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	CANbus, EBI/EMI, I²C, SCI, SPI
Peripherals	LVD, POR, PWM, WDT
Number of I/O	91
Program Memory Size	256KB (256K x 8)
Program Memory Type	FLASH
EEPROM Size	4K x 8
RAM Size	12K x 8
Voltage - Supply (Vcc/Vdd)	2.35V ~ 5.5V
Data Converters	A/D 16x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	112-LQFP
Supplier Device Package	112-LQFP (20x20)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc9s12kt256vpve

1.1.3 MC9S12KT256 Block Diagram

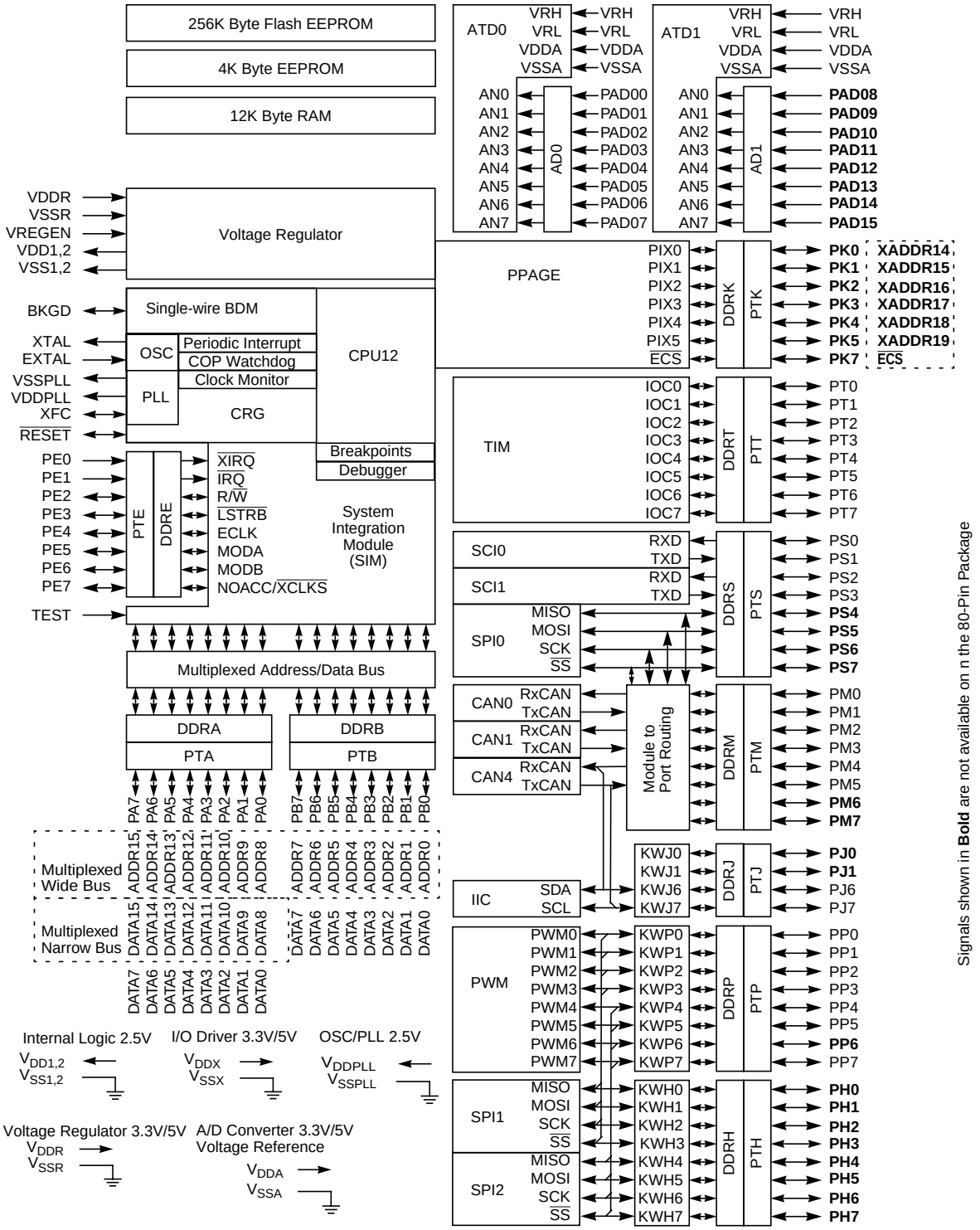


Figure 1-2. MC9S12KT256 Block Diagram

0x02C0–0x02E7 PWM (Pulse Width Modulator 8 Bit 8 Channel) (continued)

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x02D6	PWMPER2	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x02D7	PWMPER3	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x02D8	PWMPER4	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x02D9	PWMPER5	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x02DA	PWMPER6	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x02DB	PWMPER7	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x02DC	PWMDTY0	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x02DD	PWMDTY1	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x02DE	PWMDTY2	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x02DF	PWMDTY3	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x02E0	PWMDTY4	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x02E1	PWMDTY5	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x02E2	PWMDTY6	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x02E3	PWMDTY7	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x02E4	PWMSDN	R	PWMIF	PWMIE	PWMRSTRT	PWMLVL	0	PWM7IN	PWM7INL	PWM7ENA
		W								
0x02E5	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x02E6	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x02E7	Reserved	R	0	0	0	0	0	0	0	0
		W								

0x02E8–0x03FF Reserved Space

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x02E8–0x03FF	Reserved	R	0	0	0	0	0	0	0	0
		W								

FPROT register reflect the active protection scenario. See the FPHS and FPLS descriptions for additional restrictions.

