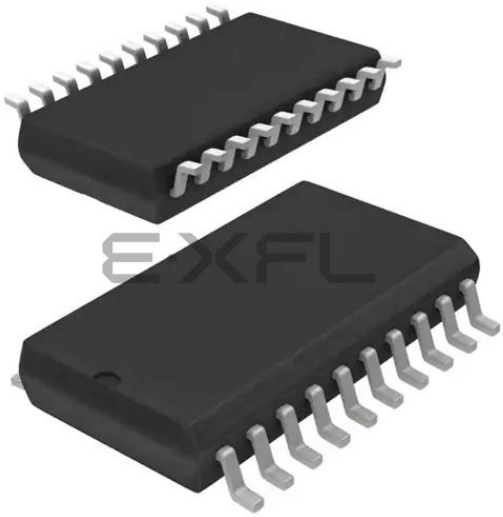


Welcome to E-XFL.COM

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	SPI
Peripherals	LVD, POR, PWM, WDT
Number of I/O	15
Program Memory Size	4KB (4K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	256 x 8
Voltage - Supply (Vcc/Vdd)	2.4V ~ 5.5V
Data Converters	A/D 7x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	20-SOIC (0.295", 7.50mm Width)
Supplier Device Package	20-SO
Purchase URL	https://www.e-xfl.com/product-detail/stmicroelectronics/st7flite15f1b6

2 PIN DESCRIPTION

Figure 2. 20-Pin SO Package Pinout

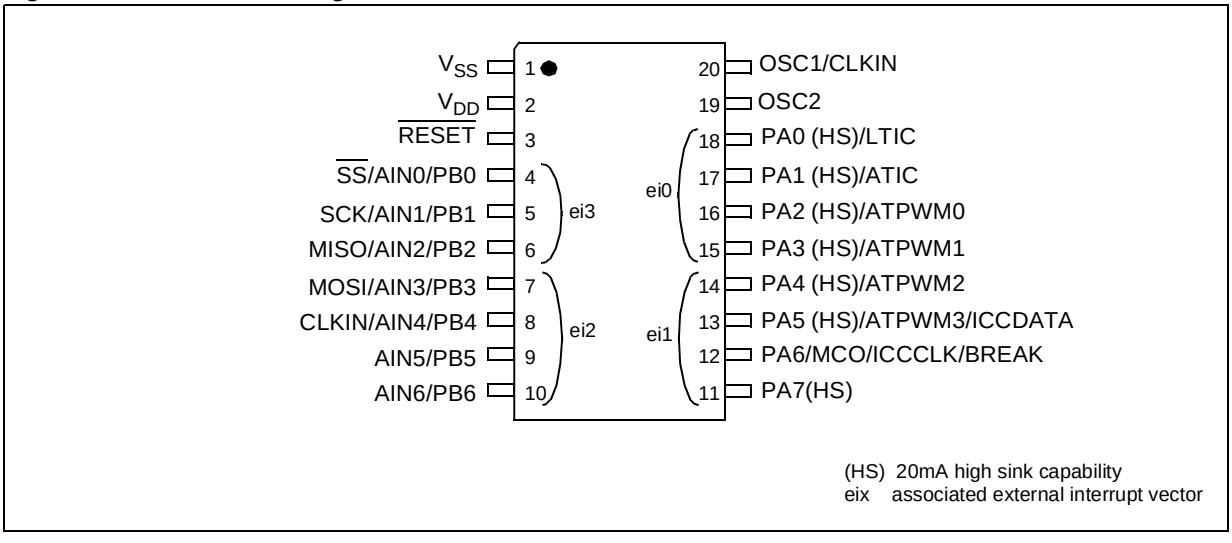
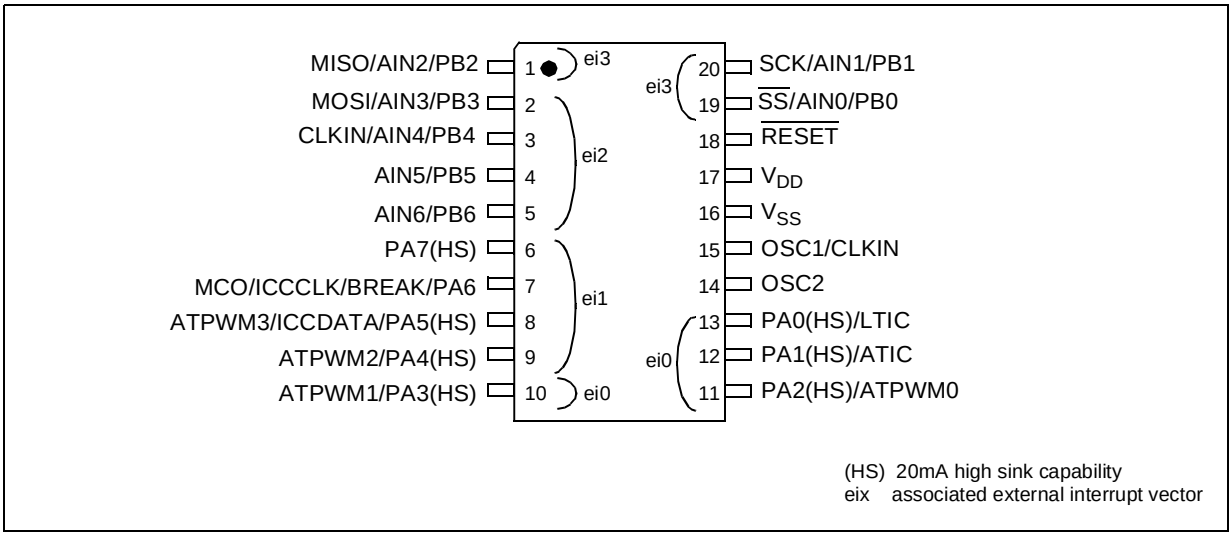


Figure 3. 20-Pin DIP Package Pinout



DATA EEPROM (Cont'd)

5.7 REGISTER DESCRIPTION

EEPROM CONTROL/STATUS REGISTER (EECSR)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
0	0	0	0	0	0	E2LAT	E2PGM

Bits 7:2 = Reserved, forced by hardware to 0.

