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	32MHz
Connectivity	CANbus, SCI, SPI
Peripherals	LVD, POR, PWM, WDT
Number of I/O	49
Program Memory Size	64KB (64K x 8)
Program Memory Type	FLASH
EEPROM Size	4K x 8
RAM Size	4K x 8
Voltage - Supply (Vcc/Vdd)	1.72V ~ 5.5V
Data Converters	A/D 10x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	64-LQFP
Supplier Device Package	64-LQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc9s12p64vlhr

Chapter 13
128 KByte Flash Module (S12FTMRC128K1V1)

NOTE

Reserved register space shown in Table 1-2 is not allocated to any module. This register space is reserved for future use. Writing to these locations have no effect. Read access to these locations returns zero.

Figure 1-2 shows S12P CPU and BDM local address translation to the global memory map. It indicates also the location of the internal resources in the memory map. Table 1-3. shows the mapping of D-Flash and unpaged P-Flash memory. The whole 256K global memory space is visible through the P-Flash window located in the 64k local memory map located at 0x8000 - 0xBFFF using the PPAGE register.

Table 1-3. MC9S12P -Family mapping for D-Flash and unpaged P-Flash

	Local 64K memory map	Global 256K memory map
D-Flash	0x0400 - 0x13FF	0x0_4400 - 0x0_53FF
P-Flash	0x1400 - 0x27FF(1)	0x3_1400 -0x3_27FF(2)
	0x4000 - 0x7FFF	0x3_4000 - 0x3_7FFF
	0xC000 - 0xFFFF	0x3_C000 - 0x3_FFFF

1. 0x2FFF for MC9S12P64 because of 4K RAM size
2. 0x3_2FFF for MC9S12P64 because of 4K RAM size

Table 1-4. Derivatives

Feature	MC9S12P32	MC9S12P64	MC9S12P96	MC9S12P128
P-Flash size	32KB	64KB	96KB	128KB
PF_LOW PPAGES	0x3_8000 0x0E - 0x0F	0x3_0000 0x0C - 0x0F	0x2_8000 0x0A - 0x0F	0x2_0000 0x08 - 0x0F
RAMSIZE	2KB	4KB	6KB	
RAM_LOW	0x0_3800	0x0_3000	0x0_2800	

1.14 S12CPMU Configuration

The bandgap reference voltage V_{BG} and the output voltage of the temperature sensor V_{HT} can be connected to the ATD channel SPECIAL17 (see Table 9-15.) using the VSEL (Voltage Access Select Bit) in CPMUHTCTL register (see Table 7-13.)

Port	Pin Name	Pin Function & Priority(1)	I/O	Description	Pin Function after Reset
M	PM5	SCK	I/O	Serial Peripheral Interface serial clock pin	GPIO
		GPIO	I/O	General purpose	
	PM4	MOSI	I/O	Serial Peripheral Interface master out/slave in pin	
		GPIO	I/O	General purpose	
	PM3	SS	I/O	Serial Peripheral Interface slave select output in master mode, input in slave mode or master mode.	
		GPIO	I/O	General purpose	
	PM2	MISO	I/O	Serial Peripheral Interface master in/slave out pin	
		GPIO	I/O	General purpose	
	PM1	TXCAN	O	MSCAN transmit pin	
		GPIO	I/O	General purpose	
	PM0	RXCAN	I	MSCAN receive	
		GPIO	I/O	General purpose	
P	PP7	GPIO/KWP7	I/O	General purpose; with interrupt	GPIO
	PP5	PWM5	I/O	Pulse Width Modulator channel 5; emergency shut-down	
		GPIO/KWP5	I/O	General purpose; with interrupt	
	PP[4:0]	PWM[4:0]	O	Pulse Width Modulator channel 4 - 0	
		GPIO/KWP[4:0]	I/O	General purpose; with interrupt	
J	PJ[7:6]	GPIO/KWJ[7:6]	I/O	General purpose; with interrupt	GPIO
	PJ[2:0]	GPIO/KWJ[2:0]	I/O	General purpose; with interrupt	
AD	PAD[9:0]	GPIO	I/O	General purpose	GPIO
		AN[9:0]	I	ATD analog	

1. Signals in brackets denote alternative module routing pins.
2. Function active when $\overline{\text{RESET}}$ asserted.

2.3 Memory Map and Register Definition

This section provides a detailed description of all Port Integration Module registers.

2.3.1 Memory Map

Table 2-2 shows the register map of the Port Integration Module.

