FT64F0AX

EEPROM Application note

目录

FT64F0Ax EEPROM 应用

1. 数据 EEPROM 和 PROM

数据 EEPROM 和闪存程序存储器是可读可写的。这两种存储器没有直接映射到数据存储器空间，而是通过特殊功能寄存器 (SFR) 间接寻址。有 7 个 SFR 用于访问这两种存储器：

- EECON1
- EECON2
- EEDATL
- EEDATH
- EEADRL
- EEADRH

当与数据 EEPROM 模块接口时，EEDATL 寄存器存放要读写的 8 位数据，EEADRL 寄存器存放被访问的 EEDATL 单元的地址。这些器件具有 128 字节的数据 EEPROM，地址范围从 0h 到 07Fh。

访问程序存储器模块时，EEDATH:EEDATL 寄存器对形成双字节字，存放要读/写的 14 位数据，而 EEADRL 和 EEADRH 寄存器形成双字节字，存放被读取的程序存储单元的 15 位地址。

EEPROM 数据存储器允许以字节为单位进行读写。EEPROM 字节写操作会自动擦除目标存储单元并写入新数据 (在写入前擦除)。写入时间由片上定时器控制。写入和擦除电压由片上电荷泵产生，此电荷泵能在器件电压范围内正常工作，用于字节或字操作。

器件能否对程序存储器的特定块执行写操作取决于配置字寄存器位 FSECPB0[7:0]的设置。然而，始终允许程序存储器的读操作。

当器件被代码保护时，调试接口将不再能访问数据或程序存储器。在代码保护时，CPU 仍可继续读写数据 EEPROM 存储器和程序存储器。

1.1. DATA EEPROM 相关寄存器汇总

名称	状态		寄存器	地址	复位值
EEADR [1]	PROM / DATA EEPROM 地址	低 8 位	EEADRL[7:0]	0x191	RW-0000 0000
		高 6 位	EEADRH[5:0]	0x192	RW-00 0000
EEDAT [2]	PROM / DATA EEPROM 数据	低 8 位	EEDATL[7:0]	0x193	RW-xxxx xxxx
		高 6 位	EEDATH[5:0]	0x194	RW-xx xxxx
EEPGD	存储器选择位	1 = 访问 PROM 0 = 访问 DATA EEPROM	EECON1[7]	0x195	RW-0
CFGS	PROM / DATA EEPROM 或配置寄存器选择位 1 = 访问配置寄存器 (读访问) 0 = 访问 PROM 或 DATA EEPROM		EECON1[6]		RW-0
FREE [3]	PROM 擦除使能位 1 = 在下一条 WR 命令执行擦除操作 (擦除完成后由硬件清零) 0 = 在下一条 WR 命令执行写操作 注：仅当 CFGS = 0 且 EEPGD = 1 时有效		EECON1[4]		RW-0
WRERR	PROM / DATA EEPROM 擦除/写错误标志位 1 = 写操作中止 (除 POR 之外的任何复位) 0 = 正常完成		EECON1[3]		RW-0
WREN [4]	编程/擦除使能位 1 = 使能　　0 = 禁止		EECON1[2]		RW-0
WR [5]	PROM / DATA EEPROM 写控制位 1 = 启动一次写或写正在进行中(完成后重置为 0) 0 = 完成		EECON1[1]		RW1-0
RD	PROM / DATA EEPROM 读控制位 1 = Yes (保持 4 个 SysClk 周期，然后 = 0) 0 = No		EECON1[0]		RW1-0
EECON2	PROM / DATA EEPROM 写操作解锁控制寄存器 在 EECON1 寄存器的 WR 置位前，必须先写 0x55，随后是 0xAA，用于解锁写操作。这些写操作必须在连续的指令周期完成		EECON2[7:0]	0x196	WO-xxxx xxxx

表 1-1　EEPROM 相关用户控制寄存器

[1] 在写周期 (约 2ms) 内，该寄存器不可写。用该寄存器访问程序存储器时，地址范围必须位于 0~0x1FFF，否则无法完成读写访问。
[2] 在写周期 (EEPROM: 3 ~ 5ms, PROM: 0.75 ~ 1.25ms) 内，该寄存器不可写。
[3] 访问数据 EEPROM 时：该位不起作用，下一条 WR 命令将启动一个擦除周期和一个写周期。
[4] 在写周期内，禁止对该寄存器位写。
[5] 软件写 1 后至少要等 1 个系统时钟才能回读。

