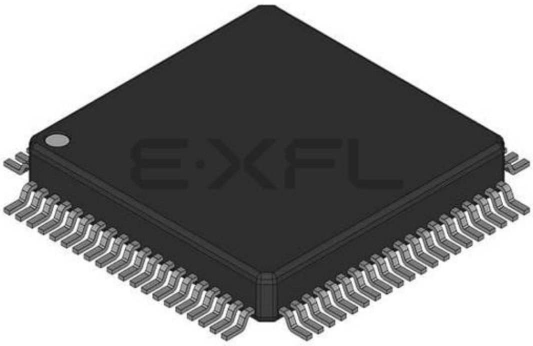


Welcome to E-XFL.COM

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	HCS12X
Core Size	16-Bit
Speed	80MHz
Connectivity	CANbus, EBI/EMI, I²C, IrDA, LINbus, SCI, SPI
Peripherals	LVD, POR, PWM, WDT
Number of I/O	59
Program Memory Size	256KB (256K x 8)
Program Memory Type	FLASH
EEPROM Size	4K x 8
RAM Size	14K x 8
Voltage - Supply (Vcc/Vdd)	2.35V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	External
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	80-QFP
Supplier Device Package	80-QFP (14x14)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mc9s12xd256vaa

1.2.3.32 PH0 / KWH0 / MISO1 — Port H I/O Pin 0

PH0 is a general-purpose input or output pin. It can be configured to generate an interrupt causing the MCU to exit stop or wait mode. It can be configured as master input (during master mode) or slave output (during slave mode) pin MISO of the serial peripheral interface 1 (SPI1).

1.2.3.33 PJ7 / KWJ7 / TXCAN4 / SCL0 / TXCAN0— PORT J I/O Pin 7

PJ7 is a general-purpose input or output pin. It can be configured to generate an interrupt causing the MCU to exit stop or wait mode. It can be configured as the transmit pin TXCAN for the scalable controller area network controller 0 or 4 (CAN0 or CAN4) or as the serial clock pin SCL of the IIC0 module.

1.2.3.34 PJ6 / KWJ6 / RXCAN4 / SDA0 / RXCAN0 — PORT J I/O Pin 6

PJ6 is a general-purpose input or output pin. It can be configured to generate an interrupt causing the MCU to exit stop or wait mode. It can be configured as the receive pin RXCAN for the scalable controller area network controller 0 or 4 (CAN0 or CAN4) or as the serial data pin SDA of the IIC0 module.

1.2.3.35 PJ5 / KWJ5 / SCL1 / $\overline{CS2}$ — PORT J I/O Pin 5

PJ5 is a general-purpose input or output pin. It can be configured to generate an interrupt causing the MCU to exit stop or wait mode. It can be configured as the serial clock pin SCL of the IIC1 module. It can be configured to provide a chip-select output.

1.2.3.36 PJ4 / KWJ4 / SDA1 / $\overline{CS0}$ — PORT J I/O Pin 4

PJ4 is a general-purpose input or output pin. It can be configured to generate an interrupt causing the MCU to exit stop or wait mode. It can be configured as the serial data pin SDA of the IIC1 module. It can be configured to provide a chip-select output.

1.2.3.37 PJ2 / KWJ2 / $\overline{CS1}$ — PORT J I/O Pin 2

PJ2 is a general-purpose input or output pins. It can be configured to generate an interrupt causing the MCU to exit stop or wait mode. It can be configured to provide a chip-select output.

1.2.3.38 PJ1 / KWJ1 / TXD2 — PORT J I/O Pin 1

PJ1 is a general-purpose input or output pin. It can be configured to generate an interrupt causing the MCU to exit stop or wait mode. It can be configured as the transmit pin TXD of the serial communication interface 2 (SCI2).

1.2.3.39 PJ0 / KWJ0 / RXD2 / $\overline{CS3}$ — PORT J I/O Pin 0

PJ0 is a general-purpose input or output pin. It can be configured to generate an interrupt causing the MCU to exit stop or wait mode. It can be configured as the receive pin RXD of the serial communication interface 2 (SCI2).It can be configured to provide a chip-select output.

