MCU Application Note

[Rev 1.1]

目录

第一章 应用设计注意事项

1 中断注意事项

芯片的中断向量地址为 04H。

程序运行过程中发生中断事件时，CPU 会将程序当前 PC 值存入堆栈，然后从 04H 地址开始执行中断程序，一直执行到 RETI 指令，取回堆栈中的 PC 值返回主程序继续执行。

中断保护现场和恢复现场程序用于保护程序进入中断时的现场信息 ,如累加器 W 和状态寄存器 STATUS 的值。如果在中断里改写了 PCLATH，则需要增加一个缓存寄存器来保护 PCLATH。如果在中断里没有改写 PCLATH 值 ,则可以不用保护。保护现场和恢复现场程序可以保证在中断发生前后的现场信息一致，中断返回后可以继续正确的执行主程序。

推荐的保护现场程序和恢复现场程序如下：

```
W_TMP     EQU     0x70    ;保护 W 值的缓存寄存器（0x70-0x7F 为公用区间）
S_TMP     EQU     0x71    ;保护 STATUS 值的缓存寄存器
ORG       0004H           ;伪指令，定义中断程序入口地址
STR       W_TMP           ;进入中断程序，开始保护现场，将 W 值存入缓存
SWAPR     STATUS,W        ;将 STATUS 高四位和低四位互换后存入 W
STR       S_TMP           ;将 W 值存入 S_TMP 缓存
BANKSEL   PORTA           ;设置 BANK 为 PORTA 所在的 BANK0
;中断处理程序             ;省略的中断处理程序
;中断处理程序             ;省略的中断处理程序
INT_RET:                  ;中断返回标号，开始恢复现场
SWAPR     S_TMP,W         ;将 S_TMP 缓存高四位和低四位互换后存入 W
STR       STATUS          ;将 W 值存入 STATUS，恢复 STATUS 值
SWAPR     W_TMP,F         ;将 W_TMP 高四位和低四位互换
SWAPR     W_TMP,W         ;将 W_TMP 高四位和低四位互换后存入 W
RETI                      ;中断返回
```

W_TMP 和 S_TMP 需要定义在 0x70-0x7F 这个所有 BANK 共用的地址区间 ,以防在其他 BANK 时进入中断会丢失中断信息。

中断保护现场后需要设置 BANK。

如果中断处理程序里要处理 BANK0 里的寄存器，那么需要将 BANK 设置在 BANK0。

如果中断处理程序里要处理 BANK1 里的寄存器，那么需要将 BANK 设置在 BANK1。

如果进入中断后不设置 BANK，那么程序在 BANK1 进入中断后，直接操作到 BANK0 里的寄存器时就会产生错误。同理程序在 BANK0 进入中断时，操作到 BANK1 里的寄存器也会产生错误。

以上为使用汇编程序编程时需要注意事项，用 C 语言编程不需要注意。

C 编译器会自动设置中断保护现场和恢复现场程序。

2 按键唤醒功能

应用需要省电时，通常会采用睡眠加按键唤醒的方式去处理。按键唤醒功能需要此 IO 具有电平变化中断功能，FMD 大部分芯片的电平变化中断功能在 PORTA 端口，也有部分芯片为下降沿变化中断（如 FT61F04X 系列的 PORTA），需要注意电平状态。

推荐的电平变化中断唤醒的程序如下：

```
GIE=0;                  //关闭总中断
ReadAPin = PORTA;       //读端口状态存入缓存，设置电平变化中断匹配初值
PAIF =0;                //清电平变化中断标志
IOCA2 =1;               //使能 PA2 的电平变化中断功能
PAIE =1;                //使能电平变化中断
NOP();                  //建议加 1 到 2 条 NOP 指令
SLEEP();                //进入睡眠
NOP();                  //睡眠后需加 1 条 NOP 指令
ReadAPin = PORTA;       //读端口状态存入缓存，消除电平变化中断匹配条件
PAIF =0;                //清电平变化中断标志
IOCA2 =0;               //关闭 PA2 的电平变化中断功能
PAIE =0;                //关闭电平变化中断
GIE=1;                  //打开总中断
```

电平变化中断唤醒需要注意以下几点：

1）此按键 IO 的状态需要在初始化里设置好。例如设置为输入口并开启内部上拉电阻功能，IO 通过按键串联电阻连接到地。这样无按键时端口的电平状态固定在高电平，按键按下时端口电平由高变低，触发电平变化中断。

2）从读端口状态到进入睡眠，中间的指令越少越好。

3）睡眠前关闭总中断 GIE 的动作，会让芯片在唤醒后继续执行 SLEEP 后的程序。

4）如果睡眠前没有关闭总中断 GIE，那么唤醒后会去跳转到中断入口地址 04H 去执行中断程序，中断返回后继续执行 SLEEP 后的程序。因此需要在中断程序里加入读端口状态和清电平变化中断标志的动作，以免因为端口电平变化中断的条件一直匹配而导致程序反复进入电平变化中断而不执行后面的程序。

推荐例程如下：

```
void interrupt ISR(void)          //中断函数
{
    if(PAIE && PAIF)              //PA 电平变化中断
    {
        ReadAPin = PORTA;         //读端口状态存入缓存，消除电平变化中断匹配条件
        PAIF = 0;                 //清电平变化中断标志
    }
}
```

3 EEPROM 注意事项

1）在使用 EEPROM 的程序时，为防止上电时电压不稳定引起读写 EEPROM 不稳定，程序上需要增加约 100ms 延时后再进行 EEPROM 操作。如果编译选项里使能了 PWRT（上电延时）则程序上只需增加约 40ms 延时再进行 EEPROM 读写操作。

