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	Active
Core Processor	HCS12
Core Size	16-Bit
Speed	25MHz
Connectivity	EBI/EMI, SCI, SPI
Peripherals	POR, PWM, WDT
Number of I/O	60
Program Memory Size	64KB (64K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	4K x 8
Voltage - Supply (Vcc/Vdd)	2.35V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	80-QFP
Supplier Device Package	80-QFP (14x14)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mcs12gc64cfue

0x00C8–0x00CF SCI (Asynchronous Serial Interface) (continued)

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x00CD	SCISR2	Read:	0	0	0	0	0	BRK13	TXDIR	RAF
		Write:								
0x00CE	SCIDRH	Read:	R8	T8	0	0	0	0	0	0
		Write:								
0x00CF	SCIDRL	Read:	R7	R6	R5	R4	R3	R2	R1	R0
		Write:	T7	T6	T5	T4	T3	T2	T1	T0

0x00D0–0x00D7 Reserved

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x00D0–0x00D7	Reserved	Read:	0	0	0	0	0	0	0	0
		Write:								

0x00D8–0x00DF SPI (Serial Peripheral Interface)

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x00D8	SPICR1	Read: Write:	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE
0x00D9	SPICR2	Read:	0	0	0	MODFEN	BIDIROE	0	SPISWAI	SPC0
		Write:								
0x00DA	SPIBR	Read:	0	SPPR2	SPPR1	SPPR0	0	SPR2	SPR1	SPR0
		Write:								
0x00DB	SPISR	Read:	SPIF	0	SPTEF	MODF	0	0	0	0
		Write:								
0x00DC	Reserved	Read:	0	0	0	0	0	0	0	0
		Write:								
0x00DD	SPIDR	Read: Write:	Bit7	6	5	4	3	2	1	Bit0
0x00DE	Reserved	Read:	0	0	0	0	0	0	0	0
		Write:								
0x00DF	Reserved	Read:	0	0	0	0	0	0	0	0
		Write:								

1.2.3 Part ID Assignments

The part ID is located in two 8-bit registers PARTIDH and PARTIDL (addresses 0x001A and ox001B after reset). The read-only value is a unique part ID for each revision of the chip. Table 1-3 shows the assigned part ID numbers for production mask sets.

Table 1-3. Assigned Part ID Numbers

Device	Mask Set Number	Part ID[1]
MC9S12C32	1L45J	$3300
MC9S12C32	2L45J	$3302
MC9S12C32	1M34C	$3311
MC9S12GC16	2L45J	$3302
MC9S12GC32	2L45J	$3302
MC9S12GC32	1M34C	$3311
MC9S12C64,MC9S12C96,MC9S12C128	2L09S	$3102
MC9S12GC64,MC9S12GC96,MC9S12GC128	2L09S	$3102
MC9S12C64,MC9S12C96,MC9S12C128	0M66G	$3103
MC9S12GC64,MC9S12GC96,MC9S12GC128	0M66G	$3103

1. The coding is as follows:
   Bit 15–12: Major family identifier
   Bit 11–8: Minor family identifier
   Bit 7–4: Major mask set revision number including FAB transfers
   Bit 3–0: Minor — non full — mask set revision

The device memory sizes are located in two 8-bit registers MEMSIZ0 and MEMSIZ1 (addresses 0x001C and 0x001D after reset). Table 1-4 shows the read-only values of these registers. Refer to Module Mapping and Control (MMC) Block Guide for further details.

Table 1-4. Memory Size Registers

Device	Register Name	Value
MC9S12GC16	MEMSIZ0	$00
	MEMSIZ1	$80
MC9S12C32, MC9S12GC32	MEMSIZ0	$00
	MEMSIZ1	$80
MC9S12C64, MC9S12GC64	MEMSIZ0	$01
	MEMSIZ1	$C0
MC9S12C96,MC9S12GC96	MEMSIZ0	$01
	MEMSIZ1	$C0
MC9S12C128, MC9S12GC128	MEMSIZ0	$01
	MEMSIZ1	$C0

the Flash EEPROM memory in the memory map (ROMCTL). At the rising edge of $\overline{\text{RESET}}$, the state of this pin is latched to the ROMON bit.

- PP6 = 1 in emulation modes equates to ROMON = 0 (ROM space externally mapped)
- PP6 = 0 in expanded modes equates to ROMON = 0 (ROM space externally mapped)

1.3.4.19 PP[5:0] / KWP[5:0] / PW[5:0] — Port P I/O Pins [5:0]

PP[5:0] are general purpose input or output pins, shared with the keypad interrupt function. When configured as inputs, they can generate interrupts causing the MCU to exit stop or wait mode.

PP[5:0] are also shared with the PWM output signals, PW[5:0]. Pins PP[2:0] are only available in the 80-pin package version. Pins PP[4:3] are not available in the 48-pin package version.

1.3.4.20 PJ[7:6] / KWJ[7:6] — Port J I/O Pins [7:6]

PJ[7:6] are general purpose input or output pins, shared with the keypad interrupt function. When configured as inputs, they can generate interrupts causing the MCU to exit stop or wait mode. These pins are not available in the 48-pin package version nor in the 52-pin package version.

1.3.4.21 PM5 / SCK — Port M I/O Pin 5

PM5 is a general purpose input or output pin and also the serial clock pin SCK for the serial peripheral interface (SPI).