Table 2-16. Flash Protection Scenario Transitions

From Protection Scenario	To Protection Scenario[1]							
	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		
6		X		X	X		X	
7	X	X	X	X	X	X	X	X

1 Allowed transitions marked with X.

2.3.2.7 Flash Status Register (FSTAT)

The banked FSTAT register defines the operational status of the module.

Module Base + 0x0005

	7	6	5	4	3	2	1	0
R	CBEIF	CCIF	PVIOL	ACCERR	DFDIF	BLANK	0	0
W								
Reset	1	1	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 2-9. Flash Status Register (FSTAT - Normal Mode)

Module Base + 0x0005

	7	6	5	4	3	2	1	0
R	CBEIF	CCIF	PVIOL	ACCERR	DFDIF	BLANK	FAIL	0
W								
Reset	1	1	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 2-10. Flash Status Register (FSTAT - Special Mode)

Table 3-8. 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 3.6, “Flash Module Security”.

3.3.2.3 Flash Test Mode Register (FTSTMOD)

The unbanked FTSTMOD register is used to control Flash test features.

Module Base + 0x0002

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

= Unimplemented or Reserved

Figure 3-6. Flash Test Mode Register (FTSTMOD)

FDFD is readable and writable while all remaining bits read 0 and are not writable in normal mode. The WRALL bit is writable only in special mode to simplify mass erase and erase verify operations. When writing to the FTSTMOD register in special mode, all unimplemented/reserved bits must be written to 0.

Table 3-9. FTSTMOD Field Descriptions

Field	Description
4 WRALL	**Write to All Register Banks** — If the WRALL bit is set, all banked registers sharing the same register address will be written simultaneously during a register write. 0 Write only to the bank selected via BKSEL. 1 Write to all register banks.
3 FDFD	**Force Double Fault Detect** — The FDFD bit allows the user to simulate a double bit fault during Flash array read operations and check the associated interrupt routine. The FDFD bit is cleared by writing a 0 to FDFD. 0 Flash array read operations will set the DFDIF flag in the FSTAT register only if a double bit fault is detected. 1 Any Flash array read operation will force the DFDIF flag in the FSTAT register to be set and an interrupt will be generated as long as the DFDIE interrupt enable in the FCNFG register is set.

3.3.2.4 Flash Configuration Register (FCNFG)

The unbanked FCNFG register enables the Flash interrupts and gates the security backdoor writes.

Module Base + 0x0008

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

= Unimplemented or Reserved

Figure 3-13. Flash Address High Register (FADDRHI)

Module Base + 0x0009

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

= Unimplemented or Reserved

Figure 3-14. Flash Address Low Register (FADDRLO)

All FADDRHI and FADDRLO bits are readable but are not writable. After an array write as part of a command write sequence, the FADDR registers will contain the mapped MCU address written. If a double bit fault is detected, as indicated by the setting of the DFDIF bit in the FSTAT register, the faulty Flash block address is stored in the FADDR registers as a word address. The faulty Flash block address remains readable until the start of the next command write sequence. The mapping of the FADDR registers to the MCU address is shown in Figure 3-15 and Figure 3-16.

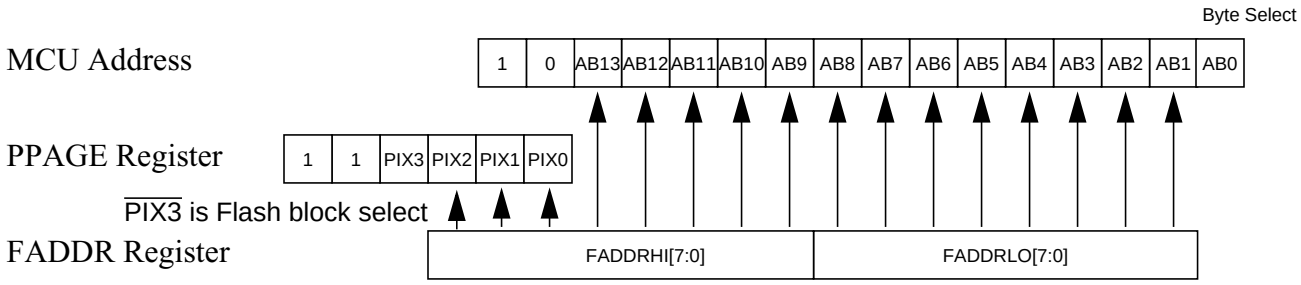


Figure 3-15. FADDR to MCU Address Mapping (Paged)

3.4.1.3.2 Data Compress Command

The data compress command is used to check Flash code integrity by compressing data from a selected portion of the Flash block into a signature analyzer. The starting address for the data compress operation is defined by the address written during the command write sequence. The number of consecutive word addresses compressed is defined by the data written during the command write sequence. The number of words that can be compressed in a single data compress operation ranges from 1 to 16,384. After launching the data compress command, the CCIF flag in the FSTAT register will set after the data compress operation has completed. The number of bus cycles required to execute the data compress operation is equal to two times the number of addresses read plus 20 bus cycles as measured from the time the CBEIF flag is cleared until the CCIF flag is set. After the CCIF flag is set, the signature generated by the data compress operation is available in the FDATA register. The signature in the FDATA register can be compared to the expected signature to determine the integrity of the selected data stored in the Flash block. If the last address of the Flash block is reached during the data compress operation, data compression will continue with the starting address of the Flash block.

