**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, LINbus, SCI, SPI |
| Peripherals | DMA, LCD, POR, PWM, WDT |
| Number of I/O | 73 |
| Program Memory Size | 64KB (64K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | 2K x 8 |
| RAM Size | 4K x 8 |
| Voltage - Supply (Vcc/Vdd) | 4.5V ~ 5.5V |
| Data Converters | A/D 4x10b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 100-LQFP |
| Supplier Device Package | 100-LQFP (14x14) |
| Purchase URL | https://www.e-xfl.com/product-detail/nxp-semiconductors/s912zvhl64f1cll |

#### 1.5.2.2 EEPROM

- Up to 2 KB EEPROM
  - 16 data bits plus 6 syndrome ECC (error correction code) bits allow single bit error correction and double fault detection
  - Erase sector size 4 bytes
  - Automated program and erase algorithm
  - User margin level setting for reads

#### 1.5.2.3 SRAM

- Up to 4 KB of general-purpose RAM with ECC
  - Single bit error correction and double bit error detection

### 1.5.3 Clocks, Reset & Power Management Unit (CPMU)

- Real Time Interrupt (RTI)
- Clock Monitor, supervising the correct function of the oscillator (CM)
- System reset generation
- Autonomous periodic interrupt (API) (combination with cyclic, watchdog)
- Low Power Operation
  - RUN mode is the main full performance operating mode with the entire device clocked.
  - WAIT mode when the internal CPU clock is switched off, so the CPU does not execute instructions.
  - Pseudo STOP - system clocks are stopped but the RTI, COP, API, RTC and LCD modules can be enabled with clock source from the osc.
  - STOP - the oscillator is stopped in this mode, all clocks are switched off and all counters and dividers remain frozen. The 32K oscillator can be enabled, RTC and LCD can be still function if enabled. The API and COP can still function if their clock source are from API clock(ACLK).

#### 1.5.3.1 Internal Phase-Locked Loop (IPLL)

- Phase-locked-loop clock frequency multiplier
  - No external components required
  - Reference divider and multiplier allow large variety of clock rates
  - Automatic bandwidth control mode for low-jitter operation
  - Automatic frequency lock detector
  - Configurable option to spread spectrum for reduced EMC radiation (frequency modulation)
  - Reference clock sources:
    - Internal 1 MHz RC oscillator (IRC)
    - External 4-20 MHz crystal oscillator/resonator

#### 1.7.2.30.2 M1COSM, M1COSP, M1SINM and M1SINP Signals

These signal are used to measure the back EMF to calibrate the pointer reset position which are associated with SSD[1].

### 1.7.2.31 Interrupt Signals — $\overline{\text{IRQ}}$ and $\overline{\text{XIRQ}}$

$\overline{\text{IRQ}}$ is a maskable level or falling edge sensitive input. $\overline{\text{XIRQ}}$ is a non-maskable level-sensitive interrupt.

### 1.7.2.32 Oscillator and Clock Signals

#### 1.7.2.32.1 4-20MHz main Oscillator Pins — EXTAL and XTAL

EXTAL and XTAL are the crystal driver. On reset, the OSC is not enabled, all the device clocks are derived from the internal reference clock. EXTAL is the oscillator input. XTAL is the oscillator output.

#### 1.7.2.32.2 32.768kHz Oscillator Pins — 32K_EXTAL and 32K_XTAL

32K_EXTAL and 32K_XTAL are the 32.768KHZ crystal driver. On reset the OSC is not enabled. 32K_EXTAL is the oscillator input. 32K_XTAL is the oscillator output. Figure 1-3 is the 32K OSC connection diagram. Refer to the Appendix Table K-1., “OSC32K DC Electrical Specifications for the $C_x$, $C_y$ and $R_F$ requirement. Both RTC and LCD clock source can from the 32K OSC. The OSC enable control is from the RTC. If the RTCCTL2[CLKSRC] is set, then it will enable the 32K OSC. After enable the OSC, it needs to wait enough time before enable the RTC and LCD. Refer to Appendix Table K-2., “OSC32K Frequency Specifications for the startup time requirement.

