钜泉 单相多功能电能计量 SOC 芯片 FAQ
——ATT7035AU/37AU/37BU/39AU
V1.6

钜泉光电科技（上海）股份有限公司

Tel: 021-51035886
Fax: 021-50277833
Email: sales@hitrendtech.com
Web: http://www.hitrendtech.com

版本更新说明

版本号	修改时间	修改内容
V0.1	2010-06-23	初始版本
V0.2	2010-11-19	1. 修改 2，3，4，5，10，11，12，13，15，16，21； 2. 删除原 17； 3. 增加 21~27；
V0.3	2010-12-6	1. 重新调整顺序，修改并增加 2，18，19，24，26；
V0.4	2011-12-30	1. 增加功率和有效值折算公式，中断使能和中断优先级寄存器操作说明；
V0.5	2011-4-22	1. 增加 VBAT 应用说明，PWM 应用说明；
V0.6	2011-5-18	1. 增加 33~37；
V0.7	2011-9-27	1. 第 12 项增加 B 版 CKCON 设置推荐； 2. 增加 38，同步时序软件规避；
V0.8	2012-1-5	针对 G 版的修改： 1. 删除失效规避； 2. OSC 外部不需要 10Mohm 电阻； 3. 措辞修改；
V0.81	2012-2-29	1. 增加 RTCCAL 的写入建议； 2. 分 4 章节，重新调整序号；
V0.82	2012-3-19	1. 修正 Data Flash 操作（2.5）； 2. 增加 RTC 定时器中断操作说明（2.13，原 2.13 内容合并入 2.7）； 3. 增加 PSRR 说明（3.10）；
V0.83	2012-4-19	1. 增加能量寄存器读取说明； 2. 增加 64K PM 空间使用说明；
V1.0	2012-5-10	1. 增加 UART0 和 UART1 的应用说明； 2. 修正笔误，增加 3.3 章节的转换系数说明； 3. RTC 定时器中断操作修改； 4. 增加按键的应用说明；
V1.1	2012-9-3	1. 增加中断使能和中断标志位操作； 2. 增加有效值寄存器的读取说明； 3. 增加如何通过 0.5mT 潜动验证；
V1.2	2013-4-27	1. 增加全失压计量作说明； 2. 增加 IO 操作注意事项； 3. 增加 SPI 操作注意事项； 4. 增加反向功率指示标识； 5. 增加适用于 ATT7037BU；
V1.3	2014-3-26	1. 增加芯片的焊接要求； 2. 增加关于加密位的说明； 3. 增加关于 KBI 模块的说明； 4. 增加仿真时的注意事项； 5. 增加 LCDCLK 在 Sleep 模式下的配置说明； 6. 增加 OSC_SLP 的配置说明； 7. 修改 IO 口操作注意事项。

V1.4	2014-4-23	1. 修改 LCDCLK 在 Sleep 模式下的配置说明。
V1.5	2014-10-8	1. 增加 GPIO 位操作 bug； 2. 增加 RTC 年月日时分秒改写问题； 3. 增加 VBAT 管脚接二极管的说明； 4. 增加上电判断的应用注意事项； 5. 增加 VBAT 应用的注意事项； 6. 更改 PSRR 为 ADC_Vregulator_PSRR_EN； 7. 修改 UART1 的应用说明； 8. 修改 KEY0 和 KEY1 的应用说明； 9. 修改当 WDT 为 4s 清狗仍无法启动时的规避方案； 10. 修改 RTC 补偿的描述； 11. 调整顺序并修正文字描述。
V1.6	2016-05-18	1. 增加 2.23 对写保护寄存器写有效的可靠性建议

注：本文档适用于 ATT7035AU, ATT7037AU, ATT7037BU, ATT7039AU.

目录

1 硬件

1.1 外部32k晶振不需10Mohm偏置电阻

A：如图 1， 系统复位后，片上低频振荡电路需外接 32KHz 晶振（已经内置 10Mohm 偏置电阻）才可正常工作，经内部 PLL 产生系统高频时钟。负载电容（15pF）的容值需根据晶振厂家的推荐。

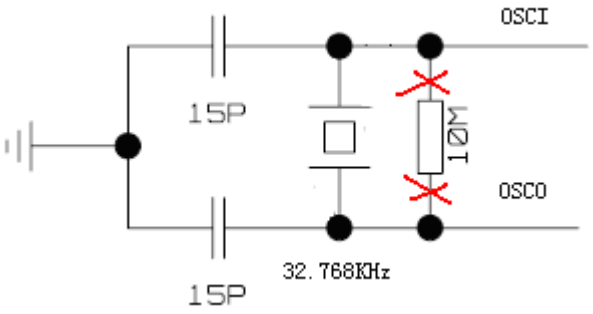

图 1

1.2 VDD1P8的用法。

A：VDD1P8 输出的是内核工作电压，推荐加 0.1uF 和 1uF 电容进行滤波，也可仅加 0.1uF。

1.3 7035A 电源输出驱动能力说明

A：7035A 有三个 VDD3P3 电源输出，分别为 pin92，51，26。

- VSYS 和最近的 VDD3P3(PIN92)之间的导通阻抗为 8 欧姆，PIN51，PIN26 相对于 PIN92 的阻抗为 5 欧姆。三个 VDD3P3 （PIN51，PIN26 和 PIN91）在芯片内部是连接在一起的，PIN51 和 PIN26 的驱动能力小于 3mA，对于接外部负载的应用，建议 PIN92 和 PIN51 在 PCB 上连接在一起，相关负载接在该电源线上。
- VDD3P3 的总的驱动能力受限于电流流过 VSYS 和 PIN92(VDD3P3)之间的电阻导致的压降。VDD3P3（PIN92）=VSYS-I×8.(I 是通过 VSYS 引脚流入 VDD3P3 的电）

例如：

VSYS=3.3V，VDD3P3 的驱动电流 I=12mA，

VDD3P3(PIN92)=3.3-0.012*8=3.2V

1.4 电源PIN的推荐用法？

A：

- VDD3P3 是内部电源输出,为了保持电压输出的稳定，减少纹波，推荐外加滤波电容 0.1uF+10uF。
- 对抗 ESD 影响，芯片的 3 个 VDD3P3 最好能分别外接滤波电容 0.1uF+10uF，或者接在一起，在此电源布线较长的情况下，沿途增加电容进行滤波。

1.5 VSYS与VBAT切换的时间？

A：电源是无缝切换的，完全切换过去的时间小于 1ms。(测试条件是在 VBAT=3.6V，VSYS=2.8V，VDD3P3 上接 0.1uF+10uF 滤波电容的情况下)，切换时间与电压切换档位、VBAT 的切换电压、电容都有关系，不

