AN201

WRITING TO FLASH FROM FIRMWARE

Relevant Devices

This application note applies to the following device families:
C8051F00x, C8051F01x, C8051F02x, C8051F04x, C8051F06x, C8051F12x-13x, C8051F2xx, C8051F30x, C8051F31x, C8051F32x, C8051F326/7, C8051F33x, C8051F34x, C8051F35x, C8051F36x, C8051F41x, and C8051F52x-53x

Table of Contents

1. Introduction 4
2. Key Points 4
3. Flash Essentials 5
3.1. Flash Organization 5
3.2. Device Specific Notes 6
3.2.1. C8051F12x Code Banking 6
3.2.2. C8051F3xx, C8051F4xx, and C8051F5xx Flash Unlock Bytes 6
3.2.3. Flash Timing 6
3.2.4. C8051F4xx and C8051F5xx VDD Monitor Level 6
3.3. Flash Read, Write, and Erase Operations 6
4. Basic Flash Operations 7
4.1. Reading A Byte 7
4.2. Writing A Byte 7
4.3. Erasing a Page 7
4.4. Example Code Implementation Notes 8
5. Advanced Flash Operations 9
6. Flash Write and Erase Guidelines 10
6.1. V_{DD} Maintenance and the V_{DD} Monitor 10
6.2. PSWE Maintenance 11
6.3. System Clock 12
7. Example Code 13
7.1. ‘F000 13
7.1.1. F000_FlashPrimitives.c 13
7.1.2. F000_FlashPrimitives.h 15
7.1.3. F000_FlashUtils.c 17
7.1.4. F000_FlashUtils.h 21
7.2. ‘F020 22
7.2.1. F020_FlashPrimitives.c 22
7.2.2. F020_FlashPrimitives.h 25
7.2.3. F020_FlashUtils.c 26
7.2.4. F020_FlashUtils.h 30

7.3. ‘F040 31
7.3.1. F040_FlashPrimitives.c 31
7.3.2. F040_FlashPrimitives.h 34
7.3.3. F040_FlashUtils.c 35
7.3.4. F040_FlashUtils.h 39
7.4. ‘F060 40
7.4.1. F060_FlashPrimitives.c 40
7.4.2. F060_FlashPrimitives.h 43
7.4.3. F060_FlashUtils.c 44
7.4.4. F060_FlashUtils.h 49
7.5. ‘F120 50
7.5.1. F120_FlashPrimitives.c 50
7.5.2. F120_FlashPrimitives.h 54
7.5.3. F120_FlashUtils.c 55
7.5.4. F120_FlashUtils.h 59
7.6. ‘F200 60
7.6.1. F200_FlashPrimitives.c 60
7.6.2. F200_FlashPrimitives.h 62
7.6.3. F200_FlashUtils.c 64
7.6.4. F200_FlashUtils.h 68
7.7. ‘F300 69
7.7.1. F300_FlashPrimitives.c 69
7.7.2. F300_FlashPrimitives.h 71
7.7.3. F300_FlashUtils.c 72
7.7.4. F300_FlashUtils.h 76
7.8. ‘F310 77
7.8.1. F310_FlashPrimitives.c 77
7.8.2. F310_FlashPrimitives.h 80
7.8.3. F310_FlashUtils.c 81
7.8.4. F310_FlashUtils.h 85
7.9. ‘F320 86
7.9.1. F320_FlashPrimitives.c 86
7.9.2. F320_FlashPrimitives.h 89
7.9.3. F320_FlashUtils.c 90
7.9.4. F320_FlashUtils.h 94
7.10. ‘F326/7 95
7.10.1. F326_FlashPrimitives.c 95
7.10.2. F326_FlashPrimitives.h 98
7.10.3. F326_FlashUtils.c 99
7.10.4. F326_FlashUtils.h 104
7.11. ‘F330 105
7.11.1. F330_FlashPrimitives.c 105
7.11.2. F330_FlashPrimitives.h 108
7.11.3. F330_FlashUtils.c 109
7.11.4. F330_FlashUtils.h 113

7.12. ‘F340 114
7.12.1. F340_FlashPrimitives.c 114
7.12.2. F340_FlashPrimitives.h 117
7.12.3. F340_FlashUtils.c 118
7.12.4. F340_FlashUtils.h 123
7.13. ‘F350 124
7.13.1. F350_FlashPrimitives.c 124
7.13.2. F350_FlashPrimitives.h 127
7.13.3. F350_FlashUtils.c 128
7.13.4. F330_FlashUtils.h 132
7.14. ‘F360 133
7.14.1. F360_FlashPrimitives.c 133
7.14.2. F360_FlashPrimitives.h 136
7.14.3. F360_FlashUtils.c 138
7.14.4. F360_FlashUtils.h 143
7.15. ‘F410 144
7.15.1. F410_FlashPrimitives.c 144
7.15.2. F410_FlashPrimitives.h 147
7.15.3. F410_FlashUtils.c 149
7.15.4. F410_FlashUtils.h 154
7.16. ‘F520 155
7.16.1. F520_FlashPrimitives.c 155
7.16.2. F520_FlashPrimitives.h 158
7.16.3. F520_FlashUtils.c 160
7.16.4. F520_FlashUtils.h 165
Document Change List 166
Contact Information 168

1. Introduction

The Flash memory on all Silicon Labs MCU devices is readable and writable from application code. This capability allows user software to store values to the Flash such as calibration constants or system parameters, and to implement a boot loading feature in which user firmware can be updated in-system from a remote site.

The Flash that is not used by application code can be treated like an EEPROM, thus negating the need to connect an external EEPROM to the device.

This document starts with the basics of accessing Flash from application code on any device, including device specific details. Then, it discusses advanced routines that can be developed using the basic routines. Finally, it describes precautions to take when writing to Flash.

Example code for the basic and advanced Flash access routines for all devices is included at the end of this application note.

2. Key Points

- It is strongly recommended that the V_{DD} monitor be enabled during Flash write and erase operations to prevent data corruption resulting from power irregularities or power-down conditions.
- Disable interrupts before setting PSWE to '1' to prevent interrupt service routines, which may access variables in xdata space, from generating MOVX writes which could corrupt Flash memory.
- Be cautious when using the 'Large' and 'Compact' memory models, which target xdata and pdata spaces for user variables, both of which generate MOVX write opcodes.
- In the C8051F3xx, C8051F4xx, and C8051F5xx series devices, a Lock and Key sequence must be executed before each Flash write or erase operation.
- In the C8051F3xx, C8051F4xx, and C8051F5xx series devices, attempts to read, write, or erase code memory locations located in RESERVED space will generate a device reset.
- Attempts to read, write, or erase code memory locations in RESERVED space will be ignored by the hardware on F0xx, F1xx, and F2xx devices.
- The CPU is stalled during Flash write and erase operations, although peripherals (UART, ADC, timers, etc.) remain active.
- Interrupts which are posted during a Flash write or erase operation will be held pending until the completion of the Flash operation, after which time they will be serviced in priority order.
- The Flash page containing the lock byte or bytes cannot be erased from application code.

3. Flash Essentials

The Flash in different device series has many similarities including page sizes, lock bits, and the instructions used to read and write to Flash. The main differences are the amount of Flash available, how the V_{DD} monitor is enabled, and how SFR registers are modified to allow Flash writes and erases.

Although the CPU is stalled during Flash write and erase operations, peripherals (UART, ADC, timers, etc.) remain active. Interrupts posted during a Flash write or erase operation are held until the Flash operation has completed, after which they are serviced in priority order.

3.1. Flash Organization

The Flash memory on most devices is organized into a set of 512-byte pages. See the Flash chapter of the device data sheet for specific information. As an example, Figure 1 shows the Flash organization for the C8051F30x series.

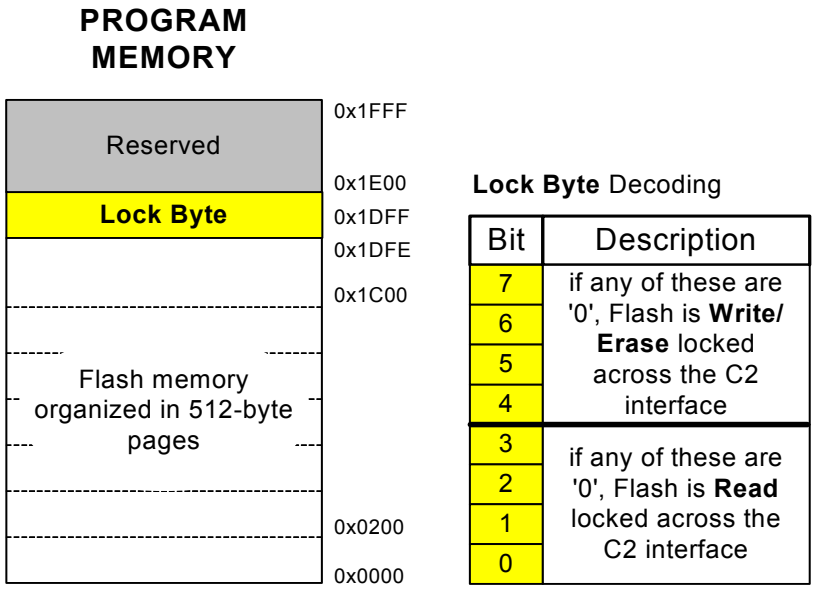

Figure 1. Flash Memory Organization and Security for the C8051F30x Series

Some devices also have a separate scratchpad area of Flash. This scratchpad area is ideal for storing constants and system parameters because of its smaller size.

3.2. Device Specific Notes

Various MCUs have features that require consideration when accessing Flash. These considerations are described below.

3.2.1. C8051F12x Code Banking

The C8051F12x family of devices have 128KB of Flash which is divided into 4 banks. Addresses in the range 0x00000 to 0x07FFF are mapped to first bank of Flash. Addresses in the range of 0x08000 to 0x0FFFF are mapped to one of the other three banks based on the settings of the PSBANK register. Whenever accessing Flash from the address range 0x08000 to 0x07FFF, precaution must be taken to select the correct bank using the PSBANK register.

3.2.2. C8051F3xx, C8051F4xx, and C8051F5xx Flash Unlock Bytes

All C8051F3xx, C8051F4xx, and C8051F5xx devices' writes and erases to Flash are protected with a lock and key function. The Flash Lock and Key Register (FLKEY) must be written with the correct key codes, in sequence, before Flash operations may be performed. The key codes are: 0xA5, 0xF1. The timing does not matter, but the codes must be written in order. If the key codes are written out of order, or the wrong codes are written, Flash writes and erases will be disabled until the next system reset. Flash writes and erases will also be disabled if a Flash write or erase is attempted before the key codes have been written properly. The Flash lock resets after each write or erase; the key codes must be written again before a following Flash operation can be performed.

3.2.3. Flash Timing

This applies to:

- C8051F0xx
- C8051F2xx
- C8051F34x
- C8051F35x
- C8051F36x
- C8051F41x

Some of the device families require a Flash timing register, FLSCL, to be set to correctly access Flash. The register value is based upon the SYSCLK speed. This register only needs to be set once. See the relevant data sheet for device specific information regarding the FLSCL register.

3.2.4. C8051F4xx and C8051F5xx VDD Monitor Level

The 'F41x and 'F52x-53x devices have two settings for the VDD monitor threshold - low and high. When writing or erasing Flash, the VDD monitor threshold should be set to the high setting. See the Reset Sources chapter in the device's data sheet for detailed information on enabling the high threshold.

3.3. Flash Read, Write, and Erase Operations

There are three basic operations that can be performed on the Flash: read, write, and erase. The basic read and write operations read or write one byte from Flash. The erase operation applies to a full page of Flash.

Flash read operations are accomplished by using the standard 8051 MOVC instruction (in the C language, MOVC instructions are generated by using pointers of memory type 'code'). Flash write and erase operations on Silicon Labs MCU devices are accomplished by using the MOVX instruction. The default target for MOVX write operations is external memory (XRAM); however, devices can be configured such that MOVX write operations target Flash memory instead. MOVX instructions are generated in C by using pointers of memory type xdata or pdata.

Flash erase operations occur on page boundaries. The erase operation sets all the bits in the Flash page to logic 1. Flash write operations, which set bits to logic 0, occur on single byte boundaries. To successfully complete a write to Flash, the target bytes must be erased to 0xFF because the write instruction can only clear bits in a byte.

MOVX write operations on all Silicon Labs MCU devices can target Flash by setting bits in the

PSCTL register. When the PSWE bit (PSCTL.0) is set to a logic 1, MOVX write opcodes target Flash memory instead of External Memory (XRAM). When both PSWE and PSEE (PSCTL.0 and PSCTL.1) are set to logic 1, MOVX write opcodes erase the Flash page containing the target address. The target address can be any address in the target page.

4. Basic Flash Operations

The basic procedure to do the three basic Flash operations is the same on all devices. Some devices will require setting additional registers to enable Flash operations. The pseudocode for the different operations is detailed below for all devices. Also included below the pseudocode are the exceptions for the various device families. The code that implements these routines for each device family is provided at the end of this application note in the files named Fxxx_Primitives.h and Fxxx_Primitives.c.

4.1. Reading A Byte

1. Disable interrupts. This is optional for most devices, but recommended for the 'F12x and 'F13x devices.
2. Read the byte.
3. Restore interrupts if originally enabled.

Exceptions:

1. Set the SFLE bit before reading if the target address is in the scratchpad area.
2. If the device is the 'F12x or 'F13x, check the address and set the correct bank in the PSBANK register.

4.2. Writing A Byte

1. Disable interrupts. This is optional for most devices, but recommended for the 'F12x and 'F13x devices.
2. Set PSWE to '1' by writing PSCTL = 0x01 (PSEE must be '0').
3. Confirm V_{DD} monitor is enabled.
4. Write the data to an erased byte.
5. Set PSWE and PSEE to '0' if no further erases are necessary.
6. Restore interrupts if originally enabled.

Exceptions:

1. Set the SFLE bit before writing if the target address is in the scratchpad are
2. If the device is a 'F3xx, 'F4xx, or 'F5xx device, write FLKEY = 0xA5 then FLKEY = 0xF1 before writing the byte.
3. If the device is the 'F12x or 'F13x, check the address and set the correct bank in the PSBANK register before writing the byte.
4. For all ‘F0xx, ‘F1xx, and 'F2xx devices, enable Flash writes in the FLSCL Register (FLSCL = 0x01) before writing the byte. Disable the bit after the byte has been written to Flash.

4.3. Erasing a Page

1. Disable interrupts. This is optional for most devices, but recommended for the 'F12x and 'F13x devices.
2. Set PSWE and PSEE to '1's by writing PSCTL = 0x03.
3. Confirm V_{DD} monitor is enabled.
4. Write a data byte to any location within the 512-byte page to be erased.
5. Set PSWE and PSEE to '0' if no further erases are necessary.
6. Restore interrupts if originally enabled.

Exceptions:

1. Set the SFLE bit before writing if the target address is in the scratchpad area.
2. If the device is a ‘F3xx, 'F4xx, or 'F5xx device, write FLKEY = 0xA5 then FLKEY = 0xF1 before writing the byte.
3. If the device is the 'F12x or 'F13x, check the address and set the correct bank in the PSBANK register before writing the byte.
4. For all ‘F0xx, ‘F1xx, and 'F2xx devices, enable Flash writes in the FLSCL Register (FLSCL = 0x01) before writing the byte. Disable the bit after the byte has been written to Flash.

4.4. Example Code Implementation Notes

The example code that comes with this application note is written for the largest Flash device in the device family. The file name Fxxx_FlashPrimitives.h contains two #defines (FLASH_LAST and FLASH_TEMP) that must be changed to reflect the target's actual Flash size.

Also, the example code explicitly enables the VDD monitor as a reset source through the RSTSRC register. The code that writes to the RSTSRC register uses a direct assignment:

RSTSRC = 0x02;

This code will only enable the VDD monitor as a reset source, and will disable any other reset sources. If your project requires other reset sources, change the example code to enable all of the required reset sources using a single assignment to the RSTSRC register. This code is in the devices's Fxxx_FlashPrimitives.c file.

5. Advanced Flash Operations

The basic routines described in Section 4 can be incorporated into more advanced routines that provide greater flexibility. The following functions are commonly used Flash routines that expand upon the basic routines. The code that implements these routines for each device family is provided at the end of this application note in the files named Fxxx_Utils.h and Fxxx_Utils.c.

Flash_Read—This routine reads multiple bytes from Flash and returns a character string.

```
char * FLASH_Read (char *dest, FLADDR src, unsigned numbytes);
```

Flash_Write—This routine writes multiple bytes to Flash. This function assumes that the target bytes have been cleared to 0xFF. If the bytes are not cleared to 0xFF, incorrect values might be stored in flash.

```
void FLASH_Write (FLADDR dest, char *src, unsigned numbytes);
```

Flash_Clear—This routines clears a string of bytes to 0xFF. The implementation of this function first copies the data to temporary space, erases the page or pages, then copies the non-cleared bytes back to the original page or pages. This routine requires a spare page of Flash to run, and it will clear up to one page size worth of data.

The FLASH_Clear routine can be used to clear bytes before they are written.

```
void FLASH_Clear (FLADDR addr, unsigned numbytes);
```

Flash_Update—This first clears the locations that are going to be written, and then writes the bytes. This routine requires a spare page of Flash to run, and it will update up to one page size worth of data.

The FLASH_Update routine combines FLASH_Clear and FLASH_Write into one routine.

```
FLASH_Update (FLADDR dest, char *src, unsigned numbytes);
```

Flash_Copy—This routine copies bytes from one location in Flash to another location in Flash. It assumes that the bytes are cleared to 0xFF. FLASH_Clear can be used to clear the bytes.

```
FLASH_Copy (FLADDR dest, FLADDR src, unsigned numbytes);
```

Flash_Fill—This routine fills bytes of Flash with a certain values. It assumes that bytes have been previously cleared to 0xFF.

```
void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill);
```

6. Flash Write and Erase Guidelines

Any system which contains routines which write or erase Flash memory from software involves some risk that the write or erase routines will execute unintentionally if the CPU is operating outside its specified operating range of V_{DD}, system clock frequency, or temperature. This accidental execution of Flash modifying code can result in alteration of Flash memory contents causing a system failure that is only recoverable by re-Flashing the code in the device.

The following guidelines are recommended for any system which contains routines which write or erase Flash from code.

6.1. V_{DD} Maintenance and the V_{DD} Monitor

1. If the system power supply is subject to voltage or current "spikes," add sufficient transient protection devices to the power supply to ensure that the supply voltages listed in the Absolute Maximum Ratings table are not exceeded.
2. Ensure that the minimum V_{DD} rise time specification of 1 ms is met. If the device has a minimum VDD rise time specification, ensure that it is met. If the system cannot meet the rise time specification, then add an external V_{DD} brownout circuit to the $\overline{RST}$ pin of the device that holds the device in reset until V_{DD} reaches 2.7 V and re-asserts $\overline{RST}$ if V_{DD} drops below 2.7 V.
3. Enable the on-chip V_{DD} monitor and enable the V_{DD} monitor as a reset source as early in code as possible. This should be the first set of instructions executed after the Reset Vector. For C-based systems, this will involve modifying the startup code added by the C compiler. See your compiler documentation for more details. Ensure that there are no delays in software between enabling the V_{DD} monitor and enabling the V_{DD} monitor as a reset source.
4. As an added precaution, explicitly enable the V_{DD} monitor and enable the V_{DD} monitor as a reset source inside the functions that write and erase Flash memory. The V_{DD} monitor enable instructions should be placed just after the instruction to set PSWE to a '1', but before the Flash write or erase operation instruction.
5. Ensure that all writes to the RSTSRC (Reset Sources) register use direct assignment operators and explicitly DO NOT use the bit-wise operators (such as AND or OR). For example, "RSTSRC = 0x02" is correct. "RSTSRC |= 0x02" is incorrect.
6. Ensure that all writes to the RSTSRC register explicitly set the PORSF bit to a '1'. Areas to check are initialization code which enables other reset sources, such as the Missing Clock Detector or Comparator, for example, and instructions which force a Software Reset. A global search on "RSTSRC" can quickly verify this.
7. If the device has a high and low threshold setting for the VDD monitor, enable the high setting.

Table 1. V_{DD} Monitor Enabling

Family	How to enable the V_{DD} monitor	How to enable as a reset source
‘F00x	(always enabled)	(always enabled)
‘F01x	(always enabled)	(always enabled)
‘F02x	Connect MONEN pin to V_{DD}	Enabled if MONEN pulled high
‘F04x	Connect MONEN pin to V_{DD}	RSTSRC = 0x02;
‘F06x	Connect MONEN pin to V_{DD}	RSTSRC = 0x02;
‘F12x, ‘F13x	Connect MONEN pin to V_{DD}	RSTSRC = 0x02;
‘F2xx	(always enabled on 48 pin package) Connect MONEN pin to V_{DD} on 32 pin package	Enabled if MONEN pulled high
‘F30x	(always enabled)	RSTSRC = 0x02;[1]
‘F31x	VDMEN = ‘1’;	RSTSRC = 0x02;[2]
‘F32x	VDMEN = ‘1’;	RSTSRC = 0x02;[2]
‘F326	VDMEN = ‘1’;	RSTSRC = 0x02;[2]
‘F33x	VDMEN = ‘1’;	RSTSRC = 0x02;[2]
‘F34x	VDMEN = ‘1’;	RSTSRC = 0x02;[2]
‘F35x	VDMEN = ‘1’;	RSTSRC = 0x02;[2]
‘F36x	VDMEN = ‘1’;	RSTSRC = 0x02;[2]
‘F41x	VDMEN = '1'; VDMLVL = '1';	RSTSRC = 0x02;[2]
‘F52x, ‘F53x	VDDMON = '1'; VDMLVL = '1';	RSTSRC = 0x02;[2]

Notes:
1. On the ‘F30x devices, enabling the V_{DD} monitor by setting the PORSF bit when the V_{DD} monitor is disabled may cause a system reset.
2. In software which writes or erases Flash memory, the V_{DD} monitor should be enabled as a reset source immediately following setting VDMEN to a '1'.

6.2. PSWE Maintenance

8. Reduce the number of places in code where the PSWE bit (b0 in PSCTL) is set to a '1'. There should be exactly one routine in code that sets PSWE to a '1' to write Flash bytes and one routine in code that sets PSWE and PSEE both to a '1' to erase Flash pages.
9. Minimize the number of variable accesses while PSWE is set to a '1'. Handle pointer address updates and loop variable maintenance outside the "PSWE = 1; ... PSWE = 0;" area.
10. Disable interrupts prior to setting PSWE to a '1' and leave them disabled until after PSWE has been reset to '0'. Any interrupts posted during the Flash write or erase operation will be serviced in priority order after the Flash operation has been completed and interrupts have been re-enabled by software.
11. Ensure that the Flash write and erase pointer variables are not located in XRAM. See your compiler documentation for instructions regarding how to explicitly locate variables in different memory areas.
12. Add address bounds checking to the routines that write or erase Flash memory to ensure that a routine called with an illegal address does not result in modification of the Flash.

6.3. System Clock

13. If operating from an external crystal, be advised that crystal performance is susceptible to electrical interference and is sensitive to layout and to changes in temperature. If the system is operating in an electrically noisy environment, use the internal oscillator or use an external CMOS clock.
14. If operating from the external oscillator, switch to the internal oscillator during Flash write or erase operations. The external oscillator can continue to run, and the CPU can switch back to the external oscillator after the Flash operation has completed.

7. Example Code

The following code examples are organized by device family because devices in the same family are code compatible. These code examples target the device that has the largest Flash size in each family. The code changes that need to be made for devices with a different Flash size are the locations of the temporary page and last page of Flash. These addresses are located in FlashPrimitives.h. See the Flash section of the device's data sheet for more information.

7.1. ‘F000

7.1.1. F000_FlashPrimitives.c

```
//-----------------------------------------------------------------------------
// F000_FlashPrimitives.c
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F0xx
// Tool chain: KEIL C51 7.06
//
// Release 1.0
//

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F000_FlashPrimitives.h"
#include <c8051F000.h>

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
void FLASH_ByteWrite (FLADDR addr, char byte);
unsigned char FLASH_ByteRead (FLADDR addr);
void FLASH_PageErase (FLADDR addr);

//-----------------------------------------------------------------------------
// Global Variables
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH Routines
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH_ByteWrite
//-----------------------------------------------------------------------------
//
// This routine writes <byte> to the linear FLASH address <addr>.
// Linear map is decoded as follows:
// Linear Address       Device Address
// ------------------------------------------------
// 0x00000 - 0x0FFFF    0x0000 - 0xFFFF
```

```
//
//
void FLASH_ByteWrite (FLADDR addr, char byte)
{
   bit EA_SAVE = EA;                   // preserve EA
   char xdata * data pwrite;           // FLASH write pointer

   EA = 0;                             // disable interrupts

   pwrite = (char xdata *) addr;       // initialize write pointer

   FLSCL  = FLASHSCALE;                // enable FLASH writes/erases
   PSCTL |= 0x01;                      // PSWE = 1

   *pwrite = byte;                     // write the byte

   PSCTL &= ~0x01;                     // PSWE = 0
   FLSCL |=  0x0F;                     // disable FLASH writes/erases

   EA = EA_SAVE;                       // restore interrupts
}

//-----------------------------------------------------------------------------
// FLASH_ByteRead
//-----------------------------------------------------------------------------
//
// This routine reads a <byte> from the linear FLASH address <addr>.
//
unsigned char FLASH_ByteRead (FLADDR addr)
{
   bit EA_SAVE = EA;                   // preserve EA
   char code * data pread;             // FLASH read pointer
   unsigned char byte;

   pread = (char code *) addr;         // initialize read pointer

   EA = 0;                             // disable interrupts

   byte = *pread;                      // read the byte

   EA = EA_SAVE;                       // restore interrupts

   return byte;
}

//-----------------------------------------------------------------------------
// FLASH_PageErase
//-----------------------------------------------------------------------------
//
// This routine erases the FLASH page containing the linear FLASH address
// <addr>.
//
void FLASH_PageErase (FLADDR addr)
{
   bit EA_SAVE = EA;                   // preserve EA
   char xdata * data pwrite;           // FLASH write pointer

   EA = 0;                             // disable interrupts

   pwrite = (char xdata *) addr;       // initialize erase pointer

   FLSCL  = FLASHSCALE;                // enable FLASH writes/erases
   PSCTL |= 0x03;                      // PSWE = 1; PSEE = 1

   *pwrite = 0;                        // initiate page erase

   PSCTL &= ~0x03;                     // PSWE = 0; PSEE = 0
   FLSCL |=  0x0F;                     // disable FLASH writes/erases

   EA = EA_SAVE;                       // restore interrupts
}
```

7.1.2. F000_FlashPrimitives.h

```
//-----------------------------------------------------------------------------
// F000_FlashPrimitives.h
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F0xx
// Tool chain: KEIL C51 7.06
//
// Release 1.0
//

#ifndef F000_FLASHPRIMITIVES_H
#define F000_FLASHPRIMITIVES_H

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

typedef unsigned long ULONG;
typedef unsigned int UINT;
typedef unsigned char UCHAR;

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

#ifndef SYSCLK
#define SYSCLK 16000000L
#endif

#ifndef FLASHSCALE

#if   (SYSCLK < 50000L)
#define FLASHSCALE 0x80
#elif (SYSCLK < 100000L)
#define FLASHSCALE 0x81
#elif (SYSCLK < 200000L)
#define FLASHSCALE 0x82
#elif (SYSCLK < 400000L)
#define FLASHSCALE 0x83
#elif (SYSCLK < 800000L)
#define FLASHSCALE 0x84
#elif (SYSCLK < 1600000L)
#define FLASHSCALE 0x85
#elif (SYSCLK < 3200000L)
#define FLASHSCALE 0x86
#elif (SYSCLK < 6400000L)
#define FLASHSCALE 0x87
#elif (SYSCLK < 12800000L)
#define FLASHSCALE 0x88
#elif (SYSCLK < 25600000L)
#define FLASHSCALE 0x89
#endif // SYSCLK test
```

```
#endif // FLASHSCALE

#ifndef FLASH_PAGESIZE
#define FLASH_PAGESIZE 512
#endif

#ifndef FLASH_SCRATCHSIZE
#define FLASH_SCRATCHSIZE 128
#endif

#ifndef FLASH_TEMP
#define FLASH_TEMP 0x07800L
#endif

#ifndef FLASH_LAST
#define FLASH_LAST 0x07A00L            // last page of FLASH
#endif

typedef UINT FLADDR;

//-----------------------------------------------------------------------------
// Exported Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
extern void FLASH_ByteWrite (FLADDR addr, char byte);
extern unsigned char FLASH_ByteRead (FLADDR addr);
extern void FLASH_PageErase (FLADDR addr);

#endif // F000_FLASHPRIMITIVES_H
```

