Freescale Semiconductor
Application Note

Document Number: AN3469
Rev. 1, 07/2007

S12 and S12XE Family Compatibility

by: Steve McAslan, MCD Applications, East Kilbride
Joachim Krücken, Microcontroller Division, Munich
Martyn Gallop, MCD Applications, East Kilbride

1 Introduction

The MC9S12XEP100 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 S12XE family's increased speed of operation almost immediately due to its 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 S12XE family. Developers currently designing applications with the S12 family should note these differences so they can take advantage of the S12XE when appropriate. A companion application note (AN2615) describes the compatibility between the S12 and the S12XD families.

All differences noted are based on the feature set of the 9S12XEP100. Other derivatives of the S12XE family might not feature all of the functions described here. Refer to the appropriate data sheet for details.

Contents

The following list summarizes the differences between the S12XE family and the S12 family.

- 50 MHz operation
- 100 MHz Enhanced XGATE peripheral coprocessor
- Enhanced CPU
- Programmable eight level interrupt controller
- Memory protection unit
- Emulated EEPROM
- Flash with error correction coding
- Enhanced memory management controller
- Internal PLL featuring frequency modulation option
- New eight channel periodic interrupt timer
- Improved enhanced capture timer with additional modulus prescaler and output compare options
- Non-multiplexed 8 Mbyte expanded memory bus
- New low-power RC trigger and fast recovery from stop modes
- Improved SCI featuring hardware bit-manipulation for LIN
- SPI with 16-bit mode
- Faster ATD conversions and 12-bit mode for analog to digital converters with enhanced trigger options
- Amplitude controlled Pierce oscillator
- Wider and deeper debug module

2 S12XE Enhancements

This section is intended for developers creating new applications for the S12 who want to maximize compatibility with the S12XE. It is also useful for developers who are familiar with the S12 family when creating applications for the S12XE. Also, this section describes how the new functionality of the S12XE 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 the overall processing requirement and the response time for events is met as expected. The S12XE, 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 or from one peripheral to another. The XGATE can accomplish all this activity, without intervention by the CPU, and run at twice the clock speed of the CPU. This makes the XGATE ideal for managing the real time requirements of most applications.

Because the XGATE is fully programmable, it can also be used to create virtual peripherals. For example, the 9S12XEP100 contains eight SCI modules. By using timer modules on the MCU, the XGATE can add further SCIs with similar performance 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, design the software architecture so that real-time managing 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 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 S12XE 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 S12XE can be used to direct the interrupt request directly to the XGATE.

2.2 CPU and Instruction Set

The S12XE features the new CPU12X-2. This CPU includes enhancements to the programmer model, stacking operations, and the instruction set.

2.2.1 Programmers Model

The S12XE 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.

15				8
U	Reserved	IPL2	IPL1	IPL0

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. The U-bit in the CCR controls the operating state of the CPU. When set, it places the CPU in user state and restricts the operation of some opcodes. Use this bit along with the memory protection unit to improve the memory integrity of your system. See Section 2.4, "System Integrity Improvements through Memory Management," for a detailed description of the behavior of this bit.

2.2.2 Interrupt Stacking Operation

The CCR has been extended to two bytes from one byte. This 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.

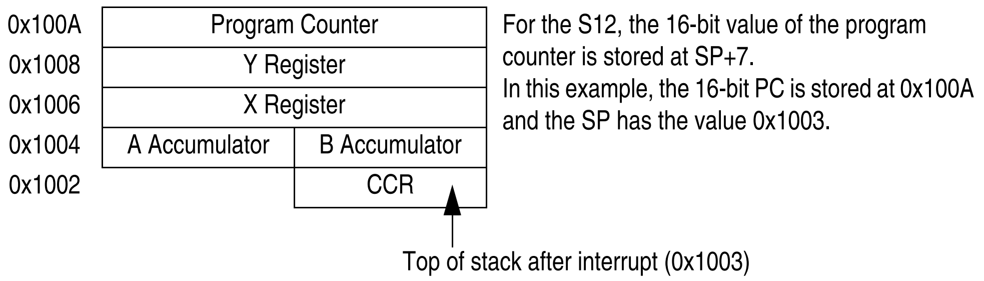
0x100A
0x1008
0x1006
0x1004
0x1002
Program Counter
Y Register
X Register
A Accumulator
B Accumulator
CCR
For the S12, the 16-bit value of the program counter is stored at SP+7.
In this example, the 16-bit PC is stored at 0x100A and the SP has the value 0x1003.
Top of stack after interrupt (0x1003)

Figure 2. Stack Frame Example for S12

Figure 3 gives an example of a stack frame on an S12XE after an interrupt has occurred.

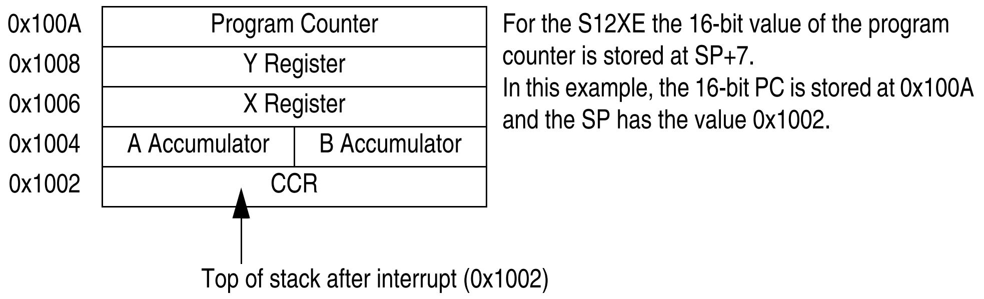
0x100A
0x1008
0x1006
0x1004
0x1002
Program Counter
Y Register
X Register
A Accumulator
B Accumulator
CCR
For the S12XE the 16-bit value of the program counter is stored at SP+7.
In this example, the 16-bit PC is stored at 0x100A and the SP has the value 0x1002.
Top of stack after interrupt (0x1002)

Figure 3. Stack Frame Example for S12XE

In practice, extracting 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, the difference between the S12 and S12XE has little impact for many users.

NOTE

Users developing code on the S12 for later use on the S12XE need to be aware of the maximum stack size and any unusual requirements such as that described. Because each interrupt stack frame is one byte larger on the S12XE, users must take account of this in increasing the memory reserved for the stack when moving to the S12XE.

2.2.3 Instruction Set

The S12XE features an enhanced instruction set over the S12. All of the existing S12 CPU instructions are retained with the exception of five fuzzy instructions (MEM, WAV, WAVR, REV, and REVW) that have been removed.

There are four classes of new instructions.

- New 16-bit operations, where only an 8-bit accumulator operation existed
- New memory access instructions, allowing access to linear banks of up to 64 Kbytes
- New instructions designed to optimize operating system use
- 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 S12XE 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 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 8-bit 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 S12XE CPU provides a set of compare instructions carrying forward the carry and also the zero flag (CPED, CPEX, CPEY, and CPES). This improves the capability to perform 32-bit compares.

While the existing architecture allows 8-bit read-modify-write instructions, the S12XE 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 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 S12XE Memory Management Controller. This allows access to any 64-Kbyte 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 0x00_0000 to 0x7F_FFFF. Each location is accessible with a single instruction from anywhere in a program (after the GPAGE register is configured for that 64-Kbyte page).

NOTE

Users developing code on the S12 for later use on the S12XE should carefully note areas of opportunity/requirement to use this feature and modify their code and their compiler and linker settings using the S12XE.

