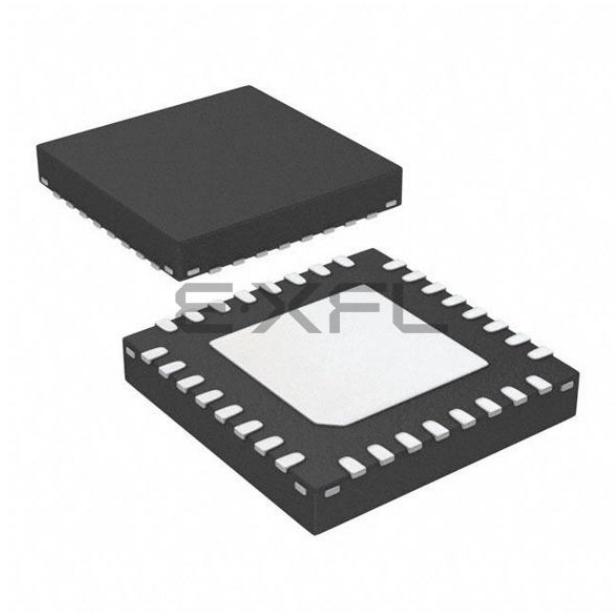


Welcome to E-XFL.COM

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	Obsolete
Core Processor	ARM® Cortex®-M0
Core Size	32-Bit Single-Core
Speed	48MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, WDT
Number of I/O	25
Program Memory Size	32KB (32K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	8K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	-
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	32-VQFN Exposed Pad
Supplier Device Package	32-HVQFN (7x7)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/em773fhn33-551

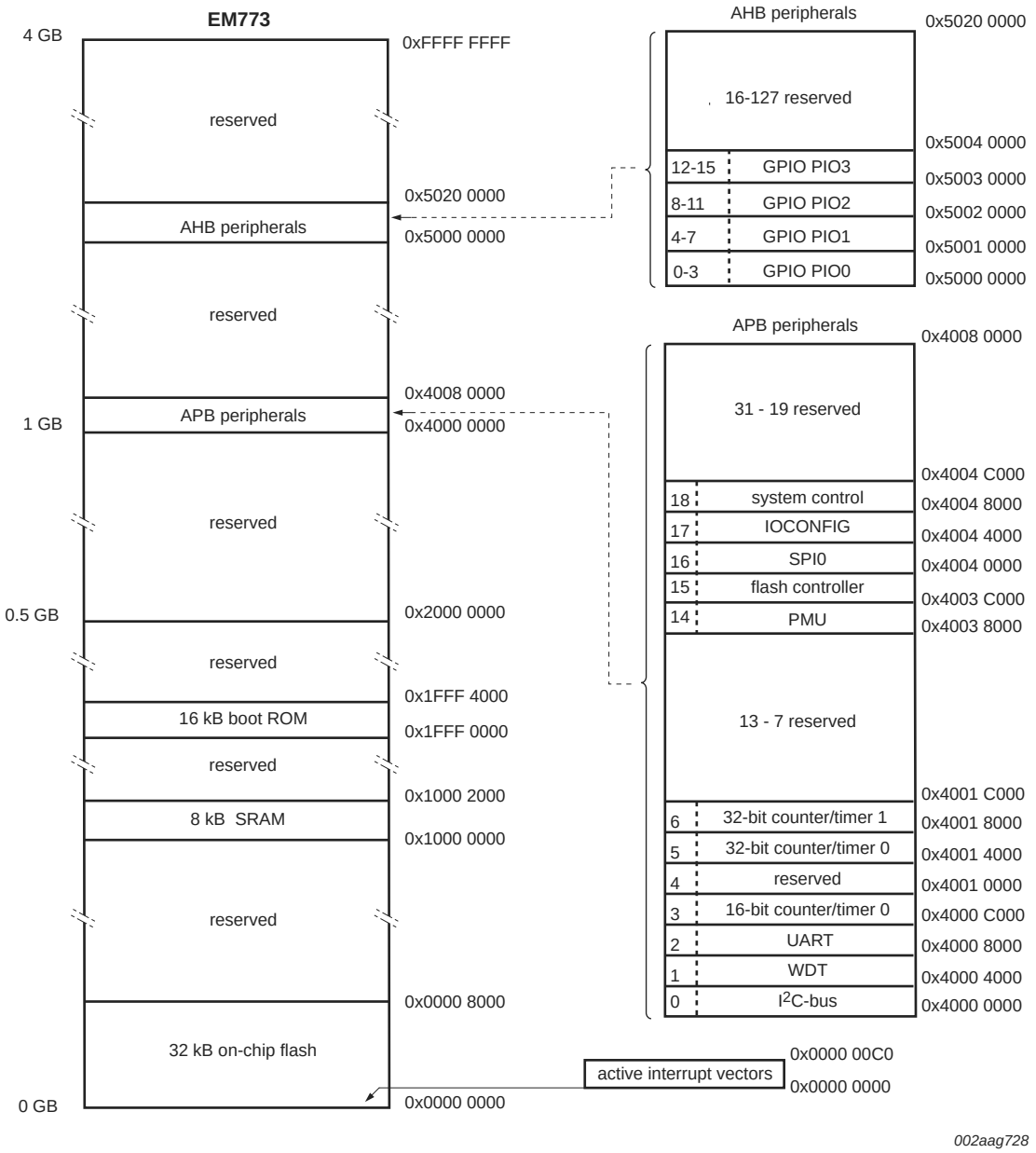


Fig 2. EM773 memory map

Table 56. IOCON_PIO0_5 register (IOCON_PIO0_5 address 0x4004 4034) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		000	Selects function PIO0_5 (open-drain pin).	
		001	Selects I2C function SDA (open-drain pin).	
		010 to 111	Reserved.	
7:3		-	Reserved.	00000
9:8	I2CMODE		Selects I2C mode (see Section 6.3.4).	00
		00[1]	Standard mode/ Fast-mode I2C.	
		01[1]	Standard I/O functionality	
		10	Fast-mode Plus I2C	
		11	Reserved.	
31:10	-	-	Reserved.	-

[1] Select Standard mode (I2CMODE = 00, default) or Standard I/O functionality (I2CMODE = 01) if the pin function is GPIO (FUNC = 000).