7.1.3. F000_FlashUtils.c

```
//-----------------------------------------------------------------------------
// F000_FlashUtils.c
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F0xx
// Tool chain: KEIL C51 7.06
//
// Release 1.0
//

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F000_FlashPrimitives.h"
#include "F000_FlashUtils.h"

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
void FLASH_Write (FLADDR dest, char *src, unsigned numbytes);
char * FLASH_Read (char *dest, FLADDR src, unsigned numbytes);
void FLASH_Clear (FLADDR addr, unsigned numbytes);

// FLASH update/copy routines
void FLASH_Update (FLADDR dest, char *src, unsigned numbytes);
void FLASH_Copy (FLADDR dest, FLADDR src, unsigned numbytes);

// FLASH test routines
void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill);

//-----------------------------------------------------------------------------
// Global Variables
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH Routines
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH_Clear
//-----------------------------------------------------------------------------
//
// This routine erases <numbytes> starting from the FLASH addressed by
// <dest> by performing a read-modify-write operation using <FLASH_TEMP> as
// a temporary holding area.  This function accepts <numbytes> up to
```

```
// <FLASH_PAGESIZE>.
//
void FLASH_Clear (FLADDR dest, unsigned numbytes)
{
   FLADDR dest_1_page_start;           // first address in 1st page
                                       // containing <dest>
   FLADDR dest_1_page_end;             // last address in 1st page
                                       // containing <dest>
   FLADDR dest_2_page_start;           // first address in 2nd page
                                       // containing <dest>
   FLADDR dest_2_page_end;             // last address in 2nd page
                                       // containing <dest>
   unsigned numbytes_remainder;        // when crossing page boundary,
                                       // number of <src> bytes on 2nd page
   unsigned FLASH_pagesize;            // size of FLASH page to update

   FLADDR  wptr;                       // write address
   FLADDR  rptr;                       // read address

   unsigned length;

   FLASH_pagesize = FLASH_PAGESIZE;

   dest_1_page_start = dest & ~(FLASH_pagesize - 1);
   dest_1_page_end = dest_1_page_start + FLASH_pagesize - 1;
   dest_2_page_start = (dest + numbytes)  & ~(FLASH_pagesize - 1);
   dest_2_page_end = dest_2_page_start + FLASH_pagesize - 1;

   if (dest_1_page_end == dest_2_page_end) {

      // 1. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP);

      // 2. Copy bytes from first byte of dest page to dest-1 to Scratch page

      wptr = FLASH_TEMP;
      rptr = dest_1_page_start;
      length = dest - dest_1_page_start;
      FLASH_Copy (wptr, rptr, length);

      // 3. Copy from (dest+numbytes) to dest_page_end to Scratch page

      wptr = FLASH_TEMP + dest - dest_1_page_start + numbytes;
      rptr = dest + numbytes;
      length = dest_1_page_end - dest - numbytes + 1;
      FLASH_Copy (wptr, rptr, length);

      // 4. Erase destination page
      FLASH_PageErase (dest_1_page_start);

      // 5. Copy Scratch page to destination page
      wptr = dest_1_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, rptr, length);

   } else {                            // value crosses page boundary
      // 1. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP);

      // 2. Copy bytes from first byte of dest page to dest-1 to Scratch page

      wptr = FLASH_TEMP;
      rptr = dest_1_page_start;
```

```
      length = dest - dest_1_page_start;
      FLASH_Copy (wptr, rptr, length);

      // 3. Erase destination page 1
      FLASH_PageErase (dest_1_page_start);

      // 4. Copy Scratch page to destination page 1
      wptr = dest_1_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, rptr, length);

      // now handle 2nd page

      // 5. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP);

      // 6. Copy bytes from numbytes remaining to dest-2_page_end to Scratch page

      numbytes_remainder = numbytes - (dest_1_page_end - dest + 1);
      wptr = FLASH_TEMP + numbytes_remainder;
      rptr = dest_2_page_start + numbytes_remainder;
      length = FLASH_pagesize - numbytes_remainder;
      FLASH_Copy (wptr, rptr, length);

      // 7. Erase destination page 2
      FLASH_PageErase (dest_2_page_start);

      // 8. Copy Scratch page to destination page 2
      wptr = dest_2_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, rptr, length);
   }
}

//-----------------------------------------------------------------------------
// FLASH_Update
//-----------------------------------------------------------------------------
//
// This routine replaces <numbytes> from <src> to the FLASH addressed by
// <dest>.  This function calls FLASH_Clear() to handle the dirty work of
// initializing all <dest> bytes to 0xff's prior to copying the bytes from
// <src> to <dest>. This function accepts <numbytes> up to <FLASH_PAGESIZE>.
//
void FLASH_Update (FLADDR dest, char *src, unsigned numbytes)
{
   // 1. Erase <numbytes> starting from <dest>
   FLASH_Clear (dest, numbytes);

   // 2. Write <numbytes> from <src> to <dest>
   FLASH_Write (dest, src, numbytes);
}


//-----------------------------------------------------------------------------
// FLASH_Write
//-----------------------------------------------------------------------------
//
// This routine copies <numbytes> from <src> to the linear FLASH address
// <dest>.
//
void FLASH_Write (FLADDR dest, char *src, unsigned numbytes)
{
```

```
   FLADDR i;

   for (i = dest; i < dest+numbytes; i++) {
      FLASH_ByteWrite (i, *src++);
   }
}

//-----------------------------------------------------------------------------
// FLASH_Read
//-----------------------------------------------------------------------------
//
// This routine copies <numbytes> from the linear FLASH address <src> to
// <dest>.
//
char * FLASH_Read (char *dest, FLADDR src, unsigned numbytes)
{
   FLADDR i;

   for (i = 0; i < numbytes; i++) {
      *dest++ = FLASH_ByteRead (src+i);
   }
   return dest;
}

//-----------------------------------------------------------------------------
// FLASH_Copy
//-----------------------------------------------------------------------------
//
// This routine copies <numbytes> from <src> to the linear FLASH address
// <dest>.
//
void FLASH_Copy (FLADDR dest, FLADDR src, unsigned numbytes)
{
   FLADDR i;

   for (i = 0; i < numbytes; i++) {

      FLASH_ByteWrite ((FLADDR) dest+i, FLASH_ByteRead((FLADDR) src+i));
   }
}


//-----------------------------------------------------------------------------
// FLASH_Fill
//-----------------------------------------------------------------------------
//
// This routine fills the FLASH beginning at <addr> with <lenght> bytes.
//
void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill)
{
   FLADDR i;

   for (i = 0; i < length; i++) {
      FLASH_ByteWrite (addr+i, fill);
   }
}
```

7.1.4. F000_FlashUtils.h

```
//-----------------------------------------------------------------------------
// F000_FlashUtils.h
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F0xx
// Tool chain: KEIL C51 7.06
//
// Release 1.0
//

#ifndef F000_FLASHUTILS_H
#define F000_FLASHUTILS_H

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F000_FlashPrimitives.h"

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Exported Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
extern void FLASH_Write (FLADDR dest, char *src, unsigned numbytes);
extern char * FLASH_Read (char *dest, FLADDR src, unsigned numbytes);
extern void FLASH_Clear (FLADDR addr, unsigned numbytes);

// FLASH update/copy routines
extern void FLASH_Update (FLADDR dest, char *src, unsigned numbytes);   // copy with destina-
tion preservation
extern void FLASH_Copy (FLADDR dest, FLADDR src, unsigned numbytes);     // low-level FLASH/
FLASH byte copy

// FLASH test routines
extern void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill);

#endif // F000_FLASHUTILS_H
```

7.2. ‘F020

7.2.1. F020_FlashPrimitives.c

```
//-----------------------------------------------------------------------------
// F020_FlashPrimitives.c
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F2xx
// Tool chain: KEIL C51 7.06
//
// Release 1.0
//

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F020_FlashPrimitives.h"
#include <c8051F020.h>

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
void FLASH_ByteWrite (FLADDR addr, char byte, bit SFLE);
unsigned char FLASH_ByteRead (FLADDR addr, bit SFLE);
void FLASH_PageErase (FLADDR addr, bit SFLE);

//-----------------------------------------------------------------------------
// Global Variables
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH Routines
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH_ByteWrite
//-----------------------------------------------------------------------------
//
// This routine writes <byte> to the linear FLASH address <addr>.
// Linear map is decoded as follows:
// Linear Address       Device Address
// ----------------------------------------------
// 0x00000 - 0x0FFFF    0x0000 - 0xFFFF
//
void FLASH_ByteWrite (FLADDR addr, char byte, bit SFLE)
{
```

```
   bit EA_SAVE = EA;                    // preserve EA
   char xdata * data pwrite;            // FLASH write pointer

   EA = 0;                              // disable interrupts

   pwrite = (char xdata *) addr;

   FLSCL |= 0x01;                       // enable FLASH writes/erases
   PSCTL |= 0x01;                       // PSWE = 1

   if (SFLE) {
      PSCTL |= 0x04;                    // set SFLE
   }

   *pwrite = byte;                      // write the byte

   if (SFLE) {
      PSCTL &= ~0x04;                   // clear SFLE
   }

   PSCTL &= ~0x01;                      // PSWE = 0
   FLSCL &= ~0x01;                      // disable FLASH writes/erases

   EA = EA_SAVE;                        // restore interrupts
}

//-----------------------------------------------------------------------------
// FLASH_ByteRead
//-----------------------------------------------------------------------------
//
// This routine reads a <byte> from the linear FLASH address <addr>.
//
unsigned char FLASH_ByteRead (FLADDR addr, bit SFLE)
{
   bit EA_SAVE = EA;                    // preserve EA
   char code * data pread;              // FLASH read pointer
   unsigned char byte;

   EA = 0;                              // disable interrupts

   pread = (char code *) addr;

   if (SFLE) {
      PSCTL |= 0x04;                    // set SFLE
   }

   byte = *pread;                       // read the byte

   if (SFLE) {
      PSCTL &= ~0x04;                   // clear SFLE
   }

   EA = EA_SAVE;                        // restore interrupts

   return byte;
}

//-----------------------------------------------------------------------------
// FLASH_PageErase
//-----------------------------------------------------------------------------
//
// This routine erases the FLASH page containing the linear FLASH address
// <addr>.
//
```

```
void FLASH_PageErase (FLADDR addr, bit SFLE)
{
   bit EA_SAVE = EA;                   // preserve EA
   char xdata * data pwrite;           // FLASH write pointer

   EA = 0;                             // disable interrupts

   pwrite = (char xdata *) addr;

   FLSCL |= 0x01;                      // enable FLASH writes/erases
   PSCTL |= 0x03;                      // PSWE = 1; PSEE = 1

   if (SFLE) {
      PSCTL |= 0x04;                   // set SFLE
   }

   *pwrite = 0;                        // initiate page erase

   if (SFLE) {
      PSCTL &= ~0x04;                  // clear SFLE
   }

   PSCTL &= ~0x03;                     // PSWE = 0; PSEE = 0
   FLSCL &= ~0x01;                     // disable FLASH writes/erases

   EA = EA_SAVE;                       // restore interrupts
}
```

7.2.2. F020_FlashPrimitives.h

```
//-----------------------------------------------------------------------------
// F020_FlashPrimitives.h
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F2xx
// Tool chain: KEIL C51 7.06
//
// Release 1.0
//

#ifndef F020_FLASHPRIMITIVES_H
#define F020_FLASHPRIMITIVES_H

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------
typedef unsigned long ULONG;
typedef unsigned int UINT;
typedef unsigned char UCHAR;

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

#ifndef FLASH_PAGESIZE
#define FLASH_PAGESIZE 512
#endif

#ifndef FLASH_SCRATCHSIZE
#define FLASH_SCRATCHSIZE 128
#endif

#ifndef FLASH_TEMP
#define FLASH_TEMP 0x0F800L
#endif

#ifndef FLASH_LAST
#define FLASH_LAST 0x0FA00L            // last page of FLASH
#endif

typedef UINT FLADDR;

//-----------------------------------------------------------------------------
// Exported Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
extern void FLASH_ByteWrite (FLADDR addr, char byte, bit SFLE);
extern unsigned char FLASH_ByteRead (FLADDR addr, bit SFLE);
extern void FLASH_PageErase (FLADDR addr, bit SFLE);

#endif // F020_FLASHPRIMITIVES_H
```

7.2.3. F020_FlashUtils.c

```
//-----------------------------------------------------------------------------
// F020_FlashUtils.c
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F2xx
// Tool chain: KEIL C51 7.06
//
// Release 1.0
//

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F020_FlashPrimitives.h"
#include "F020_FlashUtils.h"

#include <stdio.h>

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
void FLASH_Write (FLADDR dest, char *src, unsigned numbytes, bit SFLE);
char * FLASH_Read (char *dest, FLADDR src, unsigned numbytes, bit SFLE);
void FLASH_Clear (FLADDR addr, unsigned numbytes, bit SFLE);

// FLASH update/copy routines
void FLASH_Update (FLADDR dest, char *src, unsigned numbytes, bit SFLE);
void FLASH_Copy (FLADDR dest, bit destSFLE, FLADDR src, bit srcSFLE,
                 unsigned numbytes);

// FLASH test routines
void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill, bit SFLE);

//-----------------------------------------------------------------------------
// Global Variables
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH Routines
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH_Clear
//-----------------------------------------------------------------------------
//
// This routine erases <numbytes> starting from the FLASH addressed by
// <dest> by performing a read-modify-write operation using <FLASH_TEMP> as
```

```
// a temporary holding area.  This function accepts <numbytes> up to
// <FLASH_PAGESIZE>.
//
void FLASH_Clear (FLADDR dest, unsigned numbytes, bit SFLE)
{
   FLADDR dest_1_page_start;           // first address in 1st page
                                       // containing <dest>
   FLADDR dest_1_page_end;             // last address in 1st page
                                       // containing <dest>
   FLADDR dest_2_page_start;           // first address in 2nd page
                                       // containing <dest>
   FLADDR dest_2_page_end;             // last address in 2nd page
                                       // containing <dest>
   unsigned numbytes_remainder;        // when crossing page boundary,
                                       // number of <src> bytes on 2nd page
   unsigned FLASH_pagesize;            // size of FLASH page to update

   FLADDR  wptr;                       // write address
   FLADDR  rptr;                       // read address

   unsigned length;

   if (SFLE) {                         // update Scratchpad
      FLASH_pagesize = FLASH_SCRATCHSIZE;
   } else {
      FLASH_pagesize = FLASH_PAGESIZE;
   }

   dest_1_page_start = dest & ~(FLASH_pagesize - 1);
   dest_1_page_end = dest_1_page_start + FLASH_pagesize - 1;
   dest_2_page_start = (dest + numbytes)  & ~(FLASH_pagesize - 1);
   dest_2_page_end = dest_2_page_start + FLASH_pagesize - 1;

   if (dest_1_page_end == dest_2_page_end) {

      // 1. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP, 0);

      // 2. Copy bytes from first byte of dest page to dest-1 to Scratch page

      wptr = FLASH_TEMP;
      rptr = dest_1_page_start;
      length = dest - dest_1_page_start;
      FLASH_Copy (wptr, 0, rptr, SFLE, length);

      // 3. Copy from (dest+numbytes) to dest_page_end to Scratch page

      wptr = FLASH_TEMP + dest - dest_1_page_start + numbytes;
      rptr = dest + numbytes;
      length = dest_1_page_end - dest - numbytes + 1;
      FLASH_Copy (wptr, 0, rptr, SFLE, length);

      // 4. Erase destination page
      FLASH_PageErase (dest_1_page_start, SFLE);

      // 5. Copy Scratch page to destination page
      wptr = dest_1_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, SFLE, rptr, 0, length);

   } else {                            // value crosses page boundary
      // 1. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP, 0);
```

```
      // 2. Copy bytes from first byte of dest page to dest-1 to Scratch page

      wptr = FLASH_TEMP;
      rptr = dest_1_page_start;
      length = dest - dest_1_page_start;
      FLASH_Copy (wptr, 0, rptr, SFLE, length);

      // 3. Erase destination page 1
      FLASH_PageErase (dest_1_page_start, SFLE);

      // 4. Copy Scratch page to destination page 1
      wptr = dest_1_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, SFLE, rptr, 0, length);

      // now handle 2nd page

      // 5. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP, 0);

      // 6. Copy bytes from numbytes remaining to dest-2_page_end to Scratch page

      numbytes_remainder = numbytes - (dest_1_page_end - dest + 1);
      wptr = FLASH_TEMP + numbytes_remainder;
      rptr = dest_2_page_start + numbytes_remainder;
      length = FLASH_pagesize - numbytes_remainder;
      FLASH_Copy (wptr, 0, rptr, SFLE, length);

      // 7. Erase destination page 2
      FLASH_PageErase (dest_2_page_start, SFLE);

      // 8. Copy Scratch page to destination page 2
      wptr = dest_2_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, SFLE, rptr, 0, length);
   }
}


//-----------------------------------------------------------------------------
// FLASH_Update
//-----------------------------------------------------------------------------
//
// This routine replaces <numbytes> from <src> to the FLASH addressed by
// <dest>.  This function calls FLASH_Clear() to handle the dirty work of
// initializing all <dest> bytes to 0xff's prior to copying the bytes from
// <src> to <dest>. This function accepts <numbytes> up to <FLASH_PAGESIZE>.
//
void FLASH_Update (FLADDR dest, char *src, unsigned numbytes, bit SFLE)
{
   // 1. Erase <numbytes> starting from <dest>
   FLASH_Clear (dest, numbytes, SFLE);

   // 2. Write <numbytes> from <src> to <dest>
   FLASH_Write (dest, src, numbytes, SFLE);
}


//-----------------------------------------------------------------------------
// FLASH_Write
//-----------------------------------------------------------------------------
//
```

```
// This routine copies <numbytes> from <src> to the linear FLASH address
// <dest>.
//
void FLASH_Write (FLADDR dest, char *src, unsigned numbytes, bit SFLE)
{
   FLADDR i;

   for (i = dest; i < dest+numbytes; i++) {
      FLASH_ByteWrite (i, *src++, SFLE);
   }
}

//-----------------------------------------------------------------------------
// FLASH_Read
//-----------------------------------------------------------------------------
//
// This routine copies <numbytes> from the linear FLASH address <src> to
// <dest>.
//
char * FLASH_Read (char *dest, FLADDR src, unsigned numbytes, bit SFLE)
{
   FLADDR i;

   for (i = 0; i < numbytes; i++) {
      *dest++ = FLASH_ByteRead (src+i, SFLE);
   }
   return dest;
}

//-----------------------------------------------------------------------------
// FLASH_Copy
//-----------------------------------------------------------------------------
//
// This routine copies <numbytes> from <src> to the linear FLASH address
// <dest>.
//
void FLASH_Copy (FLADDR dest, bit destSFLE, FLADDR src, bit srcSFLE,
                 unsigned numbytes)
{
   FLADDR i;

   for (i = 0; i < numbytes; i++) {

      FLASH_ByteWrite ((FLADDR) dest+i,
                       FLASH_ByteRead((FLADDR) src+i, srcSFLE),
                       destSFLE);
   }
}


//-----------------------------------------------------------------------------
// FLASH_Fill
//-----------------------------------------------------------------------------
//
// This routine fills the FLASH beginning at <addr> with <lenght> bytes.
//
void FLASH_Fill (FLADDR addr, ULONG length, unsigned char fill, bit SFLE)
{
   FLADDR i;

   for (i = 0; i < length; i++) {
      FLASH_ByteWrite (addr+i, fill, SFLE);
   }
}
```

7.2.4. F020_FlashUtils.h

```
//-----------------------------------------------------------------------------
// F020_FlashUtils.h
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F2xx
// Tool chain: KEIL C51 7.06
//
// Release 1.0
//

#ifndef F020_FLASHUTILS_H
#define F020_FLASHUTILS_H

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F020_FlashPrimitives.h"

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Exported Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
extern void FLASH_Write (FLADDR dest, char *src, unsigned numbytes, bit SFLE);
extern char * FLASH_Read (char *dest, FLADDR src, unsigned numbytes, bit SFLE);
extern void FLASH_Clear (FLADDR addr, unsigned numbytes, bit SFLE);

// FLASH update/copy routines
extern void FLASH_Update (FLADDR dest, char *src, unsigned numbytes, bit SFLE);
extern void FLASH_Copy (FLADDR dest, bit destSFLE, FLADDR src, bit srcSFLE,
                 unsigned numbytes);

// FLASH test routines
extern void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill, bit SFLE);

#endif // F020_FLASHUTILS_H
```

7.3. ‘F040

7.3.1. F040_FlashPrimitives.c

```
//-----------------------------------------------------------------------------
// F040_FlashPrimitives.c
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F04x
// Tool chain: KEIL C51 7.06
//
// Release 1.0
//

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F040_FlashPrimitives.h"
#include <c8051F040.h>

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
void FLASH_ByteWrite (FLADDR addr, char byte, bit SFLE);
unsigned char FLASH_ByteRead (FLADDR addr, bit SFLE);
void FLASH_PageErase (FLADDR addr, bit SFLE);

//-----------------------------------------------------------------------------
// Global Variables
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH Routines
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH_ByteWrite
//-----------------------------------------------------------------------------
//
// This routine writes <byte> to the linear FLASH address <addr>.
// Linear map is decoded as follows:
// Linear Address       Device Address
// ----------------------------------------------
// 0x00000 - 0x0FFFF    0x0000 - 0xFFFF
//
void FLASH_ByteWrite (FLADDR addr, char byte, bit SFLE)
{
```

```
   char SFRPAGE_SAVE = SFRPAGE;        // preserve SFRPAGE
   bit EA_SAVE = EA;                   // preserve EA
   char xdata * data pwrite;           // FLASH write pointer

   EA = 0;                             // disable interrupts

   pwrite = (char xdata *) addr;       // initialize write pointer

   SFRPAGE = LEGACY_PAGE;

   FLSCL |= 0x01;                      // enable FLASH writes/erases
   PSCTL |= 0x01;                      // PSWE = 1

   if (SFLE) {
      PSCTL |= 0x04;                   // set SFLE
   }

   RSTSRC = 0x02;                      // enable VDDMON as reset source
   *pwrite = byte;                     // write the byte

   if (SFLE) {
      PSCTL &= ~0x04;                  // clear SFLE
   }
   PSCTL &= ~0x01;                     // PSWE = 0
   FLSCL &= ~0x01;                     // disable FLASH writes/erases

   SFRPAGE = SFRPAGE_SAVE;             // restore SFRPAGE
   EA = EA_SAVE;                       // restore interrupts
}

//-----------------------------------------------------------------------------
// FLASH_ByteRead
//-----------------------------------------------------------------------------
//
// This routine reads a <byte> from the linear FLASH address <addr>.
//
unsigned char FLASH_ByteRead (FLADDR addr, bit SFLE)
{
   char SFRPAGE_SAVE = SFRPAGE;        // preserve SFRPAGE
   bit EA_SAVE = EA;                   // preserve EA
   char code * data pread;             // FLASH read pointer
   unsigned char byte;

   pread = (char code *) addr;         // initialize read pointer

   EA = 0;                             // disable interrupts

   SFRPAGE = LEGACY_PAGE;

   if (SFLE) {
      PSCTL |= 0x04;                   // set SFLE
   }

   byte = *pread;                      // read the byte

   if (SFLE) {
      PSCTL &= ~0x04;                  // clear SFLE
   }

   SFRPAGE = SFRPAGE_SAVE;             // restore SFRPAGE
   EA = EA_SAVE;                       // restore interrupts

   return byte;
}
```

```
//-----------------------------------------------------------------------------
// FLASH_PageErase
//-----------------------------------------------------------------------------
//
// This routine erases the FLASH page containing the linear FLASH address
// <addr>.
//
void FLASH_PageErase (FLADDR addr, bit SFLE)
{
   char SFRPAGE_SAVE = SFRPAGE;        // preserve SFRPAGE
   bit EA_SAVE = EA;                   // preserve EA
   char xdata * data pwrite;           // FLASH write pointer

   EA = 0;                             // disable interrupts

   pwrite = (char xdata *) addr;       // initialize erase pointer

   SFRPAGE = LEGACY_PAGE;

   FLSCL |= 0x01;                      // enable FLASH writes/erases
   PSCTL |= 0x03;                      // PSWE = 1; PSEE = 1

   if (SFLE) {
      PSCTL |= 0x04;                   // set SFLE
   }

   RSTSRC = 0x02;                      // enable VDDMON as reset source
   *pwrite = 0;                        // initiate page erase

   if (SFLE) {
      PSCTL &= ~0x04;                  // clear SFLE
   }

   PSCTL &= ~0x03;                     // PSWE = 0; PSEE = 0
   FLSCL &= ~0x01;                     // disable FLASH writes/erases

   SFRPAGE = SFRPAGE_SAVE;             // restore SFRPAGE
   EA = EA_SAVE;                       // restore interrupts
}
```

7.3.2. F040_FlashPrimitives.h

```
//-----------------------------------------------------------------------------
// F040_FlashPrimitives.h
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F04x
// Tool chain: KEIL C51 7.06
//
// Release 1.0
//

#ifndef F040_FLASHPRIMITIVES_H
#define F040_FLASHPRIMITIVES_H

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

typedef unsigned long ULONG;
typedef unsigned int UINT;
typedef unsigned char UCHAR;

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

#ifndef FLASH_PAGESIZE
#define FLASH_PAGESIZE 512
#endif

#ifndef FLASH_SCRATCHSIZE
#define FLASH_SCRATCHSIZE 128
#endif

#ifndef FLASH_TEMP
#define FLASH_TEMP 0x0F800L
#endif

#ifndef FLASH_LAST
#define FLASH_LAST 0x0FA00L            // last page of FLASH
#endif

typedef UINT FLADDR;

//-----------------------------------------------------------------------------
// Exported Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
extern void FLASH_ByteWrite (FLADDR addr, char byte, bit SFLE);
extern unsigned char FLASH_ByteRead (FLADDR addr, bit SFLE);
extern void FLASH_PageErase (FLADDR addr, bit SFLE);

#endif // F040_FLASHPRIMITIVES_H
```

7.3.3. F040_FlashUtils.c

```
//-----------------------------------------------------------------------------
// F040_FlashUtils.c
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F04x
// Tool chain: KEIL C51 7.06
//
// Release 1.0
//

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F040_FlashPrimitives.h"
#include "F040_FlashUtils.h"

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
void FLASH_Write (FLADDR dest, char *src, unsigned numbytes, bit SFLE);
char * FLASH_Read (char *dest, FLADDR src, unsigned numbytes, bit SFLE);
void FLASH_Clear (FLADDR addr, unsigned numbytes, bit SFLE);

// FLASH update/copy routines
void FLASH_Update (FLADDR dest, char *src, unsigned numbytes, bit SFLE);
void FLASH_Copy (FLADDR dest, bit destSFLE, FLADDR src, bit srcSFLE,
                 unsigned numbytes);

// FLASH test routines
void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill, bit SFLE);

//-----------------------------------------------------------------------------
// Global Variables
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH Routines
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH_Clear
//-----------------------------------------------------------------------------
//
// This routine erases <numbytes> starting from the FLASH addressed by
// <dest> by performing a read-modify-write operation using <FLASH_TEMP> as
// a temporary holding area.  This function accepts <numbytes> up to
// <FLASH_PAGESIZE>.
```

```
//
void FLASH_Clear (FLADDR dest, unsigned numbytes, bit SFLE)
{
   FLADDR dest_1_page_start;           // first address in 1st page
                                       // containing <dest>
   FLADDR dest_1_page_end;             // last address in 1st page
                                       // containing <dest>
   FLADDR dest_2_page_start;           // first address in 2nd page
                                       // containing <dest>
   FLADDR dest_2_page_end;             // last address in 2nd page
                                       // containing <dest>
   unsigned numbytes_remainder;        // when crossing page boundary,
                                       // number of <src> bytes on 2nd page
   unsigned FLASH_pagesize;            // size of FLASH page to update

   FLADDR  wptr;                       // write address
   FLADDR  rptr;                       // read address

   unsigned length;

   if (SFLE) {                         // update Scratchpad
      FLASH_pagesize = FLASH_SCRATCHSIZE;
   } else {
      FLASH_pagesize = FLASH_PAGESIZE;
   }

   dest_1_page_start = dest & ~(FLASH_pagesize - 1);
   dest_1_page_end = dest_1_page_start + FLASH_pagesize - 1;
   dest_2_page_start = (dest + numbytes)  & ~(FLASH_pagesize - 1);
   dest_2_page_end = dest_2_page_start + FLASH_pagesize - 1;

   if (dest_1_page_end == dest_2_page_end) {

      // 1. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP, 0);

      // 2. Copy bytes from first byte of dest page to dest-1 to Scratch page

      wptr = FLASH_TEMP;
      rptr = dest_1_page_start;
      length = dest - dest_1_page_start;
      FLASH_Copy (wptr, 0, rptr, SFLE, length);

      // 3. Copy from (dest+numbytes) to dest_page_end to Scratch page

      wptr = FLASH_TEMP + dest - dest_1_page_start + numbytes;
      rptr = dest + numbytes;
      length = dest_1_page_end - dest - numbytes + 1;
      FLASH_Copy (wptr, 0, rptr, SFLE, length);

      // 4. Erase destination page
      FLASH_PageErase (dest_1_page_start, SFLE);

      // 5. Copy Scratch page to destination page
      wptr = dest_1_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, SFLE, rptr, 0, length);

   } else {                            // value crosses page boundary
      // 1. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP, 0);

      // 2. Copy bytes from first byte of dest page to dest-1 to Scratch page
```

```
      wptr = FLASH_TEMP;
      rptr = dest_1_page_start;
      length = dest - dest_1_page_start;
      FLASH_Copy (wptr, 0, rptr, SFLE, length);

      // 3. Erase destination page 1
      FLASH_PageErase (dest_1_page_start, SFLE);

      // 4. Copy Scratch page to destination page 1
      wptr = dest_1_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, SFLE, rptr, 0, length);

      // now handle 2nd page

      // 5. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP, 0);

      // 6. Copy bytes from numbytes remaining to dest-2_page_end to Scratch page

      numbytes_remainder = numbytes - (dest_1_page_end - dest + 1);
      wptr = FLASH_TEMP + numbytes_remainder;
      rptr = dest_2_page_start + numbytes_remainder;
      length = FLASH_pagesize - numbytes_remainder;
      FLASH_Copy (wptr, 0, rptr, SFLE, length);

      // 7. Erase destination page 2
      FLASH_PageErase (dest_2_page_start, SFLE);

      // 8. Copy Scratch page to destination page 2
      wptr = dest_2_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, SFLE, rptr, 0, length);
   }
}


//-----------------------------------------------------------------------------
// FLASH_Update
//-----------------------------------------------------------------------------
//
// This routine replaces <numbytes> from <src> to the FLASH addressed by
// <dest>.  This function calls FLASH_Clear() to handle the dirty work of
// initializing all <dest> bytes to 0xff's prior to copying the bytes from
// <src> to <dest>. This function accepts <numbytes> up to <FLASH_PAGESIZE>.
//
void FLASH_Update (FLADDR dest, char *src, unsigned numbytes, bit SFLE)
{
   // 1. Erase <numbytes> starting from <dest>
   FLASH_Clear (dest, numbytes, SFLE);

   // 2. Write <numbytes> from <src> to <dest>
   FLASH_Write (dest, src, numbytes, SFLE);
}



//-----------------------------------------------------------------------------
// FLASH_Write
//-----------------------------------------------------------------------------
//
// This routine copies <numbytes> from <src> to the linear FLASH address
```

```
// <dest>.
//
void FLASH_Write (FLADDR dest, char *src, unsigned numbytes, bit SFLE)
{
   FLADDR i;

   for (i = dest; i < dest+numbytes; i++) {
      FLASH_ByteWrite (i, *src++, SFLE);
   }
}

//-----------------------------------------------------------------------------
// FLASH_Read
//-----------------------------------------------------------------------------
//
// This routine copies <numbytes> from the linear FLASH address <src> to
// <dest>.
//
char * FLASH_Read (char *dest, FLADDR src, unsigned numbytes, bit SFLE)
{
   FLADDR i;

   for (i = 0; i < numbytes; i++) {
      *dest++ = FLASH_ByteRead (src+i, SFLE);
   }
   return dest;
}

//-----------------------------------------------------------------------------
// FLASH_Copy
//-----------------------------------------------------------------------------
//
// This routine copies <numbytes> from <src> to the linear FLASH address
// <dest>.
//
void FLASH_Copy (FLADDR dest, bit destSFLE, FLADDR src, bit srcSFLE,
                 unsigned numbytes)
{
   FLADDR i;

   for (i = 0; i < numbytes; i++) {

      FLASH_ByteWrite ((FLADDR) dest+i,
                       FLASH_ByteRead((FLADDR) src+i, srcSFLE),
                       destSFLE);
   }
}


//-----------------------------------------------------------------------------
// FLASH_Fill
//-----------------------------------------------------------------------------
//
// This routine fills the FLASH beginning at <addr> with <lenght> bytes.
//
void FLASH_Fill (FLADDR addr, ULONG length, unsigned char fill, bit SFLE)
{
   FLADDR i;

   for (i = 0; i < length; i++) {
      FLASH_ByteWrite (addr+i, fill, SFLE);
   }
}
```

7.3.4. F040_FlashUtils.h

```
//-----------------------------------------------------------------------------
// F040_FlashUtils.h
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F04x
// Tool chain: KEIL C51 7.06
//
// Release 1.0
//

#ifndef F040_FLASHUTILS_H
#define F040_FLASHUTILS_H

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F040_FlashPrimitives.h"

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Exported Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
extern void FLASH_Write (FLADDR dest, char *src, unsigned numbytes, bit SFLE);
extern char * FLASH_Read (char *dest, FLADDR src, unsigned numbytes, bit SFLE);
extern void FLASH_Clear (FLADDR addr, unsigned numbytes, bit SFLE);

// FLASH update/copy routines
extern void FLASH_Update (FLADDR dest, char *src, unsigned numbytes, bit SFLE);
extern void FLASH_Copy (FLADDR dest, bit destSFLE, FLADDR src, bit srcSFLE,
                 unsigned numbytes);

// FLASH test routines
extern void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill, bit SFLE);

#endif // F040_FLASHUTILS_H
```

