S12X_MMC
Memory Mapping Control
Block Guide
V02.04

Original Release Date: 05 Feb 2003
Revised: 01 Sept 2004

MCU Design Center Munich
Motorola, Inc.

Revision History

Version Number	Revision Date	Effective Date	Author	Description of Changes
v01.00	05 Feb. 2004	02/05/2004		Reformat the block guide to be SRS compliant
v02.00	19 May 2004	05/19/2004		S12X_MMC Block guide is meant for Bluefin2 architecture. - Major document rework. - Added bus arbitration scheme. - Changed DIRECT address from $0012 to $0011 - Moved MODE register ($000A) from S12X_EBI - Moved (EIFCTL->MEMCTL0) register ($000B) from S12X_EBI - Moved 'Modes of Operation' description from S12X_EBI. - Moved features 'chip selects' and 'Chip operating mode control' from S12X_EBI. - Moved write protection features from XSRAM.
v02.01	19 July 2004	07/19/2004		Reorganization of MEMCTL0 bits to allow integrating new features (from [7:5] to [2:0]).
v02.02	05 August 2004	08/05/2004		- Added CS3E in MEMCTL0 register (position 3) - Added third chip select (CS3), and redefined CS2
v02.03	13 August 2004	08/13/2004		- XGATE read access to a secured Flash in expanded modes is considered like a software error
v02.04	01 Sept 2004	09/01/2004		- Clarify in details unimplemented areas and External Spaces and chip selects - EROMON bit in register MMCCTL1 is (write never) instead of write once.

Section 1 Introduction

Section 2 External Signal Description

Section 3 Memory Map/Register Definition

Section 4 Functional Description

Section 5 Initialisation/Application Information

Preface

Terminology

Table 0-1 Acronyms and Abbreviations

Logic level “1”	Voltage that corresponds to Boolean true state
Logic level “0”	Voltage that corresponds to Boolean false state
$	Represents hexadecimal number
x	Represents logic level ’don’t care’
byte	8-bit data
word	16-bit data
local address	based on the 64 KBytes Memory Space (16-bit address)
globall address	based on the 8 MBytes Memory Space (23-bit address)
Bus Clock	System Clock. Refer to CRG Block Guide.
expanded modes	Normal Expanded Mode Emulation Single-Chip Mode Emulation Expanded Mode Special Test Mode
single-chip modes	Normal Single-Chip Mode Special Single-Chip Mode
emulation modes	Emulation Single-Chip Mode Emulation Expanded Mode
normal modes	Normal Single-Chip Mode Normal Expanded Mode
special modes	Special Single-Chip Mode Special Test Mode
NS	Normal Single-Chip Mode
SS	Special Single-Chip Mode
NX	Normal Expanded Mode
ES	Emulation Single-Chip Mode
EX	Emulation Expanded Mode
ST	Special Test Mode
Unimplemented areas	Areas which are accessible by the pages (RPAGE,PPAGE,EPAGE) and not implemented
External Space	Area which is accessible in the global address range 14_0000 to 3F_FFFF
external resource	Resources (Emulator, Application) connected to the MCU via the external bus on expanded modes (Unimplemented areas and External Space)
PRR	Port Replacement Registers
PRU	Port Replacement Unit located on the emulator side
MCU	MicroController Unit
NVM	Non-volatile Memory; Flash EEPROM or ROM

Section 1 Introduction

This section describes the functionality of the Module Mapping Control (S12X_MMC) sub-block of the S12X Platform. The block diagram of the S12X_MMC is shown in **Figure 1-1**.

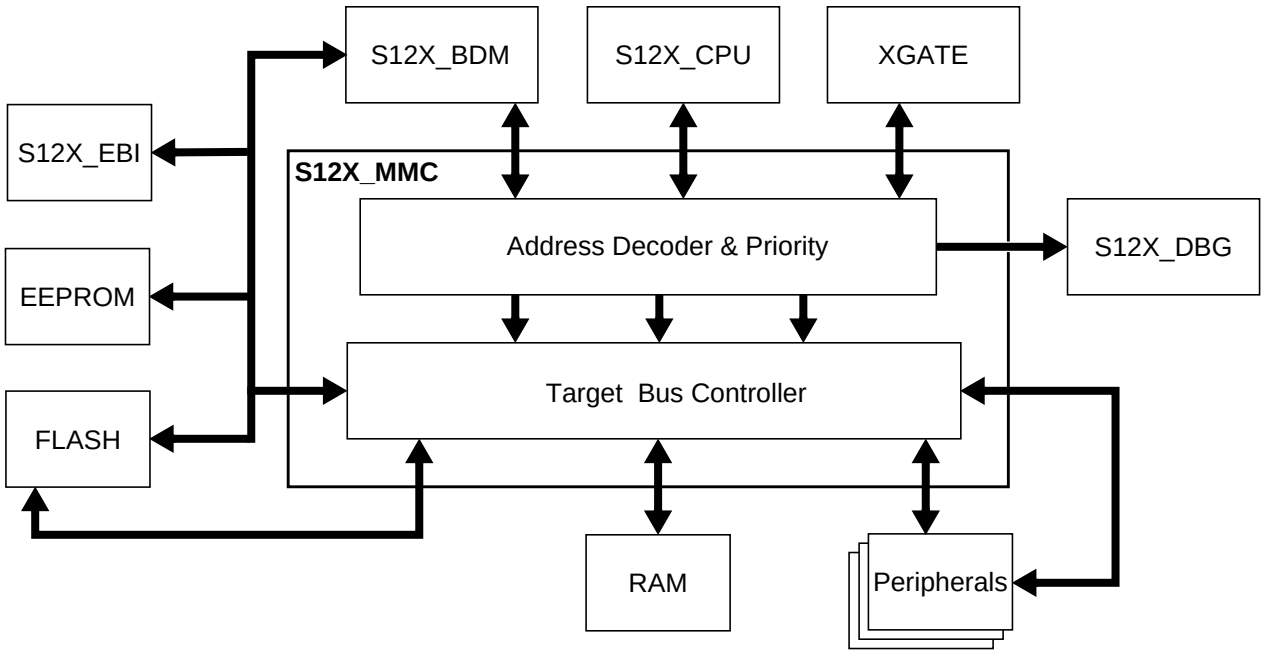


Figure 1-1 S12X_MMC Block Diagram

1.1 Overview

The S12X_MMC module controls the multi-master priority accesses, the selection of internal resources and external space. Internal buses including internal memories and peripherals are controlled in this module. The local address space for each master is translated to a global memory space.

1.2 Features

- Paging capability to support a global 8 MBytes memory address space.
- Bus arbitration between the masters S12X_CPU, S12X_BDM and XGATE.
- Simultaneous accesses to different resources[1] (internal, external and peripherals). (See **Figure 1-1**)
- Resolution of target bus access collision.

NOTES:
1. Resources are also called targets.

- Access restriction control from masters to some targets (e.g. RAM write access protection for user specified areas).
- MCU operation mode control.
- MCU security control.
- Separate memory map schemes for each master S12X_CPU, S12X_BDM and XGATE.
- ROM control bits to enable the on-chip FLASH or ROM selection.
- Port Replacement Registers access control.
- Generation of System Reset when S12X_CPU accesses an unimplemented address (i.e. an address which does not belong to any of the on-chip modules) in single-chip modes.

1.3 Modes of Operation

This subsection lists and briefly describes all operating modes supported by the S12X_MMC.

1.3.1 Power Saving Modes

- Run Mode

  S12X_MMC is functional during normal Run Mode.
- Wait Mode

  S12X_MMC is functional during Wait Mode.
- Stop Mode

  S12X_MMC is inactive during Stop mode.

1.3.2 Functional Modes

- Single Chip Modes

  In normal and special single chip mode the internal memory is used. External bus is not active.
- Expanded Modes

  Address, Data and Control signals are activated in normal expanded and special test modes when accessing the external bus.
- Emulation Modes

  External bus is active to emulate via an external tool the normal expanded or the normal single chip mode.

Section 2 External Signal Description

2.1 Overview

The user is advised to refer to the SoC Guide for port configuration and location of external bus signals. Some pins may not be bonded out in all implementations.

Table 2-1 and **Table 2-2** outline the pin names and functions. It also provides a brief description of their operation.

Table 2-1 External input signals associated to the S12X_MMC

Signal	I/O	Description	Availability
MODC	I	Mode input	Latched after $\overline{\text{RESET}}$ (active low)
MODB	I	Mode input	
MODA	I	Mode input	
EROMCTL	I	EROM Control input	
ROMCTL	I	ROM Control input	

Table 2-2 External output signals associated with the S12X_MMC

Signal	I/O	Description	Available in Modes					
			NS	SS	NX	ES	EX	ST
CS0	O	Chip select line 0	(see **Table 3-1**)					
CS1	O	Chip select line 1						
CS2	O	Chip select line 2						
CS3	O	Chip select line 3						

Section 3 Memory Map/Register Definition

A summary of the registers associated with the S12X_MMC block is shown in **Figure 3-1**. Detailed descriptions of the registers and bits are given in the subsections that follow.

Address	Name		Bit 7	6	5	4	3	2	1	Bit 0
$000A	MMCCTL0	Read	0	0	0	0	CS3E	CS2E	CS1E	CS0E
		Write								
$000B	MODE	Read	MODC	MODB	MODA	0	0	0	0	0
		Write								
$0010	GPAGE	Read	0	GP6	GP5	GP4	GP3	GP2	GP1	GP0
		Write								
$0011	DIRECT	Read	DP15	DP14	DP13	DP12	DP11	DP10	DP9	DP8
		Write								
$0012	reserved	Read	0	0	0	0	0	0	0	0
		Write								
$0013	MMCCTL1	Read	0	0	0	0	0	EROMON	ROMHM	ROMON
		Write								
$0014	reserved	Read	0	0	0	0	0	0	0	0
		Write								
$0015	reserved	Read	0	0	0	0	0	0	0	0
		Write								
$0016	RPAGE	Read	RP7	RP6	RP5	RP4	RP3	RP2	RP1	RP0
		Write								
$0017	EPAGE	Read	EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0
		Write								
$0030	PPAGE	Read	PIX7	PIX6	PIX5	PIX4	PIX3	PIX2	PIX1	PIX0
		Write								
$0031	reserved	Read	0	0	0	0	0	0	0	0
		Write								
$011C	RAMWPC	Read	RPWE	0	0	0	0	0	AVIE	AVIF
		Write								
$011D	RAMXGU	Read	1	XGU6	XGU5	XGU4	XGU3	XGU2	XGU1	XGU0
		Write								

Address	Name		Bit 7	6	5	4	3	2	1	Bit 0
$011E	RAMSHL	Read	1	SHL6	SHL5	SHL4	SHL3	SHL2	SHL1	SHL0
		Write								
$011F	RAMSHU	Read	1	SHU6	SHU5	SHU4	SHU3	SHU2	SHU1	SHU0
		Write								

= Unimplemented or Reserved

Figure 3-1 S12X_MMC Register Map Summary

3.1 Register Descriptions

3.1.1 S12X_MMC Control Register (MMCCTL0)

Address: $000A PRR

	BIT 7	6	5	4	3	2	1	BIT 0
Read:	0	0	0	0	CS3E	CS2E	CS1E	CS0E
Write:								
Reset:	0	0		0	0	0	0	$\overline{\text{ROMON}}$[1]

= Unimplemented or Reserved

Figure 3-2 S12X_MMC Control Register (MMCCTL0)

NOTES:
1. ROMON is the bit[0] of the register MMCCTL1 (see **Figure 3-9**).

Read: Anytime. In emulation modes read operations will return the data from the external bus. In all other modes the data is read from this register.

Write: Anytime. In emulation modes write operations will also be directed to the external bus.

Table 3-1 Chip Selects function activity

Register Bit	Chip Modes					
	NS	SS	NX	ES	EX	ST
CS3E, CS2E, CS1E, CS0E	disabled[1]	disabled	enabled[2]	disabled	enabled	enabled

NOTES:
1. disabled: feature always inactive.
2. enabled: activity is controlled by the appropriate register bit value.

The MMCCTL0 register is used to control external bus functions, i.e. availability of chip selects.

Disclaimer : XGATE write access to this register during an S12X_CPU access which makes use of this register could lead to unexpected results.

CS3E, CS2E, CS1E, CS0E — Chip Select Enables 2, 1, 0

Each of these bits enables one of the external chip selects $\overline{CS3}$, $\overline{CS2}$, $\overline{CS1}$ and $\overline{CS0}$ outputs which are asserted during accesses to specific external addresses. The associated global address ranges are shown in **Table 3-2** and **Figure 4-3**

Table 3-2 Chip select signals

Global address range	Asserted signal
$00_0800 - $0F_FFFF	$\overline{CS3}$
$10_0000 - $1F_FFFF	$\overline{CS2}$
$20_0000 - $3F_FFFF	$\overline{CS1}$
$40_0000 - $7F_FFFF	$\overline{CS0}$(1)

NOTES:
1. When the internal NVM is enabled (see ROMON in **3.1.5**) the $\overline{CS0}$ is not asserted in the space occupied by this on-chip memory block.