2）在使用 EEPROM 之前需要对 EEPROM 进行初始化操作，在未使用到 EEPROM 任意一个地址写入两次“0xAA”，后续程序不要操作到此地址。比如：

```
void SYSTEM_INITIAL (void)
{
    EEPROMwrite(0xFF,0xAA);
    EEPROMwrite(0xFF,0xAA);     //在未使用到的任意一个地址(0xFF)写两次 0xAA
}
```

3）写入 EEPROM 数据过程需要按照以下例程步骤进行：

```
void EEPROMwrite(unchar EEAddr,unchar Data)
{
    GIE = 0;                    //步骤 A:写数据必须关闭中断
    NOP();                      //注意:芯片运行在 1T 时，这里要加 NOP()
    while(GIE);                 //步骤 B:等待 GIE 为 0
    EEADR = EEAddr;             //步骤 C:写入 EEPROM 的地址
    EEDAT = Data;               //步骤 D:写入待写 EEPROM 的数据
    EECON1 |= 0x34;             //步骤 E:置位 WREN1,WREN2,WREN3 三个标志.
    WR = 1;                     //步骤 F:置位 WR 启动硬件 EEPROM 编程
    NOP();                      //注意:芯片运行在 1T 时，这里要加 NOP()
    while(WR);                  //步骤 G:等待 EEPROM 写入完成（过程约 2ms）
    GIE = 1;                    //步骤 H:写入完成，开启中断
}
```

如果软件启用了看门狗，那么在进入 EEPROM 写程序后需要清看门狗。

如果部分应用在中断里还有其他功能要实现（如模拟 PWM 输出），则 EEPROM 写入过程的 2ms 时间内不能关闭中断，可以将步骤 H 与步骤 G 位置互换。

4）写入 EEPROM 数据完成后，应将 EEPROM 实际值和要写入的目标值进行比较核对。如果相等则说明写入成功，否则写入失败，程序可采取重写操作。

```
EEPROMwrite(0x00,0xCC);         //往 EEPOM 的 00H 地址写入 0xCC
EEReadData = EEPROMread(0x00);//读取 0x00 地址 EEPROM 值
if(EEReadData!=0xCC)            //判断实际值是否等于 0xCC
{
    EEPROMwrite(0x00,0xCC);     //重新往 EEPOM 的 00H 地址写入 0xCC
}
```

5）如果在 EEPROM 的写入过程中发生了外部复位、WDT 溢出复位、LVR 复位或者非法指令引起的复位，那么标志位 EECON1.WRERR 会被置 1。上电初始化时可通过读取此标志位来判断前一次的 EEPROM 写入过程中是否发生异常情况，进而采取相应处理措施。

4 睡眠低功耗

在部分电池应用或需要低功耗的应用上，芯片必须使用睡眠来降低整机功耗。
FMD 芯片睡眠电流最低可以做到 3uA 以内，最低能到 0.3uA。
进入睡眠时，需要注意以下几方面：
1）芯片没有用到的功能外设尽量关闭。
2）LVD/LVR 功能启动后会有约 15uA 的电流，在睡眠时可根据实际情况来选择关闭 LVD/LVR 功能，睡眠唤醒后再打开。
3）普通 IO 在睡眠时应根据应用需要来设置。如果是普通输出口，则应设置成省电的输出。如果是普通输入口，则应使输入口保持固定的电平，不能处于浮空的状态。
4）编译选项里 FOSC 是否选为 INTOSC。如果选择 INTOSC，会导致芯片的 CLKO 脚位输出频率为 Fosc/4 的方波，影响睡眠电流。
5）部分芯片因为脚位没有全部封装出来，也需要将设置这些 IO，避免处于浮空输入的状态。
6）部分系列芯片的复位脚和 IO 口复用，且作为输入口时无内部上下拉电阻。需要特别注意如果此 IO 未使用到，需要将编译选项里 MCLRE 设置成 MCLR 功能，避免此 IO 处于浮空输入的状态。如 FT60F01X 的 PA3 和 FT60F02X 的 PA5。
7）如果应用特殊，关闭看门狗功能可以减少约 3uA 的电流。

5 内部上拉/下拉电阻功能

内部上拉电阻/下拉电阻只在 IO 处于数字输入口时生效。
1）先将 IO 口其他模拟功能先关闭（部分芯片默认是模拟口）。
2）设置相应的 IO 为输入端口。
3）打开相应的上拉电阻/下拉电阻的控制开关。
4）部分芯片的 PORTA 口还需设置 OPTION.7=0(/PUPA)来使能 PORTA 的上拉电阻。

6 PORTA 口操作注意事项

PORTA 是一个 8 位双向端口。与其相应的进出方向寄存器就是 TRISA 寄存器。
反之，将某一位设置为 0 会将该对应 PORTA 端口设置为输出端口。
在置为输出端口时，输出驱动电路会被打开，输出寄存器里的数据会被放置到输出端口。
当 IO 处于输入状态时，对 PORTA 进行读动作， PORTA 内容会是反映输入端口的状态。
在 PORTA 上进行写动作时， PORTA 内容会被写入输出寄存器。
所有的写操作都是“读-更改-写”的流程，即数据被读，然后更改，再写入输出寄存器的过程。
此特性在操作部分 LED 数码管时需要特别注意。

在部分应用中要视情况设置编译选项里的 RDCTRL 里为 LATCH，在输出状态下读 PORTA 返回的值为 PORTA 的缓存值，而非 PAD 上的真实电平值。

7 使用外部晶振注意事项

使用外部晶振时，MCU 晶振相邻脚输入输出高频脉冲时，会干扰晶振，引起晶振抖动，从而影响时序，造成计时偏差。

具体相邻脚位结合封装脚位图看 :例如 FT61F145-TRB, PC0 就是晶振邻脚 ,使用外部晶振时 ,PC0 不要用做高频信号输入输出 IO 口，具体看下图：

