**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 | 32 |
| Program Memory Size | 16KB (16K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 512 x 8 |
| Voltage - Supply (Vcc/Vdd) | 3.8V ~ 5.5V |
| Data Converters | A/D 12x10b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 44-LQFP |
| Supplier Device Package | - |
| Purchase URL | https://www.e-xfl.com/product-detail/stmicroelectronics/st72f324j4tae |

10.4.7 Interrupts . . . 86
10.4.8 Register Description . . . 87
10.5 SERIAL COMMUNICATIONS INTERFACE (SCI) . . . 90
10.5.1 Introduction . . . 90
10.5.2 Main Features . . . 90
10.5.3 General Description . . . 90
10.5.4 Functional Description . . . 92
10.5.5 Low Power Modes . . . 99
10.5.6 Interrupts . . . 99
10.5.7 Register Description . . . 100
10.6 10-BIT A/D CONVERTER (ADC) . . . 106
10.6.1 Introduction . . . 106
10.6.2 Main Features . . . 106
10.6.3 Functional Description . . . 107
10.6.4 Low Power Modes . . . 107
10.6.5 Interrupts . . . 107
10.6.6 Register Description . . . 108
**11 INSTRUCTION SET . . . 110**
11.1 CPU ADDRESSING MODES . . . 110
11.1.1 Inherent . . . 111
11.1.2 Immediate . . . 111
11.1.3 Direct . . . 111
11.1.4 Indexed (No Offset, Short, Long) . . . 111
11.1.5 Indirect (Short, Long) . . . 111
11.1.6 Indirect Indexed (Short, Long) . . . 112
11.1.7 Relative mode (Direct, Indirect) . . . 112
11.2 INSTRUCTION GROUPS . . . 113
**12 ELECTRICAL CHARACTERISTICS . . . 116**
12.1 PARAMETER CONDITIONS . . . 116
12.1.1 Minimum and Maximum values . . . 116
12.1.2 Typical values . . . 116
12.1.3 Typical curves . . . 116
12.1.4 Loading capacitor . . . 116
12.1.5 Pin input voltage . . . 116
12.2 ABSOLUTE MAXIMUM RATINGS . . . 117
12.2.1 Voltage Characteristics . . . 117
12.2.2 Current Characteristics . . . 117
12.2.3 Thermal Characteristics . . . 118
12.3 OPERATING CONDITIONS . . . 118
12.3.1 Operating Conditions . . . 118
12.4 LVD/AVD CHARACTERISTICS . . . 119
12.4.1 Operating Conditions with Low Voltage Detector (LVD) . . . 119
12.4.2 Auxiliary Voltage Detector (AVD) Thresholds . . . 119
12.5 SUPPLY CURRENT CHARACTERISTICS . . . 120
12.5.1 CURRENT CONSUMPTION . . . 120
12.5.2 Supply and Clock Managers . . . 122
12.5.3 On-Chip Peripherals . . . 123

**PIN DESCRIPTION** (Cont'd)

For external pin connection guidelines, refer to See "ELECTRICAL CHARACTERISTICS" on page 116.

**Legend / Abbreviations for** Table 1**:**

Type: I = input, O = output, S = supply

Input level: A = Dedicated analog input

In/Output level: C = CMOS $0.3V_{DD}/0.7V_{DD}$
$C_T$= CMOS $0.3V_{DD}/0.7V_{DD}$ with input trigger

Output level: HS = 20mA high sink (on N-buffer only)

Port and control configuration:

– Input: float = floating, wpu = weak pull-up, int = interrupt [1], ana = analog ports

– Output: OD = open drain [2], PP = push-pull

Refer to "I/O PORTS" on page 45 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 1. Device Pin Description**

| Pin n° | | | | Pin Name | Type | Level | | Port | | | | | | Main function (after reset) | Alternate Function | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | Input | | | | Output | | | | |
| TQFP44 | SDIP42 | TQFP32 | SDIP32 | | | Input | Output | float | wpu | int | ana | OD | PP | | | |
| 6 | 1 | 30 | 1 | PB4 (HS) | I/O | $C_T$ | HS | **X** | ei3 | | | X | X | Port B4 | | |
| 7 | 2 | 31 | 2 | PD0/AIN0 | I/O | $C_T$ | | **X** | X | | X | X | X | Port D0 | ADC Analog Input 0 | |
| 8 | 3 | 32 | 3 | PD1/AIN1 | I/O | $C_T$ | | **X** | X | | X | X | X | Port D1 | ADC Analog Input 1 | |
| 9 | 4 | | | PD2/AIN2 | I/O | $C_T$ | | **X** | X | | X | X | X | Port D2 | ADC Analog Input 2 | |
| 10 | 5 | | | PD3/AIN3 | I/O | $C_T$ | | **X** | X | | X | X | X | Port D3 | ADC Analog Input 3 | |
| 11 | 6 | | | PD4/AIN4 | I/O | $C_T$ | | **X** | X | | X | X | X | Port D4 | ADC Analog Input 4 | |
| 12 | 7 | | | PD5/AIN5 | I/O | $C_T$ | | **X** | X | | X | X | X | Port D5 | ADC Analog Input 5 | |
| 13 | 8 | 1 | 4 | $V_{AREF}$ | S | | | | | | | | | Analog Reference Voltage for ADC | | |
| 14 | 9 | 2 | 5 | $V_{SSA}$ | S | | | | | | | | | Analog Ground Voltage | | |
| 15 | 10 | 3 | 6 | PF0/MCO/AIN8 | I/O | $C_T$ | | **X** | ei1 | | X | X | X | Port F0 | Main clock out ($f_{CPU}$) | ADC Analog Input 8 |
| 16 | 11 | 4 | 7 | PF1 (HS)/BEEP | I/O | $C_T$ | HS | **X** | ei1 | | | X | X | Port F1 | Beep signal output | |
| 17 | 12 | | | PF2 (HS) | I/O | $C_T$ | HS | **X** | | ei1 | | X | X | Port F2 | | |
| 18 | 13 | 5 | 8 | 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 | 14 | 6 | 9 | PF6 (HS)/ICAP1_A | I/O | $C_T$ | HS | **X** | X | | | X | X | Port F6 | Timer A Input Capture 1 | |
| 20 | 15 | 7 | 10 | PF7 (HS)/ EXTCLK_A | I/O | $C_T$ | HS | **X** | X | | | X | X | Port F7 | Timer A External Clock Source | |
| 21 | | | | $V_{DD\_0}$ | S | | | | | | | | | Digital Main Supply Voltage | | |
| 22 | | | | $V_{SS\_0}$ | S | | | | | | | | | Digital Ground Voltage | | |
| 23 | 16 | 8 | 11 | PC0/OCMP2_B/ AIN12 | I/O | $C_T$ | | **X** | X | | X | X | X | Port C0 | Timer B Output Compare 2 | ADC Analog Input 12 |

