程序之后能够得到CPU的响应。



压栈、出栈图

Address (High/Low)	Vector	Vector Name
0xFFC0:0xFFC1	ACMP2	Vacmp2
0xFFC2:0xFFC3	ACMP1	Vacmp1
0xFFC4:0xFFC5	MSCAN Transmit	Vcantx
0xFFC6:0xFFC7	MSCAN Receive	Vcanrx
0xFFC8:0xFFC9	MSCAN errors	Vcanerr
0xFFCA:0xFFCB	MSCAN wake up	Vcanwu
0xFFCC:0xFFCD	RTC	Vrtc
0xFFCE:0xFFCF	IIC	Viic
0xFFD0:0xFFD1	ADC Conversion	Vadc
0xFFD2:0xFFD3	Port A, Port B, Port D	Vport
0xFFD4:0xFFD5	SCI2 Transmit	Vsci2tx
0xFFD6:0xFFD7	SCI2 Receive	Vsci2rx
0xFFD8:0xFFD9	SCI2 Error	Vsci2err
0xFFDA:0xFFDB	SCI1 Transmit	Vsci1tx
0xFFDC:0xFFDD	SCI1 Receive	Vsci1rx
0xFFDE:0xFFDF	SCI1 Error	Vsci1err
0xFFE0:0xFFE1	SPI	Vspi
0xFFE2:0xFFE3	TPM2 Overflow	Vtpm2ovf
0xFFE4:0xFFE5	TPM2 Channel 1	Vtpm2ch1
0xFFE6:0xFFE7	TPM2 Channel 0	Vtpm2ch0
0xFFE8:0xFFE9	TPM1 Overflow	Vtpm1ovf
0xFFEA:0xFFEB	TPM1 Channel 5	Vtpm1ch5
0xFFEC:0xFFED	TPM1 Channel 4	Vtpm1ch4
0xFFEE:0xFFEF	TPM1 Channel 3	Vtpm1ch3
0xFFF0:0xFFF1	TPM1 Channel 2	Vtpm1ch2
0xFFF2:0xFFF3	TPM1 Channel 1	Vtpm1ch1
0xFFF4:0xFFF5	TPM1 Channel 0	Vtpm1ch0
0xFFF6:0xFFF7	MCG Loss of lock	Vlol
0xFFF8:0xFFF9	Low-Voltage Detect	Vlvd
0xFFFA:0xFFFB	IRQ	Virq
0xFFFC:0xFFFD	SWI	Vswi
0xFFFE:0xFFFF	Reset	Vreset

中断向量表

1、2 外部中断引脚

外部中断由IRQ的状态和控制寄存器IRQSC管理。当IRQ使能，同步逻辑电路监测该引脚是否边沿事件或边沿－电平事件发生。IRQ可以用于唤醒STOP模式下的MCU。

1、2、1 引脚配置选项

设置IRQSC寄存器中的IRQPE＝1，那么该引脚作为IRQ输入功能。

IRQMOD寄存器决定引脚中断模式：沿触发中断或边沿－电平触发中断。

IRQEDG寄存器决定引脚中断信号边沿或电平的极性。

BIH和BIL指令可用于检测引脚信号电平。

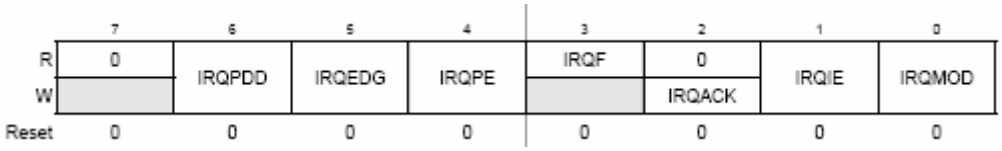
1、2、2 边沿和电平触发模式

IRQMOD控制位设置外部中断输入信号边沿有效还是电平有效。

在边沿－电平有效模式下，当检测到有效的边沿信号，IRQF标志就会置位，如果该引脚一直处于有效的触发电平状态，那么该中断标志会连续被置位。

1、3 相关寄存器

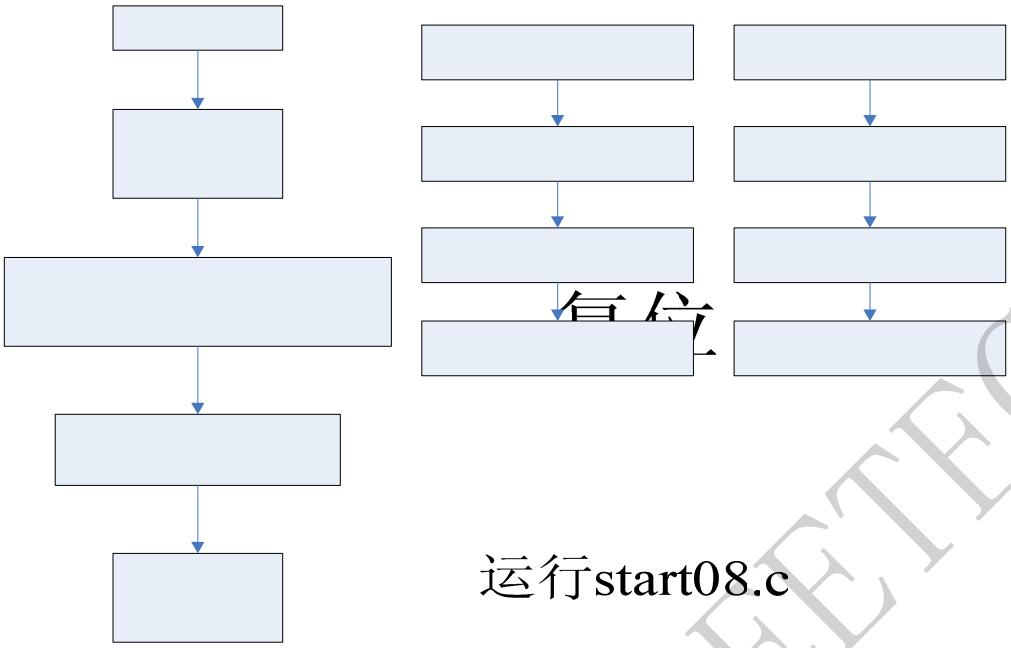
外部中断引脚状态和控制寄存器：IRQSC（$001C）



	7	6	5	4	3	2	1	0
R	0	IRQPDD	IRQEDG	IRQPE	IRQF	0	IRQIE	IRQMOD
W						IRQACK		
Reset	0	0	0	0	0	0	0	0

5 IRQEDG	IRQ 边沿选择：可读写；该位用于选择边沿极性。与 IRQMOD 位配合使用 0：下降沿或下降沿/低电平有效（电平方式） 1：上升沿或上升沿/高电平有效（电平方式）
6 IRQPDD	口线驱动使能，减少干扰； 0：使能驱动；IRQPE=1; 1：不使能驱动；IRQPE=1;
4 IRQPE	IRQ 管脚使能：可读写 0：关闭 IRQ 功能 1：IRQ 使能
3 IRQF	IRQ 中断标志 0：无 IRQ 请求 1：有 IRQ 请求
2 IRQACK	IRQ 确认：只写 写 1 清除 IRQF，但在边沿－电平触发模式下，如果 IRQ 引脚处于中断有效电平状态，向 IRQACK 写 1 不会清除 IRQF
1 IRQIE	IRQ 中断使能：可读写 0：关闭 IRQ 中断功能 1：IRQ 中断使能
0 IRQMOD	IRQ 触发方式：可读写 0：IRQ 边沿触发方式 1：高电平或低电平触发方式

2、流程图



3、实验现象

3、1 打开源代码中的 IRQ 实验.mcp 文件。

3、2 调试环境下单击运行按键，学习板上的 D5 点亮。

3、3 按一下学习板上的 SW5 按键，会使 D5 状态例如变化。

3、4 用户可对按键加消抖处理。

4、程序代码

参见源代码中的 IRQ 实验。

实验三 实时时钟实验

1．实验功能介绍

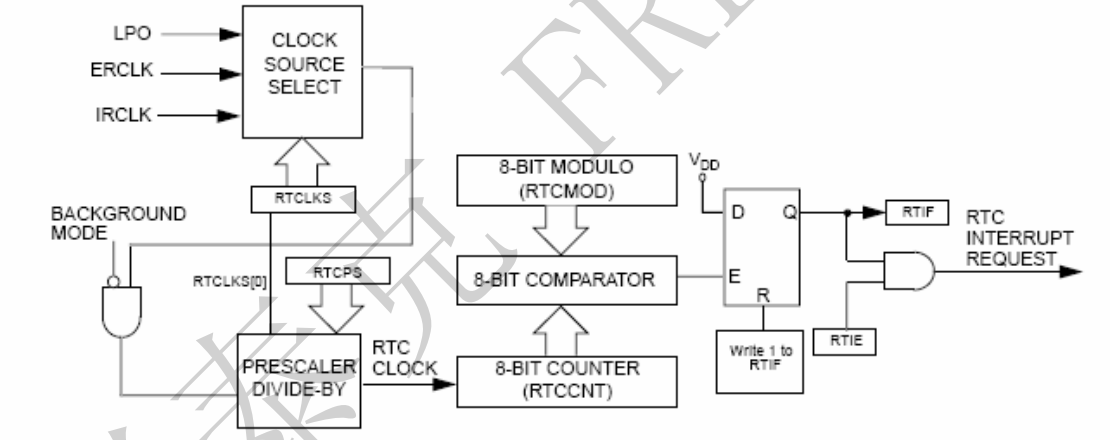
1、1 实时时钟中断

实时时钟中断可以用于产生定期中断事件。该实时中断时钟源可选择为 1kHz 内部时钟源 32k 源或外部时钟。在低功耗的应用中很多。1kHz 内部时钟源是一个独立于系统总线的时钟源，只用于 RTI 模块，COP 模块（只对某些 MCU）。如果使用外部时钟源，必须对相关控制位进行设置。寄存器 SRTISC 中的 RTICLKS 位用于选择时钟源。

实时中断在正常模式，等待模式或 STOP3 模式下可以使用内部时钟源或外部时钟源。如果在 STOP3 模式中使用外部晶振，必须设置 OSCSTEN=1，RANGE=0。只有选择内部时钟源，才能唤醒处于 STOP2 模式下的 MCU。

SRTISC 寄存器包括一个只读状态标志，一个只写确认位，3 个可读写控制位（用于选择 7 种唤醒间隔之一，同时该 3 个控制位用于关闭实时时钟的时钟源。

RTI 有自己的中断使能位 RTIE。最大可以 26.6 分钟唤醒。



1、2 系统实时时钟状态和控制寄存器－SRTISC($1808)

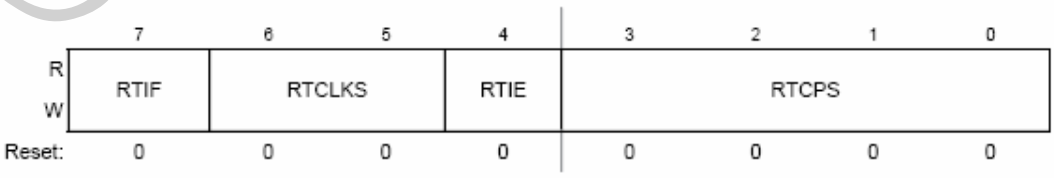


Figure 15-3. RTC Status and Control Register (RTCSC)

7 RTIF	实时时钟中断标志，只读 0：定时唤醒计数器未溢出 1：定时唤醒计数器溢出，产生中断，该位写一清零

6 RTICLKS1	
5 RTICLKS0	实时时钟中断时钟选择 00：内部 1-kHz 振荡；01：外部时钟；10，11;32K 内部源
4 RTIE	实时时钟中断使能 0：禁止实时时钟中断；1：实时时钟中断使能
3：0 RTCPS[3:0]	实时时钟定时选择，RTCPS=0 时关闭实时时钟计数功能 见下表

Table 15-3. RTC Prescaler Divide-by values

	RTCPS															
RTCLKS[0]	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	OFF	2^3	2^5	2^6	2^7	2^8	2^9	2^{10}	1	2	2^2	10	2^4	10^2	$5x10^2$	10^3
1	OFF	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	10^3	$2x10^3$	$5x10^3$	10^4	$2x10^4$	$5x10^4$	10^5	$2x10^5$

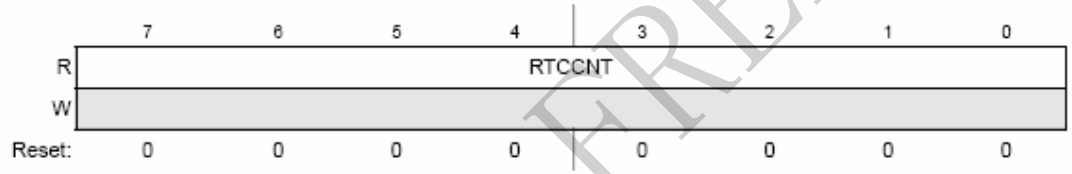


Figure 15-4. RTC Counter Register (RTCCNT)