会超过 2ms。

1.6 VBAT新电池略高于3.6v，直接接入VBAT，是否可行？

A：VBAT 最大可以接到 3.8V。正常的标称 3.6V 电池直接接入 VBAT 都不会有问题。

1.7 VBAT的应用说明

A：在不连接电池的应用中，VBAT 应连接到 VSYS，不推荐 Floating，不能直接接地（或只接法拉电容）。

若因电池电压过低导致无法唤醒，需要检查软、硬件设计，降低唤醒时系统功耗（如开启的功能模块，IIC 操作、IO 设置等）；修改 BOR 为较低的阈值。

1.8 推荐的采样输入信号。（电流信号幅度根据实际情况定）

A：

- 电压通道推荐输入为 200mV 有效值。电压信号太大，会导致串扰，以及量程溢出；太小会影响内部精度。可选择的范围 143mV～350mV，此推荐的电压范围可以保证 70%Un 以及 60%Un 输入时的精度符合国标；
- 建议采用第一电流通道作为主计量通道。

1.9 JTAG复用的串口（UART1）的应用说明

A：UART1 的 TX1，RX1 引脚与 JTAG 的 TDO，TMS 引脚复用。如果使用 UART1 来实现通讯功能，有可能会影响到正常的下载和仿真。

因此推荐用其他串口来实现通讯功能，如不可避免使用 UART1 来实现通讯功能，可以在硬件上做节点，当进行下载和仿真的时候将复用的电路断开。

1.10 按键的应用说明

A：KEY0、KEY1 引脚与 JTAG 的 TCK，TDI 引脚复用。KEY0 或 KEY1 引脚上如果外接 0.1uF 滤波电容有可能会影响 JTAG 口的下载和仿真功能。

优先推荐用 KEY2、KEY3、外部中断 0 或外部中断 1 来实现按键功能。若不可避免使用 KEY0 和 KEY1，则推荐通过使用小于 4700pF 的电容来进行规避。

1.11 芯片的焊接要求

A：芯片开封后需保存在干燥的环境中并保证在 168h 内上板；

若芯片开封后 168h 内未上板，需要在焊接前在 125℃条件下烘干 7~8 小时做防潮处理，然后才可以焊接；芯片能够承受的峰值焊接温度为 260℃。

1.12 关于VBAT管脚是否需要接二极管？

A：当 VBAT 管脚外接电池并且电池电压较低（低于 2.0V）时，若系统反复上下电的环境中，当掉电时系统切换到 VBAT 供电，VDD3P3 此时的电压高于电池电压，会反向往电池灌电，加速电池老化，减小其寿命。

若想规避此现象，可以在 VBAT 管脚接个二极管后再接降 0.3V 的锗管。但由于 VBAT 外围的负载会影响到锗管的压降，因此此时电池电压检测的结果会受影响。

2 软件

2.1 WDT设为4s清狗，但程序无法启动

A：原因是程序的变量初始化时间超过了 4S，而且这部分程序是 keil 编译器自己生成的，用户也没法在其中增加清狗指令。

建议用户在写程序时，注意以下几点：

- 缩短初始化时间，减少 ram 的占用；
- 在声明 const 变量的时候前面加 code，将这些常数放在 code 区；
- 定义变量的时候，不要带初始化，比如 unsigned char temp = 1；改为 unsigned char temp；给其赋值为 1 的动作放在用户程序中。也就是说需要赋初值的变量，不要在声明的时候赋值，放在用户程序中主动赋值；
- 也可以选择不修改程序，直接在 STARTUP.A51 中打开高频。

2.2 如何提高程序运行速度？

A：ATT703X 是单 Cycle 的 CPU，程序和数据存储器均支持零延迟读写操作，读写速度可以通过读写延迟控制寄存器 CKCON（0x8E）加以控制。

CKCON[6：4] = 111 ； Flash 空间对于一个字节读写操作的取址时间为（7+1）个振荡周期；
CKCON[6：4] = 000 ； Flash 空间对于一个字节读写操作的取址时间为（0+1）个振荡周期；
CKCON[2：0] = 001 ； XRAM 空间对于一个字节读写操作的取址时间为（1+1）个振荡周期；
CKCON[2：0] = 000 ； XRAM 空间对于一个字节读写操作的取址时间为（0+1）个振荡周期；

推荐设置 CKCON=0x11（综合速度和 EMC 等特性考虑）。

2.3 内部RAM高128bytes和SFR地址重合，操作上如何区分？

A：内部 RAM 高 128bytes 只能采用寄存器间接寻址方式，如：

```
MOV   A   @R0;
```

而 SFR 是直接寻址的，如：

```
MOV   A，  070H;
```

2.4 如何使用双指针进行数据块搬移？

A：利用 DPS 的 SEL (bit0)位选择当前的数据指针，就可实现快速数据块搬移。

```
DPS_SEL = 0    ;数据指针使用 DPTR0（default）;
DPS_SEL = 1    ;数据指针使用 DPTR1;
```

例：

将 0100H 地址开始的 10 个数据搬到 0300H 地址，程序如下：

```
        MOV    DPTR, #0100H     ;DPTR0  地址为 0100H
        MOV    R0,   #0AH       ;数据块长度 10 放入 R0
        INC    DPS              ;切换指针为 DPTR1
        MOV    DPTR, #0300H     ;DPTR1  地址为 0300H
LOOP:
        INC    DPS              ;切换指针为 DPTR0
        MOVX   A, @DPTR         ;从 0100H 取数据
```

```
        INC     DPS             ;切换指针为 DPTR1
        MOVX    @DPTR, A;       ;放入 0300H
        DJNZ    R0, LOOP        ;循环搬移
        INC     DPS             ;切换指针为 DPTR0
```

注：如果采用较老的钜泉仿真器，若芯片内已有程序设置了 DPS=1，再此下载程序时会出现下载不了的情况，原因是仿真器和 703x 的 DPTR 不同步，这种情况在使用新版本的 70XX 仿真器将不会出现此问题。

2.5 Data Flash如何操作？

A： ATT703X 包含 60K Code Flash、4K Flash（DM/PM）和 256Info Flash 三部分。

- 4K Flash 默认为 Data Memory ，若需要将其配置为 PM 空间，操作如下

  配置 PCON 的 DP_SEL 选择 60K~64K Flash 为 PM 空间：

```
BWPR = 0xCF;
BWPR = 0xDC;
PCON |= 0x40;// DP_SEL(bit6) =1
BWPR = 0;
```

