文档编号：AN_121

上海东软载波微电子有限公司

应用笔记

ES7H202x

修订历史

版本	修订日期	修改概要
V1.0	2020-3-11	1. 初版发布
V1.1	2020-9-21	1. 增加 IAP 操作擦写 Flash 的注意说明
V1.2	2025-1-8	1. 更新 ADC 模块应用注意事项的描述

地　址：中国上海市徐汇区古美路 1515 号凤凰园 12 号楼 3 楼
邮　编：200235
E-mail：support@essemi.com
电　话：+86-21-60910333
传　真：+86-21-60914991
网　址：http://www.essemi.com

目 录

第1章 ES7H202x应用注意

1. 1 内部振荡器

ES7H202x 芯片在出厂时已做好内部振荡器的校准，内部高速 RC 振荡器校准精度 16MHz±0.5%@25℃，2.2V~5.5V；16MHz±2%@40℃~85℃，2.2V~5.5V。内部低速 RC 振荡器校准精度 32KHz±5%@25℃，2.2V~5.5V。

芯片上电复位后，内部 16MHz RC 振荡器(HRC)默认作为系统时钟源。可通过配置 OSCC1 寄存器中的 SCKS 位切换系统时钟源。

LRC 时钟固定为使能，无法关闭。

1. 2 复位模块

1. 2. 1 外部MRSTN管脚复位

复用端口 MRSTN/PA4 作为外部 MRSTN 使用时，该端口内部上拉电阻（约 18K 欧）固定为使能，因此用户的复位电路 MRSTN 引脚外接上拉电阻时，需考虑该内部上拉电阻的阻值。

我们建议用户使用具有低电压检测功能的复位芯片作为外部复位电路。

当系统有低成本的要求时，也可以采用以下电路替代电压检测芯片。

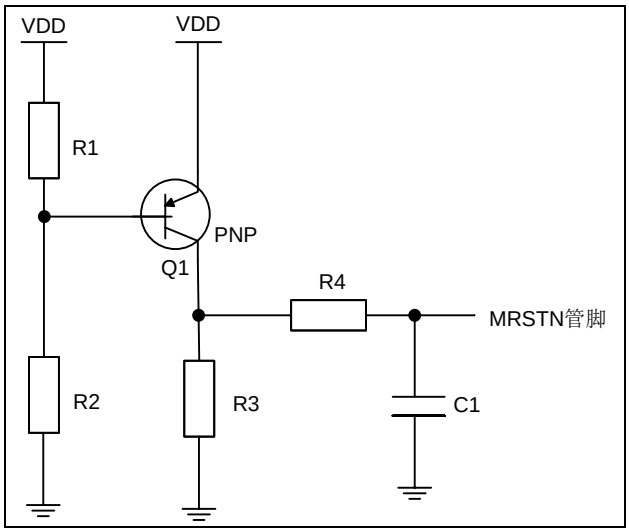

图 1-1 MRSTN 复位参考电路图

电压检测原理：当 VDD 电压下降，导致 R1 两端电压< 0.7V 时，PNP 晶体管截止， MRSTN 引脚被 R3 电阻下拉至低电平，使芯片处于复位状态。

复位电压点应满足 [R1/(R1+R2)]×VDD < 0.7V 这个条件，用户可以此进行复位电压和匹配电阻的计算。

举例：

选定 R1=2K，R2=10K，R3=20K，R4=1K，复位电压应满足：

[2K/(2K+10K)]×VDD < 0.7V，通过计算可以得到，当 VDD < 4.2V 时，PNP 晶体管处于截止状

态，MRSTN 被拉至低电平，可保证芯片处于复位状态。

1. 2. 2 BOR复位

BOR 掉电复位模块监控施加于芯片电源上的电压，一旦芯片的工作电压超出所设定的电压范围，则产生欠压复位，这样可以防止芯片 IO 端口的非正常输入/输出，有效增强系统的抗干扰性能，提高系统的稳定性。

BOR 固定为使能，建议客户设置 BORVS 在合理的电压点，以免芯片因外界干扰或电源波动而工作异常。

1. 2. 3 WDT复位

建议客户在设计产品时使能 WDT 功能。

1. 3 通用数据存储器SRAM

SRAM 的最后 1 个存储体组 SECTION 11（地址范围 0580H~05FFH），从高地址（05FFH）开始的部分或全部地址空间可分配用于 PC 硬件堆栈，用户通过配置字 STKLS 选择 PC 硬件堆栈的级数。当 STKLS 选定后，PC 硬件堆栈占用的 SRAM 地址空间是堆栈级数的 2 倍。

PC 硬件堆栈占用的 SRAM 空间程序不能读写。

1. 4 FLASH程序存储器

在线编程 ISP 模式下，FLASH 程序存储器可以通过设置配置字进行全加密和分区加密。

对已加密分区的页，IAP 读出值固定为 0。对设置了 FLASH 全加密，但没有分区加密的页，仍可通过 TBR 指令进行访问。

1. 5 低功耗模式

- 如果产品封装管脚数小于 32，未引出的 I/O 管脚需设置为输出低电平。
- 实际应用系统中，未使用的 I/O 管脚需设置为输出低电平。

1. 6 唤醒方式操作

- 在进入 IDLE 之前，需注意清零相关中断标志位，并关闭中断使能位，以免误唤醒芯片。
- 唤醒 IDLE 后，默认系统时钟可由芯片配置字来确定使用 LRC 时钟，还是进入睡眠前的系统时钟。