7.4. ‘F060

7.4.1. F060_FlashPrimitives.c

```
//-----------------------------------------------------------------------------
// F060_FlashPrimitives.c
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F06x
// Tool chain: KEIL C51 7.06
//
// Release 1.0
//

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F060_FlashPrimitives.h"
#include <c8051F060.h>

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
void FLASH_ByteWrite (FLADDR addr, char byte, bit SFLE);
unsigned char FLASH_ByteRead (FLADDR addr, bit SFLE);
void FLASH_PageErase (FLADDR addr, bit SFLE);

//-----------------------------------------------------------------------------
// Global Variables
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH Routines
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH_ByteWrite
//-----------------------------------------------------------------------------
//
// This routine writes <byte> to the linear FLASH address <addr>.
// Linear map is decoded as follows:
// Linear Address       Device Address
// ----------------------------------------------
// 0x00000 - 0x0FFFF    0x0000 - 0xFFFF
//
void FLASH_ByteWrite (FLADDR addr, char byte, bit SFLE)
{
```

```
   char SFRPAGE_SAVE = SFRPAGE;        // preserve SFRPAGE
   bit EA_SAVE = EA;                   // preserve EA
   char xdata * data pwrite;           // FLASH write pointer

   EA = 0;                             // disable interrupts

   pwrite = (char xdata *) addr;       // initialize write pointer

   SFRPAGE = LEGACY_PAGE;

   FLSCL |= 0x01;                      // enable FLASH writes/erases
   PSCTL |= 0x01;                      // PSWE = 1

   if (SFLE) {
      PSCTL |= 0x04;                   // set SFLE
   }

   RSTSRC = 0x02;                      // enable VDDMON as reset source
   *pwrite = byte;                     // write the byte

   if (SFLE) {
      PSCTL &= ~0x04;                  // clear SFLE
   }
   PSCTL &= ~0x01;                     // PSWE = 0
   FLSCL &= ~0x01;                     // disable FLASH writes/erases

   SFRPAGE = SFRPAGE_SAVE;             // restore SFRPAGE
   EA = EA_SAVE;                       // restore interrupts
}

//-----------------------------------------------------------------------------
// FLASH_ByteRead
//-----------------------------------------------------------------------------
//
// This routine reads a <byte> from the linear FLASH address <addr>.
//
unsigned char FLASH_ByteRead (FLADDR addr, bit SFLE)
{
   char SFRPAGE_SAVE = SFRPAGE;        // preserve SFRPAGE
   bit EA_SAVE = EA;                   // preserve EA
   char code * data pread;             // FLASH read pointer
   unsigned char byte;

   pread = (char code *) addr;         // initialize read pointer

   EA = 0;                             // disable interrupts

   SFRPAGE = LEGACY_PAGE;

   if (SFLE) {
      PSCTL |= 0x04;                   // set SFLE
   }

   byte = *pread;                      // read the byte

   if (SFLE) {
      PSCTL &= ~0x04;                  // clear SFLE
   }

   SFRPAGE = SFRPAGE_SAVE;             // restore SFRPAGE
   EA = EA_SAVE;                       // restore interrupts

   return byte;
}
```

```
//-----------------------------------------------------------------------------
// FLASH_PageErase
//-----------------------------------------------------------------------------
//
// This routine erases the FLASH page containing the linear FLASH address
// <addr>.
//
void FLASH_PageErase (FLADDR addr, bit SFLE)
{
   char SFRPAGE_SAVE = SFRPAGE;        // preserve SFRPAGE
   bit EA_SAVE = EA;                   // preserve EA
   char xdata * data pwrite;           // FLASH write pointer

   EA = 0;                             // disable interrupts

   pwrite = (char xdata *) addr;       // initialize erase pointer

   SFRPAGE = LEGACY_PAGE;

   FLSCL |= 0x01;                      // enable FLASH writes/erases
   PSCTL |= 0x03;                      // PSWE = 1; PSEE = 1

   if (SFLE) {
      PSCTL |= 0x04;                   // set SFLE
   }

   RSTSRC = 0x02;                      // enable VDDMON as reset source
   *pwrite = 0;                        // initiate page erase

   if (SFLE) {
      PSCTL &= ~0x04;                  // clear SFLE
   }

   PSCTL &= ~0x03;                     // PSWE = 0; PSEE = 0
   FLSCL &= ~0x01;                     // disable FLASH writes/erases

   SFRPAGE = SFRPAGE_SAVE;             // restore SFRPAGE
   EA = EA_SAVE;                       // restore interrupts
}
```

7.4.2. F060_FlashPrimitives.h

```
//-----------------------------------------------------------------------------
// F060_FlashPrimitives.h
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F06x
// Tool chain: KEIL C51 7.06
//
// Release 1.0
//

#ifndef F060_FLASHPRIMITIVES_H
#define F060_FLASHPRIMITIVES_H

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

typedef unsigned long ULONG;
typedef unsigned int UINT;
typedef unsigned char UCHAR;

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

#ifndef FLASH_PAGESIZE
#define FLASH_PAGESIZE 512
#endif

#ifndef FLASH_SCRATCHSIZE
#define FLASH_SCRATCHSIZE 128
#endif

#ifndef FLASH_TEMP
#define FLASH_TEMP 0x0F800L
#endif

#ifndef FLASH_LAST
#define FLASH_LAST 0x0FA00L            // last page of FLASH
#endif

typedef UINT FLADDR;

//-----------------------------------------------------------------------------
// Exported Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
extern void FLASH_ByteWrite (FLADDR addr, char byte, bit SFLE);
extern unsigned char FLASH_ByteRead (FLADDR addr, bit SFLE);
extern void FLASH_PageErase (FLADDR addr, bit SFLE);

#endif // F060_FLASHPRIMITIVES_H
```

7.4.3. F060_FlashUtils.c

```
//-----------------------------------------------------------------------------
// F060_FlashUtils.c
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F06x
// Tool chain: KEIL C51 7.06
//
// Release 1.0
//

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F060_FlashPrimitives.h"
#include "F060_FlashUtils.h"

#include <stdio.h>

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
void FLASH_Write (FLADDR dest, char *src, unsigned numbytes, bit SFLE);
char * FLASH_Read (char *dest, FLADDR src, unsigned numbytes, bit SFLE);
void FLASH_Clear (FLADDR addr, unsigned numbytes, bit SFLE);

// FLASH update/copy routines
void FLASH_Update (FLADDR dest, char *src, unsigned numbytes, bit SFLE);
void FLASH_Copy (FLADDR dest, bit destSFLE, FLADDR src, bit srcSFLE,
                 unsigned numbytes);

// FLASH test routines
void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill, bit SFLE);
void FLASH_Print (FLADDR addr, ULONG length, bit SFLE);

//-----------------------------------------------------------------------------
// Global Variables
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH Routines
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH_Clear
//-----------------------------------------------------------------------------
```

```
//
// This routine erases <numbytes> starting from the FLASH addressed by
// <dest> by performing a read-modify-write operation using <FLASH_TEMP> as
// a temporary holding area.  This function accepts <numbytes> up to
// <FLASH_PAGESIZE>.
//
void FLASH_Clear (FLADDR dest, unsigned numbytes, bit SFLE)
{
   FLADDR dest_1_page_start;           // first address in 1st page
                                       // containing <dest>
   FLADDR dest_1_page_end;             // last address in 1st page
                                       // containing <dest>
   FLADDR dest_2_page_start;           // first address in 2nd page
                                       // containing <dest>
   FLADDR dest_2_page_end;             // last address in 2nd page
                                       // containing <dest>
   unsigned numbytes_remainder;        // when crossing page boundary,
                                       // number of <src> bytes on 2nd page
   unsigned FLASH_pagesize;            // size of FLASH page to update

   FLADDR  wptr;                       // write address
   FLADDR  rptr;                       // read address

   unsigned length;

   if (SFLE) {                         // update Scratchpad
      FLASH_pagesize = FLASH_SCRATCHSIZE;
   } else {
      FLASH_pagesize = FLASH_PAGESIZE;
   }

   dest_1_page_start = dest & ~(FLASH_pagesize - 1);
   dest_1_page_end = dest_1_page_start + FLASH_pagesize - 1;
   dest_2_page_start = (dest + numbytes)  & ~(FLASH_pagesize - 1);
   dest_2_page_end = dest_2_page_start + FLASH_pagesize - 1;

   if (dest_1_page_end == dest_2_page_end) {

      // 1. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP, 0);

      // 2. Copy bytes from first byte of dest page to dest-1 to Scratch page

      wptr = FLASH_TEMP;
      rptr = dest_1_page_start;
      length = dest - dest_1_page_start;
      FLASH_Copy (wptr, 0, rptr, SFLE, length);

      // 3. Copy from (dest+numbytes) to dest_page_end to Scratch page

      wptr = FLASH_TEMP + dest - dest_1_page_start + numbytes;
      rptr = dest + numbytes;
      length = dest_1_page_end - dest - numbytes + 1;
      FLASH_Copy (wptr, 0, rptr, SFLE, length);

      // 4. Erase destination page
      FLASH_PageErase (dest_1_page_start, SFLE);

      // 5. Copy Scratch page to destination page
      wptr = dest_1_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, SFLE, rptr, 0, length);
```

```
   } else {                            // value crosses page boundary
      // 1. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP, 0);

      // 2. Copy bytes from first byte of dest page to dest-1 to Scratch page

      wptr = FLASH_TEMP;
      rptr = dest_1_page_start;
      length = dest - dest_1_page_start;
      FLASH_Copy (wptr, 0, rptr, SFLE, length);

      // 3. Erase destination page 1
      FLASH_PageErase (dest_1_page_start, SFLE);

      // 4. Copy Scratch page to destination page 1
      wptr = dest_1_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, SFLE, rptr, 0, length);

      // now handle 2nd page

      // 5. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP, 0);

      // 6. Copy bytes from numbytes remaining to dest-2_page_end to Scratch page

      numbytes_remainder = numbytes - (dest_1_page_end - dest + 1);
      wptr = FLASH_TEMP + numbytes_remainder;
      rptr = dest_2_page_start + numbytes_remainder;
      length = FLASH_pagesize - numbytes_remainder;
      FLASH_Copy (wptr, 0, rptr, SFLE, length);

      // 7. Erase destination page 2
      FLASH_PageErase (dest_2_page_start, SFLE);

      // 8. Copy Scratch page to destination page 2
      wptr = dest_2_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, SFLE, rptr, 0, length);
   }
}

//-----------------------------------------------------------------------------
// FLASH_Update
//-----------------------------------------------------------------------------
//
// This routine replaces <numbytes> from <src> to the FLASH addressed by
// <dest>.  This function calls FLASH_Clear() to handle the dirty work of
// initializing all <dest> bytes to 0xff's prior to copying the bytes from
// <src> to <dest>. This function accepts <numbytes> up to <FLASH_PAGESIZE>.
//
void FLASH_Update (FLADDR dest, char *src, unsigned numbytes, bit SFLE)
{
   // 1. Erase <numbytes> starting from <dest>
   FLASH_Clear (dest, numbytes, SFLE);

   // 2. Write <numbytes> from <src> to <dest>
   FLASH_Write (dest, src, numbytes, SFLE);
}


//-----------------------------------------------------------------------------
```

```
// FLASH_Write
//-----------------------------------------------------------------------------
//
// This routine copies <numbytes> from <src> to the linear FLASH address
// <dest>.
//
void FLASH_Write (FLADDR dest, char *src, unsigned numbytes, bit SFLE)
{
   FLADDR i;

   for (i = dest; i < dest+numbytes; i++) {
      FLASH_ByteWrite (i, *src++, SFLE);
   }
}

//-----------------------------------------------------------------------------
// FLASH_Read
//-----------------------------------------------------------------------------
//
// This routine copies <numbytes> from the linear FLASH address <src> to
// <dest>.
//
char * FLASH_Read (char *dest, FLADDR src, unsigned numbytes, bit SFLE)
{
   FLADDR i;

   for (i = 0; i < numbytes; i++) {
      *dest++ = FLASH_ByteRead (src+i, SFLE);
   }
   return dest;
}

//-----------------------------------------------------------------------------
// FLASH_Copy
//-----------------------------------------------------------------------------
//
// This routine copies <numbytes> from <src> to the linear FLASH address
// <dest>.
//
void FLASH_Copy (FLADDR dest, bit destSFLE, FLADDR src, bit srcSFLE,
                 unsigned numbytes)
{
   FLADDR i;

   for (i = 0; i < numbytes; i++) {

      FLASH_ByteWrite ((FLADDR) dest+i,
                       FLASH_ByteRead((FLADDR) src+i, srcSFLE),
                       destSFLE);
   }
}


//-----------------------------------------------------------------------------
// FLASH_Fill
//-----------------------------------------------------------------------------
//
// This routine fills the FLASH beginning at <addr> with <lenght> bytes.
//
void FLASH_Fill (FLADDR addr, ULONG length, unsigned char fill, bit SFLE)
{
   FLADDR i;

   for (i = 0; i < length; i++) {
```

```
      FLASH_ByteWrite (addr+i, fill, SFLE);
   }
}

//-----------------------------------------------------------------------------
// FLASH_Print
//-----------------------------------------------------------------------------
//
// This routine prints <length> bytes from the FLASH beginning at <addr>.
//
void FLASH_Print (FLADDR addr, ULONG length, bit SFLE)
{
   FLADDR i;
   unsigned char me;

   for (i = 0; i < length; i++) {
      me = FLASH_ByteRead (addr+i, SFLE);
      if ((addr+i) % 16 == 0) {
         if (sizeof (FLADDR) == 4)
            printf ("\n%05lx: %02x ", (addr+i), (unsigned) me);
         else
            printf ("\n%05x: %02x ", (addr+i), (unsigned) me);
      } else {
         printf ("%02x ", (unsigned) me);
      }
   }
}
```

7.4.4. F060_FlashUtils.h

```
//-----------------------------------------------------------------------------
// F060_FlashUtils.h
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F06x
// Tool chain: KEIL C51 7.06
//
// Release 1.0
//

#ifndef F060_FLASHUTILS_H
#define F060_FLASHUTILS_H

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F060_FlashPrimitives.h"

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Exported Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
extern void FLASH_Write (FLADDR dest, char *src, unsigned numbytes, bit SFLE);
extern char * FLASH_Read (char *dest, FLADDR src, unsigned numbytes, bit SFLE);
extern void FLASH_Clear (FLADDR addr, unsigned numbytes, bit SFLE);

// FLASH update/copy routines
extern void FLASH_Update (FLADDR dest, char *src, unsigned numbytes, bit SFLE);
extern void FLASH_Copy (FLADDR dest, bit destSFLE, FLADDR src, bit srcSFLE,
                 unsigned numbytes);

// FLASH test routines
extern void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill, bit SFLE);
extern void FLASH_Print (FLADDR addr, ULONG length, bit SFLE);

#endif // F060_FLASHUTILS_H
```

7.5. ‘F120

7.5.1. F120_FlashPrimitives.c

```
//-----------------------------------------------------------------------------
// F120_FlashPrimitives.c
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F12x
// Tool chain: KEIL C51 7.06
//
// Release 1.1
// -Upgrading release version due to change in FlashPrimitives.h
// -07 FEB 2006 (GP)
//
// Release 1.0
//

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F120_FlashPrimitives.h"
#include <c8051F120.h>

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
void FLASH_ByteWrite (FLADDR addr, char byte, bit SFLE);
unsigned char FLASH_ByteRead (FLADDR addr, bit SFLE);
void FLASH_PageErase (FLADDR addr, bit SFLE);

//-----------------------------------------------------------------------------
// Global Variables
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH Routines
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH_ByteWrite
//-----------------------------------------------------------------------------
//
// This routine writes <byte> to the linear FLASH address <addr>.
// Linear map is decoded as follows:
// Linear Address       Bank     Bank Address
// ------------------------------------------------
```

```
// 0x00000 - 0x07FFF    0        0x0000 - 0x7FFF
// 0x08000 - 0x0FFFF    1        0x8000 - 0xFFFF
// 0x10000 - 0x17FFF    2        0x8000 - 0xFFFF
// 0x18000 - 0x1FFFF    3        0x8000 - 0xFFFF
//
void FLASH_ByteWrite (FLADDR addr, char byte, bit SFLE)
{
   char SFRPAGE_SAVE = SFRPAGE;        // preserve SFRPAGE
   bit EA_SAVE = EA;                   // preserve EA
   char PSBANK_SAVE = PSBANK;          // preserve PSBANK
   char xdata * data pwrite;           // FLASH write pointer

   EA = 0;                             // disable interrupts

   SFRPAGE = LEGACY_PAGE;

   if (addr < 0x10000) {               // 64K linear address
      pwrite = (char xdata *) addr;
   } else if (addr < 0x18000) {        // BANK 2
      addr |= 0x8000;
      pwrite = (char xdata *) addr;
      PSBANK &= ~0x30;                 // COBANK = 0x2
      PSBANK |=  0x20;
   } else {                            // BANK 3
      pwrite = (char xdata *) addr;
      PSBANK &= ~0x30;                 // COBANK = 0x3
      PSBANK |=  0x30;
   }

   FLSCL |= 0x01;                      // enable FLASH writes/erases
   PSCTL |= 0x01;                      // PSWE = 1

   if (SFLE) {
      PSCTL |= 0x04;                   // set SFLE
   }

   RSTSRC = 0x02;                      // enable VDDMON as reset source
   *pwrite = byte;                     // write the byte

   if (SFLE) {
      PSCTL &= ~0x04;                  // clear SFLE
   }
   PSCTL &= ~0x01;                     // PSWE = 0
   FLSCL &= ~0x01;                     // disable FLASH writes/erases

   PSBANK = PSBANK_SAVE;               // restore PSBANK
   SFRPAGE = SFRPAGE_SAVE;             // restore SFRPAGE
   EA = EA_SAVE;                       // restore interrupts
}

//-----------------------------------------------------------------------------
// FLASH_ByteRead
//-----------------------------------------------------------------------------
//
// This routine reads a <byte> from the linear FLASH address <addr>.
//
unsigned char FLASH_ByteRead (FLADDR addr, bit SFLE)
{
   char SFRPAGE_SAVE = SFRPAGE;        // preserve SFRPAGE
   bit EA_SAVE = EA;                   // preserve EA
   char PSBANK_SAVE = PSBANK;          // preserve PSBANK
   char code * data pread;             // FLASH read pointer
   unsigned char byte;
```

```
   EA = 0;                              // disable interrupts

   SFRPAGE = LEGACY_PAGE;

   if (addr < 0x10000) {                // 64K linear address
      pread = (char code *) addr;
   } else if (addr < 0x18000) {         // BANK 2
      addr |= 0x8000;
      pread = (char code *) addr;
      PSBANK &= ~0x30;                  // COBANK = 0x2
      PSBANK |=  0x20;
   } else {                             // BANK 3
      pread = (char code *) addr;
      PSBANK &= ~0x30;                  // COBANK = 0x3
      PSBANK |=  0x30;
   }

   if (SFLE) {
      PSCTL |= 0x04;                    // set SFLE
   }

   byte = *pread;                       // read the byte

   if (SFLE) {
      PSCTL &= ~0x04;                   // clear SFLE
   }

   PSBANK = PSBANK_SAVE;                // restore PSBANK
   SFRPAGE = SFRPAGE_SAVE;              // restore SFRPAGE
   EA = EA_SAVE;                        // restore interrupts

   return byte;
}

//-----------------------------------------------------------------------------
// FLASH_PageErase
//-----------------------------------------------------------------------------
//
// This routine erases the FLASH page containing the linear FLASH address
// <addr>.
//
void FLASH_PageErase (FLADDR addr, bit SFLE)
{
   char SFRPAGE_SAVE = SFRPAGE;         // preserve SFRPAGE
   bit EA_SAVE = EA;                    // preserve EA
   char PSBANK_SAVE = PSBANK;           // preserve PSBANK
   char xdata * data pwrite;            // FLASH write pointer

   EA = 0;                              // disable interrupts

   SFRPAGE = LEGACY_PAGE;

   if (addr < 0x10000) {                // 64K linear address
      pwrite = (char xdata *) addr;
   } else if (addr < 0x18000) {         // BANK 2
      addr |= 0x8000;
      pwrite = (char xdata *) addr;
      PSBANK &= ~0x30;                  // COBANK = 0x2
      PSBANK |=  0x20;
   } else {                             // BANK 3
      pwrite = (char xdata *) addr;
      PSBANK &= ~0x30;                  // COBANK = 0x3
      PSBANK |=  0x30;
   }
```

```
   FLSCL |= 0x01;                      // enable FLASH writes/erases
   PSCTL |= 0x03;                      // PSWE = 1; PSEE = 1

   if (SFLE) {
      PSCTL |= 0x04;                   // set SFLE
   }

   RSTSRC = 0x02;                      // enable VDDMON as reset source
   *pwrite = 0;                        // initiate page erase

   if (SFLE) {
      PSCTL &= ~0x04;                  // clear SFLE
   }

   PSCTL &= ~0x03;                     // PSWE = 0; PSEE = 0
   FLSCL &= ~0x01;                     // disable FLASH writes/erases

   PSBANK = PSBANK_SAVE;               // restore PSBANK
   SFRPAGE = SFRPAGE_SAVE;             // restore SFRPAGE
   EA = EA_SAVE;                       // restore interrupts
}
```

7.5.2. F120_FlashPrimitives.h

```
//-----------------------------------------------------------------------------
// F120_FlashPrimitives.h
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F12x
// Tool chain: KEIL C51 7.06
//
//
// Release 1.1
// -Change typecast of FLASH_PAGESIZE to 1024L to fix bug in Flash_Clear()
// -07 FEB 2006 (GP)
//
// Release 1.0
//

#ifndef F120_FLASHPRIMITIVES_H
#define F120_FLASHPRIMITIVES_H

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------
typedef unsigned long ULONG;
typedef unsigned int UINT;
typedef unsigned char UCHAR;

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

#ifndef FLASH_PAGESIZE
#define FLASH_PAGESIZE 1024L
#endif

#ifndef FLASH_SCRATCHSIZE
#define FLASH_SCRATCHSIZE 256
#endif

#ifndef FLASH_TEMP
#define FLASH_TEMP 0x1F400L            // address of temp page
#endif

#ifndef FLASH_LAST
#define FLASH_LAST 0x1F800L            // last page of FLASH
#endif

typedef ULONG FLADDR;


//-----------------------------------------------------------------------------
// Exported Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
extern void FLASH_ByteWrite (FLADDR addr, char byte, bit SFLE);
extern unsigned char FLASH_ByteRead (FLADDR addr, bit SFLE);
extern void FLASH_PageErase (FLADDR addr, bit SFLE);

#endif // F120_FLASHPRIMITIVES_H
```

7.5.3. F120_FlashUtils.c

```
//-----------------------------------------------------------------------------
// F120_FlashUtils.c
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F12x
// Tool chain: KEIL C51 7.06
//
// Release 1.1
// -Upgrading release version due to change in FlashPrimitives.h
// -07 FEB 2006 (GP)
//
// Release 1.0
//

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F120_FlashPrimitives.h"
#include "F120_FlashUtils.h"

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
void FLASH_Write (FLADDR dest, char *src, unsigned numbytes, bit SFLE);
char * FLASH_Read (char *dest, FLADDR src, unsigned numbytes, bit SFLE);
void FLASH_Clear (FLADDR addr, unsigned numbytes, bit SFLE);

// FLASH update/copy routines
void FLASH_Update (FLADDR dest, char *src, unsigned numbytes, bit SFLE);
void FLASH_Copy (FLADDR dest, bit destSFLE, FLADDR src, bit srcSFLE,
                 unsigned numbytes);

// FLASH test routines
void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill, bit SFLE);

//-----------------------------------------------------------------------------
// Global Variables
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH Routines
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH_Clear
//-----------------------------------------------------------------------------
```

```
//
// This routine erases <numbytes> starting from the FLASH addressed by
// <dest> by performing a read-modify-write operation using <FLASH_TEMP> as
// a temporary holding area.  This function accepts <numbytes> up to
// <FLASH_PAGESIZE>.
//
void FLASH_Clear (FLADDR dest, unsigned numbytes, bit SFLE)
{
   FLADDR dest_1_page_start;           // first address in 1st page
                                       // containing <dest>
   FLADDR dest_1_page_end;             // last address in 1st page
                                       // containing <dest>
   FLADDR dest_2_page_start;           // first address in 2nd page
                                       // containing <dest>
   FLADDR dest_2_page_end;             // last address in 2nd page
                                       // containing <dest>
   unsigned numbytes_remainder;        // when crossing page boundary,
                                       // number of <src> bytes on 2nd page
   unsigned FLASH_pagesize;            // size of FLASH page to update

   FLADDR  wptr;                       // write address
   FLADDR  rptr;                       // read address

   unsigned length;

   if (SFLE) {                         // update Scratchpad
      FLASH_pagesize = FLASH_SCRATCHSIZE;
   } else {
      FLASH_pagesize = FLASH_PAGESIZE;
   }

   dest_1_page_start = dest & ~(FLASH_pagesize - 1);
   dest_1_page_end = dest_1_page_start + FLASH_pagesize - 1;
   dest_2_page_start = (dest + numbytes)  & ~(FLASH_pagesize - 1);
   dest_2_page_end = dest_2_page_start + FLASH_pagesize - 1;

   if (dest_1_page_end == dest_2_page_end) {

      // 1. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP, 0);

      // 2. Copy bytes from first byte of dest page to dest-1 to Scratch page

      wptr = FLASH_TEMP;
      rptr = dest_1_page_start;
      length = dest - dest_1_page_start;
      FLASH_Copy (wptr, 0, rptr, SFLE, length);

      // 3. Copy from (dest+numbytes) to dest_page_end to Scratch page

      wptr = FLASH_TEMP + dest - dest_1_page_start + numbytes;
      rptr = dest + numbytes;
      length = dest_1_page_end - dest - numbytes + 1;
      FLASH_Copy (wptr, 0, rptr, SFLE, length);

      // 4. Erase destination page
      FLASH_PageErase (dest_1_page_start, SFLE);

      // 5. Copy Scratch page to destination page
      wptr = dest_1_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, SFLE, rptr, 0, length);

   } else {                            // value crosses page boundary
      // 1. Erase Scratch page
```

```
      FLASH_PageErase (FLASH_TEMP, 0);

      // 2. Copy bytes from first byte of dest page to dest-1 to Scratch page

      wptr = FLASH_TEMP;
      rptr = dest_1_page_start;
      length = dest - dest_1_page_start;
      FLASH_Copy (wptr, 0, rptr, SFLE, length);

      // 3. Erase destination page 1
      FLASH_PageErase (dest_1_page_start, SFLE);

      // 4. Copy Scratch page to destination page 1
      wptr = dest_1_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, SFLE, rptr, 0, length);

      // now handle 2nd page

      // 5. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP, 0);

      // 6. Copy bytes from numbytes remaining to dest-2_page_end to Scratch page

      numbytes_remainder = numbytes - (dest_1_page_end - dest + 1);
      wptr = FLASH_TEMP + numbytes_remainder;
      rptr = dest_2_page_start + numbytes_remainder;
      length = FLASH_pagesize - numbytes_remainder;
      FLASH_Copy (wptr, 0, rptr, SFLE, length);

      // 7. Erase destination page 2
      FLASH_PageErase (dest_2_page_start, SFLE);

      // 8. Copy Scratch page to destination page 2
      wptr = dest_2_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, SFLE, rptr, 0, length);
   }
}



//-----------------------------------------------------------------------------
// FLASH_Update
//-----------------------------------------------------------------------------
//
// This routine replaces <numbytes> from <src> to the FLASH addressed by
// <dest>.  This function calls FLASH_Clear() to handle the dirty work of
// initializing all <dest> bytes to 0xff's prior to copying the bytes from
// <src> to <dest>. This function accepts <numbytes> up to <FLASH_PAGESIZE>.
//
void FLASH_Update (FLADDR dest, char *src, unsigned numbytes, bit SFLE)
{
   // 1. Erase <numbytes> starting from <dest>
   FLASH_Clear (dest, numbytes, SFLE);

   // 2. Write <numbytes> from <src> to <dest>
   FLASH_Write (dest, src, numbytes, SFLE);
}



//-----------------------------------------------------------------------------
// FLASH_Write
```

```
//-----------------------------------------------------------------------------
//
// This routine copies <numbytes> from <src> to the linear FLASH address
// <dest>.
//
void FLASH_Write (FLADDR dest, char *src, unsigned numbytes, bit SFLE)
{
   FLADDR i;

   for (i = dest; i < dest+numbytes; i++) {
      FLASH_ByteWrite (i, *src++, SFLE);
   }
}

//-----------------------------------------------------------------------------
// FLASH_Read
//-----------------------------------------------------------------------------
//
// This routine copies <numbytes> from the linear FLASH address <src> to
// <dest>.
//
char * FLASH_Read (char *dest, FLADDR src, unsigned numbytes, bit SFLE)
{
   FLADDR i;

   for (i = 0; i < numbytes; i++) {
      *dest++ = FLASH_ByteRead (src+i, SFLE);
   }
   return dest;
}

//-----------------------------------------------------------------------------
// FLASH_Copy
//-----------------------------------------------------------------------------
//
// This routine copies <numbytes> from <src> to the linear FLASH address
// <dest>.
//
void FLASH_Copy (FLADDR dest, bit destSFLE, FLADDR src, bit srcSFLE,
                 unsigned numbytes)
{
   FLADDR i;

   for (i = 0; i < numbytes; i++) {

      FLASH_ByteWrite ((FLADDR) dest+i,
                       FLASH_ByteRead((FLADDR) src+i, srcSFLE),
                       destSFLE);
   }
}


//-----------------------------------------------------------------------------
// FLASH_Fill
//-----------------------------------------------------------------------------
//
// This routine fills the FLASH beginning at <addr> with <lenght> bytes.
//
void FLASH_Fill (FLADDR addr, ULONG length, unsigned char fill, bit SFLE)
{
   FLADDR i;

   for (i = 0; i < length; i++) {
      FLASH_ByteWrite (addr+i, fill, SFLE);
   }
}
```

7.5.4. F120_FlashUtils.h

```
//-----------------------------------------------------------------------------
// F120_FlashUtils.h
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F12x
// Tool chain: KEIL C51 7.06
//
// Release 1.1
// -Upgrading release version due to change in FlashPrimitives.h
// -07 FEB 2006 (GP)
//
// Release 1.0
//

#ifndef F120_FLASHUTILS_H
#define F120_FLASHUTILS_H

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F120_FlashPrimitives.h"

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Exported Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
extern void FLASH_Write (FLADDR dest, char *src, unsigned numbytes, bit SFLE);
extern char * FLASH_Read (char *dest, FLADDR src, unsigned numbytes, bit SFLE);
extern void FLASH_Clear (FLADDR addr, unsigned numbytes, bit SFLE);

// FLASH update/copy routines
extern void FLASH_Update (FLADDR dest, char *src, unsigned numbytes, bit SFLE);
extern void FLASH_Copy (FLADDR dest, bit destSFLE, FLADDR src, bit srcSFLE,
                 unsigned numbytes);

// FLASH test routines
extern void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill, bit SFLE);

#endif // F120_FLASHUTILS_H
```