9.1 How to read this chapter

The UART block is identical for all EM773 parts.

9.2 Basic configuration

The UART is configured using the following registers:

1. Pins: The UART pins must be configured in the IOCONFIG register block (Section 6.4.1).

   Remark: If the modem input pins are used, the modem function location must be also selected in the UART location registers (Section 6.4.2)
2. Power: In the SYSAHBCLKCTRL register, set bit 12 (Table 17).
3. Peripheral clock: Enable the UART peripheral clock by writing to the UARTCLKDIV register (Table 19).

9.3 Features

- 16-byte receive and transmit FIFOs.
- Register locations conform to ‘550 industry standard.
- Receiver FIFO trigger points at 1, 4, 8, and 14 bytes.
- Built-in baud rate generator.
- UART allows for implementation of either software or hardware flow control.
- RS-485/EIA-485 9-bit mode support with output enable.
- Modem control.

9.4 Pin description

Table 88. UART pin description

Pin	Type	Description
RXD	Input	**Serial Input.** Serial receive data.
TXD	Output	**Serial Output.** Serial transmit data.
$\overline{\text{RTS}}$	Output	Request To Send. RS-485 direction control pin.
$\overline{\text{DTR}}$	Output	Data Terminal Ready.
$\overline{\text{CTS}}$	Input	Clear To Send.

Table 123. I²C Data buffer register (I2C0DATA_BUFFER - 0x4000 002C) bit description

Bit	Symbol	Description	Reset value
7:0	Data	This register holds contents of the 8 MSBs of the I2DAT shift register.	0
31:8	-	Reserved. The value read from a reserved bit is not defined.	0

10.8.10 I²C Mask registers (I2C0MASK[0, 1, 2, 3] - 0x4000 00[30, 34, 38, 3C])

The four mask registers each contain seven active bits (7:1). Any bit in these registers which is set to '1' will cause an automatic compare on the corresponding bit of the received address when it is compared to the I2ADDRn register associated with that mask register. In other words, bits in an I2ADDRn register which are masked are not taken into account in determining an address match.

On reset, all mask register bits are cleared to '0'.

The mask register has no effect on comparison to the General Call address ("0000000").

Bits(31:8) and bit(0) of the mask registers are unused and should not be written to. These bits will always read back as zeros.

When an address-match interrupt occurs, the processor will have to read the data register (I2DAT) to determine what the received address was that actually caused the match.

Table 124. I²C Mask registers (I2C0MASK[0, 1, 2, 3] - 0x4000 00[30, 34, 38, 3C]) bit description

Bit	Symbol	Description	Reset value
0	-	Reserved. User software should not write ones to reserved bits. This bit reads always back as 0.	0
7:1	MASK	Mask bits.	0x00
31:8	-	Reserved. The value read from reserved bits is undefined.	0

10.9 I²C operating modes

In a given application, the I²C block may operate as a master, a slave, or both. In the slave mode, the I²C hardware looks for any one of its four slave addresses and the General Call address. If one of these addresses is detected, an interrupt is requested. If the processor wishes to become the bus master, the hardware waits until the bus is free before the master mode is entered so that a possible slave operation is not interrupted. If bus arbitration is lost in the master mode, the I²C block switches to the slave mode immediately and can detect its own slave address in the same serial transfer.

10.9.1 Master Transmitter mode

In this mode data is transmitted from master to slave. Before the master transmitter mode can be entered, the I2CONSET register must be initialized as shown in Table 125. I2EN must be set to 1 to enable the I²C function. If the AA bit is 0, the I²C interface will not acknowledge any address when another device is master of the bus, so it can not enter slave mode. The STA, STO and SI bits must be 0. The SI Bit is cleared by writing 1 to the SIC bit in the I2CONCLR register. THe STA bit should be cleared after writing the slave address.