Class 3 adds new instructions, which are important for real time operating systems (RTOS). The first new instruction is BTAS (bit test and set) and is useful when requesting access to software semaphores. The second new instruction is SYS, which tasks can use to request service from an operating system.

Resources are usually locked via a status bit in RAM — when the bit is set the resource is in use. On the S12, 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 because 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 0x1020, #0x20
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 S12XE addressing modes and not only those fitting in a single postbyte. See the CPU block guide for more information on the newly added modes.

2.3 Interrupt Controller

The S12XE 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. New interrupt vectors include one for detection of spurious interrupts (where an interrupt source is removed before the vector is fetched), one for the new SYS instruction, one for CPU MPU violations, and one for XGATE MPU violations or software errors

The S12XE 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 the system reset at 0xFFFE.

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 (0xFFFF) 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 points at which interrupts are taken. After 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 S12XE, 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, the global mask (I-bit) is clear, and if its priority 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 and 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 is not taken if the I-bit is set. As with the S12, the I-bit is set automatically on entry to an interrupt. 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 continue to benefit from the seven different priority levels because the highest priority interrupt is always selected from those pending.

When the CPU returns from an interrupt, part of its new functionality is to recover the interrupt level to where it was 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 you to have multiple vector tables for multiple mode operating systems, debugging systems, and bootloaders. The IVBR is set to 0xFF 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 services the highest priority request first and can accept a single higher priority interrupt if a lower priority interrupt is already running.

The interrupt module also manages interrupts from the XGATE to the CPU. 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, this priority is programmable in the interrupt module again. Figure 5 shows the architecture of the interrupt module.

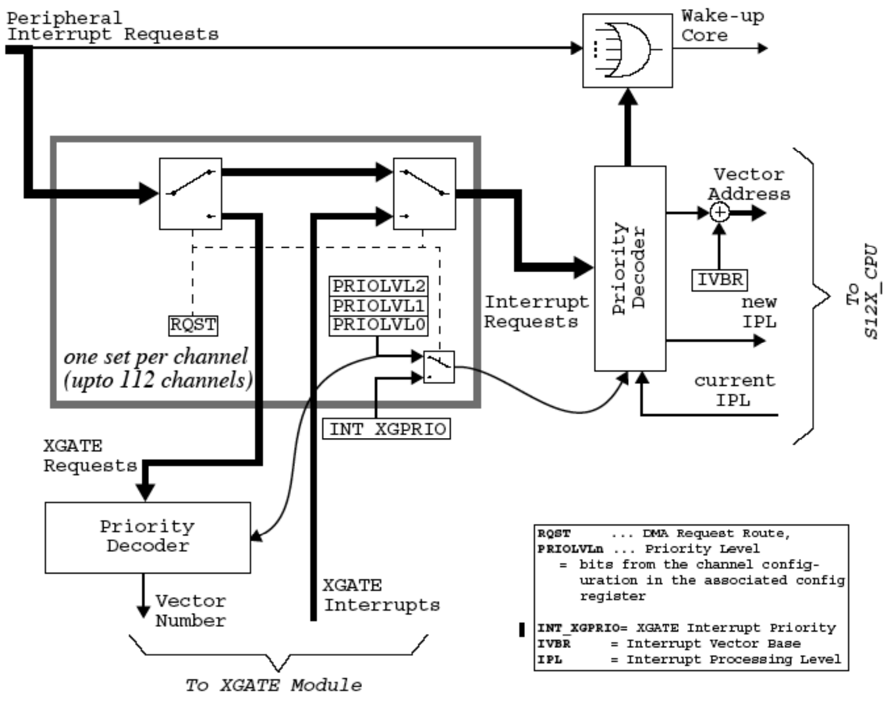
Peripheral Interrupt Requests
Wake-up Core
Vector Address
IVBR
new IPL
current IPL
To S12X_CPU
Priority Decoder
Interrupt Requests
RQST
PRIOLVL2
PRIOLVL1
PRIOLVL0
one set per channel (upto 112 channels)
INT_XGPRIO
XGATE Requests
Priority Decoder
Vector Number
XGATE Interrupts
To XGATE Module
RQST ... DMA Request Route,
PRIOLVLn ... Priority Level
= bits from the channel configuration in the associated config register
INT_XGPRIO= XGATE Interrupt Priority
IVBR = Interrupt Vector Base
IPL = Interrupt Processing Level

Figure 5. Interrupt Module

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

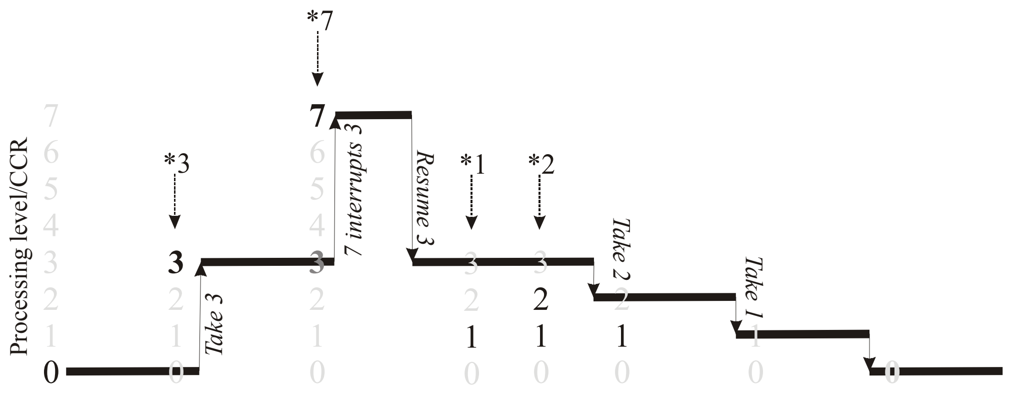

Figure 6. Interrupt Profile

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. Because these interrupts have a lower priority than the currently executing ISR, their ISRs are not executed. Instead, the interrupts 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.

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.

Take care not to set the interrupt level to 0 in the interrupt module. Doing so disables 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 S12XE 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 S12XE cannot be developed. Possible approaches to preparing for the S12XE include: polling important interrupt sources while in an ISR and enabling interrupts as required or enabling all interrupts to occur and managing less important ones as quickly as possible (see AN2617, “A Software Interrupt Priority Scheme for HCS12 Microcontrollers,” for a possible scheme).

2.4 System Integrity Improvements through Memory Management

Increasingly, embedded systems contain software provided by several suppliers. For example, an application developer may select a driver for a particular peripheral from one supplier while a different vendor provides the real-time kernel. The developer then integrates these together with the specific application code. The correct interaction of peripheral drivers, RTOS kernels, and applications is an important and complex challenge for the system integrator. It is important that the various software modules do not incorrectly access or modify another’s memory space or hardware.

The S12XE family includes significant new capabilities to help manage the on-chip and external memory through changes to the CPU and the provision of a new memory protection unit (MPU) module. These new features provide an advanced scheme that allows incorrect software behavior to be detected and averted.

2.4.1 The CPU, System States, and the MPU

The S12XE protection uses a bus master approach where eight memory access descriptors are available for controlling the access permission for any defined bus master. These descriptors are located in a new MPU and define the range of memory addresses and the type of access allowed. If a software task accesses resources outside those permitted, the system generates a non-maskable interrupt.

The MPU supports up to four bus masters. There are two bus masters allocated for the CPU, one for the XGATE and one for another module such as the Flexray controller.

The CPU can operate as two different bus masters depending on the active system state, which can be supervisor state (CPU_SS) or user state (CPU_US). In user state, the CPU is unable to execute certain instructions, which prevents a User State task from changing the system state to supervisor state. The supervisor state allows the MPU to restrict access for code executed by kernels or drivers that require access to system resources.

