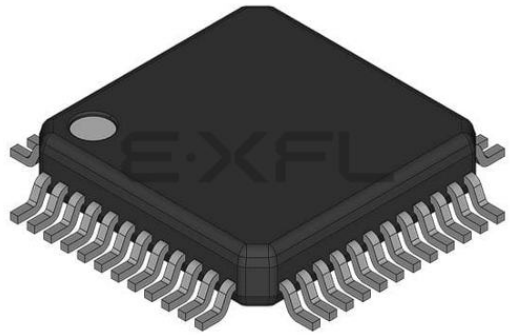

Welcome to E-XFL.COM

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	CANbus, EBI/EMI, SCI, SPI
Peripherals	POR, PWM, WDT
Number of I/O	31
Program Memory Size	96KB (96K 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 ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	48-LQFP
Supplier Device Package	48-LQFP (7x7)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mc9s12c96mfae

Chapter 1
MC9S12C and MC9S12GC Device Overview (MC9S12C128)

1.1 Introduction

The MC9S12C-Family / MC9S12GC-Family are 48/52/80 pin Flash-based MCU families, which deliver the power and flexibility of the 16-bit core to a whole new range of cost and space sensitive, general purpose industrial and automotive network applications. All MC9S12C-Family / MC9S12GC-Family members feature standard on-chip peripherals including a 16-bit central processing unit (CPU12), up to 128K bytes of Flash EEPROM, up to 4K bytes of RAM, an asynchronous serial communications interface (SCI), a serial peripheral interface (SPI), an 8-channel 16-bit timer module (TIM), a 6-channel 8-bit pulse width modulator (PWM), an 8-channel, 10-bit analog-to-digital converter (ADC).

The MC9S12C128-Family members also feature a CAN 2.0 A, B software compatible module (MSCAN12).

All MC9S12C-Family / MC9S12GC-Family devices feature full 16-bit data paths throughout. The inclusion of a PLL circuit allows power consumption and performance to be adjusted to suit operational requirements. In addition to the I/O ports available in each module, up to 10 dedicated I/O port bits are available with wake-up capability from stop or wait mode. The devices are available in 48-, 52-, and 80-pin QFP packages, with the 80-pin version pin compatible to the HCS12 A, B, and D Family derivatives.

1.1.1 Features

- 16-bit HCS12 core:
  - HCS12 CPU
    - Upward compatible with M68HC11 instruction set
    - Interrupt stacking and programmer's model identical to M68HC11
    - Instruction queue
    - Enhanced indexed addressing
  - MMC (memory map and interface)
  - INT (interrupt control)
  - BDM (background debug mode)
  - DBG12 (enhanced debug12 module, including breakpoints and change-of-flow trace buffer)
  - MEBI (multiplexed expansion bus interface) available only in 80-pin package version
- Wake-up interrupt inputs:
  - Up to 12 port bits available for wake up interrupt function with digital filtering

1.2.2 Detailed Register Map

The detailed register map of the MC9S12C128 is listed in address order below.

0x0000–0x000F MEBI Map 1 of 3 (HCS12 Multiplexed External Bus Interface)

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0000	PORTA	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
0x0001	PORTB	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
0x0002	DDRA	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
0x0003	DDRB	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
0x0004	Reserved	Read:	0	0	0	0	0	0	0	0
		Write:								
0x0005	Reserved	Read:	0	0	0	0	0	0	0	0
		Write:								
0x0006	Reserved	Read:	0	0	0	0	0	0	0	0
		Write:								
0x0007	Reserved	Read:	0	0	0	0	0	0	0	0
		Write:								
0x0008	PORTE	Read:	Bit 7	6	5	4	3	2	Bit 1	Bit 0
		Write:								
0x0009	DDRE	Read:	Bit 7	6	5	4	3	Bit 2	0	0
		Write:								
0x000A	PEAR	Read:	NOACCE	0	PIPOE	NECLK	LSTRE	RDWE	0	0
		Write:								
0x000B	MODE	Read:	MODC	MODB	MODA	0	IVIS	0	EMK	EME
		Write:								
0x000C	PUCR	Read:	PUPKE	0	0	PUPEE	0	0	PUPBE	PUPAE
		Write:								
0x000D	RDRIV	Read:	RDPK	0	0	RDPE	0	0	RDPB	RDPA
		Write:								
0x000E	EBICTL	Read:	0	0	0	0	0	0	0	ESTR
		Write:								
0x000F	Reserved	Read:	0	0	0	0	0	0	0	0
		Write:								