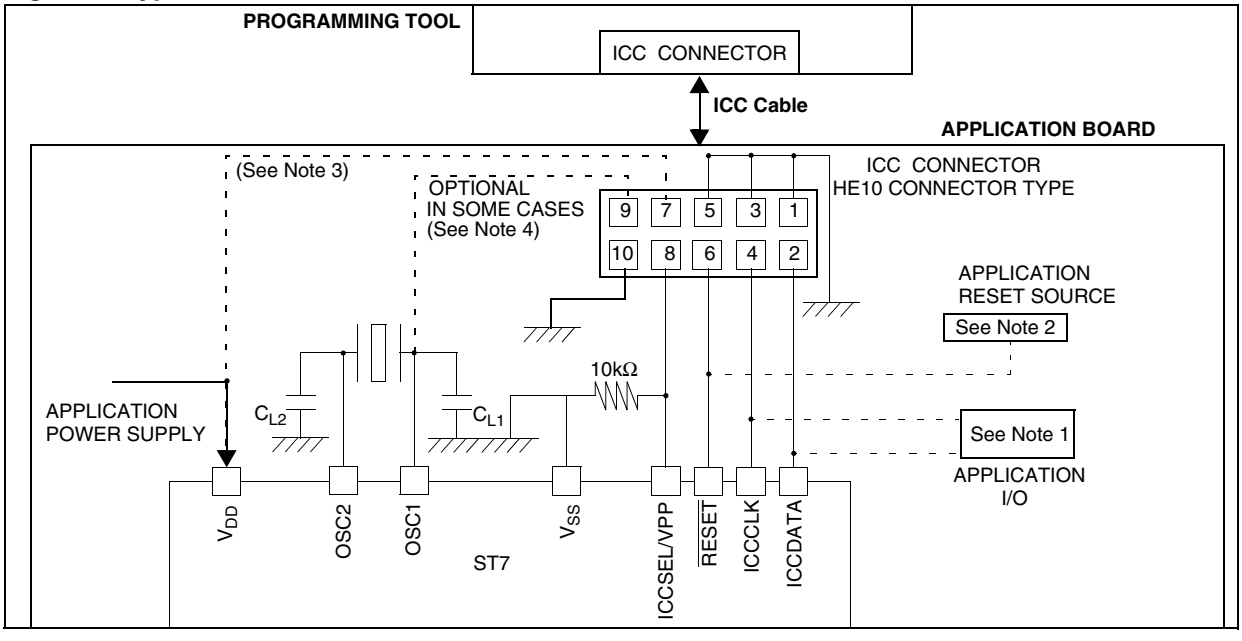
**FLASH PROGRAM MEMORY** (Cont'd)

### 4.4 ICC Interface

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

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

**Figure 7. Typical ICC Interface**




**Notes:**

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

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

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

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

# 6 SUPPLY, RESET AND CLOCK MANAGEMENT

The device includes a range of utility features for securing the application in critical situations (for example in case of a power brown-out), and reducing the number of external components. An overview is shown in Figure 11.

For more details, refer to dedicated parametric section.

**Main features**

- Optional PLL for multiplying the frequency by 2 (not to be used with internal RC oscillator in order to respect the max. operating frequency)
- Reset Sequence Manager (RSM)
- Multi-Oscillator Clock Management (MO)
  - 5 Crystal/Ceramic resonator oscillators
  - 1 Internal RC oscillator
- System Integrity Management (SI)
  - Main supply Low voltage detection (LVD)
  - Auxiliary Voltage detector (AVD) with interrupt capability for monitoring the main supply

## 6.1 PHASE LOCKED LOOP

If the clock frequency input to the PLL is in the range 2 to 4 MHz, the PLL can be used to multiply the frequency by two to obtain an $f_{OSC2}$ of 4 to 8 MHz. The PLL is enabled by option byte. If the PLL is disabled, then $f_{OSC2} = f_{OSC}/2$.

**Caution:** The PLL is not recommended for applications where timing accuracy is required.

**Caution**: The PLL must not be used with the internal RC oscillator.

**Figure 10. PLL Block Diagram**

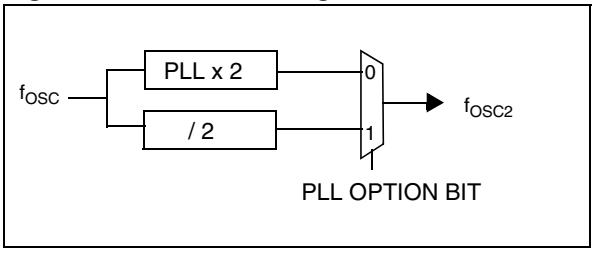
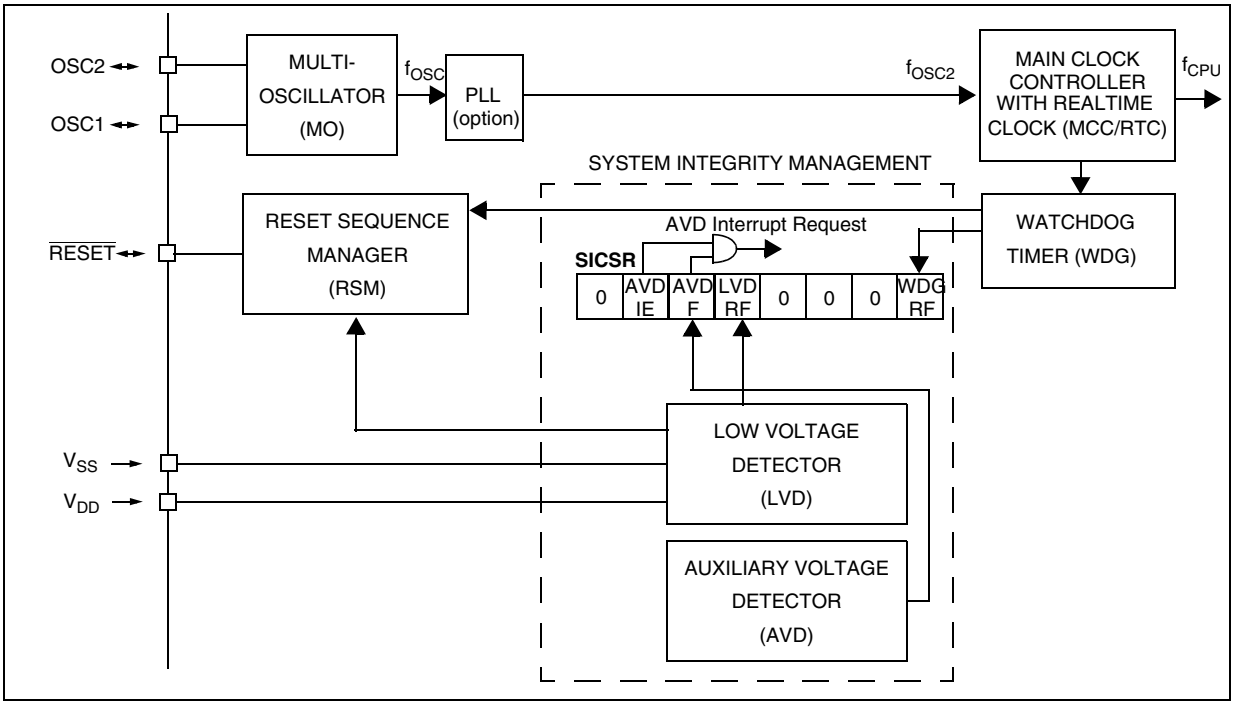



