

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	HCS12
Core Size	16-Bit
Speed	25MHz
Connectivity	EBI/EMI, SCI, SPI
Peripherals	POR, PWM, WDT
Number of I/O	35
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 ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	52-LQFP
Supplier Device Package	52-TQFP (10x10)
Purchase URL	<a href="https://www.e-xfl.com/product-detail/nxp-semiconductors/mc9s12gc96vpber">https://www.e-xfl.com/product-detail/nxp-semiconductors/mc9s12gc96vpber</a>

### 1.2.3 Part ID Assignments

The part ID is located in two 8-bit registers PARTIDH and PARTIDL (addresses 0x001A and 0x001B 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 <sup>(1)</sup>
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

## 2.2 Signal Description

This section lists and describes the signals that do connect off-chip.

Table 2-1 shows all pins and their functions that are controlled by the PIM module. If there is more than one function associated to a pin, the priority is indicated by the position in the table from top (highest priority) to down (lowest priority).

**Table 2-1. Pin Functions and Priorities**

Port	Pin Name	Pin Function	Description	Pin Function after Reset	
Port T	PT[7:0]	PWM[4:0]	PWM outputs (only available if enabled in MODRR register)	GPIO	
		IOC[7:0]	Standard timer channels		
		GPIO	General-purpose I/O		
Port S	PS3	GPIO	General-purpose I/O		
	PS2	GPIO	General purpose I/O		
	PS1	TXD	Serial communication interface transmit pin		
		GPIO	General-purpose I/O		
	PS0	RXD	Serial communication interface receive pin		
		GPIO	General-purpose I/O		
Port M	PM5	SCK	SPI clock		
	PM4	MOSI	SPI transmit pin		
	PM3	SS	SPI slave select line		
	PM2	MISO	SPI receive pin		
	PM1	TXCAN	MSCAN transmit pin		
	PM0	RXCAN	MSCAN receive pin		
Port P	PP[7:0]	PWM[5:0]	PWM outputs		
		GPIO[7:0]	General purpose I/O with interrupt		
	PP[6]	ROMON	ROMON input signal		
Port J	PJ[7:6]	GPIO	General purpose I/O with interrupt		
Port AD	PAD[7:0]	ATD[7:0]	ATD analog inputs		
		GPIO[7:0]	General purpose I/O		
Port A	PA[7:0]	ADDR[15:8]/ DATA[15:8]/ GPIO	Refer to MEBI Block Guide.		
Port B	PB[7:0]	ADDR[7:0]/ DATA[7:0]/ GPIO	Refer to MEBI Block Guide.		

## Chapter 2 Port Integration Module (PIM9C32) Block Description

Address	Name		Bit 7	6	5	4	3	2	1	Bit 0
0x0006	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x0007	MODRR	R	0	0	0	MODRR4	MODRR3	MODRR2	MODRR1	MODRR0
		W								
0x0008	PTS	R	0	0	0	0	PTS3	PTS2	PTS1	PTS0
		W								
		SCI	—	—	—	—	—	—	TXD	RXD
0x0009	PTIS	R	0	0	0	0	PTIS3	PTIS2	PTIS1	PTIS0
		W								
0x000A	DDRS	R	0	0	0	0	DDRS3	DDRS2	DDRS1	DDRS0
		W								
0x000B	RDRS	R	0	0	0	0	RDRS3	RDRS2	RDRS1	RDRS0
		W								
0x000C	PERS	R	0	0	0	0	PERS3	PERS2	PERS1	PERS0
		W								
0x000D	PPSS	R	0	0	0	0	PPSS3	PPSS2	PPSS1	PPSS0
		W								
0x000E	WOMS	R	0	0	0	0	WOMS3	WOMS2	WOMS1	WOMS0
		W								
0x000F	Reserved	R	0	0	0	0	0	0	0	0
		W								
		R	0	0	PTM5	PTM4	PTM3	PTM2	PTM1	PTM0
		W								
0x0010	PTM	MSCAN / SPI	—	—	SCK	MOSI	$\overline{SS}$	MISO	TXCAN	RXCAN
0x0011	PTIM	R	0	0	PTIM5	PTIM4	PTIM3	PTIM2	PTIM1	PTIM0
		W								
0x0012	DDRM	R	0	0	DDRM5	DDRM4	DDRM3	DDRM2	DDRM1	DDRM0
		W								
0x0013	RDRM	R	0	0	RDRM5	RDRM4	RDRM3	RDRM2	RDRM1	RDRM0
		W								
0x0014	PERM	R	0	0	PERM5	PERM4	PERM3	PERM2	PERM1	PERM0
		W								
0x0015	PPSM	R	0	0	PPSM5	PPSM4	PPSM3	PPSM2	PPSM1	PPSM0
		W								
0x0016	WOMM	R	0	0	WOMM5	WOMM4	WOMM3	WOMM2	WOMM1	WOMM0
		W								
0x0017	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x0018	PTP	R	PTP7	PTP6	PTP5	PTP4	PTP3	PTP2	PTP1	PTP0
		W								
		PWM	—	—	PWM5	PWM4	PWM3	PWM2	PWM1	PWM0
0x0019	PTIP	R	PTIP7	PTIP6	PTIP5	PTIP4	PTIP3	PTIP2	PTIP1	PTIP0
		W								

