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	16MHz
Connectivity	LINbusSCI, SPI
Peripherals	LVD, POR, PWM, WDT
Number of I/O	15
Program Memory Size	8KB (8K x 8)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	384 x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 7x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	20-SOIC (0.295", 7.50mm Width)
Supplier Device Package	20-SO
Purchase URL	https://www.e-xfl.com/product-detail/stmicroelectronics/st7flite39f2m3tr

5 DATA EEPROM

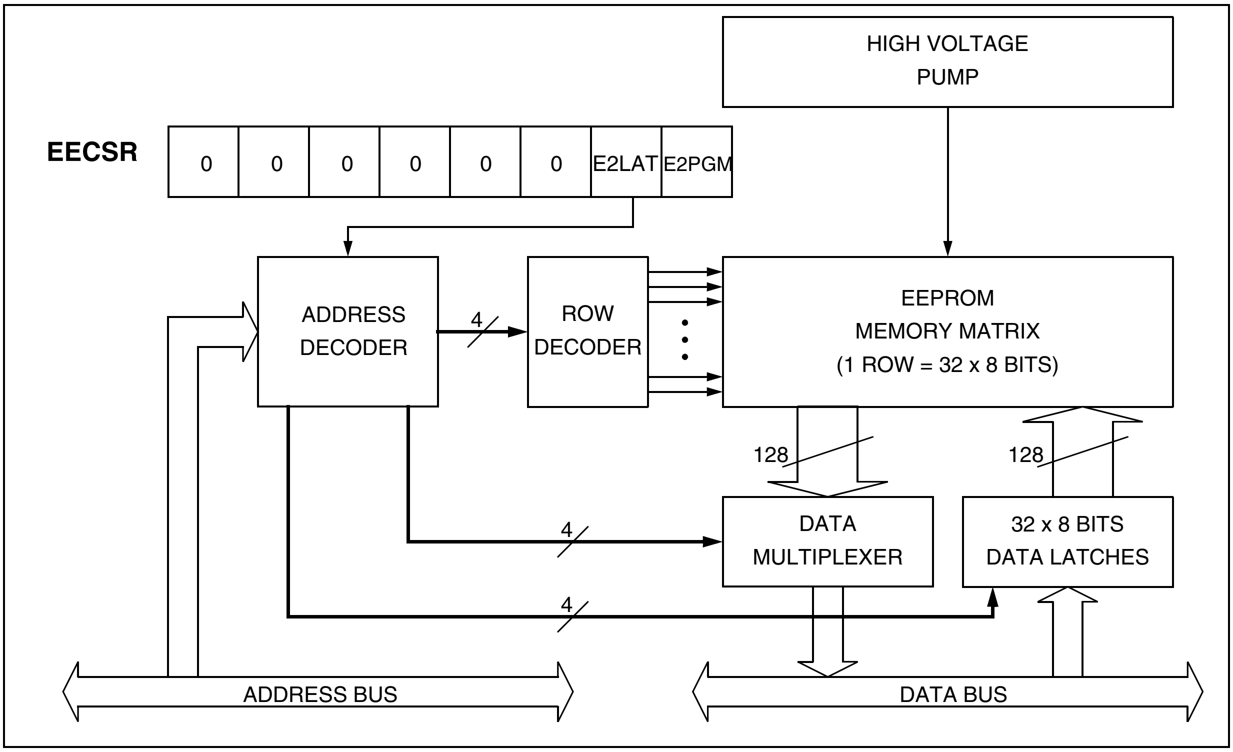
5.1 INTRODUCTION

The Electrically Erasable Programmable Read Only Memory can be used as a non volatile back-up for storing data. Using the EEPROM requires a basic access protocol described in this chapter.

5.2 MAIN FEATURES

- Up to 32 Bytes programmed in the same cycle
- EEPROM mono-voltage (charge pump)
- Chained erase and programming cycles
- Internal control of the global programming cycle duration
- WAIT mode management
- Readout protection

Figure 6. EEPROM Block Diagram



RESET SEQUENCE MANAGER (Cont'd)

The RESET pin is an asynchronous signal which plays a major role in EMS performance. In a noisy environment, it is recommended to follow the guidelines mentioned in the electrical characteristics section.

7.5.3 External Power-On RESET

If the LVD is disabled by option byte, to start up the microcontroller correctly, the user must ensure by means of an external reset circuit that the reset signal is held low until V_{DD} is over the minimum level specified for the selected f_{OSC} frequency.

A proper reset signal for a slow rising V_{DD} supply can generally be provided by an external RC network connected to the RESET pin.

7.5.4 Internal Low Voltage Detector (LVD) RESET

Two different RESET sequences caused by the internal LVD circuitry can be distinguished:

- Power-On RESET
- Voltage Drop RESET

The device RESET pin acts as an output that is pulled low when $V_{DD}<V_{IT+}$ (rising edge) or $V_{DD}<V_{IT-}$ (falling edge) as shown in Figure 16.

