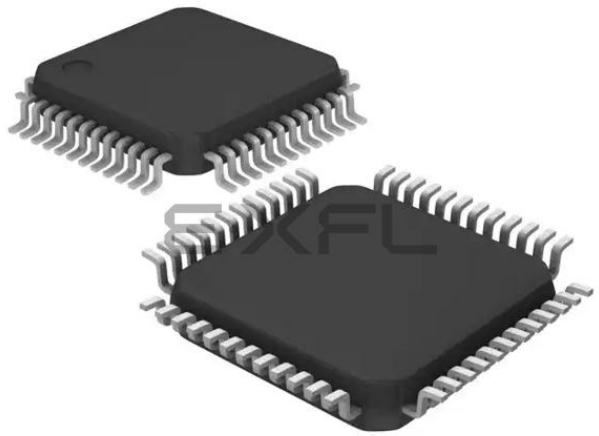


**Welcome to E-XFL.COM**

## 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 | CANbus, I²C, SCI, SPI |
| Peripherals | DMA, POR, PWM, WDT |
| Number of I/O | 28 |
| Program Memory Size | 128KB (128K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | 2K x 8 |
| RAM Size | 8K x 8 |
| Voltage - Supply (Vcc/Vdd) | 3.5V ~ 40V |
| Data Converters | A/D 10x10b; D/A 1x8b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 125°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 48-LQFP |
| Supplier Device Package | 48-LQFP (7x7) |
| Purchase URL | https://www.e-xfl.com/product-detail/nxp-semiconductors/s912zvc12f0mlf |

## 1.6 Family Memory Map

Table 1-2.shows the MC9S12ZVC-Family register memory map.

**Table 1-2. Device Register Memory Map**

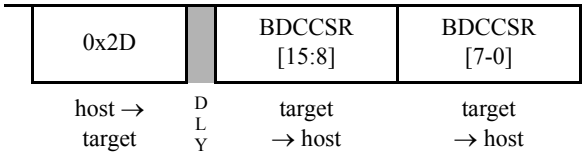
| Address | Module | Size (Bytes) |
|---|---|---|
| 0x0000–0x0003 | ID Register | 4 |
| 0x0004–0x000F | Reserved | 12 |
| 0x0010–0x001F | INT | 16 |
| 0x0020–0x006F | Reserved | 80 |
| 0x0070–0x008F | MMC | 32 |
| 0x0090–0x00FF | MMC Reserved | 112 |
| 0x0100–0x017F | DBG | 128 |
| 0x0180–0x01FF | Reserved | 128 |
| 0x0200–0x037F | PIM | 380 |
| 0x0380–0x039F | FTMRZ | 32 |
| 0x03A0–0x03BF | Reserved | 32 |
| 0x03C0–0x03CF | RAM ECC | 16 |
| 0x03D0–0x03FF | Reserved | 48 |
| 0x0400–0x042F | TIM1 | 48 |
| 0x0430–0x047F | Reserved | 48 |
| 0x0480–0x04AF | PWM0 | 48 |
| 0x04B0–0x04FF | Reserved | 80 |
| 0x0500–0x052F | PWM1 | 48 |
| 0x0530–0x05BF | Reserved | 144 |
| 0x05C0–0x05EF | TIM0 | 48 |
| 0x05F0–0x05FF | Reserved | 16 |
| 0x0600–0x063F | ADC0 | 64 |
| 0x0640–0x067F | Reserved | 64 |
| 0x0680–0x0687 | DAC | 8 |
| 0x0688–0x068F | Reserved | 128 |
| 0x0690–0x0697 | ACMP0 | 8 |
| 0x0698–0x069F | ACMP1 | 8 |
| 0x06A0–0x06BF | Reserved | 32 |
| 0x06C0–0x06DF | CPMU | 32 |
| 0x06E0–0x06EF | Reserved | 16 |
| 0x06F0–0x06F7 | BATS | 8 |
| 0x06F8–0x06FF | Reserved | 8 |
| 0x0700–0x0707 | SCI0 | 8 |
| 0x0708–0x070F | Reserved | 8 |
| 0x0710–0x0717 | SCI1 | 8 |

**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.

### 3.4.4.14 READ_BDCCSR

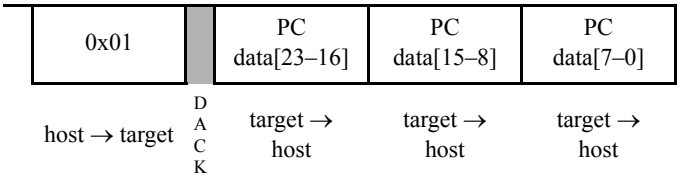
**Read BDCCSR Status Register** **Always Available**




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

### 3.4.4.15 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 returns the address of the next instruction to be executed on returning from active BDM. Thus following a write to the PC in active BDM, a SYNC_PC returns that written value.

### 4.4.2 Illegal Accesses

The S12ZMMC module monitors all memory traffic for illegal accesses. See Table 4-8 for a complete list of all illegal accesses.

