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	-
Core Size	-
Speed	-
Connectivity	-
Peripherals	-
Number of I/O	-
Program Memory Size	-
Program Memory Type	-
EEPROM Size	-
RAM Size	-
Voltage - Supply (Vcc/Vdd)	-
Data Converters	-
Oscillator Type	-
Operating Temperature	-
Mounting Type	-
Package / Case	-
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/s912zvhy32f1vlq

5.1.3.3 Low-Power Modes

5.1.3.3.1 Stop Mode

The execution of the CPU STOP instruction leads to stop mode only when all bus masters (CPU, or others, depending on the device) have finished processing. The operation during stop mode depends on the ENBDC and BDCCIS bit settings as summarized in Table 5-3

Table 5-3. BDC STOP Operation Dependencies

ENBDC	BDCCIS	Description Of Operation
0	0	BDC has no effect on STOP mode.
0	1	BDC has no effect on STOP mode.
1	0	Only BDCCLK clock continues
1	1	All clocks continue

A disabled BDC has no influence on stop mode operation. In this case the BDCSI clock is disabled in stop mode thus it is not possible to enable the BDC from within stop mode.

STOP Mode With BDC Enabled And BDCCIS Clear

If the BDC is enabled and BDCCIS is clear, then the BDC prevents the BDCCLK clock (Figure 5-5) from being disabled in stop mode. This allows BDC communication to continue throughout stop mode in order to access the BDCCSR register. All other device level clock signals are disabled on entering stop mode.

NOTE

This is intended for application debugging, not for fast flash programming. Thus the CLKSW bit must be clear to map the BDCSI to BDCCLK.

With the BDC enabled, an internal acknowledge delays stop mode entry and exit by 2 BDCSI clock + 2 bus clock cycles. If no other module delays stop mode entry and exit, then these additional clock cycles represent a difference between the debug and not debug cases. Furthermore if a BDC internal access is being executed when the device is entering stop mode, then the stop mode entry is delayed until the internal access is complete (typically for 1 bus clock cycle).

Accesses to the internal memory map are not possible when the internal device clocks are disabled. Thus attempted accesses to memory mapped resources are suppressed and the NORESP flag is set. Resources can be accessed again by the next command received following exit from Stop mode.

A BACKGROUND command issued whilst in stop mode remains pending internally until the device leaves stop mode. This means that subsequent active BDM commands, issued whilst BACKGROUND is pending, set the ILLCMD flag because the device is not yet in active BDM.

If ACK handshaking is enabled, then the first ACK, following a stop mode entry is long to indicate a stop exception. The BDC indicates a stop mode occurrence by setting the BDCCSR bit STOP. If the host attempts further communication before the ACK pulse generation then the OVRUN bit is set.

Table 6-16. DBGSCR1 Field Descriptions

Field	Description
1–0 C0SC[1:0]	Channel 0 State Control. These bits select the targeted next state whilst in State1 following a match0.
3–2 C1SC[1:0]	Channel 1 State Control. These bits select the targeted next state whilst in State1 following a match1.
5–4 C2SC[1:0]	Channel 2 State Control. These bits select the targeted next state whilst in State1 following a match2.
7–6 C3SC[1:0]	Channel 3 State Control. If EEVE !=10, these bits select the targeted next state whilst in State1 following a match3. If EEVE = 10, these bits select the targeted next state whilst in State1 following an external event.

Table 6-17. State1 Match State Sequencer Transitions

CxSC[1:0]	Function
00	Match has no effect
01	Match forces sequencer to State2
10	Match forces sequencer to State3
11	Match forces sequencer to Final State

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

6.3.2.8 Debug State Control Register 2 (DBGSCR2)

Address: 0x0108

	7	6	5	4	3	2	1	0
R W	C3SC1	C3SC0	C2SC1	C2SC0	C1SC1	C1SC0	C0SC1	C0SC0
Reset	0	0	0	0	0	0	0	0

Figure 6-10. Debug State Control Register 2 (DBGSCR2)

Read: Anytime.

Write: If DBG is not armed and PTACT is clear.

The state control register 2 selects the targeted next state whilst in State2. The matches refer to the outputs of the comparator match control logic as depicted in Figure 6-1 and described in Section 6.3.2.12”. Comparators must be enabled by setting the comparator enable bit in the associated DBGXCTL control register.

Table 6-18. DBGSCR2 Field Descriptions

Field	Description
1–0 C0SC[1:0]	Channel 0 State Control. These bits select the targeted next state whilst in State2 following a match0.
3–2 C1SC[1:0]	Channel 1 State Control. These bits select the targeted next state whilst in State2 following a match1.

accesses). Furthermore, comparators A and C can compare the data buses to values stored in DBGXD3-0 and allow data bit masking.

The comparators can monitor the buses for an exact address or an address range. The comparator configuration is controlled by the control register contents and the range control by the DBGC2 contents.

The comparator control register also allows the type of data access to be included in the comparison through the use of the RWE and RW bits. The RWE bit controls whether the access type is compared for the associated comparator and the RW bit selects either a read or write access for a valid match.