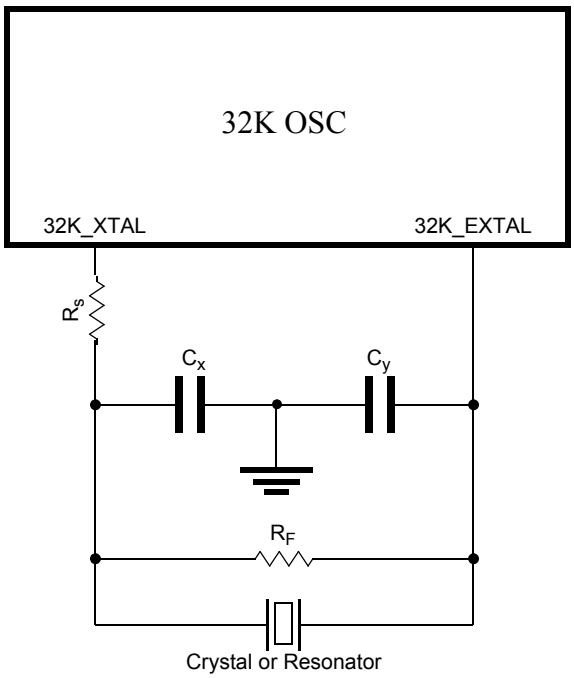



**Figure 1-3. 32K OSC Crystal/Resonator Connection**

**Table 2-3. MODRR1 Routing Register Field Descriptions**

| Field | Description |
|---|---|
| 1<br>PWM2RR | **Module Routing Register** — PWM2 routing<br><br>1 PWM2 to PA5<br>0 PWM2 to PP2 |
| 0<br>PWM0RR | **Module Routing Register** — PWM0 routing<br><br>1 PWM0 to PA4<br>0 PWM0 to PP0 |

### 2.3.2.3 Module Routing Register 2 (MODRR2)

Address 0x0202 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | SCI1RR | IIC0RR | 0 | 0 | T1IC0RR1 | T1IC0RR0 |
| W | | | | | | | | |
| | | | SCI1 | IIC0 | | | TIM1 IC0 | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-3. Module Routing Register 2 (MODRR2)**

1. Read: Anytime
   Write: Once in normal, anytime in special mode

**Table 2-4. MODRR2 Routing Register Field Descriptions**

| Field | Description |
|---|---|
| 5<br>SCI1RR | **Module Routing Register** — SCI1 routing<br><br>1 TXD1 on PP7; RXD1 on PP5<br>0 TXD1 on PC7; RXD1 on PC6 |
| 4<br>IIC0RR | **Module Routing Register** — IIC0 routing<br><br>1 SCL0 on PA2; SDA0 on PA3<br>0 SCL0 on PS4; SDA0 on PS5 |
| 1-0<br>T1IC0RR1-0 | **Module Routing Register** — TIM1 IC0 routing<br><br>11 TIM1 input capture channel 0 is connected to RXD1<br>10 TIM1 input capture channel 0 is connected to RXD0<br>01 TIM1 input capture channel 0 is connected to RTC's CALCLK<br>00 TIM1 input capture channel 0 is connected to PT0 |

### 3.1.1 Glossary

**Table 3-2. Glossary Of Terms**

| Term | Definition |
|---|---|
| MCU | Microcontroller Unit |
| CPU | S12Z Central Processing Unit |
| BDC | S12Z Background Debug Controller |
| ADC | Analog-to-Digital Converter |
| unmapped address range | Address space that is not assigned to a memory |
| reserved address range | Address space that is reserved for future use cases |
| illegal access | Memory access, that is not supported or prohibited by the S12ZMMC, e.g. a data store to NVM |
| access violation | Either an illegal access or an uncorrectable ECC error |
| byte | 8-bit data |
| word | 16-bit data |

### 3.1.2 Overview

The S12ZMMC provides access to on-chip memories and peripherals for the S12ZCPU, the S12ZBDC, and the ADC. It arbitrates memory accesses and determines all of the MCU memory maps. Furthermore, the S12ZMMC is responsible for selecting the MCUs functional mode.

### 3.1.3 Features

- S12ZMMC mode operation control
- Memory mapping for S12ZCPU, S12ZBDC, and ADC
  - Maps peripherals and memories into a 16 MByte address space for the S12ZCPU, the S12ZBDC, and the ADC
  - Handles simultaneous accesses to different on-chip resources (NVM, RAM, and peripherals)
- Access violation detection and logging
  - Triggers S12ZCPU machine exceptions upon detection of illegal memory accesses and uncorrectable ECC errors
  - Logs the state of the S12ZCPU and the cause of the access error

### 3.1.4 Modes of Operation

#### 3.1.4.1 Chip configuration modes

The S12ZMMC determines the chip configuration mode of the device. It captures the state of the MODC pin at reset and provides the ability to switch from special-single chip mode to normal single chip-mode.

**Table 5-8. BDC Command Summary (continued)**

| Command Mnemonic | Command Classification | ACK | Command Structure | Description |
|---|---|---|---|---|
| READ_SAME.sz | Non-Intrusive | Yes | (0x50+4 x sz)/dack/rd.sz | Read from location. An initial READ_MEM defines the address, subsequent READ_SAME reads return content of same address |
| READ_SAME.sz_WS | Non-Intrusive | No | (0x51+4 x sz)/d/ss/rd.sz | Read from location. An initial READ_MEM defines the address, subsequent READ_SAME reads return content of same address |
| READ_BDCCSR | Always Available | No | 0x2D/rd16 | Read the BDCCSR register |
| SYNC_PC | Non-Intrusive | Yes | 0x01/dack/rd24 | Read current PC |
| WRITE_MEM.sz | Non-Intrusive | Yes | (0x10+4 x sz)/ad24/wd.sz/dack | Write the appropriately-sized (sz) memory value to the location specified by the 24-bit address |
| WRITE_MEM.sz_WS | Non-Intrusive | No | (0x11+4 x sz)/ad24/wd.sz/d/ss | Write the appropriately-sized (sz) memory value to the location specified by the 24-bit address and report status |
| WRITE_Rn | Active Background | Yes | (0x40+CRN)/wd32/dack | Write the requested CPU register |
| WRITE_BDCCSR | Always Available | No | 0x0D/wd16 | Write the BDCCSR register |
| ERASE_FLASH | Always Available | No | 0x95/d | Mass erase internal flash |
| STEP1 (TRACE1) | Active Background | Yes | 0x09/dack | Execute one CPU command. |

1. The SYNC command is a special operation which does not have a command code.
2. The GO_UNTIL command is identical to the GO command if ACK is not enabled.

