FlashFlex Microcontroller
In-Application Programming Example Using C

Application Note
August 2008

1.0 INTRODUCTION

This application note is intended to provide the software designer an example of implementing In-Application Programming (IAP) using the C programming language. The three principal IAP operations of Erase, Write, and Read have been implemented for user reference. The C code is provided as stand-alone code, requiring no header files, and it will compile correctly with the Keil C51-compiler.

2.0 SOFTWARE DESCRIPTION

This sample IAP-centered program, shown in Appendix A, is physically located in the lower flash memory block (block 0) of SST microcontrollers. It demonstrates the usage of the IAP functions—Sector-Erase, Byte-Program, and Byte-Verify. A subroutine for each IAP function is given, and these subroutines are called from the main program to perform a typical use of IAP—program data into the memory of the other flash memory block (block 1).

In the main program, the destination sector is erased first, and then the data bytes (0,1,2,...n, where n equals the sector size) are written into the destination address BLK1_DST_ADDR in block 1. After the IAP is completed successfully, ErrorCode is 0. If an error condition occurred, the execution of the program will remain in the error function, and the ErrorCode is 1.

3.0 SUMMARY

In this application note, the user is shown C-language sample code of the principal IAP operations to use as a reference for developing their own specific IAP-based application code in C.

Application Note

APPENDIX A. SAMPLE CODE

```
/*****************************************************************************************************
* demoIAP.c - Sample IAP C51 code for SST89x516RDx/SST89x58RDx 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 SST89x516RDx/SST89x58RDx MCU, there are two blocks of flash
* memory. This code is to write data to block1 starting at 1000H
* (for the SST89x516RDx) or E000H (for the SST89x58RDx) by running IAP
*  operations from block 0. The default set up is for SST89x516RDx.
*  When using SST89x58RDx, please modify the global variable BLK1_DST_ADDR
*  following the instruction below the variable definition.
*
*  The companion SST89x516RDx/SST89x58RDx MCU data sheet should be
*  reviewed in conjunction with this sample code.
*
*  Note: This demo program is specifically for SST89x516RDx/SST89x58RDx devices.
*****************************************************************************************************/

/*****************************************************************************************************
*                       FlashFlex 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*/

/*****************************************************************************************************
*                       FlashFlex 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;
/*SST89x516RDx 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 SST89x58RDx is used*/

/*const unsigned short int BLK1_DST_ADDR = 0x0F000; */
/*SST89x58RDx 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 SST89x516RDx 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*/
  {}
}
```