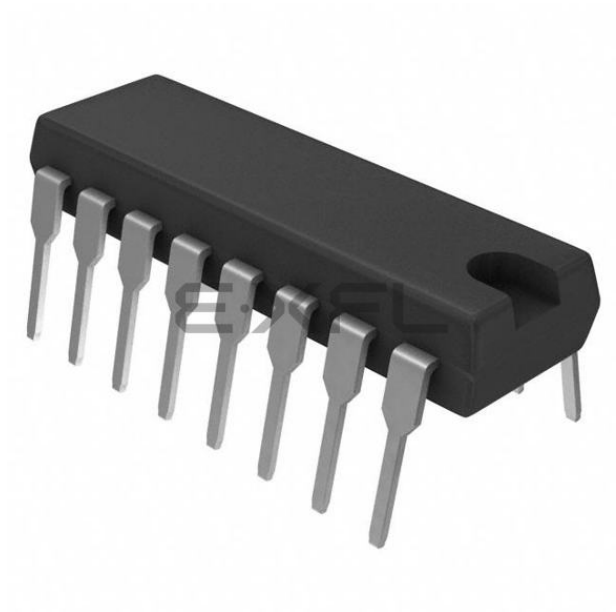


Welcome to E-XFL.COM

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	13
Program Memory Size	1KB (1K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	128 x 8
Voltage - Supply (Vcc/Vdd)	2.4V ~ 5.5V
Data Converters	A/D 5x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Through Hole
Package / Case	16-DIP (0.300", 7.62mm)
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/stmicroelectronics/st7flites5y0b6

Address	Block	Register Label	Register Name	Reset Status	Remarks
003Ah	SI	SICSR	System Integrity Control/Status Register	0xh	R/W
003Bh to 007Fh	Reserved area (69 bytes)				

Notes:

1. The contents of the I/O port DR registers are readable only in output configuration. In input configuration, the values of the I/O pins are returned instead of the DR register contents.

2. The bits associated with unavailable pins must always keep their reset value.

CPU REGISTERS (Cont'd)

Stack Pointer (SP)

Read/Write

Reset Value: 00 FFh

15							8
0	0	0	0	0	0	0	0

7							0
1	1	SP5	SP4	SP3	SP2	SP1	SP0

The Stack Pointer is a 16-bit register which is always pointing to the next free location in the stack. It is then decremented after data has been pushed onto the stack and incremented before data is popped from the stack (see Figure 12).

Since the stack is 64 bytes deep, the 10 most significant bits are forced by hardware. Following an MCU Reset, or after a Reset Stack Pointer instruction (RSP), the Stack Pointer contains its reset value (the SP5 to SP0 bits are set) which is the stack higher address.

The least significant byte of the Stack Pointer (called S) can be directly accessed by a LD instruction.

Note: When the lower limit is exceeded, the Stack Pointer wraps around to the stack upper limit, without indicating the stack overflow. The previously stored information is then overwritten and therefore lost. The stack also wraps in case of an underflow.

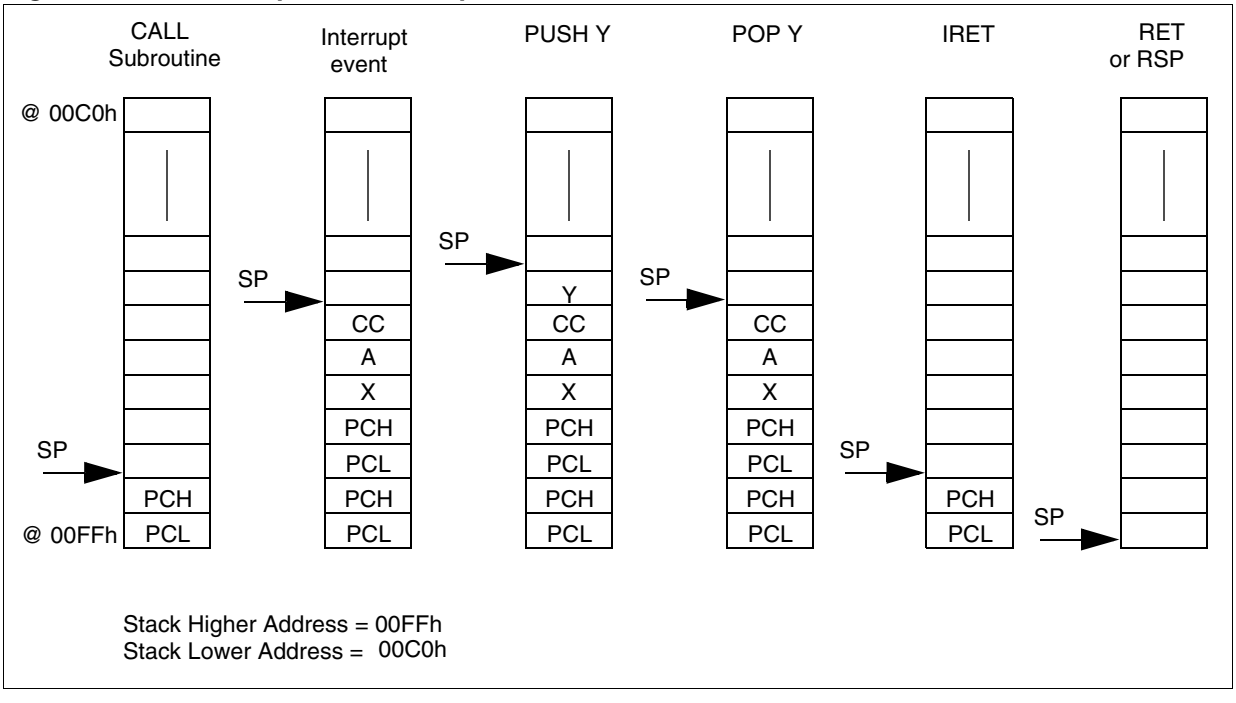
The stack is used to save the return address during a subroutine call and the CPU context during an interrupt. The user may also directly manipulate the stack by means of the PUSH and POP instructions. In the case of an interrupt, the PCL is stored at the first location pointed to by the SP. Then the other registers are stored in the next locations as shown in Figure 12.