Chip selects are only active if enabled in Normal Expanded Mode, Emulation Expanded Mode and Special Test Mode. The function disabled in all other operating modes.

1 = Chip Select is enabled
0 = Chip Select is disabled

3.1.2 MODE Register (MODE)

Address: $000B PRR

	BIT 7	6	5	4	3	2	1	BIT 0
Read:	MODC	MODB	MODA	0	0	0	0	0
Write:								
Reset:	MODC(1)	MODB1	MODA1	0	0	0	0	0

= Unimplemented or Reserved

Figure 3-3 MODE Register (MODE)

NOTES:
1. External signal (see **Table 2-1**).

Read: Anytime. In emulation modes read operations will return the data read from the external bus. In all other modes the data are read from this register.

Write: Only if a transition is allowed (see **Figure 3-4**). In emulation modes write operations will be also directed to the external bus.

The MODE bits of the MODE register are used to establish the MCU operating mode.

Disclaimer : XGATE write access to this register during an S12X_CPU access which makes use of this register could lead to unexpected results.

MODC, MODB, and MODA — Mode Select Bits

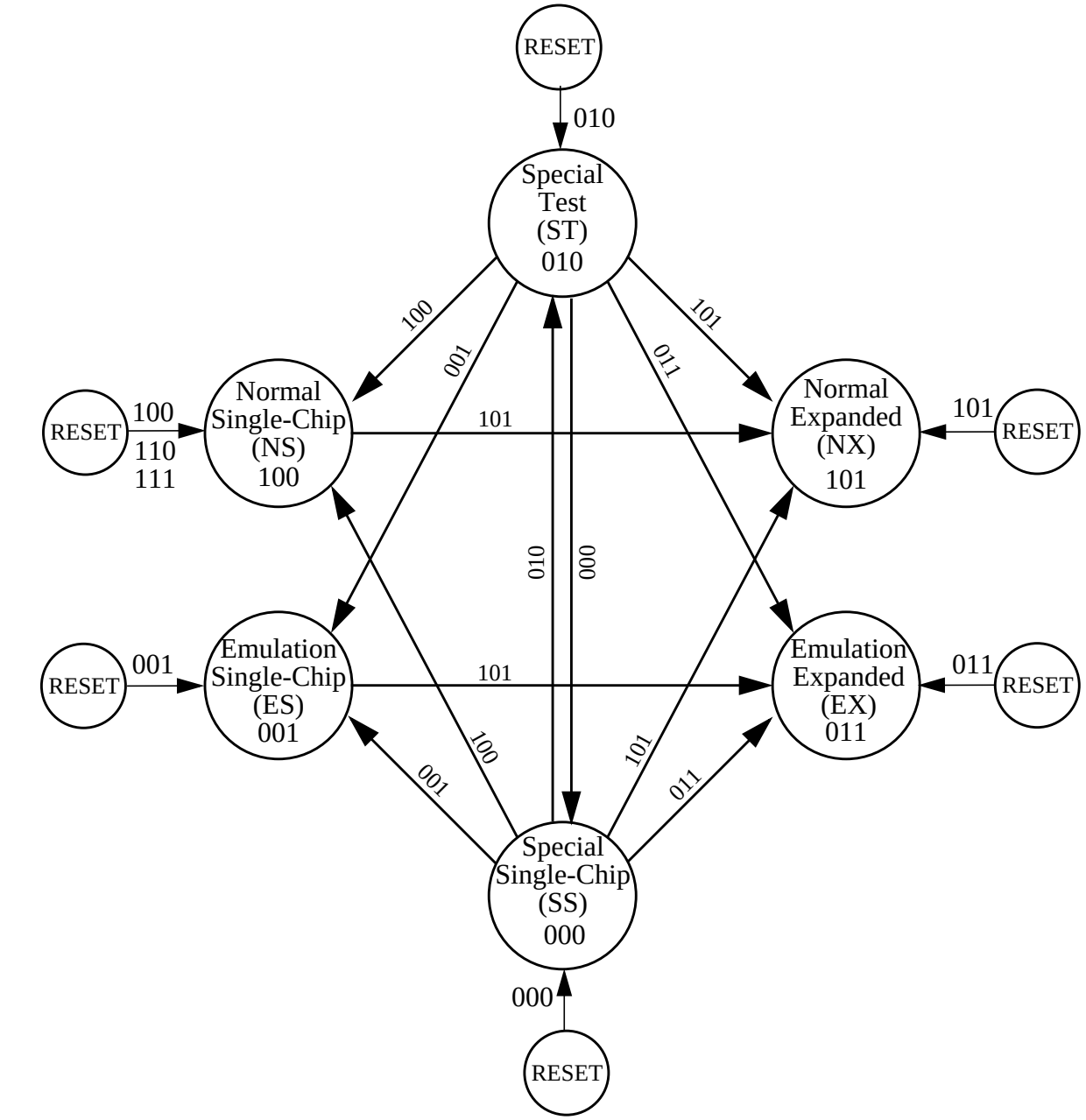
These bits control the current operating mode during $\overline{\text{RESET}}$ high (inactive). The external mode pins MODC, MODB, and MODA determine the operating mode during $\overline{\text{RESET}}$ low (active). The state of the pins is latched into the respective register bits after the $\overline{\text{RESET}}$ signal goes inactive (see **Figure 3-4**).

Write restrictions exist to disallow transitions between certain modes. **Figure 3-4** illustrates all allowed mode changes. Attempting non authorized transitions will not change the MODE bits, but it will block further writes to these register bits except in special modes.

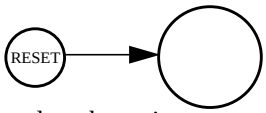
Both transitions from Normal Single-Chip Mode to Normal Expanded Mode and from Emulation Single-Chip to Emulation Expanded Mode are only executed by writing a value of 0b101 (write once). Writing any other value will not change the MODE bits, but will block further writes to these register bits.

Changes of operating modes are not allowed when the device is secured, but it will block further writes to these register bits except in special modes.

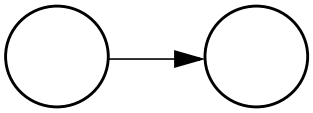
In emulation modes reading this address returns data from the external bus which has to be driven by the emulator. It is therefore responsibility of the emulator hardware to provide the expected value (i.e. a value corresponding to Normal Single Chip mode while the device is in Emulation Single-Chip Mode or a value corresponding to Normal Expanded mode while the device is in Emulation Expanded Mode).



Transition done by EXTERNAL PINS (MODC,MODB,MODA)



Transition done by write access to the MODE register



110
111
Illegal (MODC,MODB,MODA) PIN values.

Do not use. (Reserved for future use).

Figure 3-4 Mode Transition Diagram when MCU is unsecured

3.1.3 Global Page Index Register (GPAGE)

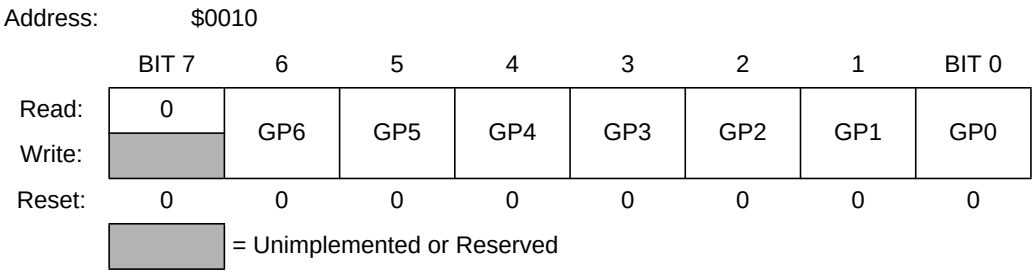


Figure 3-5 Global Page Index Register (GPAGE)

Read: Anytime
Write: Anytime

The Global Page Index Register is used only when the S12X_CPU is executing a global instruction (GLDAA, GLDAB, GLDD, GLDS, GLDX, GLDY,GSTAA, GSTAB, GSTD, GSTS, GSTX, GSTY) (see S12X_CPU Block Guide). The generated global address is the result of concatenation of the S12X_CPU local address [15:0] with the GPAGE register [22:16] (see **Figure 3-6**).

Disclaimer : XGATE write access to this register during an S12X_CPU access which makes use of this register could lead to unexpected results..

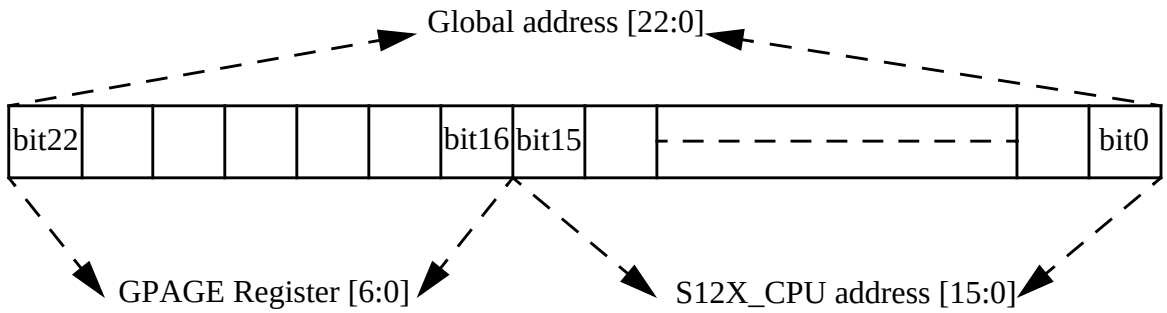


Figure 3-6 GPAGE Address Mapping

GP6–GP0 — Global Page Index Bits 6–0

These page index bits are used to select which of the 128 64-kilobyte pages is to be accessed.

Example: This example demonstrates usage of the GPAGE register:

```
LDAADR EQU $5000              ;Initialize LDADDR to the value of $5000
MOVB     #$14, GPAGE          ;Initialize GPAGE register with the value of $14
GLDAA    >LDAADR              ;Load Accu A from the global address $14_5000
```

3.1.4 Direct Page Register (DIRECT)

Address: $0011

	BIT 7	6	5	4	3	2	1	BIT 0
Read: Write:	DP15	DP14	DP13	DP12	DP11	DP10	DP9	DP8
Reset:	0	0	0	0	0	0	0	0

Figure 3-7 Direct Register (DIRECT)

Read: Anytime

Write: anytime in special modes, one time only in other modes.

This register determines the position of the direct page within the memory map.

DP15–DP8 — Direct Page Index Bits 15–8

These bits are used by the S12X_CPU when performing accesses using the direct addressing mode. The bits from this register form bits [15:8] of the address (see **Figure 3-8**).

Disclaimer : XGATE write access to this register during an S12X_CPU access which makes use of this register could lead to unexpected results.

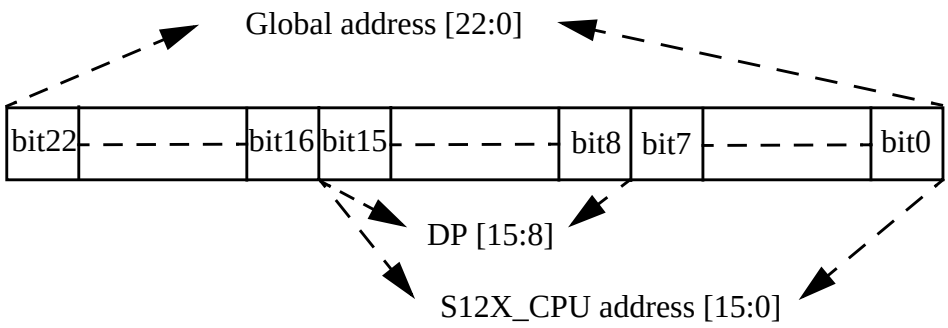


Figure 3-8 DIRECT Address Mapping

Bits [22:16] of the global address will be formed by the GPAGE[6:0] bits in case the CPU executes a global instruction in Direct Addressing Mode or by the appropriate local address to the global address expansion (refer to **4.2.1.1.1**).

