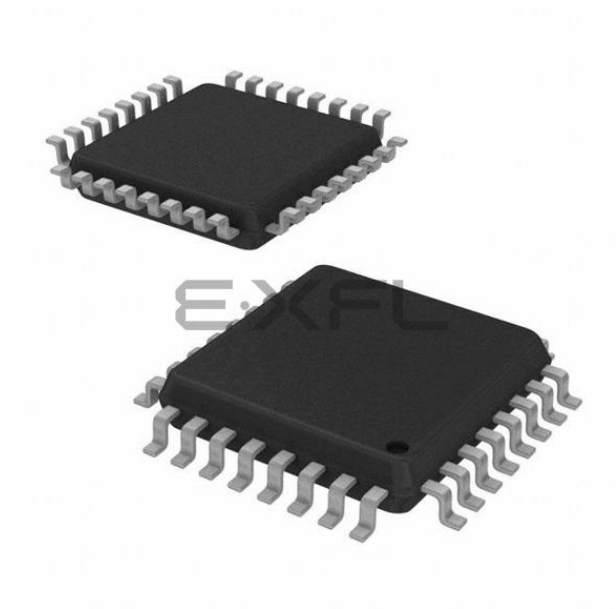


Welcome to E-XFL.COM

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	ST7
Core Size	8-Bit
Speed	8MHz
Connectivity	SCI, SPI
Peripherals	LVD, POR, PWM, WDT
Number of I/O	24
Program Memory Size	16KB (16K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	3.8V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	32-LQFP
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/stmicroelectronics/st72f324k4ta

To obtain the most recent version of this datasheet,
please check at www.st.com>products>technical literature>datasheet.

Please also pay special attention to the Section “KNOWN LIMITATIONS” on page 159.

Legend: x=undefined, R/W=read/write

Notes:

1. The contents of the I/O port DR registers are readable only in output configuration. In input configuration, the values of the I/O pins are returned instead of the DR register contents.
2. The bits associated with unavailable pins must always keep their reset value.
3. The Timer A Input Capture 2 pin is not available (not bonded).
   – In Flash devices:
   The TAIC2HR and TAIC2LR registers are not present. Bit 5 of the TACSR register (ICF2) is forced by hardware to 0. Consequently, the corresponding interrupt cannot be used.
4. The Timer A Output Compare 2 pin is not available (not bonded).
   – The TAOC2HR and TAOC2LR Registers are write only, reading them will return undefined values. Bit 4 of the TACSR register (OCF2) is forced by hardware to 0. Consequently, the corresponding interrupt cannot be used.

Caution: The TAIC2HR and TAIC2LR registers and the ICF2 and OCF2 flags are not present in Flash devices but are present in the emulator. For compatibility with the emulator, it is recommended to perform a dummy access (read or write) to the TAIC2LR and TAOC2LR registers to clear the interrupt flags.

4 FLASH PROGRAM MEMORY

4.1 Introduction

The ST7 dual voltage High Density Flash (HDFlash) is a non-volatile memory that can be electrically erased as a single block or by individual sectors and programmed on a Byte-by-Byte basis using an external V_{PP} supply.

The HDFlash devices can be programmed and erased off-board (plugged in a programming tool) or on-board using ICP (In-Circuit Programming) or IAP (In-Application Programming).

The array matrix organisation allows each sector to be erased and reprogrammed without affecting other sectors.

4.2 Main Features

- Three Flash programming modes:
  - Insertion in a programming tool. In this mode, all sectors including option bytes can be programmed or erased.
  - ICP (In-Circuit Programming). In this mode, all sectors including option bytes can be programmed or erased without removing the device from the application board.
  - IAP (In-Application Programming) In this mode, all sectors except Sector 0, can be programmed or erased without removing the device from the application board and while the application is running.
- ICT (In-Circuit Testing) for downloading and executing user application test patterns in RAM
- Read-out protection
- Register Access Security System (RASS) to prevent accidental programming or erasing

4.3 Structure

The Flash memory is organised in sectors and can be used for both code and data storage.

Depending on the overall Flash memory size in the microcontroller device, there are up to three user sectors (see Table 3). Each of these sectors can be erased independently to avoid unnecessary erasing of the whole Flash memory when only a partial erasing is required.

The first two sectors have a fixed size of 4 Kbytes (see Figure 6). They are mapped in the upper part of the ST7 addressing space so the reset and interrupt vectors are located in Sector 0 (F000h-FFFFh).

Table 3. Sectors available in Flash devices

Flash Size (bytes)	Available Sectors
4K	Sector 0
8K	Sectors 0,1
> 8K	Sectors 0,1, 2

4.3.1 Read-out Protection

Read-out protection, when selected, provides a protection against Program Memory content extraction and against write access to Flash memory. Even if no protection can be considered as totally unbreakable, the feature provides a very high level of protection for a general purpose microcontroller.

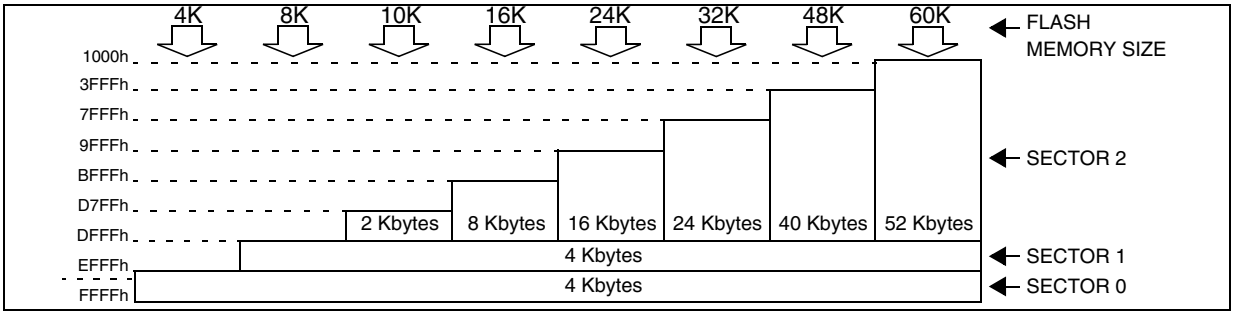
In flash devices, this protection is removed by reprogramming the option. In this case, the entire program memory is first automatically erased.

Read-out protection selection depends on the device type:

– In Flash devices it is enabled and removed through the FMP_R bit in the option byte.

– In ROM devices it is enabled by mask option specified in the Option List.

