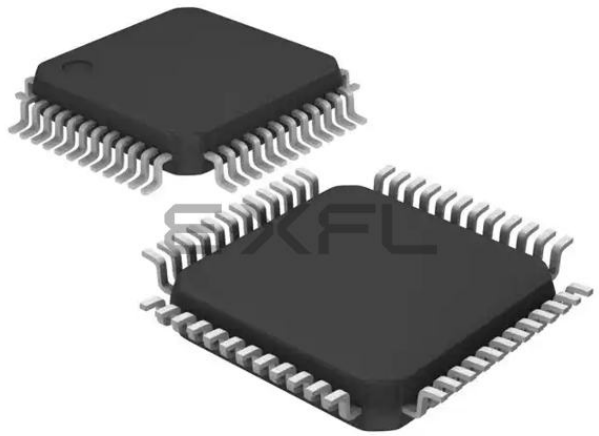

**Welcome to E-XFL.COM**

## 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 | 64KB (64K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | 1K x 8 |
| RAM Size | 4K x 8 |
| Voltage - Supply (Vcc/Vdd) | 3.5V ~ 40V |
| Data Converters | A/D 10x12b; D/A 1x8b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°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/s912zvca64f0clfr |

#### 1.7.16.2 SCK[1:0] signals

This signal is associated with the serial clock SCK functionality of the serial peripheral interface SPI0 and SPI1.

#### 1.7.16.3 MISO[1:0] signals

This signal is associated with the MISO functionality of the serial peripheral interface SPI0 and SPI1. This signal acts as master input during master mode or as slave output during slave mode.

#### 1.7.16.4 MOSI[1:0] signals

This signal is associated with the MOSI functionality of the serial peripheral interface SPI0 and SPI1. This signal acts as master output during master mode or as slave input during slave mode

### 1.7.17 SCI signals

#### 1.7.17.1 RXD[1:0] signals

These signals are associated with the receive functionality of the serial communication interfaces SCI[1:0].

#### 1.7.17.2 TXD[1:0] signals

These signals are associated with the transmit functionality of the serial communication interfaces SCI[1:0].

### 1.7.18 Timer IOC[7:0] signals

The signals IOC0[7:0] and IOC1[3:0] are associated with the input capture or output compare functionality of the timer modules TIM0 and TIM1.

### 1.7.19 PWM[7:0] signals

The signals PWM0[7:0] and PWM1[7:0] are associated with the outputs of the PWM0 and PWM1 modules.

### 1.7.20 IIC signals

#### 1.7.20.1 SDA signal

This signal is associated with the serial data pin of IIC.

#### 1.7.20.2 SCL signal

This signal is associated with the serial clock pin of IIC.

## 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 on chip bandgap reference voltage $V_{BG}$ is measured during production test. $V_{BG}$ has a narrow variation over temperature and external voltage supply. $V_{BG}$ is connected to an internal channel of the ADC module (see Table 1-8). The12-bit left justified ADC conversion result of $V_{BG}$ is stored in the flash IFR for reference, as listed in Table 1-16.

By measuring the voltage $V_{BG}$ 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:

| | |
|---|---|
| Converted AD Input: | Result of the analog to digital conversion of the desired pin |
| Converted Reference: | Result of internal channel conversion |
| Stored Reference: | Value in IFR location |
| n: | ADC resolution (12 bit) |

**NOTE**

The ADC reference voltage $V_{RH}$ must remain at a constant level throughout the conversion process.

The reference voltage $V_{BG}$ is measured under the conditions shown in Table 1-17. The value stored in the IFR is the average of 8 consecutive conversions.

**Table 5-6. INT_CFDATA0–7 Field Descriptions**

| Field | Description |
|---|---|
| 2–0<br>PRIOLVL[2:0] | **Interrupt Request Priority Level Bits** — The PRIOLVL[2:0] bits configure the interrupt request priority level of the associated interrupt request. Out of reset all interrupt requests are enabled at the lowest active level (“1”). Please also refer to Table 5-7 for available interrupt request priority levels.<br>**Note:** Write accesses to configuration data registers of unused interrupt channels are ignored and read accesses return all 0s. For information about what interrupt channels are used in a specific MCU, please refer to the Device Reference Manual for that MCU.<br>**Note:** When non I-bit maskable request vectors are selected, writes to the corresponding INT_CFDATA registers are ignored and read accesses return all 0s. The corresponding vectors do not have configuration data registers associated with them.<br>**Note:** Write accesses to the configuration register for the spurious interrupt vector request (vector base + 0x0001DC) are ignored and read accesses return 0x07 (request is handled by the CPU, PRIOLVL = 7). |