– When an interrupt is received, the SP is decremented and the context is pushed on the stack.

– On return from interrupt, the SP is incremented and the context is popped from the stack.

A subroutine call occupies two locations and an interrupt five locations in the stack area.

Figure 12. Stack Manipulation Example



9 POWER SAVING MODES

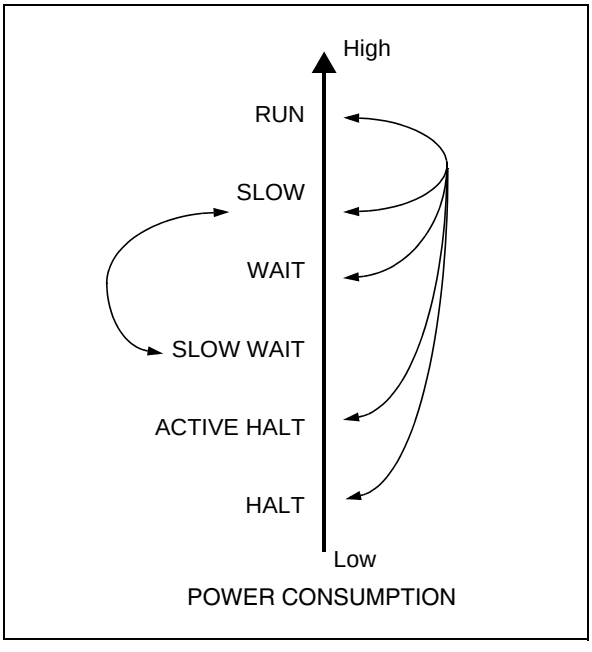
9.1 INTRODUCTION

To give a large measure of flexibility to the application in terms of power consumption, four main power saving modes are implemented in the ST7 (see Figure 22): SLOW, WAIT (SLOW WAIT), ACTIVE HALT and 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 (f_{OSC}).

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 22. 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.

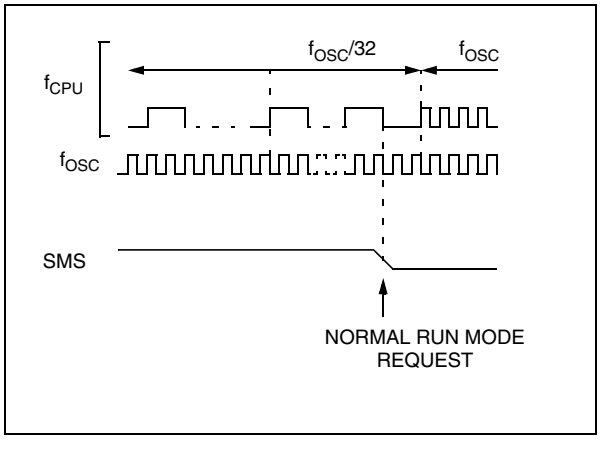
In this mode, the oscillator frequency is divided by 32. The CPU and peripherals are clocked at this lower frequency.

Notes:

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

SLOW mode has no effect on the Lite Timer which is already clocked at $F_{OSC/32}$.

Figure 23. SLOW Mode Clock Transition



LITE TIMER (Cont'd)

11.1.3 Functional Description

The value of the 8-bit counter cannot be read or written by software. After an MCU reset, it starts incrementing from 0 at a frequency of $f_{OSC}/32$. A counter overflow event occurs when the counter rolls over from F9h to 00h. If f_{OSC} = 8 MHz, then the time period between two counter overflow events is 1 ms. This period can be doubled by setting the TB bit in the LTCSR register.

When the timer overflows, the TBF bit is set by hardware and an interrupt request is generated if the TBIE is set. The TBF bit is cleared by software reading the LTCSR register.

11.1.3.1 Watchdog

The watchdog is enabled using the WDGE bit. The normal Watchdog timeout is 2ms (@ = 8 MHz f_{OSC}), after which it then generates a reset.

To prevent this watchdog reset occuring, software must set the WDGD bit. The WDGD bit is cleared by hardware after t_{WDG}. This means that software must write to the WDGD bit at regular intervals to prevent a watchdog reset occurring. Refer to Figure 32.

If the watchdog is not enabled immediately after reset, the first watchdog timeout will be shorter than 2ms, because this period is counted starting from reset. Moreover, if a 2ms period has already elapsed after the last MCU reset, the watchdog reset will take place as soon as the WDGE bit is set. For these reasons, it is recommended to enable the Watchdog immediately after reset or else to set the WDGD bit before the WGDE bit so a watchdog reset will not occur for at least 2ms.

A Watchdog reset can be forced at any time by setting the WDGRF bit. To generate a forced watchdog reset, first watchdog has to be activated by setting the WDGE bit and then the WDGRF bit has to be set.

The WDGRF bit also acts as a flag, indicating that the Watchdog was the source of the reset. It is automatically cleared after it has been read.

Caution: When the WDGRF bit is set, software must clear it, otherwise the next time the watchdog is enabled (by hardware or software), the microcontroller will be immediately reset.

Hardware Watchdog Option

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

Refer to the Option Byte description in the "device configuration and ordering information" section.

Using Halt Mode with the Watchdog (option)

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

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

If an external interrupt is received, the WDG restarts counting after 256 CPU clocks. If a reset is generated, the Watchdog is disabled (reset state).

If Halt mode with Watchdog is enabled by option byte (No watchdog reset on HALT instruction), it is recommended before executing the HALT instruction to refresh the WDG counter, to avoid an unexpected WDG reset immediately after waking up the microcontroller.

11.3 SERIAL PERIPHERAL INTERFACE (SPI)

11.3.1 Introduction

The Serial Peripheral Interface (SPI) allows full-duplex, synchronous, serial communication with external devices. An SPI system may consist of a master and one or more slaves however the SPI interface can not be a master in a multi-master system.

