S12X_MMC
Module Mapping Control
Block Guide
V01.00

Original Release Date: 05 Feb 2003
Revised: 05 Feb 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

Section 1 Introduction

Section 2 External Signal Description

Section 3 Memory Map/Register Definition

Section 4 Functional Description

Section 1 Introduction

This section describes the functionality of the Module Mapping Control (S12X_MMC) sub-block of the S12 X Platform.

1.1 Overview

The S12X_MMC is the module which controls memory map assignment and selection of internal resources and external space. Internal buses between the core and memories and between the core and peripherals is controlled in this module. The memory expansion is generated in this module.

1.2 Features

- Registers for paging of address space for on-chip RAM, EEPROM, and FLASH/External memory blocks (RPAGE, EPAGE, and PPAGE)
- Memory mapping control and selection based upon address decode and system operating mode
- Core address bus control
- Core data bus control and multiplexing
- Core security state decoding
- Global address memory expansion (GPAGE)
- Internal read visibility
- External stretch and ROM mapping control functions via the MISC register
- Reserved register for test purposes
- Direct page register for mapping the 256-byte direct page
- External address select for unassigned addresses in expanded modes
- Illegal address reset for unassigned addresses in single-chip modes. See SoC Guide for a detailed address map.

1.3 Modes of Operation

Some of the registers operate differently depending on the mode of operation (i.e., normal expanded wide, special single chip, etc.). This is best understood from the register descriptions (**3.1 Register Descriptions**)

1.4 Block Diagram

The block diagram of the MMC is shown in **Figure 1-1**.