Figure 6. Memory Map and Sector Address



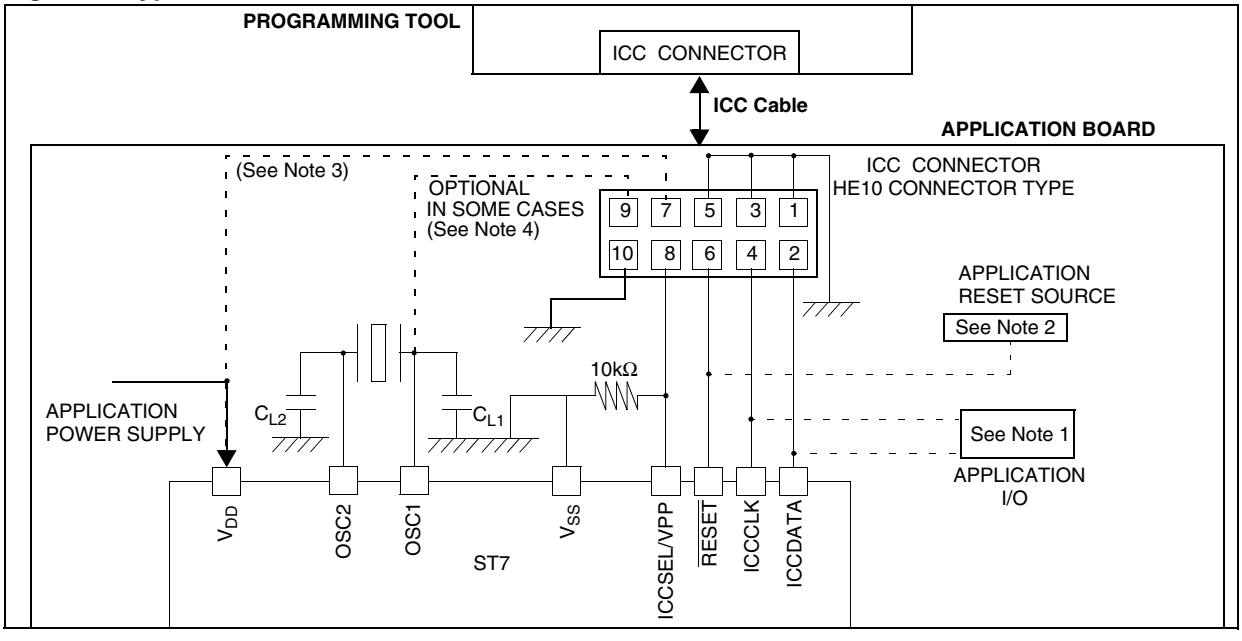
FLASH PROGRAM MEMORY (Cont'd)

4.4 ICC Interface

ICC needs a minimum of 4 and up to 6 pins to be connected to the programming tool (see Figure 7). These pins are:

- $\overline{\text{RESET}}$: device reset
- V_{SS}: device power supply ground
- ICCCLK: ICC output serial clock pin
- ICCDATA: ICC input/output serial data pin
- ICCSEL/V_{PP}: programming voltage
- OSC1(or OSCIN): main clock input for external source (optional)
- V_{DD}: application board power supply (optional, see Figure 7, Note 3)

Figure 7. Typical ICC Interface



Notes:

1. If the ICCCLK or ICCDATA pins are only used as outputs in the application, no signal isolation is necessary. As soon as the Programming Tool is plugged to the board, even if an ICC session is not in progress, the ICCCLK and ICCDATA pins are not available for the application. If they are used as inputs by the application, isolation such as a serial resistor has to implemented in case another device forces the signal. Refer to the Programming Tool documentation for recommended resistor values.

2. During the ICC session, the programming tool must control the $\overline{\text{RESET}}$ pin. This can lead to conflicts between the programming tool and the application reset circuit if it drives more than 5mA at high level (push pull output or pull-up resistor<1K). A schottky diode can be used to isolate the application RESET circuit in this case. When using a classical RC network with R>1K or a reset management IC with open drain output and pull-up resistor>1K, no additional components are needed. In all cases the user must ensure that no external reset is generated by the application during the ICC session.

3. The use of Pin 7 of the ICC connector depends on the Programming Tool architecture. This pin must be connected when using most ST Programming Tools (it is used to monitor the application power supply). Please refer to the Programming Tool manual.

4. Pin 9 has to be connected to the OSC1 or OSCIN pin of the ST7 when the clock is not available in the application or if the selected clock option is not programmed in the option byte. ST7 devices with multi-oscillator capability need to have OSC2 grounded in this case.

INTERRUPTS (Cont'd)

Table 8. Interrupt Mapping

N°	Source Block	Description	Register Label	Priority Order	Exit from HALT/ ACTIVE HALT[1)]	Address Vector
	RESET	Reset	N/A		yes	FFFEh-FFFFh
	TRAP	Software interrupt			no	FFFCh-FFFDh
0	Not used					FFFAh-FFFBh
1	MCC/RTC	Main clock controller time base interrupt	MCCSR	Higher Priority	yes	FFF8h-FFF9h
2	ei0	External interrupt port A3..0	N/A		yes	FFF6h-FFF7h
3	ei1	External interrupt port F2..0			yes	FFF4h-FFF5h
4	ei2	External interrupt port B3..0			yes	FFF2h-FFF3h
5	ei3	External interrupt port B7..4			yes	FFF0h-FFF1h
6	Not used					FFEEh-FFEFh
7	SPI	SPI peripheral interrupts	SPICSR		yes	FFECh-FFEDh
8	TIMER A	TIMER A peripheral interrupts	TASR		no	FFEAh-FFEBh
9	TIMER B	TIMER B peripheral interrupts	TBSR		no	FFE8h-FFE9h
10	SCI	SCI Peripheral interrupts	SCISR	Lower Priority	no	FFE6h-FFE7h
11	AVD	Auxiliary Voltage detector interrupt	SICSR		no	FFE4h-FFE5h

Notes:

1. In Flash devices only a RESET or MCC/RTC interrupt can be used to wake-up from Active Halt mode.

7.6 EXTERNAL INTERRUPTS

7.6.1 I/O Port Interrupt Sensitivity

The external interrupt sensitivity is controlled by the IPA, IPB and ISxx bits of the EICR register (Figure 21). This control allows to have up to 4 fully independent external interrupt source sensitivities.

Each external interrupt source can be generated on four (or five) different events on the pin:

- Falling edge
- Rising edge
- Falling and rising edge
- Falling edge and low level
- Rising edge and high level (only for ei0 and ei2)

To guarantee correct functionality, the sensitivity bits in the EICR register can be modified only when the I1 and I0 bits of the CC register are both set to 1 (level 3). This means that interrupts must be disabled before changing sensitivity.

The pending interrupts are cleared by writing a different value in the ISx[1:0], IPA or IPB bits of the EICR.

POWER SAVING MODES (Cont'd)

8.4 ACTIVE-HALT AND HALT MODES

ACTIVE-HALT and HALT modes are the two lowest power consumption modes of the MCU. They are both entered by executing the 'HALT' instruction. The decision to enter either in ACTIVE-HALT or HALT mode is given by the MCC/RTC interrupt enable flag (OIE bit in MCCSR register).

