KeyStone Architecture
Memory Protection Unit (MPU)

User Guide

Literature Number: SPRUGW5A
June 2013

Release History

Revision	Date	Chapter/Topic	Description/Comments
SPRUGW5A	June 2013	"Registers"	Added EOI value for MPU. (Page 3-6) Updated the description of AID in PROGn_MPPA to clarify the permission check feature (Page 3-7)
SPRUGW5	November 2010	All	Initial Release

Contents

List of Tables

List of Figures

Preface

About This Manual

This document describes the functionality, operational details, and programming information for the KeyStone Architecture Memory Protection Unit (MPU).

This document uses the following conventions:

- Commands and keywords are in **boldface** font.
- Arguments for which you supply values are in *italic* font.
- Terminal sessions and information the system displays are in `screen font`.
- Information you must enter is in **`boldface screen font`**.
- Elements in square brackets ([ ]) are optional.

Notes use the following conventions:

Note—Means reader take note. Notes contain helpful suggestions or references to material not covered in the publication.

The information in a caution or a warning is provided for your protection. Please read each caution and warning carefully.

CAUTION—Indicates the possibility of service interruption if precautions are not taken.

WARNING—Indicates the possibility of damage to equipment if precautions are not taken.

Related Documentation from Texas Instruments

C66x CPU and Instruction Set Reference Guide	SPRUGH7

Trademarks

TMS320C66x and C66x are trademarks of Texas Instruments Incorporated.

All other brand names and trademarks mentioned in this document are the property of Texas Instruments Incorporated or their respective owners, as applicable.

Chapter 1

Introduction

Memory Protection Units (MPU) manage access to memory. The MPU allows ranges to be defined with different permissions and can read fault information. The MPU can also notify the system of a fault through an interrupt.

1.1 Overview

The Memory Protection Unit (MPU) manages access to memory. The MPU allows multiple ranges to be defined and made secure by limiting access only to secure system masters. Access also can be granted or denied to individual system masters based on their privilege ID. The MPU can record a detected fault or invalid access, and notify the system through an interrupt.

1.2 Features

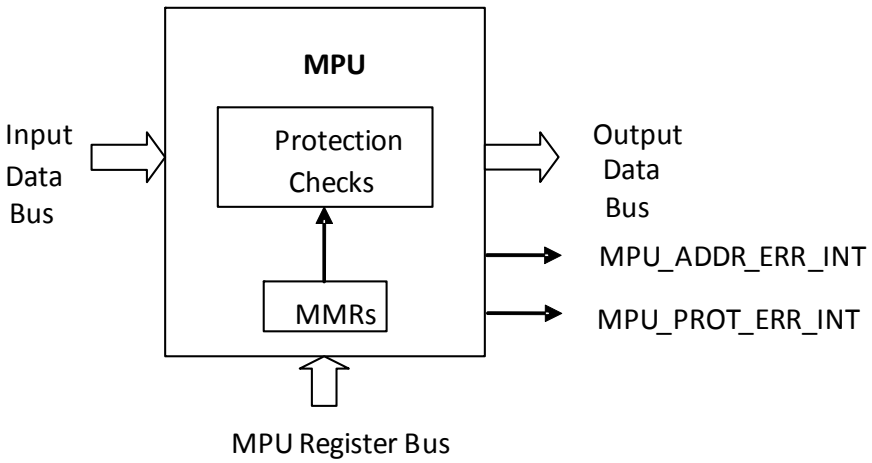
MPU supports the following features:

- Supports multiple programmable address ranges
- Supports secure and debug access privileges. Devices may or may not support secure privileges. Please see the device specific data manual to determine if security is supported
- Supports read, write, and execute access privileges
- Supports privilege ID associations with ranges
- Generates an interrupt when there is a protection violation, and saves violating transfer parameters
- Supports L1/L2 cache accesses
- Supports protection of its own registers

1.3 Functional Block Diagram

Figure 1-1 shows a block diagram of the MPU. An access to a protected memory must pass through the MPU. During an access, the MPU checks the memory address on the input data bus against fixed and programmable ranges. If allowed, the transfer is passed unmodified to the output data bus. If the transfer fails the protection check then the MPU does not pass the transfer to the output bus but rather services the transfer internally back to the input bus (to prevent a hang), returning the fault status to the requestor as well as generating an interrupt about the fault. The MPU generates two interrupts: an address error interrupt (MPU_ADDR_ERR_INT) and a protection interrupt (MPU_PROT_ERR_INT).