### 5.4.4.1 SYNC

The SYNC command is unlike other BDC commands because the host does not necessarily know the correct speed to use for serial communications until after it has analyzed the response to the SYNC command.

To issue a SYNC command, the host:

1. Ensures that the BKGD pin is high for at least 4 cycles of the slowest possible BDCSI clock without reset asserted.
2. Drives the BKGD pin low for at least 128 cycles of the slowest possible BDCSI clock.
3. Drives BKGD high for a brief speed-up pulse to get a fast rise time. (This speedup pulse is typically one cycle of the host clock which is as fast as the maximum target BDCSI clock).
4. Removes all drive to the BKGD pin so it reverts to high impedance.
5. Listens to the BKGD pin for the sync response pulse.

**Table 5-10. Field Location to Byte Access Mapping**

| Address[1:0] | Access Size | 00 | 01 | 10 | 11 | Note |
|---|---|---|---|---|---|---|
| 00 | 32-bit | Data[31:24] | Data[23:16] | Data [15:8] | Data [7:0] | |
| 01 | 32-bit | Data[31:24] | Data[23:16] | Data [15:8] | Data [7:0] | Realigned |
| 10 | 32-bit | Data[31:24] | Data[23:16] | Data [15:8] | Data [7:0] | Realigned |
| 11 | 32-bit | Data[31:24] | Data[23:16] | Data [15:8] | Data [7:0] | Realigned |
| 00 | 16-bit | Data [15:8] | Data [7:0] | | | |
| 01 | 16-bit | | Data [15:8] | Data [7:0] | | |
| 10 | 16-bit | | | Data [15:8] | Data [7:0] | |
| 11 | 16-bit | | | Data [15:8] | Data [7:0] | Realigned |
| 00 | 8-bit | Data [7:0] | | | | |
| 01 | 8-bit | | Data [7:0] | | | |
| 10 | 8-bit | | | Data [7:0] | | |
| 11 | 8-bit | | | | Data [7:0] | |
| | | | Denotes byte that is not transmitted | | | |

#### 5.4.5.2.1 FILL_MEM and DUMP_MEM Increments and Alignment

FILL_MEM and DUMP_MEM increment the previously accessed address by the previous access size to calculate the address of the current access. On misaligned longword accesses, the address bits [1:0] are forced to zero, therefore the following FILL_MEM or DUMP_MEM increment to the first address in the next 4-byte field. This is shown in Table 5-11, the address of the first DUMP_MEM.32 following READ_MEM.32 being calculated from 0x004000+4.

When misaligned word accesses are realigned, then the original address (not the realigned address) is incremented for the following FILL_MEM, DUMP_MEM command.

Misaligned word accesses can cause the same locations to be read twice as shown in rows 6 and 7. The hardware ensures alignment at an attempted misaligned word access across a 4-byte boundary, as shown in row 7. The following word access in row 8 continues from the realigned address of row 7.

**Table 5-11. Consecutive Accesses With Variable Size**

| Row | Command | Address | Address[1:0] | 00 | 01 | 10 | 11 |
|---|---|---|---|---|---|---|---|
| 1 | READ_MEM.32 | 0x004003 | 11 | Accessed | Accessed | Accessed | Accessed |
| 2 | DUMP_MEM.32 | 0x004004 | 00 | Accessed | Accessed | Accessed | Accessed |
| 3 | DUMP_MEM.16 | 0x004008 | 00 | Accessed | Accessed | | |
| 4 | DUMP_MEM.16 | 0x00400A | 10 | | | Accessed | Accessed |
| 5 | DUMP_MEM.08 | 0x00400C | 00 | Accessed | | | |
| 6 | DUMP_MEM.16 | 0x00400D | 01 | | Accessed | Accessed | |
| 7 | DUMP_MEM.16 | 0x00400E | 10 | | | Accessed | Accessed |
| 8 | DUMP_MEM.16 | 0x004010 | 01 | Accessed | Accessed | | |

Figure 7-2 shows a block diagram of the XOSCLCP.

