Freescale Semiconductor
Application Note

Document Number: AN3787
Rev. 0, 12/2008

Preventing Device Lockout via Censorship on MPC55xx and MPC563x Families

by: David Connelly
Austin, TX, USA
MSG Applications Engineer

1 Introduction

The MPC55xx and MPC563x family of microcontrollers have a feature called censorship which is a secure way to prevent unauthorized access to the device. The key to this feature is a 32-bit word that is stored in the shadow block comprised of censorship and serial boot control information, each sixteen bits in length. This 32-bit censorship control word is read and interpreted by hardware as part of the boot process. It is used in conjunction with the BOOTCFG bit(s) to enable or disable the internal flash memory and the Nexus/JTAG debug interface. The censorship control word is programmed during manufacture to be 0x55AA_55AA. This results in a device that is not censored and uses a flash based password for serial boot mode, or a preset public password (FEED_FACE CAFE_BEEF) for internal boot mode.

Contents

freescale™ semiconductor

2 Background

Censorship is a protection scheme required by automotive manufacturers to prevent access or modification to proprietary powertrain and engine control code, located internally in the microcontroller. Unauthorized changes can result in safety issues, failure to meet governmental standards, or result in internal engine or transmission damage. The powerful development features built into the chips today allow almost limitless internal access to the device with the correct tools. Censorship attempts to prevent any internal access or modification to the device after it has been setup for intended use.

WARNING

If used incorrectly, this feature can cause the developer to be locked out of the device while modifying the shadow block, making the device useless. This can happen if the device is unintentionally reset while erasing the shadow block, or there is a programming or verifying error while updating the censorship information. In this case, the transmission of trace messages and Nexus/JTAG access to memory mapped resources is not possible. External access to flash memory is also disabled.

3 Reset Options

The combination of the selected BOOTCFG pins and the censorship control word defines the boot mode and determines if flash and nexus are enabled or disabled. Boot mode options are internal, serial, and external. The external boot mode is not typically discussed as a boot option, but is used in this example as a means to work around censorship if the proper preventive steps are taken. The boot modes are based on the BOOTCFG pin(s), as illustrated in Table 1.

Table 1. Boot Modes

BOOTCFG [0:1]	Censorship Control 0x00FF_FDE0	Serial Boot Control 0x00FF_FDE2	Boot Mode Name	Internal Flash State	Nexus State	Serial Password
00	!0x55AA[1]	Any Value	Internal—Censored	Enabled	Disabled	Flash
	0x55AA		Internal—Public	Enabled	Enabled	Public
01	Any Value	0x55AA	Serial—Flash Password	Enabled	Disabled	Flash
		!0x55AA	Serial—Public Password	Disabled	Enabled	Public
10	!0x55AA	Any Value	External—No Arbitration—Censored	Disabled	Enabled	Public
	0x55AA		External—No Arbitration—Public	Enabled	Enabled	Public
11	!0x55AA	Any Value	External—External Arbitration—Censored	Disabled	Enabled	Public
	0x55AA		External—External Arbitration—Public	Enabled	Enabled	Public

1 ! = NOT, as in !0x55AA, which means all values except 0x55AA. Do not use 0x0000 or 0xFFFF for the value of the censorship control or serial boot control words.

The nexus port controller is held in reset after it is disabled. When the internal flash state is considered disabled, the flash bus interface unit returns a bus error when access is attempted to the flash module.

MPC551x devices have only one BOOTCFG signal. An external boot is not an option for these devices. The internal boot mode also uses a reset configuration half word (RCHW) and a reset vector, both

contained in the internal flash. The RCHW is a 16-bit value that contains a fixed 8-bit boot identifier and a few configuration bits. The reset vector is a 32-bit value at RCHW+0x4 that contains the address of the first instruction to be executed. The RCHW is expected to be the first half-word in one of the low-address space small flash blocks. These addresses are shown in the following table.

Table 2. Low Address Space (LAS) Block Memory Addresses

Block	Address
0	0x0000_0000
1	0x0000_4000
2	0x0001_0000
3	0x0001_C000
4	0x0002_0000
5	0x0003_0000

Table 3. LAS Configuration

Block	Address
0	0x0000_0000
1	0x0000_4000
2	0x0000_8000
3	0x0000_C000
4	0x0001_0000
5	0x0001_4000
6	0x0001_8000
7	0x0001_C000
8	0x0002_0000
9	0x0003_0000

If a valid RCHW is not located in either internal or external flash (the boot identifier field of RCHW is not 0x5A), or if BOOTCFG pin(s) are asserted for serial boot at the negation of the RESET pin, the RCHW is not then applicable and the serial boot mode is performed.

