用 C 语言实现在应用中编程(IAP)例程

1.0 简介

这篇应用文档的目的是为软件设计者提供一份用 C 语言实现在应用中编程(IAP)的例程。我们为您提供了包括擦除、写入、验证三个基本 IAP 的操作以供参考。该 C 代码提供了独立的代码，不需要头文件，并且它可以用 Keil C51 进行正确的综合的。

2.0 软件描述

下面所提供的 IAP 为中心的程序，应该驻留在 SST 微控制器的物理地址的低地址位(Block0). 它列出了用到的 IAP 功能----擦除、写入、验证。每个子程序提供了相应的 IAP 功能，这些子程序通过主程序的调用来实现 IAP 的典型应用-----往另外一个 Block(Block1)写入数据。

在主程序中，目标单元首先被擦出，然后数据字节(0,1,2,…n,这里 n 等于单元大小)被写到 Block1 中目标地址 BLK1_DST_ADDR。在 IAP 功能成功实现以后，ErrorCode 是 0. 如果错误的条件出现，执行程序将会保留错误在错误的函数中, 并且 ErrorCode 是 1。

3.0 总结

在这个应用文档中，我们向用户展示了具有主要 IAP 功能的 C 语言例程作为参考，方便用户开发自己专门需求的基于 IAP 应用的代码。

```
/*****************************************************************************************************
* demoIAP.c - Sample IAP C51 code for SST89x564RD/SST89x554RC Devices
*
*
* This sample code provides programming routines using In-Application
* Programming (IAP).
* This sample code is for the user's reference only. SST does not
* guarantee the functionality or the usefulness of the sample code.
*
* In the SST89x564RD/SST89x554RC MCU, there are two blocks of flash
* memory. This code is to write data to block1 starting at 1000H
* (for the SST89x564RD) or E000H (for the SST89x554RC) by running IAP
*  operations from block 0. The default set up is for SST89x564RD.
*  When using SST89x554RC, please modify the global variable BLK1_DST_ADDR
*  following the instruction below the variable definition.
*
*  The companion SST89x564RD/SST89x554RC MCU data sheet should be
*  reviewed in conjunction with this sample code.
*
*  Note: This demo program is specifically for SST89x564RD/SST89x554RC devices.
*****************************************************************************************************/

/*****************************************************************************************************
*                       FlashFlex51 MCU SFR Memory Addresses
*****************************************************************************************************/
sfr SFCF = 0xB1;                /*SuperFlash Configuration*/
sfr  SFCM = 0xB2;               /*SuperFlash Command*/
sfr  SFAL = 0xB3;               /*SuperFlash Address Low*/
sfr  SFAH = 0xB4;               /*SuperFlash Address High*/
sfr  SFDT = 0xB5;               /*SuperFlash Data*/
sfr  SFST = 0xB6;               /*SuperFlash Status*/

/*****************************************************************************************************
*                       FlashFlex51 MCU IAP Commands
*****************************************************************************************************/
#define SFCM_SE   0x0B;       /*Sector-Erase IAP cmd*/
#define SFCM_VB   0x0C;       /*Byte-Verify  IAP cmd*/
#define SFCM_PB   0x0E;       /*Byte-Program IAP cmd*/

/*****************************************************************************************************
*                       Global Variable Definition
*****************************************************************************************************/
const unsigned short int BLK1_DST_ADDR = 0x1000;
/*SST89x564RD destination address (in the other on-chip flash memory block)
where data will be written to, which is above BSL code space.
Please comment out this line and uncomment the following line if SST89x554RC is used*/

/*const unsigned short int BLK1_DST_ADDR = 0x0F000; */
/*SST89x554RC destination address (in the other on-chip flash memory block)
where data will be written to, which is above BSL code space.
Please comment out this line and uncomment the previous line if SST89x564RD is used*/

const unsigned char SECT_SIZE = 0x80;     /*number of bytes in a sector*/

unsigned char ErrorCode;                  /*show the result of the operation*/
```

```
/*****************************************************************************************************
*                        Function Prototype
*****************************************************************************************************/
void sector_erase(unsigned short int dataAddr);
void byte_program(unsigned short int dataAddr, unsigned char dataByte);
unsigned char byte_verify(unsigned short int dataAddr);
int  ready();
void error();

/*****************************************************************************************************
*                        MAIN PROGRAM
* To program a sector of data bytes (starting from 0, increment by 1)
* into block 1, starting address is BLK1_DST_ADDR.
* When the IAP is completed successfully, ErrorCode is 0. Otherwise,
* ErrorCode is 1.
*****************************************************************************************************/
void main()
{
  unsigned short int destAddr = BLK1_DST_ADDR;