**Table 5-7. Interrupt Priority Levels**

| Priority | PRIOLVL2 | PRIOLVL1 | PRIOLVL0 | Meaning |
|---|---|---|---|---|
| | 0 | 0 | 0 | Interrupt request is disabled |
| low | 0 | 0 | 1 | Priority level 1 |
| | 0 | 1 | 0 | Priority level 2 |
| | 0 | 1 | 1 | Priority level 3 |
| | 1 | 0 | 0 | Priority level 4 |
| | 1 | 0 | 1 | Priority level 5 |
| | 1 | 1 | 0 | Priority level 6 |
| high | 1 | 1 | 1 | Priority level 7 |

## 5.4 Functional Description

The INT module processes all exception requests to be serviced by the CPU module. These exceptions include interrupt vector requests and reset vector requests. Each of these exception types and their overall priority level is discussed in the subsections below.

### 5.4.1 S12Z Exception Requests

The CPU handles both reset requests and interrupt requests. The INT module contains registers to configure the priority level of each I-bit maskable interrupt request which can be used to implement an interrupt priority scheme. This also includes the possibility to nest interrupt requests. A priority decoder is used to evaluate the relative priority of pending interrupt requests.

### 5.4.2 Interrupt Prioritization

After system reset all I-bit maskable interrupt requests are configured to be enabled, are set up to be handled by the CPU and have a pre-configured priority level of 1. Exceptions to this rule are the non-maskable interrupt requests and the spurious interrupt vector request at (vector base + 0x0001DC)

### 6.1.1 Glossary

**Table 6-2. Glossary Of Terms**

| Term | Definition |
|---|---|
| COF | Change Of Flow.<br>Change in the program flow due to a conditional branch, indexed jump or interrupt |
| PC | Program Counter |
| BDM | Background Debug Mode.<br>In this mode CPU application code execution is halted.<br>Execution of BDC “active BDM” commands is possible. |
| BDC | Background Debug Controller |
| WORD | 16-bit data entity |
| CPU | S12Z CPU module |

### 6.1.2 Overview

The comparators monitor the bus activity of the CPU. A single comparator match or a series of matches can generate breakpoints. A state sequencer determines if the correct series of matches occurs. Similarly an external event can generate breakpoints.

### 6.1.3 Features

- Three comparators (A, B, and D)
  - Comparator A compares the full address bus and full 32-bit data bus
  - Comparator A features a data bus mask register
  - Comparators B and D compare the full address bus only
  - Each comparator can be configured to monitor PC addresses or addresses of data accesses
  - Each comparator can select either read or write access cycles
  - Comparator matches can force state sequencer state transitions
- Three comparator modes
  - Simple address/data comparator match mode
  - Inside address range mode, Addmin ≤ Address ≤ Addmax
  - Outside address range match mode, Address < Addmin or Address > Addmax
- State sequencer control
  - State transitions forced by comparator matches
  - State transitions forced by software write to TRIG
  - State transitions forced by an external event
- The following types of breakpoints
  - CPU breakpoint entering active BDM on breakpoint (BDM)
  - CPU breakpoint executing SWI on breakpoint (SWI)
  -

**Table 7-9. Memory access cycles**

| Access type | ECC error | access cycle | Internal operation | Memory content | Error indication |
|---|---|---|---|---|---|
| 1 or 3 byte write, non-aligned 2 byte write | no | 2 | read data from the memory<br>write old + new data to the memory | old + new data | — |
| | single bit | 2 | read data from the memory<br>write corrected + new data to the memory | corrected + new data | SBEEIF |
| | double bit | 2 | read data from the memory<br>ignore write data | unchanged | initiator module is informed |
| read access | no | 1 | read from memory | unchanged | - |
| | single bit | 1[1] | read data from the memory<br>write corrected data back to memory | corrected data | SBEEIF |
| | double bit | 1 | read from memory | unchanged | data mark as invalid |

1 The next back to back read access to the memory will be delayed by one clock cycle

The single bit ECC error generates an interrupt when enabled. The double bit ECC errors are reported by the SRAM_ECC module, but handled at MCU level. For more information, see the MMC description.