11.3.2 Main Features

- Full duplex synchronous transfers (on 3 lines)
- Simplex synchronous transfers (on 2 lines)
- Master or slave operation
- Six 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.

11.3.3 General Description

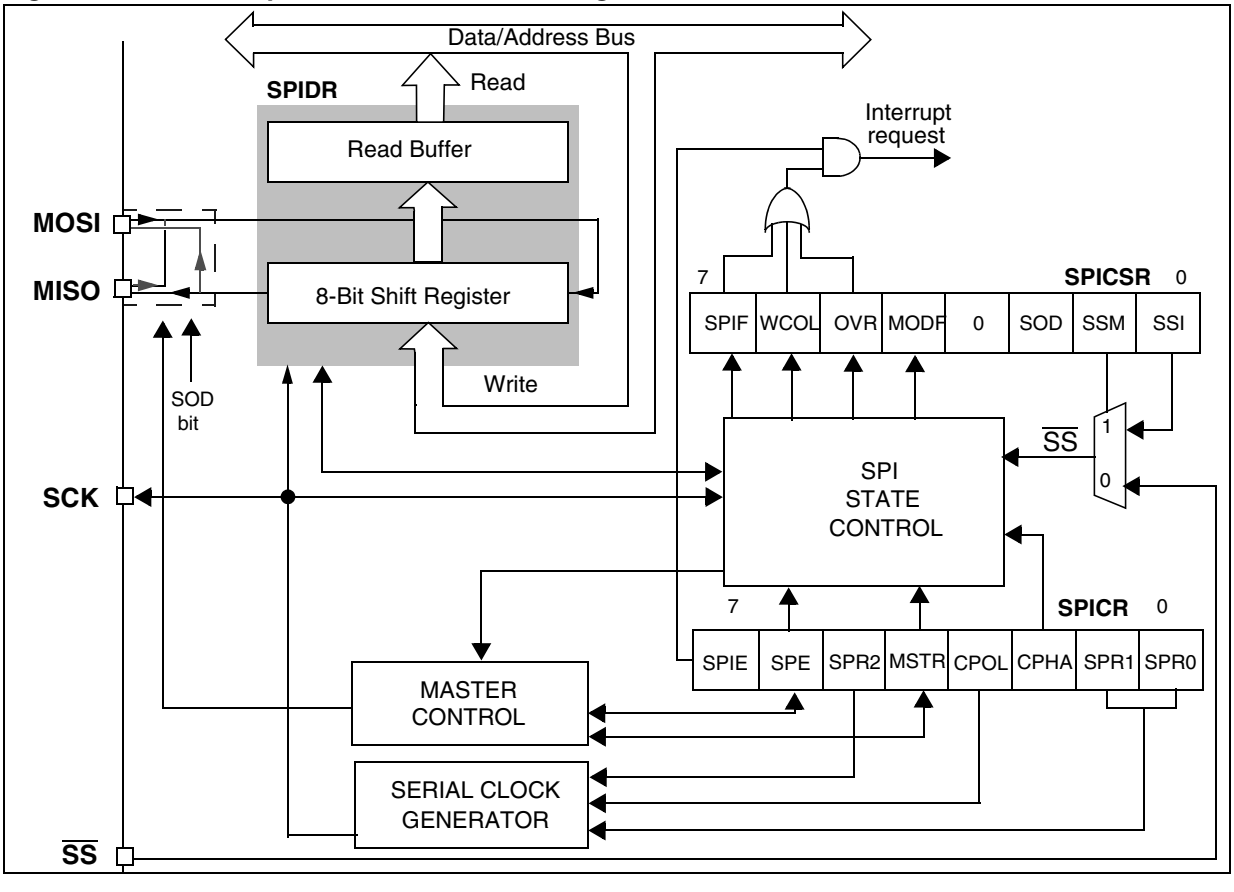
Figure 37 shows the serial peripheral interface (SPI) block diagram. There are 3 registers:

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

The SPI is connected to external devices through 3 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 MCU.

Figure 37. Serial Peripheral Interface Block Diagram



SERIAL PERIPHERAL INTERFACE (Cont'd)

11.3.6 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 MCU is woken up by an interrupt with "exit from HALT mode" capability. The data received is subsequently read from the SPIDR register when the software is running (interrupt vector fetching). If several data are received before the wake-up event, then an overrun error is generated. This error can be detected after the fetch of the interrupt routine that woke up the device.

11.3.6.1 Using the SPI to wakeup the MCU from Halt mode

In slave configuration, the SPI is able to wakeup the ST7 device from HALT mode through a SPIF interrupt. The data received is subsequently read from the SPIDR register when the software is running (interrupt vector fetch). If multiple data transfers have been performed before software clears the SPIF bit, then the OVR bit is set by hardware.

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

Caution: The SPI can wake up the ST7 from Halt mode only if the Slave Select signal (external $\overline{SS}$ pin or the SSI bit in the SPICSR register) is low when the ST7 enters Halt mode. So if Slave selection is configured as external (see Section 11.3.3.2), make sure the master drives a low level on the $\overline{SS}$ pin when the slave enters Halt mode.

11.3.7 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).

SERIAL PERIPHERAL INTERFACE (Cont'd)

11.3.8 Register Description

CONTROL REGISTER (SPICR)

Read/Write

Reset Value: 0000 xxxx (0xh)

7							0
SPIE	SPE	SPR2	MSTR	CPOL	CPHA	SPR1	SPR0

Bit 7 = **SPIE** *Serial Peripheral Interrupt Enable.*
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An SPI interrupt is generated whenever SPIF=1, MODF=1 or OVR=1 in the SPICSR register

Bit 6 = **SPE** *Serial Peripheral Output Enable.*
This bit is set and cleared by software. It is also cleared by hardware when, in master mode, $\overline{SS}$=0 (see Section 11.3.5.1 Master Mode Fault (MODF)). The SPE bit is cleared by reset, so the SPI peripheral is not initially connected to the external pins.
0: I/O pins free for general purpose I/O
1: SPI I/O pin alternate functions enabled

Bit 5 = **SPR2** *Divider Enable*.
This bit is set and cleared by software and is cleared by reset. It is used with the SPR[1:0] bits to set the baud rate. Refer to Table 15 SPI Master mode SCK Frequency.
0: Divider by 2 enabled
1: Divider by 2 disabled

Note: This bit has no effect in slave mode.

