Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	HCS12X
Core Size	16-Bit
Speed	80MHz
Connectivity	CANbus, EBI/EMI, I²C, IrDA, LINbus, SCI, SPI
Peripherals	LVD, POR, PWM, WDT
Number of I/O	119
Program Memory Size	256KB (256K x 8)
Program Memory Type	FLASH
EEPROM Size	4K x 8
RAM Size	16K x 8
Voltage - Supply (Vcc/Vdd)	2.35V ~ 5.5V
Data Converters	A/D 24x10b
Oscillator Type	External
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	144-LQFP
Supplier Device Package	144-LQFP (20x20)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mc9s12xdt256mag

Table 1-1. Device Register Memory Map (continued)

Address	Module	Size (Bytes)
0x0110–0x011B	EEPROM control register	12
0x011C–0x011F	MMC (memory map control)	4
0x0120–0x012F	INT (interrupt module)	16
0x0130–0x013F	Reserved	16
0x0130–0x0137	SCI4 (serial communications interface)	8
0x0138–0x013F	SCI5 (serial communications interface)	8
0x0140–0x017F	CAN0 (scalable CAN)	64
0x0180–0x01BF	CAN1 (scalable CAN)	64
0x0180–0x023F	Reserved	192
0x01C0–0x01FF	CAN2 (scalable CAN)	64
0x0200–0x023F	Reserved	64
0x0200–0x023F	CAN3 (scalable CAN)	64
0x0240–0x027F	PIM (port integration module)	64
0x0280–0x02BF	CAN4 (scalable CAN)	64
0x02C0–0x02DF	Reserved	32
0x02C0–0x02DF	ATD0 (analog-to-digital converter 10 bit 8-channel)	32
0x02E0–0x02EF	Unimplemented	16
0x02F0–0x02F7	Voltage regulator	8
0x02F8–0x02FF	Unimplemented	8
0x0300–0x0327	PWM (pulse-width modulator 8 channels)	40
0x0328–0x033F	Unimplemented	24
0x0340–0x0367	Periodic interrupt timer	40
0x0368–0x037F	Unimplemented	24
0x0380–0x03BF	XGATE	64
0x03C0–0x03FF	Unimplemented	64
0x0400–0x07FF	Unimplemented	1024

Table 2-8. RTI Frequency Divide Rates for RTDEC = 1

RTR[3:0]	RTR[6:4] =							
	000 ($1x10^3$)	**001 ($2x10^3$)**	**010 ($5x10^3$)**	**011 ($10x10^3$)**	**100 ($20x10^3$)**	**101 ($50x10^3$)**	**110 ($100x10^3$)**	**111 ($200x10^3$)**
0000 (÷1)	$1x10^3$	$2x10^3$	$5x10^3$	$10x10^3$	$20x10^3$	$50x10^3$	$100x10^3$	$200x10^3$
0001 (÷2)	$2x10^3$	$4x10^3$	$10x10^3$	$20x10^3$	$40x10^3$	$100x10^3$	$200x10^3$	$400x10^3$
0010 (÷3)	$3x10^3$	$6x10^3$	$15x10^3$	$30x10^3$	$60x10^3$	$150x10^3$	$300x10^3$	$600x10^3$
0011 (÷4)	$4x10^3$	$8x10^3$	$20x10^3$	$40x10^3$	$80x10^3$	$200x10^3$	$400x10^3$	$800x10^3$
0100 (÷5)	$5x10^3$	$10x10^3$	$25x10^3$	$50x10^3$	$100x10^3$	$250x10^3$	$500x10^3$	$1x10^6$
0101 (÷6)	$6x10^3$	$12x10^3$	$30x10^3$	$60x10^3$	$120x10^3$	$300x10^3$	$600x10^3$	$1.2x10^6$
0110 (÷7)	$7x10^3$	$14x10^3$	$35x10^3$	$70x10^3$	$140x10^3$	$350x10^3$	$700x10^3$	$1.4x10^6$
0111 (÷8)	$8x10^3$	$16x10^3$	$40x10^3$	$80x10^3$	$160x10^3$	$400x10^3$	$800x10^3$	$1.6x10^6$
1000 (÷9)	$9x10^3$	$18x10^3$	$45x10^3$	$90x10^3$	$180x10^3$	$450x10^3$	$900x10^3$	$1.8x10^6$
1001 (÷10)	$10\ x10^3$	$20x10^3$	$50x10^3$	$100x10^3$	$200x10^3$	$500x10^3$	$1x10^6$	$2x10^6$
1010 (÷11)	$11\ x10^3$	$22x10^3$	$55x10^3$	$110x10^3$	$220x10^3$	$550x10^3$	$1.1x10^6$	$2.2x10^6$
1011 (÷12)	$12x10^3$	$24x10^3$	$60x10^3$	$120x10^3$	$240x10^3$	$600x10^3$	$1.2x10^6$	$2.4x10^6$
1100 (÷13)	$13x10^3$	$26x10^3$	$65x10^3$	$130x10^3$	$260x10^3$	$650x10^3$	$1.3x10^6$	$2.6x10^6$
1101 (÷14)	$14x10^3$	$28x10^3$	$70x10^3$	$140x10^3$	$280x10^3$	$700x10^3$	$1.4x10^6$	$2.8x10^6$
1110 (÷15)	$15x10^3$	$30x10^3$	$75x10^3$	$150x10^3$	$300x10^3$	$750x10^3$	$1.5x10^6$	$3x10^6$
1111 (÷16)	$16x10^3$	$32x10^3$	$80x10^3$	$160x10^3$	$320x10^3$	$800x10^3$	$1.6x10^6$	$3.2x10^6$