NOTE

Since the FDATA register (or data buffer) is written to as part of the data compress operation, a command write sequence is not allowed to be buffered behind a data compress command write sequence. The CBEIF flag will not set after launching the data compress command to indicate that a command must not be buffered behind it. If an attempt is made to start a new command write sequence with a data compress operation active, the ACCERR flag in the FSTAT register will be set. A new command write sequence must only be started after reading the signature stored in the FDATA register. A Flash array read that generates a double bit fault will overwrite the contents of the FDATA register.

In order to take corrective action, it is recommended that the data compress command be executed on a Flash sector or subset of a Flash sector. If the data compress operation on a Flash sector returns an invalid signature, the Flash sector must be erased using the sector erase command and then reprogrammed using the program command.

NOTE

During the data compress operation, the Flash array is read with a sense-amp margin setting that is different from the normal array read setting. Therefore, if the data compress operation returns an invalid signature, the section of the Flash array compressed may still be functional. The failing section of the Flash array could be validated using normal array read operations.

The data compress command can be used to verify that a sector or sequential set of sectors are erased.

If the ECC logic detects a double bit fault during the data compress operation, the operation will terminate immediately and set the DFDIF and ACCERR flags in the FSTAT register. The faulty address will be stored in the FADDR registers and the ECC parity bits read at the faulty address will be stored in the FDATALO register. The CCIF flag will set after the DFDIF flag is set and the faulty information is stored in the FADDR and FDATALO registers.

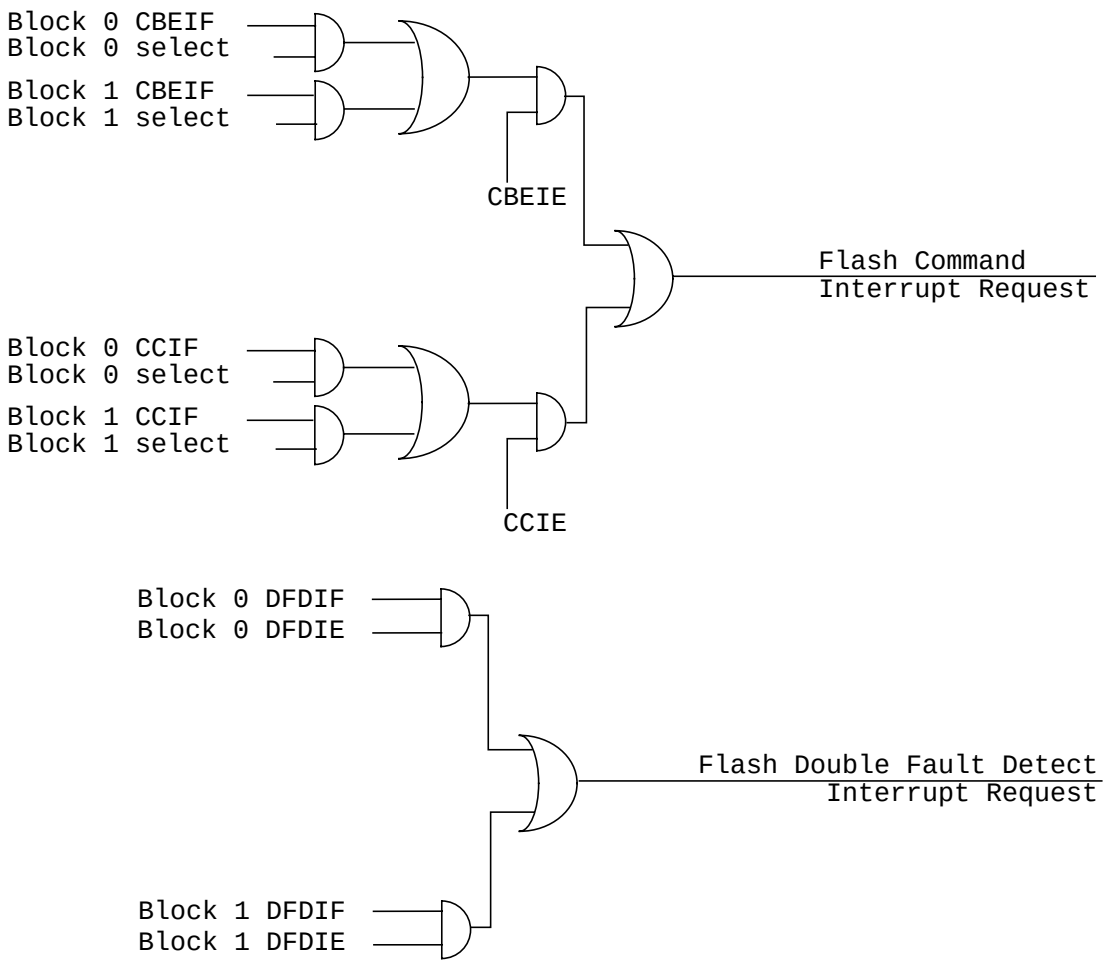


Figure 3-31. Flash Interrupt Implementation

For a detailed description of the register bits, refer to Section 3.3.2.4, “Flash Configuration Register (FCNFG)” and Section 3.3.2.7, “Flash Status Register (FSTAT)”.

Table 5-17. MODRR Field Descriptions (continued)

Field	Description
3–2 MODRR[3:2]	**CAN4 Routing Bits** — See Table 5-19.
1–0 MODRR[1:0]	**CAN0 Routing Bits** — See Table 5-18.