Bit 4 = **MSTR** *Master Mode.*
This bit is set and cleared by software. It is also cleared by hardware when, in master mode, $\overline{SS}$=0 (see Section 11.3.5.1 Master Mode Fault (MODF)).
0: Slave mode
1: Master mode. The function of the SCK pin changes from an input to an output and the functions of the MISO and MOSI pins are reversed.

Bit 3 = **CPOL** *Clock Polarity.*
This bit is set and cleared by software. This bit determines the idle state of the serial Clock. The CPOL bit affects both the master and slave modes.
0: SCK pin has a low level idle state
1: SCK pin has a high level idle state

Note: If CPOL is changed at the communication byte boundaries, the SPI must be disabled by resetting the SPE bit.

Bit 2 = **CPHA** *Clock Phase.*
This bit is set and cleared by software.
0: The first clock transition is the first data capture edge.
1: The second clock transition is the first capture edge.

Note: The slave must have the same CPOL and CPHA settings as the master.

Bits 1:0 = **SPR[1:0]** *Serial Clock Frequency.*
These bits are set and cleared by software. Used with the SPR2 bit, they select the baud rate of the SPI serial clock SCK output by the SPI in master mode.

Note: These 2 bits have no effect in slave mode.

Table 15. SPI Master mode SCK Frequency

Serial Clock	SPR2	SPR1	SPR0
$f_{CPU}/4$	1	0	0
$f_{CPU}/8$	0	0	0
$f_{CPU}/16$	0	0	1
$f_{CPU}/32$	1	1	0
$f_{CPU}/64$	0	1	0
$f_{CPU}/128$	0	1	1

12 INSTRUCTION SET

12.1 ST7 ADDRESSING MODES

The ST7 Core features 17 different addressing modes which can be classified in seven main groups:

Addressing Mode	Example
Inherent	nop
Immediate	ld A,#$55
Direct	ld A,$55
Indexed	ld A,($55,X)
Indirect	ld A,([$55],X)
Relative	jrne loop
Bit operation	bset byte,#5

The ST7 Instruction set is designed to minimize the number of bytes required per instruction: To do so, most of the addressing modes may be subdivided in two submodes called long and short:

– Long addressing mode is more powerful because it can use the full 64 Kbyte address space, however it uses more bytes and more CPU cycles.

– Short addressing mode is less powerful because it can generally only access page zero (0000h - 00FFh range), but the instruction size is more compact, and faster. All memory to memory instructions use short addressing modes only (CLR, CPL, NEG, BSET, BRES, BTJT, BTJF, INC, DEC, RLC, RRC, SLL, SRL, SRA, SWAP)

The ST7 Assembler optimizes the use of long and short addressing modes.

Table 18. ST7 Addressing Mode Overview

Mode			Syntax	Destination/ Source	Pointer Address (Hex.)	Pointer Size (Hex.)	Length (Bytes)
Inherent			nop				+ 0
Immediate			ld A,#$55				+ 1
Short	Direct		ld A,$10	00..FF			+ 1
Long	Direct		ld A,$1000	0000..FFFF			+ 2
No Offset	Direct	Indexed	ld A,(X)	00..FF			+ 0 (with X register) + 1 (with Y register)
Short	Direct	Indexed	ld A,($10,X)	00..1FE			+ 1
Long	Direct	Indexed	ld A,($1000,X)	0000..FFFF			+ 2
Short	Indirect		ld A,[$10]	00..FF	00..FF	byte	+ 2
Long	Indirect		ld A,[$10.w]	0000..FFFF	00..FF	word	+ 2
Short	Indirect	Indexed	ld A,([$10],X)	00..1FE	00..FF	byte	+ 2
Long	Indirect	Indexed	ld A,([$10.w],X)	0000..FFFF	00..FF	word	+ 2
Relative	Direct		jrne loop	PC-128/PC+127[1)]			+ 1
Relative	Indirect		jrne [$10]	PC-128/PC+127[1)]	00..FF	byte	+ 2
Bit	Direct		bset $10,#7	00..FF			+ 1
Bit	Indirect		bset [$10],#7	00..FF	00..FF	byte	+ 2
Bit	Direct	Relative	btjt $10,#7,skip	00..FF			+ 2
Bit	Indirect	Relative	btjt [$10],#7,skip	00..FF	00..FF	byte	+ 3

Note:
1. At the time the instruction is executed, the Program Counter (PC) points to the instruction following JRxx.

ST7 ADDRESSING MODES (cont'd)

12.1.6 Indirect Indexed (Short, Long)

This is a combination of indirect and short indexed addressing modes. The operand is referenced by its memory address, which is defined by the unsigned addition of an index register value (X or Y) with a pointer value located in memory. The pointer address follows the opcode.

The indirect indexed addressing mode consists of two submodes:

Indirect Indexed (Short)

The pointer address is a byte, the pointer size is a byte, thus allowing 00 - 1FE addressing space, and requires 1 byte after the opcode.

Indirect Indexed (Long)

The pointer address is a byte, the pointer size is a word, thus allowing 64 Kbyte addressing space, and requires 1 byte after the opcode.

Table 19. Instructions Supporting Direct, Indexed, Indirect and Indirect Indexed Addressing Modes

Long and Short Instructions	Function
LD	Load
CP	Compare
AND, OR, XOR	Logical Operations
ADC, ADD, SUB, SBC	Arithmetic Addition/subtraction operations
BCP	Bit Compare

Short Instructions Only	Function
CLR	Clear
INC, DEC	Increment/Decrement
TNZ	Test Negative or Zero
CPL, NEG	1 or 2 Complement
BSET, BRES	Bit Operations
BTJT, BTJF	Bit Test and Jump Operations
SLL, SRL, SRA, RLC, RRC	Shift and Rotate Operations
SWAP	Swap Nibbles
CALL, JP	Call or Jump subroutine

12.1.7 Relative Mode (Direct, Indirect)

This addressing mode is used to modify the PC register value by adding an 8-bit signed offset to it.