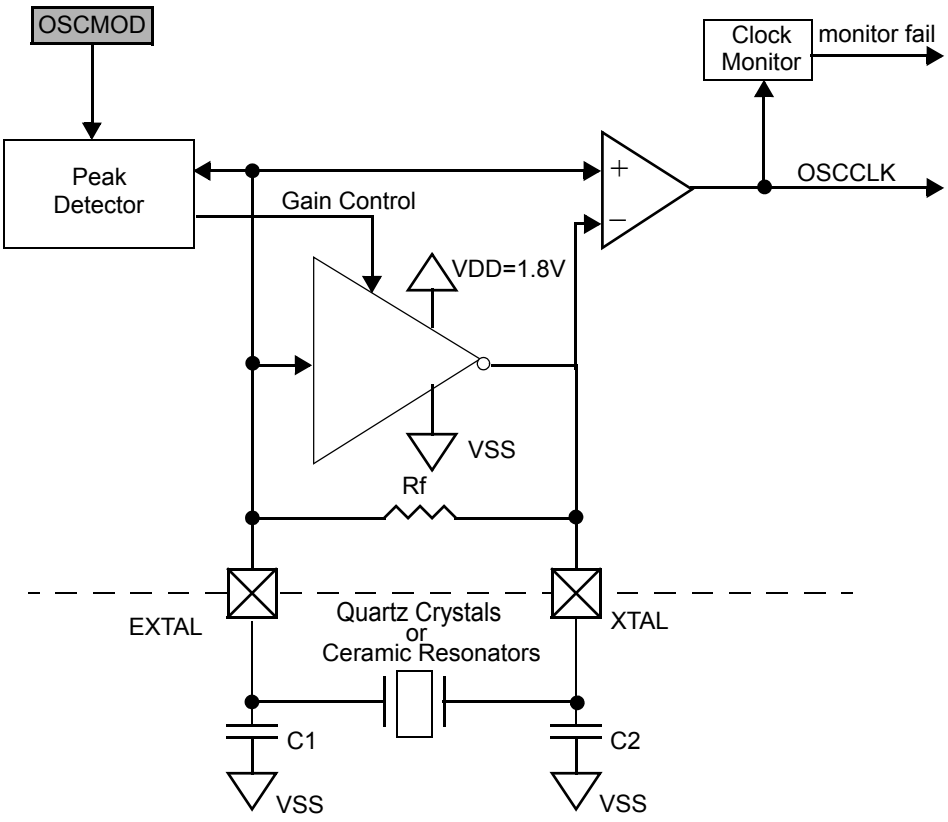



**Figure 7-2. XOSCLCP Block Diagram**

**Table 7-4. Reference Clock Frequency Selection if OSC_LCP is enabled**

| REFCLK Frequency Ranges (OSCE=1) | REFFRQ[1:0] |
|---|---|
| 1MHz <= $f_{REF}$ <= 2MHz | 00 |
| 2MHz < $f_{REF}$ <= 6MHz | 01 |
| 6MHz < $f_{REF}$ <= 12MHz | 10 |
| $f_{REF}$ >12MHz | 11 |

### 7.3.2.18 Autonomous Periodical Interrupt Rate High and Low Register (CPMUAPIRH / CPMUAPIRL)

The CPMUAPIRH and CPMUAPIRL registers allow the configuration of the autonomous periodical interrupt rate.

Module Base + 0x0014

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | APIR15 | APIR14 | APIR13 | APIR12 | APIR11 | APIR10 | APIR9 | APIR8 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 7-23. Autonomous Periodical Interrupt Rate High Register (CPMUAPIRH)**

Module Base + 0x0015

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | APIR7 | APIR6 | APIR5 | APIR4 | APIR3 | APIR2 | APIR1 | APIR0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 7-24. Autonomous Periodical Interrupt Rate Low Register (CPMUAPIRL)**

Read: Anytime

Write: Anytime if APIFE=0, Else writes have no effect.

**Table 7-22. CPMUAPIRH / CPMUAPIRL Field Descriptions**

| Field | Description |
|---|---|
| 15-0<br>APIR[15:0] | **Autonomous Periodical Interrupt Rate Bits** — These bits define the time-out period of the API. See Table 7-23 for details of the effect of the autonomous periodical interrupt rate bits. |

The period can be calculated as follows depending on logical value of the APICLK bit:

APICLK=0: Period = 2*(APIR[15:0] + 1) * (ACLK Clock Period * 2)
APICLK=1: Period = 2*(APIR[15:0] + 1) * Bus Clock Period

