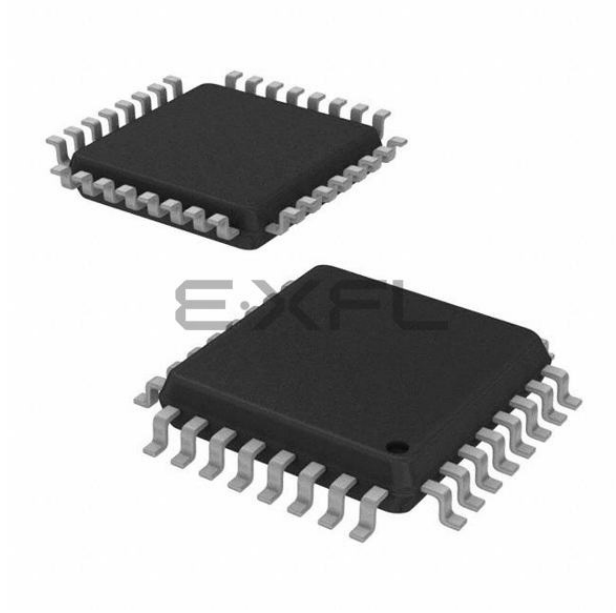


Welcome to E-XFL.COM

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	32KB (32K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	3.8V ~ 5.5V
Data Converters	A/D 12x10b
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/st72f324bk6tae

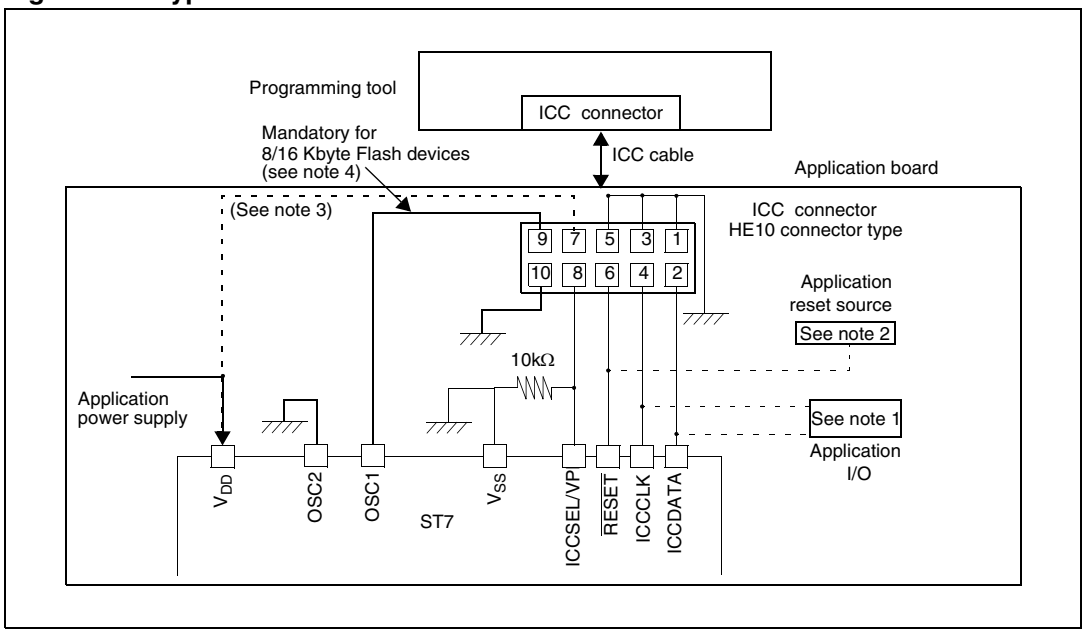
Refer to *Section 9: I/O ports on page 58* for more details on the software configuration of the I/O ports.

The reset configuration of each pin is shown in bold. This configuration is valid as long as the device is in reset state.