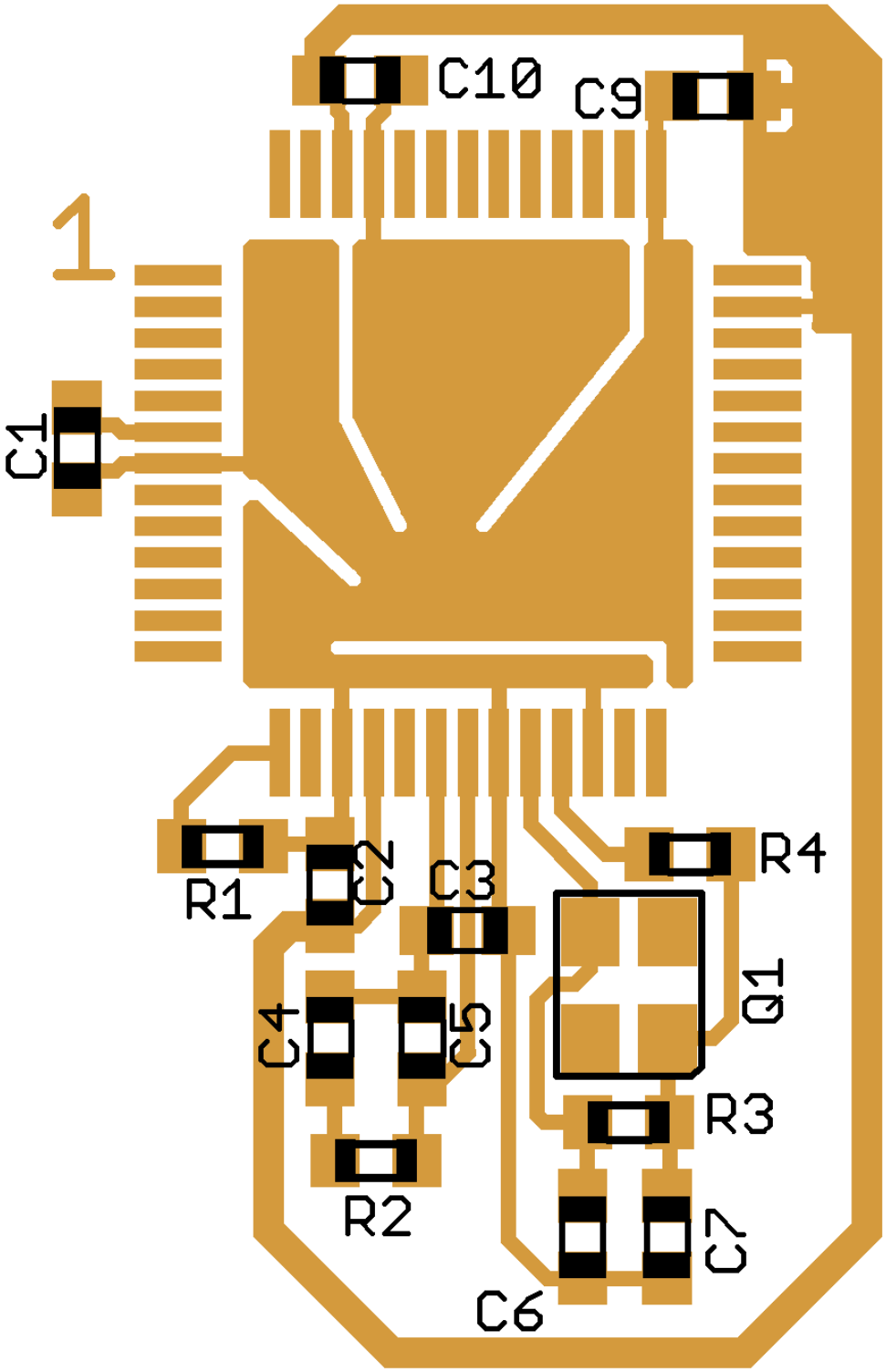

Figure 1-19. Recommended PCB Layout for 52 LQFP Pierce Oscillator

2.4.1.4 Reduced Drive Register

If the port is used as an output the register allows the configuration of the drive strength.

2.4.1.5 Pull Device Enable Register

This register turns on a pull-up or pull-down device. It becomes only active if the pin is used as an input or as a wired-or output.

2.4.1.6 Polarity Select Register

This register selects either a pull-up or pull-down device if enabled. It becomes only active if the pin is used as an input. A pull-up device can be activated if the pin is used as a wired-OR output.

2.4.2 Port Descriptions

2.4.2.1 Port T

This port is associated with the Standard Capture Timer. PWM output channels can be rerouted from port P to port pins T. In all modes, port T pins can be used for either general-purpose I/O, Standard Capture Timer I/O or as PWM channels module, if so configured by MODRR.

During reset, port T pins are configured as high-impedance inputs.

2.4.2.2 Port S

This port is associated with the serial SCI module. Port S pins PS[3:0] can be used either for general-purpose I/O, or with the SCI subsystem.

During reset, port S pins are configured as inputs with pull-up.

2.4.2.3 Port M

This port is associated with the MSCAN and SPI module. Port M pins PM[5:0] can be used either for general-purpose I/O, with the MSCAN or SPI subsystems.

During reset, port M pins are configured as inputs with pull-up.

2.4.2.4 Port AD

This port is associated with the ATD module. Port AD pins can be used either for general-purpose I/O, or for the ATD subsystem. There are 2 data port registers associated with the Port AD: PTAD[7:0], located in the PIM and PORTAD[7:0] located in the ATD.

To use PTAD[n] as a standard input port, the corresponding DDRD[n] must be cleared. To use PTAD[n] as a standard output port, the corresponding DDRD[n] must be set

NOTE: To use PORTAD[n], located in the ATD as an input port register, DDRD[n] must be cleared and ATDDIEN[n] must be set. *Please refer to ATD Block Guide for details*.

Table 4-10. RDRIV Field Descriptions

Field	Description
7 RDRK	**Reduced Drive of Port K** 0 All port K output pins have full drive enabled. 1 All port K output pins have reduced drive enabled.
4 RDPE	**Reduced Drive of Port E** 0 All port E output pins have full drive enabled. 1 All port E output pins have reduced drive enabled.
1 RDPB	**Reduced Drive of Port B** 0 All port B output pins have full drive enabled. 1 All port B output pins have reduced drive enabled.
0 RDPA	**Reduced Drive of Ports A** 0 All port A output pins have full drive enabled. 1 All port A output pins have reduced drive enabled.

4.3.2.12 External Bus Interface Control Register (EBICTL)

Module Base + 0x000E
Starting address location affected by INITRG register setting.