Figure 1-1 MPU Block Diagram



Chapter 2

Architecture

This chapter describes the details of the architecture of the MPU.

2.1 Privilege Levels

The privilege level of a memory access determines what level of permissions the originator of the memory access might have. Two privilege levels are supported: supervisor and user.

Supervisor level is generally granted access to peripheral registers and the memory protection configuration. User level is generally confined to the memory spaces that the OS specifically designates for its use.

CORE instruction and data accesses have a privilege level associated with them. The privilege level is inherited from the code running on the CORE. See the *C66x CPU and Instruction Set Reference Guide* in "Related Documentation from Texas Instruments" on page ø-viii for more details on privilege levels of the DSP CORE.

Although master peripherals like EMAC and SRIO do not execute code, they still have a privilege level associated with them.

2.2 Memory Protection Ranges

The MPU divides its assigned memory into address ranges. Each MPU can support multiple programmable address ranges. The programmable address range allows software to program the start and end addresses.

Each address range has the following set of registers:

- **Range start and end address registers (MPSAR and MPEAR):** Specifies the starting and ending address of the address range.
- **Memory protection page attribute register (MPPA):** Use to program the permission settings of the address range.

It is allowed to configure ranges such that they overlap each other. In this case, all the overlapped ranges must allow the access, otherwise the access is not allowed. The final permissions given to the access are the lowest of each type of permission from any hit range.

Addresses not covered by a range are allowed.

Note—In some cases, the amount of physical memory in use may be less than the maximum amount of memory supported by the device. For example, the device may support a total of 512 Mbytes of DDR3 SDRAM memory, but a particular design may populate only 128 Mbytes. In such cases, the unpopulated memory range must be protected in order to prevent unintended/disallowed *aliased* access to protected memory, especially secure memory. One of the programmable address ranges could be used to detect accesses to this unpopulated memory.

2.3 Permission Structures

The MPUs define a per-range permission structure with three permission fields in a 32-bit permission entry. Figure 2-1 shows the structure of a permission entry.

Figure 2-1 Permission Fields

31					26	25	24	23	22	21	20	19	18	17	16
Reserved						Allowed IDs									
						AID15	AID14	AID13	AID12	AID11	AID10	AID9	AID8	AID7	AID6
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Allowed IDs							Reserved	Security		Access Types					
AID5	AID4	AID3	AID2	AID1	AID0	AIX		NS	EMU	SR	SW	SX	UR	UW	UX

2.3.1 Requestor-ID Based Access Controls

Each master on the device has an N-bit code associated with it that identifies it for privilege purposes. This privilege ID accompanies all memory accesses made on behalf of that master. That is, when a master triggers a memory access command, the privilege ID will be carried alongside the command.

See the device data manual for the privilege ID, privilege level (supervisor vs. user), and security level (secure vs. non-secure) of each master on the device.

Each memory protection range has an allowed ID (AID) field associated with it that indicates which requestors may access the given address range. The memory protection unit maps the privilege IDs of all the possible requestors to bits in the allowed IDs field in the memory protection page attribute registers.

- AID0 through AID15 are used to specify the allowed privilege IDs.
- An additional allowed ID bit, AIDX, captures access made by all privilege IDs not covered by AID0 through AID15.

When set to 1, the AID bit grants access to the corresponding ID. When cleared to 0, the AID bit denies access to the corresponding requestor.