Table 1-8. MC9S12XD Family Power and Ground Connection Summary

Mnemonic	Pin Number			Nominal Voltage	Description
	144-Pin LQFP	112-Pin LQFP	80-Pin QFP		
$V_{DD1, 2}$	15, 87	13, 65	9, 49	2.5 V	Internal power and ground generated by internal regulator
$V_{SS1, 2}$	16, 88	14, 66	10, 50	0V	
V_{DDR1}	53	41	29	5.0 V	External power and ground, supply to pin drivers and internal voltage regulator
V_{SSR1}	52	40	28	0 V	
V_{DDX1}	139	107	77	5.0 V	External power and ground, supply to pin drivers
V_{SSX1}	138	106	76	0 V	
V_{DDX2}	26	N.A.	N.A.	5.0 V	External power and ground, supply to pin drivers
V_{SSX2}	27	N.A.	N.A.	0 V	
V_{DDR2}	82	N.A.	N.A.	5.0 V	External power and ground, supply to pin drivers
V_{SSR2}	81	N.A.	N.A.	0 V	
V_{DDA}	107	83	59	5.0 V	Operating voltage and ground for the analog-to-digital converters and the reference for the internal voltage regulator, allows the supply voltage to the A/D to be bypassed independently.
V_{SSA}	110	86	62	0 V	
V_{RL}	109	85	61	0 V	Reference voltages for the analog-to-digital converter.
V_{RH}	108	84	60	5.0 V	
V_{DDPLL}	55	43	31	2.5 V	Provides operating voltage and ground for the phased-locked loop. This allows the supply voltage to the PLL to be bypassed independently. Internal power and ground generated by internal regulator.
V_{SSPLL}	57	45	33	0 V	

2.3.1 Module Memory Map

Table 2-1 gives an overview on all CRG registers.

Table 2-1. CRG Memory Map

Address Offset	Use	Access
0x_00	CRG Synthesizer Register (SYNR)	R/W
0x_01	CRG Reference Divider Register (REFDV)	R/W
0x_02	CRG Test Flags Register (CTFLG)[1]	R/W
0x_03	CRG Flags Register (CRGFLG)	R/W
0x_04	CRG Interrupt Enable Register (CRGINT)	R/W
0x_05	CRG Clock Select Register (CLKSEL)	R/W
0x_06	CRG PLL Control Register (PLLCTL)	R/W
0x_07	CRG RTI Control Register (RTICTL)	R/W
0x_08	CRG COP Control Register (COPCTL)	R/W
0x_09	CRG Force and Bypass Test Register (FORBYP)[2]	R/W
0x_0A	CRG Test Control Register (CTCTL)[3]	R/W
0x_0B	CRG COP Arm/Timer Reset (ARMCOP)	R/W

1 CTFLG is intended for factory test purposes only.
2 FORBYP is intended for factory test purposes only.
3 CTCTL is intended for factory test purposes only.

NOTE

Register Address = Base Address + Address Offset, where the Base Address is defined at the MCU level and the Address Offset is defined at the module level.

BRK Break BRK

Operation

Put XGATE into Debug Mode (see Section 6.6.2, “Entering Debug Mode”)and signals a Software breakpoint to the S12X_DBG module (see section 4.9 of the **S12X_DBG Section**).

NOTE

It is not possible to single step over a BRK instruction. This instruction does not advance the program counter.

CCR Effects

N	Z	V	C
—	—	—	—

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code																Cycles
BRK	INH	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PAff

7.3.2.5 Timer Count Register (TCNT)

	15	14	13	12	11	10	9	8
R W	TCNT15	TCNT14	TCNT13	TCNT12	TCNT11	TCNT10	TCNT9	TCNT8
Reset	0	0	0	0	0	0	0	0

Figure 7-7. Timer Count Register High (TCNT)

	7	6	5	4	3	2	1	0
R W	TCNT7	TCNT6	TCNT5	TCNT4	TCNT3	TCNT2	TCNT1	TCNT0
Reset	0	0	0	0	0	0	0	0

Figure 7-8. Timer Count Register Low (TCNT)

Read: Anytime

Write: Has no meaning or effect

All bits reset to zero.

