Freescale Semiconductor
Application Note

Document Number: AN2615
Rev. 4, 07/2007

S12 and S12XD Family Compatibility

by: Steve McAslan, 8/16-bit Products Division, East Kilbride
Joachim Krücken, 8/16-bit Products Division, Munich

1 Introduction

The 9S12XDP512 MCU provides significant performance improvements over the existing 9S12 family through a combination of increased clock speed and enhanced functionality. Existing S12 users can take advantage of the increased speed of operation of the S12XD family almost immediately due to its very high level of backwards compatibility. Further performance benefits can be gained by optimizing the application design to take advantage of the new feature set.

This document describes the notable differences between the S12 family and the S12XD family. Developers currently designing applications with the S12 family should take note of these differences so they are well placed to take advantage of the S12XD when appropriate. A companion application note (AN3469) describes the compatability between the S12 and the S12XE families.

This document is structured in two sections. The first section (S12XD Enhancements) describes the functional

Contents

freescale semiconductor

enhancements provided by the S12XD, and is most useful for developers embarking on new designs with the S12 or S12XD. The second section (Considerations for Existing S12 Applications) deals with the basic modifications required to allow existing S12 code to run on the S12XD. To simplify understanding, each section steps through the feature set of the MCU, explaining the points to note and the benefits of the S12XD approach.

All differences noted are based on the feature set of the 9S12XDP512. Note that other derivatives of the S12XD family might not feature all of the functions described here. Refer to the appropriate data sheet for details.

The following list summarizes the differences between the S12XD family and the S12 family.

- 40MHz operation
- 80MHz XGATE peripheral coprocessor
- Extended CPU instruction set
- Programmable eight level interrupt controller
- Enhanced Memory Management Controller
- New four channel Periodic Interrupt Timer
- Improved Enhanced Capture Timer with modulus prescaler options
- Non-multiplexed 8MB expanded memory bus
- New low-power RC trigger and fast recovery from STOP modes
- Decimal prescaler for Real Time Interrupt module
- Improved SCI featuring hardware bit manipulation for LIN
- Enhanced trigger source options for Analog to Digital Converters
- Amplitude controlled Pierce oscillator
- Wider and deeper debug module

2 S12XD Enhancements

This section is intended for developers creating new applications for the S12 who wish to maximize compatibility with the S12XD. It is also useful for developers creating applications for the S12XD, who are familiar with the S12 family. It describes how the new functionality of the S12XD differs from the S12 and how to take advantage of these new features.

2.1 Software Architecture

In many modern embedded systems, a significant portion of the microcontroller performance is taken up dealing with real-time events. External stimuli and internal modules generate interrupts that require processing time that must be made available for the system to operate correctly. In the S12, the only resource available for this activity is the CPU, and care must be taken to ensure that both the overall processing requirement and the response time for events is met as expected. The S12XD, however, has a special module called the XGATE that is designed for this kind of activity.

The XGATE is a new programmable module that can respond to any peripheral interrupt, gather data from the peripheral, process it as required, and store it for use by the CPU. In addition, it can take data from the CPU and pass it to a peripheral, and can pass data from one peripheral to another. The XGATE can accomplish all this activity, without intervention by the CPU, and running at twice the clock speed of the CPU. This makes the XGATE ideal for handling the real time requirements of most applications.

Since the XGATE is fully programmable, it can also be used to create “Virtual Peripherals”. For example, the 9S12XDP512 contains six SCI modules; by using timer modules on the MCU, the XGATE can add further SCIs with similar performance, and with no impact to CPU performance. This approach also allows the creation of queued peripherals with flexible buffer structures.

To take best advantage of the benefits of the XGATE, it is necessary to design the software architecture in such a way that real-time handling code is separate from linear application code. In fact, most software is already written this way by placing real-time code into interrupt service routines (ISR). In general, it is advisable to minimize the amount of code placed in an ISR, so that the response time of the CPU to other interrupt events is optimized. This is true, to a lesser extent, of the XGATE. However, the XGATE’s architecture and fast execution time do allow significant functionality to be performed in response to an interrupt.

NOTE

Users developing code on the S12 for later use on the S12XD can create and test the functionality of a real-time interrupt handler by coding ISRs for the S12 CPU and then move this functionality to the XGATE when ready. The interrupt module on the S12XD can be used to direct the interrupt request directly to the XGATE.

2.2 CPU and Instruction Set

The S12XD features the new S12XD CPU. This CPU includes enhancements to the programmer model, stacking operations, and the instruction set.

2.2.1 Programmers Model

The S12XD CPU features an enhanced Condition Code Register (CCR). This register has been extended to 16 bits to allow stacking of the interrupt priority.

The additional bits in the CCR are shown in Figure 1. IPL[2:0] indicate the interrupt level of the CPU prior to the current interrupt.

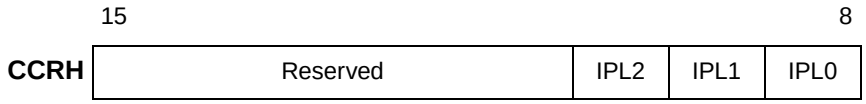

Figure 1. High Byte of Condition Code Register

The CPU automatically updates the value of IPL[2:0] to the value of the interrupt currently being serviced.

2.2.2 Interrupt Stacking Operation

Since the CCR has been extended to two bytes from one byte; this in turn causes the interrupt stack frame to be extended by one byte from nine bytes to ten bytes Therefore all stack relative accesses are modified by one byte.

Figure 2 gives an example of a stack frame on an S12 after an interrupt has occurred.

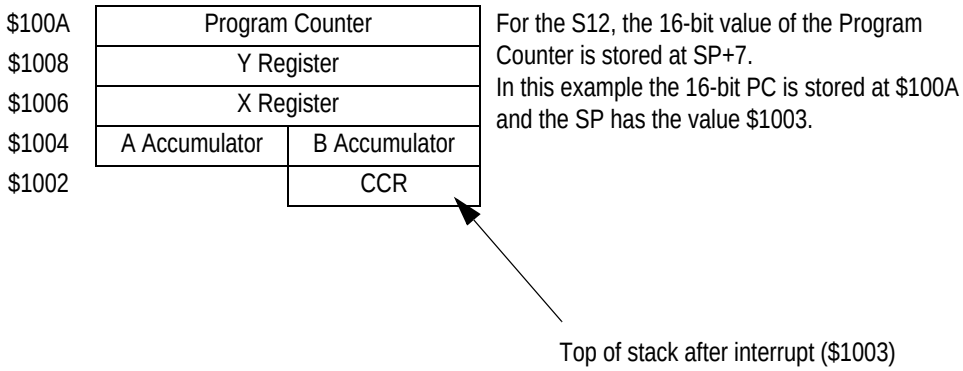

Figure 2. Stack Frame Example for S12

Figure 3 gives an example of a stack frame on the CPU after an interrupt has occurred.