2.3.2 Request-Type Based Permissions

The memory protection model defines three fundamental functional access types: read, write, and execute. Read and write refer to data accesses — accesses originating via the load/store units on the CORE or via a mater peripheral. Execute refers to accesses associated with an instruction fetch.

The memory protection model allows controlling read, write, and execute permissions independently for both user and supervisor mode. This results in six permission bits, shown in Table 2-1.

Table 2-1 Request Type Access Controls

Bit	Field	Description
5	SR	Supervisor may read
4	SW	Supervisor may write
3	SX	Supervisor may execute
2	UR	User may read
1	UW	User may write
0	UX	User may execute
End of Table 2-1		

For each bit, a 1 permits the access type, and a 0 denies it. Thus UX = 1 means that User Mode may execute from the given page. The MPU allows the programmer to specify all six of these bits separately. 64 different encodings are permitted altogether, although programs might not use all of them.

2.3.3 Non-Secure and Emulator Accesses

Together, the debug (DBG) and non-secure (NS) bits specify three valid protection levels for each range as described in Table 2-2. The term *debug access* refers to an access initiated through the emulation port of the device.

- If NS =1, then the range is non-secure and any security or debug level may access the range.
- If the NS = 0, then the range is secure and only secure level accesses are allowed.
- In secure mode if EMU = 1, then debug accesses are allowed.
- If EMU = 0, then debug accesses are not allowed.

Note—Only certain masters on the device are capable of generating secure accesses. See the device-specific data manual for more details.

Table 2-2 Protection Levels

NS	DBG	Description
0	0	Secure without debug Only secure accesses are allowed; debug accesses are not allowed.
0	1	Secure with debug Only secure and debug accesses are allowed.
1	X	Not secure All accesses (including debug accesses) are allowed.
End of Table 2-2		

2.4 Protection Check

During a memory access, the MPU checks if the address range of the input transfer overlaps one of the address ranges. When the input transfer address is within a range the transfer parameters are checked against the address range permissions.

The MPU first checks the transfer's privilege ID against the AID settings. If the AID bit is 0, then the range will not be checked. If the AID bit is 1, then the transfer secure and debug parameters are checked against the MPPA values to detect an allowed access.

- If NS =1, the range is non-secure and any security or debug level may access the range.
- If the NS = 0, the range is secure only and only secure level accesses are allowed.
- In secure mode if EMU = 1, debug accesses are allowed.
- If EMU = 0, debug accesses are not allowed.

For non-debug accesses the read, write, and execute permissions also are checked. There is a set of permissions for supervisor mode and another for user mode. For supervisor mode accesses, the SR, SW, and SX bits are checked. For user mode accesses the UR, UW, and UX bits are checked.

If the transfer address range does not match any address range then the transfer is allowed.

In the case that a transfer spans multiple address ranges, all the overlapped ranges must allow the access, otherwise the access is not allowed. The final permissions given to the access are the lowest of each type of permission from any hit range. Therefore, if a transfer matches two ranges, one that is RW and one that is RX, then the final permission is just R.

The MPU has a special mechanism for handling DSP L1/L2 cache controller read accesses. See section 2.5 "DSP L1/L2 Cache Controller Accesses" for more details.

2.5 DSP L1/L2 Cache Controller Accesses

A memory read access which originates from the DSP L1/L2 cache is treated differently to allow memory protection to be enforced by the DSP level. This is because a subsequent memory access that hits in the cache does not pass through the MPU. Instead the memory access is serviced directly by the L1/L2 memory controllers.

During a cache memory read the permission settings stored in the MPU MPPA register are passed to the L1/L2 memory controllers along with the read data. The permissions settings returned by the MPU are taken from the MPPA register that covers the address range of the original request—only the SR, SW, SX, UR, UW, UX, NS and EMU bits are passed.

If the request address is covered by multiple address ranges, then the returned value is the logical AND of all the MPPA permissions. If the transfer address range is not covered by an address range then the transfer is either allowed or disallowed based on the configuration of the MPU.

2.6 MPU Register Protection

Access to the range start and end address registers (MPSAR and MPEAR) and memory protection page attribute registers (MPPA) are also protected. All non-debug writes must be by a supervisor entity. In addition, if the NS bit of the MPPA register is in secure mode (NS = 0) then all writes must be by a secure entity. The NS bit itself is only writable by a secure entity. A protection fault can occur from a register write with invalid permissions and this triggers an interrupt just like a memory access.

A debug write is only allowed if NS = 1 or the EMU = 1 regardless of the secure or privilege attributes. Faults are not recorded (nor interrupts generated) for debug accesses.

2.7 Invalid Accesses and Exceptions

When a transfer fails the protection check, the MPU does not pass the transfer to the output bus. The MPU instead services the transfer locally to prevent a hang and returns a protection error to the requestor. The behavior of the MPU depends on whether the access was a read or a write:

- For a read: The MPU returns zeros, a permission value is 0 (no access allowed), a protection error status.
- For a write: The MPU receives all the write data and returns a protection error status.

The MPU captures system faults due to addressing or protection violations in its registers. The MPU can store the fault information for only one fault, so the first detected fault is recorded into the fault registers and an interrupt is generated. Software must use the FLTCLR register to clear the fault status so that another fault can be recorded. The MPU will not record another fault nor generate another interrupt until the existing fault has been cleared. Also, additional faults will be ignored. Faults are not recorded (no interrupts generated) for debug accesses.

2.8 Reset Considerations

After reset, the NS and EMU bits in the MPPA registers default to 1. This sets the protection level to non-secure with debug access.

2.9 Interrupt Support

2.9.1 Interrupt Events and Requests

The MPU generates two interrupts:

- An address error interrupt (MPU_ADDR_ERR_INT)
- A protection interrupt (MPU_PROT_ERR_INT).

The MPU_ADDR_ERR_INT is generated when there is an addressing violation due to an access to a non-existent location in the MPU register space.

The MPU_PROT_ERR_INT interrupt is generated when there is a protection violation of the either in the defined ranges or to the MPU registers.

The transfer parameters that caused the violation are saved in the MPU registers.

2.9.2 Interrupt Multiplexing

The interrupts from a MPU are combined into a single interrupt called MPUx_INTD. The combined interrupt is routed to the CORE interrupt controllers. See the device-specific data manual for routing of MPU interrupts.

2.10 Emulation Considerations

Memory is protected against emulation accesses by the MPU. Emulator access to memory is controlled through the debug (DBG) and non-secure (NS) bits. See section 2.3.3 "Non-Secure and Emulator Accesses" for more details.

The MPU also protects its range start and end address registers (MPSAR and MPEAR) and memory protection attribute registers (MPPA) against emulation accesses. See section 2.6 "MPU Register Protection" for more details.

Chapter 3

Registers

3.1 MPU Registers

Some registers may not exist based on the configuration of MPU. See the device-specific data manual for more specific offsets.

Table 3-1 shows offsets for MPU registers.

Table 3-1 MPU Register Offsets

Address Offset	Register
0x000	Revision
0x004	Configuration
0x010	Interrupt Raw Status/Set
0x014	Interrupt Enabled Status/Clear
0x018	Interrupt Enable
0x01C	Interrupt Enable Clear
0x020	EOI
0x024 – 0x1FC	Reserved
0x200	Programmable 1 Start Address
0x204	Programmable 1 End Address
0x208	Programmable 1 MPPA
0x20C	Reserved
0x210 – 0x2FC	Additional Programmable Range MMRs
0x300	Fault 1 Address
0x304	Fault 1 Status
0x308	Fault 1 Clear
End of Table 3-1	

3.2 Revision ID Register (REVID)

The revision ID register (REVID) contains MPU revision information.

Figure 3-1 Revision ID Register (REVID)

31	0
Reserved	
R-4E814901	

Legend: R = Read only; -*n* = value after reset

Table 3-2 Revision ID Register (REVID) Field Descriptions

Bits	Field	Description
31-0	Reserved	Revision ID
End of Table 3-2		

3.3 Configuration Register (CONFIG)

The configuration register (CONFIG) contains the configuration value of the MPU. The default value of this register is device-dependent. See the device specific data manual for details.

Figure 3-2 Configuration Register (CONFIG)

31 24	23 20	19 16	15 12	11 1	0
ADDR_WIDTH	NUM_FIXED	NUM_PROG	NUM_AIDS	Reserved	ASSUME_ALLOWED
R-Device Specific	R-Device Specific	R-Device Specific	R-Device Specific	R-0	R-1

Legend: R = Read only; -*n* = value after reset

Table 3-3 Configuration Register (CONFIG) Field Descriptions

Bits	Field	Description
31 – 24	ADDR_WIDTH	Address alignment for range checking 0 = 1-KB alignment 6 = 64-KB alignment
23 – 20	NUM_FIXED	Number of fixed address ranges
19 – 16	NUM_PROG	Number of programmable address ranges. Note: 0 represents 16 programmable ranges since the fields are 4 bits only.
15 – 12	NUM_AIDS	Number of supported AIDs
11 – 1	Reserved	Reserved. Always reads as 0.
0	ASSUME_ALLOWED	Assume allowed bit. When an address is not covered by any MPU protection range, this bit determines whether the transfer is assumed to be allowed or not. 0 = Assume disallowed 1 = Assume allowed

3.4 MPU Interrupt Registers

3.4.1 Interrupt Raw Status/Set Register (IRAWSTAT)

Reading the interrupt raw status/set register (IRAWSTAT) returns the status of all interrupts. Software can write to IRAWSTAT to manually set an interrupt. However, an interrupt is generated only if the interrupt is enabled. Writes of 0 have no effect.

Figure 3-3 Interrupt Raw Status/Set Register (IRAWSTAT) Field Descriptions

31	2	1	0
Reserved		ADDR_ERR	PROT_ERR
R-0		R/W-0	R/W-0

Legend: R = Read only; -*n* = value after reset

Table 3-4 Interrupt Raw Status/Set Register (IRAWSTAT) Field Descriptions

Bit	Field	Description
31 - 2	Reserved	Reserved. Always reads as 0.
1	ADDR_ERR	Addressing violation error. Raw status is read. Write a 1 to set the status. Writing a 0 has no effect.
0	PROT_ERR	Protection violation error. Raw status is read. Write a 1 to set the status. Writing a 0 has no effect.

3.4.2 Interrupt Enabled Status/Clear Register (IENSTAT)

Reading the interrupt enable status/clear register (IENSTAT) returns the status of only those interrupts that are enabled. Software can write to the bits of IENSTAT to clear an interrupt; the interrupt is cleared both from IENSTAT and IRAWSTAT. Writes of 0 have no effect.

Figure 3-4 Interrupt Enabled Status/Clear Register (IENSTAT)

31	2	1	0
Reserved		ENABLED_ADDR_ERR	ENABLED_PROT_ERR
R-0		R/W-0	R/W-0

Legend: R = Read only; -*n* = value after reset

Table 3-5 Interrupt Enabled Status/Clear Register (IENSTAT) Field Descriptions

Bits	Field	Description
31 - 2	Reserved	Reserved. Always reads as 0.
1	ENABLED_ADDR_ERR	Addressing violation error. Enabled status is read. Write a 1 to clear the status. Writing a 0 has no effect.
0	ENABLED_PROT_ERR	Protection violation error. Enabled status is read. Write a 1 to clear the status. Writing a 0 has no effect.

3.4.3 Interrupt Enable Register (IENSET)

Reading the interrupt enable register (IENSET) returns the interrupts that are enabled. Software can write to IENSET to enable an interrupt. Writes of 0 have no effect.

Figure 3-5 Interrupt Enabled Register (IENSET)

31	2	1	0
Reserved		ADDR_ERR_EN	PROT_ERR_EN
R-0		R/W-0	R/W-0

Legend: R = Read only; -*n* = value after reset

Table 3-6 Interrupt Enabled Register (IENSET) Field Descriptions

Bits	Name	Description
31-2	Reserved	Reserved. Always reads as 0.
1	ADDR_ERR_EN	Addressing violation error enable. Write a 1 to set the enable. Writing a 0 has no effect.
0	PROT_ERR_EN	Protection violation error enable. Write a 1 to set the enable. Writing a 0 has no effect.

3.4.4 Interrupt Enable Clear Register (IENCLR)

Reading the interrupt enable clear register (IENCLR) returns the interrupts that are enabled. Software can write to IENCLR to clear/disable an interrupt. Writes of 0 have no effect.

Figure 3-6 Interrupt Enabled Clear Register (IENCLR)

31	2	1	0
Reserved		ADDR_ERR_EN_CLR	PROT_ERR_EN_CLR
R-0		R/W-0	R/W-0

Legend: R = Read only; -*n* = value after reset

Table 3-7 Interrupt Enabled Clear Register (IENCLR) Field Descriptions

Bits	Name	Description
31-2	Reserved	Reserved. Always reads as 0.
1	ADDR_ERR_EN_CLR	Addressing violation error enable. Write a 1 to clear the enable. Writing a 0 has no effect.
0	PROT_ERR_EN_CLR	Protection violation error enable. Write a 1 to clear the enable. Writing a 0 has no effect.

3.4.5 EOI Register (EOI)

The EOI register allows software to indicate when the end of interrupt service is complete. The EOI vector value is dependent on the interrupt handling.

Figure 3-7 EOI Register (EOI) Field Descriptions

31	8	7	0
Reserved		EOI_VECTOR	
R-0		R/W-0	

Legend: R = Read only; -*n* = value after reset

Table 3-8 EOI Register (EOI) Field Descriptions

Bits	Name	Description
31-8	Reserved	Reserved. Always reads as 0.
7-0	EOI_VECTOR	EOI vector value. Write this with the interrupt distribution value in the device. This drives the mpu_eoi_vector output signal. EOI Value is 0 for MPU.

3.4.6 MPU Programmable Range Registers

3.4.6.1 Programmable Range *n* Start Address Register (PROG*n*_MPSAR)

The programmable address start register holds the start address for the range. This register is writable by a supervisor entity only. If NS = 0 (non-secure mode) in the associated MPPA register then the register is also only writable by a secure entity.

The start address must be aligned on a page boundary. The size of the page is 1 Kbyte. The size of the page determines the width of the address field in MPSAR and MPEAR. Default value of this register is device dependent. Refer to device specific datasheet for detail.

Figure 3-8 Programmable Range *n* Start Address Register (PROG*n*_MPSAR)

31	10	9	0
START_ADDR		Reserved	
R/W - Device specific		R -000h	

Legend: R = Read only; R/W = Read/Write; -*n* = value after reset

Table 3-9 Programmable Range *n* Start Address Register (PROG*n*_MPSAR) Field Descriptions

Bits	Name	Range	Description
31 – 10	START_ADDR	Programmable	Start address for range N.
9 – 0	Reserved		Reserved. Always reads as 0.

3.4.6.2 Programmable Range *n* - End Address Register (PROG*n*_MPEAR)

The programmable address end register holds the end address for the range. This register is writable by a supervisor entity only. If NS = 0 (non-secure mode) in the associated MPPA register then the register is also only writable by a secure entity. The end address must be aligned on a page boundary. Default value of this register is device dependent. Refer to device specific datasheet for detail.

Figure 3-9 Programmable Range *n* End Address Register (PROG*n*_MPEAR)

31 – 10	9 – 0
END_ADDR	Reserved
R/W -Device specific	R -3Fh

Legend: R = Read only; R/W = Read/Write; -*n* = value after reset

Table 3-10 Programmable Range *n* End Address Register (PROG*n*_MPEAR) Field Descriptions

Bits	Name	Range	Description
31 – 10	START_ADDR	Programmable	End address for range N.
9 – 0	Reserved		Reserved. Always reads as 1.

3.4.6.3 Programmable Range *n* Memory Protection Page Attribute Register (PROG*n*_MPPA)

The programmable address memory protection page attribute register holds the permissions for the region. This register is writable only by a non-debug supervisor entity. If NS = 0 (secure mode) then the register is also only writable by a non-debug secure entity. The NS bit is only writable by a non-debug secure entity. For debug accesses the register is writable only when NS = 1 or EMU = 1. Default value of this register is device-dependent. See the device-specific data manual for details.

Figure 3-10 Programmable Range *n* Memory Protection Page Attribute Register (PROG*n*_MPPA)

31 – 26	25	24	23	22	21	20	19	18	17	16	15
Reserved	AID15	AID14	AID13	AID12	AID11	AID10	AID9	AID8	AID7	AID6	AID5
R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AID4	AID3	AID2	AID1	AID0	AIDX	Reserved	NS	EMU	SR	SW	SX	UR	UW	UX
R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Legend: R = Read only; R/W = Read/Write

Table 3-11 Programmable Range *n* Memory Protection Page Attribute Register (PROG*n*_MPPA) Field Descriptions (Part 1 of 3)

Bits	Name	Description
31 – 26	Reserved	Reserved. Always reads as 0.
25	AID15	Controls permission check of ID = 15 0 = AID is not checked for permissions. 1 = AID is checked for permissions.
24	AID14	Controls permission check of ID = 14 0 = AID is not checked for permissions. 1 = AID is checked for permissions.
23	AID13	Controls permission check of ID = 13 0 = AID is not checked for permissions. 1 = AID is checked for permissions.

Table 3-11 Programmable Range *n* Memory Protection Page Attribute Register (PROG*n*_MPPA) Field Descriptions (Part 2 of 3)

Bits	Name	Description
22	AID12	Controls permission check of ID = 12 0 = AID is not checked for permissions. 1 = AID is checked for permissions.
21	AID11	Controls permission check of ID = 11 0 = AID is not checked for permissions. 1 = AID is checked for permissions.
20	AID10	Controls permission check of ID = 10 0 = AID is not checked for permissions. 1 = AID is checked for permissions.
19	AID9	Controls permission check of ID = 9 0 = AID is not checked for permissions. 1 = AID is checked for permissions.
18	AID8	Controls permission check of ID = 8 0 = AID is not checked for permissions. 1 = AID is checked for permissions.
17	AID7	Controls permission check of ID = 7 0 = AID is not checked for permissions. 1 = AID is checked for permissions.
16	AID6	Controls permission check of ID = 6 0 = AID is not checked for permissions. 1 = AID is checked for permissions.
15	AID5	Controls permission check of ID = 5 0 = AID is not checked for permissions. 1 = AID is checked for permissions.
14	AID4	Controls permission check of ID = 4 0 = AID is not checked for permissions. 1 = AID is checked for permissions.
13	AID3	Controls permission check of ID = 3 0 = AID is not checked for permissions. 1 = AID is checked for permissions.
12	AID2	Controls permission check of ID = 2 0 = AID is not checked for permissions. 1 = AID is checked for permissions.
11	AID1	Controls permission check of ID = 1 0 = AID is not checked for permissions. 1 = AID is checked for permissions.
10	AID0	Controls permission check of ID = 0 0 = AID is not checked for permissions. 1 = AID is checked for permissions.
9	AIDX	Controls permission check of ID >15 0 = AID is not checked for permissions. 1 = AID is checked for permissions.
8	Reserved	Reserved. Always reads as 0.
7	NS	Non-secure access permission 0 = Only secure access allowed. 1 = Non-secure access allowed.
6	EMU	Emulation (debug) access permission. This bit is ignored if NS = 1 0 = Debug access not allowed. 1 = Debug access allowed.
5	SR	Supervisor Read permission 0 = Access not allowed. 1 = Access allowed.

