Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

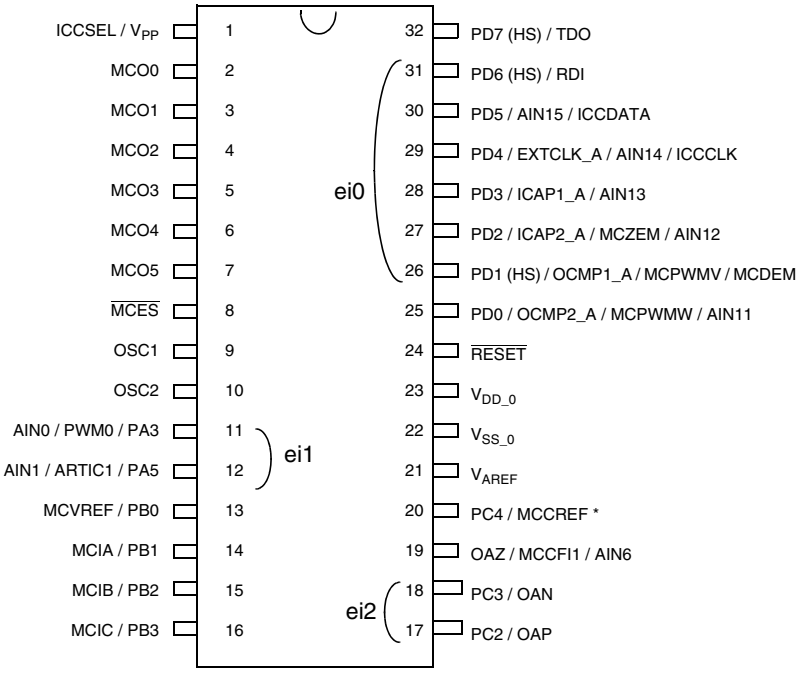
"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" 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	Not For New Designs
Core Processor	ST7
Core Size	8-Bit
Speed	8MHz
Connectivity	LINbusSCI, SPI
Peripherals	LVD, Motor Control PWM, POR, PWM, WDT
Number of I/O	26
Program Memory Size	16KB (16K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	768 x 8
Voltage - Supply (Vcc/Vdd)	3.8V ~ 5.5V
Data Converters	A/D 11x10b
Oscillator Type	External
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	44-LQFP
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/stmicroelectronics/st7fmc2s4t3

PIN DESCRIPTION (Cont'd)

Figure 4. 32-Pin SDIP Package Pinouts



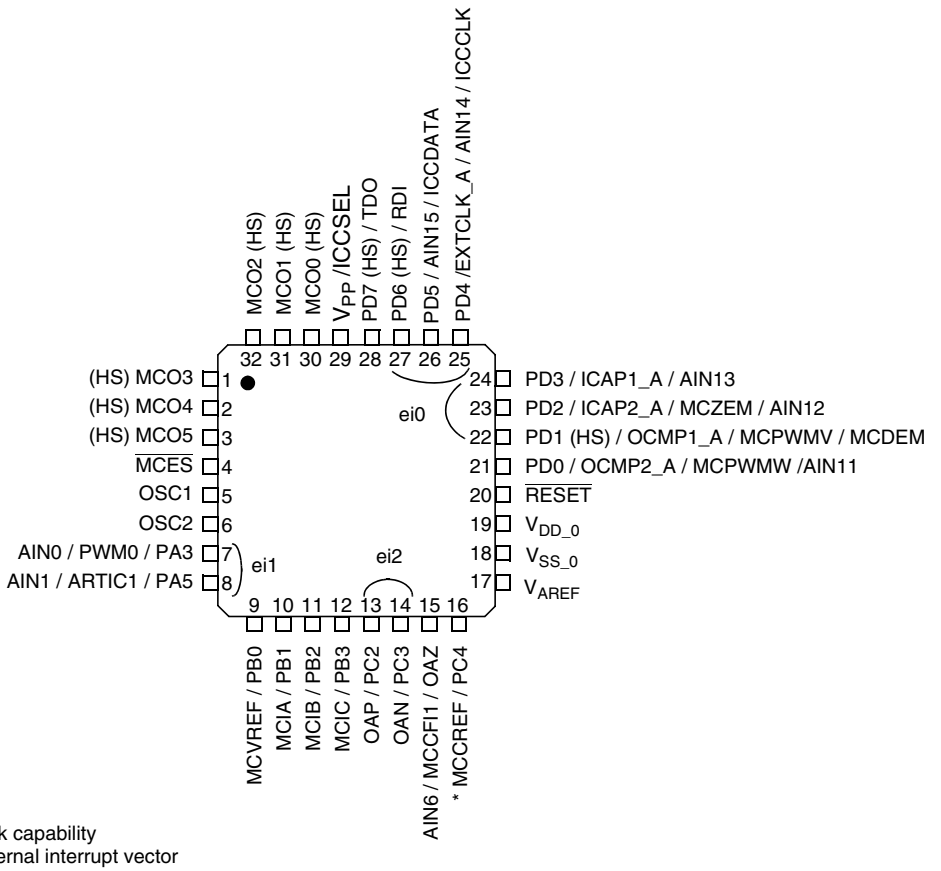
(HS) 20mA high sink capability

eix associated external interrupt vector

* Once the MTC peripheral is ON, the pin PC4 is configured to an alternate function. PC4 is no longer usable as a digital I/O

PIN DESCRIPTION (Cont'd)

Figure 7. 32-Pin LQFP 7x7 Package Pinout



(HS) 20mA high sink capability
eix associated external interrupt vector

* Once the MTC peripheral is ON, the pin PC4 is configured to an alternate function. PC4 is no longer usable as a digital I/O

SYSTEM INTEGRITY MANAGEMENT (Cont'd)

6.3.3 Clock Security System (CSS)

The Clock Security System (CSS) protects the ST7 against main clock problems. To allow the integration of the security features in the applications, it is based on a PLL which can provide a backup clock. The PLL can be enabled or disabled by option byte or by software. It requires an 8-MHz input clock and provides a 16-MHz output clock.

6.3.3.1 Safe Oscillator Control

The safe oscillator of the CSS block is made of a PLL.

If the clock signal disappears (due to a broken or disconnected resonator...) the PLL continues to provide a lower frequency, which allows the ST7 to perform some rescue operations.

Note: The clock signal must be present at start-up. Otherwise, the ST7MC will not start and will be maintained in RESET conditions.

6.3.3.2 Limitation detection

The automatic safe oscillator selection is notified by hardware setting the CSSD bit of the SICSR register. An interrupt can be generated if the CSSIE bit has been previously set.
These two bits are described in the SICSR register description.

6.3.4 Low Power Modes

Mode	Description
Wait	No effect on SI. CSS and AVD interrupts cause the device to exit from Wait mode.
Halt	The CRSR register is frozen. The CSS (including the safe oscillator) is disabled until Halt mode is exited. The previous CSS configuration resumes 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. The AVD remains active, and an AVD interrupt can be used to exit from Halt mode.