**Table 4-8. Illegal memory accesses**

| | | S12ZCPU | S12ZBDC | ADC |
|---|---|---|---|---|
| Register space | Read access | ok | ok | illegal access |
| | Write access | ok | ok | illegal access |
| | Code execution | illegal access | | |
| RAM | Read access | ok | ok | ok |
| | Write access | ok | ok | ok |
| | Code execution | ok | | |
| EEPROM | Read access | ok[1] | ok[1] | ok[1] |
| | Write access | illegal access | illegal access | illegal access |
| | Code execution | ok[1] | | |
| Reserved Space | Read access | ok | ok | illegal access |
| | Write access | only permitted in SS mode | ok | illegal access |
| | Code execution | illegal access | | |
| Reserved Read-only Space | Read access | ok | ok | illegal access |
| | Write access | illegal access | illegal access | illegal access |
| | Code execution | illegal access | | |
| NVM IFR | Read access | ok[1] | ok[1] | illegal access |
| | Write access | illegal access | illegal access | illegal access |
| | Code execution | illegal access | | |
| Program NVM | Read access | ok[1] | ok[1] | ok[1] |
| | Write access | illegal access | illegal access | illegal access |
| | Code execution | ok[1] | | |
| Unmapped Space | Read access | illegal access | illegal access | illegal access |
| | Write access | illegal access | illegal access | illegal access |
| | Code execution | illegal access | | |

1 Unsupported NVM accesses during NVM command execution (“collisions”), are treated as illegal accesses.

Illegal accesses are reported in several ways:

- All illegal accesses performed by the S12ZCPU trigger machine exceptions.
- All illegal accesses performed through the S12ZBDC interface, are captured in the ILLACC bit of the BDCCSRL register.

### 6.1.4 Modes of Operation

The DBG module can be used in all MCU functional modes.

The DBG module can issue breakpoint requests to force the device to enter active BDM or an SWI ISR. The BDC BACKGROUND command is also handled by the DBG to force the device to enter active BDM. When the device enters active BDM through a BACKGROUND command with the DBG module armed, the DBG remains armed.

### 6.1.5 Block Diagram

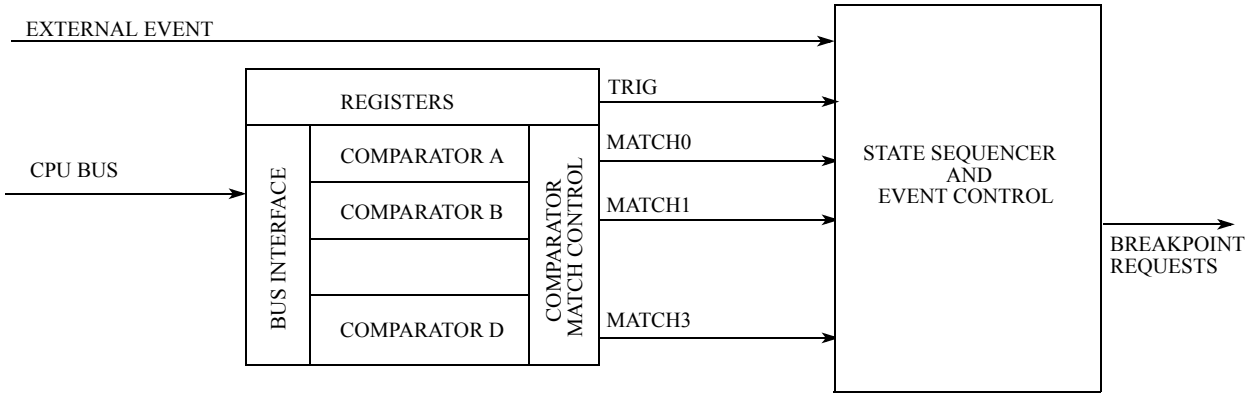



**Figure 6-1. Debug Module Block Diagram**

## 6.2 External Signal Description

### 6.2.1 External Event Input

The DBG module features an external event input signal, DBGEEV. The mapping of this signal to a device pin is specified in the device specific documentation. This function can be enabled and configured by the EEVE field in the DBGC1 control register. This signal is input only and allows an external event to force a state sequencer transition. With the external event function enabled, a falling edge at the external event pin constitutes an event. Rising edges have no effect. The maximum frequency of events is half the internal core bus frequency. The function is explained in the EEVE field description.

**NOTE**

Due to input pin synchronization circuitry, the DBG module sees external events 2 bus cycles after they occur at the pin. Thus an external event occurring less than 2 bus cycles before arming the DBG module is perceived to occur whilst the DBG is armed.

**Table 8-1. CPMURFLG Field Descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>OMRF | **Oscillator Clock Monitor Reset Flag** — OMRF is set to 1 when a loss of oscillator (crystal) clock occurs. Refer to8.5.3, "Oscillator Clock Monitor Reset" for details.This flag can only be cleared by writing a 1. Writing a 0 has no effect.<br>0 Loss of oscillator clock reset has not occurred.<br>1 Loss of oscillator clock reset has occurred. |
| 0<br>PMRF | **PLL Clock Monitor Reset Flag** — PMRF is set to 1 when a loss of PLL clock occurs. This flag can only be cleared by writing a 1. Writing a 0 has no effect.<br>0 Loss of PLL clock reset has not occurred.<br>1 Loss of PLL clock reset has occurred. |

### 8.3.2.2 S12CPMU_UHV_V7 Synthesizer Register (CPMUSYNR)

The CPMUSYNR register controls the multiplication factor of the PLL and selects the VCO frequency range.

