# NXP

**Freescale Semiconductor**
Application Note

# HCS12X Family Memory Organization

by:  John Doyle & Daniel McKenna
     MCD Applications EKB

## 1    Introduction

The HCS12X family is the successor to the HCS12 family, with many additional features. One new feature is the increased memory available to the CPU and the methods available to access it. This document focuses on the improved memory map configuration.

## 2    Memory Map

The internal memory available to the CPU has been increased to a maximum of 8M bytes. The sixteen CPU address lines result in a maximum address space of 64K bytes. Therefore, two new access schemes are available, each of which extends the memory map. These schemes are called global addressing and logical addressing. Each scheme is discussed in general terms, and then the memory configuration particular to the MC9S12XDP512 is explained.

**Contents**

*freescale*™
semiconductor

## 2.1   Memory Map Controller

The different registers used to control memory access reside within the Memory Map Controller (MMC) block and are accessible to the CPU through the normal register space of the memory map (see Figure 1).

These registers are prefixed to the CPU address lines; together, they increase the number of address lines from the sixteen from the CPU to the 23 required for the memory address decoding.
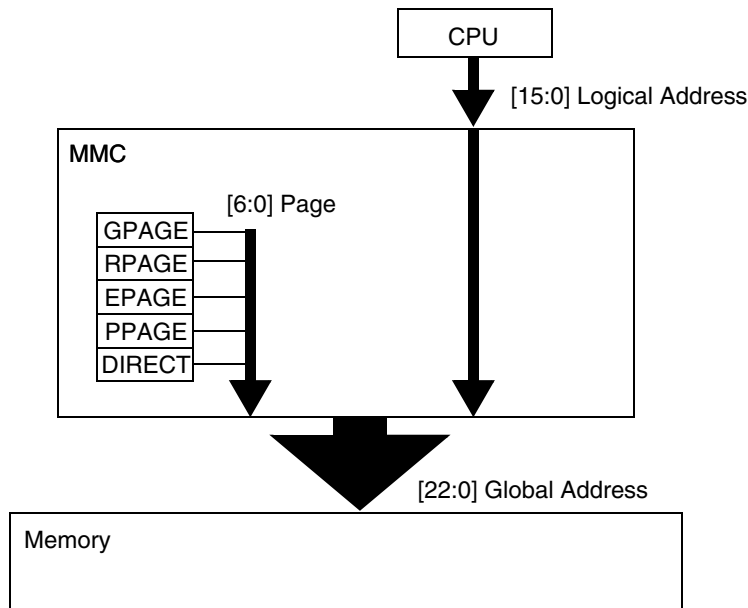


**Figure 1. MMC Module Overview**

## 2.2   Global Address Map

The global memory address space runs in a linear fashion across the full 8M bytes, from $00_0000 through to $7F_FFFF. Figure 2 shows the partitions for the different memory technologies currently used for the HCS12X family.

To extend the sixteen address lines from the CPU to the 23 required, a register called the GPAGE register is provided. The seven bits within the GPAGE register are prefixed to the CPU address lines, thereby creating the full 23-bit address space. See Figure 3.
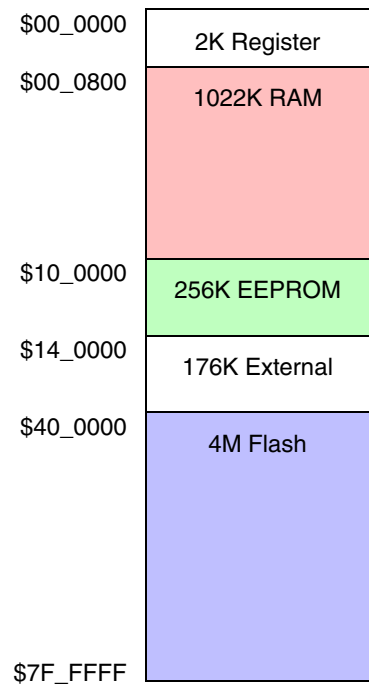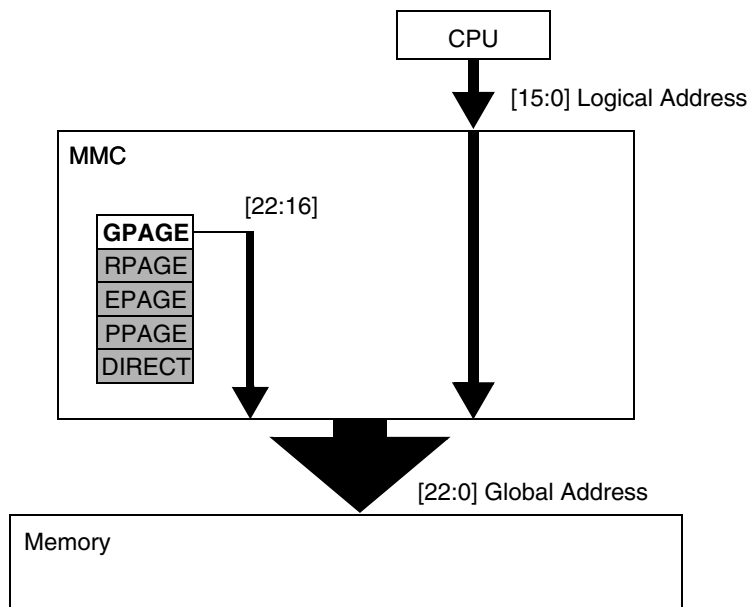
**Figure 2. Generic Global Memory Map**



**Figure 3. Memory Access Using GPAGE**

The whole global memory can be viewed as 128 pages, each comprising 64K bytes of memory, the global page register GPAGE selecting one of the 64K byte pages. Global load and store instructions have been introduced to the CPU instruction set. Using the GPAGE register with the new instructions allows data to be located almost anywhere within the 8M bytes of memory.

```
Memory Address Bits = PPP PPPP aaaa aaaa aaaa aaaa
Global Address Range = $00_0000 to $7f_FFFF

Key:
P = GPAGE bit
a = CPU address
```

After reset, the value in the GPAGE register is $00. In order to use the global method to access locations beyond the first 64K page, the GPAGE register must contain the value for the corresponding page. For example, the following assembler code stores $F5 to the first location in unpaged RAM, at global address $0F_E000.

```
__asm {
      ; store value $F5 in first location of unpaged RAM at $0F_E000
      MOVB    #$0F, $10 ; write $0F to the GPAGE register
      LDAA    #$F5
      GSTAA   $E000    ; Global store to $0F_E000
}
```

The global paging scheme can be extremely useful when programs must access large tables of data. These tables can be accessed through the GPAGE register, whilst the program is accessed via the logical address.

## 2.3    Logical Address Map

An example of a logical memory map is shown in Figure 4. The 64K byte memory range is completely accessible using the sixteen CPU address lines. In the following example, the EEPROM is accessed via the CPU address range $0800 to $0BFF; the RAM address range is $1000 to $1FFF; and the Flash is from $C000 to $FFFF.
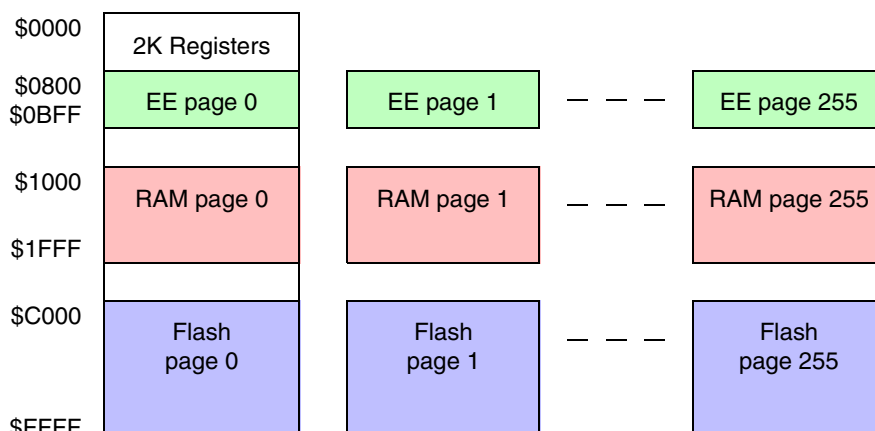
**Figure 4. Generic Logical Memory Map**

The CPU accesses a maximum of 64K bytes. The available memory is expanded by substituting pages of memory into the same logical address space. Thus, theoretically, the CPU can access up to 256 pages of each area of memory, whilst still using a 64K byte map. The pages of memory are selected by special registers within the MMC, namely EPAGE, RPAGE and PPAGE.

# 3    EEPROM Pages

In the example shown in Figure 5, the CPU can access up to 1K bytes of EEPROM in its logical address map. Therefore, only the ten least significant address bits are used, the upper six bits being ignored. The EPAGE register within the MMC module selects which bank of EEPROM is presented to the CPU, by prefixing the CPU address with the contents of the EPAGE register.

For this example, the five most significant bits of the global address have been hard coded. This fixes the EEPROM within an area of the global memory map. Thus, the possible address range for this example is:

```
Memory Address Bits = 001 00PP PPPP PPaa aaaa aaaa
Global Address Range = $10_0000 to $13_FFFF

Key:
P = EPAGE bit
a = CPU address
```
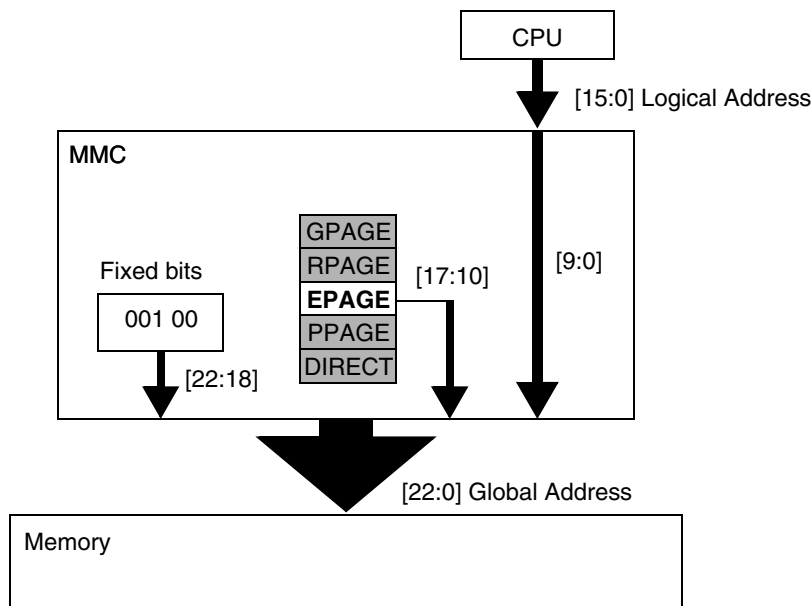
**HCS12X Family Memory Organization, Rev. 1**

**Figure 5. Memory Access Using EPAGE**

# 4 RAM Pages

The 4K RAM in the example shown in Figure 6 is located in the CPU address range $1000 to $1FFF. Bank selection is controlled via the RPAGE register in the MMC.

Here, the twelve least significant bits of the CPU address are used to locate bytes within the 4K byte RAM page. Selection of the RAM page is controlled using the RPAGE register, whose value prefixes the CPU address. The fixed bits are required to complete the 23 address lines passed to the memory. This also fixes the global address range:

```
Memory Address Bits = 000 PPPP PPPP aaaa aaaa aaaa
Global Address Range = $00_0000 to $0F_FFFF

Key:
P = RPAGE bit
a = CPU address
```

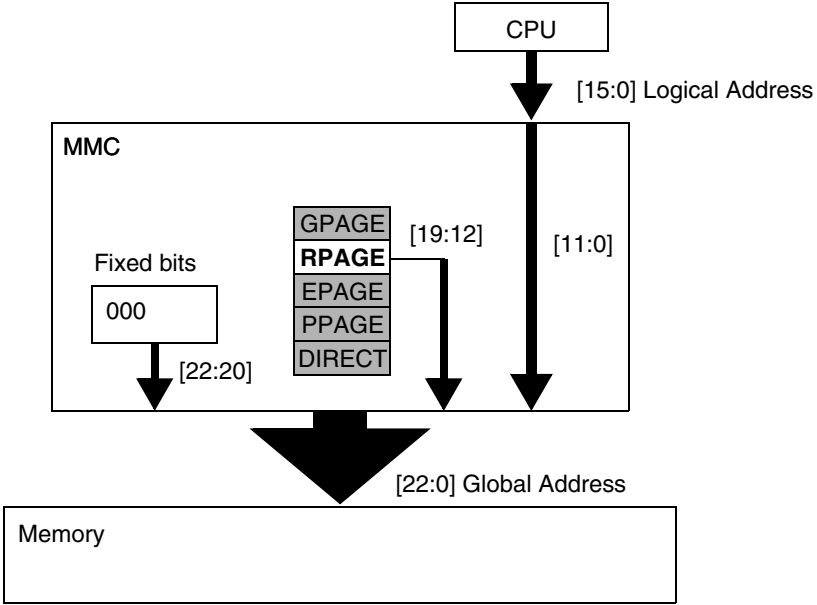**Figure 6. Memory Access Using RPAGE**

# 5    Flash Pages

The last memory technology currently available is Flash memory. In this example (see Figure 7), there are up to 256 Flash pages, each of 16K bytes. Fourteen of the CPU address lines are used to address the 16K byte space. The memory is then expanded using the eight bits of the PPAGE register.

Again, the fixed bit is used to complete the required 23 address lines to the memory, and locates the pages at the extreme of the global memory map.

```
Memory Address Bits = 1PP PPPP PPaa aaaa aaaa aaaa
Global Address Range = $40_0000 to $7F_FFFF

Key:
P = PPAGE bit
a = CPU address
```

The selected page is then accessed within the same logical address range. The RAM and Flash memories have similar addressing schema, controlled by the RPAGE and PPAGE registers, respectively. This type of access can be extremely useful when running multi-threaded operations; each thread can keep the program, variable and constants section completely isolated from other threads.
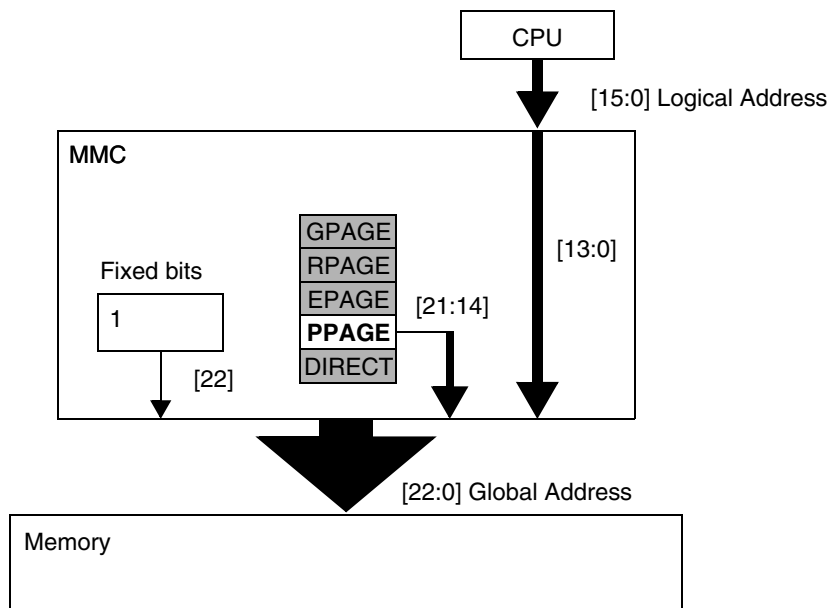
**Figure 7. Memory Access Using PPAGE**

# 6    MC9S12DPX512 Memory Map

The logical and global memory map configuration for the MC9S12DPX512 is shown in Figure 8. This device has 512K bytes of Flash, 32K bytes of RAM, and 4K bytes of EEPROM available.
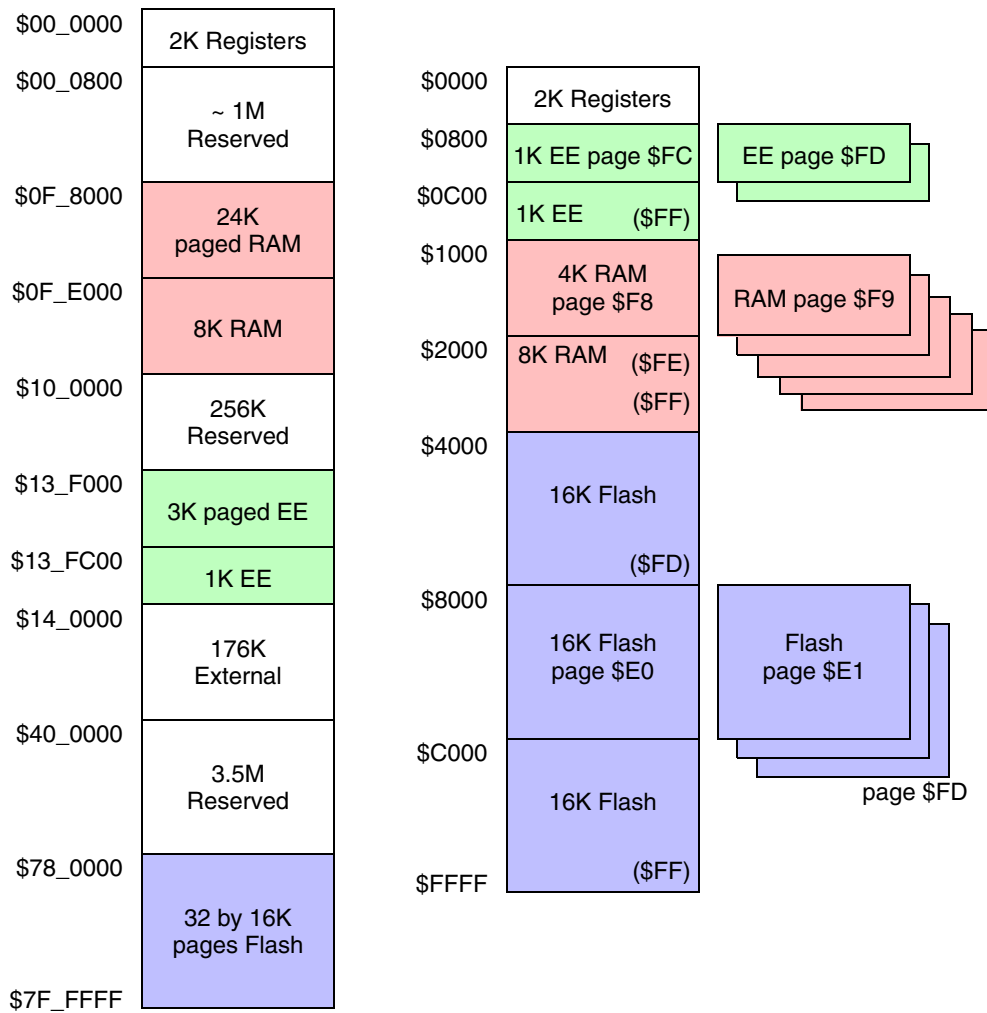
**Figure 8. MC9S12XDP512 Global and Logical Memory Map**

# 7    EEPROM

The logical address space for the paged EEPROM is from $0800 to $0BFF. There are four pages of EEPROM available, each page storing 1K bytes. At reset, the EPAGE register is equal to $FE. The range of implemented values for the EPAGE is from $FC to $FF. Setting EPAGE outside this range will result in non-valid accesses to memory.