### 7.3.1 Aligned 2 and 4 Byte Memory Write Access

During an aligned 2 or 4 byte memory write access, no ECC check is performed. The internal ECC logic generates the new ECC value based on the write data and writes the data words together with the generated ECC values into the memory.

### 7.3.2 Other Memory Write Access

Other types of write accesses are separated into a read-modify-write operation. During the first cycle, the logic reads the data from the memory and performs an ECC check. If no ECC errors were detected then the logic generates the new ECC value based on the read and write data and writes the new data word together with the new ECC value into the memory. If required both 2 byte data words are updated.

If the module detects a single bit ECC error during the read cycle, then the logic generates the new ECC value based on the corrected read and new write read. In the next cycle, the new data word and the new ECC value are written into the memory. If required both 2 byte data words are updated. The SBEEIF bit is set. Hence, the single bit ECC error was corrected by the write access. Figure 7-9 shows an example of a 2 byte non-aligned memory write access.

If the module detects a double bit ECC error during the read cycle, then the write access to the memory is blocked and the initiator module is informed about the error.

access is always a 2 byte aligned memory access, so that no ECC check is performed and no single or double bit ECC error indication is activated.

#### 7.3.7.2 ECC Debug Memory Read Access

Writing one to the ECCDR bit performs a debug read access from the memory address defined by register DPTR. If the ECCDR bit is cleared then the register DDATA contains the uncorrected read data from the memory. The register DECC contains the ECC value read from the memory. Independent of the ECCDRR register bit setting, the debug read access will not perform an automatic ECC repair during read access. During the debug read access no ECC check is performed, so that no single or double bit ECC error indication is activated.

If the ECCDW and the ECCDR bits are set at the same time, then only the debug write access is performed.

- Frequency trimming
  (A factory trim value for 1MHz is loaded from Flash Memory into the IRCTRIM register after reset, which can be overwritten by application if required)
- Temperature Coefficient (TC) trimming.
  (A factory trim value is loaded from Flash Memory into the IRCTRIM register to turn off TC trimming after reset. Application can trim the TC if required by overwriting the IRCTRIM register).

Other features of the S12CPMU_UHV_V7 include

- Oscillator clock monitor to detect loss of crystal
- Autonomous periodical interrupt (API)
- Bus Clock Generator
  - Clock switch to select either PLLCLK or external crystal/resonator as source of the Bus Clock
  - PLLCLK divider to adjust system speed
- System Reset generation from the following possible sources:
  - Power-on reset (POR)
  - Low-voltage reset (LVR)
  - COP system watchdog, COP reset on time-out, windowed COP
  - Loss of oscillation (Oscillator clock monitor fail)
  - Loss of PLL clock (PLL clock monitor fail)
  - External pin $\overline{\text{RESET}}$

frequency as shown in Table 8-2. Setting the VCOFRQ[1:0] bits incorrectly can result in a non functional PLL (no locking and/or insufficient stability).

**Table 8-2. VCO Clock Frequency Selection**

| VCOCLK Frequency Ranges | VCOFRQ[1:0] |
|---|---|
| 32MHz <= $f_{VCO}$ <= 48MHz | 00 |
| 48MHz < $f_{VCO}$ <= 64MHz | 01 |
| Reserved | 10 |
| Reserved | 11 |

### 8.3.2.3 S12CPMU_UHV_V7 Reference Divider Register (CPMUREFDIV)

The CPMUREFDIV register provides a finer granularity for the PLL multiplier steps when using the external oscillator as reference.

Module Base + 0x0005

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | REFFRQ[1:0] | | 0 | 0 | REFDIV[3:0] | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

**Figure 8-6. S12CPMU_UHV_V7 Reference Divider Register (CPMUREFDIV)**

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.

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

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

The REFFRQ[1:0] bits are used to configure the internal PLL filter for optimal stability and lock time. For correct PLL operation the REFFRQ[1:0] bits have to be selected according to the actual REFCLK frequency as shown in Table 8-3.

If IRC1M is selected as REFCLK (OSCE=0) the PLL filter is fixed configured for the 1MHz <= $f_{REF}$ <= 2MHz range. The bits can still be written but will have no effect on the PLL filter configuration.

For OSCE=1, setting the REFFRQ[1:0] bits incorrectly can result in a non functional PLL (no locking and/or insufficient stability).