名称	状态		寄存器	地址	复位值
GIE	全局中断	1 = 使能 (PEIE, EEIE 适用) 0 = 全局关闭 (唤醒不受影响)	INTCON[7]	Bank 首地址 +0x0B	RW-0
PEIE	外设总中断	1 = 使能 (EEIE 适用) 0 = 关闭 (无唤醒)	INTCON[6]		RW-0
EEIE	EEPROM 写完成中断	1 = 使能 0 = 关闭 (无唤醒)	INTCON[5]		RW-0
EEIF [6]	EEPROM 写完成中断标志位	1 = Yes (锁存) 0 = No	INTCON[2]		R_W1C-0

表 1-2 EEPROM 中断使能和状态位

1.2. EEADRL 和 EEADRH 寄存器

EEADRH:EEADRL 寄存器对可以寻址最大 128 字节的数据 EEPROM 或最大 32K 字的程序存储器。

当选择程序地址值时，地址的高字节写入 EEADRH 寄存器而低字节被写入 EEADRL 寄存器。当选择 EEPROM 地址值时，只将地址的低字节写入 EEADRL 寄存器。

1.2.1. EECON1 和 EECON2 寄存器

EECON1 是访问 EE 存储器的控制寄存器。

控制位 EEPGD 决定访问的是程序存储器还是数据存储器。当它为 0 时，任何后续操作都将针对 EEPROM 存储器进行。当置 1 时，任何后续操作都将针对程序存储器进行。复位后，EEPGD 清 0，即默认选中 EEPROM。

控制位 RD 和 WR 分别启动读和写。用软件只能将这些位置 1 而无法清零。在读或写操作完成后，由硬件将它们清零。由于无法用软件将 WR 位清零，可避免因意外而过早地终止写操作。

当 WREN 位置 1 时，允许执行写操作。上电时，WREN 位被清零。在正常运行中当写操作被复位中断时，WRERR 位置 1。在这些情况下，复位后用户可以检查 WRERR 位并执行相应的错误处理程序。当写操作完成时，PIR2 寄存器的中断标志位 EEIF 被置 1。该标志位必须用软件清零。

读 EECON2 得到的是全 0。EECON2 寄存器仅在数据 EEPROM 写过程中使用。要使能写操作，必须将特定模式写入 EECON2。

1.3. 使用数据 EEPROM

数据 EEPROM 是高耐久性、可字节寻址的阵列，已将其优化以便存储频繁更改的信息 (例如：程序变量或其他经常更新的数据)。软件开发者应将不频繁更改的变量 (例如常量、ID 和校准值等) 存储在闪存程序存储器中，以免超出 EEPROM 字节最大的写允许次数。

1.3.1. 读数据 EEPROM 存储器

要读取数据存储单元，用户需要把地址写入 EEADRL 寄存器，清零 EECON1 寄存器的 EEPGD 和 CFGS

[6] 写 1 清 0，写 0 无效。建议只使用 STR、MOVWI 指令进行写操作，而不要用 BSR 或 IOR 指令。

控制位，再置 1 控制位 RD。等待 4 个系统时钟周期后，数据将更新到 EEDATL 寄存器中，EEDATL 将把此值保持至下一次读取或用户向该单元写入数据时 (在写操作过程中) 为止。

读数据 EEPROM，示例：

```
BANKSEL     EEADRL      ;
LDWI        DATA_EE_ADDR ;
STR         EEADRL          ;Data Memory Address to read
BCR         EECON1, CFGS    ;Deselect Config space
BCR         EECON1, EEPGD ;Point to DATA memory
BSR         EECON1, RD      ;EE Read
; wait 4 system clock
LDR         EEDATL, W     ;W = EEDATL
```

注意：

1. 无论 CPB 为何值，软件总是可以读取 EEPROM；

1.3.2. 写数据 EEPROM 存储器

要写 EEPROM 数据存储单元，用户应首先将该单元的地址写入 EEADRL 寄存器并将数据写入 EEDATL 寄存器。然后用户必须按特定顺序开始写入每个字节。

如果未完全按照上述顺序 (即，首先将 0x55 写入 EECON2，随后将 0xAA 写入 EECON2，最后将 WR 位置 1) 逐字节写入，将不会启动写操作。在该代码段中应禁止中断。

此外，必须将 EECON1 中的 WREN 位置 1 以使能写操作。这种机制可防止由于代码执行错误 (异常) 导致误写数据 EEPROM。除了更新 EEPROM 时以外，用户应始终保持 WREN 位清零。WREN 位不能由硬件清零。

一个写序列启动后，清零 WREN 位将不会影响此写周期。除非 WREN 位置 1，否则 WR 位将无法置 1。写周期完成时，WR 位由硬件清零并且 EE 写完成中断标志位 (EEIF) 置 1。用户可以允许中断或查询此位。EEIF 必须用软件清零。