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	ESTR
W								
Reset:								
Peripheral	0	0	0	0	0	0	0	0
All other modes	0	0	0	0	0	0	0	1

= Unimplemented or Reserved

Figure 4-16. External Bus Interface Control Register (EBICTL)

Read: Anytime (provided this register is in the map)

Write: Refer to individual bit descriptions below

The EBICTL register is used to control miscellaneous functions (i.e., stretching of external E clock).

This register is not in the on-chip memory map in expanded and special peripheral modes. Therefore, these accesses will be echoed externally.

Table 4-11. EBICTL Field Descriptions

Field	Description
0 ESTR	**E Clock Stretches** — This control bit determines whether the E clock behaves as a simple free-running clock or as a bus control signal that is active only for external bus cycles. Normal and Emulation: write once Special: write anytime 0 E never stretches (always free running). 1 E stretches high during stretched external accesses and remains low during non-visible internal accesses. This bit has no effect in single-chip modes.

Table 6-2. BDMSTS Field Descriptions

Field	Description
7 ENBDM	**Enable BDM** — This bit controls whether the BDM is enabled or disabled. When enabled, BDM can be made active to allow firmware commands to be executed. When disabled, BDM cannot be made active but BDM hardware commands are allowed. 0 BDM disabled 1 BDM enabled **Note:** ENBDM is set by the firmware immediately out of reset in special single-chip mode. In secure mode, this bit will not be set by the firmware until after the EEPROM and FLASH erase verify tests are complete.
6 BDMACT	**BDM Active Status** — This bit becomes set upon entering BDM. The standard BDM firmware lookup table is then enabled and put into the memory map. BDMACT is cleared by a carefully timed store instruction in the standard BDM firmware as part of the exit sequence to return to user code and remove the BDM memory from the map. 0 BDM not active 1 BDM active
5 ENTAG	**Tagging Enable** — This bit indicates whether instruction tagging in enabled or disabled. It is set when the TAGGO command is executed and cleared when BDM is entered. The serial system is disabled and the tag function enabled 16 cycles after this bit is written. BDM cannot process serial commands while tagging is active. 0 Tagging not enabled or BDM active 1 Tagging enabled
4 SDV	**Shift Data Valid** — This bit is set and cleared by the BDM hardware. It is set after data has been transmitted as part of a firmware read command or after data has been received as part of a firmware write command. It is cleared when the next BDM command has been received or BDM is exited. SDV is used by the standard BDM firmware to control program flow execution. 0 Data phase of command not complete 1 Data phase of command is complete
3 TRACE	**TRACE1 BDM Firmware Command is Being Executed** — This bit gets set when a BDM TRACE1 firmware command is first recognized. It will stay set as long as continuous back-to-back TRACE1 commands are executed. This bit will get cleared when the next command that is not a TRACE1 command is recognized. 0 TRACE1 command is not being executed 1 TRACE1 command is being executed

firmware. The standard BDM firmware watches for serial commands and executes them as they are received.

The firmware commands are shown in Table 6-6.

Table 6-6. Firmware Commands

Command[1]	Opcode (hex)	Data	Description
READ_NEXT	62	16-bit data out	Increment X by 2 (X = X + 2), then read word X points to.
READ_PC	63	16-bit data out	Read program counter.
READ_D	64	16-bit data out	Read D accumulator.
READ_X	65	16-bit data out	Read X index register.
READ_Y	66	16-bit data out	Read Y index register.
READ_SP	67	16-bit data out	Read stack pointer.
WRITE_NEXT	42	16-bit data in	Increment X by 2 (X = X + 2), then write word to location pointed to by X.
WRITE_PC	43	16-bit data in	Write program counter.
WRITE_D	44	16-bit data in	Write D accumulator.
WRITE_X	45	16-bit data in	Write X index register.
WRITE_Y	46	16-bit data in	Write Y index register.
WRITE_SP	47	16-bit data in	Write stack pointer.
GO	08	None	Go to user program. If enabled, ACK will occur when leaving active background mode.
GO_UNTIL[2]	0C	None	Go to user program. If enabled, ACK will occur upon returning to active background mode.
TRACE1	10	None	Execute one user instruction then return to active BDM. If enabled, ACK will occur upon returning to active background mode.
TAGGO	18	None	Enable tagging and go to user program. There is no ACK pulse related to this command.

1. If enabled, ACK will occur when data is ready for transmission for all BDM READ commands and will occur after the write is complete for all BDM WRITE commands.
2. Both WAIT (with clocks to the S12 CPU core disabled) and STOP disable the ACK function. The GO_UNTIL command will not get an Acknowledge if one of these two CPU instructions occurs before the “UNTIL” instruction. This can be a problem for any instruction that uses ACK, but GO_UNTIL is a lot more difficult for the development tool to time-out.

6.4.5 BDM Command Structure

Hardware and firmware BDM commands start with an 8-bit opcode followed by a 16-bit address and/or a 16-bit data word depending on the command. All the read commands return 16 bits of data despite the byte or word implication in the command name.

NOTE

8-bit reads return 16-bits of data, of which, only one byte will contain valid data. If reading an even address, the valid data will appear in the MSB. If reading an odd address, the valid data will appear in the LSB.

NOTE

BDM should not be entered from a breakpoint unless the ENABLE bit is set in the BDM. Even if the ENABLE bit in the BDM is cleared, the CPU actually executes the BDM firmware code. It checks the ENABLE and returns if ENABLE is not set. If the BDM is not serviced by the monitor then the breakpoint would be re-asserted when the BDM returns to normal CPU flow.