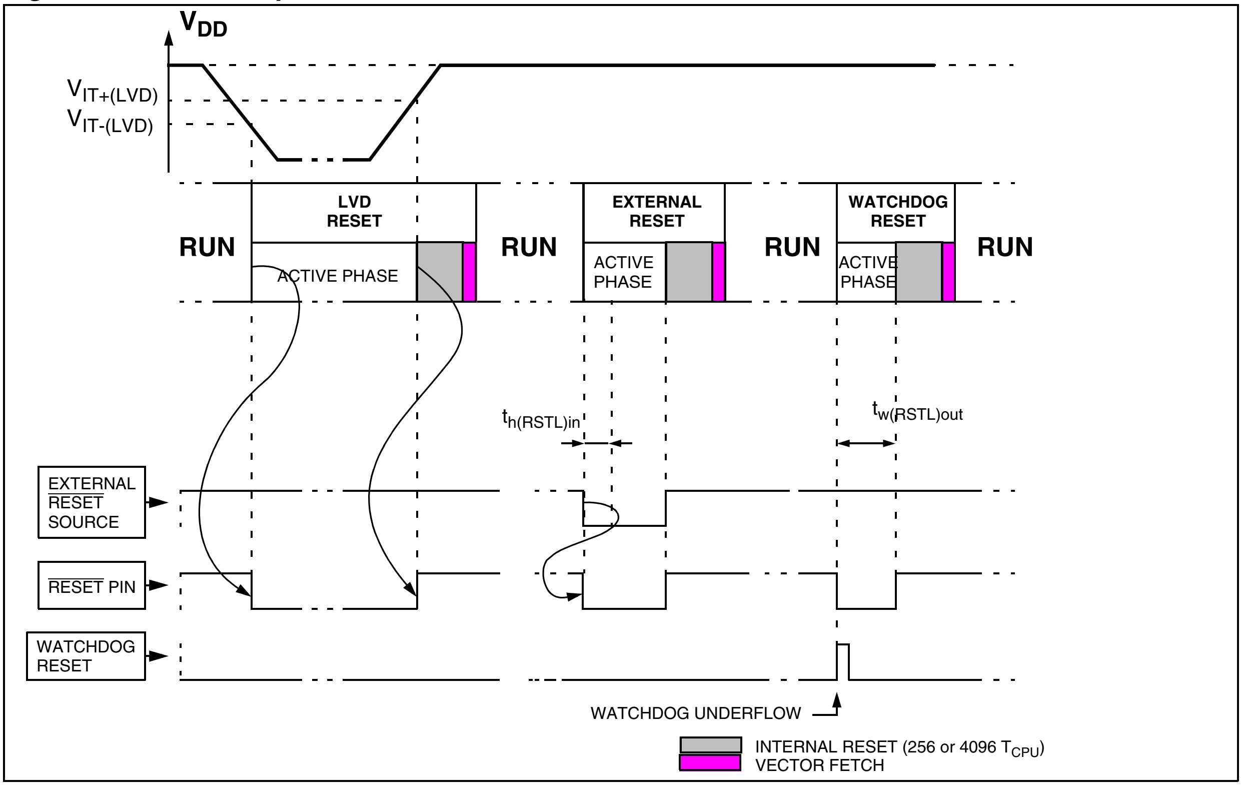
The LVD filters spikes on V_{DD} larger than $t_{g(VDD)}$ to avoid parasitic resets.

7.5.5 Internal Watchdog RESET

The RESET sequence generated by a internal Watchdog counter overflow is shown in Figure 16.

Starting from the Watchdog counter underflow, the device RESET pin acts as an output that is pulled low during at least $t_{w(RSTL)out}$.

Figure 16. RESET Sequences



8 INTERRUPTS

The ST7 core may be interrupted by one of two different methods: Maskable hardware interrupts as listed in the “interrupt mapping” table and a non-maskable software interrupt (TRAP). The Interrupt processing flowchart is shown in Figure 20.

The maskable interrupts must be enabled by clearing the I bit in order to be serviced. However, disabled interrupts may be latched and processed when they are enabled (see external interrupts subsection).

Note: After reset, all interrupts are disabled.

When an interrupt has to be serviced:

– Normal processing is suspended at the end of the current instruction execution.

– The PC, X, A and CC registers are saved onto the stack.

– The I bit of the CC register is set to prevent additional interrupts.

– The PC is then loaded with the interrupt vector of the interrupt to service and the first instruction of the interrupt service routine is fetched (refer to the Interrupt Mapping table for vector addresses).

The interrupt service routine should finish with the IRET instruction which causes the contents of the saved registers to be recovered from the stack.

Note: As a consequence of the IRET instruction, the I bit is cleared and the main program resumes.

Priority Management

By default, a servicing interrupt cannot be interrupted because the I bit is set by hardware entering in interrupt routine.

In the case when several interrupts are simultaneously pending, an hardware priority defines which one will be serviced first (see the Interrupt Mapping table).

Interrupts and Low Power Mode

All interrupts allow the processor to leave the WAIT low power mode. Only external and specifically mentioned interrupts allow the processor to leave the HALT low power mode (refer to the “Exit from HALT” column in the Interrupt Mapping table).

8.1 NON MASKABLE SOFTWARE INTERRUPT

This interrupt is entered when the TRAP instruction is executed regardless of the state of the I bit. It is serviced according to the flowchart in Figure 20.

8.2 EXTERNAL INTERRUPTS

External interrupt vectors can be loaded into the PC register if the corresponding external interrupt occurred and if the I bit is cleared. These interrupts allow the processor to leave the HALT low power mode.

The external interrupt polarity is selected through the miscellaneous register or interrupt register (if available).

An external interrupt triggered on edge will be latched and the interrupt request automatically cleared upon entering the interrupt service routine.

Caution: The type of sensitivity defined in the Miscellaneous or Interrupt register (if available) applies to the ei source. In case of a NANDed source (as described in the I/O ports section), a low level on an I/O pin, configured as input with interrupt, masks the interrupt request even in case of rising-edge sensitivity.

8.3 PERIPHERAL INTERRUPTS

Different peripheral interrupt flags in the status register are able to cause an interrupt when they are active if both:

– The I bit of the CC register is cleared.

– The corresponding enable bit is set in the control register.

If any of these two conditions is false, the interrupt is latched and thus remains pending.

Clearing an interrupt request is done by:

– Writing “0” to the corresponding bit in the status register or

– Access to the status register while the flag is set followed by a read or write of an associated register.

Note: The clearing sequence resets the internal latch. A pending interrupt (that is, waiting for being enabled) will therefore be lost if the clear sequence is executed.

INTERRUPTS (Cont'd)

EXTERNAL INTERRUPT CONTROL REGISTER (EICR)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
IS31	IS30	IS21	IS20	IS11	IS10	IS01	IS00

Bit 7:6 = **IS3[1:0]** *ei3 sensitivity*
These bits define the interrupt sensitivity for ei3 (Port B0) according to Table 7.

Bit 5:4 = **IS2[1:0]** *ei2 sensitivity*
These bits define the interrupt sensitivity for ei2 (Port B3) according to Table 7.

Bit 3:2 = **IS1[1:0]** *ei1 sensitivity*
These bits define the interrupt sensitivity for ei1 (Port A7) according to Table 7.

Bit 1:0 = **IS0[1:0]** *ei0 sensitivity*
These bits define the interrupt sensitivity for ei0 (Port A0) according to Table 7.

Note: These 8 bits can be written only when the I bit in the CC register is set.

Table 7. Interrupt Sensitivity Bits

ISx1	ISx0	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

.

EXTERNAL INTERRUPT SELECTION REGISTER (EISR)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
ei31	ei30	ei21	ei20	ei11	ei10	ei01	ei00

Bit 7:6 = **ei3[1:0]** *ei3 pin selection*
These bits are written by software. They select the Port B I/O pin used for the ei3 external interrupt according to the table below.

External Interrupt I/O pin selection

ei31	ei30	I/O Pin
0	0	No interrupt *
0	1	PB0
1	0	PB1
1	1	PB2

* Reset State

Bit 5:4 = **ei2[1:0]** *ei2 pin selection*
These bits are written by software. They select the Port B I/O pin used for the ei2 external interrupt according to the table below.