6.3.4.1 Interrupts

The CSS or AVD interrupt events generate an interrupt if the corresponding Enable Control Bit (CSSIE or AVDIE) is set and the interrupt mask in the CC register is reset (RIM instruction).

Interrupt Event	Event Flag	Enable Control Bit	Exit from Wait	Exit from Halt
CSS event detection (safe oscillator activated as main clock)	CSSD	CSSIE	Yes	No [1)]
AVD event	AVDF	AVDIE	Yes	Yes

Note 1: This interrupt allows to exit from Active-halt mode.

SYSTEM INTEGRITY MANAGEMENT (Cont'd)

SYSTEM INTEGRITY (SI) CONTROL/STATUS REGISTER (SICSR, page 1)

Reset Value: 00000000 (00h)

7							0
PAGE	0	VCOEN	LOCK	PLLEN	0	CKSEL	0

Bit 7 = **PAGE** *SICSR Register Page Selection*
This bit selects the SICSR register page. It is set and cleared by software
0: Access to SICSR register mapped in page 0.
1: Access to SICSR register mapped in page 1.

Bit 6 = Reserved, must be kept cleared.

Bit 5 = **VCOEN** *VCO Enable*
This bit is set and cleared by software.
0: VCO (Voltage Controlled Oscillator) connected to the output of the PLL charge pump (default mode), to obtain a 16-MHz output frequency (with an 8-MHz input frequency).
1: VCO tied to ground in order to obtain a 10-MHz frequency (f_{vco})

Notes:
1. During ICC session, this bit is set to 1 in order to have an internal frequency which does not depend on the input clock. Then, it can be reset in order to run faster with an external oscillator.

Bit 4 = **LOCK** *PLL Locked*
This bit is read only. It is set by hardware. It is set automatically when the PLL reaches its operating frequency.
0: PLL not locked
1: PLL locked

Bit 3 = **PLLEN** *PLL Enable*
This bit enables the PLL and the clock detector. It is set and cleared by software.
0: PLL and Clock Detector (CKD) disabled
1: PLL and Clock Detector (CKD) enabled

Notes:
1. During ICC session, this bit is set to 1.
2. PLL cannot be disabled if PLL clock source is selected (CKSEL= 1).

Bit 2 = Reserved, must be kept cleared.

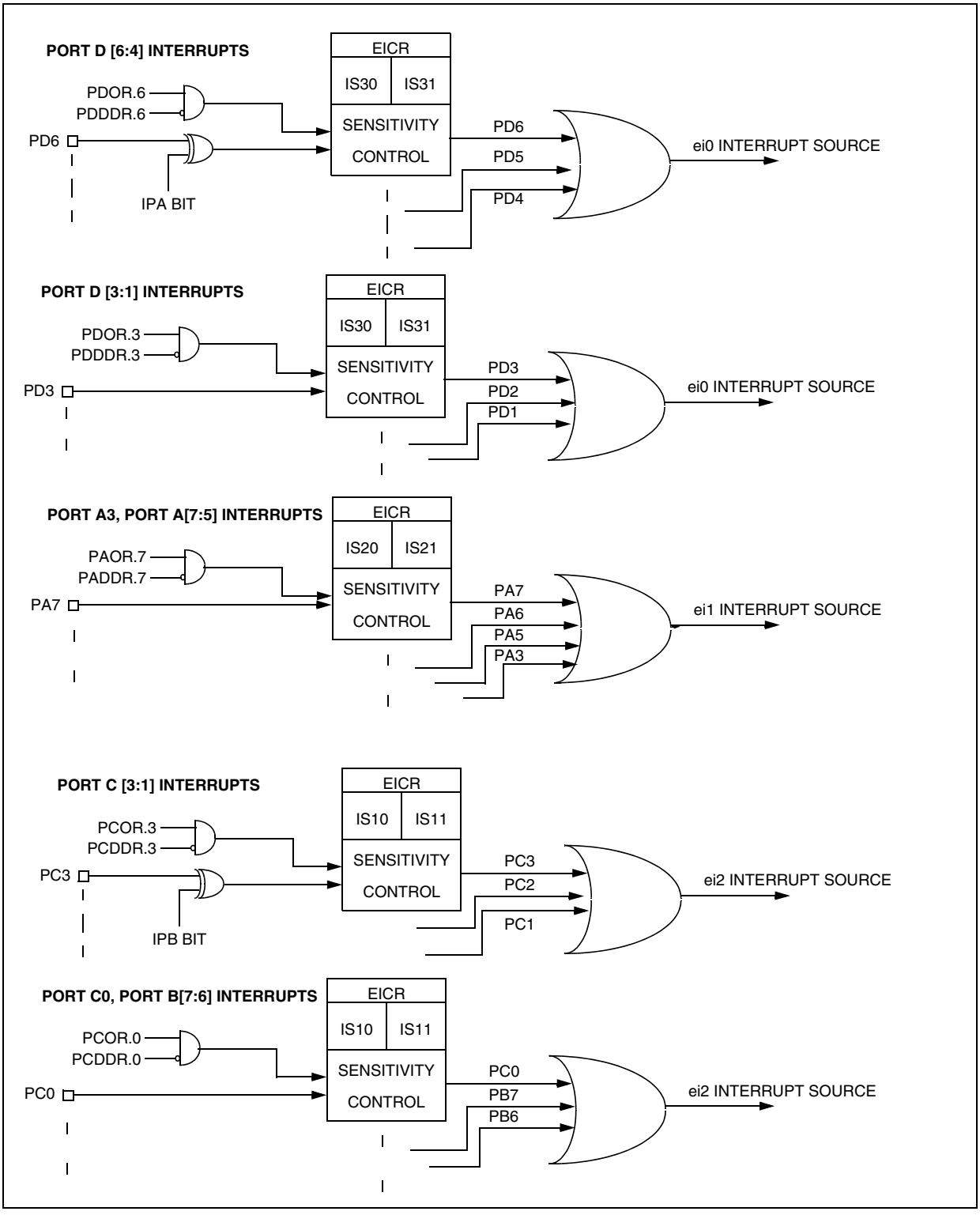
Bit 1 = **CKSEL** *Clock Source Selection*
This bit selects the clock source: oscillator clock or clock from the PLL. It is set and cleared by software. It can also be set by option byte (PLL opt)
0: Oscillator clock selected
1: PLL clock selected

Notes:
1. During ICC session, this bit is set to 1. Then, CKSEL can be reset in order to run with f_{OSC}.
2. Clock from the PLL cannot be selected if the PLL is disabled (PLLEN =0)
3. If the clock source is selected by PLL option bit, CKSEL bit selection has no effect.

Bit 0 = Reserved, must be kept cleared.