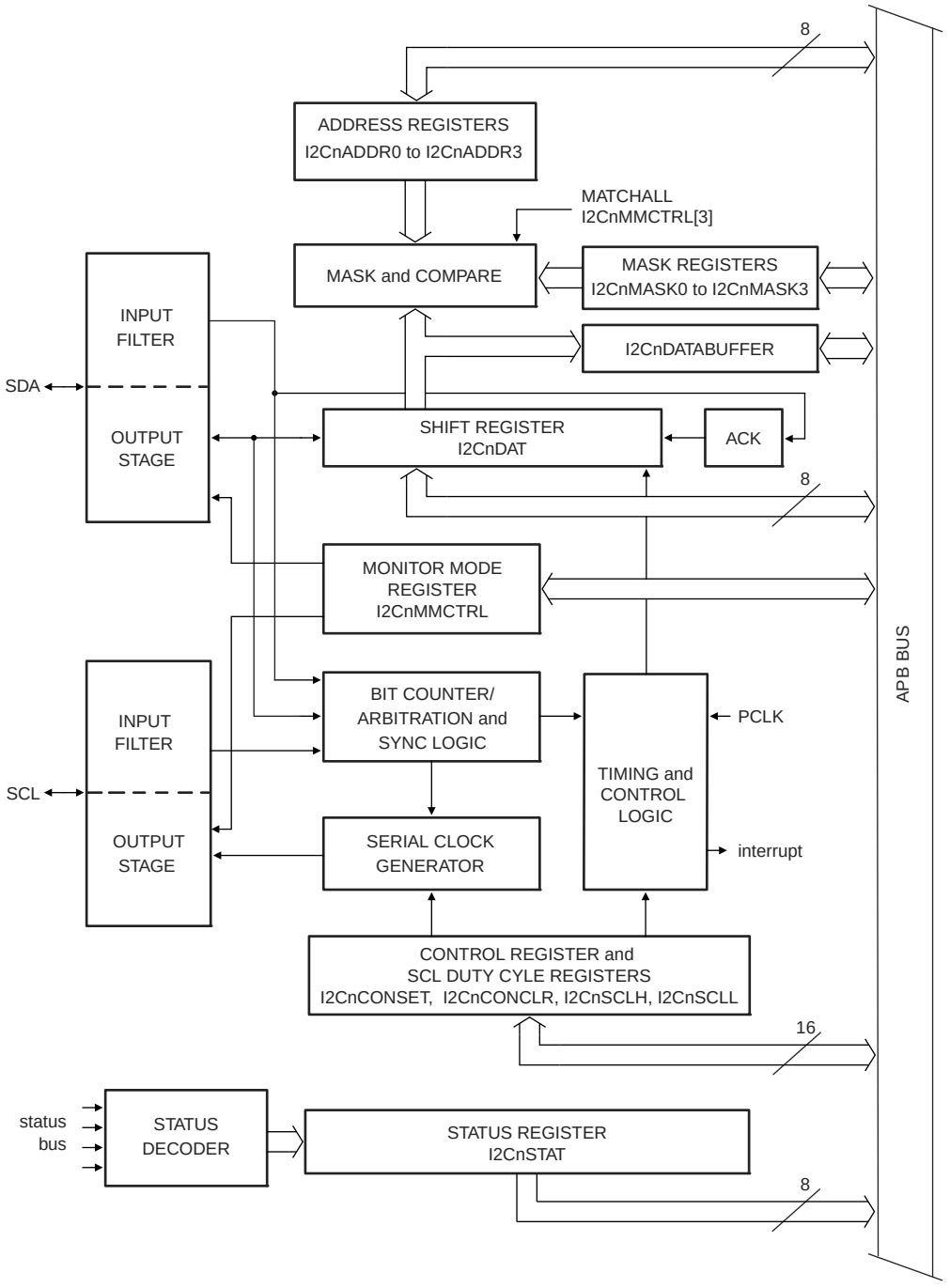
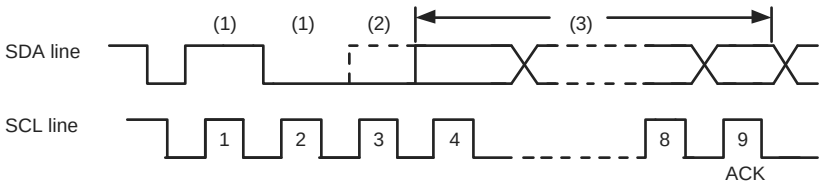


Fig 21. I²C serial interface block diagram

10.10.1 Input filters and output stages

Input signals are synchronized with the internal clock, and spikes shorter than three clocks are filtered out.

The output for I²C is a special pad designed to conform to the I²C specification.



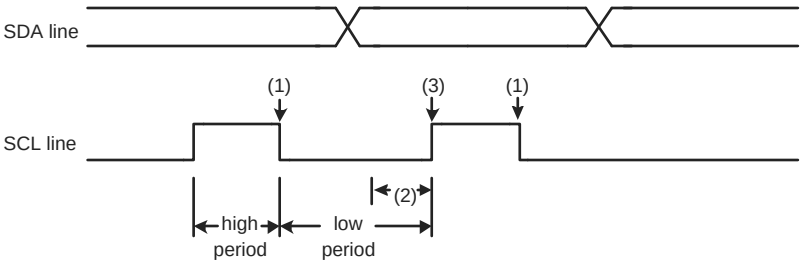
(1) Another device transmits serial data.

(2) Another device overrules a logic (dotted line) transmitted this I2C master by pulling the SDA line low. Arbitration is lost, and this I2C enters Slave Receiver mode.

(3) This I2C is in Slave Receiver mode but still generates clock pulses until the current byte has been transmitted. This I2C will not generate clock pulses for the next byte. Data on SDA originates from the new master once it has won arbitration.

Fig 22. Arbitration procedure

The synchronization logic will synchronize the serial clock generator with the clock pulses on the SCL line from another device. If two or more master devices generate clock pulses, the “mark” duration is determined by the device that generates the shortest “marks,” and the “space” duration is determined by the device that generates the longest “spaces”. Figure 23 shows the synchronization procedure.



(1) Another device pulls the SCL line low before this I2C has timed a complete high time. The other device effectively determines the (shorter) HIGH period.

(2) Another device continues to pull the SCL line low after this I2C has timed a complete low time and released SCL. The I2C clock generator is forced to wait until SCL goes HIGH. The other device effectively determines the (longer) LOW period.

(3) The SCL line is released , and the clock generator begins timing the HIGH time.