Module Base + 0x0004

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | VCOFRQ[1:0] | | SYNDIV[5:0] | | | | | |
| Reset | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |

**Figure 8-5. S12CPMU_UHV_V7 Synthesizer Register (CPMUSYNR)**

Read: Anytime

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

**NOTE**

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

If PLL has locked (LOCK=1) $f_{VCO} = 2 \times f_{REF} \times (SYNDIV + 1)$

**NOTE**

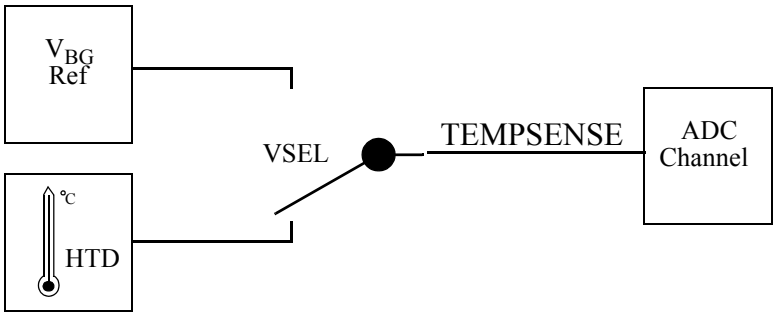
$f_{VCO}$ must be within the specified VCO frequency lock range. Bus frequency $f_{bus}$ must not exceed the specified maximum.

The VCOFRQ[1:0] bits are used to configure the VCO gain for optimal stability and lock time. For correct PLL operation the VCOFRQ[1:0] bits have to be selected according to the actual target VCOCLK

**Table 8-16. CPMUHTCTL Field Descriptions**

| Field | Description |
|---|---|
| 5<br>VSEL | **Voltage Access Select Bit** — If set, the bandgap reference voltage $V_{BG}$ can be accessed internally (i.e. multiplexed to an internal Analog to Digital Converter channel). If not set, the die temperature proportional voltage $V_{HT}$ of the temperature sensor can be accessed internally. See device level specification for connectivity. For any of these access the HTE bit must be set.<br>0 An internal temperature proportional voltage $V_{HT}$ can be accessed internally.<br>1 Bandgap reference voltage $V_{BG}$ can be accessed internally. |
| 3<br>HTE | **High Temperature Sensor/Bandgap Voltage Enable Bit** — This bit enables the high temperature sensor and bandgap voltage amplifier.<br>0 The temperature sensor and bandgap voltage amplifier is disabled.<br>1 The temperature sensor and bandgap voltage amplifier is enabled. |
| 2<br>HTDS | **High Temperature Detect Status Bit** — This read-only status bit reflects the temperature status. Writes have no effect.<br>0 Junction Temperature is below level $T_{HTID}$ or RPM.<br>1 Junction Temperature is above level $T_{HTIA}$ and FPM. |
| 1<br>HTIE | **High Temperature Interrupt Enable Bit**<br>0 Interrupt request is disabled.<br>1 Interrupt will be requested whenever HTIF is set. |
| 0<br>HTIF | **High Temperature Interrupt Flag** — HTIF is set to 1 when HTDS status bit changes. This flag can only be cleared by writing a 1.<br>Writing a 0 has no effect. If enabled (HTIE=1), HTIF causes an interrupt request.<br>0 No change in HTDS bit.<br>1 HTDS bit has changed. |

**Figure 8-18. Voltage Access Select**



Several examples of PLL divider settings are shown in Table 8-32. The following rules help to achieve optimum stability and shortest lock time:

- Use lowest possible $f_{VCO}$ / $f_{REF}$ ratio (SYNDIV value).
- Use highest possible REFCLK frequency $f_{REF}$.

**Table 8-32. Examples of PLL Divider Settings**

| $f_{osc}$ | REFDIV[3:0] | $f_{REF}$ | REFFRQ[1:0] | SYNDIV[5:0] | $f_{VCO}$ | VCOFRQ[1:0] | POSTDIV[4:0] | $f_{PLL}$ | $f_{bus}$ |
|---|---|---|---|---|---|---|---|---|---|
| off | $00 | 1MHz | 00 | $18 | 50MHz | 01 | $03 | 12.5MHz | 6.25MHz |
| off | $00 | 1MHz | 00 | $18 | 50MHz | 01 | $00 | 50MHz | 25MHz |
| 4MHz | $00 | 4MHz | 01 | $05 | 48MHz | 00 | $00 | 48MHz | 24MHz |

The phase detector inside the PLL compares the feedback clock (FBCLK = VCOCLK/(SYNDIV+1)) with the reference clock (REFCLK = (IRC1M or OSCCLK)/(REFDIV+1)). Correction pulses are generated based on the phase difference between the two signals. The loop filter alters the DC voltage on the internal filter capacitor, based on the width and direction of the correction pulse which leads to a higher or lower VCO frequency.

The user must select the range of the REFCLK frequency (REFFRQ[1:0] bits) and the range of the VCOCLK frequency (VCOFRQ[1:0] bits) to ensure that the correct PLL loop bandwidth is set.