Example: This example demonstrates usage of the Direct Addressing Mode by a global instruction:

```
LDAADR    EQU $0000          ;Initialize LDADDR with the value of $0000
MOVB      #$80,DIRECT        ;Initialize DIRECT register with the value of $80
MOVB      #$14,GPAGE         ;Initialize GPAGE register with the value of $14
GLDAA     <LDAADR            ;Load Accu A from the global address $14_8000
```

3.1.5 S12X_MMC Control Register (MMCCTL1)

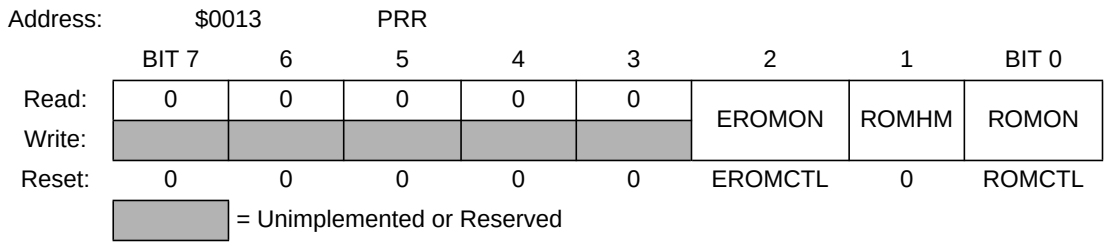


Figure 3-9 S12X_MMC Control Register (MMCCTL1)

Read: Anytime. In emulation modes read operations will return the data from the external bus. In all other modes the data are read from this register.
Write: Refer to each bit description. In emulation modes write operations will also be directed to the external bus.

Disclaimer : XGATE write access to this register during an S12X_CPU access which makes use of this register could lead to unexpected results.

EROMON — Enables emulated FLASH or ROM memory in the memory map.

Write: never.

1 = Enables the emulated FLASH or ROM in the memory map.
0 = Disables the emulated FLASH or ROM in the memory map.

ROMHM — FLASH or ROM only in higher Half of Memory Map

Write: Once in Normal and Emulation modes and anytime in Special modes.

1 = Disables access to the FLASH or ROM in the lower half of the memory map.These physical locations of the FLASH or ROM can still be accessed through the Program Page window. Accesses to $4000-$7FFF will be mapped to 14_4000-14_7FFF in the global memory space (external access).
0 = The fixed page of FLASH or ROM can be accessed in the lower half of the memory map. Accesses to $4000-$7FFF will be mapped to $7F_4000-$7F_7FFF in the global memory space.

ROMON — Enable FLASH or ROM in the memory map

Write: Once in Normal and Emulation modes and anytime in Special modes.

1 = Enables the FLASH or ROM in the memory map.
0 = Disables the FLASH or ROM from the memory map.

EROMON and ROMON control the visibility of the Flash in the memory map for S12X_CPU or S12X_BDM (not for XGATE). Both local and global memory maps are affected.

Table 3-3 Data sources when S12X_CPU or S12X_BDM is accessing Flash area

Chip MODES	ROMON	EROMON	DATA SOURCE(1)	Stretch(2)
Normal Single Chip	X	X	Internal	N
Special Single Chip				
Emulation Single Chip	X	0	Emulation Memory	N
	X	1	Internal Flash	
Normal Expanded	0	X	External Application	Y
	1	X	Internal Flash	N
Emulation Expanded	0	X	External Application	Y
	1	0	Emulation Memory	N
	1	1	Internal Flash	
Special Test	0	X	External Application	N
	1	X	Internal Flash	

NOTES:

1. Internal means resources inside the MCU are read/written.
   Internal Flash means Flash resources inside the MCU are read/written.
   Emulation Memory means resources inside the emulator are read/written (PRU registers, flash replacement, RAM, EEPROM and Register Space are always considered internal).
   External Application means resources residing outside the MCU are read/written.
2. The external access stretch mechanism is part of the EBI module (refer to S12X_EBI Block Guide for details).

3.1.6 RAM Page Index Register (RPAGE)

Address: $0016

	BIT 7	6	5	4	3	2	1	BIT 0
Read: Write:	RP7	RP6	RP5	RP4	RP3	RP2	RP1	RP0
Reset:	1	1	1	1	1	1	0	1

Figure 3-10 RAM Page Index Register (RPAGE)

Read: Anytime
Write: Anytime

The RAM Page Index Register allows accessing up to (1M minus 2K) bytes of RAM in the global memory map by using the eight page index bits to page 4K byte blocks into the RAM Page Window located in the S12X_CPU local memory map from address $1000 to address $1FFF (see **Figure 3-11**).

Disclaimer : XGATE write access to this register during an S12X_CPU access which makes use of this register could lead to unexpected results.

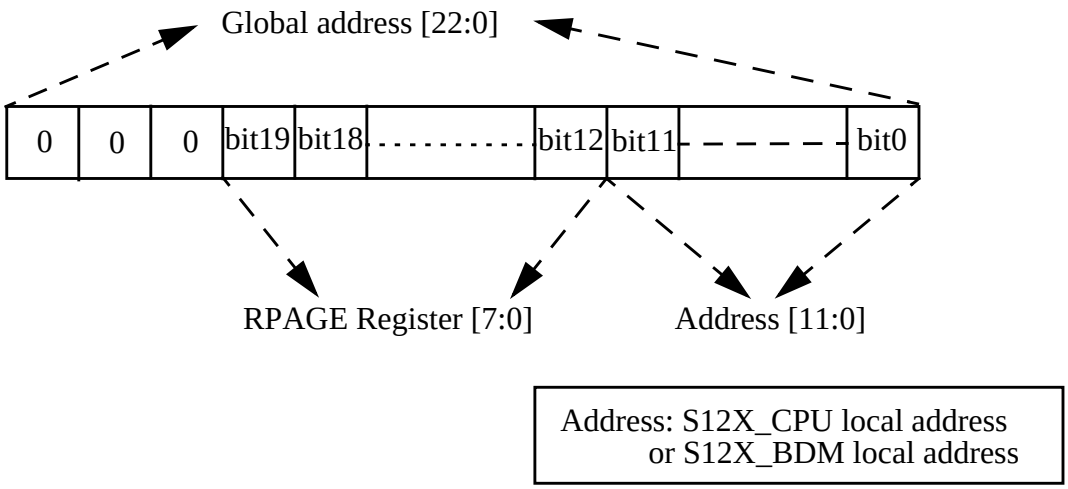


Figure 3-11 RPAGE Address Mapping

NOTE: *Because RAM page 0 has the same global address as the register space, it is possible to write to registers through the RAM space when RPAGE = $00.*

RP7–RP0 — RAM Page Index Bits 7–0

These page index bits are used to select which of the 256 RAM array pages is to be accessed in the RAM Page Window.

The reset value of $FD ensures that there is a linear RAM space available between addresses $1000 and $3FFF out of reset.

The fixed 4K page from $2000 - $2FFF of RAM is equivalent to page 254 (page number $FE).

The fixed 4K page from $3000 - $3FFF of RAM is equivalent to page 255 (page number $FF).

3.1.7 EEPROM Page Index Register (EPAGE)

Figure 3-12 EEPROM Page Index Register (EPAGE)

Address: $0017

	BIT 7	6	5	4	3	2	1	BIT 0
Read: Write:	EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0
Reset:	1	1	1	1	1	1	1	0

Read: Anytime
Write: Anytime

The EEPROM Page Index Register allows accessing up to 256K Byte of EEPROM in the global memory map by using the eight page index bits to page 1K byte blocks into the EEPROM Page Window located in the local S12X_CPU memory map from address $0800 to address $0BFF (see **Figure 3-13**).

Disclaimer : XGATE write access to this register during an S12X_CPU access which makes use of this register could lead to unexpected results.

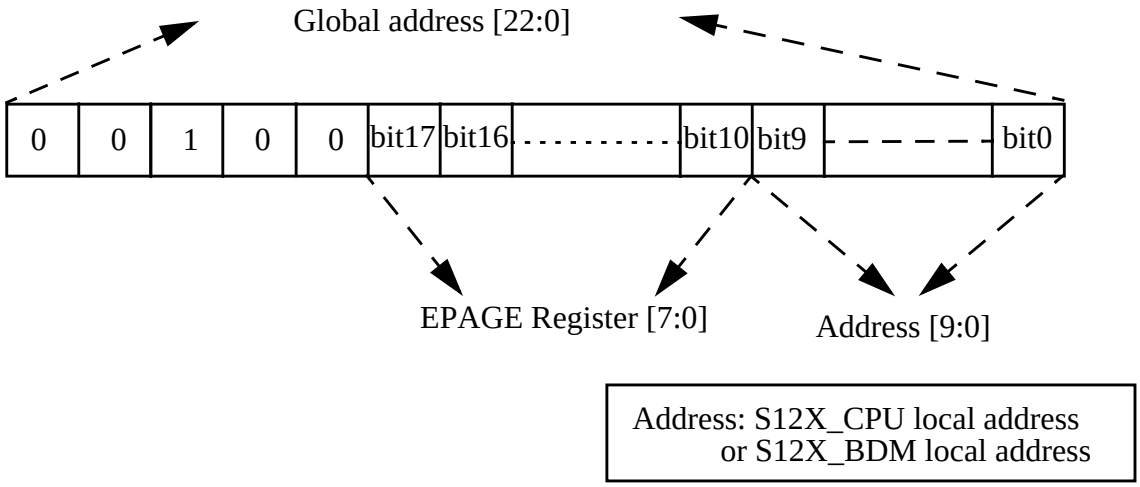


Figure 3-13 EPAGE Address Mapping

EP7–EP0 — EEPROM Page Index Bits 7–0

These page index bits are used to select which of the 256 EEPROM array pages is to be accessed in the EEPROM Page Window.

The reset value of $FE ensures that there is a linear EEPROM space available between addresses $0800 and $0FFF out of reset.

The fixed 1K page $0C00 - $0FFF of EEPROM is equivalent to page 255 (page number $FF).

3.1.8 Program Page Index Register (PPAGE)

Address: $0030

	BIT 7	6	5	4	3	2	1	BIT 0
Read: Write:	PIX7	PIX6	PIX5	PIX4	PIX3	PIX2	PIX1	PIX0
Reset:	1	1	1	1	1	1	1	0

Figure 3-14 Program Page Index Register (PPAGE)

Read: Anytime
Write: Anytime

The Program Page Index Register allows accessing up to 4M byte of FLASH or ROM in the global memory map by using the eight page index bits to page 16K byte blocks into the Program Page Window located in the S12X_CPU local memory map from address $8000 to address $BFFF (see **Figure 3-15**). The S12X_CPU has a special access to read and write this register during execution of CALL and RTC instructions.

Disclaimer : XGATE write access to this register during an S12X_CPU access which makes use of this register could lead to unexpected results.

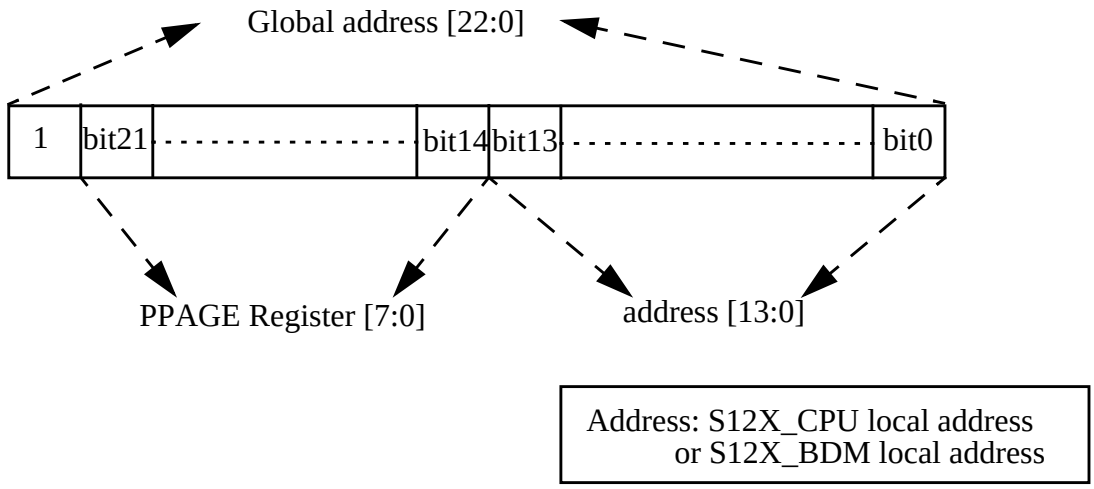


Figure 3-15 PPAGE Address Mapping

NOTE: *Writes to this register using the special access of the CALL and RTC instructions will be complete before the end of the instruction execution.*

PIX7–PIX0 — Program Page Index Bits 7–0

These page index bits are used to select which of the 256 FLASH or ROM array pages is to be accessed in the Program Page Window.

The fixed 16K page from $4000-$7FFF (when ROMHM = 0) is the page number $FD.

The reset value of $FE ensures that there is linear Flash space available between addresses $4000 and $FFFF out of reset.