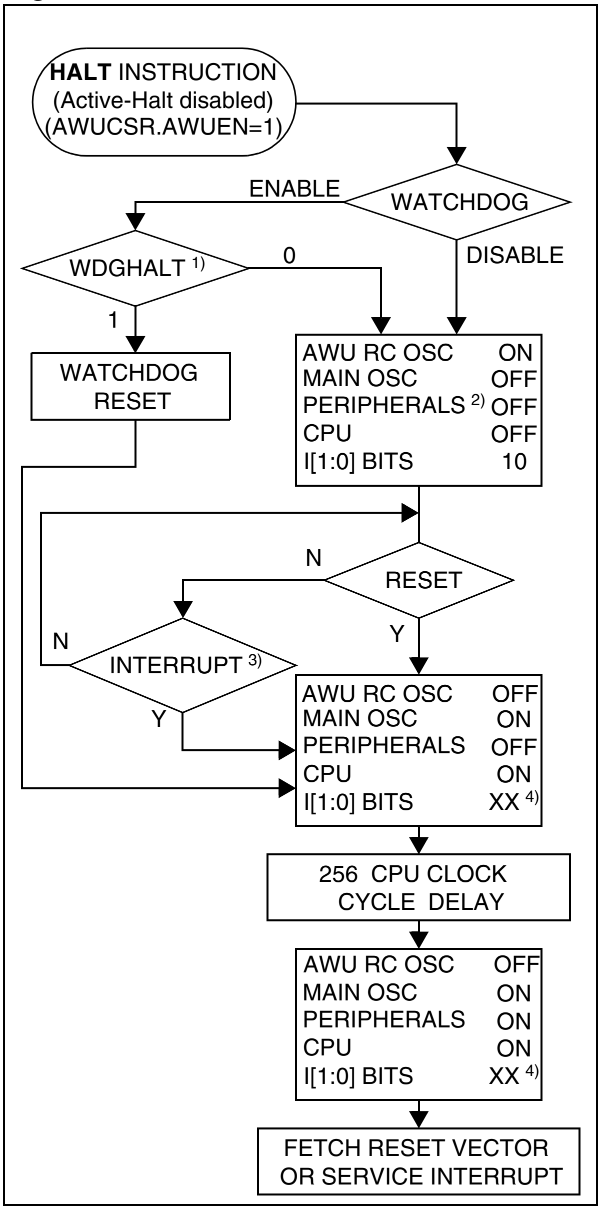
External Interrupt I/O pin selection

ei21	ei20	I/O Pin
0	0	No interrupt *
0	1	PB3
1	0	PB5
1	1	PB6

* Reset State

POWER SAVING MODES (Cont'd)

Figure 30. AWUFH Mode Flow-chart



Notes:

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

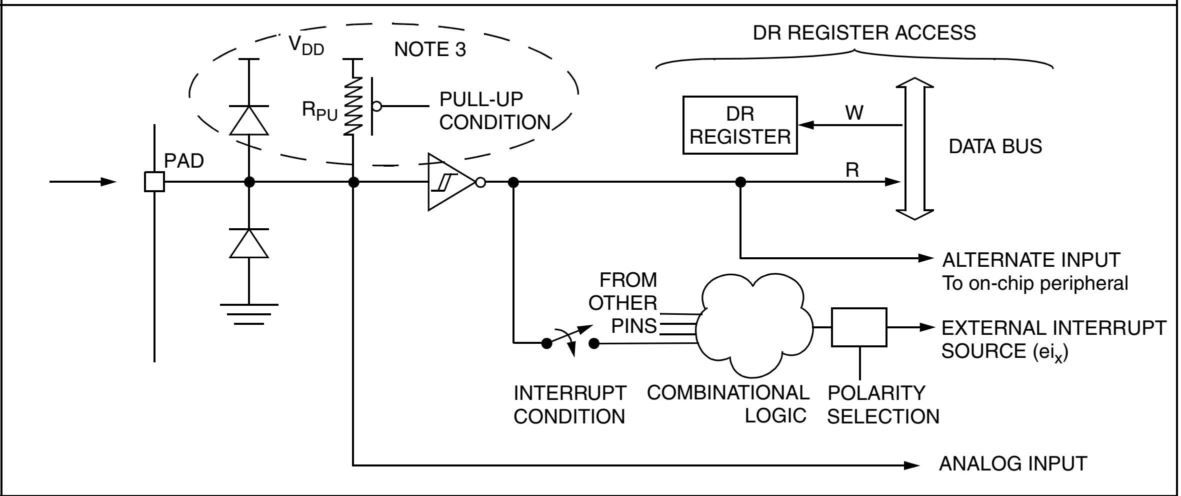
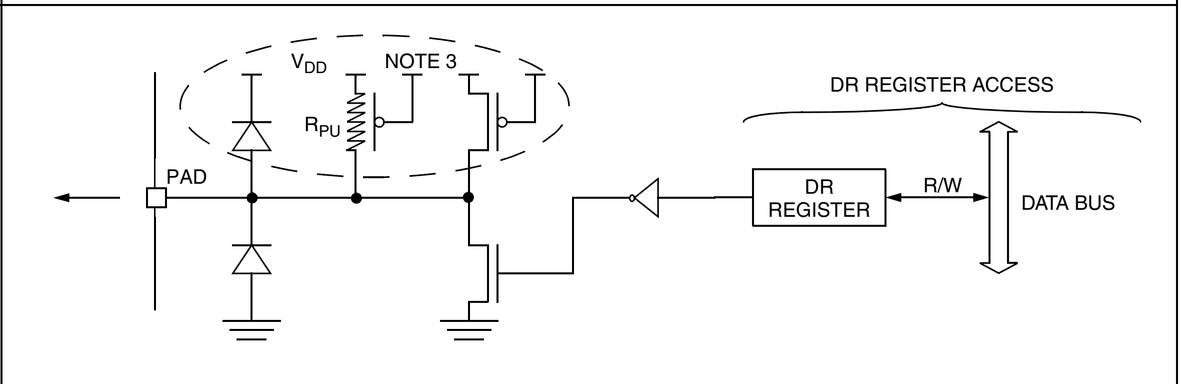
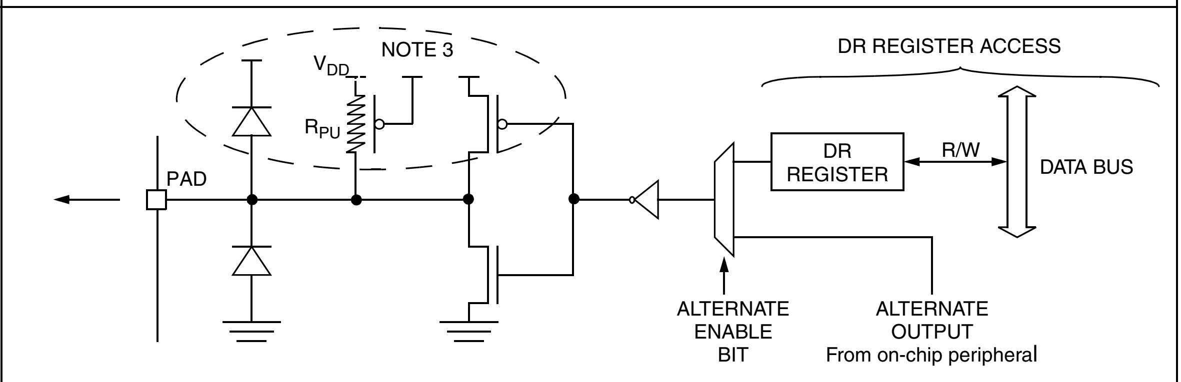
2. Peripheral clocked with an external clock source can still be active.