There is no hardware to enforce restriction of breakpoint operation if the BDM is not enabled.

When program control returns from a tagged breakpoint through an RTI or a BDM GO command, it will return to the instruction whose tag generated the breakpoint. Unless breakpoints are disabled or modified in the service routine or active BDM session, the instruction will be tagged again and the breakpoint will be repeated. In the case of BDM breakpoints, this situation can also be avoided by executing a TRACE1 command before the GO to increment the program flow past the tagged instruction.

7.4.1.4 Using Comparator C in BKP Mode

The original BKP_ST12_A module supports two breakpoints. The DBG_ST12_A module can be used in BKP mode and allow a third breakpoint using comparator C. Four additional bits, BKCEN, TAGC, RWCEN, and RWC in DBGC2 in conjunction with additional comparator C address registers, DBGCCX, DBGCCH, and DBGCCL allow the user to set up a third breakpoint. Using PAGSEL in DBGCCX for expanded memory will work differently than the way paged memory is done using comparator A and B in BKP mode. See Section 7.3.2.5, “Debug Comparator C Extended Register (DBGCCX),” for more information on using comparator C.

7.4.2 DBG Operating in DBG Mode

Enabling the DBG module in DBG mode, allows the arming, triggering, and storing of data in the trace buffer and can be used to cause CPU breakpoints. The DBG module is made up of three main blocks, the comparators, trace buffer control logic, and the trace buffer.

NOTE

In general, there is a latency between the triggering event appearing on the bus and being detected by the DBG circuitry. In general, tagged triggers will be more predictable than forced triggers.

7.4.2.1 Comparators

The DBG contains three comparators, A, B, and C. Comparator A compares the core address bus with the address stored in DBGCAH and DBGCAL. Comparator B compares the core address bus with the address stored in DBGCBH and DBGCBL except in full mode, where it compares the data buses to the data stored in DBGCBH and DBGCBL. Comparator C can be used as a breakpoint generator or as the address comparison unit in the loop1 mode. Matches on comparator A, B, and C are signaled to the trace buffer

the trigger is at the address of a change-of-flow address the trigger event will not be stored in the trace buffer.

7.4.2.9 Reading Data from Trace Buffer

The data stored in the trace buffer can be read using either the background debug module (BDM) module or the CPU provided the DBG module is enabled and not armed. The trace buffer data is read out first-in first-out. By reading CNT in DBGCNT the number of valid words can be determined. CNT will not decrement as data is read from DBGTBH:DBGTBL. The trace buffer data is read by reading DBGTBH:DBGTBL with a 16-bit read. Each time DBGTBH:DBGTBL is read, a pointer in the DBG will be incremented to allow reading of the next word.

Reading the trace buffer while the DBG module is armed will return invalid data and no shifting of the RAM pointer will occur.

NOTE

The trace buffer should be read with the DBG module enabled and in the same capture mode that the data was recorded. The contents of the trace buffer counter register (DBGCNT) are resolved differently in detail mode verses the other modes and may lead to incorrect interpretation of the trace buffer data.

7.4.3 Breakpoints

There are two ways of getting a breakpoint in DBG mode. One is based on the trigger condition of the trigger mode using comparator A and/or B, and the other is using comparator C. External breakpoints generated using the $\overline{\text{TAGHI}}$ and $\overline{\text{TAGLO}}$ external pins are disabled in DBG mode.

7.4.3.1 Breakpoint Based on Comparator A and B

A breakpoint request to the CPU can be enabled by setting DBGBRK in DBGC1. The value of BEGIN in DBGC1 determines when the breakpoint request to the CPU will occur. When BEGIN in DBGC1 is set, begin-trigger is selected and the breakpoint request will not occur until the trace buffer is filled with 64 words. When BEGIN in DBGC1 is cleared, end-trigger is selected and the breakpoint request will occur immediately at the trigger cycle.

There are two types of breakpoint requests supported by the DBG module, tagged and forced. Tagged breakpoints are associated with opcode addresses and allow breaking just before a specific instruction executes. Forced breakpoints are not associated with opcode addresses and allow breaking at the next instruction boundary. The type of breakpoint based on comparators A and B is determined by TRGSEL in the DBGC1 register (TRGSEL = 1 for tagged breakpoint, TRGSEL = 0 for forced breakpoint). Table 7-26 illustrates the type of breakpoint that will occur based on the debug run.

9.3.2.6 CRG Clock Select Register (CLKSEL)

This register controls CRG clock selection. Refer to Figure 9-17 for details on the effect of each bit.

Module Base + 0x0005

	7	6	5	4	3	2	1	0
R W	PLLSEL	PSTP	SYSWAI	ROAWAI	PLLWAI	CWAI	RTIWAI	COPWAI
Reset	0	0	0	0	0	0	0	0

Figure 9-9. CRG Clock Select Register (CLKSEL)