Table 2. Device pin description

Pin			Type	Level		Port						Main function (after reset)	Alternate function	
No.		Name		Input	Output	Input				Output				
LQFP44	LQFP32					float	wpu	int	ana	OD	PP			
6	30	PB4 (HS)	I/O	C_T	HS	**X**	ei3			X	X	Port B4		
7	31	PD0/AIN0	I/O	C_T		**X**	X		X	X	X	Port D0	ADC analog input 0	
8	32	PD1/AIN1	I/O	C_T		**X**	X		X	X	X	Port D1	ADC analog input 1	
9	-	PD2/AIN2	I/O	C_T		**X**	X		X	X	X	Port D2	ADC analog input 2	
10	-	PD3/AIN3	I/O	C_T		**X**	X		X	X	X	Port D3	ADC analog input 3	
11	-	PD4/AIN4	I/O	C_T		**X**	X		X	X	X	Port D4	ADC analog input 4	
12	-	PD5/AIN5	I/O	C_T		**X**	X		X	X	X	Port D5	ADC analog input 5	
13	1	V_{AREF}(1)	S									Analog reference voltage for ADC		
14	2	V_{SSA}(1)	S									Analog ground voltage		
15	3	PF0/MCO/AIN8	I/O	C_T		**X**	ei1		X	X	X	Port F0	Main clock out (f_{CPU})	ADC analog input 8
16	4	PF1 (HS)/BEEP	I/O	C_T	HS	**X**	ei1			X	X	Port F1	Beep signal output	
17	-	PF2 (HS)	I/O	C_T	HS	**X**	ei1			X	X	Port F2		
18	5	PF4/OCMP1_A /AIN10	I/O	C_T		**X**	X		X	X	X	Port F4	Timer A output compare 1	ADC analog Input 10
19	6	PF6 (HS)/ICAP1_A	I/O	C_T	HS	**X**	X			X	X	Port F6	Timer A input capture 1	
20	7	PF7 (HS)/EXTCLK_A	I/O	C_T	HS	**X**	X			X	X	Port F7	Timer A external clock source	
21	-	V_{DD_0}(1)	S									Digital main supply voltage		
22	-	V_{SS_0}(1)	S									Digital ground voltage		
23	8	PC0/OCMP2_B /AIN12	I/O	C_T		**X**	X		X	X	X	Port C0	Timer B output compare 2	ADC analog input 12
24	9	PC1/OCMP1_B /AIN13	I/O	C_T		**X**	X		X	X	X	Port C1	Timer B output compare 1	ADC analog input 13
25	10	PC2 (HS)/ICAP2_B	I/O	C_T	HS	**X**	X			X	X	Port C2	Timer B input capture 2	

Figure 6. Typical ICC interface



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 be 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 (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.

Caution: External clock ICC entry mode is mandatory in ST72F324B 8/16 Kbyte Flash devices. In this case pin 9 must be connected to the OSC1 (OSCIN) pin of the ST7 and OSC2 must be grounded. 32 Kbyte Flash devices may use external clock or application clock ICC entry mode.

4.5 ICP (in-circuit programming)

To perform ICP the microcontroller must be switched to ICC (in-circuit communication) mode by an external controller or programming tool.

Depending on the ICP code downloaded in RAM, Flash memory programming can be fully customized (number of bytes to program, program locations, or selection serial communication interface for downloading).

When using an STMicroelectronics or third-party programming tool that supports ICP and the specific microcontroller device, the user needs only to implement the ICP hardware interface on the application board (see *Figure 6*). For more details on the pin locations, refer to the device pinout description.

5 Central processing unit (CPU)

5.1 Introduction

This CPU has a full 8-bit architecture and contains six internal registers allowing efficient 8-bit data manipulation.

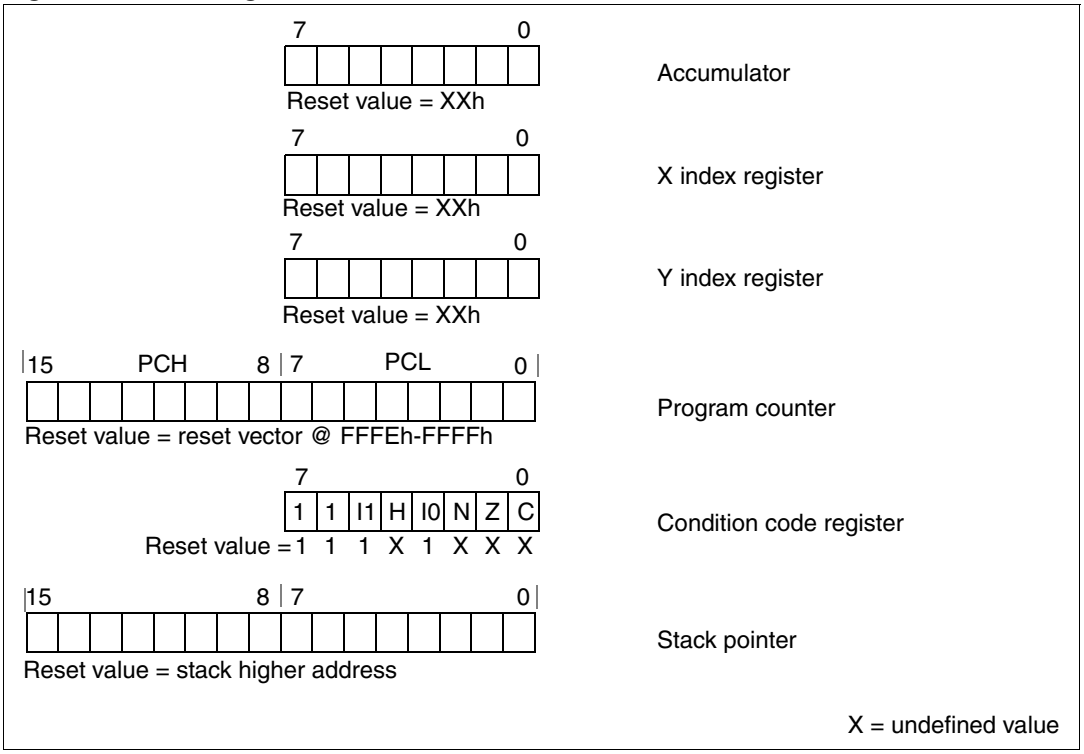
5.2 Main features

- Enable executing 63 basic instructions
- Fast 8-bit by 8-bit multiply
- 17 main addressing modes (with indirect addressing mode)
- Two 8-bit index registers
- 16-bit stack pointer
- Low power Halt and Wait modes
- Priority maskable hardware interrupts
- Non-maskable software/hardware interrupts

5.3 CPU registers

The six CPU registers shown in *Figure 7* are not present in the memory mapping and are accessed by specific instructions.