注意：
软件对 EECON1.WR 写 1 后，至少等待一个系统时钟 (NOP 或者任何别的指令) 软件才能对该位进行读判断，否则将读回 0，进而影响程序的流程 (比如误认为写结束)。

1.3.3. 防止误写操作的保护措施

有些情况下，用户并不希望向数据 EEPROM 存储器写入数据。为了防止 EEPROM 误写操作，器件内建了各种保护机制。上电时，清零 WREN。同时，上电延时定时器 (64ms 的延时) 也会阻止对 EEPROM 进行写操作。

写启动序列和 WREN 位共同防止在以下情况下发生意外写操作：

- 欠压
- 电源故障
- 软件故障

1.3.4. 关于 GIE 的清 0

在启动 EEPROM 和 PROM 写之前，需要对 EECON2 顺序写 0x55 和 0xAA，且不能被打断。所以在做该开锁动作前应把 GIE 清 0 以屏蔽可能的中断。而由于中断的响应延时为 2 个 NOP，故在第一次清 GIE 后，等两个 NOP 再次判断 GIE 是否为 0，如以下代码示：

```
GIE= 0;
NOP;
NOP;
while (GIE) { GIE= 0;};
```

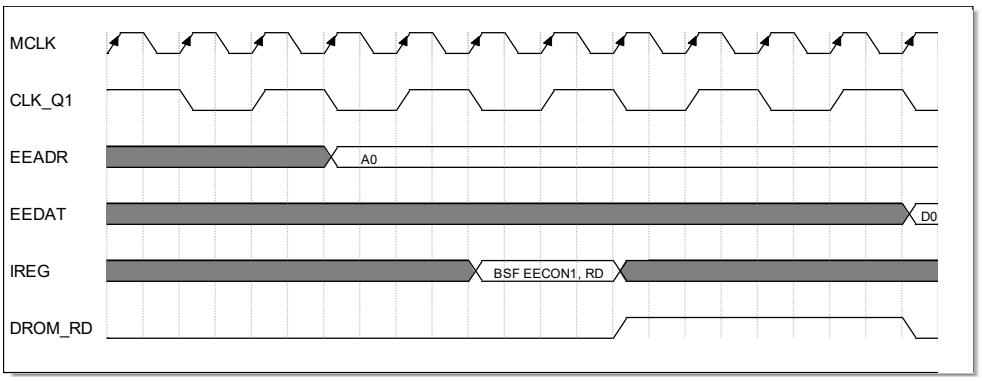

图 1-1 2T 模式下软件读 EEPROM 时序

写数据 EEPROM，示例：

```
BANKSEL EEADRL              ;
LDWI DATA_EE_ADDR           ;
STR EEADRL                  ;Data Memory Address to write
LDWI DATA_EE_DATA           ;
STR EEDATL                  ;Data Memory Value to write
BCR EECON1, CFGS            ;Deselect Configuration space
BCR EECON1, EEPGD                   ;Point to DATA memory
BSR EECON1, WREN            ;Enable writes

BCR INTCON, GIE             ;Disable INTs.
NOP
NOP
BTSC INTCON, GIE            ;Test again
GOTO $-1


LDWI 55h                    ;
STR EECON2                  ;Write 55h
LDWI 0AAh                   ;
STR EECON2                  ;Write AAh

BSR EECON1, WR              ;Set WR bit to begin write
BSR INTCON, GIE             ;Enable Interrupts
BCR EECON1, WREN            ;Disable writes
BTSC EECON1, WR             ;Wait for write to complete
LJUMP $-2                   ;Done
```

该程序流不能被中断

注意：

1. 数据 EEPROM 写周期并不暂停指令的执行。

1.4. 闪存程序存储器概述

了解闪存程序存储器结构对于擦除和编程操作非常重要。闪存程序存储器按行排列。每行包括固定数量的 14 位程序存储器字。行是用户软件可擦除的最小块大小。只有当目标地址位于未受写保护的存储器段内 (由配置字寄存器的 CPB 和 FSECPB0 定义)，才能对闪存程序存储器进行写或擦除操作。擦除某行后，用户可以对该行的部分或全部进行重新编程。写入程序存储器行的数据将被写入 14 位宽的数据写锁存器中。用户不能直接访问这些写锁存器，但是可以通过对 EEDATH:EEDATL 寄存器对的连续写入来加载写锁存器的内容。

数据写锁存器数并不等于行单元数，例如该芯片的程序存储器一行有 64 个单元，但只有 1 个写锁存器。编程时，用户软件需要填充该组写锁存器并多次启动编程操作，才能完全重新编程擦除的行 (页)。例如，具有 64 字的行大小和 1 个写锁存器的器件需要将数据装入写锁存器并启动编程操作 64 次。