The fixed 16K page from $C000-$FFFF is the page number $FF.

3.1.9 RAM Write Protection Control Register (RAMWPC)

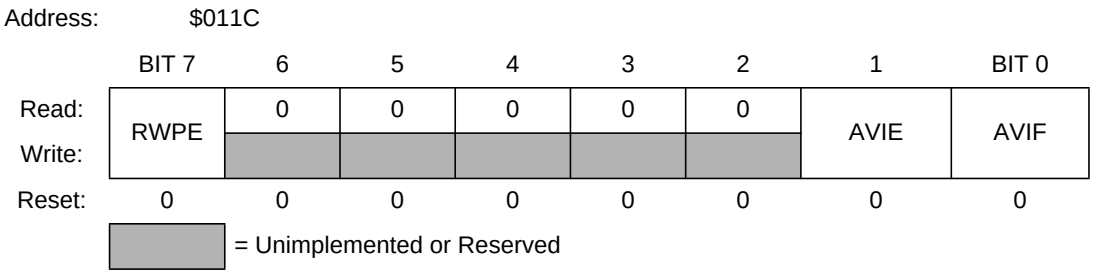


Figure 3-16 RAM Write Protection Control Register (RAMWPC)

Read: Anytime
Write: Anytime

RWPE — RAM Write Protection Enable

This bit enables the RAM write protection mechanism. When the RWPE bit is cleared, there is no write protection and any memory location is writable by the S12X_CPU module and the XGATE module. When the RWPE bit is set the write protection mechanism is enabled and write access of the S12X_CPU or to the XGATE RAM region. Write access performed by the XGATE module to outside of the XGATE RAM region or the shared region is suppressed as well in this case.

1 = RAM write protection check is enabled, region boundary registers cannot be written.
0 = RAM write protection check is disabled, region boundary registers can be written.

AVIE — S12X_CPU Access Violation Interrupt Enable

This bit enables the Access Violation Interrupt. If AVIE is set and AVIF is set, an interrupt is generated.

1 = S12X_CPU Access Violation Interrupt Enabled.
0 = S12X_CPU Access Violation Interrupt Disabled.

AVIF — S12X_CPU Access Violation Interrupt Flag

When set, this bit indicates that the S12X_CPU has tried to write a memory location inside the XGATE RAM region. This flag can be reset by writing ’1’ to the AVIF bit location.

1 = Access violation by the S12X_CPU was detected.
0 = No access violation by the S12X_CPU was detected.

3.1.10 RAM XGATE Upper Boundary Register (RAMXGU)

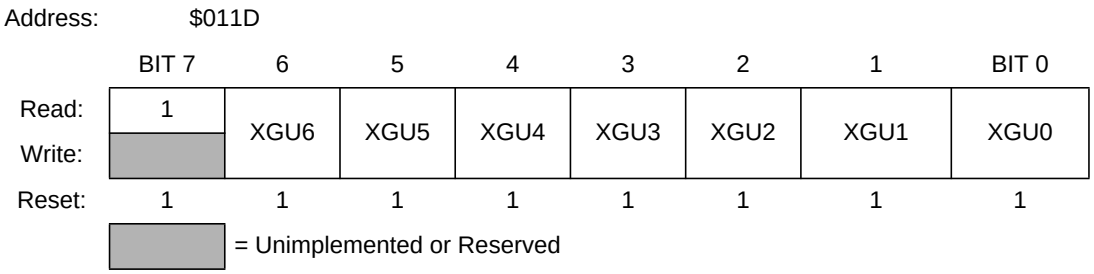


Figure 3-17 RAM XGATE Upper Boundary Register (RAMXGU)

Read:anytime

Write:anytime when RWPE=0

XGU[6:0] — XGATE Region Upper Boundary Bits 6-0

These bits define the upper boundary of the RAM region allocated to the XGATE module in multiples of 256 bytes. The 256 byte block selected by this register is included in the region. See **Figure 4-5** for details.

3.1.11 RAM Shared Region Lower Boundary Register (RAMSHL)

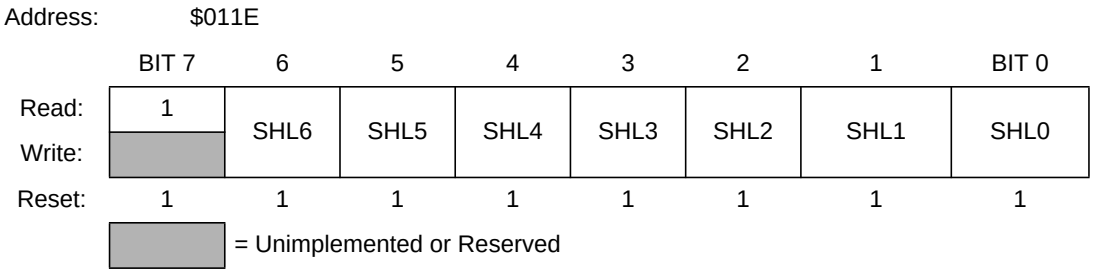


Figure 3-18 RAM Shared Region Lower Boundary Register (RAMSHL)

Read:anytime

Write:anytime when RWPE=0

SHL[6:0] — RAM Shared Region Lower Boundary Bits 6-0

These bits define the lower boundary of the shared memory region in multiples of 256 bytes. The block selected by this register is included in the region. See **Figure 4-5** for details.

3.1.12 RAM Shared Region Upper Boundary Register (RAMSHU)

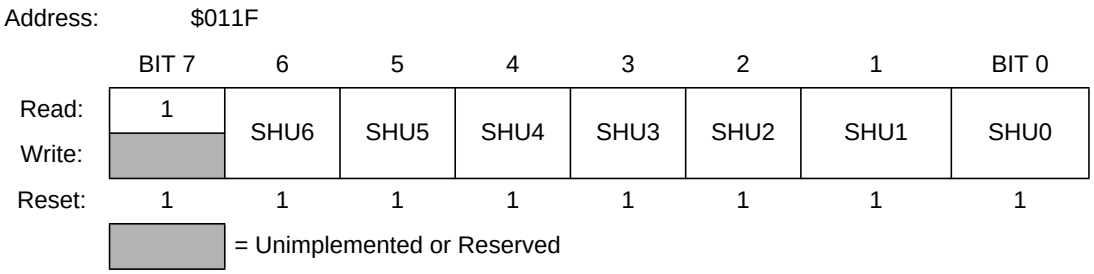


Figure 3-19 RAM Shared Region Upper Boundary Register (RAMSHU)

Read:anytime

Write:anytime when RWPE=0

SHU[6:0] — RAM Shared Region Upper Boundary Bits 6-0

These bits define the upper boundary of the shared memory in multiples of 256 bytes. The block selected by this register is included in the region. See **Figure 4-5** for details.

Section 4 Functional Description

The S12X_MMC block performs several basic functions of the S12X sub-system operation: MCU operation modes, priority control, address mapping, select signal generation and access limitations for the system. Each aspect is described in the following subsections.

4.1 MCU Operating Mode

- Normal Single-Chip Mode

  There is no external bus in this mode. The MCU program is executed from the internal memory and no external accesses are allowed.

- Special Single-Chip Mode

  This mode is generally used for debugging single-chip operation, boot-strapping or security related operations. The active Background Debug Mode is in control of the S12X_CPU code execution and the S12X_BDM firmware is waiting for serial commands sent through the BKGD pin. There is no external bus in this mode.

- Emulation Single-Chip Mode

  Tool vendors use this mode for emulation systems in which the user's target application is Normal Single-Chip Mode. Code is executed from external or internal memory depending on the set-up of the EROMON bit (see section **3.1.5**). The external bus is active in both cases to allow observation of internal operations (Internal Visibility).

- Normal Expanded Mode

  The external bus interface is configured as an up to 23-bit address bus, 8 or 16-bit data bus with dedicated bus control and status signals. This mode allows 8 or 16-bit external memory and peripheral devices to be interfaced to the system. The fastest external bus rate is half of the internal bus rate. An external signal can be used in this mode to cause the external bus to wait as desired by the external logic.

- Emulation Expanded Mode

  Tool vendors use this mode for emulation systems in which the user's target application is Normal Expanded Mode.

- Special Test Mode

  This mode is an expanded mode for factory test.

4.2 Memory Map Scheme

4.2.1 S12X_CPU and S12X_BDM Memory Map Scheme

The S12X_BDM firmware lookup tables and S12X_BDM register memory locations share addresses with other modules, however they are not visible in the memory map during user's code execution. The

S12X_BDM memory resources are enabled only during the READ_BD and WRITE_BD access cycles to distinguish between accesses to the BDM memory area and accesses to the other modules. (Refer to S12X_BDM Block Guide for further details).

When MCU enters active BDM mode the S12X_BDM firmware lookup tables and the S12X_BDM registers become visible in the local memory map between addresses $FF00 and $FFFF and the S12X_CPU begins execution of firmware commands or the S12X_BDM begins execution of hardware commands. The resources which share memory space with the S12X_BDM module will not be visible in the memory map during active BDM mode.

Please note that after the MCU enters active BDM mode the S12X_BDM firmware lookup tables and the S12X_BDM registers will also be visible between addresses $BF00 and $BFFF if the PPAGE register contains value of $FF.

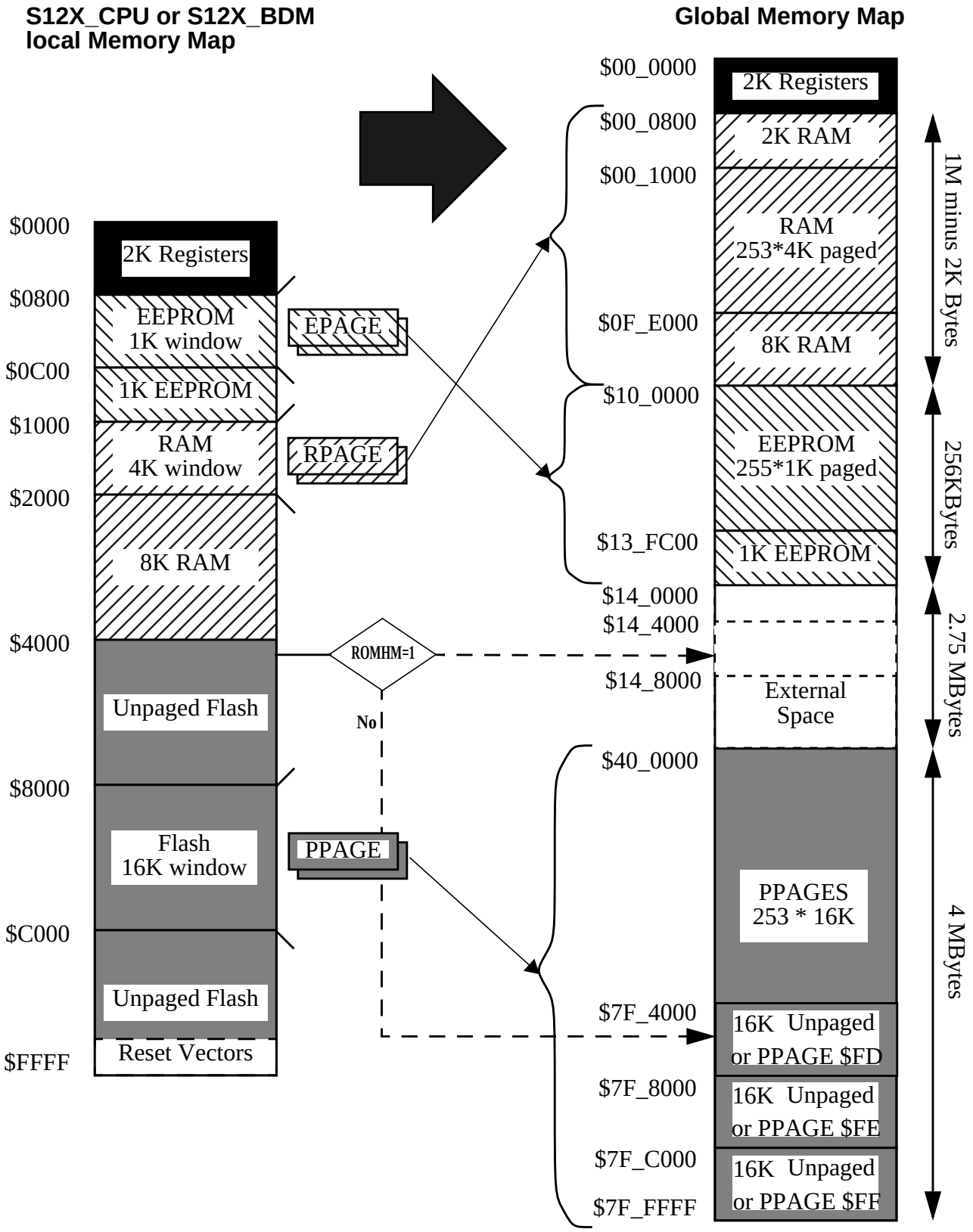


Figure 4-1 Expansion of the local address map

4.2.1.1 Expansion of the local address map

4.2.1.1.1 Expansion of the S12X_CPU local address map

The Program Page Index Register in S12X_MMC allows accessing up to 4M byte of FLASH or ROM in the global memory map by using the eight page index bits to page 256 16KByte blocks into the Program Page Window located from address $8000 to address $BFFF in the local S12X_CPU memory map.