Table 2-2. Block Memory Map

Port	Offset or Address	Register	Access	Reset Value	Section/Page
A	0x0000	PORTA—Port A Data Register	R/W	0x00	2.3.3/2-63
B	0x0001	PORTB—Port B Data Register	R/W	0x00	2.3.4/2-63
	0x0002	DDRA—Port A Data Direction Register	R/W	0x00	2.3.5/2-64
	0x0003	DDRB—Port B Data Direction Register	R/W	0x00	2.3.6/2-64

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
0x0004 Reserved	R	0	0	0	0	0	0	0	0
	W								
0x0005 Reserved	R	0	0	0	0	0	0	0	0
	W								
0x0006 Reserved	R	0	0	0	0	0	0	0	0
	W								
0x0007 Reserved	R	0	0	0	0	0	0	0	0
	W								
0x0008 PORTE	R	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0
	W								
0x0009 DDRE	R	DDRE7	DDRE6	DDRE5	DDRE4	DDRE3	DDRE2	0	0
	W								
0x000A 0x000B Non-PIM Address Range	R	Non-PIM Address Range							
	W								
0x000C PUCR	R	0	BKPUE	0	PUPEE	0	0	PUPBE	PUPAE
	W								
0x000D RDRIV	R	0	0	0	RDPE	0	0	RDPB	RDPA
	W								
0x000E–0x001B Non-PIM Address Range	R	Non-PIM Address Range							
	W								
0x001C ECLKCTL	R	NECLK	NCLKX2	DIV16	EDIV4	EDIV3	EDIV2	EDIV1	EDIV0
	W								
0x001D Reserved	R	0	0	0	0	0	0	0	0
	W								
0x001E IRQCR	R	IRQE	IRQEN	0	0	0	0	0	0
	W								
0x001F Reserved	R	0	0	0	0	0	0	0	0
	W								

= Unimplemented or Reserved

This mode is generally used for debugging operation, boot-strapping or security related operations. The active background debug mode is in control of the CPU code execution and the BDM firmware is waiting for serial commands sent through the BKGD pin.

3.4.2 Memory Map Scheme

3.4.2.1 CPU and BDM Memory Map Scheme

The BDM firmware lookup tables and BDM register memory locations share addresses with other modules; however they are not visible in the memory map during user's code execution. The BDM memory resources are enabled only during the READ_BD and WRITE_BD access cycles to distinguish between accesses to the BDM memory area and accesses to the other modules. (Refer to BDM Block Guide for further details).

When the MCU enters active BDM mode, the BDM firmware lookup tables and the BDM registers become visible in the local memory map in the range 0xFF00-0xFFFF (global address 0x3_FF00 - 0x3_FFFF) and the CPU begins execution of firmware commands or the BDM begins execution of hardware commands. The resources which share memory space with the BDM module will not be visible in the memory map during active BDM mode.

Please note that after the MCU enters active BDM mode the BDM firmware lookup tables and the BDM registers will also be visible between addresses 0xBF00 and 0xBFFF if the PPAGE register contains value of 0x0F.

3.4.2.1.1 Expansion of the Local Address Map

Expansion of the CPU Local Address Map

The program page index register in S12PMMC allows accessing up to 256KB of P-Flash in the global memory map by using the four index bits (PPAGE[3:0]) to page 16x16 KB blocks into the program page window located from address 0x8000 to address 0xBFFF in the local CPU memory map.

The page value for the program page window is stored in the PPAGE register. The value of the PPAGE register can be read or written by normal memory accesses as well as by the CALL and RTC instructions (see Section 3.6.1, "CALL and RTC Instructions).

Control registers, vector space and parts of the on-chip memories are located in unpaged portions of the 64KB local CPU address space.

The starting address of an interrupt service routine must be located in unpaged memory unless the user is certain that the PPAGE register will be set to the appropriate value when the service routine is called. However an interrupt service routine can call other routines that are in paged memory. The upper 16KB block of the local CPU memory space (0xC000–0xFFFF) is unpaged. It is recommended that all reset and interrupt vectors point to locations in this area or to the other unmapped pages sections of the local CPU memory map.

- CPU BGND instruction
- Breakpoint force or tag mechanism[1]

When BDM is activated, the CPU finishes executing the current instruction and then begins executing the firmware in the standard BDM firmware lookup table. When BDM is activated by a breakpoint, the type of breakpoint used determines if BDM becomes active before or after execution of the next instruction.

NOTE

If an attempt is made to activate BDM before being enabled, the CPU resumes normal instruction execution after a brief delay. If BDM is not enabled, any hardware BACKGROUND commands issued are ignored by the BDM and the CPU is not delayed.

In active BDM, the BDM registers and standard BDM firmware lookup table are mapped to addresses 0x3_FF00 to 0x3_FFFF. BDM registers are mapped to addresses 0x3_FF00 to 0x3_FF0B. The BDM uses these registers which are readable anytime by the BDM. However, these registers are not readable by user programs.

When BDM is activated while CPU executes code overlapping with BDM firmware space the saved program counter (PC) will be auto incremented by one from the BDM firmware, no matter what caused the entry into BDM active mode (BGND instruction, BACKGROUND command or breakpoints). In such a case the PC must be set to the next valid address via a WRITE_PC command before executing the GO command.

5.4.3 BDM Hardware Commands

Hardware commands are used to read and write target system memory locations and to enter active background debug mode. Target system memory includes all memory that is accessible by the CPU such as on-chip RAM, Flash, I/O and control registers.