Figure 7. CPU registers



They can be also set/cleared by software with the RIM, SIM, HALT, WFI, IRET and PUSH/POP instructions (see *Table 18: Dedicated interrupt instruction set*).

7.5.2 Interrupt software priority registers (ISPRx)

ISPRx Reset value: 1111 1111 (FFh)

	7	6	5	4	3	2	1	0
ISPR0	I1_3	I0_3	I1_2	I0_2	I1_1	I0_1	I1_0	I0_0
ISPR1	I1_7	I0_7	I1_6	I0_6	I1_5	I0_5	I1_4	I0_4
ISPR2	I1_11	I0_11	I1_10	I0_10	I1_9	I0_9	I1_8	I0_8
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ISPR3	1	1	1	1	I1_13	I0_13	I1_12	I0_12
	RO	RO	RO	RO	R/W	R/W	R/W	R/W

These four registers contain the interrupt software priority of each interrupt vector.

- Each interrupt vector (except reset and TRAP) has corresponding bits in these registers where its own software priority is stored. This correspondence is shown in the following *Table 17*.

Table 17. ISPRx interrupt vector correspondence

Vector address	ISPRx bits
FFFBh-FFFAh	I1_0 and I0_0 bits
FFF9h-FFF8h	I1_1 and I0_1 bits
...	...
FFE1h-FFE0h	I1_13 and I0_13 bits

- Each I1_x and I0_x bit value in the ISPRx registers has the same meaning as the I1 and I0 bits in the CC register.
- Level 0 cannot be written (I1_x = 1, I0_x = 0). In this case, the previously stored value is kept (for example, previous value = CFh, write = 64h, result = 44h).

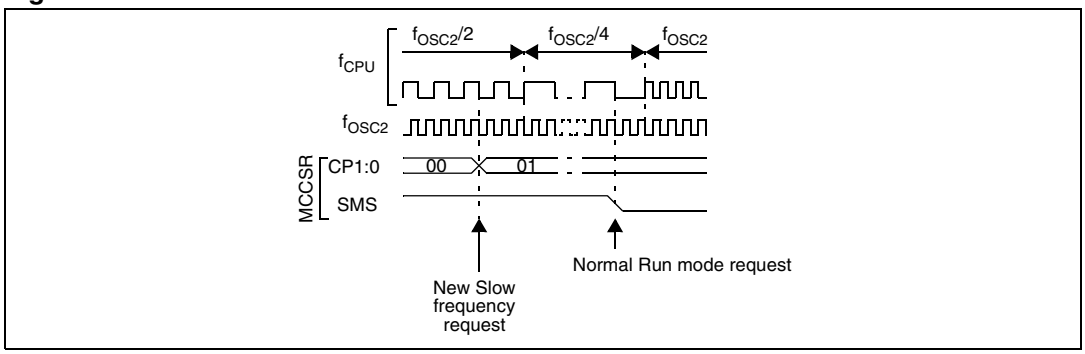
The reset, and TRAP vectors have no software priorities. When one is serviced, the I1 and I0 bits of the CC register are both set.

Caution: If the I1_x and I0_x bits are modified while the interrupt x is executed the following behavior has to be considered: If the interrupt x is still pending (new interrupt or flag not cleared) and the new software priority is higher than the previous one, the interrupt x is re-entered. Otherwise, the software priority stays unchanged up to the next interrupt request (after the IRET of the interrupt x).

Table 18. Dedicated interrupt instruction set(1)

Instruction	New description	Function/example	I1	H	I0	N	Z	C
HALT	Entering HALT mode		1		0			

