Memory Integrity Detection (MID)

NOTE: This chapter is an excerpt from the *MSP430x5xx and MSP430x6xx Family User's Guide*. The most recent version of the full user's guide is available from http://www.ti.com/lit/pdf/slau208.

Memory Integrity Detection (MID) is a program and data protection mechanism that is available on several device families (for example, MSP430F6659). It provides a high level of operation safety for fault-critical application areas. This chapter explains how to use the firmware for the level of operational safety and overall fault response that suits different applications.

Topic **Page**

1.1 MID Overview

The MID is an add-on to the MSP430 flash memory controller. MID provides additional functionality over the regular flash operation methods as described in the Flash Memory Controller chapter.

The main purpose of the MID function is to help gain higher reliability of flash content and overall system integrity in harsh environments and in applications requiring such features. The additional level of security is reached by calculating parity information.

The complete MID solution consists of the blocks *Parity Generator and Parity Check* and *MID ROM*. The *Parity Generator and Parity Check* provides all of the necessary logic elements needed to identify bit errors in the whole memory array. The on-chip MID ROM contains the MID Support Software, and this software performs all the necessary tasks to operate MID. The built-in MID functions provide all functionality to use the MID features.

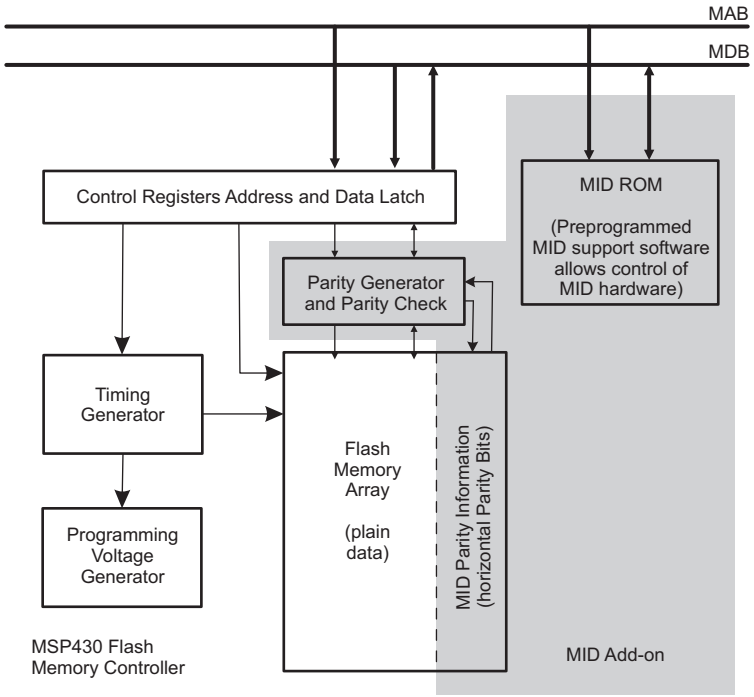
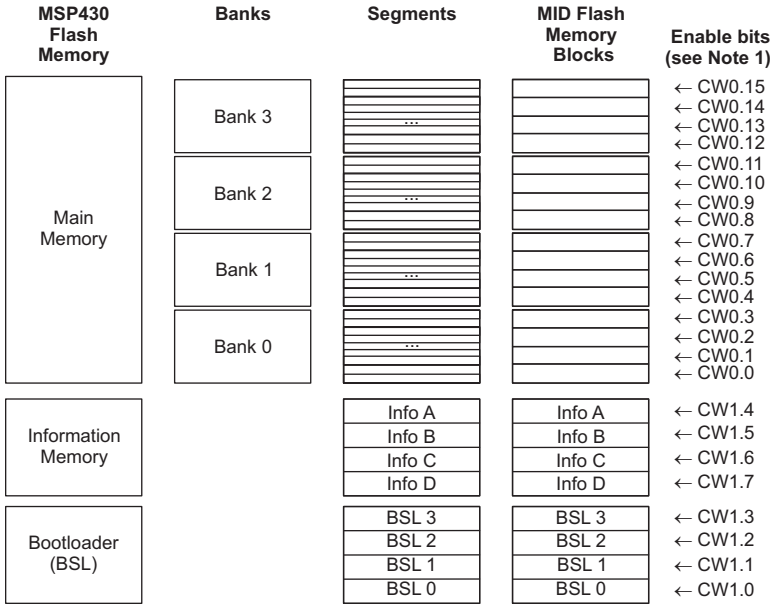