MCCSR OIE bit	Power Saving Mode entered when HALT instruction is executed
0	HALT mode
1	ACTIVE-HALT mode

8.4.1 ACTIVE-HALT MODE

ACTIVE-HALT mode is the lowest power consumption mode of the MCU with a real time clock available. It is entered by executing the 'HALT' instruction when the OIE bit of the Main Clock Controller Status register (MCCSR) is set (see Section 10.2 on page 56 for more details on the MCCSR register).

The MCU can exit ACTIVE-HALT mode on reception of either an MCC/RTC interrupt, a specific interrupt (see Table 8, "Interrupt Mapping," on page 36) or a RESET. When exiting ACTIVE-HALT mode by means of an interrupt, no 256 or 4096 CPU cycle delay occurs. The CPU resumes operation by servicing the interrupt or by fetching the reset vector which woke it up (see Figure 26).

When entering ACTIVE-HALT mode, the I[1:0] bits in the CC register are forced to '10b' to enable interrupts. Therefore, if an interrupt is pending, the MCU wakes up immediately.

In ACTIVE-HALT mode, only the main oscillator and its associated counter (MCC/RTC) are running to keep a wake-up time base. All other peripherals are not clocked except those which get their clock supply from another clock generator (such as external or auxiliary oscillator).

The safeguard against staying locked in ACTIVE-HALT mode is provided by the oscillator interrupt.

Note: As soon as the interrupt capability of one of the oscillators is selected (MCCSR.OIE bit set), entering ACTIVE-HALT mode while the Watchdog is active does not generate a RESET.
This means that the device cannot spend more than a defined delay in this power saving mode.

CAUTION: When exiting ACTIVE-HALT mode following an interrupt, OIE bit of MCCSR register must not be cleared before t_{DELAY} after the interrupt occurs (t_{DELAY} = 256 or 4096 t_{CPU} delay depending on option byte). Otherwise, the ST7 enters HALT mode for the remaining t_{DELAY} period.

Figure 25. ACTIVE-HALT Timing Overview

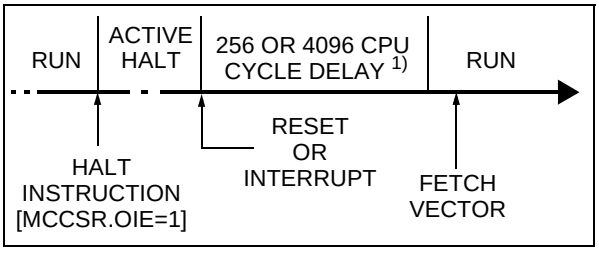
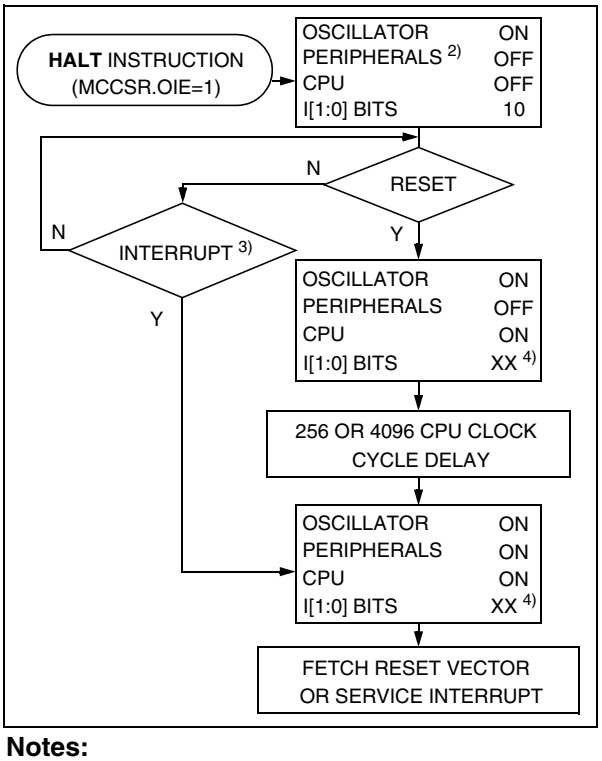


Figure 26. ACTIVE-HALT Mode Flow-chart



Notes:

1. This delay occurs only if the MCU exits ACTIVE-HALT mode by means of a RESET.
2. Peripheral clocked with an external clock source can still be active.
3. Only the MCC/RTC interrupt and some specific interrupts can exit the MCU from ACTIVE-HALT mode (such as external interrupt). Refer to Table 8, "Interrupt Mapping," on page 36 for more details.
4. Before servicing an interrupt, the CC register is pushed on the stack. The I[1:0] bits of the CC register are set to the current software priority level of the interrupt routine and restored when the CC register is popped.

16-BIT TIMER (Cont'd)

Figure 42. Output Compare Timing Diagram, $f_{TIMER} = f_{CPU}/2$

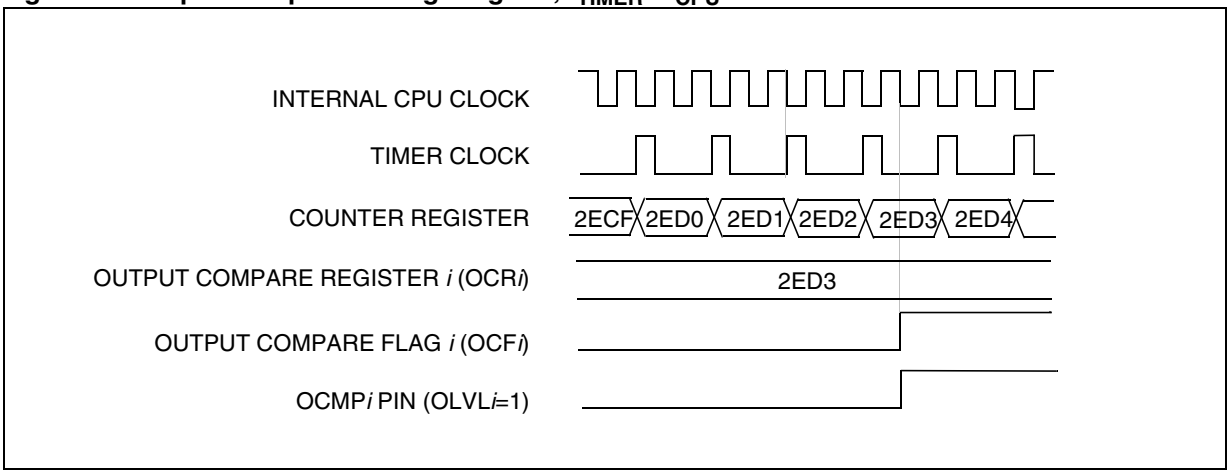
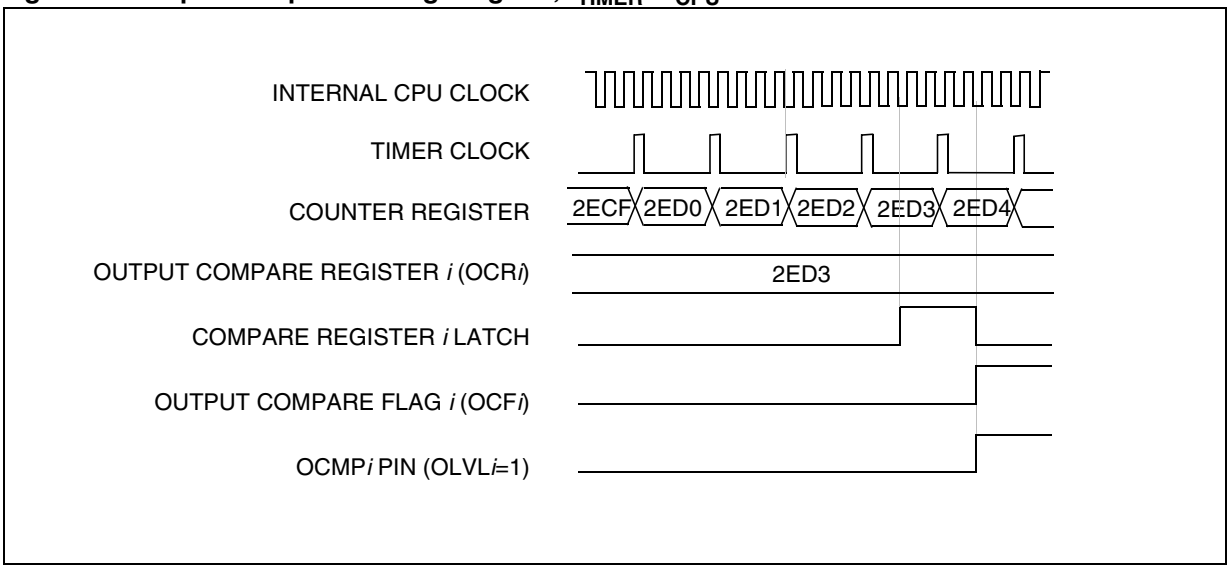