Figure 22. Slow mode clock transitions



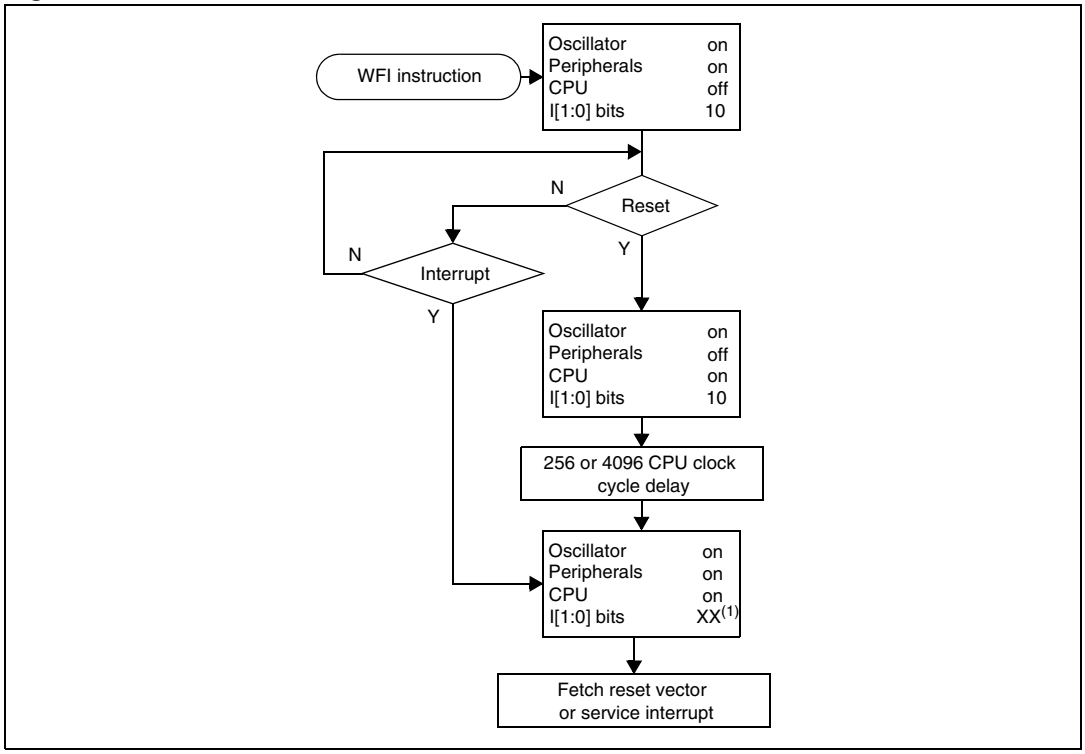
8.3 Wait mode

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 23*.

Figure 23. Wait mode flowchart



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.

10 On-chip peripherals

10.1 Watchdog timer (WDG)

10.1.1 Introduction

The Watchdog timer is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence. The Watchdog circuit generates an MCU reset on expiry of a programmed time period, unless the program refreshes the counter's contents before the T6 bit becomes cleared.

10.1.2 Main features

- Programmable free-running downcounter
- Programmable reset
- Reset (if Watchdog activated) when the T6 bit reaches zero
- Optional reset on HALT instruction (configurable by option byte)
- Hardware Watchdog selectable by option byte

10.1.3 Functional description

The counter value stored in the Watchdog Control register (WDGCR bits T[6:0]), is decremented every 16384 f_{OSC2} cycles (approx.), and the length of the timeout period can be programmed by the user in 64 increments.

If the watchdog is activated (the WDGA bit is set) and when the 7-bit timer (bits T[6:0]) rolls over from 40h to 3Fh (T6 becomes cleared), it initiates a reset cycle pulling low the reset pin for typically 30µs.

The application program must write in the WDGCR register at regular intervals during normal operation to prevent an MCU reset. This downcounter is free-running: it counts down even if the watchdog is disabled. The value to be stored in the WDGCR register must be between FFh and C0h:

- The WDGA bit is set (Watchdog enabled)
- The T6 bit is set to prevent generating an immediate reset
- The T[5:0] bits contain the number of increments which represents the time delay before the Watchdog produces a reset (see *Figure 31: Approximate timeout duration*). The timing varies between a minimum and a maximum value due to the unknown status of the prescaler when writing to the WDGCR register (see *Figure 32*).

Following a reset, the watchdog is disabled. Once activated it cannot be disabled, except by a reset.

The T6 bit can be used to generate a software reset (the WDGA bit is set and the T6 bit is cleared).

If the Watchdog is activated, the HALT instruction generates a reset.

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

$$\Delta \text{ OC}i\text{R} = \Delta t * f_{EXT}$$

Where:

Δt = Output compare period (in seconds)
f_{EXT} = External timer clock frequency (in hertz)

Clearing the output compare interrupt request (that is, clearing the OCF*i* bit) is done by:

1. Reading the SR register while the OCF*i* bit is set.
2. An access (read or write) to the OC*i*LR register.

The following procedure is recommended to prevent the OCF*i* bit from being set between the time it is read and the write to the OC*i*R register:

- Write to the OC*i*HR register (further compares are inhibited).
- Read the SR register (first step of the clearance of the OCF*i* bit, which may be already set).
- Write to the OC*i*LR register (enables the output compare function and clears the OCF*i* bit).

Note: 1 After a processor write cycle to the OCiHR register, the output compare function is inhibited until the OCiLR register is also written.

2 If the OCiE bit is not set, the OCMPi pin is a general I/O port and the OLVLi bit will not appear when a match is found but an interrupt could be generated if the OCIE bit is set.