Available Relative Direct/ Indirect Instructions	Function
JRxx	Conditional Jump
CALLR	Call Relative

The relative addressing mode consists of two submodes:

Relative (Direct)

The offset follows the opcode.

Relative (Indirect)

The offset is defined in memory, of which the address follows the opcode.

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
JRULE	Jump if (C + Z = 1)	Unsigned <=							
LD	Load	dst <= src	reg, M	M, reg			N	Z	
MUL	Multiply	X,A = X * A	A, X, Y	X, Y, A	0				0
NEG	Negate (2's compl)	neg $10	reg, M				N	Z	C
NOP	No Operation								
OR	OR operation	A = A + M	A	M			N	Z	
POP	Pop from the Stack	pop reg	reg	M					
		pop CC	CC	M	H	I	N	Z	C
PUSH	Push onto the Stack	push Y	M	reg, CC					
RCF	Reset carry flag	C = 0							0
RET	Subroutine Return								
RIM	Enable Interrupts	I = 0				0			
RLC	Rotate left true C	C <= Dst <= C	reg, M				N	Z	C
RRC	Rotate right true C	C => Dst => C	reg, M				N	Z	C
RSP	Reset Stack Pointer	S = Max allowed							
SBC	Subtract with Carry	A = A - M - C	A	M			N	Z	C
SCF	Set carry flag	C = 1							1
SIM	Disable Interrupts	I = 1				1			
SLA	Shift left Arithmetic	C <= Dst <= 0	reg, M				N	Z	C
SLL	Shift left Logic	C <= Dst <= 0	reg, M				N	Z	C
SRL	Shift right Logic	0 => Dst => C	reg, M				0	Z	C
SRA	Shift right Arithmetic	Dst7 => Dst => C	reg, M				N	Z	C
SUB	Subtraction	A = A - M	A	M			N	Z	C
SWAP	SWAP nibbles	Dst[7..4] <=> Dst[3..0]	reg, M				N	Z	
TNZ	Test for Neg & Zero	tnz lbl1					N	Z	
TRAP	S/W trap	S/W interrupt				1			
WFI	Wait for Interrupt					0			
XOR	Exclusive OR	A = A XOR M	A	M			N	Z	

13.3.2 Operating Conditions with Low Voltage Detector (LVD)

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

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$V_{IT+(LVD)}$	Reset release threshold (V_{DD} rise)	High Threshold Med. Threshold Low Threshold	4.00 [1] 3.40 [1] 2.65 [1]	4.25 3.60 2.90	4.50 3.80 3.15	V
$V_{IT-(LVD)}$	Reset generation threshold (V_{DD} fall)	High Threshold Med. Threshold Low Threshold	3.80 3.20 2.40	4.05 3.40 2.70	4.30 [1] 3.65 [1] 2.90 [1]	
V_{hys}	LVD voltage threshold hysteresis	$V_{IT+(LVD)}$-$V_{IT-(LVD)}$		200		mV
Vt_{POR}	V_{DD} rise time rate [2]		20		20000	µs/V
$t_{g(VDD)}$	Filtered glitch delay on V_{DD}	Not detected by the LVD			150	ns
$I_{DD(LVD)}$	LVD/AVD current consumption			220		µA

Notes:

1. Not tested in production.

2. Not tested in production. The V_{DD} rise time rate condition is needed to ensure a correct device power-on and LVD reset. When the V_{DD} slope is outside these values, the LVD may not ensure a proper reset of the MCU.

13.3.3 Auxiliary Voltage Detector (AVD) Thresholds

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

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$V_{IT+(AVD)}$	1=>0 AVDF flag toggle threshold (V_{DD} rise)	High Threshold Med. Threshold Low Threshold	4.40 3.90 3.20	4.70 4.10 3.40	5.00 4.30 3.60	V
$V_{IT-(AVD)}$	0=>1 AVDF flag toggle threshold (V_{DD} fall)	High Threshold Med. Threshold Low Threshold	4.30 3.70 2.90	4.60 3.90 3.20	4.90 4.10 3.40	
V_{hys}	AVD voltage threshold hysteresis	$V_{IT+(AVD)}$-$V_{IT-(AVD)}$		150		mV
ΔV_{IT-}	Voltage drop between AVD flag set and LVD reset activation	V_{DD} fall		0.45		V

13.3.4 Internal RC Oscillator and PLL

The ST7 internal clock can be supplied by an internal RC oscillator and PLL (selectable by option byte).

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$V_{DD(RC)}$	Internal RC Oscillator operating voltage		2.4		5.5	V
$V_{DD(x4PLL)}$	x4 PLL operating voltage		2.4		3.3	
$V_{DD(x8PLL)}$	x8 PLL operating voltage		3.3		5.5	
$t_{STARTUP}$	PLL Startup time			60		PLL input clock (f_{PLL}) cycles

The RC oscillator and PLL characteristics are temperature-dependent and are grouped in two 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} [1]	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[3]		+2[3]	%
$I_{DD(RC)}$	RC oscillator current consumption	T_A=25°C,V_{DD}=5V		970[3]		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[3]		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[6]		kHz
JIT_{PLL}	PLL jitter ($\Delta f_{CPU}/f_{CPU}$)			1[6]		%
$I_{DD(PLL)}$	PLL current consumption	T_A=25°C		600[3]		µA

Notes:

1. If the RC oscillator clock is selected, to improve clock stability and frequency accuracy, it is recommended to place a decoupling capacitor, typically 100nF, between the V_{DD} and V_{SS} pins as close as possible to the ST7 device.