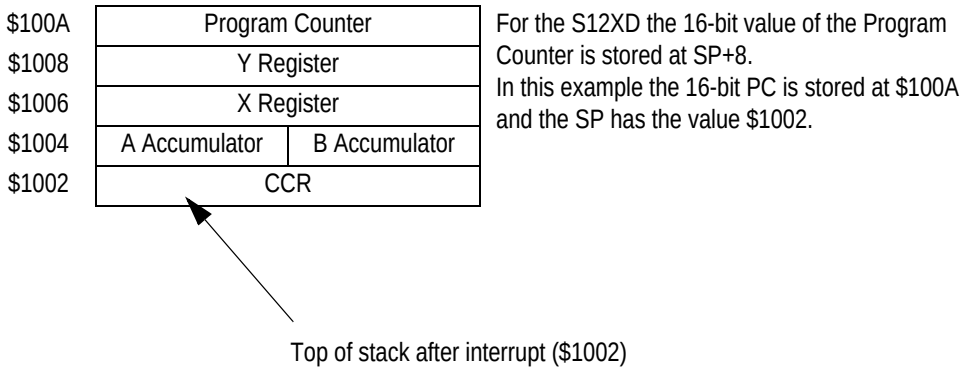

Figure 3. Stack Frame Example for S12XD

In practice, the requirement to extract information such as the Program Counter from an interrupt stack frame is an unusual activity (typically related to debug tools or perhaps task schedulers). Therefore for the vast majority of users this difference between the S12 and S12XD will have little impact.

NOTE

Users developing code on the S12 for later use on the S12XD need to be aware of the maximum stack size and any unusual requirements such as that described. Since each interrupt stack frame is one byte larger on the S12XD, users must take account of this in increasing the memory reserved for the stack when moving to the S12XD.

2.2.3 Instruction Set

The S12XD features an enhanced instruction set over the S12. All of the existing S12 CPU instructions are retained.

There are four classes of new instructions

1. New 16-bit, where only an 8-bit accumulator operation existed
2. New memory access instructions, allowing access to linear banks of up to 64 Kbytes
3. New instruction designed to optimize semaphore handling
4. New addressing modes for MOVe instructions

Class 1 improves the data manipulation capabilities of the CPU by allowing direct operation on larger data sizes. On the S12, most arithmetic and logical operations, such as addition, can only take place by using the A, B or D accumulators. The S12XD extends this capability to the X and Y registers and adds new instructions for the D register. All arithmetic and logical functions using the A or B accumulator will now have a 16-bit counterpart using the X and Y register. New instructions of this type are: ADE (add with carry), ADD (add without carry), SBE (subtract with carry), DEC (decrement) and INC (increment), SUB (subtract without carry), AND (logical AND), BIT (logical bit test), OR (logical OR), EOR (logical EXCLUSIVE OR), NEG (two's complement), COM (one's complement), CLR (clear register), TST (test register), LSL (logical shift left), ROL (rotate left), ASR (arithmetic shift right), LSR (logical shift right), and ROR (rotate right). These new instructions have the same addressing modes as their 8bit counterparts.

To improve the 32-bit capability of the D-Accumulator, ADED (add with carry) and SBED (subtract with carry) are added. In addition, the CPU provides a set of compare instructions carrying forward the carry and also the zero flag (CPED, CPEX, CPEY, CPES). This improves the capability to perform 32-bit compares.

While the existing architecture allows 8-bit read-modify-write instructions, the S12XD extends this capability to 16-bit words and provides NEGW (two's complement), COMW (one's complement), DECW (decrement 16-bit), INCW (increment 16-bit), RORW (rotate right), LSRW (logical shift right), ARSW (arithmetic shift right), ROLW (rotate left), LSLW (logical shift left), CLRW (clear memory) and TSTW (test memory). Addressing modes are the same as for their 8-bit counterparts. In general, these new 16-bit operations allow significantly faster manipulation of data compared to the S12 CPU.

Class 2 provides access to a new mode available on the S12XD Memory Management Controller . This allows access to any 64K byte page in global memory based on a new MCU register called GPAGE. The new instructions include all the available addressing modes and concatenate the GPAGE register with the 16-bit address data. Global instructions are available for the following instructions: LDAA (load accumulator A), LDAB (load accumulator B), LDD (load accumulator D), LDX (load X register), LDY (load Y register), LDS (load stack pointer), STAA (store accumulator A), STAB (store accumulator B), STD (store accumulator D), STX (store X register), STY (store Y register), and STS (store stack pointer).

The GPAGE register is seven bits wide, so that global memory runs from $00_0000 to $7F_FFFF, and each location is accessible with a single instruction from anywhere in a program (once the GPAGE register is configured for that 64 Kbyte page).

NOTE

Users developing code on the S12 for later use on the S12XD should carefully note areas of opportunity/requirement to use this feature and modify their code, and their compiler and linker settings, once using the S12XD.

Class 3 allows more efficient use of semaphores, which are important for real time operating systems (RTOS) and for sharing resources between the CPU and the XGATE. The new instruction is BTAS (bit test and set). Since this is a single instruction, it cannot be interrupted; therefore, it is useful when requesting access to resources.

Resources are usually locked via a status bit in RAM — when the bit is set the resource is in use. On the S12, users must take care that two tasks cannot both appear to have allocated the resource. This can occur if one task interrupts another immediately after a bit test instruction. Therefore, tasks typically disable interrupts while checking and allocating resources. The BTAS instruction removes this need, as it tests and sets the resource bit in a single instruction step. BTAS follows the same syntax and allows the same addressing modes as the BSET instruction, except that the test is based on the original data and not on the data written back.

A typical use for a BTAS instruction is shown in Figure 4.

```
BTAS $1020, #$20
BNE ResourceNotAvailable

<ResourceLocked>
```

Figure 4. Typical Use of BTAS instruction

Class 4 is designed to improve the opportunity for compilers to use the memory-to-memory move instructions by allowing the use of all relevant S12XD addressing modes, and not only those fitting in a single postbyte xb. See the CPU block guide for more information on the newly added modes.

2.3 Interrupt Controller

The S12XD features a new interrupt controller module that allows up to seven levels of interrupts (I-bit) to be available to the user. The addition of this module means that the HPRIO (High Priority) interrupt function in the S12 is completely removed. The XIRQ, SWI, BDM, unimplemented opcode, and system reset interrupts are available as before. In addition, a new interrupt vector is introduced to allow handling of ‘spurious’ interrupts that can occur if an interrupt source is removed before the interrupt is handled.

The S12XD also provides improved detection of invalid software operations that access areas of the MCU´s memory that contain no resources. This enhancement applies in single-chip mode and causes a

reset if the CPU accesses a memory location that does not address an on-chip memory or peripheral module. The reset vector fetched is a system reset at 0xFFFE.

Users who wish to take most advantage of the Interrupt Controller should refer to the block guide for the module. This section is intended to give an idea of the possibilities for the module and development approaches when starting from the S12.

On the S12, the priority of any interrupt is determined by its position in the interrupt vector table. Vectors closer to the top of memory ($FFFF) have a higher priority than those lower down. An exception to this is that the HPRIO register can be used to promote a single interrupt above its vector position. However, this only applies to point at which interrupts are taken — once an interrupt is being serviced, there is no way to automatically tell if any other pending interrupt has a higher priority than the current level.

On the S12XD, each interrupt source can be allocated one of seven possible interrupt levels. These levels can be changed at any time and are complemented by an eighth level, which disables the interrupt.

An interrupt can be taken only if it is enabled, and the global mask (I-bit) is clear, and if it is higher than the current working interrupt level. The CPU is aware of and stacks the interrupt level at which it is working. For example, this means that a level 1 interrupt cannot be taken by the CPU if it has not returned from processing a level 5 interrupt, even if the I-bit is clear. Conversely, a level 5 interrupt can be taken by the CPU if it is working at level 1. Of course, the level 5 interrupt will not be taken if the I-bit is set. As with the S12, the I-bit is set automatically on entry to an interrupt, so the code within each Interrupt Service Routine (ISR) must explicitly clear the I-bit using the CLI instruction, if nested interrupts are desired. Customers who do not want to use nested interrupts still benefit from the seven different priority levels, as the highest priority interrupt will always be selected from those pending.

