



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	16MHz
Connectivity	I ² C, SPI
Peripherals	LVD, POR, PWM, WDT
Number of I/O	22
Program Memory Size	8KB (8K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	256 x 8
Voltage - Supply (Vcc/Vdd)	3.2V ~ 5.5V
Data Converters	A/D 6x8b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-SOIC (0.295", 7.50mm Width)
Supplier Device Package	28-SO
Purchase URL	https://www.e-xfl.com/product-detail/stmicroelectronics/st72c254g2m6

Table of Contents

13.9.1 Asynchronous RESET Pin	121
13.9.2 ISPSEL Pin	123
13.10 TIMER PERIPHERAL CHARACTERISTICS	124
13.10.1 Watchdog Timer	124
13.10.2 16-Bit Timer	124
13.11 COMMUNICATION INTERFACE CHARACTERISTICS	125
13.11.1 SPI - Serial Peripheral Interface	125
13.11.2 I2C - Inter IC Control Interface	127
13.12 8-BIT ADC CHARACTERISTICS	128
14 PACKAGE CHARACTERISTICS	130
14.1 PACKAGE MECHANICAL DATA	130
14.2 THERMAL CHARACTERISTICS	131
14.3 SOLDERING INFORMATION	132
15 DEVICE CONFIGURATION AND ORDERING INFORMATION	133
15.1 OPTION BYTES	133
15.2 DEVICE ORDERING INFORMATION AND TRANSFER OF CUSTOMER CODE	134
15.3 DEVELOPMENT TOOLS	136
15.3.1 PACKAGE/SOCKET FOOTPRINT PROPOSAL	137
15.4 ST7 APPLICATION NOTES	138
16 SUMMARY OF CHANGES	140

2 PIN DESCRIPTION

Figure 2. 28-Pin SO Package Pinout

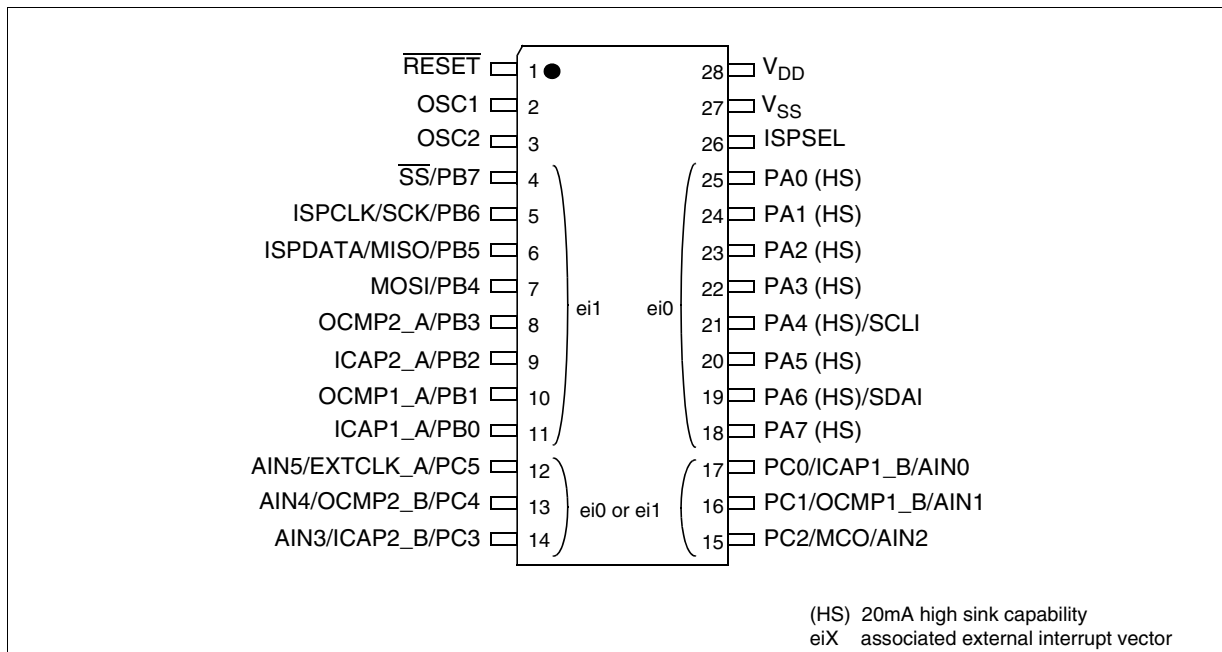
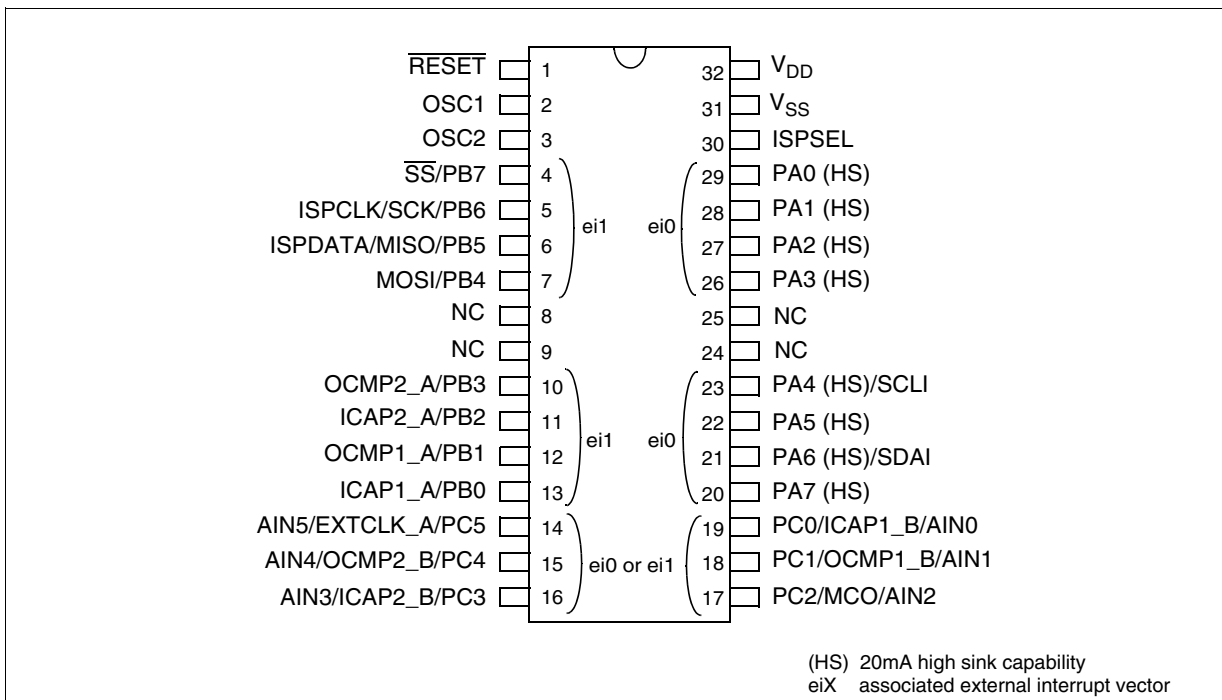


Figure 3. 32-Pin SDIP Package Pinout



Address	Block	Register Label	Register Name	Reset Status	Remarks
0031h	TIMER A	TACR2	Timer A Control Register 2	00h	R/W
0032h		TACR1	Timer A Control Register 1	00h	R/W
0033h		TASR	Timer A Status Register	xxh	Read Only
0034h		TAIC1HR	Timer A Input Capture 1 High Register	xxh	Read Only
0035h		TAIC1LR	Timer A Input Capture 1 Low Register	xxh	Read Only
0036h		TAOC1HR	Timer A Output Compare 1 High Register	80h	R/W
0037h		TAOC1LR	Timer A Output Compare 1 Low Register	00h	R/W
0038h		TACHR	Timer A Counter High Register	FFh	Read Only
0039h		TACLR	Timer A Counter Low Register	FCh	Read Only
003Ah		TAACHR	Timer A Alternate Counter High Register	FFh	Read Only
003Bh		TAACLR	Timer A Alternate Counter Low Register	FCh	Read Only
003Ch		TAIC2HR	Timer A Input Capture 2 High Register	xxh	Read Only
003Dh		TAIC2LR	Timer A Input Capture 2 Low Register	xxh	Read Only
003Eh		TAOC2HR	Timer A Output Compare 2 High Register	80h	R/W
003Fh		TAOC2LR	Timer A Output Compare 2 Low Register	00h	R/W
0040h		MISCR2	Miscellaneous Register 2	00h	R/W
0041h	TIMER B	TBCR2	Timer B Control Register 2	00h	R/W
0042h		TBCR1	Timer B Control Register 1	00h	R/W
0043h		TBSR	Timer B Status Register	xxh	Read Only
0044h		TBIC1HR	Timer B Input Capture 1 High Register	xxh	Read Only
0045h		TBIC1LR	Timer B Input Capture 1 Low Register	xxh	Read Only
0046h		TBOC1HR	Timer B Output Compare 1 High Register	80h	R/W
0047h		TBOC1LR	Timer B Output Compare 1 Low Register	00h	R/W
0048h		TBCHR	Timer B Counter High Register	FFh	Read Only
0049h		TBCLR	Timer B Counter Low Register	FCh	Read Only
004Ah		TBACHR	Timer B Alternate Counter High Register	FFh	Read Only
004Bh		TBACLR	Timer B Alternate Counter Low Register	FCh	Read Only
004Ch		TBIC2HR	Timer B Input Capture 2 High Register	xxh	Read Only
004Dh		TBIC2LR	Timer B Input Capture 2 Low Register	xxh	Read Only
004Eh		TBOC2HR	Timer B Output Compare 2 High Register	80h	R/W
004Fh		TBOC2LR	Timer B Output Compare 2 Low Register	00h	R/W
0050h to 006Fh	Reserved (32 Bytes)				
0070h	ADC	ADCDR	Data Register	00h	Read Only
0071h		ADCCSR	Control/Status Register	00h	R/W
0072h to 007Fh	Reserved (14 Bytes)				

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

Notes:

1. The contents of the I/O port DR registers are readable only in output configuration. In input configuration, the values of the I/O pins are returned instead of the DR register contents.
2. The bits associated with unavailable pins must always keep their reset value.

4 FLASH PROGRAM MEMORY

4.1 INTRODUCTION

FLASH devices have a single voltage non-volatile FLASH memory that may be programmed in-situ (or plugged in a programming tool) on a byte-by-byte basis.

4.2 MAIN FEATURES

- Remote In-Situ Programming (ISP) mode
- Up to 16 bytes programmed in the same cycle
- MTP memory (Multiple Time Programmable)
- Read-out memory protection against piracy

4.3 STRUCTURAL ORGANISATION

The FLASH program memory is organised in a single 8-bit wide memory block which can be used for storing both code and data constants.

The FLASH program memory is mapped in the upper part of the ST7 addressing space and includes the reset and interrupt user vector area .

4.4 IN-SITU PROGRAMMING (ISP) MODE

The FLASH program memory can be programmed using Remote ISP mode. This ISP mode allows the contents of the ST7 program memory to be updated using a standard ST7 programming tools after the device is mounted on the application board. This feature can be implemented with a minimum number of added components and board area impact.

An example Remote ISP hardware interface to the standard ST7 programming tool is described below. For more details on ISP programming, refer to the ST7 Programming Specification.

Remote ISP Overview

The Remote ISP mode is initiated by a specific sequence on the dedicated ISPSEL pin.

The Remote ISP is performed in three steps:

- Selection of the RAM execution mode
- Download of Remote ISP code in RAM
- Execution of Remote ISP code in RAM to program the user program into the FLASH

Remote ISP hardware configuration

In Remote ISP mode, the ST7 has to be supplied with power (V_{DD} and V_{SS}) and a clock signal (oscillator and application crystal circuit for example).

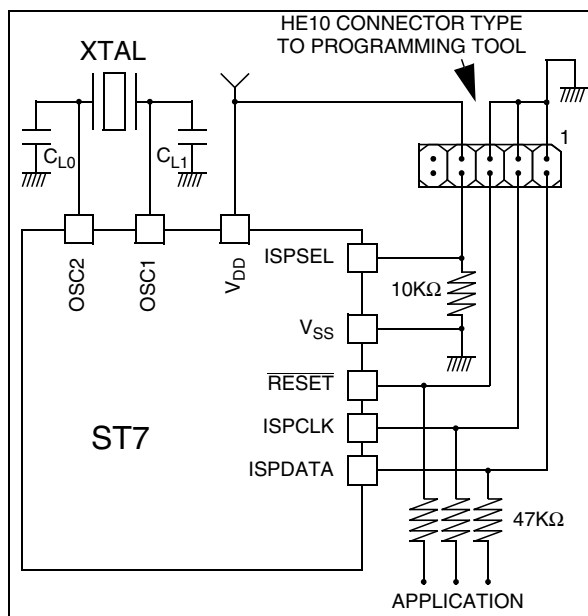
This mode needs five signals (plus the V_{DD} signal if necessary) to be connected to the programming tool. This signals are:

- **RESET**: device reset
- **V_{SS}** : device ground power supply
- **ISPCLK**: ISP output serial clock pin
- **ISPDATA**: ISP input serial data pin
- **ISPSEL**: Remote ISP mode selection. This pin must be connected to V_{SS} on the application board through a pull-down resistor.

If any of these pins are used for other purposes on the application, a serial resistor has to be implemented to avoid a conflict if the other device forces the signal level.

Figure 5 shows a typical hardware interface to a standard ST7 programming tool. For more details on the pin locations, refer to the device pinout description.

Figure 5. Typical Remote ISP Interface



4.5 MEMORY READ-OUT PROTECTION

The read-out protection is enabled through an option bit.

For FLASH devices, when this option is selected, the program and data stored in the FLASH memory are protected against read-out piracy (including a re-write protection). When this protection option is removed the entire FLASH program memory is first automatically erased. However, the E²PROM data memory (when available) can be protected only with ROM devices.

6.3 MULTI-OSCILLATOR (MO)

The main clock of the ST7 can be generated by four different source types coming from the multi-oscillator block:

- an external source
- 4 crystal or ceramic resonator oscillators
- an external RC oscillator
- an internal high frequency RC oscillator

Each oscillator is optimized for a given frequency range in terms of consumption and is selectable through the option byte. The associated hardware configuration are shown in Table 3. Refer to the electrical characteristics section for more details.

External Clock Source

In this external clock mode, a clock signal (square, sinus or triangle) with ~50% duty cycle has to drive the OSC1 pin while the OSC2 pin is tied to ground.

Crystal/Ceramic Oscillators

This family of oscillators has the advantage of producing a very accurate rate on the main clock of the ST7. The selection within a list of 4 oscillators with different frequency ranges has to be done by option byte in order to reduce consumption. In this mode of the multi-oscillator, the resonator and the load capacitors have to be placed as close as possible to the oscillator pins in order to minimize output distortion and start-up stabilization time. The loading capacitance values must be adjusted according to the selected oscillator.

These oscillators are not stopped during the RESET phase to avoid losing time in the oscillator start-up phase.

External RC Oscillator

This oscillator allows a low cost solution for the main clock of the ST7 using only an external resistor and an external capacitor. The frequency of the external RC oscillator (in the range of some MHz.) is fixed by the resistor and the capacitor values. Consequently in this MO mode, the accuracy of the clock is dependent on V_{DD} , T_A , process variations and the accuracy of the discrete components used. This option should not be used in applications that require accurate timing.

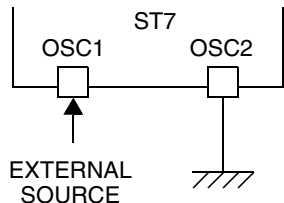
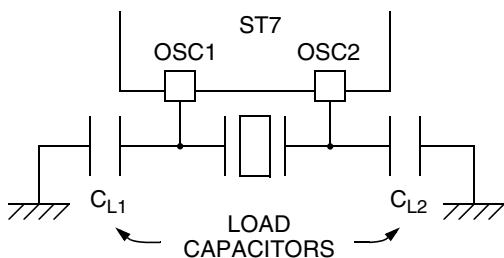
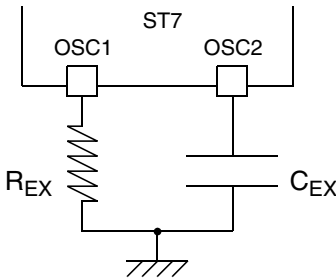
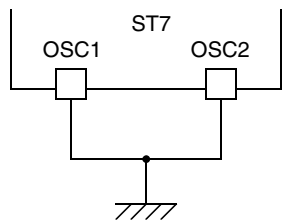
Internal RC Oscillator

The internal RC oscillator mode is based on the same principle as the external RC oscillator including the resistance and the capacitance of the device. This mode is the most cost effective one with the drawback of a lower frequency accuracy. Its frequency is in the range of several MHz. This op-

tion should not be used in applications that require accurate timing.

In this mode, the two oscillator pins have to be tied to ground.

Table 3. ST7 Clock Sources

	Hardware Configuration
External Clock	
Crystal/Ceramic Resonators	
External RC Oscillator	
Internal RC Oscillator	

7 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 1.

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 will be cleared and the main program will resume.

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).

7.1 NON-MASKABLE SOFTWARE INTERRUPT

This interrupt is entered when the TRAP instruction is executed regardless of the state of the I bit.

It will be serviced according to the flowchart on Figure 1.

7.2 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.

If several input pins, connected to the same interrupt vector, are configured as interrupts, their signals are logically NAnDED before entering the edge/level detection block.

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 on 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.

7.3 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.

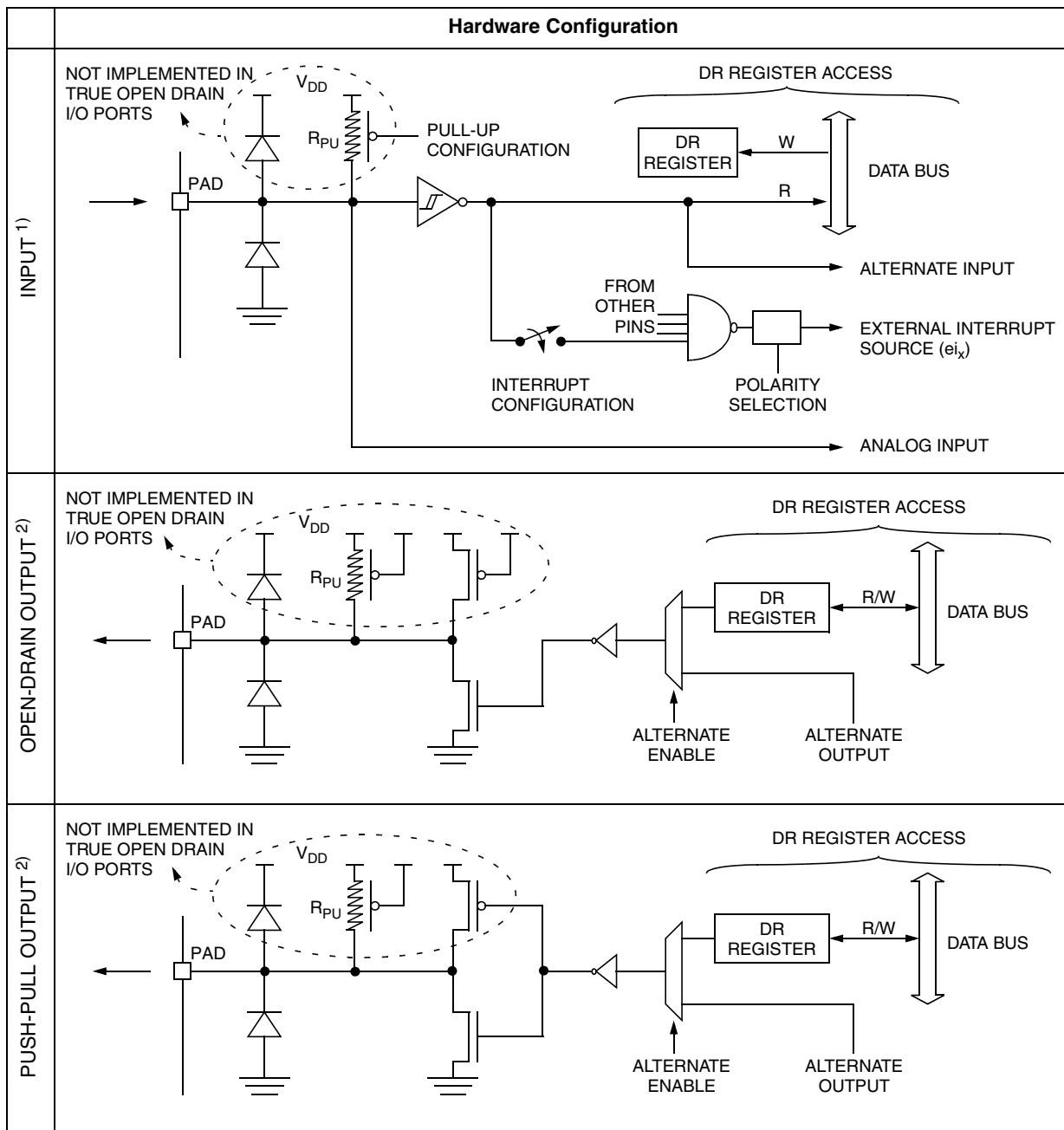
Clearing an interrupt request is done by:

- 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 to be enabled) will therefore be lost if the clear sequence is executed.

I/O PORTS (Cont'd)

Table 7. I/O Port Configurations



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.

WATCHDOG TIMER (Cont'd)**Table 11. Watchdog Timing ($f_{CPU} = 8\text{ MHz}$)**

	CR Register initial value	WDG timeout period (ms)
Max	FFh	98.304
Min	C0h	1.536

Notes: 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).

11.1.4 Hardware Watchdog Option

If Hardware Watchdog is selected by option byte, the watchdog is always active and the WDGA bit in the CR is not used.

Refer to the device-specific Option Byte description.

11.1.5 Low Power Modes**WAIT Instruction**

No effect on Watchdog.

HALT Instruction

If the Watchdog reset on HALT option is selected by option byte, a HALT instruction causes an immediate reset generation if the Watchdog is activated (WDGA bit is set).

11.1.5.1 Using Halt Mode with the WDG (option)

If the Watchdog reset on HALT option is not selected by option byte, the Halt mode can be used when the watchdog is enabled.

In this case, the HALT instruction stops the oscillator. When the oscillator is stopped, the WDG stops counting and is no longer able to generate a reset until the microcontroller receives an external interrupt or a reset.

If an external interrupt is received, the WDG restarts counting after 4096 CPU clocks. If a reset is generated, the WDG is disabled (reset state).

Recommendations

- Make sure that an external event is available to wake up the microcontroller from Halt mode.
- Before executing the HALT instruction, refresh the WDG counter, to avoid an unexpected WDG

reset immediately after waking up the microcontroller.

- When using an external interrupt to wake up the microcontroller, reinitialize the corresponding I/O as “Input Pull-up with Interrupt” before executing the HALT instruction. The main reason for this is that the I/O may be wrongly configured due to external interference or by an unforeseen logical condition.
- For the same reason, reinitialize the level sensitivity of each external interrupt as a precautionary measure.
- The opcode for the HALT instruction is 0x8E. To avoid an unexpected HALT instruction due to a program counter failure, it is advised to clear all occurrences of the data value 0x8E from memory. For example, avoid defining a constant in ROM with the value 0x8E.
- As the HALT instruction clears the I bit in the CC register to allow interrupts, the user may choose to clear all pending interrupt bits before executing the HALT instruction. This avoids entering other peripheral interrupt routines after executing the external interrupt routine corresponding to the wake-up event (reset or external interrupt).

11.1.6 Interrupts

None.

11.1.7 Register Description**CONTROL REGISTER (CR)**

Read/Write

Reset Value: 0111 1111 (7Fh)

7						0	
WDGA	T6	T5	T4	T3	T2	T1	T0

Bit 7 = **WDGA** Activation bit.

This bit is set by software and only cleared by hardware after a reset. When WDGA = 1, the watchdog can generate a reset.

0: Watchdog disabled

1: Watchdog enabled

Bit 6:0 = **T[6:0]** 7-bit timer (MSB to LSB).

These bits contain the decremented value. A reset is produced when it rolls over from 40h to 3Fh (T6 becomes cleared).

WATCHDOG TIMER (Cont'd)

Table 12. Watchdog Timer Register Map and Reset Values

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
0024h	WDGCR Reset Value	WDGA 0	T6 1	T5 1	T4 1	T3 1	T2 1	T1 1	T0 1

16-BIT TIMER (Cont'd)

Figure 30. Input Capture Block Diagram

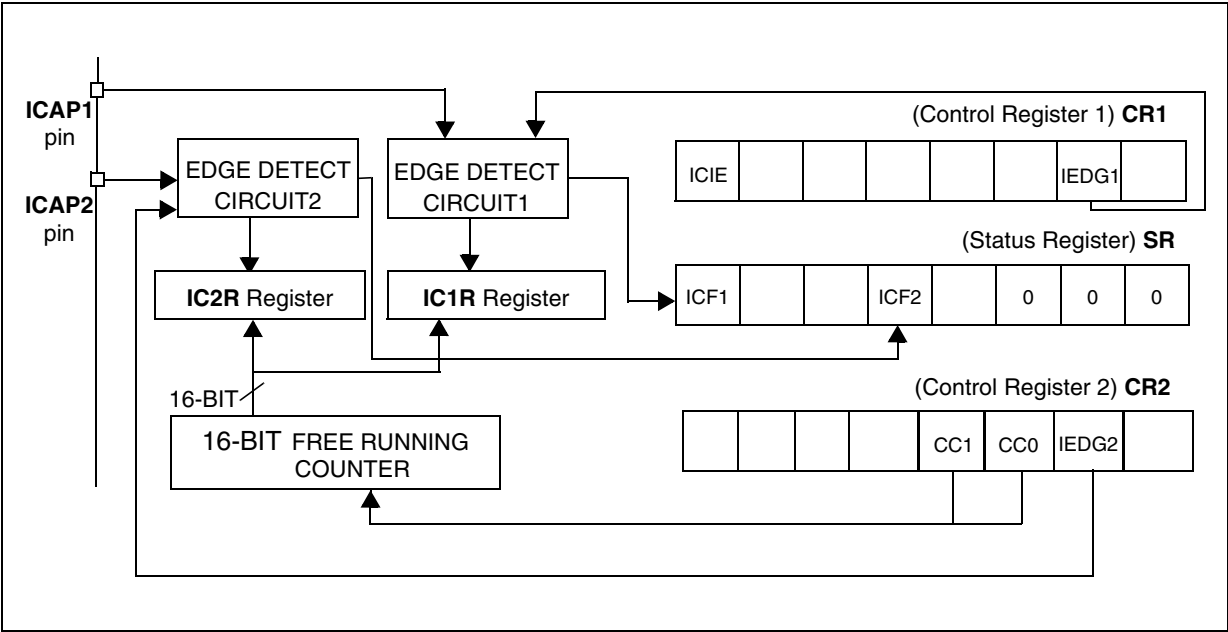
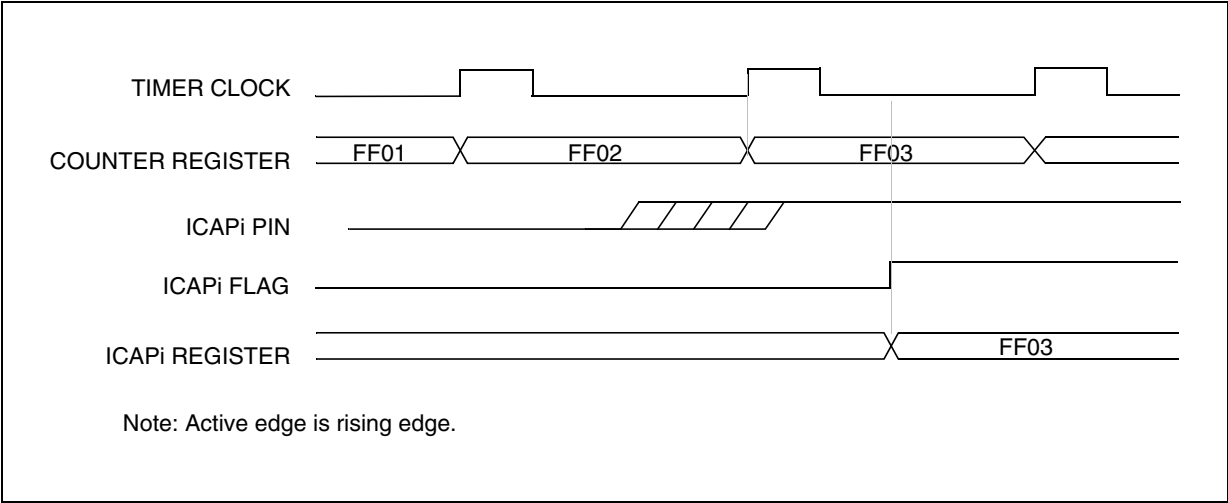


Figure 31. Input Capture Timing Diagram



16-BIT TIMER (Cont'd)**11.2.3.6 Pulse Width Modulation Mode**

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

The Pulse Width Modulation mode uses the complete Output Compare 1 function plus the OC2R register, and so these functions cannot be used when the PWM mode is activated.

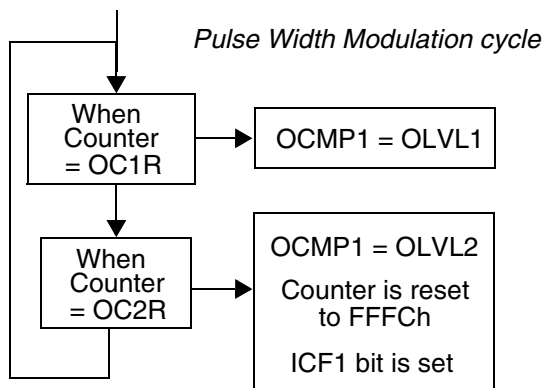
Procedure

To use Pulse Width Modulation mode:

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

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

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



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

$$\text{OC/R Value} = \frac{t \cdot f_{\text{CPU}}}{\text{PRESC}} - 5$$

Where:

t = Signal or pulse period (in seconds)

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

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

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

$$\text{OC/R} = t \cdot f_{\text{EXT}} - 5$$

Where:

t = Signal or pulse period (in seconds)

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

The Output Compare 2 event causes the counter to be initialized to FFFCh (See Figure 11)

Notes:

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

16-BIT TIMER (Cont'd)**11.2.4 Low Power Modes**

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

11.2.5 Interrupts

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

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

11.2.6 Summary of Timer modes

MODES	AVAILABLE RESOURCES			
	Input Capture 1	Input Capture 2	Output Compare 1	Output Compare 2
Input Capture (1 and/or 2)	Yes	Yes	Yes	Yes
Output Compare (1 and/or 2)	Yes	Yes	Yes	Yes
One Pulse mode	No	Not Recommended ¹⁾	No	Partially ²⁾
PWM Mode	No	Not Recommended ³⁾	No	No

¹⁾ See note 4 in Section 0.1.3.5 One Pulse Mode

²⁾ See note 5 in Section 0.1.3.5 One Pulse Mode

³⁾ See note 4 in Section 0.1.3.6 Pulse Width Modulation Mode

11.3 SERIAL PERIPHERAL INTERFACE (SPI)

11.3.1 Introduction

The Serial Peripheral Interface (SPI) allows full-duplex, synchronous, serial communication with external devices. An SPI system may consist of a master and one or more slaves or a system in which devices may be either masters or slaves.

The SPI is normally used for communication between the microcontroller and external peripherals or another microcontroller.

Refer to the Pin Description chapter for the device-specific pin-out.

11.3.2 Main Features

- Full duplex, three-wire synchronous transfers
- Master or slave operation
- Four master mode frequencies
- Maximum slave mode frequency = $f_{CPU}/4$
- Four programmable master bit rates
- Programmable clock polarity and phase
- End of transfer interrupt flag
- Write collision flag protection
- Master mode fault protection capability

11.3.3 General description

The SPI is connected to external devices through 4 alternate pins:

- MISO: Master In Slave Out pin
- MOSI: Master Out Slave In pin
- SCK: Serial Clock pin
- \overline{SS} : Slave select pin

A basic example of interconnections between a single master and a single slave is illustrated on Figure 1.

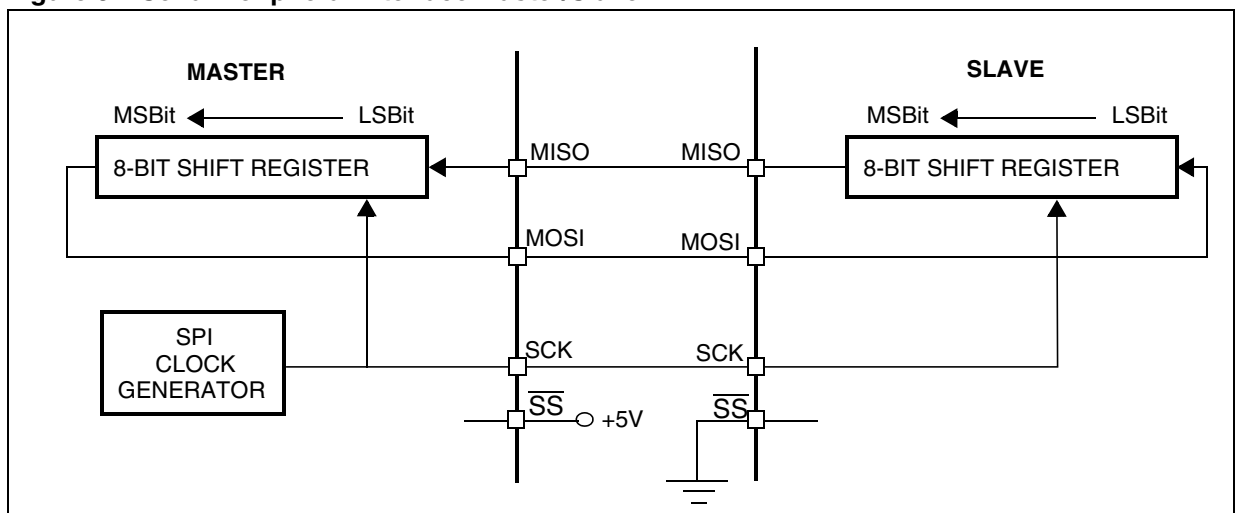
The MOSI pins are connected together as are MISO pins. In this way data is transferred serially between master and slave (most significant bit first).

When the master device transmits data to a slave device via MOSI pin, the slave device responds by sending data to the master device via the MISO pin. This implies full duplex transmission with both data out and data in synchronized with the same clock signal (which is provided by the master device via the SCK pin).

Thus, the byte transmitted is replaced by the byte received and eliminates the need for separate transmit-empty and receiver-full bits. A status flag is used to indicate that the I/O operation is complete.

Four possible data/clock timing relationships may be chosen (see Figure 4) but master and slave must be programmed with the same timing mode.

Figure 37. Serial Peripheral Interface Master/Slave



SERIAL PERIPHERAL INTERFACE (Cont'd)

11.3.4 Functional Description

Figure 1 shows the serial peripheral interface (SPI) block diagram.

This interface contains three dedicated registers:

- A Control Register (CR)
- A Status Register (SR)
- A Data Register (DR)

Refer to the CR, SR and DR registers in Section 0.1.7 for the bit definitions.

11.3.4.1 Master Configuration

In a master configuration, the serial clock is generated on the SCK pin.

Procedure

- Select the SPR0 & SPR1 bits to define the serial clock baud rate (see CR register).
- Select the CPOL and CPHA bits to define one of the four relationships between the data transfer and the serial clock (see Figure 4).
- The \overline{SS} pin must be connected to a high level signal during the complete byte transmit sequence.
- The MSTR and SPE bits must be set (they remain set only if the \overline{SS} pin is connected to a high level signal).

In this configuration the MOSI pin is a data output and to the MISO pin is a data input.

Transmit sequence

The transmit sequence begins when a byte is written to the DR register.

The data byte is parallel loaded into the 8-bit shift register (from the internal bus) during a write cycle and then shifted out serially to the MOSI pin most significant bit first.

When data transfer is complete:

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

During the last clock cycle the SPIF bit is set, a copy of the data byte received in the shift register is moved to a buffer. When the DR register is read, the SPI peripheral returns this buffered value.

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

1. An access to the SR register while the SPIF bit is set
2. A read to the DR register.

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

SERIAL PERIPHERAL INTERFACE (Cont'd)**11.3.4.7 Single Master and Multimaster Configurations**

There are two types of SPI systems:

- Single Master System
- Multimaster System

Single Master System

A typical single master system may be configured, using an MCU as the master and four MCUs as slaves (see Figure 6).

The master device selects the individual slave devices by using four pins of a parallel port to control the four \overline{SS} pins of the slave devices.

The \overline{SS} pins are pulled high during reset since the master device ports will be forced to be inputs at that time, thus disabling the slave devices.

Note: To prevent a bus conflict on the MISO line the master allows only one active slave device during a transmission.

For more security, the slave device may respond to the master with the received data byte. Then the master will receive the previous byte back from the slave device if all MISO and MOSI pins are connected and the slave has not written its DR register.

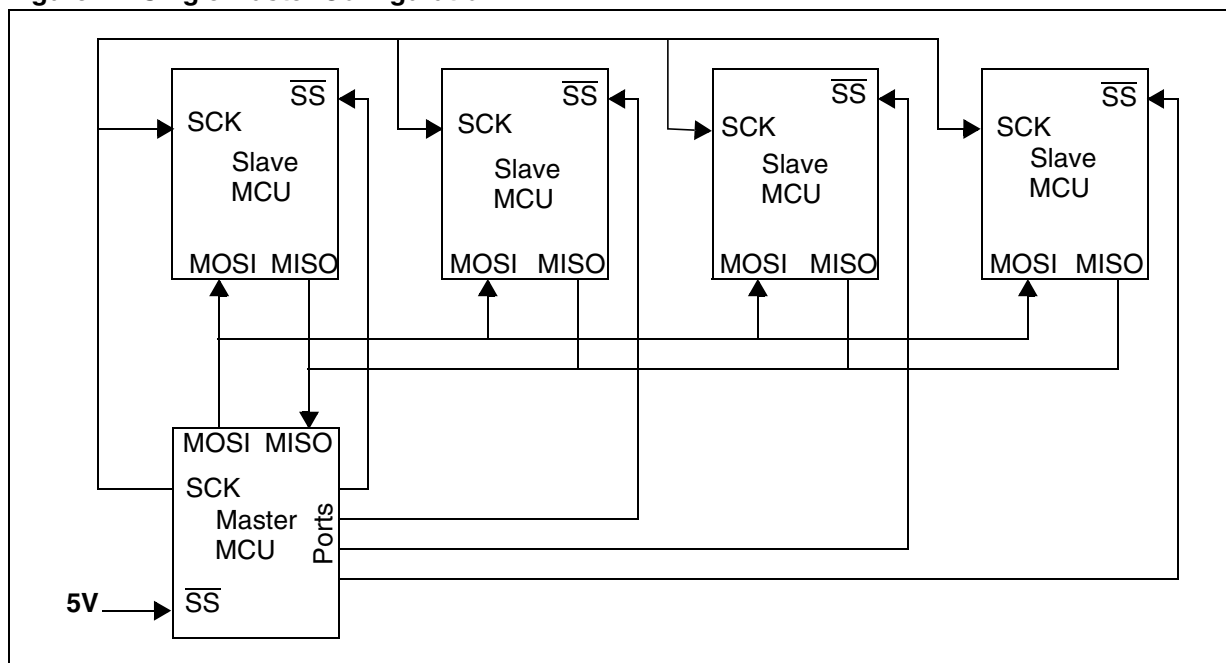
Other transmission security methods can use ports for handshake lines or data bytes with command fields.

Multi-master System

A multi-master system may also be configured by the user. Transfer of master control could be implemented using a handshake method through the I/O ports or by an exchange of code messages through the serial peripheral interface system.

The multi-master system is principally handled by the MSTR bit in the CR register and the MODF bit in the SR register.

Figure 42. Single Master Configuration



12.2 INSTRUCTION GROUPS

The ST7 family devices use an Instruction Set consisting of 63 instructions. The instructions may

be subdivided into 13 main groups as illustrated in the following table:

Load and Transfer	LD	CLR						
Stack operation	PUSH	POP	RSP					
Increment/Decrement	INC	DEC						
Compare and Tests	CP	TNZ	BCP					
Logical operations	AND	OR	XOR	CPL	NEG			
Bit Operation	BSET	BRES						
Conditional Bit Test and Branch	BTJT	BTJF						
Arithmetic operations	ADC	ADD	SUB	SBC	MUL			
Shift and Rotates	SLL	SRL	SRA	RLC	RRC	SWAP	SLA	
Unconditional Jump or Call	JRA	JRT	JRF	JP	CALL	CALLR	NOP	RET
Conditional Branch	JRxx							
Interrupt management	TRAP	WFI	HALT	IRET				
Condition Code Flag modification	SIM	RIM	SCF	RCF				

Using a pre-byte

The instructions are described with one to four bytes.

In order to extend the number of available opcodes for an 8-bit CPU (256 opcodes), three different prebyte opcodes are defined. These prebytes modify the meaning of the instruction they precede.

The whole instruction becomes:

PC-2 End of previous instruction

PC-1 Prebyte

PC Opcode

PC+1 Additional word (0 to 2) according to the number of bytes required to compute the effective address

These prebytes enable instruction in Y as well as indirect addressing modes to be implemented. They precede the opcode of the instruction in X or the instruction using direct addressing mode. The prebytes are:

PDY 90 Replace an X based instruction using immediate, direct, indexed, or inherent addressing mode by a Y one.

PIX 92 Replace an instruction using direct, direct bit, or direct relative addressing mode to an instruction using the corresponding indirect addressing mode. It also changes an instruction using X indexed addressing mode to an instruction using indirect X indexed addressing mode.

PIY 91 Replace an instruction using X indirect indexed addressing mode by a Y one.