INTERRUPTS (Cont'd)

Figure 24. External Interrupt Control bits



INTERRUPTS (Cont'd)

7.7 EXTERNAL INTERRUPT CONTROL REGISTER (EICR)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
IS11	IS10	IPB	IS21	IS20	IS31	IS30	IPA

Bit 7:6 = **IS1[1:0]** *ei2 sensitivity*
The interrupt sensitivity, defined using the IS1[1:0] bits, is applied to the following external interrupts:
- ei2 (port C3..1)

IS11	IS10	External Interrupt Sensitivity	
		IPB bit =0	IPB bit =1
0	0	Falling edge & low level	Rising edge & high level
0	1	Rising edge only	Falling edge only
1	0	Falling edge only	Rising edge only
1	1	Rising and falling edge	

- ei2 (port C0, B7..6)

IS11	IS10	External Interrupt Sensitivity
0	0	Falling edge & low level
0	1	Rising edge only
1	0	Falling edge only
1	1	Rising and falling edge

These 2 bits can be written only when I1 and I0 of the CC register are both set to 1 (level 3).

Bit 5 = **IPB** *Interrupt polarity for port C*

This bit is used to invert the sensitivity of the port C[3:1] external interrupts. It can be set and cleared by software only when I1 and I0 of the CC register are both set to 1 (level 3).
0: No sensitivity inversion
1: Sensitivity inversion

Bit 4:3= **IS2[1:0]** *ei1sensitivity*
The interrupt sensitivity, defined using the IS2[1:0] bits, is applied to the following external interrupts:
- ei1 (port A3, A5...A7)

IS21	IS20	External Interrupt Sensitivity
0	0	Falling edge & low level
0	1	Rising edge only
1	0	Falling edge only
1	1	Rising and falling edge

Bit 2:1= **IS3[1:0]** *ei0sensitivity*
The interrupt sensitivity, defined using the IS2[1:0] bits, is applied to the following external interrupts:

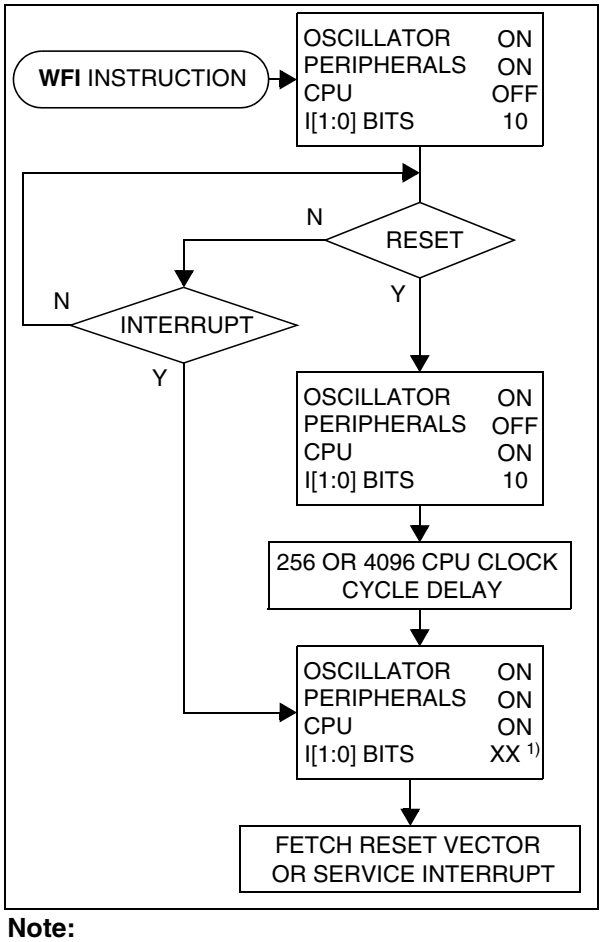
POWER SAVING MODES (Cont'd)

8.3 WAIT MODE

Wait mode places the MCU in a low power consumption mode by stopping the CPU.
This power saving mode is selected by calling the 'WFI' instruction.
All peripherals remain active. During Wait mode, the I[1:0] bits of the CC register are forced to '10', to enable all interrupts. All other registers and memory remain unchanged. The MCU remains in Wait mode until an interrupt or RESET occurs, whereupon the Program Counter branches to the starting address of the interrupt or Reset service routine.
The MCU will remain in Wait mode until a Reset or an Interrupt occurs, causing it to wake up.

Refer to Figure 27.

Figure 27. Wait Mode Flow-chart



Note:

1. 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 recovered when the CC register is popped.

POWER SAVING MODES (Cont'd)

8.4.2 HALT MODE

The Halt mode is the lowest power consumption mode of the MCU. It is entered by executing the 'HALT' instruction when the OIE bit of the Main Clock Controller Status register (MCCSR) is cleared (see section 6.4 on page 37 for more details on the MCCSR register).

The MCU can exit Halt mode on reception of either a specific interrupt (see Table 8, "Interrupt Mapping," on page 44) or a RESET. When exiting Halt mode by means of a RESET or an interrupt, the oscillator is immediately turned on and the 256 or 4096 CPU cycle delay is used to stabilize the oscillator. After the start up delay, the CPU resumes operation by servicing the interrupt or by fetching the reset vector which woke it up (see Figure 31).

When entering 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 Halt mode, the main oscillator is turned off causing all internal processing to be stopped, including the operation of the on-chip peripherals. All peripherals are not clocked except the ones which get their clock supply from another clock generator (such as an external or auxiliary oscillator).

The compatibility of Watchdog operation with Halt mode is configured by the "WDGHALT" option bit of the option byte. The HALT instruction when executed while the Watchdog system is enabled, can generate a Watchdog RESET (see section 14.1 on page 290 for more details).