The INST bit in each comparator control register is used to determine the matching condition. By setting INST, the comparator matches opcode addresses, whereby the databus, data mask, RW and RWE bits are ignored. The comparator register must be loaded with the exact opcode address.

The comparator can be configured to match memory access addresses by clearing the INST bit.

Each comparator match can force a transition to another state sequencer state (see Section 6.4.3”).

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

Comparators C and D can also be used to select an address range to trace from, when tracing CPU accesses in Detail mode. This is determined by the TRANGE bits in the DBGTCRH register. The TRANGE encoding is shown in Table 6-9. If the TRANGE bits select a range definition using comparator D and the COMPE bit is clear, then comparator D is configured for trace range definition. By setting the COMPE bit the comparator is configured for address bus comparisons, the TRANGE bits are ignored and the tracing range function is disabled. Similarly if the TRANGE bits select a range definition using comparator C and the COMPE bit is clear, then comparator C is configured for trace range definition.

Match[0, 1, 2, 3] map directly to Comparators [A, B, C, D] respectively, except in range modes (see Section 6.3.2.2”). Comparator priority rules are described in the event priority section (Section 6.4.3.5”).

6.4.2.1 Exact Address Comparator Match

With range comparisons disabled, the match condition is an exact equivalence of address bus with the value stored in the comparator address registers. Qualification of the type of access (R/W) is also possible.

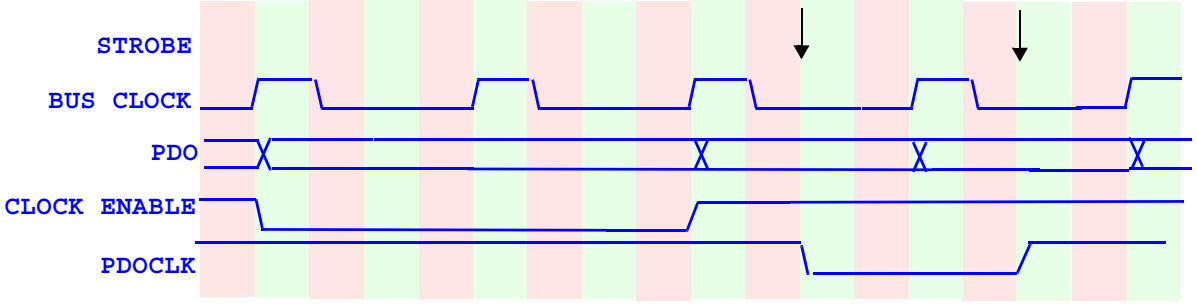
Code may contain various access forms of the same address, for example a 16-bit access of ADDR[n] or byte access of ADDR[n+1] both access n+1. The comparators ensure that any access of the address defined by the comparator address register generates a match, as shown in the example of Table 6-41. Thus if the comparator address register contains ADDR[n+1] any access of ADDR[n+1] matches. This means that a 16-bit access of ADDR[n] or 32-bit access of ADDR[n-1] also match because they also access ADDR[n+1]. The right hand columns show the contents of DBGxA that would match for each access.

Table 6-41. Comparator Address Bus Matches

Access	Address	ADDR[n]	ADDR[n+1]	ADDR[n+2]	ADDR[n+3]
32-bit	ADDR[n]	Match	Match	Match	Match
16-bit	ADDR[n]	Match	Match	No Match	No Match
16-bit	ADDR[n+1]	No Match	Match	Match	No Match

Figure 6-31 shows the profiling clock, PDOCLK, whose edges are offset from the bus clock, to ease setup and hold time requirements relative to PDO, which is synchronous to the bus clock.

Figure 6-31. PDO Profiling Clock Control



The trace buffer is used as a temporary storage medium to store COF information before it is transmitted. COF information can be transmitted whilst new information is written to the trace buffer. The trace buffer data is transmitted at PDO least significant bit first. After the first trace buffer entry is made, transmission begins in the first clock period in which no further data is written to the trace buffer.

If a trace buffer line transmission completes before the next trace buffer line is ready, then the clock output is held at a constant level until the line is ready for transfer.

6.4.6.2 Profiling Configuration, Alignment and Mode Dependencies

The PROFILE bit must be set and the DBG armed to enable profiling. Furthermore the PDOE bit must be set to configure the PDO and PDOCLK pins for profiling.

If TALIGN is configured for End-Aligned tracing then profiling begins as soon as the module is armed.

If TALIGN is configured for Begin-aligned tracing, then profiling begins when the state sequencer enters Final State and continues until a software disarm or trace buffer overflow occurs; thus profiling does not terminate after 64 line entries have been made.

Mid-Align tracing is not supported whilst profiling; if the TALIGN bits are configured for Mid-Align tracing when PROFILE is set, then the alignment defaults to end alignment.

Profiling entries continue until either a trace buffer overflow occurs or the DBG is disarmed by a state machine transition to State0. The profiling output transmission continues, even after disarming, until all trace buffer entries have been transmitted. The PTACT bit indicates if a profiling transmission is still active. The PTBOVF indicates if a trace buffer overflow has occurred.