The lock detector compares the frequencies of the FBCLK and the REFCLK. Therefore the speed of the lock detector is directly proportional to the reference clock frequency. The circuit determines the lock condition based on this comparison. So e.g. a failure in the reference clock will cause the PLL not to lock.

If PLL LOCK interrupt requests are enabled, the software can wait for an interrupt request and for instance check the LOCK bit. If interrupt requests are disabled, software can poll the LOCK bit continuously (during PLL start-up) or at periodic intervals. In either case, only when the LOCK bit is set, the VCOCLK will have stabilized to the programmed frequency.

- The LOCK bit is a read-only indicator of the locked state of the PLL.
- The LOCK bit is set when the VCO frequency is within the tolerance, $\Delta_{Lock}$, and is cleared when the VCO frequency is out of the tolerance, $\Delta_{unl}$.
- Interrupt requests can occur if enabled (LOCKIE = 1) when the lock condition changes, toggling the LOCK bit.

In case of loss of reference clock (e.g. IRCCLK) the PLL will not lock or if already locked, then it will unlock. The frequency of the VCOCLK will be very low and will depend on the value of the VCOFRQ[1:0] bits.

## 9.4 Memory Map and Register Definition

This section provides a detailed description of all registers accessible in the ADC12B_LBA.

### 9.4.1 Module Memory Map

Figure 9-3 gives an overview of all ADC12B_LBA registers.

**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.

| Address | Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0000 | ADCCTL_0 | R | ADC_EN | ADC_SR | FRZ_MOD | SWAI | ACC_CFG[1:0] | | STR_SEQA | MOD_CFG |
| | | W | | | | | | | | |
| 0x0001 | ADCCTL_1 | R | CSL_BMOD | RVL_BMOD | SMOD_ACC | AUT_RSTA | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0002 | ADCSTS | R | CSL_SEL | RVL_SEL | DBECC_ERR | Reserved | READY | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0003 | ADCTIM | R | 0 | PRS[6:0] | | | | | | |
| | | W | | | | | | | | |
| 0x0004 | ADCFMT | R | DJM | 0 | 0 | 0 | 0 | SRES[2:0] | | |
| | | W | | | | | | | | |
| 0x0005 | ADCFLWCTL | R | SEQA | TRIG | RSTA | LDOK | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0006 | ADCEIE | R | IA_EIE | CMD_EIE | EOL_EIE | Reserved | TRIG_EIE | RSTAR_EIE | LDOK_EIE | 0 |
| | | W | | | | | | | | |
| 0x0007 | ADCIE | R | SEQAD_IE | CONIF_OIE | Reserved | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0008 | ADCEiF | R | IA_EIF | CMD_EIF | EOL_EIF | Reserved | TRIG_EIF | RSTAR_EIF | LDOK_EIF | 0 |
| | | W | | | | | | | | |
| 0x0009 | ADCIF | R | SEQAD_IF | CONIF_OIF | Reserved | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x000A | ADCCONIE_0 | R | CON_IE[15:8] | | | | | | | |
| | | W | | | | | | | | |
| 0x000B | ADCCONIE_1 | R | CON_IE[7:1] | | | | | | | EOL_IE |
| | | W | | | | | | | | |
| 0x000C | ADCCONIF_0 | R | CON_IF[15:8] | | | | | | | |
| | | W | | | | | | | | |
| 0x000D | ADCCONIF_1 | R | CON_IF[7:1] | | | | | | | EOL_IF |
| | | W | | | | | | | | |
| 0x000E | ADCIMDRI_0 | R | CSL_IMD | RVL_IMD | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x000F | ADCIMDRI_1 | R | 0 | 0 | RIDX_IMD[5:0] | | | | | |
| | | W | | | | | | | | |

= Unimplemented or Reserved

**Figure 9-3. ADC12B_LBA Register Summary (Sheet 1 of 3)**

### 9.4.2.10 ADC Interrupt Flag Register (ADCIF)

After being set any of these bits can be cleared by writing a value of 1’b1 or via ADC soft-reset (bit ADC_SR). All bits are cleared if bit ADC_EN is clear. Writing any flag with value 1’b0 does not clear the flag. Writing any flag with value 1’b1 does not set the flag.

Module Base + 0x0009

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | SEQAD_IF | CONIF_OIF | Reserved | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 9-13. ADC Interrupt Flag Register (ADCIF)**

Read: Anytime

Write: Anytime

**Table 9-14. ADCIF Field Descriptions**

| Field | Description |
|---|---|
| 7<br>SEQAD_IF | **Conversion Sequence Abort Done Interrupt Flag** — This flag is set when the Sequence Abort Event has been executed except the Sequence Abort Event occurred by hardware in order to be able to enter MCU Stop Mode or Wait Mode with bit SWAI set.This flag is also not set if the Sequence Abort request occurs during execution of the last conversion command of a CSL and bit STR_SEQA being set.<br>0 No conversion sequence abort request occurred.<br>1 A conversion sequence abort request occurred. |
| 6<br>CONIF_OIF | **ADCCONIF Register Flags Overrun Interrupt Flag** — This flag indicates if an overrun situation occurred for one of the CON_IF[15:1] flags or for the EOL_IF flag. In RVL single buffer mode (RVL_BMOD clear) an overrun of the EOL_IF flag is not indicated (For more information please see Note below).<br>0 No ADCCONIF Register Flag overrun occurred.<br>1 ADCCONIF Register Flag overrun occurred. |

**NOTE**

In RVL double buffer mode a conversion interrupt flag (CON_IF[15:1]) or End Of List interrupt flag (EOL_IF) overrun is detected if one of these bits is set when it should be set again due to conversion command execution.