Figure 30. Halt Timing Overview

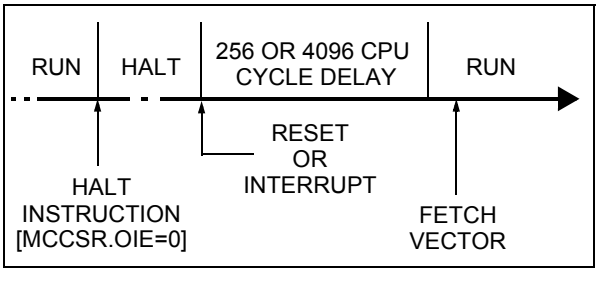
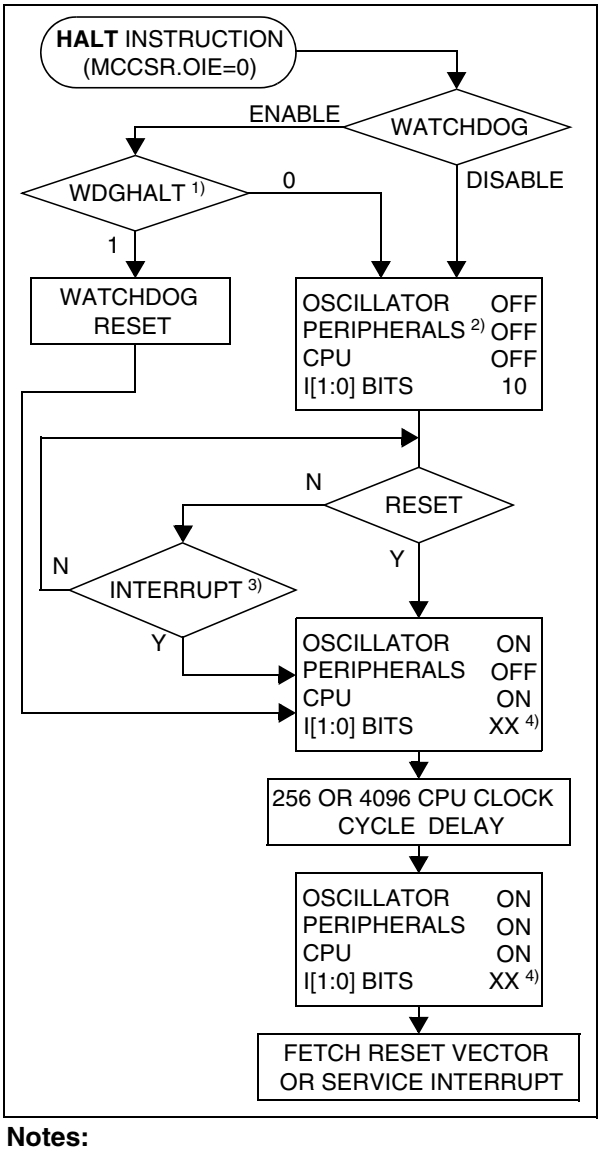


Figure 31. Halt Mode Flow-chart



Notes:

1. WDGHALT is an option bit. See option byte section for more details.

2. Peripheral clocked with an external clock source can still be active.

3. Only some specific interrupts can exit the MCU from Halt mode (such as external interrupt). Refer to Table 8, "Interrupt Mapping," on page 44 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 recovered when the CC register is popped.

WINDOW WATCHDOG (Cont'd)

Figure 36. Exact Timeout Duration (t_{min} and t_{max})

WHERE:

$t_{min0} = (LSB + 128) \times 64 \times t_{OSC2}$

$t_{max0} = 16384 \times t_{OSC2}$

$t_{OSC2} = 125ns$ if $f_{OSC2} = 8$ MHz

CNT = Value of T[5:0] bits in the WDGCR register (6 bits)
MSB and LSB are values from the table below depending on the timebase selected by the TB[1:0] bits in the MCCSR register

TB1 Bit (MCCSR Reg.)	TB0 Bit (MCCSR Reg.)	Selected MCCSR Timebase	MSB	LSB
0	0	2ms	4	59
0	1	4ms	8	53
1	0	10ms	20	35
1	1	25ms	49	54

To calculate the minimum Watchdog Timeout (t_{min}):

IF $CNT < \left\lceil \frac{MSB}{4} \right\rceil$ **THEN** $t_{min} = t_{min0} + 16384 \times CNT \times t_{osc2}$

ELSE $t_{min} = t_{min0} + \left[16384 \times \left(CNT - \left\lceil \frac{4CNT}{MSB} \right\rceil \right) + (192 + LSB) \times 64 \times \left\lceil \frac{4CNT}{MSB} \right\rceil \right] \times t_{osc2}$

To calculate the maximum Watchdog Timeout (t_{max}):

IF $CNT \leq \left\lceil \frac{MSB}{4} \right\rceil$ **THEN** $t_{max} = t_{max0} + 16384 \times CNT \times t_{osc2}$

ELSE $t_{max} = t_{max0} + \left[16384 \times \left(CNT - \left\lceil \frac{4CNT}{MSB} \right\rceil \right) + (192 + LSB) \times 64 \times \left\lceil \frac{4CNT}{MSB} \right\rceil \right] \times t_{osc2}$

Note: In the above formulae, division results must be rounded down to the next integer value.

Example:

With 2ms timeout selected in MCCSR register

Value of T[5:0] Bits in WDGCR Register (Hex.)	Min. Watchdog Timeout (ms) t_{min}	Max. Watchdog Timeout (ms) t_{max}
00	1.496	2.048
3F	128	128.552

SERIAL PERIPHERAL INTERFACE (cont'd)

10.4.6 Low Power Modes

Mode	Description
Wait	No effect on SPI. SPI interrupt events cause the device to exit from Wait mode.
Halt	SPI registers are frozen. In Halt mode, the SPI is inactive. SPI operation resumes when the device is woken up by an interrupt with "exit from Halt mode" capability. The data received is subsequently read from the SPIDR register when the software is running (interrupt vector fetching). If several data are received before the wake-up event, then an overrun error is generated. This error can be detected after the fetch of the interrupt routine that woke up the Device.

10.4.6.1 Using the SPI to wake up the device from Halt mode

In slave configuration, the SPI is able to wake up the device from Halt mode through a SPIF interrupt. The data received is subsequently read from the SPIDR register when the software is running (interrupt vector fetch). If multiple data transfers have been performed before software clears the SPIF bit, then the OVR bit is set by hardware.

Note: When waking up from Halt mode, if the SPI remains in Slave mode, it is recommended to perform an extra communications cycle to bring the SPI from Halt mode state to normal state. If the SPI exits from Slave mode, it returns to normal state immediately.

Caution: The SPI can wake up the device from Halt mode only if the Slave Select signal (external $\overline{SS}$ pin or the SSI bit in the SPICSR register) is low when the device enters Halt mode. So, if Slave selection is configured as external (see Section 10.4.3.2), make sure the master drives a low level on the $\overline{SS}$ pin when the slave enters Halt mode.

10.4.7 Interrupts

Interrupt Event	Event Flag	Enable Control Bit	Exit from Wait	Exit from Halt
SPI End of Transfer Event	SPIF	SPIE	Yes	Yes
Master Mode Fault Event	MODF			No
Overrun Error	OVR			

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

LINSCI™ SERIAL COMMUNICATION INTERFACE (SCI Mode) (cont'd)

10.5.5.3 Receiver

The SCI can receive data words of either 8 or 9 bits. When the M bit is set, word length is 9 bits and the MSB is stored in the R8 bit in the SCICR1 register.

Character reception

During a SCI reception, data shifts in least significant bit first through the RDI pin. In this mode, the SCIDR register consists or a buffer (RDR) between the internal bus and the received shift register (see Figure 62).

Procedure

– Select the M bit to define the word length.

– Select the desired baud rate using the SCIBRR and the SCIERPR registers.

– Set the RE bit, this enables the receiver which begins searching for a start bit.

When a character is received:

– The RDRF bit is set. It indicates that the content of the shift register is transferred to the RDR.

– An interrupt is generated if the RIE bit is set and the I[1:0] bits are cleared in the CCR register.

– The error flags can be set if a frame error, noise or an overrun error has been detected during reception.

Clearing the RDRF bit is performed by the following software sequence done by:

1. An access to the SCISR register

2. A read to the SCIDR register.

The RDRF bit must be cleared before the end of the reception of the next character to avoid an overrun error.