3 In both internal and external clock modes, OCFi and OCMPi are set while the counter value equals the OCiR register value (see Figure 42 on page 83 for an example with $f_{CPU}/2$ and Figure 43 on page 83 for an example with $f_{CPU}/4$). This behavior is the same in OPM or PWM mode.

4 The output compare functions can be used both for generating external events on the OCMPi pins even if the input capture mode is also used.

5 The value in the 16-bit OCiR register and the OLVi bit should be changed after each successful comparison in order to control an output waveform or establish a new elapsed timeout.

Forced output compare capability

When the FOLV*i* bit is set by software, the OLVL*i* bit is copied to the OCMP*i* pin. The OLV*i* bit has to be toggled in order to toggle the OCMP*i* pin when it is enabled (OC*i*E bit = 1). The OCF*i* bit is then not set by hardware, and thus no interrupt request is generated.

The FOLVL*i* bits have no effect in both one pulse mode and PWM mode.

Collision error will occur when the slave writes to the shift register (see *Write collision error (WCOL) on page 103*).

Figure 50. Generic $\overline{SS}$ timing diagram

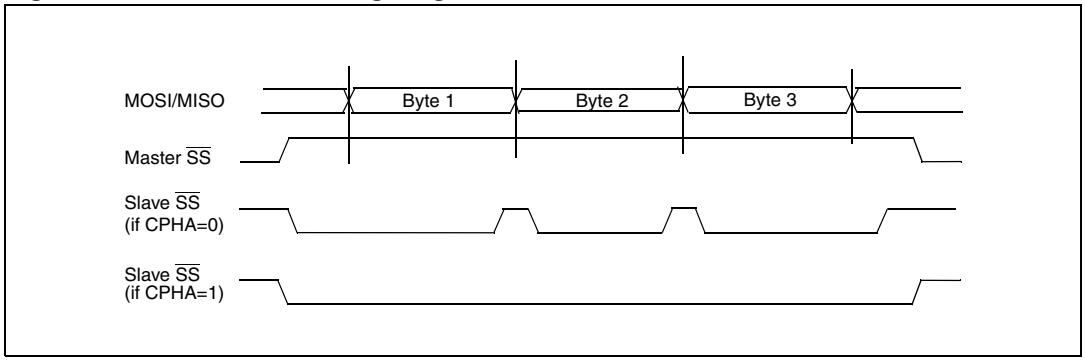
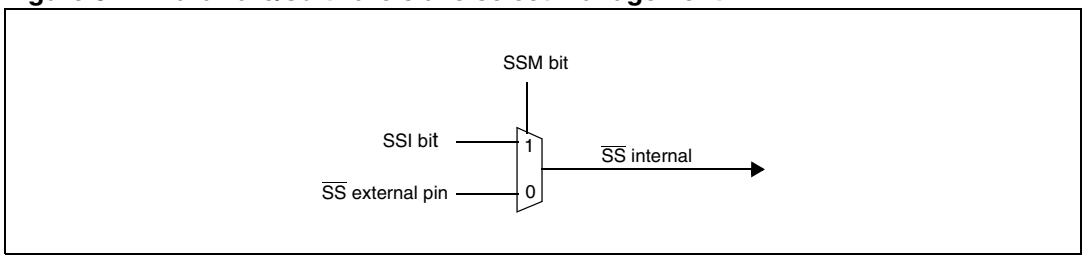


Figure 51. Hardware/software slave select management



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).

How to operate the SPI in master mode

To operate the SPI in master mode, perform the following steps in order:

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 52* 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 $\overline{SS}$ 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 $\overline{SS}$ is high.

Caution: If the SPICSR register is not written first, the SPICR register setting (MSTR bit) might not be taken into account.

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

10.4.5 Error flags

Master mode fault (MODF)

Master mode fault occurs when the master device has its $\overline{SS}$ pin 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.

Note: 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.

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.

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 is unsuccessful.

Write collisions can occur both in master and slave mode. See also *Slave Select management on page 99*.

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 MCU 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).

A software sequence clears the WCOL bit (see *Figure 53*).

10.4.6 Low power modes

Table 53. Effect of low power modes on SPI

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 MCU 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.

Using the SPI to wake up the MCU from Halt mode

In slave configuration, the SPI is able to wake up the ST7 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 ST7 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 ST7 enters Halt mode. Therefore, if Slave selection is configured as external (see *Slave Select management on page 99*), make sure the master drives a low level on the $\overline{SS}$ pin when the slave enters Halt mode.

10.4.7 Interrupts

Table 54. SPI interrupt control/wake-up capability(1)

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			

1. The SPI interrupt events are connected to the same interrupt vector (see *Section 7: Interrupts*). They generate an interrupt if the corresponding Enable Control Bit is set and the interrupt mask in the CC register is reset (RIM instruction).

10.5 Serial communications interface (SCI)

10.5.1 Introduction

The serial communications interface (SCI) offers a flexible means of full-duplex data exchange with external equipment requiring an industry standard NRZ asynchronous serial data format. The SCI offers a very wide range of baud rates using two baud rate generator systems.

