FT61F13X

EEPROM Application note

目录

FT61F13x EEPROM 应用

1. 数据 EEPROM(DATA EEPROM)

FT61F13x 片内集成有 128 x 8−bit 的非易失性 DATA EEPROM 存储区，并独立于主程序区。此数据存储区的典型擦写次数可达 100 万次。可通过指令进行读/写访问，每次可读取或写入的单位为 1 个 byte (8−bit)，没有页模式(page mode)。擦除/编程实现了硬件自定时，无需软件查询，以节省有限的代码空间。因此写操作可在后台运行，不影响 CPU 执行其他指令，甚至可进入 SLEEP 状态。

读操作需要 2 个指令时钟周期，而写操作需要的时间为 $T_{WRITE-EEPROM}$ (使能自动擦除为 2 ~ 4 ms，关闭自动擦除则为 0.7 ~ 1.3 ms)。芯片内置有电荷泵，因此不需要提供外部高压，即可对 EEPROM 区进行擦除和编程。写操作完成时将置位相应的中断标志位 EEIF。

不支持连续读(sequential READ) 或连续写(sequential WRITE), 因此每次读/写都必须更新相应的地址。

只要 $V_{DD} \geq V_{POR}$, CPU 即可在 8 MHz / 2T 的速度下运行，在高温下甚至可低至 1.5V 左右。而写 DATA EEPROM 所需的电压($V_{DD-WRITE}$) 较高。工业级和汽车 1 级的最低 $V_{DD-WRITE}$ 分别为 1.9V 和 2.2V。读 DATA EEPROM 没有此最低电压限制(参阅 $V_{DD-READ}$)。

1.1. DATA EEPROM 相关寄存器汇总

名称	状态	寄存器	地址	复位值
EEDAT	DATA EEPROM 数据	EEDAT[7:0]	0x9A	RW−0000 0000
EEADR	DATA EEPROM 地址	EEADR[7:0]	0x9B	RW−0000 0000
WREN3	DATA EEPROM 写使能 (bit 3) 111 = 使能，完成后重置为 000 (其他) = 关闭	EECON1[5]	0x9C	RW−0
WREN2	DATA EEPROM 写使能 (bit 2)	EECON1[4]		RW−0
WRERR	DATA EEPROM 写错误标志位 1 = 中止 (发生 MCLR 或 WDT 复位) 0 = 正常完成	EECON1[3]		RW−x
WREN1	DATA EEPROM 写使能 (bit 1)	EECON1[2]		RW−0
PONLY	DATA EEPROM 自动擦除 1 = No (不擦除，只写) 0 = Yes (先擦除，再写)	EECON1[1]		RW−0
RD	DATA EEPROM 读控制位 1 = Yes (保持 4 个 SysClk 周期，然后 = 0) 0 = No	EECON1[0]		RW−0
WR	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	EECON2[0]	0x9D	RW−0

表 1-1 EEPROM 相关用户控制寄存器

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]	0x0B 0x8B 0x10B 0x18B	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)	PIE1[7]	0x8C	RW-0
EEIF	EEPROM 写完成中断标志位	1 = Yes (锁存) 0 = No	PIR1[7]	0x0C	RW-0

表 1-2 EEPROM 中断使能和状态位

1.2. 写 DATA EEPROM

1. 设置 “GIE = 0”；
2. 判断 GIE，如果 “GIE = 1”, 则重复步骤 (1)；
3. 往 EEADR 写入目标地址；
4. 往 EEDAT 写入目标数据；
5. 设置 “WREN3, WREN2, WREN1” = “1, 1, 1”，并在整个编程过程中保持此设置；
6. 须立即设置 “WR = 1” 以启动写 (否则将中止)；
7. 编程完成 (编程时间请参阅 $T_{WRITE-EEPROM}$) 后，”WR” 和 “WREN3, WREN2, WREN1” 都将自动清 0；

示例程序：

```
BCR INTCON, GIE
BTSC INTCON, GIE
LJUMP $-2
BANKSEL EEADR
LDWI 55H
STR EEADR              ; 地址为 0x55
STR EEDAT              ; 数据为 0x55
LDWI 34H
STR EECON1             ; WREN3/2/1 同时置 1
BSR EECON2, 0          ; 启动写
BSR INTCON, GIE        ; GIE 置 1
```

注：

1. 当编程正在进行中时，对 Data EEPROM 进行读操作将导致读取结果错误。
2. 如果编程完成前，WREN3, WREN2 或 WREN1 任意一位被清 0，在下次编程前需清除 EEIF 标志位。

1.3. 读 DATA EEPROM

将目标地址写入 EEADR 寄存器，然后启动读 (“RD = 1”)。2 个指令时钟周期后，EEPROM 数据被写入 EEDAT 寄存器，因而必须在读指令之后紧跟一条 NOP 指令。EEDAT 寄存器将保持此值直至下一次读或写操作。

读 DATA EEPROM 的示例程序如下：

```
BANKSEL EEADR
LDWI dest_addr
STR EEADR
BSR EECON1, RD
NOP                 ; 读等待
LDR EEDAT, W        ; 此时，数据可由指令读取
```

1.4. 自动擦除功能

将数据写入字节(byte)的过程包括 2 步：先擦除字节，再编程字节。擦除操作将字节的所有 bits 擦成“1”，而编程操作会有选择地将个别 bits 写成“0”。本芯片内置自动擦除功能(设置 PONLY = 0)，即编程前会先自动执行擦除操作。除高温环境外，建议使能自动擦除功能。