The profiling timestamp feature is used only for the PTVB and PTW formats, thus differing from timestamps offered in other modes.

Profiling does not support trace buffer gating. The external pin gating feature is ignored during profiling.

When the DBG module is disarmed but profiling transmission is ongoing, register write accesses are suppressed.

When the DBG module is disarmed but profiling transmission is still ongoing, reading from the DBGTB returns the code 0xEE.

7.1.2 Modes of Operation

This subsection lists and briefly describes all operating modes supported by the S12CPMU_UHV_V5.

7.1.2.1 Run Mode

The voltage regulator is in Full Performance Mode (FPM).

NOTE

The voltage regulator is active, providing the nominal supply voltages with full current sourcing capability (see also Appendix for VREG electrical parameters). The features ACLK clock source, Low Voltage Interrupt (LVI), Low Voltage Reset (LVR) and Power-On Reset (POR) are available.

The Phase Locked Loop (PLL) is on.

The Internal Reference Clock (IRC1M) is on.

The API is available.

- **PLL Engaged Internal (PEI)**
  - This is the default mode after System Reset and Power-On Reset.
  - The Bus Clock is based on the PLLCLK.
  - After reset the PLL is configured for 50MHz VCOCLK operation.
    Post divider is 0x03, so PLLCLK is VCOCLK divided by 4, that is 12.5MHz and Bus Clock is 6.25MHz.
    The PLL can be re-configured for other bus frequencies.
  - The reference clock for the PLL (REFCLK) is based on internal reference clock IRC1M.
- **PLL Engaged External (PEE)**
  - The Bus Clock is based on the PLLCLK.
  - This mode can be entered from default mode PEI by performing the following steps:
    - Configure the PLL for desired bus frequency.
    - Program the reference divider (REFDIV[3:0] bits) to divide down oscillator frequency if necessary.
    - Enable the external oscillator (OSCE bit).
    - Wait for oscillator to start up (UPOSC=1) and PLL to lock (LOCK=1).
- **PLL Bypassed External (PBE)**
  - The Bus Clock is based on the Oscillator Clock (OSCCLK).
  - The PLLCLK is always on to qualify the external oscillator clock. Therefore it is necessary to make sure a valid PLL configuration is used for the selected oscillator frequency.
  - This mode can be entered from default mode PEI by performing the following steps:
    - Make sure the PLL configuration is valid for the selected oscillator frequency.

7.3.2.10 S12CPMU_UHV_V5 COP Control Register (CPMUCOP)

This register controls the COP (Computer Operating Properly) watchdog.

The clock source for the COP is either ACLK, IRCCLK or OSCCLK depending on the setting of the COPOSCSEL0 and COPOSCSEL1 bit (see also Table 7-8).
In Stop Mode with PSTP=1 (Pseudo Stop Mode), COPOSCSEL0=1 and COPOSCEL1=0 and PCE=1 the COP continues to run, else the COP counter halts in Stop Mode with COPOSCSEL1 =0.
In Full Stop Mode and Pseudo Stop Mode with COPOSCSEL1=1 the COP continues to run.

Module Base + 0x000C

	7	6	5	4	3	2	1	0
R	WCOP	RSBCK	0	0	0	CR2	CR1	CR0
W			WRTMASK					
Reset	F	0	0	0	0	F	F	F

After de-assert of System Reset the values are automatically loaded from the Flash memory. See Device specification for details.

= Unimplemented or Reserved

Figure 7-13. S12CPMU_UHV_V5 COP Control Register (CPMUCOP)

Read: Anytime

Write:

1. RSBCK: Anytime in Special Mode; write to “1” but not to “0” in Normal Mode
2. WCOP, CR2, CR1, CR0:
   - Anytime in Special Mode, when WRTMASK is 0, otherwise it has no effect
   - Write once in Normal Mode, when WRTMASK is 0, otherwise it has no effect.
     - Writing CR[2:0] to “000” has no effect, but counts for the “write once” condition.
     - Writing WCOP to “0” has no effect, but counts for the “write once” condition.

When a non-zero value is loaded from Flash to CR[2:0] the COP time-out period is started.

A change of the COPOSCSEL0 or COPOSCSEL1 bit (writing a different value) or loosing UPOSC status while COPOSCSEL1 is clear and COPOSCSEL0 is set, re-starts the COP time-out period.

In Normal Mode the COP time-out period is restarted if either of these conditions is true:

1. Writing a non-zero value to CR[2:0] (anytime in special mode, once in normal mode) with WRTMASK = 0.
2. Writing WCOP bit (anytime in Special Mode, once in Normal Mode) with WRTMASK = 0.
3. Changing RSBCK bit from “0” to “1”.