  修改编译器的分配地址确保 PCON|=0x40 之前的程序都运行在前 60K：

  OPTION for target->BL51 Locate->code :？ C_C51STARTUP(address), address 要定义在前 60K 地址，注意不要跟中断向量表的地址重合。

  若不作此修改，STARTUP 文件可能被 KEIL 编译器分配到 60K -64K 之间，程序中 PCON 配置的 DP_SEL 起不到作用。

- 4K DM Flash，可以代替片外的 EEPROM 用于存储掉电时需要保存的数据，地址范围为 1000H-1FFFH，包含 4 页，每页 1KB。推荐这部分空间采用第二种映射模式：

  ✧ **第二种映射模式按页擦除 Data Flash**

  例如：擦除 DM Flash Page2，程序如下：

```
        ANL   PCON,   #0EFH       ; PMW=0
        ORL   MCON,   #03H        ; PMLOCK=1,RSLOCK=1
        MOV   A,      MCON
        ANL   A,      #0F3H       ; FOP0=0
        ORL   A,      #08H        ; FOP1=1
        MOV   MCON, A             ; Flash Page Erase
        ORL   MCON, #40H          ; select the second mode
        MOV   A,    #0FFH         ; erase with FFH
        MOV   DPTR, #01400H       ; DM address, page2
        MOVX  @DPTR, A
XXX:    MOV   A, MCON
        JB    ACC.4, XXX          ; wait BUSY=0
        RET
```

例程：

```
void FlashDMPageErase(u16 Pageaddress)     //页地址就是这一页任何一个地址
{
 PCON&=0xEF;          //PMW=0
MCON|=0x03;       //PMLOCK=1,RSLOCK=1
MCON&=0xF3;
MCON|=0x08;       //FOP[1:0]=10            页擦数
MCON|=0x40;       //MAPMOD=1;
```

```
    XBYTE[Pageaddress]=0xFF;

    while(MCON&0x10);
    }
```

✧ **第二种映射模式写 Data Flash:**

例如：将 11H 写入 DM Flash 0x1400，程序如下：

```
    ANL   PCON,   #0EFH       ; PMW=0
    ORL   MCON,   #03H        ; PMLOCK=1,RSLOCK=1
    MOV   A,      MCON
    ORL   A,      #04H        ; FOP0=1
    ANL   A,      #0FBH       ; FOP1=0
    MOV   MCON, A             ; Flash Page write enable
    ORL   MCON, #40H          ; select the second mode
    MOV   A, #011H            ; write data
    MOV   DPTR, #01400H       ; DM address
    MOVX  @DPTR, A
XXX: MOV  A, MCON
    JB   ACC.4, XXX           ; wait BUSY=0
    RET
```

注：应用时，需要先将整页数据拷贝到 RAM，再擦除该页，在 RAM 中修改数据后，再将整页数据写回 Flash。

例程：

```
    void FlashWriteDM(u16 address, u8 num ,u16 size)
    {
        u16 i;
     PCON&=0xEF;          //PMW=0
    MCON|=0x03;      //PMLOCK=1,RSLOCK=1
    MCON&=0xF3;
    MCON|=0x04;      //FOP[1:0]=01        写操作
    MCON|=0x40;      //MAPMOD=1;
    for(i=0;i<size;i++)
    {
            XBYTE[address+i]=num;   //写 flash
            while(MCON&0x10);
            MCON&=0xF3;
           MCON|=0x04;      //FOP[1:0]=01        写操作
    }
    }
```

✧ **第二种映射模式读 Data Flash：**

例如：读 DM Flash 0x1400 的一个字节，程序如下

```
ANL   PCON,   #0EFH       ; PMW=0
ORL   MCON,   #40H        ; select the second mode
MOV   DPTR,   #01400H     ; read DM address
```

```
    MOVX    A,      @DPTR          ; read to ACC
```

- 这 4K DM 也可用于保存表格数据，采用新的仿真器可以直接对这部分 DM 进行烧写。

例程：

```
void Flash_Read_Mode2(unsigned int f_Addr, void *x, unsigned char len)
{
unsigned char *p;
unsigned char i;

    PCON&=0xEF;
    MCON =0x40;
    p=(unsigned char *)x;
    for(i=0;i<len;i++)
    {
        *p=XBYTE[f_Addr];
        p++;
        f_Addr++;
    }
}
```

2.6 如何配置系统电源管理？

A：系统默认：使能电池电压/温度测量、VDD 低于 2.4V 时产生 BOR 复位。

- VDCR(94H)寄存器默认值为 0x79；
- Normal 模式下：
  - 设置 VDCR[3:2]选择 VSYS 的电压阈值，当 VSYS Pin 低于此阈值，自动切换到 VBAT 电池检测端；
  - 设置 VDCR[1:0]选择 BOR 检测电压阈值，当 VDD3P3 低于此阈值，产生 BOR 复位或 BOR 中断；
  - BOR 的复位使能：BORRST=1(default)；
  - BOR 的中断使能：BORRST=0，BORIE=1；

- Sleep 模式下，定时开启 BOR 可以降低系统功耗，保证唤醒时的正常复位：
  - 设置分时开启 BOR 功能：在 BOR_EN=1(default)时，设置 BOR_DIV=1，分时开启的时间和间隔时间在 TCR 中设定；

- 在 PMSR 中可以查看 DVDD、VSYS 和 VDCIN 的状态。
- 当系统上电没有稳定时不要改写 CLKCFG。（系统电源是否正常以查看 PMSR 寄存器的 VSYSS 标志是否等于 1 为准）

2.7 如何进入SLEEP模式并实现唤醒

A：

当系统检测到掉电标志或者上电时检测到电池供电，需要进入 Sleep mode。

配置如下：

- 设置 sleep mode 下可被唤醒的中断源(WAKE_ENABLE)；
- 若配置 KEY 唤醒：按键 PIN 配置为 KEY 功能，使能相应 KEY 中断，配置 KEY 唤醒使能；

- 配置 sleep 模式下关闭 WDT。在 sleep mode 下，WDT 默认依然有效，当 WDT 计数溢出时，系统发生 WDT 复位，在进入 sleep mode 之前写入 WDT_SH=0，进入 sleep mode 后 WDT 无效。
- 打开写保护，修改 CLKCFG 的 SYSCK=0，切换到低频；
- 打开写保护，修改 PLLCFG 的 PRION=0, 关闭 PLL；这时可以加入需低频下处理的程序段；
- 切换到低频，最好经过至少 3 条指令后，再配置进入 sleep mode，这期间，可以配置 sleep 下的功能模块使能和 IO 状态，不需要的功能模块可关闭，IO 配置需配合外围，防止漏电。
- 打开写保护，配置 CLKCFG 的 mode[1：0]=11，进入 sleep mode；
- 在 VDD=VSYS，不允许进入 sleep mode，否则会出现 soft reset；只有在电池供电的情况下，才可进入 sleep mode。