Table 5-18. CAN0 Routing

MODRR[1]	MODRR[0]	RXCAN0	TXCAN0
0	0	PM0	PM1
0	1	PM2	PM3
1	0	PM4	PM5
1	1	Reserved	

Table 5-19. CAN4 Routing

MODRR[3]	MODRR[2]	RXCAN4	TXCAN4
0	0	PJ6	PJ7
0	1	PM4	PM5
1	0	PM6	PM7
1	1	Reserved	

Table 5-20. SPI0 Routing

MODRR[4]	MISO0	MOSI0	SCK0	SS0
0	PS4	PS5	PS6	PS7
1	PM2	PM4	PM5	PM3

Table 5-21. SPI1 Routing

MODRR[5]	MISO1	MOSI1	SCK1	SS1
0	PP0	PP1	PP2	PP3
1	PH0	PH1	PH2	PH3

Table 5-22. SPI2 Routing

MODRR[6]	MISO2	MOSI2	SCK2	SS2
0	PP4	PP5	PP6	PP7
1	PH4	PH5	PH6	PH7

5.3.5.3 Port H Data Direction Register (DDRH)

Module Base + 0x0022

	7	6	5	4	3	2	1	0
R W	DDRH7	DDRH6	DDRH5	DDRH4	DDRH3	DDRH2	DDRH1	DDRH0
Reset	0	0	0	0	0	0	0	0

Figure 5-33. Port H Data Direction Register (DDRH)

Read: Anytime. Write: Anytime.

This register configures each port H pin as either input or output.

Table 5-30. DDRH Field Descriptions

Field	Description
7–0 DDRH]7:0]	**Data Direction Port H** 0 Associated pin is configured as input. 1 Associated pin is configured as output.

5.3.5.4 Port H Reduced Drive Register (RDRH)

Module Base + 0x0023

	7	6	5	4	3	2	1	0
R W	RDRH7	RDRH6	RDRH5	RDRH4	RDRH3	RDRH2	RDRH1	RDRH0
Reset	0	0	0	0	0	0	0	0

Figure 5-34. Port H Reduced Drive Register (RDRH)

Read: Anytime. Write: Anytime.

This register configures the drive strength of each port H output pin as either full or reduced. If the port is used as input this bit is ignored.

Table 5-31. RDRH Field Descriptions

Field	Description
7–0 RDRH[7:0]	**Reduced Drive Port H** 0 Full drive strength at output. 1 Associated pin drives at about 1/6 of the full drive strength.

6.3.2.12 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 6-15. 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.

6.4 Functional Description

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

6.4.1 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.

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

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

Table 8-7. ATDCTL3 Field Descriptions (continued)

Field	Description
2 FIFO	**Result Register FIFO Mode** — If this bit is zero (non-FIFO mode), the A/D conversion results map into the result registers based on the conversion sequence; the result of the first conversion appears in the first result register, the second result in the second result register, and so on. If this bit is one (FIFO mode) the conversion counter is not reset at the beginning or ending of a conversion sequence; sequential conversion results are placed in consecutive result registers. In a continuously scanning conversion sequence, the result register counter will wrap around when it reaches the end of the result register file. The conversion counter value (CC2-0 in ATDSTAT0) can be used to determine where in the result register file, the current conversion result will be placed. Aborting a conversion or starting a new conversion by write to an ATDCTL register (ATDCTL5-0) clears the conversion counter even if FIFO=1. So the first result of a new conversion sequence, started by writing to ATDCTL5, will always be place in the first result register (ATDDDR0). Intended usage of FIFO mode is continuos conversion (SCAN=1) or triggered conversion (ETRIG=1). Finally, which result registers hold valid data can be tracked using the conversion complete flags. Fast flag clear mode may or may not be useful in a particular application to track valid data. 0 Conversion results are placed in the corresponding result register up to the selected sequence length. 1 Conversion results are placed in consecutive result registers (wrap around at end).
1–0 FRZ[1:0]	**Background Debug Freeze Enable** — When debugging an application, it is useful in many cases to have the ATD pause when a breakpoint (Freeze Mode) is encountered. These 2 bits determine how the ATD will respond to a breakpoint as shown in Table 8-9. Leakage onto the storage node and comparator reference capacitors may compromise the accuracy of an immediately frozen conversion depending on the length of the freeze period.

Table 8-8. Conversion Sequence Length Coding

S8C	S4C	S2C	S1C	Number of Conversions per Sequence
0	0	0	0	8
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	X	X	X	8

Table 8-9. ATD Behavior in Freeze Mode (Breakpoint)

FRZ1	FRZ0	Behavior in Freeze Mode
0	0	Continue conversion
0	1	Reserved
1	0	Finish current conversion, then freeze
1	1	Freeze Immediately

9.7.1.3 Post-Transfer Software Response

Transmission or reception of a byte will set the data transferring bit (TCF) to 1, which indicates one byte communication is finished. The IIC bus interrupt bit (IBIF) is set also; an interrupt will be generated if the interrupt function is enabled during initialization by setting the IBIE bit. Software must clear the IBIF bit in the interrupt routine first. The TCF bit will be cleared by reading from the IIC bus data I/O register (IBDR) in receive mode or writing to IBDR in transmit mode.

Software may service the IIC I/O in the main program by monitoring the IBIF bit if the interrupt function is disabled. Note that polling should monitor the IBIF bit rather than the TCF bit because their operation is different when arbitration is lost.

