FLASH PROGRAMMING VIA THE C2 INTERFACE

Relevant Devices

This application note applies to the following devices: C8051F300, C8051F301, C8051F302, and C8051F303.

Introduction

This application note describes how to program the FLASH memory on C8051F30x devices through the C2 interface. Example software is included.

C8051F30x devices have a FLASH Programming interface (FPI) that is accessed via the C2 Interface (C2I) and a set of programming registers. The FLASH access block diagram is shown in Figure 1.

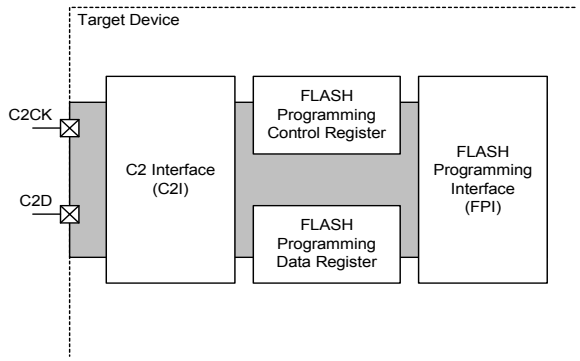

Figure 1. C2 FLASH Programming Architecture

The information needed to access the FLASH memory via the C2 Interface can be divided into the three sections listed below.

1. **C2 Interface:** This includes information about the C2 physical layer protocol, C2 registers, and C2 primitive commands.
   a. The 2-wire C2 interface (C2CK and C2D)
   b. C2 Address register and its access commands (Address Write, Address Read)
   c. C2 Data register access commands (Data Write, Data Read)
   d. Device ID and Revision ID registers
2. **FLASH Programming Registers:** This includes the functions of the FLASH Programming registers.
   a. FLASH Control register (FPCTL)
   b. FLASH Data register (FPDAT)
3. **FLASH Programming Interface:** This includes the FLASH Programming Interface (FPI) commands and procedures.
   a. FLASH Block Write
   b. FLASH Block Read
   c. FLASH Page Erase
   d. FLASH User Space Erase

C2 Interface

The C2 Interface (C2I) consists of an Address register and access to up to 256 Data registers. Access to these registers is provided by the C2 physical layer protocol. For details on the C2 protocol, see the C2 Specification on the Silicon Labs Applications website (http://www.silabs.com).

This application note assumes a basic understanding of the C2 physical layer protocol and the procedure for accessing the C2 Address and Data registers.

C2 Address Register

The C2 Address register (ADDRESS) serves two purposes in C2 FLASH programming:

1. ADDRESS selects which C2 Data register is accessed during C2 Data Read/Write frames

2. During Address Read frames, ADDRESS provides FPI status information.

Address Reads are used frequently during C2 FLASH programming as a handshaking scheme between the programmer and the FPI.

The Address Read command returns an 8-bit status code, formatted as shown in Table 1 .

Table 1. C2 Address Register Status Bits

Bit	Description
7-2	Unused
1	**InBusy:** This bit is set to '1' by the C2 Interface following a write to FPDAT. It is cleared to '0' when the FPI acknowledges the write to FPDAT.
0	**OutReady:** This bit is set to '1' by the FPI when output data is available in the FPDAT register.

InBusy should be polled following any write to FPDAT; OutReady should be polled before any reads of FPDAT.

DEVICEID: Device ID Register

The Device ID register (DEVICEID) is a read-only C2 Data register containing the 8-bit device identifier of the target C2 device. The C2 address for register DEVICEID is 0x01. The Device ID for all C8051F30x devices is 0x04.

REVID: Revision ID Register

The Revision ID register (REVID) is a read-only C2 Data register containing the 8-bit revision identifier of the target C2 device. The C2 address for register REVID is 0x01.

FLASH Programming Registers

Communication between the FPI and C2I is accomplished via two FLASH Programming registers: FPCTL and FPDAT.

Table 2. FLASH Programming Registers

Register	C2 Address
FPCTL	0x02
FPDAT	0xB4

FPCTL: FLASH Programming Control Register

The FLASH Programming Control register (FPCTL) serves to enable C2 FLASH programming.

To enable C2 FLASH programming, the following key codes must be written to FPCTL in the following sequence:

1. 0x02
2. 0x01

The key codes must be written in this sequence following a target device reset. Once the key codes have been written to FPCTL, the target device is halted until the next device reset.

FPDAT: FLASH Programming Data Register

The FLASH Programming Data register (FPDAT) is used to pass all data between the C2I and FPI. Information passed via FPDAT includes:

1. All FPI Commands (C2I-to-FPI)
2. FPI Status Information (FPI-to-C2I)
3. FLASH Addresses (C2I-to-FPI)
4. FLASH Data (both directions)

FLASH Programming Interface

The FLASH Programming Interface (FPI) performs a set of four programming commands. Each command is executed using a sequence of reads and writes of the FPDAT register. The four FPI commands are listed in Table 3 .

Table 3. C2 FLASH Programming Commands

Command	Code
FLASH Block Write	0x07
FLASH Block Read	0x06
FLASH Page Erase	0x08
Device Erase	0x03

Before performing any FPI commands, the FPI must be initialized with the following sequence:

1. Reset the target device.
2. Wait at least 2 µs.
3. Write the following key codes to FPCTL: 0x02, 0x01.
4. Wait at least 20 ms.

The FLASH Programming initialization sequence is shown in Figure 2.