In Special Mode, any write access to CPMUCOP register restarts the COP time-out period.

```
/* Procedure proposed by to setup PLL and Oscillator */
/* example for OSC = 4 MHz and Bus Clock = 25MHz, That is VCOCLK = 50MHz */

/* Initialize */
/* PLL Clock = 50 MHz, divide by one */
CPMUPOSTDIV = 0x00;

/* Generally: Whenever changing PLL reference clock (REFCLK) frequency to a higher value */
/* it is recommended to write CPMUSYNR = 0x00 in order to stay within specified */
/* maximum frequency of the MCU */
CPMUSYNR = 0x00;

/* configure PLL reference clock (REFCLK) for usage with Oscillator */
/* OSC=4MHz divide by 4 (3+1) = 1MHz, REFCLK range 1MHz to 2 MHz (REFFRQ[1:0] = 00) */
CPMUREFDV = 0x03;

/* enable external Oscillator, switch PLL reference clock (REFCLK) to OSC */
CPMUOSC = 0x80;

/* multiply REFCLK = 1MHz by 2*(24+1)*1MHz = 50MHz */
/* VCO range 48 to 80 MHz (VCOFRQ[1:0] = 01) */
CPMUSYNR = 0x58;

/* clear all flags, especially LOCKIF and OSCIF */
CPMUIFLG = 0xFF;

/* put your code to loop and wait for the LOCKIF and OSCIF or */
/* poll CPMUIFLG register until both UPOSC and LOCK status are "1" */
/* that is CPMIFLG == 0x1B */

/*..............continue to your main code execution here..............*/

/* in case later in your code you want to disable the Oscillator and use the */
/* 1MHz IRCCLK as PLL reference clock */

/* Generally: Whenever changing PLL reference clock (REFCLK) frequency to a higher value */
/* it is recommended to write CPMUSYNR = 0x00 in order to stay within specified */
/* maximum frequency of the MCU */
CPMUSYNR = 0x00;

/* disable OSC and switch PLL reference clock to IRC */
CPMUOSC = 0x00;

/* multiply REFCLK = 1MHz by 2*(24+1)*1MHz = 50MHz */
/* VCO range 48 to 80 MHz (VCOFRQ[1:0] = 01) */
CPMUSYNR = 0x58;

/* clear all flags, especially LOCKIF and OSCIF */
CPMUIFLG = 0xFF;

/* put your code to loop and wait for the LOCKIF or */
/* poll CPMUIFLG register until both LOCK status is "1" */
/* that is CPMIFLG == 0x18 */

/*..............continue to your main code execution here..............*/
```

9.2.1 PWM7 - PWM0 — PWM Channel 7 - 0

Those pins serve as waveform output of PWM channel 7 - 0.

9.3 Memory Map and Register Definition

9.3.1 Module Memory Map

This section describes the content of the registers in the scalable PWM module. The base address of the scalable PWM module is determined at the MCU level when the MCU is defined. The register decode map is fixed and begins at the first address of the module address offset. The figure below shows the registers associated with the scalable PWM and their relative offset from the base address. The register detail description follows the order they appear in the register map.

Reserved bits within a register will always read as 0 and the write will be unimplemented. Unimplemented functions are indicated by shading the bit.

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.

9.3.2 Register Descriptions

This section describes in detail all the registers and register bits in the scalable PWM module.

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
0x0000 PWME(1)	R	PWME7	PWME6	PWME5	PWME4	PWME3	PWME2	PWME1	PWME0
	W								
0x0001 PWMPOL1	R	PPOL7	PPOL6	PPOL5	PPOL4	PPOL3	PPOL2	PPOL1	PPOL0
	W								
0x0002 PWMCLK1	R	PCLK7	PCLKL6	PCLK5	PCLK4	PCLK3	PCLK2	PCLK1	PCLK0
	W								
0x0003 PWMPRCLK	R	0	PCKB2	PCKB1	PCKB0	0	PCKA2	PCKA1	PCKA0
	W								
0x0004 PWMCAE1	R	CAE7	CAE6	CAE5	CAE4	CAE3	CAE2	CAE1	CAE0
	W								
0x0005 PWMCTL1	R	CON67	CON45	CON23	CON01	PSWAI	PFRZ	0	0
	W								

= Unimplemented or Reserved

Figure 9-2. The scalable PWM Register Summary (Sheet 1 of 4)

10.4.2.14 ADC End Of List Result Information Register (ADCEOLRI)

This register is cleared when bit ADC_SR is set or bit ADC_EN is clear.

Module Base + 0x0010

	7	6	5	4	3	2	1	0
R	CSL_EOL	RVL_EOL	0	0	0	0	0	0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 10-17. ADC End Of List Result Information Register (ADCEOLRI)

Read: Anytime

Write: Never

Table 10-18. ADCEOLRI Field Descriptions

Field	Description
7 CSL_EOL	**Active CSL When "End Of List" Command Type Executed** — This bit indicates the active (used) CSL when a "End Of List" command type has been executed and related data has been stored to RAM. 0 CSL_0 active when "End Of List" command type executed. 1 CSL_1 active when "End Of List" command type executed.
6 RVL_EOL	**Active RVL When "End Of List" Command Type Executed** — This bit indicates the active (used) RVL when a "End Of List" command type has been executed and related data has been stored to RAM. 0 RVL_0 active when "End Of List" command type executed. 1 RVL_1 active when "End Of List" command type executed.

NOTE

The conversion interrupt EOL_IF occurs and simultaneously the register ADCEOLRI is updated when the "End Of List" conversion command type has been processed and related data has been stored to RAM.

