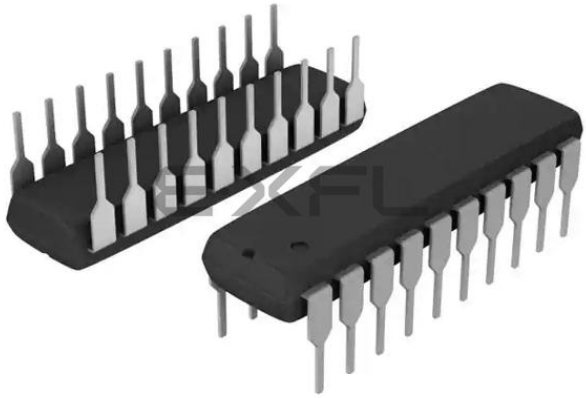


Welcome to E-XFL.COM

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	SPI
Peripherals	LVD, POR, PWM, WDT
Number of I/O	17
Program Memory Size	4KB (4K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	256 x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 7x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Through Hole
Package / Case	20-DIP (0.300", 7.62mm)
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/stmicroelectronics/st7flit10bf1b6

Table 2. Hardware Register Map

Address	Block	Register Label	Register Name	Reset Status	Remarks
0000h	Port A	PADR	Port A Data Register	FFh[1)]	R/W
0001h		PADDR	Port A Data Direction Register	00h	R/W
0002h		PAOR	Port A Option Register	40h	R/W
0003h	Port B	PBDR	Port B Data Register	FFh [1)]	R/W
0004h		PBDDR	Port B Data Direction Register	00h	R/W
0005h		PBOR	Port B Option Register	00h	R/W[2)]
0006h	Port C	PCDR	Port C Data Register	0xh	R/W
0007h		PCDDR	Port C Data Direction Register	00h	R/W
0008h	LITE TIMER 2	LTCSR2	Lite Timer Control/Status Register 2	00h	R/W
0009h		LTARR	Lite Timer Auto-reload Register	00h	R/W
000Ah		LTCNTR	Lite Timer Counter Register	00h	Read Only
000Bh		LTCSR1	Lite Timer Control/Status Register 1	0X00 0000b	R/W
000Ch		LTICR	Lite Timer Input Capture Register	00h	Read Only
000Dh	AUTO-RELOAD TIMER 2	ATCSR	Timer Control/Status Register	0X00 0000b	R/W
000Eh		CNTRH	Counter Register High	00h	Read Only
000Fh		CNTRL	Counter Register Low	00h	Read Only
0010h		ATRH	Auto-Reload Register High	00h	R/W
0011h		ATRL	Auto-Reload Register Low	00h	R/W
0012h		PWMCR	PWM Output Control Register	00h	R/W
0013h		PWM0CSR	PWM 0 Control/Status Register	00h	R/W
0014h		PWM1CSR	PWM 1 Control/Status Register	00h	R/W
0015h		PWM2CSR	PWM 2 Control/Status Register	00h	R/W
0016h		PWM3CSR	PWM 3 Control/Status Register	00h	R/W
0017h		DCR0H	PWM 0 Duty Cycle Register High	00h	R/W
0018h		DCR0L	PWM 0 Duty Cycle Register Low	00h	R/W
0019h		DCR1H	PWM 1 Duty Cycle Register High	00h	R/W
001Ah		DCR1L	PWM 1 Duty Cycle Register Low	00h	R/W
001Bh		DCR2H	PWM 2 Duty Cycle Register High	00h	R/W
001Ch		DCR2L	PWM 2 Duty Cycle Register Low	00h	R/W
001Dh		DCR3H	PWM 3 Duty Cycle Register High	00h	R/W
001Eh		DCR3L	PWM 3 Duty Cycle Register Low	00h	R/W
001Fh		ATICRH	Input Capture Register High	00h	Read Only
0020h		ATICRL	Input Capture Register Low	00h	Read Only
0021h		ATCSR2	Timer Control/Status Register 2	03h	R/W
0022h		BREAKCR	Break Control Register	00h	R/W
0023h		ATR2H	Auto-Reload Register 2 High	00h	R/W
0024h		ATR2L	Auto-Reload Register 2 Low	00h	R/W
0025h		DTGR	Dead Time Generation Register	00h	R/W
0026h		BREAKEN	Break Enable Register	03h	R/W
0027h to 002Bh	Reserved area (5 bytes)				
002Ch	Comparator Voltage Reference	VREFCR	Internal Voltage Reference Control Register	00h	R/W
002Dh	Comparator	CMPCR	Comparator and Internal Reference Control Register	00h	R/W
002Eh	WDG	WDGCR	Watchdog Control Register	7Fh	R/W

FLASH PROGRAM MEMORY (Cont'd)

4.5 Memory Protection

There are two different types of memory protection: Read Out Protection and Write/Erase Protection which can be applied individually.

4.5.1 Read out Protection

Readout protection, when selected provides a protection against program memory content extraction and against write access to Flash memory. Even if no protection can be considered as totally unbreakable, the feature provides a very high level of protection for a general purpose microcontroller. Both program and data E^2 memory are protected.

In flash devices, this protection is removed by reprogramming the option. In this case, both program and data E^2 memory are automatically erased and the device can be reprogrammed.

Read-out protection selection depends on the device type:

– In Flash devices it is enabled and removed through the FMP_R bit in the option byte.

– In ROM devices it is enabled by mask option specified in the Option List.

4.5.2 Flash Write/Erase Protection

Write/erase protection, when set, makes it impossible to both overwrite and erase program memory. It does not apply to E^2 data. Its purpose is to provide advanced security to applications and prevent any change being made to the memory content.

Warning: Once set, Write/erase protection can never be removed. A write-protected flash device is no longer reprogrammable.

Write/erase protection is enabled through the FMP_W bit in the option byte.

4.6 Related Documentation

For details on Flash programming and ICC protocol, refer to the ST7 Flash Programming Reference Manual and to the ST7 ICC Protocol Reference Manual.

4.7 Register Description

FLASH CONTROL/STATUS REGISTER (FCSR)
Read/Write
Reset Value: 000 0000 (00h)
1st RASS Key: 0101 0110 (56h)
2nd RASS Key: 1010 1110 (AEh)

7							0
0	0	0	0	0	OPT	LAT	PGM

Note: This register is reserved for programming using ICP, IAP or other programming methods. It controls the XFlash programming and erasing operations.

When an EPB or another programming tool is used (in socket or ICP mode), the RASS keys are sent automatically.

DATA EEPROM (Cont'd)

5.3 MEMORY ACCESS

The Data EEPROM memory read/write access modes are controlled by the E2LAT bit of the EEPROM Control/Status register (EECSR). The flowchart in Figure 8 describes these different memory access modes.

Read Operation (E2LAT=0)

The EEPROM can be read as a normal ROM location when the E2LAT bit of the EECSR register is cleared.