There are four different scenarios for the CRG to restart the MCU from wait mode:

- External reset
- Clock monitor reset
- COP reset
- Any interrupt

If the MCU gets an external reset or COP reset during wait mode active, the CRG asynchronously restores all configuration bits in the register space to its default settings and starts the reset generator. After completing the reset sequence processing begins by fetching the normal or COP reset vector. Wait mode is left and the MCU is in run mode again.

If the clock monitor is enabled (CME = 1) the MCU is able to leave wait mode when loss of oscillator/external clock is detected by a clock monitor fail. If the SCME bit is not asserted the CRG generates a clock monitor fail reset (CMRESET). The CRG's behavior for CMRESET is the same compared to external reset, but another reset vector is fetched after completion of the reset sequence. If the SCME bit is asserted the CRG generates a SCM interrupt if enabled (SCMIE = 1). After generating the interrupt the CRG enters self-clock mode and starts the clock quality checker (Section 2.4.1.4, "Clock Quality Checker"). Then the MCU continues with normal operation.If the SCM interrupt is blocked by SCMIE = 0, the SCMIF flag will be asserted and clock quality checks will be performed but the MCU will not wake-up from wait-mode.

If any other interrupt source (e.g., RTI) triggers exit from wait mode, the MCU immediately continues with normal operation. If the PLL has been powered-down during wait mode, the PLLSEL bit is cleared and the MCU runs on OSCCLK after leaving wait mode. The software must manually set the PLLSEL bit again, in order to switch system and core clocks to the PLLCLK.

If wait mode is entered from self-clock mode the CRG will continue to check the clock quality until clock check is successful. The PLL and voltage regulator (VREG) will remain enabled.

Table 2-12 summarizes the outcome of a clock loss while in wait mode.

2.4.3.3 System Stop Mode

All clocks are stopped in STOP mode, dependent of the setting of the PCE, PRE, and PSTP bit. The oscillator is disabled in STOP mode unless the PSTP bit is set. All counters and dividers remain frozen but do not initialize. If the PRE or PCE bits are set, the RTI or COP continues to run in pseudo stop mode. In addition to disabling system and core clocks the CRG requests other functional units of the MCU (e.g., voltage-regulator) to enter their individual power saving modes (if available). This is the main difference between pseudo stop mode and wait mode.

If the PLLSEL bit is still set when entering stop mode, the CRG will switch the system and core clocks to OSCCLK by clearing the PLLSEL bit. Then the CRG disables the PLL, disables the core clock and finally disables the remaining system clocks. As soon as all clocks are switched off, stop mode is active.

If pseudo stop mode (PSTP = 1) is entered from self-clock mode, the CRG will continue to check the clock quality until clock check is successful. The PLL and the voltage regulator (VREG) will remain enabled. If full stop mode (PSTP = 0) is entered from self-clock mode, an ongoing clock quality check will be stopped. A complete timeout window check will be started when stop mode is left again.

Wake-up from stop mode also depends on the setting of the PSTP bit.

5.4.2.2 General Purpose Digital Input Port Operation

The input channel pins can be multiplexed between analog and digital data. As analog inputs, they are multiplexed and sampled to supply signals to the A/D converter. As digital inputs, they supply external input data that can be accessed through the digital port register PORTAD (input-only).

The analog/digital multiplex operation is performed in the input pads. The input pad is always connected to the analog inputs of the ATD. The input pad signal is buffered to the digital port registers. This buffer can be turned on or off with the ATDDIEN register. This is important so that the buffer does not draw excess current when analog potentials are presented at its input.

5.4.2.3 Low Power Modes

The ATD can be configured for lower MCU power consumption in 3 different ways:

1. Stop mode: This halts A/D conversion. Exit from stop mode will resume A/D conversion, but due to the recovery time the result of this conversion should be ignored.
2. Wait mode with AWAI = 1: This halts A/D conversion. Exit from wait mode will resume A/D conversion, but due to the recovery time the result of this conversion should be ignored.
3. Writing ADPU = 0 (Note that all ATD registers remain accessible.): This aborts any A/D conversion in progress.