Bit 1 = **E2LAT** *Latch Access Transfer*
This bit is set by software. It is cleared by hardware at the end of the programming cycle. It can only be cleared by software if the E2PGM bit is cleared.
0: Read mode
1: Write mode

Bit 0 = **E2PGM** *Programming control and status*
This bit is set by software to begin the programming cycle. At the end of the programming cycle, this bit is cleared by hardware.
0: Programming finished or not yet started
1: Programming cycle is in progress

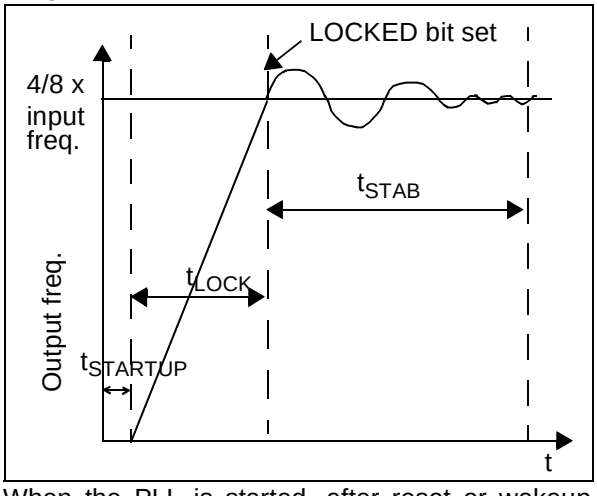
Note: if the E2PGM bit is cleared during the programming cycle, the memory data is not guaranteed

Table 3. DATA EEPROM Register Map and Reset Values

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
0030h	**EECSR**							E2LAT	E2PGM
	Reset Value	0	0	0	0	0	0	0	0

PHASE LOCKED LOOP (Cont'd)

Figure 12. PLL Output Frequency Timing Diagram



When the PLL is started, after reset or wakeup from Halt mode or AWUFH mode, it outputs the clock after a delay of $t_{STARTUP}$.

When the PLL output signal reaches the operating frequency, the LOCKED bit in the SICSCR register is set. Full PLL accuracy (ACC_{PLL}) is reached after a stabilization time of t_{STAB} (see Figure 12 and 13.3.4 Internal RC Oscillator and PLL)

Refer to section 7.6.4 on page 33 for a description of the LOCKED bit in the SICSR register.

7.3 REGISTER DESCRIPTION

MAIN CLOCK CONTROL/STATUS REGISTER (MCCSR)

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

7							0
0	0	0	0	0	0	MCO	SMS

Bits 7:2 = Reserved, must be kept cleared.

Bit 1 = **MCO** *Main Clock Out enable*
This bit is read/write by software and cleared by hardware after a reset. This bit allows to enable the MCO output clock.
0: MCO clock disabled, I/O port free for general purpose I/O.
1: MCO clock enabled.

Bit 0 = **SMS** *Slow Mode select*
This bit is read/write by software and cleared by hardware after a reset. This bit selects the input clock f_{OSC} or $f_{OSC}/32$.
0: Normal mode ($f_{CPU} = f_{OSC}$
1: Slow mode ($f_{CPU} = f_{OSC}/32$)

RC CONTROL REGISTER (RCCR)

Read / Write
Reset Value: 1111 1111 (FFh)

7							0
CR70	CR60	CR50	CR40	CR30	CR20	CR10	CR0

Bits 7:0 = **CR[7:0]** *RC Oscillator Frequency Adjustment Bits*
These bits must be written immediately after reset to adjust the RC oscillator frequency and to obtain an accuracy of 1%. The application can store the correct value for each voltage range in EEPROM and write it to this register at start-up.
00h = maximum available frequency
FFh = lowest available frequency

Note: To tune the oscillator, write a series of different values in the register until the correct frequency is reached. The fastest method is to use a dichotomy starting with 80h.

7.6 SYSTEM INTEGRITY MANAGEMENT (SI)

The System Integrity Management block contains the Low voltage Detector (LVD) and Auxiliary Voltage Detector (AVD) functions. It is managed by the SICSR register.

7.6.1 Low Voltage Detector (LVD)

The Low Voltage Detector function (LVD) generates a static reset when the V_{DD} supply voltage is below a $V_{IT-(LVD)}$ reference value. This means that it secures the power-up as well as the power-down keeping the ST7 in reset.

The $V_{IT-(LVD)}$ reference value for a voltage drop is lower than the $V_{IT+(LVD)}$ reference value for power-on in order to avoid a parasitic reset when the MCU starts running and sinks current on the supply (hysteresis).

The LVD Reset circuitry generates a reset when V_{DD} is below:

– $V_{IT+(LVD)}$when V_{DD} is rising

– $V_{IT-(LVD)}$ when V_{DD} is falling

The LVD function is illustrated in Figure 17.

The voltage threshold can be configured by option byte to be low, medium or high.

Provided the minimum V_{DD} value (guaranteed for the oscillator frequency) is above $V_{IT-(LVD)}$, the MCU can only be in two modes:

– under full software control

– in static safe reset

In these conditions, secure operation is always ensured for the application without the need for external reset hardware.

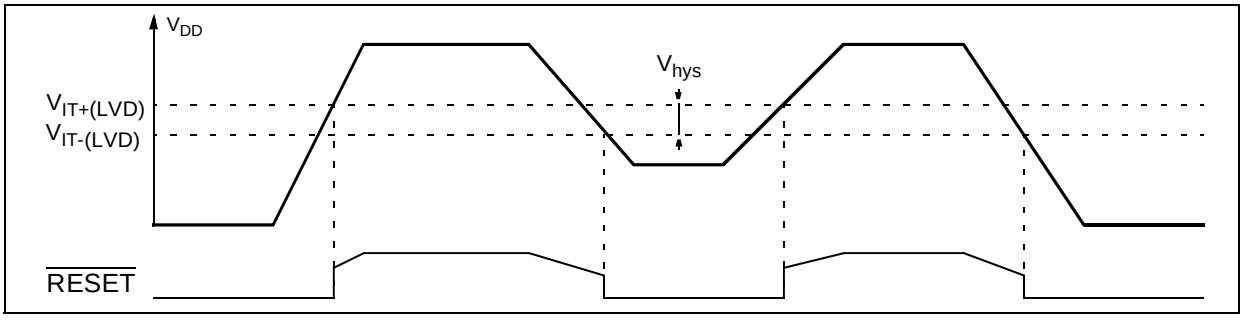
During a Low Voltage Detector Reset, the $\overline{\text{RESET}}$ pin is held low, thus permitting the MCU to reset other devices.