Fig 23. Serial clock synchronization

A slave may stretch the space duration to slow down the bus master. The space duration may also be stretched for handshaking purposes. This can be done after each bit or after a complete byte transfer. the I2C block will stretch the SCL space duration after a byte has been transmitted or received and the acknowledge bit has been transferred. The serial interrupt flag (SI) is set, and the stretching continues until the serial interrupt flag is cleared.

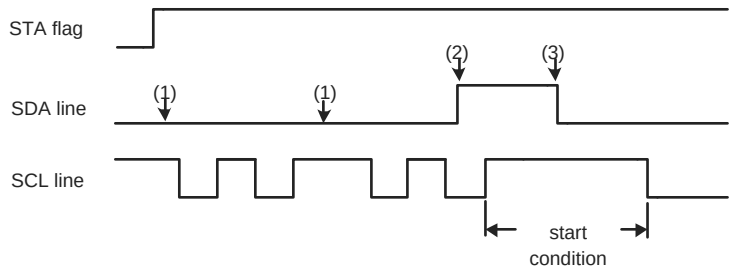
10.10.7 Serial clock generator

This programmable clock pulse generator provides the SCL clock pulses when the I2C block is in the master transmitter or master receiver mode. It is switched off when the I2C block is in a slave mode. The I2C output clock frequency and duty cycle is programmable

10.11.6.4 I²C-bus obstructed by a LOW level on SCL or SDA

An I²C-bus hang-up can occur if either the SDA or SCL line is held LOW by any device on the bus. If the SCL line is obstructed (pulled LOW) by a device on the bus, no further serial transfer is possible, and the problem must be resolved by the device that is pulling the SCL bus line LOW.

Typically, the SDA line may be obstructed by another device on the bus that has become out of synchronization with the current bus master by either missing a clock, or by sensing a noise pulse as a clock. In this case, the problem can be solved by transmitting additional clock pulses on the SCL line (see Figure 30). The I²C interface does not include a dedicated timeout timer to detect an obstructed bus, but this can be implemented using another timer in the system. When detected, software can force clocks (up to 9 may be required) on SCL until SDA is released by the offending device. At that point, the slave may still be out of synchronization, so a START should be generated to insure that all I²C peripherals are synchronized.



(1) Unsuccessful attempt to send a START condition.

(2) SDA line is released.

(3) Successful attempt to send a START condition. State 08H is entered.

Fig 30. Recovering from a bus obstruction caused by a LOW level on SDA

10.11.6.5 Bus error

A bus error occurs when a START or STOP condition is detected at an illegal position in the format frame. Examples of illegal positions are during the serial transfer of an address byte, a data bit, or an acknowledge bit.

The I²C hardware only reacts to a bus error when it is involved in a serial transfer either as a master or an addressed slave. When a bus error is detected, the I²C block immediately switches to the not addressed slave mode, releases the SDA and SCL lines, sets the interrupt flag, and loads the status register with 0x00. This status code may be used to vector to a state service routine which either attempts the aborted serial transfer again or simply recovers from the error condition as shown in Table 135.

10.11.7 I²C state service routines

This section provides examples of operations that must be performed by various I²C state service routines. This includes:

- Initialization of the I²C block after a Reset.
- I²C Interrupt Service
- The 26 state service routines providing support for all four I²C operating modes.

11.1 How to read this chapter

The SPI block is identical for all EM773 parts.

Remark: The SPI block includes the full SSP feature set, and all register names use the SSP prefix.

11.2 Basic configuration

The SPI0 is configured using the following registers:

1. Pins: The SPI pins must be configured in the IOCONFIG register block. In addition, use the IOCON_LOC register (see Section 6.4.2) to select a location for the SCK0 function.
2. Power: In the SYSAHBCLKCTRL register, set bit 11 and bit 18 (Table 17).
3. Peripheral clock: Enable the SPI0 peripheral clock by writing to the SSP0CLKDIV registers (Section 3.4.15).
4. Reset: Before accessing the SPI blocks, ensure that the SSP_RST_N bits (bit 0 and bit 2) in the PRESETCTRL register (Table 5) is set to 1. This de-asserts the reset signal to the SPI blocks.

11.3 Features

- Compatible with Motorola SPI, 4-wire TI SSI, and National Semiconductor Microwire buses.
- Synchronous Serial Communication.
- Supports master or slave operation.
- Eight-frame FIFOs for both transmit and receive.
- 4-bit to 16-bit frame.

11.4 General description

The SPI/SSP is a Synchronous Serial Port (SSP) controller capable of operation on a SPI, 4-wire SSI, or Microwire bus. It can interact with multiple masters and slaves on the bus. Only a single master and a single slave can communicate on the bus during a given data transfer. Data transfers are in principle full duplex, with frames of 4 bits to 16 bits of data flowing from the master to the slave and from the slave to the master. In practice it is often the case that only one of these data flows carries meaningful data.

The EM773 has one SPI/Synchronous Serial Port controller.

11.6 Clocking and power control

The SPI block is gated by the AHBCLKCTRL register (see Table 17). The peripheral SPI clock, which is used by the SPI clock divider and prescaler, is controlled by the SSP0CLKDIV registers (see Section 3.4.15).

The SPI0_PCLK clocks can be disabled in SSP0CLKDIV registers (see Section 3.4.15), and the SPI blocks can be disabled in the AHBCLKCTRL register (Table 17) for power savings.

Remark: Before accessing the SPI blocks, ensure that the SSP0_RST_N bit (bit 0) in the PRESETCTRL register (Table 5) is set to 1. This de-asserts the reset signal to the SSP block.

11.7 Register description

The register addresses of the SPI controllers are shown in Table 137.