1.3.4.22 PM4 / MOSI — Port M I/O Pin 4

PM4 is a general purpose input or output pin and also the master output (during master mode) or slave input (during slave mode) pin for the serial peripheral interface (SPI).

1.3.4.23 PM3 / $\overline{\text{SS}}$ — Port M I/O Pin 3

PM3 is a general purpose input or output pin and also the slave select pin $\overline{\text{SS}}$ for the serial peripheral interface (SPI).

1.3.4.24 PM2 / MISO — Port M I/O Pin 2

PM2 is a general purpose input or output pin and also the master input (during master mode) or slave output (during slave mode) pin for the serial peripheral interface (SPI).

1.3.4.25 PM1 / TXCAN — Port M I/O Pin 1

PM1 is a general purpose input or output pin and the transmit pin, TXCAN, of the CAN module if available.

1.3.4.26 PM0 / RXCAN — Port M I/O Pin 0

PM0 is a general purpose input or output pin and the receive pin, RXCAN, of the CAN module if available.

2.3.2.3.6 Port M Polarity Select Register (PPSM)

Module Base + 0x0015

	7	6	5	4	3	2	1	0
R	0	0	PPSM5	PPSM4	PPSM3	PPSM2	PPSM1	PPSM0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 2-22. Port M Polarity Select Register (PPSM)

Read: Anytime.

Write: Anytime.

Table 2-20. PPSM Field Descriptions

Field	Description
5–0 PPSM[5:0]	**Polarity Select Port M** — This register selects whether a pull-down or a pull-up device is connected to the pin. 0 A pull-up device is connected to the associated port M pin, if enabled by the associated bit in register PERM and if the port is used as input or as wired-or output. 1 A pull-down device is connected to the associated port M pin, if enabled by the associated bit in register PERM and if the port is used as input.

2.3.2.3.7 Port M Wired-OR Mode Register (WOMM)

Module Base + 0x0016

	7	6	5	4	3	2	1	0
R	0	0	WOMM5	WOMM4	WOMM3	WOMM2	WOMM1	WOMM0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 2-23. Port M Wired-OR Mode Register (WOMM)

Read: Anytime.

Write: Anytime.

Table 2-21. WOMM Field Descriptions

Field	Description
5–0 WOMM[5:0]	**Wired-OR Mode Port M** — This register configures the output pins as wired-or. If enabled the output is driven active low only (open-drain). A logic level of “1” is not driven. This bit has no influence on pins used as inputs. 0 Output buffers operate as push-pull outputs. 1 Output buffers operate as open-drain outputs.

Table 4-1. External System Pins Associated With MEBI (continued)

Pin Name	Pin Functions	Description
PE4/ECLK	PE4	General-purpose I/O pin, see PORTE and DDRE registers.
	ECLK	Bus timing reference clock, can operate as a free-running clock at the system clock rate or to produce one low-high clock per visible access, with the high period stretched for slow accesses. ECLK is controlled by the NECLK bit in PEAR, the IVIS bit in MODE, and the ESTR bit in EBICTL.
PE3/$\overline{\text{LSTRB}}$/ $\overline{\text{TAGLO}}$	PE3	General-purpose I/O pin, see PORTE and DDRE registers.
	$\overline{\text{LSTRB}}$	Low strobe bar, 0 indicates valid data on D7–D0.
	SZ8	In special peripheral mode, this pin is an input indicating the size of the data transfer (0 = 16-bit; 1 = 8-bit).
	$\overline{\text{TAGLO}}$	In expanded wide mode or emulation narrow modes, when instruction tagging is on and low strobe is enabled, a 0 at the falling edge of E tags the low half of the instruction word being read into the instruction queue.
PE2/R/$\overline{\text{W}}$	PE2	General-purpose I/O pin, see PORTE and DDRE registers.
	R/$\overline{\text{W}}$	Read/write, indicates the direction of internal data transfers. This is an output except in special peripheral mode where it is an input.
PE1/$\overline{\text{IRQ}}$	PE1	General-purpose input-only pin, can be read even if $\overline{\text{IRQ}}$ enabled.
	$\overline{\text{IRQ}}$	Maskable interrupt request, can be level sensitive or edge sensitive.
PE0/$\overline{\text{XIRQ}}$	PE0	General-purpose input-only pin.
	$\overline{\text{XIRQ}}$	Non-maskable interrupt input.
PK7/$\overline{\text{ECS}}$	PK7	General-purpose I/O pin, see PORTK and DDRK registers.
	$\overline{\text{ECS}}$	Emulation chip select
PK6/$\overline{\text{XCS}}$	PK6	General-purpose I/O pin, see PORTK and DDRK registers.
	$\overline{\text{XCS}}$	External data chip select
PK5/X19 thru PK0/X14	PK5–PK0	General-purpose I/O pins, see PORTK and DDRK registers.
	X19–X14	Memory expansion addresses

Detailed descriptions of these pins can be found in the device overview chapter.

4.3 Memory Map and Register Definition

A summary of the registers associated with the MEBI sub-block is shown in Table 4-2. Detailed descriptions of the registers and bits are given in the subsections that follow. On most chips the registers are mappable. Therefore, the upper bits may not be all 0s as shown in the table and descriptions.

The interrupt sub-block decodes the priority of all system exception requests and provides the applicable vector for processing the exception. The INT supports I-bit maskable and X-bit maskable interrupts, a non-maskable unimplemented opcode trap, a non-maskable software interrupt (SWI) or background debug mode request, and three system reset vector requests. All interrupt related exception requests are managed by the interrupt sub-block (INT).