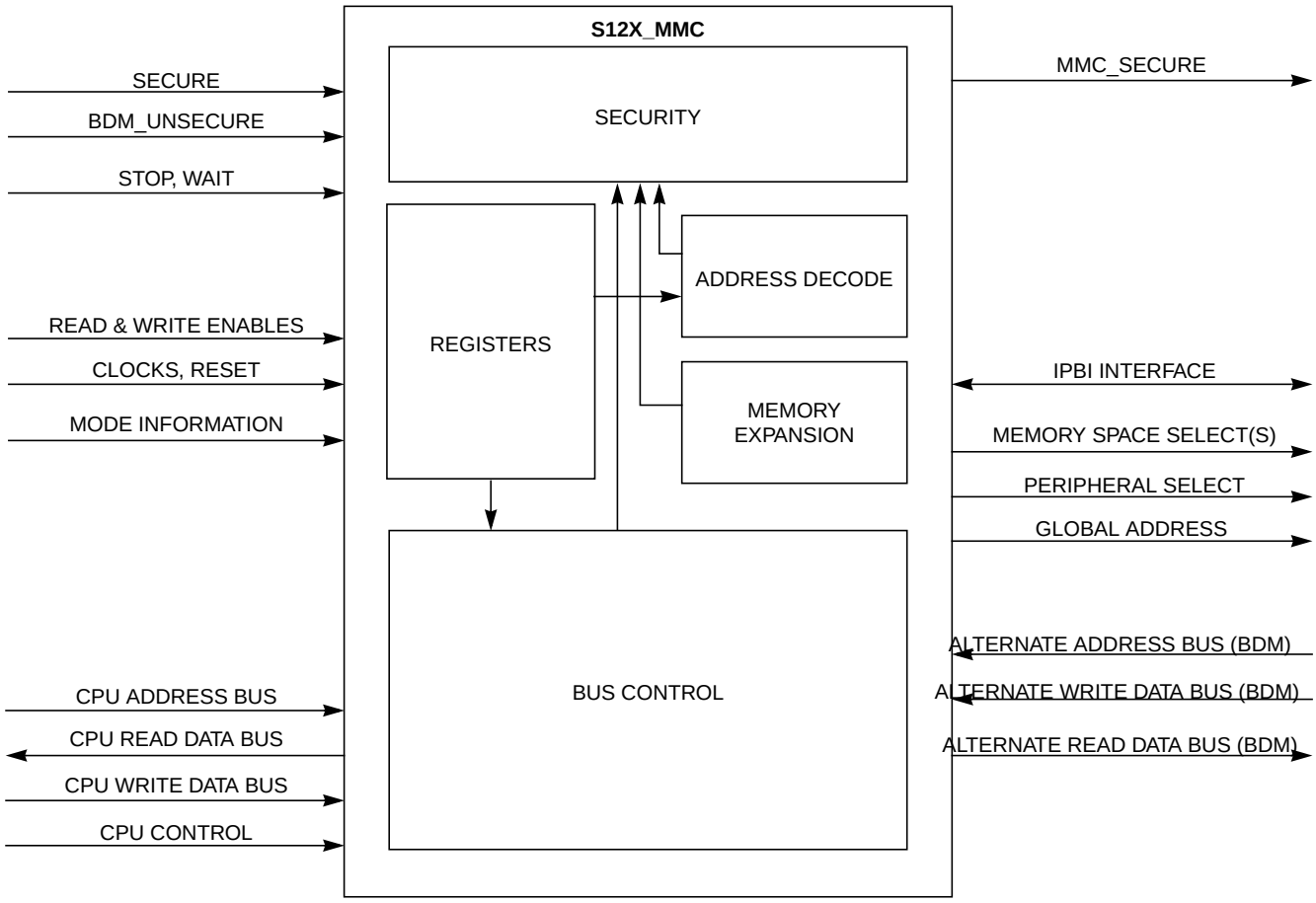

Figure 1-1 Module Mapping Control Block Diagram

Section 2 External Signal Description

S12X_MMC block has no external signals.

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
$0010	GPAGE	Read Write	0	GP6	GP5	GP4	GP3	GP2	GP1	GP0
$0011	RSVD	Read Write								
$0012	DIRECT	Read Write	DP15	DP14	DP13	DP12	DP11	DP10	DP9	DP8
$0013	MISC	Read Write	0	0	0	EROMON	EXSTR1	EXSTR0	ROMHM	ROMON
$0014	RSVD/ MTSTO	Read Write	BIT 7	6	5	4	3	2	1	BIT 0
$0015	RSVD	Read Write	BIT 7	6	5	4	3	2	1	BIT 0
$0016	RPAGE	Read Write	RP7	RP6	RP5	RP4	RP3	RP2	RP1	RP0
$0017	EPAGE	Read Write	EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0
$0030	PPAGE	Read Write	PIX7	PIX6	PIX5	PIX4	PIX3	PIX2	PIX1	PIX0
$0031	RSVD	Read Write	BIT 7	6	5	4	3	2	1	BIT 0

= Unimplemented

Figure 3-1 Module Mapping Control Register Summary

3.1 Register Descriptions

3.1.1 Global Page Index Register (GPAGE)

Register address Base + $10

	7	6	5	4	3	2	1	0
R W	0	GP6	GP5	GP4	GP3	GP2	GP1	GP0
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 3-2 Global Page Index Register (GPAGE)}

Read: Anytime
Write: Anytime

The HCS12 architecture limits the physical address space available to 64K bytes. The Global Page Index Register in HCS12X allows for integrating up to 8M byte of memory by using the seven global page index bits to page 64K byte blocks into the memory map as defined in **Figure 3-3** and **Table 3-1**.

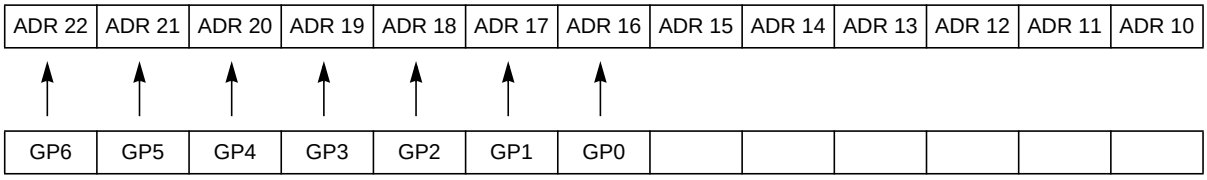

Figure 3-3 GPAGE Address Mapping

GP6–GP0 — Global Page Index Bits 6–0

These page index bits are used to select which of the 128 memory pages is to be accessed in the Global Page Window as shown in **Table 3-1**.

Table 3-1 Global Page Index Register Bits

GP6	GP5	GP4	GP3	GP2	GP1	GP0	Memory Space Selected
0	0	0	0	0	0	0	64K page 0
0	0	0	0	0	0	1	64K page 1
0	0	0	0	0	1	0	64K page 2
0	0	0	0	0	1	1	64K page 3

Table 3-1 Global Page Index Register Bits

GP6	GP5	GP4	GP3	GP2	GP1	GP0	Memory Space Selected
. . . . .	. . . . .	. . . . .	. . .	. . . . .	. . . . .	. . . . .	. . . . .
1	1	1	1	1	0	0	64K page 124
1	1	1	1	1	0	1	64K page 125
1	1	1	1	1	1	0	64K page 126
1	1	1	1	1	1	1	64K page 127

3.1.2 Direct Page Register (DIRECT)

Register address Base + $12

	7	6	5	4	3	2	1	0
R W	DP15	DP14	DP13	DP12	DP11	DP10	DP9	DP8
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 3-4 Direct Page Register (DIRECT)

Read: Anytime

Write: anytime in special modes, one time only in normal modes

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

DP15–DP8 — Direct Page Index Bits 15–8

These bits will be used as the address bits 15-8 in direct page instructions.