**NOTE**

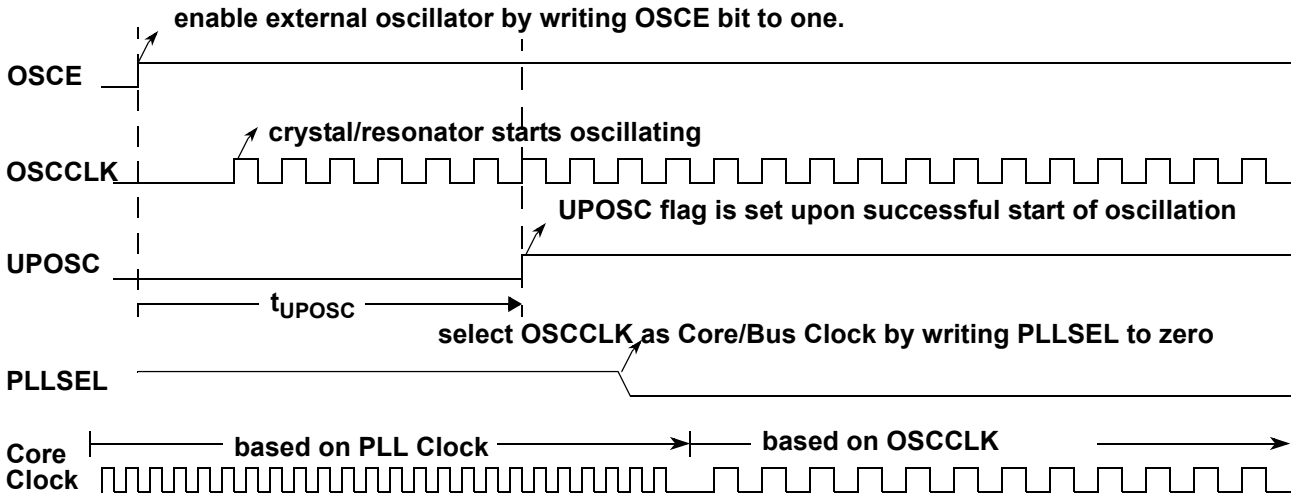
For APICLK bit clear the first time-out period of the API will show a latency time between two to three $f_{ACLK}$ cycles due to synchronous clock gate release when the API feature gets enabled (APIFE bit set).

## 7.4.5 External Oscillator

### 7.4.5.1 Enabling the External Oscillator

An example of how to use the oscillator as source of the Bus Clock is shown in Figure 7-39.

**Figure 7-39. Enabling the external oscillator**



The clock source of each PWM channel is determined by PCLKx bits in PWMCLK (see Section 9.3.2.3, “PWM Clock Select Register (PWMCLK)) and PCLKABx bits in PWMCLKAB as shown in Table 9-5 and Table 9-6.

### 9.3.2.8 PWM Scale A Register (PWMSCLA)

PWMSCLA is the programmable scale value used in scaling clock A to generate clock SA. Clock SA is generated by taking clock A, dividing it by the value in the PWMSCLA register and dividing that by two.

Clock SA = Clock A / (2 * PWMSCLA)

**NOTE**

When PWMSCLA = $00, PWMSCLA value is considered a full scale value of 256. Clock A is thus divided by 512.

Any value written to this register will cause the scale counter to load the new scale value (PWMSCLA).

Module Base + 0x0008

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

**Figure 9-10. PWM Scale A Register (PWMSCLA)**

Read: Anytime

Write: Anytime (causes the scale counter to load the PWMSCLA value)

### 9.3.2.9 PWM Scale B Register (PWMSCLB)

PWMSCLB is the programmable scale value used in scaling clock B to generate clock SB. Clock SB is generated by taking clock B, dividing it by the value in the PWMSCLB register and dividing that by two.

Clock SB = Clock B / (2 * PWMSCLB)

**NOTE**

When PWMSCLB = $00, PWMSCLB value is considered a full scale value of 256. Clock B is thus divided by 512.

Any value written to this register will cause the scale counter to load the new scale value (PWMSCLB).

Module Base + 0x0009

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

**Figure 9-11. PWM Scale B Register (PWMSCLB)**

Read: Anytime

Write: Anytime (causes the scale counter to load the PWMSCLB value).

### 10.4.2.4 ADC Timing Register (ADCTIM)

Module Base + 0x0003

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

= Unimplemented or Reserved

**Figure 10-7. ADC Timing Register (ADCTIM))**

Read: Anytime

Write: These bits are writable if bit ADC_EN is clear or bit SMOD_ACC is set

**Table 10-6. ADCTIM Field Descriptions**

| Field | Description |
|---|---|
| 6-0<br>PRS[6:0] | **ADC Clock Prescaler** — These 7bits are the binary prescaler value PRS. The ADC conversion clock frequency is calculated as follows:<br>$f_{ATDCLK} = \frac{f_{BUS}}{2 \times (PRS + 1)}$<br>Refer to Device Specification for allowed frequency range of $f_{ATDCLK}$. |

### 10.4.2.20 ADC Command Base Pointer Register (ADCCBP)

Module Base + 0x001D

| | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | CMD_PTR[23:16] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Module Base + 0x001E

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | CMD_PTR[15:8] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Module Base + 0x001F

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

= Unimplemented or Reserved

**Figure 10-23. ADC Command Base Pointer Registers (ADCCBP_0, ADCCBP_1, ADCCBP_2))**

Read: Anytime

Write: Bits CMD_PTR[23:2] writable if bit ADC_EN clear or bit SMOD_ACC set

**Table 10-27. ADCCBP Field Descriptions**

| Field | Description |
|---|---|
| 23-2<br>CMD_PTR [23:2] | **ADC Command Base Pointer Address** — These bits define the base address of the two CSL areas inside the system RAM or NVM of the memory map. They are used to calculate the final address from which the conversion commands will be loaded depending on which list is active. For more details see Section 10.5.3.2.2, “Introduction of the two Command Sequence Lists (CSLs). |

### 10.8.3 List Usage — CSL double buffer mode and RVL double buffer mode

In this use case both list types are configured for double buffer mode (CSL_BMOD=1’b1 and RVL_BMOD=1’b1) and whenever a Command Sequence List (CSL) is finished or aborted the command Sequence List is swapped by the simultaneous assertion of bits LDOK and RSTA.