型号	邻脚	晶振脚	邻脚	相同型号
FT60F02X	PA1	PA7,PA6	VDD	
FT61F02X	VDD	PA7,PA6	PA5	
FT61F04X	NC	PA6,PA7	PC5	
FT60F22X	PA5	PA6,PA7	PB5	
FT60F12X	VDD	PA7,PA6	PA5	FT60F11X
FT61F13X	GND	PC1,PC0	PB7	FT62F13X
FT62F08X	PC0	PC1,PB7	GND	FT61F18X
FT61F14X	PC0	PC1,PB7	GND	
FT64F0AX	PC0	PC1,PB7	GND	FT61F0AX

8 使用通信接口注意事项

1）I2C

外挂 FT24C02： SDA，SCL 设为开漏输出，需要外部上拉

2）SPI

外挂 FT25C64 接 4 根线，SO,SI,SCL 设为开漏输出，需要外部上拉，CS 用普通 IO 驱动

3)USART

TX 的方向寄存器要设为输出，RX 要设为输入

9 PWM 应用注意事项

虽然周期和占空比的双缓冲在很大程度上保证 PWM 输出不会产生毛刺，但如果软件非常接近 TIM 匹配时刻去写周期和占空比寄存器，特别是在 TIM 时钟频率比系统时钟快的情况下，则有可能出现不可预料的情况，导致寄存器不是期望值，所有强烈建议更新周期和占空比寄存器，只在 TIM 匹配中断里面做。

第二章 仿真注意事项

1 仿真工具版本

仿真工具分为三部分，仿真器上位机软件（IDE），仿真器（Link），仿真器的固件 FW。
IDE 的版本信息可以在 IDE 界面的“菜单栏-Help-About…”里查看，如 2.1.0。
通过官网 http://www.fremontmicro.com/可下载最新的 IDE 软件。
安装前先关杀毒软件，可以装在 D 盘，新版 IDE 内置 C 编译器，不用单独调用。
Link 的版本较少更新，版本丝印打在 PCB 面板上，如 Link_V2.5
仿真器的固件 FW 版本信息可以在 Link 插上电脑后，在 IDE 右下角显示，如 FW Ver:V0.0.17，一般随 IDE 的版本进行升级，如需手动升级或降级，可点击“菜单栏-Help-Update Firmware…”进行更新固件，更新文件保存在 IDE“安装目录\Update\AppUpdate.bin”下。

2 仿真连接方法

1）将仿真器 LINK 连接电脑，在仿真器右上角的电压选项用跳冒选择好给芯片的供电电压。
2）连接 LINK 右边的 VCC、GND、CLK、DAT 排针到芯片 VCC、GND、ICSPCLK、ICSPDAT。
3）打开工程后点击 Build All 按钮会弹出编译选项 ,选择好编译选项后点击确定 ,IDE 即可通过 LINK 将程序下载到芯片里。
4）下载完成后会自动进入 DEBUG 界面，使用 DEBUG 工具栏的图标即可进行仿真调试。
需要注意的是仿真器在下载程序时需要对芯片进行重新上电，所以目标板不能外接电源，且目标板上最好不要有大电容（100uF 以上）影响芯片的重新上电。
ICSPCLK 和 ICSPDAT 不要外接其他会影响调试通信电平状态的元器件。

3 仿真下载失败处理

仿真调试出现下载失败的情况，可分两种情况分析，一是用 Link 连单独芯片时下载失败，二是用在板调试时下载失败。
针对第一种情况，分以下几步分析：
1）检查工具版本是否最新。
2）检查 Link 和芯片的连线是否正确，是否接触良好。
3）检查编译选项里 LVR 的电压是否高过 Link 的输出电压。
4）检查芯片型号是否正确，更换芯片测试。
5）检查芯片 VCC、ICSPCLK、ICSPDAT 通信电平波形是否正常。
针对第二种情况，分以下几步分析：
1）用 Link 单独连芯片，按第一种情况排查。如可以下载，则进行下一步判断。
2）检查芯片的 VCC 和 GND 之间是否有超过 100uF 的大电容。
3）检查芯片的 CLK,DAT 脚上是否有容性器件或其他元件影响通讯电平。
4）检查芯片的复位脚功能是否开启。将编译选项里 MCLRE 设置为 PA3/PA5。

4 外部供电仿真方法

1）将仿真器的电压选择跳冒从 3.3V/3.7V/5.0V 档位换到 External。

2）将外部供电 VCC 端接到仿真器 P1 端口的 VCC。

3）将外部供电 GND 端接到仿真器 P1 端口的 GND。

4）将仿真器的 Program 四线接到仿真板上芯片的 VCC、GND、ICSPCLK、ICSPDAT。

5）外部供电电压不得超过 5.5V。

采用外部电源供电仿真时，需要通过仿真器去控制外部电源对目标板的供电，目标板和外部电源不能直接连接，须通过仿真器去控制连接。

第三章 烧录注意事项

1 连接自动烧录机台

1）烧录器电源使用 7.5V/9V 电源供电。
2）烧录器的 20PIN 牛角座背面丝印的 PROGRAM，BUSY，OK，FAIL，GND 和 GND 下方未标注的引脚（3.3V）分别连接到烧录机台串口线的 START，BUSY，OK，NG，GND，VCC。注意不要将标注 VDD 的引脚连接到机台的 VCC。
3）在烧录机台设置烧录器选项里将启动电平、BUSY 电平、OK 电平、NG 电平设置为 L 有效，时间参数均使用默认设置。

2 回读芯片校验码

1）将芯片按照烧录脚位放置在烧录器上。
2）轻按烧录器上的 KEY_MODE 按键。
3）烧录器发出“滴”提示音，并在显示屏上显示“IC’s checksum: XXXX”。
4）如果未出现此结果，需要检查是否空芯片或者芯片连接是否正常。

3 烧录器模式设置

烧录器有两大功能模式：Write 烧录模式和 Check 校验模式。
将需要校验的程序下载到烧录器后，长按 KEY_MODE 按键，直到烧录器显示屏显示“Write Flash”，进入烧录器功能选择模式，此时可通过红色按钮切换烧录器模式。
Write 功能下有以下模式（下载程序时烧录器会自动选择烧录模式，不建议手动切换）：