To support CPU system states, the S12XE CPU includes a new bit, the U-bit in the condition code register (CCR bit 15), and a new non-maskable software system call (SYS) interrupt instruction and associated vector. The SYS instruction is similar to the existing SWI instruction. In fact, it was added to prevent system calls from having to use the SWI, which is often used by debug tools.

The BDM (debug) interface on the S12XE operates independently of the system states and the MPU. This allows a debugger to access any location within the memory map without causing an error or non-maskable interrupt.

2.4.2 User State

User state is entered from supervisor state by application code setting the U-bit. User state prevents the CPU from performing several operations such that software cannot:

- Set or clear system interrupt enables (X, I)
- Set or clear stop enable (S)
- Clear the User State bit (U)
- Change interrupt priority (IPL[0:2])

These limitations affect the instructions that directly modify bits in the CCR (CLI, SEI, ANDCC and ORCC), as well as instructions that modify the entire register (RTI, PULC, EXG, TFR). If the CPU executes one of the above instructions, it modifies the unprotected bits in the CCR and leaves the protected bits unchanged.

The CPU is also unable to execute the WAI and STOP instructions. Instead, it performs the equivalent of a NOP instruction.

2.4.3 Supervisor State

The MPU restricts memory accesses of the CPU in supervisor state only when the supervisor state enable (SVSEN) bit in the MPUSEL register of the MPU is set.

The CPU returns to supervisor state from user state on entry to an interrupt service routine (ISR): the CCR with the U-bit set is stacked, and the U-bit is then cleared. The only way for the MCU to remain in supervisor mode at the end of an ISR is for the service routine code to clear the U-bit in the stacked CCR word.

2.4.4 MPU Descriptors

The MPU features eight identical descriptors that configure protection for the system.

These descriptors describe the memory access permitted by each master. A bus master cannot access memory for which it does not have a descriptor.

The descriptors contain a start and end address for the memory range. The addresses are 20 bits wide and allow access to any address in global memory with an 8-byte resolution. Each descriptor is active if it is assigned to at least one master. Descriptors can be assigned to more than one master.

The descriptors also define the memory access allowed for the given range. They can independently prevent writes to the memory and code execution from the memory. This allows the user to define four different behaviors for memory – see Table 1.

Table 1. MPU Access Combinations

Write	Execute	Behavior
No	No	Read-only data memory
No	Yes	Read-only code memory
Yes	No	Variable RAM
Yes	Yes	Code execution from RAM

2.4.5 Memory Protection Compatibility

Following reset, MPU descriptor 0 is configured, by default, for full access rights to all bus masters for the full global memory range. This enables access to all of the memory map for all bus masters. This is compatible with the default condition on the S12 family and on the S12XD family, if protection is not enabled.

NOTE

The NVM blocks also include protection schemes to prevent accidental modification.

2.5 XGATE

As previously described, the XGATE is a highly integrated but separately programmable coprocessor targeted at I/O management. The XGATE can access all of the peripheral modules and 32-Kbytes of 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 up to double the S12XE CPU bus speed. It can access RAM in cycles where the CPU also accesses RAM. For those cycles where the S12XE CPU does not access the RAM, this performance is further increased. This allows a typical sustained performance of 80 MIPS.

The instruction set is optimized for managing 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, R1, and R7 have special functions.

Code for the XGATE is typically 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 allows 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 manage all eight SCIs on the 9S12XEP100, thereby saving program space. It is equally possible to have different threads for each SCI.

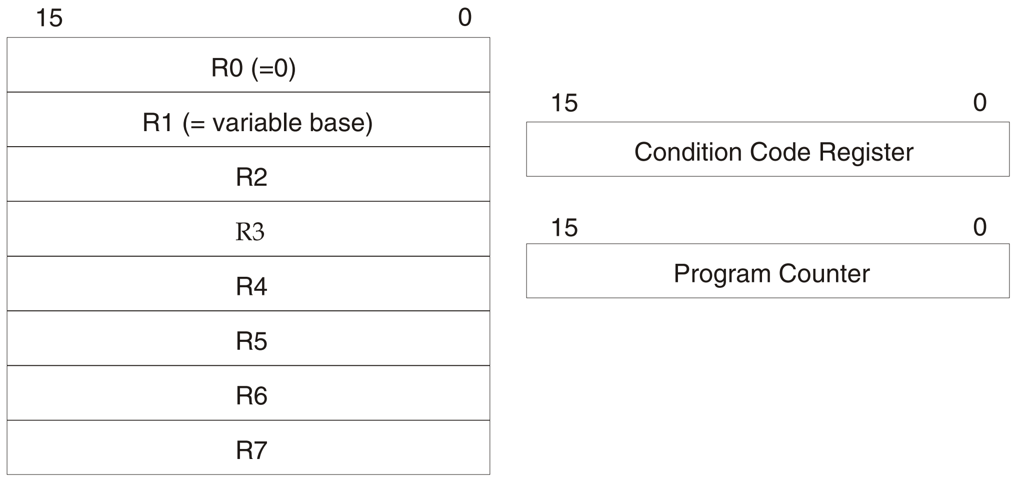

Figure 7. XGATE Programmer's Model

These threads are typically initiated by the interrupt controller, which has the capability of directing interrupts to 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 S12XE can present a number of software design challenges. Because both CPUs effectively have simultaneous access to the RAM and peripherals, the software must be designed so 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.

2.5.1 Interrupt Capability

The S12XE XGATE module supports a single level of pre-emption. Any active lower priority thread (level 1, 2 or 3) is interrupted by any new higher priority interrupt (level 4, 5, 6, or 7) routed to the XGATE.

The interrupt causes XGATE to swap to an alternative set of registers. This new context is exactly the same programmers' model, featuring general purpose registers R0 to R7, the PC, and the CCR. See Figure 8.

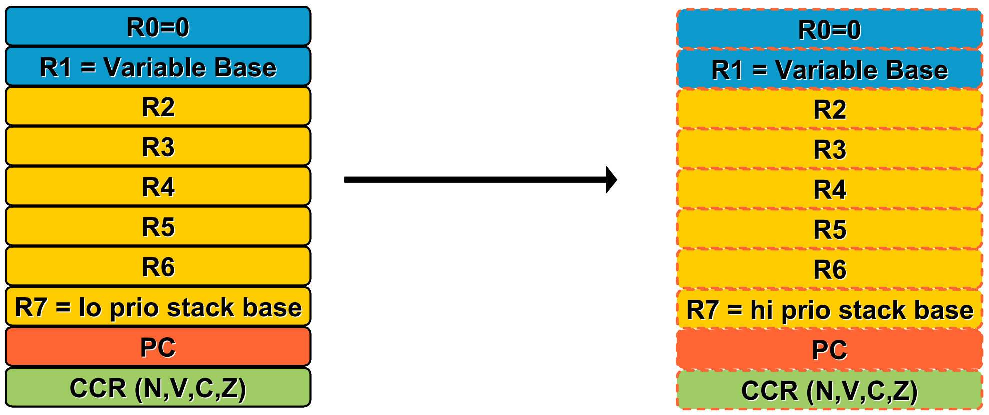

Figure 8. XGATE Context Switch

When programming the XGATE in C, the tool chain creates a software stack that it uses for function calls. The stack base address is configured by each software thread.

If a high priority thread does interrupt a low priority one, the context for the low priority thread (including any register used as a stack pointer, typically R7 for existing tools) is swapped out. At the end of any high priority exceptions, the low priority context is restored and the thread continued.