3.1.3 Miscellaneous System Control Register (MISC)

Register address Base + $13

	7	6	5	4	3	2	1	0
R	0	0	0	EROMON	EXSTR1	EXSTR0	ROMHM	ROMON
W								
Reset: Expanded or Emulation	0	0	0	—(1)	1	1	0	—(1)
Reset:Single Chip	0	0	0	1	1	1	0	1
Reset: Special Test	0	0	0	—(1	1	1	0	0

= Unimplemented or Reserved

NOTES:
1. The reset state of this bit is determined at the chip integration level.

Figure 3-5 Miscellaneous System Control Register (MISC)

Read: Anytime

Write: As stated in each bit description

This register initializes miscellaneous control functions.

EROMON — Emulation Mode Enable FLASH EEPROM or ROM

Write: never

This bit is used with the ROMON bit to enable the FLASH EEPROM or ROM memory in the memory map.

EROMON and ROMON control the status of the Flash in the memory map as described by **Table 3-2**.

Table 3-2 Data sources

MODE	CPU accesses	ROMON	EROMON	DATA SOURCE	Stretch*
Normal Single Chip	Any Address	X	X	Internal	N
Special Single Chip					
Emulation of Single Chip	Flasharea	X	1	Internal Flash	N
	Flasharea	X	0	Emulator Memory	
	Unimplemented	X	X	Emulator Memory	
	Other			Internal	
Normal Expanded Mode	Flasharea	0	X	External Application	Y
	Unimplemented	X	X	External Application	Y
	Other			Internal	N
	Flasharea	1	X	Internal	N

Table 3-2 Data sources

MODE	CPU accesses	ROMON	EROMON	DATA SOURCE	Stretch*
Emulation of Expanded Mode	Flasharea	0	X	External Application	Y
	Flasharea	1	0	Emulation Memory	N
	Flasharea	1	1	Internal	N
	Unimplemented	X	X	Application Memory	Y
	Other			Interna	N
Special Test Mode	Flasharea	1	X	Interna	N
	Flasharea	0	X	External Memory	
	Unimplemented	X	X	External Memory	
	Other			Internal	

* Stretch comment does not reflect generically stretched PRU (Port Replacement Unit) accesses.

Internal means resources internal to the MCU are read/written

Emulation Memory means resource inside the emulator are read/written (PRU registers, flash replacement; RAM, EEPROM and Register Space are always considered internal)

Application Memory means resources residing in the customers application are read/written (in Emulation of Expanded Mode, this happens through the emulator)

External Memory means resources residing outside the MCU are read/written (applicable in Special Test mode).

EXSTR1 and EXSTR0 — External Access Stretch Bits 1 and 0

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

This two bit field determines the amount of clock stretch on accesses to the external address space as shown in **Table 3-3**. In single chip and special test modes these bits have no meaning or effect.

Table 3-3 External Stretch Bit Definition

Stretch Bit EXSTR1	Stretch Bit EXSTR0	Number of E Clocks Stretched EWAIT disabled	Number of E Clocks Stretched EWAIT enabled
0	0	1; reserved for future use	>= 2 cycles; reserved
0	1	1	>= 2 cycles
1	0	2	>= 2 cycles
1	1	3	>= 3 cycles

ROMHM — FLASH EEPROM or ROM Only in Second Half of Memory Map

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

1 = Disables direct access to the FLASH EEPROM or ROM in the lower half of the memory map.These physical locations of the FLASH EEPROM or ROM can still be accessed through the Program Page window. Accesses to $4000-$7FFF will be mapped to 14_0000-14_3FFF in the global external memory space.

0 = The fixed page(s) of FLASH EEPROM or ROM in the lower half of the memory map can be accessed. Accesses to $4000-$7FFF will be mapped to $7F_8000-$7F_BFFF in the global FLASH/ROM memory space.

ROMON — Enable FLASH EEPROM or ROM

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

This bit is used to enable the FLASH EEPROM or ROM memory in the memory map.

1 = Enables the FLASH EEPROM or ROM in the memory map.

Disables the FLASH EEPROM or ROM from the memory map.EROMON and ROMON control the status of the Flash in the memory map as described by **Table 3-2**.

3.1.4 Reserved Test Register Zero (MTST0)

Register address Base + $14

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 3-6 Reserved Test Register Zero (MTST0)

Read: Anytime

Write: No effect — this register location is used for internal test purposes.