Table 7-6. TCNT Field Descriptions

Field	Description
15:0 TCNT[15:0]	**Timer Counter Bits** — The 16-bit main timer is an up counter. A read to this register will return the current value of the counter. Access to the counter register will take place in one clock cycle. **Note:** A separate read/write for high byte and low byte in test mode will give a different result than accessing them as a word. The period of the first count after a write to the TCNT registers may be a different size because the write is not synchronized with the prescaler clock.

7.3.2.31 Timer Input Capture Holding Registers 0–3 (TCxH)

	15	14	13	12	11	10	9	8
R	TC15	TC14	TC13	TC12	TC11	TC10	TC9	TC8
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 7-57. Timer Input Capture Holding Register 0 High (TC0H)

	7	6	5	4	3	2	1	0
R	TC7	TC6	TC5	TC4	TC3	TC2	TC1	TC0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 7-58. Timer Input Capture Holding Register 0 Low (TC0H)

	15	14	13	12	11	10	9	8
R	TC15	TC14	TC13	TC12	TC11	TC10	TC9	TC8
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 7-59. Timer Input Capture Holding Register 1 High (TC1H)

	7	6	5	4	3	2	1	0
R	TC7	TC6	TC5	TC4	TC3	TC2	TC1	TC0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 7-60. Timer Input Capture Holding Register 1 Low (TC1H)

	15	14	13	12	11	10	9	8
R	TC15	TC14	TC13	TC12	TC11	TC10	TC9	TC8
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 7-61. Timer Input Capture Holding Register 2 High (TC2H)

	7	6	5	4	3	2	1	0
R	TC7	TC6	TC5	TC4	TC3	TC2	TC1	TC0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 7-62. Timer Input Capture Holding Register 2 Low (TC2H)

Table 8-9. PWMSDN Field Descriptions

Field	Description
7 PWMIF	**PWM Interrupt Flag** — Any change from passive to asserted (active) state or from active to passive state will be flagged by setting the PWMIF flag = 1. The flag is cleared by writing a logic 1 to it. Writing a 0 has no effect. 0 No change on PWM7IN input. 1 Change on PWM7IN input
6 PWMIE	**PWM Interrupt Enable** — If interrupt is enabled an interrupt to the CPU is asserted. 0 PWM interrupt is disabled. 1 PWM interrupt is enabled.
5 PWMRSTRT	**PWM Restart** — The PWM can only be restarted if the PWM channel input 7 is de-asserted. After writing a logic 1 to the PWMRSTRT bit (trigger event) the PWM channels start running after the corresponding counter passes next “counter == 0” phase. Also, if the PWM7ENA bit is reset to 0, the PWM do not start before the counter passes $00. The bit is always read as “0”.
4 PWMLVL	**PWM Shutdown Output Level** If active level as defined by the PWM7IN input, gets asserted all enabled PWM channels are immediately driven to the level defined by PWMLVL. 0 PWM outputs are forced to 0 1 Outputs are forced to 1.
2 PWM7IN	**PWM Channel 7 Input Status** — This reflects the current status of the PWM7 pin.
1 PWM7INL	**PWM Shutdown Active Input Level for Channel 7** — If the emergency shutdown feature is enabled (PWM7ENA = 1), this bit determines the active level of the PWM7channel. 0 Active level is low 1 Active level is high
0 PWM7ENA	**PWM Emergency Shutdown Enable** — If this bit is logic 1, the pin associated with channel 7 is forced to input and the emergency shutdown feature is enabled. All the other bits in this register are meaningful only if PWM7ENA = 1. 0 PWM emergency feature disabled. 1 PWM emergency feature is enabled.

8.4 Functional Description

8.4.1 PWM Clock Select

There are four available clocks: clock A, clock B, clock SA (scaled A), and clock SB (scaled B). These four clocks are based on the bus clock.

Clock A and B can be software selected to be 1, 1/2, 1/4, 1/8,..., 1/64, 1/128 times the bus clock. Clock SA uses clock A as an input and divides it further with a reloadable counter. Similarly, clock SB uses clock B as an input and divides it further with a reloadable counter. The rates available for clock SA are software selectable to be clock A divided by 2, 4, 6, 8,..., or 512 in increments of divide by 2. Similar rates are available for clock SB. Each PWM channel has the capability of selecting one of two clocks, either the pre-scaled clock (clock A or B) or the scaled clock (clock SA or SB).