注：如果用户只想修改部分以前编程的行，那么必须读取整行的内容并保存在 RAM 中，然后进行擦除。

1.4.1. 读闪存程序存储器

要读程序存储单元，用户必须：

1) 将最低有效地址位和最高有效地址位写入 EEADRH:EEADRL 寄存器对
2) 将 EECON1 寄存器的 CFGS 位清零
3) 将 EECON1 寄存器的 EEPGD 控制位置 1
4) 然后，将 EECON1 寄存器的控制位 RD 置 1

一旦将读控制位置 1，闪存程序存储器控制器将使用第二个指令周期读取数据。这会导致紧跟“BSR EECON1,RD”指令后的第二条指令被忽略。在紧接着的下一个周期，EEDATH:EEDATL 寄存器对中就有数据了，因此可在随后的指令中将该数据读作两个字节。

EEDATH:EEDATL 寄存器对将把此值保存至下一次读操作或用户向该单元写入数据时为止。

注：

1. 要求程序存储器读操作后的两条指令为 NOP。这可以防止用户在 RD 位置 1 后的下一条指令执行双周期指令；
2. 不管 CP 位的设置如何，软件都可以读取闪存程序存储器；

读程序存储器，示例：

```
* This code block will read 1 word of program
* memory at the memory address: PROG_ADDR_HI: PROG_ADDR_LO
* data will be returned in the variables: PROG_DATA_HI, PROG_DATA_LO

BANKSEL EEADRL                  ; Select Bank for EEPROM
registers
LDWI PROG_ADDR_LO               ;
STR    EEADRL                   ; Store LSB of address

LDWI PROG_ADDR_HI               ;
STR    EEADRH                   ; Store MSB of address

BCR  EECON1,CFGS                ; Do not select Configuration
Space
BSR  EECON1,EEPGD               ; Select Program Memory

BCR INTCON,GIE                  ; Disable interrupts

BSR EECON1,RD                   ; Initiate read

NOP                             ; Executed(Figure 8.3.1)
NOP                             ; Ignored(Figure 8.3.1)

BSR INTCON,GIE                  ; Restore interrupts

LDR EEDATL,W                    ; Get LSB of word
STR  PROG_DATA_LO               ; Store in user location

LDR EEDATH,W                    ; Get MSB of word
STR  PROG_DATA_HI               ; Store in user location
```

1.4.2. 擦除闪存程序存储器

当执行代码时，程序存储器只能按行擦除。要擦除行：

1) 将要擦除的新行地址装入 EEADRH:EEADRL 寄存器对
2) 将 EECON1 寄存器的 CFGS 位清零
3) 将 EECON1 寄存器的 EEPGD、FREE 和 WREN 位置 1
4) 依次将 0x55 和 0xAA 写入 EECON2 (闪存编程解锁序列)
5) 将 EECON1 寄存器的控制位 WR 置 1，以开始擦除操作
6) 查询 EECON1 寄存器的 FREE 位，以确定行擦除何时结束

程序存储器的行擦除，示例

```
; This row erase routine assumes the following:
; 1. A valid address within the erase block is loaded in ADDRH:ADDRL
; 2. ADDRH and ADDRL are located in shared data memory 0x70 - 0x7F (common RAM)

BCR INTCON,GIE ; Disable ints so required sequences will execute properly

NOP
NOP

BTSC INTCON, GIE
GOTO $-1

BANKSEL EEADRL
LDR ADDRL,W     ; Load lower 8 bits of erase address boundary
STR EEADRL

LDR ADDRH,W     ; Load upper 6 bits of erase address boundary
STR EEADRH

BSR EECON1,EEPGD        ; Point to program memory
BCR EECON1,CFGS         ; Not configuration space
BSR EECON1,FREE         ; Specify an erase operation
BSR EECON1,WREN         ; Enable writes

LDWI 55h            ; Start of required sequence to initiate erase
STR EECON2          ; Write 55h
LDWI 0AAh           ;
STR EECON2          ; Write AAh

BSR EECON1,WR               ; Set WR bit to begin erase
NOP                 ; Any instructions here are ignored as processor
                    ; halts to begin erase sequence
NOP                 ; Processor will stop here and wait for erase complete.
                    ; after Erase processor continues with 3rd instruction
BCR EECON1,WREN         ; Disable writes

BSR INTCON,GIE  ; Enable interrupts
```

该程序流不能被中断

在“BSR EECON1,WR”指令后，处理器需要两个周期来设置擦除操作。用户必须在 WR 位置 1 后，执行两条 NOP 指令。处理器将暂停内部操作，通常为 2ms 擦除时间。这不是休眠模式，因为时钟和外设将继续运行。擦除周期后，处理器从 EECON1 写指令后的第三条指令继续操作。