To avoid a high priority thread corrupting the low priority thread's stack, the S12XE XGATE supports hardware initialization of register R7 at the beginning of each thread execution. This allows a different stack base address for high and low priority threads. These two stack pointer base addresses must be stored in two initial stack pointer configuration registers (XGISP31 and XGISP47) during XGATE initialization. Additionally, the XGATE compiler should be configured not to re-initialize the stack pointer at the start of each thread.

To maintain compatibility with the XGATE memory map on other S12X family members, the two stack pointer base registers are banked at the same location as the existing XGATE vector base address register (XGVBR at module base + 0x0006), and the register banking is controlled by a new initial stack pointer select register (XGISPSEL at module base + 0x0005).

The S12XE XGATE also has a channel priority level register (XGCHPL at module base + 0x0003) that indicates the interrupt level of the currently executing thread.

2.5.2 Error interrupt

The XGATE error interrupt on S12XE is a non-maskable interrupt. The sources of this interrupt are errors in XGATE software execution and memory protection violations by XGATE.

2.6 NVM – Flash and EEPROM

The S12XE family features an improved flash memory module with error correction coding (ECC) and hardware emulated EEPROM. On the S12 family, the flash and EEPROM modules are separate. On the S12XE, these are combined into a single module known as the FTM.

The FTM contains P-flash blocks intended for program storage and D-flash blocks intended primarily for NVM data storage and for emulated EEPROM storage. The FTM has a completely new memory controller to perform flash and EEPROM operations on the P-flash and D-flash blocks.

P-flash and D-flash are implemented with ECC that can resolve single bit errors and detect double bit errors. The ECC implementation on the P-flash requires that programming is done on an 8-byte basis (a P-flash phrase) starting at aligned-by-8 address boundaries. The ECC implementation on the D-flash allows programming of individual words.

FTM commands are performed by writing command information, data, and global addresses to a common command object (CCOB, a set of six banked word registers) and then clearing the CCIF flag in the FSTAT command register. Error and status flags are also available in the FSTAT register similar to the S12 family. The flash programming algorithm no longer writes the data value into the flash array.

There is no command queue as on the S12, and a flash command must complete before the CCOB can be written with the next command.

All flash commands requiring an address use the global address format. For example, the FTM's view of the P-flash on a 1 MB flash device (9S12EXP100) is a linear address map from 0x70_0000 to 0x7F_FFFF. Linear format S-records starting on an 8-byte boundary and with a line length that is a multiple of eight are expected to be the most efficient match. The 8-byte phrase length maps effectively to the 8-byte data length used by CAN bootloaders.

2.6.1 Emulated EEPROM (EEEPROM)

The S12XE family features a hardware scheme that implements EEPROM with improved performance and convenience for the user. The emulated EEPROM consists of a buffer RAM corresponding to the maximum size of the EEPROM, a data flash (D-flash), and a state machine to manage the EEPROM contents.

For an emulated EEPROM of 4 Kbytes, there is a RAM of 4 Kbytes. Software can write into the RAM and the value changes immediately. The software can read the RAM at any time and the value returned is the last value written into the memory location. Each time the software writes into the RAM, the memory controller begins a process of copying from the RAM into the D-flash. This process occurs in parallel with the normal CPU operation and requires no further intervention from user software.

The memory controller manages the contents of the D-flash so there is always erased D-flash available to program for new values in the buffer RAM. At reset, the memory controller copies the contents of D-flash into the buffer RAM.

At any time user software can determine, by examining the MGBUSY bit and the ETAG register, if the memory controller has completed its programming operations.

2.6.2 Emulated EEPROM Format Options

The memory controller allows you to format the size of the emulated EEPROM. If the application requires less than the maximum emulated EEPROM, it is possible to allocate less buffer RAM to this function. The portion of the buffer RAM not used for emulated EEPROM is available as normal RAM. The spare D-flash is then available for direct NVM operation or to extend the number of write-erase cycles on the emulated EEPROM.

You must format the emulated EEPROM once for the lifetime of the application. This operation would typically be performed during initial programming of the MCU.

For a more detailed description of the configuration and operation of the emulated EEPROM system, refer to application note AN3490.

2.6.3 NVM Compatibility

Reads of flash and emulated EEPROM behave in exactly the same way as the S12 family. Reading data directly from flash may generate ECC error warnings, but these are disabled by default.

The algorithm for programming flash and EEPROM is completely different to the approach for the S12 family. For flash programming, the algorithm is different in approach and P-flash also requires aligned 8-byte source data. The most significant change is in the size of data required for each programming operation because the algorithm, although different, uses the same flow and the same source information as the S12 family.

Software on the S12XE now programs flash only through the FTM registers and not by writing into the flash array. This change allows XGATE threads to program P-flash and D-flash. Although the XGATE cannot read the contents of most of the flash, this enhancement offers new possibilities for in-circuit programming. For example, XGATE could write status information to flash while the CPU is busy with other tasks. This could prove useful if, for example, XGATE is performing a system integrity check that discovers a defect in the CPU behavior. The defect information could be written by XGATE to flash for later diagnostic analysis. Take care when using this approach, as the NVM is unavailable while the programming operation is under way. The CPU is unable to read from or execute from NVM that the XGATE is modifying.

For emulated EEPROM programming there is no programming algorithm needed. If the user software requires to monitor the status of the programming operation, the code must be updated to account for the new module. However, you must format the emulated EEPROM before the function is available.

2.7 Memory Management Controller

The S12XE contains a new MMC that provides features not available on the S12 and creates a standard compatible memory map for devices with large or small memory configurations.

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 64 Kbytes accessible directly to the S12XE CPU, but also the full 8-Mbyte global memory space supported by the S12XE. For all S12XE devices, there is a direct correlation between the address of a memory block in the local map (0x0000 to 0xFFFF) and in the global map (0x00_0000 to 0x7F_FFFF). In the global map, the most significant byte of the address corresponds to the value in the GPAGE register.

When developing code on the S12 for later use on the S12XE, carefully consider the best approach for accessing memory on the S12XE. In many cases, use of the GPAGE register is 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 uses a simple GPAGE load on the S12XE and an alternate approach on the S12.

2.7.1 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 S12XE 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 S12XE.

The INITREG register has no direct replacement function in the S12XE because the registers are always fixed at address 0x0000. However, 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 0x0000 to 0x00FF. In that space, the optimized direct page addressing mode may be used by the CPU. On the S12XE, the need to move the registers is removed. Instead, the ability to move the direct page location is introduced. The S12XE has a register called DIRECT, which defines the upper eight bits of the direct page. The direct page address is, therefore, 0xZZ00 to 0xZZFF, where 0xZZ is the contents of the DIRECT register. The DIRECT register is writable only once.

The EEPROM and D-flash block is fixed in the local map at address 0x0800 to 0x0FFF. This 2-Kbyte block size is divided into two sections. The upper 1K from 0x0C00 to 0x0FFF 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 S12XE. 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 9S12XEP100, 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. Further swappable pages give direct access to the D-flash.

The EEPROM and D-flash block is fixed in the global map at address 0x10_0000 to 0x13_FFFF giving a maximum total size of 256 Kbytes. On the 9S12XEP100, the emulated EEPROM exists at address 0x13_F000 to 0x13_FFFF while the D-flash exists at address 0x10_0000 to 0x10_7FFF.