Hardware commands are executed with minimal or no CPU intervention and do not require the system to be in active BDM for execution, although, they can still be executed in this mode. When executing a hardware command, the BDM sub-block waits for a free bus cycle so that the background access does not disturb the running application program. If a free cycle is not found within 128 clock cycles, the CPU is momentarily frozen so that the BDM can steal a cycle. When the BDM finds a free cycle, the operation does not intrude on normal CPU operation provided that it can be completed in a single cycle. However, if an operation requires multiple cycles the CPU is frozen until the operation is complete, even though the BDM found a free cycle.

The BDM hardware commands are listed in Table 5-4.

The READ_BD and WRITE_BD commands allow access to the BDM register locations. These locations are not normally in the system memory map but share addresses with the application in memory. To distinguish between physical memory locations that share the same address, BDM memory resources are

1. This method is provided by the S12S_DBG module.

Figure 5-11 shows the ACK handshake protocol in a command level timing diagram. The READ_BYTE instruction is used as an example. First, the 8-bit instruction opcode is sent by the host, followed by the address of the memory location to be read. The target BDM decodes the instruction. A bus cycle is grabbed (free or stolen) by the BDM and it executes the READ_BYTE operation. Having retrieved the data, the BDM issues an ACK pulse to the host controller, indicating that the addressed byte is ready to be retrieved. After detecting the ACK pulse, the host initiates the byte retrieval process. Note that data is sent in the form of a word and the host needs to determine which is the appropriate byte based on whether the address was odd or even.

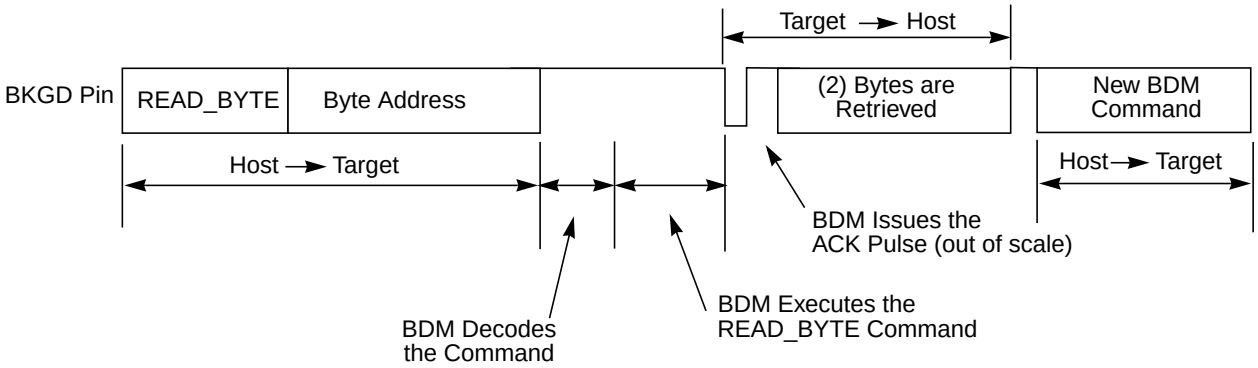

Figure 5-11. Handshake Protocol at Command Level

Differently from the normal bit transfer (where the host initiates the transmission), the serial interface ACK handshake pulse is initiated by the target MCU by issuing a negative edge in the BKGD pin. The hardware handshake protocol in Figure 5-10 specifies the timing when the BKGD pin is being driven, so the host should follow this timing constraint in order to avoid the risk of an electrical conflict in the BKGD pin.

NOTE

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

The ACK handshake protocol does not support nested ACK pulses. If a BDM command is not acknowledge by an ACK pulse, the host needs to abort the pending command first in order to be able to issue a new BDM command. When the CPU enters wait or stop while the host issues a hardware command (e.g., WRITE_BYTE), the target discards the incoming command due to the wait or stop being detected. Therefore, the command is not acknowledged by the target, which means that the ACK pulse will not be issued in this case. After a certain time the host (not aware of stop or wait) should decide to abort any possible pending ACK pulse in order to be sure a new command can be issued. Therefore, the protocol provides a mechanism in which a command, and its corresponding ACK, can be aborted.

SZ bits allow the size of access (word or byte) to be considered in the compare. Only comparators A and B feature SZE and SZ.

The TAG bit in each comparator control register is used to determine the match condition. By setting TAG, the comparator qualifies a match with the output of opcode tracking logic and a state sequencer transition occurs when the tagged instruction reaches the CPU execution stage. Whilst tagging the RW, RWE, SZE, and SZ bits and the comparator data registers are ignored; the comparator address register must be loaded with the exact opcode address.

If the TAG bit is clear (forced type match) a comparator match is generated when the selected address appears on the system address bus. If the selected address is an opcode address, the match is generated when the opcode is fetched from the memory, which precedes the instruction execution by an indefinite number of cycles due to instruction pipelining. For a comparator match of an opcode at an odd address when TAG = 0, the corresponding even address must be contained in the comparator register. Thus for an opcode at odd address (n), the comparator register must contain address (n–1).

Once a successful comparator match has occurred, the condition that caused the original match is not verified again on subsequent matches. Thus if a particular data value is verified at a given address, this address may not still contain that data value when a subsequent match occurs.