1.4.3. 写闪存程序存储器

使用以下步骤编程程序存储器：

1) 装入要编程的字的起始地址

2) 将数据装入写锁存器

3) 启动编程操作

4) 重复第 1 至 3 步，直到写入所有数据

写入程序存储器之前，要写入的字必须已擦除或者以前未写入过。程序存储器一次只能擦除一行。启动写操作时不会自动擦除。

程序存储器一次只能写入一个字，请参见**图 1-2** (对带 1 个写锁存器的程序存储器进行字写操作)。程序存储器写操作完成时，写锁存器将复位为 0x3FFF。

应完成以下步骤，以装载写锁存器和编程程序存储块。可按以下步骤操作：装载数据到写锁存器，启动编程序列。需要特殊的解锁序列才能将数据装入写锁存器或启动闪存编程操作，不应中断此解锁序列。

1) 将 EECON1 寄存器的 EEPGD 和 WREN 位置 1

2) 将 EECON1 寄存器的 CFGS 位清零

3) 将要写入的单元地址装入 EEADRH:EEADRL 寄存器对

4) 依次将 0x55 和 0xAA 写入 EECON2，然后将 EECON1 寄存器的 WR 位置 1 (闪存编程解锁序列)

5) 等待约 2ms 时间，锁存器数据写入程序存储器

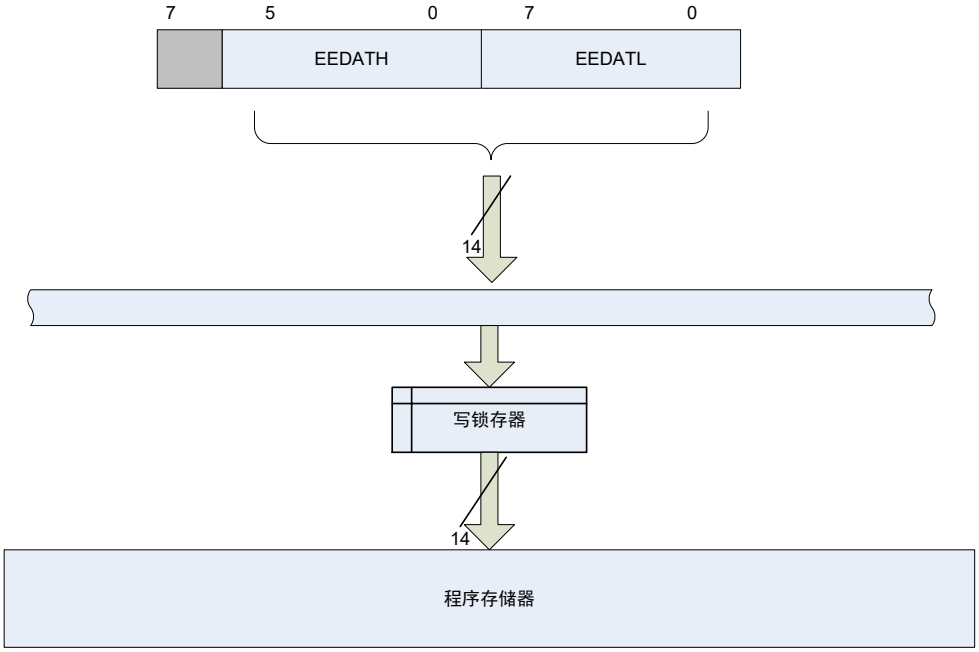

图 1-2 对带 1 个写锁存器的 PROM 进行写操作

注：示例中提供的代码序列必须重复多次，以完全编程擦除的程序存储器行。对于该系列芯片，一行 (页) 有 64 个 WORD，该代码序列需要重复 64 遍。

对包含 1 个写锁存器的 PROM 编程，示例：

```
; This write routine assumes the following:
; 1. The 2 bytes of data are loaded, starting at the address in DATA_ADDR
; 2. Each word of data to be written is made up of two adjacent bytes in DATA_ADDR, stored in little endian format
; 3. A valid starting address is loaded in ADDRH:ADDRL
; 4. ADDRH and ADDRL are located in shared data memory 0x70 - 0x7F (common RAM);
   BCR INTCON,GIE    ; Disable ints so required sequences will execute properly
   NOP
   NOP
   BTSC INTCON, GIE
   GOTO $-1
   BANKSEL EEADRH ; Bank 3
   LDR ADDRH,W       ; Load target address
   STR EEADRH        ;
   LDR ADDRL,W       ;
   STR EEADRL        ;
   LDWI LOW DATA_ADDR ; Load initial data address
   STR FSR0L ;
   LDWI HIGH DATA_ADDR ; Load initial data address
   STR FSR0H ;
   BSR EECON1,EEPGD ; Point to program memory
   BCR EECON1,CFGS              ; Not configuration space
   BSR EECON1,WREN              ; Enable writes

   MOVIW FSR0++      ; Load first data byte into lower
   STR EEDATL        ;
   MOVIW FSR0++      ; Load second data byte into upper
   STR EEDATH        ;

START_WRITE:

   LDWI 55h          ; Start of required write sequence:
   STR EECON2        ; Write 55h
   LDWI 0AAh         ;
   STR EECON2        ; Write AAh
   BSR EECON1,WR     ; Set WR bit to begin write
   NOP               ; Any instructions here are ignored as processor
                     ; halts to begin write sequence
   NOP               ; Processor will stop here and wait for write complete.
                     ; after write processor continues with 3rd instruction

   BCR EECON1,WREN              ; Disable writes
   BSR INTCON,GIE    ; Enable interrupts
```