On this device, Data EEPROM can also be used to execute machine code. Take care not to write to the Data EEPROM while executing from it. This would result in an unexpected code being executed.

Write Operation (E2LAT=1)

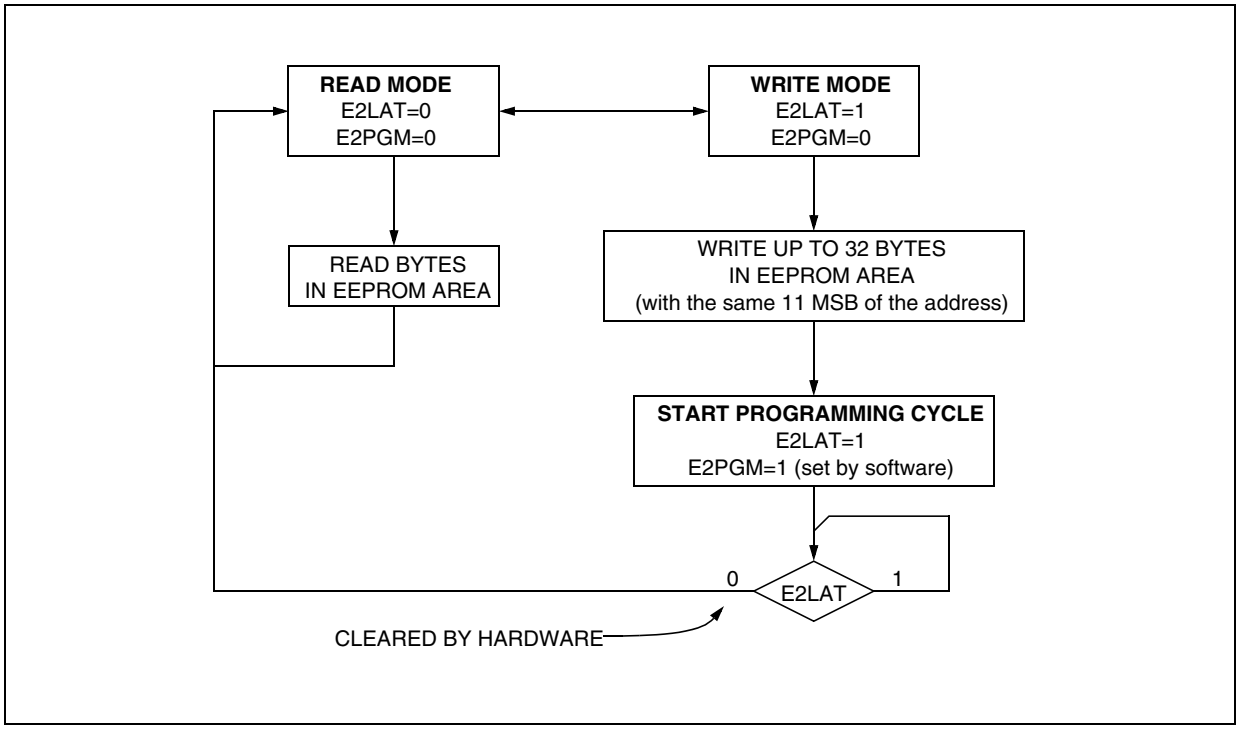
To access the write mode, the E2LAT bit has to be set by software (the E2PGM bit remains cleared). When a write access to the EEPROM area occurs, the value is latched inside the 32 data latches according to its address.

When PGM bit is set by the software, all the previous bytes written in the data latches (up to 32) are programmed in the EEPROM cells. The effective high address (row) is determined by the last EEPROM write sequence. To avoid wrong programming, the user must take care that all the bytes written between two programming sequences have the same high address: only the five Least Significant Bits of the address can change.

At the end of the programming cycle, the PGM and LAT bits are cleared simultaneously.

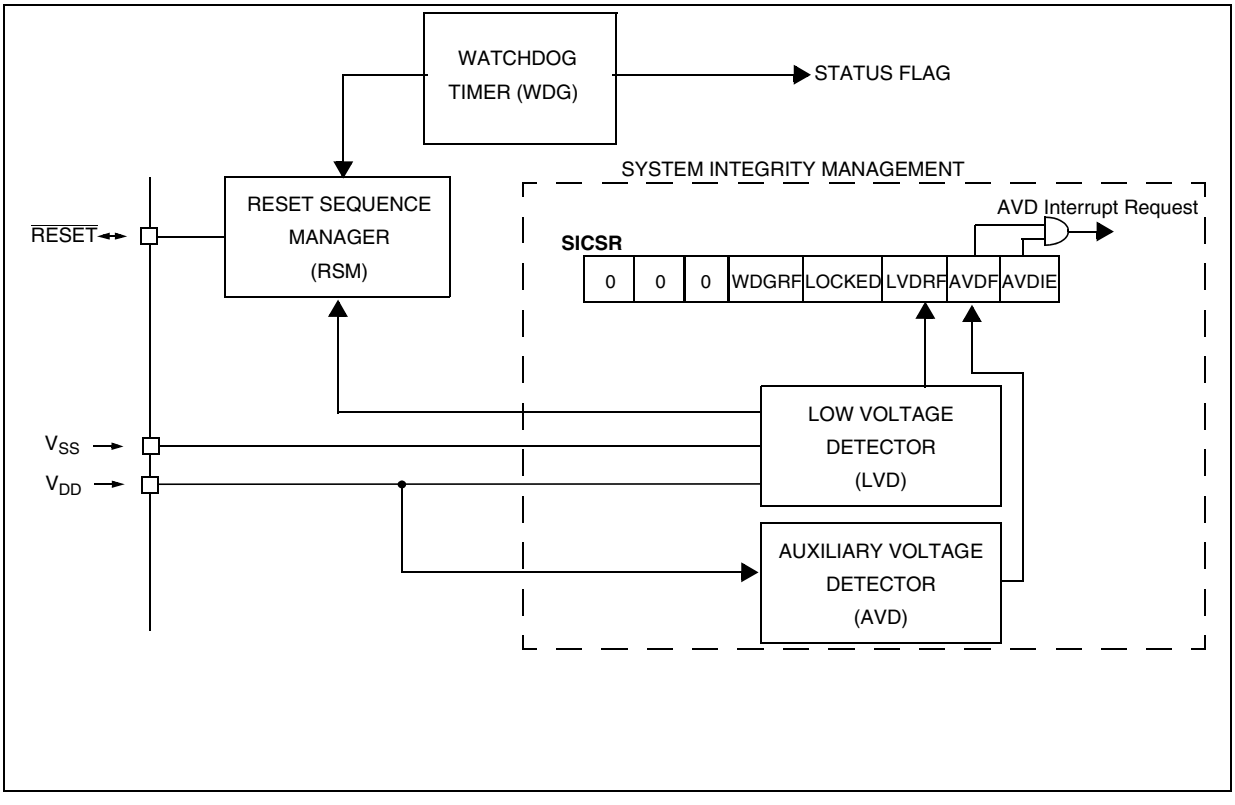
Note: Care should be taken during the programming cycle. Writing to the same memory location will over-program the memory (logical AND between the two write access data result) because the data latches are only cleared at the end of the programming cycle and by the falling edge of the E2LAT bit.
It is not possible to read the latched data.
This note is illustrated by the Figure 10.