唤醒：

- 在进入 sleep mode 后经过至少 4 个低频时钟，可以通过唤醒中断源产生唤醒复位。
- 对应外部中断 Pin 出现大于 8 个 fosc 的相同电平(根据设置的有效电平)，响应唤醒复位，程序从复位地址 0000H 开始执行。

例程：

```
void main()
{
    。。。。。。
    if(Flag.Bits.bPowerDown){
     /*iic saving */
     FreeDog();
     PowerDownConfig();//配置 sleep 模式下的功能模块开启或关闭，以及 GPIO 状态
     FreeDog();
     tmp=PMSR;
     if(tmp&0x02){//vdd=vbat
           Mode=SLEEP;
         Turn_To_ Sleep();
    }
}

void PowerDownConfig(void)
{
//turn to fosc and close PLL ------------------------
      BWPR = 0xcf;
      BWPR = 0xbc;
      CLKCFG&=0xFB;// config SYSCK=0, turn to fosc
      _nop_();
      PLLCFG&=0x7F;//PRION=0,close PLL
      _nop_();
//config function module-----------------------------
      BWPR = 0xcf;
      BWPR = 0xbc;
      SUPDC=0xA0;// open KBI,LCD
      _nop_();
```

```
        BWPR=0x0;
//IO config--------------------------------
        BWPR = 0xcf;
        BWPR = 0xdc;
        P02CFG|=0xF0;// disable Uart0/1, TOUT, PF/QF/SF
            P3CFG = 0x0;//disable PWM,T2/1/0, INT0/1,i2c
            PECFG = 0x20;//disable CLKOUT, config TMUX as SF output low level

        DDRP2 |= 0xF0;
            DDRP3 = 0xFF;// output
            DDRE|=0x1F;

        P2 &= 0x0F; //output low
            PTE&=0xE0; //output low

        RTC_WakeUp_Initial(); //若需要 RTC 唤醒，需配置
}

void Turn_To _Sleep(void)
    {
    //select wake up interrupt source------
            WAKE_EN=0;
                FreeDog();
            WAKE_EN|=WKS_KEY;
            WAKE_EN|=WKS_RTC;
    //      WAKE_EN|=WKS_RX1;
    //      WAKE_EN|=WKS_RX0;
    //      WAKE_EN|=WKS_RX2;
    //      WAKE_EN|=WKS_INT1;
             WAKE_EN|=WKS_INT0;
             WAKE_EN|=WKS_PMU_TBS;
    //      WAKE_EN|=WKS_RX2;
    //turn to sleep mode-----------------
           BWPR = 0xcf;
           BWPR = 0xbc;
           CLKCFG |= 0x83;    //MOD[1..0]=[11],     Sleep   Mode
           BWPR = 0x00;
    }
```

2.8 如何利用LVDIN PIN检测电源电压？

A：有两种方式，LVDIN 中断或定时查询 LVDIN 的状态。采用外部分压方式将电源电压分到 2V 附近接入 LVDIN PIN。

- LVDIN 中断方式：
  - 使能 LVDIN_DET 模块，系统默认 LVDIN_EN=1(PMUCFR.2, 0x93H)；
  - 开启 LVDIN 中断使能，即 LVDINIE=1(PMUIER.0, 0x97H);

- 开启 PMU 中断使能，即 EX3=1(IEN1.1, 0xB8);
- 清中断标志，即 LVDINIF=0(PMIFR.0, 0x96H)；
- 当 LVDIN 输入电压下降到低于 1.18V 或上升到高于 1.18V 时，LVDINIF=1，进入 LVDIN 中断服务程序；
  这里存在 75mv 左右的迟滞电压 ΔV，即当电压下降到(1.18-ΔV)时，或当电压上升到(1.18+ΔV)时，才能触发 LVDIN 中断。

➢ LVDIN 状态查询方式：

- 使能 LVDIN_DET 模块，系统默认 LVDIN_EN=1(PMUCFR.2, 0x93H)；
- 定时查询 PMU 状态寄存器 PMSR.0，当：
  LVDINS=1，表明 LVDIN 输入电压低于 1.18V；
  LVDINS=0，表明 LVDIN 输入电压高于 1.18V；

注：PMSR 不能被 Wake_up reset。

2.9 用LVDIN进行上电判定的注意事项

A：在上电过程中，LVDIN 不能作为判断上电的条件，只能作为预判使用。系统是否正常上电的判断应使用 VSYSS。

2.10 Uart1波特率计算公式

A：Uart1 的波特率不受 PCON 的 SMOD 位作用。

$$Uart1\_bps = \frac{\text{Fpri}}{32\times(1024-\text{S1REL})}$$

2.11 中断使能和中断优先级寄存器操作注意事项

A：对中断使能寄存器 IEN0、IEN1、IEN2，中断优先级寄存器 IP0、IP1 进行写操作，存在导致所有中断不使能的风险，必须再写 EAL=1（IEN0.7），打开所有中断使能。
应用规避：在操作 中断使能/优先级寄存器 后，写 EAL=1，开总中断。

2.12 初始化需强制高频锁定

A：
初始化时将强制锁定高频振荡，（TCR |= 0x20），例程：

```
void main(void)
{
    EA=0;
    WDTCON = WDT_000004S;
    TCR|=0x20;
    ……
}
```

2.13 RTC的补偿

A：

➢ 正常模式下的补偿方法：设置 TOUTEN[1..0] (F8H)=11 选择高频模式每 1s 补偿一次，低频模式每 20s 补偿一次。利用数字校准寄存器 RTCCAL 调整 32.768KHz 的频偏，在指定温度范围内使精度接近于

0ppm，RTCCAL 的最小解析度为 1.525ppm。

- 校准方法：配置 TOUT[2:0](F8H)=111，使其 TOUT PIN 输出未校准的频率值，测试整个温度范围内不同温度点下的频率值，计算出相应的频偏值 N，保存在非易失性存储器中。
- 打开 TBS 功能，测试环境温度，读取 TEM_DATA 计算当前的环境温度；

（注：TBS 每 1s 或 0.5s 测试一次，并根据 Diff_t[2:0]设定的温度阈值更新 TEM_DATA。）

- 根据当前的环境温度查表得到相应的频偏 N，写入 RTCCAL 寄存器；
- 配置 TOUT[2:0](F8H)=000，使其输出校正后的秒脉冲信号，可以用频率计观测补偿结果；