Notes:

The LVD allows the device to be used without any external RESET circuitry.

The LVD is an optional function which can be selected by option byte.

Figure 17. Low Voltage Detector vs Reset



SYSTEM INTEGRITY MANAGEMENT (Cont'd)

7.6.4 Register Description

SYSTEM INTEGRITY (SI) CONTROL/STATUS REGISTER (SICSR)

Read/Write

Reset Value: 0000 0xx0 (0xh)

7 0

0	0	0	WDG RF	LOCKED	LVDRF	AVDF	AVDIE

Bit 7:5 = Reserved, must be kept cleared.

Bit 4 = **WDGRF** *Watchdog reset flag*
This bit indicates that the last Reset was generated by the Watchdog peripheral. It is set by hardware (watchdog reset) and cleared by software (writing zero) or an LVD Reset (to ensure a stable cleared state of the WDGRF flag when CPU starts).
Combined with the LVDRF flag information, the flag description is given by the following table.

RESET Sources	LVDRF	WDGRF
External $\overline{\text{RESET}}$ pin	0	0
Watchdog	0	1
LVD	1	X

Bit 3 = **LOCKED** *PLL Locked Flag*
This bit is set and cleared by hardware. It is set automatically when the PLL reaches its operating frequency.
0: PLL not locked
1: PLL locked

Bit 2 = **LVDRF** *LVD reset flag*
This bit indicates that the last Reset was generated by the LVD block. It is set by hardware (LVD reset) and cleared by software (by reading). When the LVD is disabled by OPTION BYTE, the LVDRF bit value is undefined.

Bit 1 = **AVDF** *Voltage Detector flag*
This read-only bit is set and cleared by hardware. If the AVDIE bit is set, an interrupt request is generated when the AVDF bit is set. Refer to Figure 19 and to Section 7.6.2.1 for additional details.
0: V_{DD} over AVD threshold
1: V_{DD} under AVD threshold

Bit 0 = **AVDIE** *Voltage Detector interrupt enable*
This bit is set and cleared by software. It enables an interrupt to be generated when the AVDF flag is set. The pending interrupt information is automatically cleared when software enters the AVD interrupt routine.
0: AVD interrupt disabled
1: AVD interrupt enabled

Application notes

The LVDRF flag is not cleared when another RESET type occurs (external or watchdog), the LVDRF flag remains set to keep trace of the original failure.
In this case, a watchdog reset can be detected by software while an external reset can not.

8 INTERRUPTS

The ST7 core may be interrupted by one of two different methods: maskable hardware interrupts as listed in the Interrupt Mapping Table and a non-maskable software interrupt (TRAP). The Interrupt processing flowchart is shown in Figure 20.
The maskable interrupts must be enabled by clearing the I bit in order to be serviced. However, disabled interrupts may be latched and processed when they are enabled (see external interrupts subsection).

Note: After reset, all interrupts are disabled.

When an interrupt has to be serviced:

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

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

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

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

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

Note: As a consequence of the IRET instruction, the I bit 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).

8.1 NON MASKABLE SOFTWARE INTERRUPT

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

8.2 EXTERNAL INTERRUPTS

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

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

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

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.

8.3 PERIPHERAL INTERRUPTS

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

– The I bit of the CC register is cleared.

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

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

Clearing an interrupt request is done by:

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

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

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

9 POWER SAVING MODES

9.1 INTRODUCTION

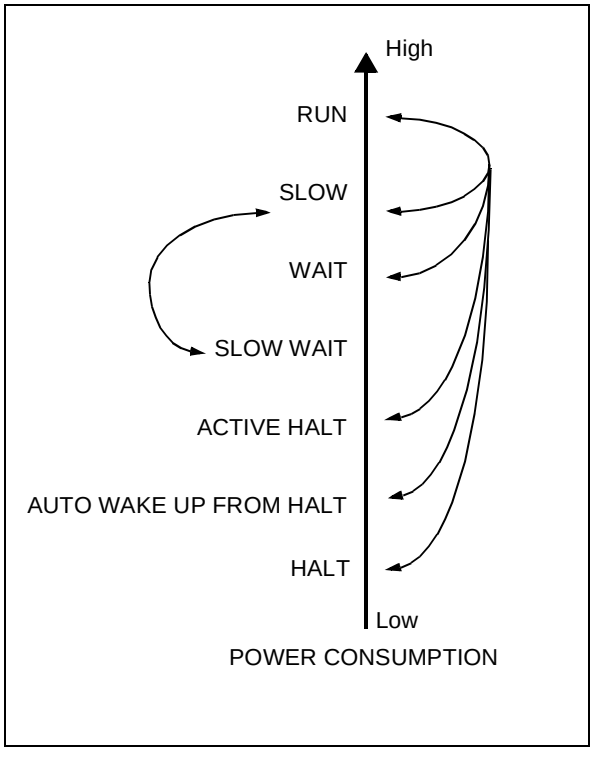
To give a large measure of flexibility to the application in terms of power consumption, five main power saving modes are implemented in the ST7 (see Figure 21):

- Slow
- Wait (and Slow-Wait)
- Active Halt
- Auto Wake up From Halt (AWUFH)
- Halt

After a RESET the normal operating mode is selected by default (RUN mode). This mode drives the device (CPU and embedded peripherals) by means of a master clock which is based on the main oscillator frequency divided or multiplied by 2 (f_{OSC2}).

From RUN mode, the different power saving modes may be selected by setting the relevant register bits or by calling the specific ST7 software instruction whose action depends on the oscillator status.

Figure 21. Power Saving Mode Transitions



9.2 SLOW MODE

This mode has two targets:

– To reduce power consumption by decreasing the internal clock in the device,

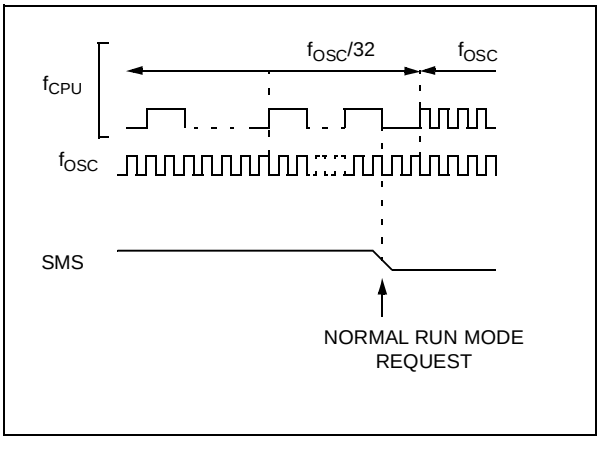
– To adapt the internal clock frequency (f_{CPU}) to the available supply voltage.