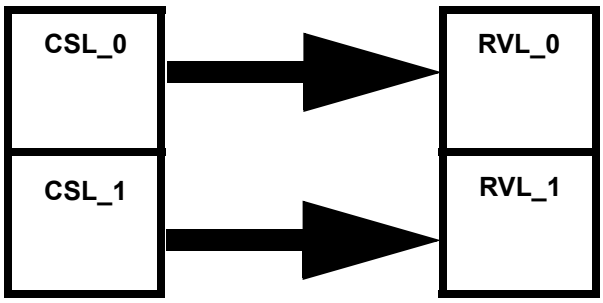



**Figure 10-37. CSL Double Buffer Mode — RVL Double Buffer Mode Diagram**

This use case can be used if the channel order or CSL length varies very frequently in an application.

### 10.8.4 List Usage — CSL double buffer mode and RVL single buffer mode

In this use case the CSL is configured for double buffer mode (CSL_BMOD=1’b1) and the RVL is configured for single buffer mode (RVL_BMOD=1’b0).

The two command lists can be different sizes and the allocated result list memory area in the RAM must be able to hold as many entries as the larger of the two command lists. Each time when the end of a Command Sequence List is reached, if bits LDOK and RSTA are set, the commands list is swapped.

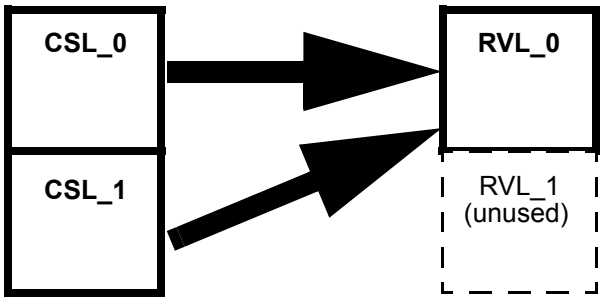



**Figure 10-38. CSL Double Buffer Mode — RVL Single Buffer Mode Diagram**

generates a receive interrupt[1] (see Section 11.4.7.3, "Receive Interrupt") to the CPU. The user's receive handler must read the received message from the RxFG and then reset the RXF flag to acknowledge the interrupt and to release the foreground buffer. A new message, which can follow immediately after the IFS field of the CAN frame, is received into the next available RxBG. If the MSCAN receives an invalid message in its RxBG (wrong identifier, transmission errors, etc.) the actual contents of the buffer will be over-written by the next message. The buffer will then not be shifted into the FIFO.

When the MSCAN module is transmitting, the MSCAN receives its own transmitted messages into the background receive buffer, RxBG, but does not shift it into the receiver FIFO, generate a receive interrupt, or acknowledge its own messages on the CAN bus. The exception to this rule is in loopback mode (see Section 11.3.2.2, "MSCAN Control Register 1 (CANCTL1)") where the MSCAN treats its own messages exactly like all other incoming messages. The MSCAN receives its own transmitted messages in the event that it loses arbitration. If arbitration is lost, the MSCAN must be prepared to become a receiver.

An overrun condition occurs when all receive message buffers in the FIFO are filled with correctly received messages with accepted identifiers and another message is correctly received from the CAN bus with an accepted identifier. The latter message is discarded and an error interrupt with overrun indication is generated if enabled (see Section 11.4.7.5, "Error Interrupt"). The MSCAN remains able to transmit messages while the receiver FIFO is being filled, but all incoming messages are discarded. As soon as a receive buffer in the FIFO is available again, new valid messages will be accepted.

### 11.4.3 Identifier Acceptance Filter

The MSCAN identifier acceptance registers (see Section 11.3.2.12, "MSCAN Identifier Acceptance Control Register (CANIDAC)") define the acceptable patterns of the standard or extended identifier (ID[10:0] or ID[28:0]). Any of these bits can be marked 'don't care' in the MSCAN identifier mask registers (see Section 11.3.2.18, "MSCAN Identifier Mask Registers (CANIDMR0–CANIDMR7)").

A filter hit is indicated to the application software by a set receive buffer full flag (RXF = 1) and three bits in the CANIDAC register (see Section 11.3.2.12, "MSCAN Identifier Acceptance Control Register (CANIDAC)"). These identifier hit flags (IDHIT[2:0]) clearly identify the filter section that caused the acceptance. They simplify the application software's task to identify the cause of the receiver interrupt. If more than one hit occurs (two or more filters match), the lower hit has priority.