Figure 1-1. Block Diagram of MID Implementation

1.2 Flash Memory With MID Support

The MSP430 flash memory is partitioned into different memory areas—main and information memory, banks, segments, and memory blocks—where MID protection can be enabled. Figure 1-2 shows an example for a typical MSP430 flash segmentation including the MID feature.



(1) The cw0.x and cw1.x bits are used to enable MID functionality for the individual memory ranges. Further information can be found in Section 1.6.1.

Figure 1-2. Overview of MSP430 Flash Memory Segmentation

The whole flash memory array consist of MID supported flash memory blocks. For a device with 512KB of flash main memory, each MID flash memory block has a size of 32KB (main memory divided by 16). Each row consists of one word of plain data (16 bits) and a horizontal parity bit (H-parity bit).

Erased segments show all ones in the data array field and horizontal parity. Writing to flash memory (with MID after reset) automatically writes the horizontal parity bits along with the data bits. Writing to the plain data field can, of course, be interrupted and continued in any order. Adding content after the horizontal parity has been written is impractical, as the horizontal parity information changes as well. The whole segment (not just a single MID memory block) would need to be erased before it can be written again. The shown method is excellent for data content of static nature like code, tables, and so on. For data acquisition into flash, other methods (for example, majority vote) are more suitable; but complete blocks of acquisition data can be protected with this method again.

1.3 MID Parity Check Logic

Any access to MID enabled flash memory causes a verification of its horizontal parity in the background. It does not matter if code or data is read from the flash memory. If a parity error is detected, the bus error event "parity error" is triggered and calls the user NMI exception handler. The application software can then react on the failure by, for example, showing an error message on the application's display.

1.4 Detecting Unprogrammed Memory Accesses

All bits are set after erasing the flash memory; this also includes the horizontal parity bit. If an erased memory range is accessed, the MID causes a NMI interrupt, because of a detected parity failure. Only programmed addresses are accessible without a MID failure interrupt; that is especially the case for the content 0xFFFF. If memory content should be 0xFFFF, it must be programmed. This ensures that the horizontal parity bit is cleared (0).

Enabling the MID functionality for nonprogrammed memory ranges allows detecting memory accesses to these nonprogrammed addresses.

1.5 MID ROM

The MID ROM is 1KB of read-only memory. The on-chip MID ROM is factory programmed with MID support software. These software functions are used to enable or disable the MID module. The start address of the MID ROM depends on the MSP430 device; see the device-specific data sheet for specifications.

1.6 MID Support Software Function

The MID is disabled by default after power-up of the device. To use the MID feature, it must be enabled within the application software. Enabling is done by calling the MidInit() function with parameters that define which MID memory blocks should be enabled or disabled.

Table 1-1 list all existing MID functions. These functions are stored in the MID ROM; its start address is defined in the device-specific data sheet.

Table 1-1. Overview of MID Support Software Functions

Function	Address Offset	Description
Revision	0x00	Content of address: 2843h
	0x02	Content of address: 80xyh, xy is the revision word
MidEnable	0x04	Initialization and enabling of MID
MidDisable	0x08	MID is disabled
MidGetErrAdr	0x0C	Returns the error location
MidCheckMem	0x10	Memory check is performed
MidSetRaw	0x14	Writing a data word and parity bit into a defined address
MidGetParity	0x18	Read out horizontal parity bit
MidCalcVParity	0x1C	Calculating vertical parity
Reserved	0x20	
Reserved	0x24	
Reserved	0x28	
Reserved	0x2C	

1.6.1 MidEnable() Function

Function

```
void MidEnable(unsigned short cw0,unsigned short cw1);
```

Function Description

This function initializes and enables MID. The argument *cw0* and *cw1* allow an explicit control what MID flash memory blocks are to be protected. MID feature is disabled after a power-up or BOR. MidEnable() function is expected to be called early after application start. Calling it again later reconfigures the settings.