Figure 8. Data EEPROM Programming Flowchart



SYSTEM INTEGRITY MANAGEMENT (Cont'd)

Figure 19. Reset and Supply Management Block Diagram



INTERRUPTS (cont'd)

Figure 21. Interrupt Processing Flowchart

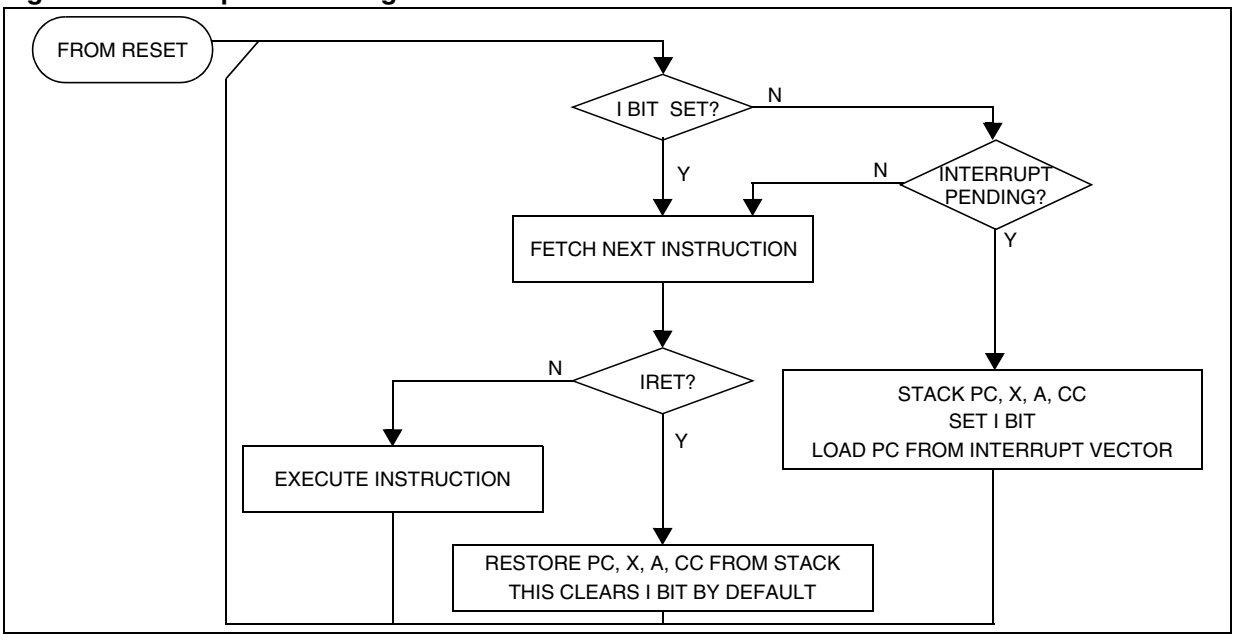


Table 5. Interrupt Mapping

N°	Source Block	Description	Register Label	Priority Order	Exit from HALT or AWUFH	Address Vector
	RESET	Reset	N/A	Highest Priority	yes	FFFEh-FFFFh
	TRAP	Software Interrupt			no	FFFCh-FFFDh
0	AWU	Auto Wake Up Interrupt	AWUCSR		yes[1)]	FFFAh-FFFBh
1	ei0	External Interrupt 0	N/A		yes	FFF8h-FFF9h
2	ei1	External Interrupt 1				FFF6h-FFF7h
3	ei2	External Interrupt 2				FFF4h-FFF5h
4	ei3	External Interrupt 3				FFF2h-FFF3h
5	LITE TIMER	LITE TIMER RTC2 interrupt	LTCSR2		no	FFF0h-FFF1h
6	Comparator	Comparator Interrupt	CMPCR		no	FFEEh-FFEFh
7	SI	AVD interrupt	SICSR		no	FFECh-FFEDh
8	AT TIMER	AT TIMER Output Compare Interrupt or Input Capture Interrupt	PWMxCSR or ATCSR		no	FFEAh-FFEBh
9		AT TIMER Overflow Interrupt	ATCSR		yes[2)]	FFE8h-FFE9h
10	LITE TIMER	LITE TIMER Input Capture Interrupt	LTCSR		no	FFE6h-FFE7h
11		LITE TIMER RTC1 Interrupt	LTCSR		yes[2)]	FFE4h-FFE5h
12	SPI	SPI Peripheral Interrupts	SPICSR	Lowest Priority	yes	FFE2h-FFE3h
13	AT TIMER	AT TIMER Overflow Interrupt	ATCSR2		no	FFE0h-FFE1h

Note 1: This interrupt exits the MCU from "Auto Wake-up from Halt" mode only.
Note 2 : These interrupts exit the MCU from "ACTIVE-HALT" mode only.

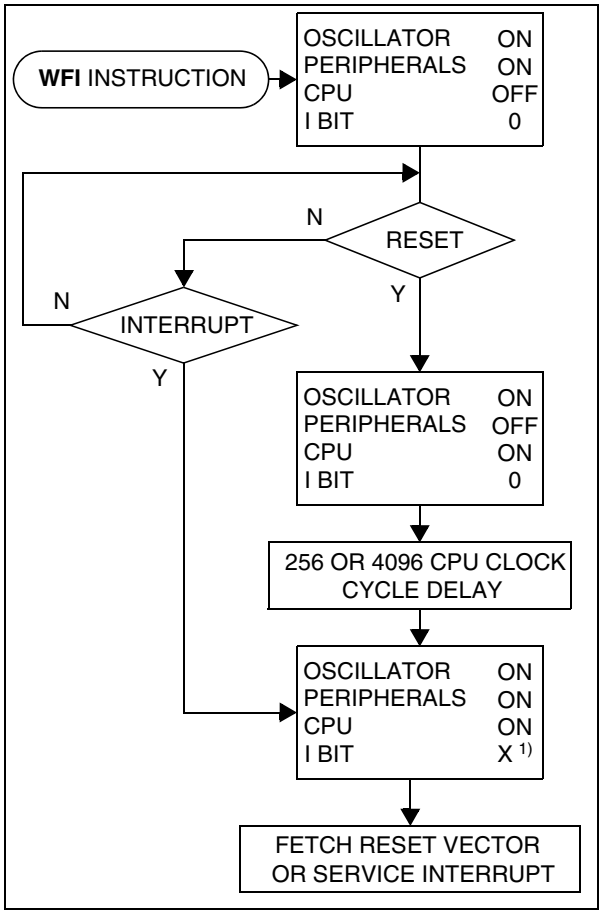
POWER SAVING MODES (Cont'd)

9.3 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 bit of the CC register is cleared, 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 bit of the CC register is set during the interrupt routine and cleared when the CC register is popped.