**Figure 11. Clock, Reset and Supply Block Diagram**



# 7 INTERRUPTS

## 7.1 INTRODUCTION

The ST7 enhanced interrupt management provides the following features:

- Hardware interrupts
- Software interrupt (TRAP)
- Nested or concurrent interrupt management with flexible interrupt priority and level management:
  - Up to 4 software programmable nesting levels
  - Up to 16 interrupt vectors fixed by hardware
  - 2 non maskable events: RESET, TRAP

This interrupt management is based on:

– Bit 5 and bit 3 of the CPU CC register (I1:0),

– Interrupt software priority registers (ISPRx),

– Fixed interrupt vector addresses located at the high addresses of the memory map (FFE0h to FFFFh) sorted by hardware priority order.

This enhanced interrupt controller guarantees full upward compatibility with the standard (not nested) ST7 interrupt controller.

## 7.2 MASKING AND PROCESSING FLOW

The interrupt masking is managed by the I1 and I0 bits of the CC register and the ISPRx registers which give the interrupt software priority level of each interrupt vector (see Table 6). The processing flow is shown in Figure 17

When an interrupt request 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.

– I1 and I0 bits of CC register are set according to the corresponding values in the ISPRx registers of the serviced interrupt vector.

– 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 “Interrupt Mapping” table for vector addresses).

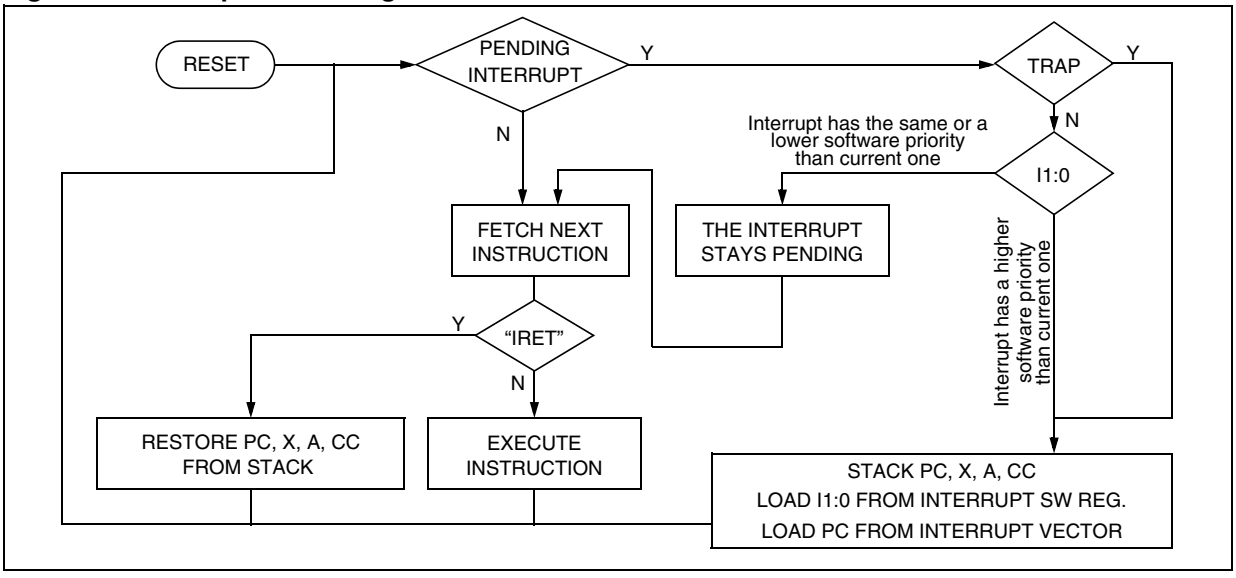
The interrupt service routine should end with the IRET instruction which causes the contents of the saved registers to be recovered from the stack.

**Note**: As a consequence of the IRET instruction, the I1 and I0 bits will be restored from the stack and the program in the previous level will resume.

**Table 6. Interrupt Software Priority Levels**

| Interrupt software priority | Level | I1 | I0 |
|---|---|---|---|
| Level 0 (main) | Low | 1 | 0 |
| Level 1 | ▼ | 0 | 1 |
| Level 2 | | 0 | 0 |
| Level 3 (= interrupt disable) | High | 1 | 1 |

**Figure 17. Interrupt Processing Flowchart**



**INTERRUPTS** (Cont'd)

**Table 7. Dedicated Interrupt Instruction Set**

| Instruction | New Description | Function/Example | I1 | H | I0 | N | Z | C |
|---|---|---|---|---|---|---|---|---|
| HALT | Entering Halt mode | | 1 | | 0 | | | |
| IRET | Interrupt routine return | Pop CC, A, X, PC | I1 | H | I0 | N | Z | C |
| JRM | Jump if I1:0=11 (level 3) | I1:0=11? | | | | | | |
| JRNM | Jump if I1:0<>11 | I1:0<>11? | | | | | | |
| POP CC | Pop CC from the Stack | Mem => CC | I1 | H | I0 | N | Z | C |
| RIM | Enable interrupt (level 0 set) | Load 10 in I1:0 of CC | 1 | | 0 | | | |
| SIM | Disable interrupt (level 3 set) | Load 11 in I1:0 of CC | 1 | | 1 | | | |
| TRAP | Software trap | Software NMI | 1 | | 1 | | | |
| WFI | Wait for interrupt | | 1 | | 0 | | | |

**Note**: During the execution of an interrupt routine, the HALT, POPCC, RIM, SIM and WFI instructions change the current software priority up to the next IRET instruction or one of the previously mentioned instructions.