= Unimplemented or Reserved

**Figure 2-2. Quick Reference to PIM Registers (Sheet 2 of 3)**

**Table 3-20. 48K Byte Physical FLASH/ROM Allocated**

Address Space	Page Window Access	ROMHM	ECS	XAB19:14
0x0000–0x3FFF	N/A	N/A	1	0x3D
0x4000–0x7FFF	N/A	0	0	0x3E
	N/A	1	1	
0x8000–0xBFFF	External	N/A	1	PIX[5:0]
	Internal	N/A	0	
0xC000–0xFFFF	N/A	N/A	0	0x3F

**Table 3-21. 64K Byte Physical FLASH/ROM Allocated**

Address Space	Page Window Access	ROMHM	ECS	XAB19:14
0x0000–0x3FFF	N/A	0	0	0x3D
	N/A	1	1	
0x4000–0x7FFF	N/A	0	0	0x3E
	N/A	1	1	
0x8000–0xBFFF	External	N/A	1	PIX[5:0]
	Internal	N/A	0	
0xC000–0xFFFF	N/A	N/A	0	0x3F



## 6.4.8 Hardware Handshake Abort Procedure

The abort procedure is based on the SYNC command. In order to abort a command, which had not issued the corresponding ACK pulse, the host controller should generate a low pulse in the BKGD pin by driving it low for at least 128 serial clock cycles and then driving it high for one serial clock cycle, providing a speedup pulse. By detecting this long low pulse in the BKGD pin, the target executes the SYNC protocol, see [Section 6.4.9, “SYNC — Request Timed Reference Pulse,”](#) and assumes that the pending command and therefore the related ACK pulse, are being aborted. Therefore, after the SYNC protocol has been completed the host is free to issue new BDM commands.

Although it is not recommended, the host could abort a pending BDM command by issuing a low pulse in the BKGD pin shorter than 128 serial clock cycles, which will not be interpreted as the SYNC command. The ACK is actually aborted when a falling edge is perceived by the target in the BKGD pin. The short abort pulse should have at least 4 clock cycles keeping the BKGD pin low, in order to allow the falling edge to be detected by the target. In this case, the target will not execute the SYNC protocol but the pending command will be aborted along with the ACK pulse. The potential problem with this abort procedure is when there is a conflict between the ACK pulse and the short abort pulse. In this case, the target may not perceive the abort pulse. The worst case is when the pending command is a read command (i.e., READ\_BYTE). If the abort pulse is not perceived by the target the host will attempt to send a new command after the abort pulse was issued, while the target expects the host to retrieve the accessed memory byte. In this case, host and target will run out of synchronism. However, if the command to be aborted is not a read command the short abort pulse could be used. After a command is aborted the target assumes the next falling edge, after the abort pulse, is the first bit of a new BDM command.

### NOTE

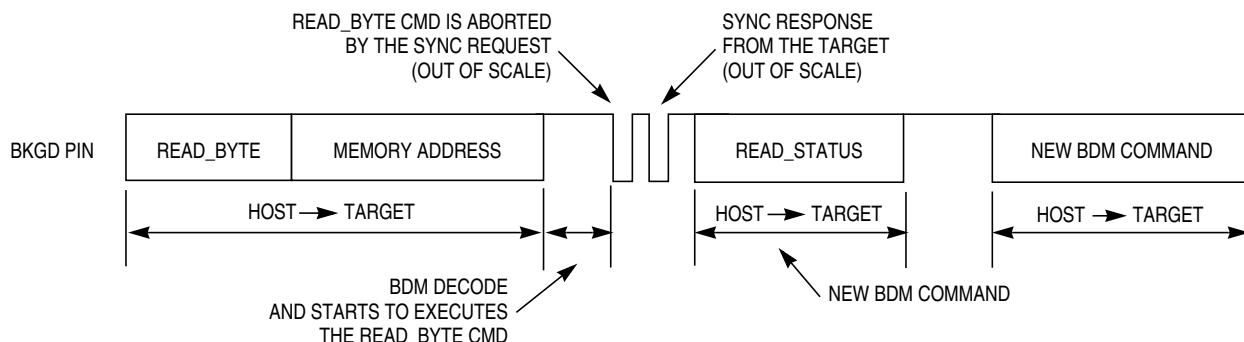
The details about the short abort pulse are being provided only as a reference for the reader to better understand the BDM internal behavior. It is not recommended that this procedure be used in a real application.

Because the host knows the target serial clock frequency, the SYNC command (used to abort a command) does not need to consider the lower possible target frequency. In this case, the host could issue a SYNC very close to the 128 serial clock cycles length. Providing a small overhead on the pulse length in order to assure the SYNC pulse will not be misinterpreted by the target. See [Section 6.4.9, “SYNC — Request Timed Reference Pulse.”](#)

[Figure 6-12](#) shows a SYNC command being issued after a READ\_BYTE, which aborts the READ\_BYTE command. Note that, after the command is aborted a new command could be issued by the host computer.