7.6. ‘F200

7.6.1. F200_FlashPrimitives.c

```
//-----------------------------------------------------------------------------
// F200_FlashPrimitives.c
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F2xx
// Tool chain: KEIL C51 7.06
//
// Release 1.0
//

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F200_FlashPrimitives.h"
#include <c8051F200.h>

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
void FLASH_ByteWrite (FLADDR addr, char byte);
unsigned char FLASH_ByteRead (FLADDR addr);
void FLASH_PageErase (FLADDR addr);

//-----------------------------------------------------------------------------
// Global Variables
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH Routines
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH_ByteWrite
//-----------------------------------------------------------------------------
//
// This routine writes <byte> to the linear FLASH address <addr>.
// Linear map is decoded as follows:
// Linear Address       Device Address
// ---------------------------------------------
// 0x00000 - 0x0FFFF    0x0000 - 0xFFFF
//
void FLASH_ByteWrite (FLADDR addr, char byte)
{
   bit EA_SAVE = EA;                   // preserve EA
   char xdata * data pwrite;           // FLASH write pointer
```

```
   EA = 0;                              // disable interrupts

   pwrite = (char xdata *) addr;        // initialize write pointer

   FLSCL  = FLASHSCALE;                 // enable FLASH writes/erases
   PSCTL |= 0x01;                       // PSWE = 1

   *pwrite = byte;                      // write the byte

   PSCTL &= ~0x01;                      // PSWE = 0
   FLSCL |=  0x0F;                      // disable FLASH writes/erases

   EA = EA_SAVE;                        // restore interrupts
}

//-----------------------------------------------------------------------------
// FLASH_ByteRead
//-----------------------------------------------------------------------------
//
// This routine reads a <byte> from the linear FLASH address <addr>.
//
unsigned char FLASH_ByteRead (FLADDR addr)
{
   bit EA_SAVE = EA;                    // preserve EA
   char code * data pread;              // FLASH read pointer
   unsigned char byte;

   pread = (char code *) addr;          // initialize read pointer

   EA = 0;                              // disable interrupts

   byte = *pread;                       // read the byte

   EA = EA_SAVE;                        // restore interrupts

   return byte;
}

//-----------------------------------------------------------------------------
// FLASH_PageErase
//-----------------------------------------------------------------------------
//
// This routine erases the FLASH page containing the linear FLASH address
// <addr>.
//
void FLASH_PageErase (FLADDR addr)
{
   bit EA_SAVE = EA;                    // preserve EA
   char xdata * data pwrite;            // FLASH write pointer

   EA = 0;                              // disable interrupts

   pwrite = (char xdata *) addr;        // initialize erase pointer

   FLSCL  = FLASHSCALE;                 // enable FLASH writes/erases
   PSCTL |= 0x03;                       // PSWE = 1; PSEE = 1

   *pwrite = 0;                         // initiate page erase

   PSCTL &= ~0x03;                      // PSWE = 0; PSEE = 0
   FLSCL |=  0x0F;                      // disable FLASH writes/erases

   EA = EA_SAVE;                        // restore interrupts
}
```

7.6.2. F200_FlashPrimitives.h

```
//-----------------------------------------------------------------------------
// F200_FlashPrimitives.h
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F2xx
// Tool chain: KEIL C51 7.06
//
// Release 1.0
//

#ifndef F200_FLASHPRIMITIVES_H
#define F200_FLASHPRIMITIVES_H

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

typedef unsigned long ULONG;
typedef unsigned int UINT;
typedef unsigned char UCHAR;

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

#ifndef SYSCLK
#define SYSCLK 16000000L
#endif

#ifndef FLASHSCALE

#if   (SYSCLK < 50000L)
#define FLASHSCALE 0x80
#elif (SYSCLK < 100000L)
#define FLASHSCALE 0x81
#elif (SYSCLK < 200000L)
#define FLASHSCALE 0x82
#elif (SYSCLK < 400000L)
#define FLASHSCALE 0x83
#elif (SYSCLK < 800000L)
#define FLASHSCALE 0x84
#elif (SYSCLK < 1600000L)
#define FLASHSCALE 0x85
#elif (SYSCLK < 3200000L)
#define FLASHSCALE 0x86
#elif (SYSCLK < 6400000L)
#define FLASHSCALE 0x87
#elif (SYSCLK < 12800000L)
#define FLASHSCALE 0x88
#elif (SYSCLK < 25600000L)
#define FLASHSCALE 0x89
#endif // SYSCLK test
```

```
#endif // FLASHSCALE

#ifndef FLASH_PAGESIZE
#define FLASH_PAGESIZE 512
#endif

#ifndef FLASH_SCRATCHSIZE
#define FLASH_SCRATCHSIZE 128
#endif

#ifndef FLASH_TEMP
#define FLASH_TEMP 0x01800L
#endif

#ifndef FLASH_LAST
#define FLASH_LAST 0x01A00L            // last page of FLASH
#endif

typedef UINT FLADDR;

//-----------------------------------------------------------------------------
// Exported Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
extern void FLASH_ByteWrite (FLADDR addr, char byte);
extern unsigned char FLASH_ByteRead (FLADDR addr);
extern void FLASH_PageErase (FLADDR addr);

#endif // F200_FLASHPRIMITIVES_H
```

7.6.3. F200_FlashUtils.c

```
//-----------------------------------------------------------------------------
// F200_FlashUtils.c
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F2xx
// Tool chain: KEIL C51 7.06
//
// Release 1.0
//

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F200_FlashPrimitives.h"
#include "F200_FlashUtils.h"

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
void FLASH_Write (FLADDR dest, char *src, unsigned numbytes);
char * FLASH_Read (char *dest, FLADDR src, unsigned numbytes);
void FLASH_Clear (FLADDR addr, unsigned numbytes);

// FLASH update/copy routines
void FLASH_Update (FLADDR dest, char *src, unsigned numbytes);
void FLASH_Copy (FLADDR dest, FLADDR src, unsigned numbytes);

// FLASH test routines
void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill);

//-----------------------------------------------------------------------------
// Global Variables
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH Routines
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH_Clear
//-----------------------------------------------------------------------------
//
// This routine erases <numbytes> starting from the FLASH addressed by
// <dest> by performing a read-modify-write operation using <FLASH_TEMP> as
// a temporary holding area.  This function accepts <numbytes> up to
```

```
// <FLASH_PAGESIZE>.
//
void FLASH_Clear (FLADDR dest, unsigned numbytes)
{
   FLADDR dest_1_page_start;           // first address in 1st page
                                       // containing <dest>
   FLADDR dest_1_page_end;             // last address in 1st page
                                       // containing <dest>
   FLADDR dest_2_page_start;           // first address in 2nd page
                                       // containing <dest>
   FLADDR dest_2_page_end;             // last address in 2nd page
                                       // containing <dest>
   unsigned numbytes_remainder;        // when crossing page boundary,
                                       // number of <src> bytes on 2nd page
   unsigned FLASH_pagesize;            // size of FLASH page to update

   FLADDR  wptr;                       // write address
   FLADDR  rptr;                       // read address

   unsigned length;

   FLASH_pagesize = FLASH_PAGESIZE;

   dest_1_page_start = dest & ~(FLASH_pagesize - 1);
   dest_1_page_end = dest_1_page_start + FLASH_pagesize - 1;
   dest_2_page_start = (dest + numbytes)  & ~(FLASH_pagesize - 1);
   dest_2_page_end = dest_2_page_start + FLASH_pagesize - 1;

   if (dest_1_page_end == dest_2_page_end) {

      // 1. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP);

      // 2. Copy bytes from first byte of dest page to dest-1 to Scratch page

      wptr = FLASH_TEMP;
      rptr = dest_1_page_start;
      length = dest - dest_1_page_start;
      FLASH_Copy (wptr, rptr, length);

      // 3. Copy from (dest+numbytes) to dest_page_end to Scratch page

      wptr = FLASH_TEMP + dest - dest_1_page_start + numbytes;
      rptr = dest + numbytes;
      length = dest_1_page_end - dest - numbytes + 1;
      FLASH_Copy (wptr, rptr, length);

      // 4. Erase destination page
      FLASH_PageErase (dest_1_page_start);

      // 5. Copy Scratch page to destination page
      wptr = dest_1_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, rptr, length);

   } else {                            // value crosses page boundary
      // 1. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP);

      // 2. Copy bytes from first byte of dest page to dest-1 to Scratch page

      wptr = FLASH_TEMP;
      rptr = dest_1_page_start;
```

```
      length = dest - dest_1_page_start;
      FLASH_Copy (wptr, rptr, length);

      // 3. Erase destination page 1
      FLASH_PageErase (dest_1_page_start);

      // 4. Copy Scratch page to destination page 1
      wptr = dest_1_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, rptr, length);

      // now handle 2nd page

      // 5. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP);

      // 6. Copy bytes from numbytes remaining to dest-2_page_end to Scratch page

      numbytes_remainder = numbytes - (dest_1_page_end - dest + 1);
      wptr = FLASH_TEMP + numbytes_remainder;
      rptr = dest_2_page_start + numbytes_remainder;
      length = FLASH_pagesize - numbytes_remainder;
      FLASH_Copy (wptr, rptr, length);

      // 7. Erase destination page 2
      FLASH_PageErase (dest_2_page_start);

      // 8. Copy Scratch page to destination page 2
      wptr = dest_2_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, rptr, length);
   }
}



//-----------------------------------------------------------------------------
// FLASH_Update
//-----------------------------------------------------------------------------
//
// This routine replaces <numbytes> from <src> to the FLASH addressed by
// <dest>.  This function calls FLASH_Clear() to handle the dirty work of
// initializing all <dest> bytes to 0xff's prior to copying the bytes from
// <src> to <dest>. This function accepts <numbytes> up to <FLASH_PAGESIZE>.
//
void FLASH_Update (FLADDR dest, char *src, unsigned numbytes)
{
   // 1. Erase <numbytes> starting from <dest>
   FLASH_Clear (dest, numbytes);

   // 2. Write <numbytes> from <src> to <dest>
   FLASH_Write (dest, src, numbytes);
}



//-----------------------------------------------------------------------------
// FLASH_Write
//-----------------------------------------------------------------------------
//
// This routine copies <numbytes> from <src> to the linear FLASH address
// <dest>.
```

```
//
void FLASH_Write (FLADDR dest, char *src, unsigned numbytes)
{
   FLADDR i;

   for (i = dest; i < dest+numbytes; i++) {
      FLASH_ByteWrite (i, *src++);
   }
}

//-----------------------------------------------------------------------------
// FLASH_Read
//-----------------------------------------------------------------------------
//
// This routine copies <numbytes> from the linear FLASH address <src> to
// <dest>.
//
char * FLASH_Read (char *dest, FLADDR src, unsigned numbytes)
{
   FLADDR i;

   for (i = 0; i < numbytes; i++) {
      *dest++ = FLASH_ByteRead (src+i);
   }
   return dest;
}

//-----------------------------------------------------------------------------
// FLASH_Copy
//-----------------------------------------------------------------------------
//
// This routine copies <numbytes> from <src> to the linear FLASH address
// <dest>.
//
void FLASH_Copy (FLADDR dest, FLADDR src, unsigned numbytes)
{
   FLADDR i;

   for (i = 0; i < numbytes; i++) {

      FLASH_ByteWrite ((FLADDR) dest+i, FLASH_ByteRead((FLADDR) src+i));
   }
}



//-----------------------------------------------------------------------------
// FLASH_Fill
//-----------------------------------------------------------------------------
//
// This routine fills the FLASH beginning at <addr> with <lenght> bytes.
//
void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill)
{
   FLADDR i;

   for (i = 0; i < length; i++) {
      FLASH_ByteWrite (addr+i, fill);
   }
}
```

7.6.4. F200_FlashUtils.h

```
//-----------------------------------------------------------------------------
// F200_FlashUtils.h
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F2xx
// Tool chain: KEIL C51 7.06
//
// Release 1.0
//

#ifndef F200_FLASHUTILS_H
#define F200_FLASHUTILS_H

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F200_FlashPrimitives.h"

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Exported Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
extern void FLASH_Write (FLADDR dest, char *src, unsigned numbytes);
extern char * FLASH_Read (char *dest, FLADDR src, unsigned numbytes);
extern void FLASH_Clear (FLADDR addr, unsigned numbytes);

// FLASH update/copy routines
extern void FLASH_Update (FLADDR dest, char *src, unsigned numbytes);   // copy with destina-
tion preservation
extern void FLASH_Copy (FLADDR dest, FLADDR src, unsigned numbytes);     // low-level FLASH/
FLASH byte copy

// FLASH test routines
extern void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill);

#endif // F200_FLASHUTILS_H
```

7.7. ‘F300

7.7.1. F300_FlashPrimitives.c

```
//-----------------------------------------------------------------------------
// F300_FlashPrimitives.c
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F30x
// Tool chain: KEIL C51 7.06
//
// Release 1.0
//

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F300_FlashPrimitives.h"
#include <c8051F300.h>

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
void FLASH_ByteWrite (FLADDR addr, char byte);
unsigned char FLASH_ByteRead (FLADDR addr);
void FLASH_PageErase (FLADDR addr);

//-----------------------------------------------------------------------------
// Global Variables
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH Routines
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH_ByteWrite
//-----------------------------------------------------------------------------
//
// This routine writes <byte> to the linear FLASH address <addr>.
//
void FLASH_ByteWrite (FLADDR addr, char byte)
{
   bit EA_SAVE = EA;                   // preserve EA
   char xdata * data pwrite;           // FLASH write pointer

   EA = 0;                             // disable interrupts

   RSTSRC = 0x06;                      // enable VDD monitor as a reset source

   pwrite = (char xdata *) addr;
```

```
   FLKEY  = 0xA5;                       // Key Sequence 1
   FLKEY  = 0xF1;                       // Key Sequence 2
   PSCTL |= 0x01;                       // PSWE = 1

   RSTSRC = 0x06;                       // enable VDD monitor as a reset source

   *pwrite = byte;                      // write the byte

   PSCTL &= ~0x01;                      // PSWE = 0

   EA = EA_SAVE;                        // restore interrupts
}

//-----------------------------------------------------------------------------
// FLASH_ByteRead
//-----------------------------------------------------------------------------
//
// This routine reads a <byte> from the linear FLASH address <addr>.
//
unsigned char FLASH_ByteRead (FLADDR addr)
{
   bit EA_SAVE = EA;                    // preserve EA
   char code * data pread;              // FLASH read pointer
   unsigned char byte;

   EA = 0;                              // disable interrupts

   pread = (char code *) addr;

   byte = *pread;                       // read the byte

   EA = EA_SAVE;                        // restore interrupts

   return byte;
}

//-----------------------------------------------------------------------------
// FLASH_PageErase
//-----------------------------------------------------------------------------
//
// This routine erases the FLASH page containing the linear FLASH address
// <addr>.
//
void FLASH_PageErase (FLADDR addr)
{
   bit EA_SAVE = EA;                    // preserve EA
   char xdata * data pwrite;            // FLASH write pointer

   EA = 0;                             // disable interrupts


  RSTSRC = 0x06;                      // enable VDD monitor as a reset source

   pwrite = (char xdata *) addr;

   FLKEY  = 0xA5;                       // Key Sequence 1
   FLKEY  = 0xF1;                       // Key Sequence 2
   PSCTL |= 0x03;                       // PSWE = 1; PSEE = 1



  RSTSRC = 0x06;                      // enable VDD monitor as a reset source
   *pwrite = 0;                         // initiate page erase

   PSCTL &= ~0x03;                      // PSWE = 0; PSEE = 0

   EA = EA_SAVE;                        // restore interrupts
}
```

7.7.2. F300_FlashPrimitives.h

```
//-----------------------------------------------------------------------------
// F300_FlashPrimitives.h
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F30x
// Tool chain: KEIL C51 7.06
//
// Release 1.0
//

#ifndef F300_FLASHPRIMITIVES_H
#define F300_FLASHPRIMITIVES_H

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

typedef unsigned long ULONG;
typedef unsigned int UINT;
typedef unsigned char UCHAR;

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

#ifndef FLASH_PAGESIZE
#define FLASH_PAGESIZE 512
#endif

#ifndef FLASH_TEMP
#define FLASH_TEMP 0x01a00L
#endif

#ifndef FLASH_LAST
#define FLASH_LAST 0x01c00L
#endif

typedef UINT FLADDR;

//-----------------------------------------------------------------------------
// Exported Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
extern void FLASH_ByteWrite (FLADDR addr, char byte);
extern unsigned char FLASH_ByteRead (FLADDR addr);
extern void FLASH_PageErase (FLADDR addr);

#endif // F300_FLASHPRIMITIVES_H
```

7.7.3. F300_FlashUtils.c

```
//-----------------------------------------------------------------------------
// F300_FlashUtils.c
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F30x
// Tool chain: KEIL C51 7.06
//
// Release 1.0
//

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F300_FlashPrimitives.h"
#include "F300_FlashUtils.h"

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
void FLASH_Write (FLADDR dest, char *src, unsigned numbytes);
char * FLASH_Read (char *dest, FLADDR src, unsigned numbytes);
void FLASH_Clear (FLADDR addr, unsigned numbytes);

// FLASH update/copy routines
void FLASH_Update (FLADDR dest, char *src, unsigned numbytes);
void FLASH_Copy (FLADDR dest, FLADDR src, unsigned numbytes);

// FLASH test routines
void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill);

//-----------------------------------------------------------------------------
// Global Variables
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH Routines
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH_Clear
//-----------------------------------------------------------------------------
//
// This routine erases <numbytes> starting from the FLASH addressed by
// <dest> by performing a read-modify-write operation using <FLASH_TEMP> as
// a temporary holding area.  This function accepts <numbytes> up to
```

```
// <FLASH_PAGESIZE>.
//
void FLASH_Clear (FLADDR dest, unsigned numbytes)
{
   FLADDR dest_1_page_start;           // first address in 1st page
                                       // containing <dest>
   FLADDR dest_1_page_end;             // last address in 1st page
                                       // containing <dest>
   FLADDR dest_2_page_start;           // first address in 2nd page
                                       // containing <dest>
   FLADDR dest_2_page_end;             // last address in 2nd page
                                       // containing <dest>
   unsigned numbytes_remainder;        // when crossing page boundary,
                                       // number of <src> bytes on 2nd page
   unsigned FLASH_pagesize;            // size of FLASH page to update

   FLADDR  wptr;                       // write address
   FLADDR  rptr;                       // read address

   unsigned length;

   FLASH_pagesize = FLASH_PAGESIZE;

   dest_1_page_start = dest & ~(FLASH_pagesize - 1);
   dest_1_page_end = dest_1_page_start + FLASH_pagesize - 1;
   dest_2_page_start = (dest + numbytes)  & ~(FLASH_pagesize - 1);
   dest_2_page_end = dest_2_page_start + FLASH_pagesize - 1;

   if (dest_1_page_end == dest_2_page_end) {

      // 1. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP);

      // 2. Copy bytes from first byte of dest page to dest-1 to Scratch page

      wptr = FLASH_TEMP;
      rptr = dest_1_page_start;
      length = dest - dest_1_page_start;
      FLASH_Copy (wptr, rptr, length);

      // 3. Copy from (dest+numbytes) to dest_page_end to Scratch page

      wptr = FLASH_TEMP + dest - dest_1_page_start + numbytes;
      rptr = dest + numbytes;
      length = dest_1_page_end - dest - numbytes + 1;
      FLASH_Copy (wptr, rptr, length);

      // 4. Erase destination page
      FLASH_PageErase (dest_1_page_start);

      // 5. Copy Scratch page to destination page
      wptr = dest_1_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, rptr, length);

   } else {                            // value crosses page boundary
      // 1. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP);

      // 2. Copy bytes from first byte of dest page to dest-1 to Scratch page

      wptr = FLASH_TEMP;
      rptr = dest_1_page_start;
```

```
      length = dest - dest_1_page_start;
      FLASH_Copy (wptr, rptr, length);

      // 3. Erase destination page 1
      FLASH_PageErase (dest_1_page_start);

      // 4. Copy Scratch page to destination page 1
      wptr = dest_1_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, rptr, length);

      // now handle 2nd page

      // 5. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP);

      // 6. Copy bytes from numbytes remaining to dest-2_page_end to Scratch page

      numbytes_remainder = numbytes - (dest_1_page_end - dest + 1);
      wptr = FLASH_TEMP + numbytes_remainder;
      rptr = dest_2_page_start + numbytes_remainder;
      length = FLASH_pagesize - numbytes_remainder;
      FLASH_Copy (wptr, rptr, length);

      // 7. Erase destination page 2
      FLASH_PageErase (dest_2_page_start);

      // 8. Copy Scratch page to destination page 2
      wptr = dest_2_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, rptr, length);
   }
}

//-----------------------------------------------------------------------------
// FLASH_Update
//-----------------------------------------------------------------------------
//
// This routine replaces <numbytes> from <src> to the FLASH addressed by
// <dest>.  This function calls FLASH_Clear() to handle the dirty work of
// initializing all <dest> bytes to 0xff's prior to copying the bytes from
// <src> to <dest>. This function accepts <numbytes> up to <FLASH_PAGESIZE>.
//
void FLASH_Update (FLADDR dest, char *src, unsigned numbytes)
{
   // 1. Erase <numbytes> starting from <dest>
   FLASH_Clear (dest, numbytes);

   // 2. Write <numbytes> from <src> to <dest>
   FLASH_Write (dest, src, numbytes);
}



//-----------------------------------------------------------------------------
// FLASH_Write
//-----------------------------------------------------------------------------
//
// This routine copies <numbytes> from <src> to the linear FLASH address
// <dest>.
//
void FLASH_Write (FLADDR dest, char *src, unsigned numbytes)
```

```
{
   FLADDR i;

   for (i = dest; i < dest+numbytes; i++) {
      FLASH_ByteWrite (i, *src++);
   }
}

//-----------------------------------------------------------------------------
// FLASH_Read
//-----------------------------------------------------------------------------
//
// This routine copies <numbytes> from the linear FLASH address <src> to
// <dest>.
//
char * FLASH_Read (char *dest, FLADDR src, unsigned numbytes)
{
   FLADDR i;

   for (i = 0; i < numbytes; i++) {
      *dest++ = FLASH_ByteRead (src+i);
   }
   return dest;
}

//-----------------------------------------------------------------------------
// FLASH_Copy
//-----------------------------------------------------------------------------
//
// This routine copies <numbytes> from <src> to the linear FLASH address
// <dest>.
//
void FLASH_Copy (FLADDR dest, FLADDR src, unsigned numbytes)
{
   FLADDR i;

   for (i = 0; i < numbytes; i++) {

      FLASH_ByteWrite ((FLADDR) dest+i, FLASH_ByteRead((FLADDR) src+i));
   }
}



//-----------------------------------------------------------------------------
// FLASH_Fill
//-----------------------------------------------------------------------------
//
// This routine fills the FLASH beginning at <addr> with <lenght> bytes.
//
void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill)
{
   FLADDR i;

   for (i = 0; i < length; i++) {
      FLASH_ByteWrite (addr+i, fill);
   }
}
```

7.7.4. F300_FlashUtils.h

```
//-----------------------------------------------------------------------------
// F300_FlashUtils.h
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F30x
// Tool chain: KEIL C51 7.06
//
// Release 1.0
//

#ifndef F300_FLASHUTILS_H
#define F300_FLASHUTILS_H

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F300_FlashPrimitives.h"

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Exported Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
extern void FLASH_Write (FLADDR dest, char *src, unsigned numbytes);
extern char * FLASH_Read (char *dest, FLADDR src, unsigned numbytes);
extern void FLASH_Clear (FLADDR addr, unsigned numbytes);

// FLASH update/copy routines
extern void FLASH_Update (FLADDR dest, char *src, unsigned numbytes);   // copy with destina-
tion preservation
extern void FLASH_Copy (FLADDR dest, FLADDR src, unsigned numbytes);     // low-level FLASH/
FLASH byte copy

// FLASH test routines
extern void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill);

#endif // F300_FLASHUTILS_H
```

7.8. ‘F310

7.8.1. F310_FlashPrimitives.c

```
//-----------------------------------------------------------------------------
// F310_FlashPrimitives.c
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F31x
// Tool chain: KEIL C51 7.06
//
// Release 1.0
//
//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F310_FlashPrimitives.h"
#include <c8051F310.h>

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
void FLASH_ByteWrite (FLADDR addr, char byte);
unsigned char FLASH_ByteRead (FLADDR addr);
void FLASH_PageErase (FLADDR addr);

//-----------------------------------------------------------------------------
// Global Variables
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH Routines
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH_ByteWrite
//-----------------------------------------------------------------------------
//
// This routine writes <byte> to the linear FLASH address <addr>.
//
void FLASH_ByteWrite (FLADDR addr, char byte)
{
   bit EA_SAVE = EA;                   // preserve EA
   char xdata * data pwrite;           // FLASH write pointer

   EA = 0;                             // disable interrupts
```

```
   // change clock speed to slow, then restore later
   VDM0CN = 0x80;                       // enable VDD monitor


   RSTSRC = 0x02;                       // enable VDD monitor as a reset source

   pwrite = (char xdata *) addr;

   FLKEY  = 0xA5;                       // Key Sequence 1
   FLKEY  = 0xF1;                       // Key Sequence 2
   PSCTL |= 0x01;                       // PSWE = 1


   VDM0CN = 0x80;                       // enable VDD monitor


   RSTSRC = 0x02;                       // enable VDD monitor as a reset source

   *pwrite = byte;                      // write the byte

   PSCTL &= ~0x01;                      // PSWE = 0

   EA = EA_SAVE;                        // restore interrupts
}

//-----------------------------------------------------------------------------
// FLASH_ByteRead
//-----------------------------------------------------------------------------
//
// This routine reads a <byte> from the linear FLASH address <addr>.
//
unsigned char FLASH_ByteRead (FLADDR addr)
{
   bit EA_SAVE = EA;                    // preserve EA
   char code * data pread;              // FLASH read pointer
   unsigned char byte;

   EA = 0;                              // disable interrupts

   pread = (char code *) addr;

   byte = *pread;                       // read the byte

   EA = EA_SAVE;                        // restore interrupts

   return byte;
}

//-----------------------------------------------------------------------------
// FLASH_PageErase
//-----------------------------------------------------------------------------
//
// This routine erases the FLASH page containing the linear FLASH address
// <addr>.
//
void FLASH_PageErase (FLADDR addr)
{
   bit EA_SAVE = EA;                    // preserve EA
   char xdata * data pwrite;            // FLASH write pointer

   EA = 0;                             // disable interrupts
   // change clock speed to slow, then restore later

   VDM0CN = 0x80;                       // enable VDD monitor
```

```
   RSTSRC = 0x02;                   // enable VDD monitor as a reset source

   pwrite = (char xdata *) addr;

   FLKEY  = 0xA5;                   // Key Sequence 1
   FLKEY  = 0xF1;                   // Key Sequence 2
   PSCTL |= 0x03;                   // PSWE = 1; PSEE = 1


   VDM0CN = 0x80;                      // enable VDD monitor


   RSTSRC = 0x02;                   // enable VDD monitor as a reset source
   *pwrite = 0;                     // initiate page erase

   PSCTL &= ~0x03;                  // PSWE = 0; PSEE = 0

   EA = EA_SAVE;                    // restore interrupts
}
```

7.8.2. F310_FlashPrimitives.h

```
//-----------------------------------------------------------------------------
// F310_FlashPrimitives.h
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F31x
// Tool chain: KEIL C51 7.06
//
// Release 1.0
//

#ifndef F310_FLASHPRIMITIVES_H
#define F310_FLASHPRIMITIVES_H

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

typedef unsigned long ULONG;
typedef unsigned int UINT;
typedef unsigned char UCHAR;

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

#ifndef FLASH_PAGESIZE
#define FLASH_PAGESIZE 512
#endif

#ifndef FLASH_TEMP
#define FLASH_TEMP 0x03a00L
#endif

#ifndef FLASH_LAST
#define FLASH_LAST 0x03c00L
#endif

typedef UINT FLADDR;

//-----------------------------------------------------------------------------
// Exported Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
extern void FLASH_ByteWrite (FLADDR addr, char byte);
extern unsigned char FLASH_ByteRead (FLADDR addr);
extern void FLASH_PageErase (FLADDR addr);

#endif // F310_FLASHPRIMITIVES_H
```

7.8.3. F310_FlashUtils.c

```
//-----------------------------------------------------------------------------
// F310_FlashUtils.c
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F31x
// Tool chain: KEIL C51 7.06
//
// Release 1.0
//

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F310_FlashPrimitives.h"
#include "F310_FlashUtils.h"

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
void FLASH_Write (FLADDR dest, char *src, unsigned numbytes);
char * FLASH_Read (char *dest, FLADDR src, unsigned numbytes);
void FLASH_Clear (FLADDR addr, unsigned numbytes);

// FLASH update/copy routines
void FLASH_Update (FLADDR dest, char *src, unsigned numbytes);
void FLASH_Copy (FLADDR dest, FLADDR src, unsigned numbytes);

// FLASH test routines
void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill);
void FLASH_Print (FLADDR addr, ULONG length);

//-----------------------------------------------------------------------------
// Global Variables
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH Routines
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH_Clear
//-----------------------------------------------------------------------------
//
// This routine erases <numbytes> starting from the FLASH addressed by
// <dest> by performing a read-modify-write operation using <FLASH_TEMP> as
```

```
// a temporary holding area.  This function accepts <numbytes> up to
// <FLASH_PAGESIZE>.
//
void FLASH_Clear (FLADDR dest, unsigned numbytes)
{
   FLADDR dest_1_page_start;           // first address in 1st page
                                       // containing <dest>
   FLADDR dest_1_page_end;             // last address in 1st page
                                       // containing <dest>
   FLADDR dest_2_page_start;           // first address in 2nd page
                                       // containing <dest>
   FLADDR dest_2_page_end;             // last address in 2nd page
                                       // containing <dest>
   unsigned numbytes_remainder;        // when crossing page boundary,
                                       // number of <src> bytes on 2nd page
   unsigned FLASH_pagesize;            // size of FLASH page to update

   FLADDR  wptr;                       // write address
   FLADDR  rptr;                       // read address

   unsigned length;

   FLASH_pagesize = FLASH_PAGESIZE;

   dest_1_page_start = dest & ~(FLASH_pagesize - 1);
   dest_1_page_end = dest_1_page_start + FLASH_pagesize - 1;
   dest_2_page_start = (dest + numbytes)  & ~(FLASH_pagesize - 1);
   dest_2_page_end = dest_2_page_start + FLASH_pagesize - 1;

   if (dest_1_page_end == dest_2_page_end) {

      // 1. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP);

      // 2. Copy bytes from first byte of dest page to dest-1 to Scratch page

      wptr = FLASH_TEMP;
      rptr = dest_1_page_start;
      length = dest - dest_1_page_start;
      FLASH_Copy (wptr, rptr, length);

      // 3. Copy from (dest+numbytes) to dest_page_end to Scratch page

      wptr = FLASH_TEMP + dest - dest_1_page_start + numbytes;
      rptr = dest + numbytes;
      length = dest_1_page_end - dest - numbytes + 1;
      FLASH_Copy (wptr, rptr, length);

      // 4. Erase destination page
      FLASH_PageErase (dest_1_page_start);

      // 5. Copy Scratch page to destination page
      wptr = dest_1_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, rptr, length);

   } else {                            // value crosses page boundary
      // 1. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP);

      // 2. Copy bytes from first byte of dest page to dest-1 to Scratch page

      wptr = FLASH_TEMP;
```

```
      rptr = dest_1_page_start;
      length = dest - dest_1_page_start;
      FLASH_Copy (wptr, rptr, length);

      // 3. Erase destination page 1
      FLASH_PageErase (dest_1_page_start);

      // 4. Copy Scratch page to destination page 1
      wptr = dest_1_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, rptr, length);

      // now handle 2nd page

      // 5. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP);

      // 6. Copy bytes from numbytes remaining to dest-2_page_end to Scratch page

      numbytes_remainder = numbytes - (dest_1_page_end - dest + 1);
      wptr = FLASH_TEMP + numbytes_remainder;
      rptr = dest_2_page_start + numbytes_remainder;
      length = FLASH_pagesize - numbytes_remainder;
      FLASH_Copy (wptr, rptr, length);

      // 7. Erase destination page 2
      FLASH_PageErase (dest_2_page_start);

      // 8. Copy Scratch page to destination page 2
      wptr = dest_2_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, rptr, length);
   }
}

//-----------------------------------------------------------------------------
// FLASH_Update
//-----------------------------------------------------------------------------
//
// This routine replaces <numbytes> from <src> to the FLASH addressed by
// <dest>.  This function calls FLASH_Clear() to handle the dirty work of
// initializing all <dest> bytes to 0xff's prior to copying the bytes from
// <src> to <dest>. This function accepts <numbytes> up to <FLASH_PAGESIZE>.
//
void FLASH_Update (FLADDR dest, char *src, unsigned numbytes)
{
   // 1. Erase <numbytes> starting from <dest>
   FLASH_Clear (dest, numbytes);

   // 2. Write <numbytes> from <src> to <dest>
   FLASH_Write (dest, src, numbytes);
}


//-----------------------------------------------------------------------------
// FLASH_Write
//-----------------------------------------------------------------------------
//
// This routine copies <numbytes> from <src> to the linear FLASH address
// <dest>.
//
void FLASH_Write (FLADDR dest, char *src, unsigned numbytes)
```

```
{
   FLADDR i;

   for (i = dest; i < dest+numbytes; i++) {
      FLASH_ByteWrite (i, *src++);
   }
}

//-----------------------------------------------------------------------------
// FLASH_Read
//-----------------------------------------------------------------------------
//
// This routine copies <numbytes> from the linear FLASH address <src> to
// <dest>.
//
char * FLASH_Read (char *dest, FLADDR src, unsigned numbytes)
{
   FLADDR i;

   for (i = 0; i < numbytes; i++) {
      *dest++ = FLASH_ByteRead (src+i);
   }
   return dest;
}

//-----------------------------------------------------------------------------
// FLASH_Copy
//-----------------------------------------------------------------------------
//
// This routine copies <numbytes> from <src> to the linear FLASH address
// <dest>.
//
void FLASH_Copy (FLADDR dest, FLADDR src, unsigned numbytes)
{
   FLADDR i;

   for (i = 0; i < numbytes; i++) {

      FLASH_ByteWrite ((FLADDR) dest+i, FLASH_ByteRead((FLADDR) src+i));
   }
}


//-----------------------------------------------------------------------------
// FLASH_Fill
//-----------------------------------------------------------------------------
//
// This routine fills the FLASH beginning at <addr> with <lenght> bytes.
//
void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill)
{
   FLADDR i;

   for (i = 0; i < length; i++) {
      FLASH_ByteWrite (addr+i, fill);
   }
}
```