Note that when an interrupt occurs at the end of the address cycle the master will always be in transmit mode, i.e. the address is transmitted. If master receive mode is required, indicated by R/W bit in IBDR, then the Tx/Rx bit should be toggled at this stage.

During slave mode address cycles (IAAS=1), the SRW bit in the status register is read to determine the direction of the subsequent transfer and the Tx/Rx bit is programmed accordingly. For slave mode data cycles (IAAS=0) the SRW bit is not valid, the Tx/Rx bit in the control register should be read to determine the direction of the current transfer.

The following is an example of a software response by a 'master transmitter' in the interrupt routine.

```
ISR         BCLR    IBSR,#$02           ;CLEAR THE IBIF FLAG
            BRCLR   IBCR,#$20,SLAVE     ;BRANCH IF IN SLAVE MODE
            BRCLR   IBCR,#$10,RECEIVE   ;BRANCH IF IN RECEIVE MODE
            BRSET   IBSR,#$01,END       ;IF NO ACK, END OF TRANSMISSION
TRANSMIT    MOVB    DATABUF,IBDR        ;TRANSMIT NEXT BYTE OF DATA
```

9.7.1.4 Generation of STOP

A data transfer ends with a STOP signal generated by the 'master' device. A master transmitter can simply generate a STOP signal after all the data has been transmitted. The following is an example showing how a stop condition is generated by a master transmitter.

```
MASTX       TST     TXCNT               ;GET VALUE FROM THE TRANSMITING COUNTER
            BEQ     END                 ;END IF NO MORE DATA
            BRSET   IBSR,#$01,END       ;END IF NO ACK
            MOVB    DATABUF,IBDR        ;TRANSMIT NEXT BYTE OF DATA
            DEC     TXCNT               ;DECREASE THE TXCNT
            BRA     EMASTX              ;EXIT
END         BCLR    IBCR,#$20           ;GENERATE A STOP CONDITION
EMASTX      RTI                         ;RETURN FROM INTERRUPT
```

If a master receiver wants to terminate a data transfer, it must inform the slave transmitter by not acknowledging the last byte of data which can be done by setting the transmit acknowledge bit (TXAK) before reading the 2nd last byte of data. Before reading the last byte of data, a STOP signal must be generated first. The following is an example showing how a STOP signal is generated by a master receiver.

by writing another byte to the Transmitter buffer (SCIDRH/SCIDRL), while the shift register is still shifting out the first byte.

To initiate an SCI transmission:

1. Configure the SCI:
   a) Select a baud rate. Write this value to the SCI baud registers (SCIBDH/L) to begin the baud rate generator. Remember that the baud rate generator is disabled when the baud rate is zero. Writing to the SCIBDH has no effect without also writing to SCIBDL.
   b) Write to SCICR1 to configure word length, parity, and other configuration bits (LOOPS,RSRC,M,WAKE,ILT,PE,PT).
   c) Enable the transmitter, interrupts, receive, and wake up as required, by writing to the SCICR2 register bits (TIE,TCIE,RIE,ILIE,TE,RE,RWU,SBK). A preamble or idle character will now be shifted out of the transmitter shift register.
2. Transmit Procedure for Each Byte:
   a. Poll the TDRE flag by reading the SCISR1 or responding to the TDRE interrupt. Keep in mind that the TDRE bit resets to one.
   d) If the TDRE flag is set, write the data to be transmitted to SCIDRH/L, where the ninth bit is written to the T8 bit in SCIDRH if the SCI is in 9-bit data format. A new transmission will not result until the TDRE flag has been cleared.
3. Repeat step 2 for each subsequent transmission.

NOTE

The TDRE flag is set when the shift register is loaded with the next data to be transmitted from SCIDRH/L, which happens, generally speaking, a little over half-way through the stop bit of the previous frame. Specifically, this transfer occurs 9/16ths of a bit time AFTER the start of the stop bit of the previous frame.

Writing the TE bit from 0 to a 1 automatically loads the transmit shift register with a preamble of 10 logic 1s (if M = 0) or 11 logic 1s (if M = 1). After the preamble shifts out, control logic transfers the data from the SCI data register into the transmit shift register. A logic 0 start bit automatically goes into the least significant bit position of the transmit shift register. A logic 1 stop bit goes into the most significant bit position.

Hardware supports odd or even parity. When parity is enabled, the most significant bit (msb) of the data character is the parity bit.

The transmit data register empty flag, TDRE, in SCI status register 1 (SCISR1) becomes set when the SCI data register transfers a byte to the transmit shift register. The TDRE flag indicates that the SCI data register can accept new data from the internal data bus. If the transmit interrupt enable bit, TIE, in SCI control register 2 (SCICR2) is also set, the TDRE flag generates a transmitter interrupt request.

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

With the misaligned character shown in Figure 11-20, the receiver counts 167 RTr cycles at the point when the count of the transmitting device is 10 bit times x 16 RTt cycles = 160 RTt cycles.

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

((167 – 160) / 167) X 100 = 4.19%

11.4.4.5.2 Fast Data Tolerance

Figure 11-21 shows how much a fast received frame can be misaligned. The fast stop bit ends at RT10 instead of RT16 but is still sampled at RT8, RT9, and RT10.