Write Flash:	烧录芯片程序 HEX
Write Flash EEPROM:	烧录芯片程序 HEX 和 EEPROM.BIN
Write Fla EEP Rcode:	烧录芯片程序 HEX 和 EEPROM.BIN，烧录滚动码
Write Fla EEP Rd FOSC:	烧录芯片程序 HEX 和 EEPROM.BIN，烧录滚动码并校准频率
Write Flash Rcode:	烧录芯片程序 HEX 和滚动码
Write Fla Rd FOSC:	烧录芯片程序 HEX，烧录滚动码并校准频率
Write Fla FOSC:	烧录芯片程序 HEX 并校准频率
Write Fla EEP FOSC:	烧录芯片程序 HEX 和 EEPROM.BIN 并校准频率
Write EEPROM:	烧录 EEPROM.BIN
Write EEPROM R_code:	烧录 EEPROM.BIN 和滚动码

Check 功能下有三个模式：

Check FOSC:	检查芯片频率（此模式会擦除芯片程序，检查完后需要重新烧录）
Check Flash:	检查程序(只适用于代码不加密的情况：CPB=Disable)
Check checksum num:	检查芯片的 Checksum（可连机台自动检测）

4 烧录器的供电电压

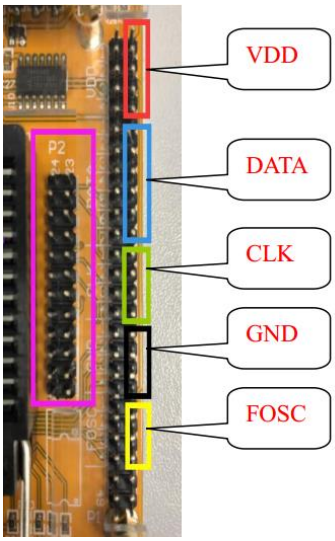
烧录器的供电有两种方式：

1）通过 USB 供电，5V 电源接 USB 线即可。

2）通过 DC 电源供电，供电电压为 7.5V 或 9V。

5 烧录器在板烧录

将烧录器上 P1 排针对应的 VDD、GND、CLK、DAT、FOSC 的右边排针和目标板上对应脚位连接好即可烧录。具体参考下图：



部分应用需要，预留烧录口没有引出 FOSC 来检测芯片频率。此时需要在烧录器软件 Setting 界面导入程序文件后，将下方的 Test Frequency 勾选项去掉，即可通过四线烧录。

6 烧录失败处理

出现烧录失败可分两种情况分析，一是单独烧录芯片时失败，二是在板烧录时失败。

针对第一种情况，分为以下几步分析：

1）检查工具版本是否最新，芯片型号是否正确。

2）检查烧录器和芯片的连线是否正确，是否接触良好。

3）检查编译选项里 LVR 的电压是否高过烧录器的输出电压（烧录器显示屏右边跳冒选择）。

4）检查芯片 VCC、ICSPCLK、ICSPDAT 通信电平波形是否正常。

5）检查程序的编译选项里的 FOSC 是否为外部晶振模式，如果采取了外部晶振模式，且 FCMEN 时钟故障检测功能被禁止，会导致芯片无法启动进入烧录模式。

针对第二种情况，分以下几步分析：

1）用烧录器单独连芯片，按第一种情况排查。如可以烧录，则进行下一步判断。

2）检查在板芯片的 VCC 和 GND 之间是否有超过 100uF 的大电容。

3）检查在板芯片的 CLK,DAT 脚上是否有容性器件或其他元件影响通讯电平。

4）检查在板芯片的复位脚功能是否开启。将编译选项里 MCLRE 设置为 PA3/PA5。

第四章 IDE 使用注意事项

1 编译错误提示

1）编译时提示“This Language Toolsuite is not Exists”

一般是 C 编译器未安装或调用的路径不对引起的。在 IDE 目录下打开一个工程后，“菜单栏-Project-Select Language Toolsuite…”在弹出的框里点击 Browse…选择编译器的安装目录下的编译器程序即可。

2）编译时提示“File not exist”或“No such file or directory”

采用汇编语言编程需要在程序起始包含头文件：

#include <头文件.inc>或#include “头文件.inc”。他们的区别在于头文件的位置：

使用#include <头文件.inc>时，编译器会在 IDE 软件目录下查找是否有此头文件。

使用#include “头文件.inc”时，编译器会在当前项目目录下查找是否有此头文件。

2 添加头文件

用户可通过 include XXX.inc 或 include XXX.h 来添加自定义的头文件。编译通过后，头文件会显示在 File View 的 Include File 目录下。暂不支持手动在 Include File 里直接添加。

3 ICSPCLK 和 ICSPDAT 的 IO 功能

仿真时 LINK 会占用 ICSPCLK 和 ICSPDAT 与芯片通信，所以仿真时 ICSPCLK 和 ICSPDAT 不能仿真普通 IO 功能。需要在程序下载完成，重新上电后，ICSPCLK 和 ICSPDAT 才可以实现普通 IO 功能。

4 仿真时关闭看门狗

仿真时，如果启用看门狗功能，要注意清看门狗，否则看门狗溢出时会产生复位，使 IDE 无法获取正确的 PC 值，从而出现仿真错误。建议仿真调试时关闭看门狗功能，需整机测试时再打开看门狗功能。

5 项目编译失败

1）头文件引起。

IDE 里使用 C 语言需包含头文件 SYSCFG.H，使用汇编语言包含芯片系列.inc 头文件。

2）C 编译器的安装目录引起。

win10 系统会禁止 IDE 修改系统盘下 C 编译器里的文件。将 C 编译器安装在非系统盘或者鼠标右键单击 IDE 主程序，选择以管理员身份运行 IDE 可以解决此问题。