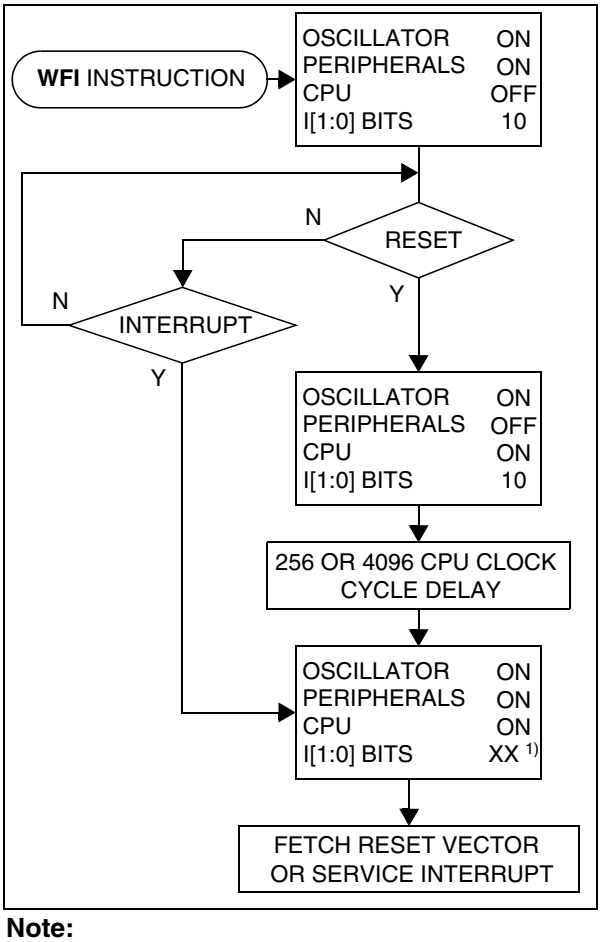
**POWER SAVING MODES** (Cont'd)

### 8.3 WAIT MODE

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

Refer to Figure 24.

**Figure 24. WAIT Mode Flow-chart**




**Note:**

1. Before servicing an interrupt, the CC register is pushed on the stack. The I[1:0] bits of the CC register are set to the current software priority level of the interrupt routine and recovered when the CC register is popped.

**POWER SAVING MODES** (Cont'd)

### 8.4.2 HALT MODE

The HALT mode is the lowest power consumption mode of the MCU. It is entered by executing the 'HALT' instruction when the OIE bit of the Main Clock Controller Status register (MCCSR) is cleared (see Section 10.2 on page 56 for more details on the MCCSR register).

The MCU can exit HALT mode on reception of either a specific interrupt (see Table 8, "Interrupt Mapping," on page 36) or a RESET. When exiting HALT mode by means of a RESET or an interrupt, the oscillator is immediately turned on and the 256 or 4096 CPU cycle delay is used to stabilize the oscillator. After the start up delay, the CPU resumes operation by servicing the interrupt or by fetching the reset vector which woke it up (see Figure 28).

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

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

The compatibility of Watchdog operation with HALT mode is configured by the "WDGHALT" option bit of the option byte. The HALT instruction when executed while the Watchdog system is enabled, can generate a Watchdog RESET (see Section 14.1 on page 150) for more details.

**Figure 27. HALT Timing Overview**

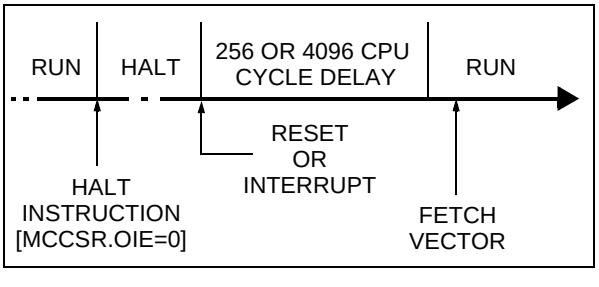
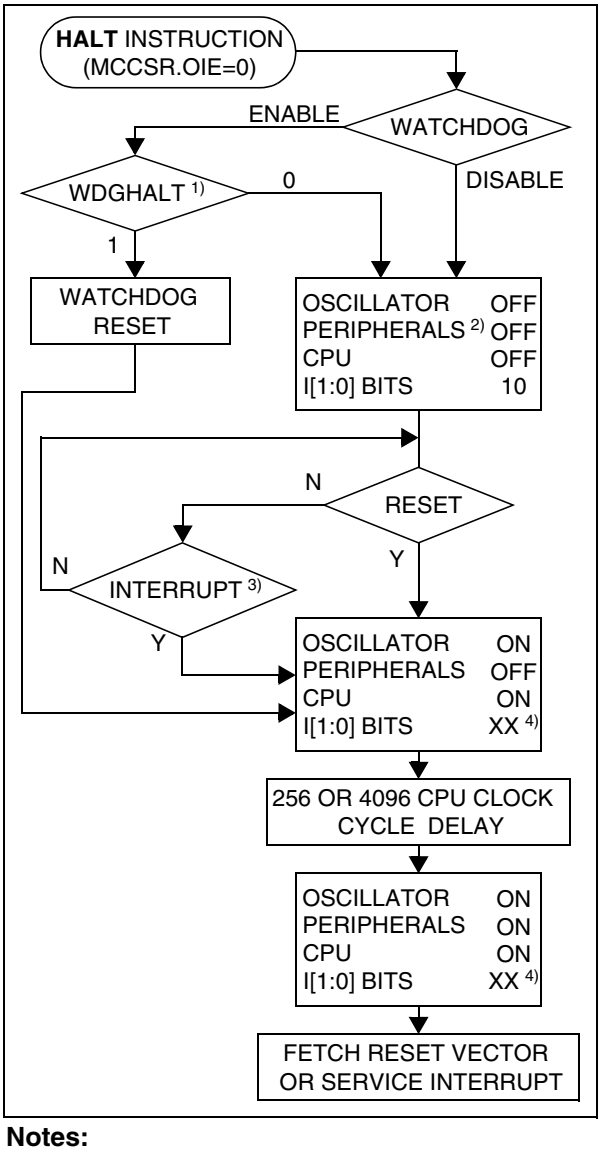



**Figure 28. HALT Mode Flow-chart**




**Notes:**

1. WDGHALT is an option bit. See option byte section for more details.
2. Peripheral clocked with an external clock source can still be active.
3. Only some specific interrupts can exit the MCU from HALT mode (such as external interrupt). Refer to Table 8, "Interrupt Mapping," on page 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.