Table 3-11 Programmable Range *n* Memory Protection Page Attribute Register (PROG*n*_MPPA) Field Descriptions (Part 3 of 3)

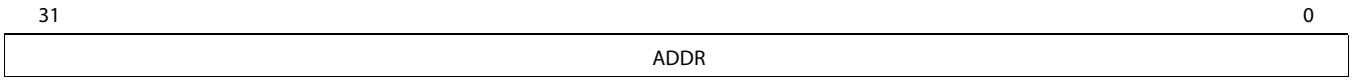
Bits	Name	Description
4	SW	Supervisor Write permission 0 = Access not allowed. 1 = Access allowed.
3	SX	Supervisor Execute permission 0 = Access not allowed. 1 = Access allowed.
2	UR	User Read permission 0 = Access not allowed. 1 = Access allowed
1	UW	User Write permission 0 = Access not allowed. 1 = Access allowed.
0	UX	User Execute permission 0 = Access not allowed. 1 = Access allowed.
End of Table 3-111		

3.4.7 MPU Fault Registers

3.4.7.1 Fault Address Register (FLTADDRR)

The fault address register holds the address of the first protection fault transfer.

Figure 3-11 Fault Address Register (FLTADDRR)



Legend: R = Read only; R/W = Read/Write; -*n* = value after reset

Table 3-12 Fault Address Register (FLTADDRR) Field Descriptions

Bits	Name	Range	Description
31 – 0	ADDR	0 – FFFF FFFFh	Memory address of fault

3.4.7.2 Fault Status Register (FLTSTAT)

The fault status register holds the status and attributes of the first protection fault transfer.

Note—Master IDs are used to determine allowed connections between masters and slaves. Unlike privilege IDs, which can be shared across different masters, master IDs are unique to each master. For more information on master IDs refer to the device-specific data manual.

Figure 3-12 Fault Status Register (FLTSTAT)

31 24	23 16	15 13	12 9	8	7	6	5 0
Reserved	MSTID	Reserved	PRIVID	Reserved	NS	Reserved	TYPE
R-x	R-0	R-0	R-0	R-0	R-0	R-0	R-0

Legend: R = Read only; -*n* = value after reset

Figure 3-13 Fault Status Register (FLTSTAT) Field Descriptions

Bits	Name	Range	Description
31 – 24	Reserved	0 – FFh	Reserved.
23 – 16	MSTID	0 – FFh	Master ID of fault transfer.
15 – 13	Reserved	0	Reserved. Always reads as 0.
12 – 9	PRIVID	0 – Fh	Privilege ID of fault transfer.
8	Reserved	0	Reserved. Always reads as 0.
7	NS	0	Security level of fault transfer. 0 = Secure access 1 = Non-secure access
6	Reserved		Reserved. Always reads as 0.
5 – 0	TYPE		Fault type.
		100000b	Supervisor read fault
		010000b	Supervisor write fault
		001000b	Supervisor execute fault
		000100b	User read fault
		000010b	User write fault
		000001b	User execute fault
		111111b	Relaxed cache line fill fault
		010010b	Relaxed cache write back fault
		000000b	No fault
End of Table 3-12			

3.4.7.3 Fault Clear Register (FLTCLR)

The fault clear register (FLTCLR) allows software to clear the current fault so that another can be captured in the fault status register as well as produce an interrupt. Only the TYPE field in the MPU fault status register is cleared when 1 is written to the CLEAR bit.

Figure 3-14 Fault Clear Register (FLTCLR)

31	1	0
Reserved		CLEAR
R-0		W-0

Legend: R = Read only; W = Write; -*n* = value after reset

Table 3-13 Fault Clear Register (FLTCLR) Field Descriptions

Bits	Name	Description
31 – 1	Reserved	Reserved. Always reads as 0.
0	CLEAR	Command to clear the current fault. Writing 0 has no effect. 0 = No effect 1 = Clear the current fault

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

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

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license 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 significant portions of TI 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. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

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

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

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