Remark: Register names use the SSP prefix to indicate that the SPI controllers have full SSP capabilities.

Table 137. Register overview: SPI0 (base address 0x4004 0000)

Name	Access	Address offset	Description	Reset Value[1]
SSP0CR0	R/W	0x000	Control Register 0. Selects the serial clock rate, bus type, and data size.	0
SSP0CR1	R/W	0x004	Control Register 1. Selects master/slave and other modes.	0
SSP0DR	R/W	0x008	Data Register. Writes fill the transmit FIFO, and reads empty the receive FIFO.	0
SSP0SR	RO	0x00C	Status Register	0x0000 0003
SSP0CPSR	R/W	0x010	Clock Prescale Register	0
SSP0IMSC	R/W	0x014	Interrupt Mask Set and Clear Register	0
SSP0RIS	RO	0x018	Raw Interrupt Status Register	0x0000 0008
SSP0MIS	RO	0x01C	Masked Interrupt Status Register	0
SSP0ICR	WO	0x020	SSPICR Interrupt Clear Register	NA

[1] Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

Table 144: SPI/SSP Raw Interrupt Status register (SSP0RIS - address 0x4004 0018) bit description

Bit	Symbol	Description	Reset Value
0	RORRIS	This bit is 1 if another frame was completely received while the RxFIFO was full. The ARM spec implies that the preceding frame data is overwritten by the new frame data when this occurs.	0
1	RTRIS	This bit is 1 if the Rx FIFO is not empty, and has not been read for a "timeout period". The timeout period is the same for master and slave modes and is determined by the SSP bit rate: 32 bits at PCLK / (CPSDVSR × [SCR+1]).	0
2	RXRIS	This bit is 1 if the Rx FIFO is at least half full.	0
3	TXRIS	This bit is 1 if the Tx FIFO is at least half empty.	1
31:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

11.7.8 SPI/SSP Masked Interrupt Status Register

This read-only register contains a 1 for each interrupt condition that is asserted and enabled in the SSPIMSC registers. When an SPI interrupt occurs, the interrupt service routine should read this register to determine the cause(s) of the interrupt.

Table 145: SPI/SSP Masked Interrupt Status register (SSP0MIS - address 0x4004 001C) bit description

Bit	Symbol	Description	Reset Value
0	RORMIS	This bit is 1 if another frame was completely received while the RxFIFO was full, and this interrupt is enabled.	0
1	RTMIS	This bit is 1 if the Rx FIFO is not empty, has not been read for a "timeout period", and this interrupt is enabled. The timeout period is the same for master and slave modes and is determined by the SSP bit rate: 32 bits at PCLK / (CPSDVSR × [SCR+1]).	0
2	RXMIS	This bit is 1 if the Rx FIFO is at least half full, and this interrupt is enabled.	0
3	TXMIS	This bit is 1 if the Tx FIFO is at least half empty, and this interrupt is enabled.	0
31:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

11.7.9 SPI/SSP Interrupt Clear Register

Software can write one or more one(s) to this write-only register, to clear the corresponding interrupt condition(s) in the SPI controller. Note that the other two interrupt conditions can be cleared by writing or reading the appropriate FIFO or disabled by clearing the corresponding bit in SSPIMSC registers.

Table 148. Register overview: 16-bit counter/timer 0 CT16B0 (base address 0x4000 C000)

Name	Access	Address offset	Description	Reset value[1]
TMR16B0IR	R/W	0x000	Interrupt Register (IR). The IR can be written to clear interrupts. The IR can be read to identify which of five possible interrupt sources are pending.	0
TMR16B0TCR	R/W	0x004	Timer Control Register (TCR). The TCR is used to control the Timer Counter functions. The Timer Counter can be disabled or reset through the TCR.	0
TMR16B0TC	R/W	0x008	Timer Counter (TC). The 16-bit TC is incremented every PR+1 cycles of PCLK. The TC is controlled through the TCR.	0
TMR16B0PR	R/W	0x00C	Prescale Register (PR). When the Prescale Counter (below) is equal to this value, the next clock increments the TC and clears the PC.	0
TMR16B0PC	R/W	0x010	Prescale Counter (PC). The 16-bit PC is a counter which is incremented to the value stored in PR. When the value in PR is reached, the TC is incremented and the PC is cleared. The PC is observable and controllable through the bus interface.	0
TMR16B0MCR	R/W	0x014	Match Control Register (MCR). The MCR is used to control if an interrupt is generated and if the TC is reset when a Match occurs.	0
TMR16B0MR0	R/W	0x018	Match Register 0 (MR0). MR0 can be enabled through the MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt every time MR0 matches the TC.	0
TMR16B0MR1	R/W	0x01C	Match Register 1 (MR1). See MR0 description.	0
TMR16B0MR2	R/W	0x020	Match Register 2 (MR2). See MR0 description.	0
TMR16B0MR3	R/W	0x024	Match Register 3 (MR3). See MR0 description.	0
TMR16B0CCR	R/W	0x028	Capture Control Register (CCR). The CCR controls which edges of the capture inputs are used to load the Capture Registers and whether or not an interrupt is generated when a capture takes place.	0
TMR16B0CR0	RO	0x02C	Capture Register 0 (CR0). CR0 is loaded with the value of TC when there is an event on the CT16B0_CAP0 input.	0
TMR16B0EMR	R/W	0x03C	External Match Register (EMR). The EMR controls the match function and the external match pins CT16B0_MAT[2:0].	0
-	-	0x040 - 0x06C	reserved	-
TMR16B0CTCR	R/W	0x070	Count Control Register (CTCR). The CTCR selects between Timer and Counter mode, and in Counter mode selects the signal and edge(s) for counting.	0
TMR16B0PWMC	R/W	0x074	PWM Control Register (PWMCON). The PWMCON enables PWM mode for the external match pins CT16B0_MAT[2:0].	0

[1] Reset value reflects the data stored in used bits only. It does not include reserved bits content.