3. Only an AWUFH interrupt and some specific interrupts can exit the MCU from HALT mode (such as external interrupt). Refer to Table 6, "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 recovered when the CC register is popped.

I/O PORTS (Cont'd)

Table 10. I/O Configurations

	Hardware Configuration
INPUT 1)	
OPEN-DRAIN OUTPUT 2)	
PUSH-PULL OUTPUT 2)	

Notes:

1. When the I/O port is in input configuration and the associated alternate function is enabled as an output, reading the DR register will read the alternate function output status.
2. When the I/O port is in output configuration and the associated alternate function is enabled as an input, the alternate function reads the pin status given by the DR register content.
3. For true open drain, these elements are not implemented.

I/O PORTS (Cont'd)

Analog alternate function

Configure the I/O as floating input to use an ADC input. The analog multiplexer (controlled by the ADC registers) switches the analog voltage present on the selected pin to the common analog rail, connected to the ADC input.

Analog Recommendations

Do not change the voltage level or loading on any I/O while conversion is in progress. Do not have clocking pins located close to a selected analog pin.

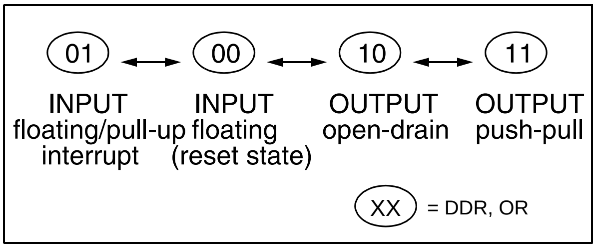
WARNING: The analog input voltage level must be within the limits stated in the absolute maximum ratings.

10.3 I/O PORT IMPLEMENTATION

The hardware implementation on each I/O port depends on the settings in the DDR and OR registers and specific I/O port features such as ADC input or open drain.

Switching these I/O ports from one state to another should be done in a sequence that prevents unwanted side effects. Recommended safe transitions are illustrated in Figure 32. Other transitions are potentially risky and should be avoided, since they may present unwanted side-effects such as spurious interrupt generation.

Figure 32. Interrupt I/O Port State Transitions



10.4 UNUSED I/O PINS

Unused I/O pins must be connected to fixed voltage levels. Refer to Section 13.8.

10.5 LOW POWER MODES

Mode	Description
WAIT	No effect on I/O ports. External interrupts cause the device to exit from WAIT mode.
HALT	No effect on I/O ports. External interrupts cause the device to exit from HALT mode.

10.6 INTERRUPTS

The external interrupt event generates an interrupt if the corresponding configuration is selected with DDR and OR registers and if the I bit in the CC register is cleared (RIM instruction).

Interrupt Event	Event Flag	Enable Control Bit	Exit from Wait	Exit from Halt
External interrupt on selected external event	-	DDRx ORx	Yes	Yes

Related Documentation

AN 970: SPI Communication between ST7 and EEPROM

AN1045: S/W implementation of I2C bus master

AN1048: Software LCD driver

I/O PORTS (Cont'd)

The I/O port register configurations are summarised as follows.

Standard Ports

PA7:0, PB6:0

MODE	DDR	OR
floating input	0	0
pull-up input	0	1
open drain output	1	0
push-pull output	1	1

Interrupt Ports

Ports where the external interrupt capability is selected using the EISR register

MODE	DDR	OR
floating input	0	0
pull-up interrupt input	0	1

Table 11. Port Configuration (Standard ports)

Port	Pin name	Input (DDR=0)		Output (DDR=1)	
		OR = 0	OR = 1	OR = 0	OR = 1
Port A	PA7:0	floating	pull-up	open drain	push-pull
Port B	PB6:0	floating	pull-up	open drain	push-pull

Note: On ports where the external interrupt capability is selected using the EISR register, the configuration will be as follows:

Table 12. Port Configuration (external interrupts)

Port	Pin name	Input with interrupt (DDR=0 ; EISR≠00)	
		OR = 0	OR = 1
Port A	PA6:1	floating	pull-up
Port B	PB5:0	floating	pull-up

Table 13. I/O Port Register Map and Reset Values

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
0000h	**PADR**	MSB							LSB
	Reset Value	1	1	1	1	1	1	1	1
0001h	**PADDR**	MSB							LSB
	Reset Value	0	0	0	0	0	0	0	0
0002h	**PAOR**	MSB							LSB
	Reset Value	0	1	0	0	0	0	0	0
0003h	**PBDR**	MSB							LSB
	Reset Value	1	1	1	1	1	1	1	1
0004h	**PBDDR**	MSB							LSB
	Reset Value	0	0	0	0	0	0	0	0
0005h	**PBOR**	MSB							LSB
	Reset Value	0	0	0	0	0	0	0	0

DUAL 12-BIT AUTORELOAD TIMER 3 (Cont'd)

11.2.3 Functional Description

11.2.3.1 PWM Mode

This mode allows up to four Pulse Width Modulated signals to be generated on the PWMx output pins.

PWM Frequency

The four PWM signals can have the same frequency (f_{PWM}) or can have two different frequencies. This is selected by the ENCNTR2 bit which enables single timer or dual timer mode (see Figure 34 and Figure 35).