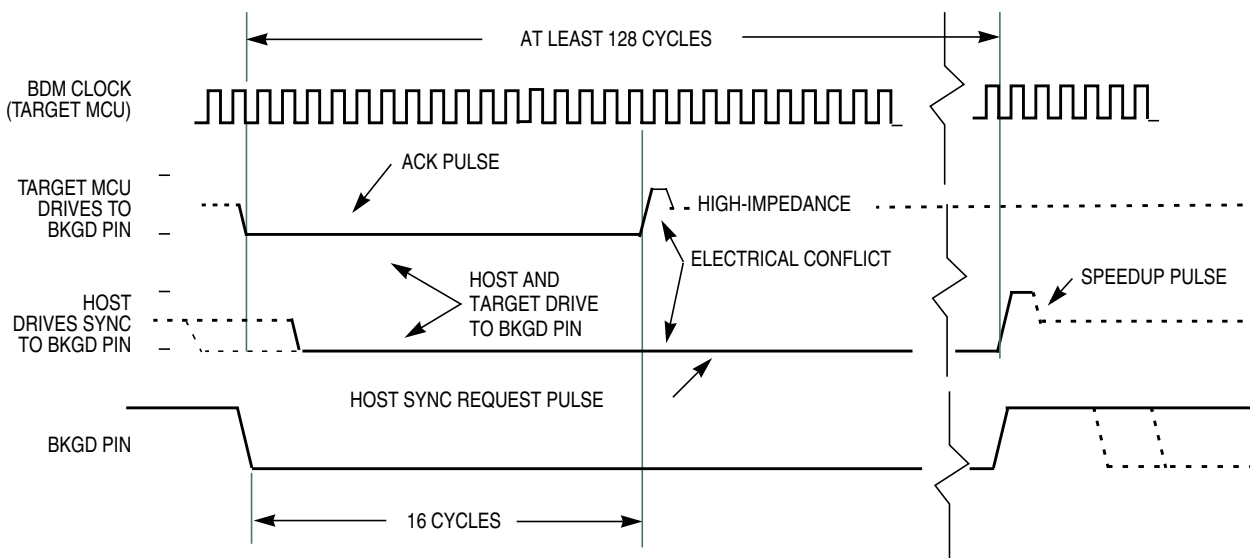
### NOTE

[Figure 6-12](#) does not represent the signals in a true timing scale



**Figure 6-12. ACK Abort Procedure at the Command Level**

Figure 6-13 shows a conflict between the ACK pulse and the SYNC request pulse. This conflict could occur if a POD device is connected to the target BKGD pin and the target is already in debug active mode. Consider that the target CPU is executing a pending BDM command at the exact moment the POD is being connected to the BKGD pin. In this case, an ACK pulse is issued along with the SYNC command. In this case, there is an electrical conflict between the ACK speedup pulse and the SYNC pulse. Because this is not a probable situation, the protocol does not prevent this conflict from happening.



**Figure 6-13. ACK Pulse and SYNC Request Conflict**

### NOTE

This information is being provided so that the MCU integrator will be aware that such a conflict could eventually occur.

The hardware handshake protocol is enabled by the `ACK_ENABLE` and disabled by the `ACK_DISABLE` BDM commands. This provides backwards compatibility with the existing POD devices which are not able to execute the hardware handshake protocol. It also allows for new POD devices, that support the hardware handshake protocol, to freely communicate with the target device. If desired, without the need for waiting for the ACK pulse.



### 7.4.2.6.3 Detail Mode

In the detail mode, address and data for all cycles except program fetch (P) and free (f) cycles are stored in trace buffer. This mode is intended to supply additional information on indexed, indirect addressing modes where storing only the destination address would not provide all information required for a user to determine where his code was in error.

### 7.4.2.6.4 Profile Mode

This mode is intended to allow a host computer to poll a running target and provide a histogram of program execution. Each read of the trace buffer address will return the address of the last instruction executed. The DBG CNT register is not incremented and the trace buffer does not get filled. The ARM bit is not used and all breakpoints and all other debug functions will be disabled.

## 7.4.2.7 Storage Memory

The storage memory is a 64 words deep by 16-bits wide dual port RAM array. The CPU accesses the RAM array through a single memory location window (DBGTBH:DBGTBL). The DBG module stores trace information in the RAM array in a circular buffer format. As data is read via the CPU, a pointer into the RAM will increment so that the next CPU read will receive fresh information. In all trigger modes except for event-only and detail capture mode, the data stored in the trace buffer will be change-of-flow addresses. change-of-flow addresses are defined as follows:

- Source address of conditional branches (long, short, BRSET, and loop constructs) taken
- Destination address of indexed JMP, JSR, and CALL instruction
- Destination address of RTI, RTS, and RTC instructions
- Vector address of interrupts except for SWI and BDM vectors

In the event-only trigger modes only the 16-bit data bus value corresponding to the event is stored. In the detail capture mode, address and then data are stored for all cycles except program fetch (P) and free (f) cycles.

## 7.4.2.8 Storing Data in Memory Storage Buffer

### 7.4.2.8.1 Storing with Begin-Trigger

Storing with begin-trigger can be used in all trigger modes. When DBG mode is enabled and armed in the begin-trigger mode, data is not stored in the trace buffer until the trigger condition is met. As soon as the trigger condition is met, the DBG module will remain armed until 64 words 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 event will be stored in the trace buffer.

### 7.4.2.8.2 Storing with End-Trigger

Storing with end-trigger cannot be used in event-only trigger modes. When DBG mode is enabled and armed in the end-trigger mode, data is stored in the trace buffer until the trigger condition is met. When the trigger condition is met, the DBG module will become de-armed and no more data will be stored. If

## 8.2 Signal Description

The ATD10B8C has a total of 12 external pins.

### 8.2.1 AN7 / ETRIG / PAD7

This pin serves as the analog input channel 7. It can be configured to provide an external trigger for the ATD conversion. It can be configured as general-purpose digital I/O.

### 8.2.2 AN6 / PAD6

This pin serves as the analog input channel 6. It can be configured as general-purpose digital I/O.