In RVL single buffer mode a conversion interrupt flag (CON_IF[15:1]) overrun is detected only. The overrun is detected if any of the conversion interrupt flags (CON_IF[15:1]) is set while the first conversion result of a CSL is stored (result of first conversion from top of CSL is stored).

#### 9.5.3.2.6 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.

### 14.4.3 Data Format

The SCI uses the standard NRZ mark/space data format. When Infrared is enabled, the SCI uses RZI data format where zeroes are represented by light pulses and ones remain low. See Figure 14-15 below.

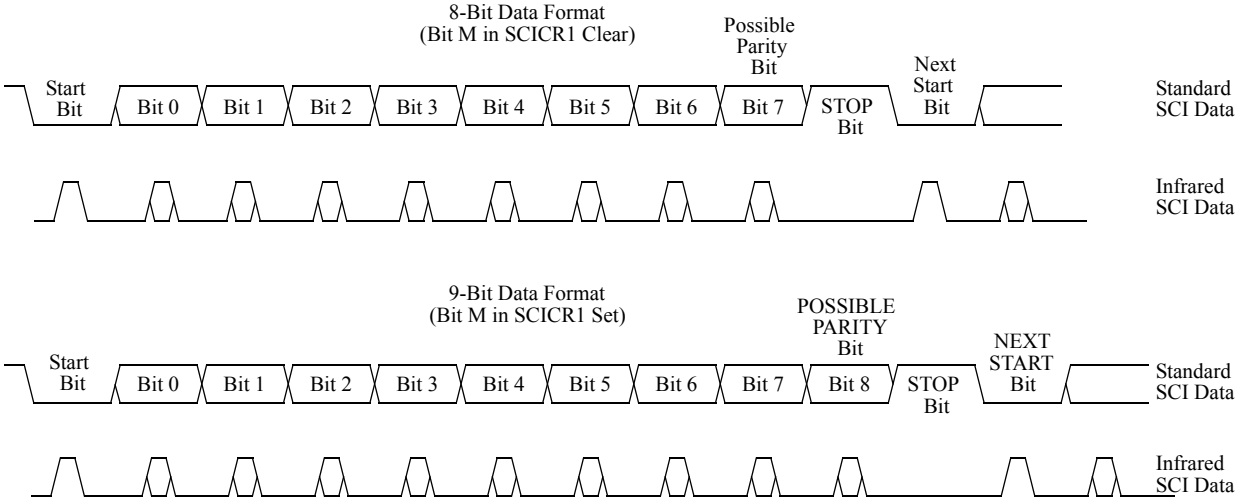



**Figure 14-15. SCI Data Formats**

Each data character is contained in a frame that includes a start bit, eight or nine data bits, and a stop bit. Clearing the M bit in SCI control register 1 configures the SCI for 8-bit data characters. A frame with eight data bits has a total of 10 bits. Setting the M bit configures the SCI for nine-bit data characters. A frame with nine data bits has a total of 11 bits.

**Table 14-14. Example of 8-Bit Data Formats**

| Start Bit | Data Bits | Address Bits | Parity Bits | Stop Bit |
|---|---|---|---|---|
| 1 | 8 | 0 | 0 | 1 |
| 1 | 7 | 0 | 1 | 1 |
| 1 | 7 | 1[1] | 0 | 1 |

[1] The address bit identifies the frame as an address character. See Section 14.4.6.6, “Receiver Wakeup”.

When the SCI is configured for 9-bit data characters, the ninth data bit is the T8 bit in SCI data register high (SCIDRH). It remains unchanged after transmission and can be used repeatedly without rewriting it. A frame with nine data bits has a total of 11 bits.

**Table 14-15. Example of 9-Bit Data Formats**

| Start Bit | Data Bits | Address Bits | Parity Bits | Stop Bit |
|---|---|---|---|---|
| 1 | 9 | 0 | 0 | 1 |
| 1 | 8 | 0 | 1 | 1 |
| 1 | 8 | 1[1] | 0 | 1 |

### 16.4.1.2 Slave Address Transmission

The first byte of data transfer immediately after the START signal is the slave address transmitted by the master. This is a seven-bit calling address followed by a R/W bit. The R/W bit tells the slave the desired direction of data transfer.

1 = Read transfer, the slave transmits data to the master.
0 = Write transfer, the master transmits data to the slave.

If the calling address is 10-bit, another byte is followed by the first byte.Only the slave with a calling address that matches the one transmitted by the master will respond by sending back an acknowledge bit. This is done by pulling the SDA low at the 9th clock (see Figure 16-10).

No two slaves in the system may have the same address. If the IIC bus is master, it must not transmit an address that is equal to its own slave address. The IIC bus cannot be master and slave at the same time.However, if arbitration is lost during an address cycle the IIC bus will revert to slave mode and operate correctly even if it is being addressed by another master.