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
0x0000 CANCTL0	R	RXFRM	RXACT	CSWAI	SYNCH	TIME	WUPE	SLPRQ	INITRQ
	W								
0x0001 CANCTL1	R	CANE	CLKSRC	LOOPB	LISTEN	BORM	WUPM	SLPAK	INITAK
	W								
0x0002 CANBTR0	R	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0
	W								
0x0003 CANBTR1	R	SAMP	TSEG22	TSEG21	TSEG20	TSEG13	TSEG12	TSEG11	TSEG10
	W								
0x0004 CANRFLG	R	WUPIF	CSCIF	RSTAT1	RSTAT0	TSTAT1	TSTAT0	OVRIF	RXF
	W								
0x0005 CANRIER	R	WUPIE	CSCIE	RSTATE1	RSTATE0	TSTATE1	TSTATE0	OVRIE	RXFIE
	W								
0x0006 CANTFLG	R	0	0	0	0	0	TXE2	TXE1	TXE0
	W								
0x0007 CANTIER	R	0	0	0	0	0	TXEIE2	TXEIE1	TXEIE0
	W								
0x0008 CANTARQ	R	0	0	0	0	0	ABTRQ2	ABTRQ1	ABTRQ0
	W								
0x0009 CANTAAK	R	0	0	0	0	0	ABTAK2	ABTAK1	ABTAK0
	W								
0x000A CANTBSEL	R	0	0	0	0	0	TX2	TX1	TX0
	W								
0x000B CANIDAC	R	0	0	IDAM1	IDAM0	0	IDHIT2	IDHIT1	IDHIT0
	W								
0x000C Reserved	R	0	0	0	0	0	0	0	0
	W								
0x000D CANMISC	R	0	0	0	0	0	0	0	BOHOLD
	W								

= Unimplemented or Reserved

Figure 11-3. MSCAN Register Summary

Module Base + 0x00XF

Access: User read/write(1)

	7	6	5	4	3	2	1	0
R	TSR7	TSR6	TSR5	TSR4	TSR3	TSR2	TSR1	TSR0
W								
Reset:	x	x	x	x	x	x	x	x

Figure 11-38. Time Stamp Register — Low Byte (TSRL)

1. Read: or transmit buffers: Anytime when TXEx flag is set (see Section 11.3.2.7, "MSCAN Transmitter Flag Register (CANTFLG)") and the corresponding transmit buffer is selected in CANTBSEL (see Section 11.3.2.11, "MSCAN Transmit Buffer Selection Register (CANTBSEL)"). For receive buffers: Anytime when RXF is set.
   Write: Unimplemented

12.3.2.9 SCI Data Registers (SCIDRH, SCIDRL)

Module Base + 0x0006

	7	6	5	4	3	2	1	0
R	R8	T8	0	0	0	Reserved	Reserved	Reserved
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 12-12. SCI Data Registers (SCIDRH)

Module Base + 0x0007

	7	6	5	4	3	2	1	0
R	R7	R6	R5	R4	R3	R2	R1	R0
W	T7	T6	T5	T4	T3	T2	T1	T0
Reset	0	0	0	0	0	0	0	0

Figure 12-13. SCI Data Registers (SCIDRL)

Read: Anytime; reading accesses SCI receive data register

Write: Anytime; writing accesses SCI transmit data register; writing to R8 has no effect

NOTE

The reserved bit SCIDRH[2:0] are designed for factory test purposes only, and are not intended for general user access. Writing to these bit is possible when in special mode and can alter the modules functionality.

Table 12-13. SCIDRH and SCIDRL Field Descriptions

Field	Description
SCIDRH 7 R8	**Received Bit 8** — R8 is the ninth data bit received when the SCI is configured for 9-bit data format (M = 1).
SCIDRH 6 T8	**Transmit Bit 8** — T8 is the ninth data bit transmitted when the SCI is configured for 9-bit data format (M = 1).
SCIDRL 7:0 R[7:0] T[7:0]	**R7:R0** — Received bits seven through zero for 9-bit or 8-bit data formats **T7:T0** — Transmit bits seven through zero for 9-bit or 8-bit formats

NOTE

If the value of T8 is the same as in the previous transmission, T8 does not have to be rewritten.The same value is transmitted until T8 is rewritten

In 8-bit data format, only SCI data register low (SCIDRL) needs to be accessed.

The SCI also sets a flag, the transmit data register empty flag (TDRE), every time it transfers data from the buffer (SCIDRH/L) to the transmitter shift register.The transmit driver routine may respond to this flag by writing another byte to the Transmitter buffer (SCIDRH/SCIDRL), while the shift register is still shifting out the first byte.

