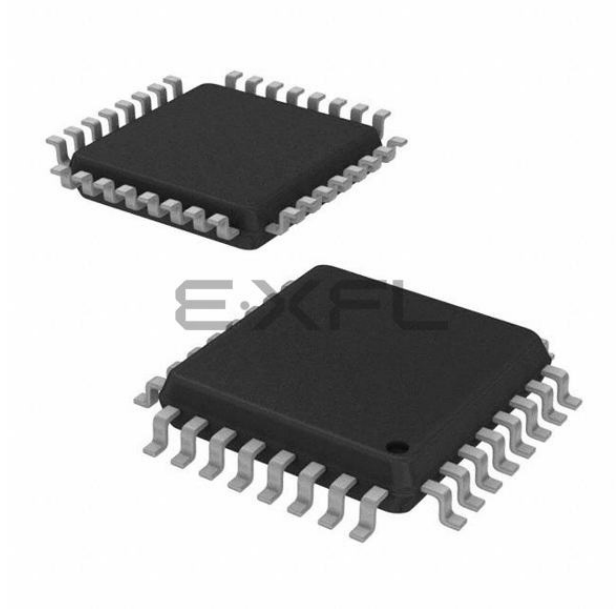


Welcome to E-XFL.COM

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	S12Z
Core Size	16-Bit
Speed	32MHz
Connectivity	I²C, IrDA, LINbus, SCI, SPI, UART/USART
Peripherals	LVD, POR, PWM, WDT
Number of I/O	19
Program Memory Size	8KB (8K x 8)
Program Memory Type	FLASH
EEPROM Size	128 x 8
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	5.5V ~ 18V
Data Converters	A/D 6x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	32-LQFP
Supplier Device Package	32-LQFP (7x7)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/s9s12zvl8f0clcr

- Amplifier output connected to ADC
- Amplifier signal reference voltage selectable from DAC, VDDA/2 or input pin

1.4.16 Supply Voltage Sensor (BATS)

- Monitoring of supply (VSUP) voltage
- Internal ADC interface from an internal resistive divider
- Generation of low or high voltage interrupts

1.4.17 On-Chip Voltage Regulator system (VREG)

- Voltage regulator
  - Linear voltage regulator directly supplied by VSUP
  - Supports 3.3V or 5V VDDX
  - Low-voltage detect on VSUP
  - Power-on reset (POR)
  - Low-voltage reset (LVR) for VDDX domain
  - External ballast device support to extend current capability and reduce internal power dissipation
  - Capable of supplying both the MCU internally plus external components
  - Over-temperature interrupt
- Internal voltage regulator
  - Linear voltage regulator with bandgap reference
  - Low-voltage detect on VDDA
  - Power-on reset (POR) circuit
  - Low-voltage reset for VDD domain

The operating mode out of reset is determined by the state of the MODC signal during reset (Table 1-9). The MODC bit in the MODE register shows the current operating mode and provides limited mode switching during operation. The state of the MODC signal is latched into this bit on the rising edge of RESET.

Table 1-9. Chip Modes

Chip Modes	MODC
Normal single chip	1
Special single chip	0

1.10.2.1 Normal Single-Chip Mode

This mode is intended for normal device operation. The opcode from the on-chip memory is being executed after reset (requires the reset vector to be programmed correctly). The processor program is executed from internal memory. To avoid unpredictable behavior do not start the device in Normal Single-Chip mode while the FLASH is erased.

1.10.2.2 Special Single-Chip Mode

This mode is used for debugging operation, boot-strapping, or security related operations. The background debug mode (BDM) is active on leaving reset in this mode.

1.10.3 Debugging Modes

The background debug mode (BDM) can be activated by the BDC module or directly when resetting into Special Single-Chip mode. Detailed information can be found in the BDC module section.

Writing to internal memory locations using the debugger, whilst code is running or at a breakpoint, can change the flow of application code.

The MC9S12ZVL-Family supports BDC communication throughout the device Stop mode. During Stop mode, writes to control registers can alter the operation and lead to unexpected results. It is thus recommended not to reconfigure the peripherals during STOP using the debugger.

1.10.4 Low Power Modes

The device has two dynamic-power modes (run and wait) and two static low-power modes (stop and pseudo stop). For a detailed description refer to the CPMU section.

- Dynamic power mode: Run
  - Run mode is the main full performance operating mode with the entire device clocked. The user can configure the device operating speed through selection of the clock source and the phase locked loop (PLL) frequency. To save power, unused peripherals must not be enabled.
- Dynamic power mode: Wait
  - This mode is entered when the CPU executes the WAI instruction. In this mode the CPU does not execute instructions. The internal CPU clock is switched off. All peripherals can be active

1.13.3 Flash IFR Mapping

Table 1-15. Flash IFR Mapping

Target																IFR Byte Address
F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0	
				ADC reference conversion using VDDA/VSSA												0x1F_C040 & 0x1F_C041
				ADC reference conversion using VRH/VRL												0x1F_C042 & 0x1F_C043

1.14 Application Information

1.14.1 ADC Calibration

For applications that do not provide external ADC reference voltages, the VDDA/VSSA supplies can be used as sources for VRH/VRL respectively. Since the VDDA must be connected to VDDX at board level in the application, the accuracy of the VDDA reference is limited by the internal voltage regulator accuracy. In order to compensate for VDDA reference voltage variation in this case, the reference voltage is measured during production test using the internal reference voltage VBG, which has a narrow variation over temperature and external voltage supply. VBG is mapped to an internal channel of the ADC module, see Table 1-7. The resulting 12-bit right justified ADC conversion results of VBG are stored to the flash IFR for reference, as listed in Table 1-15.