5.1.1 Features

The INT includes these features:

- Provides two to 122 I-bit maskable interrupt vectors (0xFF00–0xFFF2)
- Provides one X-bit maskable interrupt vector (0xFFF4)
- Provides a non-maskable software interrupt (SWI) or background debug mode request vector (0xFFF6)
- Provides a non-maskable unimplemented opcode trap (TRAP) vector (0xFFF8)
- Provides three system reset vectors (0xFFFA–0xFFFE) (reset, CMR, and COP)
- Determines the appropriate vector and drives it onto the address bus at the appropriate time
- Signals the CPU that interrupts are pending
- Provides control registers which allow testing of interrupts
- Provides additional input signals which prevents requests for servicing I and X interrupts
- Wakes the system from stop or wait mode when an appropriate interrupt occurs or whenever $\overline{\text{XIRQ}}$ is active, even if $\overline{\text{XIRQ}}$ is masked
- Provides asynchronous path for all I and X interrupts, (0xFF00–0xFFF4)
- (Optional) selects and stores the highest priority I interrupt based on the value written into the HPRIO register

5.1.2 Modes of Operation

The functionality of the INT sub-block in various modes of operation is discussed in the subsections that follow.

- **Normal operation**

  The INT operates the same in all normal modes of operation.
- **Special operation**

  Interrupts may be tested in special modes through the use of the interrupt test registers.
- **Emulation modes**

  The INT operates the same in emulation modes as in normal modes.
- **Low power modes**

  See Section 5.4.1, “Low-Power Modes,” for details

Figure 6-11 shows the ACK handshake protocol in a command level timing diagram. The READ_BYTE instruction is used as an example. First, the 8-bit instruction opcode is sent by the host, followed by the address of the memory location to be read. The target BDM decodes the instruction. A bus cycle is grabbed (free or stolen) by the BDM and it executes the READ_BYTE operation. Having retrieved the data, the BDM issues an ACK pulse to the host controller, indicating that the addressed byte is ready to be retrieved. After detecting the ACK pulse, the host initiates the byte retrieval process. Note that data is sent in the form of a word and the host needs to determine which is the appropriate byte based on whether the address was odd or even.

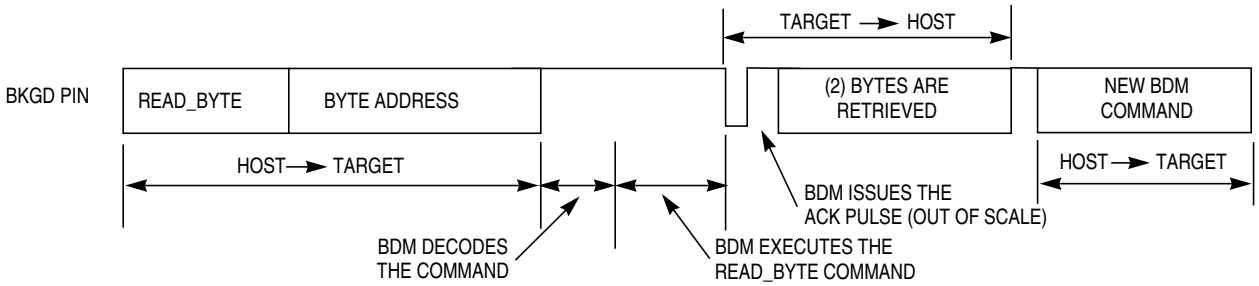


Figure 6-11. Handshake Protocol at Command Level

Differently from the normal bit transfer (where the host initiates the transmission), the serial interface ACK handshake pulse is initiated by the target MCU by issuing a falling edge in the BKGD pin. The hardware handshake protocol in Figure 6-10 specifies the timing when the BKGD pin is being driven, so the host should follow this timing constraint in order to avoid the risk of an electrical conflict in the BKGD pin.

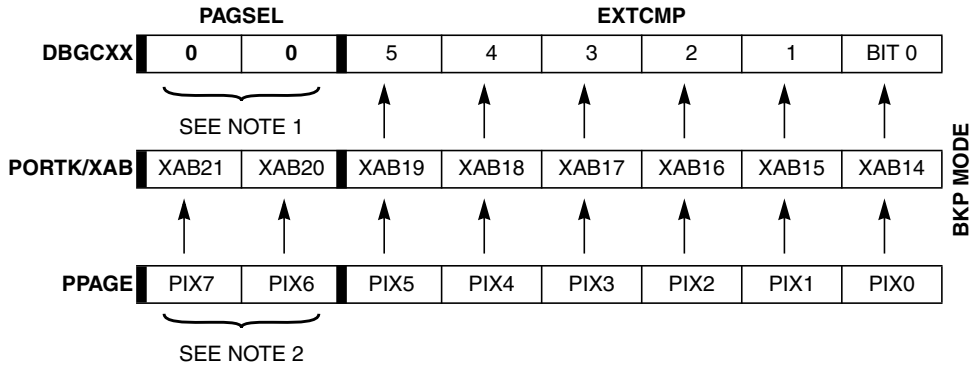
NOTE

The only place the BKGD pin can have an electrical conflict is when one side is driving low and the other side is issuing a speedup pulse (high). Other "highs" are pulled rather than driven. However, at low rates the time of the speedup pulse can become lengthy and so the potential conflict time becomes longer as well.