7.8.4. F310_FlashUtils.h

```
//-----------------------------------------------------------------------------
// F310_FlashUtils.h
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F31x
// Tool chain: KEIL C51 7.06
//
// Release 1.0
//

#ifndef F310_FLASHUTILS_H
#define F310_FLASHUTILS_H

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F310_FlashPrimitives.h"

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Exported Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
extern void FLASH_Write (FLADDR dest, char *src, unsigned numbytes);
extern char * FLASH_Read (char *dest, FLADDR src, unsigned numbytes);
extern void FLASH_Clear (FLADDR addr, unsigned numbytes);

// FLASH update/copy routines
extern void FLASH_Update (FLADDR dest, char *src, unsigned numbytes);   // copy with destina-
tion preservation
extern void FLASH_Copy (FLADDR dest, FLADDR src, unsigned numbytes);     // low-level FLASH/
FLASH byte copy

// FLASH test routines
extern void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill);

#endif // F310_FLASHUTILS_H
```

7.9. ‘F320

7.9.1. F320_FlashPrimitives.c

```
//-----------------------------------------------------------------------------
// F320_FlashPrimitives.c
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F32x
// Tool chain: KEIL C51 7.06
//
// Release 1.0
//

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F320_FlashPrimitives.h"
#include <c8051F320.h>

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
void FLASH_ByteWrite (FLADDR addr, char byte);
unsigned char FLASH_ByteRead (FLADDR addr);
void FLASH_PageErase (FLADDR addr);

//-----------------------------------------------------------------------------
// Global Variables
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH Routines
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH_ByteWrite
//-----------------------------------------------------------------------------
//
// This routine writes <byte> to the linear FLASH address <addr>.
//
// To do:
//  -- optimize to skip 0xFF bytes
//
void FLASH_ByteWrite (FLADDR addr, char byte)
{
   bit EA_SAVE = EA;                   // preserve EA
```

```
   char xdata * data pwrite;           // FLASH write pointer

   EA = 0;                             // disable interrupts

   // change clock speed to slow, then restore later

   VDM0CN = 0x80;                      // enable VDD monitor


   RSTSRC = 0x02;                      // enable VDD monitor as a reset source

   pwrite = (char xdata *) addr;

   FLKEY  = 0xA5;                      // Key Sequence 1
   FLKEY  = 0xF1;                      // Key Sequence 2
   PSCTL |= 0x01;                      // PSWE = 1


   VDM0CN = 0x80;                      // enable VDD monitor


   RSTSRC = 0x02;                      // enable VDD monitor as a reset source

   *pwrite = byte;                     // write the byte

   PSCTL &= ~0x01;                     // PSWE = 0

   EA = EA_SAVE;                       // restore interrupts
}

//-----------------------------------------------------------------------------
// FLASH_ByteRead
//-----------------------------------------------------------------------------
//
// This routine reads a <byte> from the linear FLASH address <addr>.
//
unsigned char FLASH_ByteRead (FLADDR addr)
{
   bit EA_SAVE = EA;                   // preserve EA
   char code * data pread;             // FLASH read pointer
   unsigned char byte;

   EA = 0;                             // disable interrupts

   pread = (char code *) addr;

   byte = *pread;                      // read the byte

   EA = EA_SAVE;                       // restore interrupts

   return byte;
}

//-----------------------------------------------------------------------------
// FLASH_PageErase
//-----------------------------------------------------------------------------
//
// This routine erases the FLASH page containing the linear FLASH address
// <addr>.
//
void FLASH_PageErase (FLADDR addr)
{
   bit EA_SAVE = EA;                   // preserve EA
   char xdata * data pwrite;           // FLASH write pointer
```

```
   EA = 0;                           // disable interrupts
   // change clock speed to slow, then restore later

   VDM0CN = 0x80;                       // enable VDD monitor


   RSTSRC = 0x02;                    // enable VDD monitor as a reset source

   pwrite = (char xdata *) addr;

   FLKEY  = 0xA5;                    // Key Sequence 1
   FLKEY  = 0xF1;                    // Key Sequence 2
   PSCTL |= 0x03;                    // PSWE = 1; PSEE = 1


   VDM0CN = 0x80;                       // enable VDD monitor


   RSTSRC = 0x02;                    // enable VDD monitor as a reset source
   *pwrite = 0;                      // initiate page erase

   PSCTL &= ~0x03;                   // PSWE = 0; PSEE = 0

   EA = EA_SAVE;                     // restore interrupts
}
```

7.9.2. F320_FlashPrimitives.h

```
//-----------------------------------------------------------------------------
// F320_FlashPrimitives.h
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F32x
// Tool chain: KEIL C51 7.06
//
// Release 1.0
//
#ifndef F320_FLASHPRIMITIVES_H
#define F320_FLASHPRIMITIVES_H

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

typedef unsigned long ULONG;
typedef unsigned int UINT;
typedef unsigned char UCHAR;

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

#ifndef FLASH_PAGESIZE
#define FLASH_PAGESIZE 512
#endif

#ifndef FLASH_TEMP
#define FLASH_TEMP 0x03a00L
#endif

#ifndef FLASH_LAST
#define FLASH_LAST 0x03c00L
#endif

typedef UINT FLADDR;

//-----------------------------------------------------------------------------
// Exported Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
extern void FLASH_ByteWrite (FLADDR addr, char byte);
extern unsigned char FLASH_ByteRead (FLADDR addr);
extern void FLASH_PageErase (FLADDR addr);

#endif // F320_FLASHPRIMITIVES_H
```

7.9.3. F320_FlashUtils.c

```
//-----------------------------------------------------------------------------
// F320_FlashUtils.c
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F32x
// Tool chain: KEIL C51 7.06
//
// Release 1.1
// 1/30/2006
// Fixed function header for Flash_Write()
//
// Release 1.0
//

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F320_FlashPrimitives.h"
#include "F320_FlashUtils.h"

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
void FLASH_Write (FLADDR dest, char *src, unsigned numbytes);
char * FLASH_Read (char *dest, FLADDR src, unsigned numbytes);
void FLASH_Clear (FLADDR addr, unsigned numbytes);

// FLASH update/copy routines
void FLASH_Update (FLADDR dest, char *src, unsigned numbytes);
void FLASH_Copy (FLADDR dest, FLADDR src, unsigned numbytes);

// FLASH test routines
void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill);

//-----------------------------------------------------------------------------
// Global Variables
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH Routines
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH_Clear
//-----------------------------------------------------------------------------
```

```
//
// This routine erases <numbytes> starting from the FLASH addressed by
// <dest> by performing a read-modify-write operation using <FLASH_TEMP> as
// a temporary holding area.  This function accepts <numbytes> up to
// <FLASH_PAGESIZE>.
//
void FLASH_Clear (FLADDR dest, unsigned numbytes)
{
   FLADDR dest_1_page_start;           // first address in 1st page
                                       // containing <dest>
   FLADDR dest_1_page_end;             // last address in 1st page
                                       // containing <dest>
   FLADDR dest_2_page_start;           // first address in 2nd page
                                       // containing <dest>
   FLADDR dest_2_page_end;             // last address in 2nd page
                                       // containing <dest>
   unsigned numbytes_remainder;        // when crossing page boundary,
                                       // number of <src> bytes on 2nd page
   unsigned FLASH_pagesize;            // size of FLASH page to update

   FLADDR  wptr;                       // write address
   FLADDR  rptr;                       // read address

   unsigned length;

   FLASH_pagesize = FLASH_PAGESIZE;

   dest_1_page_start = dest & ~(FLASH_pagesize - 1);
   dest_1_page_end = dest_1_page_start + FLASH_pagesize - 1;
   dest_2_page_start = (dest + numbytes)  & ~(FLASH_pagesize - 1);
   dest_2_page_end = dest_2_page_start + FLASH_pagesize - 1;

   if (dest_1_page_end == dest_2_page_end) {

      // 1. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP);

      // 2. Copy bytes from first byte of dest page to dest-1 to Scratch page

      wptr = FLASH_TEMP;
      rptr = dest_1_page_start;
      length = dest - dest_1_page_start;
      FLASH_Copy (wptr, rptr, length);

      // 3. Copy from (dest+numbytes) to dest_page_end to Scratch page

      wptr = FLASH_TEMP + dest - dest_1_page_start + numbytes;
      rptr = dest + numbytes;
      length = dest_1_page_end - dest - numbytes + 1;
      FLASH_Copy (wptr, rptr, length);

      // 4. Erase destination page
      FLASH_PageErase (dest_1_page_start);

      // 5. Copy Scratch page to destination page
      wptr = dest_1_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, rptr, length);

   } else {                            // value crosses page boundary
      // 1. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP);
```

```
      // 2. Copy bytes from first byte of dest page to dest-1 to Scratch page

      wptr = FLASH_TEMP;
      rptr = dest_1_page_start;
      length = dest - dest_1_page_start;
      FLASH_Copy (wptr, rptr, length);

      // 3. Erase destination page 1
      FLASH_PageErase (dest_1_page_start);

      // 4. Copy Scratch page to destination page 1
      wptr = dest_1_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, rptr, length);

      // now handle 2nd page

      // 5. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP);

      // 6. Copy bytes from numbytes remaining to dest-2_page_end to Scratch page

      numbytes_remainder = numbytes - (dest_1_page_end - dest + 1);
      wptr = FLASH_TEMP + numbytes_remainder;
      rptr = dest_2_page_start + numbytes_remainder;
      length = FLASH_pagesize - numbytes_remainder;
      FLASH_Copy (wptr, rptr, length);

      // 7. Erase destination page 2
      FLASH_PageErase (dest_2_page_start);

      // 8. Copy Scratch page to destination page 2
      wptr = dest_2_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, rptr, length);
   }
}


//-----------------------------------------------------------------------------
// FLASH_Update
//-----------------------------------------------------------------------------
//
// This routine replaces <numbytes> from <src> to the FLASH addressed by
// <dest>.  This function calls FLASH_Clear() to handle the dirty work of
// initializing all <dest> bytes to 0xff's prior to copying the bytes from
// <src> to <dest>. This function accepts <numbytes> up to <FLASH_PAGESIZE>.
//
void FLASH_Update (FLADDR dest, char *src, unsigned numbytes)
{
   // 1. Erase <numbytes> starting from <dest>
   FLASH_Clear (dest, numbytes);

   // 2. Write <numbytes> from <src> to <dest>
   FLASH_Write (dest, src, numbytes);
}


//-----------------------------------------------------------------------------
// FLASH_Write
//-----------------------------------------------------------------------------
//
```

```
// This routine copies <numbytes> from <src> to the linear FLASH address
// <dest>.
//
void FLASH_Write (FLADDR dest, char *src, unsigned numbytes)
{
   FLADDR i;

   for (i = dest; i < dest+numbytes; i++) {
      FLASH_ByteWrite (i, *src++);
   }
}

//-----------------------------------------------------------------------------
// FLASH_Read
//-----------------------------------------------------------------------------
//
// This routine copies <numbytes> from the linear FLASH address <src> to
// <dest>.
//
char * FLASH_Read (char *dest, FLADDR src, unsigned numbytes)
{
   FLADDR i;

   for (i = 0; i < numbytes; i++) {
      *dest++ = FLASH_ByteRead (src+i);
   }
   return dest;
}

//-----------------------------------------------------------------------------
// FLASH_Copy
//-----------------------------------------------------------------------------
//
// This routine copies <numbytes> from <src> to the linear FLASH address
// <dest>.
//
// To do:
//  -- optimize to skip 0xFF bytes
//
void FLASH_Copy (FLADDR dest, FLADDR src, unsigned numbytes)
{
   FLADDR i;

   for (i = 0; i < numbytes; i++) {

      FLASH_ByteWrite ((FLADDR) dest+i, FLASH_ByteRead((FLADDR) src+i));
   }
}


//-----------------------------------------------------------------------------
// FLASH_Fill
//-----------------------------------------------------------------------------
//
// This routine fills the FLASH beginning at <addr> with <lenght> bytes.
//
void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill)
{
   FLADDR i;

   for (i = 0; i < length; i++) {
      FLASH_ByteWrite (addr+i, fill);
   }
}
```

7.9.4. F320_FlashUtils.h

```
//-----------------------------------------------------------------------------
// F320_FlashUtils.h
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F32x
// Tool chain: KEIL C51 7.06
//
// Release 1.0
//

#ifndef F320_FLASHUTILS_H
#define F320_FLASHUTILS_H

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F320_FlashPrimitives.h"

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Exported Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
extern void FLASH_Write (FLADDR dest, char *src, unsigned numbytes);
extern char * FLASH_Read (char *dest, FLADDR src, unsigned numbytes);
extern void FLASH_Clear (FLADDR addr, unsigned numbytes);

// FLASH update/copy routines
extern void FLASH_Update (FLADDR dest, char *src, unsigned numbytes);   // copy with destina-
tion preservation
extern void FLASH_Copy (FLADDR dest, FLADDR src, unsigned numbytes);     // low-level FLASH/
FLASH byte copy

// FLASH test routines
extern void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill);

#endif // F320_FLASHUTILS_H
```

7.10. ‘F326/7

7.10.1. F326_FlashPrimitives.c

```
//-----------------------------------------------------------------------------
// F326_FlashPrimitives.c
//-----------------------------------------------------------------------------
// Copyright 2006 Silicon Laboratories, Inc.
// http://www.silabs.com
//
// Program Description:
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// FID:            326000024
// Target:         C8051F326/7
// Tool chain:     Keil C51 8.00 / Keil EVAL C51
// Command Line:   None
//
// Release 1.0
//    -Initial Revision (GP)
//    -30 JAN 2006
//

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F326_FlashPrimitives.h"
#include <c8051F326.h>

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
void FLASH_ByteWrite (FLADDR addr, char byte);
unsigned char FLASH_ByteRead (FLADDR addr);
void FLASH_PageErase (FLADDR addr);

//-----------------------------------------------------------------------------
// Global Variables
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH Routines
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH_ByteWrite
//-----------------------------------------------------------------------------
//
// Return Value : None
// Parameters   :
//   1)  FLADDR addr - target address to write to
//                     range is 0 to (FLASH_TEMP-1)
```

```
//   2)  char byte   - byte to write
//
// This routine writes <byte> to the linear FLASH address <addr>.
//
//-------------------------------------------------------------------------
void FLASH_ByteWrite (FLADDR addr, char byte)
{
   bit EA_SAVE = EA;                   // Preserve EA
   char xdata * data pwrite;           // FLASH write pointer

   EA = 0;                             // Disable interrupts

   // change clock speed to slow, then restore later

   VDM0CN = 0x80;                      // Enable VDD monitor


   RSTSRC = 0x02;                      // Enable VDD monitor as a reset source

   pwrite = (char xdata *) addr;

   FLKEY  = 0xA5;                      // Key Sequence 1
   FLKEY  = 0xF1;                      // Key Sequence 2
   PSCTL |= 0x01;                      // PSWE = 1


   VDM0CN = 0x80;                      // Enable VDD monitor


   RSTSRC = 0x02;                      // Enable VDD monitor as a reset source

   *pwrite = byte;                     // Write the byte

   PSCTL &= ~0x01;                     // PSWE = 0

   EA = EA_SAVE;                       // Restore interrupts
}

//-------------------------------------------------------------------------
// FLASH_ByteRead
//-------------------------------------------------------------------------
//
// Return Value :
//   1)  unsigned char  - byte that was read from Flash
// Parameters   :
//   1)  FLADDR addr    - target address to write to
//                        range is 0 to (FLASH_TEMP-1)
//
// This routine reads a <byte> from the linear FLASH address <addr>.
//
//-------------------------------------------------------------------------
unsigned char FLASH_ByteRead (FLADDR addr)
{
   bit EA_SAVE = EA;                   // Preserve EA
   char code * data pread;             // FLASH read pointer
   unsigned char byte;

   EA = 0;                             // Disable interrupts

   pread = (char code *) addr;

   byte = *pread;                      // Read the byte

   EA = EA_SAVE;                       // Restore interrupts
```

```
   return byte;
}

//-----------------------------------------------------------------------------
// FLASH_PageErase
//-----------------------------------------------------------------------------
//
// Return Value : None
// Parameters   :
//   1)  FLADDR addr    - target address to write to
//                        range is 0 to (FLASH_TEMP-1)
//
// This routine erases the FLASH page containing the linear FLASH address
// <addr>.
//
//-----------------------------------------------------------------------------
void FLASH_PageErase (FLADDR addr)
{
   bit EA_SAVE = EA;                   // Preserve EA
   char xdata * data pwrite;           // FLASH write pointer

   EA = 0;                             // Disable interrupts
   // change clock speed to slow, then restore later

   VDM0CN = 0x80;                      // Enable VDD monitor


   RSTSRC = 0x02;                      // enable VDD monitor as a reset source

   pwrite = (char xdata *) addr;

   FLKEY  = 0xA5;                      // Key Sequence 1
   FLKEY  = 0xF1;                      // Key Sequence 2
   PSCTL |= 0x03;                      // PSWE = 1; PSEE = 1


   VDM0CN = 0x80;                      // Enable VDD monitor


   RSTSRC = 0x02;                      // Enable VDD monitor as a reset source
   *pwrite = 0;                        // Initiate page erase

   PSCTL &= ~0x03;                     // PSWE = 0; PSEE = 0

   EA = EA_SAVE;                       // Restore interrupts
}

//-----------------------------------------------------------------------------
// End Of File
//-----------------------------------------------------------------------------
```

7.10.2. F326_FlashPrimitives.h

```
//-----------------------------------------------------------------------------
//  F326_FlashPrimitives.h
//-----------------------------------------------------------------------------
// Copyright 2006 Silicon Laboratories, Inc.
// http://www.silabs.com
//
// Program Description:
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// FID:            326000025
// Target:         C8051F326/7
// Tool chain:     Keil C51 8.00 / Keil EVAL C51
// Command Line:   None
//
// Release 1.0
//    -Initial Revision (GP)
//    -30 JAN 2006
//

#ifndef F326_FLASHPRIMITIVES_H
#define F326_FLASHPRIMITIVES_H

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

typedef unsigned long ULONG;
typedef unsigned int  UINT;
typedef unsigned char UCHAR;

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

#ifndef FLASH_PAGESIZE
#define FLASH_PAGESIZE 512
#endif

#ifndef FLASH_TEMP
#define FLASH_TEMP 0x03a00L
#endif

#ifndef FLASH_LAST
#define FLASH_LAST 0x03c00L
#endif

typedef UINT FLADDR;

//-----------------------------------------------------------------------------
// Exported Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
extern void FLASH_ByteWrite (FLADDR addr, char byte);
extern unsigned char FLASH_ByteRead (FLADDR addr);
extern void FLASH_PageErase (FLADDR addr);

#endif                                 // F326_FLASHPRIMITIVES_H

//-----------------------------------------------------------------------------
// End Of File
//-----------------------------------------------------------------------------
```

7.10.3. F326_FlashUtils.c

```
//-----------------------------------------------------------------------------
//  F326_FlashUtils.c
//-----------------------------------------------------------------------------
// Copyright 2006 Silicon Laboratories, Inc.
// http://www.silabs.com
//
// Program Description:
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// FID:            326000026
// Target:         C8051F326/7
// Tool chain:     Keil C51 8.00 / Keil EVAL C51
// Command Line:   None
//
// Release 1.0
//    -Initial Revision (GP)
//    -30 JAN 2006
//

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F326_FlashPrimitives.h"
#include "F326_FlashUtils.h"

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
void FLASH_Write (FLADDR dest, char *src, unsigned numbytes);
char * FLASH_Read (char *dest, FLADDR src, unsigned numbytes);
void FLASH_Clear (FLADDR addr, unsigned numbytes);

// FLASH update/copy routines
void FLASH_Update (FLADDR dest, char *src, unsigned numbytes);
void FLASH_Copy (FLADDR dest, FLADDR src, unsigned numbytes);

// FLASH test routines
void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill);

//-----------------------------------------------------------------------------
// Global Variables
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH Routines
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH_Clear
//-----------------------------------------------------------------------------
```

```
//
// Return Value : None
// Parameters   :
//   1)  FLADDR dest - target address where to start clearing bytes
//                     range is 0 to <FLASH_TEMP>-1
//   2)  unsigned numbytes  - number of bytes to clear
//                            range is 1 to <FLASH_PAGESIZE>
//
// This routine erases <numbytes> starting from the FLASH addressed by
// <dest> by performing a read-modify-write operation using <FLASH_TEMP> as
// a temporary holding area.
//
//-------------------------------------------------------------------------

void FLASH_Clear (FLADDR dest, unsigned numbytes)
{
   FLADDR dest_1_page_start;           // First address in 1st page
                                       // containing <dest>
   FLADDR dest_1_page_end;             // Last address in 1st page
                                       // containing <dest>
   FLADDR dest_2_page_start;           // First address in 2nd page
                                       // containing <dest>
   FLADDR dest_2_page_end;             // Last address in 2nd page
                                       // containing <dest>
   unsigned numbytes_remainder;        // When crossing page boundary,
                                       // number of <src> bytes on 2nd page
   unsigned FLASH_pagesize;            // Size of FLASH page to update

   FLADDR  wptr;                       // Write address
   FLADDR  rptr;                       // Read address

   unsigned length;

   FLASH_pagesize = FLASH_PAGESIZE;

   dest_1_page_start = dest & ~(FLASH_pagesize - 1);
   dest_1_page_end = dest_1_page_start + FLASH_pagesize - 1;
   dest_2_page_start = (dest + numbytes)  & ~(FLASH_pagesize - 1);
   dest_2_page_end = dest_2_page_start + FLASH_pagesize - 1;

   if (dest_1_page_end == dest_2_page_end) {

      // 1. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP);

      // 2. Copy bytes from first byte of dest page to dest-1 to Scratch page

      wptr = FLASH_TEMP;
      rptr = dest_1_page_start;
      length = dest - dest_1_page_start;
      FLASH_Copy (wptr, rptr, length);

      // 3. Copy from (dest+numbytes) to dest_page_end to Scratch page

      wptr = FLASH_TEMP + dest - dest_1_page_start + numbytes;
      rptr = dest + numbytes;
      length = dest_1_page_end - dest - numbytes + 1;
      FLASH_Copy (wptr, rptr, length);

      // 4. Erase destination page
      FLASH_PageErase (dest_1_page_start);

      // 5. Copy Scratch page to destination page
      wptr = dest_1_page_start;
```

```
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, rptr, length);

   } else {                            // Value crosses page boundary
      // 1. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP);

      // 2. Copy bytes from first byte of dest page to dest-1
  //    to Scratch page

      wptr = FLASH_TEMP;
      rptr = dest_1_page_start;
      length = dest - dest_1_page_start;
      FLASH_Copy (wptr, rptr, length);

      // 3. Erase destination page 1
      FLASH_PageErase (dest_1_page_start);

      // 4. Copy Scratch page to destination page 1
      wptr = dest_1_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, rptr, length);

      // now handle 2nd page

      // 5. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP);

      // 6. Copy bytes from numbytes remaining to dest-2_page_end
  //    to Scratch page

      numbytes_remainder = numbytes - (dest_1_page_end - dest + 1);
      wptr = FLASH_TEMP + numbytes_remainder;
      rptr = dest_2_page_start + numbytes_remainder;
      length = FLASH_pagesize - numbytes_remainder;
      FLASH_Copy (wptr, rptr, length);

      // 7. Erase destination page 2
      FLASH_PageErase (dest_2_page_start);

      // 8. Copy Scratch page to destination page 2
      wptr = dest_2_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, rptr, length);
   }
}

//-------------------------------------------------------------------------
// FLASH_Update
//-------------------------------------------------------------------------
//
// Return Value : None
// Parameters   :
//   1)  FLADDR dest - target address where to start updating bytes
//                     range is 0 to (FLASH_TEMP-1)
//   2)  char *src   - pointer to address where source bytes are located
//   3)  unsigned numbytes  - number of bytes to clear
//                            range is 1 to <FLASH_PAGESIZE>
//
// This routine replaces <numbytes> from <src> to the FLASH addressed by
// <dest>.  This function calls FLASH_Clear() to handle the dirty work of
```

```
// initializing all <dest> bytes to 0xff's prior to copying the bytes from
// <src> to <dest>.
//
//-----------------------------------------------------------------------------

void FLASH_Update (FLADDR dest, char *src, unsigned numbytes)
{
   // 1. Erase <numbytes> starting from <dest>
   FLASH_Clear (dest, numbytes);

   // 2. Write <numbytes> from <src> to <dest>
   FLASH_Write (dest, src, numbytes);
}

//-----------------------------------------------------------------------------
// FLASH_Write
//-----------------------------------------------------------------------------
//
// Return Value : None
// Parameters   :
//   1)  FLADDR dest - target address where to start writing bytes
//                     range is 0 to (FLASH_TEMP-1)
//   2)  char *src   - pointer to address where source bytes are located
//   3)  unsigned numbytes   - number of bytes to write
//                     range is limited by Flash size and starting location
//
// This routine copies <numbytes> from <src> to the linear FLASH address
// <dest>.
//
//-----------------------------------------------------------------------------
void FLASH_Write (FLADDR dest, char *src, unsigned numbytes)
{
   FLADDR i;

   for (i = dest; i < dest+numbytes; i++) {
      FLASH_ByteWrite (i, *src++);
   }
}
//-----------------------------------------------------------------------------
// FLASH_Read
//-----------------------------------------------------------------------------
//
// Return Value :
//   1)  char *     -  pointer to bytes read back
// Parameters   :
//   1)  char *dest -  target address where to store bytes read from Flash
//   2)  char *src   - pointer to address where source bytes are located
//   3)  unsigned numbytes   - number of bytes to read
//                     range is limited by Flash size and starting location
//
// This routine copies <numbytes> from the linear FLASH address <src> to
// <dest>.
//
//-----------------------------------------------------------------------------
char * FLASH_Read (char *dest, FLADDR src, unsigned numbytes)
{
   FLADDR i;

   for (i = 0; i < numbytes; i++) {
      *dest++ = FLASH_ByteRead (src+i);
   }
   return dest;
}
```

```
//-----------------------------------------------------------------------------
// FLASH_Copy
//-----------------------------------------------------------------------------
//
// Return Value : None
// Parameters   :
//   1)  FLADDR dest -  target address where to copy the source bytes
//   2)  FLADDR src   - target address where source bytes are located
//   3)  unsigned numbytes   - number of bytes to read
//                     range is limited by Flash size and starting location
//
// This routine copies <numbytes> from <src> to the linear FLASH address
// <dest>.
//
//-----------------------------------------------------------------------------

void FLASH_Copy (FLADDR dest, FLADDR src, unsigned numbytes)
{
   FLADDR i;

   for (i = 0; i < numbytes; i++) {

      FLASH_ByteWrite ((FLADDR) dest+i, FLASH_ByteRead((FLADDR) src+i));
   }
}


//-----------------------------------------------------------------------------
// FLASH_Fill
//-----------------------------------------------------------------------------
//
// Return Value : None
// Parameters   :
//   1)  FLADDR addr   - target address where to start filling
//   2)  ULONG length  - number of bytes to fill
//                       range is limited by Flash size and starting point
//   3)  UCHAR fill    - char to fill
//                       range is 0x00 to 0xFF
//
// This routine fills the FLASH beginning at <addr> with <length> bytes.
//
//-----------------------------------------------------------------------------
void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill)
{
   FLADDR i;

   for (i = 0; i < length; i++) {
      FLASH_ByteWrite (addr+i, fill);
   }
}

//-----------------------------------------------------------------------------
// End Of File
//-----------------------------------------------------------------------------
```

7.10.4. F326_FlashUtils.h

```
//-----------------------------------------------------------------------------
//  F326_FlashUtils.h
//-----------------------------------------------------------------------------
// Copyright 2006 Silicon Laboratories, Inc.
// http://www.silabs.com
//
// Program Description:
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// FID:            326000027
// Target:         C8051F326/7
// Tool chain:     Keil C51 8.00 / Keil EVAL C51
// Command Line:   None
//
// Release 1.0
//    -Initial Revision (GP)
//    -30 JAN 2006
//

#ifndef F326_FLASHUTILS_H
#define F326_FLASHUTILS_H

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F326_FlashPrimitives.h"

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Exported Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
extern void FLASH_Write (FLADDR dest, char *src, unsigned numbytes);
extern char * FLASH_Read (char *dest, FLADDR src, unsigned numbytes);
extern void FLASH_Clear (FLADDR addr, unsigned numbytes);

// FLASH update/copy routines

// copy with destination preservation
extern void FLASH_Update (FLADDR dest, char *src, unsigned numbytes);
// low-level FLASH/FLASH byte copy
extern void FLASH_Copy (FLADDR dest, FLADDR src, unsigned numbytes);

// FLASH test routines
extern void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill);

#endif                                 // F326_FLASHUTILS_H

//-----------------------------------------------------------------------------
// End Of File
//-----------------------------------------------------------------------------
```