Figure 43. Output Compare Timing Diagram, $f_{TIMER} = f_{CPU}/4$



16-BIT TIMER (Cont'd)

10.3.3.6 Pulse Width Modulation Mode

Pulse Width Modulation (PWM) mode enables the generation of a signal with a frequency and pulse length determined by the value of the OC1R and OC2R registers.

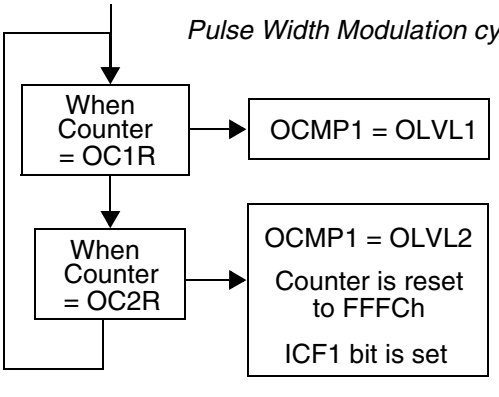
Pulse Width Modulation mode uses the complete Output Compare 1 function plus the OC2R register, and so this functionality can not be used when PWM mode is activated.

In PWM mode, double buffering is implemented on the output compare registers. Any new values written in the OC1R and OC2R registers are taken into account only at the end of the PWM period (OC2) to avoid spikes on the PWM output pin (OCMP1).

Procedure

To use pulse width modulation mode:

1. Load the OC2R register with the value corresponding to the period of the signal using the formula in the opposite column.
2. Load the OC1R register with the value corresponding to the period of the pulse if (OLVL1=0 and OLVL2=1) using the formula in the opposite column.
3. Select the following in the CR1 register:
   - Using the OLVL1 bit, select the level to be applied to the OCMP1 pin after a successful comparison with the OC1R register.
   - Using the OLVL2 bit, select the level to be applied to the OCMP1 pin after a successful comparison with the OC2R register.
4. Select the following in the CR2 register:
   - Set OC1E bit: the OCMP1 pin is then dedicated to the output compare 1 function.
   - Set the PWM bit.
   - Select the timer clock (CC[1:0]) (see Table 16 Clock Control Bits).



If OLVL1=1 and OLVL2=0 the length of the positive pulse is the difference between the OC2R and OC1R registers.

If OLVL1=OLVL2 a continuous signal will be seen on the OCMP1 pin.

The OC*i*R register value required for a specific timing application can be calculated using the following formula:

$$OCiR\ Value = \frac{t * f_{CPU}}{PRESC} - 5$$

Where:

t = Signal or pulse period (in seconds)

f_{CPU} = CPU clock frequency (in hertz)

PRESC = Timer prescaler factor (2, 4 or 8 depending on CC[1:0] bits, see Table 16)

If the timer clock is an external clock the formula is:

$$OCiR = t * f_{EXT} - 5$$

Where:

t = Signal or pulse period (in seconds)

f_{EXT} = External timer clock frequency (in hertz)

The Output Compare 2 event causes the counter to be initialized to FFFCh (See Figure 45)

Notes:

1. After a write instruction to the OC*i*HR register, the output compare function is inhibited until the OC*i*LR register is also written.
2. The OCF1 and OCF2 bits cannot be set by hardware in PWM mode therefore the Output Compare interrupt is inhibited.
3. The ICF1 bit is set by hardware when the counter reaches the OC2R value and can produce a timer interrupt if the ICIE bit is set and the I bit is cleared.
4. In PWM mode the ICAP1 pin can not be used to perform input capture because it is disconnected to the timer. The ICAP2 pin can be used to perform input capture (ICF2 can be set and IC2R can be loaded) but the user must take care that the counter is reset each period and ICF1 can also generates interrupt if ICIE is set.
5. When the Pulse Width Modulation (PWM) and One Pulse Mode (OPM) bits are both set, the PWM mode is the only active one.
6. In Flash devices, the TAOC2HR, TAOC2LR registers in Timer A are "write only". A read operation returns an undefined value.

7. In Flash devices, the ICAP2 registers (TAIC2HR, TAIC2LR) are not available in Timer A. The ICF2 bit is forced by hardware to 0.

16-BIT TIMER (Cont'd)

10.3.4 Low Power Modes

Mode	Description
WAIT	No effect on 16-bit Timer. Timer interrupts cause the device to exit from WAIT mode.
HALT	16-bit Timer registers are frozen. In HALT mode, the counter stops counting until Halt mode is exited. Counting resumes from the previous count when the MCU is woken up by an interrupt with "exit from HALT mode" capability or from the counter reset value when the MCU is woken up by a RESET. If an input capture event occurs on the ICAP*i* pin, the input capture detection circuitry is armed. Consequently, when the MCU is woken up by an interrupt with "exit from HALT mode" capability, the ICF*i* bit is set, and the counter value present when exiting from HALT mode is captured into the IC*i*R register.