To initiate an SCI transmission:

1. Configure the SCI:
   a) Select a baud rate. Write this value to the SCI baud registers (SCIBDH/L) to begin the baud rate generator. Remember that the baud rate generator is disabled when the baud rate is zero. Writing to the SCIBDH has no effect without also writing to SCIBDL.
   b) Write to SCICR1 to configure word length, parity, and other configuration bits (LOOPS,RSRC,M,WAKE,ILT,PE,PT).
   c) Enable the transmitter, interrupts, receive, and wake up as required, by writing to the SCICR2 register bits (TIE,TCIE,RIE,ILIE,TE,RE,RWU,SBK). A preamble or idle character will now be shifted out of the transmitter shift register.
2. Transmit Procedure for each byte:
   a) Poll the TDRE flag by reading the SCISR1 or responding to the TDRE interrupt. Keep in mind that the TDRE bit resets to one.
   b) If the TDRE flag is set, write the data to be transmitted to SCIDRH/L, where the ninth bit is written to the T8 bit in SCIDRH if the SCI is in 9-bit data format. A new transmission will not result until the TDRE flag has been cleared.
3. Repeat step 2 for each subsequent transmission.

NOTE

The TDRE flag is set when the shift register is loaded with the next data to be transmitted from SCIDRH/L, which happens, generally speaking, a little over half-way through the stop bit of the previous frame. Specifically, this transfer occurs 9/16ths of a bit time AFTER the start of the stop bit of the previous frame.

Writing the TE bit from 0 to a 1 automatically loads the transmit shift register with a preamble of 10 logic 1s (if M = 0) or 11 logic 1s (if M = 1). After the preamble shifts out, control logic transfers the data from the SCI data register into the transmit shift register. A logic 0 start bit automatically goes into the least significant bit position of the transmit shift register. A logic 1 stop bit goes into the most significant bit position.

Hardware supports odd or even parity. When parity is enabled, the most significant bit (MSB) of the data character is the parity bit.

The transmit data register empty flag, TDRE, in SCI status register 1 (SCISR1) becomes set when the SCI data register transfers a byte to the transmit shift register. The TDRE flag indicates that the SCI data register can accept new data from the internal data bus. If the transmit interrupt enable bit, TIE, in SCI control register 2 (SCICR2) is also set, the TDRE flag generates a transmitter interrupt request.

Figure 12-17 shows two cases of break detect. In trace RXD_1 the break symbol starts with the start bit, while in RXD_2 the break starts in the middle of a transmission. If BRKDFE = 1, in RXD_1 case there will be no byte transferred to the receive buffer and the RDRF flag will not be modified. Also no framing error or parity error will be flagged from this transfer. In RXD_2 case, however the break signal starts later during the transmission. At the expected stop bit position the byte received so far will be transferred to the receive buffer, the receive data register full flag will be set, a framing error and if enabled and appropriate a parity error will be set. Once the break is detected the BRKDIF flag will be set.

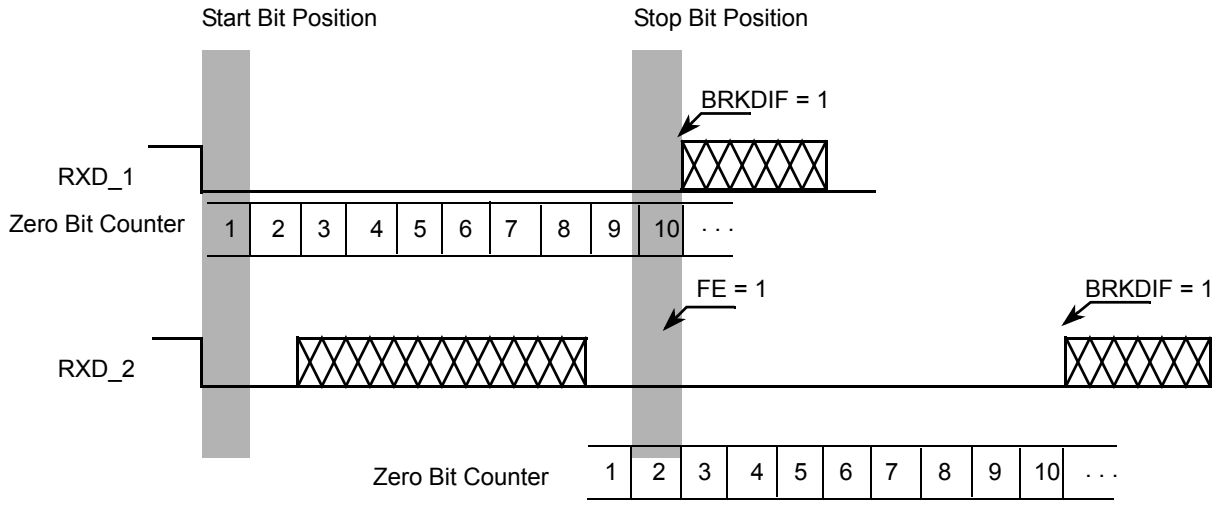


Figure 12-17. Break Detection if BRKDFE = 1 (M = 0)

12.4.5.4 Idle Characters

An idle character (or preamble) contains all logic 1s and has no start, stop, or parity bit. Idle character length depends on the M bit in SCI control register 1 (SCICR1). The preamble is a synchronizing idle character that begins the first transmission initiated after writing the TE bit from 0 to 1.