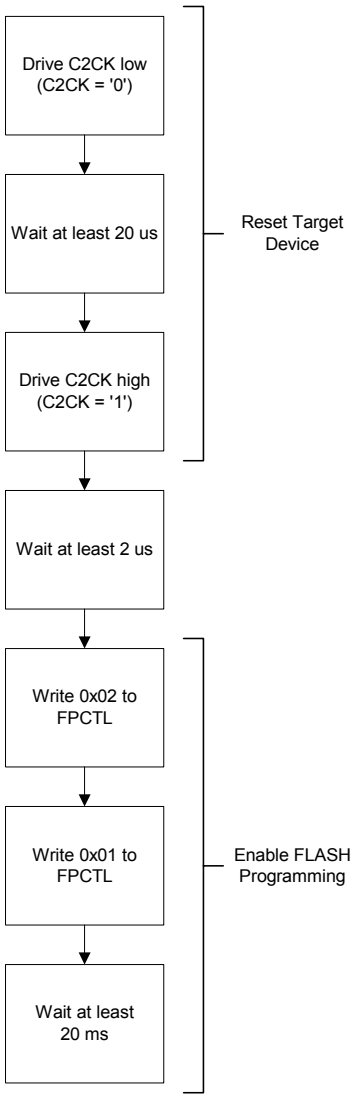

Figure 2. FPI Initialization Sequence

Writing a FLASH Block

All FLASH writes are performed with the FLASH Block Write command. The size of the FLASH block can be 1-to-256 bytes, and is user-defined during the Block Write sequence. The 8-bit Data Length Code is formatted as follows:

For (1 ≤ Length Code ≤ 255),

Block Size (in bytes) = Length Code

For (Length Code == 0),

Block Size (in bytes) = 256

The FLASH Block Write sequence is shown in Figure 3. This flow diagram assumes the FPI has been initialized by following the sequence in Figure 2.

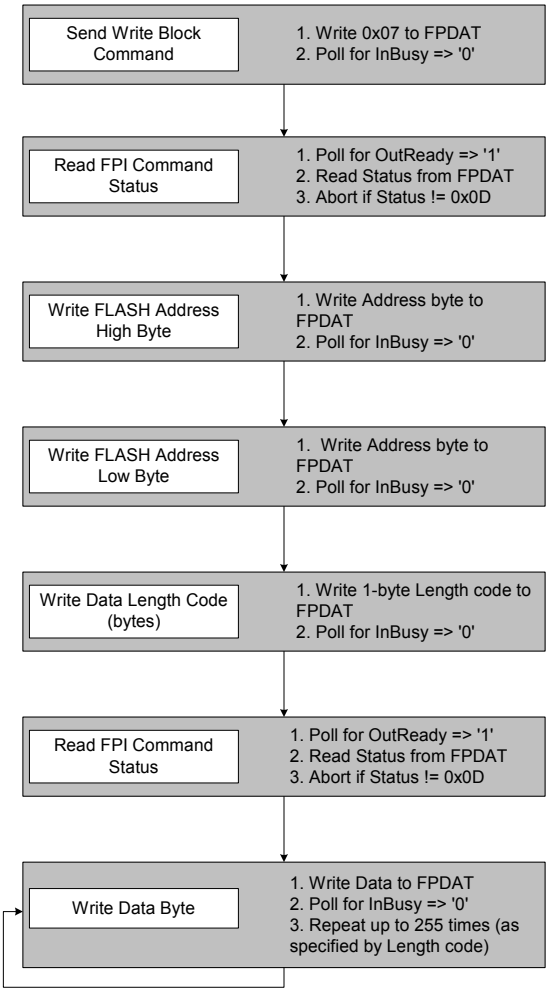

Figure 3. FLASH Block Write Sequence

Reading a FLASH Block

All FLASH reads are performed with the FLASH Block Read command. The size of the FLASH block can be 1-to-256 bytes, and is user-defined in the Block Read Sequence.

The 8-bit Data Length Code is formatted as follows:

For (1 ≤ Length Code ≤ 255),

Block Size (in bytes) = Length Code

For (Length Code == 0),

Block Size (in bytes) = 256

The FLASH Block Read sequence is shown in Figure 4. This flow diagram assumes the FPI has been initialized by following the sequence in Figure 2.