The measurement conditions of the reference conversion are listed in the device electrical parameters appendix. By measuring the voltage VBG in the application environment and comparing the result to the reference value in the IFR, it is possible to determine the current ADC reference voltage V_{RH}:

$$V_{RH} = \frac{StoredReference}{ConvertedReference} \bullet 5V$$

The exact absolute value of an analog conversion can be determined as follows:

$$Result = ConvertedADInput \bullet \frac{StoredReference \bullet 5V}{ConvertedReference \bullet 2^{n}}$$

With:

ConvertedADInput:	Result of the analog to digital conversion of the desired pin
ConvertedReference:	Result of internal channel conversion
StoredReference:	Value in IFR location
n:	ADC resolution (12 bit)

Table 3-2. ACMPC0 Register Field Descriptions (continued)

Field	Description
3-2 ACHYS1-0	**ACMP Hysteresis** — These bits select the ACMP hysteresis V_{ACMP_hyst}. 00 Select smallest hysteresis 01 Select small hysteresis 10 Select large hysteresis 11 Select largest hysteresis
1-0 ACMOD 1-0	**ACMP Mode** — These bits select the type of compare event to set ACIF. 00 Flag setting disabled 01 Flag setting on output rising edge 10 Flag setting on output falling edge 11 Flag setting on output rising or falling edge

3.6.2.2 ACMP Control Register 1 (ACMPC1)

Module Base + 0x0001 Access: User read/write[1]

	7	6	5	4	3	2	1	0
R	0	0	ACPSEL1-0		0	0	ACNSEL1-0	
W								
Reset	0	0	0	0	0	0	0	0

Figure 3-3. ACMP Control Register (ACMPC1)

[1] Read: Anytime
Write: Only if ACE=0

Table 3-3. ACMPC1 Routing Register Field Descriptions

Field	Description
5-4 ACPSEL1-0	**ACMP Positive Input Select** — These bits select the ACMP non-inverting input connected to ACMPP. 11 acmpi_1 10 acmpi_0 01 ACMP_1 00 ACMP_0
1-0 ACNSEL1-0	**ACMP Negative Input Select** — These bits select the ACMP inverting input connected to ACMPN. 11 acmpi_1 10 acmpi_0 01 ACMP_1 00 ACMP_0

NOTE

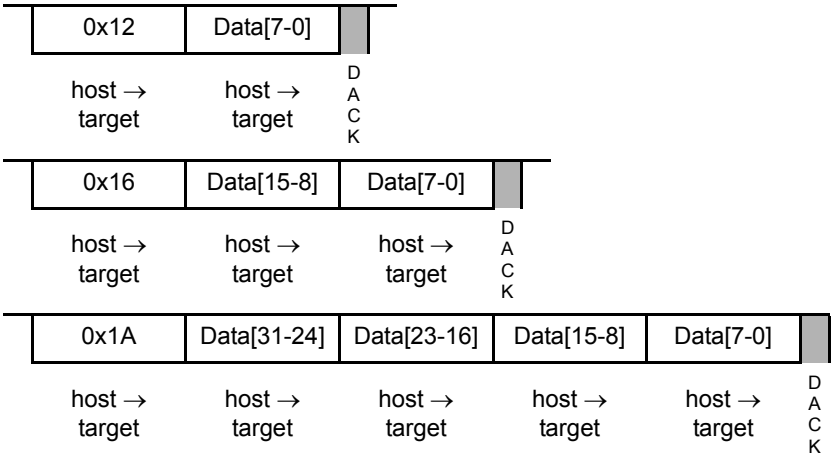
DUMP_MEM{_WS} is a valid command only when preceded by SYNC, NOP, READ_MEM{_WS}, or another DUMP_MEM{_WS} command. Otherwise, an illegal command response is returned, setting the ILLCMD bit. NOP can be used for inter-command padding without corrupting the address pointer.

The size field (sz) is examined each time a DUMP_MEM{_WS} command is processed, allowing the operand size to be dynamically altered. The examples show the DUMP_MEM.B{_WS}, DUMP_MEM.W{_WS} and DUMP_MEM.L{_WS} commands.

5.4.4.6 FILL_MEM.sz, FILL_MEM.sz_WS

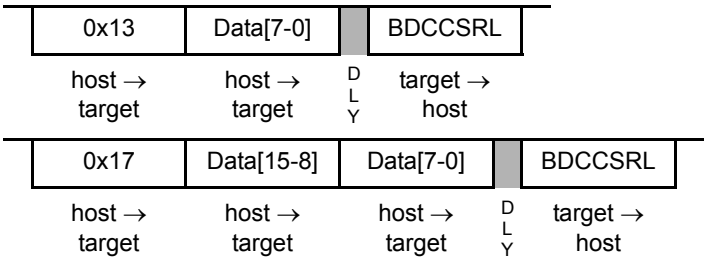
FILL_MEM.sz

Write memory specified by debug address register, then increment address **Non-intrusive**



FILL_MEM.sz_WS

Write memory specified by debug address register with status, then increment address **Non-intrusive**



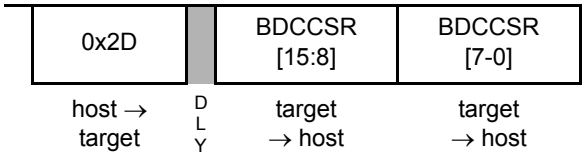
Read from location defined by the previous READ_MEM. The previous READ_MEM command defines the address, subsequent READ_SAME commands return contents of same address. The example shows the sequence for reading a 16-bit word size. Byte alignment details are described in Section 5.4.5.2, “BDC Access Of Device Memory Mapped Resources”. If enabled, an ACK pulse is driven before the data bytes are transmitted.

NOTE

READ_SAME{_WS} is a valid command only when preceded by SYNC, NOP, READ_MEM{_WS}, or another READ_SAME{_WS} command. Otherwise, an illegal command response is returned, setting the ILLCMD bit. NOP can be used for inter-command padding without corrupting the address pointer.