1. 7 外设时钟

系统程序中一定不能反复使能和关闭外设时钟，否则会有程序运行异常的风险。同理，在睡眠执行休眠指令前，也无需软件关闭外设时钟，休眠指令会关闭 Fosc 时钟，外设时钟也被自动停止。

1.8 中断处理

1.8.1 中断标志的清除

用户在使能中断前需先清除相应的中断标志，避免中断的误触发。

除只读的中断标志（由硬件清除）外，其余的中断标志必须通过软件清除。

1.8.2 GIE位和GIEL位处理

用户通过软件对中断使能位 GIE 或 GIEL 进行写零操作的时刻，如果同时发生了中断响应，则芯片会优先响应中断，本次软件写零操作无效。为确保对中断使能位 GIE 和 GIEL 的软件写零操作成功，推荐的实现方式如下：

```
while(GIE == 1)
{
    GIE = 0;
}
while(GIEL == 1)            //仅使用了向量中断才需要此语句
{
    GIEL = 0;
}
.....
GIEL = 1;                   //仅使用了向量中断才需要此语句
GIE = 1;
```

用户在对 GIE 和 GIEL 的操作中，一定要严格按照上面例程的顺序进行。

1.9 ADC模块

- 系统电源 VDD 纹波不宜过大， 否则会影响 ADC 转换结果。
- 仿真调试时仿真端口通信可能会导致 VDD 纹波过大，从而影响 ADC 转换结果，需确认系统脱机运行结果是否正常，如果正常，则可忽略仿真调试过程中的 ADC 转换问题。
- 在 A/D 转换使能位 ADEN 使能后，ADC 需要先完成自身工作建立，才能得到正确的转换结果。上述使能控制信号使能后，延时 30us 以上，启动第一次 ADC 转换（ADTRG=1），转换结束后，再延时 100us 以上，ADC 工作建立完成，后续启动 ADC 转换，即可得到正确的转换结果。

  由于 ADC 建立过程中得到的转换结果与理论值偏差极大且不可预知，所以在应用程序中需要丢弃 A/D 转换使能位 ADEN 使能后的第一次转换结果。

  因每次 A/D 转换使能位 ADEN 重新使能后，均需要执行上述 ADC 工作建立过程，所以应用中，在芯片正常运行时不建议关闭上述使能控制信号，保持 ADEN 为 1，只在进入深睡眠模式前，关闭 ADC。

- 当应用系统中对 ADC 参考电压精度和温漂特性要求高时，建议选用 VDD 或外部 VREFP 作为 ADC 模块的正参考电压。

1. 10 未使用的IO端口处理

系统中未使用的 IO 管脚建议设置为输出固定电平并悬空， 若设置为输入，须加上拉或下拉电阻接到电源或地，电阻值需小于 1K 欧姆。

1. 11 PC0 端口弱上拉处理

用户如果需要使用 PC0 端口的弱上拉功能，除了配置 PC0 为数字口，使能弱上拉外，还需同时将 PC1 口也设置成数字口（PCS 寄存器的 bit1 置 1）。

1. 12 I2C模块

I2C 从动模块支持 7 位从机地址匹配，由 I2C 主机控制发送或接收数据。

当主机向从机发送数据时，从机通常判断 I2CRBIF 标志，如果接收缓冲器不空，即接收到主机数据，则读接收缓冲器的数据。

当主机读取从机数据时，从机通常判断 I2CTBIF 标志，如果发送缓冲器未满，则依次写入需要发送的数据。

为了避免误发数据，建议每次完整的通讯结束（例如收到 STOP 标志），就采用 I2CRST 置位复位一次 I2C 模块来清空接收和发送数据缓冲器，同时再重新初始化 I2CC 和 I2CIEC 寄存器，为下次 I2C 通讯做好准备。

例程中低速传输用的是默认的中断模式，需要注意的是默认中断模式的中断入口只有一个，不同的中断源所产生的中断都会进入到同一个中断函数进行处理，如果前一个中断相关的程序执行的太久便会影响别的中断相关程序的执行。为了降低 I2C 做从机时不同中断处理对其传输速度影响，在设计程序时尽量首先判断 I2C 的中断标志位，以首先执行 I2C 中断相关程序；尽量减少每个中断源对应的中断程序运行的时间；尽量在判断不同中断源（例如定时器和 I2C 中断）时，使用如下格式的语句判断：

“if(I2CIF&&I2CIE)
{;}
else if(T10IF && T10IE)
{;}”。

使用默认中断模式传输时，考虑到不同中断源之间的影响，建议传输速度不要大于 10KHz。

实际开发中，当主机读从机，I2C 从机模块进入到地址匹配中断（I2CSRIF==1）之后，从机往往会有一些操作（比如对即将发送的数据做处理），造成数据短时间不能写入到 I2CTB 寄存器内，如果延迟时间超出了主机的最大等待时间，传输便会失败。低速传输时，主机有着充足的时间等待从机发送数据，当高速传输时，往往就会出现上述问题。解决这个问题可以使用 iDesigner 自带的优化处理方法。用户需要使用最新编译器，在编译器“项目->编译->Support interrupt vectors”选择“true”和“项目->编译->IIC slave high speed mode”选择“true”。使用编译器优化之后，当主机读从机时，从机只需要在写入数据到 I2CTB 寄存器之后释放 SCL 引脚（置位 I2CTE），主机接