The block diagram in Figure 8-18 shows the four different clocks and how the scaled clocks are created.

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.

```
MASR     DEC     RXCNT            ;DECREASE THE RXCNT
         BEQ     ENMASR           ;LAST BYTE TO BE READ
         MOVB    RXCNT,D1         ;CHECK SECOND LAST BYTE
         DEC     D1               ;TO BE READ
         BNE     NXMAR            ;NOT LAST OR SECOND LAST
LAMAR    BSET    IBCR,#$08        ;SECOND LAST, DISABLE ACK
                                  ;TRANSMITTING
         BRA     NXMAR
ENMASR   BCLR    IBCR,#$20        ;LAST ONE, GENERATE ‘STOP’ SIGNAL
NXMAR    MOVB    IBDR,RXBUF       ;READ DATA AND STORE
         RTI
```

9.7.1.5 Generation of Repeated START

At the end of data transfer, if the master continues to want to communicate on the bus, it can generate another START signal followed by another slave address without first generating a STOP signal. A program example is as shown.

```
RESTART  BSET    IBCR,#$04        ;ANOTHER START (RESTART)
         MOVB    CALLING,IBDR     ;TRANSMIT THE CALLING ADDRESS;D0=R/W
```

9.7.1.6 Slave Mode

In the slave interrupt service routine, the module addressed as slave bit (IAAS) should be tested to check if a calling of its own address has just been received. If IAAS is set, software should set the transmit/receive mode select bit (Tx/Rx bit of IBCR) according to the R/W command bit (SRW). Writing to the IBCR clears the IAAS automatically. Note that the only time IAAS is read as set is from the interrupt at the end of the address cycle where an address match occurred, interrupts resulting from subsequent data transfers will have IAAS cleared. A data transfer may now be initiated by writing information to IBDR, for slave transmits, or dummy reading from IBDR, in slave receive mode. The slave will drive SCL low in-between byte transfers, SCL is released when the IBDR is accessed in the required mode.

	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

Figure 10-16. MSCAN Reserved Register

Read: Always read 0x0000 in normal system operation modes
Write: Unimplemented in normal system operation modes

NOTE

Writing to this register when in special modes can alter the MSCAN functionality.

10.3.2.14 MSCAN Miscellaneous Register (CANMISC)

This register provides additional features.

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

= Unimplemented

Figure 10-17. MSCAN Miscellaneous Register (CANMISC)

Read: Anytime
Write: Anytime; write of ‘1’ clears flag; write of ‘0’ ignored

Table 10-19. CANMISC Register Field Descriptions

Field	Description
0 BOHOLD	**Bus-off State Hold Until User Request** — If BORM is set in Section 10.3.2.2, “MSCAN Control Register 1 (CANCTL1), this bit indicates whether the module has entered the bus-off state. Clearing this bit requests the recovery from bus-off. Refer to Section 10.5.2, “Bus-Off Recovery,” for details. 0 Module is not bus-off or recovery has been requested by user in bus-off state 1 Module is bus-off and holds this state until user request

10.3.2.15 MSCAN Receive Error Counter (CANRXERR)

This register reflects the status of the MSCAN receive error counter.

10.4.2.1 Message Transmit Background

Modern application layer software is built upon two fundamental assumptions:

- Any CAN node is able to send out a stream of scheduled messages without releasing the CAN bus between the two messages. Such nodes arbitrate for the CAN bus immediately after sending the previous message and only release the CAN bus in case of lost arbitration.
- The internal message queue within any CAN node is organized such that the highest priority message is sent out first, if more than one message is ready to be sent.

The behavior described in the bullets above cannot be achieved with a single transmit buffer. That buffer must be reloaded immediately after the previous message is sent. This loading process lasts a finite amount of time and must be completed within the inter-frame sequence (IFS) to be able to send an uninterrupted stream of messages. Even if this is feasible for limited CAN bus speeds, it requires that the CPU reacts with short latencies to the transmit interrupt.

A double buffer scheme de-couples the reloading of the transmit buffer from the actual message sending and, therefore, reduces the reactiveness requirements of the CPU. Problems can arise if the sending of a message is finished while the CPU re-loads the second buffer. No buffer would then be ready for transmission, and the CAN bus would be released.