Idle Line

When an idle line is detected, there is the same procedure as a data received character plus an interrupt if the ILIE bit is set and the I[l1:0] bits are cleared in the CCR register.

Overrun Error

An overrun error occurs when a character is received when RDRF has not been reset. Data can not be transferred from the shift register to the TDR register as long as the RDRF bit is not cleared.

When an overrun error occurs:

– The OR bit is set.

– The RDR content will not be lost.

– The shift register will be overwritten.

– An interrupt is generated if the RIE bit is set and the I[l1:0] bits are cleared in the CCR register.

The OR bit is reset by an access to the SCISR register followed by a SCIDR register read operation.

Noise Error

Oversampling techniques are used for data recovery by discriminating between valid incoming data and noise.

When noise is detected in a character:

– The NF bit is set at the rising edge of the RDRF bit.

– 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 NF bit is reset by a SCISR register read operation followed by a SCIDR register read operation.

Framing Error

A framing error is detected when:

– The stop bit is not recognized on reception at the expected time, following either a desynchronization 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.

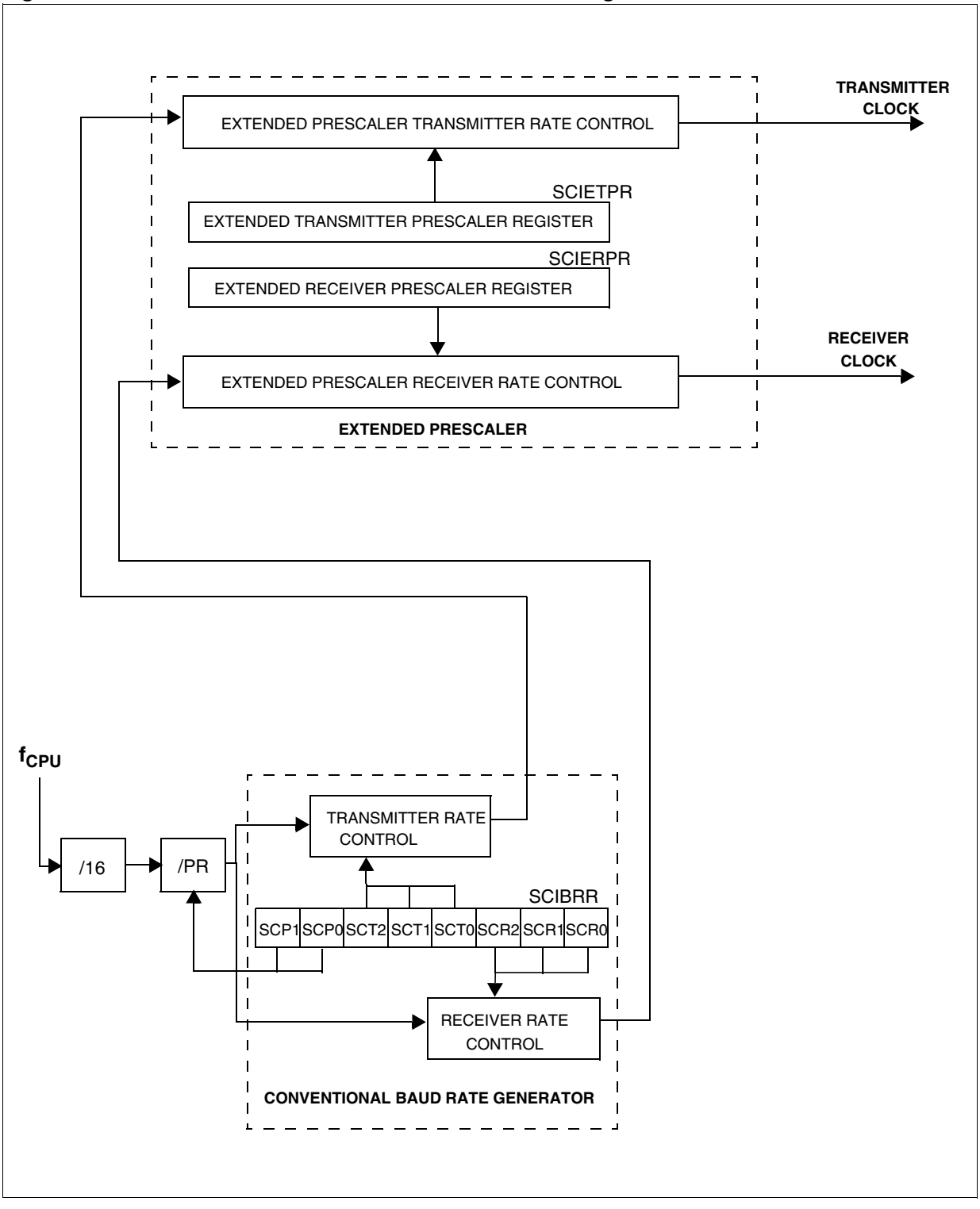
The FE bit is reset by a SCISR register read operation followed by a SCIDR register read operation.

Break Character

– When a break character is received, the SCI handles it as a framing error. To differentiate a break character from a framing error, it is necessary to read the SCIDR. If the received value is 00h, it is a break character. Otherwise it is a framing error.

LINSCI™ SERIAL COMMUNICATION INTERFACE (SCI Mode) (cont'd)

Figure 64. SCI Baud Rate and Extended Prescaler Block Diagram



MOTOR CONTROLLER (Cont'd)

10.6.6.5 Demagnetization (D) Event

At the end of the demagnetization phase, current no longer goes through the free-wheeling diodes. The voltage on the non-excited winding terminal goes from one of the power rail voltages to the common star connection voltage plus the BEMF voltage. In some cases (if the BEMF voltage is positive and the free-wheeling diodes are at ground for example) this end of demagnetization can be seen as a voltage edge on the selected MCIx input and it is called a hardware demagnetization event D_H. See Table 30.

The D event filter can be used to select the number of consecutive D events needed to generate the D_H event.

If enabled by the HDM bit in the MCRB register, the current value of the MTIM timer is captured in register MDREG when this event occurs in order to be able to simulate the demagnetization phase for the next steps.

When enabled by the SDM bit in the MCRB register, demagnetization can also be simulated by comparing the MTIM timer with the MDREG register. This kind of demagnetization is called simulated demagnetization D_S.

If the HDM and SDM bits are both set, the first event that occurs, triggers a demagnetization event. For this to work correctly, a D_S event must not precede a D_H event because the latter could be detected as a Z event.