If the TE bit is cleared during a transmission, the TXD pin becomes idle after completion of the transmission in progress. Clearing and then setting the TE bit during a transmission queues an idle character to be sent after the frame currently being transmitted.

NOTE

When queueing an idle character, return the TE bit to logic 1 before the stop bit of the current frame shifts out through the TXD pin. Setting TE after the stop bit appears on TXD causes data previously written to the SCI data register to be lost. Toggle the TE bit for a queued idle character while the TDRE flag is set and immediately before writing the next byte to the SCI data register.

If the TE bit is clear and the transmission is complete, the SCI is not the master of the TXD pin

12.5.2 Modes of Operation

12.5.2.1 Run Mode

Normal mode of operation.

To initialize a SCI transmission, see Section 12.4.5.2, “Character Transmission”.

12.5.2.2 Wait Mode

SCI operation in wait mode depends on the state of the SCISWAI bit in the SCI control register 1 (SCICR1).

- If SCISWAI is clear, the SCI operates normally when the CPU is in wait mode.
- If SCISWAI is set, SCI clock generation ceases and the SCI module enters a power-conservation state when the CPU is in wait mode. Setting SCISWAI does not affect the state of the receiver enable bit, RE, or the transmitter enable bit, TE.

  If SCISWAI is set, any transmission or reception in progress stops at wait mode entry. The transmission or reception resumes when either an internal or external interrupt brings the CPU out of wait mode. Exiting wait mode by reset aborts any transmission or reception in progress and resets the SCI.

12.5.2.3 Stop Mode

The SCI is inactive during stop mode for reduced power consumption. The STOP instruction does not affect the SCI register states, but the SCI bus clock will be disabled. The SCI operation resumes from where it left off after an external interrupt brings the CPU out of stop mode. Exiting stop mode by reset aborts any transmission or reception in progress and resets the SCI.

The receive input active edge detect circuit is still active in stop mode. An active edge on the receive input can be used to bring the CPU out of stop mode.

12.5.3 Interrupt Operation

This section describes the interrupt originated by the SCI block.The MCU must service the interrupt requests. Table 12-20 lists the eight interrupt sources of the SCI.

Table 12-20. SCI Interrupt Sources

Interrupt	Source	Local Enable	Description
TDRE	SCISR1[7]	TIE	Active high level. Indicates that a byte was transferred from SCIDRH/L to the transmit shift register.
TC	SCISR1[6]	TCIE	Active high level. Indicates that a transmit is complete.
RDRF	SCISR1[5]	RIE	Active high level. The RDRF interrupt indicates that received data is available in the SCI data register.
OR	SCISR1[3]		Active high level. This interrupt indicates that an overrun condition has occurred.
IDLE	SCISR1[4]	ILIE	Active high level. Indicates that receiver input has become idle.

14.1.2 Modes of Operation

The IIC functions the same in normal, special, and emulation modes. It has two low power modes: wait and stop modes.

14.1.3 Block Diagram

The block diagram of the IIC module is shown in Figure 14-1.