2. See "INTERNAL RC OSCILLATOR ADJUSTMENT" on page 24

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

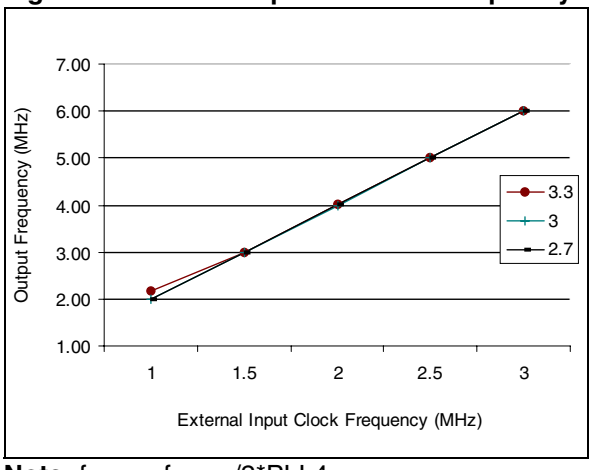
4. Averaged over a 4ms period. After the LOCKED bit is set, a period of t_{STAB} is required to reach ACC_{PLL} accuracy

5. After the LOCKED bit is set ACC_{PLL} is max. 10% until t_{STAB} has elapsed. See Figure 13 on page 25.

6. Guaranteed by design.

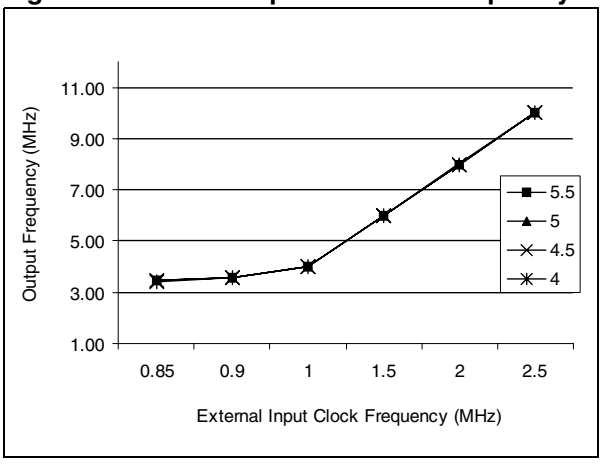
OPERATING CONDITIONS (Cont'd)

Figure 54. PLLx4 Output vs CLKIN frequency



Note: $f_{OSC} = f_{CLKIN}/2*PLL4$

Figure 55. PLLx8 Output vs CLKIN frequency



Note: $f_{OSC} = f_{CLKIN}/2*PLL8$

13.6 MEMORY CHARACTERISTICS

T_A = -40°C to 105°C, unless otherwise specified

13.6.1 RAM and Hardware Registers

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{RM}	Data retention mode [1]	HALT mode (or RESET)	1.6			V

13.6.2 FLASH Program Memory

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{DD}	Operating voltage for Flash write/erase		2.4		5.5	V
t_{prog}	Programming time for 1~32 bytes [2]	T_A=–40 to +105°C		5	10	ms
	Programming time for 1.5 kBytes	T_A=+25°C		0.24	0.48	s
t_{RET}	Data retention [4]	T_A=+55°C[3]	20			years
N_{RW}	Write erase cycles	T_A=+25°C	10K [7]			cycles
I_{DD}	Supply current	Read / Write / Erase modes f_{CPU} = 8MHz, V_{DD} = 5.5V			2.6 [6]	mA
		No Read/No Write Mode			100	µA
		Power down mode / HALT		0	0.1	µA

13.6.3 EEPROM Data Memory

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{DD}	Operating voltage for EEPROM write/erase		2.4		5.5	V
t_{prog}	Programming time for 1~32 bytes	T_A=–40 to +105°C		5	10	ms
t_{ret}	Data retention [4]	T_A=+55°C [3]	20			years
N_{RW}	Write erase cycles	T_A=+25°C	300K [7]			cycles

Notes:

1. Minimum V_{DD} supply voltage without losing data stored in RAM (in HALT mode or under RESET) or in hardware registers (only in HALT mode). Guaranteed by construction, not tested in production.

2. Up to 32 bytes can be programmed at a time.

3. The data retention time increases when the T_A decreases.

4. Data based on reliability test results and monitored in production.

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

6. Guaranteed by Design. Not tested in production.

7. Design target value pending full product characterization.

I/O PORT PIN CHARACTERISTICS (Cont'd)

13.8.2 Output Driving Current

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

Symbol	Parameter	Conditions			Min	Max	Unit
V_{OL} 1)	Output low level voltage for a standard I/O pin when 8 pins are sunk at same time (see Figure 65)	V_{DD}=5V	I_{IO}=+5mA	$T_A \le 85°C$ $T_A \ge 85°C$		1.0 1.2	V
			I_{IO}=+2mA	$T_A \le 85°C$ $T_A \ge 85°C$		0.4 0.5	
	Output low level voltage for a high sink I/O pin when 4 pins are sunk at same time (see Figure 66)		I_{IO}=+20mA,	$T_A \le 85°C$ $T_A \ge 85°C$		1.3 1.5	
			I_{IO}=+8mA	$T_A \le 85°C$ $T_A \ge 85°C$		0.75 0.85	
V_{OH} 2)	Output high level voltage for an I/O pin when 4 pins are sourced at same time (see Figure 72)		I_{IO}=-5mA,	$T_A \le 85°C$ $T_A \ge 85°C$	V_{DD}-1.5 V_{DD}-1.6		
			I_{IO}=-2mA	$T_A \le 85°C$ $T_A \ge 85°C$	V_{DD}-0.8 V_{DD}-1.0		
V_{OL} 1)3)	Output low level voltage for a standard I/O pin when 8 pins are sunk at same time (see Figure 64)	V_{DD}=3.3V	I_{IO}=+2mA	$T_A \le 85°C$ $T_A \ge 85°C$		0.5 0.6	
	Output low level voltage for a high sink I/O pin when 4 pins are sunk at same time		I_{IO}=+8mA	$T_A \le 85°C$ $T_A \ge 85°C$		0.5 0.6	
V_{OH} 2)3)	Output high level voltage for an I/O pin when 4 pins are sourced at same time		I_{IO}=-2mA	$T_A \le 85°C$ $T_A \ge 85°C$	V_{DD}-0.8 V_{DD}-1.0		
V_{OL} 1)3)	Output low level voltage for a standard I/O pin when 8 pins are sunk at same time	V_{DD}=2.7V	I_{IO}=+2mA	$T_A \le 85°C$ $T_A \ge 85°C$		0.6 0.7	
	Output low level voltage for a high sink I/O pin when 4 pins are sunk at same time		I_{IO}=+8mA	$T_A \le 85°C$ $T_A \ge 85°C$		0.6 0.7	
V_{OH} 2)3)	Output high level voltage for an I/O pin when 4 pins are sourced at same time (see Figure 69)		I_{IO}=-2mA	$T_A \le 85°C$ $T_A \ge 85°C$	V_{DD}-0.9 V_{DD}-1.0		