5.4.4.14 READ_BDCCSR

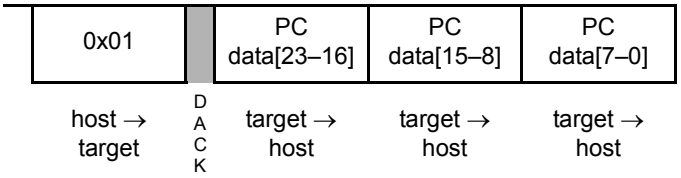
Read BDCCSR Status Register **Always Available**



Read the BDCCSR status register. This command can be executed in any mode.

5.4.4.15 SYNC_PC

Sample current PC **Non-intrusive**



This command returns the 24-bit CPU PC value to the host. Unsuccessful SYNC_PC accesses return 0xEE for each byte. If enabled, an ACK pulse is driven before the data bytes are transmitted. The value of 0xEE is returned if a timeout occurs, whereby NORESP is set. This can occur if the CPU is executing the WAI instruction, or the STOP instruction with BDCCIS clear, or if a CPU access is delayed by EWAIT. If the CPU is executing the STOP instruction and BDCCIS is set, then SYNC_PC returns the PC address of the instruction following STOP in the code listing.

This command can be used to dynamically access the PC for performance monitoring as the execution of this command is considerably less intrusive to the real-time operation of an application than a BACKGROUND/read-PC/GO command sequence. Whilst the BDC is not in active BDM, SYNC_PC returns the PC address of the instruction currently being executed by the CPU. In active BDM, SYNC_PC

7.3.2.5 Debug State Control Register 3 (DBGSCR3)

Address: 0x0109

	7	6	5	4	3	2	1	0
R	C3SC1	C3SC0	0	0	C1SC1	C1SC0	C0SC1	C0SC0
W								
Reset	0	0	0	0	0	0	0	0

Figure 7-8. Debug State Control Register 3 (DBGSCR3)

Read: Anytime.

Write: If DBG is not armed.

The state control register three selects the targeted next state whilst in State3. The matches refer to the outputs of the comparator match control logic as depicted in Figure 7-1 and described in Section 7.3.2.8, “Debug Comparator A Control Register (DBGACTL)”. Comparators must be enabled by setting the comparator enable bit in the associated DBGxCTL control register.

Table 7-11. DBGSCR3 Field Descriptions

Field	Description
1–0 C0SC[1:0]	Channel 0 State Control. These bits select the targeted next state whilst in State3 following a match0.
3–2 C1SC[1:0]	Channel 1 State Control. These bits select the targeted next state whilst in State3 following a match1.
7–6 C3SC[1:0]	Channel 3 State Control. If EEVE !=10, these bits select the targeted next state whilst in State3 following a match3. If EEVE =10, these bits select the targeted next state whilst in State3 following an external event.

Table 7-12. State3 Match State Sequencer Transitions

CxSC[1:0]	Function
00	Match has no effect
01	Match forces sequencer to State1
10	Match forces sequencer to State2
11	Match forces sequencer to Final State

In the case of simultaneous matches, the match on the higher channel number (3....0) has priority.

7.3.2.6 Debug Event Flag Register (DBGEFR)

Address: 0x010A

	7	6	5	4	3	2	1	0
R	0	TRIGF	0	EEVF	ME3	0	ME1	ME0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 7-9. Debug Event Flag Register (DBGEFR)

Table 9-12. RTI Frequency Divide Rates for RTDEC = 0

RTR[3:0]	RTR[6:4] =							
	000 (OFF)	**001 (2^{10})**	**010 (2^{11})**	**011 (2^{12})**	**100 (2^{13})**	**101 (2^{14})**	**110 (2^{15})**	**111 (2^{16})**
0000 (÷1)	OFF[1]	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}
0001 (÷2)	OFF	$2x2^{10}$	$2x2^{11}$	$2x2^{12}$	$2x2^{13}$	$2x2^{14}$	$2x2^{15}$	$2x2^{16}$
0010 (÷3)	OFF	$3x2^{10}$	$3x2^{11}$	$3x2^{12}$	$3x2^{13}$	$3x2^{14}$	$3x2^{15}$	$3x2^{16}$
0011 (÷4)	OFF	$4x2^{10}$	$4x2^{11}$	$4x2^{12}$	$4x2^{13}$	$4x2^{14}$	$4x2^{15}$	$4x2^{16}$
0100 (÷5)	OFF	$5x2^{10}$	$5x2^{11}$	$5x2^{12}$	$5x2^{13}$	$5x2^{14}$	$5x2^{15}$	$5x2^{16}$
0101 (÷6)	OFF	$6x2^{10}$	$6x2^{11}$	$6x2^{12}$	$6x2^{13}$	$6x2^{14}$	$6x2^{15}$	$6x2^{16}$
0110 (÷7)	OFF	$7x2^{10}$	$7x2^{11}$	$7x2^{12}$	$7x2^{13}$	$7x2^{14}$	$7x2^{15}$	$7x2^{16}$
0111 (÷8)	OFF	$8x2^{10}$	$8x2^{11}$	$8x2^{12}$	$8x2^{13}$	$8x2^{14}$	$8x2^{15}$	$8x2^{16}$
1000 (÷9)	OFF	$9x2^{10}$	$9x2^{11}$	$9x2^{12}$	$9x2^{13}$	$9x2^{14}$	$9x2^{15}$	$9x2^{16}$
1001 (÷10)	OFF	$10x2^{10}$	$10x2^{11}$	$10x2^{12}$	$10x2^{13}$	$10x2^{14}$	$10x2^{15}$	$10x2^{16}$
1010 (÷11)	OFF	$11x2^{10}$	$11x2^{11}$	$11x2^{12}$	$11x2^{13}$	$11x2^{14}$	$11x2^{15}$	$11x2^{16}$
1011 (÷12)	OFF	$12x2^{10}$	$12x2^{11}$	$12x2^{12}$	$12x2^{13}$	$12x2^{14}$	$12x2^{15}$	$12x2^{16}$
1100 (÷13)	OFF	$13x2^{10}$	$13x2^{11}$	$13x2^{12}$	$13x2^{13}$	$13x2^{14}$	$13x2^{15}$	$13x2^{16}$
1101 (÷14)	OFF	$14x2^{10}$	$14x2^{11}$	$14x2^{12}$	$14x2^{13}$	$14x2^{14}$	$14x2^{15}$	$14x2^{16}$
1110 (÷15)	OFF	$15x2^{10}$	$15x2^{11}$	$15x2^{12}$	$15x2^{13}$	$15x2^{14}$	$15x2^{15}$	$15x2^{16}$
1111 (÷16)	OFF	$16x2^{10}$	$16x2^{11}$	$16x2^{12}$	$16x2^{13}$	$16x2^{14}$	$16x2^{15}$	$16x2^{16}$