### 8.2.3 AN5 / PAD5

This pin serves as the analog input channel 5. It can be configured as general-purpose digital I/O.

### 8.2.4 AN4 / PAD4

This pin serves as the analog input channel 4. It can be configured as general-purpose digital I/O.

### 8.2.5 AN3 / PAD3

This pin serves as the analog input channel 3. It can be configured as general-purpose digital I/O.

### 8.2.6 AN2 / PAD2

This pin serves as the analog input channel 2. It can be configured as general-purpose digital I/O.

### 8.2.7 AN1 / PAD1

This pin serves as the analog input channel 1. It can be configured as general-purpose digital I/O.

### 8.2.8 AN0 / PAD0

This pin serves as the analog input channel 0. It can be configured as general-purpose digital I/O.

### 8.2.9 $V_{RH}$ , $V_{RL}$

$V_{RH}$  is the high reference voltage and  $V_{RL}$  is the low reference voltage for ATD conversion.

### 8.2.10 $V_{DDA}$ , $V_{SSA}$

These pins are the power supplies for the analog circuitry of the ATD10B8C block.

**Table 9-12. Outcome of Clock Loss in Pseudo-Stop Mode (continued)**

CME	SCME	SCMIE	CRG Actions
1	1	1	<p>Clock failure --&gt;</p> <ul style="list-style-type: none"> <li>– VREG enabled,</li> <li>– PLL enabled,</li> <li>– SCM activated,</li> <li>– Start Clock Quality Check,</li> <li>– SCMIF set.</li> </ul> <p>SCMIF generates Self-Clock Mode wakeup interrupt.</p> <ul style="list-style-type: none"> <li>– Exit Pseudo-Stop Mode in SCM using PLL clock (<math>f_{SCM}</math>) as system clock,</li> <li>– Continue to perform a additional Clock Quality Checks until OSCCLK is o.k. again.</li> </ul>

#### 9.4.10.2 Wake-up from Full Stop (PSTP=0)

The MCU requires an external interrupt or an external reset in order to wake-up from stop mode.

If the MCU gets an external reset during full stop mode active, the CRG asynchronously restores all configuration bits in the register space to its default settings and will perform a maximum of 50 clock *check\_windows* (see [Section 9.4.4, “Clock Quality Checker”](#)). After completing the clock quality check the CRG starts the reset generator. After completing the reset sequence processing begins by fetching the normal reset vector. Full stop mode is exited and the MCU is in run mode again.

If the MCU is woken-up by an interrupt, the CRG will also perform a maximum of 50 clock *check\_windows* (see [Section 9.4.4, “Clock Quality Checker”](#)). If the clock quality check is successful, the CRG will release all system and core clocks and will continue with normal operation. If all clock checks within the timeout-window are failing, the CRG will switch to self-clock mode or generate a clock monitor reset (CMRESET) depending on the setting of the SCME bit.

Because the PLL has been powered-down during stop mode the PLLSEL bit is cleared and the MCU runs on OSCCLK after leaving stop mode. The software must manually set the PLLSEL bit again, in order to switch system and core clocks to the PLLCLK.

#### NOTE

In full stop mode, the clock monitor is disabled and any loss of clock will not be detected.

## 9.5 Resets

This section describes how to reset the CRGV4 and how the CRGV4 itself controls the reset of the MCU. It explains all special reset requirements. Because the reset generator for the MCU is part of the CRG, this section also describes all automatic actions that occur during or as a result of individual reset conditions. The reset values of registers and signals are provided in [Section 9.3, “Memory Map and Register](#)

### 12.4.1.3 Clock Select

Each PWM channel has the capability of selecting one of two clocks. For channels 0, 1, 4, and 5 the clock choices are clock A or clock SA. For channels 2 and 3 the choices are clock B or clock SB. The clock selection is done with the PCLKx control bits in the PWMCLK register.

#### NOTE

Changing clock control bits while channels are operating can cause irregularities in the PWM outputs.

### 12.4.2 PWM Channel Timers

The main part of the PWM module are the actual timers. Each of the timer channels has a counter, a period register and a duty register (each are 8 bit). The waveform output period is controlled by a match between the period register and the value in the counter. The duty is controlled by a match between the duty register and the counter value and causes the state of the output to change during the period. The starting polarity of the output is also selectable on a per channel basis. Figure 12-35 shows a block diagram for PWM timer.