12.8.1 Interrupt Register (TMR16B0IR)

The Interrupt Register (IR) consists of four bits for the match interrupts and one bit for the capture interrupt. If an interrupt is generated then the corresponding bit in the IR will be HIGH. Otherwise, the bit will be LOW. Writing a logic one to the corresponding IR bit will reset the interrupt. Writing a zero has no effect.

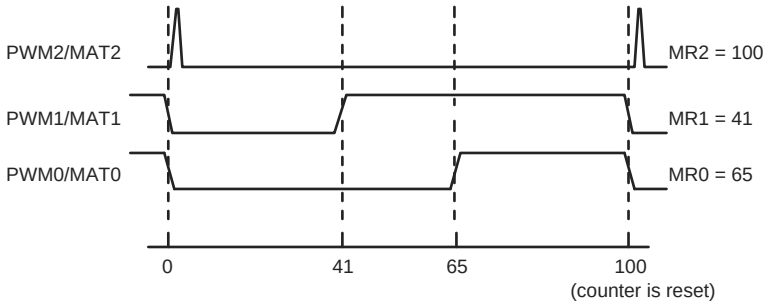


Fig 39. Sample PWM waveforms with a PWM cycle length of 100 (selected by MR3) and MAT3:0 enabled as PWM outputs by the PWCON register.

12.9 Example timer operation

Figure 40 shows a timer configured to reset the count and generate an interrupt on match. The prescaler is set to 2 and the match register set to 6. At the end of the timer cycle where the match occurs, the timer count is reset. This gives a full length cycle to the match value. The interrupt indicating that a match occurred is generated in the next clock after the timer reached the match value.

Figure 41 shows a timer configured to stop and generate an interrupt on match. The prescaler is again set to 2 and the match register set to 6. In the next clock after the timer reaches the match value, the timer enable bit in TCR is cleared, and the interrupt indicating that a match occurred is generated.

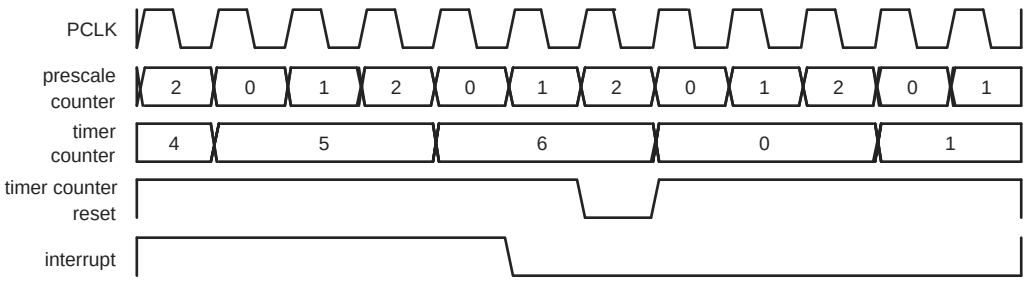


Fig 40. A timer cycle in which PR=2, MRx=6, and both interrupt and reset on match are enabled

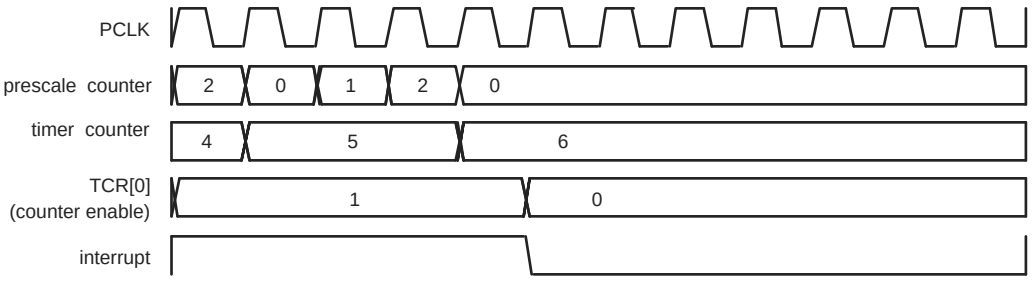


Fig 41. A timer cycle in which PR=2, MRx=6, and both interrupt and stop on match are enabled

Table 174: Watchdog Timer Warning Interrupt register (WDWARNINT - 0x4000 4014) bit description

Bit	Symbol	Description	Reset value
9:0	WARNINT	Watchdog warning interrupt compare value.	0
31:10	-	Reserved. Read value is undefined, only zero should be written.	-

14.7.6 Watchdog Timer Window register

The WDWINDOW register determines the highest WDTV value allowed when a watchdog feed is performed. If a feed valid sequence completes prior to WDTV reaching the value in WDWINDOW, a watchdog event will occur.

WDWINDOW resets to the maximum possible WDTV value, so windowing is not in effect.

Table 175: Watchdog Timer Window register (WDWINDOW - 0x4000 4018) bit description

Bit	Symbol	Description	Reset value
23:0	WINDOW	Watchdog window value.	0xFF FFFF
31:24	-	Reserved. Read value is undefined, only zero should be written.	-

14.7.7 Watchdog timing examples

The following figures illustrate several aspects of Watchdog Timer operation.

