CYGNAL Application Note

AN009 - Writing to FLASH from Application Code

Relevant Devices

This application note applies to the following devices:
C8051F000, C8051F001, C8051F002, C8051F005, C8051F006, C8051F007, C8051F010, C8051F011, and C8051F012.

Introduction

The purpose of this note is to illustrate how to program FLASH memory from application code. In-application programming allows the FLASH to be used for non-volatile information storage of configuration parameters, for example, and enables in-system boot-loading from an alternate interface than the JTAG port (such as the UART). Example code that shows how to read and write FLASH from 'C' code is included at the end of this note.

Key Points

- In-application programming allows the FLASH to be used for non-volatile information storage, similar to an EEPROM.
- The FLASH write/erase endurance is guaranteed for 10,000 cycles over the industrial temperature range of -40 to +85 degrees C.
- FLASH memory is read using the MOVC instruction (MOVC a, @A+DPTR).
- FLASH memory is written using the MOVX instruction (MOVX @DPTR, a).
- FLASH pages are aligned on 512-byte boundaries (0200h, 0400h, etc.).
- FLASH write operations can only write zeros. Therefore, before writing a FLASH byte, the page containing that byte must be erased (which sets all the bits in the page to ones).
- Before FLASH writes or erases can be performed, the FLASCL bits in FLSCL must be set based on the current system clock.
- Before a FLASH write can occur, the PSWE bit must be set to '1'.
- Before a FLASH erase can occur, both PSWE and PSEE must be set to '1's.
- A FLASH page erase operation is performed by setting PSEE and PSWE and writing to ANY byte within the 512-byte page to be erased.
- The FLASH page which contains the Read Lock Byte and the Write/Erase Lock Byte cannot be erased from user software (however this page can be written to from user software). It can only be erased through the JTAG interface.

Procedures

The FLASH is organized as a series of 512-byte pages. Additionally, there is an 128-byte page located at the top of FLASH at address 0x8000. Because a write to a FLASH byte can only write zeros, before writing an arbitrary value to a byte, the byte must be initialized to 0xFF. A FLASH erase operation initializes each byte in a page to 0xFF.

Erasing a FLASH Page

A FLASH page can be erased by the following procedure:

1. Set FLSCL based on the current system clock value as specified in the C8051F000 data sheet (FLSCL = 0x86 when using the default 2 MHz internal oscillator).
2. Set PSWE and PSEE to '1's by writing PSCTL = 0x03.
3. Write to any value to any byte within the page to be erased:

```
; initialize DPTR to address
; within page to be erased
MOV DPTR, #address

; initiate the erase operation
MOVX @DPTR, a
```

4. Set PSEE to '0' if no further erases are necessary.

An erase operation takes between 10 and 20 ms per page. Note: the CPU core is stalled during the FLASH erase process; however, the peripherals (like the ADC, UART, SMBus, and timers) remain active. Any interrupts generated during the erase process are held pending completion of the cycle. The same behavior occurs during a FLASH byte write.

Writing a FLASH Byte

The following procedure is used to write a byte to FLASH:

1. Set FLSCL based on the current system clock value as specified in the C8051F000 data sheet (FLSCL = 0x86 when using the default 2 MHz internal oscillator).
2. Set PSWE to '1' by writing PSCTL = 0x01 (PSEE must be '0').
3. Write the value to the byte:

```
; initialize DPTR to address
; within page to be erased
MOV DPTR, #address

; load byte to be written into
; acc
MOV a, #value

; initiate the write
MOVX @DPTR, a
```

4. Set PSWE to '0' if no further erases are necessary.

A FLASH write takes between 20 and 40 µs per byte.

Updating FLASH Values

When keeping an array of configuration information, it is often useful to update a single byte within a set of bytes. The generic procedure for updating a single byte (or subset of bytes) is as follows:

1. Copy the FLASH page to a temporary storage location (RAM or an erased 'scratch' FLASH page).
2. Erase the data FLASH page.
3. Copy the updated contents from the temporary storage location back to the data FLASH page.

Accessing FLASH from 'C' Code

The attached example code illustrates how to erase, write, and read FLASH contents in a 'C' program. Because FLASH reads are accomplished using the MOVC instruction, pointers into FLASH that are used for reads must be of type CODE. Because FLASH writes are accomplished by using the MOVX instruction, pointers into FLASH that are used for writes (or erases) must be of type XDATA.

Software Example

```
//-----------------------------------------------------------------------------
// FLASH_1.c
//-----------------------------------------------------------------------------
// Cygnal Integrated Products, Inc.
// AUTH: BW
//
// This program shows an example of how to erase, write, and read FLASH memory
// using a 'C' program.
//
// Target: C8051F0xx
// Tool chain: KEIL C51
//

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

#include <c8051f000.h>                          // SFR declarations
#include <stdio.h>

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

void main (void);

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

void main (void) {

   unsigned char test_string[] = "Howdy!";     // string to copy into
                                               // FLASH

   unsigned char xdata *pwrite;                // pointer in CODE space
                                               // (FLASH) to write to
   unsigned char code *pread;                  // pointer in CODE space
                                               // (FLASH) to read from
   unsigned char *pgen;                        // generic pointer
   char test;                                  // test character

   // disable watchdog timer
   WDTCN = 0xde;
   WDTCN = 0xad;

   // erase the 128-byte information page (at 0x8000)

   FLSCL = 0x86;                               // set FLASH scale register
                                               // for 2MHz system clock

   PSCTL = 0x03;                               // set PSWE and PSEE to
                                               // '1's (MOVX instructions
                                               // erase the corresponding
                                               // FLASH page
   pwrite = 0x8000;                          // set pointer to location
                                             // in page to erase

   *pwrite = 0;                              // initiate the erase
                                             // cycle.

   PSCTL = 0x01;                             // clear PSEE, leaving PSWE
                                             // set to '1'.
                                             // MOVX instructions will
                                             // write to the corresponding
                                             // FLASH address.

   // now we'll copy the contents of the string <test_string> to the
   // FLASH page, starting at address 0x8000.

   pgen = test_string;                       // pgen points to the source
                                             // string
   do {
      *pwrite++ = *pgen++;                   // copy the bytes
   } while (*pgen != '\0');                  // until the NULL terminator

   *pwrite = NULL;                           // NULL-terminate the
                                             // destination string

   PSCTL = 0;                                // disable FLASH writes
   FLSCL = 0x8f;                             // disable FLASH writes

   // now we'll read the string:

   pread = 0x8000;                           // initialize CODE read pointer
                                             // to beginning of string

   test = 0x5a;
   do {
      test = *pread++;
   } while (test != '\0');

   while (1) {                               // spin forever
   }
}
```