Match[0, 1, 2] map directly to Comparators [A, B, C] respectively, except in range modes (see 6.3.2.4). Comparator channel priority rules are described in the priority section (6.4.3.4).

6.4.2.1 Single Address Comparator Match

With range comparisons disabled, the match condition is an exact equivalence of address bus with the value stored in the comparator address registers. Further qualification of the type of access (R/W, word/byte) and databus contents is possible, depending on comparator channel.

6.4.2.1.1 Comparator C

Comparator C offers only address and direction (R/W) comparison. The exact address is compared, thus with the comparator address register loaded with address (n) a word access of address (n–1) also accesses (n) but does not cause a match.

Table 6-32. Comparator C Access Considerations

Condition For Valid Match	Comp C Address	RWE	RW	Examples
Read and write accesses of ADDR[n]	ADDR[n](1)	0	X	LDAA ADDR[n] STAA #$BYTE ADDR[n]
Write accesses of ADDR[n]	ADDR[n]	1	0	STAA #$BYTE ADDR[n]
Read accesses of ADDR[n]	ADDR[n]	1	1	LDAA #$BYTE ADDR[n]

1. A word access of ADDR[n-1] also accesses ADDR[n] but does not generate a match. The comparator address register must contain the exact address from the code.

6.4.2.1.2 Comparator B

Comparator B offers address, direction (R/W) and access size (word/byte) comparison. If the SZE bit is set the access size (word or byte) is compared with the SZ bit value such that only the specified size of

Field3 Bits in Compressed Pure PC Modes

Table 6-41. Compressed Pure PC Mode Field 3 Information Bit Encoding

INF1	INF0	TRACE BUFFER ROW CONTENT
0	0	Base PC address TB[17:0] contains a full PC[17:0] value
0	1	Trace Buffer[5:0] contain incremental PC relative to base address zero value
1	0	Trace Buffer[11:0] contain next 2 incremental PCs relative to base address zero value
1	1	Trace Buffer[17:0] contain next 3 incremental PCs relative to base address zero value

Each time that PC[17:6] differs from the previous base PC[17:6], then a new base address is stored. The base address zero value is the lowest address in the 64 address range

The first line of the trace buffer always gets a base PC address, this applies also on rollover.

6.4.5.5 Reading Data from Trace Buffer

The data stored in the Trace Buffer can be read provided the DBG module is not armed, is configured for tracing (TSOURCE bit is set) and the system not secured. When the ARM bit is written to 1 the trace buffer is locked to prevent reading. The trace buffer can only be unlocked for reading by a single aligned word write to DBGTB when the module is disarmed.

The Trace Buffer can only be read through the DBGTB register using aligned word reads, any byte or misaligned reads return 0 and do not cause the trace buffer pointer to increment to the next trace buffer address. The Trace Buffer data is read out first-in first-out. By reading CNT in DBGCNT the number of valid lines can be determined. DBGCNT does not decrement as data is read.

Whilst reading an internal pointer is used to determine the next line to be read. After a tracing session, the pointer points to the oldest data entry, thus if no rollover has occurred, the pointer points to line0, otherwise it points to the line with the oldest entry. In compressed Pure PC mode on rollover the line with the oldest data entry may also contain newer data entries in fields 0 and 1. Thus if rollover is indicated by the TBF bit, the line status must be decoded using the INF bits in field3 of that line. If both INF bits are clear then the line contains only entries from before the last rollover.
If INF0=1 then field 0 contains post rollover data but fields 1 and 2 contain pre rollover data.
If INF1=1 then fields 0 and 1 contain post rollover data but field 2 contains pre rollover data.

The pointer is initialized by each aligned write to DBGTBH to point to the oldest data again. This enables an interrupted trace buffer read sequence to be easily restarted from the oldest data entry.

The least significant word of line is read out first. This corresponds to the fields 1 and 0 of Table 6-37. The next word read returns field 2 in the least significant bits [3:0] and “0” for bits [15:4].

Reading the Trace Buffer while the DBG module is armed returns invalid data and no shifting of the RAM pointer occurs.

6.4.5.6 Trace Buffer Reset State

The Trace Buffer contents and DBGCNT bits are not initialized by a system reset. Thus should a system reset occur, the trace session information from immediately before the reset occurred can be read out and the number of valid lines in the trace buffer is indicated by DBGCNT. The internal pointer to the current

7.3.2.21 S12CPMU Oscillator Register (CPMUOSC)

This registers configures the external oscillator (OSCLCP).

0x02FA

	7	6	5	4	3	2	1	0
R	OSCE	OSCBW	0	OSCFILT[4:0]				
W								
Reset	0	0	0	0	0	0	0	0

Figure 7-28. S12CPMU Oscillator Register (CPMUOSC)

Read: Anytime

Write: If PROT=0 (CPMUPROT register) and PLLSEL=1 (CPMUCLKS register), then write anytime. Else write has no effect.

NOTE.

Write to this register clears the LOCK and UPOSC status bits.

NOTE.