The RAM block is fixed in the local map at address 0x1000 to 0x3FFF. This 12-Kbyte block size is divided into two sections. The upper 8 Kbytes from 0x2000 to 0x3FFF are always present in the memory map. The lower 4 Kbytes acts as a page window in a similar fashion to the EEPROM and flash. The RAM has a control register to determine the 4 Kbytes that is to be visible. For the RAM, this register is called RPAGE. On the 9S12XEP100, there is a total of 64 Kbytes of RAM, so there is a single fixed 8-Kbyte block and 14 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 0x00_1000 to 0x0F_FFFF, giving a maximum RAM size of under 1 Mbytes. On the 9S12XEP100, the RAM exists at address 0x0F_0000 to 0x0F_FFFF.

2.7.2 Flash Mapping

The S12XE 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 0x40_0000 to 0x7F_FFFF, giving a maximum flash size of 4 Mbytes. On the 9S12XEP100, the flash exists at address 0x70_0000 to 0x7F_FFFF.

The PPAGE register selects a block of 16 Kbytes to be visible within a window from 0x8000 and 0xBFFF in the local map. This is unchanged from the S12. The PPAGE value maps into the global map so that page 0x00 corresponds to the 16 Kbytes at the lower addresses in the global map (0x40_0000 to 0x40_3FFF). Page 0x01 corresponds to the next 16-Kbyte page (0x40_4000 to 0x40_7FFF), and so on until page 0xFF, which corresponds to the top 16 Kbytes in the global map (0x7F_C000 to 0x7F_FFFF). In addition to the PPAGE window, and like the S12, the S12XE 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 0x4000 to 0x7FFF are fixed to the global map from 0x7F_4000 to 0x7F_7FFF, and the 16 Kbytes from 0xC000 to 0xFFFF are fixed to the global map from 0x7F_C000 to 0x7F_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 0x3F maps to the top page of the 64 Kbytes (0xxx_C000 to 0xxx_FFFF), as on the S12XE. However, the second bottom page (0xxx_4000 to 0xxx_7FFF) is accessed at PPAGE 0x3E, unlike the S12XE where the value of 0xFD would be used. Typically, the first page of flash used in the window is from page 0x3D. On the S12XE, this would be 0xFE.

In practice, this means the flash page fixed at address 0x4000 to 0x7FFF on the S12 is mapped into the PPAGE window using a value of 0x3E, where the same fixed page on the S12XE is mapped using the value of 0xFD. Because 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 S12XE must ensure that any code accessing the fixed page of flash at address 0x4000 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 because this is an unusual use of flash.

2.7.3 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 external memory may be accessed in a unified manner. The primary benefit of this alternate address space is that unlike the S12, the S12XE 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 S12XE must implement this type of function as a page swap on the S12, but can replace this with a global access instruction on the S12XE

2.7.4 PPAGE Register

On S12, the PPAGE register is located at 0x0030. Address 0x0015 is reserved.

On S12XE, the PPAGE register is located at 0x0015 to simplify access to all of the memory paging registers when the register block is protected by an MPU descriptor. Address 0x0030 is reserved. Other than the change in location, the PPAGE register functions in exactly the same way as on S12. Most existing software tools support selection of a different register address for PPAGE.

2.7.5 Alternative RAM Mapping Configuration

On the S12XE, there is a new option for alternative mapping of the on-chip RAM into the 4000-7FFF fixed memory space as an alternative to page 0xFD of the program flash. This is a write-once configuration that allows the RAM to be configured as a 24-Kbyte flat memory map and 4-Kbyte paged (or 28-Kbyte flat if the application does not page the RAM).

This is enabled by the RAMHM bit (MMCCTL1, bit 3) and the existing ROMHM bit (MMCCTL1, bit 1)

When both of these bits equal 1,

- Accesses to 0x4000–0x7FFF are mapped to 0x0F_C000-0x0F_FFFF in the global Memory space
- Accesses to 0x2000-3FFF are mapped to 0x0F_A000-0x0F_BFFF in the global Memory space.

The default value of the RPAGE register remains 0xFD and should be written to 0xF9 in the application to configure a 28-Kbyte flat memory map.

Page 0xFD of the flash can continue to be accessed via PPAGE and GPAGE instructions.

2.8 Expanded Bus interface

The expanded bus interface on the S12XE 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, four 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 S12XE expanded bus, the E clock is now considered a free-running clock. It can be output at the internal bus speed, divided by any value from 1 to 512, 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 eight 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 S12XE 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 S12XE can take several approaches depending on the type of external peripheral devices. Most important is to examine the features on the S12XE 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 S12XE functionality and can therefore be directly removed when converting. If the peripheral continues to require a complex logic interface, 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 S12XE design.

On the S12XE, two different access stretch counts can be configured in EBICTL0. Each of the four chip selects can then be programmed individually to use one of the two counts or the external EWAITE signal, independently. Configuration of each chip select is via a pair of bits in MMCCTL0 register (CSnE1 and CSnE0).

2.9 Analog to Digital Converter (ATD)

The analog part of the ATD conversion module is a completely new design for S12XE that supports higher resolution (12-bit), faster sampling rates, and lower power conversion. It includes additional features, such as an internal RC oscillator for conversion in full-stop mode, analog comparison against a user programmed limit (with support for wakeup from low power modes on a match), and sample and hold capacitor discharge (for improved system reliability).

In general, the configuration and conversion sequencing are unchanged from the existing S12X ATD. This means that the general software flow is common between the two ATD modules. However, some registers and specific control bits have been modified to take account of the improved feature set and to remove redundant functionality.

This section describes the ATD configuration and functionality changes in detail.

2.9.1 New features

The ATD includes an internal oscillator that allows conversions to be carried out when the MCU is in STOP. Software can enable this new functionality by setting a bit in ATDCTL2.

The ATD also has a comparison feature that can cause an interrupt if a conversion is less than or equal to or is higher than a defined value. The comparison function is available in any mode (i.e. is not restricted to STOP mode) and frees the CPU from constantly having to check ATD results against a particular threshold. The comparison interrupt is independent of the ATD result register, allowing the CPU and XGATE to distinguish between regular conversion completion and a comparison event.

Importantly, these features can be combined and allow an MCU to monitor an external analog voltage and wake from stop if it drops below, matches, or exceeds a known value. The comparison is available on all channels and compares the ATD result against the value written into the result register.

2.9.2 ATD Pin Assignments

The S12XEP100 has two 16-channel ATD modules. The pin assignment (i.e. the mapping of analog input pins to ATD module channels) is optimized to support the two 16-channel modules.

2.9.3 ATD Register Detail

The ATDCTL1 register includes three new control bits:

- Bit 4 is now SMP_DIS. This is the discharge before sampling bit that, when set, causes the internal sample capacitor to be discharged each time before sampling the channel. This can help to detect an open circuit instead of measuring the previous sampled channel.
- Bits [6:5] are two new bits named SRES[1:0]. These select 8-bit, 10-bit or 12-bit conversion resolution.

The ATDCTL2 register contains three changes:

- Bit 7 (ADPU) has been removed. The ATD module is enabled by default and is static, consuming only leakage current when not converting. There is no longer a requirement for a 20µs setup delay.
- Bit 5 (AWAI) has been removed. If the application requires the ATD is needed to halt in wait mode, any ongoing conversions must be halted explicitly by the application software in the Wait entry code (considering the faster conversion timing and reduced power consumption this is likely to be a requirement for a continuous conversion sequence only.)
- Bit 5 is replaced by a new internal clock (enabled) in stop mode bit (ICLKSTP) that controls if the ATD converter continues activity in stop modes (including full system stop).
- Bit 0 (ACIF) is removed. This was a copy of the SCF flag. Any software reading the ACIF bit must be modified to read the SCF flag.
- Bit 0 is replaced by a new ATD compare interrupt enable flag (ACMPIE) that allows the STD to cause an interrupt if a compare condition occurs.