Read: anytime

Write: refer to each bit for individual write conditions

Table 9-4. CLKSEL Field Descriptions

Field	Description
7 PLLSEL	**PLL Select Bit** — Write anytime. Writing a 1 when LOCK = 0 and AUTO = 1, or TRACK = 0 and AUTO = 0 has no effect. This prevents the selection of an unstable PLLCLK as SYSCLK. PLLSEL bit is cleared when the MCU enters self-clock mode, stop mode or wait mode with PLLWAI bit set. 0 System clocks are derived from OSCCLK (Bus Clock = OSCCLK / 2). 1 System clocks are derived from PLLCLK (Bus Clock = PLLCLK / 2).
6 PSTP	**Pseudo-Stop Bit** — Write: anytime — This bit controls the functionality of the oscillator during stop mode. 0 Oscillator is disabled in stop mode. 1 Oscillator continues to run in stop mode (pseudo-stop). The oscillator amplitude is reduced. Refer to oscillator block description for availability of a reduced oscillator amplitude. **Note:** Pseudo-stop allows for faster stop recovery and reduces the mechanical stress and aging of the resonator in case of frequent stop conditions at the expense of a slightly increased power consumption. **Note:** Lower oscillator amplitude exhibits lower power consumption but could have adverse effects during any electro-magnetic susceptibility (EMS) tests.
5 SYSWAI	**System Clocks Stop in Wait Mode Bit** — Write: anytime 0 In wait mode, the system clocks continue to run. 1 In wait mode, the system clocks stop. **Note:** RTI and COP are not affected by SYSWAI bit.
4 ROAWAI	**Reduced Oscillator Amplitude in Wait Mode Bit** — Write: anytime — Refer to oscillator block description chapter for availability of a reduced oscillator amplitude. If no such feature exists in the oscillator block then setting this bit to 1 will not have any effect on power consumption. 0 Normal oscillator amplitude in wait mode. 1 Reduced oscillator amplitude in wait mode. **Note:** Lower oscillator amplitude exhibits lower power consumption but could have adverse effects during any electro-magnetic susceptibility (EMS) tests.
3 PLLWAI	**PLL Stops in Wait Mode Bit** — Write: anytime — If PLLWAI is set, the CRGV4 will clear the PLLSEL bit before entering wait mode. The PLLON bit remains set during wait mode but the PLL is powered down. Upon exiting wait mode, the PLLSEL bit has to be set manually if PLL clock is required. While the PLLWAI bit is set the AUTO bit is set to 1 in order to allow the PLL to automatically lock on the selected target frequency after exiting wait mode. 0 PLL keeps running in wait mode. 1 PLL stops in wait mode.

Table 10-1. CANCTL0 Register Field Descriptions (continued)

Field	Description
1 SLPRQ[5]	**Sleep Mode Request** — This bit requests the MSCAN to enter sleep mode, which is an internal power saving mode (see Section 10.4.5.4, "MSCAN Sleep Mode"). The sleep mode request is serviced when the CAN bus is idle, i.e., the module is not receiving a message and all transmit buffers are empty. The module indicates entry to sleep mode by setting SLPAK = 1 (see Section 10.3.2.2, "MSCAN Control Register 1 (CANCTL1)"). SLPRQ cannot be set while the WUPIF flag is set (see Section 10.3.2.5, "MSCAN Receiver Flag Register (CANRFLG)"). Sleep mode will be active until SLPRQ is cleared by the CPU or, depending on the setting of WUPE, the MSCAN detects activity on the CAN bus and clears SLPRQ itself. 0 Running — The MSCAN functions normally 1 Sleep mode request — The MSCAN enters sleep mode when CAN bus idle
0 INITRQ[6],[7]	**Initialization Mode Request** — When this bit is set by the CPU, the MSCAN skips to initialization mode (see Section 10.4.5.5, "MSCAN Initialization Mode"). Any ongoing transmission or reception is aborted and synchronization to the CAN bus is lost. The module indicates entry to initialization mode by setting INITAK = 1 (Section 10.3.2.2, "MSCAN Control Register 1 (CANCTL1)"). The following registers enter their hard reset state and restore their default values: CANCTL0[8], CANRFLG[9], CANRIER[10], CANTFLG, CANTIER, CANTARQ, CANTAAK, and CANTBSEL. The registers CANCTL1, CANBTR0, CANBTR1, CANIDAC, CANIDAR0-7, and CANIDMR0-7 can only be written by the CPU when the MSCAN is in initialization mode (INITRQ = 1 and INITAK = 1). The values of the error counters are not affected by initialization mode. When this bit is cleared by the CPU, the MSCAN restarts and then tries to synchronize to the CAN bus. If the MSCAN is not in bus-off state, it synchronizes after 11 consecutive recessive bits on the CAN bus; if the MSCAN is in bus-off state, it continues to wait for 128 occurrences of 11 consecutive recessive bits. Writing to other bits in CANCTL0, CANRFLG, CANRIER, CANTFLG, or CANTIER must be done only after initialization mode is exited, which is INITRQ = 0 and INITAK = 0. 0 Normal operation 1 MSCAN in initialization mode

1. The MSCAN must be in normal mode for this bit to become set.
2. See the Bosch CAN 2.0A/B specification for a detailed definition of transmitter and receiver states.
3. In order to protect from accidentally violating the CAN protocol, the TXCAN pin is immediately forced to a recessive state when the CPU enters wait (CSWAI = 1) or stop mode (see Section 10.4.5.2, "Operation in Wait Mode" and Section 10.4.5.3, "Operation in Stop Mode").
4. The CPU has to make sure that the WUPE register and the WUPIE wake-up interrupt enable register (see Section 10.3.2.6, "MSCAN Receiver Interrupt Enable Register (CANRIER)) is enabled, if the recovery mechanism from stop or wait is required.
5. The CPU cannot clear SLPRQ before the MSCAN has entered sleep mode (SLPRQ = 1 and SLPAK = 1).
6. The CPU cannot clear INITRQ before the MSCAN has entered initialization mode (INITRQ = 1 and INITAK = 1).
7. In order to protect from accidentally violating the CAN protocol, the TXCAN pin is immediately forced to a recessive state when the initialization mode is requested by the CPU. Thus, the recommended procedure is to bring the MSCAN into sleep mode (SLPRQ = 1 and SLPAK = 1) before requesting initialization mode.
8. Not including WUPE, INITRQ, and SLPRQ.
9. TSTAT1 and TSTAT0 are not affected by initialization mode.
10. RSTAT1 and RSTAT0 are not affected by initialization mode.