The frequency is controlled by the counter period and the ATR register value. In dual timer mode, PWM2 and PWM3 can be generated with a different frequency controlled by CNTR2 and ATR2.

$$f_{PWM} = f_{COUNTER} / (4096 - ATR)$$

Following the above formula,

– If $f_{COUNTER}$ is 4 Mhz, the maximum value of f_{PWM} is 2 MHz (ATR register value = 4094),the minimum value is 1 KHz (ATR register value = 0).

Duty Cycle

The duty cycle is selected by programming the DCRx registers. These are preload registers. The DCRx values are transferred in Active duty cycle registers after an overflow event if the corresponding transfer bit (TRANx bit) is set.

The TRAN1 bit controls the PWMx outputs driven by counter 1 and the TRAN2 bit controls the PWMx outputs driven by counter 2.

PWM generation and output compare are done by comparing these active DCRx values with the counter.

The maximum available resolution for the PWMx duty cycle is:

$$\text{Resolution} = 1 / (4096 - ATR)$$

where ATR is equal to 0. With this maximum resolution, 0% and 100% duty cycle can be obtained by changing the polarity.

At reset, the counter starts counting from 0.

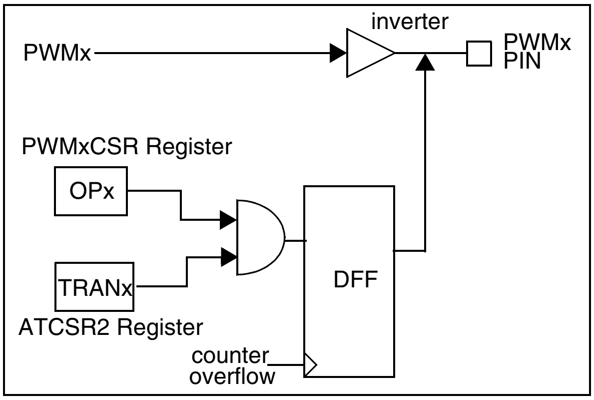
When a upcounter overflow occurs (OVF event), the preloaded Duty cycle values are transferred to the active Duty Cycle registers and the PWMx signals are set to a high level. When the upcounter matches the active DCRx value the PWMx signals are set to a low level. To obtain a signal on a PWMx pin, the contents of the corresponding active DCRx register must be greater than the contents of the ATR register.

The maximum value of ATR is 4094 because it must be lower than the DCR value which must be 4095 in this case.

Polarity Inversion

The polarity bits can be used to invert any of the four output signals. The inversion is synchronized with the counter overflow if the corresponding transfer bit in the ATCSR2 register is set (reset value). See Figure 36.

Figure 36. PWM Polarity Inversion



The Data Flip Flop (DFF) applies the polarity inversion when triggered by the counter overflow input.

Output Control

The PWMx output signals can be enabled or disabled using the OEx bits in the PWMCR register.

SERIAL PERIPHERAL INTERFACE (cont'd)

11.4.3.2 Slave Select Management

As an alternative to using the $\overline{SS}$ pin to control the Slave Select signal, the application can choose to manage the Slave Select signal by software. This is configured by the SSM bit in the SPICSR register (see Figure 51).

In software management, the external $\overline{SS}$ pin is free for other application uses and the internal $\overline{SS}$ signal level is driven by writing to the SSI bit in the SPICSR register.

In Master mode:

– $\overline{SS}$ internal must be held high continuously

In Slave Mode:

There are two cases depending on the data/clock timing relationship (see Figure 50):

If CPHA = 1 (data latched on second clock edge):

– $\overline{SS}$ internal must be held low during the entire transmission. This implies that in single slave applications the $\overline{SS}$ pin either can be tied to V_{SS}, or made free for standard I/O by managing the SS function by software (SSM = 1 and SSI = 0 in the in the SPICSR register)

If CPHA = 0 (data latched on first clock edge):

– $\overline{SS}$ internal must be held low during byte transmission and pulled high between each byte to allow the slave to write to the shift register. If $\overline{SS}$ is not pulled high, a Write Collision error will occur when the slave writes to the shift register (see Section 11.4.5.3).

Figure 50. Generic $\overline{SS}$ Timing Diagram

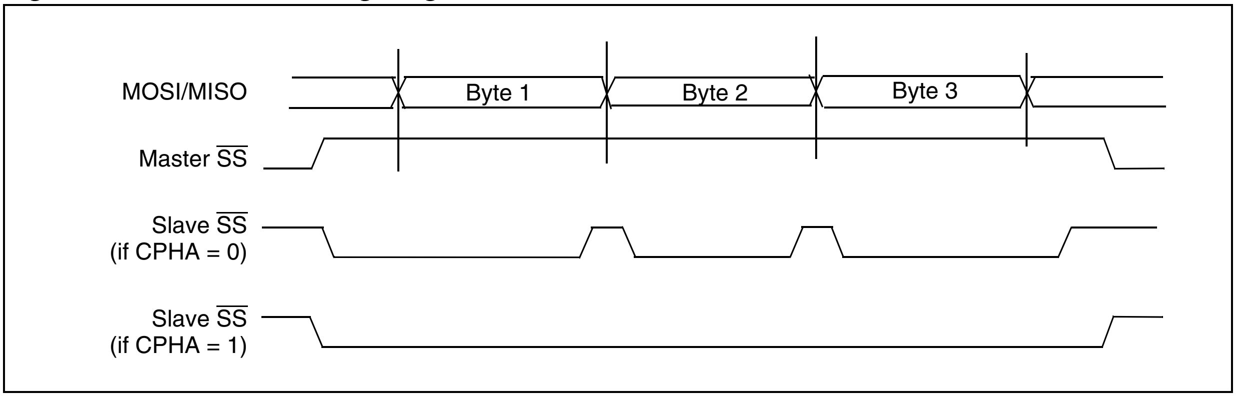
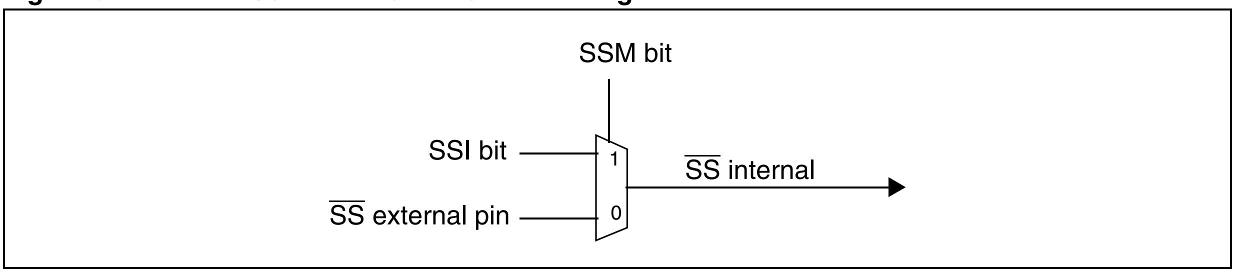


Figure 51. Hardware/Software Slave Select Management



SERIAL PERIPHERAL INTERFACE (cont'd)

11.4.5 Error Flags

11.4.5.1 Master Mode Fault (MODF)

Master mode fault occurs when the master device's $\overline{SS}$ pin is pulled low.