[1] Denotes the default value out of reset.This value should be used to disable the RTI to ensure future backwards compatibility.

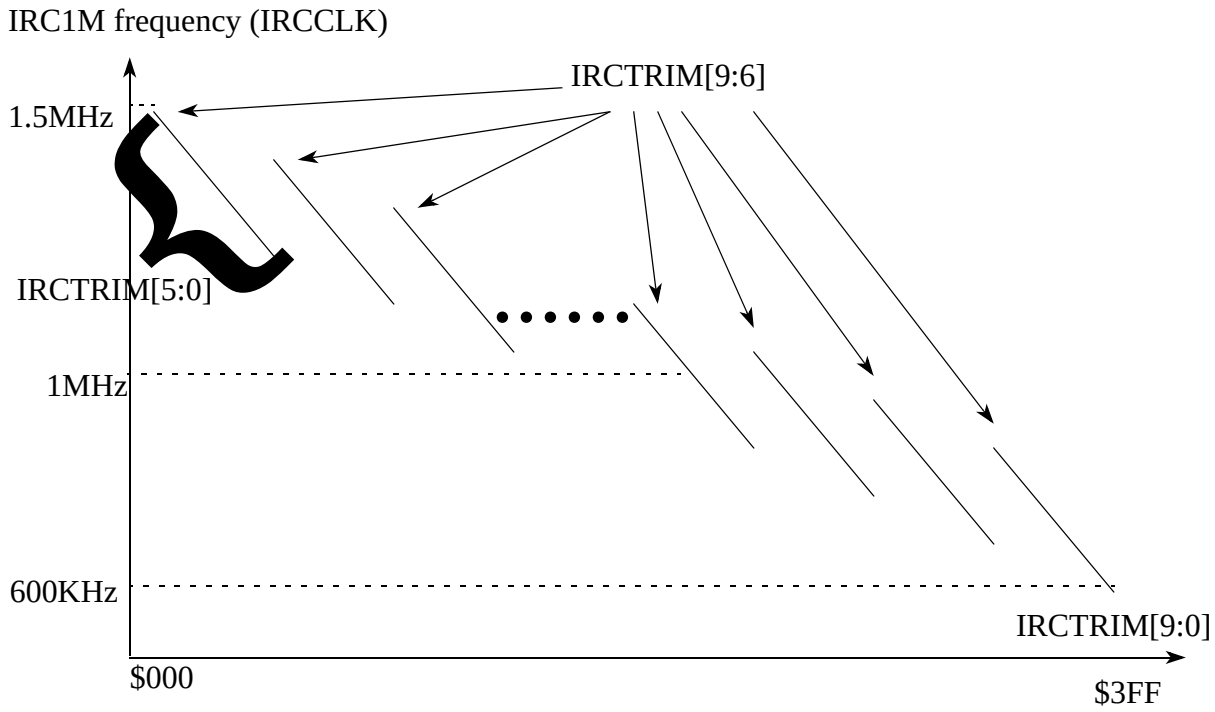


Figure 9-31. IRC1M Frequency Trimming Diagram

10.3 Key Features

- Programmer’s Model with List Based Architecture for conversion command and result value organization
- Selectable resolution of 8-bit, 10-bit, or 12-bit
- Channel select control for n external analog input channels
- Provides up to eight device internal channels (please see the device reference manual for connectivity information and Figure 10-2)
- Programmable sample time
- A sample buffer amplifier for channel sampling (improved performance in view to influence of channel input path resistance versus conversion accuracy)
- Left/right justified result data
- Individual selectable VRH_0/1 and VRL_0/1 inputs (ADC12B_LBA V1 and V2) or VRH_0/1/2 inputs (ADC12B_LBA V3) on a conversion command basis (please see Figure 10-2, Table 10-2)
- Special conversions for selected VRH_0/1 (V1 and V2) or VRH_0/1/2 (V3), VRL_0/1 (V1 and V2) or VRL_0 (V3), (VRL_0/1 + VRH_0/1) / 2 (V1 and V2) or (VRL_0 + VRH_0/1/2) / 2 (V3) (please see Table 10-2)
- 15 conversion interrupts with flexible interrupt organization per conversion result
- One dedicated interrupt for “End Of List” type commands
- Command Sequence List (CSL) with a maximum number of 64 command entries
- Provides conversion sequence abort
- Restart from top of active Command Sequence List (CSL)
- The Command Sequence List and Result Value List are implemented in double buffered manner (two lists in parallel for each function)
- Conversion Command (CSL) loading possible from System RAM or NVM
- Single conversion flow control register with software selectable access path
- Two conversion flow control modes optimized to different application use cases
- Four option bits in the conversion command for top level SoC specific feature/function implementation option (Please refer to the device reference manual for details of the top level feature/function if implemented)

Table 10-11. Summary of Conversion Flow Control Bit Scenarios

RSTA	TRIG	SEQA	LDOK	Conversion Flow Control Mode	Conversion Flow Control Scenario
0	0	0	0	Both Modes	Valid
0	0	0	1	Both Modes	Can Not Occur
0	0	1	0	Both Modes	Valid[5]
0	0	1	1	Both Modes	Can Not Occur
0	1	0	0	Both Modes	Valid[2]
0	1	0	1	Both Modes	Can Not Occur
0	1	1	0	Both Modes	Can Not Occur
0	1	1	1	Both Modes	Can Not Occur
1	0	0	0	Both Modes	Valid[4]
1	0	0	1	Both Modes	Valid[1 4]
1	0	1	0	Both Modes	Valid[3 4 5]
1	0	1	1	Both Modes	Valid[1 3 4 5]
1	1	0	0	“Restart Mode”	Error flag TRIG_EIF set
				“Trigger Mode”	Valid[2 4 6]
1	1	0	1	“Restart Mode”	Error flag TRIG_EIF set
				“Trigger Mode”	Valid[1 2 4 6]
1	1	1	0	“Restart Mode”	Error flag TRIG_EIF set
				“Trigger Mode”	Valid[2 3 4 5 6]
1	1	1	1	“Restart Mode”	Error flag TRIG_EIF set
				“Trigger Mode”	Valid[1 2 3 4 5 6]

1 Swap CSL buffer
2 Start conversion sequence
3 Prevent RSTA_EIF and LDOK_EIF
4 Load conversion command from top of CSL
5 Abort any ongoing conversion, conversion sequence and CSL
6 Bit TRIG set automatically in Trigger Mode

For a detailed description of all conversion flow control bit scenarios please see also Section 10.6.3.2.4, “The two conversion flow control Mode Configurations, Section 10.6.3.2.5, “The four ADC conversion flow control bits and Section 10.6.3.2.6, “Conversion flow control in case of conversion sequence control bit overrun scenarios

10.6.3.2.6 Conversion flow control in case of conversion sequence control bit overrun scenarios

Restart Request Overrun:
If a legal Restart Request is detected and no Restart Event is in progress, the RSTA bit is set due to the request. The set RSTA bit indicates that a Restart Request was detected and the Restart Event is in process. In case further Restart Requests occur while the RSTA bit is set, this is defined a overrun situation. This scenario is likely to occur when bit STR_SEQA is set or when a Restart Event causes a Sequence Abort Event. The request overrun is captured in a background register that always stores the last detected overrun request. Hence if the overrun situation occurs more than once while a Restart Event is in progress, only the latest overrun request is pending. When the RSTA bit is cleared, the latest overrun request is processed and RSTA is set again one cycle later.

LoadOK Overrun:
Simultaneously at any Restart Request overrun situation the LoadOK input is evaluated and the status is captured in a background register which is alternated anytime a Restart Request Overrun occurs while Load OK Request is asserted. The Load OK background register is cleared as soon as the pending Restart Request gets processed.

Trigger Overrun:
If a Trigger occurs whilst bit TRIG is already set, this is defined as a Trigger overrun situation and causes the ADC to cease conversion at the next conversion boundary and to set bit TRIG_EIF. A overrun is also detected if the Trigger Event occurs automatically generated by hardware in “Trigger Mode” due to a Restart Event and simultaneously a Trigger Event is generated via data bus or internal interface. In this case the ADC ceases operation before conversion begins to sample. In “Trigger Mode” a Restart Request Overrun does not cause a Trigger Overrun (bit TRIG_EIF not set).

Sequence Abort Request Overrun:
If a Sequence Abort Request occurs whilst bit SEQA is already set, this is defined as a Sequence Abort Request Overrun situation and the overrun request is ignored.

13.1.1 Glossary

Table 13-2. Terminology

ACK	Acknowledge of CAN message
CAN	Controller Area Network
CRC	Cyclic Redundancy Code
EOF	End of Frame
FIFO	First-In-First-Out Memory
IFS	Inter-Frame Sequence
SOF	Start of Frame
CPU bus	CPU related read/write data bus
CAN bus	CAN protocol related serial bus
oscillator clock	Direct clock from external oscillator
bus clock	CPU bus related clock
CAN clock	CAN protocol related clock

13.1.2 Block Diagram

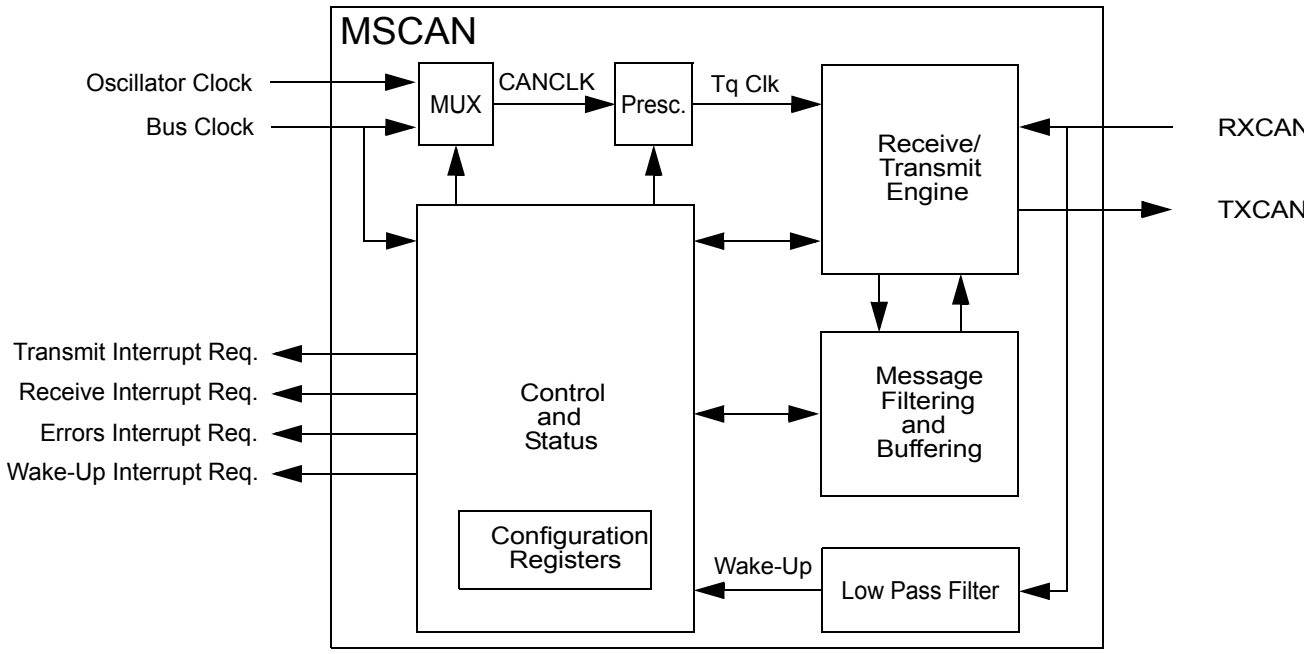


Figure 13-1. MSCAN Block Diagram

Chapter 17
Pulse-Width Modulator (S12PWM8B8CV2)

Table 17-1. Revision History

Revision Number	Revision Date	Sections Affected	Description of Changes
v02.00	Feb. 20, 2009	All	Initial revision of scalable PWM. Started from pwm_8b8c (v01.08).

17.1 Introduction

The Version 2 of S12 PWM module is a channel scalable and optimized implementation of S12 PWM8B8C Version 1. The channel is scalable in pairs from PWM0 to PWM7 and the available channel number is 2, 4, 6 and 8. The shutdown feature has been removed and the flexibility to select one of four clock sources per channel has improved. If the corresponding channels exist and shutdown feature is not used, the Version 2 is fully software compatible to Version 1.

17.1.1 Features

The scalable PWM block includes these distinctive features:

- Up to eight independent PWM channels, scalable in pairs (PWM0 to PWM7)
- Available channel number could be 2, 4, 6, 8 (refer to device specification for exact number)
- Programmable period and duty cycle for each channel
- Dedicated counter for each PWM channel
- Programmable PWM enable/disable for each channel
- Software selection of PWM duty pulse polarity for each channel
- Period and duty cycle are double buffered. Change takes effect when the end of the effective period is reached (PWM counter reaches zero) or when the channel is disabled.
- Programmable center or left aligned outputs on individual channels
- Up to eight 8-bit channel or four 16-bit channel PWM resolution
- Four clock sources (A, B, SA, and SB) provide for a wide range of frequencies
- Programmable clock select logic

17.1.2 Modes of Operation

There is a software programmable option for low power consumption in wait mode that disables the input clock to the prescaler.

NOTE

Changing the PWM output mode from left aligned to center aligned output (or vice versa) while channels are operating can cause irregularities in the PWM output. It is recommended to program the output mode before enabling the PWM channel.

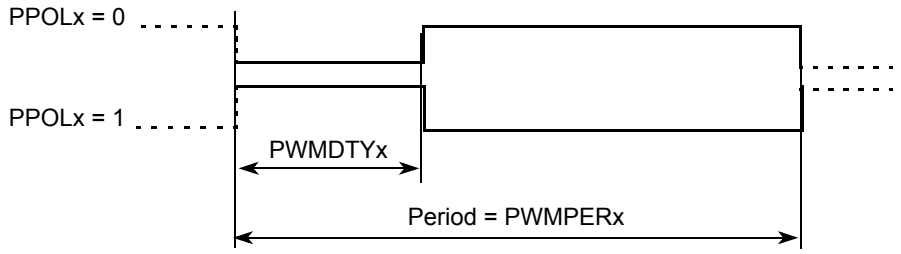


Figure 17-17. PWM Left Aligned Output Waveform

To calculate the output frequency in left aligned output mode for a particular channel, take the selected clock source frequency for the channel (A, B, SA, or SB) and divide it by the value in the period register for that channel.

- PWMx Frequency = Clock (A, B, SA, or SB) / PWMPERx
- PWMx Duty Cycle (high time as a% of period):
  - Polarity = 0 (PPOLx = 0)

    Duty Cycle = [(PWMPERx-PWMDTYx)/PWMPERx] * 100%
  - Polarity = 1 (PPOLx = 1)

    Duty Cycle = [PWMDTYx / PWMPERx] * 100%

As an example of a left aligned output, consider the following case:

Clock Source = bus clock, where bus clock = 10 MHz (100 ns period)

PPOLx = 0

PWMPERx = 4

PWMDTYx = 1

PWMx Frequency = 10 MHz/4 = 2.5 MHz

PWMx Period = 400 ns

PWMx Duty Cycle = 3/4 *100% = 75%

The output waveform generated is shown in Figure 17-18.

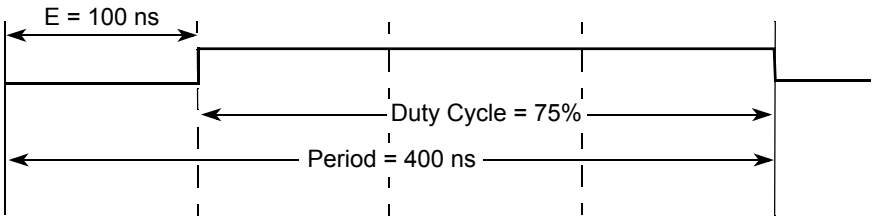


Figure 17-18. PWM Left Aligned Output Example Waveform

20.4.1.5 Repeated START Signal

As shown in Figure 20-10, a repeated START signal is a START signal generated without first generating a STOP signal to terminate the communication. This is used by the master to communicate with another slave or with the same slave in different mode (transmit/receive mode) without releasing the bus.

20.4.1.6 Arbitration Procedure

The Inter-IC bus is a true multi-master bus that allows more than one master to be connected on it. If two or more masters try to control the bus at the same time, a clock synchronization procedure determines the bus clock, for which the low period is equal to the longest clock low period and the high is equal to the shortest one among the masters. The relative priority of the contending masters is determined by a data arbitration procedure, a bus master loses arbitration if it transmits logic 1 while another master transmits logic 0. The losing masters immediately switch over to slave receive mode and stop driving SDA output. In this case the transition from master to slave mode does not generate a STOP condition. Meanwhile, a status bit is set by hardware to indicate loss of arbitration.

20.4.1.7 Clock Synchronization

Because wire-AND logic is performed on SCL line, a high-to-low transition on SCL line affects all the devices connected on the bus. The devices start counting their low period and as soon as a device's clock has gone low, it holds the SCL line low until the clock high state is reached.However, the change of low to high in this device clock may not change the state of the SCL line if another device clock is within its low period. Therefore, synchronized clock SCL is held low by the device with the longest low period. Devices with shorter low periods enter a high wait state during this time (see Figure 20-11). When all devices concerned have counted off their low period, the synchronized clock SCL line is released and pulled high. There is then no difference between the device clocks and the state of the SCL line and all the devices start counting their high periods.The first device to complete its high period pulls the SCL line low again.

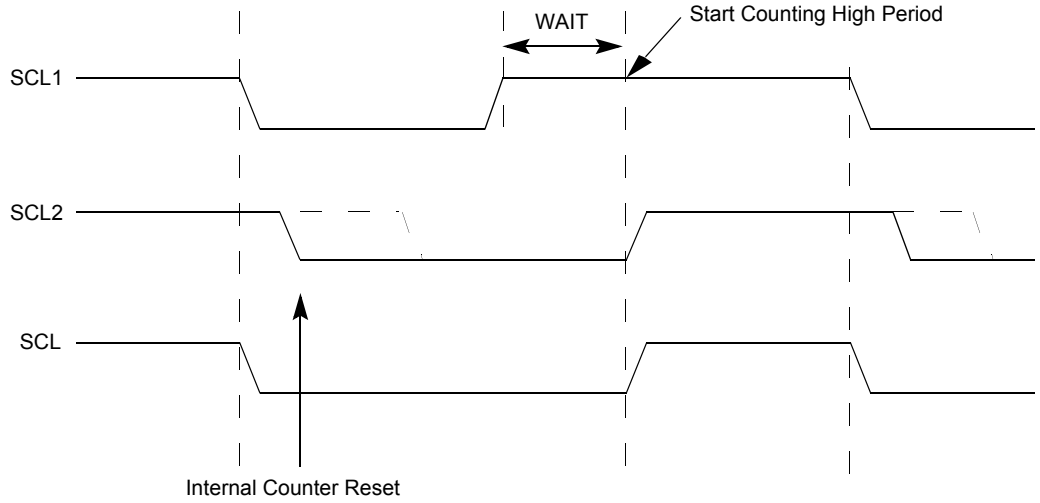


Figure 20-12. IIC-Bus Clock Synchronization

Chapter 21
LIN Physical Layer (S12LINPHYV2)

Table 21-1. Revision History Table

Rev. No. (Item No.)	Date (Submitted By)	Sections Affected	Substantial Change(s)
V02.11	19 Sep 2013	All	- Removed preliminary note. - Fixed grammar and spelling throughout the document.
V02.12	20 Sep 2013	Standby Mode	- Clarified Standby mode behavior.
V02.13	8 Oct 2013	All	- More grammar, spelling, and formating fixes throughout the document.

21.1 Introduction

The LIN (Local Interconnect Network) bus pin provides a physical layer for single-wire communication in automotive applications. The LIN Physical Layer is designed to meet the LIN Physical Layer 2.2 specification from LIN consortium.

21.1.1 Features

The LIN Physical Layer module includes the following distinctive features:

- Compliant with LIN Physical Layer 2.2 specification.
- Compliant with the SAE J2602-2 LIN standard.
- Standby mode with glitch-filtered wake-up.
- Slew rate selection optimized for the baud rates: 10.4 kbit/s, 20 kbit/s and Fast Mode (up to 250 kbit/s).
- Switchable 34 kΩ/330 kΩ pullup resistors (in shutdown mode, 330 kΩ only)
- Current limitation for LIN Bus pin falling edge.
- Overcurrent protection.
- LIN TxD-dominant timeout feature monitoring the LPTxD signal.
- Automatic transmitter shutdown in case of an overcurrent or TxD-dominant timeout.
- Fulfills the OEM “Hardware Requirements for LIN (CAN and FlexRay) Interfaces in Automotive Applications” v1.3.

The LIN transmitter is a low-side MOSFET with current limitation and overcurrent transmitter shutdown. A selectable internal pullup resistor with a serial diode structure is integrated, so no external pullup components are required for the application in a slave node. To be used as a master node, an external

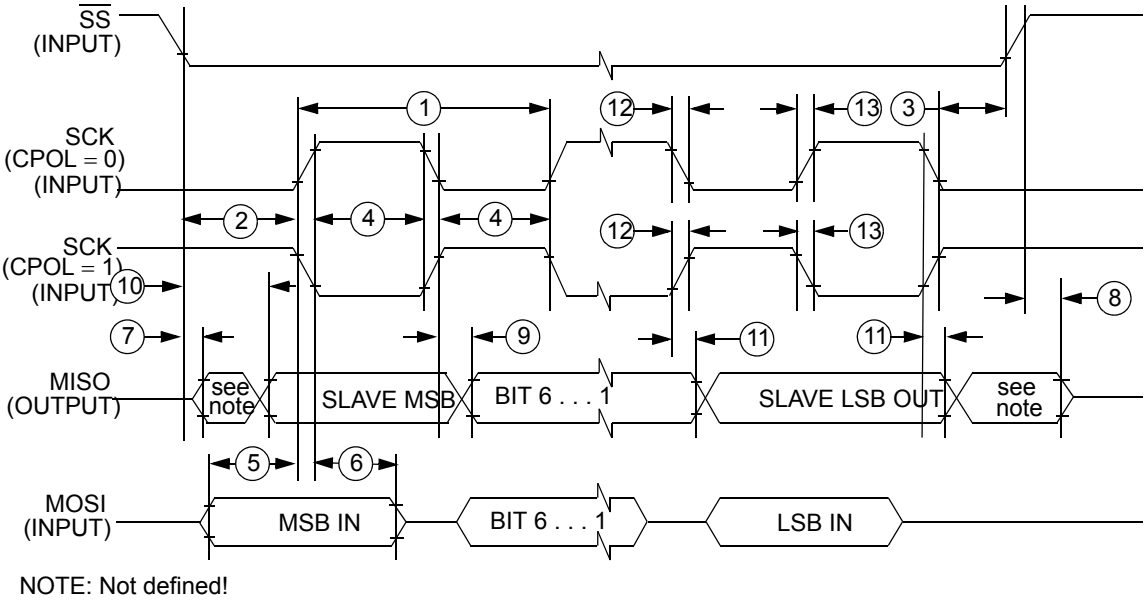


Figure K-3. SPI Slave Timing (CPHA=0)

In Figure K-4 the timing diagram for slave mode with transmission format CPHA=1 is depicted.

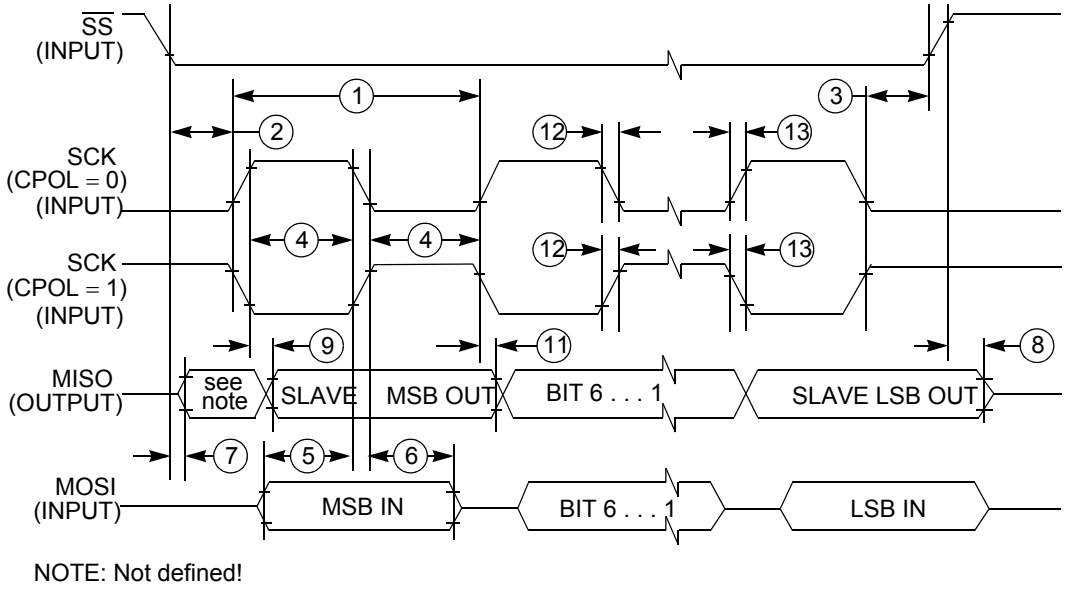


Figure K-4. SPI Slave Timing (CPHA=1)

O.21 0x0800-0x083F CAN0

Address	Name		Bit 7	6	5	4	3	2	1	Bit 0
0x0800	CANCTL0	R	RXFRM	RXACT	CSWAI	SYNCH	TIME	WUPE	SLPRQ	INITRQ
		W								
0x0802	CANBTR0	R	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0
		W								
0x0803	CANBTR1	R	SAMP	TSEG22	TSEG21	TSEG20	TSEG13	TSEG12	TSEG11	TSEG10
		W								
0x0804	CANRFLG	R	WUPIF	CSCIF	RSTAT1	RSTAT0	TSTAT1	TSTAT0	OVRIF	RXF
		W								
0x0805	CANRIER	R	WUPIE	CSCIE	RSTATE1	RSTATE0	TSTATE1	TSTATE0	OVRIE	RXFIE
		W								
0x0806	CANTFLG	R	0	0	0	0	0	TXE2	TXE1	TXE0
		W								
0x0807	CANTIER	R	0	0	0	0	0	TXEIE2	TXEIE1	TXEIE0
		W								
0x0808	CANTARQ	R	0	0	0	0	0	ABTRQ2	ABTRQ1	ABTRQ0
		W								
0x0809	CANTAAK	R	0	0	0	0	0	ABTAK2	ABTAK1	ABTAK0
		W								
0x080A	CANTBSEL	R	0	0	0	0	0	TX2	TX1	TX0
		W								
0x080B	CANIDAC	R	0	0	IDAM1	IDAM0	0	IDHIT2	IDHIT1	IDHIT0
		W								
0x080C	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x080E	CANRXERR	R	RXERR7	RXERR6	RXERR5	RXERR4	RXERR3	RXERR2	RXERR1	RXERR0
		W								
0x080F	CANTXERR	R	TXERR7	TXERR6	TXERR5	TXERR4	TXERR3	TXERR2	TXERR1	TXERR0
		W								
0x0810-0x0813	CANIDAR0–3	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		W								
0x0814-0x0817	CANIDMRx	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
		W								
0x0818-0x081B	CANIDAR4–7	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		W								
0x081C-0x081F	CANIDMR4–7	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
		W								
0x0820-0x082F	CANRXFG	R	See Section 13.3.3, “Programmer’s Model of Message Storage”							
		W								