10.3.5 Interrupts

Interrupt Event	Event Flag	Enable Control Bit	Exit from Wait	Exit from Halt
Input Capture 1 event/Counter reset in PWM mode	ICF1	ICIE	Yes	No
Input Capture 2 event	ICF2*		Yes	No
Output Compare 1 event (not available in PWM mode)	OCF1	OCIE	Yes	No
Output Compare 2 event (not available in PWM mode)	OCF2*		Yes	No
Timer Overflow event	TOF	TOIE	Yes	No

Note: The 16-bit Timer interrupt events are connected to the same interrupt vector (see Interrupts chapter). These events generate an interrupt if the corresponding Enable Control Bit is set and the interrupt mask in the CC register is reset (RIM instruction).

* In Flash devices, the ICF2 and OCF2 bits are forced by hardware to 0 in Timer A, hence there is no interrupt event for these flags.

10.3.6 Summary of Timer modes

MODES	TIMER RESOURCES			
	Input Capture 1	Input Capture 2	Output Compare 1	Output Compare 2
Input Capture (1 and/or 2)	Yes	Yes[2)5)]	Yes	Yes[4)]
Output Compare (1 and/or 2)	Yes	Yes[5)]	Yes	Yes[4)]
One Pulse Mode	No	Not Recommended[1)5)]	No	Partially [2)]
PWM Mode	No	Not Recommended[3)5)]	No	No

1) See note 4 in Section 10.3.3.5 One Pulse Mode

2) See note 5 and 6 in Section 10.3.3.5 One Pulse Mode

3) See note 4 in Section 10.3.3.6 Pulse Width Modulation Mode

4) In Flash devices, the TAOC2HR, TAOC2LR registers are write only in Timer A. Output Compare 2 event cannot be generated, OCF2 is forced by hardware to 0.

5) In Flash devices, Input Capture 2 is not implemented in Timer A. ICF2 bit is forced by hardware to 0.

SERIAL PERIPHERAL INTERFACE (Cont'd)

10.4.3.3 Master Mode Operation

In master mode, the serial clock is output on the SCK pin. The clock frequency, polarity and phase are configured by software (refer to the description of the SPICSR register).

Note: The idle state of SCK must correspond to the polarity selected in the SPICSR register (by pulling up SCK if CPOL=1 or pulling down SCK if CPOL=0).

To operate the SPI in master mode, perform the following steps in order (if the SPICSR register is not written first, the SPICR register setting (MSTR bit) may be not taken into account):

1. Write to the SPICR register:
   - Select the clock frequency by configuring the SPR[2:0] bits.
   - Select the clock polarity and clock phase by configuring the CPOL and CPHA bits. Figure 50 shows the four possible configurations.
     Note: The slave must have the same CPOL and CPHA settings as the master.
2. Write to the SPICSR register:
   - Either set the SSM bit and set the SSI bit or clear the SSM bit and tie the SS pin high for the complete byte transmit sequence.
3. Write to the SPICR register:
   - Set the MSTR and SPE bits
     Note: MSTR and SPE bits remain set only if SS is high).

The transmit sequence begins when software writes a byte in the SPIDR register.

10.4.3.4 Master Mode Transmit Sequence

When software writes to the SPIDR register, the data byte is loaded into the 8-bit shift register and then shifted out serially to the MOSI pin most significant bit first.

When data transfer is complete:

- The SPIF bit is set by hardware
- An interrupt request is generated if the SPIE bit is set and the interrupt mask in the CCR register is cleared.

Clearing the SPIF bit is performed by the following software sequence:

1. An access to the SPICSR register while the SPIF bit is set
2. A read to the SPIDR register.

Note: While the SPIF bit is set, all writes to the SPIDR register are inhibited until the SPICSR register is read.

10.4.3.5 Slave Mode Operation

In slave mode, the serial clock is received on the SCK pin from the master device.

To operate the SPI in slave mode:

1. Write to the SPICSR register to perform the following actions:
   - Select the clock polarity and clock phase by configuring the CPOL and CPHA bits (see Figure 50).
     Note: The slave must have the same CPOL and CPHA settings as the master.
   - Manage the $\overline{SS}$ pin as described in Section 10.4.3.2 and Figure 48. If CPHA=1 $\overline{SS}$ must be held low continuously. If CPHA=0 $\overline{SS}$ must be held low during byte transmission and pulled up between each byte to let the slave write in the shift register.
2. Write to the SPICR register to clear the MSTR bit and set the SPE bit to enable the SPI I/O functions.

10.4.3.6 Slave Mode Transmit Sequence

When software writes to the SPIDR register, the data byte is loaded into the 8-bit shift register and then shifted out serially to the MISO pin most significant bit first.

The transmit sequence begins when the slave device receives the clock signal and the most significant bit of the data on its MOSI pin.

When data transfer is complete:

- The SPIF bit is set by hardware
- An interrupt request is generated if SPIE bit is set and interrupt mask in the CCR register is cleared.

Clearing the SPIF bit is performed by the following software sequence:

1. An access to the SPICSR register while the SPIF bit is set.
2. A write or a read to the SPIDR register.

Notes: While the SPIF bit is set, all writes to the SPIDR register are inhibited until the SPICSR register is read.

The SPIF bit can be cleared during a second transmission; however, it must be cleared before the second SPIF bit in order to prevent an Overrun condition (see Section 10.4.5.2).

SERIAL PERIPHERAL INTERFACE (Cont'd)

CONTROL/STATUS REGISTER (SPICSR)
Read/Write (some bits Read Only)
Reset Value: 0000 0000 (00h)

7							0
SPIF	WCOL	OVR	MODF	-	SOD	SSM	SSI

Bit 7 = **SPIF** *Serial Peripheral Data Transfer Flag (Read only).*
This bit is set by hardware when a transfer has been completed. An interrupt is generated if SPIE=1 in the SPICR register. It is cleared by a software sequence (an access to the SPICSR register followed by a write or a read to the SPIDR register).
0: Data transfer is in progress or the flag has been cleared.
1: Data transfer between the device and an external device has been completed.

Note: While the SPIF bit is set, all writes to the SPIDR register are inhibited until the SPICSR register is read.

Bit 6 = **WCOL** *Write Collision status (Read only).*
This bit is set by hardware when a write to the SPIDR register is done during a transmit sequence. It is cleared by a software sequence (see Figure 51).
0: No write collision occurred
1: A write collision has been detected

Bit 5 = **OVR** S*PI Overrun error (Read only).*
This bit is set by hardware when the byte currently being received in the shift register is ready to be transferred into the SPIDR register while SPIF = 1 (See Section 10.4.5.2). An interrupt is generated if SPIE = 1 in SPICR register. The OVR bit is cleared by software reading the SPICSR register.
0: No overrun error
1: Overrun error detected

Bit 4 = **MODF** *Mode Fault flag (Read only).*
This bit is set by hardware when the $\overline{SS}$ pin is pulled low in master mode (see Section 10.4.5.1 Master Mode Fault (MODF)). An SPI interrupt can be generated if SPIE=1 in the SPICSR register. This bit is cleared by a software sequence (An access to the SPICR register while MODF=1 followed by a write to the SPICR register).
0: No master mode fault detected
1: A fault in master mode has been detected

Bit 3 = Reserved, must be kept cleared.