The page value for the Program Page Window is stored in the PPAGE register. The value of the PPAGE register can be read or written by normal memory accesses as well as by the CALL and RTC instructions (see section **5.1**).

Control registers, vector space and parts of the on-chip memories are located in unpaged portions of the 64-kilobyte local S12X_CPU address space.

The starting address of an interrupt service routine must be located in unpaged memory unless the user is certain that the PPAGE register will be set to the appropriate value when the service routine is called. However an interrupt service routine can call other routines that are in paged memory. The upper 16-kilobyte block of the local S12X_CPU memory space ($C000–$FFFF) is unpaged. It is recommended that all reset and interrupt vectors point to locations in this area or to the other upages sections of the local S12X_CPU memory map.

Table 4-1 summarizes mapping of the address bus in Flash/External space based on the address, the PPAGE register value and value of the ROMHM bit in the MMCCTL1 register.

Table 4-1 Global FLASH/ROM Allocated

local S12X_CPU address	ROMHM	External access	Global Address
$4000–$7FFF	0	No	$7F_4000 - 7F_7FFF
	1	Yes	$14_4000 - 14_7FFF
$8000–$BFFF	N/A	No(1)	$40_0000 - $7F_FFFF
	N/A	Yes[1]	
$C000–$FFFF	N/A	No	$7F_C000 - $7F_FFFF

NOTES:

1. The internal or the external bus is accessed based on the the size of the memory resources implemented on-chip. Please refer to **Figure 4-3** for further details.

The RAM Page Index Register allows accessing up to 1M-2K bytes of RAM in the global memory map by using the eight RPAGE index bits to page 4K byte blocks into the RAM Page Window located in the local S12X_CPU memory space from address $1000 to address $1FFF. The EEPROM Page Index Register EPAGE allows accessing up to 256KBytes of EEPROM in the system by using the eight EPAGE index bits to page 1Kbyte blocks into the EEPROM Page Window located in the local S12X_CPU memory space from address $0800 to address $0BFF.

4.2.1.1.2 Expansion of the S12X_BDM local address map

PPAGE, RPAGE and EPAGE registers are also used for the expansion of the S12X_BDM local address to the global address. These registers can be read and written by the S12X_BDM.

The S12X_BDM expansion scheme is the same as the S12X_CPU expansion scheme.

4.2.1.2 Global addresses based on the Global page

4.2.1.2.1 S12X_CPU Global addresses based on the Global page

The seven Global Page index bits allow access to the full 8M byte address map that can be accessed with 23 address bits. This provides an alternative way to access all of the various pages of FLASH, RAM and EEPROM as well as additional external memory.

The GPAGE Register is used only when the S12X_CPU is executing a global instruction (see section **3.1.3**). The generated global address is the result of concatenation of the S12X_CPU local address [15:0] with the GPAGE register [22:16] (see **Figure 3-6**).

4.2.1.2.2 S12X_BDM Global addresses based on the Global page

The seven BDMGPR Global Page index bits allow access to the full 8M byte address map that can be accessed with 23 address bits. This provides an alternative way to access all of the various pages of FLASH, RAM and EEPROM as well as additional external memory.

The S12X_BDM Global Page Index Register (BDMGPR) is used only in the case the S12X_CPU is executing a firmware command which uses a global instruction (like GLDD, GSTD) or by a S12X_BDM hardware command (like WRITE_W, WRITE_BYTE, READ_W, READ_BYTE). See the S12X_BDM Block Guide for further details.

The generated global address is a result of concatenation of the S12X_BDM local address with the BDMGPR register [22:16] in the case of a hardware command or concatenation of the S12X_CPU local address and the BDMGPR register [22:16] in the case of a firmware command (see **Figure 4-2**).

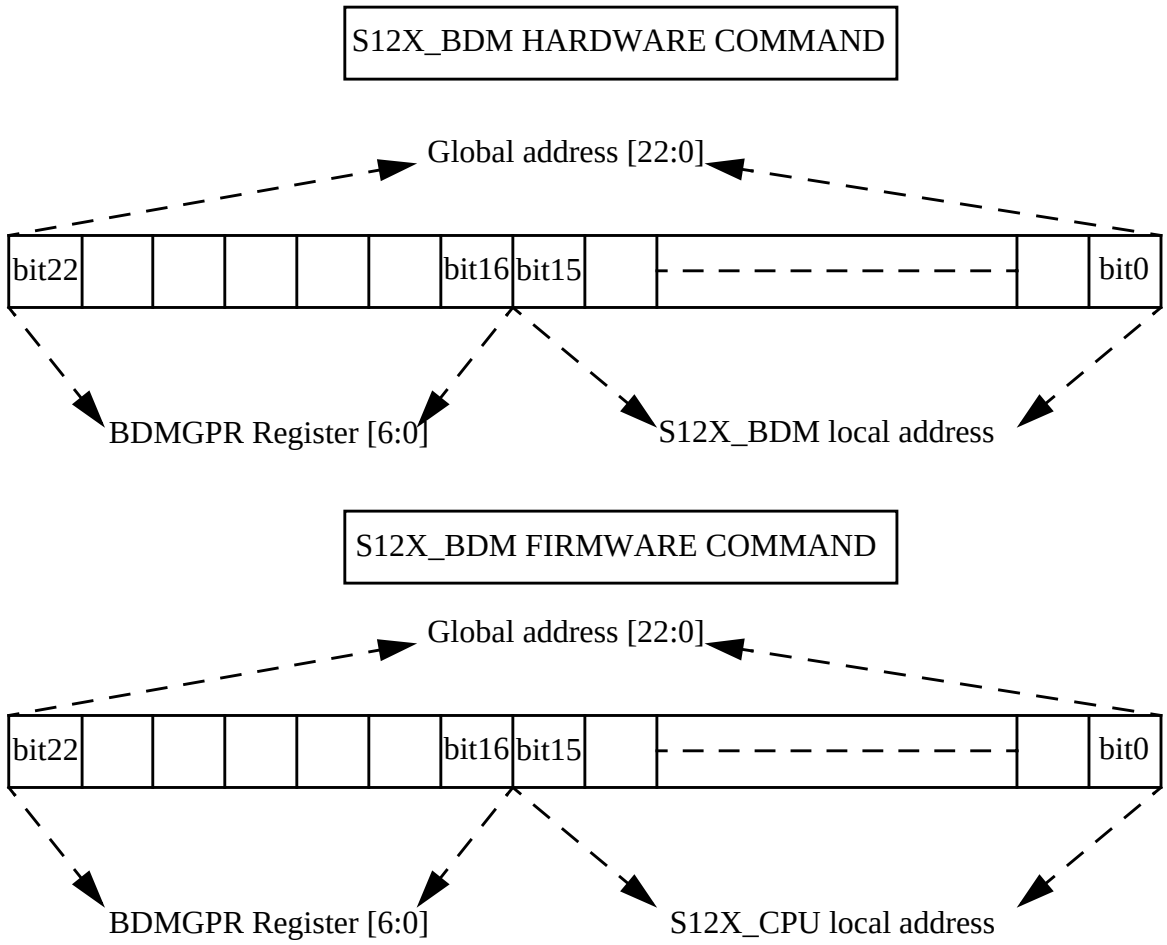


Figure 4-2 BDMGPR Address Mapping

4.2.2 Implemented Memory Map

The global memory spaces reserved for the internal resources (RAM, EEPROM and FLASH) are not determined by the S12X_MMC module. Size of the individual internal resources are however fixed in the design of the device cannot be changed by the user. Please refer to the Device User Guide for further details. **Figure 4-3** and **Table 4-2** show the memory spaces occupied by the on-chip resources. Please note that the memory spaces have fixed top addresses.

Table 4-2 Global Implemented memory space

Internal resource	Bottom address	Top address
Registers	$00_0000	$00_07FF
RAM	$10_0000 minus RAMSIZE(1)	$0F_FFFF
EEPROM	$14_0000 minus EEPROMSIZE(2)	$13_FFFF
FLASH	$80_0000 minus FLASHSIZE(3)	$7F_FFFF

NOTES:
1. RAMSIZE is the hexadecimal value of RAM SIZE in Bytes
2. EEPROMSIZE is the hexadecimal value of EEPROM SIZE in Bytes
3. FLASHSIZE is the hexadecimal value of FLASH SIZE in Bytes

When the device is operating in expanded modes except emulation single-chip mode, accesses to the global addresses which are not occupied by the on-chip resources (unimplemented areas or External Space) result in accesses to the external bus (see **Figure 4-3**).

In emulation single-chip mode, accesses to the global addresses which are not occupied by the on-chip resources (unimplemented areas) result in accesses to the external bus. S12X_CPU accesses to the global addresses which are occupied by the External Space result in an illegal access reset (system reset). The S12X_BDM accesses to the External Space are performed but the data is undefined.

In single-chip modes an access to any of the unimplemented areas (see **Figure 4-3**) by the S12X_CPU (except firmware commands) results in an illegal access reset (system reset). The S12X_BDM accesses to the unimplemented areas are performed but the data is undefined.

Misaligned word accesses to the last location (Top address) of any of the on-chip resource blocks (except RAM) by the S12X_CPU is performed in expanded modes. In single-chip modes these accesses (except Flash) result in an illegal access reset (except firmware commands).

Misaligned word accesses to the last location (top address) of the on-chip RAM by the S12X_CPU is ignored in expanded modes (read of undefined data). In single-chip modes these accesses result in an illegal access reset (except firmware commands).

No misaligned word access from the S12X_BDM module will occur. These accesses are blocked in the S12x_BDM (Refer to S12X_BDM Block Guide).

Misaligned word accesses to the last location of any global page (64 KByte) by using global instructions, is performed by accessing the last byte of the page and the first byte of the same page, considering the above mentionned misaligned access cases.

The non internal resources (unimplemented areas or External Space) are used to generate the Chip selects (CS0,CS1,CS2 and CS3) (see **Figure 4-3**), which are only active in Normal Expanded Mode, Emulation Expanded Mode and Special Test Mode (see section **3.1.1**).

Table 4-3 shows the address boundaries of each chip select and the relationship with the implemented resources (internal) parameters.

Table 4-3 Global Chip selects memory space

Chip Selects	Bottom address	Top address
$\overline{CS3}$	$00_0800	$0F_FFFF minus RAMSIZE(1)
$\overline{CS2}$	$10_0000	$13_FFFF minus EEPROMSIZE(2)
$\overline{CS2}$(3)	$14_0000	$1F_FFFF
$\overline{CS1}$	$20_0000	$3F_FFFF
$\overline{CS0}$(4)	$40_0000	$7F_FFFF minus FLASHSIZE(5)

NOTES:

1. External RPAGE accesses in (NX, EX and ST)
2. External EPAGE accesses in (NX, EX and ST)
3. When ROMHM is set (see ROMHM in **Table 4-1**) the $\overline{CS2}$ is asserted in the space occupied by this on-chip memory block.
4. When the internal NVM is enabled (see ROMON in **3.1.5**) the $\overline{CS0}$ is not asserted in the space occupied by this on-chip memory block.
5. External PPAGE accesses in (NX, EX and ST)