When a Master mode fault occurs:

– The MODF bit is set and an SPI interrupt request is generated if the SPIE bit is set.

– The SPE bit is reset. This blocks all output from the device and disables the SPI peripheral.

– The MSTR bit is reset, thus forcing the device into slave mode.

Clearing the MODF bit is done through a software sequence:

1. A read access to the SPICSR register while the MODF bit is set.

2. A write to the SPICR register.

Notes: To avoid any conflicts in an application with multiple slaves, the $\overline{SS}$ pin must be pulled high during the MODF bit clearing sequence. The SPE and MSTR bits may be restored to their original state during or after this clearing sequence.

Hardware does not allow the user to set the SPE and MSTR bits while the MODF bit is set except in the MODF bit clearing sequence.

In a slave device, the MODF bit can not be set, but in a multimaster configuration the device can be in slave mode with the MODF bit set.

The MODF bit indicates that there might have been a multimaster conflict and allows software to handle this using an interrupt routine and either perform a reset or return to an application default state.

11.4.5.2 Overrun Condition (OVR)

An overrun condition occurs when the master device has sent a data byte and the slave device has not cleared the SPIF bit issued from the previously transmitted byte.

When an Overrun occurs:

– The OVR bit is set and an interrupt request is generated if the SPIE bit is set.

In this case, the receiver buffer contains the byte sent after the SPIF bit was last cleared. A read to the SPIDR register returns this byte. All other bytes are lost.

The OVR bit is cleared by reading the SPICSR register.

11.4.5.3 Write Collision Error (WCOL)

A write collision occurs when the software tries to write to the SPIDR register while a data transfer is taking place with an external device. When this happens, the transfer continues uninterrupted and the software write will be unsuccessful.

Write collisions can occur both in master and slave mode. See also Section 11.4.3.2 Slave Select Management.

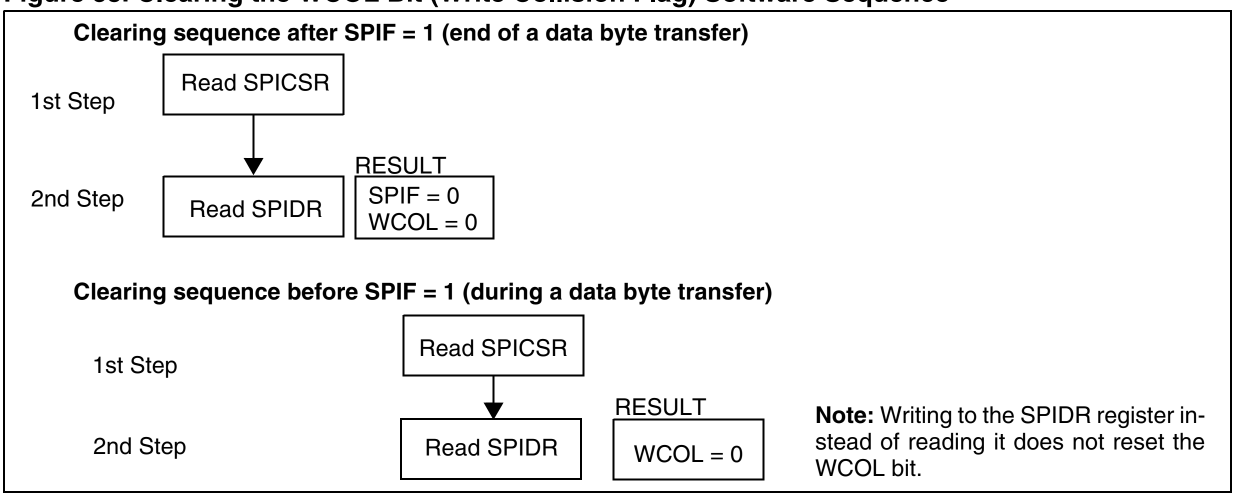
Note: A "read collision" will never occur since the received data byte is placed in a buffer in which access is always synchronous with the CPU operation.

The WCOL bit in the SPICSR register is set if a write collision occurs.

No SPI interrupt is generated when the WCOL bit is set (the WCOL bit is a status flag only).

Clearing the WCOL bit is done through a software sequence (see Figure 53).

Figure 53. Clearing the WCOL Bit (Write Collision Flag) Software Sequence



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

11.5.8 SCI Mode Register Description

STATUS REGISTER (SCISR)

Read Only

Reset Value: 1100 0000 (C0h)

7							0
TDRE	TC	RDRF	IDLE	OR1)	NF1)	FE1)	PE1)

Bit 7 = **TDRE** *Transmit data register empty*
This bit is set by hardware when the content of the TDR register has been transferred into the shift register. An interrupt is generated if the TIE = 1 in the SCICR2 register. It is cleared by a software sequence (an access to the SCISR register followed by a write to the SCIDR register).
0: Data is not transferred to the shift register
1: Data is transferred to the shift register

Bit 6 = **TC** *Transmission complete*
This bit is set by hardware when transmission of a character containing Data is complete. An interrupt is generated if TCIE = 1 in the SCICR2 register. It is cleared by a software sequence (an access to the SCISR register followed by a write to the SCIDR register).
0: Transmission is not complete
1: Transmission is complete

Note: TC is not set after the transmission of a Preamble or a Break.

Bit 5 = **RDRF** *Received data ready flag*
This bit is set by hardware when the content of the RDR register has been transferred to the SCIDR register. An interrupt is generated if RIE = 1 in the SCICR2 register. It is cleared by a software sequence (an access to the SCISR register followed by a read to the SCIDR register).
0: Data is not received
1: Received data is ready to be read

Bit 4 = **IDLE** *Idle line detected*
This bit is set by hardware when an Idle Line is detected. An interrupt is generated if the ILIE = 1 in the SCICR2 register. It is cleared by a software sequence (an access to the SCISR register followed by a read to the SCIDR register).
0: No Idle Line is detected
1: Idle Line is detected

Note: The IDLE bit will not be set again until the RDRF bit has been set itself (that is, a new idle line occurs).