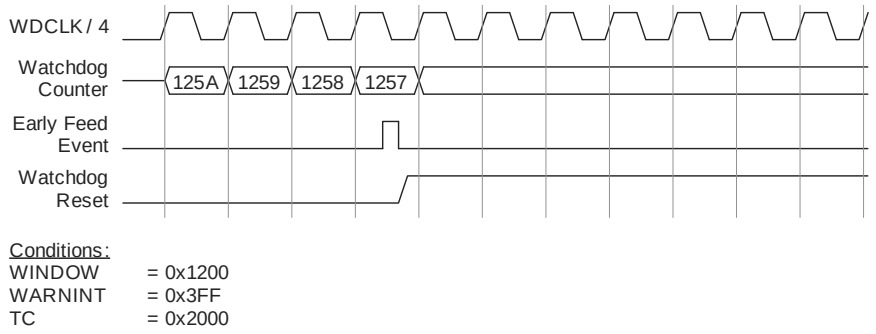


Fig 48. Early Watchdog Feed with Windowed Mode Enabled

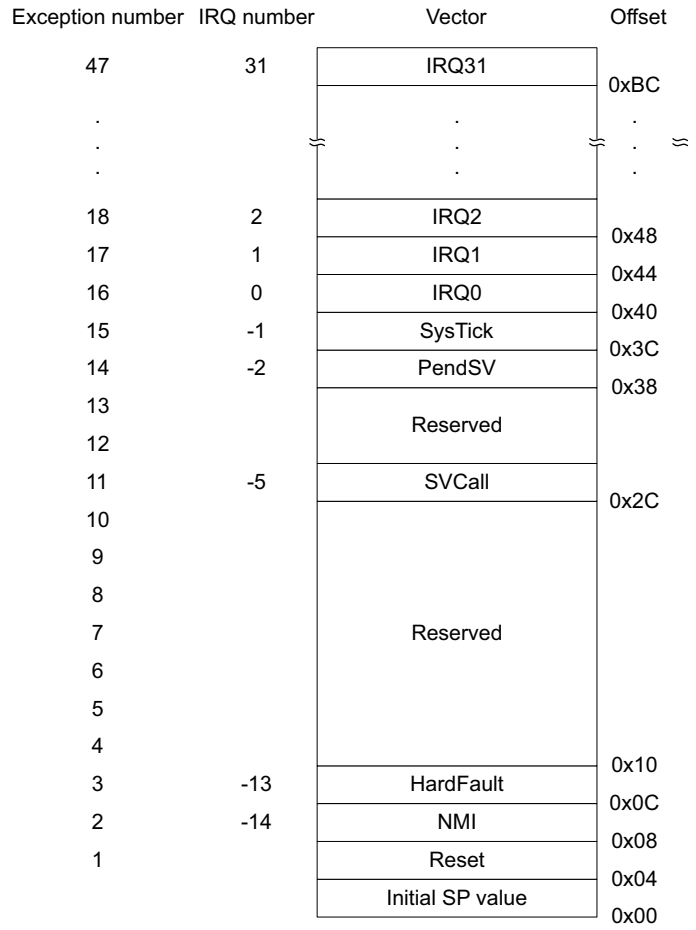


Fig 65. Vector table

The vector table is fixed at address `0x00000000`.

21.3.3.5 Exception priorities

As Table 21–230 shows, all exceptions have an associated priority, with:

- a lower priority value indicating a higher priority
- configurable priorities for all exceptions except Reset, HardFault, and NMI.

If software does not configure any priorities, then all exceptions with a configurable priority have a priority of 0. For information about configuring exception priorities see

- Section 21–21.5.3.7
- Section 21–21.5.2.6.

Remark: Configurable priority values are in the range 0-3. The Reset, HardFault, and NMI exceptions, with fixed negative priority values, always have higher priority than any other exception.

21.4.4.3.2 Operation

LDR, LDRB, U, LDRSB and LDRSH load the register specified by *Rt* with either a word, zero extended byte, zero extended halfword, sign extended byte or sign extended halfword value from memory.

STR, STRB and STRH store the word, least-significant byte or lower halfword contained in the single register specified by *Rt* into memory.

The memory address to load from or store to is the sum of the values in the registers specified by *Rn* and *Rm*.

21.4.4.3.3 Restrictions

In these instructions:

- *Rt*, *Rn*, and *Rm* must only specify R0-R7.
- the computed memory address must be divisible by the number of bytes in the load or store, see Section 21–21.4.3.4.

21.4.4.3.4 Condition flags

These instructions do not change the flags.

21.4.4.3.5 Examples

```
STR    R0, [R5, R1]         ; Store value of R0 into an address equal to
                                 ; sum of R5 and R1

LDRSH  R1, [R2, R3]         ; Load a halfword from the memory address
                                 ; specified by (R2 + R3), sign extend to 32-bits

                            ; and write to R1.
```

21.4.4.4 LDR, PC-relative

Load register (literal) from memory.

21.4.4.4.1 Syntax

LDR *Rt*, *label*

where:

Rt is the register to load.

label is a PC-relative expression. See Section 21–21.4.3.5.

21.4.4.4.2 Operation

Loads the register specified by *Rt* from the word in memory specified by *label*.

21.4.4.4.3 Restrictions

In these instructions, *label* must be within 1020 bytes of the current PC and word aligned.

21.4.4.4.4 Condition flags

These instructions do not change the flags.

21.4.4.4.5 Examples

```
LDR     R0, LookUpTable   ; Load R0 with a word of data from an address
                                ; labelled as LookUpTable.

LDR     R3, [PC, #100]    ; Load R3 with memory word at (PC + 100).
```

21.4.4.5 LDM and STM

Load and Store Multiple registers.