3.1.5 RAM Page Index Register (RPAGE)

Register address Base + $0016

	7	6	5	4	3	2	1	0
R W	RP7	RP6	RP5	RP4	RP3	RP2	RP1	RP0
Reset	1	1	1	1	1	1	0	1

= Unimplemented or Reserved

Figure 3-7 RAM Page Index Register (RPAGE)}

Read: AnytimeWrite: Anytime

The HCS12 architecture limits the RAM physical address space available to 12K bytes. In HCS12X, the RAM Page Index Register allows for integrating up to (1M- 4K) bytes of RAM into the system by using the eight page index bits to page 4K byte blocks into the RAM Page Window located from $1000 to $1FFF ($00_1000 - $0F_FFFF in the global address map) as defined in **Figure 3-8** and **Table 3-4**.

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.*

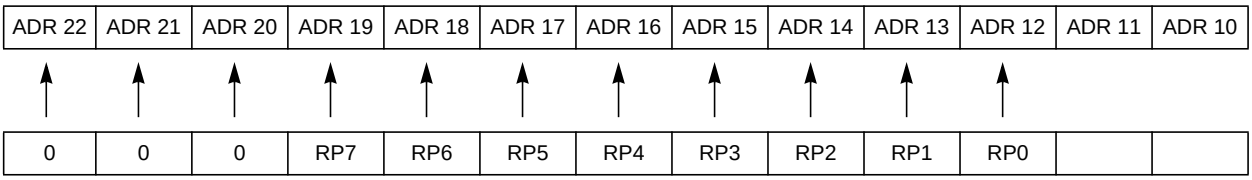

Figure 3-8 RPAGE Address Mapping

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 as shown in **Table 3-4**.

Table 3-4 RAM Page Index Register Bits

RP7	RP6	RP5	RP4	RP3	RP2	RP1	RP0	Program Space Selected
0	0	0	0	0	0	0	0	$1800 - $1FFF 2K visible only
0	0	0	0	0	0	0	1	4K page 1
0	0	0	0	0	0	1	0	4K page 2
0	0	0	0	0	0	1	1	4K page 3
. . . . .	. . . . .	. . . . .	. . . . .	. . . .	. . . . .	. . . . .	. . . . .	. . . . .
1	1	1	1	1	1	0	0	4K page 252
1	1	1	1	1	1	0	1	4K page 253
1	1	1	1	1	1	1	0	4K page 254(1)
1	1	1	1	1	1	1	1	4K page 255(1

NOTES:
1. The two fixed 4K pages from $2000 - $3FFF of RAM are equivalent to pages 254 and 255.

3.1.6 EEPROM Page Index Register (EPAGE)

Register address Base + $0017

	7	6	5	4	3	2	1	0
R W	EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0
Reset	1	1	1	1	1	1	1	0

= Unimplemented or Reserved

Figure 3-9 EEPROM Page Index Register (EPAGE)

Read: Anytime
Write: Anytime

The HCS12 architecture limits the EEPROM physical address space available to 2K bytes. In HCS12X architecture, the EEPROM Page Index Register allows for integrating up to 256K byte of EEPROM into the system by using the eight page index bits to page 1K byte blocks into the EEPROM Page Window located from $800 to $0BFF ($10_0000 - $13_FFFF in the global address map) as defined in **Figure 3-10** and **Table 3-5**.

ADR 22	ADR 21	ADR 20	ADR 19	ADR 18	ADR 17	ADR 16	ADR 15	ADR 14	ADR 13	ADR 12	ADR 11	ADR 10
0	0	1	0	0	EP7	EP6	EP3	EP2	EP3	EP2	EP1	EP0

Figure 3-10 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 as shown in **Table 3-5**.

Table 3-5 EEPROM Page Index Register Bits

EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0	EEPROM Space Selected
0	0	0	0	0	0	0	0	1K page 0
0	0	0	0	0	0	0	1	1K page 1
0	0	0	0	0	0	1	0	1K page 2
0	0	0	0	0	0	1	1	1K page 3
. . . . .	. . . . .	. . . . .	. . . . .	. . . .	. . . . .	. . . . .	. . . . .	. . . . .

Table 3-5 EEPROM Page Index Register Bits

EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0	EEPROM Space Selected
1	1	1	1	1	1	0	0	1K page 252
1	1	1	1	1	1	0	1	1K page 253
1	1	1	1	1	1	1	0	1K page 254
1	1	1	1	1	1	1	1	1K page 255(1)

NOTES:
1. The fixed 1K page of EEPROM will always be equivalent to page 255.