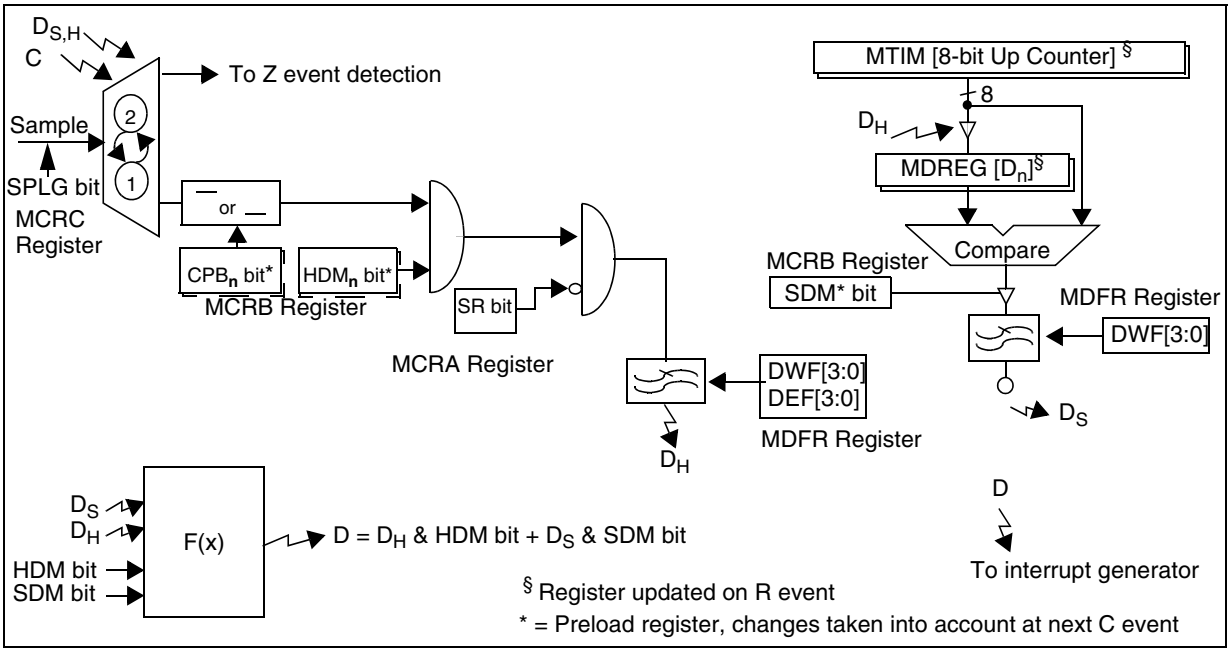
Simulated demagnetization can also be always used if the HDM bit is reset and the SDM bit is set. This mode works as a programmable masking time between the C_H and Z events. To drive the motor securely, the masking time must be always greater than the real demagnetization time in order to avoid a spurious Z event.

When an event occurs, (either D_H or D_S) the DI bit in the MISR register is set and an interrupt request is generated if the DIM bit of register MIMR is set.

Caution 1: Due to the alternate automatic capture and compare of the MTIM timer with MDREG register by D_H and D_S events, the MDREG register should be manipulated with special care.

Caution 2: Due to the event generation protection in the MZREG, MCOMP and MDREG registers for Soft Event generation (See "Built-in Checks and Controls for simulated events" on page 175.), the value written in the MDREG register in soft demagnetisation mode (SDM=1) is checked by hardware after the C event. If this value is less than or equal to the MTIM counter value at this moment, the Software demagnetisation event is generated immediately and the MTIM current value overwrites the value in the MDREG register to be able to reuse the right demagnetisation time for another simulated event generation.

Figure 81. D Event Generation Mechanism



MOTOR CONTROLLER (Cont'd)

10.6.6.10 Commutation Noise Filter

For D event detection and for Z event detection (when SPLG bit is set while DS[3:0] bits are reset), sampling is done at f_{SCF} during the PWM ON or OFF time ("Sampling block" on page 159). To avoid any erroneous detection due to PWM commutation noise, an hardware filter of 1µs (for $f_{PER-IPH}$ = 4Mhz) when PWM is put ON and when PWM is put OFF has been implemented. This means that, with sampling at 1MHz (1µs), due to this filter, 1 sample are ignored directly after the commutation.

This filter is active all the time for the D event and it is active for the Z event when the SPLG bit is set and DS[3:0] bits are cleared (meaning that the Z event is sampled at high frequency during the PWM ON or OFF time).

Table 34. Sensor/sensorless mode and D & Z event selection

SR bit	SPLG bit	DS[3:0] bits	Mode	OS[2:0] bits use	Event detection sampling clock	Sampling behaviour for Z event detection	Window and Event Filters	Behaviour of the output PWM
0	0	000	Sensors not used	Enabled	D: f_{SCF} Z: SA&OT config. PWM frequency	At the end of the off time of the PWM signal	D Window Filter DWF[3:0] after C event D Event Filter DEF[3:0] after DWF Z Window Filter ZWF[3:0] after D event Z Event Filter ZEF[3:0] after ZWF See Table 30 on page 152	"Before D" behaviour, "between D and Z" behaviour and "after Z" behaviour
0	1	000	Sensors not used	Enabled	D: f_{SCF} Z: f_{SCF}	During off time or ON time of the PWM signal		"Before D" behaviour, "between D and Z" behaviour and "after Z" behaviour
0	0	Not equal to 000	Sensors not used	Enabled	D: f_{SCF} Z: SA&OT config. PWM frequency	During ON time of the PWM signal		"Before D" behaviour, "between D and Z" behaviour and "after Z" behaviour
0	1	Not equal to 000	Sensors not used	Enabled	D: f_{SCF} Z: f_{SCF}	During ON time of the PWM signal		"Before D" behaviour, "between D and Z" behaviour and "after Z" behaviour
1	x	xxx	Position Sensors used	OS1 disabled	Z: f_{SCF}	During OFF time or ON time of the PWM signal	No Z Window Filter Only Z Event Filter is active in Sensor mode	"Before Z" behaviour and "after Z" behaviour

Note: For f_{SCF} selection, see Table 82

MOTOR CONTROLLER (Cont'd)

When using hardware commutation C_H, the sequence of events needed is C_H then D and finally Z events and the value written in the registers are checked at different times.

If SDM bit is set, meaning simulated demagnetisation, a value must be written in the MDREG register to generate the simulated demagnetisation. This value must be written after the C (either C_s or C_H) event preceding the simulated demagnetisation.

If SZ bit is set, meaning simulated zero-crossing event, a value must be written in the MZREG register to generate the simulated zero-crossing. This value must be written after the D event (D_H or D_S) preceding the simulated zero-crossing.

When using simulated commutation (C_S), the result of the 8*8 hardware multiplication of the delay manager is not taken in account and must be overwritten if the SC bit has been set in a Z event interrupt and the sequence of events is broken meaning that several consecutive simulated commutations can be implemented.

As soon as the SC bit is set in the MCRC register, the system won't necessarily expect a D event after a C event. This can be used for an application in sensor mode with only one Hall Effect sensor for example.

Be careful that the D and Z events are not ignored by the peripheral, this means that for example if a Z event occurs, the MTIM timer is reset. In Simulated Commutation mode, the sequence D -> Z is expected, and this order must be repected.