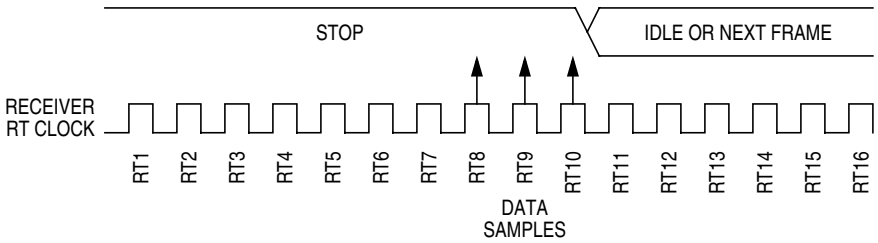


Figure 11-21. Fast Data

For an 8-bit data character, it takes the receiver 9 bit times x 16 RTr cycles + 10 RTr cycles = 154 RTr cycles to finish data sampling of the stop bit.

With the misaligned character shown in Figure 11-21, the receiver counts 154 RTr cycles at the point when the count of the transmitting device is 10 bit times x 16 RTt cycles = 160 RTt cycles.

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

((160 – 154) / 160) x 100 = 3.75%

For a 9-bit data character, it takes the receiver 10 bit times x 16 RTr cycles + 10 RTr cycles = 170 RTr cycles to finish data sampling of the stop bit.

With the misaligned character shown in Figure 11-21, the receiver counts 170 RTr cycles at the point when the count of the transmitting device is 11 bit times x 16 RTt cycles = 176 RTt cycles.

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

((176 – 170) / 176) x 100 = 3.40%

11.4.4.6 Receiver Wakeup

To enable the SCI to ignore transmissions intended only for other receivers in multiple-receiver systems, the receiver can be put into a standby state. Setting the receiver wakeup bit, RWU, in SCI control register 2 (SCICR2) puts the receiver into standby state during which receiver interrupts are disabled.The SCI will still load the receive data into the SCIDRH/L registers, but it will not set the RDRF flag.

Read: Anytime

Write: Anytime

Table 14-3. TIOS Field Descriptions

Field	Description
7:0 IOS[7:0]	**Input Capture or Output Compare Channel Configuration** 0 The corresponding channel acts as an input capture. 1 The corresponding channel acts as an output compare.

14.3.2.2 Timer Compare Force Register (CFORC)

Module Base + 0x0001

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W	FOC7	FOC6	FOC5	FOC4	FOC3	FOC2	FOC1	FOC0
Reset	0	0	0	0	0	0	0	0

Figure 14-7. Timer Compare Force Register (CFORC)

Read: Anytime but will always return 0x0000 (1 state is transient)

Write: Anytime

Table 14-4. CFORC Field Descriptions

Field	Description
7:0 FOC[7:0]	**Force Output Compare Action for Channel 7:0** — A write to this register with the corresponding data bit(s) set causes the action which is programmed for output compare “x” to occur immediately. The action taken is the same as if a successful comparison had just taken place with the TCx register except the interrupt flag does not get set. **Note:** A successful channel 7 output compare overrides any channel 6:0 compares. If forced output compare on any channel occurs at the same time as the successful output compare then forced output compare action will take precedence and interrupt flag won’t get set.

14.3.2.3 Output Compare 7 Mask Register (OC7M)

Module Base + 0x0002

	7	6	5	4	3	2	1	0
R W	OC7M7	OC7M6	OC7M5	OC7M4	OC7M3	OC7M2	OC7M1	OC7M0
Reset	0	0	0	0	0	0	0	0

Figure 14-8. Output Compare 7 Mask Register (OC7M)

Read: Anytime

Write: Anytime

14.3.2.17 Pulse Accumulators Count Registers (PACNT)

Module Base + 0x0022

	15	14	13	12	11	10	9	0
R W	PACNT15	PACNT14	PACNT13	PACNT12	PACNT11	PACNT10	PACNT9	PACNT8
Reset	0	0	0	0	0	0	0	0

Figure 14-26. Pulse Accumulator Count Register High (PACNTH)

Module Base + 0x0023

	7	6	5	4	3	2	1	0
R W	PACNT7	PACNT6	PACNT5	PACNT4	PACNT3	PACNT2	PACNT1	PACNT0
Reset	0	0	0	0	0	0	0	0

Figure 14-27. Pulse Accumulator Count Register Low (PACNTL)

Read: Anytime

Write: Anytime

These registers contain the number of active input edges on its input pin since the last reset.

When PACNT overflows from 0xFFFF to 0x0000, the Interrupt flag PAOVF in PAFLG (0x0021) is set.

Full count register access should take place in one clock cycle. A separate read/write for high byte and low byte will give a different result than accessing them as a word.

NOTE

Reading the pulse accumulator counter registers immediately after an active edge on the pulse accumulator input pin may miss the last count because the input has to be synchronized with the bus clock first.

14.4 Functional Description

This section provides a complete functional description of the timer TIM16B8CV1 block. Please refer to the detailed timer block diagram in Figure 14-28 as necessary.