POWER SAVING MODES (Cont'd)

9.4.1 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, re-initialize 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, re-initialize 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 program memory 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).

9.5 ACTIVE-HALT MODE

ACTIVE-HALT mode is the lowest power consumption mode of the MCU with a real time clock available. It is entered by executing the 'HALT' instruction. The decision to enter either in ACTIVE-HALT or HALT mode is given by the LTCSR/ATCSR register status as shown in the following table:

LTCSR1 TB1IE bit	ATCSR OVFIE bit	ATCSR CK1 bit	ATCSR CK0 bit	Meaning
0	x	x	0	ACTIVE-HALT mode disabled
0	0	x	x	
1	x	x	x	ACTIVE-HALT mode enabled
x	1	0	1	

The MCU can exit ACTIVE-HALT mode on reception of a specific interrupt (see Table 5, "Interrupt Mapping," on page 37) or a RESET.

– When exiting ACTIVE-HALT mode by means of a RESET, a 256 or 4096 CPU cycle delay occurs. After the start up delay, the CPU resumes operation by fetching the reset vector which woke it up (see Figure 28).

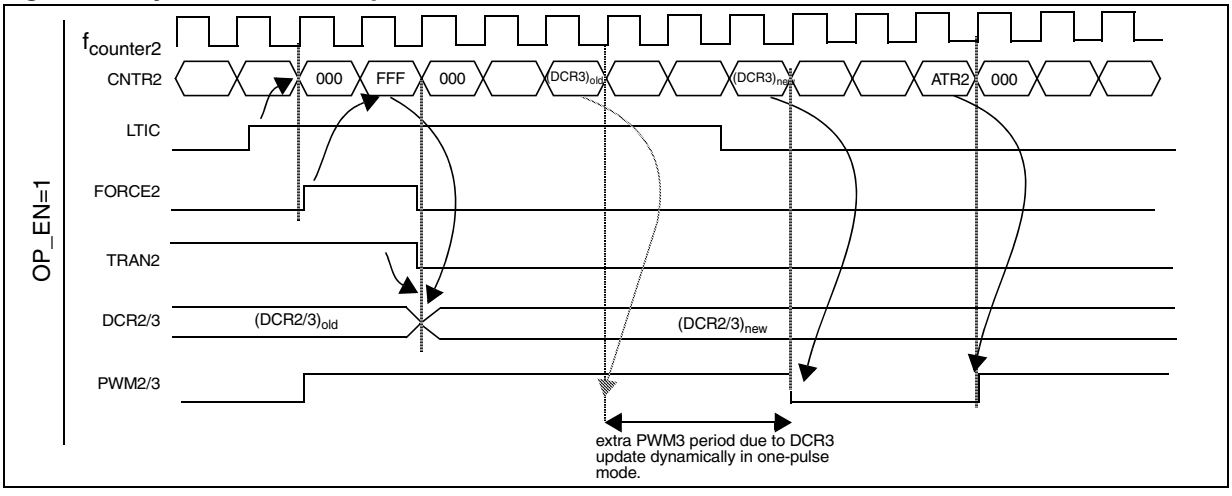
– When exiting ACTIVE-HALT mode by means of an interrupt, the CPU immediately resumes operation by servicing the interrupt vector which woke it up (see Figure 28).

When entering ACTIVE-HALT mode, the I bit in the CC register is cleared to enable interrupts. Therefore, if an interrupt is pending, the MCU wakes up immediately (see Note 3).

In ACTIVE-HALT mode, only the main oscillator and the selected timer counter (LT/AT) are running to keep a wake-up time base. All other peripherals are not clocked except those which get their clock supply from another clock generator (such as external or auxiliary oscillator).

Note: As soon as ACTIVE-HALT is enabled, executing a HALT instruction while the Watchdog is active does not generate a RESET.
This means that the device cannot spend more than a defined delay in this power saving mode.

Figure 49. Dynamic DCR2/3 update in One Pulse Mode



ON-CHIP PERIPHERALS (cont'd)

11.4 SERIAL PERIPHERAL INTERFACE (SPI)

11.4.1 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.

11.4.2 Main Features

- Full duplex synchronous transfers (on three lines)
- Simplex synchronous transfers (on two lines)
- Master or slave operation
- 6 master mode frequencies (f_{CPU}/4 max.)
- f_{CPU}/2 max. slave mode frequency (see note)
- $\overline{SS}$ Management by software or hardware
- Programmable clock polarity and phase
- End of transfer interrupt flag
- Write collision, Master Mode Fault and Overrun flags

Note: In slave mode, continuous transmission is not possible at maximum frequency due to the software overhead for clearing status flags and to initiate the next transmission sequence.

11.4.3 General Description

Figure 1 on page 3 shows the serial peripheral interface (SPI) block diagram. There are three registers:

- SPI Control Register (SPICR)
- SPI Control/Status Register (SPICSR)
- SPI Data Register (SPIDR)

The SPI is connected to external devices through four pins:

- MISO: Master In / Slave Out data
- MOSI: Master Out / Slave In data
- SCK: Serial Clock out by SPI masters and input by SPI slaves
- $\overline{SS}$: Slave select:
  This input signal acts as a 'chip select' to let the SPI master communicate with slaves individually and to avoid contention on the data lines. Slave $\overline{SS}$ inputs can be driven by standard I/O ports on the master Device.

SERIAL PERIPHERAL INTERFACE (cont'd)

11.4.3.3 Master Mode Operation

In master mode, the serial clock is output on the SCK pin. The clock frequency, polarity and phase are configured by software (refer to the description of the SPICSR register).

Note: The idle state of SCK must correspond to the polarity selected in the SPICSR register (by pulling up SCK if CPOL = 1 or pulling down SCK if CPOL = 0).

How to operate the SPI in master mode

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

1. Write to the SPICR register:
   - Select the clock frequency by configuring the SPR[2:0] bits.
   - Select the clock polarity and clock phase by configuring the CPOL and CPHA bits. Figure 5 shows the four possible configurations.
     Note: The slave must have the same CPOL and CPHA settings as the master.
2. Write to the SPICSR register:
   - Either set the SSM bit and set the SSI bit or clear the SSM bit and tie the SS pin high for the complete byte transmit sequence.
3. Write to the SPICR register:
   - Set the MSTR and SPE bits
     Note: MSTR and SPE bits remain set only if SS is high).