### 16.4.1.3 Data Transfer

As soon as successful slave addressing is achieved, the data transfer can proceed byte-by-byte in a direction specified by the R/W bit sent by the calling master

All transfers that come after an address cycle are referred to as data transfers, even if they carry sub-address information for the slave device.

Each data byte is 8 bits long. Data may be changed only while SCL is low and must be held stable while SCL is high as shown in Figure 16-10. There is one clock pulse on SCL for each data bit, the MSB being transferred first. Each data byte has to be followed by an acknowledge bit, which is signalled from the receiving device by pulling the SDA low at the ninth clock. So one complete data byte transfer needs nine clock pulses.

If the slave receiver does not acknowledge the master, the SDA line must be left high by the slave. The master can then generate a stop signal to abort the data transfer or a start signal (repeated start) to commence a new calling.

If the master receiver does not acknowledge the slave transmitter after a byte transmission, it means 'end of data' to the slave, so the slave releases the SDA line for the master to generate STOP or START signal.Note in order to release the bus correctly,after no-acknowledge to the master,the slave must be immediately switched to receiver and a following dummy reading of the IBDR is necessary.

### 16.4.1.4 STOP Signal

The master can terminate the communication by generating a STOP signal to free the bus. However, the master may generate a START signal followed by a calling command without generating a STOP signal first. This is called repeated START. A STOP signal is defined as a low-to-high transition of SDA while SCL at logical 1 (see Figure 16-10).

The master can generate a STOP even if the slave has generated an acknowledge at which point the slave must release the bus.

# Chapter 17
# CAN Physical Layer (S12CANPHYV3)

**Table 17-1. Revision History Table**

| Revision Number | Revision Date | Sections Affected | Description of Changes |
|---|---|---|---|
| V02.00 | 05 Nov 2012 | | • Added CPTXD-dominant timeout feature |
| V03.00 | 15 Apr 2013 | | • Made transmit driver (CANH & CANL) independent of CPCHVL condition<br>• Changed CPCLVL condition to disable CANL only<br>• Added mode to cover separation of CANH and CANL drivers<br>• Added configurable wake-up filter |

## 17.1 Introduction

The CAN Physical Layer provides a physical layer for high speed CAN area network communication in automotive applications. It serves as an integrated interface to the CAN bus lines for the internally connected MSCAN controller through the pins CANH, CANL and SPLIT.

The CAN Physical Layer is designed to meet the CAN Physical Layer ISO 11898-2 and ISO 11898-5 standards.

### 17.1.1 Features

The CAN Physical Layer module includes these distinctive features:

- High speed CAN interface for baud rates of up to 1 Mbit/s
- ISO 11898-2 and ISO 11898-5 compliant for 12 V battery systems
- SPLIT pin driver for bus recessive level stabilization
- Low power mode with remote CAN wake-up handled by MSCAN module
- Configurable wake-up pulse filtering
- Over-current shutdown for CANH and CANL
- Voltage monitoring on CANH and CANL
- CPTXD-dominant timeout feature monitoring the CPTXD signal
- Fulfills the OEM “Hardware Requirements for (LIN,) CAN (and FlexRay) Interfaces in Automotive Applications” v1.3

Module Base + 0x00X2

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | ID14 | ID13 | ID12 | ID11 | ID10 | ID9 | ID8 | ID7 |
| Reset: | x | x | x | x | x | x | x | x |

**Figure 18-28. Identifier Register 2 (IDR2) — Extended Identifier Mapping**

**Table 18-28. IDR2 Register Field Descriptions — Extended**

| Field | Description |
|---|---|
| 7-0<br>ID[14:7] | **Extended Format Identifier —** The identifiers consist of 29 bits (ID[28:0]) for the extended format. ID28 is the most significant bit and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number. |

Module Base + 0x00X3

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | ID6 | ID5 | ID4 | ID3 | ID2 | ID1 | ID0 | RTR |
| Reset: | x | x | x | x | x | x | x | x |

**Figure 18-29. Identifier Register 3 (IDR3) — Extended Identifier Mapping**

**Table 18-29. IDR3 Register Field Descriptions — Extended**

| Field | Description |
|---|---|
| 7-1<br>ID[6:0] | **Extended Format Identifier —** The identifiers consist of 29 bits (ID[28:0]) for the extended format. ID28 is the most significant bit and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number. |
| 0<br>RTR | **Remote Transmission Request** — This flag reflects the status of the remote transmission request bit in the CAN frame. In the case of a receive buffer, it indicates the status of the received frame and supports the transmission of an answering frame in software. In the case of a transmit buffer, this flag defines the setting of the RTR bit to be sent.<br>0 Data frame<br>1 Remote frame |

# 18.5 Initialization/Application Information

## 18.5.1 MSCAN initialization

The procedure to initially start up the MSCAN module out of reset is as follows:

1. Assert CANE
2. Write to the configuration registers in initialization mode
3. Clear INITRQ to leave initialization mode

If the configuration of registers which are only writable in initialization mode shall be changed:

1. Bring the module into sleep mode by setting SLPRQ and awaiting SLPAK to assert after the CAN bus becomes idle.
2. Enter initialization mode: assert INITRQ and await INITAK
3. Write to the configuration registers in initialization mode
4. Clear INITRQ to leave initialization mode and continue

## 18.5.2 Bus-Off Recovery

The bus-off recovery is user configurable. The bus-off state can either be left automatically or on user request.