Note that the reset value for the ADPU bit is zero. Therefore, when this module is reset, it is reset into the power down state.

5.5 Resets

At reset the ATD is in a power down state. The reset state of each individual bit is listed within the Register Description section (see Section 5.3, “Memory Map and Register Definition”), which details the registers and their bit-field.

5.6 Interrupts

The interrupt requested by the ATD is listed in Table 5-24. Refer to the device overview chapter for related vector address and priority.

Table 5-24. ATD Interrupt Vectors

Interrupt Source	CCR Mask	Local Enable
Sequence complete interrupt	I bit	ASCIE in ATDCTL2

See register descriptions for further details.

When PACN1 overflows from 0x00FF to 0x0000, the interrupt flag PBOVF in PBFLG is set.

Full count register access will take place in one clock cycle.

NOTE

A separate read/write for high byte and low byte will give a different result than accessing them as a word.

When clocking pulse and write to the registers occurs simultaneously, write takes priority and the register is not incremented.

Either left aligned or center aligned output mode can be used in concatenated mode and is controlled by the low order CAEx bit. The high order CAEx bit has no effect.

Table 8-11 is used to summarize which channels are used to set the various control bits when in 16-bit mode.

Table 8-11. 16-bit Concatenation Mode Summary

CONxx	PWMEx	PPOLx	PCLKx	CAEx	PWMx Output
CON67	PWME7	PPOL7	PCLK7	CAE7	PWM7
CON45	PWME5	PPOL5	PCLK5	CAE5	PWM5
CON23	PWME3	PPOL3	PCLK3	CAE3	PWM3
CON01	PWME1	PPOL1	PCLK1	CAE1	PWM1

8.4.2.8 PWM Boundary Cases

Table 8-12 summarizes the boundary conditions for the PWM regardless of the output mode (left aligned or center aligned) and 8-bit (normal) or 16-bit (concatenation).

Table 8-12. PWM Boundary Cases

PWMDTYx	PWMPERx	PPOLx	PWMx Output
$00 (indicates no duty)	>$00	1	Always low
$00 (indicates no duty)	>$00	0	Always high
XX	$00[1] (indicates no period)	1	Always high
XX	$00[1] (indicates no period)	0	Always low
>= PWMPERx	XX	1	Always high
>= PWMPERx	XX	0	Always low

[1] Counter = $00 and does not count.

8.5 Resets

The reset state of each individual bit is listed within the Section 8.3.2, “Register Descriptions” which details the registers and their bit-fields. All special functions or modes which are initialized during or just following reset are described within this section.

- The 8-bit up/down counter is configured as an up counter out of reset.
- All the channels are disabled and all the counters do not count.

9.3.2.2 IIC Frequency Divider Register (IBFD)

	7	6	5	4	3	2	1	0
R W	IBC7	IBC6	IBC5	IBC4	IBC3	IBC2	IBC1	IBC0
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 9-4. IIC Bus Frequency Divider Register (IBFD)

Read and write anytime

Table 9-2. IBFD Field Descriptions

Field	Description
7:0 IBC[7:0]	**I Bus Clock Rate 7:0** — This field is used to prescale the clock for bit rate selection. The bit clock generator is implemented as a prescale divider — IBC7:6, prescaled shift register — IBC5:3 select the prescaler divider and IBC2-0 select the shift register tap point. The IBC bits are decoded to give the tap and prescale values as shown in Table 9-3.

Table 9-3. I-Bus Tap and Prescale Values

IBC2-0 (bin)	SCL Tap (clocks)	SDA Tap (clocks)
000	5	1
001	6	1
010	7	2
011	8	2
100	9	3
101	10	3
110	12	4
111	15	4

IBC5-3 (bin)	scl2start (clocks)	scl2stop (clocks)	scl2tap (clocks)	tap2tap (clocks)
000	2	7	4	1
001	2	7	4	2
010	2	9	6	4
011	6	9	6	8
100	14	17	14	16
101	30	33	30	32
110	62	65	62	64
111	126	129	126	128

