

文档版本	V1.0
发布日期	20210610

APT32F102 IFC 应用指南

APT'CHIP



目录

1 概述	1
2. 适用的硬件.....	1
3. 应用方案代码说明	1
3.1 IFC 配置.....	1
3.2 IFC 写数据	2
3.3 IFC 读数据	5
4. 程序下载和运行	7

1 概述

本文介绍了在APT32F102中使用IFC的应用范例。

2. 适用的硬件

该例程使用于 APT32F102x 系列学习板

3. 应用方案代码说明

3.1 IFC 配置

基于 APT32F102 完整的库文件系统，可以对 IFC 进行配置。

● 硬件配置：

APT32F102X 系列片上带有 64K/32K 字节的闪存(PROM),支持通过 ISP 来更新闪存。

由程序存储单元(PROM)，数据存储单元(DROM)，用户配置单元(User Option)，保护选项和客户信息区域构成。

2Kbytes 的独立数据闪存，可以让用户在掉电前存储一些程序数据。

最小的擦除和烧写单元为页空间，不能擦除或者烧写某个指定的字节(Word)。建议使用数据闪存空间来存放应用所需要存储的信息，而不是使用程序闪存。

- DROM 地址：0x10000000~0x100007FF

● 注意事项：

1. 写数据时，起始地址必须是 4 的倍数
2. 在同一页中写数据时，若起始地址并非该页的开始地址，则该地址前面的数据会被擦除掉
3. 写闪存操作不要使用并行模式

● 软件配置:

可在 apt32f102_initial.c 文件中进行初始化的配置;

```

/*****/
//ifc config
//EntryParameter:NONE
//ReturnValue:NONE
/*****/
void IFC_CONFIG(void)
{
    EnIFCClk;                //使能 IFC 时钟
    IFC->MR |= 0x10002;      //高速模式, 延迟 2 个周期
}
    
```

● 注意:

闪存控制器支持最大 16MHz 系统频率下的 0-wait 读取。当频率超过 16MHz 时, CPU 读取闪存时需要增加额外的等待周期

	WAIT	SPEED
24MHz < SYSCLK ≤ 48MHz	2	1
16MHz < SYSCLK ≤ 24MHz	1	1
SYSCLK ≤ 16MHz	0	0

图 3.1.1 等待周期

3.2 IFC 写数据

选择内部主频 48MHz 作为系统时钟, 在内部 DROM 地址 0x10000040 写入 8 个字节数据。

```

//一般模式, 调用一次完成, 会在程序中延时 4.2MS
void Page_ProgramData(unsigned int FlashAdd,unsigned int DataSize,volatile unsigned char *BufArry)
{
    int i,DataBuffer;
    //页缓存擦除 1
    SetUserKey;
    IFC->CMR=0x07;                //页缓存擦除
    IFC->FM_ADDR=FlashAdd;
    IFC->CR=0X01;                //Start Program
    while(IFC->CR!=0x0);          //等待操作完成
    //向写缓存中写数据 2
    for(i=0;i<((DataSize+3)/4);i++) //sizeof structure
    {
        DataBuffer=*BufArry+(*(BufArry+1)<<8)+*(BufArry+2)<<16)+*(BufArry+3)<<24);
    }
}
    
```

```

        *(volatile unsigned int *) (FlashAdd+4*i)=DataBuffer;

        BufArray +=4;
    }

    //预编程操作设定 3
    SetUserKey;
    IFC->CMR=0x06;                //预编程操作设定
    IFC->FM_ADDR=FlashAdd;
    IFC->CR=0X01;                //Start Program
    while(IFC->CR!=0x0);         //等待操作完成
    //执行预编程 4
    SetUserKey;
    IFC->CMR=0x01;                //执行预编程
    IFC->FM_ADDR=FlashAdd;        //
    IFC->CR=0X01;                //Start Program
    while(IFC->CR!=0x0);         //等待操作完成
    //页擦除 5
    SetUserKey;
    IFC->CMR=0x02;                //页擦除
    IFC->FM_ADDR=FlashAdd;        //
    IFC->CR=0X01;                //Start Program
    while(IFC->CR!=0x0);         //等待操作完成
    //将页缓存的数据写入闪存中 6
    SetUserKey;
    IFC->CMR=0x01;                //将页缓存的数据写入闪存中
    IFC->FM_ADDR=FlashAdd;        //
    IFC->CR=0X01;                //Start Program
    while(IFC->CR!=0x0);         //等待操作完成
}

/*****/
//ifc config
//EntryParameter:NONE
//ReturnValue:NONE
/*****/
void IFC_CONFIG(void)
{
    EnIFCCLK;                    //使能 IFC 时钟
    IFC->MR |= 0x10002;           //高速模式, 延迟 2 个周期
}

/*****/
/*****/
//APT32F102_init                /
//EntryParameter:NONE           /
//ReturnValue:NONE              /
/*****/