3.1.7 Program Page Index Register (PPAGE)

Register address Base + $30

	7	6	5	4	3	2	1	0
R W	PIX7	PIX6	PIX5	PIX4	PIX3	PIX2	PIX1	PIX0
Reset	1	1	1	1	1	1	1	0

= Unimplemented or Reserved

Figure 3-11 Program Page Index Register (PPAGE)}

Read: Anytime
Write: Anytime

The HCS12 architecture limits the physical address space available to 64K bytes. In HCS12X, the Program Page Index Register allows for integrating up to 4M byte of FLASH or ROM into the system by using the eight page index bits to page 16K byte blocks into the Program Page Window located from $8000 to $BFFF ($40_0000 - $7F_FFFF in the global map) as defined in **Table 3-6** and **Figure 3-12**. CALL and RTC instructions have special access to read and write this register without using the address bus.

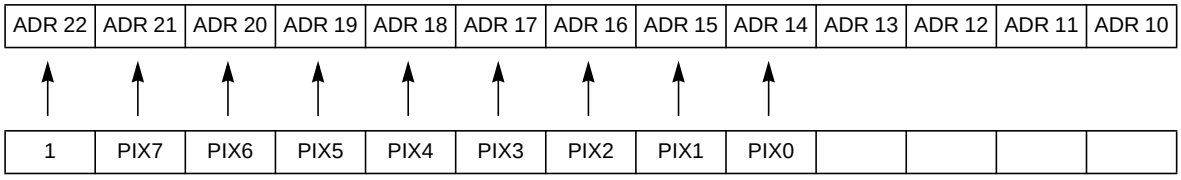

Figure 3-12 PPAGE Address Mapping

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

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 as shown in **Table 3-6**.

Table 3-6 Program Page Index Register Bits

PIX7	PIX6	PIX5	PIX4	PIX3	PIX2	PIX1	PIX0	Program Space Selected
0	0	0	0	0	0	0	0	16K page 0
0	0	0	0	0	0	0	1	16K page 1
0	0	0	0	0	0	1	0	16K page 2
0	0	0	0	0	0	1	1	16K page 3
. . . . .	. . . . .	. . . . .	. . . . .	. . . .	. . . . .	. . . . .	. . . . .	. . . . .
1	1	1	1	1	1	0	0	16K page 252
1	1	1	1	1	1	0	1	16K page 253(1)
1	1	1	1	1	1	1	0	16K page 254(2)
1	1	1	1	1	1	1	1	16K page 255(3)

NOTES:
1. Equivalent to fixed page at $4000-$7FFF ($7F_4000-$7F_7FFF) when ROMHM=0
2. Reset Condition, resulting in a linear address map out of reset
3. Equivalent to fixed page at $C000-$FFFF ($7F_C000-$7F_FFFF)

Section 4 Functional Description

The S12X_MMC block performs four basic functions of the S12X sub-system operation: bus control, address decoding and select signal generation, memory expansion, and security decoding for the system. Each aspect is described in the following subsections.

4.1 Bus Control

The S12X_MMC controls the address bus and data buses that interface the S12X sub-system with the rest of the system. This includes the multiplexing of the input data buses onto the main S12X_CPU read data bus and control of data flow from the S12X_CPU to the output address and data buses of the S12X sub-system. In addition, the S12X_MMC handles all S12X_CPU read data bus swapping operations.

4.2 Address Decoding

As data flows on the Core address bus, the S12X_MMC decodes the address information, generates the proper global address, determines whether the S12X_BDM firmware space, the register space or an external or array space is being addressed and generates the correct select signal. This decoding operation also interprets the mode of operation of the system and the state of the global signals from the S12X_CPU and S12X_BDM in order to generate the proper select and address.

4.2.1 Mode Considerations

In expanded modes, all address space not used by internal resources is by default external memory space. In single-chip modes, any S12X_CPU access to an address not used by internal resources will result in an illegal access reset, except in active background debug mode.

4.3 Memory Expansion

4.3.1 Flash/External Memory Expansion

The HCS12 Core architecture limits the physical address space available to 64K bytes. The Program Page Index Register in HCS12X allows for integrating up to 4M byte of FLASH or ROM into the system by using the eight page index bits to page 16K byte blocks into the Program Page Window located from $8000 to $BFFF in the physical memory space. ~~The paged memory space can consist of solely on-chip memory or a combination of on-chip and off-chip memory. This partitioning is configured at system integration through the use of the parameters specifying the lowest address of on-chip FLASH or ROM. The options available to the integrator are explained in the S12X Core Integration Guide.~~

The PPAGE register holds the page select value for the Program Page Window. The value of the PPAGE register can be manipulated by normal read and write instructions as well as the CALL and RTC instructions.