**POWER SAVING MODES** (Cont'd)

#### 8.4.2.1 Halt Mode Recommendations

– Make sure that an external event is available to wake up the microcontroller from Halt mode.

– 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 sensitiveness 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 interrupt mask 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).

**16-BIT TIMER** (Cont'd)

### 10.3.3.4 Output Compare

In this section, the index, *i*, may be 1 or 2 because there are 2 output compare functions in the 16-bit timer.

This function can be used to control an output waveform or indicate when a period of time has elapsed.

When a match is found between the Output Compare register and the free running counter, the output compare function:

- Assigns pins with a programmable value if the OC*i*E bit is set
- Sets a flag in the status register
- Generates an interrupt if enabled

Two 16-bit registers Output Compare Register 1 (OC1R) and Output Compare Register 2 (OC2R) contain the value to be compared to the counter register each timer clock cycle.

| | MS Byte | LS Byte |
|---|---|---|
| OC*i*R | OC*i*HR | OC*i*LR |

These registers are readable and writable and are not affected by the timer hardware. A reset event changes the OC*i*R value to 8000h.

Timing resolution is one count of the free running counter: ($f_{CPU/CC[1:0]}$).

**Procedure:**

To use the output compare function, select the following in the CR2 register:

- Set the OC*i*E bit if an output is needed then the OCMP*i* pin is dedicated to the output compare *i* signal.
- Select the timer clock (CC[1:0]) (see Table 16 Clock Control Bits).

And select the following in the CR1 register:

- Select the OLVL*i* bit to applied to the OCMP*i* pins after the match occurs.
- Set the OCIE bit to generate an interrupt if it is needed.

When a match is found between OCRi register and CR register:

- OCF*i* bit is set.
- The OCMP*i* pin takes OLVL*i* bit value (OCMP*i* pin latch is forced low during reset).
- A timer interrupt is generated if the OCIE bit is set in the CR1 register and the I bit is cleared in the CC register (CC).

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

$$\Delta \text{ OC}i\text{R} = \frac{\Delta t * f_{CPU}}{\text{PRESC}}$$

Where:

$\Delta t$ = Output compare period (in seconds)

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

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

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

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

Where:

$\Delta t$ = Output compare period (in seconds)

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

Clearing the output compare interrupt request (i.e. 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).

**16-BIT TIMER** (Cont'd)

**CONTROL/STATUS REGISTER (CSR)**

Read Only (except bit 2 R/W)

Reset Value: xxxx x0xx (xxh)

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| ICF1 | OCF1 | TOF | ICF2 | OCF2 | TIMD | 0 | 0 |

Bit 7 = **ICF1** *Input Capture Flag 1.*
0: No input capture (reset value).
1: An input capture has occurred on the ICAP1 pin or the counter has reached the OC2R value in PWM mode. To clear this bit, first read the SR register, then read or write the low byte of the IC1R (IC1LR) register.

Bit 6 = **OCF1** *Output Compare Flag 1.*
0: No match (reset value).
1: The content of the free running counter has matched the content of the OC1R register. To clear this bit, first read the SR register, then read or write the low byte of the OC1R (OC1LR) register.

Bit 5 = **TOF** *Timer Overflow Flag.*
0: No timer overflow (reset value).
1: The free running counter rolled over from FFFFh to 0000h. To clear this bit, first read the SR register, then read or write the low byte of the CR (CLR) register.

**Note:** Reading or writing the ACLR register does not clear TOF.

Bit 4 = **ICF2** *Input Capture Flag 2.*
0: No input capture (reset value).
1: An input capture has occurred on the ICAP2 pin. To clear this bit, first read the SR register, then read or write the low byte of the IC2R (IC2LR) register.

**Note:** In Flash devices, this bit is not available for Timer A and is forced by hardware to 0.

Bit 3 = **OCF2** *Output Compare Flag 2.*
0: No match (reset value).
1: The content of the free running counter has matched the content of the OC2R register. To clear this bit, first read the SR register, then read or write the low byte of the OC2R (OC2LR) register.

**Note:** In Flash devices, this bit is not available for Timer A and is forced by hardware to 0.

Bit 2 = **TIMD** *Timer disable.*
This bit is set and cleared by software. When set, it freezes the timer prescaler and counter and disabled the output functions (OCMP1 and OCMP2 pins) to reduce power consumption. Access to the timer registers is still available, allowing the timer configuration to be changed, or the counter reset, while it is disabled.
0: Timer enabled
1: Timer prescaler, counter and outputs disabled

Bits 1:0 = Reserved, must be kept cleared.

**16-BIT TIMER** (Cont'd)

**OUTPUT COMPARE 2 HIGH REGISTER (OC2HR)**

Read/Write
Reset Value: 1000 0000 (80h)

This is an 8-bit register that contains the high part of the value to be compared to the CHR register.

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| MSB | | | | | | | LSB |

**Note:** In Flash devices, the Timer A OC2HR register is write-only.

**OUTPUT COMPARE 2 LOW REGISTER (OC2LR)**

Read/Write
Reset Value: 0000 0000 (00h)

This is an 8-bit register that contains the low part of the value to be compared to the CLR register.

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| MSB | | | | | | | LSB |

**Note:** In Flash devices, the Timer A OC2LR register is write-only.

**COUNTER HIGH REGISTER (CHR)**

Read Only
Reset Value: 1111 1111 (FFh)

This is an 8-bit register that contains the high part of the counter value.

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| MSB | | | | | | | LSB |

**COUNTER LOW REGISTER (CLR)**

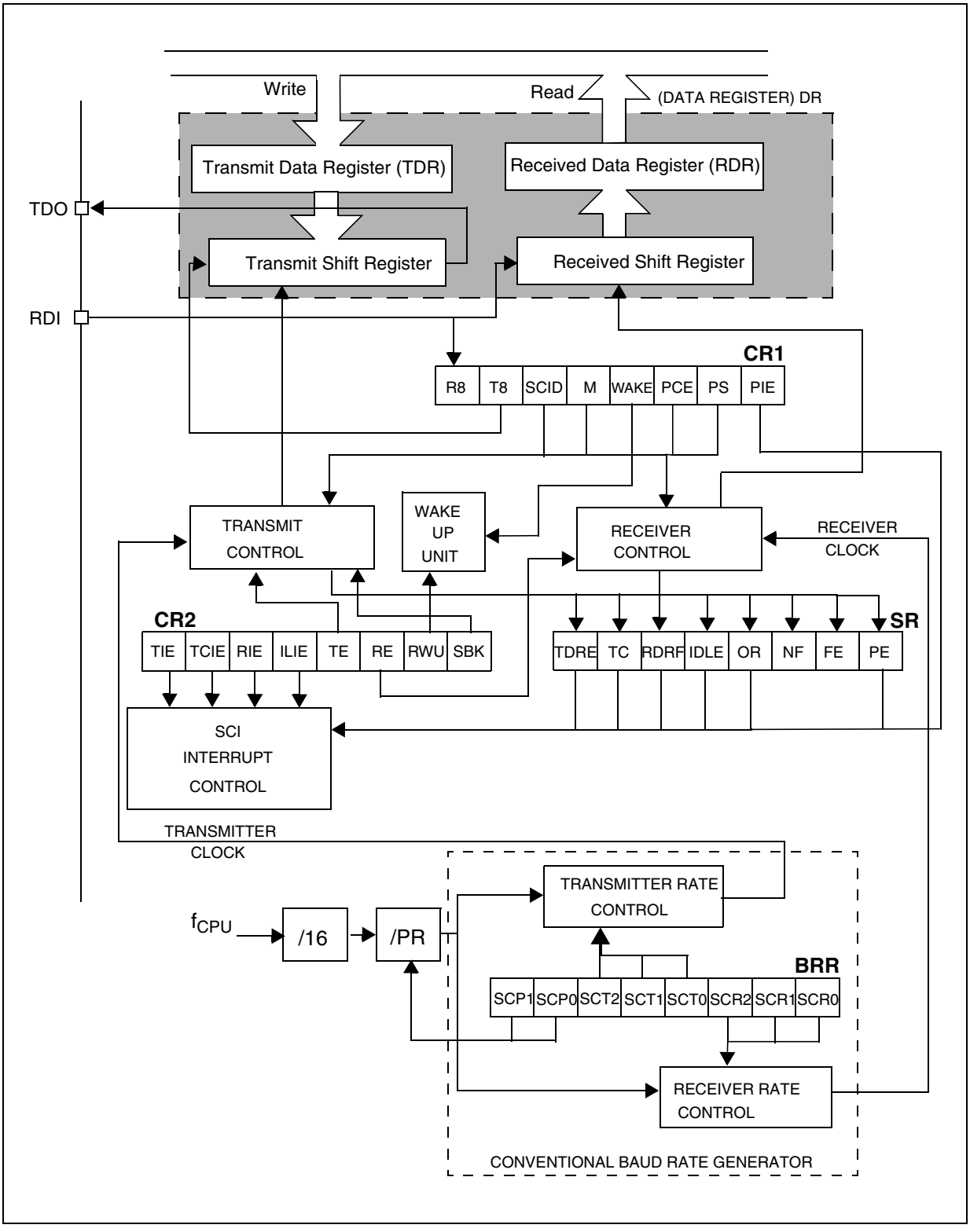
Read Only
Reset Value: 1111 1100 (FCh)

This is an 8-bit register that contains the low part of the counter value. A write to this register resets the counter. An access to this register after accessing the CSR register clears the TOF bit.

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| MSB | | | | | | | LSB |

**SERIAL COMMUNICATIONS INTERFACE** (Cont'd)

**Figure 53. SCI Block Diagram**



**SERIAL COMMUNICATIONS INTERFACE** (Cont'd)

### 10.5.4.9 Clock Deviation Causes

The causes which contribute to the total deviation are:

- $D_{TRA}$: Deviation due to transmitter error (Local oscillator error of the transmitter or the transmitter is transmitting at a different baud rate).
- $D_{QUANT}$: Error due to the baud rate quantization of the receiver.
- $D_{REC}$: Deviation of the local oscillator of the receiver: This deviation can occur during the reception of one complete SCI message assuming that the deviation has been compensated at the beginning of the message.
- $D_{TCL}$: Deviation due to the transmission line (generally due to the transceivers)

All the deviations of the system should be added and compared to the SCI clock tolerance:

$D_{TRA} + D_{QUANT} + D_{REC} + D_{TCL} < 3.75\%$

### 10.5.4.10 Noise Error Causes

See also description of Noise error in Section 0.1.4.3 .

**Start bit**

The noise flag (NF) is set during start bit reception if one of the following conditions occurs:

1. A valid falling edge is not detected. A falling edge is considered to be valid if the 3 consecutive samples before the falling edge occurs are detected as '1' and, after the falling edge occurs, during the sampling of the 16 samples, if one of the samples numbered 3, 5 or 7 is detected as a "1".
2. During sampling of the 16 samples, if one of the samples numbered 8, 9 or 10 is detected as a "1".

Therefore, a valid Start Bit must satisfy both the above conditions to prevent the Noise Flag getting set.

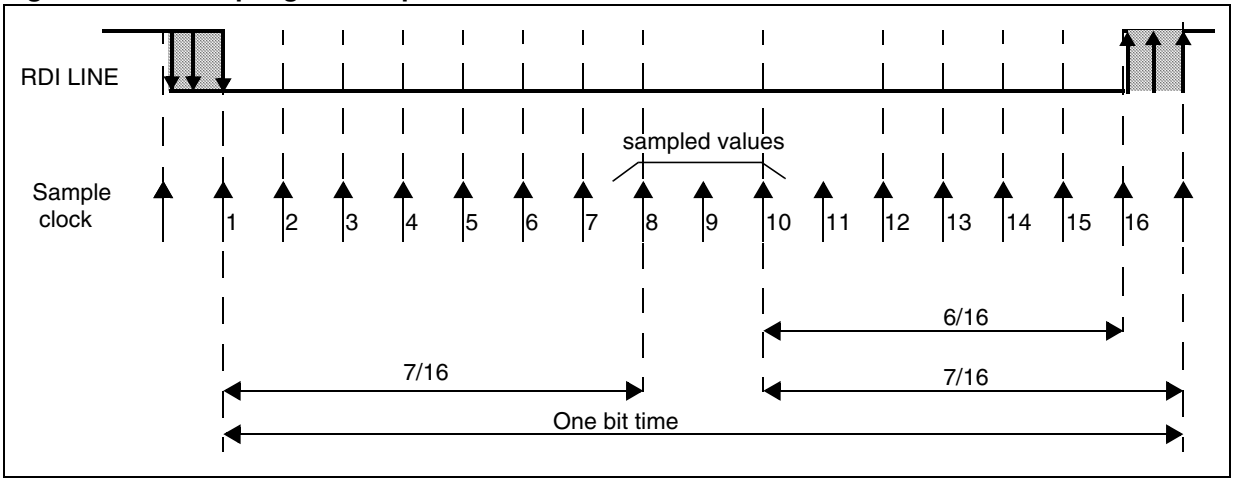
**Data Bits**

The noise flag (NF) is set during normal data bit reception if the following condition occurs:

- During the sampling of 16 samples, if all three samples numbered 8, 9 and10 are not the same. The majority of the 8th, 9th and 10th samples is considered as the bit value.

Therefore, a valid Data Bit must have samples 8, 9 and 10 at the same value to prevent the Noise Flag getting set.

**Figure 56. Bit Sampling in Reception Mode**



**SERIAL COMMUNICATIONS INTERFACE** (Cont'd)

**CONTROL REGISTER 2 (SCICR2)**

Read/Write

Reset Value: 0000 0000 (00h)

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| TIE | TCIE | RIE | ILIE | TE | RE | RWU | SBK |

Bit 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

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.

**CAUTION:** The TDO pin is free for general purpose I/O only when the TE and RE bits are both cleared (or if TE is never set).

Bit 2 = **RE** *Receiver enable.*
This bit enables the receiver. It is set and cleared by software.
0: Receiver is disabled
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

**Note:** Before selecting Mute mode (setting the RWU bit), the SCI must receive some data first, otherwise it cannot function in Mute mode with wake-up by idle line detection.

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 sends a BREAK word at the end of the current word.

**SERIAL COMMUNICATION INTERFACE** (Cont'd)

**Table 22. SCI Register Map and Reset Values**

| Address (Hex.) | Register Label | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0050h | **SCISR** | TDRE | TC | RDRF | IDLE | OR | NF | FE | PE |
| | Reset Value | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0051h | **SCIDR** | MSB | | | | | | | LSB |
| | Reset Value | x | x | x | x | x | x | x | x |
| 0052h | **SCIBRR** | SCP1 | SCP0 | SCT2 | SCT1 | SCT0 | SCR2 | SCR1 | SCR0 |
| | Reset Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0053h | **SCICR1** | R8 | T8 | SCID | M | WAKE | PCE | PS | PIE |
| | Reset Value | x | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0054h | **SCICR2** | TIE | TCIE | RIE | ILIE | TE | RE | RWU | SBK |
| | Reset Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0055h | **SCIERPR** | MSB | | | | | | | LSB |
| | Reset Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0057h | **SCIPETPR** | MSB | | | | | | | LSB |
| | Reset Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 10.6 10-BIT A/D CONVERTER (ADC)

### 10.6.1 Introduction

The on-chip Analog to Digital Converter (ADC) peripheral is a 10-bit, successive approximation converter with internal sample and hold circuitry. This peripheral has up to 16 multiplexed analog input channels (refer to device pin out description) that allow the peripheral to convert the analog voltage levels from up to 16 different sources.

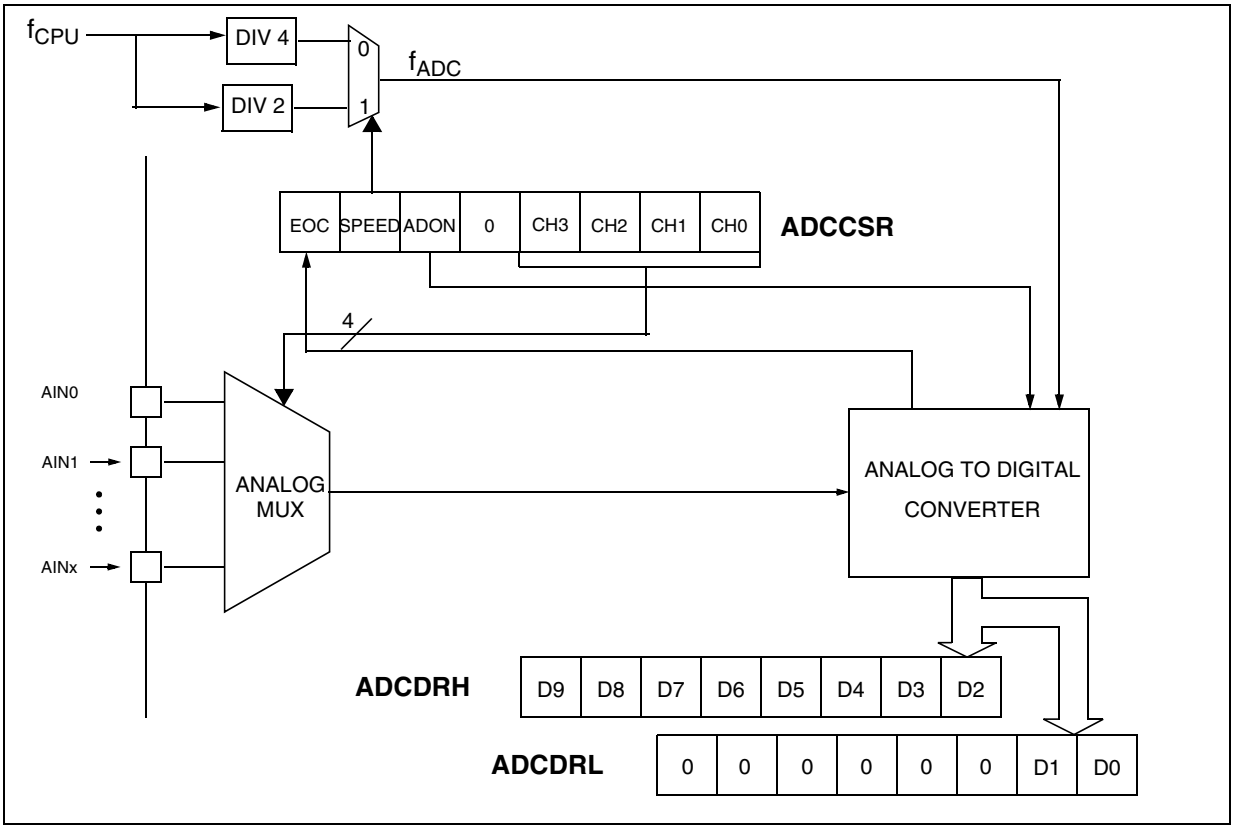
The result of the conversion is stored in a 10-bit Data Register. The A/D converter is controlled through a Control/Status Register.

### 10.6.2 Main Features

- 10-bit conversion
- Up to 16 channels with multiplexed input
- Linear successive approximation
- Data register (DR) which contains the results
- Conversion complete status flag
- On/off bit (to reduce consumption)

The block diagram is shown in Figure 57.

**Figure 57. ADC Block Diagram**



**CLOCK CHARACTERISTICS** (Cont'd)

### 12.6.5 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)] | Flash ST72F324, $f_{OSC}$ = 4 MHz. | | 1.0 | 2.5 | % |
| | | Flash ST72F324, $f_{OSC}$ = 2 MHz. | | 2.5 | 4.0 | |

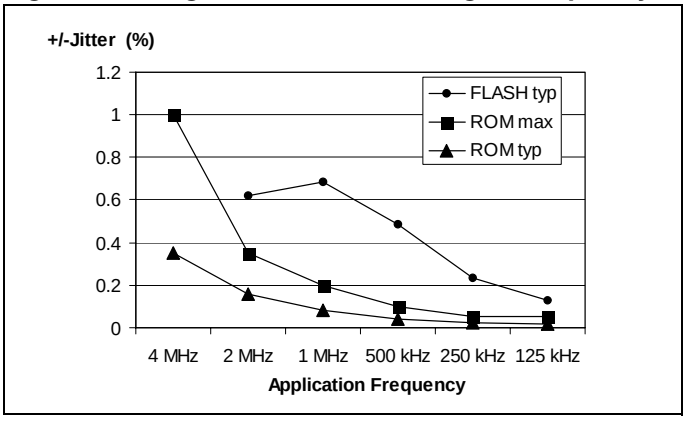
**Note:**

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

**Figure 68. Integrated PLL Jitter vs signal frequency**[1]




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

**EMC CHARACTERISTICS** (Cont'd)

### 12.8.2 Electro Magnetic Interference (EMI)

Based on a simple application running on the product (toggling 2 LEDs through the I/O ports), the product is monitored in terms of emission. This emission test is in line with the norm SAE J 1752/ 3 which specifies the board and the loading of each pin.

| Symbol | Parameter | Conditions | Device/ Package | Monitored Frequency Band | Max vs. [$f_{OSC}/f_{CPU}$] | | Unit |
|---|---|---|---|---|---|---|---|
| | | | | | 8/4MHz | 16/8MHz | |
| $S_{EMI}$ | Peak level | $V_{DD}$=5V, $T_A$=+25°C conforming to SAE J 1752/3 | 8/16K Flash/ TQFP44 | 0.1MHz to 30MHz | 12 | 18 | dBµV |
| | | | | 30MHz to 130MHz | 19 | 25 | |
| | | | | 130MHz to 1GHz | 15 | 22 | |
| | | | | SAE EMI Level | 3 | 3.5 | - |
| | | | 32K Flash/TQFP44 | 0.1MHz to 30MHz | 20 | 21 | dBµV |
| | | | | 30MHz to 130MHz | 26 | 31 | |
| | | | | 130MHz to 1GHz | 22 | 28 | |
| | | | | SAE EMI Level | 3.5 | 4.0 | - |
| | | | Flash/TQFP32 | 0.1MHz to 30MHz | 25 | 27 | dBµV |
| | | | | 30MHz to 130MHz | 30 | 36 | |
| | | | | 130MHz to 1GHz | 18 | 23 | |
| | | | | SAE EMI Level | 3.0 | 3.5 | - |

**Notes:**

1. Data based on characterization results, not tested in production.
2. Refer to Application Note AN1709 for data on other package types.

# 15 KNOWN LIMITATIONS

## 15.1 ALL DEVICES

### 15.1.1 External RC option

The External RC clock source option described in previous datasheet revisions is no longer supported and has been removed from this specification.

### 15.1.2 CSS Function

The Clock Security System function has been removed from the datasheet.

### 15.1.3 Safe Connection of OSC1/OSC2 Pins

The OSC1 and/or OSC2 pins must not be left unconnected otherwise the ST7 main oscillator may start and, in this configuration, could generate an $f_{OSC}$ clock frequency in excess of the allowed maximum (>16MHz.), putting the ST7 in an unsafe/undefined state. Refer to Section 6.2 on page 24.

### 15.1.4 Unexpected Reset Fetch

If an interrupt request occurs while a “POP CC” instruction is executed, the interrupt controller does not recognise the source of the interrupt and, by default, passes the RESET vector address to the CPU.

**Workaround**

To solve this issue, a “POP CC” instruction must always be preceded by a “SIM” instruction.

### 15.1.5 Clearing active interrupts outside interrupt routine

When an active interrupt request occurs at the same time as the related flag is being cleared, an unwanted reset may occur.

**Note:** clearing the related interrupt mask will not generate an unwanted reset

**Concurrent interrupt context**

The symptom does not occur when the interrupts are handled normally, i.e.

when:

– The interrupt flag is cleared within its own interrupt routine

– The interrupt flag is cleared within any interrupt routine

– The interrupt flag is cleared in any part of the code while this interrupt is disabled

If these conditions are not met, the symptom can be avoided by implementing the following sequence:

Perform SIM and RIM operation before and after resetting an active interrupt request.

Example:

SIM

reset interrupt flag

RIM

**Nested interrupt context:**

The symptom does not occur when the interrupts are handled normally, i.e.

when:

– The interrupt flag is cleared within its own interrupt routine

– The interrupt flag is cleared within any interrupt routine with higher or identical priority level

– The interrupt flag is cleared in any part of the code while this interrupt is disabled

If these conditions are not met, the symptom can be avoided by implementing the following sequence:

PUSH CC

SIM

reset interrupt flag

POP CC

### 15.1.6 External Interrupt Missed

To avoid any risk of generating a parasitic interrupt, the edge detector is automatically disabled for one clock cycle during an access to either DDR and OR. Any input signal edge during this period will not be detected and will not generate an interrupt.

This case can typically occur if the application refreshes the port configuration registers at intervals during runtime.

**Workaround**

The workaround is based on software checking the level on the interrupt pin before and after writing to the PxOR or PxDDR registers. If there is a level change (depending on the sensitivity programmed for this pin) the interrupt routine is invoked using the call instruction with three extra PUSH instructions before executing the interrupt routine (this is to make the call compatible with the IRET instruction at the end of the interrupt service routine).

But detection of the level change does ensure that edge occurs during the critical 1 cycle duration and the interrupt has been missed. This may lead to occurrence of same interrupt twice (one hardware and another with software call).