Notes:

1. The I_{IO} current sunk must always respect the absolute maximum rating specified in Section 13.2.2 and the sum of I_{IO} (I/O ports and control pins) must not exceed I_{VSS}.

2. The I_{IO} current sourced must always respect the absolute maximum rating specified in Section 13.2.2 and the sum of I_{IO} (I/O ports and control pins) must not exceed I_{VDD}.

3. Not tested in production, based on characterization results.

13.9 CONTROL PIN CHARACTERISTICS

13.9.1 Asynchronous $\overline{\text{RESET}}$ Pin

T_A = -40°C to 105°C, unless otherwise specified

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
V_{IL}	Input low level voltage			V_{SS} - 0.3		0.3xV_{DD}	V
V_{IH}	Input high level voltage			0.7xV_{DD}		V_{DD} + 0.3	
V_{hys}	Schmitt trigger voltage hysteresis [1]				2		V
V_{OL}	Output low level voltage [2]	V_{DD}=5V	I_{IO}=+5mA $T_A \leq 85°C$ $T_A \leq 105°C$		0.5	1.0 1.2	V
			I_{IO}=+2mA $T_A \leq 85°C$ $T_A \leq 105°C$		0.2	0.4 0.5	
R_{ON}	Pull-up equivalent resistor [3] [1]	V_{DD}=5V		20	40	80	kΩ
$t_{w(RSTL)out}$	Generated reset pulse duration	Internal reset sources			30		µs
$t_{h(RSTL)in}$	External reset pulse hold time [4]			20			µs
$t_{g(RSTL)in}$	Filtered glitch duration				200		ns

Notes:

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

2. The I_{IO} current sunk must always respect the absolute maximum rating specified in section 13.2.2 on page 82 and the sum of I_{IO} (I/O ports and control pins) must not exceed I_{VSS}.

3. The R_{ON} pull-up equivalent resistor is based on a resistive transistor. Specified for voltages on $\overline{\text{RESET}}$ pin between V_{ILmax} and V_{DD}

4. To guarantee the reset of the device, a minimum pulse has to be applied to the $\overline{\text{RESET}}$ pin. All short pulses applied on $\overline{\text{RESET}}$ pin with a duration below $t_{h(RSTL)in}$ can be ignored.

ADC CHARACTERISTICS (Cont'd)

Figure 82. R_{AIN} max. vs f_{ADC} with C_{AIN}=0pF[1)]

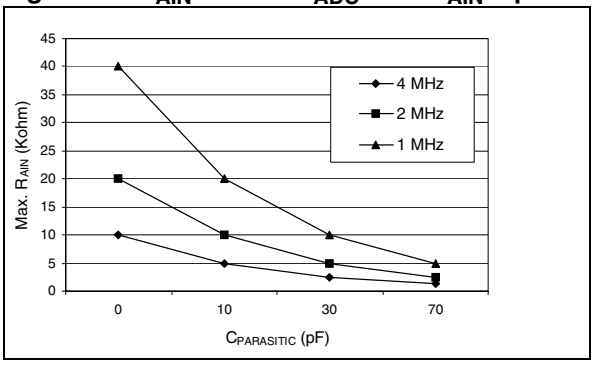
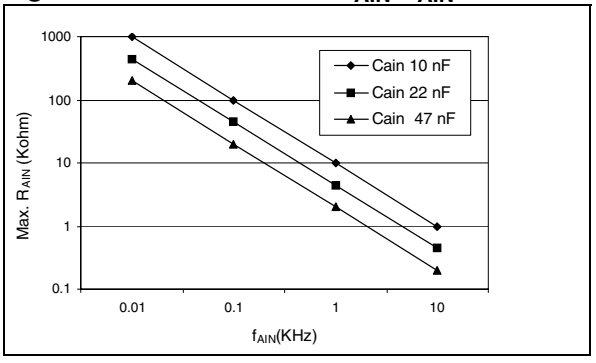


Figure 83. Recommended C_{AIN}/R_{AIN} values[2)]



Notes:

1. $C_{PARASITIC}$ represents the capacitance of the PCB (dependent on soldering and PCB layout quality) plus the pad capacitance (3pF). A high $C_{PARASITIC}$ value will downgrade conversion accuracy. To remedy this, f_{ADC} should be reduced.

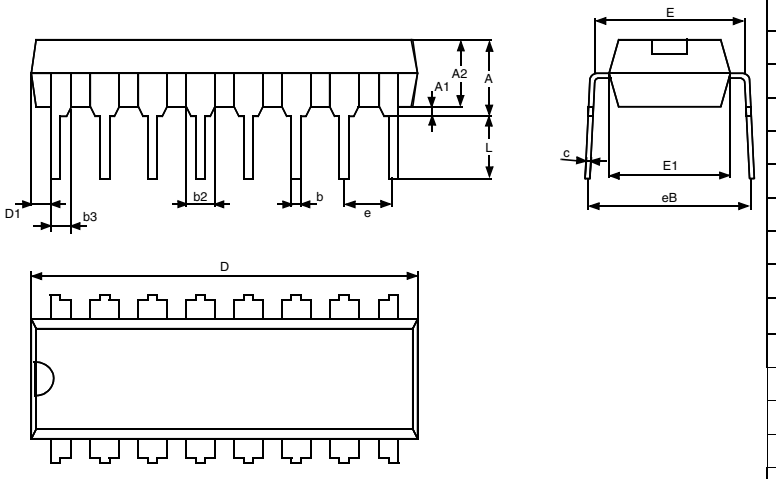
2. This graph shows that depending on the input signal variation (f_{AIN}), C_{AIN} can be increased for stabilization and to allow the use of a larger serial resistor (R_{AIN}). It is valid for all f_{ADC} frequencies ≤ 4MHz.