➢ 休眠状态下的补偿方法：

- 可以设置固定时间（如 2hours）唤醒一次，做时钟补偿，方法同上。
- 注意：在 sleep 下关闭 TBS，唤醒打开 TBS，得到一个正确温度数据的时间为 1mS(重新开启 TBS 后，开始读出的第一个温度数据是不准确的，要去掉)。

➢ RTCCAL 的写入建议：RTCCAL 写入建议避开补偿时刻 (即：9s，19s，29s，39s，49s，59s)，在补偿时刻写入 RTCCAL 存在时钟走时延后 1s 的可能。建议 RTCCAL 可以在秒中断中写入，或避免在 9s，19s，29s，39s，49s，59s 时刻写入。

2.14 RTC的定时器中断标志位操作注意事项

A：RTC 中断标志寄存器（0xB4）的 RTC 定时器 1 中断标志位 RTC1IF（Bit5）和 RTC 定时器 2 中断标志位 RTC2IF（Bit6）存在重复置位进中断的问题，必须在每次进中断后关闭对应定时器后再使能。例程：

```
void RTC_int (void) interrupt 12
{
    u8 temp;
    temp=RTCIF;
    RTCIF=0;
    if(temp&0x40)     //RTC 定时器 2
    {
        RTCCON&=0xBF;      //关闭 RTC 定时器 2
        RTCCON|=0x40;      //重新打开 RTC 定时器 2
       //user code ...
    }

   if(temp&0x20)     //RTC 定时器 1
    {
        RTCCON&=0xDF;      //关闭 RTC 定时器 1
        RTCCON|=0x20;      //重新打开 RTC 定时器 1
       //user code ...
    }
}
```

2.15 中断使能和中断标志位操作

A：中断标志位已置位（中断标志位的置位与中断使能无关，如 RTC 开启后，每分钟对应的分钟标志置位），后开启中断使能，会立即进入中断函数。需在中断使能前清零对应的中断标志位（包括子中断标志位）。

2.16 PTA操作注意事项（仅针对ATT7039A）

A：ATT7039AU（不适用于其他芯片）的 PA 口 PA.4~PA.7 如果配置为 IO 输出口时，不可用与或操作某单独 bit（否则会影响该口的其他 bit），因此建议如果程序中涉及到对 PA 口的操作，应建立一个中间变量并赋予寄存器 PTA 的初值，程序中所有操作均对中间变量进行，操作完成后再将变量的值赋给寄存器 PTA。

2.17 GPIO口的位操作问题

A：对 GPIO 的操作需要先配置方向寄存器为输出，然后配置数据寄存器。

错误应用举例：

```
unsigned char a;

void main(void)
{
    BWPR=0xCF;
    BWPR=0xDC;
    P02CFG = 0x00;              //将 P0 口配置为 GPIO 口
    DDRP0 = 0x00;               //P0 口方向寄存器配置为输入

    P0 = 0x00;              //P0 口数据寄存器配置为 0x00
    P0_7 = 1;           //将 P0_7 设置为 1

    DDRP0 = 0xFF;               //配置 P0 口的方向寄存器为全部输出

    while(1)
    {
            a = P0;         //设计思路上预期此时 P0 口的状态为 0x00，而实际上读出为 0xFF
    }
}
```

主要原因就在 P0_7=1 这个操作，这条指令看起来是只有 SET P0.7 这一个动作。实际在硬件执行的时候是执行了 3 个动作，分别为 读-修改-写，而且这 3 个动作是针对整个 P0 口的，而不是单独的一个 P0.7,而读的动作是从外部 GPIO 口上读取状态，而不是从 P0 的数据寄存器中读取状态，所以此时还在输入状态下的除了 P0.7 之外的其他所有引脚之前配置的 P0=0x00 这个赋值就无效了，被重新赋值了外部引脚的输入状态，从而导致 P0 口这时读取的数据是 0xFF。

2.18 SPI操作注意事项

A：SPI 相关的 SPCON 寄存器操作需在使能 SPI 功能后，即 SUPDC 中的 SPI 使能 bit 使能后，才可以对 SPCON 寄存器进行写操作，否则无效。

2.19 关于KEY模块的配置说明

A：

1. 建议在整个程序中 KBI 模块保持开启；
2. 如果一定要关闭再重新开启 KBI 模块，那么在 KBI 模块关闭期间不要对 KEY 端口进行操作。

2.20 LCDCLK在Sleep模式下的推荐配置

A：推荐在进入 Sleep 模式前将 LCDCLK 的值配置为默认值 0x90。

2.21 OSC_SLP的配置说明

A：OSC_SLP=1，大电流模式：如果在 Sleep 模式下将 OSC_SLP 配置成大电流模式，可以保证 RTC 补偿的时钟频率在 Normal 模式和 Sleep 模式下一致，但会增加 1μA 的功耗；

OSC_SLP=0，小电流模式：Sleep 模式下配置成此模式可节省功耗，但 OSC 振幅会降低，影响 Sleep 模式下 RTC 补偿的时钟频率。

推荐客户在 Sleep 模式下按照自己的需求配置此位。

2.22 RTC模块的年月日时分秒寄存器改写时注意事项

A：年月日时分秒寄存器修改时建议软件做如下处理：

Unsigned char temp；
temp=EA; //保存全局中断的值
EA=0； //关闭全局中断；
****； //修改年月日时分秒寄存器；
RTCIF=0；//清 RTC 标志
EA=temp; ////恢复全局中断的值

2.23 对写保护寄存器写有效的可靠性建议

当系统使用单 cycle 运行指令时，为确保具有写保护 register 能够被有效配置，建议：

1，打开写保护配置寄存器之后增加 NOP 指令等待写操作执行有效，防止时序偏差导致的写无效；

2，更改写保护类型之前关闭前一次写保护。

例如：

```
void LCD_Init(void)
{
    BWPR = 0xCF;
    BWPR = 0xBC;
    SUPDC |= 0x20;    //enable LCD module
    _nop_();
    BWPR = 0x00;

    LCDCLK = 0x90;    //select 1/4 bias 1/4 duty
    _nop_();
    LCDCR = 0xF0;
    _nop_();

    BWPR = 0xCF;
    BWPR = 0xDC;
    LCDCFG = 0xFF;
    _nop_();
    P02CFG|=0x0F;
    _nop_();
```