Important note: if the SPICSR register is not written first, the SPICR register setting (MSTR bit) may be not taken into account.

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

11.4.3.4 Master Mode Transmit Sequence

When software writes to the SPIDR register, the data byte is loaded into the 8-bit shift register 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 request is generated if the SPIE bit is set and the interrupt mask in the CCR register is cleared.

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

1. An access to the SPICSR register while the SPIF bit is set
2. A read to the SPIDR register

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

11.4.3.5 Slave Mode Operation

In slave mode, the serial clock is received on the SCK pin from the master device.

To operate the SPI in slave mode:

1. Write to the SPICSR register to perform the following actions:
   - Select the clock polarity and clock phase by configuring the CPOL and CPHA bits (see Figure 5).
     Note: The slave must have the same CPOL and CPHA settings as the master.
   - Manage the $\overline{SS}$ pin as described in Section 0.1.3.2 and Figure 3. If CPHA = 1 $\overline{SS}$ must be held low continuously. If CPHA = 0 $\overline{SS}$ must be held low during byte transmission and pulled up between each byte to let the slave write in the shift register.
2. Write to the SPICR register to clear the MSTR bit and set the SPE bit to enable the SPI I/O functions.

11.4.3.6 Slave Mode Transmit Sequence

When software writes to the SPIDR register, the data byte is loaded into the 8-bit shift register and then shifted out serially to the MISO pin most significant bit first.

The transmit sequence begins when the slave device receives the clock signal and the most significant bit of the data on its MOSI pin.

When data transfer is complete:

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

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

1. An access to the SPICSR register while the SPIF bit is set
2. A write or a read to the SPIDR register

Notes: While the SPIF bit is set, all writes to the SPIDR register are inhibited until the SPICSR register is read.

The SPIF bit can be cleared during a second transmission; however, it must be cleared before the second SPIF bit in order to prevent an Overrun condition (see Section 0.1.5.2).

SERIAL PERIPHERAL INTERFACE (Cont'd)

Table 17. SPI Register Map and Reset Values

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
0031h	**SPIDR** Reset Value	MSB x	x	x	x	x	x	x	LSB x
0032h	**SPICR** Reset Value	SPIE 0	SPE 0	SPR2 0	MSTR 0	CPOL x	CPHA x	SPR1 x	SPR0 x
0033h	**SPICSR** Reset Value	SPIF 0	WCOL 0	OVR 0	MODF 0	0	SOD 0	SSM 0	SSI 0

11.5 10-BIT A/D CONVERTER (ADC)

11.5.1 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 7 multiplexed analog input channels (refer to device pin out description) that allow the peripheral to convert the analog voltage levels from up to 7 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.

11.5.2 Main Features

- 10-bit conversion
- Up to 7 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 60.

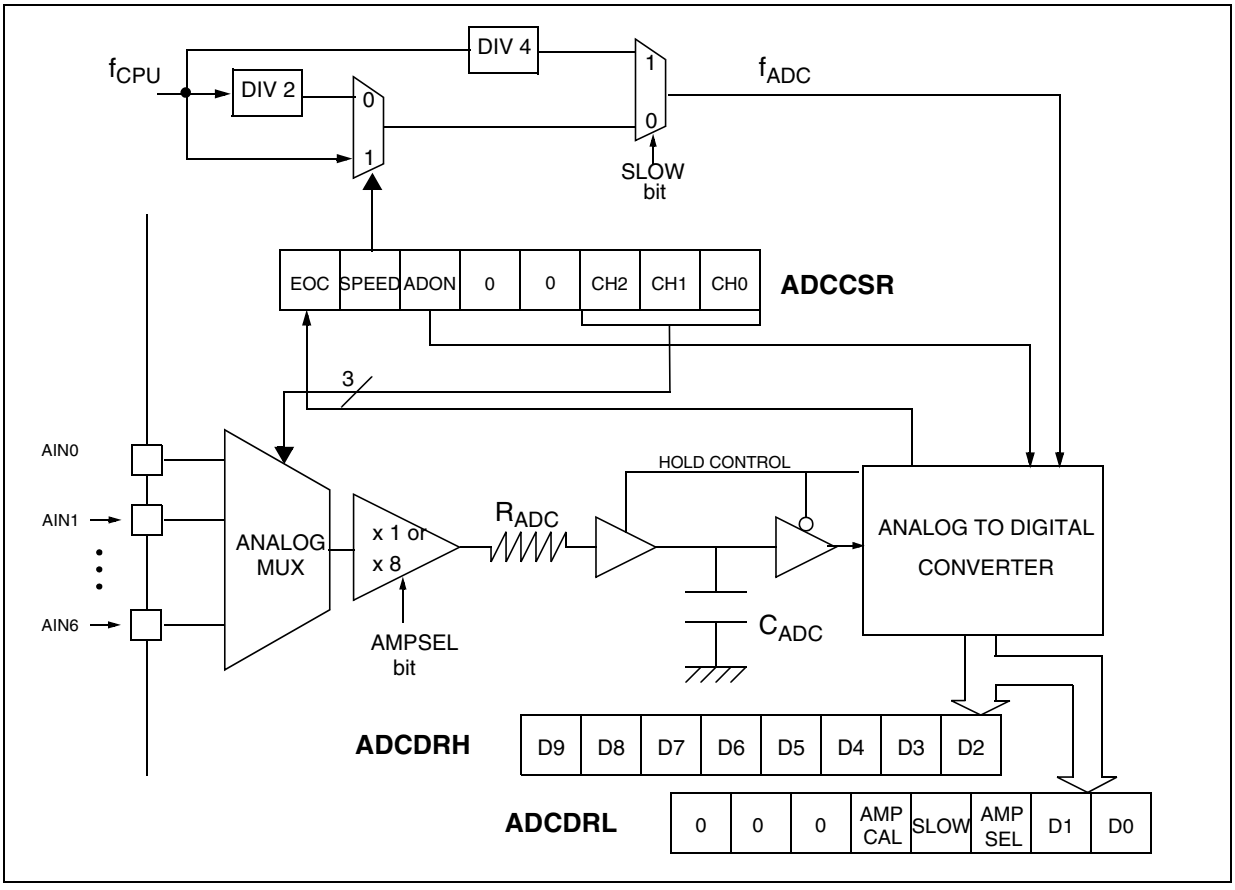
11.5.3 Functional Description

11.5.3.1 Analog Power Supply

V_{DDA} and V_{SSA} are the high and low level reference voltage pins. In some devices (refer to device pin out description) they are internally connected to the V_{DD} and V_{SS} pins.

Conversion accuracy may therefore be impacted by voltage drops and noise in the event of heavily loaded or badly decoupled power supply lines.

Figure 60. ADC Block Diagram



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							
Interruption management	TRAP	WFI	HALT	IRET				
Condition Code Flag modification	SIM	RIM	SCF	RCF				

Using a prebyte

The instructions are described with 1 to 4 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.

12.2.1 Illegal Opcode Reset

In order to provide enhanced robustness to the device against unexpected behavior, a system of illegal opcode detection is implemented. If a code to be executed does not correspond to any opcode or prebyte value, a reset is generated. This, combined with the Watchdog, allows the detection and recovery from an unexpected fault or interference.

Note: A valid prebyte associated with a valid opcode forming an unauthorized combination does not generate a reset.

INSTRUCTION GROUPS (cont'd)

Mnemo	Description	Function/Example	Dst	Src	H	I	N	Z	C
ADC	Add with Carry	A = A + M + C	A	M	H		N	Z	C
ADD	Addition	A = A + M	A	M	H		N	Z	C
AND	Logical And	A = A . M	A	M			N	Z	
BCP	Bit compare A, Memory	tst (A . M)	A	M			N	Z	
BRES	Bit Reset	bres Byte, #3	M						
BSET	Bit Set	bset Byte, #3	M						
BTJF	Jump if bit is false (0)	btjf Byte, #3, Jmp1	M						C
BTJT	Jump if bit is true (1)	btjt Byte, #3, Jmp1	M						C
CALL	Call subroutine								
CALLR	Call subroutine relative								
CLR	Clear		reg, M				0	1	
CP	Arithmetic Compare	tst(Reg - M)	reg	M			N	Z	C
CPL	One Complement	A = FFH-A	reg, M				N	Z	1
DEC	Decrement	dec Y	reg, M				N	Z	
HALT	Halt					0			
IRET	Interrupt routine return	Pop CC, A, X, PC			H	I	N	Z	C
INC	Increment	inc X	reg, M				N	Z	
JP	Absolute Jump	jp [TBL.w]							
JRA	Jump relative always								
JRT	Jump relative								
JRF	Never jump	jrf *							
JRIH	Jump if ext. interrupt = 1								
JRIL	Jump if ext. interrupt = 0								
JRH	Jump if H = 1	H = 1 ?							
JRNH	Jump if H = 0	H = 0 ?							
JRM	Jump if I = 1	I = 1 ?							
JRNM	Jump if I = 0	I = 0 ?							
JRMI	Jump if N = 1 (minus)	N = 1 ?							
JRPL	Jump if N = 0 (plus)	N = 0 ?							
JREQ	Jump if Z = 1 (equal)	Z = 1 ?							
JRNE	Jump if Z = 0 (not equal)	Z = 0 ?							
JRC	Jump if C = 1	C = 1 ?							
JRNC	Jump if C = 0	C = 0 ?							
JRULT	Jump if C = 1	Unsigned <							
JRUGE	Jump if C = 0	Jmp if unsigned >=							
JRUGT	Jump if (C + Z = 0)	Unsigned >							

13.2 ABSOLUTE MAXIMUM RATINGS

Stresses above those listed as “absolute maximum ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device under these conditions is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

13.2.1 Voltage Characteristics

Symbol	Ratings	Maximum value	Unit
V_{DD} - V_{SS}	Supply voltage	7.0	V
V_{IN}	Input voltage on any pin [1) & 2)]	V_{SS}-0.3 to V_{DD}+0.3	
$V_{ESD(HBM)}$	Electrostatic discharge voltage (Human Body Model)	see section 13.7.3 on page 128	
$V_{ESD(MM)}$	Electrostatic discharge voltage (Machine Model)		

13.2.2 Current Characteristics

Symbol	Ratings	Maximum value	Unit
I_{VDD}	Total current into V_{DD} power lines (source) [3)]	75	mA
I_{VSS}	Total current out of V_{SS} ground lines (sink) [3)]	150	
I_{IO}	Output current sunk by any standard I/O and control pin	20	
	Output current sunk by any high sink I/O pin	40	
	Output current source by any I/Os and control pin	- 25	
$I_{INJ(PIN)}$ [2) & 4)]	Injected current on ISPSEL pin	± 5	
	Injected current on $\overline{RESET}$ pin	± 5	
	Injected current on OSC1 and OSC2 pins	± 5	
	Injected current on PB0 pin [5)]	+5	
	Injected current on any other pin [6)]	± 5	
$\Sigma I_{INJ(PIN)}$ [2)]	Total injected current (sum of all I/O and control pins) [6)]	± 20	

13.2.3 Thermal Characteristics

Symbol	Ratings	Value	Unit
T_{STG}	Storage temperature range	-65 to +150	°C
T_J	Maximum junction temperature (see Table 24, “THERMAL CHARACTERISTICS,” on page 147)		

Notes:

1. Directly connecting the $\overline{RESET}$ and I/O pins to V_{DD} or V_{SS} could damage the device if an unintentional internal reset is generated or an unexpected change of the I/O configuration occurs (for example, due to a corrupted program counter). To guarantee safe operation, this connection has to be done through a pull-up or pull-down resistor (typical: 4.7kΩ for $\overline{RESET}$, 10kΩ for I/Os). Unused I/O pins must be tied in the same way to V_{DD} or V_{SS} according to their reset configuration.
2. $I_{INJ(PIN)}$ must never be exceeded. This is implicitly insured if V_{IN} maximum is respected. If V_{IN} maximum cannot be respected, the injection current must be limited externally to the $I_{INJ(PIN)}$ value. A positive injection is induced by $V_{IN}>V_{DD}$ while a negative injection is induced by $V_{IN}<V_{SS}$. For true open-drain pads, there is no positive injection current, and the corresponding V_{IN} maximum must always be respected
3. All power (V_{DD}) and ground (V_{SS}) lines must always be connected to the external supply.
4. Negative injection disturbs the analog performance of the device. In particular, it induces leakage currents throughout the device including the analog inputs. To avoid undesirable effects on the analog functions, care must be taken:
- Analog input pins must have a negative injection less than 0.8 mA (assuming that the impedance of the analog voltage is lower than the specified limits)
- Pure digital pins must have a negative injection less than 1.6mA. In addition, it is recommended to inject the current as far as possible from the analog input pins.
5. No negative current injection allowed on PB0 pin.
6. When several inputs are submitted to a current injection, the maximum $\Sigma I_{INJ(PIN)}$ is the absolute sum of the positive and negative injected currents (instantaneous values). These results are based on characterisation with $\Sigma I_{INJ(PIN)}$ maximum current injection on four I/O port pins of the device.

13.3.6 Operating conditions with ADC

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

Symbol	Parameter	Typ	Unit
$I_{INJ(ANA)}$ [1]	Injected current on any analog pin	0	mA

Note:
1. Current injection (negative or positive) not allowed on any analog pin.

13.4 SUPPLY CURRENT CHARACTERISTICS

The following current consumption specified for the ST7 functional operating modes over temperature range does not take into account the clock source current consumption. To get the total device consumption, the two current values must be added (except for HALT mode for which the clock is stopped).