As the sequence of events may not be the same when using simulated commutation, as soon as the SC bit is set, the capture/compare feature and protection on MCOMP register is reestablished only after a write to the MCOMP register. This means that as soon as the SC bit is set, if no write access is done to the MCOMP register, no commutation event will be generated, whatever the value of MCOMP compared to MTIM at the time SC is set. This does not depend on the running mode: switched or autoswitched mode (SWA bit). If software commutation event is used with a normal sequence of events C-->D-->Z, it is recommended to write the MCOMP register during the Z interrupt routine to avoid any spurious comparison as several consecutive C_s events can be generated.

Note that two different simulated events can be used in the same step (like D_S followed by Z_S).

Note also that for more precision, it is recommended to use the value captured from the preceding hardware event to compute the value used to generate simulated events.

Figure 95, Figure 96 and Figure 97 shows details of simulated event generation.

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 >								

12.2 ABSOLUTE MAXIMUM RATINGS

Stresses above those listed as “absolute maximum ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device under these conditions is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

12.2.1 Voltage Characteristics

Symbol	Ratings	Maximum value	Unit
V_{DD} - V_{SS}	Supply voltage	6.5	V
V_{PP} - V_{SS}	Programming Voltage	13	
V_{IN}	Input voltage on any pin 1) & 2)	V_{SS}-0.3 to V_{DD}+0.3	
$\lvert\Delta V_{DDx}\rvert$ and $\lvert\Delta V_{SSx}\rvert$	Variations between different digital power pins	50	mV
$\lvert V_{SSA} - V_{SSx}\rvert$	Variations between digital and analog ground pins	50	
$V_{ESD(HBM)}$	Electro-static discharge voltage (Human Body Model)	see section 12.7.3 on page 263	
$V_{ESD(CDM)}$	Electro-static discharge voltage (Charged Device Model)		

12.2.2 Current Characteristics

Symbol	Ratings		Maximum value	Unit
I_{VDD}	Total current into V_{DD} power lines (source) 3)	32-pin devices	75	mA
		44-pin devices	125	
		56, 64, 80-pin devices	175	
I_{VSS}	Total current out of V_{SS} ground lines (sink) 3)	32-pin devices	75	
		44-pin devices	125	
		56, 64, 80-pin devices	175	
I_{IO}	Output current sunk by any standard I/O and control pin		25	
	Output current sunk by any high sink I/O pin		50	
	Output current source by any I/Os and control pin		- 25	
$I_{INJ(PIN)}$ 2) & 4)	Injected current on V_{PP} pin		± 5	
	Injected current on $\overline{\text{RESET}}$ pin		± 5	
	Injected current on OSC1 and OSC2 pins		± 5	
	Injected current on any other pin 5)		± 5	
$\Sigma I_{INJ(PIN)}$ 2)	Total injected current (sum of all I/O and control pins) 5)		± 20	

Notes:

1. Directly connecting the $\overline{\text{RESET}}$ and I/O pins to V_{DD} or V_{SS} could damage the device if an unintentional internal reset is generated or an unexpected change of the I/O configuration occurs (for example, due to a corrupted program counter). To guarantee safe operation, this connection has to be done through a pull-up or pull-down resistor (typical: 4.7kΩ for $\overline{\text{RESET}}$, 10kΩ for I/Os). For the same reason, unused I/O pins must not be directly tied to V_{DD} or V_{SS}.
2. $I_{INJ(PIN)}$ must never be exceeded. This is implicitly insured if V_{IN} maximum is respected. If V_{IN} maximum cannot be respected, the injection current must be limited externally to the $I_{INJ(PIN)}$ value. A positive injection is induced by $V_{IN}>V_{DD}$ while a negative injection is induced by $V_{IN}<V_{SS}$.
3. All power (V_{DD}) and ground (V_{SS}) lines must always be connected to the external supply.
4. Negative injection disturbs the analog performance of the device. See note in “ADC Accuracy with VDD=5.0V” on page 284.
For best reliability, it is recommended to avoid negative injection of more than 1.6mA
5. When several inputs are submitted to a current injection, the maximum $\Sigma I_{INJ(PIN)}$ is the absolute sum of the positive and negative injected currents (instantaneous values). These results are based on characterisation with $\Sigma I_{INJ(PIN)}$ maximum current injection on four I/O port pins of the device.

12.10 TIMER PERIPHERAL CHARACTERISTICS

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

Refer to I/O port characteristics for more details on the input/output alternate function characteristics (output compare, input capture, external clock, PWM output...).

12.10.1 8-Bit PWM-ART Auto-Reload Timer

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$t_{res(PWM)}$	PWM resolution time		1			t_{CPU}
		f_{CPU}=8MHz	125			ns
f_{EXT}	ART external clock frequency		0		$f_{CPU}/2$	MHz
f_{PWM}	PWM repetition rate		0		$f_{CPU}/2$	
Res_{PWM}	PWM resolution				8	bit
V_{OS}	PWM/DAC output step voltage	V_{DD}=5V, Res=8-bits		20		mV

12.10.2 16-Bit Timer

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$t_{w(ICAP)in}$	Input capture pulse time		1			t_{CPU}
$t_{res(PWM)}$	PWM resolution time		2			t_{CPU}
		f_{CPU}=8MHz	250			ns
f_{EXT}	Timer external clock frequency		0		$f_{CPU}/4$	MHz
f_{PWM}	PWM repetition rate		0		$f_{CPU}/4$	MHz
Res_{PWM}	PWM resolution				16	bit