A very flexible programmable generic identifier acceptance filter has been introduced to reduce the CPU interrupt loading. The filter is programmable to operate in four different modes:

- Two identifier acceptance filters, each to be applied to:
  - The full 29 bits of the extended identifier and to the following bits of the CAN 2.0B frame:
    - Remote transmission request (RTR)
    - Identifier extension (IDE)
    - Substitute remote request (SRR)
  - The 11 bits of the standard identifier plus the RTR and IDE bits of the CAN 2.0A/B messages. This mode implements two filters for a full length CAN 2.0B compliant extended identifier. Although this mode can be used for standard identifiers, it is recommended to use the four or eight identifier acceptance filters.

1. The receive interrupt occurs only if not masked. A polling scheme can be applied on RXF also.

## 21.3 Memory Map and Registers

This section describes the memory map and registers for the Flash module. Read data from unimplemented memory space in the Flash module is undefined. Write access to unimplemented or reserved memory space in the Flash module will be ignored by the Flash module.

**CAUTION**

Writing to the Flash registers while a Flash command is executing (that is indicated when the value of flag CCIF reads as ’0’) is not allowed. If such action is attempted, the result of the write operation will be unpredictable.

Writing to the Flash registers is allowed when the Flash is not busy executing commands (CCIF = 1) and during initialization right after reset, despite the value of flag CCIF in that case (refer to Section 21.6 for a complete description of the reset sequence).

.

**Table 21-2. FTMRZ Memory Map**

| Global Address (in Bytes) | Size (Bytes) | Description |
|---|---|---|
| 0x0_0000 – 0x0_0FFF | 4,096 | Register Space |
| 0x10_0000 – 0x10_07FF | 2,048 | EEPROM memory |
| 0x1F_4000 – 0x1F_FFFF | 49,152 | NVM Resource Area(1) (see Figure 21-3) |
| 0xFF_0000 – 0xFF_FFFF | 65,536 | P-Flash Memory |

1. See NVM Resource area description in Section 21.4.4

### 21.3.1 Module Memory Map

The S12Z architecture places the P-Flash memory between global addresses 0xFF_0000 and 0xFF_FFFF as shown in Table 21-3

The P-Flash memory map is shown in Figure 21-2.

**Table 21-3. P-Flash Memory Addressing**

| Global Address | Size (Bytes) | Description |
|---|---|---|
| 0xFF_0000 – 0xFF_FFFF | 64 K | P-Flash Block<br>Contains Flash Configuration Field<br>(see Table 21-4) |

The FPROT register, described in Section 21.3.2.9, can be set to protect regions in the Flash memory from accidental program or erase. Three separate memory regions, one growing upward from global address 0xFF_8000 in the Flash memory (called the lower region), one growing downward from global address 0xFF_FFFF in the Flash memory (called the higher region), and the remaining addresses in the Flash

### 21.4.7.3 Erase Verify P-Flash Section Command

The Erase Verify P-Flash Section command will verify that a section of code in the P-Flash memory is erased. The Erase Verify P-Flash Section command defines the starting point of the code to be verified and the number of phrases.

**Table 21-37. Erase Verify P-Flash Section Command FCCOB Requirements**

| Register | FCCOB Parameters | |
|---|---|---|
| FCCOB0 | 0x03 | Global address [23:16] of a P-Flash block |
| FCCOB1 | Global address [15:0] of the first phrase to be verified | |
| FCCOB2 | Number of phrases to be verified | |

Upon clearing CCIF to launch the Erase Verify P-Flash Section command, the Memory Controller will verify the selected section of Flash memory is erased. The CCIF flag will set after the Erase Verify P-Flash Section operation has completed. If the section is not erased, it means blank check failed, both MGSTAT bits will be set.

**Table 21-38. Erase Verify P-Flash Section Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 010 at command launch |
| | | Set if command not available in current mode (see Table 21-29) |
| | | Set if an invalid global address [23:0] is supplied see Table 21-3) |
| | | Set if a misaligned phrase address is supplied (global address [2:0] != 000) |
| | | Set if the requested section crosses a the P-Flash address boundary |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read or if blank check failed. |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read or if blank check failed. |

### 21.4.7.4 Read Once Command

The Read Once command provides read access to a reserved 64 byte field (8 phrases) located in the nonvolatile information register of P-Flash. The Read Once field is programmed using the Program Once command described in Section 21.4.7.6. The Read Once command must not be executed from the Flash block containing the Program Once reserved field to avoid code runaway.