When the CPU returns from an interrupt, part of its new functionality is to recover the interrupt level at which is was working before the interrupt was taken. This is stored in the upper byte of the Condition Code Register (CCR) (see Section 2.2.1, “Programmers Model”).

An additional feature of the Interrupt module is the ability to specify the location of the interrupt vector table in memory. This is achieved by using the Interrupt Vector Base Register (IVBR). The IVBR specifies the top eight bits of the vector table 16-bit address, and can be changed at any time. The vector table always exists in the main 64 Kbyte map of the CPU. This ability to move the vector table allows users to have multiple vector tables for multiple mode operating systems, debugging systems, and bootloaders. The IVBR is set to $FF out of reset, for compatibility with the S12.

Finally, the interrupt module has the ability to route interrupts to and from the XGATE. Each interrupt has the option of being allocated to the CPU or the XGATE and given a priority. The XGATE will service the highest priority request first; however, it is not possible to interrupt an XGATE thread that is already running.

The interrupt module also handles interrupts from the XGATE. These typically occur when the XGATE has completed some function and requests the CPU to perform some action. All XGATE interrupts are given the same level of priority, however, again this priority is programmable in the Interrupt Module. Figure 5 shows the architecture of the Interrupt Module.

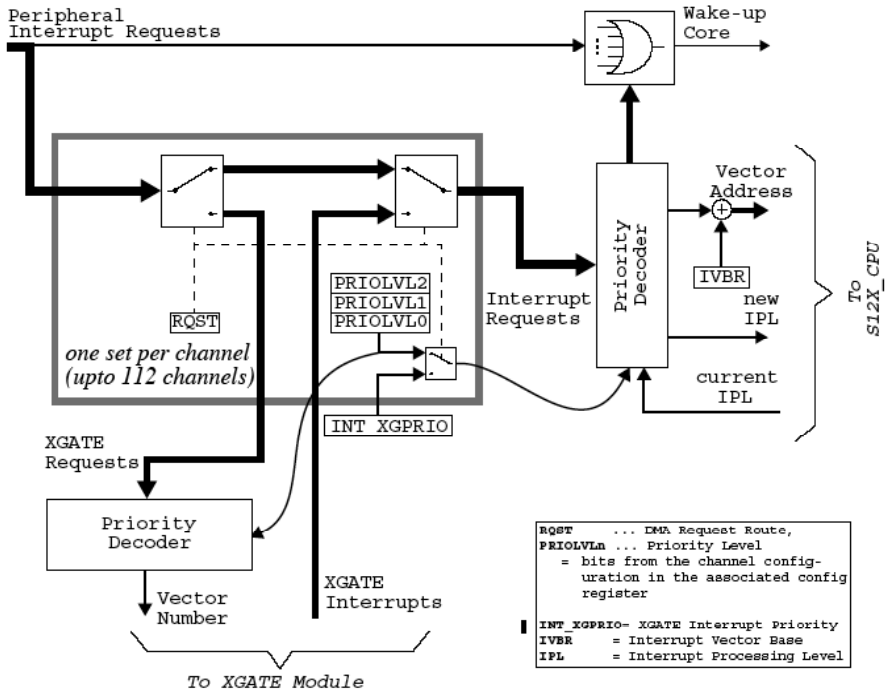

Figure 5. Interrupt Module

Figure 6 illustrates a typical profile of the interrupt processing levels possible when using the interrupt controller.

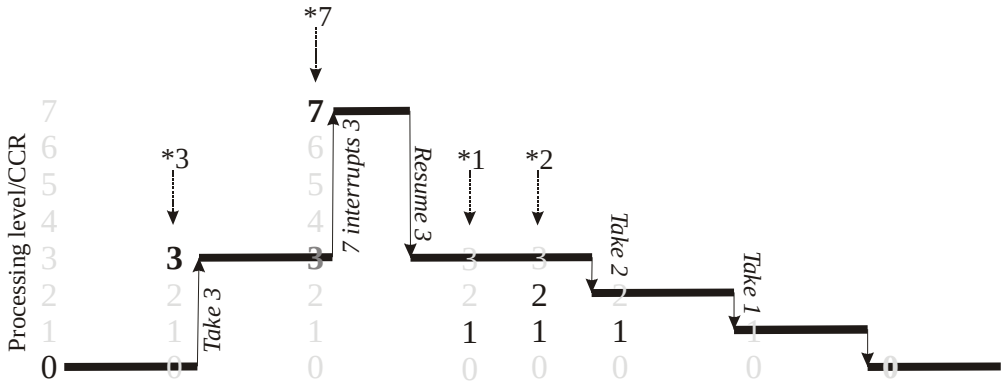

Figure 6. Interrupt Profile

In this example, the CPU is initially not executing an interrupt service routine. At the point marked by *3, an interrupt with a priority level of three is recognized by the CPU and the CPU begins executing the Interrupt Service Routine (ISR). While executing the level three ISR, an interrupt with a level seven priority occurs and begins to execute at the point marked by *7. This ISR runs to completion and the level three ISR then resumes execution.

During the remaining execution time of the level three ISR, a level one and level two interrupt, marked by *1 and *2 respectively, occur. Note that because these interrupts have a lower priority than the currently executing ISR, their ISRs are not executed, instead the interrupts simply remain pending.

At the completion of the level three ISR, the level two ISR is executed first because it has the highest priority of the two pending interrupts. Finally, at the completion of the level two ISR, the level one ISR executes and runs to completion.

Note that for interrupt nesting to occur as shown in this example, the I-bit in the CPU's Condition Code Register must be cleared at the start of each ISR.

Users should take care not to set the interrupt level to 0 in the interrupt module. Doing so will disable all interrupts associated with a peripheral regardless of the settings of the peripheral's local interrupt enable bits.

NOTE

Users developing code on the S12 for later use on the S12XD can develop interrupt service routines (ISRs) as normal. It is not possible to alter the interrupt priority (except for the HPRIO option) on the S12, so a true interrupt emulator for the S12XD cannot be developed. Possible approaches to preparing for the S12XD include: polling important interrupt sources while in an ISR and enabling interrupts as required, or simply enabling all interrupts to occur and handling less important ones as quickly as possible (see AN2617: “A Software Interrupt Priority Scheme for HCS12 Microcontrollers” for a possible scheme).

2.4 XGATE

The XGATE is a completely new module on the S12XD. As previously described, it is a highly integrated but separately programmable coprocessor targeted at I/O management. The XGATE can access all of the peripheral modules and RAM, manipulate the data, and interact with the interrupt module, all independently of the CPU. The XGATE can also read part of the FLASH.

Specifically, the XGATE is a 16-bit RISC processor running at double the CPU bus speed. It accesses RAM and peripherals during the “half cycle” that the CPU uses to prepare or complete its instruction. This allows a typical sustained performance of 70 MIPS. For those cycles where the CPU does not access the RAM, then this performance is further increased.

The instruction set is optimized for handling data movement and logic operations. See Figure 7 for a block diagram of the programmer's model. There are eight 16-bit registers, of which R0 and R1 have special functions.

Code for the XGATE is stored in RAM and is initiated by a service request from the interrupt controller or by software. XGATE code exists as threads that operate on the data space pointed to by register R1. This register is initialized automatically by the service request. Using one of the XGATE registers to point to a thread's variables and data allow multiple peripherals to share a single instance of the XGATE code in RAM. For example, a simple XGATE thread may copy bytes from an SCI into an area of RAM. By writing this code carefully, it is possible to have the same thread handle all six SCIs on the 9S12XDP512, thereby saving program space. It is equally possible to have different threads for each SCI.