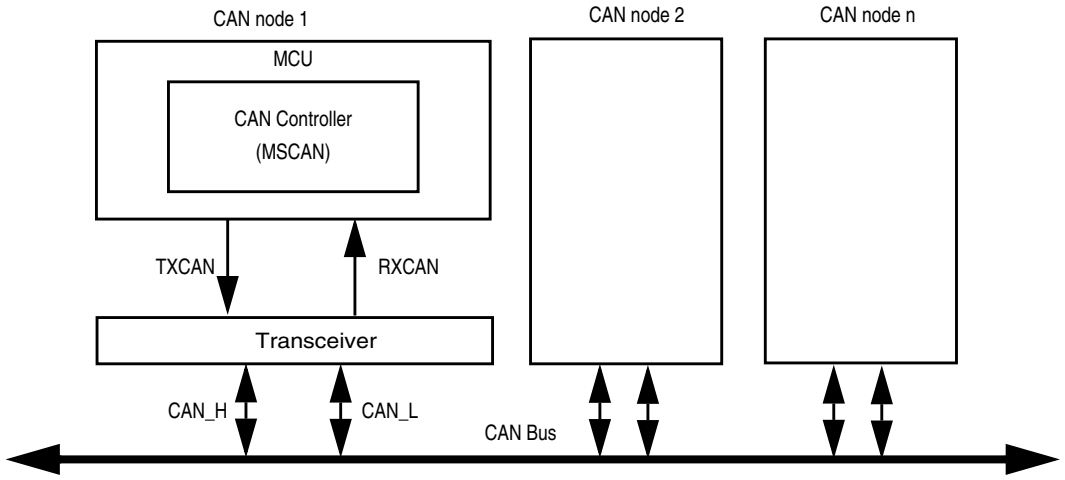

Figure 10-2. CAN System

10.3 Memory Map and Register Definition

This section provides a detailed description of all registers accessible in the MSCAN.

10.3.1 Module Memory Map

Figure 10-3 gives an overview on all registers and their individual bits in the MSCAN memory map. The *register address* results from the addition of *base address* and *address offset*. The *base address* is determined at the MCU level and can be found in the MCU memory map description. The *address offset* is defined at the module level.

The MSCAN occupies 64 bytes in the memory space. The base address of the MSCAN module is determined at the MCU level when the MCU is defined. The register decode map is fixed and begins at the first address of the module address offset.

The detailed register descriptions follow in the order they appear in the register map.

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
0x0000 CANCTL0	R	RXFRM	RXACT	CSWAI	SYNCH	TIME	WUPE	SLPRQ	INITRQ
	W								
0x0001 CANCTL1	R	CANE	CLKSRC	LOOPB	LISTEN	BORM	WUPM	SLPAK	INITAK
	W								

= Unimplemented or Reserved u = Unaffected

Figure 10-3. MSCAN Register Summary

When the transmit shift register is not transmitting a frame, the TXD pin goes to the idle condition, logic 1. If at any time software clears the TE bit in SCI control register 2 (SCICR2), the transmitter enable signal goes low and the transmit signal goes idle.

If software clears TE while a transmission is in progress (TC = 0), the frame in the transmit shift register continues to shift out. To avoid accidentally cutting off the last frame in a message, always wait for TDRE to go high after the last frame before clearing TE.

To separate messages with preambles with minimum idle line time, use this sequence between messages:

1. Write the last byte of the first message to SCIDRH/L.
2. Wait for the TDRE flag to go high, indicating the transfer of the last frame to the transmit shift register.
3. Queue a preamble by clearing and then setting the TE bit.
4. Write the first byte of the second message to SCIDRH/L.

11.4.5.3 Break Characters

Writing a logic 1 to the send break bit, SBK, in SCI control register 2 (SCICR2) loads the transmit shift register with a break character. A break character contains all logic 0s and has no start, stop, or parity bit. Break character length depends on the M bit in SCI control register 1 (SCICR1). As long as SBK is at logic 1, transmitter logic continuously loads break characters into the transmit shift register. After software clears the SBK bit, the shift register finishes transmitting the last break character and then transmits at least one logic 1. The automatic logic 1 at the end of a break character guarantees the recognition of the start bit of the next frame.

The SCI recognizes a break character when there are 10 or 11(M = 0 or M = 1) consecutive zero received. Depending if the break detect feature is enabled or not receiving a break character has these effects on SCI registers.

If the break detect feature is disabled (BKDFE = 0):

- Sets the framing error flag, FE
- Sets the receive data register full flag, RDRF
- Clears the SCI data registers (SCIDRH/L)
- May set the overrun flag, OR, noise flag, NF, parity error flag, PE, or the receiver active flag, RAF (see 3.4.4 and 3.4.5 SCI Status Register 1 and 2)

If the break detect feature is enabled (BKDFE = 1) there are two scenarios[1]

The break is detected right from a start bit or is detected during a byte reception.

- Sets the break detect interrupt flag, BLDIF
- Does not change the data register full flag, RDRF or overrun flag OR
- Does not change the framing error flag FE, parity error flag PE.
- Does not clear the SCI data registers (SCIDRH/L)
- May set noise flag NF, or receiver active flag RAF.

1. A Break character in this context are either 10 or 11 consecutive zero received bits

11.4.6.5 Baud Rate Tolerance

A transmitting device may be operating at a baud rate below or above the receiver baud rate. Accumulated bit time misalignment can cause one of the three stop bit data samples (RT8, RT9, and RT10) to fall outside the actual stop bit. A noise error will occur if the RT8, RT9, and RT10 samples are not all the same logical values. A framing error will occur if the receiver clock is misaligned in such a way that the majority of the RT8, RT9, and RT10 stop bit samples are a logic zero.