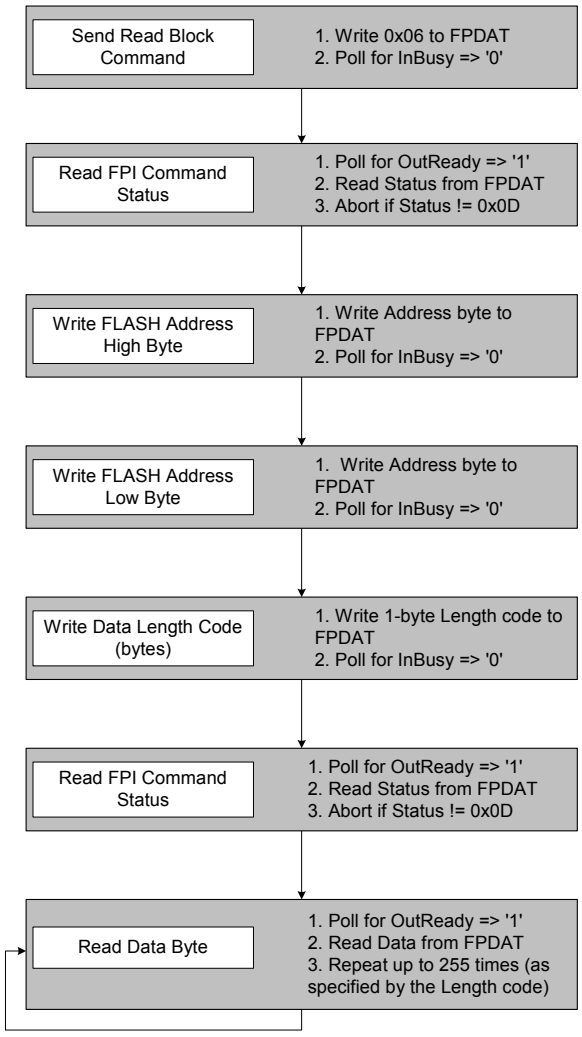

Figure 4. FLASH Block Read Sequence

Erasing a FLASH Page

FLASH memory is erased in 512-byte pages using the FLASH Page Erase command. The FLASH page to be erased is selected in the FLASH Page Erase procedure shown in Figure 5. Table 4 lists the target page numbers and addresses for C8051F30x devices.

Table 4. C8051F30x FLASH Memory Pages

Target Page	Address Range
0	0x0000 - 0x01FF
1	0x0200 - 0x03FF
2	0x0400 - 0x05FF
3	0x0600 - 0x07FF
4	0x0800 - 0x09FF
5	0x0A00 - 0x0BFF
6	0x0C00 - 0x0DFF
7	0x0E00 - 0x0FFF
8	0x1000 - 0x11FF
9	0x1200 - 0x13FF
10	0x1400 - 0x15FF
11	0x1600 - 0x17FF
12	0x1800 - 0x19FF
13	0x1A00 - 0x1BFF
14	0x1C00 - 0x1DFF
RESERVED	0x1E00 - 0x1FFF

This flow diagram in Figure 5 assumes the FPI has been initialized by following the sequence in Figure 2.

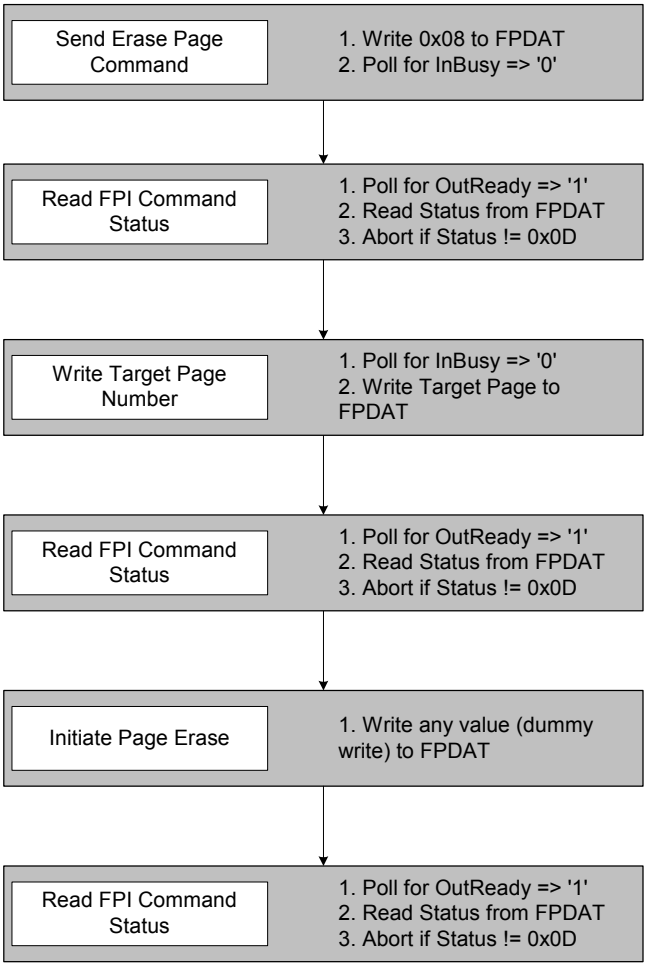

Figure 5. FLASH Page Erase Sequence

Erasing the Device

The FLASH Device Erase command performs a mass erase of the FLASH memory (addresses 0x0000 - 0x1DFF). A three-byte arming sequence must be written to the FPI to enable this procedure; the arming sequence is:

1. 0xDE
2. 0xAD
3. 0xA5

The erase operation is executed following the last byte of the arming sequence.

The Device Erase programming sequence is shown in Figure 6. This flow diagram assumes the FPI has been initialized by following the sequence in Figure 2.

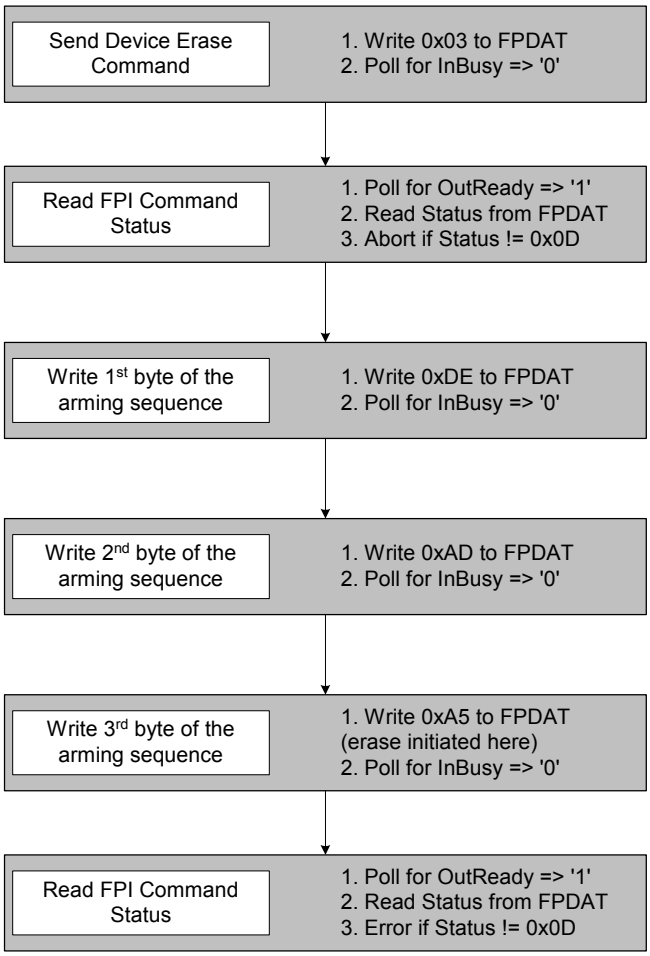

Figure 6. Device Erase Sequence

Software Example

The software example included with this application note is written for a C8051F00x or C8051F01x device acting as the programmer. The software can be ported to any other Silicon Labs C8051Fxxx device with little modification (timer and port initialization).

The interconnection diagram used with the example software is shown in Figure 7. This connection diagram assumes that the C2CK and/or C2D pins on the target device are not used by the target application.

If the C2 FLASH programming is to be performed on a target device installed in the user application, C2 isolation circuitry may be necessary. To determine if C2 isolation circuitry is necessary, see application note "AN024 - Pin Sharing Techniques for the C2 Interface" on the Silicon Labs website (http://www.silabs.com).

```
//-----------------------------------------------------------------------------
//
// CYGNAL INTEGRATED PRODUCTS, INC.
//
// FILE NAME: AN027.c
// TARGET DEVICE: C8051F00x
// CREATED ON: 12/01
// CREATED BY: JS
//
// Example code to program a C8051F30x device through its C2 interface.
//
// This software targets a C8051F00x device, though it could be easily
// ported to any Cygnal C8051Fxxx device.
//
// Code assumes the following Port pin connections:
//    C8051F0xx:  |  C8051F30x:
//    P1.0        ->  C2CK
//    P1.1        ->  C2D
//    VDD         ->  VDD
//    DGND        ->  GND
//
// Program includes:
//    - FLASH programming routines (Block Write, Block Read, Page Erase,
//       Device Erase)
//    - Primitive C2 communication routines (Address Write, Address Read,
```

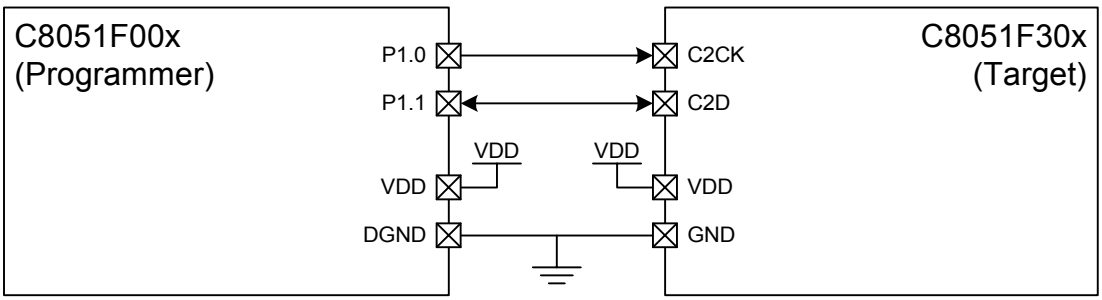

Figure 7. Interconnection Diagram for Software Example

```
//       Data Write, Data Read, Target Reset)
//    - Test software for the above routines
//
// Calling sequences for FLASH Programming Routines:
//
// C2 Initialization Procedure:
//    1) Call C2_Reset()
//    2) Call C2_Init()
//
// Note: All following routines assume the C2 Initialization Procedure
// has been executed prior to their calling
//
// Block Read Procedure:
//    1) Set <FLASH_ADDR> to the target FLASH starting address
//    2) Set <NUM_BYTES> to the number of bytes to read
//    3) Initialize the pointer <C2_PTR> to the location to store the
//       read data
//    4) Call C2_BlockRead() (will return a '1' if successful,
//       '0' otherwise)
//
// Block Write Procedure:
//    1) Set <FLASH_ADDR> to the target FLASH starting address
//    2) Set <NUM_BYTES> to the number of bytes to write
//    3) Initialize the pointer <C2_PTR> to the location of the data
//       to be written
//    4) Call C2_BlockWrite() (will return a '1' if successful,
//       '0' otherwise)
//
// Page Erase Procedure:
//    1) Set <FLASH_ADDR> to a location within the page to be erased
//    2) Call C2_PageErase()
//
// Device Erase Procedure:
//    1) Call C2_DeviceErase()
//
//-------------------------------------------------------------------------------
// Includes
//-------------------------------------------------------------------------------
#include <c8051f000.h>      // SFR declarations
#include <stdio.h>
#include <string.h>

//-------------------------------------------------------------------------------
// Global CONSTANTS
//-------------------------------------------------------------------------------

// FLASH information
#define  FLASH_SIZE        8192           // FLASH size in bytes
#define  NUM_PAGES         FLASH_SIZE/512 // Number of 512-byte FLASH pages


// C2 status return codes
#define  INVALID_COMMAND   0x00
#define  COMMAND_FAILED    0x02
#define  COMMAND_OK        0x0D

// C2 interface commands
#define  GET_VERSION       0x01
#define  BLOCK_READ        0x06
```

```
#define  BLOCK_WRITE       0x07
#define  PAGE_ERASE        0x08
#define  DEVICE_ERASE      0x03

// C2 Registers
#define  FPDAT             0xB4
#define  FPCTL             0x02
#define  DEVICEID          0x00
#define  REVID             0x01

// Program MACROS
#define  Poll_OutReady     while(!(C2_ReadAR()&0x01))
#define  Poll_InBusy       while((C2_ReadAR()&0x02))
#define  StrobeC2CK        C2CK = LOW; C2CK = HIGH

#define  C2D_DriverOn      PRT1CF |= 0x02               // Configures C2D pin as
                                                        // push-pull output

#define  C2D_DriverOff     PRT1CF &= ~(0x02);P1 |= 0x02  // Configures C2D pin as
                                                        // open-drain input
// Misc constants
#define  SYSCLK            16000000
#define  LOW               0
#define  HIGH              1

// C2 Signals
sbit  C2CK = P1 ^ 0;                  // Assign Port pin P1.0 as C2CK
sbit  C2D = P1 ^ 1;                   // Assign Port pin P1.1 as C2D

//-----------------------------------------------------------------------------
// Global VARIABLES
//-----------------------------------------------------------------------------

unsigned char NUM_BYTES;
unsigned int  FLASH_ADDR;
unsigned char idata *C2_PTR;
unsigned char idata R_BUF[128];

//-----------------------------------------------------------------------------
// Function PROTOTYPES
//-----------------------------------------------------------------------------

// FLASH programming functions
void C2_Init();
unsigned char C2_GetDevID(void);
char C2_BlockRead(void);
char C2_BlockWrite(void);
char C2_PageErase(void);
char C2_DeviceErase(void);

// Primitive C2 functions
void C2_Reset(void);
void C2_WriteAR(char);
unsigned char C2_ReadAR(void);
void C2_WriteDR(char);
unsigned char C2_ReadDR(void);

// Utility functions
void Timer3us(unsigned int);
```

```
void Port_Init(void);

//-----------------------------------------------------------------------------
// MAIN Routine
//-----------------------------------------------------------------------------

void main()
{
   unsigned int i, j;                  // Misc. counters
   unsigned char W_BUF[] = "Hook 'em!";

   WDTCN = 0xDE;                       // Disable Watchdog Timer
   WDTCN = 0xAD;

   OSCICN |= 0x03;                     // Configure internal oscillator for
                                       // highest setting (16MHz)
   Port_Init();                        // Configure P1.0 as C2CK and P1.1 as C2D

   // Read Device ID
   C2_Reset();                         // Reset target
   j = C2_GetDevID();                  // Device ID should be 0x04 for 'F30x devices

   if (j != 0x04)                      // Spin here if an 'F30x device is not found
      while (1);

   // Initiate C2 FLASH Programming
   C2_Reset();                         // Start with a target device reset
   C2_Init();                          // Enable FLASH programming via C2

   // Erase entire FLASH code space
   C2_DeviceErase();                   // Erase entire code space

   // Read back entire FLASH memory (should be all '1's)
   FLASH_ADDR = 0x0000;                // Set target addresss (0x0000)
   NUM_BYTES = 128;                    // Set number of bytes to be read
   for (i=0;i<60;i++)                  // Perform 60 128-byte block reads (15 512-byte
   {                                   // FLASH pages, excluding reserved area)
      C2_PTR = R_BUF;                  // Initialize C2 pointer to the read buffer
      C2_BlockRead();                  // Initiate FLASH read
      for (j=0;j<128;j++)              // Check read data
      {
         if (R_BUF[j] != 0xFF)
            while (1);                 // Spin here if any FLASH byte != 0xFF
      }
   }

   // Write FLASH block
   FLASH_ADDR = 0x0100;                // Set target FLASH address (0x0100)
   NUM_BYTES = strlen(W_BUF) + 1;      // Set number of bytes to be written (size of
                                       // string stored in W_BUF, +1 for null)
   C2_PTR = W_BUF;                     // Initialize C2 pointer to the target string
   C2_BlockWrite();                    // Initiate block write

   // Read back FLASH block
   FLASH_ADDR = 0x0100;                // Set target FLASH address
   NUM_BYTES = strlen(W_BUF) + 1;      // Set number of bytes to be read (size of
                                       // previously written string, +1 for null)
   C2_PTR = R_BUF;                     // Initialize C2 pointer to the read buffer
   C2_BlockRead();                     // Initiate FLASH read
```

```
   if (strcmp(R_BUF, W_BUF) != 0)      // Compare written string with read data
      while (1);                       // Spin here if strings do not match

   // Erase FLASH block
   FLASH_ADDR = 0x0100;                // Set target FLASH address (page containing
                                       // this address will be erased)
   C2_PageErase();                     // Perform Erase

   // Read back FLASH block
   FLASH_ADDR = 0x0100;                // Set target FLASH address (0x0100)
   NUM_BYTES = strlen(W_BUF) + 1;      // Set number of bytes to be read (size of
                                       // previously written string, +1 for null)
   C2_PTR = R_BUF;                     // Initialize C2 pointer to the read buffer
   C2_BlockRead();                     // Initiate FLASH read

   // Check read data (should be all 1's)
   for (i=0;i<NUM_BYTES;i++)
   {
      if (R_BUF[i] != 0xFF)            // If read data != 0xFF, an error occured
         while (1);                    // Spin here to indicate error
   }

   while(1);                           // Program successful
}

//-----------------------------------------------------------------------------
// FLASH Programming Routines (High Level)
//-----------------------------------------------------------------------------
//
// These high-level routines perform the FLASH Programming Interface (FPI)
// command sequences.

//-----------------------------------------------------------------------------
// C2_Init()
//-----------------------------------------------------------------------------
// - Initializes the C2 Interface for FLASH programming
//
void C2_Init()
{
   C2_Reset();                         // Reset the target device
   Timer3us(2);                        // Delay for at least 2us

   C2_WriteAR(FPCTL);                  // Target the C2 FLASH Programming
                                       // Control register (FPCTL) for C2 Data
                                       // register accesses

   C2_WriteDR(0x02);                   // Write the first key code to enable
                                       // C2 FLASH programming
   C2_WriteDR(0x01);                   // Write the second key code to enable
                                       // C2 FLASH programming

   Timer3us(20000);                    // Delay for at least 20ms to ensure the
                                       // target is ready for C2 FLASH programming
}

//-----------------------------------------------------------------------------
// C2_GetDevID()
//-----------------------------------------------------------------------------
```

```
// - Reads the target Devcie ID register and Revision ID register
//
unsigned char C2_GetDevID()
{
   C2_WriteAR(DEVICEID);               // Select DeviceID regsiter for C2 Data
                                       // register accesses
   return C2_ReadDR();                 // Read and return the DeviceID register
}

//-------------------------------------------------------------------------------------
// C2_BlockRead()
//-------------------------------------------------------------------------------------
// - Reads a block of FLASH memory starting at <FLASH_ADDR>
// - The size of the block is defined by <NUM_BYTES>
// - Stores the read data at the location targeted by the pointer <C2_PTR>
// - Assumes that FLASH accesses via C2 have been enabled prior to the function call
// - Function call returns a '1' if successful; returns a '0' if unsuccessful
//
char C2_BlockRead()
{
   unsigned char i;                    // Counter
   unsigned char status;               // FPI status information holder

   C2_WriteAR(FPDAT);                  // Select the FLASH Programming Data register
                                       // for C2 Data register accesses
   C2_WriteDR(BLOCK_READ);             // Send FLASH block read command
   Poll_InBusy;                        // Wait for input acknowledge

   // Check status before starting FLASH access sequence
   Poll_OutReady;                      // Wait for status information
   status = C2_ReadDR();               // Read FLASH programming interface status
   if (status != COMMAND_OK)
      return 0;                        // Exit and indicate error

   C2_WriteDR(FLASH_ADDR >> 8);        // Send address high byte to FPDAT
   Poll_InBusy;                        // Wait for input acknowledge
   C2_WriteDR(FLASH_ADDR & 0x00FF);    // Send address low byte to FPDAT
   Poll_InBusy;                        // Wait for input acknowledge
   C2_WriteDR(NUM_BYTES);              // Send block size
   Poll_InBusy;                        // Wait for input acknowledge

   // Check status before reading FLASH block
   Poll_OutReady;                      // Wait for status information
   status = C2_ReadDR();               // Read FLASH programming interface status
   if (status != COMMAND_OK)
      return 0;                        // Exit and indicate error

   // Read FLASH block
   for (i=0;i<NUM_BYTES;i++)
   {
      Poll_OutReady;                   // Wait for data ready indicator
      *C2_PTR++ = C2_ReadDR();         // Read data from the FPDAT register
   }
   return 1;                           // Exit and indicate success
}

//-------------------------------------------------------------------------------------
// C2_BlockWrite()
//-------------------------------------------------------------------------------------
```

```
// - Writes a block of FLASH memory starting at <FLASH_ADDR>
// - The size of the block is defined by <NUM_BYTES>
// - Writes the block stored at the location targetted by <C2_PTR>
// - Assumes that FLASH accesses via C2 have been enabled prior to the function call
// - Function call returns a '1' if successful; returns a '0' if unsuccessful
//
char C2_BlockWrite()
{
   unsigned char i;                    // Counter
   unsigned char status;               // FPI status information holder

   C2_WriteAR(FPDAT);                  // Select the FLASH Programming Data register
                                       // for C2 Data register accesses
   C2_WriteDR(BLOCK_WRITE);            // Send FLASH block write command
   Poll_InBusy;                        // Wait for input acknowledge

   // Check status before starting FLASH access sequence