```

```

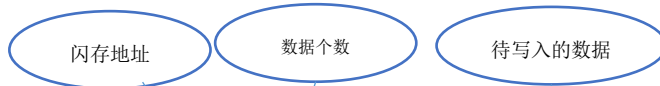
/*****/
/*****/
void APT32F102_init(void)
{
//-----/
//Peripheral clock enable and disable
//EntryParameter:NONE
//ReturnValue:NONE
//-----/
    SYSCON->PCER0=0xFFFFFFFF;           //PCLK Enable  0x00410071
    SYSCON->PCER1=0xFFFFFFFF;           //PCLK Enable
    while(!(SYSCON->PCSR0&0x1));         //Wait PCLK enabled
//-----/
//ISOSC/IMOSC/EMOSC/SYSCLK/IWDT/LVD/EM_CMFAIL/EM_CMRCV/CMD_ERR OSC stable interrupt
//EntryParameter:NONE
//ReturnValue:NONE
//-----/
    SYSCON_CONFIG();                   //syscon initial
    CK_CPU_EnAllNormalIrq();           //enable all IRQ
//-----/
//Other IP config
//-----/
    GPIO_CONFIG();                     //GPIO initial
    IFC_CONFIG();
}
unsigned char FlashDataArry0[8] = {1,2,3,4,5,6,7,8};
unsigned char DataBuffer0[2] = {0};
/*****/
//main
/*****/
int main(void)
{
    APT32F102_init();
    while(1)
    {
        Page_ProgramData(0x10000040,8,&FlashDataArry0[0]);
    }
}

```

● 代码说明:

Page_ProgramData(); ---- 用于写入数据。

● 函数参数说明:



```
Page_ProgramData(0x10000040,8,&FlashDataArry0[0]);
```

3.3 IFC 读数据

选择内部主频 48MHz 作为系统时钟，读取内部 DROM 地址 0x10000040 的数据验证。

```

/*****
//ReadFlashData fuction return Data arry save in Flash
//EntryParameter:RdStartAdd、 DataLength、 *DataArryPoint
//ReturnValue:NONE
*****/

void ReadDataArry_U8(unsigned int RdStartAdd,unsigned int DataLength,volatile unsigned char *DataArryPoint)
{
    unsigned int i;
    for (i=0;i<DataLength;i++)
    {
        if((i!=0)&&(i%4==0))
        {
            RdStartAdd +=4;
        }
        *DataArryPoint=(U8_T*)(RdStartAdd+ (i%4));
        DataArryPoint++;
    }
}

/*****
//ifc config
//EntryParameter:NONE
//ReturnValue:NONE
*****/

void IFC_CONFIG(void)
{
    EnIFCClk;                //使能 IFC 时钟
    IFC->MR |= 0x10002;       //高速模式，延迟 2 个周期
}

unsigned char FlashDataArry0[8] = {1,2,3,4,5,6,7,8};
unsigned char DataBuffer0[8] = {0};
unsigned char BitIFCflag ;

/*****

```

```

//main
/*****/

int main(void)
{
    APT32F102_init();
    while(1)
    {
        //SYSCON_IWDCNT_Reload();
        if(BitIFCflag)
        {
            BitIFCflag = 0;
            GPIO_Write_Low(GPIOA0,1);
            Page_ProgramData(0x10000040,8,&FlashDataArry0[0]);

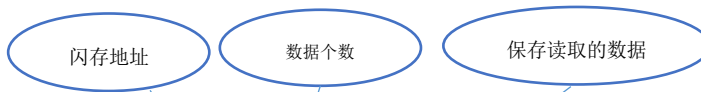
        }
        else
        {
            ReadDataArry_U8(0x10000040,8,&DataBuffer0[0]);
            if(DataBuffer0[0] == 1)
            {
                GPIO_Write_High(GPIOA0,1);
            }
        }
    }
}

```

● 代码说明:

ReadDataArry_U8(); ----用于读取保存在 Flash 中数据

● 函数参数说明:



ReadDataArry_U8(0x10000040,8,&DataBuffer0[0]);

● 数据验证:

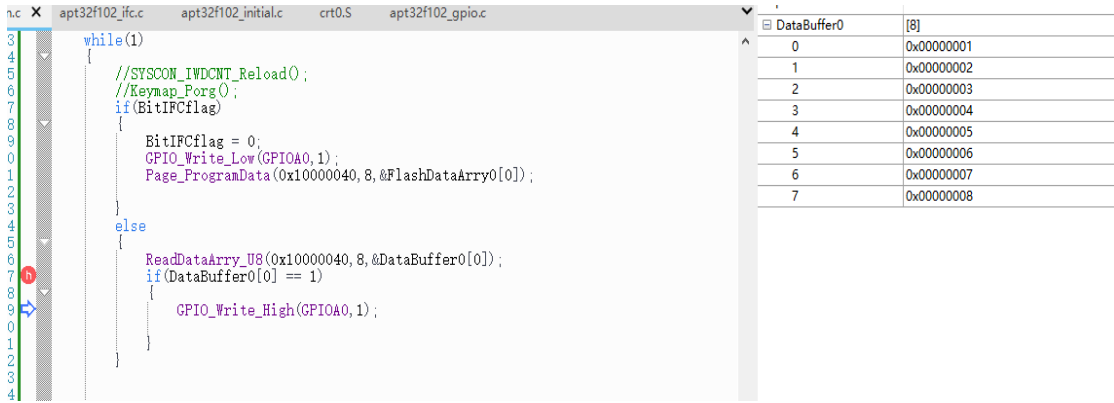


图 3.3.1 IFC 写入验证

4. 程序下载和运行

1. 将目标板与仿真器连接，分别为 VDD SCLK SWIO GND
2. 程序编译后仿真运行
3. 由图 3.3.1 得知 DataBuffer0 得到的就是保存在闪存中的数据