实际的计数值，复位或初始化后为零。

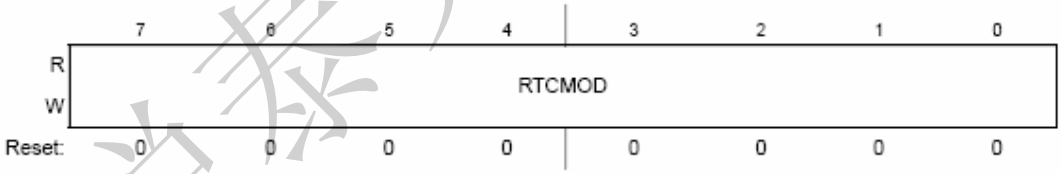
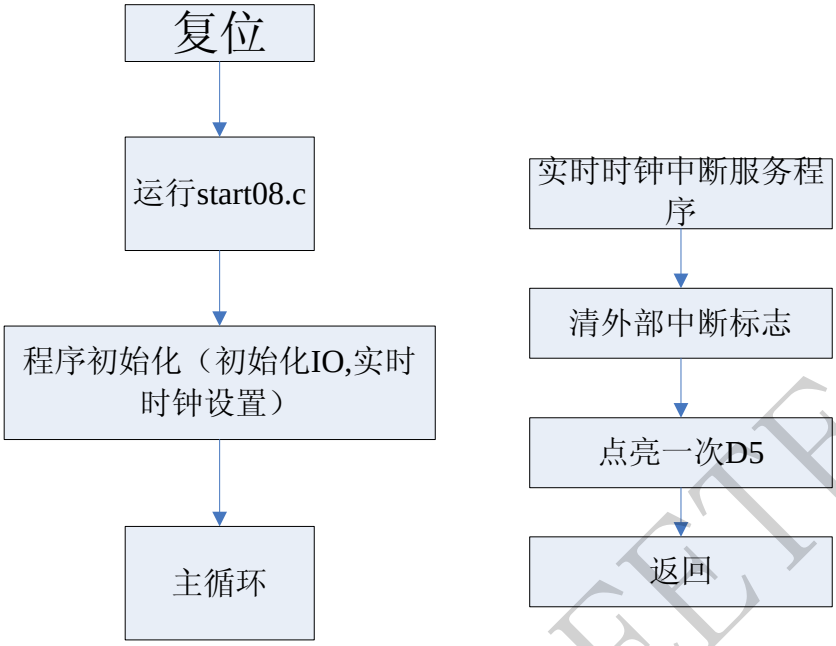


Figure 15-5. RTC Modulo Register (RTCMOD)

溢出的模数设定值，设定溢出中断值。计数值计到该数时会启动中断。

2、流程图



3、实验现象

3、1 打开 RTI 实验.mcp。

3、2 调试环境下单击全速运行按键，D5 会一秒种闪一次。蜂咛器尖叫。

4、程序代码

参见 RTI 实验。

实验四 低电压检测实验

1、实验功能介绍

1、1 低压检测功能介绍

15	P	Low-voltage detection threshold — high range V_{DD} falling V_{DD} rising	V_{LVDH}	4.2 4.3	4.3 4.4	4.4 4.5	V
16	P	Low-voltage detection threshold — low range V_{DD} falling V_{DD} rising	V_{LVDL}	2.48 2.54	2.56 2.62	2.64 2.7	V
17	P	Low-voltage warning threshold — high range V_{DD} falling V_{DD} rising	V_{LVWH}	4.2 4.3	4.3 4.4	4.4 4.5	V
18	P	Low-voltage warning threshold — low range V_{DD} falling V_{DD} rising	V_{LVWL}	2.48 2.54	2.56 2.62	2.64 2.7	V

低电压检测系统用于保护处于低电压条件下的存储器中数据和在电压波动条件下控制 MCU 系统状态。该系统包括上电复位（POR）和低电压检测(LVD)电路。

SPMSC1 寄存器中的 LVDE＝1，LVD 使能。

SPMSC2 寄存器中的 LVDV 位选择跌落电压。

当 MCU 处于 STOP 模式下，LVDSE 未设置，那么 LVD 功能关闭。

当 LVDSE 和 LVDE 位都被置位，那么 MCU 不能进入 STOP2 模式。

1、2 上电复位

上电或电源跌落到 V_{POR} 以下，POR 电路将产生复位条件。当电源电压上升，LVD 电路使 MCU 处于复位状态，直到电源电压大于 V_{LVDL} 。所以上电复位之后，SRS 寄存器中的 POR 和 LVD 为均为 1。

1、3 LVD 复位

如果 LVDRE=1，那么 LVD 电路检测到低电压情况下将产生一个复位。直到电源电压大于 V_{LVDV} ，MCU 一直处于复位状态。LVD 复位或 POR 后，SRS 的 LVD 位均被置位。

1、4 LVD 中断

设置 LVDE=1，LVDIE=1，LVDRE=0，LVD 电路检测到低电压产生一个中断，LVDF 被置为 1。

1、5 低电压告警

LVD 系统有一个低电压报警标志 LVW，该标志用于表征电源电压接近但大于 LVD 电压。LVW 置位不会产生中断；用户程序可以设定两个电压报警值，V_{LVWH} 和 V_{LVWL} 。

1、2 相关寄存器介绍

1、2、1 系统电源管理状态和控制寄存器 1－SPMSC1（$1809）

	7	6	5	4	3	2	1[1]	0
R	LVDF	0	LVDIE	LVDRE[2]	LVDSE[2]	LVDE[2]		BGBE
W		LVDACK						
Reset	0	0	0	1	1	1	0	0

= Unimplemented or Reserved

7 LVDF	低电压检测标志，只读 1：表明低电压事件发生
6 LVDACK	低电压检测确认，只写 写 1 清除 LVDF
5 LVDIE	低电压检测中断使能控制位 0：中断禁止；1：中断使能
4 LVDRE	低电压检测复位使能控制位 0：LVDF 不产生硬件复位；1：LVDF=1 时产生强制复位
3 LVDSE	低电压检测是否在停止模式可用 0：STOP 模式下 LVD 禁止；1：STOP 模式下 LVD 可运行
2 LVDE	LVD 使能 0：LVD 关闭；1：LVD 使能
0 BGBE	能系缓冲使能：ADC 模块的每一通道上使能内部带隙电压控制位 0：关闭能系参考电压；1：使能能系参考电压

1、2、2 系统电源管理状态和控制寄存器 2－SPMSC2（$180A）

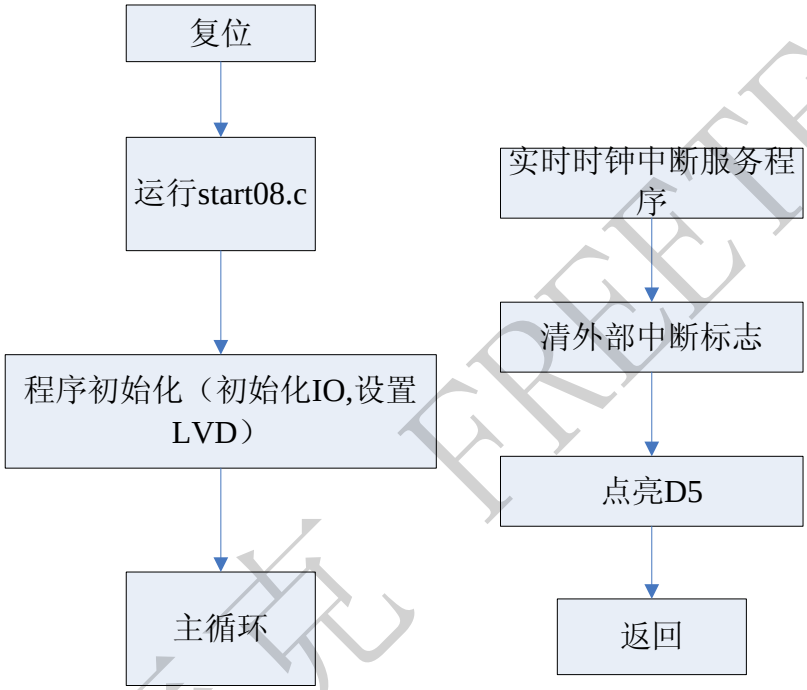
	7	6	5	4	3	2	1	0
R	LVWF	0	LVDV	LVWV	PPDF	0		PPDC[1]
W		LVWACK				PPDACK		
Power-on reset:	0[2]	0	0	0	0	0	0	0
LVD reset:	0[2]	0	U	U	0	0	0	0
Any other reset:	0[2]	0	U	U	0	0	0	0

= Unimplemented or Reserved　　U = Unaffected by reset

7 LVWF	低电压报警标志 0：未出现低电压；1：出现或出现过低电压
6 LVWACK	低电压报警确认 写 1 清除 LVWF
5 LVDV	低电压检测电压选择 0：V_{LVD}= V_{LVDL} 1：V_{LVD}= V_{LVDH}
4 LVWV	低电压报警电压选择 0：V_{LVW}= V_{LVWL} 1：V_{LVW}= V_{LVWH}
3 PPDF	该位表明 MCU 是从 STOP2 模式恢复 0：MCU 不是从 STOP2 模式唤醒

	1：MCU 从 STOP2 模式唤醒
2 PPDACK	清除 PPDF 位 写 1 清除 PPDF
0 PPDC	该位用于选择 STOP2 或 STOP3 模式 0：STOP3 模式允许 1：STOP2 模式允许

2、流程图



3、实验现象

3、1 打开 LVD 实验.mcp，把程序下载到学习板中，关闭调试窗口。

3、3 按一下复位按键 SW7，D5 并未点亮。

3、4 加入外部芯片电源，减小 VDD 到 1.8V 以下。然后逐渐加大 VDD，在 1.9V 左右，会观测到 RESET 部分的 D3 亮。这说明 VDD 在 1.9V 时，单片机处于复位状态，外部复位引脚被 MCU 拉低。继续加大 VDD，在 3.1V 作用，会观测到 D3 熄灭，D5 点亮。D5 的点亮就是低电压检测中断复位程序的工作。

4、程序代码

参见 LVD 实验。

实验五 熟悉存储器

1、实验功能介绍

1、1 寄存器地址和位操作

1、1、1 直接页寄存器

存储器的头 112 个字节为直接页寄存器，可进行位操作。

1、1、2 高页寄存器

存储器的 0X1800 开始为高页寄存器。

1、1、3 非易失性寄存器

FLASH 的 0XFFB0～0XFFBF 的 16 个字节。

1、2 RAM

RAM 的 0X100 以下可位操作

初始化 SP 的方法：

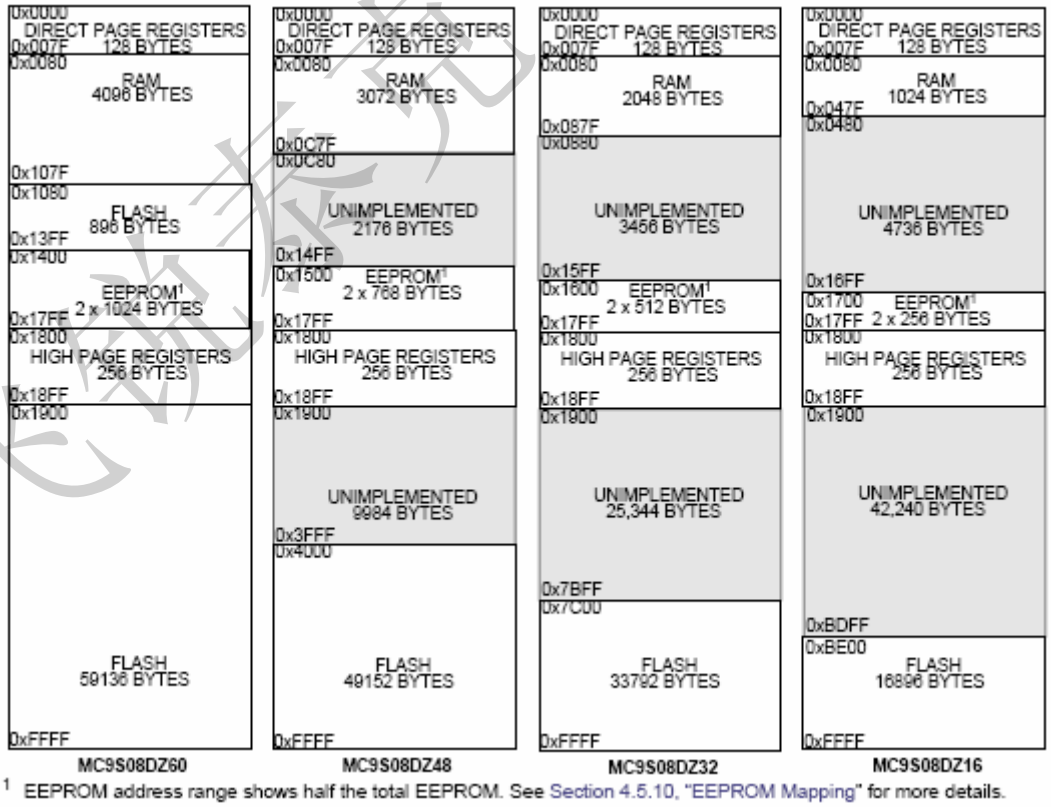
汇编语言：

```
LDHX #RamLast +1
TXS
```

C 语言：

```
asm(LDHX #RamLast+1);
asm(TXS);
```

1、3 FLASH



[1] EEPROM address range shows half the total EEPROM. See Section 4.5.10, "EEPROM Mapping" for more details.