At least three transmit buffers are required to meet the first of the above requirements under all circumstances. The MSCAN has three transmit buffers.

The second requirement calls for some sort of internal prioritization which the MSCAN implements with the “local priority” concept described in Section 10.4.2.2, “Transmit Structures.”

10.4.2.2 Transmit Structures

The MSCAN triple transmit buffer scheme optimizes real-time performance by allowing multiple messages to be set up in advance. The three buffers are arranged as shown in Figure 10-39.

All three buffers have a 13-byte data structure similar to the outline of the receive buffers (see Section 10.3.3, “Programmer’s Model of Message Storage”). An additional Section 10.3.3.4, “Transmit Buffer Priority Register (TBPR) contains an 8-bit local priority field (PRIO) (see Section 10.3.3.4, “Transmit Buffer Priority Register (TBPR)”). The remaining two bytes are used for time stamping of a message, if required (see Section 10.3.3.5, “Time Stamp Register (TSRH–TSRL)”).

To transmit a message, the CPU must identify an available transmit buffer, which is indicated by a set transmitter buffer empty (TXEx) flag (see Section 10.3.2.7, “MSCAN Transmitter Flag Register (CANTFLG)”). If a transmit buffer is available, the CPU must set a pointer to this buffer by writing to the CANTBSEL register (see Section 10.3.2.11, “MSCAN Transmit Buffer Selection Register (CANTBSEL)”). This makes the respective buffer accessible within the CANTXFG address space (see Section 10.3.3, “Programmer’s Model of Message Storage”). The algorithmic feature associated with the CANTBSEL register simplifies the transmit buffer selection. In addition, this scheme makes the handler software simpler because only one address area is applicable for the transmit process, and the required address space is minimized.

The CPU then stores the identifier, the control bits, and the data content into one of the transmit buffers. Finally, the buffer is flagged as ready for transmission by clearing the associated TXE flag.

12.4.3 Transmission Formats

During an SPI transmission, data is transmitted (shifted out serially) and received (shifted in serially) simultaneously. The serial clock (SCK) synchronizes shifting and sampling of the information on the two serial data lines. A slave select line allows selection of an individual slave SPI device; slave devices that are not selected do not interfere with SPI bus activities. Optionally, on a master SPI device, the slave select line can be used to indicate multiple-master bus contention.

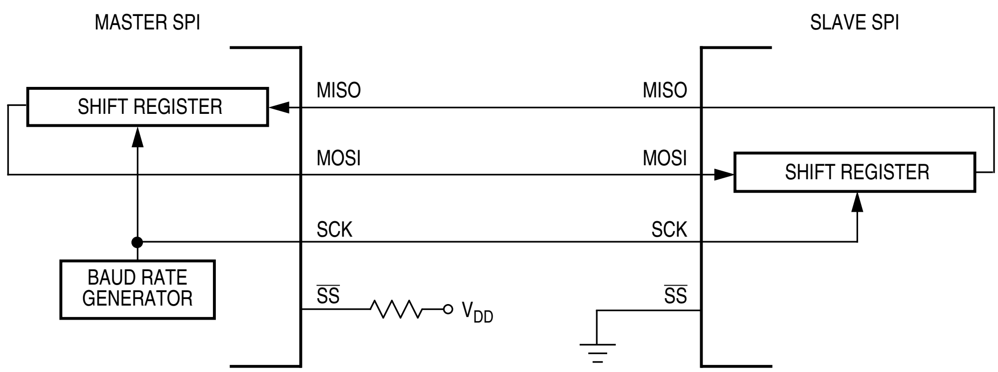


Figure 12-10. Master/Slave Transfer Block Diagram

12.4.3.1 Clock Phase and Polarity Controls

Using two bits in the SPI control register 1, software selects one of four combinations of serial clock phase and polarity.

The CPOL clock polarity control bit specifies an active high or low clock and has no significant effect on the transmission format.

The CPHA clock phase control bit selects one of two fundamentally different transmission formats.

Clock phase and polarity should be identical for the master SPI device and the communicating slave device. In some cases, the phase and polarity are changed between transmissions to allow a master device to communicate with peripheral slaves having different requirements.