下来便会开始接收从机所发的数据。

需要特殊注意的是，若使用编译器优化，在编程时用户必须使用向量中断，中断全局寄存器 INTG 中的 INTV<1:0>必须为 0b11，使得 I2C 的中断优先级位于最高位，而优先级比 I2C 中断 IG6 高一级的 IG7 中断对 I2C 的传输速度也会产生影响，所以用户应尽量避免使用 IG7 中断。不同的应用环境下 I2C 的传输速度会有差别，高速传输时建议传输的最大速度小于 600KHz，当用户传输的速度大于 400KHz 时，建议关闭 I2C 滤波功能（I2CX16 = 0）。

PC0 和 PC1 端口的输入施密特窗口大于 PC6 和 PC7，建议用户优先选用 PC0 和 PC1 作为 I2C 管脚。

如果需要使能 I2C 数据端口 SDA 为内部弱上拉，建议用户通过如下方式实现：设置 I2C 开漏输出使能 I2COD=1，并且设置复用为 SDA 功能的对应 IO 端口的内部弱上拉控制位为使能 PCPUx=1。禁止通过配置 I2CPU=1 使能 I2C 内部弱上拉。

1. 13 ICD调试注意事项

在 ICD 调试模式下，如果产生断点暂停，T10 停止计数，T11/T12/T13/T20/T21 均保持计数。

1. 14 硬件除法器模块

硬件除法器完成 32 位被除数 DIVE（由 4 个 8 位寄存器 DIVEU，DIVEH，DIVEM，DIVL 组成）除以 16 位除数 DIVS（由 2 个 8 位寄存器 DIVSH，DIVSL 组成）的操作，所得结果 32 位商存储于 32 位寄存器 DIVQ（由 4 个 8 位寄存器 DIVQU，DIVQH，DIVQM，DIVQL 组成），16 位余数存储于 16 位寄存器 DIVR（由 2 个 8 位寄存器 DIVRH，DIVRL 组成）。

硬件除法器为多节拍除法器，完成 1 次除法运算最多需要 36 个机器周期。写入除数和被除数后，通过控制寄存器 DIVC 启动除法运算，除法运算完成后结果自动载入相应结果寄存器，同时触发除法完成中断 DIVIF。用户可通过相应状态位判断运算过程是否出错。

常规操作步骤如下：

1）设置寄存器控制位 VGO（DIVC<0>）为 1，启动除法运算；

2）等待 36 个指令周期；

3）软件清零 VGO 位；

4）读取除法运算结果。

```
typedef union {
        uint16_t hWord;
        uint8_t    chByte[2];
}hword_to_byte_t;

typedef union {
        uint32_t Word;
        uint8_t    chByte[4];
}word_to_byte_t;

word_to_byte_t TC_dive;
```

```
word_to_byte_t TC_divq;
hword_to_byte_t TC_divs;
hword_to_byte_t TC_divr;

void DIV(uint32_t dive, uint16_t divs,uint32_t *divq, uint16_t *divr)
{
    uint8_t i;
    TC_dive.Word = dive;
    TC_divs.hWord = divs;
    memcpy(&DIVEL,TC_dive.chByte,4);
    memcpy(&DIVSL,TC_divs.chByte,2);
    VGO = 1;
    /*等待36个时钟周期，实测39个时钟周期*/
    for(i = 0;i<4;i++)
    {
        __asm{NOP}
    }
    VGO = 0;
    memcpy(TC_divq.chByte,&DIVQL,4);
    memcpy(TC_divr.chByte,&DIVRL,2);
    *divq = TC_divq.Word;
    *divr = TC_divr.hWord;
}

int main()
{
    uint32_t param_dive; //被除数
    uint16_t param_divs; //除数
    uint32_t param_divq; //商
    uint16_t param_divr; //余数
    param_dive = 0x12345678;
    param_divs = 0x1234;
    DIV(param_dive,param_divs,&param_divq,&param_divr);
    while(1);
}
```

1. 15 单步调试进定时器中断

T10，T11/T12/T13，T20/T21 这些定时器在调试暂停状态下仍然运行，这就导致了单步调试时可能会进入定时器中断函数，给单步调试带来不便。为了避免这种情况，可在 iDesigner 的调试菜单下的条件断点选项里勾选"单步执行跳过中断函数"，这样在单步执行时不会进中断函数，也不会停在中断函数的断点处，但是全速运行时可以进中断并且也可以停在中断函数的断点处。

1. 16 硬件乘法器

用户在使用芯片硬件乘法器时，要注意中断服务程序可能会改变乘数寄存器 MULA 和 MULB，最终导致程序运行获取到一个错误的乘积。用户有二种方式来规避这种风险。

方式一：用户在使用硬件乘法器之前，先禁止全局中断使能（GIE=0），以免在中断处理过程中，乘数寄存器被改写。乘法运算完成后，将乘积读出，再恢复全局中断使能（GIE=1）。

方式二：用户在使用硬件乘法器之前，先将乘数和被乘数备份在特定的变量中。这样，编译器会在中断服务程序中自动备份和恢复乘数寄存器。注：需使用 v1.2.0.113 及以上版本的工具链。

方式二示例如下：

```
unsigned char __MULA__ @0x0;
unsigned char __MULB__ @0x1;
#define SET_MULA(a) {__MULA__ = (a); MULA = __MULA__;}
#define SET_MULB(a) {__MULB__ = (a); MULB = __MULB__;}

main()
{
SET_MULA(12);
SET_MULB(15);
}
```