As the receiver samples an incoming frame, it re-synchronizes the RT clock on any valid falling edge within the frame. Re synchronization within frames will correct a misalignment between transmitter bit times and receiver bit times.

11.4.6.5.1 Slow Data Tolerance

Figure 11-28 shows how much a slow received frame can be misaligned without causing a noise error or a framing error. The slow stop bit begins at RT8 instead of RT1 but arrives in time for the stop bit data samples at RT8, RT9, and RT10.

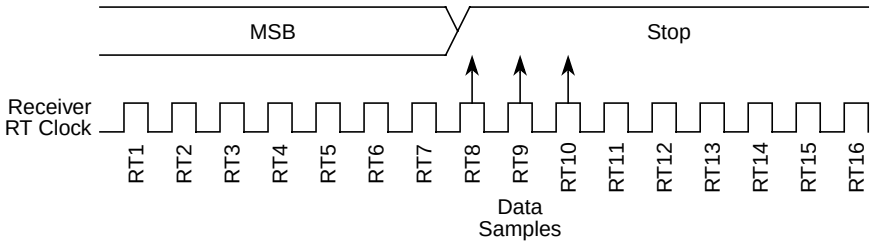

Figure 11-28. Slow Data

Let’s take RTr as receiver RT clock and RTt as transmitter RT clock.

For an 8-bit data character, it takes the receiver 9 bit times x 16 RTr cycles +7 RTr cycles = 151 RTr cycles to start data sampling of the stop bit.

With the misaligned character shown in Figure 11-28, the receiver counts 151 RTr cycles at the point when the count of the transmitting device is 9 bit times x 16 RTt cycles = 144 RTt cycles.

The maximum percent difference between the receiver count and the transmitter count of a slow 8-bit data character with no errors is:

((151 – 144) / 151) x 100 = 4.63%

For a 9-bit data character, it takes the receiver 10 bit times x 16 RTr cycles + 7 RTr cycles = 167 RTr cycles to start data sampling of the stop bit.

With the misaligned character shown in Figure 11-28, the receiver counts 167 RTr cycles at the point when the count of the transmitting device is 10 bit times x 16 RTt cycles = 160 RTt cycles.

The maximum percent difference between the receiver count and the transmitter count of a slow 9-bit character with no errors is:

((167 – 160) / 167) X 100 = 4.19%

Chapter 17
Memory Mapping Control (S12XMMCV2)

17.1 Introduction

This section describes the functionality of the module mapping control (MMC) sub-block of the S12X platform. The block diagram of the MMC is shown in Figure 1-1.

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

17.1.1 Features

The main features of this block are:

- Paging capability to support a global 8 Mbytes memory address space
- Bus arbitration between the masters CPU, BDM, and XGATE
- Simultaneous accesses to different resources[1] (internal, external, and peripherals) (see Figure 1-1)
- Resolution of target bus access collision
- Access restriction control from masters to some targets (e.g., RAM write access protection for user specified areas)
- MCU operation mode control
- MCU security control
- Separate memory map schemes for each master CPU, BDM, and XGATE
- ROM control bits to enable the on-chip FLASH or ROM selection
- Port replacement registers access control
- Generation of system reset when CPU accesses an unimplemented address (i.e., an address which does not belong to any of the on-chip modules) in single-chip modes

17.1.2 Modes of Operation

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

17.1.2.1 Power Saving Modes

- Run mode

  MMC is functional during normal run mode.

1. Resources are also called targets.

```
;automatically select direct mode.
```

18.3.2.5 MMC Control Register (MMCCTL1)

Address: 0x0013 PRR

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	EROMON	ROMHM	ROMON
W								
Reset	0	0	0	0	0	EROMCTL	0	ROMCTL

= Unimplemented or Reserved

Figure 18-10. MMC Control Register (MMCCTL1)

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

CAUTION

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

Table 18-10. MMCCTL1 Field Descriptions

Field	Description
2 EROMON	**Enables emulated Flash or ROM memory in the memory map** Write: Never This bit is used in some modes to define the placement of the Emulated Flash or ROM (Refer to Table 18-11) 0 Disables the emulated Flash or ROM in the memory map. 1 Enables the emulated Flash or ROM in the memory map.
1 ROMHM	**FLASH or ROM only in higher Half of Memory Map** Write: Once in normal and emulation modes and anytime in special modes 0 The fixed page of Flash or ROM can be accessed in the lower half of the memory map. Accesses to 0x4000–0x7FFF will be mapped to 0x7F_4000-0x7F_7FFF in the global memory space. 1 Disables access to the Flash or ROM in the lower half of the memory map.These physical locations of the Flash or ROM can still be accessed through the program page window. Accesses to 0x4000–0x7FFF will be mapped to 0x14_4000-0x14_7FFF in the global memory space (external access).
0 ROMON	**Enable FLASH or ROM in the memory map** Write: Once in normal and emulation modes and anytime in special modes. This bit is used in some modes to define the placement of the ROM (Refer to Table 18-11) 0 Disables the Flash or ROM from the memory map. 1 Enables the Flash or ROM in the memory map.