SLOW mode is controlled by the SMS bit in the MCCSR register which enables or disables Slow mode.

In this mode, the oscillator frequency is divided by 32. The CPU and peripherals are clocked at this lower frequency.

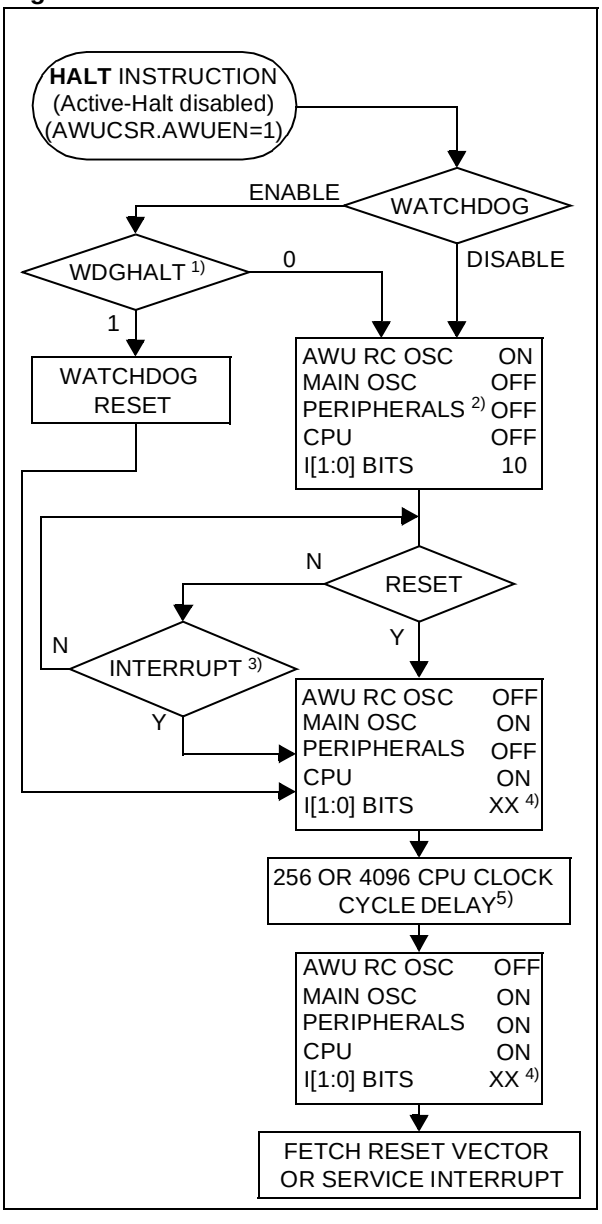
Note: SLOW-WAIT mode is activated when entering WAIT mode while the device is already in SLOW mode.

Figure 22. SLOW Mode Clock Transition



POWER SAVING MODES (Cont'd)

Figure 30. AWUFH Mode Flow-chart



Notes:

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

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

3. Only an AWUFH interrupt and some specific interrupts can exit the MCU from HALT mode (such as external interrupt). Refer to Table 5, "Interrupt Mapping," on page 35 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.

5. If the PLL is enabled by option byte, it outputs the clock after an additional delay of $t_{STARTUP}$ (see Figure 12).

POWER SAVING MODES (Cont'd)

9.6.0.1 Register Description

AWUFH CONTROL/STATUS REGISTER (AWUCSR)
Read/Write
Reset Value: 0000 0000 (00h)

7							0
0	0	0	0	0	AWUF	AWUM	AWUEN

Bits 7:3 = Reserved.

Bit 1= **AWUF** *Auto Wake Up Flag*
This bit is set by hardware when the AWU module generates an interrupt and cleared by software on reading AWUCSR. Writing to this bit does not change its value.
0: No AWU interrupt occurred
1: AWU interrupt occurred

Bit 1= **AWUM** *Auto Wake Up Measurement*
This bit enables the AWU RC oscillator and connects its output to the inputcapture of the 12-bit Auto-Reload timer. This allows the timer to be used to measure the AWU RC oscillator dispersion and then compensate this dispersion by providing the right value in the AWUPRE register.
0: Measurement disabled
1: Measurement enabled

Bit 0 = **AWUEN** *Auto Wake Up From Halt Enabled*
This bit enables the Auto Wake Up From Halt feature: once HALT mode is entered, the AWUFH wakes up the microcontroller after a time delay dependent on the AWU prescaler value. It is set and cleared by software.
0: AWUFH (Auto Wake Up From Halt) mode disabled
1: AWUFH (Auto Wake Up From Halt) mode enabled

AWUFH PRESCALER REGISTER (AWUPR)
Read/Write

7							0
AWUPR7	AWUPR6	AWUPR5	AWUPR4	AWUPR3	AWUPR2	AWUPR1	AWUPR0

Bits 7:0= **AWUPR[7:0]** *Auto Wake Up Prescaler*
These 8 bits define the AWUPR Dividing factor (as explained below:

AWUPR[7:0]	Dividing factor
00h	Forbidden
01h	1
...	...
FEh	254
FFh	255

In AWU mode, the period that the MCU stays in Halt Mode (t_{AWU} in Figure 29 on page 43) is defined by

$$t_{AWU} = 64 \times AWUPR \times \frac{1}{f_{AWURC}} + t_{RCSTRT}$$

This prescaler register can be programmed to modify the time that the MCU stays in Halt mode before waking up automatically.

Note: If 00h is written to AWUPR, depending on the product, an interrupt is generated immediately after a HALT instruction, or the AWUPR remains inchanged.