该程序流不能被中断

1.5. 修改闪存程序存储器

修改程序存储器行中的现有数据，且该行内的数据需保留时，必须先读取它并将其保存在 RAM 映像中。

使用以下步骤修改程序存储器：

1) 装入要修改的行的起始地址
2) 从行中读取现有数据并将其保存到 RAM 映像中
3) 修改 RAM 映像以包含要写入到程序存储器的新数据
4) 装入要重新写入的行的起始地址
5) 擦除程序存储器行
6) 将来自 RAM 映像的数据装入写锁存器
7) 启动编程操作

根据需要重复第 6 至 7 步多次，以对擦除行进行重新编程。

1.6. 配置字 UCFGx/FCFGx 读访问

当 EECON1 寄存器中的 CFGS = 1 时，不是访问程序存储器或 EEPROM 数据存储器，而是访问配置字 UCFGx。这是 PC[15] = 1 时指向的区域，即当软件发起读操作时，地址是 [0x8000 + EADDR]，但不是所有的地址都可访问，对于未实现的单元，读返回未定义。

1.7. 写校验

根据具体应用，将写入数据 EEPROM 或者程序存储器中的值对照期望写入的值进行校验 (见如下例程) 是一个良好的编程习惯。

对数据 EEPROM 写校验，示例：

```
BANKSEL     EEDATL          ;
LDR         EEDATL, W       ;EEDATL not changed from previous write
BSR         EECON1, RD      ;YES, Read the value written
XORWR       EEDATL, W       ;
BTSS        STATUS, Z       ;Is data the same
LJUMP       WRITE_ERR       ;No, handle error
                            ;Yes, continue
```

1.8. PROM 内容保护

PROM 控制器内置加密保护单元，具备以下特性：

- 全区加密，由 CPB 位控制
- 分扇区加密，1 扇区=1k words，由 FSECPB0 寄存器控制
- 解除加密只能通过执行一次包含 UCFG 页在内的全芯片擦除

全加密和分扇区加密区别如下表：

加密方式	CPU 取指	软件读	软件写	串口读	串口写
无	√	√	√(2)	√	√
全区	√	√	√(2)	×(1)	×(4)
分扇区	√	×(1)	×(3)	×(3)	×(5)

注意：

1) EEDAT 保持旧值不变；
2) 软件不可以编程或擦除 UCFG 页；
3) 只可以读未加密的扇区；
4) 只允许串口做包括 UCFG 在内的全芯片擦除；
5) 只允许串口做包括 UCFG 在内的全芯片擦除，或者对未加密的扇区做页擦除，编程；