为了方便用户使用，变量声明和宏定义，都已经添加在芯片的头文件中了。用户在实际使用时，只需执行 SET_MULA(12)和 SET_MULB(15)这二处即可。

第2章 ES7H202x模块例程

2. 1 Flash IAP

功能说明：

ES7H202x 内部最大有 17K 字节数据 FLASH 存储器，分 34 页，每页 512 字节，支持查表读写操作，地址范围为 0000H~43FFH。

注：由于程序在编译时，默认将算法库函数分配在 0x4000-0x43FF，因此不建议客户对此区域进行 IAP 操作，以免造成不可预期的程序运行异常。

由于 ES7H202x 全 FLASH 区支持读写操作，因此建议用户定义数组作为数据扇区的空间预留，以免 IAP 操作地址与代码区域冲突。预留区域定义方法如下：const unsigned char DataFlash[128] @ 0x3E00 = {0,0.....,0,0};

当配置位 IAPEN 使能时，ES7H202x 的程序存储器可以进行擦写和读取的访问操作。如果某个地址已经被写入数据，则改写该地址的数据前，需要预先对该地址所属的页进行擦除操作。

查表读操作通过查表读指令将 FRA 所指向的地址单元中的一个字读入 ROMD 中。

数据 FLASH 的写以字为单位，写操作前必须先擦除所写单元所在的页，故写数据 FLASH 时包含三个基本操作：数据备份，页擦除和字编程。

芯片页擦除时间至少为 2ms，此期间芯片处于暂停状态。

当 FLASH 存储器进行 IAP 擦除或 IAP 写入操作时 CPU 内核暂停执行，外设可按预设状态继续运行，外设的中断请求将置位相应的中断标志。当 IAP 擦除或 IAP 写入操作完成时，CPU 内核恢复执行。

在使用 IAP 进行 flash 空间的读/擦/写期间，需要临时关闭中断，即在需要操作 FRAH/FRAL/ROMDH/ROMDL 寄存器进行一轮 IAP 读/擦/写之前临时关闭中断，直到一轮 IAP 读/擦/写结束后才可恢复中断。

对数据 FLASH 存储器读/写操作前，芯片配置字中需使能 FREN 位。

在编译中断子程序的过程中，编译器会保存 FRAH/FRAL 的地址，并在退出中断时，通过 TBR 指令，再次读取 FRAH/FRAL 对应地址单元的值到 ROMDH/ROMDL 中。因此不建议用户在完成读、写、擦操作后更改 FRAH/FRAL 的值。从而对程序的功能造成影响。

推荐	不推荐

例 1：u8 ReadFlash()
{
while(GIE == 1)
{
GIE = 0;
}
FRAH = 0x40;
	u8 ReadFlash()
	{
	while(GIE == 1)
	{
	GIE = 0;
	}
	FRAH = 0x40;

```
    FRAL = 0x00;
    __asm
    {
        TBR
    }
    FlashData.Byte[1] = ROMDH;
    FlashData.Byte[0] = ROMDL;
    GIE=1;
    retrun   ROMDL;
}
```

```
    FRAL = 0x00;
    __asm
    {
        TBR
    }
    FlashData.Byte[1] = ROMDH;
    FlashData.Byte[0] = ROMDL;
    FRAH=0X47;
    FRAL=0X00;
    GIE=1;
    return ROMDL;
}
```

在上面的例程中，执行右边的程序，会发现最后返回的 ROMDL 结果并不一定是 0x4000 地址单元存储的数据，在打开 GIE 后，如果此时有一个中断需要响应，在退出中断子程序之前会执行“隐含”的 TBR 指令，而操作对象，正是在打开 GIE 之前刚被修改的 0x4700 地址，因此在执行完中断子程序再返回到这段代码后，程序返回的将是 0x4700 地址单元的数据。

为了保证编程及擦除操作的正确执行，芯片加入了地址取反机制，该机制有效保证了编程及擦除操作的可靠性。建议用户在编写程序时，尽可能采用立即数赋值的方式，以最大程度的发挥该机制的优势。

推荐	不推荐
例 1：addr = 0x2E00; _addr = 0xD1FF; FRAH = (addr >> 8) & 0xFF; FRAL = addr & 0xFF; FRAHN = (_addr >> 8) & 0xFF; FRALN = _addr & 0xFF ;	addr = 0x2E00; FRAH = (addr >> 8) & 0xFF; FRAL = addr & 0xFF; FRAHN = (~addr >> 8) & 0xFF; FRALN = ~addr & 0xFF ;

例 2：int Main()
{
U16 temp,addr,_addr;
addr = temp;
_addr = ~temp;
Erase(addr,_addr);
While(1);
}
void Erase(u16 addr,u16 _addr)
{ ...
FRAH = (addr >> 8) & 0xFF;
FRAL = addr & 0xFF;
FRAHN = (_addr >> 8) & 0xFF;
FRALN = _addr & 0xFF;
...
}
	Int Main()
	{
	U16 addr;
	Erase(addr);
	While(1);
	}
	void Erase(u16 addr)
	{
	...
	FRAH = (addr >> 8) & 0xFF;
	FRAL = addr & 0xFF;
	FRAHN = (~addr >> 8) & 0xFF;
	FRALN = ~addr & 0xFF ;
	...
	}

建议用户在执行页擦除操作时关闭 WDT，避免执行擦除操作时因看门狗复位导致芯片误操作。

建议用户在执行擦写操作前关闭总中断使能位 GIE，避免执行擦写操作时因中断导致芯片误操作。关于 Flash 存储器的可靠性操作方法详见《AN062_应用笔记_MCU 片内非易失性存储器操作》

注：阈值电压设置是为了验证当用户操作电压低于设置值时，flash 擦写将出现错误，在线调试时可通过设置断点的形式观测现象。本例程已将阈值电压的设置屏蔽掉，若用户想要在线调试复现 flash 擦写错误现象，可取消阈值电压的屏蔽，并配合 IDesigner 工具栏中的<工具_仿真电压设置>选项卡使用。

实现步骤：

a) 用 eeprom.asm 程序 CSEG 和 DW 伪指令初始化 0x0A00 起始地址的数据

b) 用查表读指令将一页内容备份至数据存储空间

c) 修改数据存储空间需更新的数据，16 个字节

d) 执行页擦除（需按固定指令序列进行）

e) 通过设置寄存器 FRAL 和 FRAH 选择需要更新的地址，及设置寄存器 ROMDL 和 ROMDH 需要更新的数据

f) 将 ROMDL 和 ROMDH 中的内容按固定的写序列写入 FRA 所指定的页中地址

g) 重复 d)，e)直至完成整页编程

h) 用查表读指令读出数据

2. 2 ADC程序模块

功能说明：

使用 ES7H202x 芯片的 ADC 模块，采用查询方式实现对模拟输入电压的数字量转换。ADC 转换包括采样和转换两个过程。

ES7H202x 最多支持 14 个输入通道，最大转换分辨率为 12bit。本例程采用双通道(1/4 VDD 通道、AIN4 通道)进行 ADC 演示，其中 1/4 VDD 通道采样，以内部参考 VREF 为基准，通道 4 以 VDD 为参考基准电压。

注：若两通道同时采样，且参考基准电压不同，会导致转换结果误差较大，因此本例程屏蔽了 1/4 VDD 通道采样转换配置。

实现步骤：

a) 初始化系统和端口

b) 初始化 ADC 模块并对相应的寄存器赋值

c) 使能 ADC 模块，启动 A/D 转换

d) 查询并等待 A/D 转换结束

e) 保存 A/D 转换结果

2. 3 UART通信

功能说明：

异步收发器支持 8/9 位数据格式，支持高速低速模式。波特率计算公式：

低速模式：BRGHn=0，baud rate=Fosc/(64×(BRnR<7:0>+1))

高速模式：BRGHn=1，baud rate=Fosc/(16×(BRnR<7:0>+1))

UART 通讯时，接收中断标志 RXnIF 由接收数据寄存器的读操作清零或 RXnEN 置零来清零。接收数据必须及时读取，否则可能导致接收 FIFO 溢出。发送中断标志 TXnIF 由该位的写操作，或 TXnEN 置零来清零。

注：本例程以 UART1 为例进行演示，其余三组只需在此例程基础上配置相应数据接收和发送端口，并对应更改 UART 寄存器即可。

实现步骤：

a) 初始化 PC 口为数字 I/O，并将 TX0 方向(PC7)设成输出，RX0 方向(PC6)设成输入

b) 设置波特率 9600，高速模式，波特率寄存器 BR0R=103

c) 设置 8/9 位数据格式

d) 使能发送/接收器

e) 发送时，发送缓存为空时写入发送数据；接收时，通过查询 RX0IF 中断标志位来判断是否收到完整的一帧数据

2. 4 I2C从动模块

功能说明：

当 I2CSRIF 标志位置 1，且 I2CRW 位为 0 时表示主机写从机，即从机接收数据。从机接收数据缓冲器满时，中断标志位 I2CRBIF 置位，读取一次接收数据缓冲器 I2CRB，I2CRBIF 标志位自动清零，其他使用注意事项请参考芯片数据手册。

当 I2CSRIF 标志位置位，且 I2CRW 位为 1 时表示主机读从机，即从机发送数据。I2CTB 发送数据缓冲器不满时，中断标志位 I2CTBIF 置位，写发送数据缓冲器 I2CTB，若数据缓冲器写满，I2CTBIF 标志位自动清零，若发出一个字节，数据缓冲器未满，则中断标志位 I2CTBIF 置位。

从机接收到 STOP 信号时，收发数据缓冲器不会自动清零，需置位 I2CRST 位来软件复位 I2C 模块。复位后，需重新初始化 I2C 模块。

关于 I2C 的传输共有两个例程即高速 I2C 传输和低速 I2C 传输用例，每个例程主机写一个数据给从机，再读回两个数据。从机将接收到的数据以原码和反码的形式发回给主机。

低速传输实现步骤：

a) 设置 SCL，SDA 所在的端口方向为输入

b) 设置从机的地址

c) 初始化从 I2C 模块，使能 I2C 模块及 I2C 通讯

d) 设置 I2C 中断使能寄存器 I2CIEC

e) 若收到地址匹配中断，清地址匹配中断标志， I2CRW 为 0 时写从机，使能 I2CRBIE，禁止 I2CTBIE；I2CRW 为 1 时读从机，禁止 I2CRBIE，使能 I2CTBIE

f) 若主机读从机，从机写 I2CTB 数据缓冲器发送数据

g) 若主机写从机，从机读 I2CRB 数据缓冲器接收数据

h) 若收到 STOP 中断， 清 STOP 中断标志， I2CRST 执行一次置位，复位 I2C 模块，清空收发数据缓冲器，禁止 I2C 中断使能寄存器，然后重新初始化 I2CIEC 和 I2CC 寄存器

高速传输实现步骤：

a) 设置编译器“项目->编译->Support interrupt vectors”为“true”；

“项目->编译-> IIC slave high speed mode”加入“true”

b) 设置 SDA 所在的端口方向为输入，SCL 引脚为输出，低电平

c) 设置从机的地址

d) 初始化从 I2C 模块，使能 I2C 模块及 I2C 通讯

e) 设置 I2C 中断使能寄存器 I2CIEC

f) 若收到地址匹配中断，清地址匹配中断标志， I2CRW 为 0 时写从机，使能 I2CRBIE，禁止 I2CTBIE；I2CRW 为 1 时读从机，禁止 I2CRBIE，使能 I2CTBIE，释放 SCL 引脚

g) 若主机读从机，从机写 I2CTB 数据缓冲器发送数据，程序需要在写入 I2CTB 数据之后释放 SCL 引脚，主机在从机释放 SCL 引脚以后开始读数据

h) 若主机写从机，从机读 I2CRB 数据缓冲器接收数据

i) 若收到 STOP 中断， 清 STOP 中断标志， I2CRST 执行一次置位，复位 I2C 模块，清空收发数据缓冲器，禁止 I2C 中断使能寄存器，然后重新初始化 I2CIEC 和 I2CC 寄存器

2. 5 看门狗定时唤醒（WDT）

功能说明：

通过 PA1 口输出周期为 16ms，占空比为 50%的方波观察定时唤醒效果。需打开配置字中的 WDT，在 IDLE 模式下，WDT 计数溢出会唤醒 CPU，每次对 WDTC 寄存器进行写操作前都必须对 WDT 解锁，即向 WDTUL 寄存器写入 0xA5。

实现步骤：

a) 清零 RAM

b) 初始化 PA1 端口

c) 初始化 WDT

d) 进入 IDLE 模式，并等溢出唤醒

2. 6 向量中断模式

功能说明：

每组硬件中断可以分别设置高低优先级，响应中断嵌套。本例程实现外部中断PINT0以及定时器T10中断的配置，在触发外部中断之前，通过T10计数溢出中断在PA1端口输出周期为4ms，占空比为50%的方波，当触发外部中断时PA1端口电平置1。

注：当用户新建向量中断项目时，需将项目属性下<编译>选型卡下的Support interrupt vectors选项置为ture（然后切记点击保存工程，否则该操作将不会被记录，编译将会出错）。

实现步骤：

a) 清零 RAM

b) 初始化向量中断模式

c) 初始化 PA1 为数字输出 I/O，PC0 为数字输入 I/O

d) 初始化外部中断 PINT0 并使能，初始化 T10 定时器中断并使能

e) 配置向量入口地址，并根据相应中断程序翻转 I/O 电平，清标志位

2. 7 T10 定时器

功能说明：

使用芯片的 T10 定时器模块，在 PA1 端口输出一个周期为 4ms，占空比为 50%的方波。芯片使用 16MHz 系统时钟，则对应的 T10 定时器时钟源周期为（0.0625 × 2）us。将预分频器分配给 T10 定时器，分频比为 1：64。T10 寄存器初始值的计算公式为：

2ms/（0.0625 × 2）us = （255 - T10 + 1）×（预分频比），计算得到T10 = 6（0x06）。

实现步骤：

a) 清零 RAM

b) 初始化 PA1 为数字输出 I/O

c) 初始化定时器模式，设置预分频比，对寄存器赋初值

d) 使能中断，进入中断，清标志位，翻转 I/O

2. 8 多功能定时器T11

12 位多功能定时器 T1n 支持 5 种工作模式：定时器模式、异步计数器模式、双精度 PWM 模式、互补 PWM 模式和单脉冲发射模式。双精度 PWM 模式下，每路可同时支持 2 个独立的占空比匹配值，互补 PWM 模式支持可编程死区控制。

2.8.1 T11 定时器

功能说明：

使用芯片的 T11 定时器模块，在 PA1 端口输出一个周期为 4ms，占空比为 50%的方波。芯片使用 16MHz 系统时钟，则对应的 T11 定时器时钟源周期为 0.0625us。将预分频器分配给 T11 定时器，分频比为 1：64。T11P 寄存器初始值的计算公式为：

2ms/0.0625us = T11 × 64（后分频比），计算得到T11P = 500（0x01F4）。

实现步骤：

a) 清零 RAM

b) 初始化 PA1 为数字输出 I/O

c) 初始化定时器模式，设置预分频比，对寄存器赋初值

d) 使能中断，进入中断，清标志位，翻转 I/O

2.8.2 T11 双精度PWM

功能说明：

使用芯片的双精度 PWM 模式，在 PB0 端口及 PB1 端口实现周期为 100us~50us，占空比分别为 25%及 50%的方波输出（其中周期分四次递减至 50us，再将周期重新装载为 100us，占空比不变，以此反复）。

芯片使用 16MHz 系统时钟，则对应的 T11 定时器时钟源周期为 1/16MHz。T20 的预分频采用：1：1，T11 周期寄存器 T11P 初始值计算公式应为：

PWM 周期 100us = T11P × （1/16MHz）× 1（预分频比），计算得 T11P 初值 PERIOD = 1600（0x0640）。

PWM 脉宽 50us = T11R × （1/16MHz）× 1（预分频比），计算得 T11R 初值 DUTY11 = 800（0x0320），这里以 PB1 端口的输出方波为例。

PWM 占空比 50% = T11R / T11P

实现步骤：

a) 清零 RAM

b) 声明周期、精度变量

c) 初始化 PB0、PB1 为 PWM 输出口

d) 设置双精度 PWM 模式、使能并设置预分频比

e) 设置周期与精度初值

f) 进入中断更新 PWM 周期

2.8.3 T11 互补PWM

功能说明：

使用芯片的 T11 互补 PWM 模式，在 PB0、PB1 端口实现周期为 100us，死区时间为 2us，占空比为 80%~20%分四次递减，再逆向递增以此反复的方波输出。芯片使用 16MHz 系统时钟，则对应的 T11 定时器时钟源周期为 1/16MHz。T11 预分频采用 1：1，T11 周期寄存器 T11P 初始值的计算公式应为：

PWM 周期 100us = T11P × （1/16MHz）× 1（预分频比），计算得 T11P 初值 PERIOD = 1600（0x0640）。

PWM 脉宽 80us = T11R0 × （1/16MHz）× 1（预分频比），计算得 T11R0 初值 DUTY =1280（0x0500），这里以 PB1 端口的输出方波为例。

PWM 占空比 50% = T11R0 / T11P

PWM 死区延时 2us = T11R1 ×（1/16MHz）

实现步骤：

a) 清零 RAM

b) 初始化 PB0、PB1 为数字输出 I/O

c) 设置互补 PWM 输出模式

d) 使能 PWM 输出

e) 设置周期、精度、死区延时初值

f) 使能中断，进入中断，清零中断标志位，循环更新精度值

2.8.4 T11 单脉冲发射

功能说明：

使用芯片的 T11 单脉冲发射模式，每路支持 2 个单脉冲输出端口 PWM1n0 和 PWM1n1， 每个输出端口均可独立设置输出极性。本例程以 PW111 输出为例，在 PB1 口输出一个精度为 200us 的单脉冲。并且以外部中断的方式触发脉冲发射，中断一次，输出一次单脉冲。

等待周期 = T1n 计数时钟周期 x (4096 x T1nR0+ T1nR1)

发射周期 = T1n 计数时钟周期 x T1nP

PERIOD =（1/16MHz）× 2（预分频系数）× T11P

实现步骤：

a) 清零 RAM

b) 初始化 PB1 为数字输出口

c) 初始化 PC1 为数字输入口

d) 使能外部中断，选择中断触发方式

e) 进入中断，标志位置 1

f) 进入循环触发单脉冲发射

2.8.5 T11 PWM自动关断和重启

功能说明：

芯片支持两类关断事件，EPAS0/1 管脚输入“0”关断事件和 PINT 触发关断事件。本例程在双精度 PWM 输出的基础上，实现 T11 PWM 自动关断和重启的功能。关断前 PWM111（PB1）输出周期 100us，占空比为 50%的方波。EPAS0（PE0）端口为 0 时触发关断事件，PWM111 输出口被持续置 1，关断事件撤销并且关断标志位被清零后，在下一个 PWM 周期正常输出。

PWM 周期 100us = T11P × （1/16MHz）× 1（预分频比），计算得 T11P 初值 PERIOD= 1600

（0x0640）。

PWM 脉宽 50us = T11R × （1/16MHz）× 1（预分频比），计算得 T11R 初值 DUYT11 = 800（0x0320）

PWM 占空比 50% = T11R / T11P

实现步骤：

a) 清零 RAM

b) 初始化 PB1 为数字输出 I/O

c) 初始化 PE0 为数字输入 I/O

d) 设置双精度 PWM 模式，设置周期精度

e) 设置 PWM 自动关断和重启模式

2.8.6 T11 异步计数

功能说明：

使用芯片的 T11 异步计数模式，通过定时计数溢出唤醒芯片，设置定时时间 50ms 并翻转一次 PB0 端口电平，以此反复循环输出。

实现步骤：

a) 清零 RAM

b) 切换外部时钟

c) 初始化 I/O 端口

d) 设置异步计数模式，并赋异步计数初值

e) 进入低功耗模式，等待溢出唤醒，翻转 I/O

2. 9 多功能定时器T20

16 位多功能定时器 T2n 支持 7 种工作模式，定时器模式、同步计数器模式、异步计数器模式，双精度 PWM 模式、互补 PWM 模式、捕捉器模式、比较器模式。

2.9.1 T20 定时器

功能说明：

使用芯片的T20定时器模块，在PC1端口输出一个周期为4ms，占空比为50%的方波。芯片使用16MHz系统时钟，则对应的T20定时器时钟源周期为0.0625us，设置预分频比为1：8。T20P寄存器初始值的计算公式应为：

2ms/0.0625us = T20 × 8（预分频比），计算得到T20P =4000（0x0FA0）。

实现步骤：

a) 清零 RAM

b) 初始化 PC1 端口为数字输出 I/O

c) 初始化定时器，设置预分频比，寄存器初值

d) 使能定时器及全局中断，进入中断服务程序，清中断标志位，翻转 I/O

2.9.2 T20 双精度PWM

功能说明：

使用芯片的双精度 PWM 模式，在 PC0 端口及 PC1 端口实现周期为 100us~50us，占空比分别为 25%及 50%的方波输出（其中周期分四次递减至 50us，再将周期重新装载，占空比不变，以此反复）。

芯片使用 16MHz 系统时钟，则对应的 T20 定时器时钟源周期为 1/16MHz。T20 的预分频采用：1:1，T20 周期寄存器 T20P 初始值计算公式应为：

PWM 周期 100us = T20P × （1/16MHz）× 1（预分频比），计算得 T20P 初值 PERIOD = 1600（0x0640）。

PWM 脉宽 50us = T20PR × （1/16MHz）× 1（预分频比），计算得 T20R 初值 DUTY21 = 800（0x0320），这里以 PC1 端口的输出方波为例。

PWM 占空比 50% = T20R / T20P

实现步骤：

a) 清零 RAM

b) 声明周期、精度变量

c) 初始化 PC0、PC1 为 PWM 输出口

d) 设置双精度 PWM 模式、使能并设置预分频比

e) 设置周期与精度初值

f) 进入中断更新 PWM 周期

2.9.3 T20 比较器

功能说明：

使用芯片的 T20 比较器模式，设定 T20CP1 数值，当计数器递增计数值与 T20CP1 寄存器中的比较值相同时，PWM201 端口输出 0 或 1（本例程设置为 1）并保持持续输出为 1。

实现步骤：

a) 清零 RAM

b) 初始化 PC1 端口为输出口、高电平

c) 初始化 T20 为比较器模式

d) 延时、保持 PC1 端口电平状态

e) 赋值比较寄存器，使能定时计数同时翻转 PC1 端口电平

f) 递增计数，匹配比较寄存器给定值，执行匹配事件

2. 10 T21 捕捉器

功能说明：

采用 T21 捕捉信号脉冲的上升沿，设定每 4 个上升沿捕捉 1 次，数组连续捕捉 100 次从头开始。主频 16MHz，设定捕捉时钟分频 1：16，则时钟周期为 1us。

注：在使用硬件仿真调试时不能使用 T20 捕捉器模块，因为其对应输入捕捉口同时复用为仿真通讯口，会造成持续的误触发中断。

以外挂 1kHz 脉冲源为例，数组计数值为：

1/1kHz × 4（匹配上升沿次数）/（1/16MHz × 16）= 4000（0x0FA0）

实现步骤：

a) 清零 RAM

b) 初始化 PC0 端口

c) 设定捕捉模式，使能定时器及多功能中断

d) 进入中断，读取捕捉寄存器计数值，循环 100 次放入数组

e) 数组清零，中断标志位清零

2. 11 大电流输出驱动

功能说明：

PB0~PB7 端口支持 100mA 灌电流驱动能力。大电流模式下，同时只能有一个端口可提供高达 100mA 的驱动能力。如果用 LED 驱动，需采用共阴极动态扫描的驱动方式。

注：本例程延时函数的延时时间存在误差，因为存在循环操作时的指令周期延时，而不仅仅是 NOP 指令的机器周期。

实现步骤：

a) 清零 RAM

b) MCU 初始化，PB4 为数字输出 I/O，设置 PB4 为大电流口

c) 设置延时函数每 100ms 大电流端口电平状态翻转一次

2. 12 系统时钟切换

功能说明：

将系统时钟切换至 32kHz 外部时钟（本例程被注释的部分为外部时钟切换程序，若用户不方便外挂单独的晶振可直接采用切换内部时钟例程，内部时钟精度误差较外部时钟大）。采用中断的方式，在 PA1 端口实现周期为 30ms，占空比为 50%的方波输出。

15ms/（0.03125×2）ms =（255-T10+1），计算得到 T10 = 16(0x10)。

实现步骤：

a) 清零 RAM

b) 切换外部时钟，并等待时钟切换完成

c) 初始化 PA1 为数字输出 I/O

d) 初始化 T10 定时器，设置预分频比，赋计数初值

e) 进入中断，反转 I/O

2. 13 PINT按键唤醒

功能说明：

本例程实现外部按键中断唤醒，唤醒 400us，利用 T11 的双精度 PWM 模式，实现在 PB1 端口输出周期为 100us 占空比 50% 的方波，之后再次继续进入睡眠，等待下一个中断来临，主函数里调用的延时函数主要是保证在延时时间里正常输出 PWM 波，否则程序会直接进入低功耗模式。

注：延时函数的延时时间存在误差，因为存在循环操作时的指令周期延时，而不仅仅是 NOP 指令的机器周期。

实现步骤：

a) 清零 RAM

b) 初始化 I/O

c) 选择 T11 双精度 PWM 模式，并使能外部中断 PINT0

d) 设置 PWM 周期与精度，调用延时子函数

e) 进入睡眠模式

f) 触发中断，清中断标志位，设置唤醒标志位，进入唤醒后的 PWM 输出

2. 14 LVD中断

功能说明：

本例程实现 LVD 中断模式，通过 PA0 端口的电平状态翻转观察，将电源电压 VDD 同预先设定的阈值电压进行比较，当目标电压低于预置电压阈值时，LVDO 位被置高，表明检测到低电压产生。

当 LVDO 变化时，产生 LVD 中断标志，而中断由 LVDO 位的边沿触发，本例程采用的下降沿触发方式 。

实现步骤：

a) 清零 RAM

b) 设置 LVD 模式

c) 进入中断，清中断标志位，翻转 I/O

2. 15 VGEN LCD驱动

功能说明：

本例程利用 I/O 直接驱动段式 LCD。用于驱动 5V，1/4 复用，1/2 偏置的段式 LCD，由于 COM 需要输出 1/2 VDD，而 PB 端口弱上拉电阻、弱下拉电阻阻值不同，不推荐作为 COM 口。

实现步骤：

a) 清零 RAM

b) 初始化 COM 口，端口电压为 1/2 VDD，并初始化 seg 口

c) 调用延时函数

d) 通过 COM、seg 口电平变换，刷新 LCD 显示

2. 16 WDT溢出唤醒IDLE

功能说明：

增加 WDT 溢出唤醒 IDLE 例程，需要在配置字中使能 WDT。

实现步骤：

a) 清零 RAM，系统时钟及 IO 初始化

b) 初始化 WDT 模块，约 4s 溢出

c) WDT 溢出翻转 PA6 端口电平

d) 进入 IDLE，进入前无须关闭外设时钟，等待 WDT 溢出唤醒