Bit 2 = **SOD** *SPI Output Disable.*
This bit is set and cleared by software. When set, it disables the alternate function of the SPI output (MOSI in master mode / MISO in slave mode)
0: SPI output enabled (if SPE=1)
1: SPI output disabled

Bit 1 = **SSM** $\overline{SS}$ *Management.*
This bit is set and cleared by software. When set, it disables the alternate function of the SPI $\overline{SS}$ pin and uses the SSI bit value instead. See Section 10.4.3.2 Slave Select Management.
0: Hardware management ($\overline{SS}$ managed by external pin)
1: Software management (internal $\overline{SS}$ signal controlled by SSI bit. External $\overline{SS}$ pin free for general-purpose I/O)

Bit 0 = **SSI** $\overline{SS}$ *Internal Mode.*
This bit is set and cleared by software. It acts as a 'chip select' by controlling the level of the $\overline{SS}$ slave select signal when the SSM bit is set.
0: Slave selected
1: Slave deselected

DATA I/O REGISTER (SPIDR)
Read/Write
Reset Value: Undefined

7							0
D7	D6	D5	D4	D3	D2	D1	D0

The SPIDR register is used to transmit and receive data on the serial bus. In a master device, a write to this register will initiate transmission/reception of another byte.

Notes: During the last clock cycle the SPIF bit is set, a copy of the received data byte in the shift register is moved to a buffer. When the user reads the serial peripheral data I/O register, the buffer is actually being read.

While the SPIF bit is set, all writes to the SPIDR register are inhibited until the SPICSR register is read.

Warning: A write to the SPIDR register places data directly into the shift register for transmission.

A read to the SPIDR register returns the value located in the buffer and not the content of the shift register (see Figure 46).

SERIAL PERIPHERAL INTERFACE (Cont'd)

Table 19. SPI Register Map and Reset Values

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
0021h	**SPIDR**	MSB							LSB
	Reset Value	x	x	x	x	x	x	x	x
0022h	**SPICR**	SPIE	SPE	SPR2	MSTR	CPOL	CPHA	SPR1	SPR0
	Reset Value	0	0	0	0	x	x	x	x
0023h	**SPICSR**	SPIF	WCOL	OR	MODF		SOD	SSM	SSI
	Reset Value	0	0	0	0	0	0	0	0

SERIAL COMMUNICATIONS INTERFACE (Cont'd)

Framing Error

A framing error is detected when:

– The stop bit is not recognized on reception at the expected time, following either a de-synchronization or excessive noise.

– A break is received.

When the framing error is detected:

– the FE bit is set by hardware

– Data is transferred from the Shift register to the SCIDR register.

– No interrupt is generated. However this bit rises at the same time as the RDRF bit which itself generates an interrupt.

The FE bit is reset by a SCISR register read operation followed by a SCIDR register read operation.

10.5.4.4 Conventional Baud Rate Generation

The baud rate for the receiver and transmitter (Rx and Tx) are set independently and calculated as follows:

$$Tx = \frac{f_{CPU}}{(16*PR)*TR} \qquad Rx = \frac{f_{CPU}}{(16*PR)*RR}$$

with:

PR = 1, 3, 4 or 13 (see SCP[1:0] bits)

TR = 1, 2, 4, 8, 16, 32, 64,128

(see SCT[2:0] bits)

RR = 1, 2, 4, 8, 16, 32, 64,128

(see SCR[2:0] bits)

All these bits are in the SCIBRR register.

Example: If f_{CPU} is 8 MHz (normal mode) and if PR = 13 and TR = RR = 1, the transmit and receive baud rates are 38400 baud.

Note: The baud rate registers MUST NOT be changed while the transmitter or the receiver is enabled.

10.5.4.5 Extended Baud Rate Generation

The extended prescaler option gives a very fine tuning on the baud rate, using a 255 value prescaler, whereas the conventional Baud Rate Generator retains industry standard software compatibility.

The extended baud rate generator block diagram is described in the Figure 3.

The output clock rate sent to the transmitter or to the receiver is the output from the 16 divider divided by a factor ranging from 1 to 255 set in the SCIERPR or the SCIETPR register.

Note: the extended prescaler is activated by setting the SCIETPR or SCIERPR register to a value other than zero. The baud rates are calculated as follows:

$$Tx = \frac{f_{CPU}}{16*ETPR*(PR*TR)} \qquad Rx = \frac{f_{CPU}}{16*ERPR*(PR*RR)}$$

with:

ETPR = 1,..,255 (see SCIETPR register)

ERPR = 1,.. 255 (see SCIERPR register)

10.5.4.6 Receiver Muting and Wake-up Feature

In multiprocessor configurations it is often desirable that only the intended message recipient should actively receive the full message contents, thus reducing redundant SCI service overhead for all non addressed receivers.

The non addressed devices may be placed in sleep mode by means of the muting function.

Setting the RWU bit by software puts the SCI in sleep mode:

All the reception status bits can not be set.

All the receive interrupts are inhibited.

A muted receiver may be awakened by one of the following two ways:

– by Idle Line detection if the WAKE bit is reset,

– by Address Mark detection if the WAKE bit is set.

Receiver wakes-up by Idle Line detection when the Receive line has recognized an Idle Frame. Then the RWU bit is reset by hardware but the IDLE bit is not set.

Receiver wakes-up by Address Mark detection when it received a "1" as the most significant bit of a word, thus indicating that the message is an address. The reception of this particular word wakes up the receiver, resets the RWU bit and sets the RDRF bit, which allows the receiver to receive this word normally and to use it as an address word.

CAUTION: In Mute mode, do not write to the SCICR2 register. If the SCI is in Mute mode during the read operation (RWU = 1) and a address mark wake up event occurs (RWU is reset) before the write operation, the RWU bit is set again by this write operation. Consequently the address byte is lost and the SCI is not woken up from Mute mode.