任何情况下，软件都不可以做全芯片擦除以及 FCFG 区的编程和擦除。

2. 应用范例

```
;=============================================================
/* 文件名：ASM_64F0Ax_EEPROM.ASM
* 功能：     FT64F0Ax_EEPROM 功能演示
*  IC：      FT64F0A5       TSSOP20
* 内部：     16M/2T
* 说明：     此演示程序位 64F0Ax_EEPROM 的演示程序.
*            该程序读取 0x12 地址的值,取反后存入 0x13 地址
*
*            FT64F0A5   TSSOP20
*                  ------------------
* NC----------|1(PA5)         (PA4)20|----------NC
* NC----------|2(PA6)         (PA3)19|----------NC
* NC----------|3(PA7)         (PA2)18|----------NC
* NC----------|4(PC0)         (PA1)17|----------NC
* NC----------|5(PC1)         (PA0)16|----------NC
* NC----------|6(PB7)         (PB0)15|----------NC
* GND---------|7(GND)         (PB1)14|----------NC
* NC----------|8(PB6)         (PB2)13|----------NC
* VDD---------|9(VDD)         (PB3)12|----------NC
* NC----------|10(PB5)        (PB4)11|----------NC
*                  ------------------
*/
;=============================================================
#INCLUDE   <FT64F0AX.INC>;
;=============================================================
;RAM DEFINE
;=============================================================
    TEMP1           EQU         0X41
    TEMP2           EQU         0X42

    W_TMP           EQU         0X43
    S_TMP           EQU         0X44

    EEDATTEMP       EQU         0X45
    EEADRTEMP       EQU         0X46
;=============================================================
;CONSTANT DEFINE
;=============================================================
    INTCON_DEF      EQU         B'00000000' ;禁止所有中断
    OSCCON_DEF      EQU         B'01110001' ;16MHz,1:1

    WPUA_DEF        EQU         B'00000000' ;弱上拉的开关，0-关，1-开
    WPUB_DEF        EQU         B'00000000'
```

```
    WPUC_DEF    EQU     B'00000000'

    WPDA_DEF    EQU     B'00000000' ;弱下拉的开关，0-关，1-开
    WPDB_DEF    EQU     B'00000000'
    WPDC_DEF    EQU     B'00000000'

    TRISA_DEF   EQU     B'00000000' ;输入输出设置，0-输出，1-输入
    TRISB_DEF   EQU     B'00000000'
    TRISC_DEF   EQU     B'00000000'

    PSRC0_DEF   EQU     B'11111111'  ;源电流设置最大
    PSRC1_DEF   EQU     B'11111111'
    PSRC2_DEF   EQU     B'00001111'

    PSINK0_DEF  EQU     B'11111111'  ;灌电流设置最大
    PSINK1_DEF  EQU     B'11111111'
    PSINK2_DEF  EQU     B'00000011'

    ANSELA_DEF  EQU     B'00000000' ;设置对应的 IO 为数字 IO
;============================================================
;PROGRAM  START
;============================================================
    ORG         0x0000
    LJUMP       RESTART
    ORG         0x0004
    STR         W_TMP
    SWAPR       STATUS,W
    STR         S_TMP
    LJUMP       INT_PROGRAM
;============================================================
;SYSTEM    START
;============================================================
RESTART:
    LCALL       DELAY_10MS
    LCALL       INITIAL
MAIN:
    BANKSEL     EEADRTEMP
    LDWI        0X12
    STR         EEADRTEMP
    LCALL       EEPROM_READ
    BANKSEL     EEADRTEMP
    LDWI        0X13
    STR         EEADRTEMP
    COMR        EEDATTEMP,R
```

```
    LCALL       EEPROM_WRITE
MAIN_LOOP:
    NOP
    LJUMP       MAIN_LOOP
;=========================================================
;INT_PROGRAM
;=========================================================
INT_PROGRAM:
    SWAPR       S_TMP,0
    STR         STATUS
    SWAPR       W_TMP,1
    SWAPR       W_TMP,0
    RETI
;=========================================================
;SYSTEM    INITIAL
;=========================================================
INITIAL:
    BANKSEL     OSCCON
    LDWI        OSCCON_DEF
    STR         OSCCON

    BANKSEL     INTCON
    LDWI        INTCON_DEF
    STR         INTCON

    BANKSEL     PORTA
    LDWI        0X00
    STR         PORTA
    STR         PORTB
    STR         PORTC

    BANKSEL     TRISA
    LDWI        TRISA_DEF
    STR         TRISA
    LDWI        TRISB_DEF
    STR         TRISB
    LDWI        TRISC_DEF
    STR         TRISC

    BANKSEL     WPUA
    LDWI        WPUA_DEF
    STR         WPUA
    LDWI        WPUB_DEF
    STR         WPUB
```

```
    LDWI        WPUC_DEF
    STR         WPUC

    BANKSEL     WPDA
    LDWI        WPDA_DEF
    STR         WPDA
    LDWI        WPDB_DEF
    STR         WPDB
    LDWI        WPDC_DEF
    STR         WPDC

    BANKSEL     PSRC0
    LDWI        PSRC0_DEF
    STR         PSRC0
    LDWI        PSRC1_DEF
    STR         PSRC1
    LDWI        PSRC2_DEF
    STR         PSRC2

    BANKSEL     PSINK0
    LDWI        PSINK0_DEF
    STR         PSINK0
    LDWI        PSINK1_DEF
    STR         PSINK1
    LDWI        PSINK2_DEF
    STR         PSINK2

    BANKSEL     ANSELA
    LDWI        ANSELA_DEF
    STR         ANSELA
;***************Clear  SRAM**********************
    BANKSEL     PORTA
    LDWI        0X00
    STR         FSR0H
 CLEAR_RAM_BANK0:
    LDWI        20H
    STR         FSR0L
CLEAR_RAM_BANK0_LOOP:
    CLRR        INDF0
    INCR        FSR0L,F
    LDWI        80H
    XORWR       FSR0L,W
    BTSS        STATUS,Z
    LJUMP       CLEAR_RAM_BANK0_LOOP
```

```
CLEAR_RAM_BANK1:
    LDWI        0A0H
    STR         FSR0L
CLEAR_RAM_BANK1_LOOP:
    CLRR        INDF0
    INCR        FSR0L,F
    LDWI        00H
    XORWR       FSR0L,W
    BTSS        STATUS,Z
    LJUMP       CLEAR_RAM_BANK1_LOOP
    INCR        FSR0H,F
CLEAR_RAM_LOOP:
    LDWI        10
    SUBWR       FSR0H,W
    BTSS        STATUS,0
    LJUMP       CLEAR_RAM_BANK0
    RET
;=============================================================================
;UNLOCK_FLASH
;=============================================================================
UNLOCK_FLASH:
    MOVLW 0x03
    MOVWF BSREG
    MOVLW 0x55
    MOVWF EECON2 & 0x7F
    MOVLW 0xAA
    MOVWF EECON2 & 0x7F
    BSF     EECON1 & 0x7F,1    //WR=1;
    NOP
    NOP
    RET
;============================================================
;EEPROM_READ
;============================================================
EEPROM_READ:
    BANKSEL     INTCON
    BTSS        INTCON,GIE
    LJUMP       $+5
    BCR         INTCON,GIE
    NOP
    NOP
    LJUMP       $-5

    BANKSEL     EEADRTEMP
```

```
    LDR         EEADRTEMP,W
    BANKSEL     EEADRL
    STR         EEADRL
    BCR         EECON1,CFGS
    BCR         EECON1,EEPGD
    BSR         EECON1,RD
    NOP
    NOP
    NOP
    NOP
    LDR         EEDATL,W
    BANKSEL     EEDATTEMP
    STR         EEDATTEMP
    RET
;=============================================================
;EEPROM_WRITE
;=============================================================
EEPROM_WRITE:
    BANKSEL     INTCON
    BTSS        INTCON,GIE
    LJUMP       $+5
    BCR         INTCON,GIE
    NOP
    NOP
    LJUMP       $-5

    BANKSEL     EEADRTEMP
    LDR         EEADRTEMP,W
    BANKSEL     EEADRL
    STR         EEADRL
    BANKSEL     EEDATTEMP
    LDR         EEDATTEMP,W
    BANKSEL     EEDATL
    STR         EEDATL
    BANKSEL     EECON1
    BCR         EECON1,EEPGD
    BCR         EECON1,CFGS
    BSR         EECON1,WREN

    BANKSEL     INTCON
    BCR         INTCON,EEIF

    LCALL       UNLOCK_FLASH
    NOP
```

```
    NOP
    NOP
    NOP

    BANKSEL     INTCON
    BSR         INTCON,GIE
    BANKSEL     EECON1
    BCR         EECON1,WREN
    BTSC        EECON1,WR
    LJUMP       $-2

    RET
;=============================================================
;DELAY_10MS(16M/2T）
;=============================================================
DELAY_10MS:
    LDWI        H'C8'
    STR         TEMP1
DELAY_10MS_LOOP1:
    LDWI        H'64'
    STR         TEMP2
 DELAY_10MS_LOOP2:
    CLRWDT
    DECRSZ      TEMP2,F
    LJUMP       DELAY_10MS_LOOP2
    DECRSZ      TEMP1,F
    LJUMP       DELAY_10MS_LOOP1
    RET

    END
```

联系信息

Fremont Micro Devices (SZ) Corporation

#5−8, 10/F, Changhong Building
Ke-Ji Nan 12 Road, Nanshan District,
Shenzhen, Guangdong, PRC 518057

Tel: (+86 755) 8611 7811
Fax: (+86 755) 8611 7810

Fremont Micro Devices (HK) Corporation

#16, 16/F, Block B, Veristrong Industrial Centre,
34−36 Au Pui Wan Street, Fotan, Shatin, Hong Kong SAR

Tel: (+852) 2781 1186
Fax: (+852) 2781 1144

http://www.fremontmicro.com/

* Information furnished is believed to be accurate and reliable. However, Fremont Micro Devices (SZ) Corporation assumes no responsibility for the consequences of use of such information or for any infringement of patents of other rights of third parties, which may result from its use. No license is granted by implication or otherwise under any patent rights of Fremont Micro Devices (SZ) Corporation. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. Fremont Micro Devices (SZ) Corporation products are not authorized for use as critical components in life support devices or systems without express written approval of Fremont Micro Devices (SZ) Corporation. The FMD logo is a registered trademark of Fremont Micro Devices (SZ) Corporation. All other names are the property of their respective owners.