

KF8F2320——DATA EEPROM 样例程序

引言

本应用笔记提供了 KF8F3132—DATA EEPROM 相关的配置信息以及如何能够快速的理解并上手使用该模块的一些配置方式。

本应用笔记须与 KF8F3132 数据手册结合使用。

寄存器

寄存器使用说明：（此文中 EE 代指 DATA EEPROM）

OSCCTL（系统控制寄存器）

寄存器OSCCTL：系统频率控制寄存器(地址:2FH)

| 复位值 0011 ---- | bit7 | | | | | | | bit0 |
|------------------|-------|-------|-------|-------|---|---|---|------|
| | CLKOE | IRCS2 | IRCS1 | IRCS0 | - | - | - | - |
| | R/W | R/W | R/W | R/W | U | U | U | U |

图注：R = 可读 W = 可写 P = 可编程 U = 未使用
 - = 读为0 x = 状态未知

TR0（P0 方向控制寄存器）

寄存器TR0：P0口方向控制寄存器(地址: 25H)

| 复位值 -11 1111 | bit7 | | | | | | | bit0 |
|-----------------|------|---|------|------|------|------|------|------|
| | - | - | TR05 | TR04 | TR03 | TR02 | TR01 | TR00 |
| | U | U | R/W | R/W | R/W | R/W | R/W | R/W |

TR1（P1 口方向控制寄存器）

TR1：P1口方向控制寄存器(地址: 27H)

| 复位值 -11 1111 | bit7 | | | | | | | bit0 |
|-----------------|------|---|------|------|------|------|------|------|
| | - | - | TR15 | TR14 | TR13 | TR12 | TR11 | TR10 |
| | U | U | R/W | R/W | R/W | R/W | R/W | R/W |

P1LR（P1 口输出锁存控制寄存器）

寄存器P1LR：P1口输出锁存寄存器(地址: 47H)

| 复位值 --xx xxxx | bit7 | | | | | | | bit0 |
|------------------|------|---|-------|-------|-------|-------|-------|-------|
| | - | - | P1LR5 | P1LR4 | P1LR3 | P1LR2 | P1LR1 | P1LR0 |
| | U | U | R/W | R/W | R/W | R/W | R/W | R/W |

NVMADDRL（存放 EE 的 8 位地址）

NVMDATAL（存放 EE 写入或读出的数据）

INTCTL（中断控制寄存器）

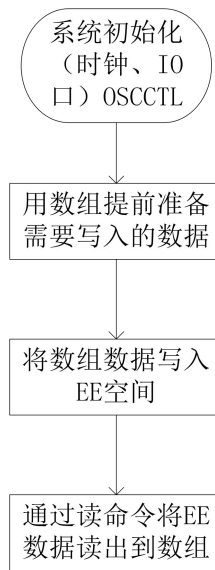
寄存器INTCTL：中断控制寄存器(地址: 0BH)

| 复位值 0000 0000 | bit7 | | | | | | | bit0 |
|------------------|------|------|------|--------|------|------|--------|------|
| | AIE | PUIE | TOIE | INTOIE | POIE | TOIF | INTOIF | POIF |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

NVMCTL0（EE 控制寄存器 0）：执行 EE 的读写命令

NVMCTL1（EE 控制寄存器 1）：执行 EE 的写命令

EE 读写样例程序框图



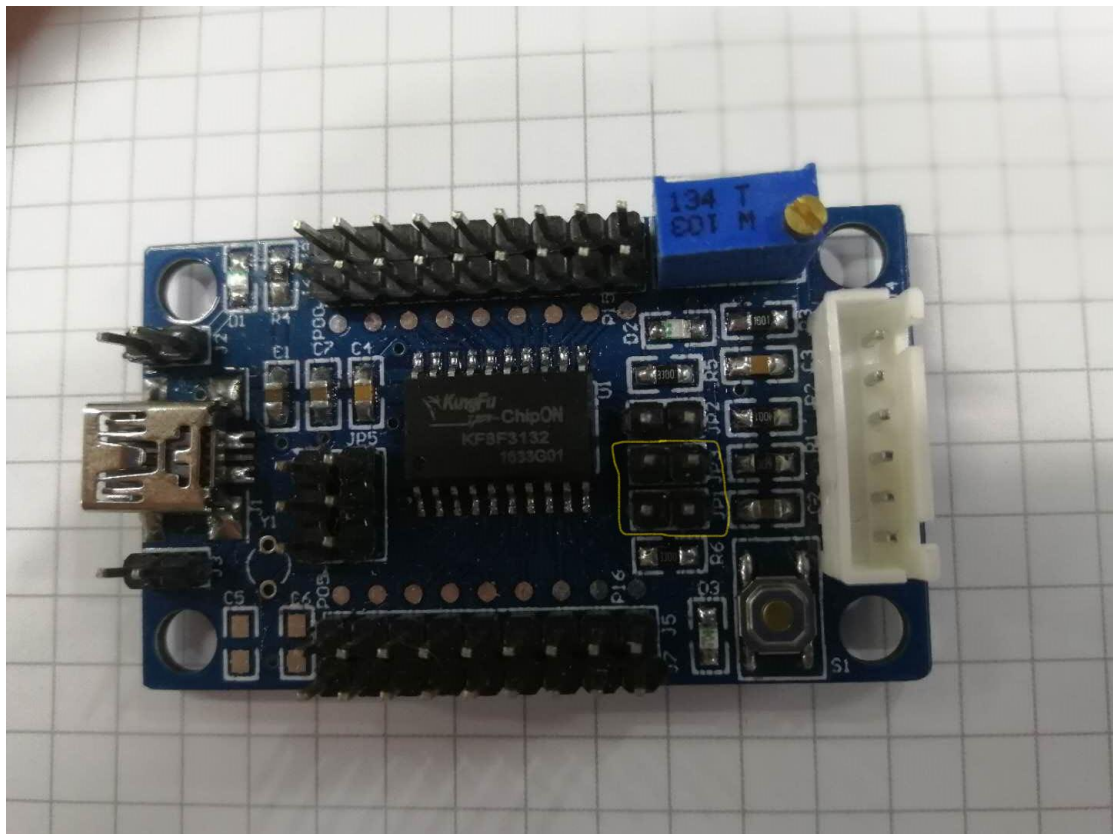
注：对于 EEPROM 的读和写操作有固定的程序步骤来执行，只需要直接使用即可，不需要用户自行编写读写程序，具体程序内容详见后面样例程序。