Parameters

Name	Type	Description
cw0	unsigned short R12.W	Configuration word 0. It is used to activate the MID feature for certain memory ranges. The main memory is divided into 16 blocks. The LSB bit of cw0 activates the lowest address range (see Figure 1-3).
cw1	unsigned short R13.W	Configuration word 1. This bit is used the same as cw0, the only difference is that BAL and Info memory are used instead of main memory (see Figure 1-4).

Figure 1-3. cw0 Parameter

15	14	13	12	11	10	9	8
cw0.15	cw0.14	cw0.13	cw0.12	cw0.11	cw0.10	cw0.9	cw0.8
7	6	5	4	3	2	1	0
cw0.7	cw0.6	cw0.5	cw0.4	cw0.3	cw0.2	cw0.1	cw0.0

Bit	Field	Description
15-0	cw0.x	Main memory is split into MID flash memory blocks. Each MID flash memory block has a size of main memory divided by 16 (for example, for a 512KB main memory, the MID memory block size is 32KB). The cw0.x bits allow to enable MID support for the different flash memory blocks. For example, cw0.0 activates the lowest flash memory block, and cw0.15 activates the highest flash memory block. 0 = MID support is deactivated 1 = MID support is active

Figure 1-4. cw1 Parameter

15	14	13	12	11	10	9	8
Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
7	6	5	4	3	2	1	0
cw1.7	cw1.6	cw1.5	cw1.4	cw1.3	cw1.2	cw1.1	cw1.0

Bit	Field	Description
15-8	Reserved	These bits are reserved. It is strongly recommended to reset (0) these bits.
7	cw1.7	Enables or disables MID for the flash information memory segment D. 0 = MID support is deactivated 1 = MID support is active
6	cw1.6	Enables or disables MID for the flash information memory segment C. 0 = MID support is deactivated 1 = MID support is active
5	cw1.5	Enables or disables MID for the flash information memory segment B. 0 = MID support is deactivated 1 = MID support is active
4	cw1.4	Enables or disables MID for the flash information memory segment A. 0 = MID support is deactivated 1 = MID support is active

Bit	Field	Description
3	cw1.3	Enables or disables MID for the BSL memory 3. 0 = MID support is deactivated 1 = MID support is active
2	cw1.2	Enables or disables MID for the BSL memory 2. 0 = MID support is deactivated 1 = MID support is active
1	cw1.1	Enables or disables MID for the BSL memory 1. 0 = MID support is deactivated 1 = MID support is active
0	cw1.0	Enables or disables MID for the BSL memory 0. 0 = MID support is deactivated 1 = MID support is active

1.6.2 MidDisable() Function

Function

```
void MidDisable(void);
```

Function Description

This function clears the cw0 and cw1 parameters that were set during MidEnable() function call and it disables the MID hardware.

1.6.3 MidGetErrAdr() Function

Function

```
unsigned short * MidGetErrAdr(void);
```

Function Description

This function returns the error location is there was a memory integrity failure. If there is no valid failure address or error location, the function returns the value F'FFFFh.

Note that the MidGetErrAdr() function returns only the correct error address when this function is called prior to a read access of SYSBERRIV register. The code example in Section 1.7 shows where the MidGetErrAdr() function call should be placed.