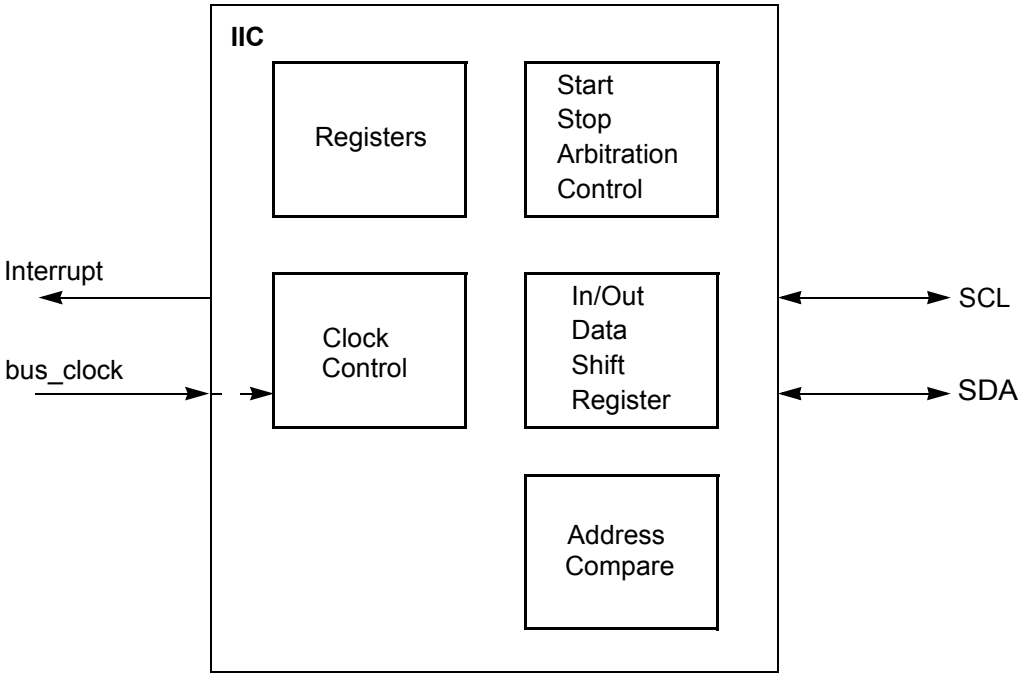


Figure 14-1. IIC Block Diagram

- In linear attack operation SSGAMPB (SSGAMP's buffer) will be increased by SSGAA every SSGSSGDUR + 1 tone cycle. In linear decay operation, SSGAMPB will be decreased by SSGAA every SSGDUR + 1 tone cycle. See below linear attack/decay formula.
- In gong attack operation SSGAMPB will be increased by SSGAMPB/32 every SSGDUR + 1 tone cycle. In gong decay operation SSGAMPB will be decreased by SSGAMPB/32 every SSGDUR + 1 tone cycle. See below gong attack/decay formula.
- In exponential attack operation SSGAMPB will multiply with 2 then add 1 every SSGDUR + 1 tone cycle. In exponential decay operation, SSGAMPB will be divided by 2 every SSGDUR + 1 tone cycle. See below exponential attack/decay formula.

Linear attack operation:

```
SSGAMPB = SSGAMP;
do{
    SSGAMPB = SSGAMPB + SSGAA_buf;
} While (SSGAMPB < AT_buf)
```

Where : AT_buf is the internal buffer of amplitude threshold register SSGAT.
SSGAA_buf is the internal buffer of SSGAA.

Linear decay operation:

```
SSGAMPB = SSGAMP;
do{
    SSGAMPB = SSGAMPB - SSGAA_buf;
} While (SSGAMPB > AT_buf)
```

Where : AT_buf is the internal buffer of amplitude threshold register SSGAT.
SSGAA_buf is the internal buffer of SSGAA.

21.1 Introduction

The FTMRZ64K2K module implements the following:

- 64 KB of P-Flash (Program Flash) memory
- 2 KB of EEPROM memory

The Flash memory is ideal for single-supply applications allowing for field reprogramming without requiring external high voltage sources for program or erase operations. The Flash module includes a memory controller that executes commands to modify Flash memory contents. The user interface to the memory controller consists of the indexed Flash Common Command Object (FCCOB) register which is written to with the command, global address, data, and any required command parameters. The memory controller must complete the execution of a command before the FCCOB register can be written to with a new command.

CAUTION

A Flash word or phrase must be in the erased state before being programmed. Cumulative programming of bits within a Flash word or phrase is not allowed.

The Flash memory may be read as bytes and aligned words. Read access time is one bus cycle for bytes and aligned words. For misaligned words access, the CPU has to perform twice the byte read access command. For Flash memory, an erased bit reads 1 and a programmed bit reads 0.

It is possible to read from P-Flash memory while some commands are executing on EEPROM memory. It is not possible to read from EEPROM memory while a command is executing on P-Flash memory. Simultaneous P-Flash and EEPROM operations are discussed in Section 21.4.6.

Both P-Flash and EEPROM memories are implemented with Error Correction Codes (ECC) that can resolve single bit faults and detect double bit faults. For P-Flash memory, the ECC implementation requires that programming be done on an aligned 8 byte basis (a Flash phrase). Since P-Flash memory is always read by half-phrase, only one single bit fault in an aligned 4 byte half-phrase containing the byte or word accessed will be corrected.

21.1.1 Glossary

Command Write Sequence — An MCU instruction sequence to execute built-in algorithms (including program and erase) on the Flash memory.

EEPROM Memory — The EEPROM memory constitutes the nonvolatile memory store for data.

EEPROM Sector — The EEPROM sector is the smallest portion of the EEPROM memory that can be erased. The EEPROM sector consists of 4 bytes.

21.4.5.3 Valid Flash Module Commands

Table 21-29 present the valid Flash commands, as enabled by the combination of the functional MCU mode (Normal SingleChip NS, Special Singlechip SS) with the MCU security state (Unsecured, Secured).

Table 21-29. Flash Commands by Mode and Security State

FCMD	Command	Unsecured		Secured	
		NS (1)	SS(2)	NS (3)	SS(4)
0x01	Erase Verify All Blocks	*	*	*	
0x02	Erase Verify Block	*	*	*	
0x03	Erase Verify P-Flash Section	*	*	*	
0x04	Read Once	*	*	*	
0x06	Program P-Flash	*	*	*	
0x07	Program Once	*	*	*	
0x08	Erase All Blocks		*		
0x09	Erase Flash Block	*	*	*	
0x0A	Erase P-Flash Sector	*	*	*	
0x0B	Unsecure Flash		*		
0x0C	Verify Backdoor Access Key	*		*	
0x0D	Set User Margin Level	*	*	*	
0x0E	Set Field Margin Level		*		
0x10	Erase Verify EEPROM Section	*	*	*	
0x11	Program EEPROM	*	*	*	
0x12	Erase EEPROM Sector	*	*	*	
0x13	Protection Override	*	*	*	

1. Unsecured Normal Single Chip mode
2. Unsecured Special Single Chip mode.
3. Secured Normal Single Chip mode.
4. Secured Special Single Chip mode.Please refer to Section 21.5.2.