The ACK handshake protocol does not support nested ACK pulses. If a BDM command is not acknowledge by an ACK pulse, the host needs to abort the pending command first in order to be able to issue a new BDM command. When the CPU enters WAIT or STOP while the host issues a command that requires CPU execution (e.g., WRITE_BYTE), the target discards the incoming command due to the WAIT or STOP being detected. Therefore, the command is not acknowledged by the target, which means that the ACK pulse will not be issued in this case. After a certain time the host should decide to abort the ACK sequence in order to be free to issue a new command. Therefore, the protocol should provide a mechanism in which a command, and therefore a pending ACK, could be aborted.

NOTE

Differently from a regular BDM command, the ACK pulse does not provide a time out. This means that in the case of a WAIT or STOP instruction being executed, the ACK would be prevented from being issued. If not aborted, the ACK would remain pending indefinitely. See the handshake abort procedure described in Section 6.4.8, "Hardware Handshake Abort Procedure."



NOTES:
1. In BKP mode, PAGSEL has no functionality. Therefore, set PAGSEL to 00 (reset state).
2. Current HCS12 implementations are limited to six PPAGE bits, PIX[5:0].

Figure 7-16. Comparators A and B Extended Comparison in BKP Mode

7.3.2.10 Debug Comparator A Register (DBGCA)

Module Base + 0x002B
Starting address location affected by INITRG register setting.

	15	14	13	12	11	10	9	8
R W	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reset	0	0	0	0	0	0	0	0

Figure 7-17. Debug Comparator A Register High (DBGCAH)

Module Base + 0x002C
Starting address location affected by INITRG register setting.

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

Figure 7-18. Debug Comparator A Register Low (DBGCAL)

Table 7-21. DBGCA Field Descriptions

Field	Description
15:0 15:0	**Comparator A Compare Bits** — The comparator A compare bits control whether comparator A compares the address bus bits [15:0] to a logic 1 or logic 0. See Table 7-20. 0 Compare corresponding address bit to a logic 0 1 Compare corresponding address bit to a logic 1

9.3.2.12 CRG COP Timer Arm/Reset Register (ARMCOP)

This register is used to restart the COP time-out period.

Module Base + 0x000B

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

Figure 9-15. ARMCOP Register Diagram

Read: always reads 0x0000

Write: anytime

When the COP is disabled (CR[2:0] = “000”) writing to this register has no effect.

When the COP is enabled by setting CR[2:0] nonzero, the following applies:

Writing any value other than 0x0055 or 0x00AA causes a COP reset. To restart the COP time-out period you must write 0x0055 followed by a write of 0x00AA. Other instructions may be executed between these writes but the sequence (0x0055, 0x00AA) must be completed prior to COP end of time-out period to avoid a COP reset. Sequences of 0x0055 writes or sequences of 0x00AA writes are allowed. When the WCOP bit is set, 0x0055 and 0x00AA writes must be done in the last 25% of the selected time-out period; writing any value in the first 75% of the selected period will cause a COP reset.

9.4 Functional Description

This section gives detailed informations on the internal operation of the design.

9.4.1 Phase Locked Loop (PLL)

The PLL is used to run the MCU from a different time base than the incoming OSCCLK. For increased flexibility, OSCCLK can be divided in a range of 1 to 16 to generate the reference frequency. This offers a finer multiplication granularity. The PLL can multiply this reference clock by a multiple of 2, 4, 6,... 126,128 based on the SYNR register.

$$\mathrm{PLLCLK} = 2 \times \mathrm{OSCCLK} \times \frac{[\mathrm{SYNR} + 1]}{[\mathrm{REFDV} + 1]}$$

CAUTION

Although it is possible to set the two dividers to command a very high clock frequency, do not exceed the specified bus frequency limit for the MCU.
If (PLLSEL = 1), Bus Clock = PLLCLK / 2

The PLL filter can be manually or automatically configured into one of two possible operating modes:

- Acquisition mode

  In acquisition mode, the filter can make large frequency corrections to the VCO. This mode is used at PLL start-up or when the PLL has suffered a severe noise hit and the VCO frequency is far off the desired frequency. When in acquisition mode, the TRACK status bit is cleared in the CRGFLG register.
- Tracking mode

  In tracking mode, the filter makes only small corrections to the frequency of the VCO. PLL jitter is much lower in tracking mode, but the response to noise is also slower. The PLL enters tracking mode when the VCO frequency is nearly correct and the TRACK bit is set in the CRGFLG register.

The PLL can change the bandwidth or operational mode of the loop filter manually or automatically.

In automatic bandwidth control mode (AUTO = 1), the lock detector automatically switches between acquisition and tracking modes. Automatic bandwidth control mode also is used to determine when the PLL clock (PLLCLK) is safe to use as the source for the system and core clocks. If PLL LOCK interrupt requests are enabled, the software can wait for an interrupt request and then check the LOCK bit. If CPU interrupts are disabled, software can poll the LOCK bit continuously (during PLL start-up, usually) or at periodic intervals. In either case, only when the LOCK bit is set, is the PLLCLK clock safe to use as the source for the system and core clocks. If the PLL is selected as the source for the system and core clocks and the LOCK bit is clear, the PLL has suffered a severe noise hit and the software must take appropriate action, depending on the application.