10.5.2 Main features

- Full duplex, asynchronous communications
- NRZ standard format (mark/space)
- Dual baud rate generator systems
- Independently programmable transmit and receive baud rates up to 500K baud.
- Programmable data word length (8 or 9 bits)
- Receive buffer full, Transmit buffer empty and End of Transmission flags
- 2 receiver wake-up modes
  - Address bit (MSB)
  - Idle line
- Muting function for multiprocessor configurations
- Separate enable bits for Transmitter and Receiver
- 4 error detection flags
  - Overrun error
  - Noise error
  - Frame error
  - Parity error
- 5 interrupt sources with flags
  - Transmit data register empty
  - Transmission complete
  - Receive data register full
  - Idle line received
  - Overrun error detected
- Parity control
  - Transmits parity bit
  - Checks parity of received data byte
- Reduced power consumption mode

Table 63. SCICR1 register description (continued)

Bit	Name	Function
3	WAKE	Wake-Up method This bit determines the SCI Wake-Up method, it is set or cleared by software. 0: Idle line 1: Address mark
2	PCE	Parity Control Enable This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission). 0: Parity control disabled 1: Parity control enabled
1	PS	Parity Selection This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity will be selected after the current byte. 0: Even parity 1: Odd parity
0	PIE	Parity Interrupt Enable This bit enables the interrupt capability of the hardware parity control when a parity error is detected (PE bit set). It is set and cleared by software. 0: Parity error interrupt disabled 1: Parity error interrupt enabled

SCI Control Register 2 (SCICR2)

SCICR2 Reset value: 0000 0000 (00h)

7	6	5	4	3	2	1	0
TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 64. SCICR2 register description

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

11 Instruction set

11.1 CPU addressing modes

The CPU features 17 different addressing modes which can be classified in 7 main groups (see *Table 75*).

Table 75. Addressing mode 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 CPU Instruction Set is designed to minimize the number of bytes required per instruction: To do so, most of the addressing modes may be divided 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 76. CPU addressing mode overview

Mode			Syntax	Destination	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
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

12.6 Clock and timing characteristics

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

12.6.1 General timings

Table 93. General timings

Symbol	Parameter	Conditions	Min	Typ(1)	Max	Unit
$t_{c(INST)}$	Instruction cycle time		2	3	12	t_{CPU}
		f_{CPU} = 8 MHz	250	375	1500	ns
$t_{v(IT)}$	Interrupt reaction time $t_{v(IT)} = \Delta t_{c(INST)} + 10$(2)		10		22	t_{CPU}
		f_{CPU} = 8 MHz	1.25		2.75	µs

1. Data based on typical application software.
2. Time measured between interrupt event and interrupt vector fetch. $\Delta t_{c(INST)}$ is the number of t_{CPU} cycles needed to finish the current instruction execution.

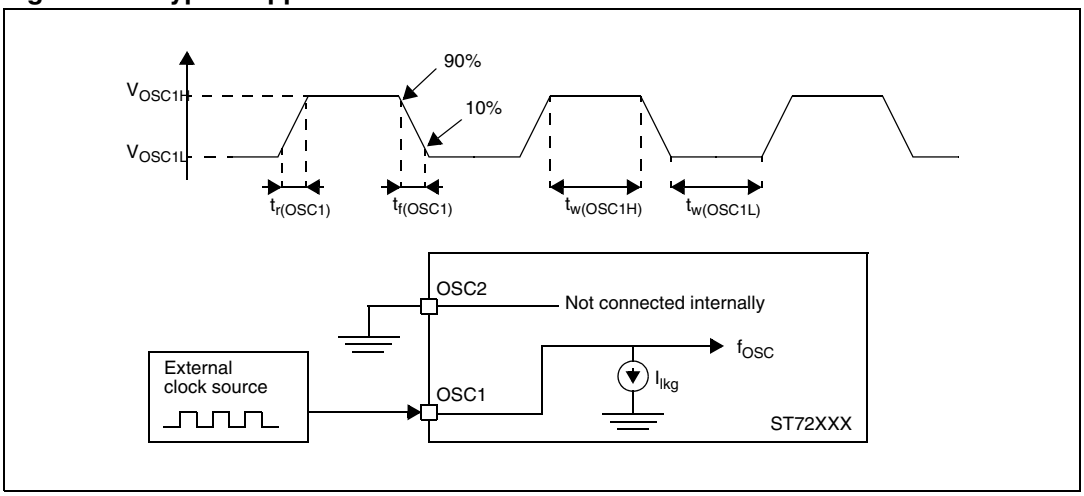
12.6.2 External clock source

Table 94. External clock source

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{OSC1H}	OSC1 input pin high level voltage	See *Figure 63.*	V_{DD}-1		V_{DD}	V
V_{OSC1L}	OSC1 input pin low level voltage		V_{SS}		V_{SS}+1	
$t_{w(OSC1H)}$ $t_{w(OSC1L)}$	OSC1 high or low time(1)		5			ns
$t_{r(OSC1)}$ $t_{f(OSC1)}$	OSC1 rise or fall time(1)				15	
I_{lkg}	OSC1 input leakage current	$V_{SS} \le V_{IN} \le V_{DD}$			±1	µA

1. Data based on design simulation and/or technology characteristics, not tested in production.

Figure 63. Typical application with an external clock source



Note: To reduce disturbance to the RC oscillator, it is recommended to place decoupling capacitors between V_{DD} and V_{SS} as shown in Figure 85 on page 173.

12.6.5 PLL characteristics

Table 98. PLL characteristics

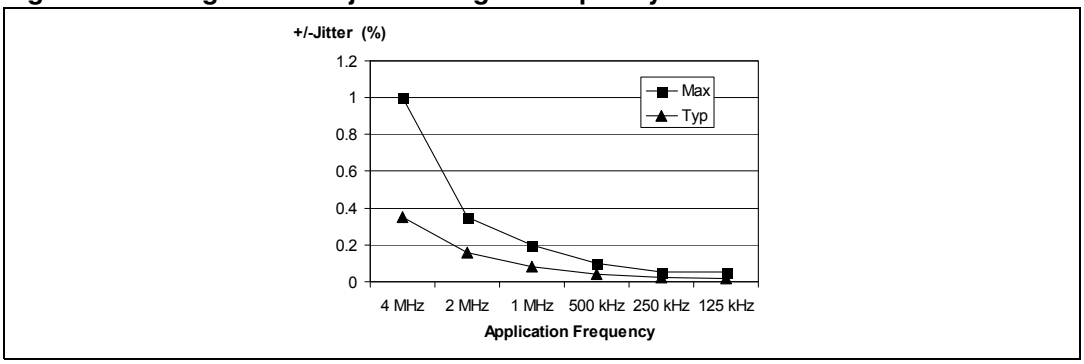
Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{OSC}	PLL input frequency range		2		4	MHz
$\Delta f_{CPU}/f_{CPU}$	Instantaneous PLL jitter(1)	f_{OSC} = 4 MHz		0.7	2	%

1. Data characterized but not tested

The user must take the PLL jitter into account in the application (for example in serial communication or sampling of high frequency signals). The PLL jitter is a periodic effect, which is integrated over several CPU cycles. Therefore the longer the period of the application signal, the less it will be impacted by the PLL jitter.

Figure 67 shows the PLL jitter integrated on application signals in the range 125 kHz to 2 MHz. At frequencies of less than 125 kHz, the jitter is negligible.

Figure 67. Integrated PLL jitter vs signal frequency(1)



1. Measurement conditions: f_{CPU} = 8 MHz

12.7 Memory characteristics

12.7.1 RAM and hardware registers

Table 99. RAM and hardware registers

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{RM}	Data retention mode(1)	Halt mode (or reset)	1.6			V

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). Not tested in production.

2. Not tested in production.

12.8.3 Absolute maximum ratings (electrical sensitivity)

Based on three different tests (ESD, LU and DLU) using specific measurement methods, the product is stressed in order to determine its performance in terms of electrical sensitivity. For more details, refer to the application note AN1181.

Electrostatic discharge (ESD)

Electrostatic discharges (a positive then a negative pulse separated by 1 second) are applied to the pins of each sample according to each pin combination. The sample size depends on the number of supply pins in the device (3 parts*(n+1) supply pin). Two models can be simulated for standard microcontrollers: Human Body Model and Charged Device Model. These tests conform to standards JESD22-A114 and JESD22-C101. There is an additional model for automotive microcontrollers: Machine Model, JESD22-A115.

Table 103. Absolute maximum ratings

Symbol	Ratings	Conditions	Maximum value(1)	Unit
$V_{ESD(HBM)}$	Electrostatic discharge voltage (human body model)	T_A = +25°C	2000	V
$V_{ESD(MM)}$	Electrostatic discharge voltage (machine model)		200	
$V_{ESD(CDM)}$	Electrostatic discharge voltage (charged device model)		750	

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

Static and dynamic Latch-Up

- **LU**: 3 complementary static tests are required on 10 parts to assess the latch-up performance. A supply overvoltage (applied to each power supply pin) and a current injection (applied to each input, output and configurable I/O pin) are performed on each sample. This test conforms to the EIA/JESD 78 IC latch-up standard. For more details, refer to the application note AN1181.
- **DLU**: Electrostatic discharges (one positive then one negative test) are applied to each pin of three samples when the micro is running to assess the latch-up performance in dynamic mode. Power supplies are set to the typical values, the oscillator is connected as near as possible to the pins of the micro and the component is put in reset mode. This test conforms to the IEC1000-4-2 and SAEJ1752/3 standards. For more details, refer to the application note AN1181.

14.3 Development tools

14.3.1 Introduction

Development tools for the ST7 microcontrollers include a complete range of hardware systems and software tools from STMicroelectronics and third-party tool suppliers. The range of tools includes solutions to help you evaluate microcontroller peripherals, develop and debug your application, and program your microcontrollers.