If the chosen VCOCLK-to-OSCCLK ratio divided by two is not an integer number, then the filter can not be used and the OSCFILT[4:0] bits must be set to 0.

Table 7-22. CPMUOSC Field Descriptions

Field	Description
7 OSCE	**Oscillator Enable Bit** — This bit enables the external oscillator (OSCLCP). The UPOSC status bit in the CPMUFLG register indicates when the oscillation is stable and OSCCLK can be selected as Bus Clock or source of the COP or RTI. A loss of oscillation will lead to a clock monitor reset. 0 External oscillator is disabled. REFCLK for PLL is IRCCLK. 1 External oscillator is enabled.Clock monitor is enabled. REFCLK for PLL is the external oscillator clock divided by REFDIV. **Note:** When starting up the external oscillator (either by programming OSCE bit to 1 or on exit from Full Stop Mode with OSCE bit is already 1) the software must wait for a minimum time equivalent to the startup-time of the external oscillator t_{UPOSC} before entering Pseudo Stop Mode.
6 OSCBW	**Oscillator Filter Bandwidth Bit —** If the VCOCLK frequency exceeds 25 MHz wide bandwidth must be selected.The Oscillator Filter is described in more detail at Section 7.4.5.2, “The Adaptive Oscillator Filter. 0 Oscillator filter bandwidth is narrow (window for expected OSCCLK edge is one VCOCLK cycle). 1 Oscillator filter bandwidth is wide (window for expected OSCCLK edge is three VCOCLK cycles).
4-0 OSCFILT	**Oscillator Filter Bits** — When using the oscillator a noise filter can be enabled, which filters noise from the OSCCLK and detects if the OSCCLK is qualified or not (quality status shown by bit UPOSC). The f_{VCO} -to- f_{OSC} ratio divided by two must be an integer value. The OSCFILT[4:0] bits must be set to the calculated integer value to enable the oscillator filter). 0x0000 Oscillator Filter disabled. else Oscillator Filter enabled:

7.4 Functional Description

7.4.1 Phase Locked Loop with Internal Filter (PLL)

The PLL is used to generate a high speed PLLCLK based on a low frequency REFCLK.

The REFCLK is by default the IRCCLK which is trimmed to f_{IRC1M_TRIM}=1MHz.

If using the oscillator (OSCE=1) REFCLK will be based on OSCCLK. For increased flexibility, OSCCLK can be divided in a range of 1 to 16 to generate the reference frequency REFCLK using the REFDIV[3:0] bits. Based on the SYNDIV[5:0] bits the PLL generates the VCOCLK by multiplying the reference clock by a 2, 4, 6,... 126, 128. Based on the POSTDIV[4:0] bits the VCOCLK can be divided in a range of 1,2, 3, 4, 5, 6,... to 32 to generate the PLLCLK.

If oscillator is enabled (OSCE=1) $$f_{REF} = \frac{f_{OSC}}{(REFDIV + 1)}$$

If oscillator is disabled (OSCE=0) $$f_{REF} = f_{IRC1M}$$

$$f_{VCO} = 2 \times f_{REF} \times (SYNDIV + 1)$$

If PLL is locked (LOCK=1) $$f_{PLL} = \frac{f_{VCO}}{(POSTDIV + 1)}$$

If PLL is not locked (LOCK=0) $$f_{PLL} = \frac{f_{VCO}}{4}$$

If PLL is selected (PLLSEL=1) $$f_{bus} = \frac{f_{PLL}}{2}$$

NOTE

Although it is possible to set the dividers to command a very high clock frequency, do not exceed the specified bus frequency limit for the MCU.

Module Base + 0x0004 Access: User read/write(1)

	7	6	5	4	3	2	1	0
R	WUPIF	CSCIF	RSTAT1	RSTAT0	TSTAT1	TSTAT0	OVRIF	RXF
W								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 8-8. MSCAN Receiver Flag Register (CANRFLG)

1. Read: Anytime
   Write: Anytime when not in initialization mode, except RSTAT[1:0] and TSTAT[1:0] flags which are read-only; write of 1 clears flag; write of 0 is ignored

NOTE

The CANRFLG register is held in the reset state[1] when the initialization mode is active (INITRQ = 1 and INITAK = 1). This register is writable again as soon as the initialization mode is exited (INITRQ = 0 and INITAK = 0).