For reasons of backwards compatibility, the MSCAN defaults to automatic recovery after reset. In this case, the MSCAN will become error active again after counting 128 occurrences of 11 consecutive recessive bits on the CAN bus (see the Bosch CAN 2.0 A/B specification for details).

If the MSCAN is configured for user request (BORM set in MSCAN Control Register 1 (CANCTL1)), the recovery from bus-off starts after both independent events have become true:

- 128 occurrences of 11 consecutive recessive bits on the CAN bus have been monitored
- BOHOLD in MSCAN Miscellaneous Register (CANMISC) has been cleared by the user

These two events may occur in any order.

### 21.8.4.2 Single-buffered Transmission without Pause Pulse

This option offers the most up-to-date transmit data. The disadvantage for software is that the preparation of the transmit data has to occur in much less time (less than the length of the calibration pulse, meaning less than 56 unit-time ticks) compared to the double-buffered transmission option.

The software uses the Transmission Complete interrupt (TC) to prepare new data for the next transmission. An example for this case is provided in Figure 21-4 below.

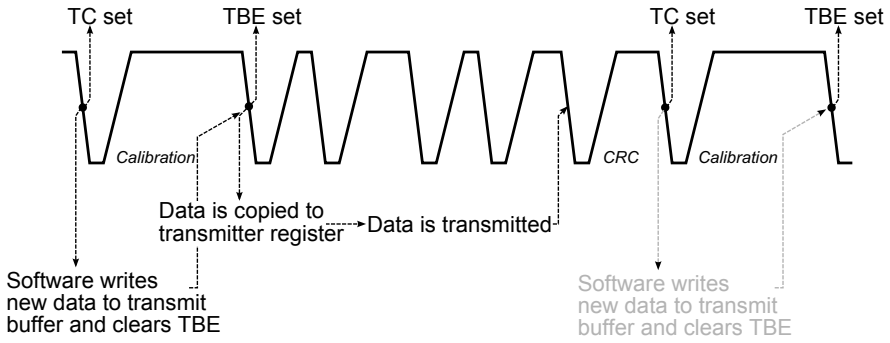



**Figure 21-4. Transmission Complete driven SENT transfer without Pause Pulse**

### 21.8.4.3 Double-buffered Transmission with Pause Pulse

This option is similar to the double-buffered transmission without pause-pulse (for details please refer to Section 21.8.4.1, “Double-buffered Transmission without Pause Pulse). The only difference is that due to the pause pulse the message periods become longer offering even more time for the software to prepare new data.

The software uses the Transmit Buffer Empty interrupt (TBE) to prepare new data for the next transmission. An example for this case is provided in Figure 21-5 below.

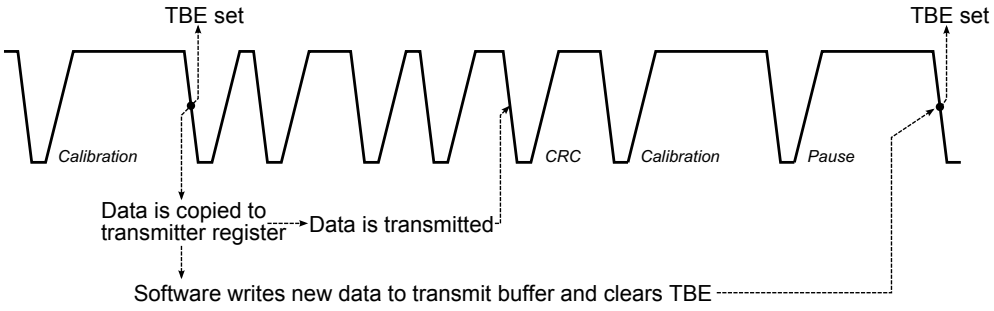



**Figure 21-5. Transmit-Buffer Empty driven SENT transfer with Pause Pulse**

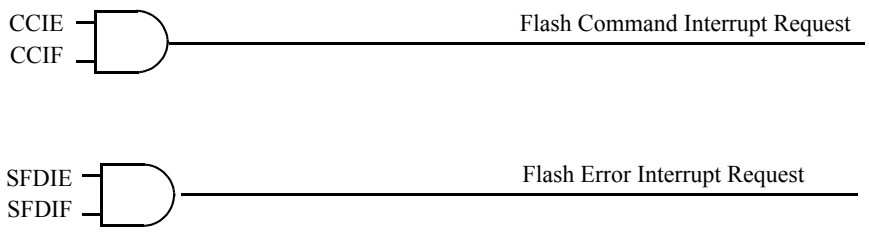



**Figure 22-31. Flash Module Interrupts Implementation**

### 22.4.9 Wait Mode

The Flash module is not affected if the MCU enters wait mode. The Flash module can recover the MCU from wait via the CCIF interrupt (see Section 22.4.8, “Interrupts”).