Bit 3 = **OR** *Overrun error*

The OR bit is set by hardware when the word currently being received in the shift register is ready to be transferred into the RDR register whereas RDRF is still set. An interrupt is generated if RIE = 1 in the SCICR2 register. It is cleared by a software sequence (an access to the SCISR register followed by a read to the SCIDR register).
0: No Overrun error
1: Overrun error detected

Note: When this bit is set, RDR register contents will not be lost but the shift register will be overwritten.

Bit 2 = **NF** *Character Noise flag*
This bit is set by hardware when noise is detected on a received character. It is cleared by a software sequence (an access to the SCISR register followed by a read to the SCIDR register).
0: No noise
1: Noise is detected

Note: This bit does not generate interrupt as it appears at the same time as the RDRF bit which itself generates an interrupt.

Bit 1 = **FE** *Framing error*
This bit is set by hardware when a desynchronization, excessive noise or a break character is detected. It is cleared by a software sequence (an access to the SCISR register followed by a read to the SCIDR register).
0: No Framing error
1: Framing error or break character detected

Note: This bit does not generate an interrupt as it appears at the same time as the RDRF bit which itself generates an interrupt. If the word currently being transferred causes both a frame error and an overrun error, it will be transferred and only the OR bit will be set.

Bit 0 = **PE** *Parity error*
This bit is set by hardware when a byte parity error occurs (if the PCE bit is set) in receiver mode. It is cleared by a software sequence (a read to the status register followed by an access to the SCIDR data register). An interrupt is generated if PIE = 1 in the SCICR1 register.
0: No parity error
1: Parity error detected

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

CONTROL REGISTER 2 (SCICR2)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
TIE	TCIE	RIE	ILIE	TE	RE	RWU[1)]	SBK[1)]

[1)]This bit has a different function in LIN mode, please refer to the LIN mode register description.

Bit 7 = **TIE** *Transmitter interrupt enable*
This bit is set and cleared by software.
0: Interrupt is inhibited
1: In SCI interrupt is generated whenever TDRE = 1 in the SCISR register

Bit 6 = **TCIE** *Transmission complete interrupt enable*
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An SCI interrupt is generated whenever TC = 1 in the SCISR register

Bit 5 = **RIE** *Receiver interrupt enable*
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An SCI interrupt is generated whenever OR = 1 or RDRF = 1 in the SCISR register

Bit 4 = **ILIE** *Idle line interrupt enable*
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An SCI interrupt is generated whenever IDLE = 1 in the SCISR register.

Bit 3 = **TE** *Transmitter enable*
This bit enables the transmitter. It is set and cleared by software.
0: Transmitter is disabled
1: Transmitter is enabled

Notes:

– During transmission, a “0” pulse on the TE bit (“0” followed by “1”) sends a preamble (idle line) after the current word.

– When TE is set there is a 1 bit-time delay before the transmission starts.

Bit 2 = **RE** *Receiver enable*
This bit enables the receiver. It is set and cleared by software.
0: Receiver is disabled in the SCISR register
1: Receiver is enabled and begins searching for a start bit

Bit 1 = **RWU** *Receiver wake-up*
This bit determines if the SCI is in mute mode or not. It is set and cleared by software and can be cleared by hardware when a wake-up sequence is recognized.
0: Receiver in active mode
1: Receiver in mute mode

Notes:

– Before selecting Mute mode (by setting the RWU bit) the SCI must first receive a data byte, otherwise it cannot function in Mute mode with wake-up by Idle line detection.

– In Address Mark Detection Wake-Up configuration (WAKE bit = 1) the RWU bit cannot be modified by software while the RDRF bit is set.

Bit 0 = **SBK** *Send break*
This bit set is used to send break characters. It is set and cleared by software.
0: No break character is transmitted
1: Break characters are transmitted

Note: If the SBK bit is set to “1” and then to “0”, the transmitter will send a BREAK word at the end of the current word.

DATA REGISTER (SCIDR)

Read/Write

Reset Value: Undefined

Contains the Received or Transmitted data character, depending on whether it is read from or written to.

7							0
DR7	DR6	DR5	DR4	DR3	DR2	DR1	DR0

The Data register performs a double function (read and write) since it is composed of two registers, one for transmission (TDR) and one for reception (RDR).
The TDR register provides the parallel interface between the internal bus and the output shift register (see Figure 55).
The RDR register provides the parallel interface between the input shift register and the internal bus (see Figure 55).

12 INSTRUCTION SET

12.1 ST7 ADDRESSING MODES

The ST7 Core features 17 different addressing modes which can be classified in seven main groups:

Addressing Mode	Example
Inherent	nop
Immediate	ld A,#$55
Direct	ld A,$55
Indexed	ld A,($55,X)
Indirect	ld A,([$55],X)
Relative	jrne loop
Bit operation	bset byte,#5

The ST7 Instruction set is designed to minimize the number of bytes required per instruction: To do so, most of the addressing modes may be subdivided in two submodes called long and short:

– Long addressing mode is more powerful because it can use the full 64 Kbyte address space, however it uses more bytes and more CPU cycles.

– Short addressing mode is less powerful because it can generally only access page zero (0000h - 00FFh range), but the instruction size is more compact, and faster. All memory to memory instructions use short addressing modes only (CLR, CPL, NEG, BSET, BRES, BTJT, BTJF, INC, DEC, RLC, RRC, SLL, SRL, SRA, SWAP)

The ST7 Assembler optimizes the use of long and short addressing modes.

Table 23. ST7 Addressing Mode Overview

Mode			Syntax	Destination/ Source	Pointer Address (Hex.)	Pointer Size (Hex.)	Length (Bytes)
Inherent			nop				+ 0
Immediate			ld A,#$55				+ 1
Short	Direct		ld A,$10	00..FF			+ 1
Long	Direct		ld A,$1000	0000..FFFF			+ 2
No Offset	Direct	Indexed	ld A,(X)	00..FF			+ 0 (with X register) + 1 (with Y register)
Short	Direct	Indexed	ld A,($10,X)	00..1FE			+ 1
Long	Direct	Indexed	ld A,($1000,X)	0000..FFFF			+ 2
Short	Indirect		ld A,[$10]	00..FF	00..FF	byte	+ 2
Long	Indirect		ld A,[$10.w]	0000..FFFF	00..FF	word	+ 2
Short	Indirect	Indexed	ld A,([$10],X)	00..1FE	00..FF	byte	+ 2
Long	Indirect	Indexed	ld A,([$10.w],X)	0000..FFFF	00..FF	word	+ 2
Relative	Direct		jrne loop	PC-128/PC+127[1)]			+ 1
Relative	Indirect		jrne [$10]	PC-128/PC+127[1)]	00..FF	byte	+ 2
Bit	Direct		bset $10,#7	00..FF			+ 1
Bit	Indirect		bset [$10],#7	00..FF	00..FF	byte	+ 2
Bit	Direct	Relative	btjt $10,#7,skip	00..FF			+ 2
Bit	Indirect	Relative	btjt [$10],#7,skip	00..FF	00..FF	byte	+ 3

Note:
1. At the time the instruction is executed, the Program Counter (PC) points to the instruction following JRxx.