**Table 8-15. COP Watchdog Rates if COPOSCSEL1=1.**

| CR2 | CR1 | CR0 | COPCLK Cycles to time-out (COPCLK is ACLK divided by 2) |
|---|---|---|---|
| 0 | 0 | 0 | COP disabled |
| 0 | 0 | 1 | $2^{7}$ |
| 0 | 1 | 0 | $2^{9}$ |
| 0 | 1 | 1 | $2^{11}$ |
| 1 | 0 | 0 | $2^{13}$ |
| 1 | 0 | 1 | $2^{15}$ |
| 1 | 1 | 0 | $2^{16}$ |
| 1 | 1 | 1 | $2^{17}$ |

## 8.4.6 System Clock Configurations

### 8.4.6.1 PLL Engaged Internal Mode (PEI)

This mode is the default mode after System Reset or Power-On Reset.

The Bus Clock is based on the PLLCLK, the reference clock for the PLL is internally generated (IRC1M). The PLL is configured to 50 MHz VCOCLK with POSTDIV set to 0x03. If locked (LOCK=1) this results in a PLLCLK of 12.5 MHz and a Bus Clock of 6.25 MHz. The PLL can be re-configured to other bus frequencies.

The clock sources for COP and RTI can be based on the internal reference clock generator (IRC1M) or the RC-Oscillator (ACLK).

### 8.4.6.2 PLL Engaged External Mode (PEE)

In this mode, the Bus Clock is based on the PLLCLK as well (like PEI). The reference clock for the PLL is based on the external oscillator.

The clock sources for COP and RTI can be based on the internal reference clock generator or on the external oscillator clock or the RC-Oscillator (ACLK).

This mode can be entered from default mode PEI by performing the following steps:

1. Configure the PLL for desired bus frequency.
2. Enable the external Oscillator (OSCE bit).
3. Wait for oscillator to start-up and the PLL being locked (LOCK = 1) and (UPOSC =1).
4. Clear all flags in the CPMUIFLG register to be able to detect any future status bit change.
5. Optionally status interrupts can be enabled (CPMUINT register).

Loosing PLL lock status (LOCK=0) means loosing the oscillator status information as well (UPOSC=0).

The impact of loosing the oscillator status (UPOSC=0) in PEE mode is as follows:

- The PLLCLK is derived from the VCO clock (with its actual frequency) divided by four until the PLL locks again.

Application software needs to be prepared to deal with the impact of loosing the oscillator status at any time.

### 9.4.2.2 ADC Control Register 1 (ADCCTL_1)

Module Base + 0x0001

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | CSL_BMOD | RVL_BMOD | SMOD_ACC | AUT_RSTA | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 9-5. ADC Control Register 1 (ADCCTL_1)**

Read: Anytime

Write:

- Bit CSL_BMOD and RVL_BMOD writable if bit ADC_EN clear or bit SMOD_ACC set
- Bit SMOD_ACC only writable in MCU Special Mode
- Bit AUT_RSTA writable anytime

**Table 9-4. ADCCTL_1 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>CSL_BMOD | **CSL Buffer Mode Select Bit** — This bit defines the CSL buffer mode. This bit is only writable if ADC_EN is clear.<br>0 CSL single buffer mode.<br>1 CSL double buffer mode. |
| 6<br>RVL_BMOD | **RVL Buffer Mode Select Bit** — This bit defines the RVL buffer mode.<br>0 RVL single buffer mode<br>1 RVL double buffer mode |
| 5<br>SMOD_ACC | **Special Mode Access Control Bit** — This bit controls register access rights in MCU Special Mode. This bit is automatically cleared when leaving MCU Special Mode.<br>Note: When this bit is set also the ADCCMD register is writeable via the data bus to allow modification of the current command for debugging purpose. But this is only possible if the current command is not already processed (conversion not started).<br>Please see access details given for each register.<br>Care must be taken when modifying ADC registers while bit SMOD_ACC is set to not corrupt a possible ongoing conversion.<br>0 Normal user access - Register write restrictions exist as specified for each bit.<br>1 Special access - Register write restrictions are lifted. |
| 4<br>AUT_RSTA | **Automatic Restart Event after exit from MCU Stop and Wait Mode (SWAI set)** — This bit controls if a Restart Event is automatically generated after exit from MCU Stop Mode or Wait Mode with bit SWAI set. It can be configured for ADC conversion flow control mode “Trigger Mode” and “Restart Mode” (anytime during application runtime).<br>0 No automatic Restart Event after exit from MCU Stop Mode.<br>1 Automatic Restart Event occurs after exit from MCU Stop Mode. |