13.4.1 Supply Current

T_A = -40 to +85°C unless otherwise specified

Symbol	Parameter	Conditions		Typ	Max	Unit
I_{DD}	Supply current in RUN mode	V_{DD}=5.5V	f_{CPU}=8MHz 1)	7	9	mA
	Supply current in WAIT mode		f_{CPU}=8MHz 2)	3	3.6	
	Supply current in SLOW mode		f_{CPU}=250kHz 3)	0.7	0.9	
	Supply current in SLOW WAIT mode7		f_{CPU}=250kHz 4)	0.5	0.8	
	Supply current in HALT mode5)		-40°C≤T_A≤+125°C	<1	6	µA
	Supply current in AWUFH mode 6)7)		-40°C≤T_A≤+125°C	20		

Notes:

1. CPU running with memory access, all I/O pins in input mode with a static value at V_{DD} or V_{SS} (no load), all peripherals in reset state; clock input (CLKIN) driven by external square wave, LVD disabled.

2. All I/O pins in input mode with a static value at V_{DD} or V_{SS} (no load), all peripherals in reset state; clock input (CLKIN) driven by external square wave, LVD disabled.

3. SLOW mode selected with f_{CPU} based on f_{OSC} divided by 32. All I/O pins in input mode with a static value at V_{DD} or V_{SS} (no load), all peripherals in reset state; clock input (CLKIN) driven by external square wave, LVD disabled.

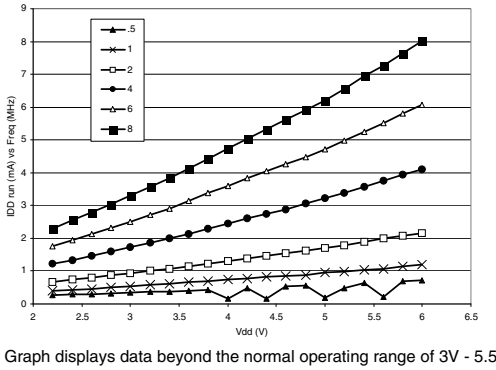
4. SLOW-WAIT mode selected with f_{CPU} based on f_{OSC} divided by 32. All I/O pins in input mode with a static value at V_{DD} or V_{SS} (no load), all peripherals in reset state; clock input (CLKIN) driven by external square wave, LVD disabled.

5. All I/O pins in output mode with a static value at V_{SS} (no load), LVD disabled. Data based on characterization results, tested in production at V_{DD} max and f_{CPU} max.

6. All I/O pins in input mode with a static value at V_{DD} or V_{SS} (no load). Data tested in production at V_{DD} max. and f_{CPU} max.

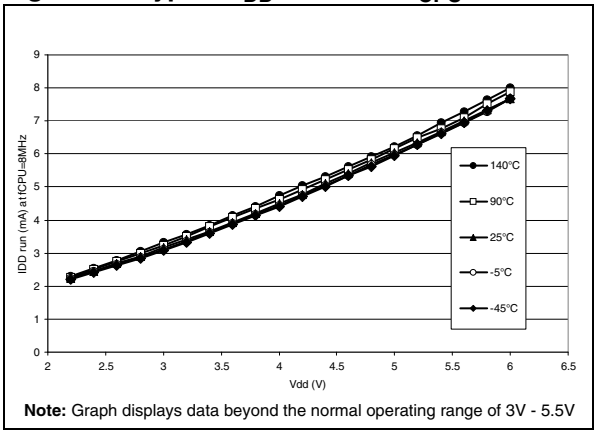
7. This consumption refers to the Halt period only and not the associated run period which is software dependent.

Figure 71. Typical I_{DD} in RUN vs. f_{CPU}



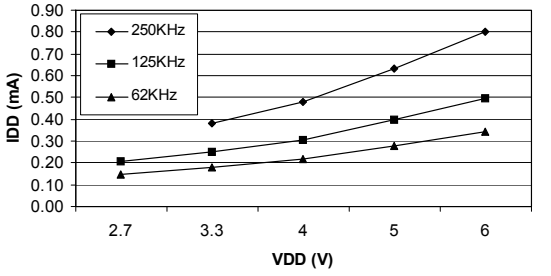
Note: Graph displays data beyond the normal operating range of 3V - 5.5V

Figure 72. Typical I_{DD} in RUN at f_{CPU} = 8MHz



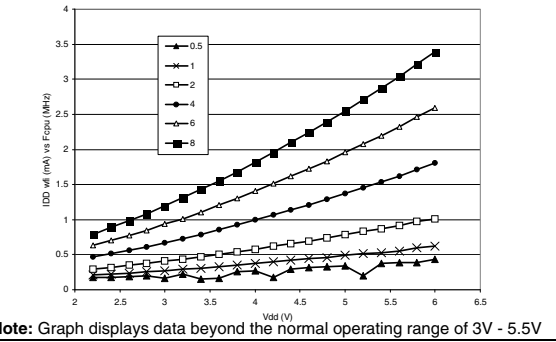
Note: Graph displays data beyond the normal operating range of 3V - 5.5V

Figure 73. Typical I_{DD} in SLOW vs. f_{CPU}



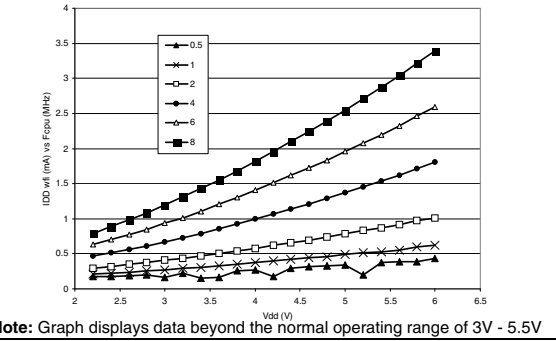
Note: Graph displays data beyond the normal operating range of 3V - 5.5V

Figure 74. Typical I_{DD} in WAIT vs. f_{CPU}



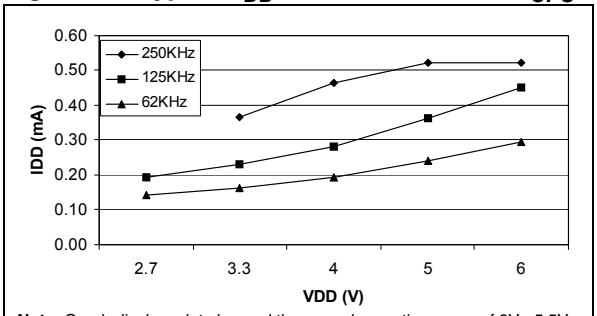
Note: Graph displays data beyond the normal operating range of 3V - 5.5V