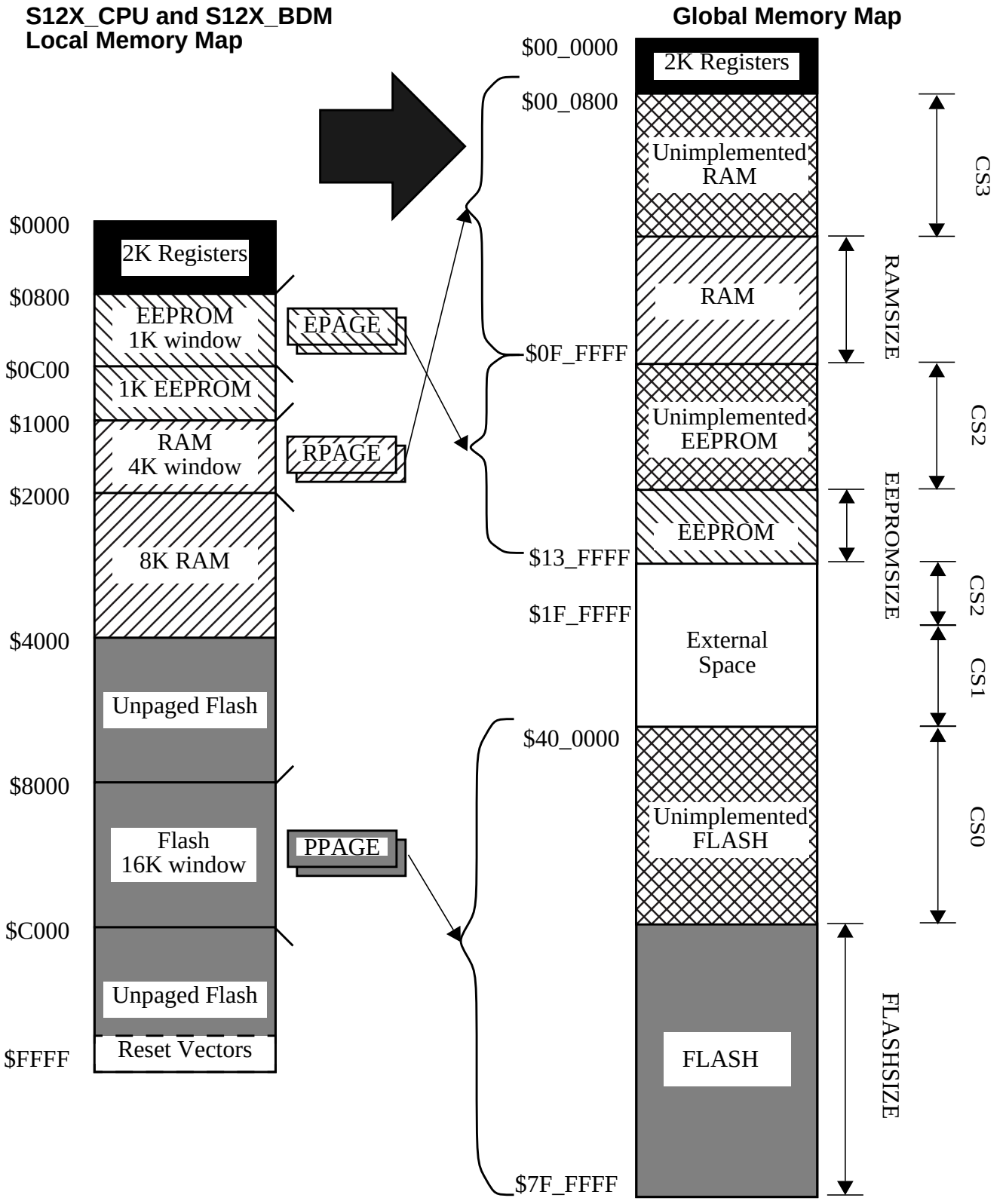


Figure 4-3 Local to Implemented Global Address Mapping (Without GPAGE)

4.2.3 XGATE Memory Map Scheme

4.2.3.1 Expansion of the XGATE local address map

The XGATE 64K byte memory space allows access to internal resources only (Registers, RAM and FLASH). The 2 Kilobyte register address range is the same register address range as for the S12X_CPU and the S12X_BDM module (see **Table 4-4**).

XGATE can access the FLASH in single chip modes, even when the MCU is secured. In expanded modes, XGATE can not access the FLASH when MCU is secured.

The local address of the XGATE RAM access is translated to the the global RAM address range. The XGATE shares the RAM resource with the S12X_CPU and the S12X_BDM module (see **Table 4-4**).

XGATE RAM size (XGRAMSIZE) could be lower or equal than the MCU RAM size (RAMSIZE).

The local address of the XGATE FLASH access is translated to the global address as defined by **Table 4-4**.

Table 4-4 XGATE Implemented memory space

Internal resource	Bottom address	Top address
Registers	$00_0000	$00_07FF
RAM	$10_0000 minus XGRAMSIZE(1)	$0F_FFFF
FLASH	$80_0000 minus FLASHSIZE plus $800(2)	Bottom address plus $F800 minus XGRAMSIZE minus $1(3)

NOTES:
1. XGRAMSIZE is the hexadecimal value of XGATE RAM SIZE in Bytes.
2. FLASHSIZE is the hexadecimal value of FLASH SIZE in Bytes.
3. $F800 is the hexadecimal value of the 64 Kilobytes minus 2 Kilobytes (Registers).

Example:

The MCU FLASHSIZE is 64 Kbytes ($10000) and MCU RAMSIZE is 32 Kbytes ($8000).

The XGATE RAMSIZE is 16 Kbytes ($4000).

The space occupied by the XGATE RAM in the global address space will be:

Bottom address: ($10_0000 minus $4000) = $0F_C000

Top address: $0F_FFFF

XGATE accesses to local address range $0800 - $BFFF will result in accesses to the following FLASH block in the global address space:

Bottom address: ($80_0000 minus $01_0000 plus $800) = $7F_0800

Top address: ($7F_0800 plus ($F800 minus $4000 minus $1)) = $7F_BFFF

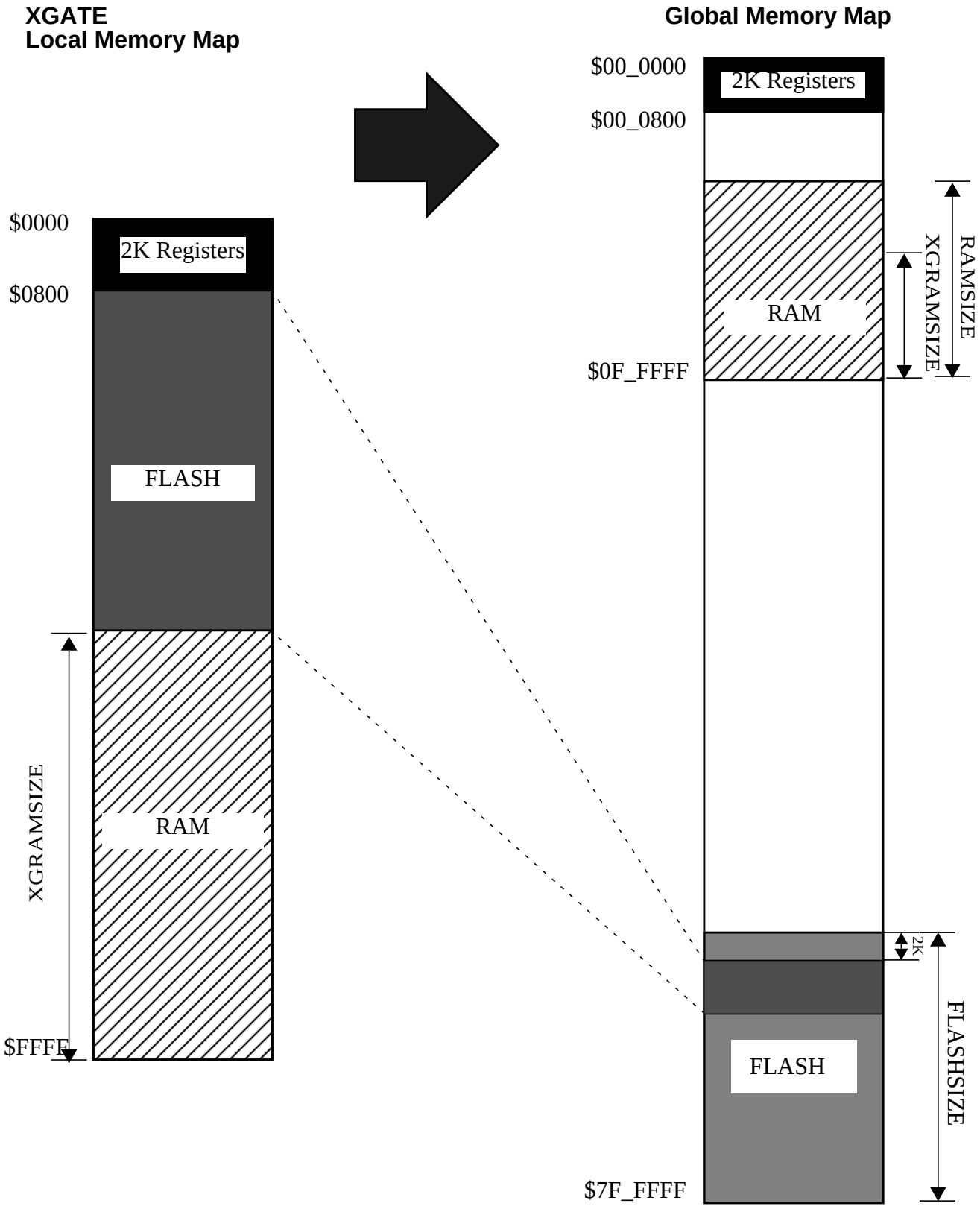


Figure 4-4 local to Global Address Mapping (XGATE)

4.3 Chip Access Restrictions

4.3.1 Illegal XGATE Accesses

A possible access error is flagged by the S12X_MMC and signalled to XGATE under the following conditions:

- XGATE performs misaligned word (in case of load-store or opcode or vector fetch accesses)

- XGATE accesses the register space (in case of opcode or vector fetch)

- XGATE performs a write to Flash in any modes (in case of load-store access)

- XGATE performs an access to a secured Flash in expanded modes (in case of load-store or opcode or vector fetch accesses)

- XGATE performs a write to non-XGATE region in RAM (RAM protection mechanism) (in case of load-store access)

For further details refer to the XGATE Block Guide.

4.3.2 Illegal S12X_CPU Accesses

After programming the protection mechanism registers (see figures **Figure 3-16**, **Figure 3-17**, **Figure 3-18** and **Figure 3-19**) and setting the RWPE bit (see **Figure 3-16**) there are 3 regions recognized by the S12X_MMC module:

- XGATE RAM region
- S12X_CPU RAM region
- Shared Region (XGATE AND S12X_CPU)

If the RWPE bit is set the S12X_CPU write accesses into the XGATE RAM region are blocked. If the S12X_CPU tries to write the XGATE RAM region the AVIF bit is set and an interrupt is generated if enabled. Furthermore if the XGATE tries to write to outside of the XGATE RAM or shared regions and the the RWPE bit is set, the write access is suppressed and the access error will be flagged to the XGATE module (see section **4.3.1** and the XGATE Block Guide).

The bottom address of the XGATE RAM region always starts at the lowest implemented RAM address.

The values stored in the boundary registers define the boundary addresses in 256 byte steps. The 256 byte block selected by any of the registers is always included in the respective region. For example setting the shared region lower boundary register (RAMSHL) to $C1 and the shared region upper boundary register (RAMSHU) to $E0 defines the shared region from address $0F_C100 to address $0F_E0FF in the global memory space (see **Figure 4-5**).

The interrupt requests generated by the S12X_MMC are listed in **Table 4-5**. Refer to the Device User Guide for the related interrupt vector address and interrupt priority.

The following conditions must be satisfied to ensure correct operation of the RAM protection mechanism:

- Value stored in RAMXGU must be lower than the value stored in RAMSHL.
- Value storred RAMSHL must be lower or equal than the value stored in RAMSHU.