### 9.4.2.7 ADC Error Interrupt Enable Register (ADCEIE)

Module Base + 0x0006

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | IA_EIE | CMD_EIE | EOL_EIE | Reserved | TRIG_EIE | RSTAR_EIE | LDOK_EIE | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 9-10. ADC Error Interrupt Enable Register (ADCEIE)**

Read: Anytime

Write: Anytime

**Table 9-11. ADCEIE Field Descriptions**

| Field | Description |
|---|---|
| 7<br>IA_EIE | **Illegal Access Error Interrupt Enable Bit** — This bit enables the illegal access error interrupt.<br>0 Illegal access error interrupt disabled.<br>1 Illegal access error interrupt enabled. |
| 6<br>CMD_EIE | **Command Value Error Interrupt Enable Bit** — This bit enables the command value error interrupt.<br>0 Command value interrupt disabled.<br>1 Command value interrupt enabled. |
| 5<br>EOL_EIE | **"End Of List" Error Interrupt Enable Bit** — This bit enables the "End Of List" error interrupt.<br>0 "End Of List" error interrupt disabled.<br>1 "End Of List" error interrupt enabled. |
| 3<br>TRIG_EIE | **Conversion Sequence Trigger Error Interrupt Enable Bit** — This bit enables the conversion sequence trigger error interrupt.<br>0 Conversion sequence trigger error interrupt disabled.<br>1 Conversion sequence trigger error interrupt enabled. |
| 2<br>RSTAR_EIE | **Restart Request Error Interrupt Enable Bit**— This bit enables the restart request error interrupt.<br>0 Restart Request error interrupt disabled.<br>1 Restart Request error interrupt enabled. |
| 1<br>LDOK_EIE | **Load OK Error Interrupt Enable Bit** — This bit enables the Load OK error interrupt.<br>0 Load OK error interrupt disabled.<br>1 Load OK error interrupt enabled. |

**Table 9-23. Analog Input Channel Select**

| CH_SEL[5] | CH_SEL[4] | CH_SEL[3] | CH_SEL[2] | CH_SEL[1] | CH_SEL[0] | Analog Input Channel |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 | Reserved |
| 0 | 0 | 1 | 0 | 0 | 0 | Reserved |
| 0 | 0 | 1 | 0 | 0 | 1 | Internal_1<br>(Vreg_3v3 sense) |
| 0 | 0 | 1 | 0 | 1 | 0 | Internal_2 |
| 0 | 0 | 1 | 0 | 1 | 1 | Internal_3 |
| 0 | 0 | 1 | 1 | 0 | 0 | Internal_4 |
| 0 | 0 | 1 | 1 | 0 | 1 | Internal_5 |
| 0 | 0 | 1 | 1 | 1 | 0 | Internal_6 |
| 0 | 0 | 1 | 1 | 1 | 1 | Internal_7 |
| 0 | 1 | 0 | 0 | 0 | 0 | AN0 |
| 0 | 1 | 0 | 0 | 0 | 1 | AN1 |
| 0 | 1 | 0 | 0 | 1 | 0 | AN2 |
| 0 | 1 | 0 | 0 | 1 | 1 | AN3 |
| 0 | 1 | 0 | 1 | 0 | 0 | AN4 |
| 0 | 1 | x | x | x | x | ANx |
| 1 | x | x | x | x | x | Reserved |

**NOTE**

ANx in Table 9-23 is the maximum number of implemented analog input channels on the device. Please refer to the device overview of the reference manual for details regarding number of analog input channels.

## 9.8.7 Conversion flow control application information

The ADC12B_LBA provides various conversion control scenarios to the user accomplished by the following features.

The ADC conversion flow control can be realized via the data bus only, the internal interface only, or by both access methods. The method used is software configurable via bits ACC_CFG[1:0].

The conversion flow is controlled via the four conversion flow control bits: SEQA, TRIG, RSTA, and LDOK.

Two different conversion flow control modes can be configured: Trigger Mode or Restart Mode

Single or double buffer configuration of CSL and RVL.

### 9.8.7.1 Initial Start of a Command Sequence List

At the initial start of a Command Sequence List after device reset all entries for at least one of the two CSL must have been completed and data must be valid. Depending on if the CSL_0 or the CSL_1 should be executed at the initial start of a Command Sequence List the following conversion control sequence must be applied:

If CSL_0 should be executed at the initial conversion start after device reset:
A Restart Event and a Trigger Event must occur (depending to the selected conversion flow control mode the events must occur one after the other or simultaneously) which causes the ADC to start conversion with commands loaded from CSL_0.