Table 7. AWU Register Map and Reset Values

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
0049h	**AWUPR** Reset Value	AWUPR7 1	AWUPR6 1	AWUPR5 1	AWUPR4 1	AWUPR3 1	AWUPR2 1	AWUPR1 1	AWUPR0 1
004Ah	**AWUCSR** Reset Value	0	0	0	0	0	AWUF	AWUM	AWUEN

I/O PORTS (Cont'd)

Analog alternate function

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

Analog Recommendations

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

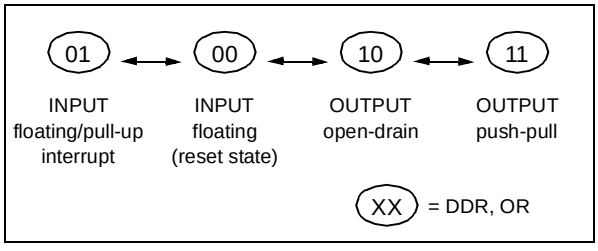
WARNING: The analog input voltage level must be within the limits stated in the absolute maximum ratings.

10.3 I/O PORT IMPLEMENTATION

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

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

Figure 32. Interrupt I/O Port State Transitions



10.4 UNUSED I/O PINS

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

10.5 LOW POWER MODES

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

10.6 INTERRUPTS

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

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

11.2 12-BIT AUTORELOAD TIMER 2 (AT2)

11.2.1 Introduction

The 12-bit Autoreload Timer can be used for general-purpose timing functions. It is based on a free-running 12-bit upcounter with an input capture register and four PWM output channels. There are 6 external pins:

– Four PWM outputs

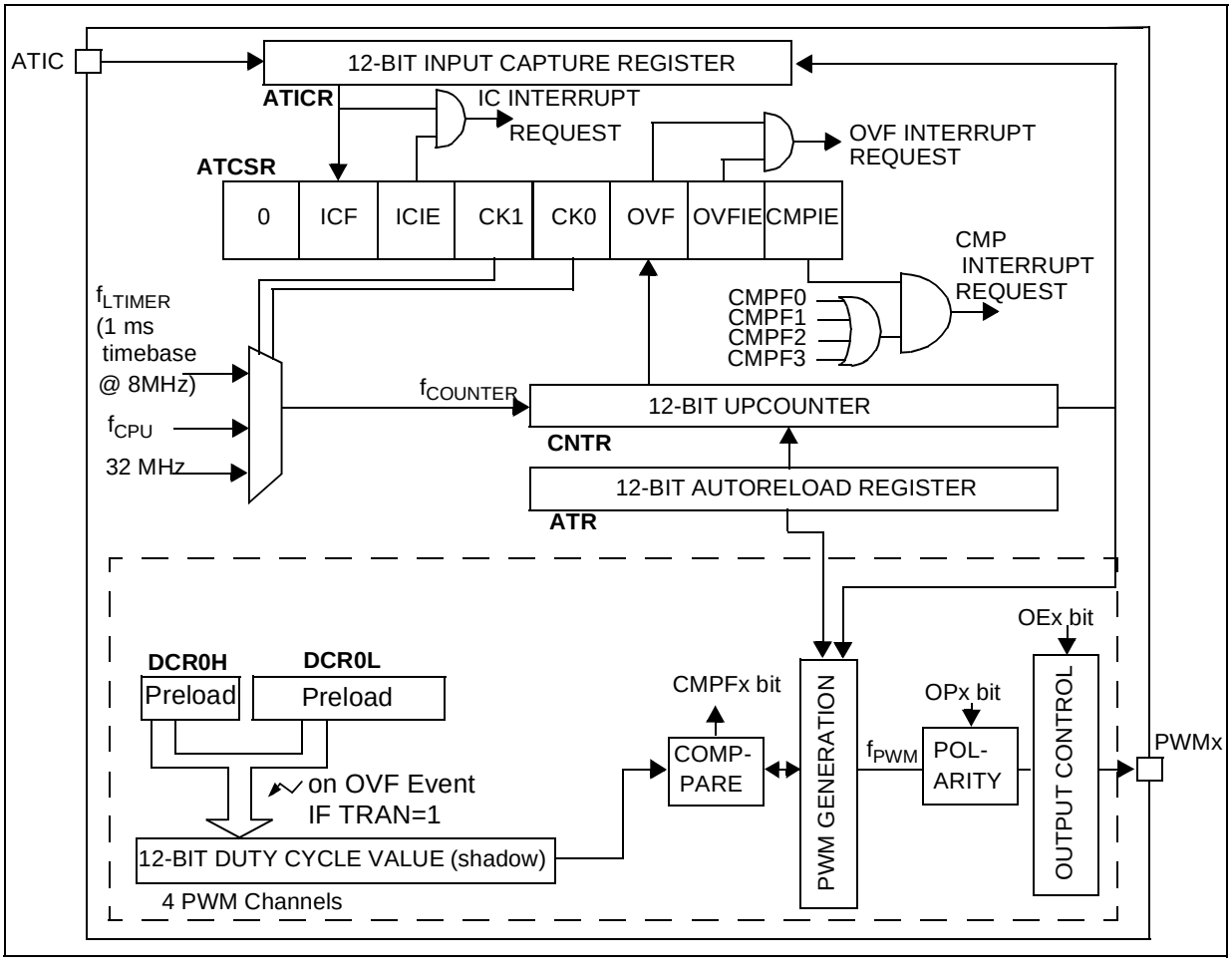
– ATIC pin for the Input Capture function

– BREAK pin for forcing a break condition on the PWM outputs

11.2.2 Main Features

- 12-bit upcounter with 12-bit autoreload register (ATR)
- Maskable overflow interrupt
- Generation of four independent PWMx signals
- Frequency 2KHz-4MHz (@ 8 MHz f_{CPU})
  - Programmable duty-cycles
  - Polarity control
  - Programmable output modes
  - Maskable Compare interrupt
- Input Capture
  - 12-bit input capture register (ATICR)
  - Triggered by rising and falling edges
  - Maskable IC interrupt

Figure 34. Block Diagram



12-BIT AUTORELOAD TIMER (Cont'd)

11.2.4 Low Power Modes

Mode	Description
SLOW	The input frequency is divided by 32
WAIT	No effect on AT timer
ACTIVE-HALT	AT timer halted except if CK0=1, CK1=0 and OVFIE=1
HALT	AT timer halted

11.2.5 Interrupts

Interrupt Event[1)]	Event Flag	Enable Control Bit	Exit from Wait	Exit from Halt	Exit from Active-Halt
Overflow Event	OVF	OVIE	Yes	No	Yes[2)]
IC Event	ICF	ICIE	Yes	No	No
CMP Event	CMPF0	CMPIE	Yes	No	No

Note 1: The CMP and IC events are connected to the same interrupt vector.