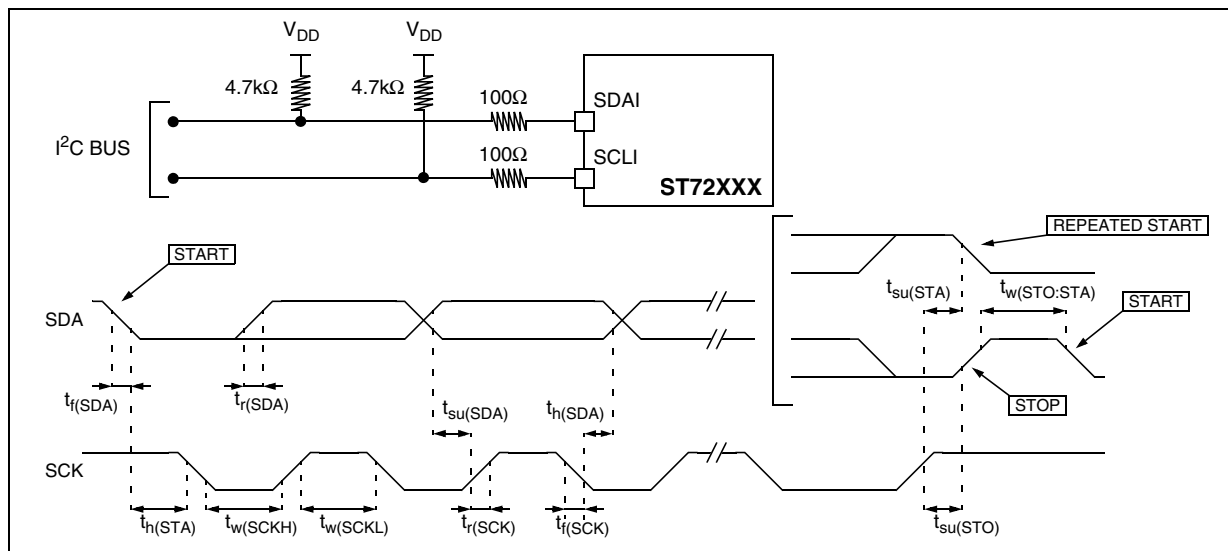
COMMUNICATION INTERFACE CHARACTERISTICS (Cont'd)

13.11.2 I²C - Inter IC Control Interface

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 (SDAI and SCLI). The ST7 I²C interface meets the requirements of the Standard I²C communication protocol described in the following table.

Symbol	Parameter	Standard mode I ² C		Fast mode I ² C		Unit
		Min ¹⁾	Max ¹⁾	Min ¹⁾	Max ¹⁾	
$t_{w(SCLL)}$	SCL clock low time	4.7		1.3		μs
$t_{w(SCLH)}$	SCL clock high time	4.0		0.6		
$t_{su(SDA)}$	SDA setup time	250		100		ns
$t_h(SDA)$	SDA data hold time	0 ³⁾		0 ²⁾	900 ³⁾	
$t_r(SDA)$ $t_r(SCL)$	SDA and SCL rise time		1000	$20+0.1C_b$	300	
$t_f(SDA)$ $t_f(SCL)$	SDA and SCL fall time		300	$20+0.1C_b$	300	
$t_h(STA)$	START condition hold time	4.0		0.6		μs
$t_{su(STA)}$	Repeated START condition setup time	4.7		0.6		
$t_{su(STO)}$	STOP condition setup time	4.0		0.6		ns
$t_{w(STO:STA)}$	STOP to START condition time (bus free)	4.7		1.3		ms
C_b	Capacitive load for each bus line		400		400	pF

Figure 94. Typical Application with I²C Bus and Timing Diagram ⁴⁾

Notes:

1. Data based on standard I²C protocol requirement, not tested in production.
2. The device must internally provide a hold time of at least 300ns for the SDA signal in order to bridge the undefined region of the falling edge of SCL.
3. The maximum hold time of the START condition has only to be met if the interface does not stretch the low period of SCL signal.
4. Measurement points are done at CMOS levels: $0.3 \times V_{DD}$ and $0.7 \times V_{DD}$.

Signature:

15.3 DEVELOPMENT TOOLS

STMicroelectronics offers a range of hardware and software development tools for the ST7 microcontroller family. Full details of tools available for the ST7 from third party manufacturers can be obtained from the STMicroelectronics Internet site:

→ <http://mcu.st.com>.

Third Party Tools

- ACTUM
- BP
- COSMIC
- CMX
- DATA I/O
- HITEK
- HIWARE
- ISYSTEM
- KANDA
- LEAP

Tools from these manufacturers include C compilers, emulators and gang programmers.

STMicroelectronics Tools

Three types of development tool are offered by ST, all of them connect to a PC via a parallel (LPT) port: see Table 26 and Table 27 for more details.

Table 26. STMicroelectronics Tool Features

	In-Circuit Emulation	Programming Capability ¹⁾	Software Included
ST7 Development Kit	Yes. (Same features as HDS2 emulator but without logic analyzer)	Yes (DIP packages only)	ST7 CD ROM with: <ul style="list-style-type: none"> – ST7 Assembly toolchain – STVD7 and WGDB7 powerful Source Level Debugger for Win 3.1, Win 95 and NT – C compiler demo versions – ST Realizer for Win 3.1 and Win 95. – Windows Programming Tools for Win 3.1, Win 95 and NT
ST7 HDS2 Emulator	Yes, powerful emulation features including trace/ logic analyzer	No	
ST7 Programming Board	No	Yes (All packages)	

Table 27. Dedicated STMicroelectronics Development Tools

Supported Products	ST7 Development Kit	ST7 HDS2 Emulator	ST7 Programming Board
ST72254G1, ST72C254G1 ST72254G2, ST72C254G2 ST72215G2, ST72C215G2 ST72216G1, ST72C216G1 ST72104G1, ST72C104G1, ST72104G2, ST72C104G2	ST7MDT1-DVP2	ST7MDT1-EMU2B	ST7MDT1-EPB2/EU ST7MDT1-EPB2/US ST7MDT1-EPB2/UK

Note:

1. In-Situ Programming (ISP) interface for FLASH devices.

DEVELOPMENT TOOLS (Cont'd)**15.3.1 PACKAGE/SOCKET FOOTPRINT PROPOSAL****Table 28. Suggested List of SDIP32 Socket Types**

Package / Probe	Adaptor / Socket Reference	Same Footprint	Socket Type
SDIP32 EMU PROBE	TEXT TOOL 232-1291-00	X	Textool

Table 29. Suggested List of SO28 Socket Types

Package / Probe	Adaptor / Socket Reference	Same Footprint	Socket Type
SO28	ENPLAS OTS-28-1.27-04		Open Top
	YAMAICHI IC51-0282-334-1		Clamshell
EMU PROBE	Adapter from SO28 to SDIP32 footprint (delivered with emulator)	X	SMD to SDIP