Control registers, vector spaces, and a portion of on-chip memory are located in unpaged portions of the 64K byte physical address space.

The starting address of a service routine must be located in unpaged memory unless the user is certain that PPAGE will be set to an appropriate value when the service routine is executed. However, a service routine can call other routines that are in paged memory. The upper 16K byte block of memory space ($C000–$FFFF) is unpaged. It is recommended that all reset and interrupt vectors point to locations in this area.

A graphical example of a memory paging for a system configured as 1M byte on-chip FLASH/ROM is given in **Figure 4-1**.

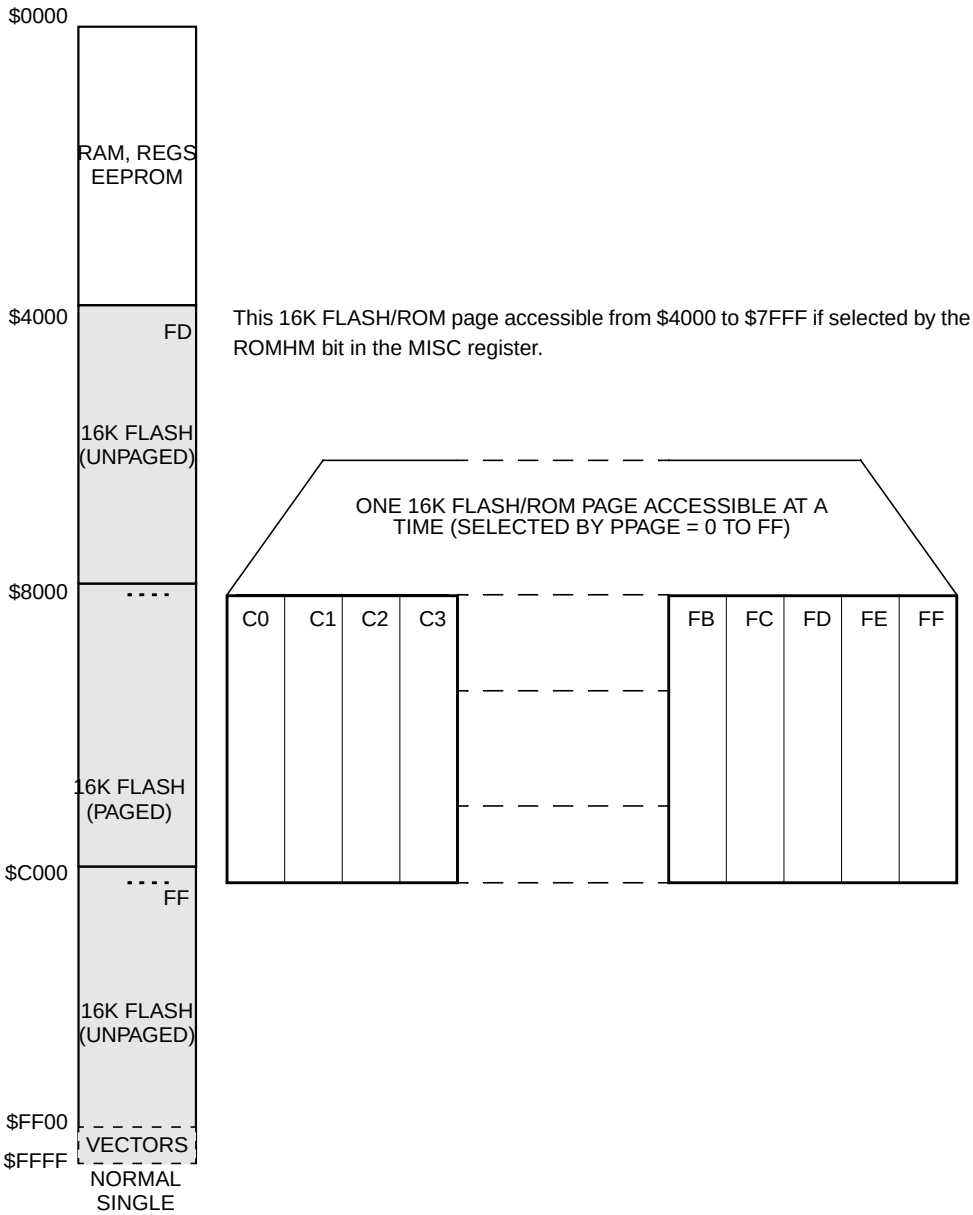

Figure 4-1 Memory Paging Example: 1M Byte On-Chip FLASH/ROM

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

Table 4-1 Physical FLASH/ROM Allocated

Address Space	Page Window Access	ROMHM	External select	AB[22:14]
$4000–$7FFF	N/A	0	0	$7FD
	N/A	1	1	$050
$8000–$BFFF	External	N/A	1	$1,PIX7:0
	Internal	N/A	0	
$C000–$FFFF	N/A	N/A	0	$7FF

4.3.2 CALL and Return from Call Instructions

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

During the execution of a CALL instruction, the S12X_CPU:

- Writes the old 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 old PPAGE value onto the stack.
- Calculates the effective address of the subroutine, refills the queue, and begins execution at the new address on the selected page of the expansion window.