10.3.2.2 MSCAN Control Register 1 (CANCTL1)

The CANCTL1 register provides various control bits and handshake status information of the MSCAN module as described below.

10.3.2.3 MSCAN Bus Timing Register 0 (CANBTR0)

The CANBTR0 register configures various CAN bus timing parameters of the MSCAN module.

Module Base + 0x0002

	7	6	5	4	3	2	1	0
R W	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0
Reset:	0	0	0	0	0	0	0	0

Figure 10-6. MSCAN Bus Timing Register 0 (CANBTR0)

Read: Anytime
Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1)

Table 10-3. CANBTR0 Register Field Descriptions

Field	Description
7:6 SJW[1:0]	**Synchronization Jump Width** — The synchronization jump width defines the maximum number of time quanta (Tq) clock cycles a bit can be shortened or lengthened to achieve resynchronization to data transitions on the CAN bus (see Table 10-4).
5:0 BRP[5:0]	**Baud Rate Prescaler** — These bits determine the time quanta (Tq) clock which is used to build up the bit timing (see Table 10-5).

Table 10-4. Synchronization Jump Width

SJW1	SJW0	Synchronization Jump Width
0	0	1 Tq clock cycle
0	1	2 Tq clock cycles
1	0	3 Tq clock cycles
1	1	4 Tq clock cycles

Table 10-5. Baud Rate Prescaler

BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	Prescaler value (P)
0	0	0	0	0	0	1
0	0	0	0	0	1	2
0	0	0	0	1	0	3
0	0	0	0	1	1	4
:	:	:	:	:	:	:
1	1	1	1	1	1	64

13.4.1 Data Format

The SCI uses the standard NRZ mark/space data format illustrated in Figure 13-10 below.

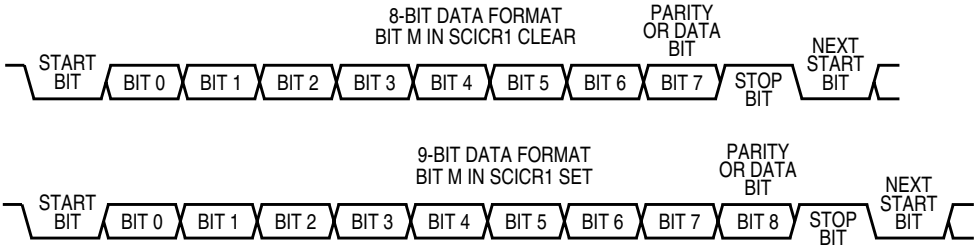

Figure 13-10. SCI Data Formats

Each data character is contained in a frame that includes a start bit, eight or nine data bits, and a stop bit. Clearing the M bit in SCI control register 1 configures the SCI for 8-bit data characters.A frame with eight data bits has a total of 10 bits. Setting the M bit configures the SCI for nine-bit data characters. A frame with nine data bits has a total of 11 bits

Table 13-8. Example of 8-Bit Data Formats

Start Bit	Data Bits	Address Bits	Parity Bits	Stop Bit
1	8	0	0	1
1	7	0	1	1
1	7	1[(1)]	0	1

1. The address bit identifies the frame as an address character. See Section 13.4.4.6, “Receiver Wakeup”.

When the SCI is configured for 9-bit data characters, the ninth data bit is the T8 bit in SCI data register high (SCIDRH). It remains unchanged after transmission and can be used repeatedly without rewriting it. A frame with nine data bits has a total of 11 bits.

Table 13-9. Example of 9-Bit Data Formats

Start Bit	Data Bits	Address Bits	Parity Bits	Stop Bit
1	9	0	0	1
1	8	0	1	1
1	8	1[(1)]	0	1

1. The address bit identifies the frame as an address character. See Section 13.4.4.6, “Receiver Wakeup”.

Table 13-11. Start Bit Verification

RT3, RT5, and RT7 Samples	Start Bit Verification	Noise Flag
100	Yes	1
101	No	0
110	No	0
111	No	0

If start bit verification is not successful, the RT clock is reset and a new search for a start bit begins.

To determine the value of a data bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. Table 13-12 summarizes the results of the data bit samples.

Table 13-12. Data Bit Recovery

RT8, RT9, and RT10 Samples	Data Bit Determination	Noise Flag
000	0	0
001	0	1
010	0	1
011	1	1
100	0	1
101	1	1
110	1	1
111	1	0

NOTE

The RT8, RT9, and RT10 samples do not affect start bit verification. If any or all of the RT8, RT9, and RT10 start bit samples are logic 1s following a successful start bit verification, the noise flag (NF) is set and the receiver assumes that the bit is a start bit (logic 0).

To verify a stop bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. Table 13-13 summarizes the results of the stop bit samples.