12.4.3.2 CPHA = 0 Transfer Format

The first edge on the SCK line is used to clock the first data bit of the slave into the master and the first data bit of the master into the slave. In some peripherals, the first bit of the slave's data is available at the slave's data out pin as soon as the slave is selected. In this format, the first SCK edge is issued a half cycle after $\overline{SS}$ has become low.

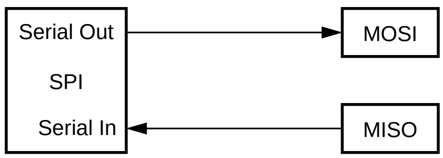
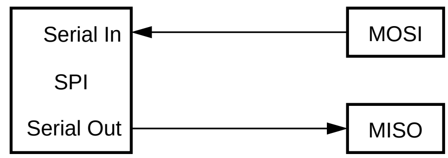
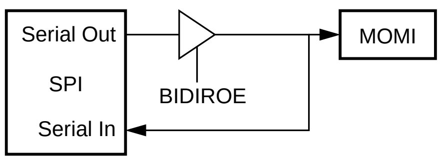
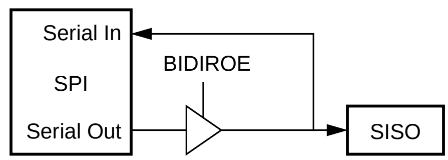
A half SCK cycle later, the second edge appears on the SCK line. When this second edge occurs, the value previously latched from the serial data input pin is shifted into the LSB or MSB of the shift register, depending on LSBFE bit.

After this second edge, the next bit of the SPI master data is transmitted out of the serial data output pin of the master to the serial input pin on the slave. This process continues for a total of 16 edges on the SCK line, with data being latched on odd numbered edges and shifted on even numbered edges.

12.4.5.2 Bidirectional Mode (MOMI or SISO)

The bidirectional mode is selected when the SPC0 bit is set in SPI control register 2 (see Table 12-8). In this mode, the SPI uses only one serial data pin for the interface with external device(s). The MSTR bit decides which pin to use. The MOSI pin becomes the serial data I/O (MOMI) pin for the master mode, and the MISO pin becomes serial data I/O (SISO) pin for the slave mode. The MISO pin in master mode and MOSI pin in slave mode are not used by the SPI.

Table 12-8. Normal Mode and Bidirectional Mode

When SPE = 1	Master Mode MSTR = 1	Slave Mode MSTR = 0
Normal Mode SPC0 = 0		
Bidirectional Mode SPC0 = 1		

The direction of each serial I/O pin depends on the BIDIROE bit. If the pin is configured as an output, serial data from the shift register is driven out on the pin. The same pin is also the serial input to the shift register.

- The SCK is output for the master mode and input for the slave mode.
- The $\overline{SS}$ is the input or output for the master mode, and it is always the input for the slave mode.
- The bidirectional mode does not affect SCK and $\overline{SS}$ functions.

NOTE

In bidirectional master mode, with mode fault enabled, both data pins MISO and MOSI can be occupied by the SPI, though MOSI is normally used for transmissions in bidirectional mode and MISO is not used by the SPI. If a mode fault occurs, the SPI is automatically switched to slave mode. In this case MISO becomes occupied by the SPI and MOSI is not used. This must be considered, if the MISO pin is used for another purpose.