This sequence is uninterruptable; there is no need to inhibit interrupts during CALL execution. A CALL can be performed from any address in memory to any other address.

The PPAGE value supplied by the instruction is part of the effective address. 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 CALL, 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 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. RTC unstacks the PPAGE value and the return address and refills the queue. Execution resumes with the next instruction after the CALL.

During the execution of an RTC instruction, the S12X_CPU:

- Pulls the old PPAGE value from the stack
- Pulls the 16-bit return address from the stack and loads it into the PC
- Writes the old PPAGE value into the PPAGE register
- Refills the queue and resumes execution at the return address

This sequence is uninterruptable; an RTC can be executed from anywhere in memory, even from a different page of extended memory in the expansion window.

The CALL and RTC instructions behave like JSR and RTS, except they use more execution cycles. Therefore, routinely substituting CALL/RTC for JSR/RTS is not recommended. JSR and RTS can be used to access subroutines that are on the same page in expanded memory. However, a subroutine in expanded memory that can be called from other pages must be terminated with an RTC. And the RTC unstacks a PPAGE value. So any access to the subroutine, even from the same page, must use a CALL instruction so that the correct PPAGE value is in the stack.

4.3.3 RAM and EEPROM Memory Expansion

Similarly, the RAM Page Index Register allows for integrating up to 1M-4K byte of RAM into the system by using the eight page index bits to page 4K byte blocks into the RAM Page Window located from $1000 to $1FFF in the physical memory space, and the EEPROM Page Index Register allows for integrating up to 256K byte of EEPROM into the system by using the eight page index bits to page 1K byte blocks into the EEPROM Page Window located from $0800 to $0BFF in the physical memory space. Unimplemented portions within the 1M RAM and 256K EEPROM spaces will be defined by parameters at chip integration.

4.3.4 Other Memory Expansion

In addition to these memory expansion mechanisms, there is another called 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, consolidated way to access all of the various pages of FLASH, RAM and EEPROM, as well as additional external memory.

Figure 4-3, **Figure 4-4**, **Figure 4-5**, and **Figure 4-2** show how logical addresses are mapped to global addresses.