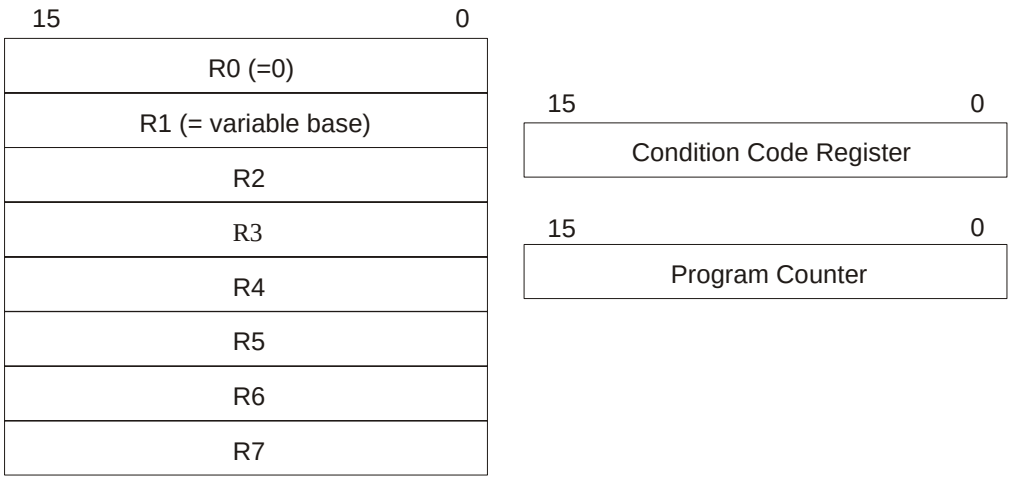

Figure 7. XGATE Programmer's Model

These threads are typically initiated by the interrupt controller, which has the capability of directing interrupts to either the CPU or the XGATE. The interrupt source may be a peripheral module, such as a timer channel, or the CPU may initiate a service request through the XGATE software trigger register. The XGATE itself has a configurable vector table containing one entry for each interrupt source. Each entry includes a pointer to the start of the XGATE thread and a pointer to the base of the data space used by the thread. The XGATE also has the capability to generate an interrupt (see Section 2.3, “Interrupt Controller”).

The addition of a second independent CPU to the S12XD can present a number of software design issues. Since both CPUs effectively have simultaneous access to the RAM and peripherals, the software must be designed in such a manner that both CPUs do not attempt to utilize the same resource at the same time. To assist the software designer in this task, the XGATE module provides a set of eight dedicated semaphore flags.

Additionally, since the XGATE code resides in RAM the provides a configurable protection system that can be used to prevent accidental corruption of the XGATE or CPU code and variables. The protection mechanism permits the on-chip RAM to be partitioned in to three variable side areas: XGATE RAM, shared RAM, and CPU RAM.

2.4.1 XGATE RAM

The XGATE RAM is typically used to store XGATE code. The CPU will copy the XGATE code into RAM then set a protection bit. Once the process is complete, the protection mechanism is enabled by the CPU preventing it from performing any further writes to this area. Note that the CPU is still permitted to read from this area.

2.4.2 Shared RAM

The shared RAM area allows full read/write access to both the XGATE and CPU. Typically, this area of RAM is used to exchange data between the two CPUs.

2.4.3 CPU RAM

The CPU RAM area is defined for exclusive access by the CPU to prevent the XGATE from unintentionally corrupting CPU data.

The XGATE is a formidable addition to the feature set of the S12XD family. Careful design will allow its significant performance to be released in a wide variety of applications.

NOTE

Users developing code on the S12 for later use on the S12XD can partition code functionality as described in the Software Architecture section. Structuring RAM allocation in a manner similar to the segments described above will simplify integration and conversion of the code.

2.5 Memory Management Controller

The S12XD contains a new MMC that provides features not available on the S12, and creates a standard memory map that is compatible for devices with large or small memory maps.

The standard memory map provided by the MMC defines the locations in memory of all five memory blocks: registers, RAM, EEPROM, FLASH and external peripherals. In doing so, it fixes, not only the local 64K bytes accessible directly to the CPU, but also the full 8 Mbyte global memory space supported by the S12XD. For all S12XD devices, there is a direct correlation between the address of a memory block in the local map ($0000 to $FFFF) and in the global map ($00_0000 to $7F_FFFF). In the global map, the most significant byte corresponds to the value in the GPAGE register.

NOTE

Users developing code on the S12 for later use on the S12XD should consider carefully the best approach for accessing memory on the S12XD. In many cases, use of the GPAGE register will be a more efficient way of accessing memory. This is particularly the case where access is required on a large data array. Cases such as these may be coded using a macro that will use a simple GPAGE load on the S12XD and an alternate approach on the S12.

2.5.1 RAM, EEPROM and Register Mapping

In creating the new MMC, some features found on the S12 were removed. This includes removing the capability to change the location of memory blocks. This means that all memory is always usable, unlike on the S12 where portions of RAM or registers can overlie the EEPROM and prevent its use. Therefore, the S12XD no longer features the INITRM, INITREG and INITEE registers that defined the starting location of the RAM, registers and EEPROM respectively. The functionality provided by these registers is implemented in an alternate way on the S12XD.

The INITREG register has no direct replacement function in the S12XD because the registers are always fixed at address $0000. However, since in practice users only moved the register block because they had a requirement to use the direct page area of the map, an alternative solution is offered. On the S12 the direct page is fixed at $0000 to $00FF, and in that space the optimized direct page addressing mode may be used

by the CPU. On the S12XD, the need to move the registers is removed and, instead, the ability to move the direct page location is introduced. The S12XD has a new register called DIRECT, which defines the upper eight bits of the direct page. The new direct page address is therefore $ZZ00 to $ZZFF, where $ZZ is the contents of the DIRECT register. The DIRECT register is writable only once.

The EEPROM block is fixed in the local map at address $0800 to $0FFF. This 2 Kbyte block size is divided into two sections. The upper 1 Kbyte from $0C00 to $0FFF is always present in the memory map. The lower 1 Kbyte acts as a page window, in a similar fashion to that offered in the FLASH on the S12 and S12XD. Like the FLASH window the EEPROM window has a control register to determine the 1 Kbytes of EEPROM that is to be visible. For the EEPROM this register is called EPAGE (c.f. PPAGE). On the 9S12XDP512 there is a total of 4 Kbytes of EEPROM so there is a single fixed 1 Kbyte and three further swappable pages of 1 Kbyte. The fixed 1 Kbyte space can also be made visible in this window should this be required.

The EEPROM is fixed in the global map at address $10_0000 to $13_FFFF giving a maximum EEPROM size of 256K bytes. On the 9S12XDP512 the EEPROM exists at address $13_F000 to $13_FFFF.

The RAM block is fixed in the local map at address $1000 to $3FFF. This 12 Kbyte block size is divided into two sections. The upper 8 Kbytes from $2000 to $3FFF are always present in the memory map. The lower 4 Kbyte acts as a page window in a similar fashion to the EEPROM and FLASH. The RAM has a control register to determine the 4 Kbyte that is to be visible. For the RAM, this register is called RPAGE. On the 9S12XDP512, there is a total of 32 Kbytes of RAM, so there is a single fixed 8 Kbyte block and six further swappable pages of 4 Kbytes. The fixed 8 Kbyte space can also be swapped into this window as two 4 Kbyte pages, should this be required.

The RAM is fixed in the global map at address $00_1000 to $0F_FFFF, giving a maximum RAM size of just under 1 Mbytes. On the 9S12XDP512, the RAM exists at address $0F_8000 to $0F_FFFF.

2.5.2 FLASH Mapping

The S12XD has a more linear arrangement of FLASH pages than the S12. This leads to a slight change in the selection of FLASH pages using the PPAGE register. No change to code is required; however, depending on use, the linker configuration may have to be slightly modified.

The FLASH is fixed in the global map at address $40_0000 to $7F_FFFF, giving a maximum FLASH size of 4 Mbytes. On the 9S12XDP512, the FLASH exists at address $78_0000 to $7F_FFFF.

The PPAGE register selects a block of 16 Kbytes to be visible within a window from $8000 and $BFFF in the local map — this is unchanged from the S12. The PPAGE value maps into the global map such that page $00 corresponds to the 16 Kbytes at the lower addresses in the global map ($40_0000 to $40_3FFF). Page $01 corresponds to the next 16 Kbyte page ($40_4000 to $40_7FFF), and so on until page $FF, which corresponds to the top 16 Kbytes in the global map ($7F_C000 to $7F_FFFF). In addition to the PPAGE window, and like the S12, the S12XD has 32 Kbytes fixed to certain addresses in the global map. This means that these FLASH pages are always visible in the local map. In the local map, the 16 Kbytes from $4000 to $7FFF are fixed to the global map from $7F_4000 to $7F_7FFF, and the 16 Kbytes from $C000 to $FFFF are fixed to the global map from $7F_C000 to $7F_FFFF

This linear arrangement is different to that found on the S12, where the PPAGE register maps pages in a slightly different order. The PPAGE register is limited to six bits on the S12, and the second top two pages

are swapped in order. For the top 64 Kbytes of FLASH, page $3F will map to the top page of the 64 Kbytes ($xx_C000 to $xx_FFFF), as on the S12XD. However, the second bottom page ($xx_4000 to $xx_7FFF) is accessed at PPAGE $3E, unlike the S12XD where the value of $FD would be used. Typically then, the first page of FLASH used in the window is from page $3D, whereas on the S12XD this would be $FE.

In practice, this means that the FLASH page fixed at address $4000 to $7FFF on the S12 is mapped into the PPAGE window using a value of $3E, where the same fixed page on the S12XD is mapped using the value of $FD. Of course, since the FLASH pages are not accessible on the S12 through any other method, the actual topology of the FLASH is irrelevant when selecting pages.

NOTE

Users developing code on the S12 for later use on the S12XD must ensure that any code accessing the fixed page of FLASH at address $4000 within the PPAGE window uses the correct PPAGE value. This is largely the duty of the linker in use. It is unlikely to be relevant during code execution, since this is an unusual use of FLASH.

2.5.3 Global Memory Mapping

As discussed in previous sections, the MMC maps all on-chip resources into a unified 8 Mbyte global, linear address space. Using the new global load and store instructions in conjunction with the GPAGE register, on-chip Flash, RAM, EEPROM, I/O registers and even external memory may be accessed in a unified manner. The primary benefit of this alternate address space is that unlike the S12, the S12XD allows code in any page of FLASH to access data in any other page in FLASH, removing the need for page swaps via intermediate memory. In addition, large data tables (<= 64K) may be accessed by code executing from anywhere in the Flash, RAM or EEPROM.

NOTE

Users developing code on the S12 for later use on the S12XD must implement this type of function as a page swap on the S12, but can simply replace this with a global access instruction on the S12XD

2.6 Expanded Bus interface

The expanded bus interface on the S12XD is modified from the S12 to allow use of the increase in bus speed and to provide a glueless interface to asynchronous memories, etc. The bus is non-multiplexed and features a 23-bit address bus, 16-bit data bus, three chip selects, read and write enables, wait control input line, and enable clock. These features allow the expanded bus to be used in many applications with no further glue logic.

With the S12XD expanded bus, the E clock is now considered a free-running clock. It can be output at the internal bus speed, divided by 2, 3 or 4, and can be stopped completely. The minimum number of clocks for the external interface is two cycles. The E-clock and R/W signal functionality is replaced by RE (read enable) and WE (write enable) signals.

Similarly to the S12, slow peripherals can be accommodated on the expanded bus by the insertion of up to three wait states on the interface timing. This is an acceptable solution if the peripheral has a fixed access time.

However, there are peripherals, such as graphics controllers, that have varying access times. To support connection to such peripherals the expanded bus also features the EWAIT signal. By connecting the peripheral to the EWAIT input, it can suspend the CPU operation until it is ready to accept/write the required data. The inserted wait states continue indefinitely until the EWAIT signal is released.

NOTE

Users developing hardware on the S12 for later use on the S12XD can take several approaches depending on the type of external peripheral devices. Most important is to examine the features on the S12XD and decide how well supported the peripheral is with the new expanded bus. If the peripheral requires little or no logic then a good approach is to design the S12 interface such that the logic builds some S12XD functionality and can therefore be directly removed when converting. If the peripheral continues to require a complex logic interface, then options such as programmable logic devices may be more effective. In either case, the de-multiplexing logic required for the S12 must be removed for the S12XD design.

2.7 Debug Module

The debug module on the S12XD is extended to support the new features of the processor including the full 8 Mbyte memory map and the XGATE.

Users developing applications on the S12 for later use on the S12XD can be sure that the level of support provided by the debug module is fully extended to support the new features on the S12XD.

2.8 Oscillator

The S12XD features an oscillator module that is different from that provided on the S12. The oscillator module provides an amplitude controlled Pierce configuration. The S12 family features either an amplitude controlled Colpitts configuration, or a choice of an amplitude controlled Colpitts or a full amplitude Pierce configuration.

The Pierce configuration improves on the current and EMC performance of the Pierce on the S12 and approaches the current supply requirements of the Colpitts.

NOTE

Users developing hardware on the S12 for later use on the S12XD should begin with the Pierce option, unless low power is an important design consideration. If the low-power option is required, then the Colpitts option (available only on the S12) should be chosen. In either case, consideration should be given to the final layout such that changes can be made as required to suit the S12XD.

As with all oscillator designs, the choice of external components and the layout of the PCB are critical. It is strongly advised that the user carefully consults the advice given the in block guide of the Oscillator.

2.9 Enhanced Capture Timer

The ECT on the S12XD contains improvements that allow more precise timing events to be created or measured. The ECT remains fully backwards compatible with the S12 but, if the user chooses, then prescalers with much finer resolution may be selected.

The new prescalers apply to the main counter, the modulus down counter, and the input capture filter delay counter. These prescalers on the S12 allow steps of 1, 2, 4, 8, 16, 32, 64, and 128. The prescalers on the S12XD allow steps of 1, 2, 3, 4, 5, 6 up to 256.

NOTE

Users developing applications on the S12 for later use on the S12XD should review the range of timing values required. In those cases where the S12 is unable to accurately provide a particular time interval, a smaller time tick may be implemented with a separate software counter. The additional software may then be removed and replaced with a new prescaler value on the S12XD. The new prescalers can also be used to maintain time intervals on an S12XD when running at a higher bus speed than the original S12.

2.10 Real Time Interrupt

Like the Enhanced Capture Timer, the RTI timer also features an enhanced prescaler. Like the ECT, it is fully backwards compatible.