If CSL_1 should be executed at the initial conversion start after device reset:
Bit LDOK must be set simultaneously with the Restart Event followed by a Trigger Event (depending on the selected conversion flow control mode the Trigger events must occur simultaneously or after the Restart Event is finished). As soon as the Trigger Event gets executed the ADC starts conversion with commands loaded from CSL_1.

As soon as a new valid Restart Event occurs the flow for ADC register load at conversion sequence start as described in Section 9.5.3.3, “ADC List Usage and Conversion/Conversion Sequence Flow Description applies.

### 9.8.7.2 Restart CSL execution with currently active CSL

To restart a Command Sequence List execution it is mandatory that the ADC is idle (no conversion or conversion sequence is ongoing).

If necessary, a possible ongoing conversion sequence can be aborted by the Sequence Abort Event (setting bit SEQA). As soon as bit SEQA is cleared by the ADC, the current conversion sequence has been aborted and the ADC is idle (no conversion sequence or conversion ongoing).

After a conversion sequence abort is executed it is mandatory to request a Restart Event (bit RSTA set). After the Restart Event is finished (bit RSTA is cleared), the ADC accepts a new Trigger Event (bit TRIG can be set) and begins conversion from the top of the currently active CSL. In conversion flow control

**Table 10-5. BATIF Register Field Descriptions**

| Field | Description |
|---|---|
| 1<br>BVHIF | **BATS Interrupt Flag High Detect** — The flag is set to 1 when BVHC status bit changes.<br><br>0 No change of the BVHC status bit since the last clearing of the flag.<br>1 BVHC status bit has changed since the last clearing of the flag. |
| 0<br>BVLIF | **BATS Interrupt Flag Low Detect** — The flag is set to 1 when BVLC status bit changes.<br><br>0 No change of the BVLC status bit since the last clearing of the flag.<br>1 BVLC status bit has changed since the last clearing of the flag. |

### 10.3.2.5 Reserved Register

Module Base + 0x0006
Module Base + 0x0007

Access: User read/write[1]

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved |
| Reset | x | x | x | x | x | x | x | x |

**Figure 10-8. Reserved Register**

[1] Read: Anytime
Write: Only in special mode

**NOTE**

These reserved registers are designed for factory test purposes only and are not intended for general user access. Writing to these registers when in special mode can alter the module's functionality.

## 10.4 Functional Description

### 10.4.1 General

The BATS module allows measuring the voltage on the VSUP pin. The voltage at the VSUP pin can be routed via an internal voltage divider to an internal Analog to Digital Converter Channel. Also the BATS module can be configured to generate a low and high voltage interrupt based on VSUP. The trigger level of the high and low interrupt are selectable.

### 10.4.2 Interrupts

This section describes the interrupt generated by the BATS module. The interrupt is only available in CPU run mode. Entering and exiting CPU stop mode has no effect on the interrupt flags.

To make sure the interrupt generation works properly the bus clock frequency must be higher than the Voltage Warning Low Pass Filter frequency ($f_{VWLP\_filter}$).

**Table 16-4. I-Bus Tap and Prescale Values**

| IBC2-0 (bin) | SCL Tap (clocks) | SDA Tap (clocks) |
|---|---|---|
| 000 | 5 | 1 |
| 001 | 6 | 1 |
| 010 | 7 | 2 |
| 011 | 8 | 2 |
| 100 | 9 | 3 |
| 101 | 10 | 3 |
| 110 | 12 | 4 |
| 111 | 15 | 4 |

**Table 16-5. Prescale Divider Encoding**

| IBC5-3 (bin) | scl2start (clocks) | scl2stop (clocks) | scl2tap (clocks) | tap2tap (clocks) |
|---|---|---|---|---|
| 000 | 2 | 7 | 4 | 1 |
| 001 | 2 | 7 | 4 | 2 |
| 010 | 2 | 9 | 6 | 4 |
| 011 | 6 | 9 | 6 | 8 |
| 100 | 14 | 17 | 14 | 16 |
| 101 | 30 | 33 | 30 | 32 |
| 110 | 62 | 65 | 62 | 64 |
| 111 | 126 | 129 | 126 | 128 |

**Table 16-6. Multiplier Factor**

| IBC7-6 | MUL |
|---|---|
| 00 | 01 |
| 01 | 02 |
| 10 | 04 |
| 11 | RESERVED |

The number of clocks from the falling edge of SCL to the first tap (Tap[1]) is defined by the values shown in the scl2tap column of Table 16-4, all subsequent tap points are separated by $2^{IBC5-3}$ as shown in the tap2tap column in Table 16-5. The SCL Tap is used to generated the SCL period and the SDA Tap is used to determine the delay from the falling edge of SCL to SDA changing, the SDA hold time.