   Poll_OutReady;                      // Wait for status information
   status = C2_ReadDR();               // Read FLASH programming interface status
   if (status != COMMAND_OK)
      return 0;                        // Exit and indicate error

   C2_WriteDR(FLASH_ADDR >> 8);        // Send address high byte to FPDAT
   Poll_InBusy;                        // Wait for input acknowledge
   C2_WriteDR(FLASH_ADDR & 0x00FF);    // Send address low byte to FPDAT
   Poll_InBusy;                        // Wait for input acknowledge
   C2_WriteDR(NUM_BYTES);              // Send block size
   Poll_InBusy;                        // Wait for input acknolwedge

   // Check status before writing FLASH block
   Poll_OutReady;                      // Wait for status information
   status = C2_ReadDR();               // Read FLASH programming interface status
   if (status != COMMAND_OK)
      return 0;                        // Exit and indicate error

   // Write FLASH block
   for (i=0;i<NUM_BYTES;i++)
   {
      C2_WriteDR(*C2_PTR++);           // Write data to the FPDAT register
      Poll_InBusy;                     // Wait for input acknowledge
   }

   Poll_OutReady;                      // Wait for last FLASH write to complete
   return 1;                           // Exit and indicate success
}

//-------------------------------------------------------------------------------------
// C2_PageErase()
//-------------------------------------------------------------------------------------
// - Erases a 512-byte FLASH page
// - Targets the FLASH page containing the address <FLASH_ADDR>
// - Assumes that FLASH accesses via C2 have been enabled prior to the function call
// - Function call returns a '1' if successful; returns a '0' if unsuccessful
//
char C2_PageErase()
{
   unsigned char page;                 // Target FLASH page
   unsigned char status;               // FPI status information holder
```

```
   page = (unsigned char)(FLASH_ADDR >> 9);
                                       // <page> is the 512-byte sector containing
                                       // the target <FLASH_ADDR>.

   if (page >= NUM_PAGES - 1)          // Check that target page is within range
                                       // (NUM_PAGES minus 1 for reserved area)
      return 0;                        // Indicate error if out of range
   C2_WriteAR(FPDAT);                  // Select the FLASH Programming Data register
                                       // for C2 Data register accesses
   C2_WriteDR(PAGE_ERASE);             // Send FLASH page erase command
   Poll_InBusy;                        // Wait for input acknowledge


   // Check status before starting FLASH access sequence
   Poll_OutReady;                      // Wait for status information
   status = C2_ReadDR();               // Read FLASH programming interface status
   if (status != COMMAND_OK)
      return 0;                        // Exit and indicate error

   C2_WriteDR(page);                   // Send FLASH page number
   Poll_InBusy;                        // Wait for input acknowledge

   Poll_OutReady;                      // Wait for ready indicator
   status = C2_ReadDR();               // Read FLASH programming interface status
   if (status != COMMAND_OK)
      return 0;                        // Exit and indicate error

   C2_WriteDR(0x00);                   // Dummy write to initiate erase
   Poll_InBusy;                        // Wait for input acknowledge

   Poll_OutReady;                      // Wait for erase operation to complete
   return 1;                           // Exit and indicate success
}

//-----------------------------------------------------------------------------
// C2_Device_Erase()
//-----------------------------------------------------------------------------
// - Erases the entire FLASH memory space
// - Assumes that FLASH accesses via C2 have been enabled prior to the function call
// - Function call returns a '1' if successful; returns a '0' if unsuccessful
//
char C2_DeviceErase()
{
   unsigned char status;               // FPI status information holder

   C2_WriteAR(FPDAT);                  // Select the FLASH Programming Data register
                                       // for C2 Data register accesses
   C2_WriteDR(DEVICE_ERASE);           // Send Device Erase command
   Poll_InBusy;                        // Wait for input acknowledge

   // Check status before starting FLASH access sequence
   Poll_OutReady;                      // Wait for status information
   status = C2_ReadDR();               // Read FLASH programming interface status
   if (status != COMMAND_OK)
      return 0;                        // Exit and indicate error

   // Send a three-byte arming sequence to enable the device erase. If the sequence
   // is not received correctly, the command will be ignored.
   // Sequence: 0xDE, 0xAD, 0xA5.
```

```
   C2_WriteDR(0xDE);                    // Arming sequence command 1
   Poll_InBusy;                         // Wait for input acknowledge

   C2_WriteDR(0xAD);                    // Arming sequence command 2
   Poll_InBusy;                         // Wait for input acknowledge

   C2_WriteDR(0xA5);                    // Arming sequence command 3
   Poll_InBusy;                         // Wait for input acknowledge

   Poll_OutReady;                       // Wait for erase operation to complete
   return 1;                            // Exit and indicate success
}

//-----------------------------------------------------------------------------
// Primitive C2 Command Routines
//-----------------------------------------------------------------------------
//
// These routines perform the low-level C2 commands:
// 1. Address Read
// 2. Address Write
// 3. Data Read
// 4. Data Write
// 5. Device Reset

//-----------------------------------------------------------------------------
// C2_ReadAR()
//-----------------------------------------------------------------------------
// - Performs a C2 Address register read
// - Returns the 8-bit register content
//
unsigned char C2_ReadAR()
{
   unsigned char i;                     // Bit counter
   unsigned char addr;                  // Address register read content

   // START field
   StrobeC2CK;                          // Strobe C2CK with C2D driver disabled

   // INS field (10b, LSB first)
   C2D = LOW;
   C2D_DriverOn;                        // Enable C2D driver (output)
   StrobeC2CK;