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

18.3.2.7 EEPROM Page Index Register (EPAGE)

Address: 0x0017

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

Figure 18-13. EEPROM Page Index Register (EPAGE)

Read: Anytime

Write: Anytime

These eight index bits are used to page 1 KByte blocks into the EEPROM page window located in the local (CPU or BDM) memory map from address 0x0800 to address 0x0BFF (see Figure 18-14). This supports accessing up to 256 Kbytes of EEPROM (in the Global map) within the 64 KByte Local map. The EEPROM page index register is effectively used to construct paged EEPROM addresses in the Local map format.

CAUTION

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

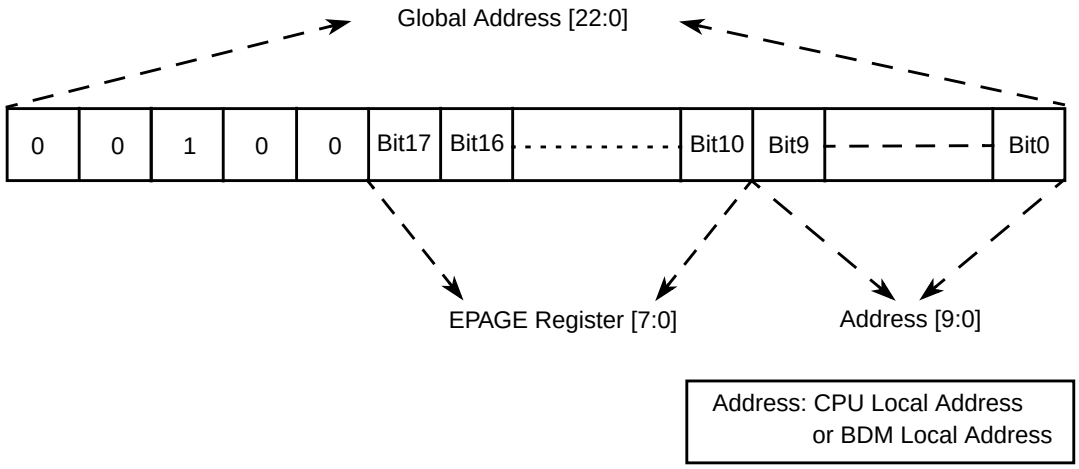

Figure 18-14. EPAGE Address Mapping

Table 18-13. EPAGE Field Descriptions

Field	Description
7–0 EP[7:0]	**EEPROM Page Index Bits 7–0** — These page index bits are used to select which of the 256 EEPROM array pages is to be accessed in the EEPROM Page Window.

The reset value of 0xFE ensures that there is a linear EEPROM space available between addresses 0x0800 and 0x0FFF out of reset.

The fixed 1K page 0x0C00–0x0FFF of EEPROM is equivalent to page 255 (page number 0xFF).

Table 19-2. External System Pins Associated With DBG

Pin Name	Pin Functions	Description
$\overline{\text{TAGHI}}$ (See DUG)	$\overline{\text{TAGHI}}$	When instruction tagging is on, tags the high half of the instruction word being read into the instruction queue.
$\overline{\text{TAGLO}}$ (See DUG)	$\overline{\text{TAGLO}}$	When instruction tagging is on, tags the low half of the instruction word being read into the instruction queue.

19.4.4.1 Final State

On entering final state a trigger may be issued to the trace buffer according to the trace position control as defined by the TALIGN field (see Section 19.3.1.3, “Debug Trace Control Register (DBGTCR)”). If the TSOURCE bits in the trace control register DBGTCR are cleared then the trace buffer is disabled and the transition to final state can only generate a breakpoint request. In this case or upon completion of a tracing session when tracing is enabled, the ARM bit in the DBGC1 register is cleared, returning the module to the disarmed state0. If tracing is enabled a breakpoint request can occur at the end of the tracing session.

19.4.5 Trace Buffer Operation

The trace buffer is a 64 lines deep by 64-bits wide RAM array. The DBG module stores trace information in the RAM array in a circular buffer format. The CPU accesses the RAM array through a register window (DBGTBH:DBGTBL) using 16-bit wide word accesses. After each complete 64-bit trace buffer line is read via the CPU, an internal pointer into the RAM is incremented so that the next read will receive fresh information. Data is stored in the format shown in Table 19-39. After each store the counter register bits DBGCNT[6:0] are incremented. Tracing of CPU activity is disabled when the BDM is active but tracing of XGATE activity is still possible. Reading the trace buffer while the BDM is active returns invalid data and the trace buffer pointer is not incremented.

19.4.5.1 Trace Trigger Alignment

Using the TALIGN bits (see Section 19.3.1.3, “Debug Trace Control Register (DBGTCR)”) it is possible to align the trigger with the end, the middle or the beginning of a tracing session.