The serial boot mode allows the user to download a program into internal SRAM via the eSCI or the FlexCAN and then execute. This mode is selected by default if internal flash is blank or invalid while internal boot mode is selected. A 64-bit user programmable password is used for protection in serial-boot mode. If more detail is required regarding serial boot mode, refer to the *MPC5566 Reference Manual*.

External boot mode is a mode intended primarily for verification and test purposes and can be used for additional storage external to the device. However, in external boot mode there are no restrictions on the use of Nexus/JTAG. The internal flash is disabled when a device is in censored mode. The Nexus/JTAG module allows communication to the device. This mode provides a mechanism for code to be downloaded and executed from SRAM in the event of a lockout.

4 Preserving Censorship Information

The censorship information is located in the shadow block at address 0x00FF_FDE0 on the MPC55xx. Careful steps must be taken to preserve these values while modifying the shadow block. Before any attempts to erase the shadow block, it is recommended to setup a proper RCHW and reset vector in one of the LAS boot blocks. It is not necessary to have the recovery code in this block. This RCHW and reset vector enable code to be executed in the event of a lockout. With this method, access to RSTCFG, PLLCFG, and BOOTCFG pins is required. If a lockout situation occurs, the device can be recovered by using a combination of external and internal boot mode. First download code to SRAM using external boot mode. Nexus can be used regardless of the censorship word. Then, the internal boot mode can be used to take advantage of a reset vector pointing to SRAM that allows code to execute. The code erases and restores the censorship information in the flash block. The following sequence shows the recommended steps to modify the shadow block:

1. Prior to modifying shadow block, erase block0 of the flash
2. Program the RCHW value (0x005A_0000) at the first 32-bit word location of block0, and the reset vector value (0x4000_0000) at location +0x4 to point to SRAM space.

NOTE

Reset vector can point to any SRAM location where code resides.

3. Erase the shadow block or any modification desired
4. Before any reset, program the default censorship word back into the shadow block for the device to remain unlocked
5. Perform a program verify of the shadow block
6. Erase block0 and program user data back into block0, if applicable

If the censorship word is not programmed correctly with error in step 3, 4, or 5, the device then locks and the recovery code that runs out of the SRAM can be downloaded and executed.

5 Restoring Censorship Word

The following steps allows the user to run code out of SRAM to unlock the device, if the RCHW and the reset vector are programmed correctly prior to being locked out. External boot mode allows Nexus to download the code into SRAM, the internal boot mode is then enabled to execute the code.

1. Set external boot mode
   a) RSTCFG = 0b0
   b) BOOTCFG[0:1] = 0b10

NOTE

For MPC551x use BOOTCFG[0] = 1 for serial boot mode

   c) PLLCFG[0:1] = 0b01
2. Connect Nexus/JTAG and power up device.

NOTE

In this mode, Nexus is enabled for downloading code and the internal flash is disabled.

3. Load the program into the SRAM that erases and restores censorship word. The entry point of the code needs to start at the reset vector location, see step 2
4. Set internal boot mode, BOOTCFG[1:0] = 0b00 and reset the device (no power down)

After setting the internal boot mode and resetting the device, the code in SRAM erases the shadow block and reprograms the censorship word. The device is then successfully unlocked. The default serial boot information also needs to be programmed into the proper locations see Table 4

Table 4. Serial Boot Flash Password

Address: 0x00FF_FDD8 Value: 0xFEED

MSB	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Binary value	1	1	1	1	1	1	1	0	1	1	1	0	1	1	0	1
Hex value	F				E				E				D			

Address: 0x00FF_FDDA Value: 0xFACE

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Binary value	1	1	1	1	1	0	1	0	1	1	0	0	1	1	1	0
Hex value	F				A				C				E			

Address: 0x00FF_FDDC Value: 0xCAFE

MSB	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
Binary value	1	1	0	0	1	0	1	0	1	1	1	1	1	1	1	0
Hex value	C				A				F				E			

Address: 0x00FF_FDDE Value: 0xBEEF

	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
Binary value	1	0	1	1	1	1	1	0	1	1	1	0	1	1	1	1
Hex value	B				E				E				F			

These change only if a custom serial boot password is required.

It is important to mention that the serial boot mode also allows Nexus access when the serial boot word is anything other than 0x55AA. The word Censorship is a don't care in serial boot mode. Therefore, this provides a means to download to SRAM, similar to the external boot mode detailed in Table 4.