IBC7–6 defines the multiplier factor MUL. The values of MUL are shown in the Table 16-6.

**Table 18-8. Time Segment 2 Values**

| TSEG22 | TSEG21 | TSEG20 | Time Segment 2 |
|---|---|---|---|
| 0 | 0 | 0 | 1 Tq clock cycle[1] |
| 0 | 0 | 1 | 2 Tq clock cycles |
| : | : | : | : |
| 1 | 1 | 0 | 7 Tq clock cycles |
| 1 | 1 | 1 | 8 Tq clock cycles |

[1] This setting is not valid. Please refer to Table 18-36 for valid settings.

**Table 18-9. Time Segment 1 Values**

| TSEG13 | TSEG12 | TSEG11 | TSEG10 | Time segment 1 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 Tq clock cycle[1] |
| 0 | 0 | 0 | 1 | 2 Tq clock cycles[1] |
| 0 | 0 | 1 | 0 | 3 Tq clock cycles[1] |
| 0 | 0 | 1 | 1 | 4 Tq clock cycles |
| : | : | : | : | : |
| 1 | 1 | 1 | 0 | 15 Tq clock cycles |
| 1 | 1 | 1 | 1 | 16 Tq clock cycles |

[1] This setting is not valid. Please refer to Table 18-36 for valid settings.

The bit time is determined by the oscillator frequency, the baud rate prescaler, and the number of time quanta (Tq) clock cycles per bit (as shown in Table 18-8 and Table 18-9).

*Eqn. 18-1*

$$\text{Bit Time} = \frac{(\text{Prescaler value})}{f_{CANCLK}} \bullet (1 + \text{TimeSegment1} + \text{TimeSegment2})$$

### 18.3.2.5 MSCAN Receiver Flag Register (CANRFLG)

A flag can be cleared only by software (writing a 1 to the corresponding bit position) when the condition which caused the setting is no longer valid. Every flag has an associated interrupt enable bit in the CANRIER register.

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 18-8. MSCAN Receiver Flag Register (CANRFLG)**

#### 18.3.3.1.1 IDR0–IDR3 for Extended Identifier Mapping

Module Base + 0x00X0

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | ID28 | ID27 | ID26 | ID25 | ID24 | ID23 | ID22 | ID21 |
| Reset: | x | x | x | x | x | x | x | x |

**Figure 18-26. Identifier Register 0 (IDR0) — Extended Identifier Mapping**

**Table 18-26. IDR0 Register Field Descriptions — Extended**

| Field | Description |
|---|---|
| 7-0<br>ID[28:21] | **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 + 0x00X1

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | ID20 | ID19 | ID18 | SRR (=1) | IDE (=1) | ID17 | ID16 | ID15 |
| Reset: | x | x | x | x | x | x | x | x |

**Figure 18-27. Identifier Register 1 (IDR1) — Extended Identifier Mapping**

**Table 18-27. IDR1 Register Field Descriptions — Extended**

| Field | Description |
|---|---|
| 7-5<br>ID[20:18] | **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. |
| 4<br>SRR | **Substitute Remote Request** — This fixed recessive bit is used only in extended format. It must be set to 1 by the user for transmission buffers and is stored as received on the CAN bus for receive buffers. |
| 3<br>IDE | **ID Extended** — This flag indicates whether the extended or standard identifier format is applied in this buffer. In the case of a receive buffer, the flag is set as received and indicates to the CPU how to process the buffer identifier registers. In the case of a transmit buffer, the flag indicates to the MSCAN what type of identifier to send.<br>0 Standard format (11 bit)<br>1 Extended format (29 bit) |
| 2-0<br>ID[17:15] | **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. |

| Global Address | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x033E | PIRL | R | 0 | 0 | 0 | 0 | 0 | 0 | PIRL1 | PIRL0 |
| | | W | | | | | | | | |
| 0x033F | PTTEL | R | 0 | 0 | 0 | 0 | 0 | 0 | PTTEL1 | PTTEL0 |
| | | W | | | | | | | | |
| 0x0340–0x037F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