The OVF event is mapped on a separate vector (see Interrupts chapter).
They generate an interrupt if the enable bit is set in the ATCSR register and the interrupt mask in the CC register is reset (RIM instruction).

Note 2: Only if CK0=1 and CK1=0

12-BIT AUTORELOAD TIMER (Cont'd)

Table 14. Register Map and Reset Values

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
0D	**ATCSR**	0	ICF	ICIE	CK1	CK0	OVF	OVFIE	CMPIE
	Reset Value		0	0	0	0	0	0	0
0E	**CNTRH**	0	0	0	0	CNTR11	CNTR10	CNTR9	CNTR8
	Reset Value					0	0	0	0
0F	**CNTRL**	CNTR7	CNTR8	CNTR7	CNTR6	CNTR3	CNTR2	CNTR1	CNTR0
	Reset Value	0	0	0	0	0	0	0	0
10	**ATRH**	0	0	0	0	ATR11	ATR10	ATR9	ATR8
	Reset Value					0	0	0	0
11	**ATRL**	ATR7	ATR6	ATR5	ATR4	ATR3	ATR2	ATR1	ATR0
	Reset Value	0	0	0	0	0	0	0	0
12	**PWMCR**	0	OE3	0	OE2	0	OE1	0	OE0
	Reset Value		0		0		0		0
13	**PWM0CSR**	0	0	0	0	0	0	OP0	CMPF0
	Reset Value							0	0
14	**PWM1CSR**	0	0	0	0	0	0	OP1	CMPF1
	Reset Value							0	0
15	**PWM2CSR**	0	0	0	0	0	0	OP2	CMPF2
	Reset Value							0	0
16	**PWM3CSR**	0	0	0	0	0	0	OP3	CMPF3
	Reset Value							0	0
17	**DCR0H**	0	0	0	0	DCR11	DCR10	DCR9	DCR8
	Reset Value					0	0	0	0
18	**DCR0L**	DCR7	DCR6	DCR5	DCR4	DCR3	DCR2	DCR1	DCR0
	Reset Value	0	0	0	0	0	0	0	0
19	**DCR1H**	0	0	0	0	DCR11	DCR10	DCR9	DCR8
	Reset Value					0	0	0	0
1A	**DCR1L**	DCR7	DCR6	DCR5	DCR4	DCR3	DCR2	DCR1	DCR0
	Reset Value	0	0	0	0	0	0	0	0
1B	**DCR2H**	0	0	0	0	DCR11	DCR10	DCR9	DCR8
	Reset Value					0	0	0	0
1C	**DCR2L**	DCR7	DCR6	DCR5	DCR4	DCR3	DCR2	DCR1	DCR0
	Reset Value	0	0	0	0	0	0	0	0
1D	**DCR3H**	0	0	0	0	DCR11	DCR10	DCR9	DCR8
	Reset Value					0	0	0	0
1E	**DCR3L**	DCR7	DCR6	DCR5	DCR4	DCR3	DCR2	DCR1	DCR0
	Reset Value	0	0	0	0	0	0	0	0
1F	**ATICRH**	0	0	0	0	ICR11	ICR10	ICR9	ICR8
	Reset Value					0	0	0	0
20	**ATICRL**	ICR7	ICR6	ICR5	ICR4	ICR3	ICR2	ICR1	ICR0
	Reset Value	0	0	0	0	0	0	0	0

LITE TIMER (Cont'd)

LITE TIMER COUNTER 2 (LTCNTR)
Read only
Reset Value: 0000 0000 (00h)

7							0
CNT7	CNT7	CNT7	CNT7	CNT3	CNT2	CNT1	CNT0

Bits 7:0 = **CNT[7:0]** *Counter 2 Reload Value.*
This register is read by software. The LTARR value is automatically loaded into Counter 2 (LTCNTR) when an overflow occurs.

LITE TIMER CONTROL/STATUS REGISTER (LTCSR1)
Read / Write
Reset Value: 0x00 0000 (x0h)

7							0
ICIE	ICF	TB	TB1IE	TB1F			

Bit 7 = **ICIE** *Interrupt Enable.*
This bit is set and cleared by software.
0: Input Capture (IC) interrupt disabled
1: Input Capture (IC) interrupt enabled

Bit 6 = **ICF** *Input Capture Flag.*
This bit is set by hardware and cleared by software by reading the LTICR register. Writing to this bit does not change the bit value.
0: No input capture
1: An input capture has occurred

Note: After an MCU reset, software must initialise the ICF bit by reading the LTICR register

Bit 5 = **TB** *Timebase period selection.*
This bit is set and cleared by software.
0: Timebase period = t_{OSC} * 8000 (1ms @ 8 MHz)
1: Timebase period = t_{OSC} * 16000 (2ms @ 8 MHz)

Bit 4 = **TB1IE** *Timebase Interrupt enable*.
This bit is set and cleared by software.
0: Timebase (TB1) interrupt disabled
1: Timebase (TB1) interrupt enabled

Bit 3 = **TB1F** *Timebase Interrupt Flag*.
This bit is set by hardware and cleared by software reading the LTCSR register. Writing to this bit has no effect.
0: No counter overflow
1: A counter overflow has occurred

LITE TIMER INPUT CAPTURE REGISTER (LTICR)
Read only
Reset Value: 0000 0000 (00h)

7							0
ICR7	ICR6	ICR5	ICR4	ICR3	ICR2	ICR1	ICR0

Bits 7:0 = **ICR[7:0]** *Input Capture Value*
These bits are read by software and cleared by hardware after a reset. If the ICF bit in the LTCSR is cleared, the value of the 8-bit up-counter will be captured when a rising or falling edge occurs on the LTIC pin.

SERIAL PERIPHERAL INTERFACE (Cont'd)

11.4.6 Low Power Modes

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

11.4.6.1 Using the SPI to wake-up the Device from Halt mode

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

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

Caution: The SPI can wake-up the Device from Halt mode only if the Slave Select signal (external $\overline{SS}$ pin or the SSI bit in the SPICSR register) is low when the Device enters Halt mode. So if Slave selection is configured as external (see Section 11.4.3.2), make sure the master drives a low level on the $\overline{SS}$ pin when the slave enters Halt mode.