  unsigned char byteCnt ;        /*byte count*/
  unsigned char origData;        /*store the data byte for IAP operation*/
  unsigned char verifyData;      /*verify the data byte */


  sector_erase(destAddr);        /*erase sector area before writing there*/
  origData = 0;
  for(byteCnt=0; byteCnt<SECT_SIZE; byteCnt++)
 {
    byte_program(destAddr, origData);                  /*program a byte*/
                  verifyData=byte_verify(destAddr);    /*verify byte programmed correctly*/
                  if(verifyData!=origData)
                  error();                             /*go to error if programmed incorrectly*/
                  destAddr++;
                  origData++;
  }

  ErrorCode=0;                   /*IAP correct*/
  while(1)
  {}
}

/*****************************************************************************************************
*                        IAP SUBROUTINES
 * 1. Sector-Erase
 * 2. Byte-Program
 * 3. Byte-Verify
*****************************************************************************************************/

/*****************************************************************************************************
*                        Sector-Erase Subroutine
*****************************************************************************************************/
void sector_erase(unsigned short int dataAddr)
{
  unsigned short int destAddr = dataAddr;
  SFCF = SFCF | 0x40;            /*enable IAP */
  SFAH = destAddr>>8;            /*load high order address byte*/
  SFAL = destAddr;               /*load low order address byte */
  SFCM = SFCM_SE;                /*issue sector erase command  */

  if(!ready())
     error();
  return;
}
```

```
/*****************************************************************************************************
*                          Byte-Program Subroutine
*****************************************************************************************************/
void byte_program(unsigned short int dataAddr, unsigned char dataByte)
{
  unsigned short int destAddr = dataAddr;
  SFCF = SFCF | 0x40;          /*enable IAP */
  SFAH = destAddr>>8;          /*load high order address byte*/
  SFAL = destAddr;             /*load low order address byte */
  SFDT = dataByte;             /*load data to be programmed  */
  SFCM = SFCM_PB;              /*issue byte program command  */

  if(!ready())
     error();
  return;
}

/*****************************************************************************************************
*                          Byte-Verify Subroutine
*****************************************************************************************************/
unsigned char byte_verify(unsigned short int dataAddr)
{
  unsigned short int destAddr = dataAddr;
  unsigned char readByte;
  SFCF = SFCF | 0x40;          /*enable IAP */
  SFAH = destAddr>>8;          /*load high order address byte*/
  SFAL = destAddr;             /*load low order address byte */
  SFCM = SFCM_VB;              /*issue byte verify command   */
  readByte = SFDT;

  SFCF = SFCF & 0xBF;          /*turn off IAP*/
  SFDT = 0;
  return readByte;
}

/*****************************************************************************************************
*                          Ready Subroutine
*
* Purpose: To check if the IAP operation is completed.
* When it is done, turn off IAP configuration.
*****************************************************************************************************/
int ready()
{
  unsigned long int TimeOut = 0;

  while (TimeOut < 100000)
     {
                    if ((SFST&4) == 0)                  /* Check if IAP is done */
                      {                                 /* IAP is done */
                              SFCF = SFCF & 0xBF;       /* turn off IAP*/
                              SFDT = 0;                 /* any value other than 0x55 */
                      return 1;                         /* IAP operation is completed*/
                      }
                    TimeOut++;
     }

  SFCF = SFCF & 0xBF;          /*turn off IAP*/
  SFDT = 0;                    /*any value other than 0x55*/
  return 0;                    /*IAP operation is NOT completed before time out*/
}
```

```
/*******************************************************************************************************************
*                          Error Function
*******************************************************************************************************************/
void error()
{
  ErrorCode=1;                  /*IAP error*/
  while(1)                      /*software trap*/
  {}
}
```