Table 8-11. CANRFLG Register Field Descriptions

Field	Description
7 WUPIF	**Wake-Up Interrupt Flag** — If the MSCAN detects CAN bus activity while in sleep mode (see Section 8.4.5.5, "MSCAN Sleep Mode,") and WUPE = 1 in CANTCTL0 (see Section 8.3.2.1, "MSCAN Control Register 0 (CANCTL0)"), the module will set WUPIF. If not masked, a wake-up interrupt is pending while this flag is set. 0 No wake-up activity observed while in sleep mode 1 MSCAN detected activity on the CAN bus and requested wake-up
6 CSCIF	**CAN Status Change Interrupt Flag** — This flag is set when the MSCAN changes its current CAN bus status due to the actual value of the transmit error counter (TEC) and the receive error counter (REC). An additional 4-bit (RSTAT[1:0], TSTAT[1:0]) status register, which is split into separate sections for TEC/REC, informs the system on the actual CAN bus status (see Section 8.3.2.6, "MSCAN Receiver Interrupt Enable Register (CANRIER)"). If not masked, an error interrupt is pending while this flag is set. CSCIF provides a blocking interrupt. That guarantees that the receiver/transmitter status bits (RSTAT/TSTAT) are only updated when no CAN status change interrupt is pending. If the TECs/RECs change their current value after the CSCIF is asserted, which would cause an additional state change in the RSTAT/TSTAT bits, these bits keep their status until the current CSCIF interrupt is cleared again. 0 No change in CAN bus status occurred since last interrupt 1 MSCAN changed current CAN bus status
5-4 RSTAT[1:0]	**Receiver Status Bits** — The values of the error counters control the actual CAN bus status of the MSCAN. As soon as the status change interrupt flag (CSCIF) is set, these bits indicate the appropriate receiver related CAN bus status of the MSCAN. The coding for the bits RSTAT1, RSTAT0 is: 00 RxOK: 0 ≤ receive error counter ≤ 96 01 RxWRN: 96 < receive error counter ≤ 127 10 RxERR: 127 < receive error counter 11 Bus-off(1): transmit error counter > 255

1. The RSTAT[1:0], TSTAT[1:0] bits are not affected by initialization mode.

1. Read: Anytime when TXEx flag is set (see Section 8.3.2.7, "MSCAN Transmitter Flag Register (CANTFLG)") and the corresponding transmit buffer is selected in CANTBSEL (see Section 8.3.2.11, "MSCAN Transmit Buffer Selection Register (CANTBSEL)")
   Write: Unimplemented

8.4 Functional Description

8.4.1 General

This section provides a complete functional description of the MSCAN.

Table 8-38. CPU vs. MSCAN Operating Modes

CPU Mode	MSCAN Mode			
	Normal	Reduced Power Consumption		
		Sleep	Power Down	Disabled (CANE=0)
RUN	CSWAI = X[1] SLPRQ = 0 SLPAK = 0	CSWAI = X SLPRQ = 1 SLPAK = 1		CSWAI = X SLPRQ = X SLPAK = X
WAIT	CSWAI = 0 SLPRQ = 0 SLPAK = 0	CSWAI = 0 SLPRQ = 1 SLPAK = 1	CSWAI = 1 SLPRQ = X SLPAK = X	CSWAI = X SLPRQ = X SLPAK = X
STOP			CSWAI = X SLPRQ = X SLPAK = X	CSWAI = X SLPRQ = X SLPAK = X

1. 'X' means don't care.

8.4.5.1 Operation in Run Mode

As shown in Table 8-38, only MSCAN sleep mode is available as low power option when the CPU is in run mode.

8.4.5.2 Operation in Wait Mode

The WAI instruction puts the MCU in a low power consumption stand-by mode. If the CSWAI bit is set, additional power can be saved in power down mode because the CPU clocks are stopped. After leaving this power down mode, the MSCAN restarts and enters normal mode again.

While the CPU is in wait mode, the MSCAN can be operated in normal mode and generate interrupts (registers can be accessed via background debug mode).

8.4.5.3 Operation in Stop Mode

The STOP instruction puts the MCU in a low power consumption stand-by mode. In stop mode, the MSCAN is set in power down mode regardless of the value of the SLPRQ/SLPAK and CSWAI bits (Table 8-38).

8.4.5.4 MSCAN Normal Mode

This is a non-power-saving mode. Enabling the MSCAN puts the module from disabled mode into normal mode. In this mode the module can either be in initialization mode or out of initialization mode. See Section 8.4.4.5, "MSCAN Initialization Mode".

8.4.7.3 Receive Interrupt

A message is successfully received and shifted into the foreground buffer (RxFG) of the receiver FIFO. This interrupt is generated immediately after receiving the EOF symbol. The RXF flag is set. If there are multiple messages in the receiver FIFO, the RXF flag is set as soon as the next message is shifted to the foreground buffer.

8.4.7.4 Wake-Up Interrupt

A wake-up interrupt is generated if activity on the CAN bus occurs during MSCAN sleep or power-down mode.

NOTE

This interrupt can only occur if the MSCAN was in sleep mode (SLPRQ = 1 and SLPAK = 1) before entering power down mode, the wake-up option is enabled (WUPE = 1), and the wake-up interrupt is enabled (WUPIE = 1).

8.4.7.5 Error Interrupt

An error interrupt is generated if an overrun of the receiver FIFO, error, warning, or bus-off condition occurrs. MSCAN Receiver Flag Register (CANRFLG) indicates one of the following conditions:

- **Overrun** — An overrun condition of the receiver FIFO as described in Section 8.4.2.3, "Receive Structures," occurred.
- **CAN Status Change** — The actual value of the transmit and receive error counters control the CAN bus state of the MSCAN. As soon as the error counters skip into a critical range (Tx/Rx-warning, Tx/Rx-error, bus-off) the MSCAN flags an error condition. The status change, which caused the error condition, is indicated by the TSTAT and RSTAT flags (see Section 8.3.2.5, "MSCAN Receiver Flag Register (CANRFLG)" and Section 8.3.2.6, "MSCAN Receiver Interrupt Enable Register (CANRIER)").