Table 19-3. DBGC1 Field Descriptions

Field	Description
7 ARM	**Arm Bit** — The ARM bit controls whether the DBG module is armed. This bit can be set and cleared by user software and is automatically cleared on completion of a tracing session, or if a breakpoint is generated with tracing not enabled. On setting this bit the state sequencer enters State1. When ARM is set, the only bits in the DBG module registers that can be written are ARM and TRIG. 0 Debugger disarmed 1 Debugger armed
6 TRIG	**Immediate Trigger Request Bit** — This bit when written to 1 requests an immediate trigger independent of comparator or external tag signal status. When tracing is complete a forced breakpoint may be generated depending upon DBGBRK and BDM bit settings. This bit always reads back a “0”. Writing a “0” to this bit has no effect. If both TSOURCE bits are clear no tracing is carried out. If tracing has already commenced using BEGIN- or mid-trigger alignment, it continues until the end of the tracing session as defined by the TALIGN bit settings, thus TRIG has no affect. In secure mode tracing is disabled and writing to this bit has no effect. 0 Do not trigger until the state sequencer enters the final state. 1 Enter final state immediately and issue forced breakpoint request when trace buffer is full.
5 XGSBPE	**XGATE S/W Breakpoint Enable** — The XGSBPE bit controls whether an XGATE S/W breakpoint request is passed to the CPU. The XGATE S/W breakpoint request is handled by the DBG module, which can request an CPU breakpoint depending on the state of this bit. 0 XGATE S/W breakpoint request is disabled 1 XGATE S/W breakpoint request is enabled
4 BDM	**Background Debug Mode Enable** — This bit determines if a CPU breakpoint causes the system to enter background debug mode (BDM) or initiate a software interrupt (SWI). It has no affect on DBG functionality. This bit must be set if the BDM is enabled by the ENBDM bit in the BDM module to map breakpoints to BDM and must be cleared if the BDM module is disabled to map breakpoints to SWI. 0 Go to software interrupt on a breakpoint 1 Go to BDM on a breakpoint.
3–2 DBGBRK	**DBG Breakpoint Enable Bits** — The DBGBRK bits control whether the debugger will request a breakpoint to either CPU, XGATE or both upon reaching the state sequencer final state. If tracing is enabled, the breakpoint is generated on completion of the tracing session. If tracing is not enabled, the breakpoint is generated immediately. Please refer to Section 19.4.7, “Breakpoints” for further details. XGATE generated breakpoints are independent of the DBGBRK bits. XGATE generates a forced breakpoint to the CPU only. See Table 19-4.
1–0 COMRV	**Comparator Register Visibility Bits** — These bits determine which bank of comparator register is visible in the 8-byte window of the DBG module address map, located between 0x0028 to 0x002F. Furthermore these bits determine which state control register is visible at the address 0x0027. See Table 19-5.

Table 19-4. DBGBRK Encoding

DBGBRK	Resource Halted by Breakpoint
00	No breakpoint generated
01	XGATE breakpoint generated
10	CPU breakpoint generated
11	Breakpoints generated for CPU and XGATE

Table 19-5. COMRV Encoding

COMRV	Visible Comparator	Visible State Control Register
00	Comparator A	DBGSCR1

Table 24-4. PORTA Field Descriptions

Field	Description
7–0 PA[7:0]	Port A — Port A pins 7–0 can be used as general purpose I/O. If the data direction bits of the associated I/O pins are set to logic level "1", a read returns the value of the port register, otherwise the buffered pin input state is read.

24.0.5.2 Port B Data Register (PORTB)

	7	6	5	4	3	2	1	0
R W	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
Reset	0	0	0	0	0	0	0	0

Figure 24-4. Port B Data Register (PORTB)

Read: Anytime.

Write: Anytime.

Table 24-5. PORTB Field Descriptions

Field	Description
7–0 PB[7:0]	**Port B** — Port B pins 7–0 can be used as general purpose I/O. If the data direction bits of the associated I/O pins are set to logic level "1", a read returns the value of the port register, otherwise the buffered pin input state is read.

24.0.5.3 Port A Data Direction Register (DDRA)

	7	6	5	4	3	2	1	0
R W	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0
Reset	0	0	0	0	0	0	0	0

Figure 24-5. Port A Data Direction Register (DDRA)

Read: Anytime.

Write: Anytime.

Table 24-6. DDRA Field Descriptions

Field	Description
7–0 DDRA[7:0]	Data Direction Port A — This register controls the data direction for port A. DDRA determines whether each pin is an input or output. A logic level "1" causes the associated port pin to be an output and a logic level "0" causes the associated pin to be a high-impedance input. 0 Associated pin is configured as input. 1 Associated pin is configured as output. **Note:** Due to internal synchronization circuits, it can take up to 2 bus clock cycles until the correct value is read on PORTA after changing the DDRA register.