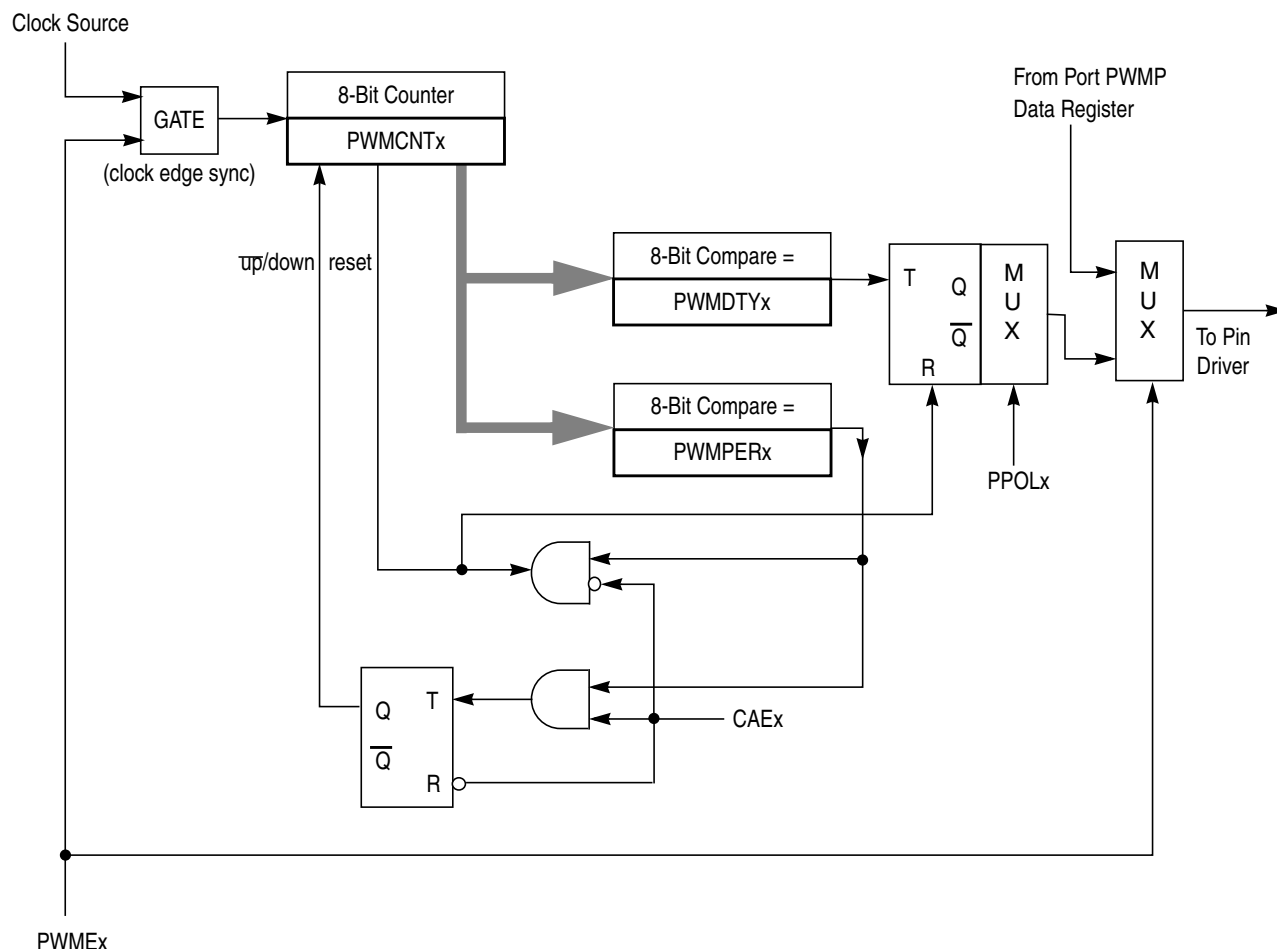


Figure 12-35. PWM Timer Channel Block Diagram

### 13.5.2.2 Interrupt Descriptions

The SCI only originates interrupt requests. The following is a description of how the SCI makes a request and how the MCU should acknowledge that request. The interrupt vector offset and interrupt number are chip dependent. The SCI only has a single interrupt line (**SCI Interrupt Signal**, active high operation) and all the following interrupts, when generated, are ORed together and issued through that port.

#### 13.5.2.2.1 TDRE Description

The TDRE interrupt is set high by the SCI when the transmit shift register receives a byte from the SCI data register. A TDRE interrupt indicates that the transmit data register (SCIDRH/L) is empty and that a new byte can be written to the SCIDRH/L for transmission. Clear TDRE by reading SCI status register 1 with TDRE set and then writing to SCI data register low (SCIDRL).

#### 13.5.2.2.2 TC Description

The TC interrupt is set by the SCI when a transmission has been completed. A TC interrupt indicates that there is no transmission in progress. TC is set high when the TDRE flag is set and no data, preamble, or break character is being transmitted. When TC is set, the TXD pin becomes idle (logic 1). Clear TC by reading SCI status register 1 (SCISR1) with TC set and then writing to SCI data register low (SCIDRL). TC is cleared automatically when data, preamble, or break is queued and ready to be sent.

#### 13.5.2.2.3 RDRF Description

The RDRF interrupt is set when the data in the receive shift register transfers to the SCI data register. A RDRF interrupt indicates that the received data has been transferred to the SCI data register and that the byte can now be read by the MCU. The RDRF interrupt is cleared by reading the SCI status register one (SCISR1) and then reading SCI data register low (SCIDRL).

#### 13.5.2.2.4 OR Description

The OR interrupt is set when software fails to read the SCI data register before the receive shift register receives the next frame. The newly acquired data in the shift register will be lost in this case, but the data already in the SCI data registers is not affected. The OR interrupt is cleared by reading the SCI status register one (SCISR1) and then reading SCI data register low (SCIDRL).

### 13.5.2.3 IDLE Description

The IDLE interrupt is set when 10 consecutive logic 1s (if M = 0) or 11 consecutive logic 1s (if M = 1) appear on the receiver input. Once the IDLE is cleared, a valid frame must again set the RDRF flag before an idle condition can set the IDLE flag. Clear IDLE by reading SCI status register 1 (SCISR1) with IDLE set and then reading SCI data register low (SCIDRL).

### 13.5.3 Recovery from Wait Mode

The SCI interrupt request can be used to bring the CPU out of wait mode.