计量

2.24 如何使用第二路电流通道实现防窃电功能？

A：用户可以手动选择相应的通道进行计量，也可选择自动防窃电进行自动切换。

- **手动方式：**用户在程序中判断是否发生窃电。
  - 开启第二电流 ADC 通道：ADC_I2ON=1（5BH.5）；
  - 起动第二路数字高通滤波器：HPFONI2=1（52H.1）；
  - 选择用功率或者电流有效值方式判断：设置 TampSel（51H.7）；
  - 设置窃电判断的阈值（功率或有效值）：设置 ITAMP（59H）；
  - 设置窃电判断两个通道相差百分比的阈值：设置 ICHK（50H）；
  - 设置窃电发生时，两路电流的计量模式：设置 CIADD（51H.3），如果 CIADD=0，有效计量通道由 CHNSEL 决定；
  - 通过 I2GAIN 对第二通道的输出校正，保证同样的输入电流时，两个通道的有效值和功率输出一致；
  - 定时查询 TEMP（51H.6），当 TEMP=1 时表示发生窃电，这时可以根据 I2GTI1（51H.5）的状态确定哪一路的电流大；
  - 通过设置 CHNSEL（51H.4）切换当前的计量通道；

- **自动防窃电方式：**
  - 起动第二路数字高通滤波器：HPFONI2=1（52H.1）；
  - 选择用功率或者电流有效值方式判断：设置 TampSel（51H.7）；
  - 设置窃电判断的阈值（功率或有效值）：设置 ITAMP（59H）；
  - 设置窃电判断两个通道相差百分比的阈值：设置 ICHK（50H）；
  - 通过 I2GAIN 对第二通道的输出校正，保证同样的输入电流时，两个通道的有效值输出一致；
  - 开启自动防窃电功能：FLTON=1（52H.5）；
  - 定时察看窃电标志 TAMP（51H.6）：当 TAMP=1 表示发生窃电，并且这时当前有效计量通道已经自动发生切换，指示在 CHNSEL/I2GTI1（51H）；
  - 窃电恢复正常后的计量通道选择：如果 I1 接火线，I2 接零线，以有效值为例，采用功率判断的公式相同，用 Power1 替代 I1rms，Power2 替代 I2rms 即可：

    I2rms>I1rms*（1+ICHK）时，自动判断发生窃电，窃电标志 TAMP 置位，采用 I2 计量；

    I2rms<=I1rms*（1+ICHK）时，TAMP=0，仍采用 I2 计量；

    I1rms>I2rms*（1+ICHK）时，恢复 I1 计量。

注：这里采用一个滞回区域，是为了防止电流在 I1*（1+CHK）点抖动，导致计量通道频繁切换而不能正常计量。

2.25 P_offset和Q_offset校验公式

A：Femu=900KHz 时，公式如下：

- **P_offset 的公式如下：**

$$P\_offset = \frac{Preal \times EC \times HFConst \times 2\text{^}23 \times (-Err\%)}{1.0322 \times 10\text{^}11}$$

其中：

- Preal 是标准表上的功率值 ；
- EC 是表常数 ；

- HFConst 是高频脉冲常数。

P_offset 是一个带符号的数，如果按以上公式计算得到的值<0，将其补码写入 5CH，
即：（256- P_offset）取整后，写入 5CH。

Q_offset 公式同上。

➢ **采用功率法校验 P_offset：**
Err%的计算方法：

$$Err\% = \frac{Pdisp - Preal}{Preal} \times 100\%$$

其中：
- Pdisp 是液晶上面显示的功率值；

代入 Poffset 计算公式可得：

$$P\_offset = \frac{EC \times HFCONST \times 2\text{^}23 \times (Preal - Pdisp)}{1.0322 \times 10\text{^}11}$$

其中：
- 注意，Pdisp 是寄存器值乘以了 Kpqs 系数后 Ib 的显示值；
- 建议客户多读几次 Pdisp 做平均后再代入公式计算；

Q_offset 公式同上。

如果 Femu=450KHz，1.0322 的系数减半。

2.26 考虑到offset校验和使用第二路电流通道的校表流程

A：参看如下简单的过程，校表细节和公式请参考用户手册。

➢ 校表开始
➢ 如果初始化打开第二电流通道，校表还要执行 7~12，否则可以跳过；（注意 I1、I2 的切换）
1. HFConst 校正或初始化；
2. 第一通道增益 GP1、GQ1、GS1 校正；
3. 第一通道相位 GPhase1 校正；
4. 有效值转换系数 K_I1rms, K_Urms 校正（非内部寄存器，需 MCU 进行处理）；
5. 功率增益转换系数 Kpqs1 校正（非内部寄存器，需 MCU 进行处理）；

➢ 如果电压频率不是 50Hz，或者 femu 不为 900KHz, 这时的无功相位精度需要校验，执行 6，否则可以跳过;
6. 进行 QPhsCal 校正；

➢ 检查小信号误差，如果误差略大，执行 7，否则可以跳过；
7. 进行 P1-offset，Q1-offset 校正；

➢ 第二路电流通道校正，执行 8 到 12
8. 校验 I2GAIN；
9. 第二通道增益 GP2、GQ2、GS2 校正；
10. 第二通道相位 GPhase2 校正；
11. 第二路电流 K_I2rms 校正（非内部寄存器，需 MCU 进行处理；理论上 K_I2rms=K_I1rms）；
12. 第二路功率增益转换系数 Kpqs2 校正（非内部寄存器，需 MCU 进行处理）；

➢ 如果校验了 P1offset，执行 13，否则可以跳过；

13. 进行 P2-offset，Q2-offset，校正；
- 如果进行了第一电流通道的 QPhsCal 校正，执行 14，否则可以跳过。

14. 进行第二通道的 QPhsCal 校正；
- 结束

2.27 EMU模块的开启条件

A：EMU 模块的电源来自于 AVCC，推荐的应用是 VSYS 和 AVCC 引脚接系统电源，VBAT 引脚接电池，这时：
- EMU 模块在系统电源正常工作时，系统时钟切换到高频后才能被开启；
- 电源切换到电池供电时，EMU 模块将自动关闭，这时即使写入 SUPDC 使能 EMU 也无效；
- 当 AVCC 掉到 3V 以下时，计量不可靠，所以建议在判断到掉电后及时关闭 EMU 模块。

2.28 SUPDC控制EMU失效

A：
问题描述：VSYS 由市电切换到电池，再切换到市电的过程；或者电压下降到一定幅度又恢复时，并未发生复位，但 EMU 实际被关闭了。
解决：电池模式下，EMU 会自动关闭，为确保应用可靠，建议在模式切换或者电压波动恢复正常后，增加配置 SUPDC 开 EMU 使能的语句，保证 EMU 的正常工作。