11.4.7 Interrupts

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

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

10-BIT A/D CONVERTER (ADC) (Cont'd)

11.5.3.2 Input Voltage Amplifier

The input voltage can be amplified by a factor of 8 by enabling the AMPSEL bit in the ADCDRL register.

When the amplifier is enabled, the input range is 0V to $V_{DD}/8$.

For example, if V_{DD} = 5V, then the ADC can convert voltages in the range 0V to 430mV with an ideal resolution of 0.6mV (equivalent to 13-bit resolution with reference to a V_{SS} to V_{DD} range).

For more details, refer to the Electrical characteristics section.

Note: The amplifier is switched on by the ADON bit in the ADCCSR register, so no additional start-up time is required when the amplifier is selected by the AMPSEL bit.

11.5.3.3 Digital A/D Conversion Result

The conversion is monotonic, meaning that the result never decreases if the analog input does not and never increases if the analog input does not.

If the input voltage (V_{AIN}) is greater than V_{DDA} (high-level voltage reference) then the conversion result is FFh in the ADCDRH register and 03h in the ADCDRL register (without overflow indication).

If the input voltage (V_{AIN}) is lower than V_{SSA} (low-level voltage reference) then the conversion result in the ADCDRH and ADCDRL registers is 00 00h.

The A/D converter is linear and the digital result of the conversion is stored in the ADCDRH and ADCDRL registers. The accuracy of the conversion is described in the Electrical Characteristics Section.

R_{AIN} is the maximum recommended impedance for an analog input signal. If the impedance is too high, this will result in a loss of accuracy due to leakage and sampling not being completed in the alloted time.

11.5.3.4 A/D Conversion

The analog input ports must be configured as input, no pull-up, no interrupt. Refer to the «I/O ports» chapter. Using these pins as analog inputs does not affect the ability of the port to be read as a logic input.

In the ADCCSR register:

– Select the CS[2:0] bits to assign the analog channel to convert.

ADC Conversion mode

In the ADCCSR register:

Set the ADON bit to enable the A/D converter and to start the conversion. From this time on, the ADC performs a continuous conversion of the selected channel.

When a conversion is complete:

– The EOC bit is set by hardware.
– The result is in the ADCDR registers.

A read to the ADCDRH resets the EOC bit.

To read the 10 bits, perform the following steps:

1. Poll EOC bit
2. Read ADCDRL
3. Read ADCDRH. This clears EOC automatically.

To read only 8 bits, perform the following steps:

1. Poll EOC bit
2. Read ADCDRH. This clears EOC automatically.

11.5.4 Low Power Modes

Note: The A/D converter may be disabled by resetting the ADON bit. This feature allows reduced power consumption when no conversion is needed and between single shot conversions.

Mode	Description
WAIT	No effect on A/D Converter
HALT	A/D Converter disabled. After wakeup from Halt mode, the A/D Converter requires a stabilization time t_{STAB} (see Electrical Characteristics) before accurate conversions can be performed.

11.5.5 Interrupts

None.

OPERATING CONDITIONS (Cont'd)

The RC oscillator and PLL characteristics are temperature-dependent and are grouped in four tables.

13.3.4.1 Devices with '"6" order code suffix (tested for T_A = -40 to +85°C) @ V_{DD} = 4.5 to 5.5V

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{RC}	Internal RC oscillator frequency	RCCR = FF (reset value), T_A=25°C,V_{DD}=5V		760		kHz
		RCCR = RCCR0[2],T_A=25°C,V_{DD}=5V		1000		
ACC_{RC}	Accuracy of Internal RC oscillator with RCCR=RCCR0[2]	T_A=25°C,V_{DD}=4.5 to 5.5V	-1		+1	%
		T_A=-40 to +85°C,V_{DD}=5V	-5		+2	%
		T_A=0 to +85°C,V_{DD}=4.5 to 5.5V	-2[1]		+2[1]	%
$I_{DD(RC)}$	RC oscillator current consumption	T_A=25°C,V_{DD}=5V		970[1]		µA
$t_{su(RC)}$	RC oscillator setup time	T_A=25°C,V_{DD}=5V			10[2]	µs
f_{PLL}	x8 PLL input clock		1[1]			MHz
t_{LOCK}	PLL Lock time[5]			2		ms
t_{STAB}	PLL Stabilization time[5]			4		ms
ACC_{PLL}	x8 PLL Accuracy	f_{RC} = 1MHz@T_A=25°C,V_{DD}=4.5 to 5.5V		0.1[4]		%
		f_{RC} = 1MHz@T_A=-40 to +85°C,V_{DD}=5V		0.1[4]		%
$t_{w(JIT)}$	PLL jitter period	f_{RC} = 1MHz		8[3]		kHz
JIT_{PLL}	PLL jitter ($\Delta f_{CPU}/f_{CPU}$)			1[3]		%
$I_{DD(PLL)}$	PLL current consumption	T_A=25°C		600[1]		µA

Notes:

1. Data based on characterization results, not tested in production
2. RCCR0 is a factory-calibrated setting for 1000kHz with ±0.2 accuracy @ T_A =25°C, V_{DD}=5V. See "INTERNAL RC OSCILLATOR ADJUSTMENT" on page 23
3. Guaranteed by design.
4. Averaged over a 4ms period. After the LOCKED bit is set, a period of t_{STAB} is required to reach ACC_{PLL} accuracy.
5. After the LOCKED bit is set ACC_{PLL} is max. 10% until t_{STAB} has elapsed. See Figure 12 on page 24.

13.8 I/O PORT PIN CHARACTERISTICS

13.8.1 General Characteristics

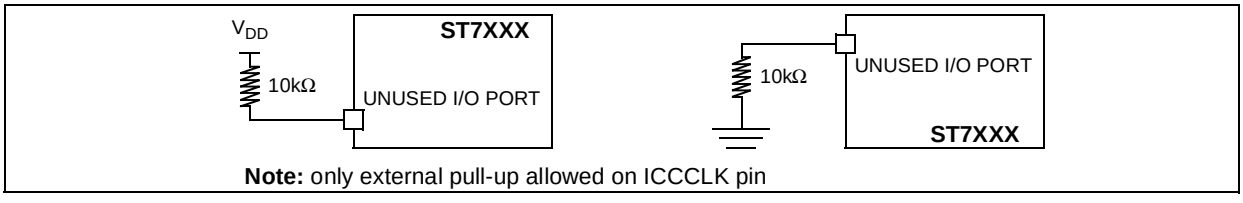
Subject to general operating conditions for V_{DD}, f_{OSC}, and T_A unless otherwise specified.

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
V_{IL}	Input low level voltage					$0.3xV_{DD}$	V
V_{IH}	Input high level voltage			$0.7xV_{DD}$			
V_{hys}	Schmitt trigger voltage hysteresis [1]				400		mV
I_L	Input leakage current	$V_{SS} \le V_{IN} \le V_{DD}$				±1	µA
I_S	Static current consumption [2]	Floating input mode				200	
R_{PU}	Weak pull-up equivalent resistor [3]	$V_{IN}=V_{SS}$	V_{DD}=5V	50	120	250	kΩ
			V_{DD}=3V		160		
C_{IO}	I/O pin capacitance				5		pF
$t_{f(IO)out}$	Output high to low level fall time [1]	C_L=50pF Between 10% and 90%			25		ns
$t_{r(IO)out}$	Output low to high level rise time [1]				25		
$t_{w(IT)in}$	External interrupt pulse time [4]			1			t_{CPU}

Notes:

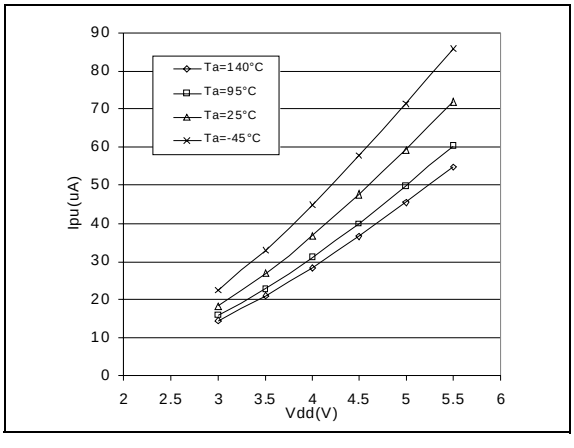
1. Data based on characterization results, not tested in production.
2. Configuration not recommended, all unused pins must be kept at a fixed voltage: using the output mode of the I/O for example or an external pull-up or pull-down resistor (see Figure 71). Data based on design simulation and/or technology characteristics, not tested in production.
3. The R_{PU} pull-up equivalent resistor is based on a resistive transistor (corresponding I_{PU} current characteristics described in Figure 72).
4. To generate an external interrupt, a minimum pulse width has to be applied on an I/O port pin configured as an external interrupt source.

Figure 71. Two typical Applications with unused I/O Pin



Note: only external pull-up allowed on ICCCLK pin

Figure 72. Typical I_{PU} vs. V_{DD} with $V_{IN}=V_{SS}$



PACKAGE CHARACTERISTICS (Cont'd)

Table 21. THERMAL CHARACTERISTICS

Symbol	Ratings	Value	Unit
R_{thJA}	Package thermal resistance (junction to ambient)	TBD	°C/W
P_D	Power dissipation [1]	500	mW
T_{Jmax}	Maximum junction temperature [2]	150	°C

Notes:

1. The power dissipation is obtained from the formula $P_D=P_{INT}+P_{PORT}$ where P_{INT} is the chip internal power ($I_{DD}xV_{DD}$) and P_{PORT} is the port power dissipation determined by the user.
2. The average chip-junction temperature can be obtained from the formula $T_J = T_A + P_D$ x RthJA.

IDENTIFICATION	DESCRIPTION
AN 982	USING ST7 WITH CERAMIC RESONATOR
AN1014	HOW TO MINIMIZE THE ST7 POWER CONSUMPTION
AN1015	SOFTWARE TECHNIQUES FOR IMPROVING MICROCONTROLLER EMC PERFORMANCE
AN1040	MONITORING THE VBUS SIGNAL FOR USB SELF-POWERED DEVICES
AN1070	ST7 CHECKSUM SELF-CHECKING CAPABILITY
AN1324	CALIBRATING THE RC OSCILLATOR OF THE ST7FLITE0 MCU USING THE MAINS
AN1477	EMULATED DATA EEPROM WITH XFLASH MEMORY
AN1502	EMULATED DATA EEPROM WITH ST7 HDFLASH MEMORY
AN1529	EXTENDING THE CURRENT & VOLTAGE CAPABILITY ON THE ST7265 VDDF SUPPLY
AN1530	ACCURATE TIMEBASE FOR LOW-COST ST7 APPLICATIONS WITH INTERNAL RC OSCILLATOR
PROGRAMMING AND TOOLS	
AN 978	KEY FEATURES OF THE STVD7 ST7 VISUAL DEBUG PACKAGE
AN 983	KEY FEATURES OF THE COSMIC ST7 C-COMPILER PACKAGE
AN 985	EXECUTING CODE IN ST7 RAM
AN 986	USING THE INDIRECT ADDRESSING MODE WITH ST7
AN 987	ST7 SERIAL TEST CONTROLLER PROGRAMMING
AN 988	STARTING WITH ST7 ASSEMBLY TOOL CHAIN
AN 989	GETTING STARTED WITH THE ST7 HIWARE C TOOLCHAIN
AN1039	ST7 MATH UTILITY ROUTINES
AN1064	WRITING OPTIMIZED HIWARE C LANGUAGE FOR ST7
AN1071	HALF DUPLEX USB-TO-SERIAL BRIDGE USING THE ST72611 USB MICROCONTROLLER
AN1106	TRANSLATING ASSEMBLY CODE FROM HC05 TO ST7
AN1179	PROGRAMMING ST7 FLASH MICROCONTROLLERS IN REMOTE ISP MODE (IN-SITU PROGRAMMING)
AN1446	USING THE ST72521 EMULATOR TO DEBUG A ST72324 TARGET APPLICATION
AN1478	PORTING AN ST7 PANTA PROJECT TO CODEWARRIOR IDE
AN1527	DEVELOPING A USB SMARTCARD READER WITH ST7SCR
AN1575	ON-BOARD PROGRAMMING METHODS FOR XFLASH AND HDFLASH ST7 MCUS