**NOTE**

*When the EPAGE register is equal to $FF, the MMC actually points to the unpaged EEPROM at logical address $0C00 to $0FFF.*

In the global memory map, the 4K byte EEPROM is accessible in the linear address range from $13_F000 to $13_FFFF.

# 8    RAM

The logical address range for each 4K byte page of RAM locations is from $1000 to $1FFF.

On the MC9S12XDP512, a total of 32K bytes of paged RAM are available with the RPAGE values of $F8 to $FF. Setting RPAGE outside this range will result in non-valid accesses to memory. On reset, the RPAGE value is set to $FD.

When the RPAGE register is set to $FE or $FF, the MMC actually uses the two non-paged 4K byte blocks at $2000 and $3000 respectively. When the RPAGE register is set to $00, it is possible to write to the registers through the RAM space.

In the global mapping scheme, the RAM is located as a linear 32 byte block from $0F_8000 to $0F_FFFF.

# 9    Flash

On the MC9S12XDP512, there is a total of 512K bytes available within the PPAGE values of $E0 to $FF. Each 16K byte page is at the logical address of $8000 to $BFFF. The reset value for PPAGE is $FE.

When PPAGE = $FD, the 16K byte page at $4000 to $7FFF is mapped to the locations $8000 to $BFFF. When PPAGE = $FF, the range $C000 to $FFFF is mapped to $8000 to $BFFF.

# 10    Direct Page

The last type of access is to the 'zero page' of the CPU, i.e. logical addresses $0000 to $00FF. The 'zero page' is usually reserved for frequently accessed variables; efficient use of the 'zero page' leads to faster execution and smaller instruction code. On the HCS12X family, some of this area is taken up by the registers. The 'zero page' can be moved to anywhere within the global memory map, via the DIRECT register. The DIRECT register can be written only once after reset. Note that moving the 'zero page' means that you cannot access the register values at $0000 to $07FF, through the normal direct access; instead, extended addressing must be used for the registers $0000 to $00FF.

A useful technique is to move the 'zero page' to the paged RAM locations, i.e. at $1000 to $10FF. Then, by using the RPAGE register, each RAM page has a 1K byte 'zero page' within it, giving up to 256 zero pages. On the MC9S12XDP512 there are eight RAM pages; by careful configuration, up to 1280 'zero page' locations are available. For example, the following example code uses direct writes and reads from different pages of RAM.

```
__asm {
  ; assembler to test moving zero page to paged RAM
  ;
  ; set direct page to paged RAM  at $00_1000
  ; extended writes to RPAGE will move the currently
  ; visible paged RAM
  ; then Direct access to Paged RAM for reads/writes

                    ; $0F_B000 is page0 of RAM
      MOVB    #$FB, $16 ;RPAGE
      MOVB    #$00, $10 ;GPAGE
      MOVB    #$10, $12 ;DIRECT = $00_10xx
```

```
                        ; direct write to logical $02 = global $0F_B002
        LDAA    #$A0
        STAA    $02

        MOVB    #$FC, $0016;RPAGE
                        ; direct write to logical $02 = global $0F_C002
        LDAA    #$50
        STAA    $02

                        ;direct read from logical $02 = global $0F_B002
        MOVB    #$FB, $0016;RPAGE
        LDAA    $02     ; should be $A0

                        ;direct read from logical $02 = global $0F_C002
        MOVB    #$FC, $0016;RPAGE
        LDAA    $02                             ;should be $50

                        ;move to RPAGE=$00, accesses now effect registers
        MOVB    #$00, $0016;RPAGE
        LDAA    #$A0
        STAA    $01     ; change PortB
}
```

**How to Reach Us:**

**Home Page:**
www.freescale.com

**Web Support:**
http://www.freescale.com/support

**USA/Europe or Locations Not Listed:**
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

**Europe, Middle East, and Africa:**
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

**Japan:**
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

**Asia/Pacific:**
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: AN2734
Rev. 1
02/2007