2.29 如何实现全失压计量？

A：采用功率互感器安装在火线电流回路上，利用火线电流通过功率互感器产生的感应电源来给系统供电，使电表在零线接地窃电方式下仍然能够正常工作。同时利用视在功率 S 来计算功率和能量，在这种情况下，不再考虑功率因素，而采取计量负载做为纯阻型的情况下的功率值，此时可以认为纯阻型负载的有功功率是和视在功率一致的。

具体方法如下：
- **ATT7035AU/ATT7037AU/ATT7037BU 的三路 ADC 可以分别控制打开或关闭：**推荐用户只打开计量电流 ADC 通道，并使用校表时固定写入电压寄存器 UCONST（64H）的电压值来计算视在功率，这时功耗最小。只有在关闭电压 ADC 通道的情况下，UCONST 才有效，这时 Urms 寄存器始终为 0。
- 校准 UCONST：输入 220V，读取 Urms 寄存器的值，将其右移 7 位，其结果写入 UCONST 保存；
- 修改 Fadc 为较低的值(450kHz or 225kHz)，再修改 ADC 转换功率控制为低功耗模式，即 ADCI_CTRL=0，这时一路 ADC 消耗功耗为 5uA；
- 只开启一路当前电流通道的 ADC：ADC_IxON=1（5BH）；
- 计算当前视在功率，并替代当前有功功率：这时 P=S= Irms * （UCONST*2^7）。
- 配置 PSSel（52H.3）来使能 PF 输出视在脉冲。
- 输出功率受火线电流的大小影响。为了降低功耗，我们只开启一路电流 ADC，打开 EMU , I2C ,LCD Module，关闭其他模块。

  程序代码如下：

```
//正常情况下，校验保存 UCONST
EPADR = ECR_WR_EN;                          //ECADR Written protect open
ECADR =   0x64;  //UCONST
ECDATH =0x04; //正常电压读出的 URMS 值，右移 7 位，即除以 2^7，得到 Uconst 值
ECDATL =0x88; //  写入 0x64
```

```
//初始化更改
EPADR = ECR_WR_EN;                          //ECADR Written protect open
ECADR =   0x5B;
ECDATH =0x00; //关闭电压通道，电流通道 2，仅开启电流通道 1，ADC 时钟设置为 225K
ECDATL =0x43;      // ADCi_Ctrl  设置为“0”，为低功耗模式

BWPR = 0xcf;
BWPR = 0xbc;
PLLCFG = 0x80;//  配置 CPU 时钟为 0.68M，降低系统功耗
BWPR = 0xcf;

BWPR = 0xcf;
BWPR = 0xbc;
SUPDC = 0x38;// open EMU, I2C, LCD Module，关闭其他模块，降低系统功耗。
BWPR = 0x0;
```

注：

1. 只可先修改 Fadc<=450K，才可修改 ADCI_CTRL=0，其它修改方式会导致错误！
2. 为了使计量结果的非线性误差更好，需要在正常情况下校准 Irms_offset，窃电时 Irms_offset 不会变。
3. 感应线圈随着线圈通过的电流的增加，感应电压会随着增大，设计时需考虑电源的安全性。

2.30 功率及有效值(RMS) 折计算公式

A：功率、有效值的显示可以直接乘上相应的系数 Kx 即可。为了检验 Kx 是否正确，可参照下面的近似折算公式。

- **有功功率折算公式：**

$$Px = \frac{Preg * 10332 * 10\text{^}7}{2\text{^}23 * EC * HFConst}$$

Px —— 为计算出的实际的功率值，单位：W
Preg —— 为从功率寄存器 PowerP 读取的功率值
EC —— 为电表的脉冲常数
HFconst —— 为 ATT703x 的 HFConst 寄存器

- **无功功率折算公式：**

$$Px = \frac{Preg * 10332 * 10\text{^}7}{2\text{^}23 * EC * HFConst}$$

Px —— 为计算出的实际的功率值，单位：Var
Preg —— 为从功率寄存器 PowerQ 读取的功率值
EC —— 为电表的脉冲常数
HFconst —— 为 ATT703x 的 HFConst 寄存器

- **IRMS 折算公式：**

$$Ix = \frac{It}{Ris * Gain * 1.064 * 2\text{^}23}$$

Ix —— 为计算出的实际的电流值，单位：A
It —— 从 IRMS 读取的电流有效值数据
Ris —— 电流取样电阻阻值
Gain —— 电流通道增益
注：如果使用电流互感器取样，还需要乘以互感器的变比

- **URMS 折算公式：**

$$Ux = \frac{Ut * Rt}{Rus * Gain * 1.064 * 2\text{^}23}$$

Ux —— 为计算出的实际的电压值，单位：V
Ut —— 从 URMS 读取的电压有效值数据
Rt —— 电压通道电阻串的总电阻值
Rus —— 为电压取样的分压电阻阻值
Gain —— 电压通道增益

注：以上公式计算出的功率值是比较精确的，但有效值只能作为粗调，还需要参考用户手册有效值输出寄存器的说明进行精确校验。

2.31 单相三线的应用

A：设置两路计量通道的电流相加模式为“绝对值和模式”（51H，ADCCFG），在输入反向的情况下，也可直接用于单相三线的计量功能。

2.32 关于EMC改善主要以下几点需要注意

- 退耦电容的摆放，注意形成有效退耦（电源引线先进入退耦电容再到芯片引脚），且尽可能贴近芯片
- 需要退耦电容的引脚主要有： AVCC/Vsys/Reset，以及前端 5V 转 3.3V 的 LDO。
- PCB 铺地，数字地和模拟地不分开，形成大面积铺地。
- 铺地尽可能形成成片的地，不要切割的太厉害，芯片下面上、下两层尽量铺成整块地。
- PCB 走线注意爬电距离，尤其电源等敏感线，不要单独暴露在铺地外面，容易形成天线效应。

2.33 ADC_VREGULATOR_PSRR_EN和ADC_Chopper说明

A：

1. VDCR(94H) 增加说明：建议关闭 ADC 的 Chopper 功能（CHOP_ADC_EN = 0，94H.4）；
2. EMCON.7，用于控制 ADC_VREGULATOR_PSRR_EN 的 Regulator 开关，该位默认为 1，打开 Regulator。