### 14.4.1 Master Mode

The SPI operates in master mode when the MSTR bit is set. Only a master SPI module can initiate transmissions. A transmission begins by writing to the master SPI Data Register. If the shift register is empty, the byte immediately transfers to the shift register. The byte begins shifting out on the MOSI pin under the control of the serial clock.

- S-clock

The SPR2, SPR1, and SPR0 baud rate selection bits in conjunction with the SPPR2, SPPR1, and SPPR0 baud rate preselection bits in the SPI Baud Rate register control the baud rate generator and determine the speed of the transmission. The SCK pin is the SPI clock output. Through the SCK pin, the baud rate generator of the master controls the shift register of the slave peripheral.

- MOSI and MISO Pins

In master mode, the function of the serial data output pin (MOSI) and the serial data input pin (MISO) is determined by the SPC0 and BIDIROE control bits.

- $\overline{SS}$  Pin

If MODFEN and SSOE bit are set, the  $\overline{SS}$  pin is configured as slave select output. The  $\overline{SS}$  output becomes low during each transmission and is high when the SPI is in idle state.

If MODFEN is set and SSOE is cleared, the  $\overline{SS}$  pin is configured as input for detecting mode fault error. If the  $\overline{SS}$  input becomes low this indicates a mode fault error where another master tries to drive the MOSI and SCK lines. In this case, the SPI immediately switches to slave mode, by clearing the MSTR bit and also disables the slave output buffer MISO (or SISO in bidirectional mode). So the result is that all outputs are disabled and SCK, MOSI and MISO are inputs. If a transmission is in progress when the mode fault occurs, the transmission is aborted and the SPI is forced into idle state.

This mode fault error also sets the mode fault (MODF) flag in the SPI Status Register (SPISR). If the SPI interrupt enable bit (SPIE) is set when the MODF flag gets set, then an SPI interrupt sequence is also requested.

When a write to the SPI Data Register in the master occurs, there is a half SCK-cycle delay. After the delay, SCK is started within the master. The rest of the transfer operation differs slightly, depending on the clock format specified by the SPI clock phase bit, CPHA, in SPI Control Register 1 (see [Section 14.4.3](#), “Transmission Formats”).

#### NOTE

A change of the bits CPOL, CPHA, SSOE, LSBFE, MODFEN, SPC0, BIDIROE with SPC0 set, SPPR2–SPPR0 and SPR2–SPR0 in master mode will abort a transmission in progress and force the SPI into idle state. The remote slave cannot detect this, therefore the master has to ensure that the remote slave is set back to idle state.

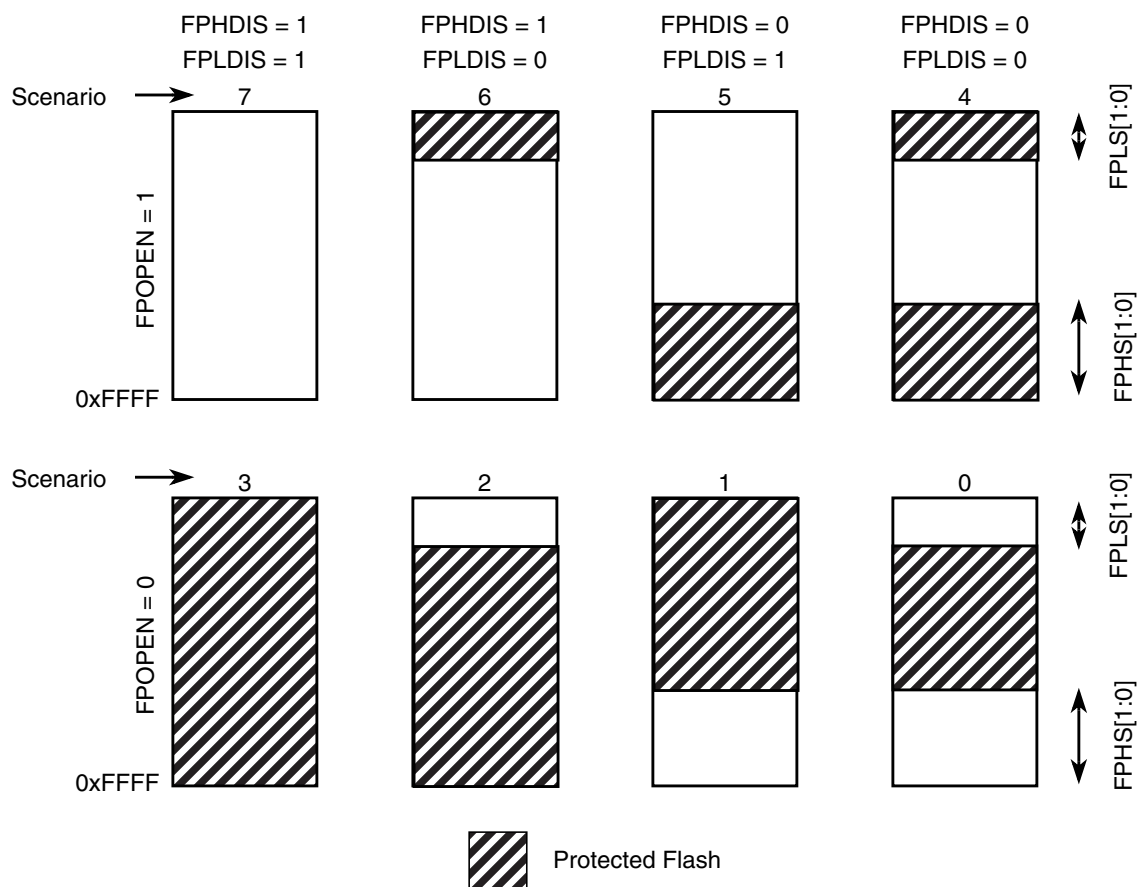


Figure 18-9. Flash Protection Scenarios

### 18.3.2.5.1 Flash Protection Restrictions

The general guideline is that protection can only be added, not removed. All valid transitions between Flash protection scenarios are specified in [Table 18-12](#). Any attempt to write an invalid scenario to the FPROT register will be ignored and the FPROT register will remain unchanged. The contents of the FPROT register reflect the active protection scenario.