Table 13-13. Stop Bit Recovery

RT8, RT9, and RT10 Samples	Framing Error Flag	Noise Flag
000	1	0
001	1	1
010	1	1
011	0	1
100	1	1
101	0	1
110	0	1
111	0	0

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
0x000D TSCR2	R	TOI	0	0	0	TCRE	PR2	PR1	PR0
	W								
0x000E TFLG1	R	C7F	C6F	C5F	C4F	C3F	C2F	C1F	C0F
	W								
0x000F TFLG2	R	TOF	0	0	0	0	0	0	0
	W								
0x0010–0x001F TCxH–TCxL	R	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	W								
	R	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	W								
0x0020 PACTL	R	0	PAEN	PAMOD	PEDGE	CLK1	CLK0	PAOVI	PAI
	W								
0x0021 PAFLG	R	0	0	0	0	0	0	PAOVF	PAIF
	W								
0x0022 PACNTH	R	PACNT15	PACNT14	PACNT13	PACNT12	PACNT11	PACNT10	PACNT9	PACNT8
	W								
0x0023 PACNTL	R	PACNT7	PACNT6	PACNT5	PACNT4	PACNT3	PACNT2	PACNT1	PACNT0
	W								
0x0024–0x002F Reserved	R								
	W								

= Unimplemented or Reserved

Figure 15-5. TIM16B8CV1 Register Summary (continued)

15.3.2.1 Timer Input Capture/Output Compare Select (TIOS)

Module Base + 0x0000

	7	6	5	4	3	2	1	0
R W	IOS7	IOS6	IOS5	IOS4	IOS3	IOS2	IOS1	IOS0
Reset	0	0	0	0	0	0	0	0

Figure 15-6. Timer Input Capture/Output Compare Select (TIOS)

Read: Anytime

18.4.1.2 Command Write Sequence

The Flash command controller is used to supervise the command write sequence to execute program, erase, and erase verify algorithms.

Before starting a command write sequence, the ACCERR and PVIOL flags in the FSTAT register must be clear and the CBEIF flag should be tested to determine the state of the address, data, and command buffers. If the CBEIF flag is set, indicating the buffers are empty, a new command write sequence can be started. If the CBEIF flag is clear, indicating the buffers are not available, a new command write sequence will overwrite the contents of the address, data, and command buffers.

A command write sequence consists of three steps which must be strictly adhered to with writes to the Flash module not permitted between the steps. However, Flash register and array reads are allowed during a command write sequence. The basic command write sequence is as follows:

1. Write to a valid address in the Flash array memory.
2. Write a valid command to the FCMD register.
3. Clear the CBEIF flag in the FSTAT register by writing a 1 to CBEIF to launch the command.

The address written in step 1 will be stored in the FADDR registers and the data will be stored in the FDATA registers. When the CBEIF flag is cleared in step 3, the CCIF flag is cleared by the Flash command controller indicating that the command was successfully launched. For all command write sequences, the CBEIF flag will set after the CCIF flag is cleared indicating that the address, data, and command buffers are ready for a new command write sequence to begin. A buffered command will wait for the active operation to be completed before being launched. Once a command is launched, the completion of the command operation is indicated by the setting of the CCIF flag in the FSTAT register. The CCIF flag will set upon completion of all active and buffered commands.

18.4.1.3.3 Sector Erase Command

The sector erase operation will erase all addresses in a 512 byte sector of the Flash array using an embedded algorithm.

An example flow to execute the sector erase operation is shown in Figure 18-24. The sector erase command write sequence is as follows:

1. Write to a Flash array address to start the command write sequence for the sector erase command. The Flash address written determines the sector to be erased while MCU address bits [8:0] and the data written are ignored.
2. Write the sector erase command, 0x40, to the FCMD register.
3. Clear the CBEIF flag in the FSTAT register by writing a 1 to CBEIF to launch the sector erase command.

If a Flash sector to be erased is in a protected area of the Flash array, the PVIOL flag in the FSTAT register will set and the sector erase command will not launch. Once the sector erase command has successfully launched, the CCIF flag in the FSTAT register will set after the sector erase operation has completed unless a new command write sequence has been buffered.

Table 20-2. Flash Array Memory Map Summary

MCU Address Range	PPAGE	Protectable Low Range	Protectable High Range	Array Relative Address[1]
0x0000–0x3FFF[2]	Unpaged (0x3D)	N.A.	N.A.	0x14000–0x17FFF
0x4000–0x7FFF	Unpaged (0x3E)	0x4000–0x43FF 0x4000–0x47FF 0x4000–0x4FFF 0x4000–0x5FFF	N.A.	0x18000–0x1BFFF
0x8000–0xBFFF	0x38	N.A.	N.A.	0x00000–0x03FFF
	0x39	N.A.	N.A.	0x04000–0x07FFF
	0x3A	N.A.	N.A.	0x08000–0x0BFFF
	0x3B	N.A.	N.A.	0x0C000–0x0FFFF
	0x3C	N.A.	N.A.	0x10000–0x13FFF
	0x3D	N.A.	N.A.	0x14000–0x17FFF
	0x3E	0x8000–0x83FF 0x8000–0x87FF 0x8000–0x8FFF 0x8000–0x9FFF	N.A.	0x18000–0x1BFFF
	0x3F	N.A.	0xB800–0xBFFF 0xB000–0xBFFF 0xA000–0xBFFF 0x8000–0xBFFF	0x1C000–0x1FFFF
0xC000–0xFFFF	Unpaged (0x3F)	N.A.	0xF800–0xFFFF 0xF000–0xFFFF 0xE000–0xFFFF 0xC000–0xFFFF	0x1C000–0x1FFFF