Figure 75. Typical I_{DD} in WAIT at f_{CPU}= 8MHz



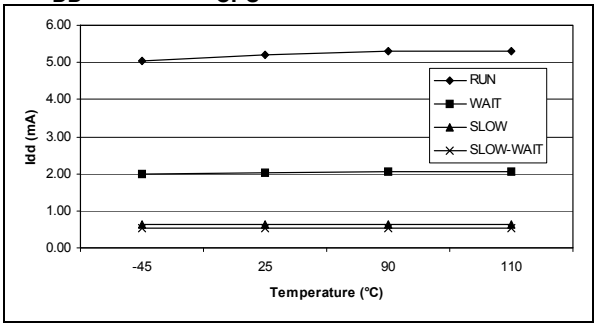
Note: Graph displays data beyond the normal operating range of 3V - 5.5V

Figure 76. Typical I_{DD} in SLOW-WAIT vs. f_{CPU}



Note: Graph displays data beyond the normal operating range of 3V - 5.5V

Figure 77. Typical I_{DD} vs. Temperature at V_{DD} = 5V and f_{CPU} = 8MHz



13.5 CLOCK AND TIMING CHARACTERISTICS

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

13.5.1 General Timings

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

Notes:

1. Guaranteed by Design. Not tested in production.

2. Data based on typical application software.

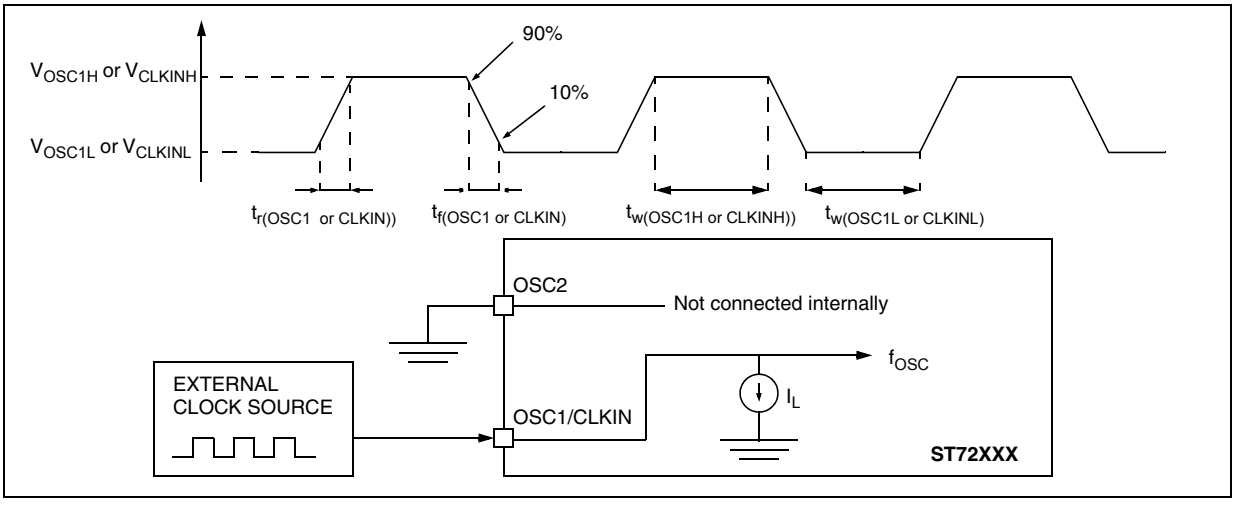
3. Time measured between interrupt event and interrupt vector fetch. $\Delta t_{c(INST)}$ is the number of t_{CPU} cycles needed to finish the current instruction execution.

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

13.5.2 External Clock Source

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{OSC1H} or V_{CLKIN_H}	OSC1/CLKIN input pin high level voltage	see Figure 78	0.7xV_{DD}		V_{DD}	V
V_{OSC1L} or V_{CLKIN_L}	OSC1/CLKIN input pin low level voltage		V_{SS}		0.3xV_{DD}	
$t_{w(OSC1H)}$ or $t_{w(CLKINH)}$ $t_{w(OSC1L)}$ or $t_{w(CLKINL)}$	OSC1/CLKIN high or low time [4]		15			ns
$t_{r(OSC1)}$ or $t_{r(CLKIN)}$ $t_{f(OSC1)}$ or $t_{f(CLKIN)}$	OSC1/CLKIN rise or fall time [4]				15	
I_L	OSCx/CLKIN Input leakage current	$V_{SS} \le V_{IN} \le V_{DD}$			±1	µA

Figure 78. Typical Application with an External Clock Source



13.5.3 Auto Wakeup from Halt Oscillator (AWU)

Symbol	Parameter [1]	Conditions	Min	Typ	Max	Unit
f_{AWU}	AWU Oscillator Frequency		50	125	250	kHz
t_{RCSRT}	AWU Oscillator startup time				50	µs

Note: 1. Guaranteed by Design. Not tested in production.

15.3 DEVELOPMENT TOOLS

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

15.3.1 Starter kits

ST offers complete, affordable **starter kits**. Starter kits are complete hardware/software tool packages that include features and samples to help you quickly start developing your application.

15.3.2 Development and debugging tools

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

The range of hardware tools includes full-featured **ST7-EMU3 series emulators**, cost effective **ST7-DVP3 series emulators** and the low-cost **RLink** in-circuit debugger/programmer. These tools are supported by the **ST7 Toolset** from STMicroelectronics, which includes the STVD7 integrated development environment (IDE) with high-level language debugger, editor, project manager and integrated programming interface.

15.3.3 Programming tools

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

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

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

15.3.4 Order Codes for Development and Programming Tools

Table 28 below lists the ordering codes for the ST7LITE1xB development and programming tools. For additional ordering codes for spare parts and accessories, refer to the online product selector at www.st.com/mcu.

15.3.5 Order codes for ST7LITE1xB development tools

Table 28. Development tool order codes for the ST7LITE1xB family

MCU	In-circuit Debugger, RLink Series[1]		Emulator		Programming Tool	
	Starter Kit without Demo Board	Starter Kit with Demo Board	DVP Series	EMU Series	In-circuit Programmer	ST Socket Boards and EPBs
ST7FLIT1xBF0 ST7FLIT1xBF1 ST7FLIT1xBY0 ST7FLIT1xBY1	STX-RLINK[2]	ST7FLITE-SK/RAIS[2]	ST7MDT10-DVP3[4]	ST7MDT10-EMU3	STX-RLINK ST7-STICK[3][5]	ST7SB10-123[3]

Notes:

1. Available from ST or from Raisonance, www.raisonance.com
2. USB connection to PC
3. Add suffix /EU, /UK or /US for the power supply for your region
4. Includes connection kit for DIP16/SO16 only. See "How to order an EMU or DVP" in ST product and tool selection guide for connection kit ordering information
5. Parallel port connection to PC