Application Report
SPNA099A–October 2006–Revised February 2007

Modifying the Memory Security Module Keys of Catalog TMS470 Devices

John Mangino *TMS470 Applications*

ABSTRACT

This document shows how to reprogram the Memory Security Module (MSM) keys on the TMS470 devices. These examples are intended as a reference to enable the user to modify the MSM keys and unlock the part using the IAR tools. The reference design includes TMS470 software for use with the IAR Embedded Workbench™.

Contents

1 Introduction

This application report introduces an understanding of the Memory Security Module (MSM), unlocking the MSM, and modifying the MSM keys on the TMS470 flash devices. These examples are using the IAR Embedded Workbench and have information for code generation specific to the IAR tools.

Note: Do not overwrite the flash protection keys and the Memory Security Module (MSM) keys. If the keys are rewritten and the data is not known, the part cannot be reprogrammed or accessed in the case of the MSM.

1.1 Memory Security Module (MSM) Overview

The MSM provides a hardware firewall to prevent unauthorized users from accessing on-chip memories via the debug/test ports or an external memory interface.

The security modules use a password or key to protect against read and write accesses to protected memory. The device is secured by its 128-bit password (four 32-bit words). The device is unsecured by executing the password match flow (PMF) described in the *TMS470R1x Memory Security Module Reference Guide* (TI literature number SPNU243).

If code is currently running from inside secure memory, secure code can access secure data. If code is running from unsecured memory, all read/write accesses to secure memories are blocked. User code can dynamically jump into and out of secure memory, thereby allowing secure function calls from unsecured memory. For instance, interrupt service routines can be placed in secure memory, even if the main program loop is run from unsecured memory.

1.2 Unlocking TMS470 Devices With Memory Security

There are two considerations for unlocking the MSM on TMS470 devices. The first is when the device has the default keys of 0xFFFFFFFF, and the second is when the keys have been modified.

1.2.1 Unlocking Devices With the Default MSM Keys

To unlock the device when the MSM keys contain the default values of 0xFFFFFFFF, the user must perform four reads of the password or MSM key locations in the flash (MSMPWL 0–3). These are located in the second-from-the-last four words of the first sector of flash memory. The flash protection keys are the last four words in the first sector (the MSM keys precede the flash protection keys).

Perform dummy reads of the four consecutive MSMPWL locations (MSMPWL 0–3). Secure memory is fully accessible immediately after this operation is completed.

Note: A dummy read of the password location registers must be performed before reading, writing, or programming secure memory, even though the device is not protected with a password.

1.2.2 Unlocking Devices With Modified MSM Keys

To unlock the device when the MSM keys contain modified values, the user must perform four reads of the password or MSM key locations in the flash (MSMPWL 0–3). These reads are followed by writing the modified values of the Write password into the MSMKEY registers. If the password in the MSMKEY registers matches the password, the device becomes unsecured. If the passwords do not match, it stays secure.

The MSM keys (MSMPWL 0–3) are located in the second-from-the-last four words of the first sector of flash memory. The flash protection keys are the last four words in the first sector (the MSM keys precede the flash protection keys). You may need to configure the TMS470 memory map to unsecure the MSM and provide access to on-chip device memories accessing the password locations. In flash devices, you can change a password any time if you know the old password. These registers are only accessible in *privileged mode*.

2 MSM Matching and Modification

The MSM keys are modified by writing to the second to the last four words of the first sector in flash. The process flow is:

1. Perform four dummy reads of the MSM password flash locations (MSMPWL 0–3).
2. If the MSM password is modified from the default values, write the current password to the MSM key registers (0xFFFFF700, 0xFFFFF704, 0xFFFFF708, 0xFFFFF70C).
3. Match the current flash protection keys to unlock the flash.
4. Erase the flash.
5. Write the new MSM password and flash protection keys.

2.1 Matching the MSM Password

As with any accesses to MSM protected flash or RAM, the device must be unlocked.

2.1.1 IAR Workbench

The IAR workbench requires three files to implement the MSM key lock and unlock.