14.3.2 Evaluation tools and starter kits

ST offers complete, affordable **starter kits** and full-featured **evaluation boards** that allow you to evaluate microcontroller features and quickly start developing ST7 applications. Starter kits are complete, affordable hardware/software tool packages that include features and samples to help you quickly start developing your application. ST evaluation boards are open-design, embedded systems, which are developed and documented to serve as references for your application design. They include sample application software to help you demonstrate, learn about and implement your ST7's features.

14.3.3 Development and debugging tools

Application development for ST7 is supported by fully optimizing **C Compilers** and the **ST7 Assembler-Linker** toolchain, which are all seamlessly integrated in the ST7 integrated development environments in order to facilitate the debugging and fine-tuning of your application. The Cosmic C Compiler is available in a free version that outputs up to 16 Kbytes of code.

The range of hardware tools includes cost effective ST7-DVP3 series emulators. These tools are supported by the ST7 Toolset from STMicroelectronics, which includes the STVD7 integrated development environment (IDE) with high-level language debugger, editor, project manager and integrated programming interface.

14.3.4 Programming tools

During the development cycle, the ST7-DVP3 and ST7-EMU3 series emulators and the RLink provide in-circuit programming capability for programming the Flash microcontroller on your application board.

ST also provides dedicated a low-cost dedicated in-circuit programmer, the ST7-STICK, as well as ST7 socket boards which provide all the sockets required for programming any of the devices in a specific ST7 subfamily on a platform that can be used with any tool with in-circuit programming capability for ST7.

For production programming of ST7 devices, ST's third-party tool partners also provide a complete range of gang and automated programming solutions, which are ready to integrate into your production environment.

For additional ordering codes for spare parts, accessories and tools available for the ST7 (including from third party manufacturers), refer to the online product selector at www.st.com/mcu.

Table 120. STMicroelectronics development tools

Supported products	Emulation				Programming
	ST7 DVP3 series		ST7 EMU3 series		ICC socket board
	Emulator	Connection kit	Emulator	Active probe and TEB	
ST72324BJ, ST72F324BJ	ST7MDT20-DVP3	ST7MDT20-T44/DVP	ST7MDT20J-EMU3	ST7MDT20J-TEB	ST7SB20J/xx(1)
ST72324BK, ST72F324BK		ST7MDT20-T32/DVP			

1. Add suffix /EU, /UK, /US for the power supply of your region.

14.3.5 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 121*.

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 LQFP44 10x10 and www.ironwoodelectronics.com for LQFP32 7x7).

Table 121. Suggested list of socket types

Device	Socket (supplied with ST7MDT20J-EMU3)	Emulator adapter (supplied with ST7MDT20J-EMU3)
LQFP32 7X7	IRONWOOD SF-QFE32SA-L-01	IRONWOOD SK-UGA06/32A-01
LQFP44 10X10	YAMAICHI IC149-044-*52-*5	YAMAICHI ICP-044-5

14.4 ST7 Application notes

All relevant ST7 application notes can be found on www.st.com.

In the same way, as long as the SBK bit is set, break characters are sent to the TDO pin. This may lead to generate one break more than expected.

Occurrence

The occurrence of the problem is random and proportional to the baud rate. With a transmit frequency of 19200 baud (f_{CPU} = 8MHz and SCIBRR = 0xC9), the wrong break duration occurrence is around 1%.

Workaround

If this wrong duration is not compliant with the communication protocol in the application, software can request that an Idle line be generated before the break character. In this case, the break duration is always correct assuming the application is not doing anything between the idle and the break. This can be ensured by temporarily disabling interrupts.

The exact sequence is:

1. Disable interrupts
2. Reset and set TE (IDLE request)
3. Set and reset SBK (break request)
4. Re-enable interrupts

15.2 8/16 Kbyte Flash devices only

15.2.1 39-pulse ICC entry mode

ICC mode entry using ST7 application clock (39 pulses) is not supported. External clock mode must be used (36 pulses). Refer to the *ST7 Flash Programming Reference Manual*.

15.2.2 Negative current injection on pin PB0

Negative current injection on pin PB0 degrades the performance of the device and is not allowed on this pin.

15.3 8/16 Kbyte ROM devices only

15.3.1 Readout protection with LVD

Readout protection is not supported if the LVD is enabled.

15.3.2 I/O Port A and F configuration

When using an external quartz crystal or ceramic resonator, a few f_{OSC2} clock periods may be lost when the signal pattern in *Table 122* occurs. This is because this pattern causes the device to enter test mode and return to user mode after a few clock periods. User program execution and I/O status are not changed, only a few clock cycles are lost.

This happens with either one of the following configurations

- PA3 = 0, PF4 = 1, PF1 = 0 while PLL option is disabled and PF0 is toggling
- PA3 = 0, PF4 = 1, PF1 = 0, PF0 = 1 while PLL option is enabled

This is detailed in *Table 122*