21.4.4.5.1 Syntax

LDM *Rn*{!}, reglist

STM *Rn*!, reglist

where:

Rn is the register on which the memory addresses are based.

! writeback suffix.

reglist is a list of one or more registers to be loaded or stored, enclosed in braces. It can contain register ranges. It must be comma separated if it contains more than one register or register range, see Section 21–21.4.4.5.5.

LDMIA and LDMFD are synonyms for LDM. LDMIA refers to the base register being Incremented After each access. LDMFD refers to its use for popping data from Full Descending stacks.

STMIA and STMEA are synonyms for STM. STMIA refers to the base register being Incremented After each access. STMEA refers to its use for pushing data onto Empty Ascending stacks.

21.4.4.5.2 Operation

LDM instructions load the registers in *reglist* with word values from memory addresses based on *Rn*.

STM instructions store the word values in the registers in *reglist* to memory addresses based on *Rn*.

The memory addresses used for the accesses are at 4-byte intervals ranging from the value in the register specified by *Rn* to the value in the register specified by *Rn* + 4 * (*n*-1), where *n* is the number of registers in *reglist*. The accesses happens in order of increasing register numbers, with the lowest numbered register using the lowest memory address and the highest number register using the highest memory address. If the writeback suffix is specified, the value in the register specified by *Rn* + 4 **n* is written back to the register specified by *Rn*.

21.4.4.5.3 Restrictions

In these instructions:

- *reglist* and *Rn* are limited to R0-R7.
- the writeback suffix must always be used unless the instruction is an LDM where reglist also contains *Rn*, in which case the writeback suffix must not be used.

Table 239. Branch and control instructions

Mnemonic	Brief description	See
BL	Branch with Link	Section 21–21.4.6.1
BLX	Branch indirect with Link	Section 21–21.4.6.1
BX	Branch indirect	Section 21–21.4.6.1

21.4.6.1 B, BL, BX, and BLX

Branch instructions.

21.4.6.1.1 Syntax

B{*cond*} *label*

BL *label*

BX *Rm*

BLX *Rm*

where:

cond is an optional condition code, see Section 21–21.4.3.6.

label is a PC-relative expression. See Section 21–21.4.3.5.

Rm is a register providing the address to branch to.

21.4.6.1.2 Operation

All these instructions cause a branch to the address indicated by *label* or contained in the register specified by *Rm*. In addition:

- The BL and BLX instructions write the address of the next instruction to LR, the link register R14.
- The BX and BLX instructions result in a HardFault exception if bit[0] of *Rm* is 0.

BL and BLX instructions also set bit[0] of the LR to 1. This ensures that the value is suitable for use by a subsequent POP {PC} or BX instruction to perform a successful return branch.

Table 240 shows the ranges for the various branch instructions.

Table 240. Branch ranges

Instruction	Branch range
B *label*	−2 KB to +2 KB
B*cond label*	−256 bytes to +254 bytes
BL *label*	−16 MB to +16 MB
BX *Rm*	Any value in register
BLX *Rm*	Any value in register

21.4.6.1.3 Restrictions

In these instructions:

- Do not use SP or PC in the BX or BLX instruction.

21.4.7.8.4 Condition flags

This instruction does not change the flags.

21.4.7.8.5 Examples

```
NOP  ; No operation
```

21.4.7.9 SEV

Send Event.

21.4.7.9.1 Syntax

SEV

21.4.7.9.2 Operation

SEV causes an event to be signaled to all processors within a multiprocessor system. It also sets the local event register, see Section 21–21.3.5.

See also Section 21–21.4.7.11.

21.4.7.9.3 Restrictions

There are no restrictions.

21.4.7.9.4 Condition flags

This instruction does not change the flags.

21.4.7.9.5 Examples

```
SEV ; Send Event
```

21.4.7.10 SVC

Supervisor Call.

21.4.7.10.1 Syntax

SVC #*imm*

where:

imm is an integer in the range 0-255.

21.4.7.10.2 Operation

The SVC instruction causes the SVC exception.

imm is ignored by the processor. If required, it can be retrieved by the exception handler to determine what service is being requested.

21.4.7.10.3 Restrictions

There are no restrictions.

21.4.7.10.4 Condition flags

This instruction does not change the flags.

21.4.7.10.5 Examples

```
SVC  #0x32 ; Supervisor Call (SVC handler can extract the immediate value
               ; by locating it via the stacked PC)
```

21.4.7.11 WFE

Wait For Event.

Remark: The WFE instruction is not implemented on the EM773.

21.4.7.11.1 Syntax

WFE

21.4.7.11.2 Operation

If the event register is 0, WFE suspends execution until one of the following events occurs:

- an exception, unless masked by the exception mask registers or the current priority level
- an exception enters the Pending state, if SEVONPEND in the System Control Register is set
- a Debug Entry request, if debug is enabled
- an event signaled by a peripheral or another processor in a multiprocessor system using the SEV instruction.

If the event register is 1, WFE clears it to 0 and completes immediately.

For more information see Section 21–21.3.5.

Remark: WFE is intended for power saving only. When writing software assume that WFE might behave as NOP.

21.4.7.11.3 Restrictions

There are no restrictions.

21.4.7.11.4 Condition flags

This instruction does not change the flags.

21.4.7.11.5 Examples

```
WFE  ; Wait for event
```

21.4.7.12 WFI

Wait for Interrupt.

21.4.7.12.1 Syntax

WFI