Table 4-5 RAM write protection Interrupt Vectors

Interrupt Source	CCR Mask	Local Enable
S12X_CPU access violation	I Bit	AVIE in RAMWPC

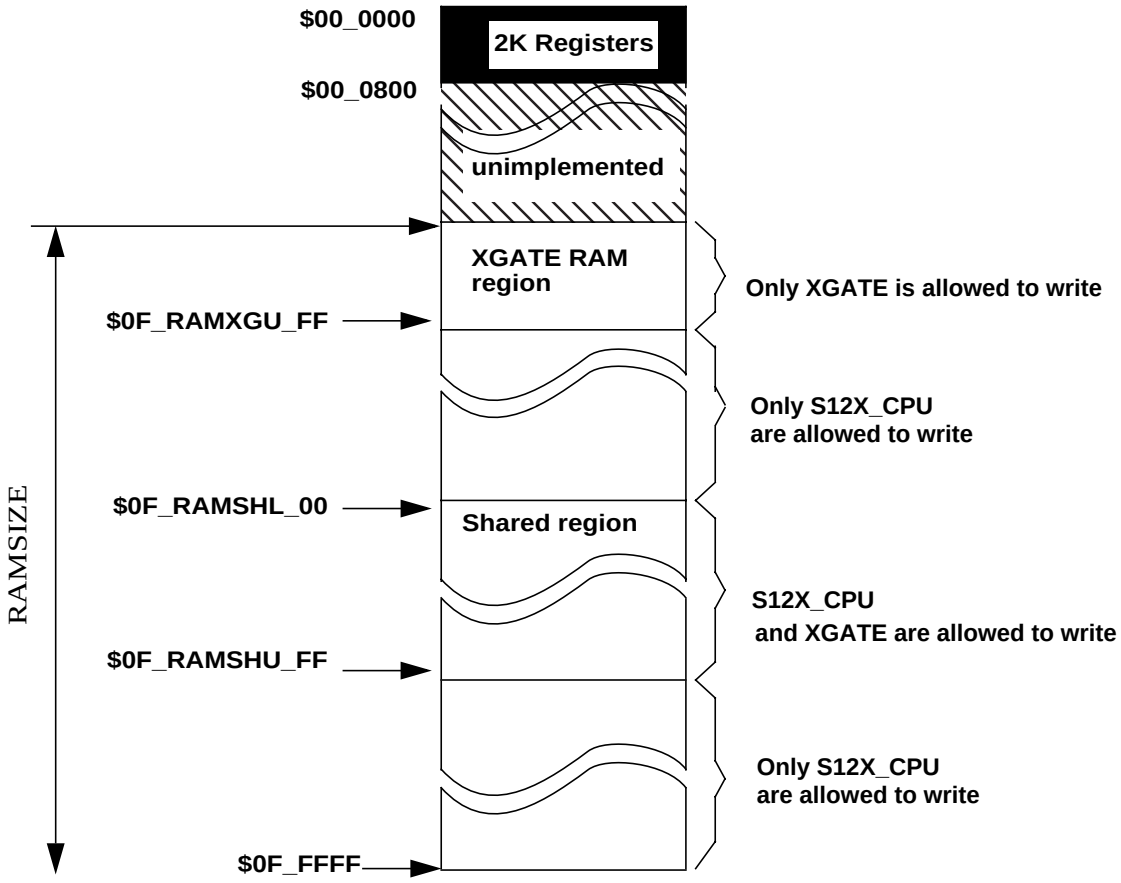


Figure 4-5 RAM write protection scheme

4.4 Chip Bus Control

The S12X_MMC controls the address busses and the data buses that interface the S12X masters (S12X_CPU, S12X_BDM and XGATE) with the rest of the system (master busses). In addition the S12X_MMC handles all S12X_CPU read data bus swapping operations. All internal and external resources are connected to specific target buses (see **Figure 4-6**).

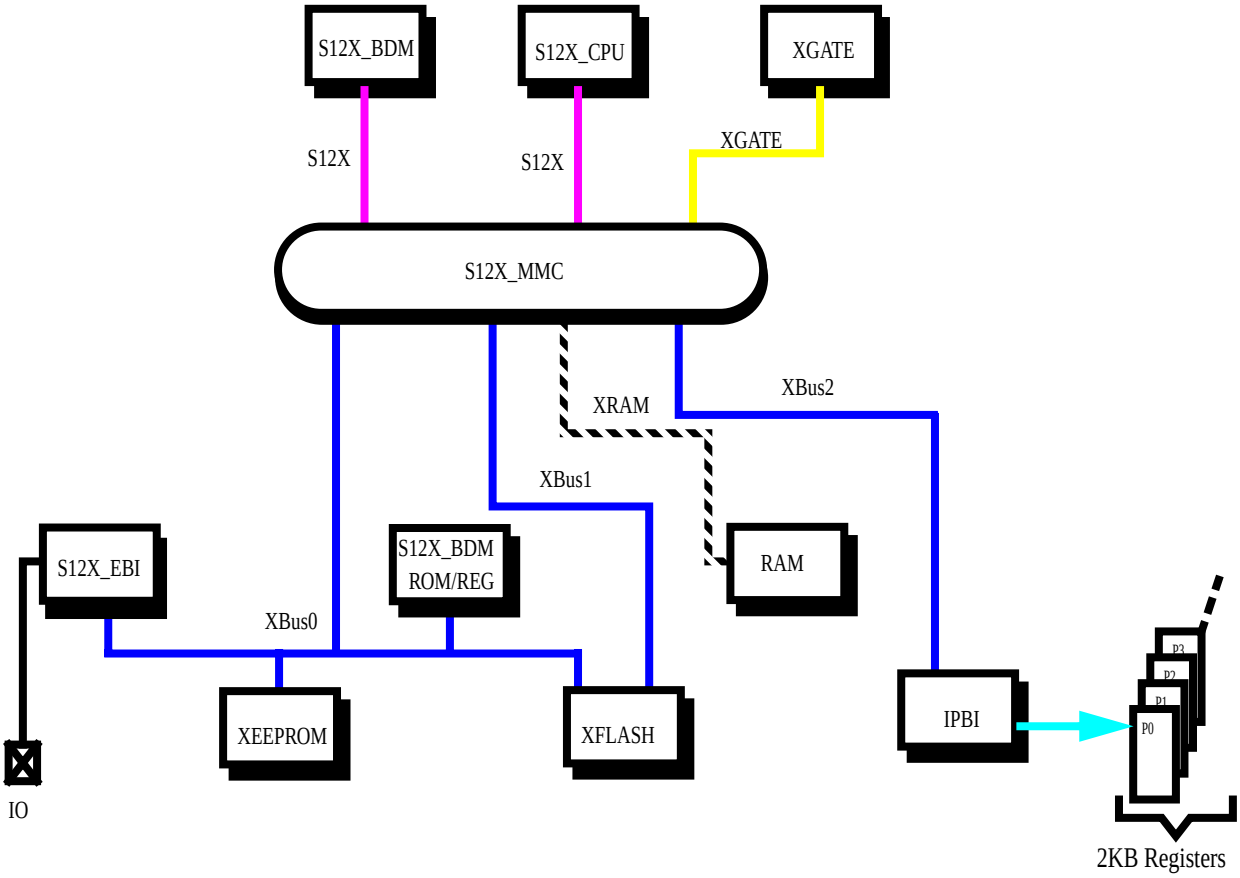


Figure 4-6 S12X Architecture

4.4.1 Master Bus Prioritisation

The following rules apply when prioritizing accesses over master buses:

- The S12X_CPU has priority over the S12X_BDM, unless the S12X_BDM access is stalled for more than 128 cycles. In the later case the S12X_CPU will be stalled after finishing the current operation and the S12X_BDM will gain access to the bus.

- XGATE access to PRU registers constitutes a special case. It is always granted and stalls the S12X_CPU and S12X_BDM for its duration.

4.4.2 Access conflicts on target busses

The arbitration scheme allows only one master to be connected to a target at any given time. The following rules apply when prioritizing accesses from different masters to the same target bus:

- S12X_CPU has always priority over XGATE

- S12X_BDM access has priority over XGATE

- XGATE access to PRU registers constitutes a special case. It is always granted and stalls the S12X_CPU and S12X_BDM for its duration.

- In emulation modes all internal accesses are visible on the external bus as well.

- During access to the PRU registers the external bus is reserved.

4.5 Interrupts

4.5.1 Outgoing Interrupt Requests

The following interrupt requests can be triggered by the S12X_MMC module:

S12X_CPU access violation: The S12X_CPU access violation signals to the S12X_CPU detection of an error condition in the S12X_CPU application code which is resulted in write access to the protected XGATE RAM area (see section **4.3.2**).

Section 5 Initialisation/Application Information

5.1 CALL and RTC Instructions

CALL and RTC instructions are uninterruptable S12X_CPU instructions that automate page switching in the program page window. The CALL instruction is similar to the JSR instruction, but the subroutine that is called can be located anywhere in the local address space or in any Flash or ROM page visible through the program page window. The CALL instruction calculates and stacks a return address, stacks the current PPAGE value and writes a new instruction-supplied value to the PPAGE register. The PPAGE value controls which of the 256 possible pages is visible through the 16K byte program page window in the 64K byte local S12X_CPU memory map. Execution then begins at the address of the called subroutine.

During the execution of the CALL instruction the S12X_CPU performs the following steps:

Writes the current PPAGE value into an internal temporary register and writes the new instruction-supplied PPAGE value into the PPAGE register.

Calculates the address of the next instruction after the CALL instruction (the return address) and pushes this 16-bit value onto the stack.

Pushes the temporarily stored PPAGE value onto the stack.

Calculates the effective address of the subroutine, refills the queue and begins execution at the new address.

This sequence is uninterruptible. There is no need to inhibit interrupts during the CALL instruction execution. A CALL instruction can be performed from any address to any other address in the local S12X_CPU memory space.

The PPAGE value supplied by the instruction is part of the effective address of the S12X_CPU. For all addressing mode variations (except indexed-indirect modes) the new page value is provided by an immediate operand in the instruction. In indexed-indirect variations of the CALL instruction a pointer specifies memory locations where the new page value and the address of the called subroutine are stored. Using indirect addressing for both the new page value and the address within the page allows usage of values calculated at run time rather than immediate values that must be known at the time of assembly.

The RTC instruction terminates subroutines invoked by a CALL instruction. The RTC instruction unstacks the PPAGE value and the return address and refills the queue. Execution resumes with the next instruction after the CALL instruction.

During the execution of an RTC instruction the S12X_CPU performs the following steps:

Pulls the previously stored PPAGE value from the stack.

Pulls the 16-bit return address from the stack and loads it into the PC.

Writes the PPAGE value into the PPAGE register.

Refills the queue and resumes execution at the return address.

This sequence is uninterruptible. The RTC can be executed from anywhere in the local S12X_CPU memory space.

The CALL and RTC instructions behave like JSR and RTS instruction, they however require more execution cycles. Usage of JSR/RTS intructions is therefore recommended when possible and CALL/RTC instructions should only be used when needed. The JSR and RTS instructions can be used to access subroutines that are already present in the local S12X_CPU memory map (i.e. in the same page in the program memory page window for example). However calling a function located in a different page requires usage of the CALL instruction. The function must be terminated by the RTC instruction. Because the RTC instruction restores contents of the PPAGE register from the stack, functions terminated with the RTC instruction must be called using the CALL instruction even when the correct page is already present in the memory map. This is to make sure that the correct PPAGE value will be present on stack at the time of the RTC instruction execution.

5.2 Port Replacement Registers (PRRs)

Registers used for emulation purposes must be rebuilt by the In Circuit Emulator hardware to achieve full emulation of single chip mode operation. These registers are called Port Replacement registers (PRRs) (see **Table 5-1**). PRRs are accessible from all masters using different access types (word aligned, word-misaligned and byte). Each access to PRRs will be extended to 2 bus cycles for write or read accesses independent of the operating mode. In emulation modes all write operations result in writing into the internal registers (peripheral access) and into the emulated registers (external access) located in the PRU in the emulator at the same time. All read operations are performed from external registers (external access) in emulation modes. In all other modes the read operations are performed from the internal registers (peripheral access).

Due to internal visibility of S12X_CPU accesses the S12X_CPU will be halted during XGATE or S12X_BDM access to any PRR. This rule applies also in normal modes to ensure that operation of the device is the same as in emulation modes.

A summary of PRR accesses is the following:

- An aligned word access to a PRR will take 2 bus cycles.
- A misaligned word access to a PRRs will take 4 cycles. If one of the two bytes accessed by the misaligned word access is not a PRR, the access will take only 3 cycles.
- A byte access to a PRR will take 2 cycles.

Table 5-1 PRR listing

PRR Name	PRR local Address	PRR Location
PORTA	$0000	PIM
PORTB	$0001	PIM
DDRA	$0002	PIM
DDRB	$0003	PIM
PORTC	$0004	PIM
PORTD	$0005	PIM
DDRC	$0006	PIM
DDRD	$0007	PIM
PORTE	$0008	PIM
DDRE	$0009	PIM
MMCCTL0	$000A	S12X_MMC
MODE	$000B	S12X_MMC
PUCR	$000C	PIM
RDRIV	$000D	PIM
EBICTL0	$000E	S12X_EBI
EBICTL1	$000F	S12X_EBI
reserved	$0012	S12X_MMC
MMCCTL1	$0013	S12X_MMC
ECLKCTL	$001C	PIM
reserved	$001D	PIM
PORTK	$0032	PIM
DDRK	$0033	PIM

5.3 On-Chip ROM Control

5.3.1 Overview

The MCU offers two modes to support emulation. In the first mode (called generator) the emulator provides the data instead of the internal FLASH and traces the S12X_CPU actions. In the other mode (called observer) the internal FLASH provides the data and all internal actions are made visible to the emulator.

5.3.2 ROM control in single chip modes

In single-chip modes the MCU has no external bus. All memory accesses and program fetches are internal (see **Figure 5-1**).