If end or mid tracing is selected, tracing begins when the ARM bit in DBGC1 is set and State1 is entered. The transition to final state if end is selected signals the end of the tracing session. The transition to final state if mid is selected signals that another 32 lines will be traced before ending the tracing session. Tracing with begin-trigger starts at the opcode of the trigger.

19.4.5.1.1 Storing with Begin-Trigger

Storing with begin-trigger, data is not stored in the trace buffer until the final state is entered. Once the trigger condition is met the DBG module will remain armed until 64 lines are stored in the trace buffer. If the trigger is at the address of the change-of-flow instruction the change of flow associated with the trigger will be stored in the trace buffer. Using begin-trigger together with tagging, if the tagged instruction is about to be executed then the trace is started. Upon completion of the tracing session the breakpoint is generated, thus the breakpoint does not occur at the tagged instruction boundary.

19.4.5.1.2 Storing with Mid-Trigger

Storing with mid-trigger, data is stored in the trace buffer as soon as the DBG module is armed. When the trigger condition is met, another 32 lines will be traced before ending the tracing session, irrespective of the number of lines stored before the trigger occurred, then the DBG module is disarmed and no more data is stored. If the trigger is at the address of a change of flow instruction the trigger event is not stored in the trace buffer. Using mid-trigger with tagging, if the tagged instruction is about to be executed then the trace

Table 21-1. External System Signals Associated with XEBI

Signal	I[1]/O	EBI Signal Multiplex (T)ime[2] (F)unction[3]		Description	Available in Modes					
					NS	SS	NX	ES	EX	ST
$\overline{RE}$	O	—	—	Read Enable, indicates external read access	No	No	Yes	No	No	No
ADDR[22:20]	O	T	—	External address	No	No	Yes	Yes	Yes	Yes
ACC[2:0]	O		—	Access source	No	No	No	Yes	Yes	Yes
ADDR[19:16]	O	T	—	External address	No	No	Yes	Yes	Yes	Yes
IQSTAT[3:0]	O		—	Instruction Queue Status	No	No	No	Yes	Yes	Yes
ADDR[15:1]	O	T	—	External address	No	No	Yes	Yes	Yes	Yes
IVD[15:1]	O		—	Internal visibility read data (IVIS = 1)	No	No	No	Yes	Yes	Yes
ADDR0	O	T	F	External address	No	No	No	Yes	Yes	Yes
IVD0	O			Internal visibility read data (IVIS = 1)	No	No	No	Yes	Yes	Yes
$\overline{UDS}$	O	—		Upper Data Select, indicates external access to the high byte DATA[15:8]	No	No	Yes	No	No	No
$\overline{LSTRB}$	O	—	F	Low Strobe, indicates valid data on DATA[7:0]	No	No	No	Yes	Yes	Yes
$\overline{LDS}$	O	—		Lower Data Select, indicates external access to the low byte DATA[7:0]	No	No	Yes	No	No	No
R/$\overline{W}$	O	—	F	Read/Write, indicates the direction of internal data transfers	No	No	No	Yes	Yes	Yes
$\overline{WE}$	O	—		Write Enable, indicates external write access	No	No	Yes	No	No	No
DATA[15:8]	I/O	—	—	Bidirectional data (even address)	No	No	Yes	Yes	Yes	Yes
DATA[7:0]	I/O	—	—	Bidirectional data (odd address)	No	No	Yes	Yes	Yes	Yes
$\overline{EWAIT}$	I	—	—	External control for external bus access stretches (adding wait states)	No	No	Yes	No	Yes	No

1 All inputs are capable of reducing input threshold level

2 Time-multiplex means that the respective signals share the same pin on chip level and are active alternating in a dedicated time slot (in modes where applicable).

3 Function-multiplex means that one of the respective signals sharing the same pin on chip level continuously uses the pin depending on configuration and reset state.

22.3.2.31 Port M Input Register (PTIM)

	7	6	5	4	3	2	1	0
R	PTIM7	PTIM6	PTIM5	PTIM4	PTIM3	PTIM2	PTIM1	PTIM0
W								
Reset[1]	—	—	—	—	—	—	—	—

= Unimplemented or Reserved

Figure 22-33. Port M Input Register (PTIM)

1 These registers are reset to zero. Two bus clock cycles after reset release the register values are updated with the associated pin values.

Read: Anytime.

Write: Never, writes to this register have no effect.

This register always reads back the buffered state of the associated pins. This can also be used to detect overload or short circuit conditions on output pins.

22.4.2.3 Port C and D

Port C pins PC[7:0] and port D pins PD[7:0] can be used for either general-purpose I/O, or, in 144-pin packages, also with the external bus interface. In this case port C and port D are associated with the external data bus inputs/outputs DATA15–DATA8 and DATA7–DATA0, respectively.