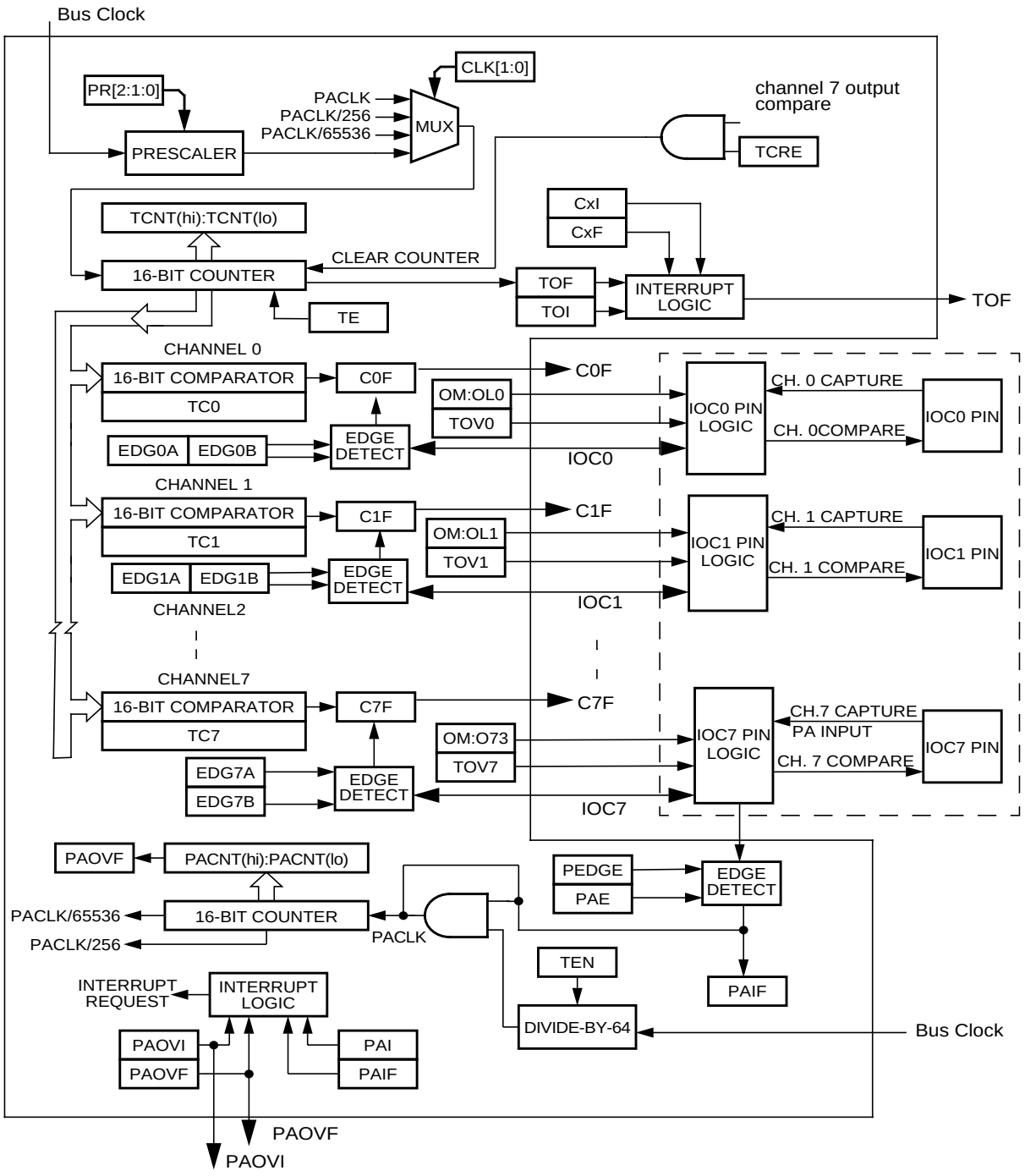


Figure 14-28. Detailed Timer Block Diagram

14.4.1 Prescaler

The prescaler divides the bus clock by 1,2,4,8,16,32,64 or 128. The prescaler select bits, PR[2:0], select the prescaler divisor. PR[2:0] are in timer system control register 2 (TSCR2).

The commands are described as follows:

- ACK_ENABLE — enables the hardware handshake protocol. The target will issue the ACK pulse when a CPU command is executed by the CPU. The ACK_ENABLE command itself also has the ACK pulse as a response.
- ACK_DISABLE — disables the ACK pulse protocol. In this case, the host needs to use the worst case delay time at the appropriate places in the protocol.

The default state of the BDM after reset is hardware handshake protocol disabled.

All the read commands will ACK (if enabled) when the data bus cycle has completed and the data is then ready for reading out by the BKGD serial pin. All the write commands will ACK (if enabled) after the data has been received by the BDM through the BKGD serial pin and when the data bus cycle is complete. See Section 16.4.3, “BDM Hardware Commands,” and Section 16.4.4, “Standard BDM Firmware Commands,” for more information on the BDM commands.

The ACK_ENABLE sends an ACK pulse when the command has been completed. This feature could be used by the host to evaluate if the target supports the hardware handshake protocol. If an ACK pulse is issued in response to this command, the host knows that the target supports the hardware handshake protocol. If the target does not support the hardware handshake protocol the ACK pulse is not issued. In this case, the ACK_ENABLE command is ignored by the target because it is not recognized as a valid command.