### 22.4.10 Stop Mode

If a Flash command is active (CCIF = 0) when the MCU requests stop mode, the current Flash operation will be completed before the MCU is allowed to enter stop mode.

## 22.5 Security

The Flash module provides security information to the MCU. The Flash security state is defined by the SEC bits of the FSEC register (see Table 22-10). During reset, the Flash module initializes the FSEC register using data read from the security byte of the Flash configuration field at global address 0xFF_FE0F. The security state out of reset can be permanently changed by programming the security byte assuming that the MCU is starting from a mode where the necessary P-Flash erase and program commands are available and that the upper region of the P-Flash is unprotected. If the Flash security byte is successfully programmed, its new value will take affect after the next MCU reset.

The following subsections describe these security-related subjects:

- Unsecuring the MCU using Backdoor Key Access
- Unsecuring the MCU in Special Single Chip Mode using BDM
- .Mode and Security Effects on Flash Command Availability

### 22.5.1 Unsecuring the MCU using Backdoor Key Access

The MCU may be unsecured by using the backdoor key access feature which requires knowledge of the contents of the backdoor keys (four 16-bit words programmed at addresses 0xFF_FE00-0xFF_FE07). If the KEYEN[1:0] bits are in the enabled state (see <st-blue>Section 22.3.2.2 Flash Security Register (FSEC)), the Verify Backdoor Access Key command (see <st-blue>Section 22.4.7.11 Verify Backdoor Access Key Command) allows the user to present four prospective keys for comparison to the keys stored

In **Table D-2.** the timing characteristics for slave mode are listed.

**Table D-2. SPI Slave Mode Timing Characteristics (Junction Temperature From –40°C To +175°C)**

| Num | Characteristic | Symbol | | | | Unit |
|---|---|---|---|---|---|---|
| | | | Min | Typ | Max | |
| 1 | SCK Frequency | $f_{sck}$ | DC | — | 1/4 | $f_{bus}$ |
| 1 | SCK Period | $t_{sck}$ | 4 | — | ∞ | $t_{bus}$ |
| 2 | Enable Lead Time | $t_{lead}$ | 4 | — | — | $t_{bus}$ |
| 3 | Enable Lag Time | $t_{lag}$ | 4 | — | — | $t_{bus}$ |
| 4 | Clock (SCK) High or Low Time | $t_{wsck}$ | 4 | — | — | $t_{bus}$ |
| 5 | Data Setup Time (Inputs) | $t_{su}$ | 8 | — | — | ns |
| 6 | Data Hold Time (Inputs) | $t_{hi}$ | 8 | — | — | ns |
| 7 | Slave Access Time (time to data active) | $t_a$ | — | — | 20 | ns |
| 8 | Slave MISO Disable Time | $t_{dis}$ | — | — | 22 | ns |
| 9 | Data Valid after SCK Edge | $t_{vsck}$ | — | — | $30 + t_{bus}$ [1] | ns |
| 10 | Data Valid after $\overline{SS}$ fall | $t_{vss}$ | — | — | $30 + t_{bus}$ [1] | ns |
| 11 | Data Hold Time (Outputs) | $t_{ho}$ | 20 | — | — | ns |
| 12 | Rise and Fall Time Inputs | $t_{rfi}$ | — | — | 8 | ns |
| 13 | Rise and Fall Time Outputs | $t_{rfo}$ | — | — | 8 | ns |

[1] $t_{bus}$ added due to internal synchronization delay

| Global Address | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0315–0x031E | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x031F | WOMJ | R | 0 | 0 | 0 | 0 | 0 | 0 | WOMJ1 | WOMJ0 |
| | | W | | | | | | | | |
| 0x0320–0x032F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0330 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0331 | PTIL | R | 0 | 0 | 0 | 0 | 0 | 0 | PTIL1 | PTIL0 |
| | | W | | | | | | | | |
| 0x0332 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0333 | PTPSL | R | 0 | 0 | 0 | 0 | 0 | 0 | PTPSL1 | PTPSL0 |
| | | W | | | | | | | | |
| 0x0334 | PPSL | R | 0 | 0 | 0 | 0 | 0 | 0 | PPSL1 | PPSL0 |
| | | W | | | | | | | | |
| 0x0335 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0336 | PIEL | R | 0 | 0 | 0 | 0 | 0 | 0 | PIEL1 | PIEL0 |
| | | W | | | | | | | | |
| 0x0337 | PIFL | R | 0 | 0 | 0 | 0 | 0 | 0 | PIFL1 | PIFL0 |
| | | W | | | | | | | | |
| 0x0338–0x0339 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x033A | PTABYPL | R | 0 | 0 | 0 | 0 | 0 | 0 | PTABYPL1 | PTABYPL0 |
| | | W | | | | | | | | |
| 0x033B | PTADIRL | R | 0 | 0 | 0 | 0 | 0 | 0 | PTADIRL1 | PTADIRL0 |
| | | W | | | | | | | | |
| 0x033C | DIENL | R | 0 | 0 | 0 | 0 | 0 | 0 | DIENL1 | DIENL0 |
| | | W | | | | | | | | |
| 0x033D | PTAENL | R | 0 | 0 | 0 | 0 | 0 | 0 | PTAENL1 | PTAENL0 |
| | | W | | | | | | | | |