The ATDCTL4 register contains a single bit change:

- Bit 7 (SRES8) has been removed. ATD resolution is now configured by the new SRES[1:0] bits in ATDCTL1.
- Bit 7 is replaced by an additional sample length configuration bit SMP2 (adjacent to existing SMP1 and SMP0). Sample time encoding is not compatible and user software must configure the ATD sample times so that it is compatible with the application characteristics (see Section 2.9.5, “Additional Comments on the ATD”).

The ATDCTL5 register contains two changes:

- Bit 6 (DSGN) has been removed. Applications requiring signed offset data should manipulate the results data in software.
- Bit 6 is replaced by the SC bit from ATDTEST1 bit 0. Conversion of the external channels or internal references are now initiated by the conversion start access to ATDCTL5.
- Bit 7 (DJM) has been moved to ATDCTL3 bit 7. The data justification mode affects the format of the conversion results written by the converter to the results register and also the justification of the comparison values data written to the results registers by the CPU or XGATE. On S12XE this bit is reserved.

The ATDTEST0 and ATDTEST1 registers are replaced by a new 16-bit ATD compare enable register (ATDCMPE). After reset, the register disables the ATD compare and, as long as it is not written, it is compatible. The only compatibility issue is that any writes to the SC bit to enable conversion of the internal references should now be part of a modified write to ATDCTL5.

The PORTAD0 and PORTAD1 registers are replaced by a new 16-bit ATD compare higher than register (ATDCMPHT). Application code that accesses the PORTAD registers have to be modified to access the alternative PTxADx registers in the port integration module (PIM).

The ATDDRn registers are read-only on the S12. On the S12XE they are also used as comparison registers and so the software can write to these registers. Writing to ATDDRn registers causes the last ATD conversion result to be lost.

2.9.4 Conversion Complete Flag Clearing

The default flag clearing mechanism (i.e. with AFFC = 0) for the conversion complete flags (CCFx) in ATDSTAT2 has been updated. On the S12, the conversion complete flags are cleared by a single read of the ATDSTAT2 register followed by a read of the appropriate conversion result register(s). On the S12XE, the conversion complete flags must be cleared by writing a 1 to the set CCFx bit (standard flag clearing method on S12).

The fast flag clear mode (i.e. with AFFC = 1) remains compatible.

2.9.5 Additional Comments on the ATD

The 12-bit resolution is intended to improve the relative sampling resolution. The absolute error of the converter is expected to be the same as for the current 10-bit ATD.

The S12XE analog design does not include a buffer amplifier (as on the S12). Therefore, conversion accuracy is going to be more sensitive to the correct sample time than on S12. To assist with setting a suitable sample time, the encoding of the sample timing (SMP bits) is enhanced.

2.9.6 ATD Compatibility

The significant new functionality on the ATD requires a number of changes to the configuration registers on the module. The general flow of the software is unchanged so software changes are expected to be relatively minor. Although, it is important that the changes are implemented correctly.

2.10 Debug Module

The debug module on the S12XE 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 S12XE can be sure that the level of support provided by the debug module is fully extended to support the new features on the S12XE.

2.11 Oscillator

The S12XE features an oscillator module different from that provided on the S12. The oscillator module provides an amplitude controlled Pierce configuration. The S12 family features 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 S12XE should begin with the Pierce option, unless low power is an important design consideration. If the low-power option is required, the Colpitts option (available only on the S12) should be chosen. In either case, consideration should be given to the final layout so that changes can be made as required to suit the S12XE.

As with all oscillator designs, the choice of external components and the layout of the PCB are critical. Carefully consult the advice given the in block guide of the oscillator.

2.12 Internal Phase Locked Loop (IPLL)

Unlike the PLL on the S12 family, the S12XE internal PLL (IPLL) does not require an external filter circuit on the PCB (there is no XFC pin). The S12XE also has a new register configuration and provides support for frequency modulation (which can assist with EMC emissions in some applications).

Software must configure the CRG to generate a VCO frequency within a specified range. There are new controls to configure the IPLL filter and VCO gain for specific input reference frequency ranges and the VCO output frequency ranges respectively. The VCO output clock is then divided down to provide the required IPLL clock output. The IPLL frequency modulation can be enabled and selected for three different modulation amplitudes.

2.12.1 CRG Register Detail

The SYNR register includes two new control bits:

- Bits [7:6] are two new bits named VCOFRQ[1:0]. These bits control the gain of the VCO and user software must select the correct configuration based on the target frequency.

The REFDV register includes two new control bits:

- Bits [7:6] are two new bits named REFFRQ[1:0]. These bits configure the internal filter and software must select the correct configuration based on the reference clock frequency.

There is no longer any distinction between acquisition and tracking mode. This leads to changes to two bits in the PLLCTL register:

- Bit 4 ACQ is removed.
- Bit 5 AUTO is removed.
- Bits [5:4] are two new bits named FM[1:0]. These bits enable and select frequency modulation on PLLCLK for reduced noise emission.

This also causes changes in the CRGFLG register:

- Bit 2 TRACK status bit is removed.
- Bit 2 becomes the illegal address reset flag (ILAF) from the CRGINT register, so all CRG flags are now located in the CRGFLG register.
- Bit 6 in the CRGINT register is now reserved.

The CLKSEL register contains a new bit:

- Bit 5 is a new bit named XCLKS. This read-only bit reflects the selected (latched) oscillator configuration.

The CRG also features a new register POSTDIV that divides the VCO output clock (VCOCLK) down to the desired system clock frequency. This replaces the CTFLG register on the S12 family, which was used only for factory test.

2.13 Enhanced Capture Timer

The ECT on the S12XE contains improvements that allow more precise timing events to be created or measured. The ECT remains fully backwards compatible with the S12, but 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 S12XE allow steps of 1, 2, 3, 4, 5, 6, and up to 256.

NOTE

Users developing applications on the S12 for later use on the S12XE 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 S12XE. The new prescalers can also be used to maintain time intervals on an S12XE when running at a higher bus speed than the original S12.

The ECT timer has an additional output compare pin disconnect (OCPD) register, which can support glitch free initialization of the output state of any output compare channel. On the S12, the output compare configuration OMx = 0, OLx = 0 is described as timer disconnected from output pin logic. This is a typical configuration where the capture compare interrupt is only for internal interrupt timing. In this case control of the associated port pin is automatically switched to the PIM port control registers.

On the S12XE, the output compare configuration OMx = 0, OLx = 0 is more specifically described as no output compare action on the timer output signal. The internal interrupt behavior is the same as on the S12, but the OCPD register explicitly controls the connection of the OC channel logic or the IO port control logic to the port pin.

The reset state of the OCPD register is for the OC logic to be connected to the pin when a channel is configured for output compare. For compatibility with S12, the relevant OCPD bit must be set during configuration if GPIO functionality is required while using the OC channel as an internal interrupt timer.

2.14 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 S12XE 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 S12XE 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 S12XE.

2.15 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.

2.16 Periodic Interrupt Timer

The PIT is an internal timer module. You can use the PIT to create interrupts or trigger the ADC peripheral at 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 that does not take up any external pins.