The following conditions apply when the PLL is in automatic bandwidth control mode (AUTO = 1):

- The TRACK bit is a read-only indicator of the mode of the filter.
- The TRACK bit is set when the VCO frequency is within a certain tolerance, Δ_{trk}, and is clear when the VCO frequency is out of a certain tolerance, Δ_{unt}.
- The LOCK bit is a read-only indicator of the locked state of the PLL.
- The LOCK bit is set when the VCO frequency is within a certain tolerance, Δ_{Lock}, and is cleared when the VCO frequency is out of a certain tolerance, Δ_{unl}.
- CPU interrupts can occur if enabled (LOCKIE = 1) when the lock condition changes, toggling the LOCK bit.

The PLL can also operate in manual mode (AUTO = 0). Manual mode is used by systems that do not require an indicator of the lock condition for proper operation. Such systems typically operate well below the maximum system frequency (f_{sys}) and require fast start-up. The following conditions apply when in manual mode:

- ACQ is a writable control bit that controls the mode of the filter. Before turning on the PLL in manual mode, the ACQ bit should be asserted to configure the filter in acquisition mode.
- After turning on the PLL by setting the PLLON bit software must wait a given time (t_{acq}) before entering tracking mode (ACQ = 0).
- After entering tracking mode software must wait a given time (t_{al}) before selecting the PLLCLK as the source for system and core clocks (PLLSEL = 1).

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

- External reset
- Clock monitor reset
- COP reset
- Self-clock mode interrupt
- Real-time interrupt (RTI)

If the MCU gets an external 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 reset vector. Wait mode is exited 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 (see Section 9.4.4, "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 9-11 summarizes the outcome of a clock loss while in wait mode.

Table 10-10. CANRIER Register Field Descriptions (continued)

Field	Description
1 OVRIE	**Overrun Interrupt Enable** 0 No interrupt request is generated from this event. 1 An overrun event causes an error interrupt request.
0 RXFIE	**Receiver Full Interrupt Enable** 0 No interrupt request is generated from this event. 1 A receive buffer full (successful message reception) event causes a receiver interrupt request.

1. WUPIE and WUPE (see Section 10.3.2.1, "MSCAN Control Register 0 (CANCTL0)") must both be enabled if the recovery mechanism from stop or wait is required.
2. Bus-off state is defined by the CAN standard (see Bosch CAN 2.0A/B protocol specification: for only transmitters. Because the only possible state change for the transmitter from bus-off to TxOK also forces the receiver to skip its current state to RxOK, the coding of the RXSTAT[1:0] flags define an additional bus-off state for the receiver (see Section 10.3.2.5, "MSCAN Receiver Flag Register (CANRFLG)").

10.3.2.7 MSCAN Transmitter Flag Register (CANTFLG)

The transmit buffer empty flags each have an associated interrupt enable bit in the CANTIER register.

Module Base + 0x0006

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	TXE2	TXE1	TXE0
W								
Reset:	0	0	0	0	0	1	1	1

= Unimplemented

Figure 10-10. MSCAN Transmitter Flag Register (CANTFLG)

NOTE

The CANTFLG register is held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK = 1). This register is writable when not in initialization mode (INITRQ = 0 and INITAK = 0).

Read: Anytime
Write: Anytime for TXEx flags when not in initialization mode; write of 1 clears flag, write of 0 is ignored

The MSCAN then schedules the message for transmission and signals the successful transmission of the buffer by setting the associated TXE flag. A transmit interrupt (see Section 10.4.7.2, "Transmit Interrupt") is generated[1] when TXEx is set and can be used to drive the application software to re-load the buffer.

If more than one buffer is scheduled for transmission when the CAN bus becomes available for arbitration, the MSCAN uses the local priority setting of the three buffers to determine the prioritization. For this purpose, every transmit buffer has an 8-bit local priority field (PRIO). The application software programs this field when the message is set up. The local priority reflects the priority of this particular message relative to the set of messages being transmitted from this node. The lowest binary value of the PRIO field is defined to be the highest priority. The internal scheduling process takes place whenever the MSCAN arbitrates for the CAN bus. This is also the case after the occurrence of a transmission error.

When a high priority message is scheduled by the application software, it may become necessary to abort a lower priority message in one of the three transmit buffers. Because messages that are already in transmission cannot be aborted, the user must request the abort by setting the corresponding abort request bit (ABTRQ) (see Section 10.3.2.9, "MSCAN Transmitter Message Abort Request Register (CANTARQ)".) The MSCAN then grants the request, if possible, by:

1. Setting the corresponding abort acknowledge flag (ABTAK) in the CANTAAK register.
2. Setting the associated TXE flag to release the buffer.
3. Generating a transmit interrupt. The transmit interrupt handler software can determine from the setting of the ABTAK flag whether the message was aborted (ABTAK = 1) or sent (ABTAK = 0).

10.4.2.3 Receive Structures

The received messages are stored in a five stage input FIFO. The five message buffers are alternately mapped into a single memory area (see Figure 10-38). The background receive buffer (RxBG) is exclusively associated with the MSCAN, but the foreground receive buffer (RxFG) is addressable by the CPU (see Figure 10-38). This scheme simplifies the handler software because only one address area is applicable for the receive process.

All receive buffers have a size of 15 bytes to store the CAN control bits, the identifier (standard or extended), the data contents, and a time stamp, if enabled (see Section 10.3.3, "Programmer's Model of Message Storage").

The receiver full flag (RXF) (see Section 10.3.2.5, "MSCAN Receiver Flag Register (CANRFLG)") signals the status of the foreground receive buffer. When the buffer contains a correctly received message with a matching identifier, this flag is set.