DATA EEPROM 读写样例简述：

开发环境：ChipON IDE

功能简述：对 EEPROM 的 0~63 空间进行多字节连续读写测试，完毕后 LED2 熄灭。程序等待按键按下。64~127 空间进行单字节读写测试，完毕后 LED3 熄灭。若中间有错误，则 LED2 点亮。至此程序结束，此时可用 PRO 将 EEPROM 空间全部读出来，验证写 EE 是否符合预期。

硬件连接：JP2 和 JP3 连接（图中黄色框中的插针接跳线帽）



EEPROM 样例程序:

MCU 初始化:

```
void init_mcu()
{
    OSCCTL = 0x70;           //设置为16M
    /*****端口初始化*****/
    TR03=1;    //将P03口设置为输入模式

    TR1 = 0;           //设置P1端口输出低电平
    P1LR = 0;
}
```

EEPROM 读:

```
unsigned char EE_READ_ONE(unsigned char address)
{
    unsigned char INTCTL_TEMP,OSCCTL_TEMP,DATA_buf;
    NVMADDR1 = address;
    INTCTL_TEMP = INTCTL;
    OSCCTL_TEMP = OSCCTL;

    INTCTL = 0X00;
    OSCCTL = 0X20;

    NVMCTL0 = 0X01;
    _NOP();
    DATA_buf = NVMDATAL;

    OSCCTL = OSCCTL_TEMP;
    INTCTL = INTCTL_TEMP;
    return DATA_buf;
}
```

EEPROM 写:

```
void EE_WRITE_ONE(unsigned char address,unsigned char value)
{
    unsigned char INTCTL_TEMP,OSCCTL_TEMP;
    NVMADDR1 = address;
    NVMDATAL = value;

    INTCTL_TEMP = INTCTL;
    OSCCTL_TEMP = OSCCTL;

    // EEIF = 0;
    INTCTL = 0X00;
    OSCCTL = 0X20;

    // ;;以下时序不可更改
    NVMCTL0 = 0X04;
    NVMCTL1 = 0X69;
    NVMCTL1 = 0X96;
    NVMCTL0 |= 0X02;
    _NOP();
    _NOP();
    _NOP();
    _NOP();
    _NOP();
    _NOP();
    _NOP();
    _NOP();
    _NOP();
    _NOP();
    _NOP();
    _NOP();
    NVMCTL0 = 0X00;

    while(!EEIF);
    EEIF = 0;
    OSCCTL = OSCCTL_TEMP;
    INTCTL = INTCTL_TEMP;
}
```

多字节连续读 EE:

```
void EE_REEAD_BUF(unsigned char address,unsigned char length)
{
    unsigned char i;
    if((length<33) &&(length>0))
    {
        for(i=0;i<32;i++)
        {
            if(i<length)
                Data_buf[i]= EE_READ_ONE(address+i);
            else
                Data_buf[i]=0;
        }
    }
}
```

多字节连续写 EE:

```
void EE_WRITE_BUF(unsigned char address,unsigned char length)
{
    unsigned char i;
    if((length<33) &&(length>0))
    {
        for(i=0;i<length;i++)
        {
            EE_WRITE_ONE((address+i),Data_buf[i]);
        }
    }
}
```

主函数:

```

void main()
{
    unsigned int i=0, k=0;
    init_mcu(); //点亮LED2和LED3

    for(i=0;i<32;i++) //将要写的内容提前填充到Data_buf数组
    {
        Data_buf[i]= i;
    }
    EE_WRITE_BUF(0,32); //将Data_buf内容写入EEPROM空间的0~31

    EE_REEAD_BUF(k,32);
    k=32;
    EE_WRITE_BUF(k,32); //将0~31的内容, 复制到32~63的空间

    P1LR2 =1; //熄灭LED2

    while(P03); //等待按键按下

    for(i=64;i<128;i++) //连续读写EEPROM混合测试
    {
        EE_WRITE_ONE(i,0X55);
        DATA_TEMP=EE_READ_ONE(i);
        if(0x55!=DATA_TEMP)
            P1LR2 =0; //若有错误, LED2点亮
    }
    P1LR1 =1; //LED3

    while(1)
    {

    }
}

```

注意事项：

1、对于 EEPROM 的读写操作需要使用固定的格式及固定的频率，用户直接使用即可。

2、对于字节的连续读写，只需要使用数组和循环语句来重复操作单字节的读写就可以达到目的。

3、当器件被代码保护时，器件编程器将不再能访问 DATA EEPROM。在代码保护时，CPU 仍可读写 DATA EEPROM。

4、CPU 写 DATA EEPROM 时，不管 DATAP 设置为何值，都能写入正确的数据，也可以在任何情况下读取数据。

5、EEIF 中断标志位必须使用软件清 0。