8.4.7.6 Interrupt Acknowledge

Interrupts are directly associated with one or more status flags in either the MSCAN Receiver Flag Register (CANRFLG) or the MSCAN Transmitter Flag Register (CANTFLG). Interrupts are pending as long as one of the corresponding flags is set. The flags in CANRFLG and CANTFLG must be reset within the interrupt handler to handshake the interrupt. The flags are reset by writing a 1 to the corresponding bit position. A flag cannot be cleared if the respective condition prevails.

NOTE

It must be guaranteed that the CPU clears only the bit causing the current interrupt. For this reason, bit manipulation instructions (BSET) must not be used to clear interrupt flags. These instructions may cause accidental clearing of interrupt flags which are set after entering the current interrupt service routine.

Address	Name		Bit 7	6	5	4	3	2	1	Bit 0
0x0020	ATDDR8	R W	See Section 9.3.2.12.1, “Left Justified Result Data (DJM=0)” and Section 9.3.2.12.2, “Right Justified Result Data (DJM=1)”							
0x0022	ATDDR9	R W	See Section 9.3.2.12.1, “Left Justified Result Data (DJM=0)” and Section 9.3.2.12.2, “Right Justified Result Data (DJM=1)”							
0x0024 - 0x002F	Unimple-mented	R	0	0	0	0	0	0	0	0
		W								

= Unimplemented or Reserved

Figure 9-2. ADC12B10C Register Summary (Sheet 3 of 3)

9.3.2 Register Descriptions

This section describes in address order all the ADC12B10C registers and their individual bits.

9.3.2.1 ATD Control Register 0 (ATDCTL0)

Writes to this register will abort current conversion sequence.

Module Base + 0x0000

	7	6	5	4	3	2	1	0
R	Reserved	0	0	0	WRAP3	WRAP2	WRAP1	WRAP0
W								
Reset	0	0	0	0	1	1	1	1

= Unimplemented or Reserved

Figure 9-3. ATD Control Register 0 (ATDCTL0)

Read: Anytime

Write: Anytime, in special modes always write 0 to Reserved Bit 7.

Table 9-1. ATDCTL0 Field Descriptions

Field	Description
3-0 WRAP[3-0]	**Wrap Around Channel Select Bits** — These bits determine the channel for wrap around when doing multi-channel conversions. The coding is summarized in Table 9-2.

Table 9-2. Multi-Channel Wrap Around Coding

WRAP3	WRAP2	WRAP1	WRAP0	Multiple Channel Conversions (MULT = 1) Wraparound to AN0 after Converting
0	0	0	0	Reserved(1)
0	0	0	1	AN1
0	0	1	0	AN2
0	0	1	1	AN3
0	1	0	0	AN4
0	1	0	1	AN5

between the PWM counter and the period register behaves differently depending on what output mode is selected as shown in Figure 10-35 and described in Section 10.4.2.5, “Left Aligned Outputs,” and Section 10.4.2.6, “Center Aligned Outputs.”

Each channel counter can be read at anytime without affecting the count or the operation of the PWM channel.

Any value written to the counter causes the counter to reset to 0x0000, the counter direction to be set to up, the immediate load of both duty and period registers with values from the buffers, and the output to change according to the polarity bit. When the channel is disabled (PWMEx = 0), the counter stops. When a channel becomes enabled (PWMEx = 1), the associated PWM counter continues from the count in the PWMCNTx register. This allows the waveform to resume when the channel is re-enabled. When the channel is disabled, writing 0 to the period register will cause the counter to reset on the next selected clock.

NOTE

If the user wants to start a new “clean” PWM waveform without any “history” from the old waveform, the user must write to channel counter (PWMCNTx) prior to enabling the PWM channel (PWMEx = 1).

Generally, writes to the counter are done prior to enabling a channel to start from a known state. However, writing a counter can also be done while the PWM channel is enabled (counting). The effect is similar to writing the counter when the channel is disabled except that the new period is started immediately with the output set according to the polarity bit.

NOTE

Writing to the counter while the channel is enabled can cause an irregular PWM cycle to occur.

The counter is cleared at the end of the effective period (see Section 10.4.2.5, “Left Aligned Outputs,” and Section 10.4.2.6, “Center Aligned Outputs,” for more details).

Table 10-11. PWM Timer Counter Conditions

Counter Clears (0x0000)	Counter Counts	Counter Stops
When PWMCNTx register written to any value	When PWM channel is enabled (PWMEx = 1). Counts from last value in PWMCNTx.	When PWM channel is disabled (PWMEx = 0)
Effective period ends		

10.4.2.5 Left Aligned Outputs

The PWM timer provides the choice of two types of outputs, left aligned or center aligned outputs. They are selected with the CAEx bits in the PWMCAE register. If the CAEx bit is cleared (CAEx = 0), the corresponding PWM output will be left aligned.