   C2D = HIGH;
   StrobeC2CK;

   C2D_DriverOff;                       // Disable C2D driver (input)

   // ADDRESS field
   addr = 0;
   for (i=0;i<8;i++)                    // Shift in 8 bit ADDRESS field
   {                                    // LSB-first
      addr >>= 1;
      StrobeC2CK;
      if (C2D)
         addr |= 0x80;
   }

   // STOP field
   StrobeC2CK;                          // Strobe C2CK with C2D driver disabled
```

```
   return addr;                         // Return Address register read value
}

//-----------------------------------------------------------------------------
// C2_WriteAR()
//-----------------------------------------------------------------------------
// - Performs a C2 Address register write (writes the <addr> input
//   to Address register)
//
void C2_WriteAR(unsigned char addr)
{
   unsigned char i;                     // Bit counter

   // START field
   StrobeC2CK;                          // Strobe C2CK with C2D driver disabled

   // INS field (11b, LSB first)
   C2D = HIGH;
   C2D_DriverOn;                        // Enable C2D driver (output)
   StrobeC2CK;
   C2D = HIGH;
   StrobeC2CK;

   // ADDRESS field
   for(i=0;i<8;i++)                     // Shift out 8-bit ADDRESS field
   {
      C2D = (addr & 0x01);
      StrobeC2CK;
      addr >>= 1;
   }

   // STOP field
   C2D_DriverOff;                       // Disable C2D driver
   StrobeC2CK;                          // Strobe C2CK with C2D driver disabled

   return;
}

//-----------------------------------------------------------------------------
// C2_ReadDR()
//-----------------------------------------------------------------------------
// - Performs a C2 Data register read
// - Returns the 8-bit register content
//
unsigned char C2_ReadDR()
{
   unsigned char i;                     // Bit counter
   unsigned char dat;                   // Data register read content

   // START field
   StrobeC2CK;                          // Strobe C2CK with C2D driver disabled

   // INS field (00b, LSB first)
   C2D = LOW;
   C2D_DriverOn;                        // Enable C2D driver (output)
   StrobeC2CK;
   C2D = LOW;
   StrobeC2CK;
```

```
   // LENGTH field (00b -> 1 byte)
   C2D = LOW;
   StrobeC2CK;
   C2D = LOW;
   StrobeC2CK;

   // WAIT field
   C2D_DriverOff;                      // Disable C2D driver for input
   do
   {
      StrobeC2CK;
   }
   while (!C2D);                       // Strobe C2CK until target transmits a '1'

   // DATA field
   dat = 0;
   for (i=0;i<8;i++)                   // Shift in 8-bit DATA field
   {                                   // LSB-first
      dat >>= 1;
      StrobeC2CK;
      if (C2D)
         dat  |= 0x80;
   }

   // STOP field
   StrobeC2CK;                         // Strobe C2CK with C2D driver disabled

   return dat;
}

//-----------------------------------------------------------------------------
// C2_WriteDR()
//-----------------------------------------------------------------------------
// - Performs a C2 Data register write (writes <dat> input to data register)
//
void C2_WriteDR(unsigned char dat)
{
   unsigned char i;                    // Bit counter

   // START field
   StrobeC2CK;                         // Strobe C2CK with C2D driver disabled

   // INS field (01b, LSB first)
   C2D = HIGH;
   C2D_DriverOn;                       // Enable C2D driver
   StrobeC2CK;
   C2D = LOW;
   StrobeC2CK;

   // LENGTH field (00b -> 1 byte)
   C2D = LOW;
   StrobeC2CK;
   C2D = LOW;
   StrobeC2CK;

   // DATA field
   for (i=0;i<8;i++)                   // Shift out 8-bit DATA field
   {                                   // LSB-first
```

```
      C2D = (dat & 0x01);
      StrobeC2CK;
      dat >>= 1;
   }

   // WAIT field
   C2D_DriverOff;                      // Disable C2D driver for input
   do
   {
      StrobeC2CK;                      // Strobe C2CK until target transmits a '1'
   }
   while (!C2D);

   // STOP field
   StrobeC2CK;                         // Strobe C2CK with C2D driver disabled

   return;
}

//-----------------------------------------------------------------------------
// C2_Reset()
//-----------------------------------------------------------------------------
// - Performs a target device reset by pulling the C2CK pin low for >20us
//
void C2_Reset()
{
   C2CK = LOW;                         // Put target device in reset state by pulling
   Timer3us(20);                       // C2CK low for >20us

   C2CK = HIGH;                        // Release target device from reset
}

//-----------------------------------------------------------------------------
// Initialization / Utility Routines
//-----------------------------------------------------------------------------
//
//-----------------------------------------------------------------------------
// Timer3us
//-----------------------------------------------------------------------------
// - Uses Timer3 to delay <us> micro-seconds.
//
void Timer3us(unsigned int us)
{
   unsigned int i;                     // usec counter

   TMR3CN = 0 ;                        // Stop Timer3 and clear interrupt-pending flag
   TMR3CN |= 0x02;                     // Select SYSCLK as timebase

   TMR3RL = (-SYSCLK/1000000);         // Set Timer3 to overflow in 1us
   TMR3 = TMR3RL;                      // Initialize Timer3 for first overflow

   for (i = 0; i < us; i++)            // Count <us> overflows
   {
      TMR3CN |= 0x04;                  // START Timer3
      while (!(TMR3CN & 0x80));        // Wait for overflow
      TMR3CN &= ~0x04;                 // STOP Timer3
      TMR3CN &= ~(0x80);               // Clear overflow indicator
   }
}
```

```
//-----------------------------------------------------------------------------
// Port_Init()
//-----------------------------------------------------------------------------
// - Configures Port I/O for C2 functions
//
void Port_Init()
{
   XBR0 = 0x00;                        // No peripherals routed through Crossbar
   XBR1 = 0x00;                        // Crossbar
   XBR2 = 0x40;                        // Enable Crossbar (for GPIO)

   PRT1CF = 0x01;                      // Initialize C2CK to push-pull
                                       // and C2D to open-drain (driver disabled)
}
```

Notes:

Contact Information

Silicon Laboratories Inc.
4635 Boston Lane
Austin, TX 78735
Tel: 1+(512) 416-8500
Fax: 1+(512) 416-9669
Toll Free: 1+(877) 444-3032

Email: productinfo@silabs.com
Internet: www.silabs.com