24.0.5.9 ECLK Control Register (ECLKCTL)

	7	6	5	4	3	2	1	0	
R	NECLK	NCLKX2	0	0	0	0	EDIV1	EDIV0	
W									
Reset[1]	Mode Dependent	1	0	0	0	0	0	0	Mode
SS	0	1	0	0	0	0	0	0	Special Single-Chip
ES	1	1	0	0	0	0	0	0	Emulation Single-Chip
ST	0	1	0	0	0	0	0	0	Special Test
EX	0	1	0	0	0	0	0	0	Emulation Expanded
NS	1	1	0	0	0	0	0	0	Normal Single-Chip
NX	0	1	0	0	0	0	0	0	Normal Expanded

(shaded) = Unimplemented or Reserved

Figure 24-11. ECLK Control Register (ECLKCTL)

1. Reset values in emulation modes are identical to those of the target mode.

Read: Anytime.

Write: Anytime.

The ECLKCTL register is used to control the availability of the free-running clocks and the free-running clock divider.

Table 24-12. ECLKCTL Field Descriptions

Field	Description
7 NECLK	**No ECLK** — This bit controls the availability of a free-running clock on the ECLK pin. Clock output is always active in emulation modes and if enabled in all other operating modes. 0 ECLK enabled 1 ECLK disabled
6 NCLKX2	**No ECLKX2** — This bit controls the availability of a free-running clock on the ECLKX2 pin. This clock has a fixed rate of twice the internal bus clock. Clock output is always active in emulation modes and if enabled in all other operating modes. 0 ECLKX2 is enabled 1 ECLKX2 is disabled
1–0 EDIV[1:0]	**Free-Running ECLK Divider** — These bits determine the rate of the free-running clock on the ECLK pin. The usage of the bits is shown in Table 24-13. Divider is always disabled in emulation modes and active as programmed in all other operating modes.

28.4.2.5 Mass Erase Command

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

An example flow to execute the mass erase operation is shown in Figure 28-30. The mass erase command write sequence is as follows:

1. Write to a Flash block address to start the command write sequence for the mass erase command. The address and data written will be ignored. Multiple Flash blocks can be simultaneously mass erased by writing to the same relative address in each Flash block.
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 block 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.

29.4.2.5 Mass Erase Command

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

An example flow to execute the mass erase operation is shown in Figure 29-28. The mass erase command write sequence is as follows:

1. Write to a Flash block 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 block 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.

0x02C0–0x02DF Analog-to-Digital Converter 10-Bit 8-Channel (ATD0) Map (continued)

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x02D6	ATD0DR3H	R	Bit15	14	13	12	11	10	9	Bit8
		W								
0x02D7	ATD0DR3L	R	Bit7	Bit6	0	0	0	0	0	0
		W								
0x02D8	ATD0DR4H	R	Bit15	14	13	12	11	10	9	Bit8
		W								
0x02D9	ATD0DR4L	R	Bit7	Bit6	0	0	0	0	0	0
		W								
0x02DA	ATD0DR5H	R	Bit15	14	13	12	11	10	9	Bit8
		W								
0x02DB	ATD0DR5L	R	Bit7	Bit6	0	0	0	0	0	0
		W								
0x02DC	ATD0DR6H	R	Bit15	14	13	12	11	10	9	Bit8
		W								
0x02DD	ATD0DR6L	R	Bit7	Bit6	0	0	0	0	0	0
		W								
0x02DE	ATD0DR7H	R	Bit15	14	13	12	11	10	9	Bit8
		W								
0x02DF	ATD0DR7L	R	Bit7	Bit6	0	0	0	0	0	0
		W								

0x02E0–0x02EF Reserved

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x02E0–0x02EF	Reserved	R	0	0	0	0	0	0	0	0
		W								

0x02F0–0x02F7 Voltage Regulator (VREG_3V3) Map

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x02F0	VREGHTCL	R	Reserved for Factory Test							
		W								
0x02F1	VREGCTRL	R	0	0	0	0	0	LVDS	LVIE	LVIF
		W								
0x02F2	VREGAPICL	R	APICLK	0	0	0	0	APIFE	APIE	APIF
		W								
0x02F3	VREGAPITR	R	APITR5	APITR4	APITR3	APITR2	APITR1	APITR0	0	0
		W								
0x02F4	VREGAPIRH	R	0	0	0	0	APIR11	APIR10	APIR9	APIR8
		W								
0x02F5	VREGAPIRL	R	APIR7	APIR6	APIR5	APIR4	APIR3	APIR2	APIR1	APIR0
		W								
0x02F6	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x02F7	Reserved	R	0	0	0	0	0	0	0	0
		W								