These pins are configured for reduced input threshold in certain operating modes (refer to S12X_EBI section).

NOTE

Port C and D are neither available in 112-pin nor in 80-pin packages.

22.4.2.4 Port E

Port E is associated with the external bus control outputs R/$\overline{W}$, $\overline{LSTRB}$, $\overline{LDS}$ and $\overline{RE}$, the free-running clock outputs ECLK and ECLK2X, as well as with the $\overline{TAGHI}$, $\overline{TAGLO}$, MODA and MODB and interrupt inputs $\overline{IRQ}$ and $\overline{XIRQ}$.

Port E pins PE[7:2] can be used for either general-purpose I/O or with the alternative functions.

Port E pin PE[7] an be used for either general-purpose I/O or as the free-running clock ECLKX2 output running at the core clock rate. The clock output is always enabled in emulation modes.

Port E pin PE[4] an be used for either general-purpose I/O or as the free-running clock ECLK output running at the bus clock rate or at the programmed divided clock rate. The clock output is always enabled in emulation modes.

Port E pin PE[1] can be used for either general-purpose input or as the level- or falling edge-sensitive $\overline{IRQ}$ interrupt input. $\overline{IRQ}$ will be enabled by setting the IRQEN configuration bit (Section 22.3.2.14, “IRQ Control Register (IRQCR)”) and clearing the I-bit in the CPU’s condition code register. It is inhibited at reset so this pin is initially configured as a simple input with a pull-up.

Port E pin PE[0] can be used for either general-purpose input or as the level-sensitive $\overline{XIRQ}$ interrupt input. $\overline{XIRQ}$ can be enabled by clearing the X-bit in the CPU’s condition code register. It is inhibited at reset so this pin is initially configured as a high-impedance input with a pull-up.

Port E pins PE[5] and PE[6] are configured for reduced input threshold in certain modes (refer to S12X_EBI section).

22.4.2.5 Port K

Port K pins PK[7:0] can be used for either general-purpose I/O, or, in 144-pin packages, also with the external bus interface. In this case port K pins PK[6:0] are associated with the external address bus outputs ADDR22–ADDR16 and PK7 is associated to the $\overline{EWAIT}$ input.

Port K pin PE[7] is configured for reduced input threshold in certain modes (refer to S12X_EBI section).

NOTE

Port K is not available in 80-pin packages. PK[6] is not available in 112-pin packages.

then 182 kHz. In this case, the EEPROM program and erase algorithm timings are increased over the optimum target by:

$$(200 - 182) / 200 \times 100 \; = \; 9\%$$

CAUTION

Program and erase command execution time will increase proportionally with the period of EECLK. Because of the impact of clock synchronization on the accuracy of the functional timings, programming or erasing the EEPROM memory cannot be performed if the bus clock runs at less than 1 MHz. Programming or erasing the EEPROM memory with EECLK < 150 kHz should be avoided. Setting ECLKDIV to a value such that EECLK < 150 kHz can destroy the EEPROM memory due to overstress. Setting ECLKDIV to a value such that (1/EECLK+Tbus) < 5 µs can result in incomplete programming or erasure of the EEPROM memory cells.

If the ECLKDIV register is written, the EDIVLD bit is set automatically. If the EDIVLD bit is 0, the ECLKDIV register has not been written since the last reset. If the ECLKDIV register has not been written to, the EEPROM command loaded during a command write sequence will not execute and the ACCERR flag in the ESTAT register will set.

0x000E–0x000F External Bus Interface (S12XEBI) Map

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x000E	EBICTL0	R	ITHRS	0	HDBE	ASIZ4	ASIZ3	ASIZ2	ASIZ1	ASIZ0
		W								
0x000F	EBICTL1	R	EWAITE	0	0	0	0	EXSTR2	EXSTR1	EXSTR0
		W								

0x0010–0x0017 Module Mapping Control (S12XMMC) Map 2 of 4

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0010	GPAGE	R	0	GP6	GP5	GP4	GP3	GP2	GP1	GP0
		W								
0x0011	DIRECT	R	DP15	DP14	DP13	DP12	DP11	DP10	DP9	DP8
		W								
0x0012	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x0013	MMCCTL1	R	0	0	0	0	0	EROMON	ROMHM	ROMON
		W								
0x0014	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x0015	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x0016	RPAGE	R	RP7	RP6	RP5	RP4	RP3	RP2	RP1	RP0
		W								
0x0017	EPAGE	R	EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0
		W								

0x0018–0x001B Miscellaneous Peripheral

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0018	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x0019	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x001A	PARTIDH	R	1	1	0	0	0	1	0	0
		W								
0x001B	PARTIDL	R	0	0	0	0	0	0	0	0
		W								

0x001C–0x001F Port Integration Module (PIM) Map 3 of 5

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x001C	ECLKCTL	R	NECLK	NCLKX2	0	0	0	0	EDIV1	EDIV0
		W								