INSTRUCTION GROUPS (cont'd)

Mnemo	Description	Function/Example	Dst	Src	H	I	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					0			
IRET	Interrupt routine return	Pop CC, A, X, PC			H	I	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. interrupt = 1								
JRIL	Jump if ext. interrupt = 0								
JRH	Jump if H = 1	H = 1 ?							
JRNH	Jump if H = 0	H = 0 ?							
JRM	Jump if I = 1	I = 1 ?							
JRNM	Jump if I = 0	I = 0 ?							
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 >							

13.6 MEMORY CHARACTERISTICS

T_A = -40°C to 125°C, unless otherwise specified

13.6.1 RAM and Hardware Registers

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{RM}	Data retention mode [1]	HALT mode (or RESET)	1.6			V

13.6.2 FLASH Program Memory

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{DD}	Operating voltage for Flash write/erase	Refer to operating range of V_{DD} with T_A, section 13.3.1 on page 133	2.7		5.5	V
t_{prog}	Programming time for 1~32 bytes [2]	T_A=–40 to +125°C		5	10	ms
	Programming time for 1.5 kBytes	T_A=+25°C		0.24	0.48	s
t_{RET}	Data retention [4]	T_A=+55°C [3]	20			years
N_{RW}	Write erase cycles	T_A=+25°C	10K			cycles
I_{DD}	Supply current	Read / Write / Erase modes f_{CPU} = 8MHz, V_{DD} = 5.5V			2.6 [6]	mA
		No Read/No Write Mode			100	µA
		Power down mode / HALT		0	0.1	µA

13.6.3 EEPROM Data Memory

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{DD}	Operating voltage for EEPROM write/erase	Refer to operating range of V_{DD} with T_A, section 13.3.1 on page 133	2.7		5.5	V
t_{prog}	Programming time for 1~32 bytes	T_A=–40 to +125°C		5	10	ms
t_{ret}	Data retention [4]	T_A=+55°C [3]	20			years
N_{RW}	Write erase cycles	T_A=+25°C	300K			cycles

Notes:

1. Minimum V_{DD} supply voltage without losing data stored in RAM (in HALT mode or under RESET) or in hardware registers (only in HALT mode). Guaranteed by construction, not tested in production.

2. Up to 32 bytes can be programmed at a time.

3. The data retention time increases when the T_A decreases.

4. Data based on reliability test results and monitored in production.

5. Data based on characterization results, not tested in production.

6. Guaranteed by Design. Not tested in production.

OPTION BYTES (Cont'd)

	OPTION BYTE 0								OPTION BYTE 1							
	7							0	7							0
	AWUCK	OSCRANGE 2:0			SEC1	SEC0	FMPR	FMPW	PLL x4x8	PLL OFF	Res.	OSC	LVD 1:0		WDG SW	WDG HALT
Default Value	1	1	1	1	1	1	0	0	1	1	1	0	1	1	1	1

OPTION BYTE 1

OPT 7 = **PLLx4x8** *PLL Factor Selection*.
0: PLLx4
1: PLLx8

OPT 6 = **PLLOFF** *PLL Disable*
This option bit enables or disables the PLL.
0: PLL enabled
1: PLL disabled (bypassed)

OPT 5 = Reserved. Must always be set to 1.

OPT 4 = **OSC** *RC Oscillator Selection*
This option bit enables to select the internal RC Oscillator.
0: RC Oscillator on
1: RC Oscillator off

Notes:
- RC oscillator available on ST7LITE35 and ST7LITE39 devices only
- If the RC oscillator is selected, then to improve clock stability and frequency accuracy, it is recommended to place a decoupling capacitor, typically 100nF, between the V_{DD} and V_{SS} pins as close as possible to the ST7 device.

OPT 3:2 = **LVD[1:0]** *Low Voltage Selection*
These option bits enable the voltage detection block (LVD and AVD) with a selected threshold to the LVD and AVD.

Configuration	VD1	VD0
LVD Off	1	1
Highest Voltage Threshold	1	0
Medium Voltage Threshold	0	1
Lowest Voltage Threshold	0	0

OPT 1 = **WDGSW** *Hardware or Software Watchdog*
0: Hardware (watchdog always enabled)
1: Software (watchdog to be enabled by software)

OPT 0 = **WDG HALT** *Watchdog Reset on Halt*
0: No reset generation when entering HALT mode
1: Reset generation when entering HALT mode

15.4 ST7 APPLICATION NOTES

Table 27. 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

IMPORTANT NOTES (Cont'd)

Impact on application

Software may execute the interrupt routine twice after header reception.

Moreover, in reception mode, as the receiver is no longer in mute mode, an interrupt will be generated on each data byte reception.

Workaround

The problem can be detected in the LINSCI interrupt routine. In case of timeout error (LHE is set and LHLR is loaded with 00h), the software can check the RWU bit in the SCICR2 register. If RWU is cleared, it can be set by software. Refer to Figure 110. Workaround is shown in bold characters.

Figure 110. LINSCI Interrupt routine

```
@interrupt void LINSCI_IT ( void ) /* LINSCI interrupt routine */
{
     /* clear flags */
     SCISR_buffer = SCISR;
     SCIDR_buffer = SCIDR;

     if ( SCISR_buffer & LHE )/* header error ? */
     {
          if (!LHLR)/* header time-out? */
          {
               if ( !(SCICR2 & RWU) )/* active mode ? */
               {
                     _asm("sim");/* disable interrupts */
                     SCISR;
                     SCIDR;/* Clear RDRF flag */
                     SCICR2 |= RWU;/* set mute mode */
                     SCISR;
                     SCIDR;/* Clear RDRF flag */
                     SCICR2 |= RWU;/* set mute mode */
                     _asm("rim");/* enable interrupts */
               }
          }
     }
}
```

Example using Cosmic compiler syntax