INSTRUCTION SET OVERVIEW (Cont'd)

Mnemo	Description	Function/Example	Dst	Src	I1	H	I0	N	Z	C
ADC	Add with Carry	A = A + M + C	A	M		H		N	Z	C
ADD	Addition	A = A + M	A	M		H		N	Z	C
AND	Logical And	A = A . M	A	M				N	Z	
BCP	Bit compare A, Memory	tst (A . M)	A	M				N	Z	
BRES	Bit Reset	bres Byte, #3	M							
BSET	Bit Set	bset Byte, #3	M							
BTJF	Jump if bit is false (0)	btjf Byte, #3, Jmp1	M							C
BTJT	Jump if bit is true (1)	btjt Byte, #3, Jmp1	M							C
CALL	Call subroutine									
CALLR	Call subroutine relative									
CLR	Clear		reg, M					0	1	
CP	Arithmetic Compare	tst(Reg - M)	reg	M				N	Z	C
CPL	One Complement	A = FFH-A	reg, M					N	Z	1
DEC	Decrement	dec Y	reg, M					N	Z	
HALT	Halt				1		0			
IRET	Interrupt routine return	Pop CC, A, X, PC			I1	H	I0	N	Z	C
INC	Increment	inc X	reg, M					N	Z	
JP	Absolute Jump	jp [TBL.w]								
JRA	Jump relative always									
JRT	Jump relative									
JRF	Never jump	jrf *								
JRIH	Jump if ext. INT pin = 1	(ext. INT pin high)								
JRIL	Jump if ext. INT pin = 0	(ext. INT pin low)								
JRH	Jump if H = 1	H = 1?								
JRNH	Jump if H = 0	H = 0?								
JRM	Jump if I1:0 = 11	I1:0 = 11?								
JRNM	Jump if I1:0 <> 11	I1:0 <> 11?								
JRMI	Jump if N = 1 (minus)	N = 1?								
JRPL	Jump if N = 0 (plus)	N = 0?								
JREQ	Jump if Z = 1 (equal)	Z = 1?								
JRNE	Jump if Z = 0 (not equal)	Z = 0?								
JRC	Jump if C = 1	C = 1?								
JRNC	Jump if C = 0	C = 0?								
JRULT	Jump if C = 1	Unsigned <								
JRUGE	Jump if C = 0	Jmp if unsigned >=								
JRUGT	Jump if (C + Z = 0)	Unsigned >								

INSTRUCTION SET OVERVIEW (Cont'd)

Mnemo	Description	Function/Example	Dst	Src	I1	H	I0	N	Z	C
JRULE	Jump if (C + Z = 1)	Unsigned <=								
LD	Load	dst <= src	reg, M	M, reg				N	Z	
MUL	Multiply	X,A = X * A	A, X, Y	X, Y, A		0				0
NEG	Negate (2's compl)	neg $10	reg, M					N	Z	C
NOP	No Operation									
OR	OR operation	A = A + M	A	M				N	Z	
POP	Pop from the Stack	pop reg	reg	M						
		pop CC	CC	M	I1	H	I0	N	Z	C
PUSH	Push onto the Stack	push Y	M	reg, CC						
RCF	Reset carry flag	C = 0								0
RET	Subroutine Return									
RIM	Enable Interrupts	I1:0 = 10 (level 0)			1		0			
RLC	Rotate left true C	C <= A <= C	reg, M					N	Z	C
RRC	Rotate right true C	C => A => C	reg, M					N	Z	C
RSP	Reset Stack Pointer	S = Max allowed								
SBC	Substract with Carry	A = A - M - C	A	M				N	Z	C
SCF	Set carry flag	C = 1								1
SIM	Disable Interrupts	I1:0 = 11 (level 3)			1		1			
SLA	Shift left Arithmetic	C <= A <= 0	reg, M					N	Z	C
SLL	Shift left Logic	C <= A <= 0	reg, M					N	Z	C
SRL	Shift right Logic	0 => A => C	reg, M					0	Z	C
SRA	Shift right Arithmetic	A7 => A => C	reg, M					N	Z	C
SUB	Substraction	A = A - M	A	M				N	Z	C
SWAP	SWAP nibbles	A7-A4 <=> A3-A0	reg, M					N	Z	
TNZ	Test for Neg & Zero	tnz lbl1						N	Z	
TRAP	S/W trap	S/W interrupt			1		1			
WFI	Wait for Interrupt				1		0			
XOR	Exclusive OR	A = A XOR M	A	M				N	Z	

12.12 COMMUNICATION INTERFACE CHARACTERISTICS

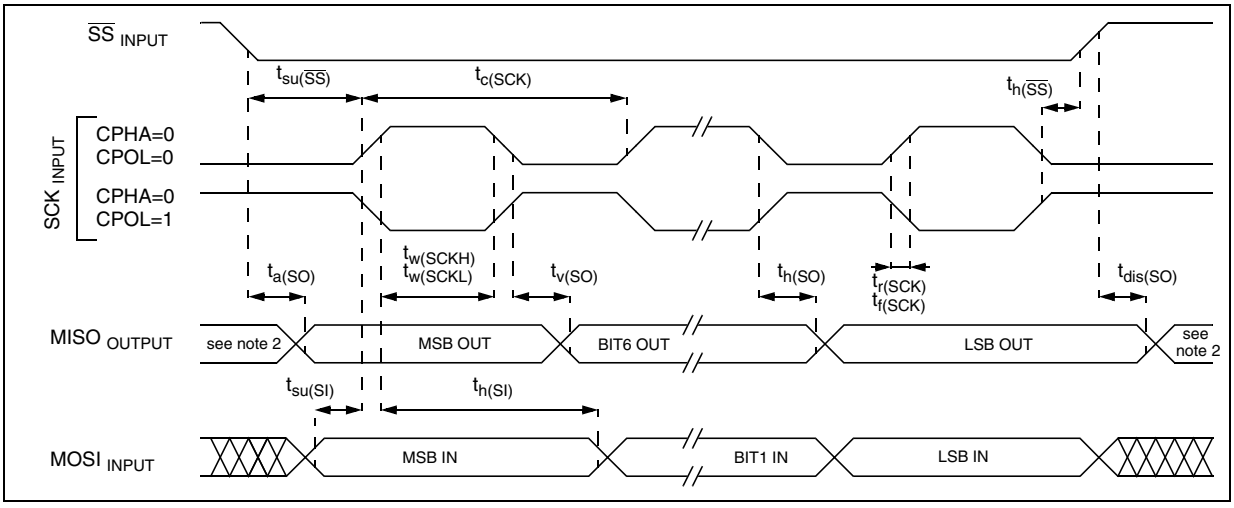
12.12.1 SPI - Serial Peripheral Interface

Subject to general operating conditions for V_{DD}, f_{CPU}, and T_A unless otherwise specified. Data based on design simulation and/or characterisation results, not tested in production.

When no communication is on-going the data output line of the SPI (MOSI in master mode, MISO in slave mode) has its alternate function capability released. In this case, the pin status depends on the I/O port configuration. Refer to I/O port characteristics for more details on the input/output alternate function characteristics ($\overline{SS}$, SCK, MOSI, MISO).