ADC_VREGULATOR_PSRR_EN 是用来改善不稳定的电源电压对计量精度的影响，打开此功能，可以保证在较恶劣的电源环境下，仍然具有稳定的计量精度输出。

注意：进入 Sleep 模式后建议关闭 ADC_VREGULATOR_PSRR_EN，唤醒后再开启（降低功耗）。

2.34 能量寄存器的读取说明

A：建议使用读后不清零能量寄存器来保存电能量。读后清零寄存器存在读后不清零的隐患。

2.35 有效值寄存器的读取说明

A：建议有效值寄存器（电流有效值、电压有效值寄存器）读取后与上 0x7FFFFF 再做处理（默认为 24 位无符号数，最高位 bit23 为无效位，但接近满量程时最高位 bit23 有置位风险）。

例如：

锰铜 175u，增益 16 （含模拟与数字），当电流为 180A 时，此时电流有效值为 175u*16*180=504mV （最大值 504*1.414=713mv<800mv，未溢出）。这种情况下，电流有效值寄存器正确值应为 0x408312 左右，但由于实际处理时第 24 位是第 23 位的扩展，所以第 24 位也会置 1，从而实际读到的值为 0xC08312。

规避方法：

读出有效值寄存器，将最高位清零后再做处理。如 Irms = （电流有效值寄存器）& 0x7FFFFF；Urms = （电压有效值寄存器）& 0x7FFFFF。

2.36 如何通过0.5mT潜动验证

A：由于 0.5mT 工频电磁场试验过程中有功功率的大小变化，导致有功功率在试验过程的部分时间会超过计量芯片的 启动/潜动 阈值，从而在超过阈值的时间里，芯片寄存器 PFCNT 会累计电能。当这个时间足够长，则会导致芯片发出一个脉冲。

0.5mT 工频电磁场对于电表的影响，主要是对其电流采样器件锰铜的影响，想要解决这个问题的源头是磁场对锰铜的影响，锰铜的摆放方式，锰铜线的绞线方式，锰铜线是否采用特殊的屏蔽线，这些都会大大降低工频电磁场对锰铜的影响。

如果用户已经通过硬件上降低了工频磁场对锰铜的影响，那么 0.5mT 工频电磁场试验对芯片而言不需要做任何额外的软件处理，否则用户需要在软件上增加更多的判断来区分是由于工频电磁场的试验导致了功率的大小变化还是由于用户的负载变化导致功率的大小变化。

芯片发脉冲之前，对于小于 1 个脉冲的能量，提供了 PFCNT 寄存器来指示，同时有功功率小于用户设定 启动/潜动 阈值的情况下，会给出芯片潜动标志位（小于阈值则置位），根据这两个功能和 0.5mT 工频电磁场试验下有功功率忽大忽小变化的特点，用户可以在潜动标志位置位的情况下，清零 PFCNT 寄存器，从而达到在规定的长时间工频电磁场试验情况下，也不会出脉冲的目的。

2.37 反向功率指示标识

A：由 EMUSR 中的 bit REVP 和 bit REVQ 用于指示有功和无功功率方向。当 EMCON 中配置能量累加为绝对值累加模式时，ATT7039AU 里的 bit REVP 和 bit REVQ（EMUSR 寄存器中）恒为 0，但 ATT7035AU、ATT7037AU、ATT7037BU 中的 bit REVP 和 bit REVQ（EMUSR 寄存器中）依然实时反映功率方向。不同芯片有所不同，应用中加以区分。

3 仿真

3.1 如何避免参数传递错误，保证系统可靠性？

A：

- **避免编译时，当前工作寄存器组发生切换：**

  CPU 由 PSW 中的 RS1 和 RS0 位决定当前用于参数传递的工作寄存器组（默认为 00，即 Bank0），在头文件中使用如下宏命令，可以防止当前工作寄存器组切换，从而避免出现参数传递错误：

  #pragma NOAREGS　//forbidden work registers' bank changed

  此条宏定义需要包含在所有头文件中，且保证系统运行时可以被立刻执行，避免在调用子程序发生寄存器组切换。

  使用该宏命令后，中断中的 using 指定的工作寄存器组语句将无效，都使用 PSW 中指定的工作寄存器组，以提高系统的可靠性。

- **保证 WDT 在上电时稳定：**

  为避免上电初始化时间过长引起 WDT 复位，在 STARTUP.A51 文件的 STARTUP1 标号后面添加一条清狗指令，将 WDT 溢出时间修改为 4s：

  MOV　0C9H, #0E5H;　　//修改 WDT 溢出时间为 4s

3.2 仿真器使用中的注意事项

A：

- **在进行与/JTAG 口相关的操作（比如下载、仿真）时，应确保/JTAG_WDTEN 与 GND 短接，否则会提示 Connect Error；**
- **下载完程序不能自动运行：**

  JTAG 模式下，WDT 拉高，程序不会运行；这时需要将 WDTEN 先拉高，再将 ATT703x 复位，程序可以运行；这是目前仿真器的问题，使用新的仿真器下载完程序后会自动复位，将不会有此问题。

3.3 单步调试时，受写保护的寄存器为何无法被改写？

A：在线单步调试时，打开写保护和改写受保护的寄存器操作不可单步执行。在改写受保护寄存器语句之后设置断点，全速运行到断点，寄存器就可被正确改写。这是因为单步时，操作写保护的两条语句之间会超过 30 条 CPU 指令时间，从而写使能自动关闭，造成修改无效。

3.4 下载成功，能正常运行，却无法仿真

A：

- **读保护被使能：**

ATT7035A/7037A/7037B 具有 Flash 加密功能，0x20FF（Info Flash）为非 0xFF 值时使能 Flash 加密功能。在线仿真需要实时读取 Flash 内容，所以不能在 Flash 加密使能的时候正常在线仿真。0x20FF 默认值为 0xFF，若需要对程序进行加密，可参考如下方式对其进行操作。

```
void FlashEncrypt()
{
    unsigned char tmp_ea = EA;
```

```
    EA = 0;

    BWCR=0xCF;
    BWCR=0xDC;

    PCON &= 0xEF;//PCON.PMW=0, Enable DM operation
    MCON &= 0xF7; //FOP[1:0]=01,PMLOCK=1,RSLOCK=1
    MCON |= 0x07;

    BWCR=0x00;

    while(MCON&0x10)
            ;

      XBYTE[0x20FF] = 0x00;   //Define the value of 0x20FF in info flash，enable the encryption of flash

    while(MCON&0x10)
            ;

    BWCR=0xCF;
    BWCR=0xDC;

    PCON &= 0xEF;//PCON.PMW=0, Enable DM operation
    MCON &= 0xF0; //FOP[1:0]=00,PMLOCK=0,RSLOCK=0

    BWCR=0x00;

    EA = tmp_ea;
}
```