



Writing to FLASH from Application Code on 'F30x Devices

Relevant Devices

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

Introduction

The FLASH memory on C8051F300 family devices is 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.

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 operations on Cygnal devices are accomplished by using the MOVX instruction. The default target for MOVX write operations is external memory (XRAM); however, when the SFR bit PSWE (PSCTL.1) is set to a logic 1, MOVX write operations target FLASH memory instead. MOVX instructions are generated in 'C' by using pointers of memory type 'xdata'.

- In 'C', the pointer used to de-reference FLASH writes should NOT be located in *xdata* space (*data* and *idata* are appropriate).
- Be cautious when using the 'Large' and 'Compact' memory models, which target *xdata* space for user variables.
- 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.
- A Lock and Key sequence must be executed before *each* FLASH write or erase operation.
- Attempts to read, write, or erase code memory locations located in RESERVED space will generate a device reset.
- The CPU is stalled during FLASH write and erase operations, although peripherals (UART, ADC, timers, etc.) remain active.

Key Points

- Because FLASH write operations use the MOVX instruction, care must be taken to ensure that all MOVX write operations that occur when PSWE = '1' are strictly controlled.



FLASH Organization

The FLASH memory is organized into a set of 512-byte pages, as shown in Figure 1. 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. 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 time they are serviced in priority order.

MOVX and PSCTL

Writes to FLASH memory are supported by the MOVX write instruction. When the PSWE bit (PSCTL.1) is set to a logic 1, MOVX writes target FLASH memory instead of External Memory (XRAM). When both PSWE and PSEE (PSCTL.1 and PSCTL.2) are set to logic 1, MOVX writes erase the FLASH page containing the target address.

Lock and Key: FLKEY

To protect code space from an inadvertent FLASH write operation, a lock and key mechanism is supported. Before each FLASH write or erase operation, the “FLKEY” SFR register must be loaded with the following byte sequence: 0xA5 followed by 0xF1. FLASH read operations, which use the MOVX instruction, do not require a key sequence.

The MOVX write following the write of 0xF1 to FLKEY will initiate the FLASH write or erase operation. If the FLKEY sequence is not properly executed prior to the MOVX write, the write or erase operation will have no effect.

FLASH write or erase operations that target the RESERVED address space will automatically fail, regardless of the lock and key sequence, and will generate a device RESET. Following this reset, the FERROR bit (RSTSRC.6) will be set to a logic 1. Writes to the FLASH page containing the FLASH Lock Byte are permitted from user code. Erases to that FLASH page are prohibited and can only be accomplished through the C2 interface.

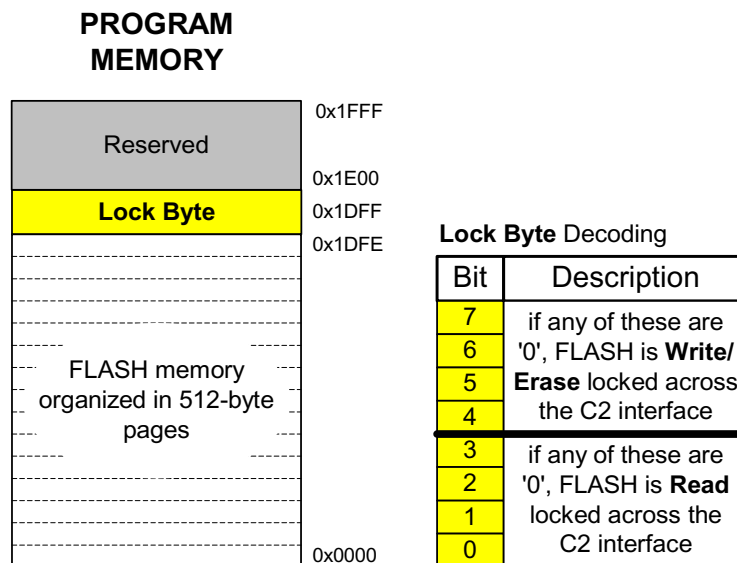


Figure 1. FLASH Memory Organization and Security



Software Example

```
//-----  
// FLASH_1.c  
//-----  
// Copyright 2002 Cygnal Integrated Products, Inc.  
//  
// AUTH: BW  
// DATE: 09 JAN 02  
//  
// This program shows an example of how to read, erase, and write FLASH  
// memory on an 'F30x device from application code.  
//  
// Target: C8051F30x  
// Tool chain: KEIL C51 6.03 / KEIL EVAL C51  
//  
  
//-----  
// Includes  
//-----  
  
#include <c8051f300.h>           // SFR declarations  
  
//-----  
// 16-bit SFR Definitions for 'F30x  
//-----  
  
sfr16 DP      = 0x82;           // data pointer  
sfr16 TMR2RL  = 0xca;           // Timer2 reload value  
sfr16 TMR2    = 0xcc;           // Timer2 counter  
sfr16 PCA0CP1 = 0xe9;           // PCA0 Module 1 Capture/Compare  
sfr16 PCA0CP2 = 0xeb;           // PCA0 Module 2 Capture/Compare  
sfr16 PCA0    = 0xf9;           // PCA0 counter  
sfr16 PCA0CP0 = 0xfb;           // PCA0 Module 0 Capture/Compare  
  
//-----  
// Global CONSTANTS  
//-----  
  
//-----  
// Function PROTOTYPES  
//-----  
  
//-----  
// Global VARIABLES  
//-----  
  
//-----  
// MAIN Routine  
//-----  
  
void main (void) {  
    unsigned char xdata *pwrite;           // pointer to FLASH used for writes  
                                           // NOTE: this pointer must be located  
                                           // in <data> or <idata> space!  
    unsigned char code *pread;            // pointer to FLASH used for reads  
    // test string
```

AN029 - Writing to FLASH from Application Code on 'F30x Devices



```
unsigned char code test_string[] = "Howdy!";

// Disable Watchdog timer
PCA0MD &= ~0x40;           // WDTE = 0 (clear watchdog timer
                           // enable)

// erase the FLASH page at 0x1000
EA = 0;                    // disable interrupts (precautionary)

// initialize write/erase pointer
pwrite = (unsigned char xdata *) 0x1000;
PSCTL = 0x03;              // MOVX writes erase FLASH page

FLKEY = 0xA5;              // FLASH lock and key sequence 1
FLKEY = 0xF1;              // FLASH lock and key sequence 2
*pwrite = 0;               // initiate page erase

PSCTL = 0;                 // MOVX writes target XRAM
EA = 1;                    // re-enable interrupts

// copy a string to FLASH memory at address 0x1000
// initialize FLASH read pointer
pread = (unsigned char code *) test_string;

EA = 0;                    // disable interrupts (precautionary)
pwrite = 0x1000;           // initialize FLASH write pointer
PSCTL = 0x01;              // MOVX writes target FLASH memory

while (*pread != '\0') {   // copy until NULL is detected

    FLKEY = 0xA5;          // FLASH lock and key sequence 1
    FLKEY = 0xF1;          // FLASH lock and key sequence 2
    *pwrite = *pread;      // copy byte

    pread++;               // advance pointers
    pwrite++;
}

FLKEY = 0xA5;              // FLASH lock and key sequence 1
FLKEY = 0xF1;              // FLASH lock and key sequence 2
*pwrite = '\0';            // NULL-terminate string

PSCTL = 0x00;              // MOVX writes target XRAM
EA = 1;                    // re-enable interrupts

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