14.4 ST7 APPLICATION NOTES

Table 93. ST7 Application Notes

IDENTIFICATION	DESCRIPTION
APPLICATION EXAMPLES	
AN1658	SERIAL NUMBERING IMPLEMENTATION
AN1720	MANAGING THE READ-OUT PROTECTION IN FLASH MICROCONTROLLERS
AN1755	A HIGH RESOLUTION/PRECISION THERMOMETER USING ST7 AND NE555
AN1756	CHOOSING A DALI IMPLEMENTATION STRATEGY WITH ST7DALI
AN1812	A HIGH PRECISION, LOW COST, SINGLE SUPPLY ADC FOR POSITIVE AND NEGATIVE INPUT VOLTAGES
EXAMPLE DRIVERS	
AN 969	SCI COMMUNICATION BETWEEN ST7 AND PC
AN 970	SPI COMMUNICATION BETWEEN ST7 AND EEPROM
AN 971	I²C COMMUNICATION BETWEEN ST7 AND M24CXX EEPROM
AN 972	ST7 SOFTWARE SPI MASTER COMMUNICATION
AN 973	SCI SOFTWARE COMMUNICATION WITH A PC USING ST72251 16-BIT TIMER
AN 974	REAL TIME CLOCK WITH ST7 TIMER OUTPUT COMPARE
AN 976	DRIVING A BUZZER THROUGH ST7 TIMER PWM FUNCTION
AN 979	DRIVING AN ANALOG KEYBOARD WITH THE ST7 ADC
AN 980	ST7 KEYPAD DECODING TECHNIQUES, IMPLEMENTING WAKE-UP ON KEYSTROKE
AN1017	USING THE ST7 UNIVERSAL SERIAL BUS MICROCONTROLLER
AN1041	USING ST7 PWM SIGNAL TO GENERATE ANALOG OUTPUT (SINUSOÏD)
AN1042	ST7 ROUTINE FOR I²C SLAVE MODE MANAGEMENT
AN1044	MULTIPLE INTERRUPT SOURCES MANAGEMENT FOR ST7 MCUS
AN1045	ST7 S/W IMPLEMENTATION OF I²C BUS MASTER
AN1046	UART EMULATION SOFTWARE
AN1047	MANAGING RECEPTION ERRORS WITH THE ST7 SCI PERIPHERALS
AN1048	ST7 SOFTWARE LCD DRIVER
AN1078	PWM DUTY CYCLE SWITCH IMPLEMENTING TRUE 0% & 100% DUTY CYCLE
AN1082	DESCRIPTION OF THE ST72141 MOTOR CONTROL PERIPHERALS REGISTERS
AN1083	ST72141 BLDC MOTOR CONTROL SOFTWARE AND FLOWCHART EXAMPLE
AN1105	ST7 PCAN PERIPHERAL DRIVER
AN1129	PWM MANAGEMENT FOR BLDC MOTOR DRIVES USING THE ST72141
AN1130	AN INTRODUCTION TO SENSORLESS BRUSHLESS DC MOTOR DRIVE APPLICATIONS WITH THE ST72141
AN1148	USING THE ST7263 FOR DESIGNING A USB MOUSE
AN1149	HANDLING SUSPEND MODE ON A USB MOUSE
AN1180	USING THE ST7263 KIT TO IMPLEMENT A USB GAME PAD
AN1276	BLDC MOTOR START ROUTINE FOR THE ST72141 MICROCONTROLLER
AN1321	USING THE ST72141 MOTOR CONTROL MCU IN SENSOR MODE
AN1325	USING THE ST7 USB LOW-SPEED FIRMWARE V4.X
AN1445	EMULATED 16-BIT SLAVE SPI
AN1475	DEVELOPING AN ST7265X MASS STORAGE APPLICATION
AN1504	STARTING A PWM SIGNAL DIRECTLY AT HIGH LEVEL USING THE ST7 16-BIT TIMER
AN1602	16-BIT TIMING OPERATIONS USING ST7262 OR ST7263B ST7 USB MCUS
AN1633	DEVICE FIRMWARE UPGRADE (DFU) IMPLEMENTATION IN ST7 NON-USB APPLICATIONS
AN1712	GENERATING A HIGH RESOLUTION SINEWAVE USING ST7 PWMART
AN1713	SMBUS SLAVE DRIVER FOR ST7 I2C PERIPHERALS
AN1753	SOFTWARE UART USING 12-BIT ART

Table 93. ST7 Application Notes

IDENTIFICATION	DESCRIPTION
AN1947	ST7MC PMAC SINE WAVE MOTOR CONTROL SOFTWARE LIBRARY
GENERAL PURPOSE	
AN1476	LOW COST POWER SUPPLY FOR HOME APPLIANCES
AN1526	ST7FLITE0 QUICK REFERENCE NOTE
AN1709	EMC DESIGN FOR ST MICROCONTROLLERS
AN1752	ST72324 QUICK REFERENCE NOTE
PRODUCT EVALUATION	
AN 910	PERFORMANCE BENCHMARKING
AN 990	ST7 BENEFITS VS INDUSTRY STANDARD
AN1077	OVERVIEW OF ENHANCED CAN CONTROLLERS FOR ST7 AND ST9 MCUS
AN1086	U435 CAN-DO SOLUTIONS FOR CAR MULTIPLEXING
AN1103	IMPROVED B-EMF DETECTION FOR LOW SPEED, LOW VOLTAGE WITH ST72141
AN1150	BENCHMARK ST72 VS PC16
AN1151	PERFORMANCE COMPARISON BETWEEN ST72254 & PC16F876
AN1278	LIN (LOCAL INTERCONNECT NETWORK) SOLUTIONS
PRODUCT MIGRATION	
AN1131	MIGRATING APPLICATIONS FROM ST72511/311/214/124 TO ST72521/321/324
AN1322	MIGRATING AN APPLICATION FROM ST7263 REV.B TO ST7263B
AN1365	GUIDELINES FOR MIGRATING ST72C254 APPLICATIONS TO ST72F264
AN1604	HOW TO USE ST7MDT1-TRAIN WITH ST72F264
AN2200	GUIDELINES FOR MIGRATING ST7LITE1X APPLICATIONS TO ST7FLITE1XB
PRODUCT OPTIMIZATION	
AN 982	USING ST7 WITH CERAMIC RESONATOR
AN1014	HOW TO MINIMIZE THE ST7 POWER CONSUMPTION
AN1015	SOFTWARE TECHNIQUES FOR IMPROVING MICROCONTROLLER EMC PERFORMANCE
AN1040	MONITORING THE VBUS SIGNAL FOR USB SELF-POWERED DEVICES
AN1070	ST7 CHECKSUM SELF-CHECKING CAPABILITY
AN1181	ELECTROSTATIC DISCHARGE SENSITIVE MEASUREMENT
AN1324	CALIBRATING THE RC OSCILLATOR OF THE ST7FLITE0 MCU USING THE MAINS
AN1502	EMULATED DATA EEPROM WITH ST7 HDFLASH MEMORY
AN1529	EXTENDING THE CURRENT & VOLTAGE CAPABILITY ON THE ST7265 VDDF SUPPLY
AN1530	ACCURATE TIMEBASE FOR LOW-COST ST7 APPLICATIONS WITH INTERNAL RC OSCILLATOR
AN1605	USING AN ACTIVE RC TO WAKEUP THE ST7LITE0 FROM POWER SAVING MODE
AN1636	UNDERSTANDING AND MINIMIZING ADC CONVERSION ERRORS
AN1828	PIR (PASSIVE INFRARED) DETECTOR USING THE ST7FLITE05/09/SUPERLITE
AN1946	SENSORLESS BLDC MOTOR CONTROL AND BEMF SAMPLING METHODS WITH ST7MC
AN1953	PFC FOR ST7MC STARTER KIT
AN1971	ST7LITE0 MICROCONTROLLED BALLAST
PROGRAMMING AND TOOLS	
AN 978	ST7 VISUAL DEVELOP SOFTWARE KEY DEBUGGING FEATURES
AN 983	KEY FEATURES OF THE COSMIC ST7 C-COMPILER PACKAGE
AN 985	EXECUTING CODE IN ST7 RAM
AN 986	USING THE INDIRECT ADDRESSING MODE WITH ST7
AN 987	ST7 SERIAL TEST CONTROLLER PROGRAMMING
AN 988	STARTING WITH ST7 ASSEMBLY TOOL CHAIN
AN1039	ST7 MATH UTILITY ROUTINES