如果使能自动擦除，多次编程 FF 数据实际为多次擦除相应字节。然而多次编程非 FF 数据实际只对相应字节进行了一次编程，因为每次编程前都会先自动擦除。只有当自动擦除功能关闭时，重复编程才会有累积效应。某些情况下，比如在非常高的温度下，可能会需要关闭自动擦除功能，并进行重复编程以确保编程成功。流程如下：

1. 确保自动擦除使能。
2. 擦除字节。
3. 读 DATA EEPROM。
4. 如果字节数据为 FF 则继续，否则返回步骤(2)。
5. 再执行相同次数的步骤(2)即擦除操作，以确保擦除强度。
6. 关闭自动擦除。
7. 编程期望值。
8. 读 DATA EEPROM。
9. 如果字节数据为期望值则继续，否则返回步骤(7)。

再执行相同次数的步骤(7)即累积编程，以确保编程强度。

2. 应用范例

```
// Project:     test61F13x_EEPROM.c
// Device:      FT61F13X
// Memory:    Flash 3Kx14b, EEPROM 128x8b, SRAM 256x8b
// Company:
// Version:
/*  文件名：  test61F13x_EEPROM.c
*  功能：      FT61F13x-EEPROM 功能演示
*    IC:          FT61F135 SOP20
*  晶振：      16M/2T
*  说明：      此演示程序为 61F13x ERROM 的演示程序.
*               把 0x55 写入地址 0x13,再读出该值
*
*                   FT61F135    SOP20
*                        ----------------
*    VDD----------|1(VDD)    (VSS)20|----------VSS
*    NC-----------|2(PC1)     (PA0)19|----------NC
*    NC-----------|3(PC0)     (PA1)18|----------NC
*    NC-----------|4(PB7)     (PA2)17|----------NC
*    NC-----------|5(PB6)     (PA3)16|----------NC
*    NC-----------|6(PB5)     (PA4)15|----------NC
*    NC-----------|7(PB4)     (PA5)14|----------NC
*    NC-----------|8(PB3)     (PA6)13|----------NC
*    NC-----------|9(PB2)     (PA7)12|----------NC
*    NC-----------|10(PB1)    (PB0)11|----------NC
*                        ----------------
*/
//============================================================
#include "SYSCFG.h";
//============================================================
//Variable definition
//============================================================
#define   unchar         unsigned char
unchar EEReadData=0;
/*---------------------------------------------
 *   函数名：POWER_INITIAL
 *   功能：    上电系统初始化
 *   输入：
 *   输出：    无
 ---------------------------------------------*/
void POWER_INITIAL (void)
{
     OSCCON = 0B01110001;       //IRCF=111=16MHz/2T=8MHz，0.125μs
     INTCON = 0;                //暂禁止所有中断
```

```
    PORTA   = 0B00000000;
    TRISA   = 0B00000000;       //PA 输入输出  0-输出  1-输入
    PORTC   = 0B00000000;
    TRISC   = 0B00000000;       //PC 输入输出  0-输出  1-输入

    WPUA    = 0B00000000;       //PA 端口上拉控制  1-开上拉  0-关上拉
    WPUC    = 0B00000000;       //PC 端口上拉控制  1-开上拉  0-关上拉

    OPTION = 0B00001000;        //Bit3=1 WDT MODE
                                //Bit[2:0]=000 WDT RATE 1:1
}
//=============================================================
//函数名称：interrupt ISR
//输入：      无
//输出：      无
//=============================================================
void interrupt ISR(void)
{
}
/*-------------------------------------------------
 *   函数名称：EEPROMread
 *   功能：      读 EEPROM 数据
 *   输入：      EEAddr  需读取数据的地址
 *   输出：      ReEEPROMread  对应地址读出的数据
 -------------------------------------------------*/
unchar EEPROMread(unchar EEAddr)
{
    unchar ReEEPROMread;

    EEADR = EEAddr;
    RD = 1;
    ReEEPROMread = EEDAT;       //EEPROM 的读数据  ReEEPROMread = EEDATA;
    return ReEEPROMread;
}
/*-------------------------------------------------
 *   函数名称：EEPROMwrite
 *   功能：      写数据到 EEPROM
 *   输入：      EEAddr  需要写入数据的地址
 *               Data  需要写入的数据
 *   输出：      无
 -------------------------------------------------*/
void EEPROMwrite(unchar EEAddr,unchar Data)
{
```

```
    GIE = 0;                        //写数据必须关闭中断
    while(GIE);                     //等待 GIE 为 0
    EEADR = EEAddr;                 //EEPROM 的地址
    EEDAT = Data;                   //EEPROM 的写数据   EEDATA = Data
    EEIF = 0;                       //清 0 中断标志位
    EECON1 |= 0x34;                 //置位 WREN1，WREN2，WREN3 三个变量
    WR = 1;                         //置位 WR 启动编程
    while(WR);                      //等待 EE 写入完成
    GIE = 1;
}
//=============================================================
//函数名称：main
//功能：    主程序
//输入：    无
//输出：    无
//=============================================================
main()
{
    POWER_INITIAL();                //系统初始化

    EEPROMwrite(0x13,0x55);         //0x55 写入地址 0x13
    EEReadData = EEPROMread(0x13); //读取 0x13 地址 EEPROM 值
    while(1)
    {
        EEReadData=0;
    }
}
```

联系信息

Fremont Micro Devices 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) Limited

#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