Table 18-12. Flash Protection Scenario Transitions

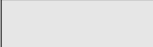
From Protection Scenario	To Protection Scenario <sup>(1)</sup>							
	0	1	2	3	4	5	6	7
0	X	X	X	X				
1		X		X				
2			X	X				
3				X				
4				X	X			
5			X	X	X	X		

### 19.3.2.14 RESERVED6

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

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 19-23. RESERVED6**

All bits read 0 and are not writable.

## 19.4 Functional Description

### 19.4.1 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

#### 19.4.1.1 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



Table 20-15. FCMD Field Descriptions

Field	Description
6, 5, 2, 0 CMDB[6:5] CMDB[2] CMDB[0]	Valid Flash commands are shown in Table 20-16. An attempt to execute any command other than those listed in Table 20-16 will set the ACCERR bit in the FSTAT register (see Section 20.3.2.6).

Table 20-16. Valid Flash Command List

CMDB	NVM Command
0x05	Erase verify
0x20	Word program
0x40	Sector erase
0x41	Mass erase

### 20.3.2.8 RESERVED2

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

Module Base + 0x0007

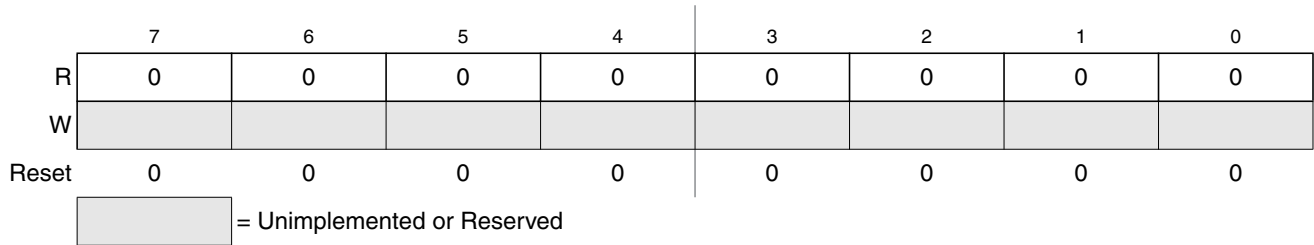


Figure 20-14. RESERVED2

All bits read 0 and are not writable.

### 20.3.2.9 Flash Address Register (FADDR)

FADDRHI and FADDRLO are the Flash address registers.

Module Base + 0x0008

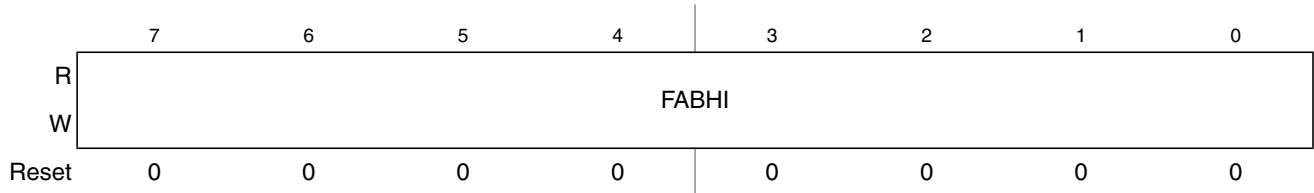


Figure 20-15. Flash Address High Register (FADDRHI)

then FCLKDIV register bits PRDIV8 and FDIV[5:0] are to be set as described in [Figure 20-23](#).

For example, if the oscillator clock frequency is 950 kHz and the bus clock is 10 MHz, FCLKDIV bits FDIV[5:0] should be set to 4 (000100) and bit PRDIV8 set to 0. The resulting FCLK is then 190 kHz. As a result, the Flash algorithm timings are increased over optimum target by:

$$(200 - 190) / 200 \times 100 = 5\%$$

Command execution time will increase proportionally with the period of FCLK.

### CAUTION

Because of the impact of clock synchronization on the accuracy of the functional timings, programming or erasing the Flash array cannot be performed if the bus clock runs at less than 1 MHz. Programming or erasing the Flash array with an input clock < 150 kHz should be avoided. Setting FCLKDIV to a value such that FCLK < 150 kHz can destroy the Flash array due to overstress. Setting FCLKDIV to a value such that  $(1/\text{FCLK} + T_{\text{bus}}) < 5\mu\text{s}$  can result in incomplete programming or erasure of the Flash array cells.

If the FCLKDIV register is written, the bit FDIVLD is set automatically. If the FDIVLD bit is 0, the FCLKDIV register has not been written since the last reset. If the FCLKDIV register has not been written to, the Flash command loaded during a command write sequence will not execute and the ACCERR flag in the FSTAT register will set.