In left aligned output mode, the 8-bit counter is configured as an up counter only. It compares to two registers, a duty register and a period register as shown in the block diagram in Figure 10-35. When the

Table 10-12 is used to summarize which channels are used to set the various control bits when in 16-bit mode.

Table 10-12. 16-bit Concatenation Mode Summary

CONxx	PWMEx	PPOLx	PCLKx	CAEx	PWMx Output
CON45	PWME5	PPOL5	PCLK5	CAE5	PWM5
CON23	PWME3	PPOL3	PCLK3	CAE3	PWM3
CON01	PWME1	PPOL1	PCLK1	CAE1	PWM1

10.4.2.8 PWM Boundary Cases

Table 10-13 summarizes the boundary conditions for the PWM regardless of the output mode (left aligned or center aligned) and 8-bit (normal) or 16-bit (concatenation):

Table 10-13. PWM Boundary Cases

PWMDTYx	PWMPERx	PPOLx	PWMx Output
0x0000 (indicates no duty)	>0x0000	1	Always Low
0x0000 (indicates no duty)	>0x0000	0	Always High
XX	0x0000[1] (indicates no period)	1	Always High
XX	0x0000[1] (indicates no period)	0	Always Low
>= PWMPERx	XX	1	Always High
>= PWMPERx	XX	0	Always Low

1. Counter = 0x0000 and does not count.

10.5 Resets

The reset state of each individual bit is listed within the register description section (see Section 10.3, “Memory Map and Register Definition,” which details the registers and their bit-fields. All special functions or modes which are initialized during or just following reset are described within this section.

- The 8-bit up/down counter is configured as an up counter out of reset.
- All the channels are disabled and all the counters don’t count.

10.6 Interrupts

The PWM8B6CV1 module has only one interrupt which is generated at the time of emergency shutdown, if the corresponding enable bit (PWMIE) is set. This bit is the enable for the interrupt. The interrupt flag PWMIF is set whenever the input level of the PWM5 channel changes while PWM5ENA=1 or when PWMENA is being asserted while the level at PWM5 is active.

A description of the registers involved and affected due to this interrupt is explained in Section 10.3.2.15, “PWM Shutdown Register (PWMSDN).”

As long as no more than one slave device drives the system slave's serial data output line, it is possible for several slaves to receive the same transmission from a master, although the master would not receive return information from all of the receiving slaves.

If the CPHA bit in SPI control register 1 is clear, odd numbered edges on the SCK input cause the data at the serial data input pin to be latched. Even numbered edges cause the value previously latched from the serial data input pin to shift into the LSB or MSB of the SPI shift register, depending on the LSBFE bit.

If the CPHA bit is set, even numbered edges on the SCK input cause the data at the serial data input pin to be latched. Odd numbered edges cause the value previously latched from the serial data input pin to shift into the LSB or MSB of the SPI shift register, depending on the LSBFE bit.

When CPHA is set, the first edge is used to get the first data bit onto the serial data output pin. When CPHA is clear and the $\overline{SS}$ input is low (slave selected), the first bit of the SPI data is driven out of the serial data output pin. After the nth[1] shift, the transfer is considered complete and the received data is transferred into the SPI data register. To indicate transfer is complete, the SPIF flag in the SPI status register is set.

NOTE

A change of the bits CPOL, CPHA, SSOE, LSBFE, MODFEN, SPC0, or BIDIROE with SPC0 set in slave mode will corrupt a transmission in progress and must be avoided.

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.

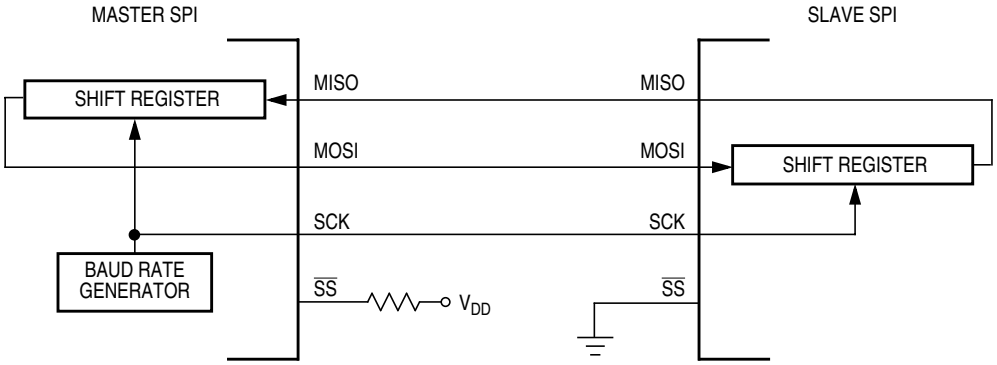

Figure 12-11. Master/Slave Transfer Block Diagram

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.

1. n depends on the selected transfer width, please refer to Section 12.3.2.2, "SPI Control Register 2 (SPICR2)