Symbol	Parameter	Conditions	Min	Max	Unit
f_{SCK} $1/t_{c(SCK)}$	SPI clock frequency	Master f_{CPU}=8MHz	$f_{CPU}/128$ 0.0625	$f_{CPU}/4$ 2	MHz
		Slave f_{CPU}=8MHz	0	$f_{CPU}/2$ 4	
$t_{r(SCK)}$ $t_{f(SCK)}$	SPI clock rise and fall time		see I/O port pin description		
$t_{su(\overline{SS})}$	$\overline{SS}$ setup time	Slave	120		ns
$t_{h(\overline{SS})}$	$\overline{SS}$ hold time	Slave	120		
$t_{w(SCKH)}$ $t_{w(SCKL)}$	SCK high and low time	Master Slave	100 90		
$t_{su(MI)}$ $t_{su(SI)}$	Data input setup time	Master Slave	100 100		
$t_{h(MI)}$ $t_{h(SI)}$	Data input hold time	Master Slave	100 100		
$t_{a(SO)}$	Data output access time	Slave	0	120	
$t_{dis(SO)}$	Data output disable time	Slave		240	
$t_{v(SO)}$	Data output valid time	Slave (after enable edge)		90	
$t_{h(SO)}$	Data output hold time		0		
$t_{v(MO)}$	Data output valid time	Master (before capture edge)	0.25		t_{CPU}
$t_{h(MO)}$	Data output hold time		0.25		

Figure 80. SPI Slave Timing Diagram with CPHA=0 [1)]



Notes:

1. Measurement points are done at CMOS levels: $0.3xV_{DD}$ and $0.7xV_{DD}$.

12.13 10-BIT ADC CHARACTERISTICS

Subject to general operating conditions for V_{DD}, f_{CPU}, and T_A unless otherwise specified.

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{ADC}	ADC clock frequency		0.4		2	MHz
V_{AREF}	Analog reference voltage	$0.7*V_{DD} \leq V_{AREF} \leq V_{DD}$	3.8		V_{DD}	V
V_{AIN}	Conversion voltage range [1]		V_{SSA}		V_{AREF}	
I_{lkg}	Positive input leakage current for analog input[2]	$-40°C \leq T_A \leq +85°C$			±250	nA
		$+85°C \leq T_A \leq +125°C$			±1	µA
R_{AIN}	External input impedance				see Figure 83 and Figure 84[2][3][4]	kΩ
C_{AIN}	External capacitor on analog input					pF
f_{AIN}	Variation freq. of analog input signal					Hz
C_{ADC}	Internal sample and hold capacitor			12		pF
t_{ADC}	Conversion time (Sample+Hold) f_{CPU}=8MHz, SPEED=0 f_{ADC}=2MHz			7.5		µs
t_{ADC}	- No of sample capacitor loading cycles - No. of Hold conversion cycles			4 11		$1/f_{ADC}$

Notes:

1. Any added external serial resistor will downgrade the ADC accuracy (especially for resistance greater than 10kΩ). Data based on characterization results, not tested in production.

2.For Flash devices: injecting negative current on any of the analog input pins significantly reduces the accuracy of any conversion being performed on any analog input. Analog pins of ST72F324 devices can be protected against negative injection by adding a Schottky diode (pin to ground). Injecting negative current on digital input pins degrades ADC accuracy especially if performed on a pin close to the analog input pins. Any positive injection current within the limits specified for $I_{INJ(PIN)}$ and $\Sigma I_{INJ(PIN)}$ in Section 12.9 does not affect the ADC accuracy.

14.4.1 Socket and Emulator Adapter Information

For information on the type of socket that is supplied with the emulator, refer to the suggested list of sockets in Table 29.

Note: Before designing the board layout, it is recommended to check the overall dimensions of the socket as they may be greater than the dimensions of the device.

For footprint and other mechanical information about these sockets and adapters, refer to the manufacturer's datasheet (www.yamaichi.de for TQFP44 10 x 10 and www.ironwoodelectronics.com for TQFP32 7 x 7).

Table 29. Suggested List of Socket Types

Device	Socket (supplied with ST7MDT20J-EMU3)	Emulator Adapter (supplied with ST7MDT20J-EMU3)
TQFP32 7 X 7	IRONWOOD SF-QFE32SA-L-01	IRONWOOD SK-UGA06/32A-01
TQFP44 10 X10	YAMAICHI IC149-044-*52-*5	YAMAICHI ICP-044-5

KNOWN LIMITATIONS (Cont'd)

To avoid this, a semaphore is set to '1' before checking the level change. The semaphore is changed to level '0' inside the interrupt routine. When a level change is detected, the semaphore status is checked and if it is '1' this means that the last interrupt has been missed. In this case, the interrupt routine is invoked with the call instruction.

There is another possible case, that is, if writing to PxOR or PxDDR is done with global interrupts disabled (interrupt mask bit set). In this case, the semaphore is changed to '1' when the level change is detected. Detecting a missed interrupt is done after the global interrupts are enabled (interrupt mask bit reset) and by checking the status of the semaphore. If it is '1' this means that the last interrupt was missed and the interrupt routine is invoked with the call instruction.

To implement the workaround, the following software sequence is to be followed for writing into the PxOR/PxDDR registers. The example is for Port PF1 with falling edge interrupt sensitivity. The software sequence is given for both cases (global interrupt disabled/enabled).

Case 1: Writing to PxOR or PxDDR with Global Interrupts Enabled:

```
LD A,#01
LD sema,A      ; set the semaphore to '1'
LD A,PFDR
AND A,#02
LD X,A         ; store the level before writing to PxOR/PxDDR
LD A,#$90
LD PFDDR,A     ; Write to PFDDR
LD A,#$ff
LD  PFOR,A     ; Write to PFOR
LD A,PFDR
AND A,#02
LD Y,A         ; store the level after writing to PxOR/PxDDR
LD A,X         ; check for falling edge
cp A,#02
jrne OUT
TNZ Y
jrne OUT
LD A,sema      ; check the semaphore status if edge is detected
CP A,#01
jrne OUT
call call_routine; call the interrupt routine
OUT:LD A,#00
LD sema,A
.call_routine  ; entry to call_routine
PUSH A
PUSH X
PUSH CC
.ext1_rt       ; entry to interrupt routine
LD A,#00
LD sema,A
IRET
```

Case 2: Writing to PxOR or PxDDR with Global Interrupts Disabled:

```
SIM            ; set the interrupt mask
LD A,PFDR
AND A,#$02
LD X,A         ; store the level before writing to PxOR/PxDDR
LD A,#$90
LD PFDDR,A     ; Write into PFDDR
LD A,#$ff
LD PFOR,A      ; Write to PFOR
LD A,PFDR
AND A,#$02
LD Y,A         ; store the level after writing to PxOR/PxDDR
LD A,X         ; check for falling edge
cp A,#$02
jrne OUT
TNZ Y
jrne OUT
LD A,#$01
LD sema,A      ; set the semaphore to '1' if edge is detected
RIM            ; reset the interrupt mask
LD A,sema      ; check the semaphore status
CP A,#$01
jrne OUT
call call_routine; call the interrupt routine
RIM
OUT:           RIM
```