### 20.4.1.3.2 Program Command

The program operation will program a previously erased word in the Flash array using an embedded algorithm.

An example flow to execute the program operation is shown in [Figure 20-25](#). The program command write sequence is as follows:

1. Write to a Flash array address to start the command write sequence for the program command. The data written will be programmed to the Flash array address written.
2. Write the program command, 0x20, to the FCMD register.
3. Clear the CBEIF flag in the FSTAT register by writing a 1 to CBEIF to launch the program command.

If a word to be programmed is in a protected area of the Flash array, the PVIOL flag in the FSTAT register will set and the program command will not launch. Once the program command has successfully launched, the CCIF flag in the FSTAT register will set after the program operation has completed unless a new command write sequence has been buffered. By executing a new program command write sequence on sequential words after the CBEIF flag in the FSTAT register has been set, up to 55% faster programming time per word can be effectively achieved than by waiting for the CCIF flag to set after each program operation.

#### 20.4.1.3.4 Mass Erase Command

The mass erase operation will erase all addresses in a Flash array using an embedded algorithm.

An example flow to execute the mass erase operation is shown in [Figure 20-27](#). The mass erase command write sequence is as follows:

1. Write to a Flash array address to start the command write sequence for the mass erase command. The address and data written will be ignored.
2. Write the mass erase command, 0x41, to the FCMD register.
3. Clear the CBEIF flag in the FSTAT register by writing a 1 to CBEIF to launch the mass erase command.

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

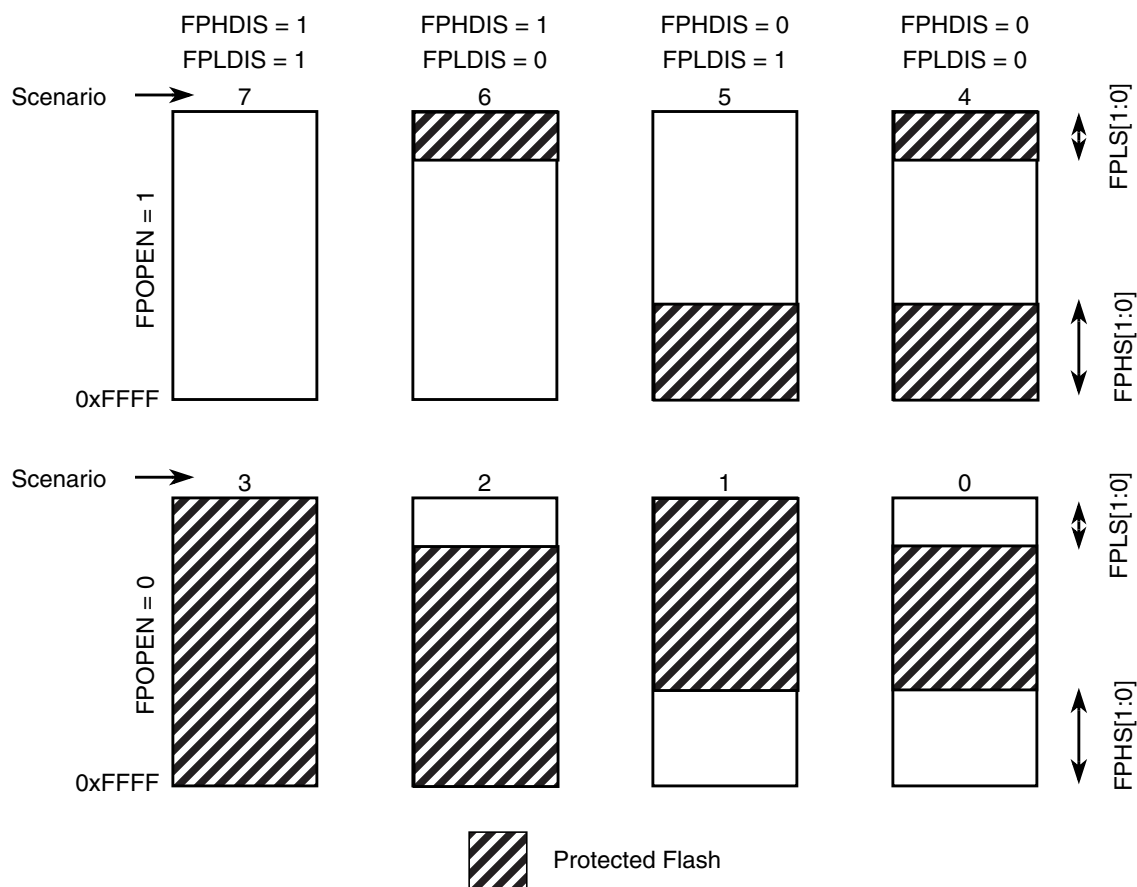


Figure 21-9. Flash Protection Scenarios

### 21.3.2.5.1 Flash Protection Restrictions

The general guideline is that protection can only be added, not removed. All valid transitions between Flash protection scenarios are specified in [Table 21-12](#). Any attempt to write an invalid scenario to the FPROT register will be ignored and the FPROT register will remain unchanged. The contents of the FPROT register reflect the active protection scenario.

Table 21-12. Flash Protection Scenario Transitions

From Protection Scenario	To Protection Scenario <sup>(1)</sup>							
	0	1	2	3	4	5	6	7
0	X	X	X	X				
1		X		X				
2			X	X				
3				X				
4				X	X			
5			X	X	X	X		