The S12XD RTI features an option to use a decimal rather than a binary prescaler. This is useful where, for example, a 5 ms tick is required from a 4 MHz oscillator. Using a binary divide, the closest tick value is 5.12 ms (= 4 MHz/(5*4096)), whereas the decimal prescaler can achieve exactly 5 ms (= 4 MHz/(2*10000)).

The decimal RTI prescaler also allows divisions up to 15 times greater than the binary prescaler.

NOTE

Users developing applications on the S12 for later use on the S12XD should consider uses for the new prescaler on the RTI. Where a time tick unavailable on the S12 is required, users may consider using an alternate timing method (e.g. smaller RTI divider and separate software counter), and replacing this overhead with a simple write to the RTI prescaler on the S12XD.

2.11 COP — Watchdog

The COP rate and type (window or non-window COP) are loaded from the FLASH during the reset procedure. Therefore the COP can be enabled out of reset without CPU intervention for safety critical systems. Full backwards compatibility to the existing COP is maintained when the reserved FLASH byte is erased.

2.12 Periodic Interrupt Timer

The PIT is a completely new module for the S12XD. The user can use the PIT to create interrupts or trigger the ADC peripheral at very precise time intervals. The PIT operates independently of the Enhanced Capture Timer and allows 24-bit count depths. This provides extremely large count values for long timeout periods or shorter periods with high resolution. The PIT is a completely internal module which does not take up any external pins.

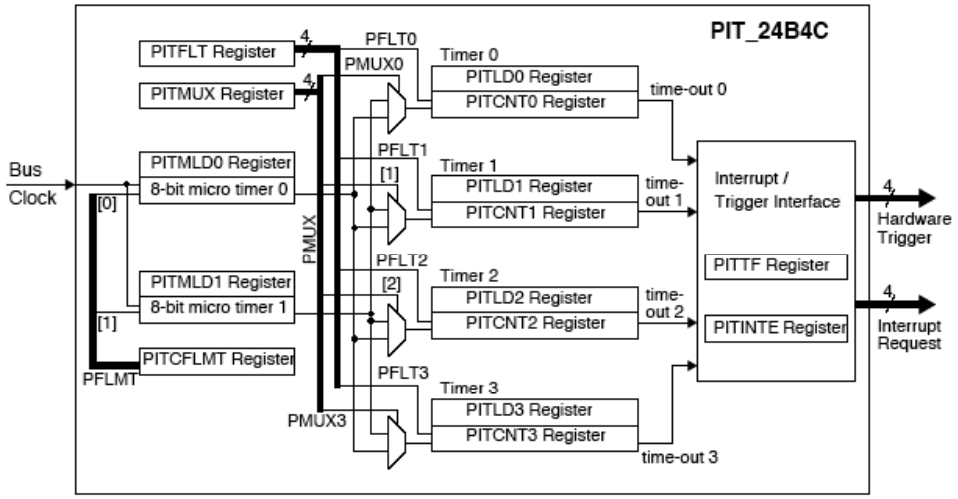

Figure 8. Periodic Interrupt Timer

As can be seen in Figure 8, the PIT is implemented as two stages with four 16-bit modulus down-counters and two 8-bit modulus down-counters. Each 16-bit counter may be driven from the output of either 8-bit counter. This allows for a wide variety of timeout periods that may or may not have an integer relation to each other. For example, if an application has the need to execute several tasks at different time intervals, then interrupts of 1 ms, 4 ms and 10 ms are possible, in addition to a regular 823 µs tick (to activate an A/D conversion, for example). All of these timeouts are possible (at 40 MHz bus clock), with no further intervention by the CPU. Equally, interrupts of 21.75 µs, 3.93 ms, 29.1 ms and 0.419 s could be generated (at 40 MHz bus clock).

NOTE

Users developing code on the S12 for later use on the S12XD can use the existing output compare functions on the ECT to create the interrupts at approximate intervals to these. The ECT has no way to trigger the ADC directly, so this function would have to be implemented in software.

2.13 Miscellaneous Modules

The following modules have minor changes, which should be borne in mind when developing a new application that is intended to run on S12XD at some point. Some of the new features are not trivial and may make the S12XD more suitable for particular applications.

2.13.1 Analog to Digital Converter

On the S12XD, both of the ADC modules can be triggered from additional sources. The new triggers are in addition to the conversion modes on the S12. On the 9S12DX512, the ADC can be triggered from PWM channel 0 or 2, PIT timer channel 0, or any of the external ADC pins.

2.13.2 SCI

The SCI features new hardware support for the following: IrDA protocol, LIN detection of break signal and bus signal collisions, wake up interrupts on active edges, and alternate pin polarity on the receive and transmit pins. The SCI is fully backwards compatible with the S12.

2.13.3 msCAN

The msCAN module provides a new manual control for recovery from Bus-off conditions. In the manual mode, the Bus-off recovery is under software control not earlier than after 128 * 11 received bits. The S12 automatic recovery after 128 * 11 received bits is still available, and is the default configuration.

2.13.4 Background Debug Module

The BDM module on the S12XD includes changes to account for the new Memory Management Controller and also to take advantage automatically of the blank check capability of the FLASH and EEPROM hardware interfaces.

2.13.5 Low Power and STOP Mode Support

The S12XD features new support for those applications where low power is important and frequent use of the STOP mode is required.

Many applications require very low power usage over an extended period, while not losing the ability of the processor to respond to external events. STOP mode is very often used for this type of application, and the S12XD provides two new features to reduce power consumption to a minimum:

- Fast response to external events by starting external oscillator only as required
- Low-power internal interval timer to allow regular timed wake-ups

Fast response from STOP mode is provided by allowing the user to run code from the internal VCO very soon after an event is detected. Since the VCO begins oscillating very quickly, this avoids the delay in starting the external crystal circuit. In many cases, the processor can examine the event and a full recovery can be avoided if not required. Should a recovery be required, then user code simply restarts the external oscillator and begins running code as normal.

Where a regular check on external activity is required, the S12XD can make use of a low-power internal RC oscillator with programmable time ticks from 0.2 ms to 819 ms. When enabled, the RC oscillator wakes the CPU at the specified interval and allows user code to examine the state of the application. Again the user can decide to start up using the VCO or the external crystal.

3 Considerations for Existing S12 Applications

This section is intended for developers who wish to simply convert an existing S12 design to operate on an S12XD. The main benefit of such a conversion is that the S12XD features a faster maximum operating speed. This section describes the changes that a developer must make to allow an existing S12 design to run on the S12XD. No attempt is made to describe new features or modules of the S12XD.

Like the previous section, each module is described in turn.

3.1 General Comments

The S12XD is highly compatible with the S12. Most code will run unchanged when users move from the S12. Of course, to take advantage of the higher bus clock speed, users must make alterations to their code. The following list describes areas of code that may need attention when changing to a higher bus speed.

- PLL Configuration
- Timer prescalers
- SCI baud rate
- SPI baud rate
- msCAN configuration
- PWM prescalers
- ADC clock prescaler
- FLASH and EEPROM prescalers
- IIC baud rate

There may be other time dependent areas of coding that need to be reconsidered if the bus clock speed is considered.

3.2 CPU and Instruction Set

The S12XD retains all of the instruction set of the S12 unchanged. The only exception that requires consideration is that the interrupt stack frame on the S12XD is increased by one byte due to the extended Condition Code Register.

S12 users should carefully check the maximum number of interrupt stack frames that can simultaneously exist and increment the stack space by one byte for every frame.

3.3 Interrupt Controller

The interrupt controller provides significant new functionality for the S12XD. The default configuration for this module is that all maskable (I-bit) interrupts are allocated the same interrupt level. In this state, the interrupt behavior is very similar to the S12 in that each interrupt's priority is determined by its position in the interrupt vector table. Compatibility with existing code depends on how interrupts are used in the system.

The I-bit in the Condition Code Register is automatically set when entering an interrupt service routine. This prevents further interrupts from occurring and is unchanged on the S12XD. Users have the option of

clearing this bit and thus allowing nested interrupts where a new interrupt can be serviced before the current one is complete.

3.3.1 Nested Interrupts Not Used

For systems that do not use nested interrupts the S12XD´s reset condition will cause interrupts to behave in the same way as on the S12.

3.3.2 Nested Interrupts Used

For systems that do use nested interrupts then the compatibility is less straightforward and users may have to adjust the interrupt priority levels. The reason is that the interrupt controller does not allow interrupts to occur where the CPU is already operating at the same level as the interrupt. For example, if the interrupt is of priority 1 and the CPU is already servicing an interrupt at priority 1 then the interrupt will not be allowed until the CPU returns to priority 0. This is different to the S12 where any interrupt will be taken if the I-bit is cleared.

The solution in this case depends on the usage of the nested interrupts. If a particular interrupt is more urgent or can be serviced in a shorter time then it may be better to assign it to a higher priority. In general, systems will have a natural hierarchy of interrupts where some are more important than others, so users may find that the changes required for the new interrupt controller actually reduce the amount of code required and optimize the performance.

3.3.3 Other Considerations

A consequence of the new module is that the S12 HPRIO register has been removed. This promoted a single interrupt to a level higher than its position in the interrupt table. If the HPRIO register is used then it may improve system performance if that interrupt is promoted to a higher priority level instead.

The S12XD also has a new vector to deal with spurious interrupts. These can occur if an interrupt is triggered and then disabled before the interrupt vector can be fetched.

S12 users should:

- Use the priority registers (INT_CFDRx) to assign new priorities for any nested interrupts
- Remove any references to the HPRIO register and consider increasing the priority of any interrupt that used it
- Create an interrupt vector and handler for the spurious interrupt

3.4 Memory Configuration and Access

The 9S12XDP512 differs from the 9S12DP512 in that the access to non-FLASH portions of memory is altered. The ability to move RAM, registers, and EEPROM is removed in the S12XD, and so the compatibility of the products depends heavily on how these features have been used. Additionally, the default memory map of the two 512K byte parts is different.

If users have moved the RAM on the S12 to take advantage of the direct page addressing mode, then this can be addressed by moving the direct page area to the corresponding area of RAM using the S12XD DIRECT register.

Figure 9 shows the two default memory maps side by side.

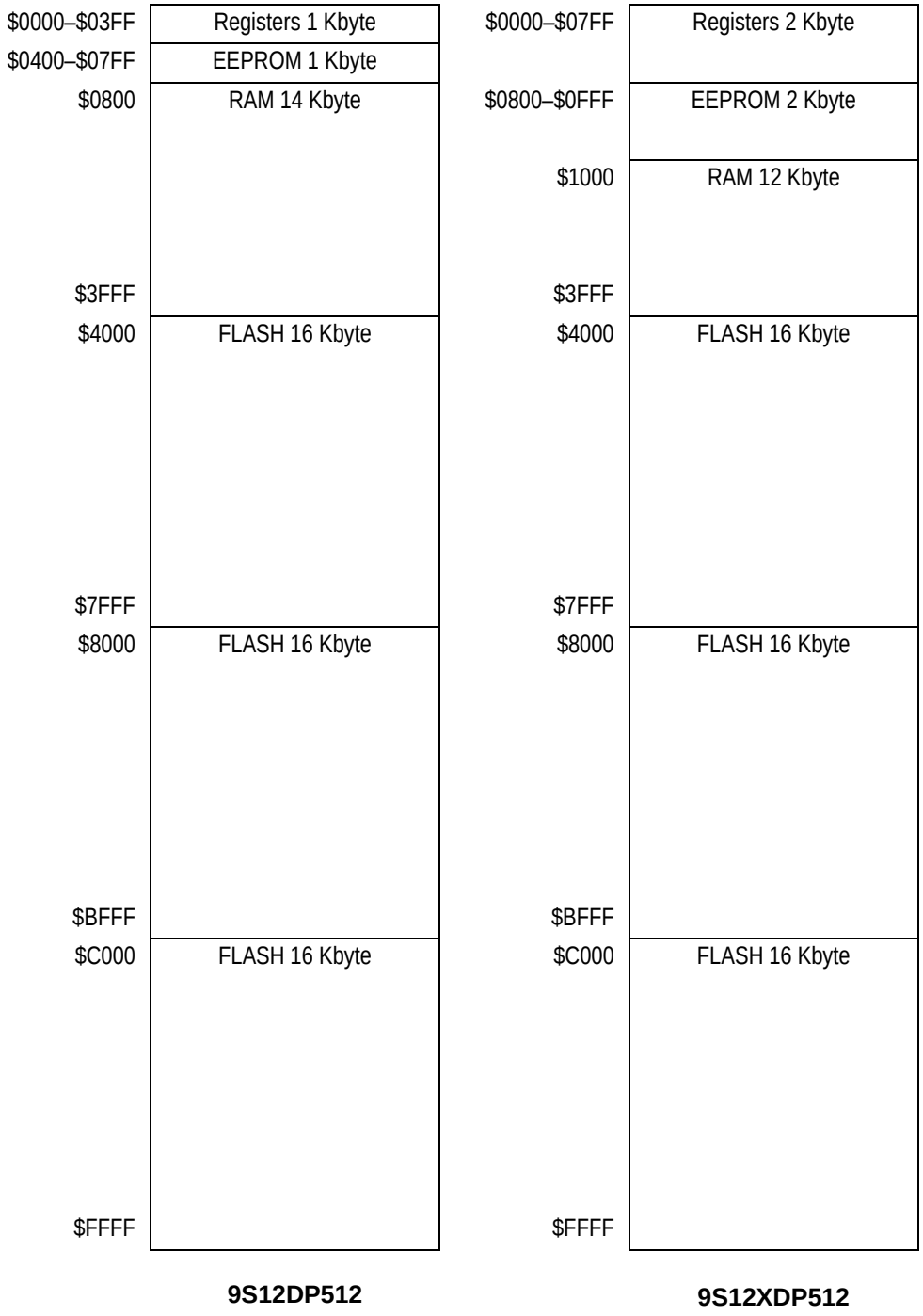

Figure 9. 9S12DP512 AND 9S12XDP512 Default Memory Maps

On the S12 device, the size of the RAM block visible directly after reset is 2 Kbytes larger than on the S12XD device. This is most likely the difference that will cause the greatest difficulty for users porting from S12 to S12XD. The S12XD has the same size of EEPROM available (accessible in a different manner), but has a much larger RAM block (from 14 Kbytes to 32 Kbytes).

Users porting S12 devices with less RAM and EEPROM than the DP512 may find little difference in the use of memory, with the exception of the start address of each block.

3.4.1 Default Memory Use

Users of the S12 who use the default memory arrangement must recompile the code so that the memory access is to the correct address. If access to the additional RAM and EEPROM is required, then new code must be prepared to allow this. The additional RAM and EEPROM is accessible via new page registers — EPAGE and RPAGE. These swap blocks of 1 Kbytes (EEPROM) or 4 Kbytes (RAM) into the memory map. The swappable block of each memory type is the lower portion.