7.11. ‘F330

7.11.1. F330_FlashPrimitives.c

```
//-----------------------------------------------------------------------------
// F330_FlashPrimitives.c
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F33x
// Tool chain: KEIL C51 7.06
//
// Release 1.0
//

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F330_FlashPrimitives.h"
#include <c8051F330.h>

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
void FLASH_ByteWrite (FLADDR addr, char byte);
unsigned char FLASH_ByteRead (FLADDR addr);
void FLASH_PageErase (FLADDR addr);

//-----------------------------------------------------------------------------
// Global Variables
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH Routines
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH_ByteWrite
//-----------------------------------------------------------------------------
//
// This routine writes <byte> to the linear FLASH address <addr>.
//
// To do:
//  -- optimize to skip 0xFF bytes
//
void FLASH_ByteWrite (FLADDR addr, char byte)
{
   bit EA_SAVE = EA;                   // preserve EA
```

```
   char xdata * data pwrite;           // FLASH write pointer

   EA = 0;                             // disable interrupts

   // change clock speed to slow, then restore later

   VDM0CN = 0x80;                      // enable VDD monitor


   RSTSRC = 0x02;                      // enable VDD monitor as a reset source

   pwrite = (char xdata *) addr;

   FLKEY  = 0xA5;                      // Key Sequence 1
   FLKEY  = 0xF1;                      // Key Sequence 2
   PSCTL |= 0x01;                      // PSWE = 1


   VDM0CN = 0x80;                      // enable VDD monitor


   RSTSRC = 0x02;                      // enable VDD monitor as a reset source

   *pwrite = byte;                     // write the byte

   PSCTL &= ~0x01;                     // PSWE = 0

   EA = EA_SAVE;                       // restore interrupts
}

//-----------------------------------------------------------------------------
// FLASH_ByteRead
//-----------------------------------------------------------------------------
//
// This routine reads a <byte> from the linear FLASH address <addr>.
//
unsigned char FLASH_ByteRead (FLADDR addr)
{
   bit EA_SAVE = EA;                   // preserve EA
   char code * data pread;             // FLASH read pointer
   unsigned char byte;

   EA = 0;                             // disable interrupts

   pread = (char code *) addr;

   byte = *pread;                      // read the byte

   EA = EA_SAVE;                       // restore interrupts

   return byte;
}

//-----------------------------------------------------------------------------
// FLASH_PageErase
//-----------------------------------------------------------------------------
//
// This routine erases the FLASH page containing the linear FLASH address
// <addr>.
//
void FLASH_PageErase (FLADDR addr)
{
   bit EA_SAVE = EA;                   // preserve EA
   char xdata * data pwrite;           // FLASH write pointer
```

```
   EA = 0;                          // disable interrupts
   // change clock speed to slow, then restore later

   VDM0CN = 0x80;                      // enable VDD monitor


   RSTSRC = 0x02;                   // enable VDD monitor as a reset source

   pwrite = (char xdata *) addr;

   FLKEY  = 0xA5;                   // Key Sequence 1
   FLKEY  = 0xF1;                   // Key Sequence 2
   PSCTL |= 0x03;                   // PSWE = 1; PSEE = 1


   VDM0CN = 0x80;                      // enable VDD monitor


   RSTSRC = 0x02;                   // enable VDD monitor as a reset source
   *pwrite = 0;                     // initiate page erase

   PSCTL &= ~0x03;                  // PSWE = 0; PSEE = 0

   EA = EA_SAVE;                    // restore interrupts
}
```

7.11.2. F330_FlashPrimitives.h

```
//-----------------------------------------------------------------------------
// F330_FlashPrimitives.h
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F33x
// Tool chain: KEIL C51 7.06
//
// Release 1.0
//
#ifndef F330_FLASHPRIMITIVES_H
#define F330_FLASHPRIMITIVES_H

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

typedef unsigned long ULONG;
typedef unsigned int UINT;
typedef unsigned char UCHAR;

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

#ifndef FLASH_PAGESIZE
#define FLASH_PAGESIZE 512
#endif

#ifndef FLASH_TEMP
#define FLASH_TEMP 0x01a00L
#endif

#ifndef FLASH_LAST
#define FLASH_LAST 0x01c00L
#endif

typedef UINT FLADDR;

//-----------------------------------------------------------------------------
// Exported Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
extern void FLASH_ByteWrite (FLADDR addr, char byte);
extern unsigned char FLASH_ByteRead (FLADDR addr);
extern void FLASH_PageErase (FLADDR addr);

#endif // F310_FLASHPRIMITIVES_H
```

7.11.3. F330_FlashUtils.c

```
//-----------------------------------------------------------------------------
// F330_FlashUtils.c
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F33x
// Tool chain: KEIL C51 7.06
//
// Release 1.0
//

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F330_FlashPrimitives.h"
#include "F330_FlashUtils.h"

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
void FLASH_Write (FLADDR dest, char *src, unsigned numbytes);
char * FLASH_Read (char *dest, FLADDR src, unsigned numbytes);
void FLASH_Clear (FLADDR addr, unsigned numbytes);

// FLASH update/copy routines
void FLASH_Update (FLADDR dest, char *src, unsigned numbytes);
void FLASH_Copy (FLADDR dest, FLADDR src, unsigned numbytes);

// FLASH test routines
void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill);

//-----------------------------------------------------------------------------
// Global Variables
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH Routines
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH_Clear
//-----------------------------------------------------------------------------
//
// This routine erases <numbytes> starting from the FLASH addressed by
// <dest> by performing a read-modify-write operation using <FLASH_TEMP> as
// a temporary holding area.  This function accepts <numbytes> up to
```

```
// <FLASH_PAGESIZE>.
//
void FLASH_Clear (FLADDR dest, unsigned numbytes)
{
   FLADDR dest_1_page_start;           // first address in 1st page
                                       // containing <dest>
   FLADDR dest_1_page_end;             // last address in 1st page
                                       // containing <dest>
   FLADDR dest_2_page_start;           // first address in 2nd page
                                       // containing <dest>
   FLADDR dest_2_page_end;             // last address in 2nd page
                                       // containing <dest>
   unsigned numbytes_remainder;        // when crossing page boundary,
                                       // number of <src> bytes on 2nd page
   unsigned FLASH_pagesize;            // size of FLASH page to update

   FLADDR  wptr;                       // write address
   FLADDR  rptr;                       // read address

   unsigned length;

   FLASH_pagesize = FLASH_PAGESIZE;

   dest_1_page_start = dest & ~(FLASH_pagesize - 1);
   dest_1_page_end = dest_1_page_start + FLASH_pagesize - 1;
   dest_2_page_start = (dest + numbytes)  & ~(FLASH_pagesize - 1);
   dest_2_page_end = dest_2_page_start + FLASH_pagesize - 1;

   if (dest_1_page_end == dest_2_page_end) {

      // 1. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP);

      // 2. Copy bytes from first byte of dest page to dest-1 to Scratch page

      wptr = FLASH_TEMP;
      rptr = dest_1_page_start;
      length = dest - dest_1_page_start;
      FLASH_Copy (wptr, rptr, length);

      // 3. Copy from (dest+numbytes) to dest_page_end to Scratch page

      wptr = FLASH_TEMP + dest - dest_1_page_start + numbytes;
      rptr = dest + numbytes;
      length = dest_1_page_end - dest - numbytes + 1;
      FLASH_Copy (wptr, rptr, length);

      // 4. Erase destination page
      FLASH_PageErase (dest_1_page_start);

      // 5. Copy Scratch page to destination page
      wptr = dest_1_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, rptr, length);

   } else {                            // value crosses page boundary
      // 1. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP);

      // 2. Copy bytes from first byte of dest page to dest-1 to Scratch page

      wptr = FLASH_TEMP;
      rptr = dest_1_page_start;
```

```
      length = dest - dest_1_page_start;
      FLASH_Copy (wptr, rptr, length);

      // 3. Erase destination page 1
      FLASH_PageErase (dest_1_page_start);

      // 4. Copy Scratch page to destination page 1
      wptr = dest_1_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, rptr, length);

      // now handle 2nd page

      // 5. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP);

      // 6. Copy bytes from numbytes remaining to dest-2_page_end to Scratch page

      numbytes_remainder = numbytes - (dest_1_page_end - dest + 1);
      wptr = FLASH_TEMP + numbytes_remainder;
      rptr = dest_2_page_start + numbytes_remainder;
      length = FLASH_pagesize - numbytes_remainder;
      FLASH_Copy (wptr, rptr, length);

      // 7. Erase destination page 2
      FLASH_PageErase (dest_2_page_start);

      // 8. Copy Scratch page to destination page 2
      wptr = dest_2_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, rptr, length);
   }
}


//-----------------------------------------------------------------------------
// FLASH_Update
//-----------------------------------------------------------------------------
//
// This routine replaces <numbytes> from <src> to the FLASH addressed by
// <dest>.  This function calls FLASH_Clear() to handle the dirty work of
// initializing all <dest> bytes to 0xff's prior to copying the bytes from
// <src> to <dest>. This function accepts <numbytes> up to <FLASH_PAGESIZE>.
//
void FLASH_Update (FLADDR dest, char *src, unsigned numbytes)
{
   // 1. Erase <numbytes> starting from <dest>
   FLASH_Clear (dest, numbytes);

   // 2. Write <numbytes> from <src> to <dest>
   FLASH_Write (dest, src, numbytes);
}


//-----------------------------------------------------------------------------
// FLASH_Write
//-----------------------------------------------------------------------------
//
// This routine copies <numbytes> from <src> to the linear FLASH address
// <dest>.
//
void FLASH_Write (FLADDR dest, char *src, unsigned numbytes)
```

```
{
   FLADDR i;

   for (i = dest; i < dest+numbytes; i++) {
      FLASH_ByteWrite (i, *src++);
   }
}

//-----------------------------------------------------------------------------
// FLASH_Read
//-----------------------------------------------------------------------------
//
// This routine copies <numbytes> from the linear FLASH address <src> to
// <dest>.
//
char * FLASH_Read (char *dest, FLADDR src, unsigned numbytes)
{
   FLADDR i;

   for (i = 0; i < numbytes; i++) {
      *dest++ = FLASH_ByteRead (src+i);
   }
   return dest;
}

//-----------------------------------------------------------------------------
// FLASH_Copy
//-----------------------------------------------------------------------------
//
// This routine copies <numbytes> from <src> to the linear FLASH address
// <dest>.
//
void FLASH_Copy (FLADDR dest, FLADDR src, unsigned numbytes)
{
   FLADDR i;

   for (i = 0; i < numbytes; i++) {

      FLASH_ByteWrite ((FLADDR) dest+i, FLASH_ByteRead((FLADDR) src+i));
   }
}


//-----------------------------------------------------------------------------
// FLASH_Fill
//-----------------------------------------------------------------------------
//
// This routine fills the FLASH beginning at <addr> with <lenght> bytes.
//
void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill)
{
   FLADDR i;

   for (i = 0; i < length; i++) {
      FLASH_ByteWrite (addr+i, fill);
   }
}
```

7.11.4. F330_FlashUtils.h

```
//-----------------------------------------------------------------------------
// F330_FlashUtils.h
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F33x
// Tool chain: KEIL C51 7.06
//
// Release 1.0
//

#ifndef F330_FLASHUTILS_H
#define F330_FLASHUTILS_H

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F330_FlashPrimitives.h"

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Exported Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
extern void FLASH_Write (FLADDR dest, char *src, unsigned numbytes);
extern char * FLASH_Read (char *dest, FLADDR src, unsigned numbytes);
extern void FLASH_Clear (FLADDR addr, unsigned numbytes);

// FLASH update/copy routines
extern void FLASH_Update (FLADDR dest, char *src, unsigned numbytes);   // copy with destina-
tion preservation
extern void FLASH_Copy (FLADDR dest, FLADDR src, unsigned numbytes);     // low-level FLASH/
FLASH byte copy

// FLASH test routines
extern void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill);

#endif // F330_FLASHUTILS_H
```

7.12. ‘F340

7.12.1. F340_FlashPrimitives.c

```
//-----------------------------------------------------------------------------
// F340_FlashPrimitives.c
//-----------------------------------------------------------------------------
// Copyright 2006 Silicon Laboratories, Inc.
// http://www.silabs.com
//
// Program Description:
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// FID:            34X000024
// Target:         C8051F34x
// Tool chain:     Keil C51 8.00 / Keil EVAL C51
// Command Line:   None
//
// Release 1.0
//    -Initial Revision (GP)
//    -30 JAN 2006
//

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F340_FlashPrimitives.h"
#include <c8051F340.h>

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
void FLASH_ByteWrite (FLADDR addr, char byte);
unsigned char FLASH_ByteRead (FLADDR addr);
void FLASH_PageErase (FLADDR addr);

//-----------------------------------------------------------------------------
// Global Variables
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH Routines
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH_ByteWrite
//-----------------------------------------------------------------------------
//
// Return Value : None
// Parameters   :
//   1)  FLADDR addr - target address to write to
//                     range is 0 to (FLASH_TEMP-1)
```

```
//   2)  char byte   - byte to write
//
// This routine writes <byte> to the linear FLASH address <addr>.
//
//-------------------------------------------------------------------------
void FLASH_ByteWrite (FLADDR addr, char byte)
{
   bit EA_SAVE = EA;                   // Preserve EA
   char xdata * data pwrite;           // FLASH write pointer

   EA = 0;                             // Disable interrupts

   // change clock speed to slow, then restore later

   VDM0CN = 0x80;                      // Enable VDD monitor


   RSTSRC = 0x02;                      // Enable VDD monitor as a reset source

   pwrite = (char xdata *) addr;

   FLKEY  = 0xA5;                      // Key Sequence 1
   FLKEY  = 0xF1;                      // Key Sequence 2
   PSCTL |= 0x01;                      // PSWE = 1


   VDM0CN = 0x80;                      // Enable VDD monitor


   RSTSRC = 0x02;                      // Enable VDD monitor as a reset source

   *pwrite = byte;                     // Write the byte

   PSCTL &= ~0x01;                     // PSWE = 0

   EA = EA_SAVE;                       // Restore interrupts
}

//-------------------------------------------------------------------------
// FLASH_ByteRead
//-------------------------------------------------------------------------
//
// Return Value :
//   1)  unsigned char  - byte that was read from Flash
// Parameters   :
//   1)  FLADDR addr    - target address to write to
//                        range is 0 to (FLASH_TEMP-1)
//
// This routine reads a <byte> from the linear FLASH address <addr>.
//
//-------------------------------------------------------------------------
unsigned char FLASH_ByteRead (FLADDR addr)
{
   bit EA_SAVE = EA;                   // Preserve EA
   char code * data pread;             // FLASH read pointer
   unsigned char byte;

   EA = 0;                             // Disable interrupts

   pread = (char code *) addr;

   byte = *pread;                      // Read the byte

   EA = EA_SAVE;                       // Restore interrupts
```

```
   return byte;
}

//-----------------------------------------------------------------------------
// FLASH_PageErase
//-----------------------------------------------------------------------------
//
// Return Value : None
// Parameters   :
//   1)  FLADDR addr    - target address to write to
//                        range is 0 to (FLASH_TEMP-1)
//
// This routine erases the FLASH page containing the linear FLASH address
// <addr>.
//
//-----------------------------------------------------------------------------
void FLASH_PageErase (FLADDR addr)
{
   bit EA_SAVE = EA;                   // Preserve EA
   char xdata * data pwrite;           // FLASH write pointer

   EA = 0;                             // Disable interrupts
   // change clock speed to slow, then restore later

   VDM0CN = 0x80;                      // Enable VDD monitor


   RSTSRC = 0x02;                      // enable VDD monitor as a reset source

   pwrite = (char xdata *) addr;

   FLKEY  = 0xA5;                      // Key Sequence 1
   FLKEY  = 0xF1;                      // Key Sequence 2
   PSCTL |= 0x03;                      // PSWE = 1; PSEE = 1


   VDM0CN = 0x80;                      // Enable VDD monitor


   RSTSRC = 0x02;                      // Enable VDD monitor as a reset source
   *pwrite = 0;                        // Initiate page erase

   PSCTL &= ~0x03;                     // PSWE = 0; PSEE = 0

   EA = EA_SAVE;                       // Restore interrupts
}

//-----------------------------------------------------------------------------
// End Of File
//-----------------------------------------------------------------------------
```

7.12.2. F340_FlashPrimitives.h

```
//-----------------------------------------------------------------------------
//  F340_FlashPrimitives.h
//-----------------------------------------------------------------------------
// Copyright 2006 Silicon Laboratories, Inc.
// http://www.silabs.com
//
// Program Description:
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// FID:            34X000025
// Target:         C8051F34x
// Tool chain:     Keil C51 8.00 / Keil EVAL C51
// Command Line:   None
//
// Release 1.0
//    -Initial Revision (GP)
//    -30 JAN 2006
//

#ifndef F340_FLASHPRIMITIVES_H
#define F340_FLASHPRIMITIVES_H

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

typedef unsigned long ULONG;
typedef unsigned int  UINT;
typedef unsigned char UCHAR;

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

#ifndef FLASH_PAGESIZE
#define FLASH_PAGESIZE 512
#endif

#ifndef FLASH_TEMP
#define FLASH_TEMP 0x0F800L
#endif

#ifndef FLASH_LAST
#define FLASH_LAST 0x0FA00L
#endif

typedef UINT FLADDR;

//-----------------------------------------------------------------------------
// Exported Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
extern void FLASH_ByteWrite (FLADDR addr, char byte);
extern unsigned char FLASH_ByteRead (FLADDR addr);
extern void FLASH_PageErase (FLADDR addr);

#endif                                 // F340_FLASHPRIMITIVES_H

//-----------------------------------------------------------------------------
// End Of File
//-----------------------------------------------------------------------------
```

7.12.3. F340_FlashUtils.c

```
//-----------------------------------------------------------------------------
//  F340_FlashUtils.c
//-----------------------------------------------------------------------------
// Copyright 2006 Silicon Laboratories, Inc.
// http://www.silabs.com
//
// Program Description:
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// FID:            34X000026
// Target:         C8051F34x
// Tool chain:     Keil C51 8.00 / Keil EVAL C51
// Command Line:   None
//
// Release 1.0
//    -Initial Revision (GP)
//    -30 JAN 2006
//

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F340_FlashPrimitives.h"
#include "F340_FlashUtils.h"

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
void FLASH_Write (FLADDR dest, char *src, unsigned numbytes);
char * FLASH_Read (char *dest, FLADDR src, unsigned numbytes);
void FLASH_Clear (FLADDR addr, unsigned numbytes);

// FLASH update/copy routines
void FLASH_Update (FLADDR dest, char *src, unsigned numbytes);
void FLASH_Copy (FLADDR dest, FLADDR src, unsigned numbytes);

// FLASH test routines
void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill);

//-----------------------------------------------------------------------------
// Global Variables
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH Routines
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH_Clear
//-----------------------------------------------------------------------------
```

```
//
// Return Value : None
// Parameters   :
//   1)  FLADDR dest - target address where to start clearing bytes
//                     range is 0 to (FLASH_TEMP-1)
//   2)  unsigned numbytes   - number of bytes to clear
//                     range is 1 to <FLASH_PAGESIZE>
//
// This routine erases <numbytes> starting from the FLASH addressed by
// <dest> by performing a read-modify-write operation using <FLASH_TEMP> as
// a temporary holding area.
//
//-------------------------------------------------------------------------

void FLASH_Clear (FLADDR dest, unsigned numbytes)
{
   FLADDR dest_1_page_start;           // First address in 1st page
                                       // containing <dest>
   FLADDR dest_1_page_end;             // Last address in 1st page
                                       // containing <dest>
   FLADDR dest_2_page_start;           // First address in 2nd page
                                       // containing <dest>
   FLADDR dest_2_page_end;             // Last address in 2nd page
                                       // containing <dest>
   unsigned numbytes_remainder;        // When crossing page boundary,
                                       // number of <src> bytes on 2nd page
   unsigned FLASH_pagesize;            // Size of FLASH page to update

   FLADDR  wptr;                       // Write address
   FLADDR  rptr;                       // Read address

   unsigned length;

   FLASH_pagesize = FLASH_PAGESIZE;

   dest_1_page_start = dest & ~(FLASH_pagesize - 1);
   dest_1_page_end = dest_1_page_start + FLASH_pagesize - 1;
   dest_2_page_start = (dest + numbytes)  & ~(FLASH_pagesize - 1);
   dest_2_page_end = dest_2_page_start + FLASH_pagesize - 1;

   if (dest_1_page_end == dest_2_page_end) {

      // 1. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP);

      // 2. Copy bytes from first byte of dest page to dest-1 to Scratch page

      wptr = FLASH_TEMP;
      rptr = dest_1_page_start;
      length = dest - dest_1_page_start;
      FLASH_Copy (wptr, rptr, length);

      // 3. Copy from (dest+numbytes) to dest_page_end to Scratch page

      wptr = FLASH_TEMP + dest - dest_1_page_start + numbytes;
      rptr = dest + numbytes;
      length = dest_1_page_end - dest - numbytes + 1;
      FLASH_Copy (wptr, rptr, length);

      // 4. Erase destination page
      FLASH_PageErase (dest_1_page_start);

      // 5. Copy Scratch page to destination page
      wptr = dest_1_page_start;
```

```
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, rptr, length);

   } else {                            // Value crosses page boundary
      // 1. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP);

      // 2. Copy bytes from first byte of dest page to dest-1
  //    to Scratch page

      wptr = FLASH_TEMP;
      rptr = dest_1_page_start;
      length = dest - dest_1_page_start;
      FLASH_Copy (wptr, rptr, length);

      // 3. Erase destination page 1
      FLASH_PageErase (dest_1_page_start);

      // 4. Copy Scratch page to destination page 1
      wptr = dest_1_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, rptr, length);

      // now handle 2nd page

      // 5. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP);

      // 6. Copy bytes from numbytes remaining to dest-2_page_end
  //    to Scratch page

      numbytes_remainder = numbytes - (dest_1_page_end - dest + 1);
      wptr = FLASH_TEMP + numbytes_remainder;
      rptr = dest_2_page_start + numbytes_remainder;
      length = FLASH_pagesize - numbytes_remainder;
      FLASH_Copy (wptr, rptr, length);

      // 7. Erase destination page 2
      FLASH_PageErase (dest_2_page_start);

      // 8. Copy Scratch page to destination page 2
      wptr = dest_2_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, rptr, length);
   }
}

//-------------------------------------------------------------------------
// FLASH_Update
//-------------------------------------------------------------------------
//
// Return Value : None
// Parameters   :
//   1)  FLADDR dest - target address where to start updating bytes
//                     range is 0 to (FLASH_TEMP-1)
//   2)  char *src   - pointer to address where source bytes are located
//   3)  unsigned numbytes  - number of bytes to update
//                            range is 1 to <FLASH_PAGESIZE>
//
// This routine replaces <numbytes> from <src> to the FLASH addressed by
// <dest>.  This function calls FLASH_Clear() to handle the dirty work of
```

```
// initializing all <dest> bytes to 0xff's prior to copying the bytes from
// <src> to <dest>.
//
//-----------------------------------------------------------------------------

void FLASH_Update (FLADDR dest, char *src, unsigned numbytes)
{
   // 1. Erase <numbytes> starting from <dest>
   FLASH_Clear (dest, numbytes);

   // 2. Write <numbytes> from <src> to <dest>
   FLASH_Write (dest, src, numbytes);
}

//-----------------------------------------------------------------------------
// FLASH_Write
//-----------------------------------------------------------------------------
//
// Return Value : None
// Parameters   :
//   1)  FLADDR dest - target address where to start writing bytes
//                     range is 0 to (FLASH_TEMP-1)
//   2)  char *src   - pointer to address where source bytes are located
//   3)  unsigned numbytes   - number of bytes to write
//                     range is limited by Flash size and starting location
//
// This routine copies <numbytes> from <src> to the linear FLASH address
// <dest>.
//
//-----------------------------------------------------------------------------
void FLASH_Write (FLADDR dest, char *src, unsigned numbytes)
{
   FLADDR i;

   for (i = dest; i < dest+numbytes; i++) {
      FLASH_ByteWrite (i, *src++);
   }
}
//-----------------------------------------------------------------------------
// FLASH_Read
//-----------------------------------------------------------------------------
//
// Return Value :
//   1)  char *     -  pointer to bytes read back
// Parameters   :
//   1)  char *dest -  target address where to store bytes read from Flash
//   2)  char *src   - pointer to address where source bytes are located
//   3)  unsigned numbytes   - number of bytes to read
//                     range is limited by Flash size and starting location
//
// This routine copies <numbytes> from the linear FLASH address <src> to
// <dest>.
//
//-----------------------------------------------------------------------------
char * FLASH_Read (char *dest, FLADDR src, unsigned numbytes)
{
   FLADDR i;

   for (i = 0; i < numbytes; i++) {
      *dest++ = FLASH_ByteRead (src+i);
   }
   return dest;
}
```

```
//-----------------------------------------------------------------------------
// FLASH_Copy
//-----------------------------------------------------------------------------
//
// Return Value : None
// Parameters   :
//   1)  FLADDR dest -  target address where to copy the source bytes
//   2)  FLADDR src   - target address where source bytes are located
//   3)  unsigned numbytes   - number of bytes to read
//                     range is limited by Flash size and starting location
//
// This routine copies <numbytes> from <src> to the linear FLASH address
// <dest>.
//
//-----------------------------------------------------------------------------

void FLASH_Copy (FLADDR dest, FLADDR src, unsigned numbytes)
{
   FLADDR i;

   for (i = 0; i < numbytes; i++) {

      FLASH_ByteWrite ((FLADDR) dest+i, FLASH_ByteRead((FLADDR) src+i));
   }
}


//-----------------------------------------------------------------------------
// FLASH_Fill
//-----------------------------------------------------------------------------
//
// Return Value : None
// Parameters   :
//   1)  FLADDR addr   - target address where to start filling
//   2)  ULONG length  - number of bytes to fill
//                       range is limited by Flash size and starting point
//   3)  UCHAR fill    - char to fill
//                       range is 0x00 to 0xFF
//
// This routine fills the FLASH beginning at <addr> with <length> bytes.
//
//-----------------------------------------------------------------------------
void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill)
{
   FLADDR i;

   for (i = 0; i < length; i++) {
      FLASH_ByteWrite (addr+i, fill);
   }
}

//-----------------------------------------------------------------------------
// End Of File
//-----------------------------------------------------------------------------
```

7.12.4. F340_FlashUtils.h

```
//-----------------------------------------------------------------------------
//  F340_FlashUtils.h
//-----------------------------------------------------------------------------
// Copyright 2006 Silicon Laboratories, Inc.
// http://www.silabs.com
//
// Program Description:
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// FID:            34X000027
// Target:         C8051F34x
// Tool chain:     Keil C51 8.00 / Keil EVAL C51
// Command Line:   None
//
// Release 1.0
//    -Initial Revision (GP)
//    -30 JAN 2006
//

#ifndef F340_FLASHUTILS_H
#define F340_FLASHUTILS_H

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F340_FlashPrimitives.h"

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Exported Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
extern void FLASH_Write (FLADDR dest, char *src, unsigned numbytes);
extern char * FLASH_Read (char *dest, FLADDR src, unsigned numbytes);
extern void FLASH_Clear (FLADDR addr, unsigned numbytes);

// FLASH update/copy routines

// copy with destination preservation
extern void FLASH_Update (FLADDR dest, char *src, unsigned numbytes);
// low-level FLASH/FLASH byte copy
extern void FLASH_Copy (FLADDR dest, FLADDR src, unsigned numbytes);

// FLASH test routines
extern void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill);

#endif                                 // F340_FLASHUTILS_H

//-----------------------------------------------------------------------------
// End Of File
//-----------------------------------------------------------------------------
```

7.13. ‘F350

7.13.1. F350_FlashPrimitives.c

```
//-----------------------------------------------------------------------------
// F350_FlashPrimitives.c
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F35x
// Tool chain: KEIL C51 7.06
//
// Release 1.0
//

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F350_FlashPrimitives.h"
#include <c8051F350.h>

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
void FLASH_ByteWrite (FLADDR addr, char byte);
unsigned char FLASH_ByteRead (FLADDR addr);
void FLASH_PageErase (FLADDR addr);

//-----------------------------------------------------------------------------
// Global Variables
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH Routines
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH_ByteWrite
//-----------------------------------------------------------------------------
//
// This routine writes <byte> to the linear FLASH address <addr>.
//
void FLASH_ByteWrite (FLADDR addr, char byte)
{
   bit EA_SAVE = EA;                   // preserve EA
   char xdata * data pwrite;           // FLASH write pointer

   EA = 0;                             // disable interrupts
```

```
   // change clock speed to slow, then restore later

   VDM0CN = 0x80;                       // enable VDD monitor


   RSTSRC = 0x02;                       // enable VDD monitor as a reset source

   pwrite = (char xdata *) addr;

   FLKEY  = 0xA5;                       // Key Sequence 1
   FLKEY  = 0xF1;                       // Key Sequence 2
   PSCTL |= 0x01;                       // PSWE = 1


   VDM0CN = 0x80;                       // enable VDD monitor


   RSTSRC = 0x02;                       // enable VDD monitor as a reset source

   *pwrite = byte;                      // write the byte

   PSCTL &= ~0x01;                      // PSWE = 0

   EA = EA_SAVE;                        // restore interrupts
}

//-----------------------------------------------------------------------------
// FLASH_ByteRead
//-----------------------------------------------------------------------------
//
// This routine reads a <byte> from the linear FLASH address <addr>.
//
unsigned char FLASH_ByteRead (FLADDR addr)
{
   bit EA_SAVE = EA;                    // preserve EA
   char code * data pread;              // FLASH read pointer
   unsigned char byte;

   EA = 0;                              // disable interrupts

   pread = (char code *) addr;

   byte = *pread;                       // read the byte

   EA = EA_SAVE;                        // restore interrupts

   return byte;
}

//-----------------------------------------------------------------------------
// FLASH_PageErase
//-----------------------------------------------------------------------------
//
// This routine erases the FLASH page containing the linear FLASH address
// <addr>.
//
void FLASH_PageErase (FLADDR addr)
{
   bit EA_SAVE = EA;                    // preserve EA
   char xdata * data pwrite;            // FLASH write pointer

   EA = 0;                             // disable interrupts
   // change clock speed to slow, then restore later
```

```
   VDM0CN = 0x80;                      // enable VDD monitor


   RSTSRC = 0x02;                   // enable VDD monitor as a reset source

   pwrite = (char xdata *) addr;

   FLKEY  = 0xA5;                   // Key Sequence 1
   FLKEY  = 0xF1;                   // Key Sequence 2
   PSCTL |= 0x03;                   // PSWE = 1; PSEE = 1


   VDM0CN = 0x80;                      // enable VDD monitor


   RSTSRC = 0x02;                   // enable VDD monitor as a reset source
   *pwrite = 0;                     // initiate page erase

   PSCTL &= ~0x03;                  // PSWE = 0; PSEE = 0

   EA = EA_SAVE;                    // restore interrupts
}
```

7.13.2. F350_FlashPrimitives.h

```
//-----------------------------------------------------------------------------
// F350_FlashPrimitives.h
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F35x
// Tool chain: KEIL C51 7.06
//
// Release 1.0
//

#ifndef F350_FLASHPRIMITIVES_H
#define F350_FLASHPRIMITIVES_H

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

typedef unsigned long ULONG;
typedef unsigned int UINT;
typedef unsigned char UCHAR;

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

#ifndef FLASH_PAGESIZE
#define FLASH_PAGESIZE 512
#endif

#ifndef FLASH_TEMP
#define FLASH_TEMP 0x01a00L
#endif

#ifndef FLASH_LAST
#define FLASH_LAST 0x01c00L
#endif

typedef UINT FLADDR;

//-----------------------------------------------------------------------------
// Exported Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
extern void FLASH_ByteWrite (FLADDR addr, char byte);
extern unsigned char FLASH_ByteRead (FLADDR addr);
extern void FLASH_PageErase (FLADDR addr);

#endif // F350_FLASHPRIMITIVES_H
```

7.13.3. F350_FlashUtils.c

```
//-----------------------------------------------------------------------------
// F350_FlashUtils.c
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F35x
// Tool chain: KEIL C51 7.06
//
// Release 1.0
//

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F350_FlashPrimitives.h"
#include "F350_FlashUtils.h"

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
void FLASH_Write (FLADDR dest, char *src, unsigned numbytes);
char * FLASH_Read (char *dest, FLADDR src, unsigned numbytes);
void FLASH_Clear (FLADDR addr, unsigned numbytes);

// FLASH update/copy routines
void FLASH_Update (FLADDR dest, char *src, unsigned numbytes);
void FLASH_Copy (FLADDR dest, FLADDR src, unsigned numbytes);

// FLASH test routines
void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill);

//-----------------------------------------------------------------------------
// Global Variables
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH Routines -- no SFLE
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// FLASH_Clear
//-----------------------------------------------------------------------------
//
// This routine erases <numbytes> starting from the FLASH addressed by
// <dest> by performing a read-modify-write operation using <FLASH_TEMP> as
// a temporary holding area.  This function accepts <numbytes> up to
```

```
// <FLASH_PAGESIZE>.
//
void FLASH_Clear (FLADDR dest, unsigned numbytes)
{
   FLADDR dest_1_page_start;           // first address in 1st page
                                       // containing <dest>
   FLADDR dest_1_page_end;             // last address in 1st page
                                       // containing <dest>
   FLADDR dest_2_page_start;           // first address in 2nd page
                                       // containing <dest>
   FLADDR dest_2_page_end;             // last address in 2nd page
                                       // containing <dest>
   unsigned numbytes_remainder;        // when crossing page boundary,
                                       // number of <src> bytes on 2nd page
   unsigned FLASH_pagesize;            // size of FLASH page to update

   FLADDR  wptr;                       // write address
   FLADDR  rptr;                       // read address

   unsigned length;

   FLASH_pagesize = FLASH_PAGESIZE;

   dest_1_page_start = dest & ~(FLASH_pagesize - 1);
   dest_1_page_end = dest_1_page_start + FLASH_pagesize - 1;
   dest_2_page_start = (dest + numbytes)  & ~(FLASH_pagesize - 1);
   dest_2_page_end = dest_2_page_start + FLASH_pagesize - 1;

   if (dest_1_page_end == dest_2_page_end) {

      // 1. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP);

      // 2. Copy bytes from first byte of dest page to dest-1 to Scratch page

      wptr = FLASH_TEMP;
      rptr = dest_1_page_start;
      length = dest - dest_1_page_start;
      FLASH_Copy (wptr, rptr, length);

      // 3. Copy from (dest+numbytes) to dest_page_end to Scratch page

      wptr = FLASH_TEMP + dest - dest_1_page_start + numbytes;
      rptr = dest + numbytes;
      length = dest_1_page_end - dest - numbytes + 1;
      FLASH_Copy (wptr, rptr, length);

      // 4. Erase destination page
      FLASH_PageErase (dest_1_page_start);

      // 5. Copy Scratch page to destination page
      wptr = dest_1_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, rptr, length);

   } else {                            // value crosses page boundary
      // 1. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP);

      // 2. Copy bytes from first byte of dest page to dest-1 to Scratch page

      wptr = FLASH_TEMP;
      rptr = dest_1_page_start;
```

```
      length = dest - dest_1_page_start;
      FLASH_Copy (wptr, rptr, length);

      // 3. Erase destination page 1
      FLASH_PageErase (dest_1_page_start);

      // 4. Copy Scratch page to destination page 1
      wptr = dest_1_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, rptr, length);

      // now handle 2nd page

      // 5. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP);

      // 6. Copy bytes from numbytes remaining to dest-2_page_end to Scratch page

      numbytes_remainder = numbytes - (dest_1_page_end - dest + 1);
      wptr = FLASH_TEMP + numbytes_remainder;
      rptr = dest_2_page_start + numbytes_remainder;
      length = FLASH_pagesize - numbytes_remainder;
      FLASH_Copy (wptr, rptr, length);

      // 7. Erase destination page 2
      FLASH_PageErase (dest_2_page_start);

      // 8. Copy Scratch page to destination page 2
      wptr = dest_2_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, rptr, length);
   }
}



//-----------------------------------------------------------------------------
// FLASH_Update
//-----------------------------------------------------------------------------
//
// This routine replaces <numbytes> from <src> to the FLASH addressed by
// <dest>.  This function calls FLASH_Clear() to handle the dirty work of
// initializing all <dest> bytes to 0xff's prior to copying the bytes from
// <src> to <dest>. This function accepts <numbytes> up to <FLASH_PAGESIZE>.
//
void FLASH_Update (FLADDR dest, char *src, unsigned numbytes)
{
   // 1. Erase <numbytes> starting from <dest>
   FLASH_Clear (dest, numbytes);

   // 2. Write <numbytes> from <src> to <dest>
   FLASH_Write (dest, src, numbytes);
}



//-----------------------------------------------------------------------------
// FLASH_Write
//-----------------------------------------------------------------------------
//
// This routine copies <numbytes> from <src> to the linear FLASH address
// <dest>.
```

```
//
void FLASH_Write (FLADDR dest, char *src, unsigned numbytes)
{
   FLADDR i;

   for (i = dest; i < dest+numbytes; i++) {
      FLASH_ByteWrite (i, *src++);
   }
}

//-----------------------------------------------------------------------------
// FLASH_Read
//-----------------------------------------------------------------------------
//
// This routine copies <numbytes> from the linear FLASH address <src> to
// <dest>.
//
char * FLASH_Read (char *dest, FLADDR src, unsigned numbytes)
{
   FLADDR i;

   for (i = 0; i < numbytes; i++) {
      *dest++ = FLASH_ByteRead (src+i);
   }
   return dest;
}

//-----------------------------------------------------------------------------
// FLASH_Copy
//-----------------------------------------------------------------------------
//
// This routine copies <numbytes> from <src> to the linear FLASH address
// <dest>.
//
void FLASH_Copy (FLADDR dest, FLADDR src, unsigned numbytes)
{
   FLADDR i;

   for (i = 0; i < numbytes; i++) {

      FLASH_ByteWrite ((FLADDR) dest+i, FLASH_ByteRead((FLADDR) src+i));
   }
}



//-----------------------------------------------------------------------------
// FLASH_Fill
//-----------------------------------------------------------------------------
//
// This routine fills the FLASH beginning at <addr> with <lenght> bytes.
//
void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill)
{
   FLADDR i;

   for (i = 0; i < length; i++) {
      FLASH_ByteWrite (addr+i, fill);
   }
}
```

7.13.4. F330_FlashUtils.h

```
//-----------------------------------------------------------------------------
// F330_FlashUtils.h
//-----------------------------------------------------------------------------
// Copyright 2004 Silicon Laboratories, Inc.
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// AUTH: BW & GP
// DATE: 21 JUL 04
//
// Target: C8051F35x
// Tool chain: KEIL C51 7.06
//
// Release 1.0
//

#ifndef F350_FLASHUTILS_H
#define F350_FLASHUTILS_H

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F350_FlashPrimitives.h"

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Exported Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
extern void FLASH_Write (FLADDR dest, char *src, unsigned numbytes);
extern char * FLASH_Read (char *dest, FLADDR src, unsigned numbytes);
extern void FLASH_Clear (FLADDR addr, unsigned numbytes);

// FLASH update/copy routines
extern void FLASH_Update (FLADDR dest, char *src, unsigned numbytes);   // copy with destina-
tion preservation
extern void FLASH_Copy (FLADDR dest, FLADDR src, unsigned numbytes);     // low-level FLASH/
FLASH byte copy

// FLASH test routines
extern void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill);

#endif // F350_FLASHUTILS_H
```

7.14. ‘F360

7.14.1. F360_FlashPrimitives.c

```
//-----------------------------------------------------------------------------
// F360_FlashPrimitives.c
//-----------------------------------------------------------------------------
// Copyright 2006 Silicon Laboratories, Inc.
// http://www.silabs.com
//
// Program Description:
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// FID:            36X000004
// Target:         C8051F36x
// Tool chain:     Keil C51 8.00
// Command Line:   None
//
//
// Release 1.0
//    -Initial Revision (GP / TP)
//    -24 OCT 2006
//

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include <C8051F360.h>
#include “F360_FlashPrimitives.h”

//-----------------------------------------------------------------------------
// Function Prototypes
//-----------------------------------------------------------------------------

void          FLASH_ByteWrite (FLADDR addr, char byte);
unsigned char FLASH_ByteRead  (FLADDR addr);
void          FLASH_PageErase (FLADDR addr);

//-----------------------------------------------------------------------------
// FLASH_ByteWrite
//-----------------------------------------------------------------------------
//
// Return Value : None
// Parameters   :
//   1) FLADDR addr - address of the byte to write to
//                    valid range is from 0x0000 to 0x7DFE for 32K devices
//                    valid range is from 0x0000 to 0x3FFE for 16K devices
//   2) char byte - byte to write to Flash.
//
// This routine writes <byte> to the linear FLASH address <addr>.
//-----------------------------------------------------------------------------

void FLASH_ByteWrite (FLADDR addr, char byte)
{
   bit EA_SAVE = EA;                   // Preserve EA
   char xdata * data pwrite;           // FLASH write pointer

   EA = 0;                             // Disable interrupts

   VDM0CN = 0xA0;                      // Enable VDD monitor and high threshold

   RSTSRC = 0x02;                      // Enable VDD monitor as a reset source
```

```
   pwrite = (char xdata *) addr;

   FLKEY  = 0xA5;                      // Key Sequence 1
   FLKEY  = 0xF1;                      // Key Sequence 2
   PSCTL |= 0x01;                      // PSWE = 1 which enables writes

   VDM0CN = 0xA0;                      // Enable VDD monitor and high threshold

   RSTSRC = 0x02;                      // Enable VDD monitor as a reset source

   *pwrite = byte;                     // Write the byte

   PSCTL &= ~0x01;                     // PSWE = 0 which disable writes

   EA = EA_SAVE;                       // Restore interrupts
}

//-----------------------------------------------------------------------------
// FLASH_ByteRead
//-----------------------------------------------------------------------------
//
// Return Value :
//      unsigned char - byte read from Flash
// Parameters   :
//   1) FLADDR addr - address of the byte to read to
//                    valid range is from 0x0000 to 0x7DFE for 32K devices
//                    valid range is from 0x0000 to 0x3FFE for 16K devices
//
// This routine reads a <byte> from the linear FLASH address <addr>.
//-----------------------------------------------------------------------------

unsigned char FLASH_ByteRead (FLADDR addr)
{
   bit EA_SAVE = EA;                   // Preserve EA
   char code * data pread;             // FLASH read pointer
   unsigned char byte;

   EA = 0;                             // Disable interrupts

   pread = (char code *) addr;

   byte = *pread;                      // Read the byte

   EA = EA_SAVE;                       // Restore interrupts

   return byte;
}

//-----------------------------------------------------------------------------
// FLASH_PageErase
//-----------------------------------------------------------------------------
//
// Return Value : None
// Parameters   :
//   1) FLADDR addr - address of any byte in the page to erase
//                    valid range is from 0x0000 to 0x7BFF for 32K devices
//                    valid range is from 0x0000 to 0x3DFF for 16K devices
//
// This routine erases the FLASH page containing the linear FLASH address
// <addr>.  Note that the page of Flash containing the Lock Byte cannot be
// erased if the Lock Byte is set.
//
//-----------------------------------------------------------------------------
```

```
void FLASH_PageErase (FLADDR addr)
{
   bit EA_SAVE = EA;                   // Preserve EA
   char xdata * data pwrite;           // FLASH write pointer

   EA = 0;                             // Disable interrupts

   VDM0CN = 0x80;                      // Enable VDD monitor

   RSTSRC = 0x02;                      // Enable VDD monitor as a reset source

   pwrite = (char xdata *) addr;

   FLKEY  = 0xA5;                      // Key Sequence 1
   FLKEY  = 0xF1;                      // Key Sequence 2
   PSCTL |= 0x03;                      // PSWE = 1; PSEE = 1


   VDM0CN = 0x80;                      // Enable VDD monitor

   RSTSRC = 0x02;                      // Enable VDD monitor as a reset source
   *pwrite = 0;                        // Initiate page erase

   PSCTL &= ~0x03;                     // PSWE = 0; PSEE = 0

   EA = EA_SAVE;                       // Restore interrupts
}

//-----------------------------------------------------------------------------
// End Of File
//-----------------------------------------------------------------------------
```

7.14.2. F360_FlashPrimitives.h

```
//-----------------------------------------------------------------------------
// F360_FlashPrimitives.h
//-----------------------------------------------------------------------------
// Copyright 2006 Silicon Laboratories, Inc.
// http://www.silabs.com
//
// Program Description:
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// FID:            36X000005
// Target:         C8051F36x
// Tool chain:     Keil C51 8.00
// Command Line:   None
//
//
// Release 1.0
//    -Initial Revision (GP / TP)
//    -24 OCT 2006
//

//-----------------------------------------------------------------------------
// Open Header #define
//-----------------------------------------------------------------------------

#ifndef _F360_FLASHPRIMITIVES_H_
#define _F360_FLASHPRIMITIVES_H_

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

typedef unsigned long ULONG;
typedef unsigned int  UINT;
typedef unsigned char UCHAR;
typedef UINT FLADDR;

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

#ifndef FLASH_PAGESIZE
#define FLASH_PAGESIZE 512
#endif

#ifndef FLASH_TEMP
#define FLASH_TEMP 0x7400L             // For 32K Flash devices
//#define FLASH_TEMP 0x3800L           // For 16K Flash devices
#endif

#ifndef FLASH_LAST
#define FLASH_LAST 0x7800L             // For 32K Flash devices
//#define FLASH_LAST 0x3C00L           // For 16K Flash devices
#endif

//-----------------------------------------------------------------------------
// Exported Function Prototypes
//-----------------------------------------------------------------------------

void          FLASH_ByteWrite (FLADDR addr, char byte);
unsigned char FLASH_ByteRead  (FLADDR addr);
void          FLASH_PageErase (FLADDR addr);
```

```
//-----------------------------------------------------------------------------
// Close Header #define
//-----------------------------------------------------------------------------

#endif    // _F360_FLASHPRIMITIVES_H_

//-----------------------------------------------------------------------------
// End Of File
//-----------------------------------------------------------------------------
```

7.14.3. F360_FlashUtils.c

```
//-----------------------------------------------------------------------------
// F360_FlashUtils.c
//-----------------------------------------------------------------------------
// Copyright 2006 Silicon Laboratories, Inc.
// http://www.silabs.com
//
// Program Description:
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// FID:            36X000006
// Target:         C8051F36x
// Tool chain:     Keil C51 8.00
// Command Line:   None
//
//
// Release 1.0
//    -Initial Revision (GP / TP)
//    -24 OCT 2006
//

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F360_FlashPrimitives.h"
#include "F360_FlashUtils.h"

//-----------------------------------------------------------------------------
// Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
void FLASH_Write (FLADDR dest, char *src, unsigned int numbytes);
char * FLASH_Read (char *dest, FLADDR src, unsigned int numbytes);
void FLASH_Clear (FLADDR addr, unsigned int numbytes);

// FLASH update/copy routines
void FLASH_Update (FLADDR dest, char *src, unsigned int numbytes);
void FLASH_Copy (FLADDR dest, FLADDR src, unsigned int numbytes);

// FLASH test routines
void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill);

//-----------------------------------------------------------------------------
// FLASH_Clear
//-----------------------------------------------------------------------------
//
// Return Value : None
// Parameters   :
//   1) FLADDR addr - address of the byte to write to
//                    valid range is 0x0000 to 0x79FF for 32K devices
//                    valid range is 0x0000 to 0x3BFF for 16K devices
//   2) unsigned int numbytes - the number of bytes to clear to 0xFF
//                    valid range is 0 to FLASH_PAGESIZE
//
// This routine erases <numbytes> starting from the FLASH addressed by
// <dest> by performing a read-modify-write operation using <FLASH_TEMP> as
// a temporary holding area.
// <addr> + <numbytes> must be less than 0x79FF/0x3BFF.
//
//-----------------------------------------------------------------------------
```

```
void FLASH_Clear (FLADDR dest, unsigned int numbytes)
{
   FLADDR dest_1_page_start;           // First address in 1st page
                                       // containing <dest>
   FLADDR dest_1_page_end;             // Last address in 1st page
                                       // containing <dest>
   FLADDR dest_2_page_start;           // First address in 2nd page
                                       // containing <dest>
   FLADDR dest_2_page_end;             // Last address in 2nd page
                                       // containing <dest>
   unsigned numbytes_remainder;        // When crossing page boundary,
                                       // number of <src> bytes on 2nd page
   unsigned FLASH_pagesize;            // Size of FLASH page to update

   FLADDR  wptr;                       // Write address
   FLADDR  rptr;                       // Read address

   unsigned length;

   FLASH_pagesize = FLASH_PAGESIZE;

   dest_1_page_start = dest & ~(FLASH_pagesize - 1);
   dest_1_page_end   = dest_1_page_start + FLASH_pagesize - 1;
   dest_2_page_start = (dest + numbytes)  & ~(FLASH_pagesize - 1);
   dest_2_page_end   = dest_2_page_start + FLASH_pagesize - 1;

   if (dest_1_page_end == dest_2_page_end)
   {
      // 1. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP);

      // 2. Copy bytes from first byte of dest page to dest-1 to Scratch page

      wptr = FLASH_TEMP;
      rptr = dest_1_page_start;
      length = dest - dest_1_page_start;
      FLASH_Copy (wptr, rptr, length);

      // 3. Copy from (dest+numbytes) to dest_page_end to Scratch page

      wptr = FLASH_TEMP + dest - dest_1_page_start + numbytes;
      rptr = dest + numbytes;
      length = dest_1_page_end - dest - numbytes + 1;
      FLASH_Copy (wptr, rptr, length);

      // 4. Erase destination page
      FLASH_PageErase (dest_1_page_start);

      // 5. Copy Scratch page to destination page
      wptr = dest_1_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, rptr, length);

   }
   else
   {                              // value crosses page boundary
      // 1. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP);

      // 2. Copy bytes from first byte of dest page to dest-1 to temp page

      wptr = FLASH_TEMP;
```

```
      rptr = dest_1_page_start;
      length = dest - dest_1_page_start;
      FLASH_Copy (wptr, rptr, length);

      // 3. Erase destination page 1
      FLASH_PageErase (dest_1_page_start);

      // 4. Copy Scratch page to destination page 1
      wptr = dest_1_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, rptr, length);

      // Now handle 2nd page

      // 5. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP);

      // 6. Copy bytes from numbytes remaining to dest-2_page_end to temp page

      numbytes_remainder = numbytes - (dest_1_page_end - dest + 1);
      wptr = FLASH_TEMP + numbytes_remainder;
      rptr = dest_2_page_start + numbytes_remainder;
      length = FLASH_pagesize - numbytes_remainder;
      FLASH_Copy (wptr, rptr, length);

      // 7. Erase destination page 2
      FLASH_PageErase (dest_2_page_start);

      // 8. Copy Scratch page to destination page 2
      wptr = dest_2_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, rptr, length);
   }
}

//-----------------------------------------------------------------------------
// FLASH_Update
//-----------------------------------------------------------------------------
//
// Return Value : None
// Parameters   :
//   1) FLADDR dest - starting address of the byte(s) to write to
//                    valid range is 0x0000 to 0x79FF for 32K devices
//                    valid range is 0x0000 to 0x3BFF for 16K devices
//   2) char *src - pointer to source bytes
//   3) unsigned int numbytes - the number of bytes to update
//                              valid range is 0 to FLASH_PAGESIZE
//
// This routine replaces <numbytes> from <src> to the FLASH addressed by
// <dest>.  This function calls FLASH_Clear() to initialize all <dest> bytes
// to 0xff's prior to copying the bytes from <src> to <dest>.
// <dest> + <numbytes> must be less than 0x7DFF/0x3FFF.
//
//-----------------------------------------------------------------------------

void FLASH_Update (FLADDR dest, char *src, unsigned int numbytes)
{
   // 1. Erase <numbytes> starting from <dest>
   FLASH_Clear (dest, numbytes);

   // 2. Write <numbytes> from <src> to <dest>
   FLASH_Write (dest, src, numbytes);
```

```
}


//-----------------------------------------------------------------------------
// FLASH_Write
//-----------------------------------------------------------------------------
//
// Return Value : None
// Parameters   :
//   1) FLADDR dest - starting address of the byte(s) to write to
//                    valid range is 0x0000 to 0x7DFE for 32K Flash devices
//                    valid range is 0x0000 to 0x3FFE for 16K Flash devices
//   2) char *src - pointer to source bytes
//   3) unsigned int numbytes - the number of bytes to write
//                              valid range is is range of integer
//
// This routine copies <numbytes> from <src> to the linear FLASH address
// <dest>.  The bytes must be erased to 0xFF before writing.
// <dest> + <numbytes> must be less than 0x7DFF/0x3FFF.
//
//-----------------------------------------------------------------------------

void FLASH_Write (FLADDR dest, char *src, unsigned int numbytes)
{
   FLADDR i;

   for (i = dest; i < dest+numbytes; i++) {
      FLASH_ByteWrite (i, *src++);
   }
}

//-----------------------------------------------------------------------------
// FLASH_Read
//-----------------------------------------------------------------------------
//
// Return Value : None
//   1) char *dest - pointer to destination bytes
// Parameters   :
//   1) char *dest - pointer to destination bytes
//   2) FLADDR src - address of source bytes in Flash
//                   valid range is 0x0000 to 0x7DFF for 32K Flash devices
//                   valid range is 0x0000 to 0x3FFF for 16K Flash devices
//   3) unsigned int numbytes - the number of bytes to read
//                              valid range is range of integer
//
// This routine copies <numbytes> from the linear FLASH address <src> to
// <dest>.
// <src> + <numbytes> must be less than 0x7DFF/0x3FFF.
//
//-----------------------------------------------------------------------------

char * FLASH_Read (char *dest, FLADDR src, unsigned int numbytes)
{
   FLADDR i;

   for (i = 0; i < numbytes; i++) {
      *dest++ = FLASH_ByteRead (src+i);
   }
   return dest;
}

//-----------------------------------------------------------------------------
// FLASH_Copy
//-----------------------------------------------------------------------------
```

```
//
// Return Value : None
// Parameters   :
//   1) FLADDR dest - pointer to destination bytes in Flash
//                    valid range is 0x0000 to 0x7DFF for 32K Flash devices
//                    valid range is 0x0000 to 0x3FFF for 16K Flash devices
//   2) FLADDR src - address of source bytes in Flash
//                    valid range is 0x0000 to 0x7DFF for 32K Flash devices
//                    valid range is 0x0000 to 0x3FFF for 16K Flash devices
//   3) unsigned int numbytes - the number of bytes to copy
//                              valid range is range of integer
//
// This routine copies <numbytes> from <src> to the linear FLASH address
// <dest>.  The destination bytes must be erased to 0xFF before writing.
// <src>/<dest> + <numbytes> must be less than 0x7DFF.
//
//-------------------------------------------------------------------------

void FLASH_Copy (FLADDR dest, FLADDR src, unsigned int numbytes)
{
   FLADDR i;

   for (i = 0; i < numbytes; i++) {
      FLASH_ByteWrite ((FLADDR) dest+i, FLASH_ByteRead((FLADDR) src+i));
   }
}

//-------------------------------------------------------------------------
// FLASH_Fill
//-------------------------------------------------------------------------
//
// Return Value : None
// Parameters   :
//   1) FLADDR addr - starting address of bytes to fill in Flash
//                    valid range is 0x0000 to 0x7DFF for 32K Flash devices
//                    valid range is 0x0000 to 0x3FFF for 16K Flash devices
//   2) ULONG length - number of bytes to fill
//                     range is total Flash size
//   3) UCHAR fill - the character used the Flash should be filled with
//
// This routine fills the FLASH beginning at <addr> with <length> bytes.
// The target bytes must be erased before writing to them.
// <addr> + <length> must be less than 0x7DFF.
//
//-------------------------------------------------------------------------

void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill)
{
   FLADDR i;

   for (i = 0; i < length; i++) {
      FLASH_ByteWrite (addr+i, fill);
   }
}

//-------------------------------------------------------------------------
// End Of File
//-------------------------------------------------------------------------
```

7.14.4. F360_FlashUtils.h

```
//-----------------------------------------------------------------------------
// F360_FlashUtils.h
//-----------------------------------------------------------------------------
// Copyright 2006 Silicon Laboratories, Inc.
// http://www.silabs.com
//
// Program Description:
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// FID:            36X000007
// Target:         C8051F36x
// Tool chain:     Keil C51 8.00
// Command Line:   None
//
//
// Release 1.0
//    -Initial Revision (GP / TP)
//    -24 OCT 2006
//

//-----------------------------------------------------------------------------
// Open Header #define
//-----------------------------------------------------------------------------

#ifndef _F360_FLASHUTILS_H_
#define _F360_FLASHUTILS_H_

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F360_FlashPrimitives.h"

//-----------------------------------------------------------------------------
// Exported Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
void FLASH_Write (FLADDR dest, char *src, unsigned int numbytes);
char * FLASH_Read (char *dest, FLADDR src, unsigned int numbytes);
void FLASH_Clear (FLADDR addr, unsigned int numbytes);

// FLASH update/copy routines
void FLASH_Update (FLADDR dest, char *src, unsigned int numbytes);
void FLASH_Copy (FLADDR dest, FLADDR src, unsigned int numbytes);

// FLASH test routines
void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill);

//-----------------------------------------------------------------------------
// Close Header #define
//-----------------------------------------------------------------------------

#endif    // _F360_FLASHUTILS_H_

//-----------------------------------------------------------------------------
// End Of File
//-----------------------------------------------------------------------------
```

7.15. ‘F410

7.15.1. F410_FlashPrimitives.c

```
//-----------------------------------------------------------------------------
// F410_FlashPrimitives.c
//-----------------------------------------------------------------------------
// Copyright 2006 Silicon Laboratories, Inc.
// http://www.silabs.com
//
// Program Description:
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// FID:            41X000050
// Target:         C8051F410/1/2/3
// Tool chain:     Keil C51 8.00
// Command Line:   None
//
//
// Release 1.0
//    -Initial Revision (GP)
//    -29 SEP 2006
//

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include <c8051F410.h>
#include "F410_FlashPrimitives.h"

//-----------------------------------------------------------------------------
// Function Prototypes
//-----------------------------------------------------------------------------

void          FLASH_ByteWrite (FLADDR addr, char byte);
unsigned char FLASH_ByteRead  (FLADDR addr);
void          FLASH_PageErase (FLADDR addr);

//-----------------------------------------------------------------------------
// FLASH_ByteWrite
//-----------------------------------------------------------------------------
//
// Return Value : None
// Parameters   :
//   1) FLADDR addr - address of the byte to write to
//                    valid range is from 0x0000 to 0x7DFE for 32K devices
//                    valid range is from 0x0000 to 0x3FFE for 16K devices
//   2) char byte - byte to write to Flash.
//
// This routine writes <byte> to the linear FLASH address <addr>.
//-----------------------------------------------------------------------------

void FLASH_ByteWrite (FLADDR addr, char byte)
{
   bit EA_SAVE = EA;                   // Preserve EA
   char xdata * data pwrite;           // FLASH write pointer

   EA = 0;                             // Disable interrupts

   VDM0CN = 0xA0;                      // Enable VDD monitor and high threshold

   RSTSRC = 0x02;                      // Enable VDD monitor as a reset source
```

```
   pwrite = (char xdata *) addr;

   FLKEY  = 0xA5;                      // Key Sequence 1
   FLKEY  = 0xF1;                      // Key Sequence 2
   PSCTL |= 0x01;                      // PSWE = 1 which enables writes

   VDM0CN = 0xA0;                      // Enable VDD monitor and high threshold

   RSTSRC = 0x02;                      // Enable VDD monitor as a reset source

   *pwrite = byte;                     // Write the byte

   PSCTL &= ~0x01;                     // PSWE = 0 which disable writes

   EA = EA_SAVE;                       // Restore interrupts
}

//-----------------------------------------------------------------------------
// FLASH_ByteRead
//-----------------------------------------------------------------------------
//
// Return Value :
//      unsigned char - byte read from Flash
// Parameters   :
//   1) FLADDR addr - address of the byte to read to
//                    valid range is from 0x0000 to 0x7DFE for 32K devices
//                    valid range is from 0x0000 to 0x3FFE for 16K devices
//
// This routine reads a <byte> from the linear FLASH address <addr>.
//-----------------------------------------------------------------------------

unsigned char FLASH_ByteRead (FLADDR addr)
{
   bit EA_SAVE = EA;                   // Preserve EA
   char code * data pread;             // FLASH read pointer
   unsigned char byte;

   EA = 0;                             // Disable interrupts

   pread = (char code *) addr;

   byte = *pread;                      // Read the byte

   EA = EA_SAVE;                       // Restore interrupts

   return byte;
}

//-----------------------------------------------------------------------------
// FLASH_PageErase
//-----------------------------------------------------------------------------
//
// Return Value : None
// Parameters   :
//   1) FLADDR addr - address of any byte in the page to erase
//                    valid range is from 0x0000 to 0x7BFF for 32K devices
//                    valid range is from 0x0000 to 0x3DFF for 16K devices
//
// This routine erases the FLASH page containing the linear FLASH address
// <addr>.  Note that the page of Flash containing the Lock Byte cannot be
// erased if the Lock Byte is set.
//
//-----------------------------------------------------------------------------
```

```
void FLASH_PageErase (FLADDR addr)
{
   bit EA_SAVE = EA;                   // Preserve EA
   char xdata * data pwrite;           // FLASH write pointer

   EA = 0;                             // Disable interrupts

   VDM0CN = 0xA0;                      // Enable VDD monitor and high threshold

   RSTSRC = 0x02;                      // Enable VDD monitor as a reset source

   pwrite = (char xdata *) addr;

   FLKEY  = 0xA5;                      // Key Sequence 1
   FLKEY  = 0xF1;                      // Key Sequence 2
   PSCTL |= 0x03;                      // PSWE = 1; PSEE = 1


   VDM0CN = 0xA0;                      // Enable VDD monitor and high threshold

   RSTSRC = 0x02;                      // Enable VDD monitor as a reset source
   *pwrite = 0;                        // Initiate page erase

   PSCTL &= ~0x03;                     // PSWE = 0; PSEE = 0

   EA = EA_SAVE;                       // Restore interrupts
}

//-----------------------------------------------------------------------------
// End Of File
//-----------------------------------------------------------------------------
```

7.15.2. F410_FlashPrimitives.h

```
//-----------------------------------------------------------------------------
// F410_FlashPrimitives.h
//-----------------------------------------------------------------------------
// Copyright 2006 Silicon Laboratories, Inc.
// http://www.silabs.com
//
// Program Description:
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// FID:            41X000051
// Target:         C8051F410/1/2/3
// Tool chain:     Keil C51 8.00
// Command Line:   None
//
//
// Release 1.0
//    -Initial Revision (GP)
//    -29 SEP 2006
//

//-----------------------------------------------------------------------------
// Open Header #define
//-----------------------------------------------------------------------------

#ifndef _F410_FLASHPRIMITIVES_H_
#define _F410_FLASHPRIMITIVES_H_

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

typedef unsigned long ULONG;
typedef unsigned int  UINT;
typedef unsigned char UCHAR;
typedef UINT FLADDR;

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

#ifndef FLASH_PAGESIZE
#define FLASH_PAGESIZE 512
#endif

#ifndef FLASH_TEMP
#define FLASH_TEMP 0x7a00L             // For 32K Flash devices
//#define FLASH_TEMP 0x3C00L           // For 16K Flash devices
#endif

#ifndef FLASH_LAST
#define FLASH_LAST 0x7c00L             // For 32K Flash devices
//#define FLASH_LAST 0x3E00L           // For 16K Flash devices
#endif

//-----------------------------------------------------------------------------
// Exported Function Prototypes
//-----------------------------------------------------------------------------

void          FLASH_ByteWrite (FLADDR addr, char byte);
unsigned char FLASH_ByteRead  (FLADDR addr);
void          FLASH_PageErase (FLADDR addr);
```

```
//-----------------------------------------------------------------------------
// Close Header #define
//-----------------------------------------------------------------------------

#endif    // _F410_FLASHPRIMITIVES_H_

//-----------------------------------------------------------------------------
// End Of File
//-----------------------------------------------------------------------------
```