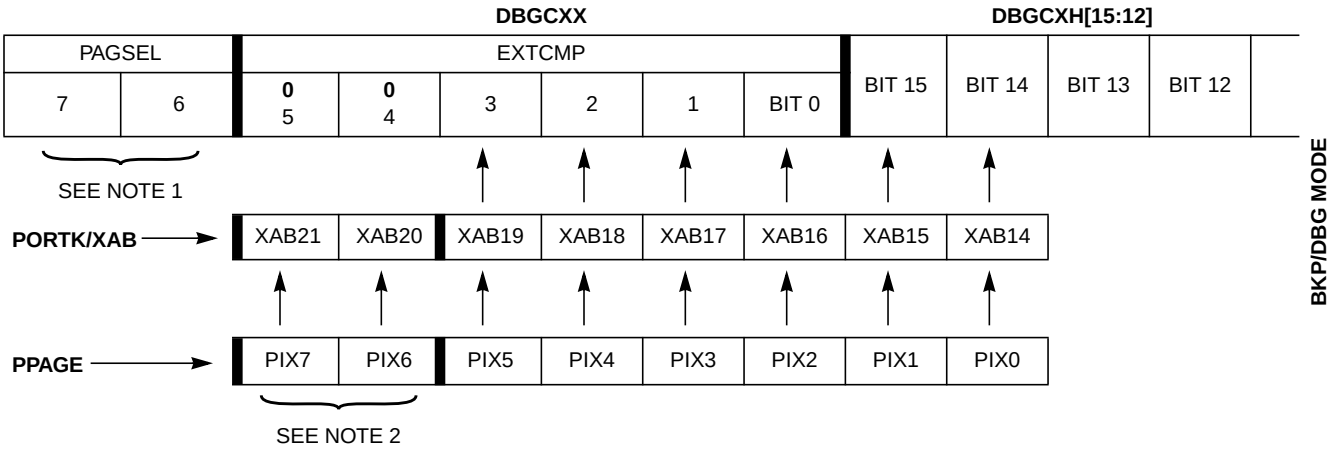
The BACKGROUND command will issue an ACK pulse when the CPU changes from normal to background mode. The ACK pulse related to this command could be aborted using the SYNC command.

The GO command will issue an ACK pulse when the CPU exits from background mode. The ACK pulse related to this command could be aborted using the SYNC command.

The GO_UNTIL command is equivalent to a GO command with exception that the ACK pulse, in this case, is issued when the CPU enters into background mode. This command is an alternative to the GO command and should be used when the host wants to trace if a breakpoint match occurs and causes the CPU to enter active background mode. Note that the ACK is issued whenever the CPU enters BDM, which could be caused by a breakpoint match or by a BGND instruction being executed. The ACK pulse related to this command could be aborted using the SYNC command.

The TRACE1 command has the related ACK pulse issued when the CPU enters background active mode after one instruction of the application program is executed. The ACK pulse related to this command could be aborted using the SYNC command.

The TAGGO command will not issue an ACK pulse because this would interfere with the tagging function shared on the same pin.



NOTES:
1. In BKP and DBG mode, PAGSEL selects the type of paging as shown in Table 17-11.
2. Current HCS12 implementations are limited to six PPAGE bits, PIX[5:0]. Therefore, EXTCMP[5:4] = 00.

Figure 17-10. Comparator C Extended Comparison in BKP/DBG Mode

17.3.2.6 Debug Comparator C Register (DBGCC)

Module Base + 0x0026
Starting address location affected by INITRG register setting.

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

= Unimplemented or Reserved

Figure 17-11. Debug Comparator C Register High (DBGCCH)

Module Base + 0x0027
Starting address location affected by INITRG register setting.

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

= Unimplemented or Reserved

Figure 17-12. Debug Comparator C Register Low (DBGCCL)

DBGC2 being logic 1 or logic 0, respectively. BDM requests have a higher priority than SWI requests. No data breakpoints are allowed in this mode.

TAGAB in DBGC2 selects whether the breakpoint mode is forced or tagged. The BKxMBH:L bits in DBGC3 select whether or not the breakpoint is matched exactly or is a range breakpoint. They also select whether the address is matched on the high byte, low byte, both bytes, and/or memory expansion. The RWx and RWxEN bits in DBGC3 select whether the type of bus cycle to match is a read, write, or read/write when performing forced breakpoints.

17.4.1.2 Full Breakpoint Mode

Full breakpoint mode requires a match on address and data for a breakpoint to occur. Upon a successful match, the system will enter background debug mode or initiate a software interrupt based upon the state of BDM in DBGC2 being logic 1 or logic 0, respectively. BDM requests have a higher priority than SWI requests. R/W matches are also allowed in this mode.

TAGAB in DBGC2 selects whether the breakpoint mode is forced or tagged. When TAGAB is set in DBGC2, only addresses are compared and data is ignored. The BKAMBH:L bits in DBGC3 select whether or not the breakpoint is matched exactly, is a range breakpoint, or is in page space. The BKBMBH:L bits in DBGC3 select whether the data is matched on the high byte, low byte, or both bytes. RWA and RWAEN bits in DBGC2 select whether the type of bus cycle to match is a read or a write when performing forced breakpoints. RWB and RWBEN bits in DBGC2 are not used in full breakpoint mode.

NOTE

The full trigger mode is designed to be used for either a word access or a byte access, but not both at the same time. Confusing trigger operation (seemingly false triggers or no trigger) can occur if the trigger address occurs in the user program as both byte and word accesses.

17.4.1.3 Breakpoint Priority

Breakpoint operation is first determined by the state of the BDM module. If the BDM module is already active, meaning the CPU is executing out of BDM firmware, breakpoints are not allowed. In addition, while executing a BDM TRACE command, tagging into BDM is not allowed. If BDM is not active, the breakpoint will give priority to BDM requests over SWI requests. This condition applies to both forced and tagged breakpoints.

In all cases, BDM related breakpoints will have priority over those generated by the Breakpoint sub-block. This priority includes breakpoints enabled by the $\overline{\text{TAGLO}}$ and $\overline{\text{TAGHI}}$ external pins of the system that interface with the BDM directly and whose signal information passes through and is used by the breakpoint sub-block.