3.4.2 Customized Memory Use

Users of the S12 who use the INITRM, INITRG and INITEE should consider various options in moving to the S12XD. These initialization registers are no longer available on the S12XD, and there are two options for replacing this functionality.

Users can rewrite the code so that banks of RAM and EEPROM are swapped in and out of memory, as required. With this option, 12 Kbytes of RAM are available from 32 Kbytes at all times. RAM is swapped in 4 Kbyte blocks, meaning that there is a total of a further three different 4 Kbyte blocks available to be swapped. EEPROM is swapped in 1 Kbyte blocks and has a total of 4 Kbytes on chip. This means that there is also a total of a further three different 1 Kblocks available to be swapped. Note that, for both the RAM and EEPROM, the fixed blocks of 8 Kbytes and 1 Kbytes respectively can also be swapped into the windows, should this be required.

The second option is to use the new GPAGE register. This identifies a block of 64 Kbytes on chip that is accessible through new CPU instructions. This is an ideal solution where the swapped out banks are infrequently used, or only used by a small number of software tasks. These tasks then do not have to keep track of the bank of memory currently swapped in.

For many users this changed memory scheme will be an improvement, as it is a more flexible arrangement for making memory available on the MCU. With the S12 architecture, trade-offs often had to be made between the various memory types.

3.5 Expanded Bus interface

The expanded bus interface on the S12XD takes a new approach to allow faster operation and reduced complexity. For this reason, users of the S12 will have to create a new hardware layout for their design.

The degree of change will depend on the type of external devices used; in most cases less interface logic will be needed on the S12XD. Key differences to note and consider are that the S12XD uses a non-multiplexed address and data bus with 23 address lines and 16 (separate) data lines. Also, the R/W signal is replaced with separate RE (read enable) and WE (write enable) lines.

3.6 Oscillator

The S12XD features an oscillator module that is different from that provided on the S12. The oscillator module provides an amplitude controlled Pierce configuration. The S12 family features either an amplitude controlled Colpitts configuration, or a choice of an amplitude controlled Colpitts or a full amplitude Pierce configuration. A resistive controlled logic level on the XCLKS pin provides this configuration.

The following paragraphs describe approaches to converting each design. As with all oscillator designs, the choice of external components and layout of the PCB are critical. It is strongly advised that the user carefully consults the advice given the block guide of the oscillator.

3.6.1 S12 Pierce Users

The S12XD Pierce oscillator is a new design and, therefore, although the basic configuration of the external oscillator components is the same, care should be taken to review the PCB layout and components chosen. Users should carefully consult the advice given in the block guide of the Oscillator. The XCLKS configuration resistor is no longer needed for the S12XD.

3.6.2 S12 Colpitts Users

The S12XD uses an oscillator configuration that is different from the Colpitts configuration. The Pierce configuration is closer to the “traditional” oscillator layout used on many MCUs. For this reason, the PCB layout and component choice for the S12XD is different from the S12. Users converting must redesign this portion of the PCB and change components to be able to use the S12XD oscillator.

3.6.3 S12 Direct Clock Users

For those users who provide a digital clock (no analogue oscillator used) directly to the S12, then no change to application design is required.

3.7 TEST and Reset Pins

On the S12XD, the TEST pin features a pull-down resistor. The specification defines that this pin must be connected to Vss, so there is no change to users.

The Reset pin on the S12XD is provided with a pull-up resistor. Users should review any connected reset circuitry, to ensure that its behavior is not affected.

3.8 Background Debug Module

Changes are made to the S12XD BDM to make it fully compatible with features of the new MCU. The BDM features a global page register with the same functionality as the GPAGE. If enabled by setting bit 7 of this register, all BDM read and write addresses are concatenated with the upper seven bits, forming a 23-bit address. This allows debug tools to address the whole 8Mbyte address space without touching one of the user registers like EPAGE, RPAGE, GPAGE, or PPAGE.

In practice, few end-users will be concerned with the changes in this module. In most cases, the only requirement for use of the debug module is to update development tools to be compatible with the new architecture.

3.9 Debug Module

The debug module on the S12XD is enhanced over the S12 to extend the buffer space to include the full memory space and to allow captures from the XGATE. In practice, few end-users will be concerned with the changes in this module.

In most cases, the only requirement for use of the debug module is to update development tools to be compatible with the new architecture.

3.10 Backwards Compatible Modules

The following modules are backwards compatible with the S12 modules. For these, typically a new bit is defined that enables the new features. Users should carefully check that their existing applications software does not inadvertently set/clear these bits. On the S12, this action would be ignored; however, it could cause unforeseen problems if the code is then used on the S12XD.

3.10.1 Enhanced Capture Timer

The ECT is further improved on the S12XD to introduce new timer prescaler options. New features on the module are enabled via a “legacy” bit on the module. Out of the reset, the ECT is compatible with the S12. Users should carefully check their code for the ECT to ensure that this bit is not inadvertently written to.

However, for users upgrading to a faster bus clock, the new prescalers may be a useful addition, since they allow a much finer granularity of clock division. For example, software that relies on a 24MHz bus clock and a timer prescaler of 16 may use the new prescaler options to give a divide by 24 to allow an identical timer tick at a bus speed of 36MHz. This divide option is not possible using the S12 prescaler.

3.10.2 Real Time Interrupt Timer

The RTI is again enhanced over the S12. Once again, a new control bit is added to the module to enable a new decimal prescaler mode. Out of the reset the RTI is compatible with the S12. Again the module is fully backwards compatible with the S12, although users should carefully check their code for the RTI to ensure that this bit (most significant bit of prescaler register) is not written to.

Again users should carefully examine the new prescaler options, as they may offer the simplest route to a compatible timeout period for the RTI timer.

3.10.3 Analog to Digital Converter

The ADC is backwards compatible with the S12, with new trigger sources being the only change. Out of the reset, the ADC is compatible with the S12. Although these are unlikely to occur, users should carefully check their code to ensure that one of these new valid trigger sources is not accidentally enabled.

3.10.4 Serial Communications Interface

The new features of the SCI are enabled using the AMAP bit in the most significant position of SCICS2. Users should carefully check their code to ensure that this bit is not inadvertently written. Out of reset, the SCI is compatible with the S12.

The new features on the SCI are primarily to add support for the IrDA and LIN protocols, and to provide new interrupt and polarity options. Users of these protocols may find that the new features allow them to optimize their code when moving to the S12XD.

3.10.5 msCAN Module

The msCAN module on the S12XD includes a new option for manual recovery from Bus-off mode. This is selectable via a bit in the CANCTL1 register, and the manual action is activated in a new register CANMISC. Out of reset, the msCAN is compatible with the S12. Users should carefully check their code to ensure that these bits are not inadvertently written.

3.11 Unchanged Modules

The following modules are identical on the S12 and S12XD. There is no change required in software or hardware design to utilize them:

- Serial Peripheral Interface (SPI)
- IIC
- General purpose Input/Output ports (GPIO)

THIS PAGE IS INTENTIONALLY BLANK

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

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)
support@freescale.com

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: AN2615
Rev. 4
07/2007

RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics as their non-RoHS-compliant and/or non-Pb-free counterparts. For further information, see http://www.freescale.com or contact your Freescale sales representative.

For information on Freescale’s Environmental Products program, go to http://www.freescale.com/epp.

freescale™
semiconductor