7.15.3. F410_FlashUtils.c

```
//-----------------------------------------------------------------------------
// F410_FlashUtils.c
//-----------------------------------------------------------------------------
// Copyright 2006 Silicon Laboratories, Inc.
// http://www.silabs.com
//
// Program Description:
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// FID:            41X000053
// Target:         C8051F410/1/2/3
// Tool chain:     Keil C51 8.00
// Command Line:   None
//
//
// Release 1.0
//    -Initial Revision (GP)
//    -29 SEP 2006
//

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F410_FlashPrimitives.h"
#include "F410_FlashUtils.h"

//-----------------------------------------------------------------------------
// Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
void FLASH_Write (FLADDR dest, char *src, unsigned int numbytes);
char * FLASH_Read (char *dest, FLADDR src, unsigned int numbytes);
void FLASH_Clear (FLADDR addr, unsigned int numbytes);

// FLASH update/copy routines
void FLASH_Update (FLADDR dest, char *src, unsigned int numbytes);
void FLASH_Copy (FLADDR dest, FLADDR src, unsigned int numbytes);

// FLASH test routines
void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill);

//-----------------------------------------------------------------------------
// FLASH_Clear
//-----------------------------------------------------------------------------
//
// Return Value : None
// Parameters   :
//   1) FLADDR addr - address of the byte to write to
//                    valid range is 0x0000 to 0x79FF for 32K devices
//                    valid range is 0x0000 to 0x3BFF for 16K devices
//   2) unsigned int numbytes - the number of bytes to clear to 0xFF
//                    valid range is 0 to FLASH_PAGESIZE
//
// This routine erases <numbytes> starting from the FLASH addressed by
// <dest> by performing a read-modify-write operation using <FLASH_TEMP> as
// a temporary holding area.
// <addr> + <numbytes> must be less than 0x79FF/0x3BFF.
//
//-----------------------------------------------------------------------------
```

```
void FLASH_Clear (FLADDR dest, unsigned int numbytes)
{
   FLADDR dest_1_page_start;           // First address in 1st page
                                       // containing <dest>
   FLADDR dest_1_page_end;             // Last address in 1st page
                                       // containing <dest>
   FLADDR dest_2_page_start;           // First address in 2nd page
                                       // containing <dest>
   FLADDR dest_2_page_end;             // Last address in 2nd page
                                       // containing <dest>
   unsigned numbytes_remainder;        // When crossing page boundary,
                                       // number of <src> bytes on 2nd page
   unsigned FLASH_pagesize;            // Size of FLASH page to update

   FLADDR  wptr;                       // Write address
   FLADDR  rptr;                       // Read address

   unsigned length;

   FLASH_pagesize = FLASH_PAGESIZE;

   dest_1_page_start = dest & ~(FLASH_pagesize - 1);
   dest_1_page_end   = dest_1_page_start + FLASH_pagesize - 1;
   dest_2_page_start = (dest + numbytes)  & ~(FLASH_pagesize - 1);
   dest_2_page_end   = dest_2_page_start + FLASH_pagesize - 1;

   if (dest_1_page_end == dest_2_page_end)
   {
      // 1. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP);

      // 2. Copy bytes from first byte of dest page to dest-1 to Scratch page

      wptr = FLASH_TEMP;
      rptr = dest_1_page_start;
      length = dest - dest_1_page_start;
      FLASH_Copy (wptr, rptr, length);

      // 3. Copy from (dest+numbytes) to dest_page_end to Scratch page

      wptr = FLASH_TEMP + dest - dest_1_page_start + numbytes;
      rptr = dest + numbytes;
      length = dest_1_page_end - dest - numbytes + 1;
      FLASH_Copy (wptr, rptr, length);

      // 4. Erase destination page
      FLASH_PageErase (dest_1_page_start);

      // 5. Copy Scratch page to destination page
      wptr = dest_1_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, rptr, length);

   }
   else
   {                              // value crosses page boundary
      // 1. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP);

      // 2. Copy bytes from first byte of dest page to dest-1 to temp page

      wptr = FLASH_TEMP;
```

```
      rptr = dest_1_page_start;
      length = dest - dest_1_page_start;
      FLASH_Copy (wptr, rptr, length);

      // 3. Erase destination page 1
      FLASH_PageErase (dest_1_page_start);

      // 4. Copy Scratch page to destination page 1
      wptr = dest_1_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, rptr, length);

      // Now handle 2nd page

      // 5. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP);

      // 6. Copy bytes from numbytes remaining to dest-2_page_end to temp page

      numbytes_remainder = numbytes - (dest_1_page_end - dest + 1);
      wptr = FLASH_TEMP + numbytes_remainder;
      rptr = dest_2_page_start + numbytes_remainder;
      length = FLASH_pagesize - numbytes_remainder;
      FLASH_Copy (wptr, rptr, length);

      // 7. Erase destination page 2
      FLASH_PageErase (dest_2_page_start);

      // 8. Copy Scratch page to destination page 2
      wptr = dest_2_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, rptr, length);
   }
}

//-----------------------------------------------------------------------------
// FLASH_Update
//-----------------------------------------------------------------------------
//
// Return Value : None
// Parameters   :
//   1) FLADDR dest - starting address of the byte(s) to write to
//                    valid range is 0x0000 to 0x79FF for 32K devices
//                    valid range is 0x0000 to 0x3BFF for 16K devices
//   2) char *src - pointer to source bytes
//   3) unsigned int numbytes - the number of bytes to update
//                              valid range is 0 to FLASH_PAGESIZE
//
// This routine replaces <numbytes> from <src> to the FLASH addressed by
// <dest>.  This function calls FLASH_Clear() to initialize all <dest> bytes
// to 0xff's prior to copying the bytes from <src> to <dest>.
// <dest> + <numbytes> must be less than 0x7DFF/0x3FFF.
//
//-----------------------------------------------------------------------------

void FLASH_Update (FLADDR dest, char *src, unsigned int numbytes)
{
   // 1. Erase <numbytes> starting from <dest>
   FLASH_Clear (dest, numbytes);

   // 2. Write <numbytes> from <src> to <dest>
   FLASH_Write (dest, src, numbytes);
```

```
}


//-----------------------------------------------------------------------------
// FLASH_Write
//-----------------------------------------------------------------------------
//
// Return Value : None
// Parameters   :
//   1) FLADDR dest - starting address of the byte(s) to write to
//                    valid range is 0x0000 to 0x7DFE for 32K Flash devices
//                    valid range is 0x0000 to 0x3FFE for 16K Flash devices
//   2) char *src - pointer to source bytes
//   3) unsigned int numbytes - the number of bytes to write
//                              valid range is is range of integer
//
// This routine copies <numbytes> from <src> to the linear FLASH address
// <dest>.  The bytes must be erased to 0xFF before writing.
// <dest> + <numbytes> must be less than 0x7DFF/0x3FFF.
//
//-----------------------------------------------------------------------------

void FLASH_Write (FLADDR dest, char *src, unsigned int numbytes)
{
   FLADDR i;

   for (i = dest; i < dest+numbytes; i++) {
      FLASH_ByteWrite (i, *src++);
   }
}

//-----------------------------------------------------------------------------
// FLASH_Read
//-----------------------------------------------------------------------------
//
// Return Value : None
//   1) char *dest - pointer to destination bytes
// Parameters   :
//   1) char *dest - pointer to destination bytes
//   2) FLADDR src - address of source bytes in Flash
//                   valid range is 0x0000 to 0x7DFF for 32K Flash devices
//                   valid range is 0x0000 to 0x3FFF for 16K Flash devices
//   3) unsigned int numbytes - the number of bytes to read
//                              valid range is range of integer
//
// This routine copies <numbytes> from the linear FLASH address <src> to
// <dest>.
// <src> + <numbytes> must be less than 0x7DFF/0x3FFF.
//
//-----------------------------------------------------------------------------

char * FLASH_Read (char *dest, FLADDR src, unsigned int numbytes)
{
   FLADDR i;

   for (i = 0; i < numbytes; i++) {
      *dest++ = FLASH_ByteRead (src+i);
   }
   return dest;
}

//-----------------------------------------------------------------------------
// FLASH_Copy
//-----------------------------------------------------------------------------
```

```
//
// Return Value : None
// Parameters   :
//   1) FLADDR dest - pointer to destination bytes in Flash
//                    valid range is 0x0000 to 0x7DFF for 32K Flash devices
//                    valid range is 0x0000 to 0x3FFF for 16K Flash devices
//   2) FLADDR src - address of source bytes in Flash
//                    valid range is 0x0000 to 0x7DFF for 32K Flash devices
//                    valid range is 0x0000 to 0x3FFF for 16K Flash devices
//   3) unsigned int numbytes - the number of bytes to copy
//                              valid range is range of integer
//
// This routine copies <numbytes> from <src> to the linear FLASH address
// <dest>.  The destination bytes must be erased to 0xFF before writing.
// <src>/<dest> + <numbytes> must be less than 0x7DFF.
//
//-------------------------------------------------------------------------

void FLASH_Copy (FLADDR dest, FLADDR src, unsigned int numbytes)
{
   FLADDR i;

   for (i = 0; i < numbytes; i++) {
      FLASH_ByteWrite ((FLADDR) dest+i, FLASH_ByteRead((FLADDR) src+i));
   }
}

//-------------------------------------------------------------------------
// FLASH_Fill
//-------------------------------------------------------------------------
//
// Return Value : None
// Parameters   :
//   1) FLADDR addr - starting address of bytes to fill in Flash
//                    valid range is 0x0000 to 0x7DFF for 32K Flash devices
//                    valid range is 0x0000 to 0x3FFF for 16K Flash devices
//   2) ULONG length - number of bytes to fill
//                     range is total Flash size
//   3) UCHAR fill - the character used the Flash should be filled with
//
// This routine fills the FLASH beginning at <addr> with <length> bytes.
// The target bytes must be erased before writing to them.
// <addr> + <length> must be less than 0x7DFF.
//
//-------------------------------------------------------------------------

void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill)
{
   FLADDR i;

   for (i = 0; i < length; i++) {
      FLASH_ByteWrite (addr+i, fill);
   }
}

//-------------------------------------------------------------------------
// End Of File
//-------------------------------------------------------------------------
```

7.15.4. F410_FlashUtils.h

```
//-----------------------------------------------------------------------------
// F410_FlashUtils.h
//-----------------------------------------------------------------------------
// Copyright 2006 Silicon Laboratories, Inc.
// http://www.silabs.com
//
// Program Description:
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// FID:            41X000052
// Target:         C8051F410/1/2/3
// Tool chain:     Keil C51 8.00
// Command Line:   None
//
//
// Release 1.0
//    -Initial Revision (GP)
//    -29 SEP 2006
//

//-----------------------------------------------------------------------------
// Open Header #define
//-----------------------------------------------------------------------------

#ifndef _F410_FLASHUTILS_H_
#define _F410_FLASHUTILS_H_

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F410_FlashPrimitives.h"

//-----------------------------------------------------------------------------
// Exported Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
void FLASH_Write (FLADDR dest, char *src, unsigned int numbytes);
char * FLASH_Read (char *dest, FLADDR src, unsigned int numbytes);
void FLASH_Clear (FLADDR addr, unsigned int numbytes);

// FLASH update/copy routines
void FLASH_Update (FLADDR dest, char *src, unsigned int numbytes);
void FLASH_Copy (FLADDR dest, FLADDR src, unsigned int numbytes);

// FLASH test routines
void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill);

//-----------------------------------------------------------------------------
// Close Header #define
//-----------------------------------------------------------------------------

#endif    // _F410_FLASHUTILS_H_

//-----------------------------------------------------------------------------
// End Of File
//-----------------------------------------------------------------------------
```

7.16. ‘F520

7.16.1. F520_FlashPrimitives.c

```
//-----------------------------------------------------------------------------
// F520_FlashPrimitives.c
//-----------------------------------------------------------------------------
// Copyright 2006 Silicon Laboratories, Inc.
// http://www.silabs.com
//
// Program Description:
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// FID:            52X000001
// Target:         C8051F520/1/2/3/4/5/6/7; C8051F530/1/2/3/4/5/6/7
// Tool chain:     Keil C51 8.00
// Command Line:   None
//
//
// Release 1.0
//    -Initial Revision (GP)
//    -29 SEP 2006
//

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include <c8051F520.h>
#include "F520_FlashPrimitives.h"

//-----------------------------------------------------------------------------
// Function Prototypes
//-----------------------------------------------------------------------------

void          FLASH_ByteWrite (FLADDR addr, char byte);
unsigned char FLASH_ByteRead  (FLADDR addr);
void          FLASH_PageErase (FLADDR addr);

//-----------------------------------------------------------------------------
// FLASH_ByteWrite
//-----------------------------------------------------------------------------
//
// Return Value : None
// Parameters   :
//   1) FLADDR addr - address of the byte to write to
//                    valid range is 0x0000 to 0x1DFE for 8K Flash devices
//                    valid range is 0x0000 to 0x0FFE for 4K Flash devices
//                    valid range is 0x0000 to 0x07FE for 2K Flash devices
//   2) char byte - byte to write to Flash.
//
// This routine writes <byte> to the linear FLASH address <addr>.
//-----------------------------------------------------------------------------

void FLASH_ByteWrite (FLADDR addr, char byte)
{
   bit EA_SAVE = EA;                   // Preserve EA
   char xdata * data pwrite;           // FLASH write pointer

   EA = 0;                             // Disable interrupts

   VDM0CN = 0xA0;                      // Enable VDD monitor and high threshold
```

```
   RSTSRC = 0x02;                      // Enable VDD monitor as a reset source

   pwrite = (char xdata *) addr;

   FLKEY  = 0xA5;                      // Key Sequence 1
   FLKEY  = 0xF1;                      // Key Sequence 2
   PSCTL |= 0x01;                      // PSWE = 1 which enables writes

   VDM0CN = 0xA0;                      // Enable VDD monitor and high threshold

   RSTSRC = 0x02;                      // Enable VDD monitor as a reset source

   *pwrite = byte;                     // Write the byte

   PSCTL &= ~0x01;                     // PSWE = 0 which disable writes

   EA = EA_SAVE;                       // Restore interrupts
}

//-----------------------------------------------------------------------------
// FLASH_ByteRead
//-----------------------------------------------------------------------------
//
// Return Value :
//      unsigned char - byte read from Flash
// Parameters   :
//   1) FLADDR addr - address of the byte to read to
//                    valid range is 0x0000 to 0x1DFE for 8K Flash devices
//                    valid range is 0x0000 to 0x0FFE for 4K Flash devices
//                    valid range is 0x0000 to 0x07FE for 2K Flash devices
//
// This routine reads a <byte> from the linear FLASH address <addr>.
//-----------------------------------------------------------------------------

unsigned char FLASH_ByteRead (FLADDR addr)
{
   bit EA_SAVE = EA;                   // Preserve EA
   char code * data pread;             // FLASH read pointer
   unsigned char byte;

   EA = 0;                             // Disable interrupts

   pread = (char code *) addr;

   byte = *pread;                      // Read the byte

   EA = EA_SAVE;                       // Restore interrupts

   return byte;
}

//-----------------------------------------------------------------------------
// FLASH_PageErase
//-----------------------------------------------------------------------------
//
// Return Value : None
// Parameters   :
//   1) FLADDR addr - address of any byte in the page to erase
//                    valid range is 0x0000 to 0x1BFF for 8K Flash devices
//                    valid range is 0x0000 to 0x0DFF for 4K Flash devices
//                    valid range is 0x0000 to 0x05FF for 2K Flash devices
//
// This routine erases the FLASH page containing the linear FLASH address
// <addr>.  Note that the page of Flash containing the Lock Byte cannot be
```

```
// erased from application code.
//-----------------------------------------------------------------------------

void FLASH_PageErase (FLADDR addr)
{
   bit EA_SAVE = EA;                   // Preserve EA
   char xdata * data pwrite;           // FLASH write pointer

   EA = 0;                             // Disable interrupts

   VDM0CN = 0xA0;                      // Enable VDD monitor and high threshold

   RSTSRC = 0x02;                      // Enable VDD monitor as a reset source

   pwrite = (char xdata *) addr;

   FLKEY  = 0xA5;                      // Key Sequence 1
   FLKEY  = 0xF1;                      // Key Sequence 2
   PSCTL |= 0x03;                      // PSWE = 1; PSEE = 1


   VDM0CN = 0xA0;                      // Enable VDD monitor and high threshold

   RSTSRC = 0x02;                      // Enable VDD monitor as a reset source
   *pwrite = 0;                        // Initiate page erase

   PSCTL &= ~0x03;                     // PSWE = 0; PSEE = 0

   EA = EA_SAVE;                       // Restore interrupts
}

//-----------------------------------------------------------------------------
// End Of File
//-----------------------------------------------------------------------------
```

7.16.2. F520_FlashPrimitives.h

```
//-----------------------------------------------------------------------------
// F520_FlashPrimitives.h
//-----------------------------------------------------------------------------
// Copyright 2006 Silicon Laboratories, Inc.
// http://www.silabs.com
//
// Program Description:
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// FID:            52X000002
// Target:         C8051F520/1/2/3/4/5/6/7; C8051F530/1/2/3/4/5/6/7
// Tool chain:     Keil C51 8.00
// Command Line:   None
//
//
// Release 1.0
//    -Initial Revision (GP)
//    -29 SEP 2006
//

//-----------------------------------------------------------------------------
// Open Header #define
//-----------------------------------------------------------------------------

#ifndef _F520_FLASHPRIMITIVES_H_
#define _F520_FLASHPRIMITIVES_H_

//-----------------------------------------------------------------------------
// Structures, Unions, Enumerations, and Type Definitions
//-----------------------------------------------------------------------------

typedef unsigned long ULONG;
typedef unsigned int  UINT;
typedef unsigned char UCHAR;
typedef UINT FLADDR;

//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------

#ifndef FLASH_PAGESIZE
#define FLASH_PAGESIZE 512
#endif

#ifndef FLASH_TEMP
#define FLASH_TEMP 0x1A00L             // For 8K Flash devices
//#define FLASH_TEMP 0x0C00L           // For 4K Flash devices
//#define FLASH_TEMP 0x0400L           // For 2K Flash devices
#endif

#ifndef FLASH_LAST
#define FLASH_LAST 0x1C00L             // For 8K Flash devices
//#define FLASH_LAST 0x0E00L           // For 4K Flash devices
//#define FLASH_LAST 0x0600L           // For 2K Flash devices
#endif

//-----------------------------------------------------------------------------
// Exported Function Prototypes
//-----------------------------------------------------------------------------

void          FLASH_ByteWrite (FLADDR addr, char byte);
```

```
unsigned char FLASH_ByteRead  (FLADDR addr);
void          FLASH_PageErase (FLADDR addr);

//-----------------------------------------------------------------------------
// Close Header #define
//-----------------------------------------------------------------------------

#endif    // _F520_FLASHPRIMITIVES_H_

//-----------------------------------------------------------------------------
// End Of File
//-----------------------------------------------------------------------------
```

7.16.3. F520_FlashUtils.c

```
//-----------------------------------------------------------------------------
// F520_FlashUtils.c
//-----------------------------------------------------------------------------
// Copyright 2006 Silicon Laboratories, Inc.
// http://www.silabs.com
//
// Program Description:
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// FID:            52X000003
// Target:         C8051F520/1/2/3/4/5/6/7; C8051F530/1/2/3/4/5/6/7
// Tool chain:     Keil C51 8.00
// Command Line:   None
//
//
// Release 1.0
//    -Initial Revision (GP)
//    -29 SEP 2006
//

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F520_FlashPrimitives.h"
#include "F520_FlashUtils.h"

//-----------------------------------------------------------------------------
// Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
void FLASH_Write (FLADDR dest, char *src, unsigned int numbytes);
char * FLASH_Read (char *dest, FLADDR src, unsigned int numbytes);
void FLASH_Clear (FLADDR addr, unsigned int numbytes);

// FLASH update/copy routines
void FLASH_Update (FLADDR dest, char *src, unsigned int numbytes);
void FLASH_Copy (FLADDR dest, FLADDR src, unsigned int numbytes);

// FLASH test routines
void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill);

//-----------------------------------------------------------------------------
// FLASH_Clear
//-----------------------------------------------------------------------------
//
// Return Value : None
// Parameters   :
//   1) FLADDR addr - address of the byte to write to
//                    valid range is 0x0000 to 0x19FF for 8K Flash devices
//                    valid range is 0x0000 to 0x0BFF for 4K Flash devices
//                    valid range is 0x0000 to 0x03FF for 2K Flash devices
//   2) unsigned int numbytes - the number of bytes to clear to 0xFF
//                    valid range is 0 to FLASH_PAGESIZE
//
// This routine erases <numbytes> starting from the FLASH addressed by
// <dest> by performing a read-modify-write operation using <FLASH_TEMP> as
// a temporary holding area.
// <addr> + <numbytes> must be less than 0x19FF/0x0BFF/0x03FF.
//
//-----------------------------------------------------------------------------
```

```
void FLASH_Clear (FLADDR dest, unsigned int numbytes)
{
   FLADDR dest_1_page_start;           // First address in 1st page
                                       // containing <dest>
   FLADDR dest_1_page_end;             // Last address in 1st page
                                       // containing <dest>
   FLADDR dest_2_page_start;           // First address in 2nd page
                                       // containing <dest>
   FLADDR dest_2_page_end;             // Last address in 2nd page
                                       // containing <dest>
   unsigned numbytes_remainder;        // When crossing page boundary,
                                       // number of <src> bytes on 2nd page
   unsigned FLASH_pagesize;            // Size of FLASH page to update

   FLADDR  wptr;                       // Write address
   FLADDR  rptr;                       // Read address

   unsigned length;

   FLASH_pagesize = FLASH_PAGESIZE;

   dest_1_page_start = dest & ~(FLASH_pagesize - 1);
   dest_1_page_end   = dest_1_page_start + FLASH_pagesize - 1;
   dest_2_page_start = (dest + numbytes)  & ~(FLASH_pagesize - 1);
   dest_2_page_end   = dest_2_page_start + FLASH_pagesize - 1;

   if (dest_1_page_end == dest_2_page_end)
   {
      // 1. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP);

      // 2. Copy bytes from first byte of dest page to dest-1 to Scratch page

      wptr = FLASH_TEMP;
      rptr = dest_1_page_start;
      length = dest - dest_1_page_start;
      FLASH_Copy (wptr, rptr, length);

      // 3. Copy from (dest+numbytes) to dest_page_end to Scratch page

      wptr = FLASH_TEMP + dest - dest_1_page_start + numbytes;
      rptr = dest + numbytes;
      length = dest_1_page_end - dest - numbytes + 1;
      FLASH_Copy (wptr, rptr, length);

      // 4. Erase destination page
      FLASH_PageErase (dest_1_page_start);

      // 5. Copy Scratch page to destination page
      wptr = dest_1_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, rptr, length);

   }
   else                                // Value crosses page boundary
   {
      // 1. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP);

      // 2. Copy bytes from first byte of dest page to dest-1 to temp page

      wptr = FLASH_TEMP;
      rptr = dest_1_page_start;
```

```
      length = dest - dest_1_page_start;
      FLASH_Copy (wptr, rptr, length);

      // 3. Erase destination page 1
      FLASH_PageErase (dest_1_page_start);

      // 4. Copy Scratch page to destination page 1
      wptr = dest_1_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, rptr, length);

      // Now handle 2nd page

      // 5. Erase Scratch page
      FLASH_PageErase (FLASH_TEMP);

      // 6. Copy bytes from numbytes remaining to dest-2_page_end to temp page

      numbytes_remainder = numbytes - (dest_1_page_end - dest + 1);
      wptr = FLASH_TEMP + numbytes_remainder;
      rptr = dest_2_page_start + numbytes_remainder;
      length = FLASH_pagesize - numbytes_remainder;
      FLASH_Copy (wptr, rptr, length);

      // 7. Erase destination page 2
      FLASH_PageErase (dest_2_page_start);

      // 8. Copy Scratch page to destination page 2
      wptr = dest_2_page_start;
      rptr = FLASH_TEMP;
      length = FLASH_pagesize;
      FLASH_Copy (wptr, rptr, length);
   }
}

//-----------------------------------------------------------------------------
// FLASH_Update
//-----------------------------------------------------------------------------
//
// Return Value : None
// Parameters   :
//   1) FLADDR dest - starting address of the byte(s) to write to
//                    valid range is 0x0000 to 0x19FF for 8K Flash devices
//                    valid range is 0x0000 to 0x0BFF for 4K Flash devices
//                    valid range is 0x0000 to 0x03FF for 2K Flash devices
//   2) char *src - pointer to source bytes
//   3) unsigned int numbytes - the number of bytes to update
//                              valid range is 0 to FLASH_PAGESIZE
//
// This routine replaces <numbytes> from <src> to the FLASH addressed by
// <dest>.  This function calls FLASH_Clear() to initialize all <dest> bytes
// to 0xff's prior to copying the bytes from <src> to <dest>.
// <dest> + <numbytes> must be less than 0x19FF/0x0BFF/0x03FF.
//
//-----------------------------------------------------------------------------

void FLASH_Update (FLADDR dest, char *src, unsigned int numbytes)
{
   // 1. Erase <numbytes> starting from <dest>
   FLASH_Clear (dest, numbytes);

   // 2. Write <numbytes> from <src> to <dest>
   FLASH_Write (dest, src, numbytes);
```

```
}


//-----------------------------------------------------------------------------
// FLASH_Write
//-----------------------------------------------------------------------------
//
// Return Value : None
// Parameters   :
//   1) FLADDR dest - starting address of the byte(s) to write to
//                    valid range is 0x0000 to 0x1DFE for 8K Flash devices
//                    valid range is 0x0000 to 0x0FFE for 4K Flash devices
//                    valid range is 0x0000 to 0x07FE for 2K Flash devices
//   2) char *src - pointer to source bytes
//   3) unsigned int numbytes - the number of bytes to write
//                              valid range is is range of integer
//
// This routine copies <numbytes> from <src> to the linear FLASH address
// <dest>.  The bytes must be erased to 0xFF before writing.
// <dest> + <numbytes> must be less than 0x1DFE/0x0FFE/0x07FE.
//
//-----------------------------------------------------------------------------

void FLASH_Write (FLADDR dest, char *src, unsigned int numbytes)
{
   FLADDR i;

   for (i = dest; i < dest+numbytes; i++) {
      FLASH_ByteWrite (i, *src++);
   }
}

//-----------------------------------------------------------------------------
// FLASH_Read
//-----------------------------------------------------------------------------
//
// Return Value :
//   1) char *dest - pointer to destination bytes
// Parameters   :
//   1) char *dest - pointer to destination bytes
//   2) FLADDR src - address of source bytes in Flash
//                    valid range is 0x0000 to 0x1DFE for 8K Flash devices
//                    valid range is 0x0000 to 0x0FFE for 4K Flash devices
//                    valid range is 0x0000 to 0x07FE for 2K Flash devices
//   3) unsigned int numbytes - the number of bytes to read
//                              valid range is range of integer
//
// This routine copies <numbytes> from the linear FLASH address <src> to
// <dest>.
// <dest> + <numbytes> must be less than 0x1DFE/0x0FFE/0x07FE.
//
//-----------------------------------------------------------------------------

char * FLASH_Read (char *dest, FLADDR src, unsigned int numbytes)
{
   FLADDR i;

   for (i = 0; i < numbytes; i++) {
      *dest++ = FLASH_ByteRead (src+i);
   }
   return dest;
}

//-----------------------------------------------------------------------------
```

```
// FLASH_Copy
//-------------------------------------------------------------------------
//
// Return Value : None
// Parameters   :
//   1) FLADDR dest - pointer to destination bytes in Flash
//                    valid range is 0x0000 to 0x1DFE for 8K Flash devices
//                    valid range is 0x0000 to 0x0FFE for 4K Flash devices
//                    valid range is 0x0000 to 0x07FE for 2K Flash devices
//   2) FLADDR src - address of source bytes in Flash
//                    valid range is 0x0000 to 0x1DFE for 8K Flash devices
//                    valid range is 0x0000 to 0x0FFE for 4K Flash devices
//                    valid range is 0x0000 to 0x07FE for 2K Flash devices
//   3) unsigned int numbytes - the number of bytes to copy
//                              valid range is range of integer
//
// This routine copies <numbytes> from <src> to the linear FLASH address
// <dest>.  The destination bytes must be erased to 0xFF before writing.
// <dest>/<src> + <numbytes> must be less than 0x1DFE/0x0FFE/0x07FE.
//
//-------------------------------------------------------------------------

void FLASH_Copy (FLADDR dest, FLADDR src, unsigned int numbytes)
{
   FLADDR i;

   for (i = 0; i < numbytes; i++)
   {
      FLASH_ByteWrite ((FLADDR) dest+i, FLASH_ByteRead((FLADDR) src+i));
   }
}

//-------------------------------------------------------------------------
// FLASH_Fill
//-------------------------------------------------------------------------
//
// Return Value : None
// Parameters   :
//   1) FLADDR addr - starting address of bytes to fill in Flash
//                    valid range is 0x0000 to 0x1DFE for 8K Flash devices
//                    valid range is 0x0000 to 0x0FFE for 4K Flash devices
//                    valid range is 0x0000 to 0x07FE for 2K Flash devices
//   2) ULONG length - number of bytes to fill
//                     range is total Flash size
//   3) UCHAR fill - the character used the Flash should be filled with
//
// This routine fills the FLASH beginning at <addr> with <length> bytes.
// The target bytes must be erased before writing to them.
// <addr> + <length> must be less than 0x1DFE/0x0FFE/0x07FE.
//
//-------------------------------------------------------------------------

void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill)
{
   FLADDR i;

   for (i = 0; i < length; i++) {
      FLASH_ByteWrite (addr+i, fill);
   }
}

//-------------------------------------------------------------------------
// End Of File
//-------------------------------------------------------------------------
```

7.16.4. F520_FlashUtils.h

```
//-----------------------------------------------------------------------------
// F520_FlashUtils.h
//-----------------------------------------------------------------------------
// Copyright 2006 Silicon Laboratories, Inc.
// http://www.silabs.com
//
// Program Description:
//
// This program contains several useful utilities for writing and updating
// FLASH memory.
//
// FID:            52X000004
// Target:         C8051F520/1/2/3/4/5/6/7; C8051F530/1/2/3/4/5/6/7
// Tool chain:     Keil C51 8.00
// Command Line:   None
//
//
// Release 1.0
//    -Initial Revision (GP)
//    -29 SEP 2006
//

//-----------------------------------------------------------------------------
// Open Header #define
//-----------------------------------------------------------------------------

#ifndef _F520_FLASHUTILS_H_
#define _F520_FLASHUTILS_H_

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include "F520_FlashPrimitives.h"

//-----------------------------------------------------------------------------
// Exported Function Prototypes
//-----------------------------------------------------------------------------

// FLASH read/write/erase routines
void FLASH_Write (FLADDR dest, char *src, unsigned int numbytes);
char * FLASH_Read (char *dest, FLADDR src, unsigned int numbytes);
void FLASH_Clear (FLADDR addr, unsigned int numbytes);

// FLASH update/copy routines
void FLASH_Update (FLADDR dest, char *src, unsigned int numbytes);
void FLASH_Copy (FLADDR dest, FLADDR src, unsigned int numbytes);

// FLASH test routines
void FLASH_Fill (FLADDR addr, ULONG length, UCHAR fill);

//-----------------------------------------------------------------------------
// Close Header #define
//-----------------------------------------------------------------------------

#endif    // _F520_FLASHUTILS_H_

//-----------------------------------------------------------------------------
// End Of File
//-----------------------------------------------------------------------------
```

DOCUMENT CHANGE LIST

Revision 0.1 to Revision 0.2

- Updated list of relevant devices.
- Add all new content to "6. Flash Write and Erase Guidelines" on page 10.
- Added example code for 'F340 devices.
- Added example code for 'F326/7 devices.
- Fixed example code for 'F320 devices.

Revision 0.2 to Revision 0.3

- Added C8051F41x and C8051F52x-53x documentation
- Added clarification of the RSTSRC register in "4.4. Example Code Implementation Notes" on page 8.

Revision 0.3 to Revision 0.4

- Added C8051F36x documentation.

NOTES:

CONTACT INFORMATION

Silicon Laboratories Inc.

400 West Cesar Chavez
Austin, TX 78701
Tel: 1+(512) 416-8500
Fax: 1+(512) 416-9669
Toll Free: 1+(877) 444-3032

Email: MCUinfo@silabs.com
Internet: www.silabs.com