1.6.4 MidCheckMem() Function

Function

```
void MidCheckMem(unsigned short * startAdr, unsigned short * endAdr);
```

Function Description

This function allows doing a memory integrity check. First, the MidEnable function should be called. This function enables MID and it defines the memory blocks that should be protected. After that the MidCheckMem function can be called. Its parameter list defines an address range that is accessed with wordwise reads. An UNMI interrupt (MID interrupt) is generated in case a parity error occurs and the read address is enabled for MID protection.

Parameters

Name	Type	Description
startAdr	unsigned short *R12.W	Start address for the memory integrity check. The startAdr must be an even number.
endAdr	unsigned short R13.W	End address for the memory integrity check. The endAdr must be an even number. The address defined with endAdr is included in the memory integrity check.

1.6.5 MidSetRaw() Function

Function

```
void MidSetRaw(unsigned short data, unsigned short parity, unsigned short * adr,
          unsigned short flashKey);
```

Function Description

This function writes one word (data) and a separately definable parity bit (parity) to an MID memory address (adr). The Flash memory key is needed to allow access to flash control registers; this parameter is passed through the argument flashKey (see the follwoing example).

Parameters

Name	Type	Description
data	unsigned short R12.W	Data to be written
parity	unsigned short R13.W	If parity = 0, the parity bit 0 is written. If parity <> 0, the parity bit 1 is written.
adr	unsigned short *R14.A	Destination address where raw information is written
flashKey	unsigned short R15.W	Flash memory key. This is needed to allow the MidSetRaw function access to the flash control registers. The passing parameter is usually defined in the standard MSP430 header files; therefore, "FWKEY" can be used here.

Example

```
#include <msp430f6659.h>
const unsigned short FlashAdr=0xFF00; // Flash memory address will be reprogrammed
void main(void)
{ static unsigned short Data;   // variable for data the will be read
  static unsigned short Parity; // H-parity bit that will be read
  WDTCTL=WTDPW+WDTHOLD;         // disable Watchdog
  Data=0x5A5A;                  // data that will be written into flash memory
  Parity=0;                     // parity bit that will be written
  MIDSetRaw(Data,Parity,&FlashAdr,FWKEY); // write data and parity bit
  while(1);
}
```

1.6.6 MidGetParity() Function

Function

```
unsigned short MidGetParity(unsigned short * adr);
```

Function Description

This function returns the parity bit of the appropriate address. Reading the parity bit works only when MID was enabled before calling MidGetParity() function and the appropriate MID memory block is enabled.

Parameters

Name	Type	Description
adr	unsigned short *R12.A	Defines the address where the parity bit should be read.

1.6.7 MidCalcVParity() Function

Function

```
unsigned short MidCalcVParity(unsigned short * startAdr, unsigned short * endAdr);
```

Function Description

This function allows to calculate a vertical parity for a defined memory range.

Parameters

Name	Type	Description
startAdr	unsigned short *R12.A	Defines the start address for calculating vertical parity. The startAdr must be an even number.
endAdr	unsigned short *R13.A	End address for calculating vertical parity. The endAdr must be an even number. The address defined with endAdr is included in the vertical parity calculation.

1.7 User's UNMI Interrupt Handler

If an error is detected, the on-chip MID generates an UNMI interrupt. The application software must manage error handling.

UNMI handler framework for MID error handling:

```
__interrupt void unmi_isr(void)
{   switch(__even_in_range(SYSUNIV, 0x08))
    {
    case 0x00: break;
    case 0x02: break;                       // NMIIFG
    case 0x04: break;                       // OFIFG
    case 0x06: break;                       // ACCVIFG
    case 0x08:                              // BUSIFG
               // If needed, obtain the flash error location here.
               ErrorLocation = MidGetErrAdr();

               switch(__even_in_range(SYSBERRIV, 0x08))
               {   case 0x00: break;        // no bus error
                   case 0x02: break;        // USB bus error
                   case 0x04: break;     // reserved
                   case 0x06:                  // MID error
                   <place your MID error handler code here>
                               break;
                   case 0x08: break;
                   default:   break;
               }
               break;
    default:   break;
    }
```

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ('TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources 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.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/sampterms.htm).

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