Another way to restore censorship information is to erase and program all of the boot code into block0 prior to modifying the shadow block. The shadow block is a special case and is considered to be part of all partitions, therefore there are restrictions with this method. Flash partitions are comprised of at least two

blocks and are grouped this way to allow read while write (RWW) access to one partition while programming or erasing another. Because the intention is to run code out of the main flash block to erase and program the shadow block, run the program and erase operations from SRAM. The user code executed from the flash has to copy the functions to SRAM and then execute them from there. An example of how to do this with the standard software drivers (SSD) is included in Section 6, “Censorship Recovery Code”.

When using this method, if the user is locked out and the appropriate block was configured with all of the user code, switch to internal boot mode (BOOTCFG=[0:0]) to run the code out of block0. To run the recovery code out of flash, the RCHW and reset vector needs to be properly setup for block0, or any other valid block. If block0 started at 0x0000_0000, then:

@ 0x0000_0000 = 0x005A_0000 (RCHW)

@ 0x0000_0004 = 0x0000_0008 (Reset Vector)

The RCHW is recognized as valid and then the reset vector tells the core where to start executing code. The entry point of the flash code for this example would start at address 0x00000008.

6 Censorship Recovery Code

The recovery code erases the shadow row, restores the censorship key, and the serial boot code. Example 1 is a custom example code that shows how to do this:

Example 1. Custom Censorship Recovery Code

```
#include "m_h7f.h"

#define W32(addr, value) (*((volatile unsigned int *) (addr)) = (value))
#define FLASH (*( struct H7F_tag *) 0xC3F88000)

#define SHADOW_ADDRESS              0x00FFFC00
#define CENSORSHIP_ADDR             0x00FFFDE0
#define SERIAL_BOOT1_ADDR           0x00FFFDD8
#define SERIAL_BOOT2_ADDR           0x00FFFDDC
#define SERIAL_PASSWORD1            0xFEEDFACE
#define SERIAL_PASSWORD2            0xCAFEBEEF
#define LOW_ADDR_KEY                0xA1A11111
#define S_LOW_ADDR_KEY              0xC3C33333
#define LOW_ADDR_SR_UNLOCK          0x800FFFFF

void main(void)
{
        /* enable shadow space */
        FLASH.LML.R = LOW_ADDR_KEY;
        FLASH.LML.R = LOW_ADDR_SR_UNLOCK;
        FLASH.SLL.R = S_LOW_ADDR_KEY;
        FLASH.SLL.R = LOW_ADDR_SR_UNLOCK;

        /* step1. erase shadow row */
        FLASH.MCR.B.ERS = 1;
        W32(SHADOW_ADDRESS, 0x0);                    //interlock write
        FLASH.MCR.B.EHV = 1;
        while (FLASH.MCR.B.DONE==0);
        FLASH.MCR.B.EHV = 0;                         //clear EHV
        FLASH.MCR.B.ERS = 0;                         //clear ERS
```

```
        /* step2. program the serial boot word */
        FLASH.MCR.B.PGM = 1;
        W32(SERIAL_BOOT1_ADDR, SERIAL_PASSWORD1);   //interlock write
        W32(SERIAL_BOOT2_ADDR, SERIAL_PASSWORD2);   //interlock write
        FLASH.MCR.B.EHV = 1;
        while (FLASH.MCR.B.DONE==0) ;
        FLASH.MCR.B.EHV = 0;                        //clear EHV
        FLASH.MCR.B.PGM = 0;                        //clear PGM

        /* step3. program the censorship control word  */
        FLASH.MCR.B.PGM = 1;
        W32(CENSORSHIP_ADDR, 0x55aa55aa);           //interlock write
        W32(CENSORSHIP_ADDR+4, 0xffffffff);         //interlock write
        FLASH.MCR.B.EHV = 1;
        while (FLASH.MCR.B.DONE==0);
        FLASH.MCR.B.EHV = 0;                        //clear EHV
        FLASH.MCR.B.PGM = 0;                        //clear PGM

        while(1);
}
```

In addition to the custom code, there are a set of Standard Software Drivers that are available for general use. These can also be used for the censorship recovery scheme, as shown here:

Example 2. Example 2 – SSD Censorship Recovery Code

```
#include "m_h7f.h"
#include "ssd_h7f.h"

typedef unsigned int UINT32;
#define NO_INTERRUPTS 1 //** Define this globally in your project if needed - used in
z650_init_debug.c **
#define W32(addr, value)  (*((UINT32 *) (addr)) = (value))     //write 32 bits of data
#define FLASH               (*( struct H7F_tag *) 0xC3F88000)  //0xFFFF8000 for MPC551x

#define SHADOW_ADDRESS0x00FFFC00
#define CENSORSHIP_ADDR0x00FFFDE0
#define SERIAL_BOOT1_ADDR0x00FFFDD8
#define SERIAL_BOOT2_ADDR0x00FFFDDC
#define SERIAL_PASSWORD10xFEEDFACE
#define SERIAL_PASSWORD20xCAFEBEEF
#define LOW_ADDR_KEY0xA1A11111
#define S_LOW_ADDR_KEY0xC3C33333
#define CENSORSHIP_WORD0x55AA55AA
#define LOW_ADDR_SR_UNLOCK0x800FFFFF
#define LBLOCK0           0x00000001
#define NOBLOCK           0
#define DRIVER_BUFFER_SIZE150                                  //max size of driver in words

SSD_CONFIG ssdConfig = {
    H7F_CTL_REG_BASE,       /* H7F control register base */
    FLASH_ARRAY_BASE,       /* base of main array */
    0,                      /* size of main array */
    0,                          /* base of shadow row */
    0,                          /* size of shadow row */
    0,                      /* block number in low address space */
```

```
    0,                      /* block number in middle address space */
    0,                      /* block number in high address space */
    0,                             /* flash page size selection */
    FALSE                   /* debug mode selection */
};

extern const UINT32 FlashErase_C[];
extern const UINT32 FlashProgram_C[];

UINT32 ssd_RAM[DRIVER_BUFFER_SIZE];
void CopyFunction(const UINT32 function[], UINT32 ram_destination[], UINT32 size);   //only
used if running code out of flash

// RUN_FROM_FLASH -->> Used for executing this code out of flash.
// Functions running from flash block need to be copied to RAM in order to erase shadow block
since
// they are in the same flash partition (RWW restriction)
#define RUN_FROM_FLASH  1

void main(void)
{
UINT32 data[2], status;

        pFLASHERASE    FlashErase    = (pFLASHERASE)    FlashErase_C;
        pFLASHPROGRAM  FlashProgram  = (pFLASHPROGRAM)  FlashProgram_C;

        // enable shadow space
        FLASH.LML.R = LOW_ADDR_KEY;
        FLASH.LML.R = LOW_ADDR_SR_UNLOCK;
        FLASH.SLL.R = S_LOW_ADDR_KEY;
        FLASH.SLL.R = LOW_ADDR_SR_UNLOCK;

        // Setup ssd with proper parameters for shadow row
        ssdConfig.h7fRegBase = H7F_CTL_REG_BASE;
        ssdConfig.mainArrayBase = FLASH_ARRAY_BASE;
        ssdConfig.shadowRowBase = SHADOW_BASE_1K;
        ssdConfig.shadowRowSize = ONE_KB;
        ssdConfig.pageSize = H7FA_PAGE_SIZE;

        #if RUN_FROM_FLASH
                CopyFunction(FlashErase_C, ssd_RAM, DRIVER_BUFFER_SIZE);
                FlashErase = (pFLASHERASE) ssd_RAM;
        #endif

        // Erase Shadow Row
        status = FlashErase(&ssdConfig, TRUE, NOBLOCK, NOBLOCK, NOBLOCK, NULL_CALLBACK);

        #if RUN_FROM_FLASH
                CopyFunction(FlashProgram_C, ssd_RAM, DRIVER_BUFFER_SIZE);
                FlashProgram = (pFLASHPROGRAM) ssd_RAM;
        #endif

        // program serial password
        data[0] = SERIAL_PASSWORD1;
        data[1] = SERIAL_PASSWORD2;
        status = FlashProgram(&ssdConfig, 0x00FFFDD8, 8, (UINT32)data, NULL_CALLBACK);
```

```
        // program censorship and serial boot info
        data[0] = CENSORSHIP_WORD;
        data[1] = 0xffffffff;
        status = FlashProgram(&ssdConfig, 0x00FFFDE0, 8, (UINT32)data, NULL_CALLBACK);

        while(1);
}

void CopyFunction(const UINT32 function[], UINT32 sram_addr[], UINT32 size)
{
        UINT32 i;

        //Number of 32 bit words passed in, just copy it over
        for (i = 0; i < size; i++)
        {
                sram_addr[i] = function[i];
        }
}
```

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or +1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

AN3787

Rev. 0
12/2008