13.11.1 General PCB Design Guidelines

To obtain best results, some general design and layout rules should be followed when designing the application PCB to shield the noise-sensitive, analog physical interface from noise-generating CMOS logic signals.

Properly place components and route the signal traces on the PCB to shield the analog inputs. Analog signals paths should run over the analog ground plane and be as short as possible. Isolate analog signals from digital signals that may switch while the analog inputs are being sampled by the A/D converter. Do not toggle digital outputs on the same I/O port as the A/D input being converted.

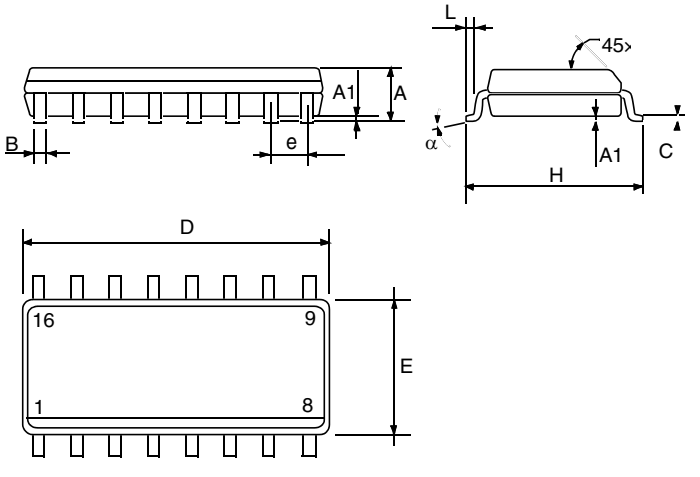
Figure 87. 16-Pin Plastic Dual In-Line Package, 300-mil Width



Dim	mm			inches[1]		
	Min	Typ	Max	Min	Typ	Max
A			5.33			0.2098
A1	0.38			0.0150		
A2	2.92	3.30	4.95	0.1150	0.1299	0.1949
b	0.36	0.46	0.56	0.0142	0.0181	0.0220
b2	1.14	1.52	1.78	0.0449	0.0598	0.0701
b3	0.76	0.99	1.14	0.0299	0.0390	0.0449
c	0.20	0.25	0.36	0.0079	0.0098	0.0142
D	18.67	19.18	19.69	0.7350	0.7551	0.7752
D1	0.13			0.0051		
e		2.54			0.1000	
E	7.62	7.87	8.26	0.3000	0.3098	0.3252
E1	6.10	6.35	7.11	0.2402	0.2500	0.2799
L	2.92	3.30	3.81	0.1150	0.1299	0.1500
eB			10.92			0.4299
	Number of Pins					
N	16					

Note 1. Values in inches are converted from mm and rounded to 4 decimal digits.

Figure 88. 16-Pin Plastic Small Outline Package, 150-mil Width



Dim.	mm			inches[1]		
	Min	Typ	Max	Min	Typ	Max
A	1.35		1.75	0.0531		0.0689
A1	0.10		0.25	0.0039		0.0098
B	0.33		0.51	0.0130		0.0201
C	0.19		0.25	0.0075		0.0098
D	9.80		10.00	0.3858		0.3937
E	3.80		4.00	0.1496		0.1575
e		1.27			0.0500	
H	5.80		6.20	0.2283		0.2441
α	0°		8°	0°		8°
L	0.40		1.27	0.0157		0.0500
	Number of Pins					
N	16					

Note 1. Values in inches are converted from mm and rounded to 4 decimal digits.

Table 24. ST7 Application Notes

IDENTIFICATION	DESCRIPTION
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 AN ST72324 TARGET APPLICATION
AN1477	EMULATED DATA EEPROM WITH XFLASH MEMORY
AN1527	DEVELOPING A USB SMARTCARD READER WITH ST7SCR
AN1575	ON-BOARD PROGRAMMING METHODS FOR XFLASH AND HDFLASH ST7 MCUS
AN1576	IN-APPLICATION PROGRAMMING (IAP) DRIVERS FOR ST7 HDFLASH OR XFLASH MCUS
AN1577	DEVICE FIRMWARE UPGRADE (DFU) IMPLEMENTATION FOR ST7 USB APPLICATIONS
AN1601	SOFTWARE IMPLEMENTATION FOR ST7DALI-EVAL
AN1603	USING THE ST7 USB DEVICE FIRMWARE UPGRADE DEVELOPMENT KIT (DFU-DK)
AN1635	ST7 CUSTOMER ROM CODE RELEASE INFORMATION
AN1754	DATA LOGGING PROGRAM FOR TESTING ST7 APPLICATIONS VIA ICC
AN1796	FIELD UPDATES FOR FLASH BASED ST7 APPLICATIONS USING A PC COMM PORT
AN1900	HARDWARE IMPLEMENTATION FOR ST7DALI-EVAL
AN1904	ST7MC THREE-PHASE AC INDUCTION MOTOR CONTROL SOFTWARE LIBRARY
AN1905	ST7MC THREE-PHASE BLDC MOTOR CONTROL SOFTWARE LIBRARY
SYSTEM OPTIMIZATION	
AN1711	SOFTWARE TECHNIQUES FOR COMPENSATING ST7 ADC ERRORS
AN1827	IMPLEMENTATION OF SIGMA-DELTA ADC WITH ST7FLITE05/09
AN2009	PWM MANAGEMENT FOR 3-PHASE BLDC MOTOR DRIVES USING THE ST7FMC
AN2030	BACK EMF DETECTION DURING PWM ON TIME BY ST7MC