1. tms470r1flash.mac – Flash macro file for the IAR workbench
2. tms470r1a288_lnk_flash.xcl – Supplied linker file (use project-specific linker file)
3. msm_key.c – File containing new MSM keys

2.1.1.1 Flash Macro File

The IAR workbench has a flash macro file that configures the memory prior to loading the flash programming routines. For the TMS470R1B1M the file name is “FlashTMS470R1A288.mac”. This file is also used to unlock the MSM protected device. To unlock the device through the IAR workbench:

1. Set a parameter that prevents the dummy reads from stopping the workbench from continuing from a read error generated from the access to protected memory (__emulatorStatusCheckOnRead(0);).
2. Configure the memory map.
3. Perform the four dummy reads of the MSM key location in flash (device specific):
   __readMemory32(0xxxxxxxxx, "Memory");
   __readMemory32(0xxxxxxxxx, "Memory");
   __readMemory32(0xxxxxxxxx, "Memory");
   __readMemory32(0xxxxxxxxx, "Memory");
4. If the MSM password is modified from default values, write the current values to the MSMKEY registers. This step may be omitted if the MSM keys are the default values 0xFFFFFFFF.
   __writeMemory32(0xxxxxxxxx, 0xFFFFF700, "Memory");
   __writeMemory32(0xxxxxxxxx, 0xFFFFF704, "Memory");
   __writeMemory32(0xxxxxxxxx, 0xFFFFF708, "Memory");
   __writeMemory32(0xxxxxxxxx, 0xFFFFF70C, "Memory");
5. Set the parameter that enables error reports from (__emulatorStatusCheckOnRead(1);).
6. The part is now unlocked.

Below is a sample of the macro file:

```
// ---------------------------------------------------------------------------
// This file contains the flash set up configuration for the TMS470R1B1M.
//
// The Memory Security Module (MSM) unlocking functionality is included.
//
// Revision : 0.04 JM 05/14/06
//
//----------------------------------------------------------------------------

PrepareMemory( )
{
// B1M
  __message "Executing memory setup macro B1M Flash\n";

  __emulatorStatusCheckOnRead(0);

  /* disable illegal address reset */
  __writeMemory32( 0x00004007, 0xffffffe0, "Memory" ) ;  // SYSECR

  /* set memory select 0 block size to 1M */
  __writeMemory32( 0x000000B0, 0xfffffe04, "Memory" ) ;  // MFBALR0
  __writeMemory32( 0x00000000, 0xfffffe00, "Memory"); // MFBAHR0

  /* activate RAM at 0x00000000 0k that is connected to memory select 1 */
  __writeMemory32(0x00000000, 0xfffffe08, "Memory"); // MFBALR1
  __writeMemory32(0x00000000, 0xfffffe0c, "Memory"); // MFBAHR1

  /* activate RAM at 0x00300000 32k that is connected to memory select 2 */
  __writeMemory32( 0x00000040, 0xfffffe10, "Memory" ) ;  // MFBAHR2
  __writeMemory32( 0x00000070, 0xfffffe14, "Memory" ) ;  // MFBALR2

  /* activate RAM at 0x00000000 0k that is connected to memory select 3 */
  __writeMemory32( 0x00000000, 0xfffffe18, "Memory" ) ;  // MFBAHR3
  __writeMemory32( 0x00000000, 0xfffffe1c, "Memory" ) ;  // MFBALR3 0K

  /* activate HET RAM at 0x00800000 1k that is connected to memory select 4 */
  __writeMemory32( 0x00000080, 0xfffffe20, "Memory" ) ;  // MFBAHR4
  __writeMemory32( 0x00000020, 0xfffffe24, "Memory" ) ;  // MFBALR4

  /* activate new mapping by writing 1 to MFBALR0.8 */
  __writeMemory32( 0x000001B0, 0xfffffe04, "Memory" ) ;  // MFBALR0

  __writeMemory32(0x80000000, 0xfffff724, "Memory");

  __readMemory32(0x0000ffe0, "Memory");
  __readMemory32(0x0000ffe4, "Memory");
  __readMemory32(0x0000ffe8, "Memory");
  __readMemory32(0x0000ffec, "Memory");

  __writeMemory32(0xFFFFFFFF, 0xfffff700, "Memory");  // Enter MSM password 0 here
  __writeMemory32(0xFFFFFFFF, 0xfffff704, "Memory");  // Enter MSM password 1 here
  __writeMemory32(0xFFFFFFFF, 0xfffff708, "Memory");  // Enter MSM password 2 here
  __writeMemory32(0xFFFFFFFF, 0xfffff70c, "Memory");  // Enter MSM password 3 here

  __emulatorStatusCheckOnRead(1);

  __writeMemory32( 0x00000003, 0xfffffffdc, "Memory" ) ;  // GCR (SYSCLK * 8)/4 = SYSCLK
}
```

2.1.2 Flash API Routines

Unlocking the device using the flash API routines is the same procedure as with the IAR workbench. To unlock the device through the IAR workbench:

1. Use the current memory map configuration or modify it to the desired configuration.
2. Perform the four dummy reads (these variables are device specific and verify name and memory location):

   TMP = MSMPWL0;

   TMP = MSMPWL1;

   TMP = MSMPWL2;

   TMP = MSMPWL3;
3. If the MSM password is modified from default values, write the current values to the MSMKEY registers:

   MSMKEY0 = NewKey0;

   MSMKEY1 = NewKey1;

   MSMKEY2 = NewKey2;

   MSMKEY3 = NewKey3;
4. The part is now unlocked.