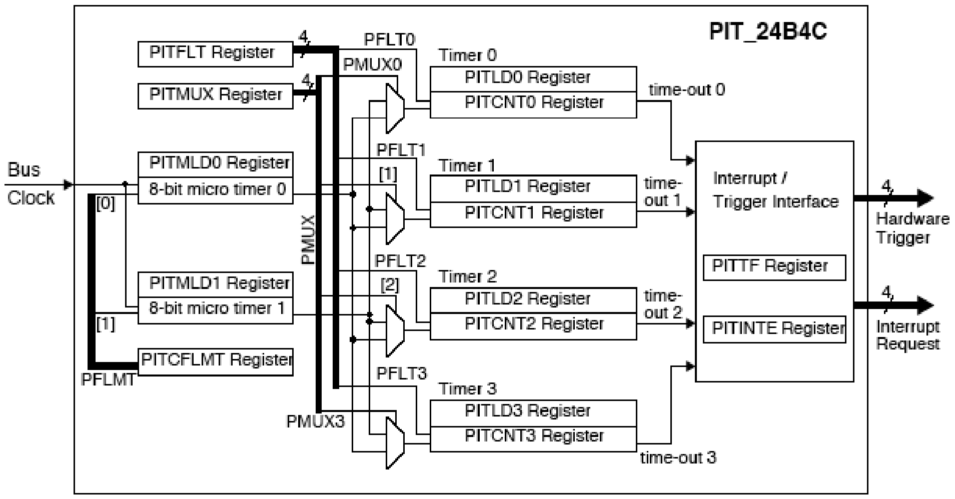

Figure 9. Periodic Interrupt Timer

As can be seen in Figure 9, 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, 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 S12XE 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.

2.17 Miscellaneous Modules

The following modules have minor changes, to keep in mind when developing a new application intended to run on S12XE. Some of the new features may make the S12XE more suitable for particular applications.

2.17.1 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. As a result of a design change, the delay before a transmission starts is increased by one half bit time. Otherwise, the SCI is fully backwards compatible with the S12.

2.17.2 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 available and is the default configuration.

2.17.3 Background Debug Module

The BDM module on the S12XE includes changes to account for the new memory management controller. It also automatically takes advantage of the blank check capability of the flash and EEPROM hardware interfaces.

2.17.4 Low Power and STOP Mode Support

The S12XE 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 often used for this type of application, and the S12XE 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 soon after an event is detected. Because 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 recovery be required, user code restarts the external oscillator and begins running code as normal.

Where a regular check on external activity is required, the S12XE can make use of a low-power internal RC oscillator with programmable time ticks from 0.2 ms to 13 s. 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.

2.18 Voltage Regulator

The on-chip voltage regulator (VREG) generates supply voltages for the core logic, and the NVM and oscillator/PLL modules.

On the S12, the VREGEN pin allows the use of an external voltage regulator for the core power supply. In this case, the core supply pins are connected together on the application PCB.

On the S12XE, the VREGEN pin is not available, and the internal voltage regulator is always enabled. It is recommended that the core supply pins are not connected together on the PCB and that each pair of supply pins has a dedicated decoupling capacitor. S12 designs using an external voltage regulator to supply the 2.5V core supplies have to be modified appropriately.

V_{DD1}, V_{DD2}, and V_{DDPLL} are all nominally 2.5V on S12.

Table 2. S12 V_{DD} Values

		Characteristic	Symbol	Min	Typ	Max	Unit
7	P	Low Voltage Interrupt Assert Level Deassert Level	V_{LIVA} V_{LVID}	4.1 4.25	4.37 4.52	4.66 4.77	V V
8	P	Low Voltage Reset Assert Level	V_{LVRA}	2.25	—	—	V
9	C	Power-on Reset Assert Level Deassert Level	V_{PORA} V_{PORD}	0.97 —	— —	— 2.05	V V

V_{DD} (= V_{DD1}) and V_{DDPLL} are nominally 1.8V and V_{DDF} (==V_{DD2}) is nominally 2.8V on S12XE.

Table 3. S12XE V_{DD} Values

		Characteristic	Symbol	Min	Typ	Max	Unit
5	P	Low Voltage Interrupt Assert Level Deassert Level	V_{LIVA} V_{LVID}	4.06 4.19	4.21 4.34	4.36 4.49	V V
6	P	V_{DD} Low Voltage Reset Assert Level	V_{LVRA}	1.62	—	—	V
7	P	V_{DDF} Low Voltage Reset Assert Level	V_{LVRFA}	2.6	—	—	V
8	P	V_{DDX} Low Voltage Reset Assert Level	V_{LVRXA}	3.13	—	—	V

NOTE

On the S12XE, with the minimum LVR level for the input supply voltage equaling 3.13V, the device is held in reset for any supply effectively below the lower supply input voltage (= 3.15V), preventing out-of-spec operation. This is the lowest voltage the device can operate at for any functionality.

On S12XE, the maximum allowable decoupling capacitance on V_{DD1} and V_{DD2} is 440nF.

2.19 IIC

The IIC module has two additional bits to support new general call address and 10-bit addressing features, which are controlled via a new control register IBCR2 located at module base +0x0005. This is a reserved register on S12.

The new features default to disabled and the default IIC is fully compatible with S12 and S12X.

2.20 Serial Peripheral Interface (SPI)

The SPI on the S12XE additionally supports 16-bit data transfers as well as 8-bit transfers. This is controlled via a new transfer word (XFRW) in a control register (IBCR2 bit 5).

The S12 8-bit data register SPIDR is renamed SPIDRL and is used for the least significant eight data bits for 8-bit or 16-bit transfer modes.

In 16-bit transfer mode, the upper byte of the 16-bit data (SPIDRH) is located at module base +0x0004. This is a reserved register on S12.

This SPI new feature defaults to disabled and the default mode is fully compatible with S12X.

3 Considerations for Existing S12 Applications

This section is intended for developers who wish to convert an existing S12 design to operate on an S12XE. The main benefit of such a conversion is that the S12XE 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 S12XE. No attempt is made to describe new features or modules of the S12XE.

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

3.1 General Comments

The S12XE is highly compatible with the S12. Most code runs unchanged when users move from the S12. To take advantage of the higher bus clock speed, make alterations to the 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
- IIC baudrate

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 S12XE retains all of the instruction set of the S12 unchanged with the exception of the four fuzzy logic instructions. The only exception that requires consideration is that the interrupt stack frame on the S12XE 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. Ensure that application software does not use the fuzzy logic instructions (MEM, REV, REVW, and WAV/WAVR)

3.3 Interrupt Controller

The interrupt controller provides significant new functionality for the S12XE. 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 similar to the S12 because 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 S12XE. You 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 S12XE reset condition causes interrupts to behave in the same way as on the S12.

3.3.2 Nested Interrupts Used

For systems that do use nested interrupts, the compatibility is less straightforward and you 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, the interrupt is not allowed until the CPU returns to priority 0. This is different from the S12 where any interrupt is 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, it may be better to assign it to a higher priority. In general, systems have a natural hierarchy of interrupts where some are more important than others. 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 interrupt 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, it may improve system performance if that interrupt is promoted to a higher priority level instead.

The S12XE 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 9S12XEP100 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 S12XE. The compatibility of the products depends heavily on how these features have been used. Additionally, the default memory map of the two parts is different.

If the RAM on the S12 has been moved to take advantage of the direct page addressing mode, this can be addressed by moving the direct page area to the corresponding area of RAM using the S12XE direct register.

Figure 10 shows the two default memory maps side by side.