1、3、1 **编程和擦除时间:**

任何编程或擦除命令执行之前，FLASH 时钟分频寄存器（FCDIV）必须将内部时钟设置设置好，即 f_{FCLK} 在 150kHz~200kHz 之间。这个寄存器只能写入一次，所以在复位初始化过程中设置好该寄存器。如果在 FSTAT 寄存器中的 FACCERR＝1，FCDIV 不允许写入。故在写入 FCDIV 之前，应查询 FACCERR 标志。执行时间如下：

Table 4-6. Program and Erase Times

Parameter	Cycles of FCLK	Time if FCLK = 200 kHz
Byte program	9	45 µs
Burst program	4	20 µs[1]
Sector erase	4000	20 ms
Mass erase	20,000	100 ms
Sector erase abort	4	20 µs[1]

[1] Excluding start/end overhead

注：1 页 512 字节

1、3、2 FLASH 时钟分频寄存器（FCDIV）：

该寄存器复位后只允许写一次

Figure 4-5. Flash and EEPROM Clock Divider Register (FCDIV)

7 DIVLD	0：FCDIV 寄存器在复位之后未被写过；擦除和编程操作禁止； 1：FCDIV 寄存器复位之后被写过；可以对 FLASH 进行擦除和编程
6 PRDIV8	预分频系数 8 允许位 0：总线时钟直接输入 FLASH 时钟驱动器 1：总线时钟 8 分频后再输入 FLASH 时钟驱动器
5 DIV[5:0]	分频系数：FLASH 时钟有总线时钟驱动。见下表

$$\text{if PRDIV8} = 0 \text{ — } f_{FCLK} = f_{Bus} \div ([DIV5:DIV0] + 1) \qquad \textit{Eqn. 4-1}$$

$$\text{if PRDIV8} = 1 \text{ — } f_{FCLK} = f_{Bus} \div (8 \times ([DIV5:DIV0] + 1)) \qquad \textit{Eqn. 4-2}$$

Table 4-8. Flash and EEPROM Clock Divider Settings

f_{Bus}	PRDIV8 (Binary)	DIV (Decimal)	f_{FCLK}	Program/Erase Timing Pulse (5 µs Min, 6.7 µs Max)
20 MHz	1	12	192.3 kHz	5.2 µs
10 MHz	0	49	200 kHz	5 µs
8 MHz	0	39	200 kHz	5 µs
4 MHz	0	19	200 kHz	5 µs
2 MHz	0	9	200 kHz	5 µs
1 MHz	0	4	200 kHz	5 µs
200 kHz	0	0	200 kHz	5 µs
150 kHz	0	0	150 kHz	6.7 µs

flash 时钟分频器设置

1、3、3 FLASH 选择寄存器（FOPT 和 NVOPT）

复位期间，NVOPT 的内容从 FLASH 复制到 FOPT。任意时刻都可对该寄存器读，但写操作无效。

	7	6	5	4	3	2	1	0
R	KEYEN	FNORED	EPGMOD	0	0	0	SEC	
W								
Reset	F	F	F	0	0	0	F	F

= Unimplemented or Reserved　　F = loaded from nonvolatile location NVOPT during reset

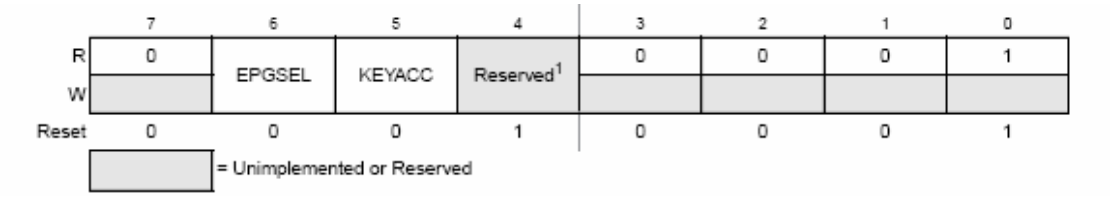
Figure 4-6. Flash and EEPROM Options Register (FOPT)

7 KEYEN	后门密匙机制允许位：通过硬件临时解密 0：后门密匙访问禁止；1：通过输入密匙值并与 NVBACKKEY－NVBACKEY+7 匹配，加密机制被屏蔽直到下次复位
6 FNORED	向量重指允许位 0：向量重指允许；1：禁止向量重指
EPGMOD	EPROM 模式位：0：4 个字节　1: 8 个字节方式
1：0 SEC0[1:0]	安全状态代码，见下表

SEC01:SEC00	Description
0:0	secure
0:1[1]	secure
1:0	unsecured
1:1	secure

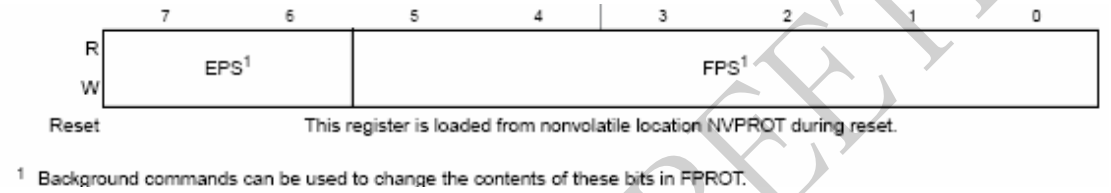
加密状态

1、3、4　**FLASH 配置寄存器（FCNFG）**



5 KEYACC	访问密匙允许位 0：写 0Xffb0～0Xffb7 操作被认为是擦除或编程 FLASH 的开始 1：写 0Xffb0～0Xffb7 操作被认为是比较密匙
6 EPGSEL	EEPROM 页选位 0：零页使用　　1：1 页使用

1、3、5 FLASH 保护寄存器（FPROT 和 NVPROT）



1 Background commands can be used to change the contents of these bits in FPROT.

EPS	Address Area Protected	Memory Size Protected (bytes)	Number of Sectors Protected
0x3	N/A	0	0
0x2	0x17F0 - 0x17FF	32	4
0x1	0x17E0 - 0x17FF	64	8
0x0	0x17C0–0x17FF	128	16

Table 4-14. Flash Block Protection

FPS	Address Area Protected	Memory Size Protected (bytes)	Number of Sectors Protected
0x3F	N/A	0	0
0x3E	0xFA00–0xFFFF	1.5K	2
0x3D	0xF400–0xFFFF	3K	4
0x3C	0xEE00–0xFFFF	4.5K	6
0x3B	0xE800–0xFFFF	6K	8
...	...	...	...
0x37	0xD000–0xFFFF	12K	16
0x36	0xCA00–0xFFFF	13.5K	18
0x35	0xC400–0xFFFF	15K	20
0x34	0xBE00–0xFFFF	16.5K	22
...	...	...	...
0x2C	0x8E00–0xFFFF	28.5K	38
0x2B	0x8800–0xFFFF	30K	40
0x2A	0x8200–0xFFFF	31.5K	42
0x29	0x7C00–0xFFFF	33K	44
...	...	...	...
0x22	0x5200–0xFFFF	43.5K	58
0x21	0x4C00–0xFFFF	45K	60
0x20	0x4600–0xFFFF	46.5K	62
0x19	0x4000–0xFFFF	48K	64
...	...	...	...

0x1B	0x2800–0xFFFF	54K	72
0x1A	0x2200–0xFFFF	55.5K	74
0x19	0x1C00–0xFFFF	57K	76
0x18–0x00	0x0000–0xFFFF	64K	88

1、3、6 FLASH 状态寄存器（FSTAT）

	7	6	5	4	3	2	1	0
R	FCBEF	FCCF	FPVIOL	FACCERR	0	FBLANK	0	0
W								
Reset	1	1	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 4-9. Flash and EEPROM Status Register (FSTAT)

7 FCBEF	FLASH 命令缓冲区空标志：用于发起命令。指示命令缓冲为空，所以一个新的命令可以执行（当执行快速编程）。清除方法：

	1） 向该位写 1 2） 一个快速编程命令传送给编程的阵列 0：命令缓冲区满；1：一个新的快速编程命令可以写入命令缓冲区
6 FCCF	FLASH 命令完成标志：当命令缓冲为空，没有命令执行时该位置位。当一个新的命令开始后，该位自动清除。 0：命令执行中；1：所有命令执行完成
5 FPVIOL	FLASH 保护冲突标志：清除该位方法为向该位写 1。 0：无保护冲突；1：试图擦除或编程一个受保护的区域，发生冲突
4 FACCERR	FLASH 访问错误标志：1）未遵循正确的命令顺序 1) FCDIV 未初始化而进行编程或擦除操作 2) 命令执行过程钟 MCU 进入 STOP 模式 清除该位的方法：向该位写 1。 0：无访问错误发生 1：发生一个访问错误
2 FBLANK	FLASH 查空核实标志 0：查空命令之后，FCCF=1，FBLANK=0 表明 FLASH 阵列未完全擦除； 1：查空命令之后，FCCF=1, FBLANK=1 表明 FLASH 阵列完成擦除

1、3、7 FLASH 命令寄存器（FCMD）

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W	FCMD7	FCMD6	FCMD5	FCMD4	FCMD3	FCMD2	FCMD1	FCMD0
Reset	0	0	0	0	0	0	0	0

Command	FCMD	Equate File Label
Blank check	$05	mBlank
Byte program	$20	mByteProg
Byte program — burst mode	$25	mBurstProg
Page erase (512 bytes/page)	$40	mPageErase
Mass erase (all FLASH)	$41	mMassErase

FLASH 操作命令

1、3、8 编程擦除命令执行过程

在执行下面操作之前，应清除所有的与 flash 有关的错误标志，并设置好 FCDIV。

执行步骤：

1. 向 FLASH 阵列的一个地址写入一个数据。这次写的地址和数据信息被闩锁进 FLASH 接口。该预写步骤是任何命令序列的第一个步。
   对于擦除和查空命令，数据值并不重要。
   对于页擦除命令，地址应为要擦除的 512 字节 FLASH 块中任意一个位置地址。
   对于块擦除命令，地址应为 FLASH 存储器地址空间中任意一个位置地址。

对于字节编程，那么该地址为要编程的地址，数据为要编程的数据。

注意：正确擦除操作之后，只允许编程一次。对未擦除的地址进行编程会破坏该地址中的数据。

2. FCMD 写入命令。命令如下：

$05:查空

$20:字节编程

$25:快速编程

$40:页擦除

$41:块擦除

3. FSTAT 寄存器中 FCBEF 位写 1，启动 FLASH 操作命令（包括地址和数据信息）。

在第 1 步和第 3 步骤之间，可以手动的设置 FCBEF=0，可中止该次操作。而在开始新的命令之前，必须清除 FACCERR 访问错误标志。

必须遵守这一严格过程，否则命令不会被接受。FCCF 位表明命令是否执行完成。

1、3、9 快速编程过程

快速编程用于编程连续地址的FLASH，时间上少于通过标准编程命令方式。这是由于在两次编程操作之间，不必关闭高压。通常，当发出一个擦除或编程命令，内部电压泵被打开，为FLASH阵列提供高压。一旦命令执行完成，电压泵自动关闭。而当发出一个快速编程命令，电压泵使能并保持（在一次快速编程完成之后，遇到以下情形）：

- 在当前编程操作完成之前，下一个快速编程命令已经列队好
- 下一个连续地址与当前编程地址为同一行。一行FLASH包括64个字节。一行内的地址通过A0~A5来定位。当A0~A5全为0时，开始新的一行。

快速编程模式下，第一个字节编程时间与标准编程模式时间相同。接下的字节编程时间为快速编程时间，前提是满足上面两个条件之一。

当连续字节的地址为新的一行的第一个字节，其编程时间与标准编程时间相同，这是因为，高压必须关闭，给另一个FLASH阵列供电。在当前命令完成之前，没有新的快速编程命令，电荷泵将关闭，高压移除该阵列。

1、3、10 FLASH 加密

当加密机制使能，FLASH和RAM认为是加密的区域；而低页，高页寄存器和后台调试控制器被认为是未加密区域。在加密的存储器执行的程序可以访问任何MCU存储器和资源。反之，**在非安全区域的程序或后台调试接口试图访问一个加密区域的话，将被阻止（写操作忽略，读操作返回0值）。**

加密机制使能或禁止是由FOPT寄存器中的非易失性位SEC01:SEC00位决定。复位期间，NVOPT中的值装载到FOPT中。因此在程序烧写到存储器时，可以设置好NVOPT寄存器。1：0的组合取消加密机制，其他3种组合均使能加密机制。故擦除之后，组合（1：1）会使MCU加密。在开发过程中，当FLASH被擦除，编程SEC01:SEC00=1:0，从而MCU处于未加密状态，方便程序的开发。

当MCU处于加密状态时，片上调试模块不能使能。但独立的后台调试控制器仍用于后台存储器访问命令，但MCU不能进入后台调试模式（除非在复位的上升沿，BKGD/MS维持

低）。

通过后门密匙，可以临时解密。如果NVOPT/FOPT中的KEYEN位为0，则后门密匙不允许，除了完全擦除FLASH外，没有其他方法取消加密。如果KEYEN=1，一个加密的用户程序可以临时取消加密状态，方法如下：

1. 向FCNFG寄存器中的KEYACC写1。从而对FLASH的NVBACKKEY~NVBACKKEY+7位置的写操作是比较后门密匙，而不是作为FLASH编程或擦除命令的第一步。
2. 向NVBACKKEY～NVBACKKEY+7位置写入用户输入的密匙值。写过程必须按一定顺序进行，开始于NVBACKKEY，结束于NVBACKKEY+7。STHX指令不应用于该写过程，因为对NVBACKKEY～NVBACKKEY+7写操作不能在接近总线周期完成。用户软件通常从通过通信接口从外部系统获得密匙。
3. 向FCNFG寄存器的KEYACC位写0。如果刚写入的8字节密匙与存储在FLASH的密匙相同，SEC01:SEC00自动变为1：0，直到下次复位，那么MCU的加密状态取消。

加密密匙只能由加密存储区域（RAM或FLASH）的程序修改，因此不能通过后台调试命令输入。

后门比较密匙位于FLASH空间，与中断和复位向量同属一个512字节块。

通过后台调试接口执行以下步骤可以取消加密机制：

1. 写FPROT位取消任何块保护。FPROT只能通过后台调试命令修改，而不能通过应用软件。
2. 块擦除FLASH
3. 查空。

为避免下次复位之后返回安全状态，应编程 SEC01:SEC00=1:0。

1、3、11 向量重指

当任意块受到保护，那么复位和中断向量也将受到保护。向量重指允许用户程序调整中断向量信息（不包括引导程序区域和复位向量）。设置NVOPT寄存器中的FNORED=0，允许向量重指。为了能够重指，至少一部分但不是所有FLASH空间进行块保护。所有中断向量（0XFFC0~0XFFD）都被重指，而复位向量不改变。

例如，如果FLASH的512字节即（0XFE00~0XFFFF）受到保护。那么中断向量（0XFFC0~0XFFFD）被重指到0XFDC0~0XFDFD。如果现在SPI中断发生，那么0XFDE0:FDE1处的中断向量被使用。利用这一特点，用户可以对未保护的区域进行编程（包括新的中断向量值），而受块保护的区域包括默认的向量地址不会改变。

1、3、12 FLASH块保护

块保护是指被保护的FLASH区域可以避免被编程或擦除。块保护是通过FLASH保护寄存器（FPROT）控制的。当块保护使能，块保护将保护FLASH地址0XFFFF以下任何以512个字节为单位的块。

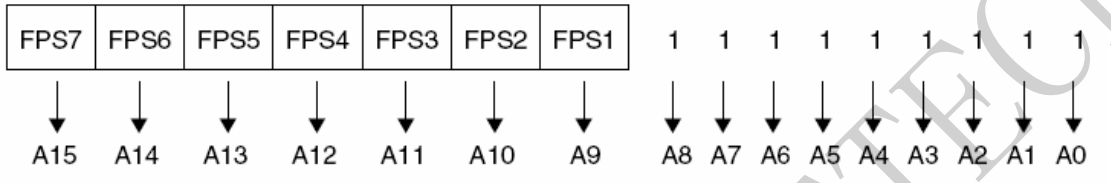
复位之后，NVPROT内的值装载到FPROT中。FPROT不能被应用软件直接改变，所以用户程序不能改变块保护设置。因为NVPROT位于FLASH的最后512个字节，如果任何数量的存储器被保护，那么NVPROT自身也受保护，不允许修改。

后台调试命令可以修改FPORT，所以后台调试命令可以擦除和再次编程受保护的

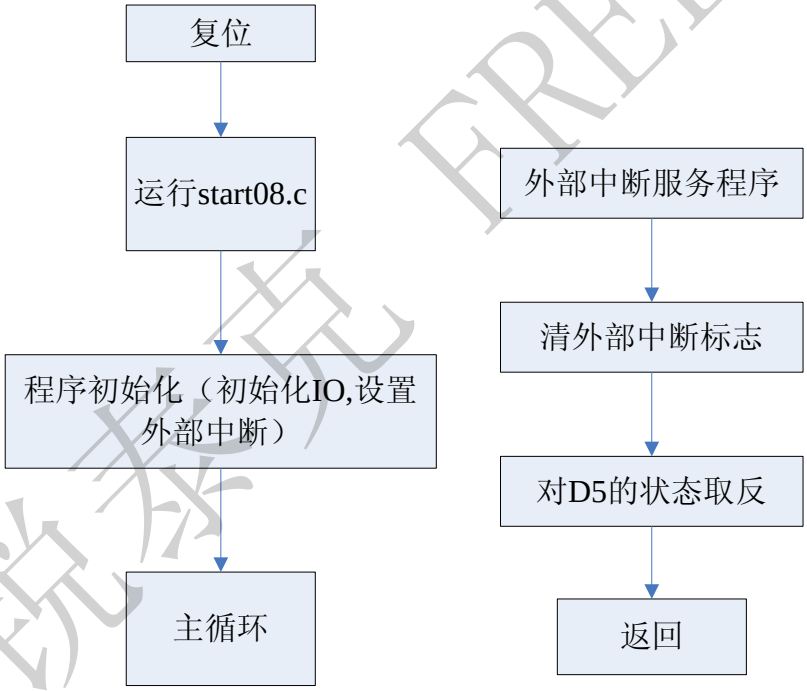
FLASH存储器。

例如为了保护0XE000~0XFFFF存储器，应设置FPS＝1101，111。这样从0X0000～0XDFFF为未保护区域。同时为了能够修改FPS值，首先应该设置FPDIS=0。所以寄存器NVPROT的值应设置为0XDE。

用户可以对存放引导程序的区域进行块保护，这样引导程序就可以对其他FLASH空间擦除和编程。由于引导程序受到保护，所以在擦除和编程期间，即使电源掉电，也不会受到破坏。



2、流程图



3、实验现象

3、1 打开 flash 实验的 flash 实验.mcp。

3、2 修改 const byte nvopt_user@0xFFBF=0XF0;

//程序下载到 flash 时，地址 0xFFBF 处的值被指定为 0XF0。FLASH 被加密。

3、3 单击调试按键，程序会下载到目标芯片中，但在调试窗口中看不到源程序。关闭调试窗口。

3、4 按下复位键 SW7，测试外部中断按键 SW6，会发现程序是正常运行的。

3、5 再次单击调试按键，会出现如下警告窗口：

说明芯片已被加密。解密方法可参见第三章。

注意：芯片电压要在 5V 左右。

3、6 修改 const byte nvopt_user@0xFFBF=0X82;

const byte nvprot_user@0xFFBD=0XFB;//0XFE00~0XFFFF 被保护

3、7 单击调试按键，进入调试窗口。单击全速运行，会发现外部中断按键 SW6 并不起作用。这是因为中断向量发生了重指。

3、8 修改 MC9S08dz60.H 文件中的“#define Virq 0x0000FFFA”为“#define Virq 0x0000FDFA”。

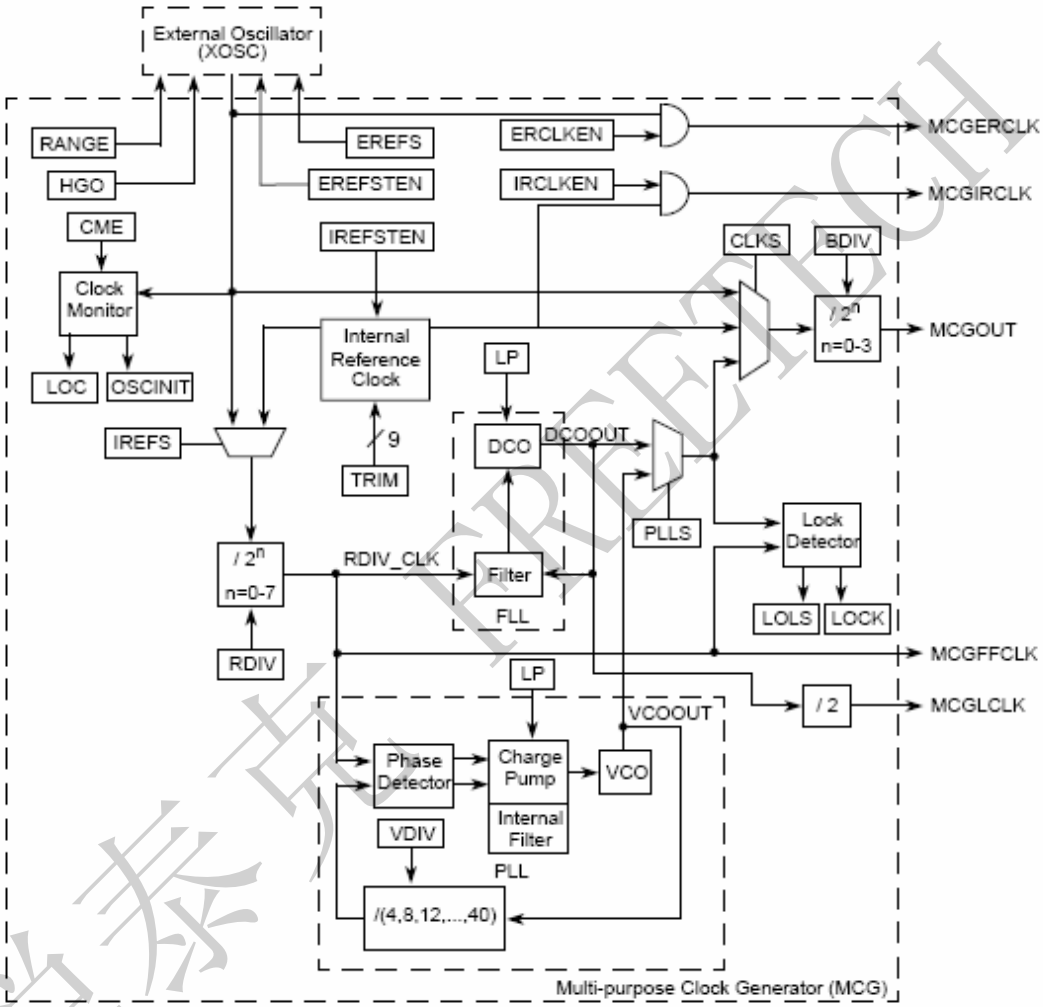
3、9 单击调试按键进入调试环境，单击全速运行，发现外部中断按键起作用了。

4、程序代码

参见 flash 实验。

实验六 内部时钟

1、实验功能介绍



外接 4M 晶振总线实现 8M，并通过 MCLK 进行输出 4M 进行。

2、流程图

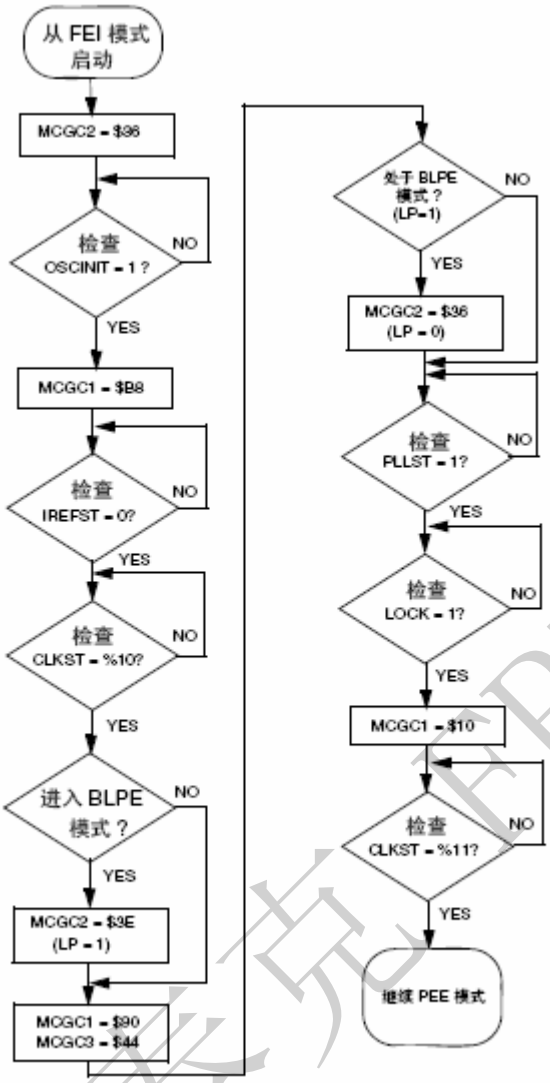


图 8-9. 使用 4 MHz 晶体从 FEI 转换到 PEE 模式的流程图

3、实验现象

发光管 d5 为总线频率闪亮，pta0 输出总线 2 分频

4、代码

见附件

实验七 MCU 运行模式

1、实验功能介绍

1、1 特点

- 用于代码调试的后台调试模式
- 等待模式

  －CPU 挂起

－系统时钟运行
－稳压模块维持

- 停止模式
  －CPU 和总线停止
  －STOP2：部分内部电路进入节电模式，RAM 数据保持
  －STOP3：所有内部电路仍供电，可用于快速恢复

1、2 正常运行模式

MCU 复位之后，BKGD/MS 为高，进入正常运行模式。CPU 预取 0XFFE:0XFFFF 处代码，也就是复位向量处的代码。

1、3 后台调试模式

BDC：后台调试控制器

DBG：片上调试模块

BDC 和 DBG 一起工作提供了分析 MCU 操作的一种方式，用于调试用户代码。

5 种进入后台调试模式的方式：

- MCU 复位信号的上升沿时 BKGD/MS 引脚为低
- MCU 的 BKGD 引脚接收到 BACKGROUND 命令
- MCU 执行 BGND 命令
- MCU 遇到 BDC 断点
- MCU 遇到 DBG 断点

进入后台调试模式后，CPU 挂起等待后台调试命令而不会执行用户程序。按后台调试指令按是否影响到 CPU 可以分为两种类型后台调试命令：

- 非干扰指令（硬 BDM 指令）：用户程序正常运行下的 BDM 指令，这些也可以在后台调试下运行。包括：
  －存储器访问指令
  －带状态返回存储器访问指令
  －BDC 寄存器访问指令
  －BACKGROUND 指令
- 干扰指令：只能在后台调试模式下执行
  －CPU 寄存器指令
  －单步跟踪用户程序
  －离开后台调试模式，进入正常运行模式(GO)

实际上我们在调试过程中，进入调试界面时，MCU 处于活跃的后台调试模式。

以上两种模式我们只要了解即可。下面主要介绍 WAIT 模式和 STOP 模式和相关实验。

1、4 WAIT 模式

执行完 WAIT 指令，CPU 进入低功耗状态。进入 WAIT 模式之前，设置 CCR 寄存器的 I＝0，使能等待模式下中断功能，从而使该中断唤醒 CPU，执行中断服务程序。

在等待模式下，BACKGROUND 和带状态返回访问存储器后台调试指令可以执行。BACKGROUND 命令可以唤醒 CPU 进入后台调试模式；带状态返回访问存储器命令不会访问存储器，而是返回一个错误标志，表明 MCU 处于停止状态或等待状态。

执行 WAIT 指令后，MCU 进入 WAIT 模式。硬件中断可以唤醒 WAIT 模式下的 MCU，软中断指令 SWI 不能唤醒 WAIT 模式下的 MCU。当外部中断唤醒 MCU 后，MCU 执行中断服务程序。

注：不要在中断服务程序中执行 WAIT 指令。

1、5 STOP 模式

设置 SOPT 中 STOPE=1，执行 STOP 指令，MCU 进入 2 种停止模式之一。任何一种方式下，总线和 CPU 时钟均停止。如果执行 STOP 命令之前，STOPE 位未置位，那么执行 STOP 命令不会进入 STOP 模式，而是作为非法操作码而发生复位。而通过设置 SPMSC2 中的相应位可以选择不同的停止模式。

Mode	PPDC	CPU, Digital Peripherals, FLASH	RAM	ICG	ADC	Regulator	I/O Pins	RTI
Stop2	1	Off	Standby	Off	Disabled	Standby	States held	Optionally on
Stop3	0	Standby	Standby	Off[1]	Optionally on	Standby	States held	Optionally on

[1] Crystal oscillator can be configured to run in stop3. Please see the ICG registers.

1、5、1 STOP3

在执行 STOP 命令之前，设置 PPDC=0，STOPE=1，那么执行 STOP 指令，MCU 进入 STOP3 模式。一旦进入 STOP3 模式，MCU 内的所有时钟包括振荡器均停止。ICG 处于待命状态。内部寄存器和逻辑电路以及 RAM 中的数据均维持。I/O 引脚状态不被闩锁。

复位引脚上的复位信号，异步中断或实时中断可以唤醒 STOP3 模式下的 MCU。

复位引脚上的复位信号唤醒 MCU 的话，MCU 按照复位状态运行。而通过异步中断或实时中断唤醒的话，MCU 执行相应的中断复位程序。

一个单独的自时钟源（大约 1kHz）为实时中断提供时钟。当 RTIS2:RTIS1:RTIS0=0:0:0，实时中断和 1kHz 时钟源均关闭，从而进一步降低功耗。

从以上对 STOP2 和 STOP3 模式介绍来看，如何 STOP3 模式能够满足系统的功耗要求，那么选择将单片机在系统空闲时进入 STOP3 模式是一个不错的选择。

1、5、2 STOP2

STOP2 模式下 MCU 功耗很低，同时 RAM 中的数据和 I/O 引脚的当前状态处于维持状态。欲进入 STOP2 模式，那么在执行 STOP 指令之前设置 PPDC=1，STOPE=1。如果在执行 STOP 命令之前，LVD 使能，那么执行 STOP 命令不会进入 STOP2 模式，而是进入 STOP3 模式。

在进入 STOP2 模式之前，用户必须保存 I/O 端口寄存器中的数据到 RAM 中。一旦从 STOP2 模式退出后，用户应通过软件将保存的值恢复。

当 MCU 处于 STOP2 模式，所有的由电压稳压器供电的内部电路均关闭，RAM 除外。I/O 引脚闩锁。当中断使 MCU 从 STOP2 模式退出后，I/O 状态仍处于闩锁状态，直到向 SPMSC2 寄存器中的 PPDACK 位写 1。

当由于复位引脚上的复位信号或外部中断或 RTI 中断可以将 MCU 从 STOP2 模式退出。当 MCU 处于 STOP2 模式，IRQ 总是对低电平敏感而不管进入 STOP2 模式之前的对 IRQ

的配置。

一旦 MCU 从 STOP2 模式退出，那么 MCU 将像上电复位那样运行，当然除了引脚状态仍处于闩锁状态外。CPU 运行复位向量处的程序。MCU 的系统和所有外围模块处于其默认状态必须进行初始化。

当 MCU 从 STOP2 模式唤醒，SPMSC2 寄存器中的 PPDF 位置位。通过该标志位用户可以执行从 STOP2 恢复服务程序。

当 I/O 配置为通用 I/O 时，在进入 STOP2 模式之前，将 I/O 寄存器的状态保存到 RAM 中，在 MCU 退出 STOP2 模式后，用户软件从 RAM 中恢复保存的状态。如果在写 PPDACK 之前未恢复的话，那么 I/O 引脚转变为复位状态。

该模式是一种更加低功耗模式。该模式下只会维持 RAM 中的数据和 I/O 引脚的当前状态，因此在进入 STOP2 模式之前，应将必要的 I/O 数据寄存器或其他映射寄存器保存到 RAM 中。

进入 STOP2 模式的条件为：

STOPE=1;//允许执行 STOP 指令

PPDC=1;//选择 STOP2 模式

LVDSE=0 或 LVDE=0;//关闭 LVD

唤醒 STOP2 的方法：

1、复位

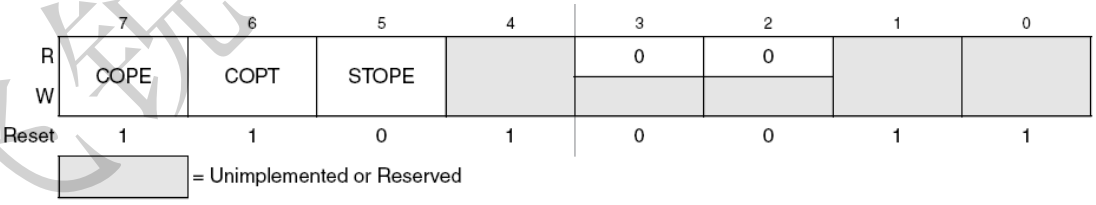
2、RTI

3、IRQ:IRQ 用于唤醒的话，MCU 会认为低电平有效中断，所以用户外部需将 IRQ 引脚上拉或进入 STOP2 模式之前使能 IRQ 的上拉。

中断或复位可以唤醒 STOP2，但如果是中断唤醒的话，唤醒后并不执行中断服务程序，而是从复位向量处执行程序。那么如何判断是复位还是中断唤醒的 STOP2 模式的 MCU 呢？通过判断 PPDF 位即可；如复位唤醒的 PPDF=0;所有的 I/O 或外围功能模块需要重新初始化；如是中断唤醒的 PPDF=1;此时我们可以将在进入 STOP2 模式前保存到 RAM 的数据恢复，然后清除 PPDF 位即可。

1、6 相关寄存器

1、6、1 系统选项寄存器：SOPT（$1802）



该寄存器在任何时候均可以读取。复位之后，只允许写一次。用户在复位之后初始化过程中，应写一次该寄存器，尽管写入的值与该寄存器复位之后相同。

7 COPE	看门狗使能位 0：关闭看门狗；1：看门狗使能
6 COPT	看门狗溢出时间 0：较短的溢出时间间隔（2 的 13 次方个总线周期）；1：较长的溢出时间间隔（2 的 18 次方个总线周期）

5 STOPE	STOP 模式使能：如果 STOPE 设置为 0，而执行 STOP 指令，那么会产生一个非法操作码复位。 0：禁止进入 STOP 模式；1：允许进入 STOP 模式

1、6、2 系统电源管理状态和控制寄存器 2－SPMSC2（$180A）

	7	6	5	4	3	2	1	0
R	LVWF	0	LVDV	LVWV	PPDF	0		PPDC[1]
W		LVWACK				PPDACK		
Power-on reset:	0[2]	0	0	0	0	0	0	0
LVD reset:	0[2]	0	U	U	0	0	0	0
Any other reset:	0[2]	0	U	U	0	0	0	0

= Unimplemented or Reserved　　U = Unaffected by reset

7 LVWF	低电压报警标志 0：未出现低电压；1：出现或出现过低电压
6 LVWACK	低电压报警确认 写 1 清除 LVWF
5 LVDV	低电压检测电压选择 0：V_{LVD}= V_{LVDL} 1：V_{LVD}= V_{LVDH}
4 LVWV	低电压报警电压选择 0：V_{LVW}= V_{LVWL} 1：V_{LVW}= V_{LVWH}
3 PPDF	该位表明 MCU 是从 STOP2 模式恢复 0：MCU 不是从 STOP2 模式唤醒 1：MCU 从 STOP2 模式唤醒
2 PPDACK	清除 PPDF 位 写 1 清除 PPDF
0 PPDC	该位用于选择 STOP2 或 STOP3 模式 0：STOP3 模式允许 1：STOP2 模式允许

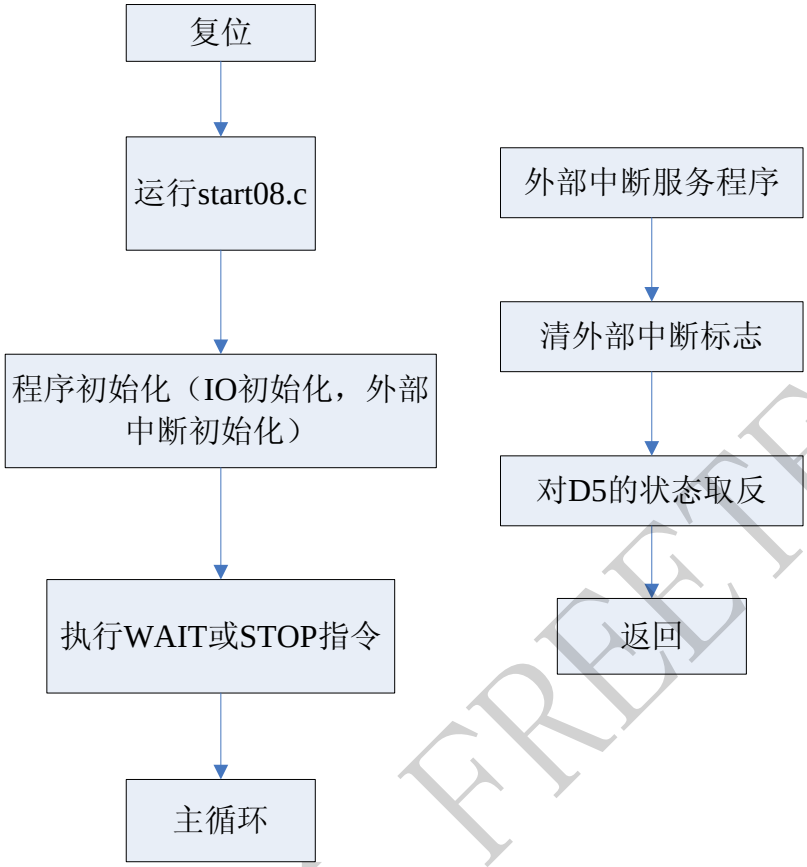
1、6、3 系统电源管理状态和控制寄存器 1－SPMSC1（$1809）

	7	6	5	4	3	2	1[1]	0
R	LVDF	0	LVDIE	LVDRE[2]	LVDSE[2]	LVDE[2]		BGBE
W		LVDACK						
Reset	0	0	0	1	1	1	0	0

= Unimplemented or Reserved

7 LVDF	低电压检测标志，只读 1：表明低电压事件发生
6 LVDACK	低电压检测确认，只写 写 1 清除 LVDF
5 LVDIE	低电压检测中断使能控制位 0：中断禁止；1：中断使能
4 LVDRE	低电压检测复位使能控制位 0：LVDF 不产生硬件复位；1：LVDF=1 时产生强制复位
3 LVDSE	低电压检测是否在停止模式可用 0：STOP 模式下 LVD 禁止；1：STOP 模式下 LVD 可运行
2 LVDE	LVD 使能 0：LVD 关闭；1：LVD 使能
0 BGBE	能系缓冲使能：ADC 模块的每一通道上使能内部带隙电压控制位 0：关闭能系参考电压；1：使能能系参考电压

2、流程图



3、实验现象

3、1 打开 wait 实验.mcp；该程序初始化设置后，会使 MCU 进入 WAIT 状态；外部中断按键 SW6 可唤醒 WAIT 状态下的 MCU，MCU 唤醒后会执行外部中断的服务程序，对 D5 的状态取反。

3、2 打开 stop3 实验.mcp；该程序初始化设置后，会使 MCU 进入 stop3 状态；外部中断按键 SW6 可唤醒 stop3 状态下的 MCU，MCU 唤醒后会执行外部中断的服务程序，对 D5 的状态取反。

3、3 打开 stop2 实验.mcp；该程序初始化设置后，会使 MCU 进入 stop2 状态；外部中断按键 SW6 可唤醒 stop2 状态下的 MCU，MCU 唤醒后不会执行外部中断的服务程序，我们会观测到 D5 状态并未改变。再次按一下外部中断按键 SW6，我们会观测到 D5 状态会发生改变。

4、实验代码

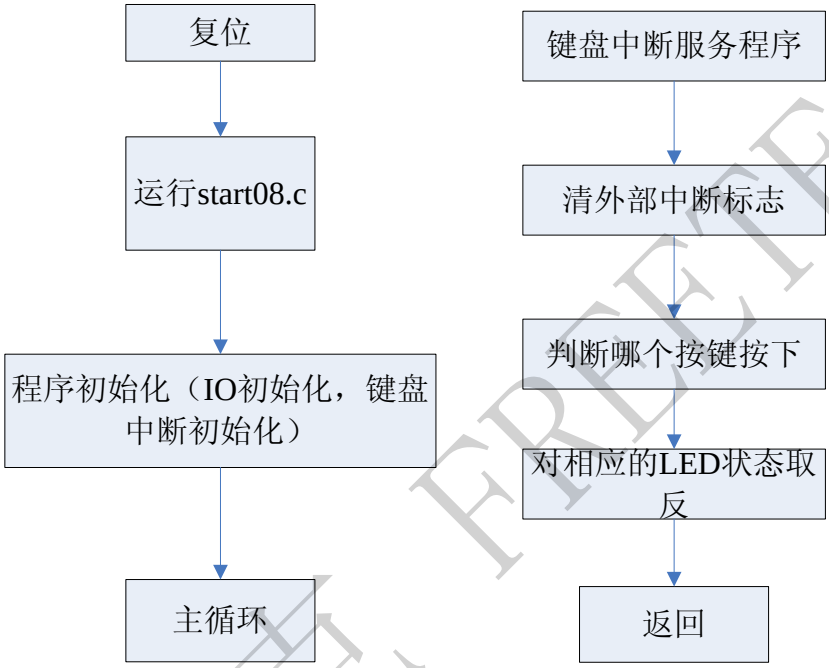
参见 STOP2 实验，STOP3 实验，WAIT 实验。

实验八 键盘中断实验

1、实验功能介绍

通过按动按键，驱动相应的灯进行指示。采用 i/o 的沿触发中断方式。

2、流程图



3、实验现象

按一下 SW1，D5 点亮，再按一下 SW1，D5 熄灭
按一下 SW2，D7 点亮，再按一下 SW2，D7 熄灭

4、程序代码

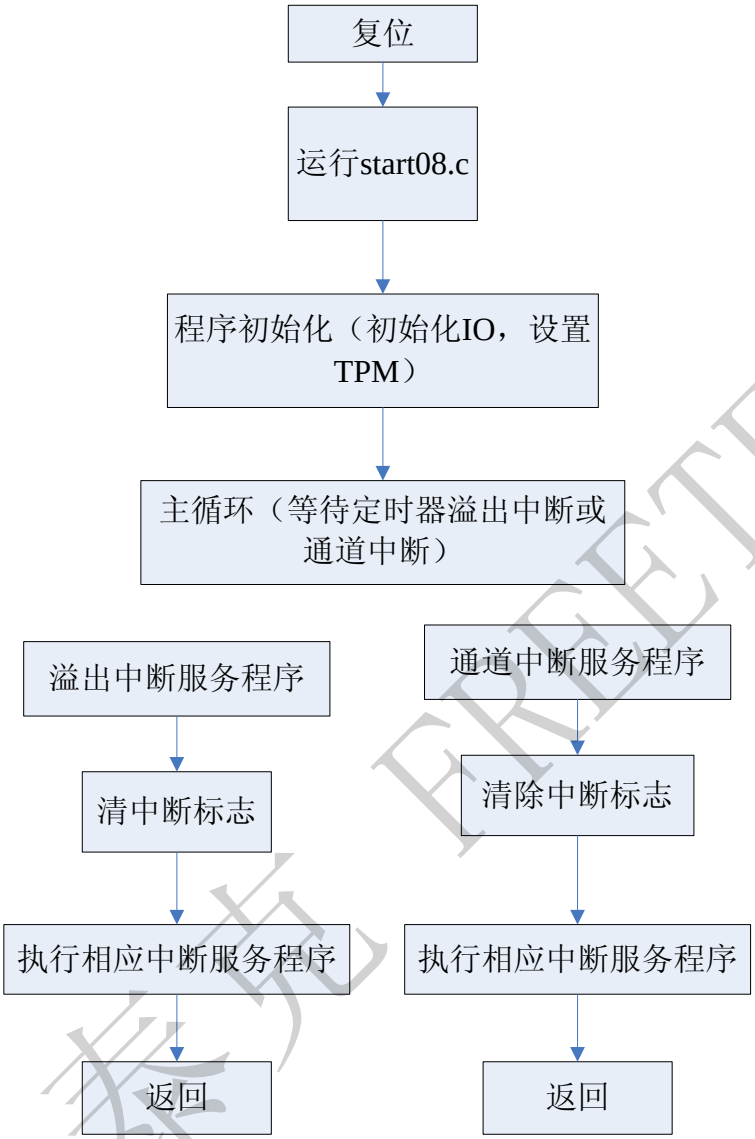
参见 kb 实验。

实验十 TPM 实验

1、实验功能介绍

熟悉定时器的输入捕捉，输出比较，及 pwm 输出功能。

2、流程图



3、实验现象

3、1 运行 TPM 自由计数实验，会观测到 D5 以较快的频率闪烁，D7 的闪烁频率很慢。

3、2 运行 TPM 边沿对齐实验，可以观测通道输出波形。

4、程序代码

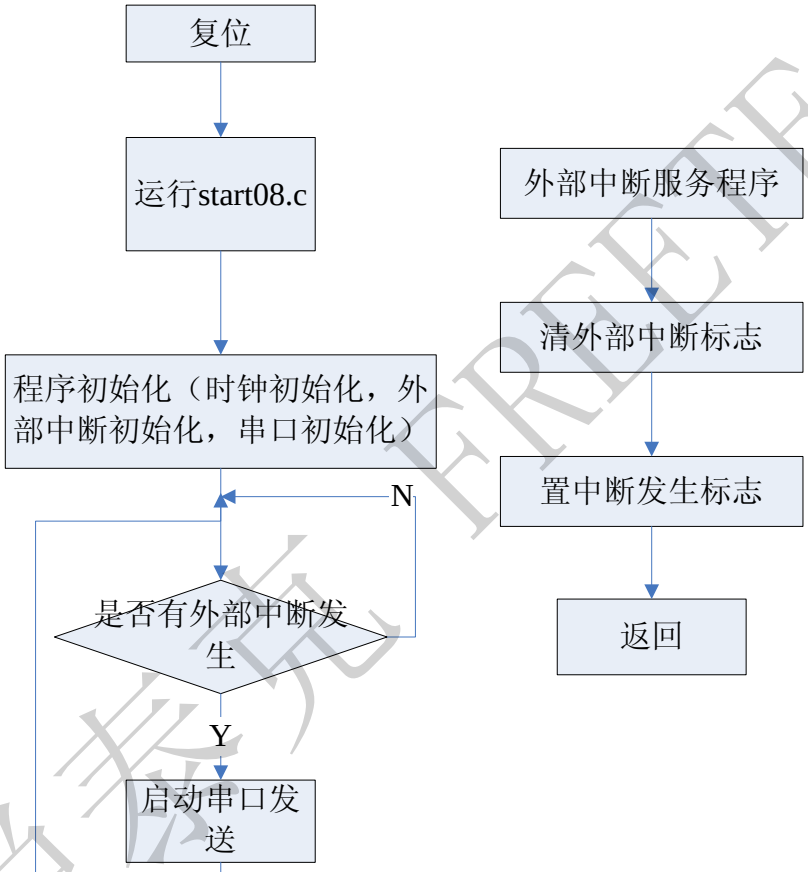
参见“TPM 自由计数实验”， “TPM 边沿对齐 PWM 实验”。

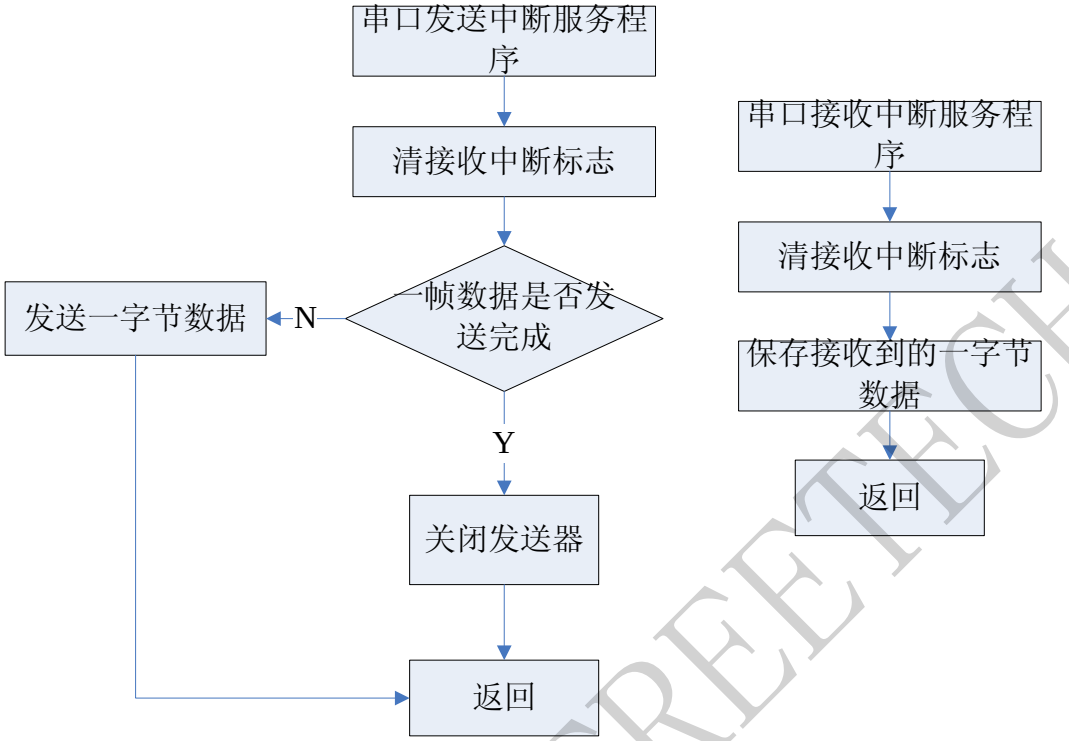
实验十一 UART 实验

1、实验功能介绍

通过串口1与pc机进行232的通讯。

2、流程图





3、实验现象

本实验内部时钟工作于 FBE 模式，外接晶体 12M，总线频率 6M。

串口设置：通信波特率 9600bps；8 位数据位，无奇偶校验。

打开 PC 串口调试助手，设置好通信波特率和数据格式。按下外部中断按键 SW6，串口助手就会接收到一帧数据；串口助手发送数据给学习板，但学习板只会保存接收到数据的最后一个字节。再次按外部中断按键 SW6，可以观测接收到的数据是否正确。

4、源代码

参见 UART1 实验。

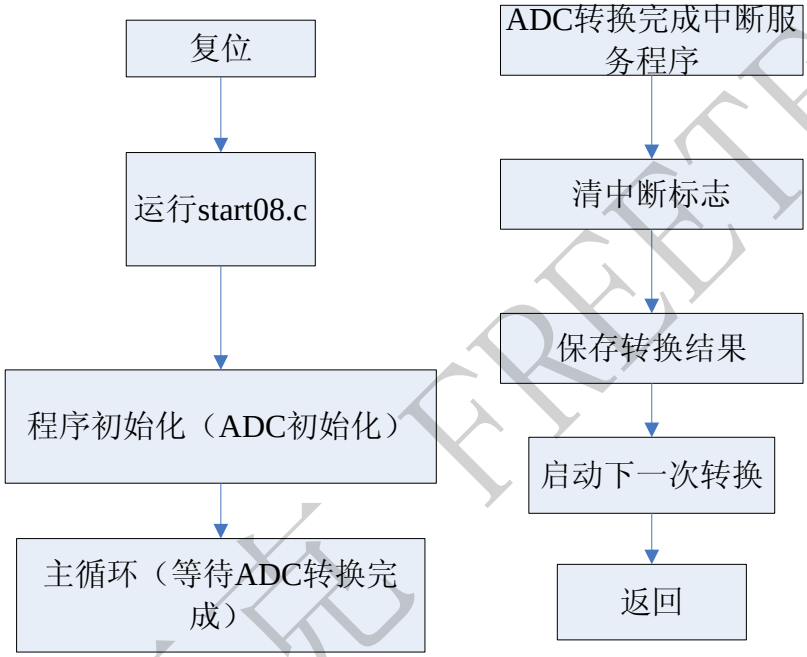
实验十二 ADC 实验

1、实验功能介绍

调节电位器改变 ad 采样数值，通过仿真界面进行追踪显示。

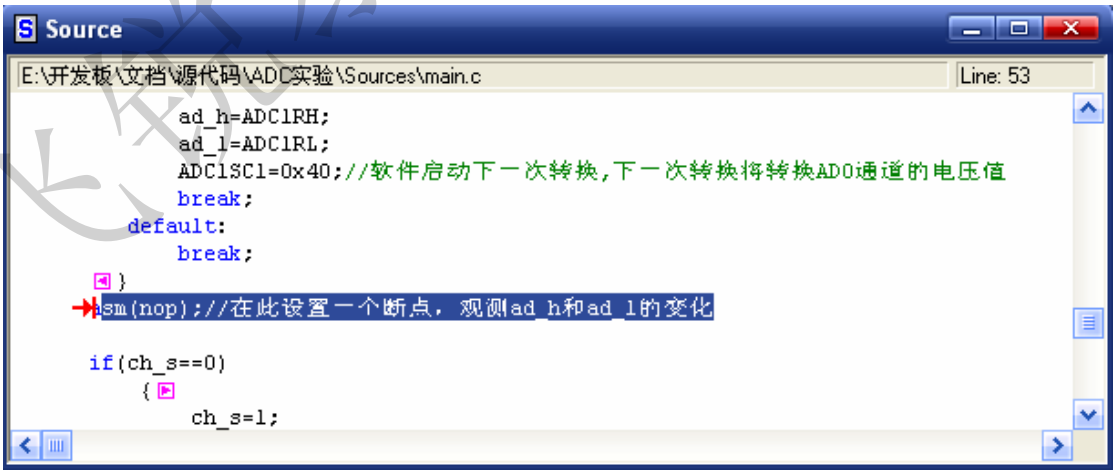
通道 ad1 和 ad10；输入范围 0-5v,对应输出 0-4096, 12 位

2、流程图

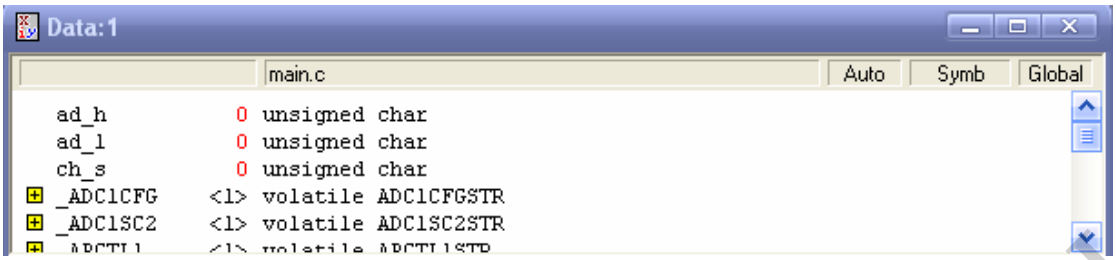


3、实验现象

3、1 打开 ADC 实验.mcp，在 ADC 转换完成中断服务程序中设置一个断点，如下图所示：



3、2 单击全速运行按键，程序停在断点处。观察 Data:1 窗口，如下图所示



说明通道 0 转换结果为 0。用万用表测试学习板 ADC 部分的 ADC_AD1 测试点的电压，会发现电压值为 0。

3、3 再单击全速运行按键，观察 Data:1 窗口，如下图所示：

说明通道 1 转换结果为 0X3FF。用万用表测试学习板 ADC 部分的 ADC_AD2 测试点的电压，会发现电压值为 4.2V 左右。即该通道模拟输入电压等于参考电压。

3、4 旋转电位器 RV2 将改变 ADC_AD1 的电压值；旋转电位器 RV1 将改变 ADC_AD2 的电压值。按上面的方法观测 ad_h 和 ad_l 的变化。

4、程序代码

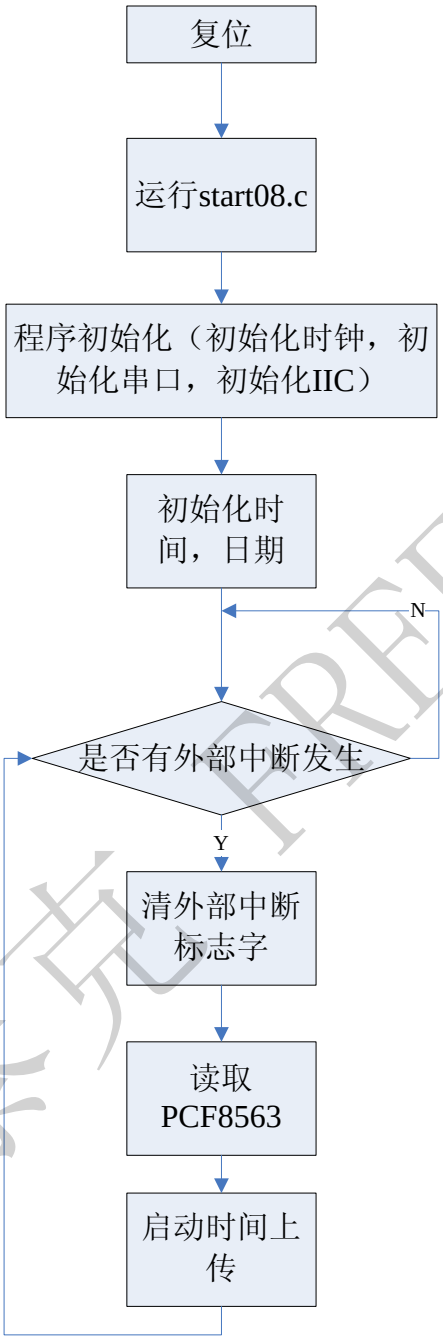
参见 ADC 实验。

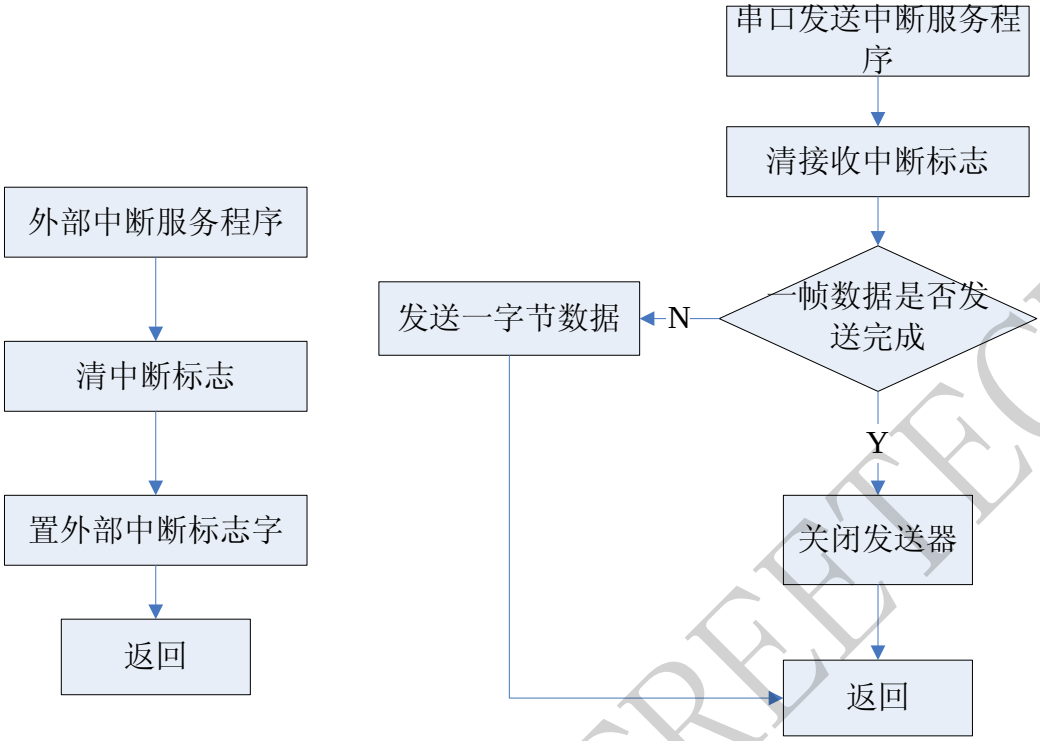
实验十三 IIC 模块实验

1、实验功能介绍

标准的 iic 接口的时钟芯片，可以实现 iic 主从通讯。读取时时的时钟

2、流程图





3、实验现象

3、1 将 IIC 实验程序下载到学习板中。

3、2 打开串口调试助手，波特率 9600bps，无校验位。

3、3 按一下外部中断按键 SW6，学习板会向 PC 发送一帧数据；依次为“秒，分，时，日，周，月，年”。

4、程序代码

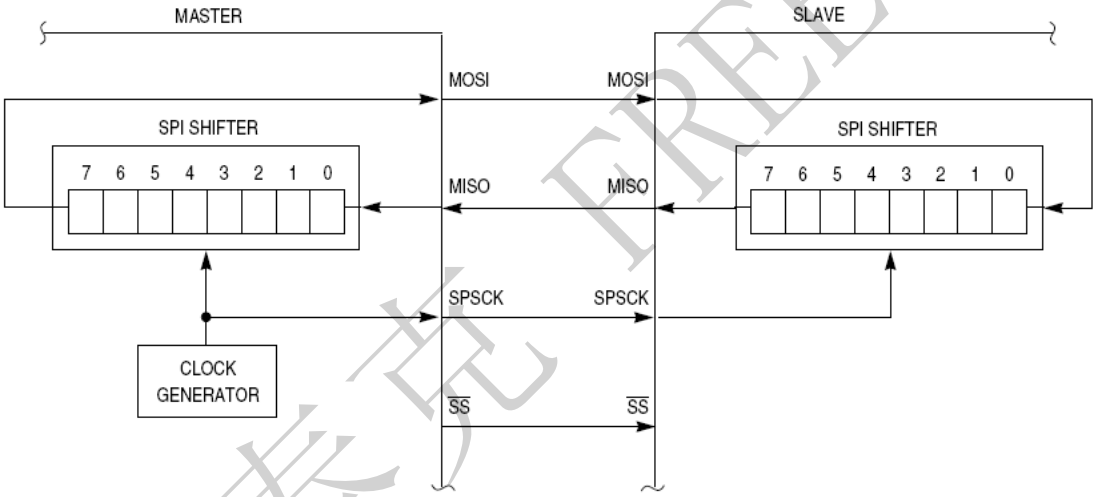
参见 IIC 实验。

实验十四 SPI 接口

1、实验功能介绍

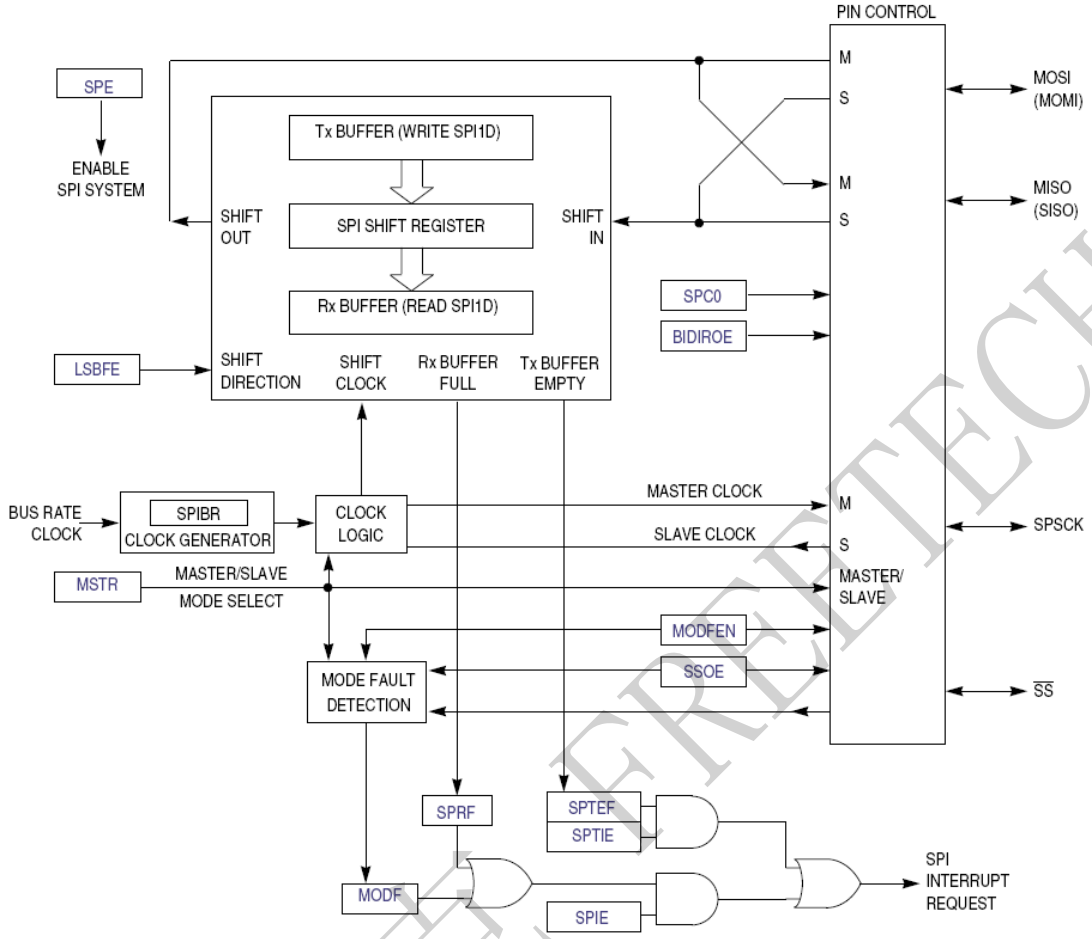
1、1 特点

- 可设置为主机或从机工作方式
- 全双工模式或单线双向模式
- 可编程的传送速率
- 使用相互独立的发送和接受数据寄存器，实现双缓存冲操作
- 可设置时钟极性和相位
- 从机选择输出
- 可选择从最高位或最低位开始传送



SPI 系统连接图

主机初始化 SPI 接口。在一次 SPI 传送发生时，一个字节通过主出从入（MOSI）引脚移位输出；同时另一个字节从主入从出（MISO）引脚输入。这个传送过程可以理解为主从 SPI 移位寄存器相互交换数据。SPSCK 为主设备输入给从设备的时钟信号。从设备的 SS(低有效)引脚必须拉低。



SPI 原理框图

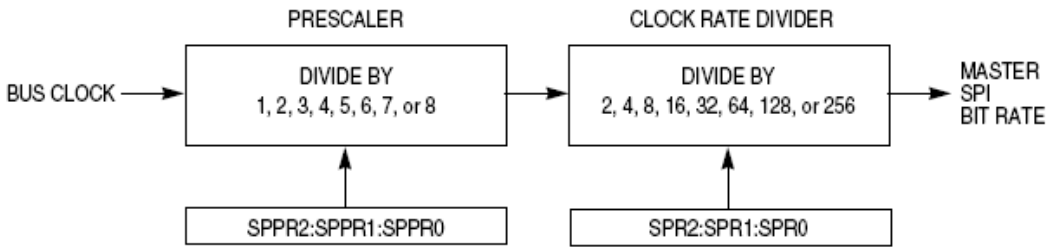
SPI 的中心元件为 SPI 移位寄存器。当写 SPI1D 时实际上是将数据写入双缓冲冲器中，然后该数据通过 SPI 移位寄存器，开始数据转移。当字节数据移位完毕后，数据转移到双缓冲接收器中，通过执行读取 SPI1D 来读取该数据。

当 SPI 配置为主设备，SPSCK 引脚作为时钟输出；MOSI 作为移位寄存器的输出；MISO 作为移位寄存器的输入。

当 SPI 配置为从设备，SPSCK 引脚作为时钟输入，MISO 作为移位寄存器的输出；MOSI 作为移位寄存器的输入。

SPI 的外部连接：主从设备的 SPSCK 相连；MISO 相连；MOSI 相连。

1、2 SPI 波特率发生器



SPI 波特率发生器

时钟源为总线时钟，经过两级分频得到 SPI 通信的 1 位时钟。

1、3 相关寄存器

1、3、1 SPI 控制寄存器 1（SPI1C1）

	7	6	5	4	3	2	1	0
R W	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE
Reset	0	0	0	0	0	1	0	0

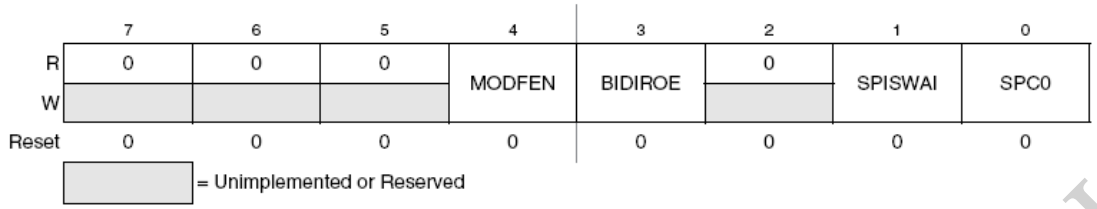
7 SPIE	SPI 中断允许位 0：不允许由 SPRF 和 MODF 产生中断 1：允许 SPI 接收缓存满标志 SPRF 或模式错误标志 MODF 产生中断
6 SPE	SPI 系统使能：禁止 SPI 功能，将停止正在传送的过程，清除数据缓冲；SPRF 置 0，SPTEF 置 1(表明 SPI 发送数据缓冲为空) 0:停止 SPI 1：允许 SPI
5 SPTIE	SPI 传送中断允许位 0：不允许 SPTEF 产生中断；1：允许 SPI 发送缓冲器空标志 SPTEF 产生中断
4 MSTR	主/从模式选择控制位 0：SPI 模块配置为从机模式；1：配置为主机模式
3 CPOL	时钟极性控制位 0：选择高电平有效时钟，SPSCK 空闲状态为低电平； 1：选择低电平有效时钟，SPSCK 空闲状态为高电平
2 CPHA	时钟相位控制位 0：在 SPI 数据传送的第 1 个周期的中间时刻产生第一个 SPSCK 跳变沿 1：在 SPI 的数据传送的第 1 个周期的开始时刻产生第一个 SPSCK 跳变沿
1 SSOE	从机选通控制位 该控制位与 SPIC2 中的 MODFEN 控制位一起控制 SS 引脚，见 SS 引脚功能表
0 LSBFE	SPI 数据传送起始位置控制位 0：SPI 数据传送从最高位开始 1：SPI 数据传送从最低位开始

MODFEN	SSOE	Master Mode	Slave Mode
0	0	General-purpose I/O (not SPI)	Slave select input
0	1	General-purpose I/O (not SPI)	Slave select input
1	0	$\overline{SS}$ input for mode fault	Slave select input
1	1	Automatic $\overline{SS}$ output	Slave select input