# N.6 0x0380-0x039F FTMRZ192K2K

| Address | Name | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0380 | FCLKDIV | R | FDIVLD | FDIVLCK | FDIV5 | FDIV4 | FDIV3 | FDIV2 | FDIV1 | FDIV0 |
| | | W | | | | | | | | |
| 0x0381 | FSEC | R | KEYEN1 | KEYEN0 | RNV5 | RNV4 | RNV3 | RNV2 | SEC1 | SEC0 |
| | | W | | | | | | | | |
| 0x0382 | FCCOBIX | R | 0 | 0 | 0 | 0 | 0 | CCOBIX2 | CCOBIX1 | CCOBIX0 |
| | | W | | | | | | | | |
| 0x0383 | FPSTAT | R | FPOVRD | 0 | 0 | 0 | 0 | 0 | 0 | WSTAT ACK |
| | | W | | | | | | | | |
| 0x0384 | FCNFG | R | CCIE | 0 | ERSAREQ | IGNSF | WSTAT[1:0] | | FDFD | FSFD |
| | | W | | | | | | | | |
| 0x0385 | FERCNFG | R | 0 | 0 | 0 | 0 | 0 | 0 | DFDIE | SFDIE |
| | | W | | | | | | | | |
| 0x0386 | FSTAT | R | CCIF | 0 | ACCERR | FPVIOL | MGBUSY | RSVD | MGSTAT1 | MGSTAT0 |
| | | W | | | | | | | | |
| 0x0387 | FERSTAT | R | 0 | 0 | 0 | 0 | 0 | 0 | DFDIF | SFDIF |
| | | W | | | | | | | | |
| 0x0388 | FPROT | R | FPOPEN | RNV6 | FPHDIS | FPHS1 | FPHS0 | FPLDIS | FPLS1 | FPLS0 |
| | | W | | | | | | | | |
| 0x0389 | DFPROT | R | DPOPEN | 0 | 0 | 0 | DPS3 | DPS2 | DPS1 | DPS0 |
| | | W | | | | | | | | |
| 0x038A | FOPT | R | NV7 | NV6 | NV5 | NV4 | NV3 | NV2 | NV1 | NV0 |
| | | W | | | | | | | | |
| 0x038B | FRSV1 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

# N.11 0x05C0-0x05EF TIM0 (continued)

| Address | Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x05C7 | TIM0TTOV | R | 0 | 0 | 0 | 0 | TOV3 | TOV2 | TOV1 | TOV0 |
| | | W | | | | | | | | |
| 0x05C8 | TIM0TCTL1 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x05C9 | TIM0TCTL2 | R | OM3 | OL3 | OM2 | OL2 | OM1 | OL1 | OM0 | OL0 |
| | | W | | | | | | | | |
| 0x05CA | TIM0TCTL3 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x05CB | TIM0TCTL4 | R | EDG3B | EDG3A | EDG2B | EDG2A | EDG1B | EDG1A | EDG0B | EDG0A |
| | | W | | | | | | | | |
| 0x05CC | TIM0TIE | R | 0 | 0 | 0 | 0 | C3I | C2I | C1I | C0I |
| | | W | | | | | | | | |
| 0x05CD | TIM0TSCR2 | R | TOI | 0 | 0 | 0 | | PR2 | PR1 | PR0 |
| | | W | | | | | | | | |
| 0x05CE | TIM0TFLG1 | R | 0 | 0 | 0 | 0 | C3F | C2F | C1F | C0F |
| | | W | | | | | | | | |
| 0x05CF | TIM0TFLG2 | R | TOF | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x05D0 | TIM0TC0H | R | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| | | W | | | | | | | | |
| 0x05D1 | TIM0TC0L | R | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| | | W | | | | | | | | |
| 0x05D2 | TIM0TC1H | R | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| | | W | | | | | | | | |
| 0x05D3 | TIM0TC1L | R | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| | | W | | | | | | | | |
| 0x05D4 | TIM0TC2H | R | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| | | W | | | | | | | | |
| 0x05D5 | TIM0TC2L | R | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| | | W | | | | | | | | |
| 0x05D6 | TIM0TC3H | R | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| | | W | | | | | | | | |
| 0x05D7 | TIM0TC3L | R | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| | | W | | | | | | | | |
| 0x05D8– 0x05DF | Reserved | R | | | | | | | | |
| | | W | | | | | | | | |