6 导出 EEPROM 文件

1）打开工程，选择菜单栏 Project->Export EEPROM Data 会生成一个.bin 类型的文件。
2）由于 IDE 不支持 EEPROM 文件的在线编辑 ,建议使用第三方文本编辑软件对 EEPROM 文件进行编辑。以 UltraEdit 为例，用 UltraEdit 打开文件，在菜单栏选择编辑->十六进制函数->十六进制编辑，就可以对 256 字节的 EEPROM 数据进行编辑。

7 IDE 烧录 HEX 文件

目前只能通过 IDE 的 Build All 来进行工程的编译和下载 ,单独 HEX 文件只能通过烧录器烧录芯片。后续 IDE 会增加单独烧录 HEX 文件的功能。

8 更新 IDE 后校验码不一样

因为早期 IDE 版本开放过 LVRS=1.8V 的功能，所以部分程序编译选项使用 LVRS=1.8V 的工程使用新的 IDE 编译后会默认将 LVRS 提升至 2.0V。程序校验码会包含编译选项里的内容，所以校验码会和之前的校验码有少许差异。
如校验码差异较大，建议检查 C 编译器是否为限制版（编译信息会提示 Lite Mode）。

9 库文件打包设置

1）建立工程文档。
2）将需要打包的程序写成一个 C 文件，包含进工程文档。
3）编译完成后，工程目录下会生成与 C 文件同名的 lpp 文件。
4）将 lpp 文件包含进工程文档 LIB File，删除子文件后重新编译即可。

第五章 烧录软件注意事项

1 烧录器联机

1）烧录器通过 USB 线连接电脑。
2）打开烧录器软件，软件左上角会显示软件版本号，如 FMDWriter V5.1。
3）烧录器软件右上方 FMD 图标下的 DRV Ver 后会显示固件版本号：如 DRV Ver:V2.1.0。如果未显示版本号，说明烧录器未连接成功，建议拔插 USB 接线重新连接。
4）如果烧录器固件版本较低，与烧录软件连接后会提示固件版本已过期，选择更新后，在弹出对话框里选择确定，等待两分钟左右，烧录器自动完成更新。固件更新完成后需要重新拔插 USB 重新连接，重新连接成功后 DRV Ver 后面会显示当前固件版本号。
5）通过点击烧录器软件右上方 FMD 图标可以进行手动更新固件。

2 导入文件

点击“器件\设置”按钮，烧录器软件会自动弹出设置选项框。
选择芯片的系列和型号后，在下方文件选择 Flash 文件和 EEPROM 文件。
如果需要烧录程序文件，则点击浏览选择待烧录 Flash 文件。
如果需要烧录 EEPROM，则点击浏览选择待烧录 EEPROM 文件。
单独选择程序文件，烧录器只烧录芯片程序区，不影响芯片原有的 EEPROM 数据。
单独选择 EEPROM 文件，烧录器只烧录芯片 EEPROM 区，不影响芯片原有的程序数据。
如果不选择文件，那么只能通过烧录软件去读取芯片数据。
导入文件后，烧录器的其他功能设置：
1）测频：5%（默认使能）
通过芯片 FOSC 测试芯片内部频率是否偏差超过±5%，如超过±5%判断为 NG 芯片。
2）校频：1%（默认关闭）
通过芯片 FOSC 测试芯片内部频率是否偏差超过±1% ,如超过±1%则烧录器会校准频率至±1%内，如果芯片无法校准在±1%以内，则判断为 NG 芯片。因芯片出厂测试频率偏差范围为±1.5%内，勾选此项会导致部分正常芯片因无法校准在±1%范围内而被判为 NG，所以一般情况不建议客户勾选此项。部分特殊应用需要芯片内部频率偏差范围不超过±1%，则可勾选次选项。
3）烧录数起始值和最大值
用于设置烧录器烧录数量，烧录芯片达到最大值时，烧录器不再烧录，需重新导入文件。
4）滚码
用于设置芯片滚动码，该滚动码存于芯片 EEPROM 区。

3 读取芯片的校验码

芯片的校验码可以通过烧录器上的 Key_Mode 按键去读回并显示在烧录器显示屏上。
另外，烧录器软件还可以读取芯片的程序区内容和 EEPROM 区内容。
如果芯片 CPB 使能加密了，那只能读取 EEPROM 文件。
读取回的数据可以点击 Flash in IC 或者 EEPROM in IC 查看对应的数据。

4 烧录器软件各按钮的功能

Auto Program： 烧录器便会依次执行“擦除”、“程序烧录”、“烧录校验”动作
Erase： 仅对芯片进行擦除操作（擦除可选择 Flash 和 EEPROM）
Blank Check： 仅对芯片进行空白检查
Program： 仅对芯片执行烧录动作,不校验是否烧录成功
Verify： 检查芯片中的代码是否正确，(如果烧录程序使能了 CPB，则无法校验)
Read： 读出芯片信息（如果芯片有加密则只能读出校验码和 EEPROM 数据）
Setting： 设置烧录数量
Rolling Code： 滚码设置
Calibration Frequency：校准频率（勾选后，检测到频率偏移芯片就会自动校准）
Test Frequency：检测频率（勾选后，检测到频率偏移的芯片就会按 NG 处理）

5 烧录器软件不显示 FT61F04X 芯片

点击 Device/Load 按钮后，会弹出选择芯片型号的对话框。
如果没有当前芯片的系列号，需要检查：
1、烧录器是否正常联机。（DRV Ver 是否显示烧录器固件号？）
2、烧录器是否为黄色版本烧录器。早期黑色/蓝色/绿色烧录器不能烧录 4K 及以上容量芯片。
3、如果为黄色版本，请确认烧录器固件号是否为最新的版本。

6 烧录器报警提示说明

Write Flash fail： 烧录 Flash 失败
Write UCFG fail： 烧录用户配置失败
Write EEP fail： 烧录 eeprom 失败
Write UID fail： 烧录滚码失败
NO Flash File： 烧录器里面没有烧录程序
No Eeprom File： 烧录器里面没有 eeprom 程序
Uid Not Loaded： 烧录器里面没有滚码

ALL UIDs USED：	所有滚码已经使用完毕
Rd CPU_STAT Err：	芯片引脚接错了或者接触不良
Read ICID Err：	烧录芯片与烧录器所下载的 HEX 对应型号不匹配
ICID_1T MATCH NG：	FT61F14X、FT61F0AX、FT64F0AX、FT67F0AX 某些版本不支持 1T 工作模式
fosc check fail：	测频失败，芯偏频偏过大或者 CLKO 脚位接错接触不良等原因造成
fosc cal fail：	芯片频率校准失败
Read FCFG Err：	若是加密程序，则说明密码与芯片配置的秘密不一样，须确认双方密码之后再次烧录；再或者就是芯片内部信息错误。
No ICSeriesinfo：	跟新固件后会提示，让用户导入烧录程序

第六章 汇编语言编程注意事项

1 伪指令 BANKSEL

伪指令 BANKSEL 用于设置寄存器的 BANK，作用是设置 BANK 为 BANKSEL 后面的寄存器所在的 BANK。

如 FT60F01X 和 FT60F02X：

```
BANKSEL   PORTA; 因为 PORTA 在 BANK0，所以指令等同于 BCR   STATUS,5
BANKSEL   TRISA;  因为 TRISA 在 BANK1，所以指令等同于 BSR   STATUS,5
```

在 FT61F02X 中 ,寄存器 BANK 共有 3 个 :BANK0 ,BANK1 ,BANK2 ,需要通过 STATUS 的 BIT5，BIT6 去设置，所以一条 BANKSEL 等同于两条指令。

```
BANKSEL   PORTA，等同于：
    BCR     STATUS,5
    BCR     STATUS,6
BANKSEL   TRISA，等同于：
    BSR     STATUS,5
    BCR     STATUS,6
```

这在需要精确计算指令数时要特别注意。

2 汇编指令周期

汇编指令周期指执行一条汇编指令所需要的时间。

按照不同系列的芯片，指令周期的时间会有所不同。

目前我们有 1T、2T、4T 的指令周期。

1T 的指令周期指的是执行一条汇编指令周期需要 1 个系统时钟周期。

2T 的指令周期指的是执行一条汇编指令周期需要 2 个系统时钟周期。

4T 的指令周期指的是执行一条汇编指令周期需要 4 个系统时钟周期。

例如：FT60F01X 为 4T 指令周期。

选择 OSCCON.IRCF[2-0]=111 时，内部振荡频率 Fosc=16M。系统时钟周期为 1/16M 秒。

此时一个指令周期即为 4*1/16M=1/4M 秒=0.25us。即执行一个 NOP 需要耗时 0.25us。

FT60F02X 为 2T/4T 可选指令周期。可以在编译选项的 Tsel 选择为 2T 指令周期。

选择 OSCCON.IRCF[2-0]=111 时，内部振荡频率 Fosc=16M。系统时钟周期为 1/16M 秒。

此时一个指令周期即为 2*1/16M=1/8M 秒=0.125us。即执行一个 NOP 需要耗时 0.125us。

FMD 汇编指令大部分都是单周期指令，即一条指令占用一个指令周期。

部分有跳转 PCL 功能的指令为双周期指令，如 LJUMP、LCALL、RET、RETI、RETW。

部分判断跳转指令在满足跳转条件时为双指令周期，不满足跳转条件时为单指令周期。如 BTSS、BTSC、INCRSZ、DECRSZ。

3 汇编指令功能说明

助记符	影响标志	周期	功能说明
BCR R,b	NONE	1	寄存器 R 的第 b 位清 0
BSR R,b	NONE	1	寄存器 R 的第 b 位置 1
BTSC R,b	NONE	1 或 2	如果寄存器 R 的第 b 位为 0 则跳过下一条指令，发生跳转时为 2 周期
BTSS R,b	NONE	1 或 2	如果寄存器 R 的第 b 位为 1 则跳过下一条指令，发生跳转时为 2 周期
NOP	NONE	1	空指令
CLRWDT	/PF, /TF	1	清看门狗
SLEEP	/PF, /TF	1	睡眠指令
STR R	NONE	1	将累加器 W 的值存入寄存器 R
LDR R,d	Z	1	将寄存器 R 的值导入累加器 W(d=W/0)或寄存器 R(d=F/1)
SWAPR R,d	NONE	1	将寄存器 R 的高四位和低四位交换后存入累加器 W(d=W/0)或寄存器 R(d=F/1)
INCR R,d	Z	1	寄存器 R 的值加 1,结果存入累加器 W(d=W/0)或寄存器 R(d=F/1)
INCRSZ R,d	NONE	1 或 2	寄存器 R 的值加 1,结果存入累加器 W(d=W/0)或寄存器 R(d=F/1) 如果结果等于 0，则跳过下一条指令，发生跳转时为 2 周期
ADDWR R,d	C, HC, Z	1	将累加器 W 的值和寄存器 R 的值相加 ,结果存入累加器 W(d=W/0)或寄存器 R(d=F/1) 如果结果大于 255，则 C=1
SUBWR R,d	C, HC, Z	1	将寄存器 R 的值减去累加器 W 的值，结果存入累加器 W(d=W/0)或寄存器 R(d=F/1) 如果 R>=W，则 C=1
DECR R,d	Z	1	寄存器 R 的值减 1 结果存入累加器 W(d=W/0)或寄存器 R(d=F/1)
DECRSZ R,d	NONE	1 或 2	寄存器 R 的值减 1,结果存入累加器 W(d=W/0)或寄存器 R(d=F/1) 如果结果等于 0，则跳过下一条指令，发生跳转时为 2 周期
ANDWR R,d	Z	1	寄存器 R 的值和累加器 W 的值相与,结果存入累加器 W(d=W/0)或寄存器 R(d=F/1)
IORWR R,d	Z	1	寄存器 R 的值和累加器 W 的值相或,结果存入累加器 W(d=W/0)或寄存器 R(d=F/1)
XORWR R,d	Z	1	寄存器 R 的值和累加器 W 的值异或,结果存入累加器 W(d=W/0)或寄存器 R(d=F/1)
COMR R,d	Z	1	寄存器 R 的值取反,结果存入累加器 W(d=W/0)或寄存器 R(d=F/1)
RRR R,d	C	1	寄存器 R 的值带 C 位右移，结果存入累加器 W(d=W/0)或寄存器 R(d=F/1)
RLR R,d	C	1	寄存器 R 的值带 C 位左移，结果存入累加器 W(d=W/0)或寄存器 R(d=F/1)
CLRW	Z	1	清累加器 W
CLRR R	Z	1	清寄存器 R
RETI	NONE	2	中断返回
RET	NONE	2	子程序返回
LCALL N	NONE	2	调用子程序 N
LJUMP N	NONE	2	跳转到标号 N
LDWI I	NONE	1	将立即数 I 导入累加器 W
ANDWI I	Z	1	累加器 W 的值和立即数 I 相与，结果存入累加器 W
IORWI I	Z	1	累加器 W 的值和立即数 I 相或，结果存入累加器 W
XORWI I	Z	1	累加器 W 的值和立即数 I 异或，结果存入累加器 W
RETW I	NONE	2	子程序返回，并将立即数 I 存入累加器 W
ADDWI I	C, HC, Z	1	累加器 W 的值和立即数 I 相加，结果存入累加器 W，如果结果大于 255，则 C=1
SUBWI I	C, HC, Z	1	累加器 W 的值和立即数 I 相减，结果存入累加器 W，如果 I>=W，则 C=1

4 汇编查表

在汇编指令中处理查表功能时需要特别注意 PCLATH 的处理。

汇编里通常通过 RETW XXX 来返回表格数据，常用的数码管查表程序例程如下：

```
        LDR     DIG_2,W     ;将第二位赋值给 W
        LCALL   GET_CODE    ;查表
        STR     TEMP0       ;数据存入 TEMP0

        ORG     0X300       ;定义表格程序地址到 300H 地址
GET_CODE:
        LDWI    0X03
        STR     PCLATH      ;设置 PCLATH=3（跟随表格实际地址）
        LDR     BCD,W       ;查表
        ADDWR   PCL,F       ;--GFEDCBA
        RETW    B'00111111' ;0
        RETW    B'00000110' ;1
        RETW    B'01011011' ;2
        RETW    B'01001111' ;3
        RETW    B'01100110' ;4
        RETW    B'01101101' ;5
        RETW    B'01111101' ;6
        RETW    B'00000111' ;7
        RETW    B'01111111' ;8
        RETW    B'01101111' ;9
        RETW    B'00000000' ;OFF
```

5 LJUMP 和 LCALL 指令

因为 PC 宽度限制的问题，LJUMP 和 LCALL 最多只能访问到 2KROM 空间。

程序空间在 2K 内的芯片使用 LJUMP 和 LCALL 指令时无需考虑跳转出错的问题。

2K 容量以上的芯片在使用 LJUMP 和 LCALL 访问 2K 以上的空间时，需要设置 PCLATH。

超过 4K 程序空间的芯片，建议使用 C 语言编程，编译器会自动处理。

6 标号的使用

在汇编里设置子程序标号或者位置标号时，需要注意在标号增加英文冒号：

不加冒号会被编译器识别成宏定义，如果头文件里没有这个宏定义就会引起编译器报错。

第七章 C 语言编程注意事项

1 包含头文件

在程序起始位置必须用#include 指令包含编译器提供的“syscfg.h”文件 ,以实现特殊功能寄存器和其它特殊符号的声明。不用单独包含芯片型号的头文件，IDE 会自动调用。
其他子程序文件和头文件也需要在 Main 文件里用#include 指令包含。

2 变量类型

常用变量的类型如下表：需要注意的是 char 默认为无符号字符变量。

类型	长度（位）	数学表达
bit	1	布尔型变量
char	8	字符变量，默认为无符号字符变量
signed char	8	有符号字符变量
unsigned char	8	无符号字符变量
int	16	整型变量，默认为有符号整型变量
signed int	16	有符号整型变量
unsigned int	16	无符号整型变量
long	32	长整型变量，默认为有符号长整型变量
signed long	32	有符号长整型变量
unsigned long	32	无符号长整型变量
float	24	浮点数
double	24	双精度浮点数

3 变量声明

const：常数型变量

如果在变量定义时加前缀 const 声明，那么此变量就会成为常数型变量，程序运行过程中不能对其修改，通常用于常量数组的定义申明。

volatile：常驻内存型变量

C 编译器会对程序进行优化，部分在程序里无实际作用的变量会被优化处理，在调试过程中就无法观察变量实际值。如果希望变量不被优化，需要在此变量定义时加 volatile 的前缀进行声明。

persistent：非初始化变量

在程序上电运行时会将所有定义的变量进行清零或设置初始值。但在许多实际应用中不运行程序对部分变量进行初始化，此时可以在此变量定义前加入 persistent 前缀声明。

4 绝对地址变量

部分应用需要将 8 个位变量放在同一个字节中进行批量操作，可以通过两种方式：

1）通过定义一个位域结构和一个字节变量联合来实现：

```
union {
    struct {
        unsigned bit0:    1;
        unsigned bit1:    1;
        unsigned bit2:    1;
        unsigned bit3:    1;
        unsigned bit4:    1;
        unsigned bit5:    1;
        unsigned bit6:    1;
        unsigned bit7:    1;
    } oneBit;
    unsigned char allBits;
} TESTFlag;
```

设置其中某一位时操作如下： TESTFlag.oneBit.bit5=1; //bit5 位置 1

位变量批量清零时操作如下： TESTFlag.allBits=0; //位变量批量清 0

2）通过定义变量的绝对地址的方式来实现

```
volatile unsigned charTESTFlag @ 0x20; // TESTFlag 定义在地址 0x20
bit TESTBit0 @((unsigned)&TESTFlag*8)+0;//TESTBit0 对应于 TESTFlag 第 0 位
bit TESTBit1 @((unsigned)&TESTFlag*8)+1;//TESTBit1 对应于 TESTFlag 第 1 位
bit TESTBit2 @((unsigned)&TESTFlag*8)+2;//TESTBit2 对应于 TESTFlag 第 2 位
bit TESTBit3 @((unsigned)&TESTFlag*8)+3;//TESTBit3 对应于 TESTFlag 第 3 位
bit TESTBit4 @((unsigned)&TESTFlag*8)+4;//TESTBit4 对应于 TESTFlag 第 4 位
bit TESTBit5 @((unsigned)&TESTFlag*8)+5;//TESTBit5 对应于 TESTFlag 第 5 位
bit TESTBit6 @((unsigned)&TESTFlag*8)+6;//TESTBit6 对应于 TESTFlag 第 6 位
bit TESTBit7 @((unsigned)&TESTFlag*8)+7;//TESTBit7 对应于 TESTFlag 第 7 位
```

需要注意的是：编译器对绝对定位的变量不保留地址空间。

上面例程里变量 TESTFlag 的地址是 0x20，但最后 0x20 地址有可能又被编译器分配给了其它变量使用，这样就发生了地址冲突。因此用户自定义变量的绝对地址时需要在定义变量时加上前缀 volatile 声明，防止编译器自动优化分配地址。

这样的用法会降低 RAM 的使用效率，所以在一般的程序设计中不建议定义变量的绝对地址。

第八章 具体型号使用注意事项

1．FT60F01X

1）PA3 的应用注意事项：

PA3 是单方向输入端口，当输入口用，必须外部上拉

只有当 PA3 设设为复位脚时，才会开启内部弱上拉

FT60F011A-RB 实际项目工程中没用到 PA3 时，也要设为复位脚，减少功耗

FT60F010A-URB PA3 没封装出来，也必须设为复位脚

2．FT60F02X

1）PA0,PA1 设为内部上拉：

关闭比较器模拟功能,CxIN 设为数字 IO 管脚：CMCON0：CM[2:0]=111

3．FT60F12X/FT60F11X

1）CLKO：

E 版之后 CLKO 映射到 PC5，而不时 PA6

2）外部晶振

设置外部晶振：FOSC:XT(PA6,PA7 接高速晶体 4~20Mhz)

A,IESO:Enable(使能双速时钟)，IRCF[2:0]=000，必须设置为 000

B,IESO:Disable(不使能双速时钟)，IRCF 可以随便设置

3）PWM

当 PWM 占空比- PWM 周期 = 1，PWM 输出的电平会翻转，需要用户写程序去规避这个问题

4．FT61F02X

1）PA0 PA1 PA1 PA3 设置输入上拉

A,关比较器 CMCON0 = 0B00000111，

B,关模拟口 ANSEL = 0X00;

C,设为输入 TRISA = 0B11111111;

D,上拉 WPUA = 0xFF;

5．FT61F04X

1）运放

运放只能输入负电压，如果输入信号为正电压，建议客户用 FT64F0AX

6．FT61F14X

1）PWM　IO 映射　[TIM1_CH2]

不能单独 PB0 输出 PWM,只能 PA1 单独输出，设置 PB0 时，PA1 也会输出 PWM

TIM2_CH1 输出使能时 ,PB0 不能配置为 TIM1_CH2 输出 ,否则 PB0 将输出 TIM2_CH1 波形。

2）TIM4 接外部晶振 32.769k 低功耗应用

如果系统时钟选 HIRC，TIM4 用 LP,睡眠的时候，系统时钟不能关，功耗做不小，目前可以设置，进入睡眠，系统时钟设为内部 32K，TIM4 中断唤醒　6014 7001 6006 都是一样的

起振电容建议用 33pF

7．FT62F08X/FT61F08X

1）中断函数应用注意事项：

C 语言程序中，中断函数必须保留，否则编译报错

2）扇区加密不能读 CHECKSUM

CPB 使能，可以读 CHECKSUM.

FSECPB0 扇区加密使能，就不能读 CHECKSUM,要注意，其他扇区加密不影响

3）ADC 应用注意事项：

1，ADC 校正，一定要寄存器设置好后才能校正

2，模拟通道 1/4VDD 是电阻分压取样，睡眠前 CHS[3:0]要切换到其他通道，否则会耗电，导致睡眠电流比较大

4）FT62F088-LRB,用做触摸功能的时候，PC7 被触摸库占用，必须悬空，不能做普通 IO 口用

8．FT62F21X/FT60F21X

1）单独输出 P1D　PWM 功能的时候:

P1DDTL 的低 7 位不能同时为 0，例如 P1DDTL=0X80,时，P1D 就输出低电平，而不是 PWM 波形，使用映射也是一样的，这个目前只能用软件去规避，P1CDTH 要赋值才可以，一定要注意

9．FT62F13X/FT61F13X

1）PA3,PA5 没有封装出来的型号，是和 GND 打到一起的，软件不要把他们置高，以免引起漏电

2）FT61F131B-RB,　PB3(PC0)合封，PA1(PA4)合封，一个 IO 口要设置为浮空输入，才能正常是用另一个 IO 的功能，要注意

3）指令周期设置为 4T，IO 的 EFT 抗干扰特性会更强