On reception, each message is checked to see whether it passes the filter (see Section 10.4.3, "Identifier Acceptance Filter") and simultaneously is written into the active RxBG. After successful reception of a valid message, the MSCAN shifts the content of RxBG into the receiver FIFO[2], sets the RXF flag, and generates a receive interrupt (see Section 10.4.7.3, "Receive Interrupt") to the CPU[3]. The user's receive handler must read the received message from the RxFG and then reset the RXF flag to acknowledge the interrupt and to release the foreground buffer. A new message, which can follow immediately after the IFS field of the CAN frame, is received into the next available RxBG. If the MSCAN receives an invalid

1. The transmit interrupt occurs only if not masked. A polling scheme can be applied on TXEx also.
2. Only if the RXF flag is not set.
3. The receive interrupt occurs only if not masked. A polling scheme can be applied on RXF also.

13.4.3 Transmitter

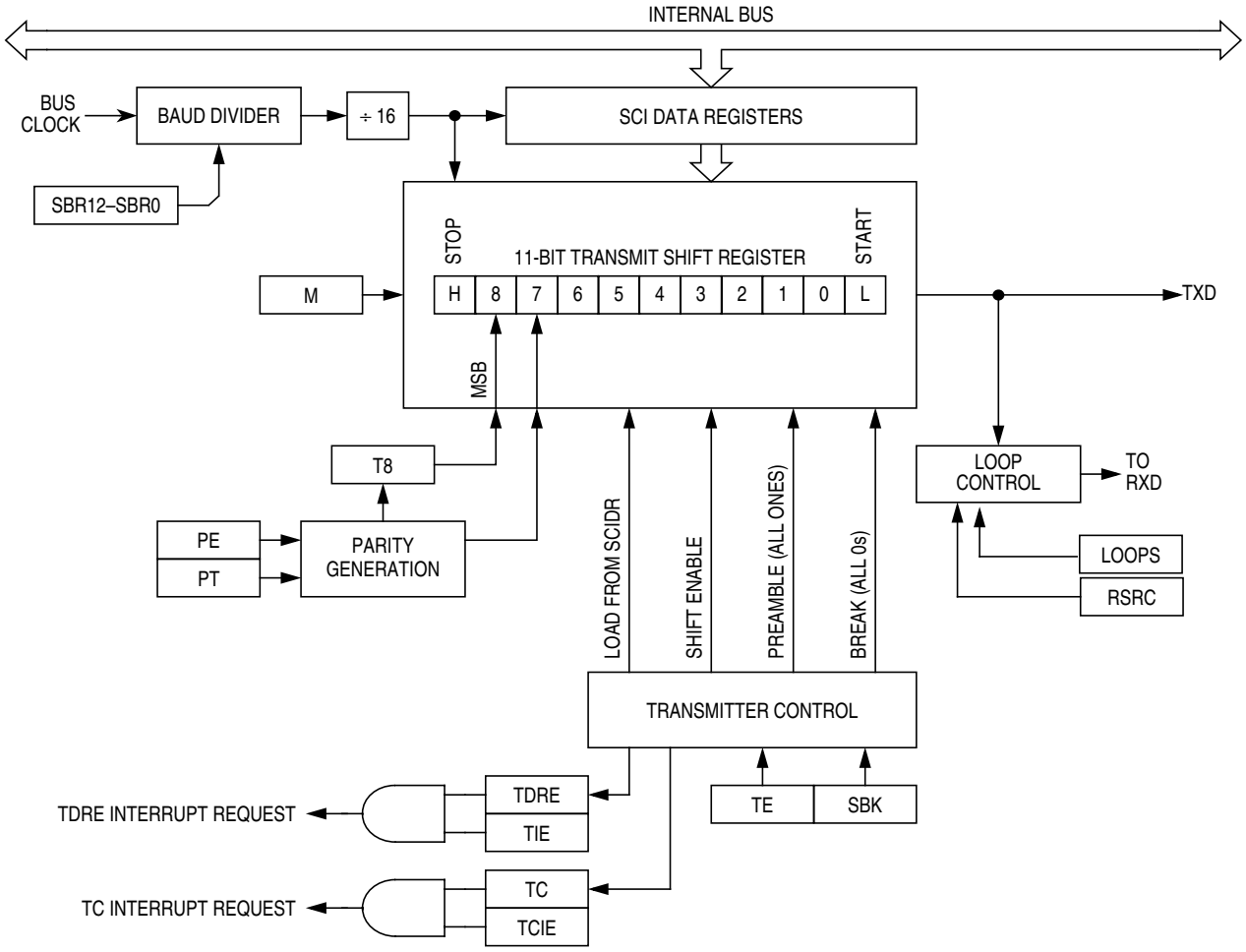


Figure 13-11. Transmitter Block Diagram

13.4.3.1 Transmitter Character Length

The SCI transmitter can accommodate either 8-bit or 9-bit data characters. The state of the M bit in SCI control register 1 (SCICR1) determines the length of data characters. When transmitting 9-bit data, bit T8 in SCI data register high (SCIDRH) is the ninth bit (bit 8).

13.4.3.2 Character Transmission

To transmit data, the MCU writes the data bits to the SCI data registers (SCIDRH/SCIDRL), which in turn are transferred to the transmitter shift register. The transmit shift register then shifts a frame out through the **Tx output** signal, after it has prefaced them with a start bit and appended them with a stop bit. The SCI data registers (SCIDRH and SCIDRL) are the write-only buffers between the internal data bus and the transmit shift register.