1. Inside Flash block.
2. If allowed by MCU.

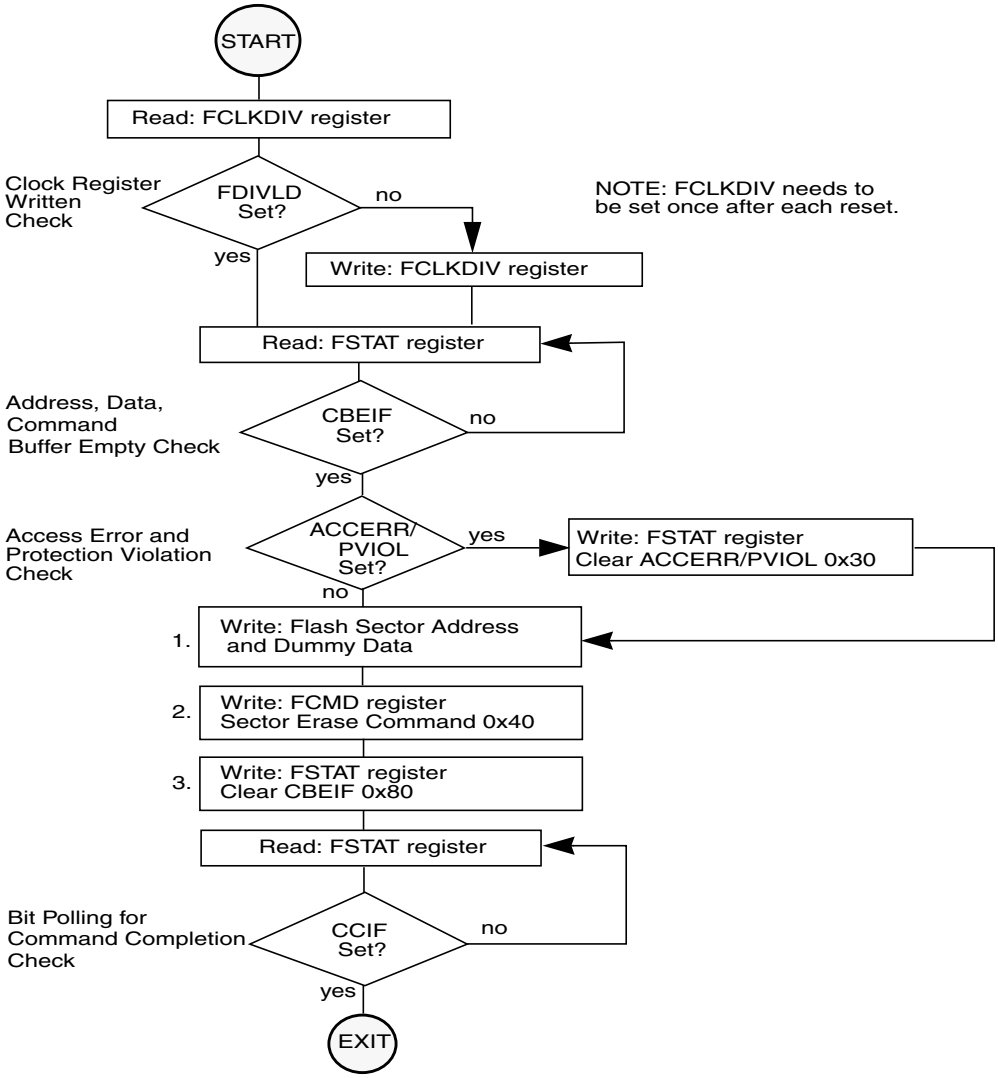

Figure 21-24. Example Sector Erase Command Flow

addresses sequentially staring with 0xFF00-0xFF01 and ending with 0xFF06–0xFF07. The values 0x0000 and 0xFFFF are not permitted as keys. When the KEYACC bit is set, reads of the Flash array will return invalid data.

The user code stored in the Flash array must have a method of receiving the backdoor key from an external stimulus. This external stimulus would typically be through one of the on-chip serial ports.

If KEYEN[1:0] = 1:0 in the FSEC register, the MCU can be unsecured by the backdoor key access sequence described below:

1. Set the KEYACC bit in the FCNFG register
2. Write the correct four 16-bit words to Flash addresses 0xFF00–0xFF07 sequentially starting with 0xFF00
3. Clear the KEYACC bit in the FCNFG register
4. If all four 16-bit words match the backdoor key stored in Flash addresses 0xFF00–0xFF07, the MCU is unsecured and bits SEC[1:0] in the FSEC register are forced to the unsecure state of 1:0

The backdoor key access sequence is monitored by the internal security state machine. An illegal operation during the backdoor key access sequence will cause the security state machine to lock, leaving the MCU in the secured state. A reset of the MCU will cause the security state machine to exit the lock state and allow a new backdoor key access sequence to be attempted. The following illegal operations will lock the security state machine:

1. If any of the four 16-bit words does not match the backdoor key programmed in the Flash array
2. If the four 16-bit words are written in the wrong sequence
3. If more than four 16-bit words are written
4. If any of the four 16-bit words written are 0x0000 or 0xFFFF
5. If the KEYACC bit does not remain set while the four 16-bit words are written

After the backdoor key access sequence has been correctly matched, the MCU will be unsecured. The Flash security byte can be programmed to the unsecure state, if desired.

In the unsecure state, the user has full control of the contents of the four word backdoor key by programming bytes 0xFF00–0xFF07 of the Flash configuration field.

The security as defined in the Flash security/options byte at address 0xFF0F is not changed by using the backdoor key access sequence to unsecure. The backdoor key stored in addresses 0xFF00–0xFF07 is unaffected by the backdoor key access sequence. After the next reset sequence, the security state of the Flash module is determined by the Flash security/options byte at address 0xFF0F. The backdoor key access sequence has no effect on the program and erase protection defined in the FPROT register.

It is not possible to unsecure the MCU in special single chip mode by executing the backdoor key access sequence in background debug mode.