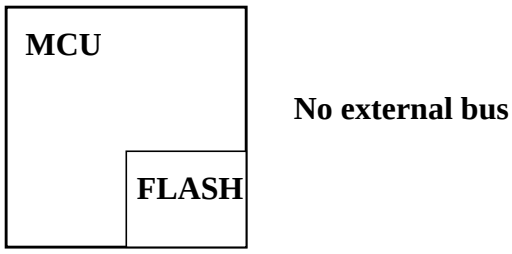


Figure 5-1 ROM in single chip modes

5.3.3 ROM control in emulation singl chip mode

In emulation single-chip mode the external bus is connected to the emulator. If the EROMON bit is set, the internal FLASH provides the data and the emulator can observe all internal S12X_CPU actions on the external bus. If the EROMON bit is cleared, the emulator provides the data (generator) and traces the all S12X_CPU actions (see **Figure 5-2**).

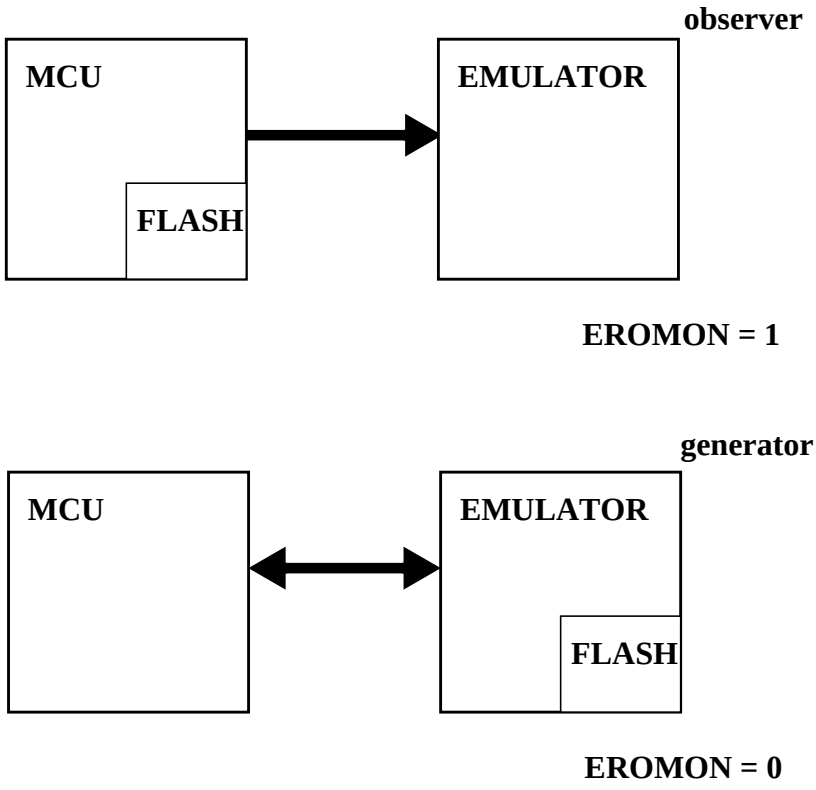


Figure 5-2 ROM in emulation single chip mode

5.3.4 ROM control in normal expanded mode

In normal expanded mode the external bus will be connected to the application. If the ROMON bit is set, the internal FLASH provides the data. If the ROMON bit is cleared, the application memory provides the data (see **Figure 5-3**).

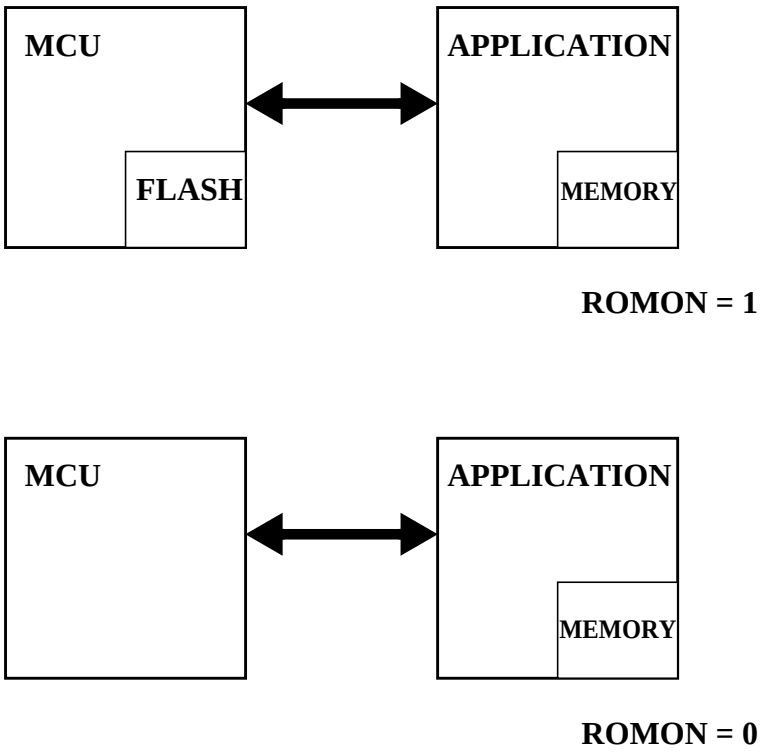


Figure 5-3 ROM in normal expanded mode

5.3.5 ROM control in emulation expanded mode

In emulation expanded mode the external bus will be connected to the emulator and to the application. If the ROMON bit is set, the internal FLASH provides the data. If the EROMON bit is set as well the emulator observes all S12X_CPU internal actions, otherwise the emulator provides the data and traces all S12X_CPU actions (see **Figure 5-4**). When the ROMON bit is cleared, the application memory provides the data and the emulator will observe the S12X_CPU internal actions (see **Figure 5-5**).

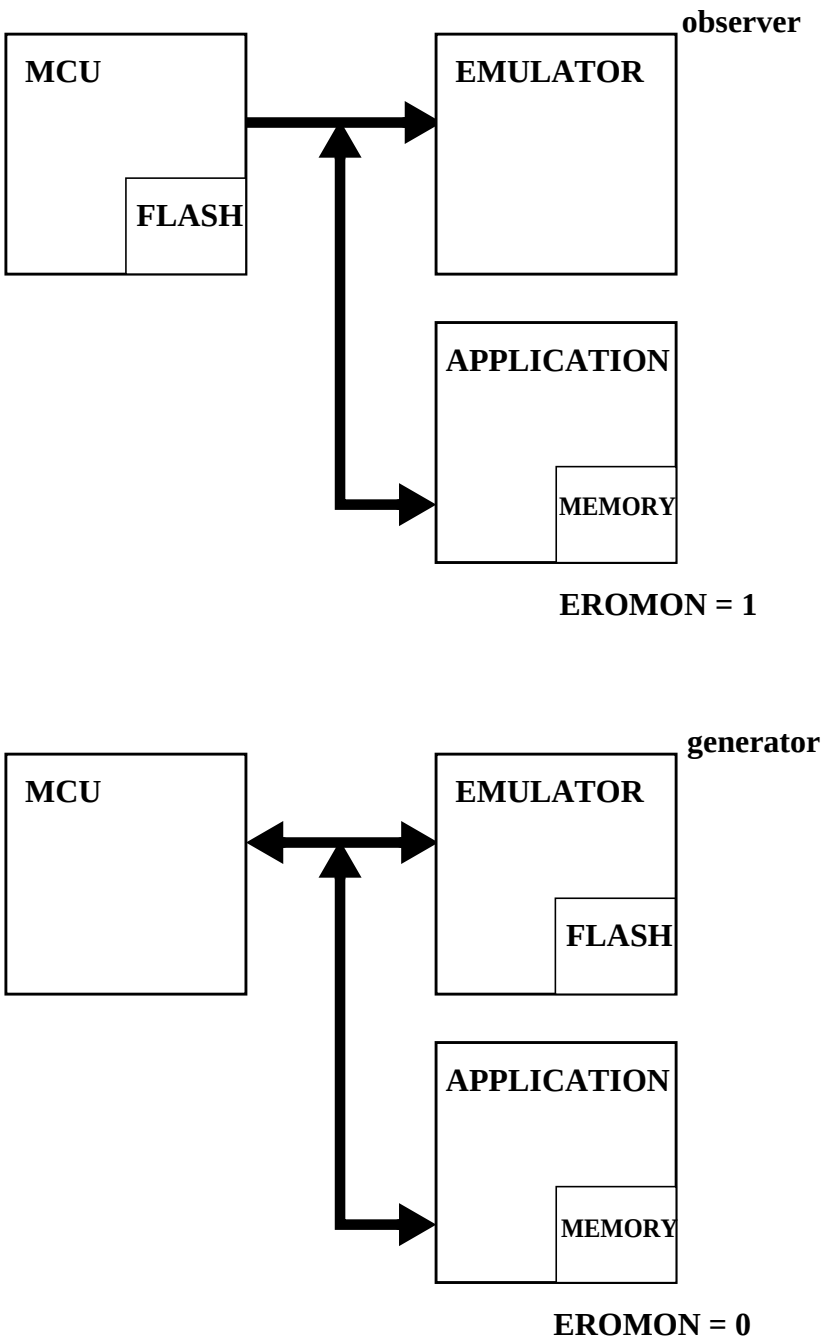


Figure 5-4 ROMON = 1 in emulation expanded mode

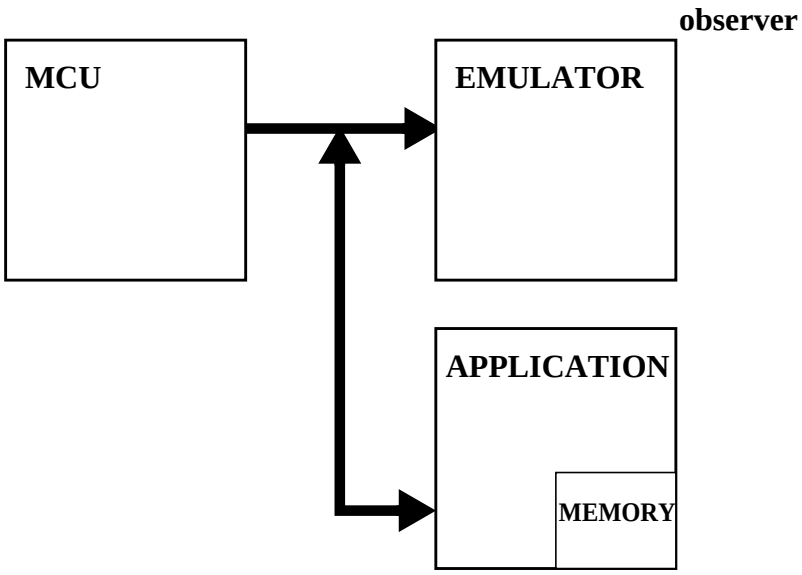


Figure 5-5 ROMON = 0 in emulation expanded mode

5.3.6 ROM control in special test mode

In special test mode the external bus is connected to the application. If the ROMON bit is set, the internal FLASH provides the data, otherwise the application memory provides the data (see **Figure 5-6**).

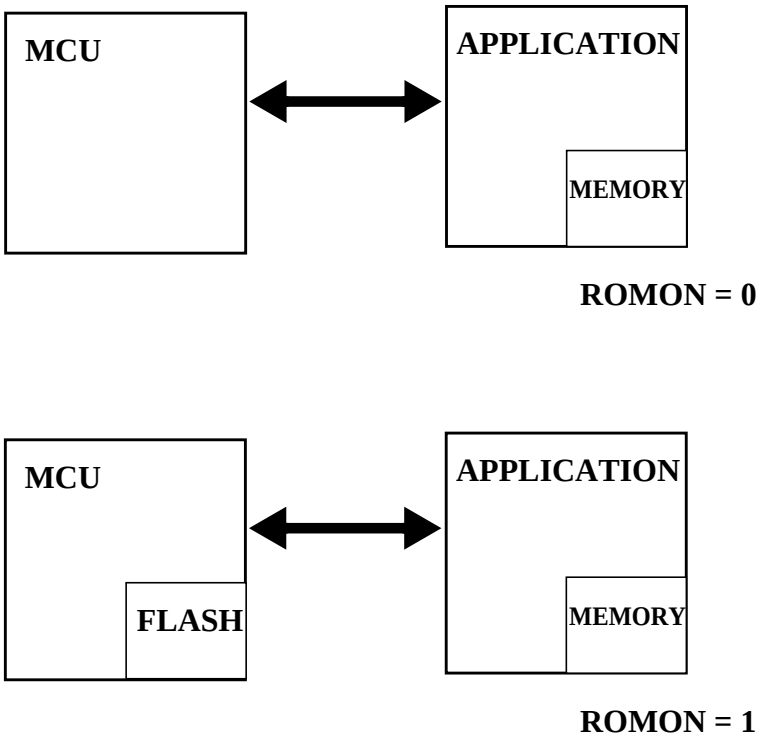


Figure 5-6 ROM in special test mode