The SCI also sets a flag, the transmit data register empty flag (TDRE), every time it transfers data from the buffer (SCIDRH/L) to the transmitter shift register.The transmit driver routine may respond to this flag by

Table 17-4. FSEC Field Descriptions

Field	Description
7–6 KEYEN[1:0]	**Backdoor Key Security Enable Bits** — The KEYEN[1:0] bits define the enabling of the backdoor key access to the Flash module as shown in Table 17-5.
5–2 NV[5:2]	**Nonvolatile Flag Bits** — The NV[5:2] bits are available to the user as nonvolatile flags.
1–0 SEC[1:0]	**Flash Security Bits** — The SEC[1:0] bits define the security state of the MCU as shown in Table 17-6. If the Flash module is unsecured using backdoor key access, the SEC[1:0] bits are forced to 1:0.

Table 17-5. Flash KEYEN States

KEYEN[1:0]	Status of Backdoor Key Access
00	DISABLED
01(1)	DISABLED
10	ENABLED
11	DISABLED

1. Preferred KEYEN state to disable Backdoor Key Access.

Table 17-6. Flash Security States

SEC[1:0]	Status of Security
00	Secured
01(1)	Secured
10	Unsecured
11	Secured

1. Preferred SEC state to set MCU to secured state.

The security function in the Flash module is described in Section 17.4.3, “Flash Module Security”.

17.3.2.3 RESERVED1

This register is reserved for factory testing and is not accessible to the user.

Module Base + 0x0002

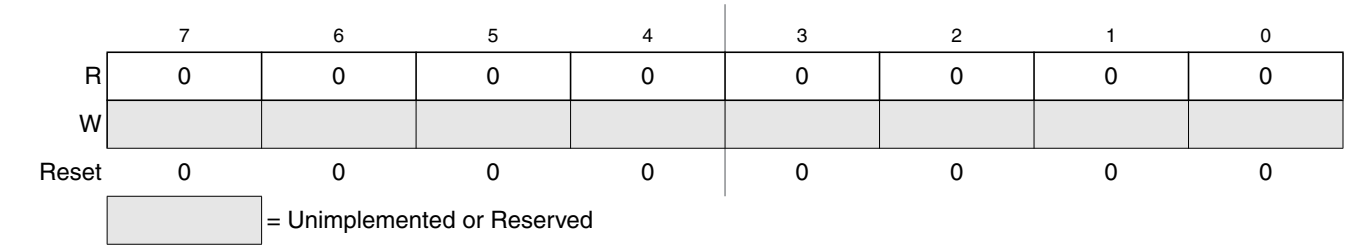


Figure 17-6. RESERVED1

All bits read 0 and are not writable.

18.3.2 Register Descriptions

The Flash module contains a set of 16 control and status registers located between module base + 0x0000 and 0x000F. A summary of the Flash module registers is given in Figure 18-3. Detailed descriptions of each register bit are provided.

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
0x0000 FCLKDIV	R	FDIVLD	PRDIV8	FDIV5	FDIV4	FDIV3	FDIV2	FDIV1	FDIV0
	W								
0x0001 FSEC	R	KEYEN1	KEYEN0	NV5	NV4	NV3	NV2	SEC1	SEC0
	W								
0x0002 RESERVED1 (1)	R	0	0	0	0	0	0	0	0
	W								
0x0003 FCNFG	R	CBEIE	CCIE	KEYACC	0	0	0	0	0
	W								
0x0004 FPROT	R	FPOPEN	NV6	FPHDIS	FPHS1	FPHS0	FPLDIS	FPLS1	FPLS0
	W								
0x0005 FSTAT	R	CBEIF	CCIF	PVIOL	ACCERR	0	BLANK	FAIL	DONE
	W								
0x0006 FCMD	R	0	CMDB6	CMDB5	0	0	CMDB2	0	CMDB0
	W								
0x0007 RESERVED2[1]	R	0	0	0	0	0	0	0	0
	W								
0x0008 FADDRHI[1]	R	0	0	FABHI					
	W								
0x0009 FADDRLO[1]	R	FABLO							
	W								
0x000A FDATAHI[1]	R	FDHI							
	W								
0x000B FDATALO[1]	R	FDLO							
	W								
0x000C RESERVED3[1]	R	0	0	0	0	0	0	0	0
	W								
0x000D RESERVED4[1]	R	0	0	0	0	0	0	0	0
	W								
0x000E RESERVED5[1]	R	0	0	0	0	0	0	0	0
	W								
0x000F RESERVED6[1]	R	0	0	0	0	0	0	0	0
	W								

= Unimplemented or Reserved

Figure 18-3. Flash Register Summary

1. Intended for factory test purposes only.

Table 18-9. Flash Protection Function

FPOPEN	FPHDIS	FPHS[1]	FPHS[0]	FPLDIS	FPLS[1]	FPLS[0]	Function(1)
1	1	x	x	1	x	x	No protection
1	1	x	x	0	x	x	Protect low range
1	0	x	x	1	x	x	Protect high range
1	0	x	x	0	x	x	Protect high and low ranges
0	1	x	x	1	x	x	Full Flash array protected
0	0	x	x	1	x	x	Unprotected high range
0	1	x	x	0	x	x	Unprotected low range
0	0	x	x	0	x	x	Unprotected high and low ranges