SS 引脚功能表

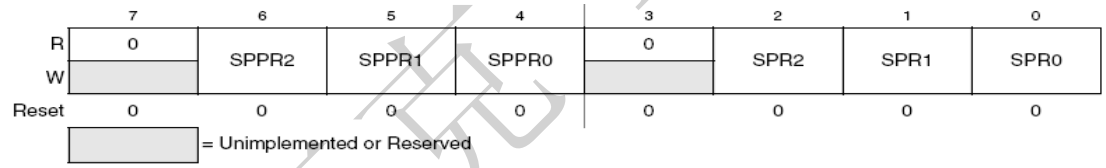
注意：不要在 SPI 禁止的情况下，改变 CPHA 位的值。

1、3、2 SPI 控制寄存器 2（SPI1C2）



4 MODFEN	主机模式检错功能允许控制位。该位对于从设备没有意义。作为主设备时，该位与 SPI1C1 的 SSOE 位一起作用。见上表。
3 BIDIROE	半双向模式输出允许控制位。该位只有在 SPC0=1 时有效 0：SPI 的 I/O 作为输入端口；1：SPI 的 I/O 作为输出端口
1 SPISWAI	SPI 在 WAIT 模式下的停止控制位 0：在 WAIT 模式仍继续运行；1：在 WAIT 模式下，停止 SPI 时钟
0 SPC0	SPI 引脚控制位：如果 MSTR=0 即 MCU 作为从机，SPI 模式使用 MISO 引脚作为半双向的数据传送；如何 MSTR=1 即 MCU 作为主机，SPI 模块使用 MOSI 引脚作为半双向的数据传送。 0：SPI 进入全双工模式（输入/输出采用独立的信号线） 1：SPI 进入半双工工作模式

1、3、3 SPI 波特率寄存器（SPI1BR）



6：4 SPPR[2:0]	SPI 波特率预分频器
2：0 SPR[2:0]	SPI 分频因子

SPPR2:SPPR1:SPPR0	Prescaler Divisor
0:0:0	1
0:0:1	2
0:1:0	3
0:1:1	4
1:0:0	5
1:0:1	6
1:1:0	7
1:1:1	8

SPI 波特率预分频器

SPR2:SPR1:SPR0	Rate Divisor
0:0:0	2
0:0:1	4
0:1:0	8
0:1:1	16
1:0:0	32
1:0:1	64
1:1:0	128
1:1:1	256

SPI 分频因子

1、3、4 SPI 状态寄存器（SPI1S）

	7	6	5	4	3	2	1	0
R	SPRF	0	SPTEF	MODF	0	0	0	0
W								
Reset	0	0	1	0	0	0	0	0

= Unimplemented or Reserved

7 SPRF	SPI 接收缓冲满标志位－当 SPI 数据传送完成后，SPRF 被置 1，说明接收到的数据可以从 SPI 数据寄存器读出。清除该位方法：先读取 SPRF 标志位，然后读取 SPI 数据寄存器 0：接收数据缓冲器中无有效数据；1：接收数据缓冲器中数据有效
5 SPTEF	SPI 发送缓存空标志位。当 SPI 发送缓存空时，该标志置 1。清除方法：通过读取 SP1S 寄存器，接着向 SPI1D 写入一个数据。在向 SPI1D 写数据时前，必须先读取 SPI1S，否则向 SPI1D 写数据被忽律。 如果 SPTIE 位置位，SPTEF 置位将产生一个 SPTEF 中断。 SPTEF 会自动设置当数据从发送缓冲转移到发送移位寄存器。 空闲模式下，当数据从发送缓存转移到移位寄存器，几乎同时该位被置位，从而允许下一个数据写入到发送缓冲器中。当在移位寄存器中的数据传送完毕后，那么发送缓冲器中的数据就会自动移到移位器中，SPTEF 置位。 0：SPI 发送缓存非空 1：SPI 发送缓存空
4 MODF	主机模式错误标志－当 SPI 配置为主机模式，从机选择引脚变低，表明其他 SPI 设备也配置为主机模式。SS 引脚作为模式错误标志当 MSTR=1, MODFEN=1, SSOE=0；否则 MODF 不会置位。通过读取 MODF 位，然向 SPI1C1 寄存器写数据，可以清除该位。 0：无模式错误 1：检测到模式错误

1、3、5 SPI 数据寄存器（SPI1D）

	7	6	5	4	3	2	1	0
R W	Bit 7	6	5	4	3	2	1	Bit 0
Reset	0	0	0	0	0	0	0	0

读取该寄存器返回接收数据缓存中的数据。写该寄存器将数据写入发送数据缓存。当 SPI 配置为主机，向发送数据缓存写数据将启动 SPI 发送。

不应向数据发送缓冲器写入数据，除非 SPTEF 置位。

SPRF 位置位之后，在传送另一个数据之前，可以读取 SPI1D 中的数据。如果在转移一个新数据之前，未从接收数据缓冲器中读出数据，那么将导致接收溢出，造成数据丢失。

1、4 功能描述

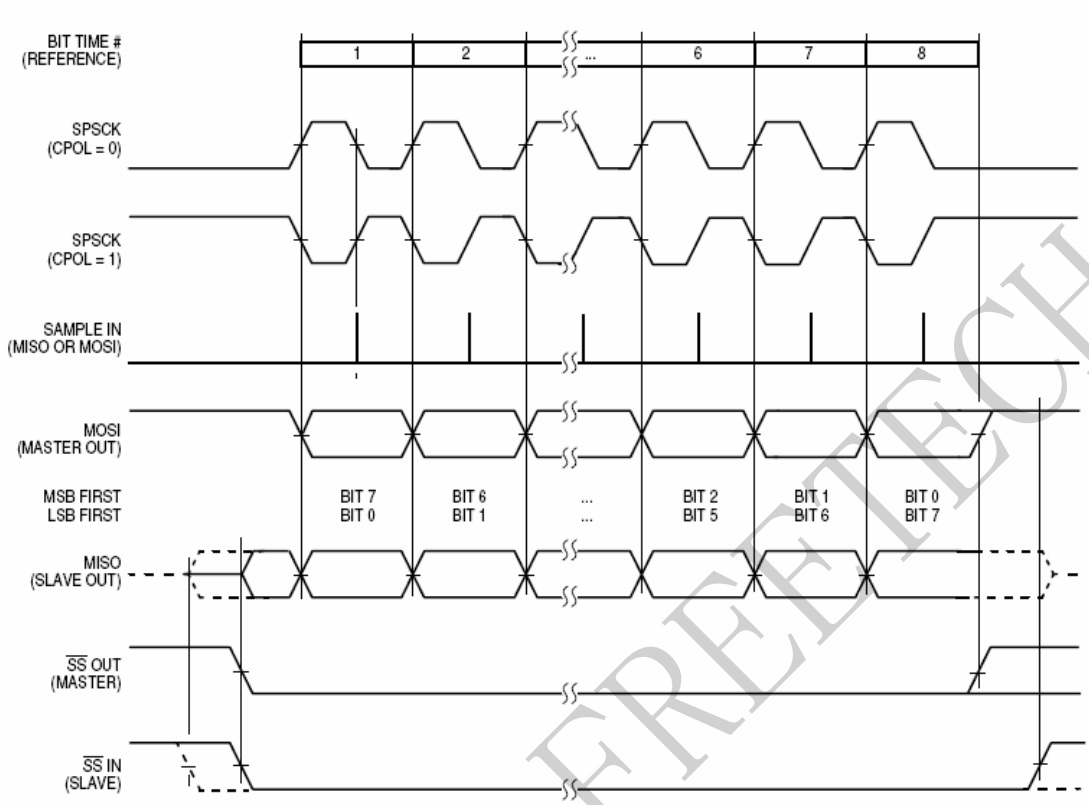
主机模式下，启动 SPI 数据转移的方法为：检查 SPI 发送缓冲器是否为空即 SPTEF 是否为 1，然后向 SPI 数据寄存器写入要发送的数据。当 SPI 的移位寄存器可用，该数据从发送数据缓冲转移到移位器，此时 SPTEF 会置位，表明发送数据缓冲可以接收另一个数据。SPI 数据发送开始。

在 SPI 发送期间，在 SPSCK 的边沿对 MISO 引脚进行抽样，移位；半个 SPSCK 时钟周期后，MOSI 引脚的电平改变为下一位值。8 个 SPSCK 周期后，主机移位寄存器中的数据从 MOSI 引脚移出，而从机将数据从 MISO 引脚移入到数据移位寄存器。在传送的最后，主机接收到的数据从移位器转移到接收数据缓冲，同时 SPRF 置位，表明数据可被读取。如果此时在发送缓冲有一个等待发送的数据，那么该数据转移到移位器中，SPTEF 置位，开始一个新的发送。

通常，SPI 数据先发送最高位。如果 LSBFE 置位，SPI 数据先发送最低位。

当 SPI 配置为从机模式，它的 SS 引脚在传送前必须被驱动到低电平，在整个传送期间，SS 必须处于低电平状态。当 CPHA=0，在连续传送之间，SS 必须被拉高。如果 CPHA=1，SS 在连续传送之间可保持低电平。

由于发送器和接收器是双缓冲。SPTEF 标志表明当发送缓冲有为新数据的空间。SPRF 标志表明在接收数据缓存中有一个有效数据。在下一次传送之前，接收到的字符必须从接收缓存中读出否则产生超载错误。**在超载情况下，新数据会丢失，因为接收缓存会一直保存着以前接收到的字符。因此应用程序开发者必须确保先前接收到的数据必须从接收缓存读出，才能启动下一次发送。**



SPI 时序图

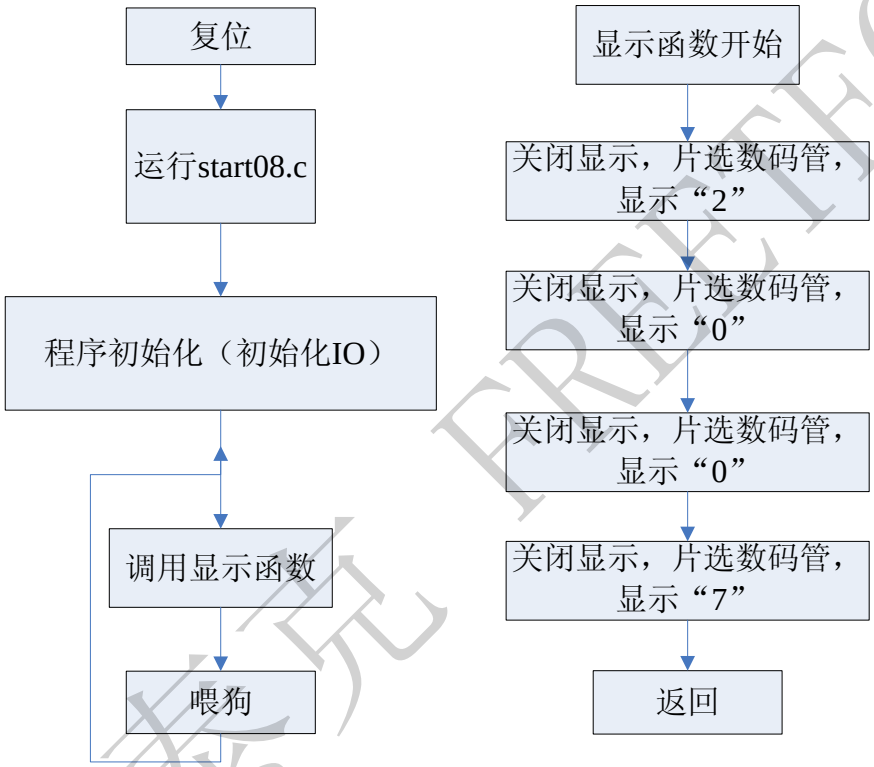
从上图可以看出，在 SPSCK 的第一个跳变沿，主设备将其移位器的第一个数据放到其 MOSI 引脚，从设备将其移位器的第一数据放到其 MISO 引脚；SPSCK 的第二个跳变沿，主，从设备对其 MISO 和 MOSI 引脚采样；在 SPSCK 的第三个跳变沿，主从设备对其采样到的数据移位，将第二个数据放到数据引脚。

实验十五 4 位数码管实验

1、实验功能介绍

本实验利用单片机的 IO 将要显示的数据发送给 74LS164，74LS164 驱动 4 位数码管；利用 74LS164 串入并出的功能实现了单片机 IO 的扩展。

2、流程图



3、实验现象

运行“显示实现”，4 位数码管显示“2007”。

4、程序代码

参见 4 位数码管实验

实验十六 单线协议 lin 实验

1、实验功能介绍

通过与子板 lin 接口进行通讯。

2、流程图

3、实现现象

4、程序代码

参见 lin 实验。

实验十七 汽车 CAN 总线实验

1、实验功能介绍

通过主板与从板的 can 通讯接口，实现 can 的通讯。掌握 can 通讯的流程。详细寄存器说明参见附件《MC9S08DZ60 中文数据手册》第 12 章。

2、流程图

见附件

3、实验现象

当数据祯正确子板根据主板发送数据进行点灯。

同主板灯指示一致。

下载程序时

3.1 取下主板上的 dz60 小板，断开 bdm 连线。

3.2 连接电源和 bdm 连线到子板。

3.3 下载 can2.mcp 到子板。

3.4 断开子板，装好主板上的 dz60 小板。连好 bdm。连好 can 接口线。

3.5 下载 can1.mcp 到子板。

4、程序代码

参见 can 实验。

第 5 章 注意事项

1、硬件方面

1、1 该学习板集成了程序下载调试器，所以在使用过程中用户应连接好 USB 数据线，BDM 线缆和安装好 CW6.0。

1、2 该学习板力求能够对 MC9S08DZ 系列单片机内部每个功能模块予以体现，在使用过程中会进行一些跳线，所以用户需熟悉学习板的跳线说明。

1、3 该学习板由底板和子板组成，其中底板上的 Open Source BDM 部分可脱离学习板作为一个独立的程序下载调试器，通过 6 芯线缆与用户的电路板连接，此时用户的电路板应外部供电。

2、软件方面

2、1 对于飞思卡尔新推出的 8 位单片机型号，已安装的 CW6.0 软件会不支持，那么可登陆 www.freescale.com 网站下载对应新型号的升级补丁即可。

2、2 用户如果在同一台计算机上安装了不同版本的 CW 软件，那么在用户双击.mcp 文件时会调用最新安装的版本。如果用户想用以前的版本打开.mcp 文件，请更改.mcp 文件的打开方式。

2、3 CW6.0 软件提供了丰富的帮助文档，用户可使用该文档解决遇到的问题。

2、4 用户在使用过程中遇到问题可通过以下方式联系：Tel：010-59831537－8011 或者 email: huadong@free-tech.com.cn