2.1.3 MSM Password Locations

The MSM passwords are four 32-bit words in flash, located at the end of [(sector 0) – 8 words] in the TMS470 devices.

Device	Flash Protection Key Location
TMS470R1A64	0x00001FE0–0x00001FEF
TMS470R1A128	0x00001FE0–0x00001FEF
TMS470R1A256	0x00001FE0–0x00001FEF
TMS470R1A288	0x00001FE0–0x00001FEF
TMS470R1A384	0x00001FE0–0x00001FEF
TMS470R1B512	0x00003FE0–0x00003FEF
TMS470R1B768	0x00003FE0–0x00003FEF
TMS470R1B1M	0x0000FFE0–0x0000FFEF

2.2 Modifying MSM Passwords

The MSM passwords are modified by writing to the MSM password locations in flash.

2.2.1 IAR Workbench

The MSM passwords and flash protection keys are protected from writes with the –allownewkeys command in the IAR workbench. The command line is located in the **Options** > **Debugger** > **Download** pane of the tool. Editing the flash loader overview with the following command unlocks the flash:

--allownewkeys --clock12000

Note: The MSM and flash protection keys are independent mechanisms. The MSM does not have to be unlocked to update the flash, if internal code is making the modification. For the workbench to flash program the TMS470, the part must be unlocked and the flash protection keys must be matched. Since the MSM prevent unauthorized accesses, the flash protection keys can remain in the default value. The specific application may deem that both should be modified, such as if the internal flash update code is invoked from a port command. In this case, both the MSM and flash protection keys may be modified, and the flash protection keys may be sent through the port to allow flash modification.

If the flash keys are modified, use the following command line instruction:

--flashkey0 0xFFFFFFFF --flashkey1 0xFFFFFFFF --flashkey2 0xFFFFFFFF --flashkey3 0xFFFFFFFF --allownewkeys --clock12000

The **–flashkeyx** command inserts the existing flash keys to unlock the flash for reprogramming. 0xFFFFFFFF is the default of the keys, and it is not necessary to enter these keys because the workbench automatically inserts them when programming the flash. However, if the keys are changed, the existing keys must be entered.

The **–allownewkeys** command signals the IAR workbench to allow the flash protection key and MSM key locations to be modified. This is required to protect the user from unintentionally rewriting the flash protection and MSM keys.

Note: The **–clock** command sets the delay factor for flash programming. On the A256 EVM, a 12-MHz system clock frequency is used for reprogramming. The 12000 entry represents 12 MHz and creates a delay factor of 6. See the flash API documentation for a full explanation of the clock frequency and flash programming.

The C program below assigns the MSM and FlashKeys array to the FLASHKEYS segment MSMKEYS segment defined in the linker file that points to the MSM and flash protection key location for the specific device. The **_root** directive prevents the compiler from optimizing out the MSMKeys and FlashKeys variables.

```
//-------------------------------------------------------------------------------
// This module modifies flash keys
//-------------------------------------------------------------------------------


#pragma location="FLASHKEYS"
__root const unsigned long FlashKeys[4] =
{
  0xFFFFFFFF,
  0xFFFFFFFF,
  0xFFFFFFFF,
  0xFFFFFFFF
};
```

```
//-------------------------------------------------------------------------------
// This module modifies MSM keys
//-------------------------------------------------------------------------------


#pragma location="MSMKEYS"
__root const unsigned long MSMKeys[4] =
{
  0xFFFFFFFF,
  0xFFFFFFFF,
  0xFFFFFFFF,
  0xFFFFFFFF
};
```

The Flash Key Segment is defined in the following example segment. This segment is added to the linker files and is device specific.

```
////////////////////////////////////////////////////////////////////////
// Flash Key Segment
////////////////////////////////////////////////////////////////////////

-Z(CODE)FLASHKEYS=(ROMSTART+0x1FF0):+0x10

////////////////////////////////////////////////////////////////////////
// MSM Key Segment
////////////////////////////////////////////////////////////////////////

-Z(CODE)MSMKEYS=(ROMSTART+0x1FE0):+0x10
```

2.2.2 Flash API Modules

When using the flash API modules, the user creates the routine to write the flash. If an external access is not needed to update the flash, the MSM can remain locked. If new code is loaded from a port, the MSM may need to be unlocked, depending on the circumstances of the port access. Once the part is unlocked and the flash memory is opened, the flash protection key and MSM key locations can be overwritten as ordinary flash memory.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
Low Power Wireless	www.ti.com/lpw	Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments
Post Office Box 655303 Dallas, Texas 75265