1. For range sizes refer to Table 18-10 and or Table 18-11.

Table 18-10. Flash Protection Higher Address Range

FPHS[1:0]	Address Range	Range Size
00	0xF800–0xFFFF	2 Kbytes
01	0xF000–0xFFFF	4 Kbytes
10	0xE000–0xFFFF	8 Kbytes
11	0xC000–0xFFFF	16 Kbytes

Table 18-11. Flash Protection Lower Address Range

FPLS[1:0]	Address Range	Range Size
00	0x4000–0x41FF	512 bytes
01	0x4000–0x43FF	1 Kbyte
10	0x4000–0x47FF	2 Kbytes
11	0x4000–0x4FFF	4 Kbytes

Figure 18-9 illustrates all possible protection scenarios. Although the protection scheme is loaded from the Flash array after reset, it is allowed to change in normal modes. This protection scheme can be used by applications requiring re-programming in single chip mode while providing as much protection as possible if no re-programming is required.

Module Base + 0x000F

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

= Unimplemented or Reserved

Figure 18-20. RESERVED6

All bits read 0 and are not writable.

18.4 Functional Description

18.4.1 Flash Command Operations

Write operations are used for the program, erase, and erase verify algorithms described in this section. The program and erase algorithms are controlled by a state machine whose timebase FCLK is derived from the oscillator clock via a programmable divider. The FCMD register as well as the associated FADDR and FDATA registers operate as a buffer and a register (2-stage FIFO) so that a new command along with the necessary data and address can be stored to the buffer while the previous command is still in progress. This pipelined operation allows a time optimization when programming more than one word on a specific row, as the high voltage generation can be kept active in between two programming commands. The pipelined operation also allows a simplification of command launching. Buffer empty as well as command completion are signalled by flags in the FSTAT register with corresponding interrupts generated, if enabled.

The next sections describe:

- How to write the FCLKDIV register
- Command write sequence used to program, erase or erase verify the Flash array
- Valid Flash commands
- Errors resulting from illegal Flash operations

18.4.1.1 Writing the FCLKDIV Register

Prior to issuing any Flash command after a reset, it is first necessary to write the FCLKDIV register to divide the oscillator clock down to within the 150-kHz to 200-kHz range. Since the program and erase timings are also a function of the bus clock, the FCLKDIV determination must take this information into account.

If we define:

- FCLK as the clock of the Flash timing control block
- Tbus as the period of the bus clock
- INT(x) as taking the integer part of x (e.g., INT(4.323) = 4),

FPHDIS is cleared. The FPROT register is loaded from Flash address 0xFF0D during the reset sequence, indicated by F in Figure 19-10.

To change the Flash protection that will be loaded on reset, the upper sector of the Flash array must be unprotected, then the Flash protection byte located at Flash address 0xFF0D must be written to.

A protected Flash sector is disabled by FPHDIS and FPLDIS while the size of the protected sector is defined by FPHS[1:0] and FPLS[1:0] in the FPROT register.

Trying to alter any of the protected areas will result in a protect violation error and the PVIOL flag will be set in the FSTAT register (see Section 19.3.2.6). A mass erase of the whole Flash array is only possible when protection is fully disabled by setting the FPOPEN, FPLDIS, and FPHDIS bits. An attempt to mass erase a Flash array while protection is enabled will set the PVIOL flag in the FSTAT register.

Table 19-9. FPROT Field Descriptions

Field	Description
7 FPOPEN	**Protection Function for Program or Erase** — It is possible using the FPOPEN bit to either select address ranges to be protected using FPHDIS, FPLDIS, FPHS[1:0] and FPLS[1:0] or to select the same ranges to be unprotected. When FPOPEN is set, FPxDIS enables the ranges to be protected, whereby clearing FPxDIS enables protection for the range specified by the corresponding FPxS[1:0] bits. When FPOPEN is cleared, FPxDIS defines unprotected ranges as specified by the corresponding FPxS[1:0] bits. In this case, setting FPxDIS enables protection. Thus the effective polarity of the FPxDIS bits is swapped by the FPOPEN bit as shown in Table 19-10. This function allows the main part of the Flash array to be protected while a small range can remain unprotected for EEPROM emulation. 0 The FPHDIS and FPLDIS bits define Flash address ranges to be unprotected 1 The FPHDIS and FPLDIS bits define Flash address ranges to be protected
6 NV6	**Nonvolatile Flag Bit** — The NV6 bit should remain in the erased state for future enhancements.
5 FPHDIS	**Flash Protection Higher Address Range Disable** — The FPHDIS bit determines whether there is a protected/unprotected area in the higher space of the Flash address map. 0 Protection/unprotection enabled 1 Protection/unprotection disabled
4–3 FPHS[1:0]	**Flash Protection Higher Address Size** — The FPHS[1:0] bits determine the size of the protected/unprotected sector as shown in Table 19-11. The FPHS[1:0] bits can only be written to while the FPHDIS bit is set.
2 FPLDIS	**Flash Protection Lower Address Range Disable** — The FPLDIS bit determines whether there is a protected/unprotected sector in the lower space of the Flash address map. 0 Protection/unprotection enabled 1 Protection/unprotection disabled
1–0 FPLS[1:0]	**Flash Protection Lower Address Size** — The FPLS[1:0] bits determine the size of the protected/unprotected sector as shown in Table 19-12. The FPLS[1:0] bits can only be written to while the FPLDIS bit is set.