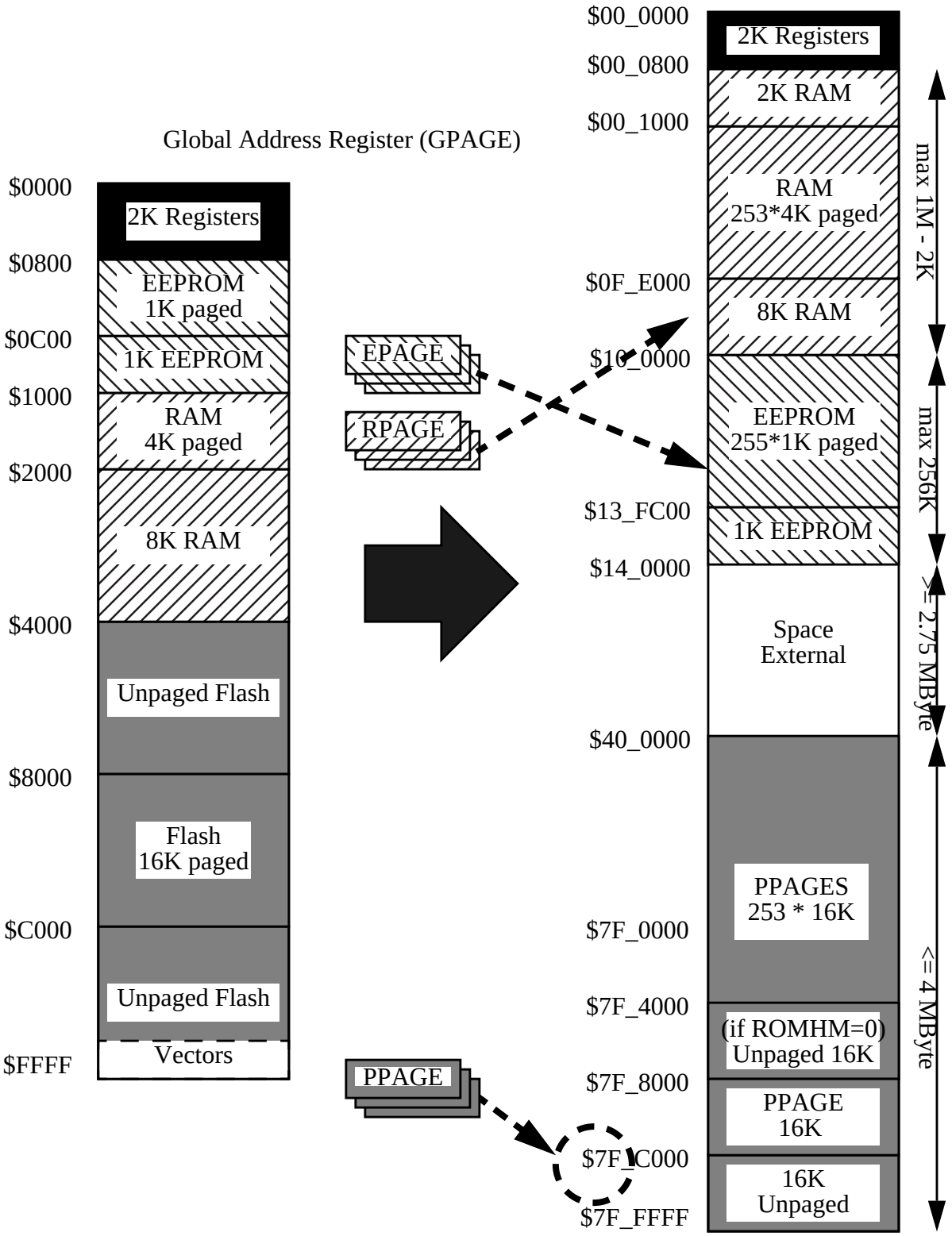

Figure 4-2 Logical to Global Address Mapping

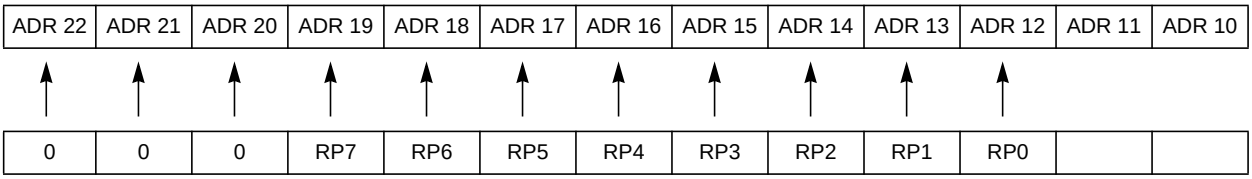

Figure 4-3 RPAGE Address Mapping (copy of Figure 3-8)

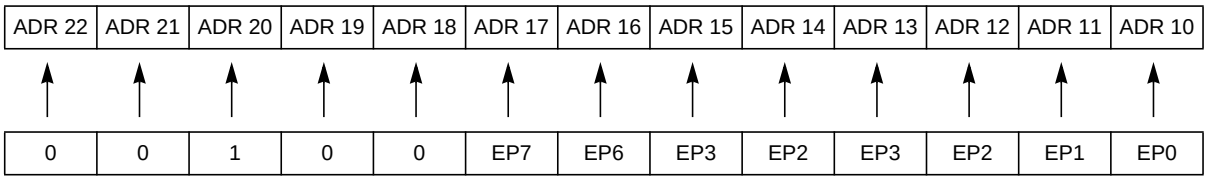

Figure 4-4 EPAGE Address Mapping (copy of Figure 3-10)

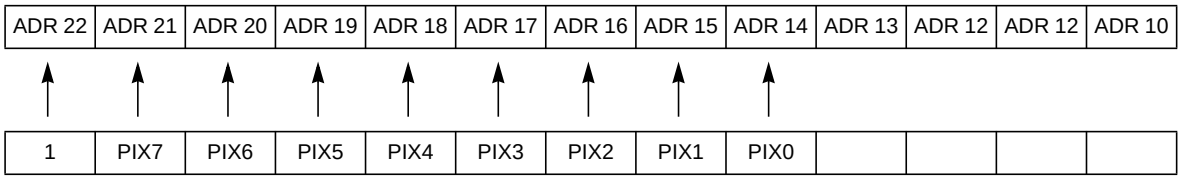

Figure 4-5 PPAGE Address Mapping (copy of Figure 3-12)