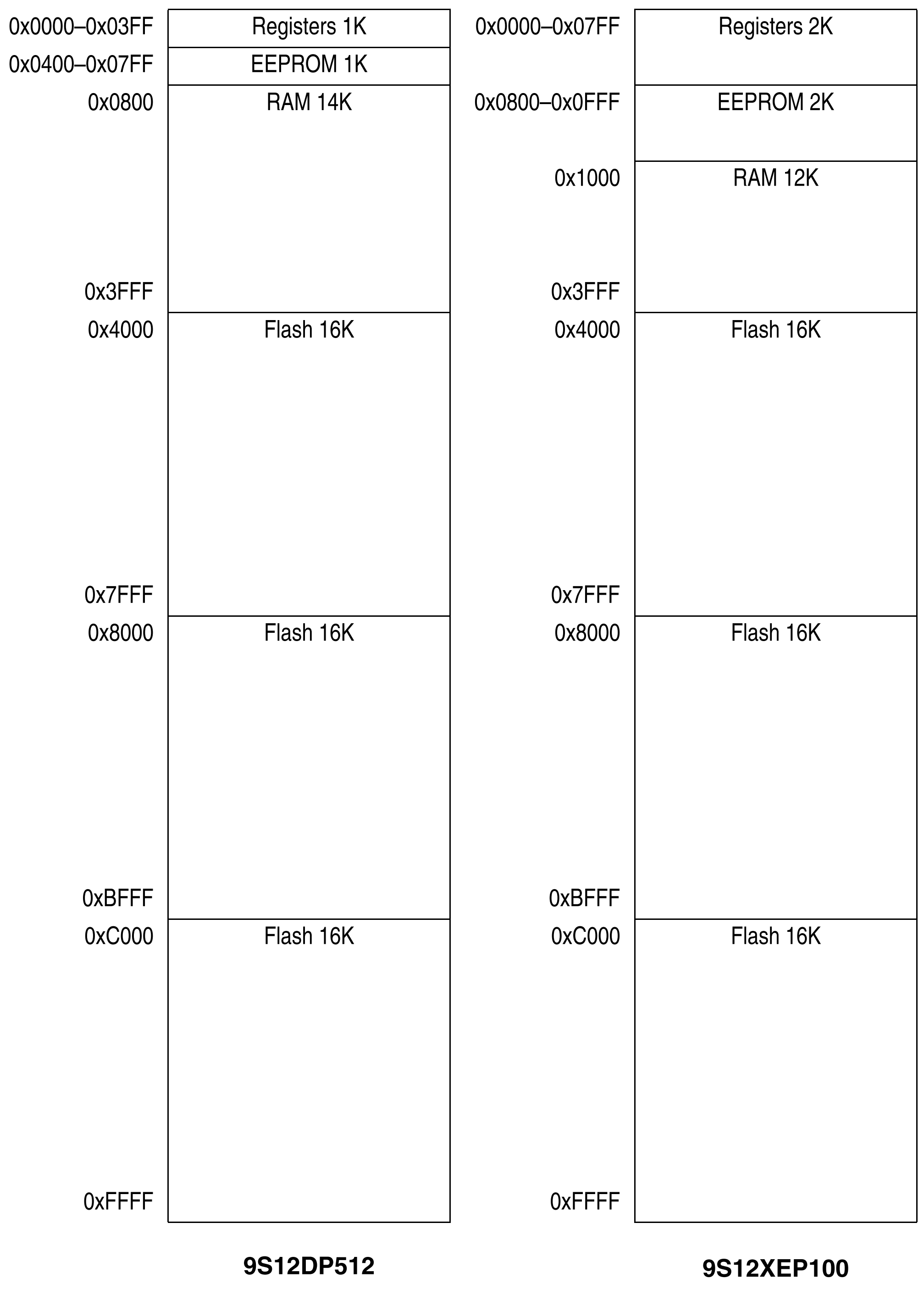

Figure 10. 9S12DP512 AND 9S12XEP100 Default Memory Maps

On the S12 device, the size of the RAM block visible directly after reset is 2 Kbytes larger than on the S12XE device. This is most likely the difference that causes the greatest difficulty for users porting from S12 to S12XE. The S12XE has the same size of EEPROM available (accessible in a different manner), but has a much larger RAM block (from 14 Kbytes to 64 Kbytes).

When porting S12 devices with less RAM and EEPROM than the DP512, you 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

When using the default memory arrangement of the S12, recompile the code so that the memory access is to the correct address. If access to the additional RAM and EEPROM is required, 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

When using the INITRM, INITRG, and INITEE of the S12, consider various options in moving to the S12XE. These initialization registers are no longer available on the S12XE, and there are two options for replacing this functionality.

You 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 64 Kbytes at all times. RAM is swapped in 4-Kbyte blocks, meaning that there is a total of a 16 different 4-Kbytes 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 a total of a four different 1-Kbyte blocks available to be swapped. For 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 a chip 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 is an improvement because 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 S12XE takes a new approach to allow faster operation and reduced complexity. For this reason, users of the S12 have to create a new hardware layout for their design.

The degree of change depends on the type of external devices used. In most cases, less interface logic is needed on the S12XE. Key differences to note and consider are that the S12XE 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 S12XE features an oscillator module different from that provided on the S12. The oscillator module provides an amplitude controlled Pierce configuration. The S12 family features 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 to carefully consult the advice given in the block guide of the oscillator.

3.6.1 S12 Pierce Users

The S12XE Pierce oscillator is a new design and, therefore, care should be taken to review the PCB layout and components chosen although the basic configuration of the external oscillator components is the same. Carefully consult the advice given in the block guide of the oscillator. The XCLKS configuration resistor is no longer needed for the S12XE.

3.6.2 S12 Colpitts Users

The S12XE uses an oscillator configuration 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 S12XE is different from the S12. When converting, redesign this portion of the PCB and change components to be able to use the S12XE oscillator.

3.6.3 S12 Direct Clock Users

When providing a digital clock (no analogue oscillator used) directly to the S12, the input voltage of the clock source must change from 2.5V to 1.8V.

3.7 TEST and Reset Pins

On the S12XE, 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 S12XE is provided with a pull-up resistor. Review any connected reset circuitry to ensure that its behavior is not affected.

3.8 Background Debug Module

Changes are made to the S12XE 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 8-Mbyte address space without touching one of the user registers like EPAGE, RPAGE, GPAGE, or PPAGE.

In practice, few end-users are 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 S12XE is enhanced over the S12 to extend the buffer space to include the full memory map and to allow captures from the XGATE. In practice, few end-users are 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, a new bit is typically defined that enables the new features. Carefully check that the existing application's 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 S12XE.

3.10.1 Enhanced Capture Timer

The ECT is further improved on the S12XE to introduce new timer prescaler options and new output compare disconnect 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. Carefully check the code for the ECT to ensure that this bit is not inadvertently written to.

However, when upgrading to a faster bus clock, the new prescalers may be a useful addition because they allow a much finer granularity of clock division. For example, software that relies on a 24-MHz 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 36 MHz. 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. 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, you should carefully check the code for the RTI to ensure that this bit (most significant bit of prescaler register) is not written to.

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

3.10.3 Serial Communications Interface

The new features of the SCI are enabled using the AMAP bit in the most significant position of SCICS2. Carefully check the 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 S12XE.

3.10.4 Serial Parallel Interface

The new features of the SPI are enabled using the XFRW bit in SPICR2. Carefully check the code to ensure this bit is not inadvertently written. Out of reset, the SPI is compatible with the S12.

3.10.5 msCAN Module

The msCAN module on the S12XE 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. Carefully check their code to ensure that these bits are not inadvertently written.

3.10.6 IIC

The IIC module has two additional bits to support new general call address and 10-bit addressing features. Carefully check the code to ensure these bits are not inadvertently written. Out of reset, the IIC is compatible with the S12.

3.11 Unchanged Modules

The general purpose input/output ports (GPIO) module is identical on the S12 and S12XE. There is no change required in software or hardware design to utilize it.

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: AN3469
Rev. 1
07/2007