**Table 21-39. Read Once Command FCCOB Requirements**

| Register | FCCOB Parameters | |
|---|---|---|
| FCCOB0 | 0x04 | Not Required |
| FCCOB1 | Read Once phrase index (0x0000 - 0x0007) | |
| FCCOB2 | Read Once word 0 value | |

### 22.3.2.3 BATS Interrupt Enable Register (BATIE)

Module Base + 0x0002 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | BVHIE | BVLIE |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 22-6. BATS Interrupt Enable Register (BATIE)**

1. Read: Anytime
   Write: Anytime

**Table 22-4. BATIE Register Field Descriptions**

| Field | Description |
|---|---|
| 1<br>BVHIE | **BATS Interrupt Enable High** — Enables High Voltage Interrupt **.**<br><br>0 No interrupt will be requested whenever BVHIF flag is set .<br>1 Interrupt will be requested whenever BVHIF flag is set |
| 0<br>BVLIE | **BATS Interrupt Enable Low** — Enables Low Voltage Interrupt .<br><br>0 No interrupt will be requested whenever BVLIF flag is set .<br>1 Interrupt will be requested whenever BVLIF flag is set . |

### 22.3.2.4 BATS Interrupt Flag Register (BATIF)

Module Base + 0x0003 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | BVHIF | BVLIF |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 22-7. BATS Interrupt Flag Register (BATIF)**

1. Read: Anytime
   Write: Anytime, write 1 to clear

# Appendix B
# ADC Electricals

This section describes the characteristics of the analog-to-digital converter.

## B.1 ADC Operating Characteristics

The Table B-1 shows conditions under which the ADC operates.

The following constraints exist to obtain full-scale, full range results:

$VSSA \le VRL \le V_{IN} \le VRH \le VDDA.$

This constraint exists since the sample buffer amplifier can not drive beyond the power supply levels that it ties to. If the input level goes outside of this range it will effectively be clipped.

**Table B-1. ADC Operating Characteristics**

| Supply voltage 4.5V < $V_{DDA}$ < 5.5 V, -40°C < TJ < 150°C | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Num** | **C** | **Rating** | **Symbol** | **Min** | **Typ** | **Max** | **Unit** |
| 1 | D | Reference potential<br>Low<br>High | <br>$V_{RL}$<br>$V_{RH}$ | <br>$V_{SSA}$<br>$V_{DDA}/2$ | <br>—<br>— | <br>$V_{DDA}/2$<br>$V_{DDA}$ | <br>V<br>V |
| 2 | D | Voltage difference $V_{DDX}$ to $V_{DDA}$ | $\Delta_{VDDX}$ | -0.1 | 0 | 0.1 | V |
| 3 | D | Voltage difference $V_{SSX}$ to $V_{SSA}$ | $\Delta_{VSSX}$ | –0.1 | 0 | 0.1 | V |
| 4 | C | Differential reference voltage(1) | $V_{RH}$-$V_{RL}$ | 3.13 | 5.0 | 5.5 | V |
| 5 | C | ADC Clock Frequency (derived from bus clock via the prescaler bus) | $f_{ADCCLk}$ | 0.25 | | 8.0 | MHz |
| 6 | C | Buffer amplifier turn on time (delay after module start/recovery from Stop mode) | $t_{REC}$ | — | — | 1 | µs |
| 7 | D | ADC disable then re-enable require time | $t_{DISABLE}$ | — | — | 3 | Bus Cycles |
| 8 | D | ADC Conversion Period(2)<br>10 bit resolution:<br>8 bit resolution: | <br>$N_{CONV10}$<br>$N_{CONV8}$ | <br>18<br>16 | | <br>38<br>36 | ADC clock Cycles |

1. The accuracy is reduced if differential reference voltage is less than 4.50 V
2. The minimum time assumes a sample time of 4 ADC clock cycles. The maximum time assumes a sample time of 24 ADC clock cycles.

### B.1.1 Factors Influencing Accuracy

Source resistance, source capacitance and current injection have an influence on the accuracy of the ADC. A further factor is that PortAD pins that are configured as output drivers switching.

## 0x03C0–0x03CF SRAM ECC Generator (SRAM_ECC)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x03C0 | ECCSTAT | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | RDY |
| | | W | | | | | | | | |
| 0x03C1 | ECCIE | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | SBEEIE |
| | | W | | | | | | | | |
| 0x03C2 | ECCIF | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | SBEEIF |
| | | W | | | | | | | | |
| 0x03C3-0x03C6 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x03C7 | ECCDPTRH | R | DPTR[23:16] | | | | | | | |
| | | W | | | | | | | | |
| 0x03C8 | ECCDPTRM | R | DPTR[15:8] | | | | | | | |
| | | W | | | | | | | | |
| 0x03C9 | ECCDPTRL | R | DPTR[7:1] | | | | | | | 0 |
| | | W | | | | | | | | |
| 0x03CA-0x3CB | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x03CC | ECCDDH | R | DDATA[15:8] | | | | | | | |
| | | W | | | | | | | | |
| 0x03CD | ECCDDL | R | DDATA[7:0] | | | | | | | |
| | | W | | | | | | | | |
| 0x03CE | ECCDE | R | 0 | 0 | DECC[5:0] | | | | | |
| | | W | | | | | | | | |
| 0x03CF | ECCDCMD | R | ECCDRR | 0 | 0 | 0 | 0 | 0 | ECCDW | ECCDR |
| | | W | | | | | | | | |

## 0x03D0–0x03FF Reserved

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x03D0-0x03FF | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |