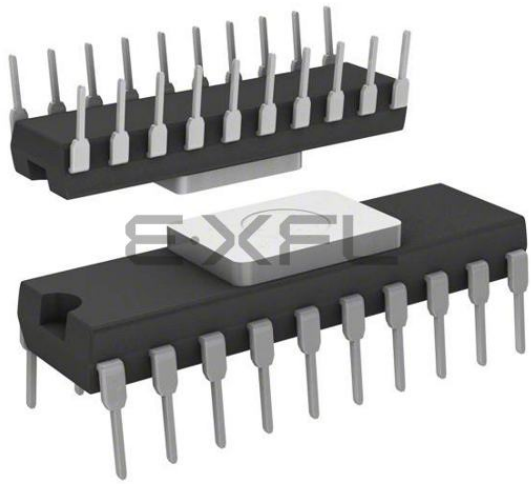


Welcome to E-XFL.COM

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	ST6
Core Size	8-Bit
Speed	8MHz
Connectivity	-
Peripherals	LVD, POR, WDT
Number of I/O	12
Program Memory Size	4KB (4K x 8)
Program Memory Type	EPROM, UV
EEPROM Size	-
RAM Size	64 x 8
Voltage - Supply (Vcc/Vdd)	3V ~ 6V
Data Converters	A/D 8x8b
Oscillator Type	Internal
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Through Hole
Package / Case	20-CDIP (0.300", 7.62mm) Window
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/stmicroelectronics/st62e20cf1

Table of Contents

1 GENERAL DESCRIPTION

1.1 INTRODUCTION

The ST62T08C,T09C,T10C,T20C and ST62E20C devices are low cost members of the ST62xx 8-bit HCMOS family of microcontrollers, which is targeted at low to medium complexity applications. All ST62xx devices are based on a building block approach: a common core is surrounded by a number of on-chip peripherals.

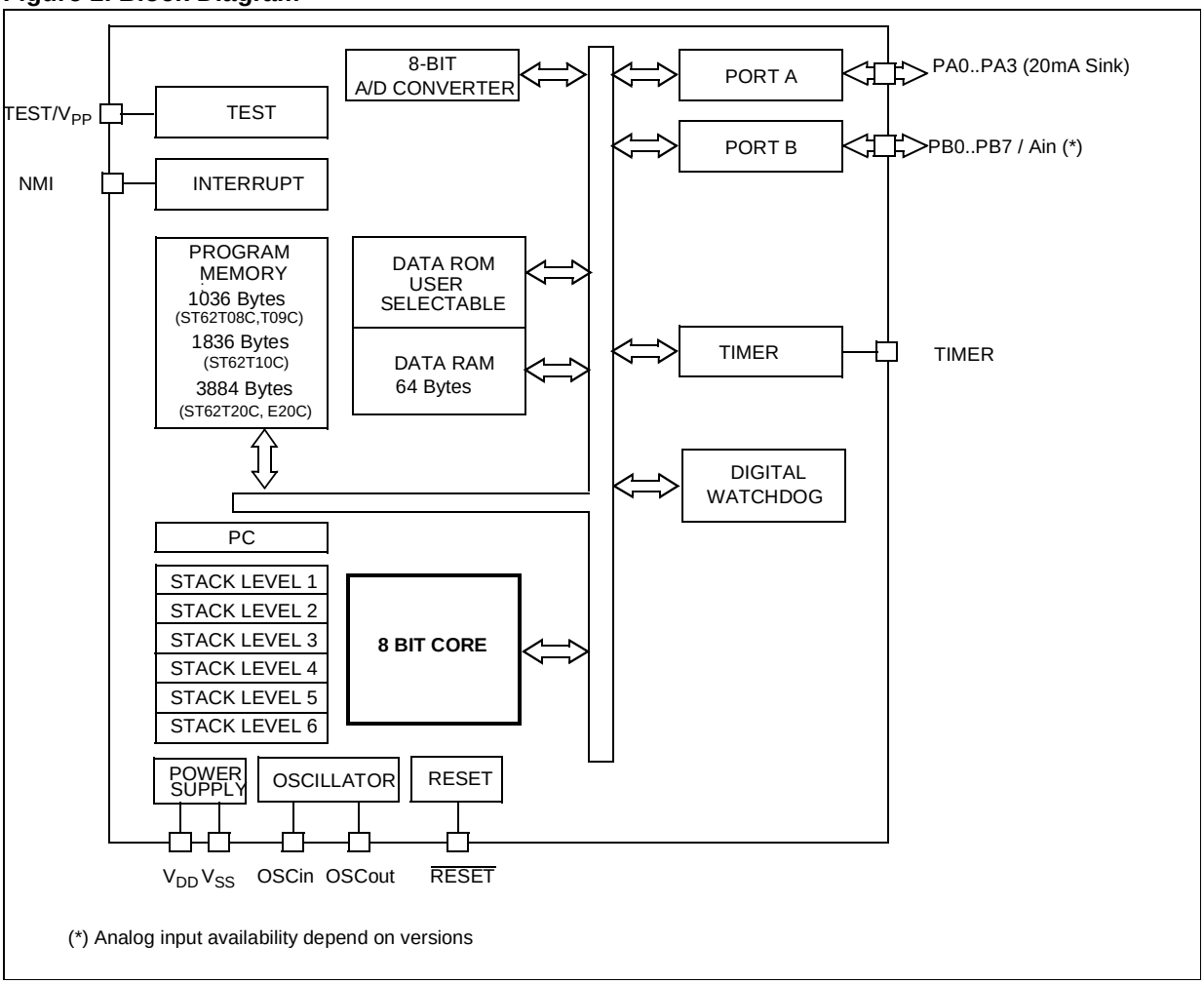
The ST62E20C is the erasable EPROM version of the ST62T08C,T09C,T10C and T20C device, which may be used to emulate the ST62T08C,T09C,T10C and T20C device, as well as the respective ST6208C,09C,10C,20C ROM devices.

OTP and EPROM devices are functionally identical. The ROM based versions offer the same functionality selecting as ROM options the options defined in the programmable option bytes of the OTP/EPROM versions.

OTP devices offer all the advantages of user programmability at low cost, which make them the ideal choice in a wide range of applications where frequent code changes, multiple code versions or last minute programmability are required.

These compact low-cost devices feature a Timer comprising an 8-bit counter and a 7-bit programmable prescaler,an 8-bit A/D Converter with up to 8 analog inputs and a Digital Watchdog timer, making them well suited for a wide range of automotive, appliance and industrial applications.

Figure 1. Block Diagram



(*) Analog input availability depend on versions

MEMORY MAP (Cont'd)

1.3.5 Data Window Register (DWR)

The Data read-only memory window is located from address 0040h to address 007Fh in Data space. It allows direct reading of 64 consecutive bytes located anywhere in program memory, between address 0000h and 0FFFh (top memory address depends on the specific device). All the program memory can therefore be used to store either instructions or read-only data. Indeed, the window can be moved in steps of 64 bytes along the program memory by writing the appropriate code in the Data Window Register (DWR).

The DWR can be addressed like any RAM location in the Data Space, it is however a write-only register and therefore cannot be accessed using single-bit operations. This register is used to position the 64-byte read-only data window (from address 40h to address 7Fh of the Data space) in program memory in 64-byte steps. The effective address of the byte to be read as data in program memory is obtained by concatenating the 6 least significant bits of the register address given in the instruction (as least significant bits) and the content of the DWR register (as most significant bits), as illustrated in Figure 5 below. For instance, when addressing location 0040h of the Data Space, with 0 loaded in the DWR register, the physical location addressed in program memory is 00h. The DWR register is not cleared on reset, therefore it must be written to prior to the first access to the Data read-only memory window area.

Data Window Register (DWR)

Address: 0C9h — Write Only

7							0
-	-	DWR5	DWR4	DWR3	DWR2	DWR1	DWR0

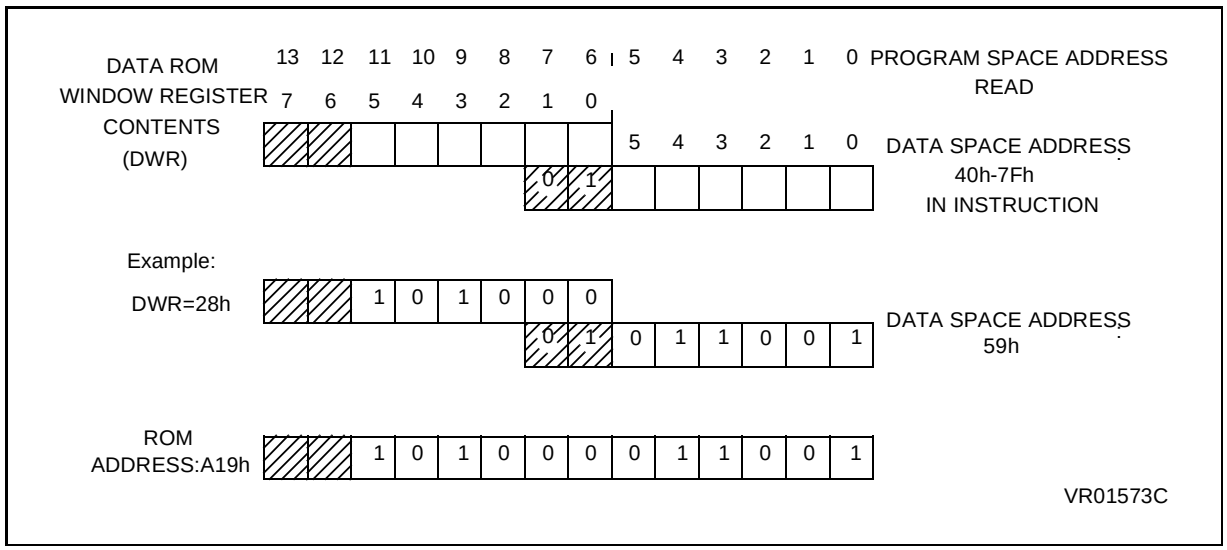
Bits 6, 7 = Not used.

Bit 5-0 = **DWR5-DWR0:** *Data read-only memory Window Register Bits.* These are the Data read-only memory Window bits that correspond to the upper bits of the data read-only memory space.

Caution: *This register is undefined on reset. Neither read nor single bit instructions may be used to address this register.*

Note: Care is required when handling the DWR register as it is write only. For this reason, the DWR contents should not be changed while executing an interrupt service routine, as the service routine cannot save and then restore the register's previous contents. If it is impossible to avoid writing to the DWR during the interrupt service routine, an image of the register must be saved in a RAM location, and each time the program writes to the DWR, it must also write to the image register. The image register must be written first so that, if an interrupt occurs between the two instructions, the DWR is not affected.

Figure 5. Data read-only memory Window Memory Addressing



CPU REGISTERS (Cont'd)

However, if the program space contains more than 4096 bytes, the additional memory in program space can be addressed by using the Program Bank Switch register.

The PC value is incremented after reading the address of the current instruction. To execute relative jumps, the PC and the offset are shifted through the ALU, where they are added; the result is then shifted back into the PC. The program counter can be changed in the following ways:

- JP (Jump) instructionPC=Jump address
- CALL instructionPC= Call address
- Relative Branch Instruction.PC= PC +/- offset
- Interrupt PC=Interrupt vector
- Reset PC= Reset vector
- RET & RETI instructionsPC= Pop (stack)
- Normal instructionPC= PC + 1

Flags (C, Z). The ST6 CPU includes three pairs of flags (Carry and Zero), each pair being associated with one of the three normal modes of operation: Normal mode, Interrupt mode and Non Maskable Interrupt mode. Each pair consists of a CARRY flag and a ZERO flag. One pair (CN, ZN) is used during Normal operation, another pair is used during Interrupt mode (CI, ZI), and a third pair is used in the Non Maskable Interrupt mode (CNMI, ZNMI).

The ST6 CPU uses the pair of flags associated with the current mode: as soon as an interrupt (or a Non Maskable Interrupt) is generated, the ST6 CPU uses the Interrupt flags (resp. the NMI flags) instead of the Normal flags. When the RETI instruction is executed, the previously used set of flags is restored. It should be noted that each flag set can only be addressed in its own context (Non Maskable Interrupt, Normal Interrupt or Main routine). The flags are not cleared during context switching and thus retain their status.

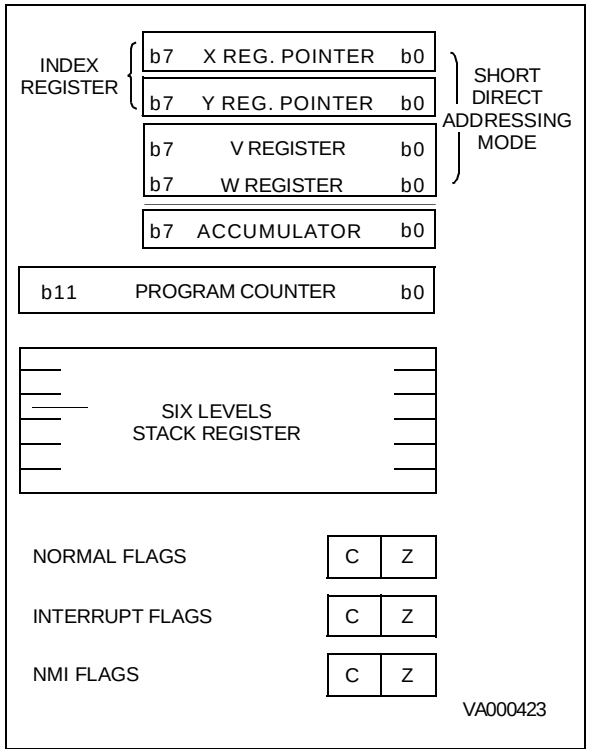
The Carry flag is set when a carry or a borrow occurs during arithmetic operations; otherwise it is cleared. The Carry flag is also set to the value of the bit tested in a bit test instruction; it also participates in the rotate left instruction.

The Zero flag is set if the result of the last arithmetic or logical operation was equal to zero; otherwise it is cleared.

Switching between the three sets of flags is performed automatically when an NMI, an interrupt or a RETI instructions occurs. As the NMI mode is automatically selected after the reset of the MCU, the ST6 core uses at first the NMI flags.

Stack. The ST6 CPU includes a true LIFO hardware stack which eliminates the need for a stack pointer. The stack consists of six separate 12-bit RAM locations that do not belong to the data space RAM area. When a subroutine call (or interrupt request) occurs, the contents of each level are shifted into the next higher level, while the content of the PC is shifted into the first level (the original contents of the sixth stack level are lost). When a subroutine or interrupt return occurs (RET or RETI instructions), the first level register is shifted back into the PC and the value of each level is popped back into the previous level. Since the accumulator, in common with all other data space registers, is not stored in this stack, management of these registers should be performed within the subroutine. The stack will remain in its "deepest" position if more than 6 nested calls or interrupts are executed, and consequently the last return address will be lost. It will also remain in its highest position if the stack is empty and a RET or RETI is executed. In this case the next instruction will be executed.

Figure 7. ST6 CPU Programming Mode



3.2 RESETS

The MCU can be reset in four ways:

– by the external Reset input being pulled low;

– by Power-on Reset;

– by the digital Watchdog peripheral timing out.

– by Low Voltage Detection (LVD)

3.2.1 RESET Input

The $\overline{\text{RESET}}$ pin may be connected to a device of the application board in order to reset the MCU if required. The $\overline{\text{RESET}}$ pin may be pulled low in RUN, WAIT or STOP mode. This input can be used to reset the MCU internal state and ensure a correct start-up procedure. The pin is active low and features a Schmitt trigger input. The internal Reset signal is generated by adding a delay to the external signal. Therefore even short pulses on the $\overline{\text{RESET}}$ pin are acceptable, provided V_{DD} has completed its rising phase and that the oscillator is running correctly (normal RUN or WAIT modes). The MCU is kept in the Reset state as long as the $\overline{\text{RESET}}$ pin is held low.

If $\overline{\text{RESET}}$ activation occurs in the RUN or WAIT modes, processing of the user program is stopped (RUN mode only), the Inputs and Outputs are configured as inputs with pull-up resistors and the main Oscillator is restarted. When the level on the RESET pin then goes high, the initialization sequence is executed following expiry of the internal delay period.

If $\overline{\text{RESET}}$ pin activation occurs in the STOP mode, the oscillator starts up and all Inputs and Outputs are configured as inputs with pull-up resistors. When the level of the $\overline{\text{RESET}}$ pin then goes high, the initialization sequence is executed following expiry of the internal delay period.

3.2.2 Power-on Reset

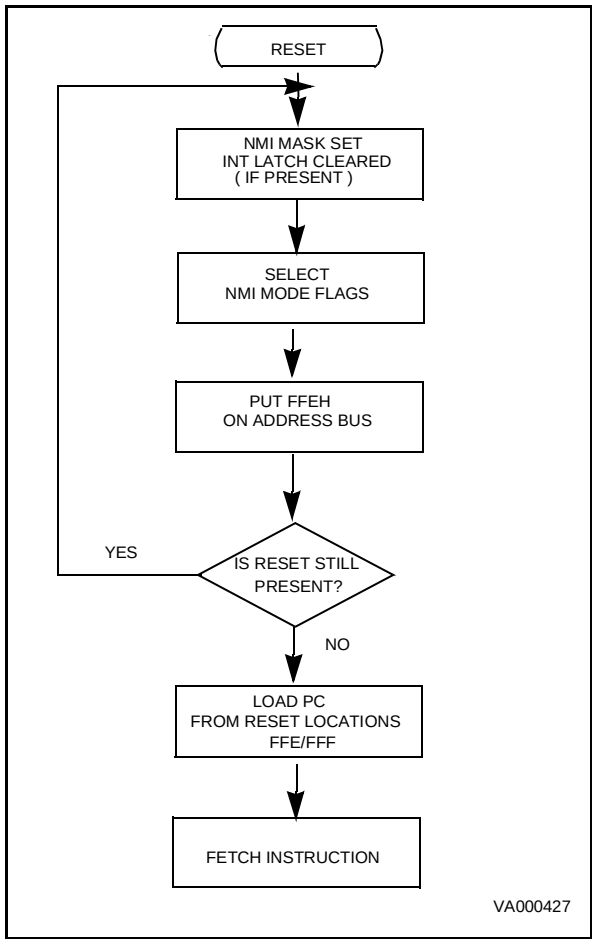
The function of the POR circuit consists in waking up the MCU by detecting around 2V a dynamic (rising edge) variation of the VDD Supply. At the beginning of this sequence, the MCU is configured in the Reset state: all I/O ports are configured as inputs with pull-up resistors and no instruction is executed. When the power supply voltage rises to a sufficient level, the oscillator starts to operate, whereupon an internal delay is initiated, in order to allow the oscillator to fully stabilize before executing the first instruction. The initialization sequence is executed immediately following the internal delay.

To ensure correct start-up, the user should take care that the VDD Supply is stabilized at a sufficient level for the chosen frequency (see recommended operation) before the reset signal is released. In addition, supply rising must start from 0V.

As a consequence, the POR does not allow to supervise static, slowly rising, or falling, or noisy (presenting oscillation) VDD supplies.

An external RC network connected to the $\overline{\text{RESET}}$ pin, or the LVD reset can be used instead to get the best performances.

Figure 13. Reset and Interrupt Processing



DIGITAL WATCHDOG (Cont'd)

3.3.1 Digital Watchdog Register (DWDR)

Address: 0D8h — Read/Write

Reset status: 1111 1110b

7							0
T0	T1	T2	T3	T4	T5	SR	C

Bit 0 = **C**: *Watchdog Control bit*

If the hardware option is selected, this bit is forced high and the user cannot change it (the Watchdog is always active). When the software option is selected, the Watchdog function is activated by setting bit C to 1, and cannot then be disabled (save by resetting the MCU).

When C is kept low the counter can be used as a 7-bit timer.

This bit is cleared to "0" on Reset.

Bit 1 = **SR**: *Software Reset bit*

This bit triggers a Reset when cleared.

When C = "0" (Watchdog disabled) it is the MSB of the 7-bit timer.

This bit is set to "1" on Reset.

Bits 2-7 = **T5-T0**: *Downcounter bits*

It should be noted that the register bits are reversed and shifted with respect to the physical counter: bit-7 (T0) is the LSB of the Watchdog downcounter and bit-2 (T5) is the MSB.

These bits are set to "1" on Reset.

3.3.2 Application Notes

The Watchdog plays an important supporting role in the high noise immunity of ST62xx devices, and should be used wherever possible. Watchdog related options should be selected on the basis of a trade-off between application security and STOP mode availability.

When STOP mode is not required, hardware activation without EXTERNAL STOP MODE CONTROL should be preferred, as it provides maximum security, especially during power-on.

When STOP mode is required, hardware activation and EXTERNAL STOP MODE CONTROL should be chosen. NMI should be high by default, to allow STOP mode to be entered when the MCU is idle.

The NMI pin can be connected to an I/O line (see Figure 18) to allow its state to be controlled by software. The I/O line can then be used to keep NMI low while Watchdog protection is required, or to avoid noise or key bounce. When no more processing is required, the I/O line is released and the device placed in STOP mode for lowest power consumption.

When software activation is selected and the Watchdog is not activated, the downcounter may be used as a simple 7-bit timer (remember that the bits are in reverse order).

The software activation option should be chosen only when the Watchdog counter is to be used as a timer. To ensure the Watchdog has not been unexpectedly activated, the following instructions should be executed within the first 27 instructions:

```
jrr 0, WD, #+3
ldi WD, 0FDH
```

DIGITAL WATCHDOG (Cont'd)

These instructions test the C bit and Reset the MCU (i.e. disable the Watchdog) if the bit is set (i.e. if the Watchdog is active), thus disabling the Watchdog.

In all modes, a minimum of 28 instructions are executed after activation, before the Watchdog can generate a Reset. Consequently, user software should load the watchdog counter within the first 27 instructions following Watchdog activation (software mode), or within the first 27 instructions executed following a Reset (hardware activation).

It should be noted that when the GEN bit is low (interrupts disabled), the NMI interrupt is active but cannot cause a wake up from STOP/WAIT modes.

Figure 18. A typical circuit making use of the EXERNAL STOP MODE CONTROL feature

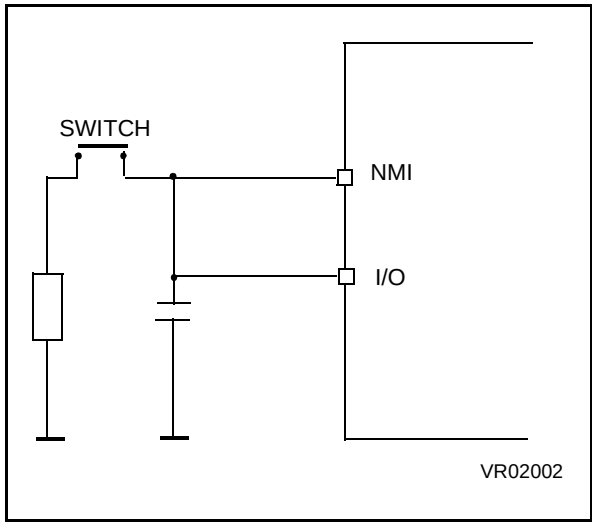
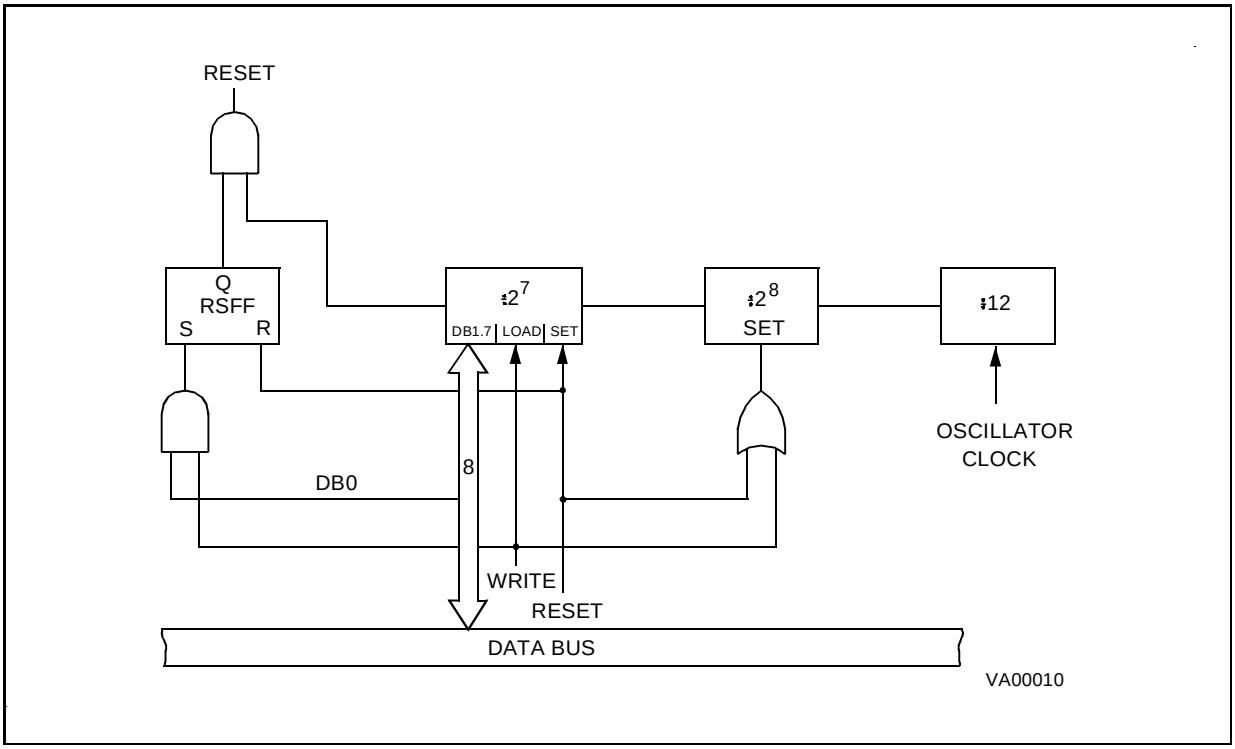


Figure 19. Digital Watchdog Block Diagram



INTERRUPTS (Cont'd)

3.4.2 Interrupt Procedure

The interrupt procedure is very similar to a call procedure, indeed the user can consider the interrupt as an asynchronous call procedure. As this is an asynchronous event, the user cannot know the context and the time at which it occurred. As a result, the user should save all Data space registers which may be used within the interrupt routines. There are separate sets of processor flags for normal, interrupt and non-maskable interrupt modes, which are automatically switched and so do not need to be saved.

The following list summarizes the interrupt procedure:

MCU

– The interrupt is detected.
– The C and Z flags are replaced by the interrupt flags (or by the NMI flags).
– The PC contents are stored in the first level of the stack.
– The normal interrupt lines are inhibited (NMI still active).
– The first internal latch is cleared.
– The associated interrupt vector is loaded in the PC.

WARNING: In some circumstances, when a maskable interrupt occurs while the ST6 core is in NORMAL mode and especially during the execution of an "ldi IOR, 00h" instruction (disabling all maskable interrupts): if the interrupt arrives during the first 3 cycles of the "ldi" instruction (which is a 4-cycle instruction) the core will switch to interrupt mode BUT the flags CN and ZN will NOT switch to the interrupt pair CI and ZI.

User

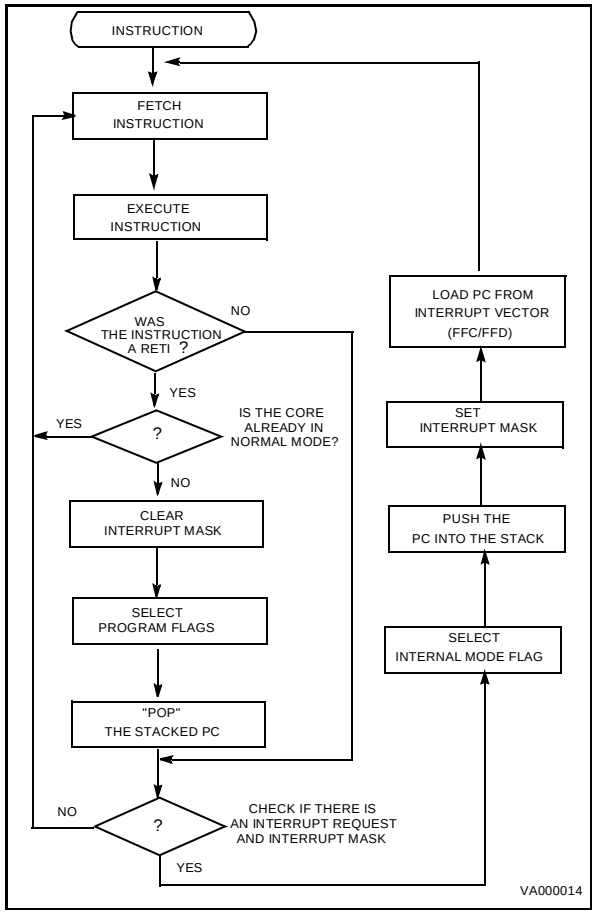
– User selected registers are saved within the interrupt service routine (normally on a software stack).
– The source of the interrupt is found by polling the interrupt flags (if more than one source is associated with the same vector).
– The interrupt is serviced.
– Return from interrupt (RETI)

MCU

– Automatically the MCU switches back to the normal flag set (or the interrupt flag set) and pops the previous PC value from the stack.

The interrupt routine usually begins by the identifying the device which generated the interrupt request (by polling). The user should save the registers which are used within the interrupt routine in a software stack. After the RETI instruction is executed, the MCU returns to the main routine.

Figure 20. Interrupt Processing Flow Chart



INTERRUPTS (Cont'd)

3.4.3 Interrupt Option Register (IOR)

The Interrupt Option Register (IOR) is used to enable/disable the individual interrupt sources and to select the operating mode of the external interrupt inputs. This register is write-only and cannot be accessed by single-bit operations.

Address: 0C8h — Write Only

Reset status: 00h

7							0
-	LES	ESB	GEN	-	-	-	-

Bit 7, Bits 3-0 = *Unused*.

Bit 6 = **LES**: *Level/Edge Selection bit*.

When this bit is set to one, the interrupt source #1 is level sensitive. When cleared to zero the edge sensitive mode for interrupt request is selected.

Bit 5 = **ESB**: *Edge Selection bit*.

The bit ESB selects the polarity of the interrupt source #2.

Bit 4 = **GEN**: *Global Enable Interrupt*. When this bit is set to one, all interrupts are enabled. When this bit is cleared to zero all the interrupts (excluding NMI) are disabled.

When the GEN bit is low, the NMI interrupt is active but cannot cause a wake up from STOP/WAIT modes.

This register is cleared on reset.

3.4.4 Interrupt Sources

Interrupt sources available on the ST62E20C/T20C are summarized in the Table 9 with associated mask bit to enable/disable the interrupt request.

Table 9. Interrupt Requests and Mask Bits

Peripheral	Register	Address Register	Mask bit	Masked Interrupt Source	Interrupt vector
GENERAL	IOR	C8h	GEN	All Interrupts, excluding NMI	
TIMER	TSCR	D4h	ETI	TMZ: TIMER Overflow	Vector 3
A/D CONVERTER(*)	ADCR	D1h	EAI	EOC: End of Conversion	Vector 4
Port PAn	ORPA-DRPA	C4h-CCh	ORPAn-DRPAn	PAn pin	Vector 1
Port PBn	ORPB-DRPB	C5h-CDh	ORPBn-DRPBn	PBn pin	Vector 2

*Except ST62T08C

3.5 POWER SAVING MODES

The WAIT and STOP modes have been implemented in the ST62xx family of MCUs in order to reduce the product's electrical consumption during idle periods. These two power saving modes are described in the following paragraphs.

In addition, the Low Frequency Auxiliary Oscillator (LFAO) can be used instead of the main oscillator to reduce power consumption in RUN and WAIT modes.

3.5.1 WAIT Mode

The MCU goes into WAIT mode as soon as the WAIT instruction is executed. The microcontroller can be considered as being in a "software frozen" state where the core stops processing the program instructions, the RAM contents and peripheral registers are preserved as long as the power supply voltage is higher than the RAM retention voltage. In this mode the peripherals are still active.

WAIT mode can be used when the user wants to reduce the MCU power consumption during idle periods, while not losing track of time or the capability of monitoring external events. The active oscillator (main oscillator or LFAO) is not stopped in order to provide a clock signal to the peripherals. Timer counting may be enabled as well as the Timer interrupt, before entering the WAIT mode: this allows the WAIT mode to be exited when a Timer interrupt occurs. The same applies to other peripherals which use the clock signal.

If the power consumption has to be further reduced, the Low Frequency Auxiliary Oscillator (LFAO) can be used in place of the main oscillator, if its operating frequency is lower. If required, the LFAO must be switched on before entering the WAIT mode.

If the WAIT mode is exited due to a Reset (either by activating the external pin or generated by the Watchdog), the MCU enters a normal reset procedure. If an interrupt is generated during WAIT mode, the MCU's behaviour depends on the state of the processor core prior to the WAIT instruction, but also on the kind of interrupt request which is generated. This is described in the following paragraphs. The processor core does not generate a delay following the occurrence of the interrupt, because the oscillator clock is still available and no stabilisation period is necessary.

3.5.2 STOP Mode

If the Watchdog is disabled, STOP mode is available. When in STOP mode, the MCU is placed in the lowest power consumption mode. In this operating mode, the microcontroller can be considered as being "frozen", no instruction is executed, the oscillator is stopped, the RAM contents and peripheral registers are preserved as long as the power supply voltage is higher than the RAM retention voltage, and the ST62xx core waits for the occurrence of an external interrupt request or a Reset to exit the STOP state.

If the STOP state is exited due to a Reset (by activating the external pin) the MCU will enter a normal reset procedure. Behaviour in response to interrupts depends on the state of the processor core prior to issuing the STOP instruction, and also on the kind of interrupt request that is generated.

This case will be described in the following paragraphs. The processor core generates a delay after occurrence of the interrupt request, in order to wait for complete stabilisation of the oscillator, before executing the first instruction.

I/O PORTS (Cont'd)

4.1.3 I/O Port Option Registers

ORA/B (CCh PA, CDh PB) Read/Write

7							0
Px7	Px6	Px5	Px4	Px3	Px2	Px1	Px0

Bit 7-0 = **Px7 - Px0**: *Port A and B Option Register bits.*

4.1.4 I/O Port Data Direction Registers

DDRA/B (C4h PA, C5h PB) Read/Write

7							0
Px7	Px6	Px5	Px4	Px3	Px2	Px1	Px0

Bit 7-0 = Px7 - Px0: *Port A and B Data Direction Registers bits.*

4.1.5 I/O Port Data Registers

DRA/B (C0h PA, C1h PB) Read/Write

7							0
Px7	Px6	Px5	Px4	Px3	Px2	Px1	Px0

Bit 7-0 = **Px7 - Px0**: *Port A and B Data Registers bits.*

Note: X = Don't care

4.2 TIMER

The MCU features an on-chip Timer peripheral, consisting of an 8-bit counter with a 7-bit programmable prescaler, giving a maximum count of 2^{15}. The peripheral may be configured in three different operating modes.

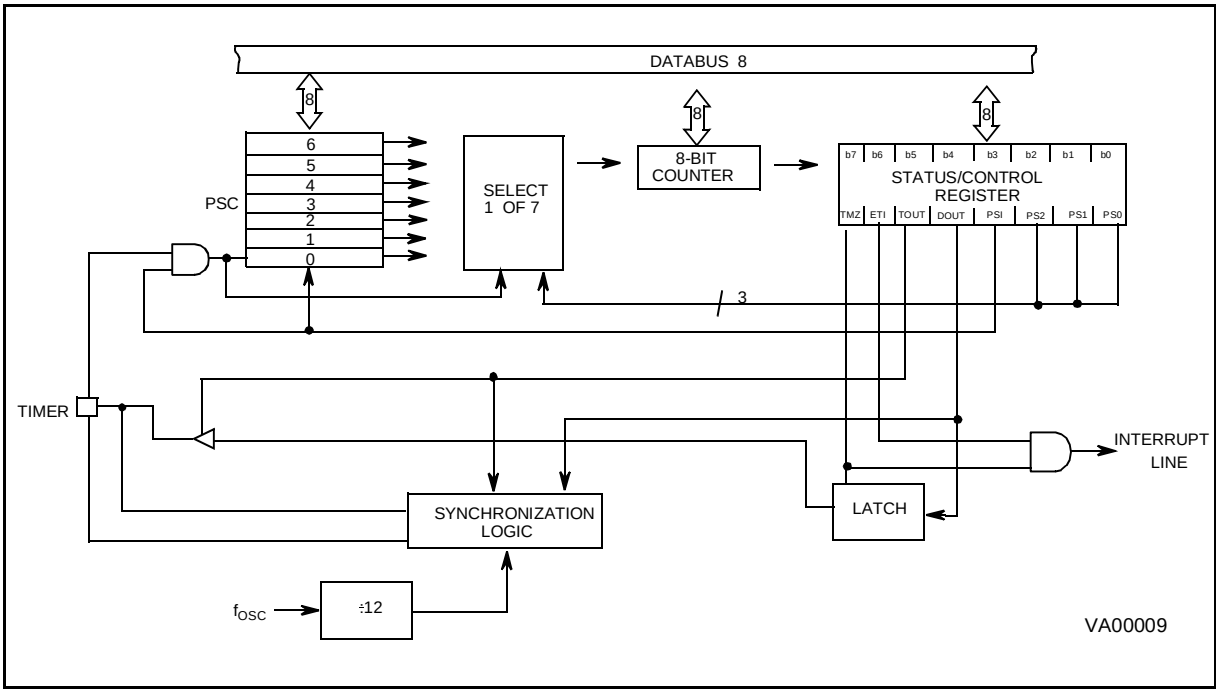
Figure 24 shows the Timer Block Diagram. The external TIMER pin is available to the user. The content of the 8-bit counter can be read/written in the Timer/Counter register, TCR, while the state of the 7-bit prescaler can be read in the PSC register. The control logic device is managed in the TSCR register as described in the following paragraphs.

The 8-bit counter is decremented by the output (rising edge) coming from the 7-bit prescaler and can be loaded and read under program control. When it decrements to zero then the TMZ (Timer Zero) bit in the TSCR is set to “1”. If the ETI (Enable Timer Interrupt) bit in the TSCR is also set to “1”, an interrupt request is generated as described in the Interrupt Chapter. The Timer interrupt can be used to exit the MCU from WAIT mode.

The prescaler input can be the internal frequency f_{INT} divided by 12 or an external clock applied to the TIMER pin. The prescaler decrements on the rising edge. Depending on the division factor programmed by PS2, PS1 and PS0 bits in the TSCR. The clock input of the timer/counter register is multiplexed to different sources. For division factor 1, the clock input of the prescaler is also that of timer/counter; for factor 2, bit 0 of the prescaler register is connected to the clock input of TCR. This bit changes its state at half the frequency of the prescaler input clock. For factor 4, bit 1 of the PSC is connected to the clock input of TCR, and so forth. The prescaler initialize bit, PSI, in the TSCR register must be set to “1” to allow the prescaler (and hence the counter) to start. If it is cleared to “0”, all the prescaler bits are set to “1” and the counter is inhibited from counting. The prescaler can be loaded with any value between 0 and 7Fh, if bit PSI is set to “1”. The prescaler tap is selected by means of the PS2/PS1/PS0 bits in the control register.

Figure 25 illustrates the Timer’s working principle.

Figure 24. Timer Block Diagram



5.3 INSTRUCTION SET

The ST6 core offers a set of 40 basic instructions which, when combined with nine addressing modes, yield 244 usable opcodes. They can be divided into six different types: load/store, arithmetic/logic, conditional branch, control instructions, jump/call, and bit manipulation. The following paragraphs describe the different types.

All the instructions belonging to a given type are presented in individual tables.

Load & Store. These instructions use one, two or three bytes in relation with the addressing mode. One operand is the Accumulator for LOAD and the other operand is obtained from data memory using one of the addressing modes.

For Load Immediate one operand can be any of the 256 data space bytes while the other is always immediate data.

Table 14. Load & Store Instructions

Instruction	Addressing Mode	Bytes	Cycles	Flags	
				Z	C
LD A, X	Short Direct	1	4	Δ	*
LD A, Y	Short Direct	1	4	Δ	*
LD A, V	Short Direct	1	4	Δ	*
LD A, W	Short Direct	1	4	Δ	*
LD X, A	Short Direct	1	4	Δ	*
LD Y, A	Short Direct	1	4	Δ	*
LD V, A	Short Direct	1	4	Δ	*
LD W, A	Short Direct	1	4	Δ	*
LD A, rr	Direct	2	4	Δ	*
LD rr, A	Direct	2	4	Δ	*
LD A, (X)	Indirect	1	4	Δ	*
LD A, (Y)	Indirect	1	4	Δ	*
LD (X), A	Indirect	1	4	Δ	*
LD (Y), A	Indirect	1	4	Δ	*
LDI A, #N	Immediate	2	4	Δ	*
LDI rr, #N	Immediate	3	4	*	*

Notes:
X,Y. Indirect Register Pointers, V & W Short Direct Registers
. Immediate data (stored in ROM memory)
rr. Data space register
Δ. Affected
* . Not Affected

INSTRUCTION SET (Cont'd)

Arithmetic and Logic. These instructions are used to perform the arithmetic calculations and logic operations. In AND, ADD, CP, SUB instructions one operand is always the accumulator while the other can be either a data space memory content or an immediate value in relation with the addressing mode. In CLR, DEC, INC instructions the operand can be any of the 256 data space addresses. In COM, RLC, SLA the operand is always the accumulator.

Table 15. Arithmetic & Logic Instructions

Instruction	Addressing Mode	Bytes	Cycles	Flags	
				Z	C
ADD A, (X)	Indirect	1	4	Δ	Δ
ADD A, (Y)	Indirect	1	4	Δ	Δ
ADD A, rr	Direct	2	4	Δ	Δ
ADDI A, #N	Immediate	2	4	Δ	Δ
AND A, (X)	Indirect	1	4	Δ	Δ
AND A, (Y)	Indirect	1	4	Δ	Δ
AND A, rr	Direct	2	4	Δ	Δ
ANDI A, #N	Immediate	2	4	Δ	Δ
CLR A	Short Direct	2	4	Δ	Δ
CLR r	Direct	3	4	*	*
COM A	Inherent	1	4	Δ	Δ
CP A, (X)	Indirect	1	4	Δ	Δ
CP A, (Y)	Indirect	1	4	Δ	Δ
CP A, rr	Direct	2	4	Δ	Δ
CPI A, #N	Immediate	2	4	Δ	Δ
DEC X	Short Direct	1	4	Δ	*
DEC Y	Short Direct	1	4	Δ	*
DEC V	Short Direct	1	4	Δ	*
DEC W	Short Direct	1	4	Δ	*
DEC A	Direct	2	4	Δ	*
DEC rr	Direct	2	4	Δ	*
DEC (X)	Indirect	1	4	Δ	*
DEC (Y)	Indirect	1	4	Δ	*
INC X	Short Direct	1	4	Δ	*
INC Y	Short Direct	1	4	Δ	*
INC V	Short Direct	1	4	Δ	*
INC W	Short Direct	1	4	Δ	*
INC A	Direct	2	4	Δ	*
INC rr	Direct	2	4	Δ	*
INC (X)	Indirect	1	4	Δ	*
INC (Y)	Indirect	1	4	Δ	*
RLC A	Inherent	1	4	Δ	Δ
SLA A	Inherent	2	4	Δ	Δ
SUB A, (X)	Indirect	1	4	Δ	Δ
SUB A, (Y)	Indirect	1	4	Δ	Δ
SUB A, rr	Direct	2	4	Δ	Δ
SUBI A, #N	Immediate	2	4	Δ	Δ

Notes:
X,Y.Indirect Register Pointers, V & W Short Direct RegistersD. Affected
. Immediate data (stored in ROM memory)* . Not Affected
rr. Data space register

Opcode Map Summary. The following table contains an opcode map for the instructions used by the ST6

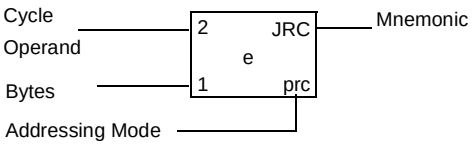
LOW / HI	**0 0000**	**1 0001**	**2 0010**	**3 0011**	**4 0100**	**5 0101**	**6 0110**	**7 0111**	LOW / HI
0 0000	2 JRNZ e 1 pcr	4 CALL abc 2 ext	2 JRNC e 1 pcr	5 JRR b0,rr,ee 3 bt	2 JRZ e 1 pcr	#	2 JRC e 1 prc	4 LD a,(x) 1 ind	**0 0000**
1 0001	2 JRNZ e 1 pcr	4 CALL abc 2 ext	2 JRNC e 1 pcr	5 JRS b0,rr,ee 3 bt	2 JRZ e 1 pcr	4 INC x 1 sd	2 JRC e 1 prc	4 LDI a,nn 2 imm	**1 0001**
2 0010	2 JRNZ e 1 pcr	4 CALL abc 2 ext	2 JRNC e 1 pcr	5 JRR b4,rr,ee 3 bt	2 JRZ e 1 pcr	#	2 JRC e 1 prc	4 CP a,(x) 1 ind	**2 0010**
3 0011	2 JRNZ e 1 pcr	4 CALL abc 2 ext	2 JRNC e 1 pcr	5 JRS b4,rr,ee 3 bt	2 JRZ e 1 pcr	4 LD a,x 1 sd	2 JRC e 1 prc	4 CPI a,nn 2 imm	**3 0011**
4 0100	2 JRNZ e 1 pcr	4 CALL abc 2 ext	2 JRNC e 1 pcr	5 JRR b2,rr,ee 3 bt	2 JRZ e 1 pcr	#	2 JRC e 1 prc	4 ADD a,(x) 1 ind	**4 0100**
5 0101	2 JRNZ e 1 pcr	4 CALL abc 2 ext	2 JRNC e 1 pcr	5 JRS b2,rr,ee 3 bt	2 JRZ e 1 pcr	4 INC y 1 sd	2 JRC e 1 prc	4 ADDI a,nn 2 imm	**5 0101**
6 0110	2 JRNZ e 1 pcr	4 CALL abc 2 ext	2 JRNC e 1 pcr	5 JRR b6,rr,ee 3 bt	2 JRZ e 1 pcr	#	2 JRC e 1 prc	4 INC (x) 1 ind	**6 0110**
7 0111	2 JRNZ e 1 pcr	4 CALL abc 2 ext	2 JRNC e 1 pcr	5 JRS b6,rr,ee 3 bt	2 JRZ e 1 pcr	4 LD a,y 1 sd	2 JRC e 1 prc	#	**7 0111**
8 1000	2 JRNZ e 1 pcr	4 CALL abc 2 ext	2 JRNC e 1 pcr	5 JRR b1,rr,ee 3 bt	2 JRZ e 1 pcr	#	2 JRC e 1 prc	4 LD (x),a 1 ind	**8 1000**
9 1001	2 RNZ e 1 pcr	4 CALL abc 2 ext	2 JRNC e 1 pcr	5 JRS b1,rr,ee 3 bt	2 JRZ e 1 pcr	4 INC v 1 sd	2 JRC e 1 prc	#	**9 1001**
A 1010	2 JRNZ e 1 pcr	4 CALL abc 2 ext	2 JRNC e 1 pcr	5 JRR b5,rr,ee 3 bt	2 JRZ e 1 pcr	#	2 JRC e 1 prc	4 AND a,(x) 1 ind	**A 1010**
B 1011	2 JRNZ e 1 pcr	4 CALL abc 2 ext	2 JRNC e 1 pcr	5 JRS b5,rr,ee 3 bt	2 JRZ e 1 pcr	4 LD a,v 1 sd	2 JRC e 1 prc	4 ANDI a,nn 2 imm	**B 1011**
C 1100	2 JRNZ e 1 pcr	4 CALL abc 2 ext	2 JRNC e 1 pcr	5 JRR b3,rr,ee 3 bt	2 JRZ e 1 pcr	#	2 JRC e 1 prc	4 SUB a,(x) 1 ind	**C 1100**
D 1101	2 JRNZ e 1 pcr	4 CALL abc 2 ext	2 JRNC e 1 pcr	5 JRS b3,rr,ee 3 bt	2 JRZ e 1 pcr	4 INC w 1 sd	2 JRC e 1 prc	4 SUBI a,nn 2 imm	**D 1101**
E 1110	2 JRNZ e 1 pcr	4 CALL abc 2 ext	2 JRNC e 1 pcr	5 JRR b7,rr,ee 3 bt	2 JRZ e 1 pcr	#	2 JRC e 1 prc	4 DEC (x) 1 ind	**E 1110**
F 1111	2 JRNZ e 1 pcr	4 CALL abc 2 ext	2 JRNC e 1 pcr	5 JRS b7,rr,ee 3 bt	2 JRZ e 1 pcr	4 LD a,w 1 sd	2 JRC e 1 prc	#	**F 1111**

Abbreviations for Addressing Modes:

dir	Direct
sd	Short Direct
imm	Immediate
inh	Inherent
ext	Extended
b.d	Bit Direct
bt	Bit Test
pcr	Program Counter Relative
ind	Indirect

Legend:

#	Indicates Illegal Instructions
e	5 Bit Displacement
b	3 Bit Address
rr	1byte dataspace address
nn	1 byte immediate data
abc	12 bit address
ee	8 bit Displacement



6 ELECTRICAL CHARACTERISTICS

6.1 ABSOLUTE MAXIMUM RATINGS

This product contains devices to protect the inputs against damage due to high static voltages, however it is advisable to take normal precaution to avoid application of any voltage higher than the specified maximum rated voltages.

For proper operation it is recommended that V_I and V_O be higher than V_{SS} and lower than V_{DD}. Reliability is enhanced if unused inputs are connected to an appropriate logic voltage level (V_{DD} or V_{SS}).

Power Considerations.The average chip-junction temperature, Tj, in Celsius can be obtained from:

Tj=TA + PD x RthJA

Where:TA = Ambient Temperature.

RthJA =Package thermal resistance (junction-to ambient).

PD = Pint + Pport.

Pint =IDD x VDD (chip internal power).

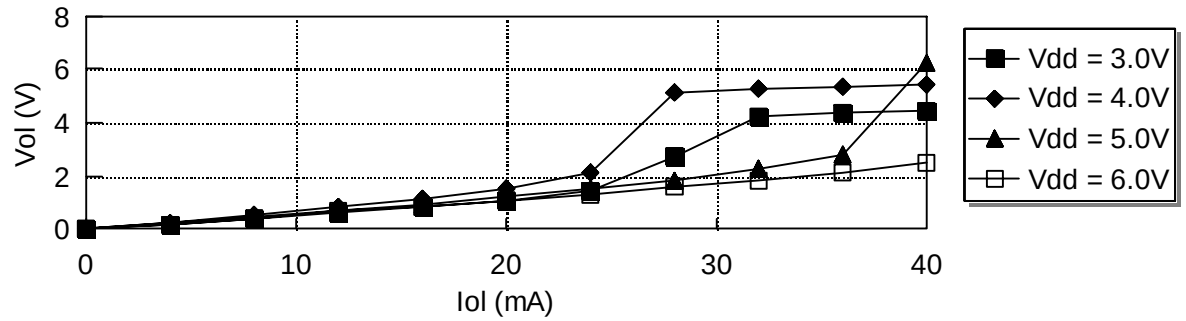
Pport =Port power dissipation (determined by the user).

Symbol	Parameter	Value	Unit
V_{DD}	Supply Voltage	-0.3 to 7.0	V
V_I	Input Voltage	V_{SS} - 0.3 to V_{DD} + 0.3(1)	V
V_O	Output Voltage	V_{SS} - 0.3 to V_{DD} + 0.3(1)	V
IV_{DD}	Total Current into V_{DD} (source)	80	mA
IV_{SS}	Total Current out of V_{SS} (sink)	100	mA
Tj	Junction Temperature	150	°C
T_{STG}	Storage Temperature	-60 to 150	°C

Notes:

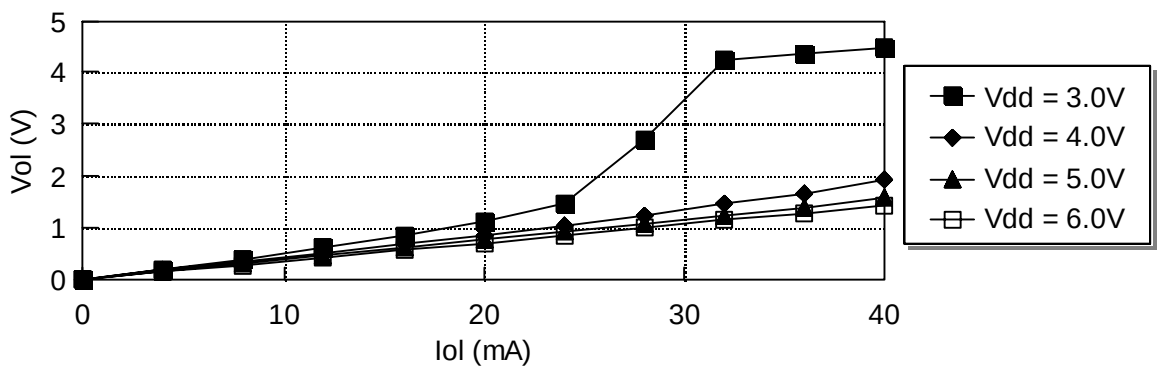
- Stresses above those listed as “absolute maximum ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these conditions is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.
- (1) Within these limits, clamping diodes are guarantee to be not conductive. Voltages outside these limits are authorised as long as injection current is kept within the specification.

Figure 36. Vol versus Iol on all I/O port at T=25°C



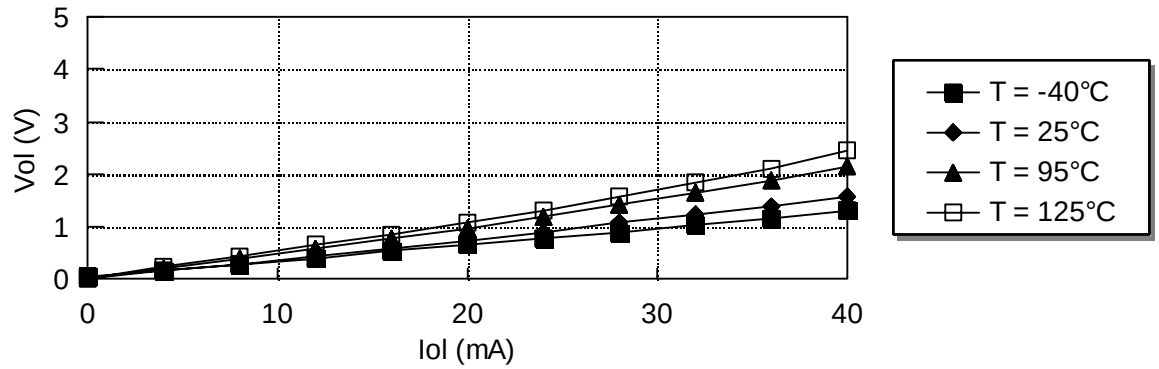
This curves represents typical variations and is given for guidance only

Figure 37. Vol versus Iol for High sink (20mA) I/Oports at T=25°C



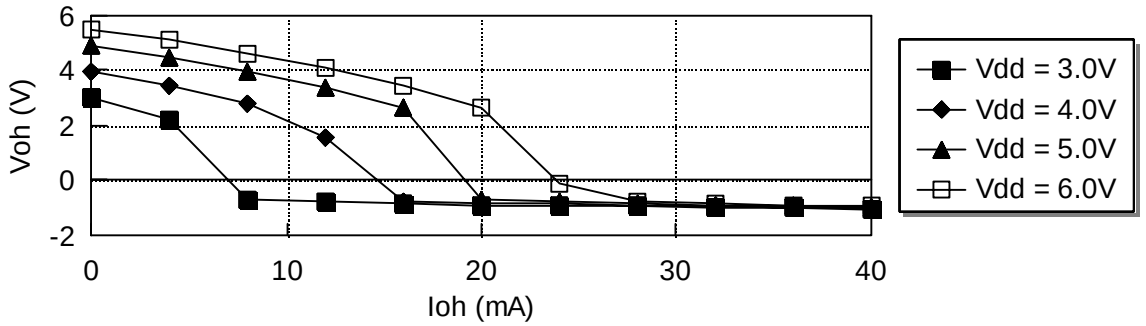
This curves represents typical variations and is given for guidance only

Figure 38. Vol versus Iol for High sink (20mA) I/O ports at Vdd=5V



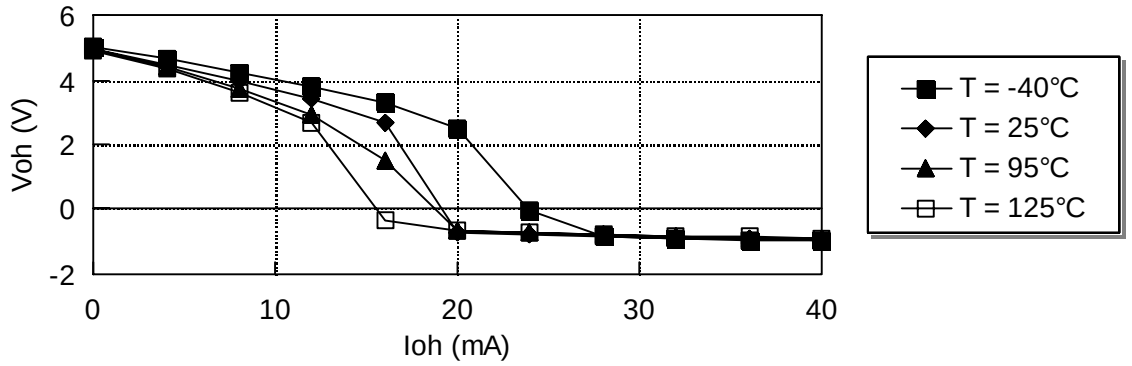
This curves represents typical variations and is given for guidance only

Figure 39. Voh versus Ioh on all I/O port at 25°C



This curves represents typical variations and is given for guidance only

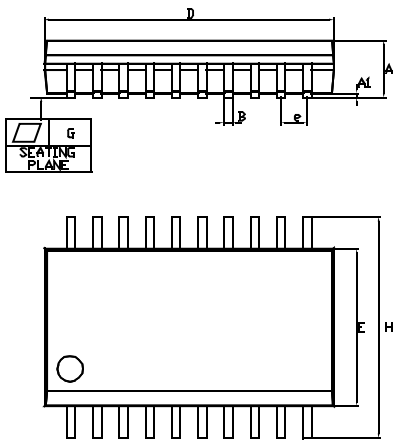
Figure 40. Voh versus Ioh on all I/O port at Vdd=5V



This curves represents typical variations and is given for guidance only

PACKAGE MECHANICAL DATA (Cont'd)

Figure 43. 20-Pin Plastic Small Outline Package, 300-mil Width



SO20

Dim.	mm			inches		
	Min	Typ	Max	Min	Typ	Max
A	2.35		2.65	0.0926		0.1043
A1	0.10			0.0040		
B	0.33		0.51	0.0130		0.0200
C			0.32			0.0125
D	4.98		13.00	0.1961		0.5118
E	7.40		7.60	0.2914		0.2992
e		1.27			0.050	
H	10.01		10.64	0.394		0.419
h	0.25		0.74	0.010		0.029
K	0°		8°	0°		8°
L	0.41		1.27	0.016		0.050
G			0.10			0.004
	Number of Pins					
N	20					

7.2 .ORDERING INFORMATION

Table 20. OTP/EPROM VERSION ORDERING INFORMATION

Sales Type	I/O	Program Memory (Bytes)	Analog input	Temperature Range	Package
ST62E20CF1	12	3884 (EPROM)	8	0 to +70°C	CDIP20W
ST62T08CB6		1036 (OTP)	None	-40 to + 85°C	PDIP20
ST62T08CM6					PSO20
ST62T09CB6		1036 (OTP)	4		PDIP20
ST62T09CM6					PSO20
ST62T10CB6		1836 (OTP)	8		PDIP20
ST62T10CM6					PSO20
ST62T20CB6		3884 (OTP)			PDIP20
ST62T20CM6					PSO20
ST62T20CB3		3884 (OTP)		-40 to + 125°C	PDIP20
ST62T20CM3					PSO20

ORDERING INFORMATION (Cont'd)

Table 4. ROM version Ordering Information

Sales Type	ROM	Analog inputs	Temperature Range	Package
ST6208CB1/XXX ST6208CB6/XXX ST6208CB3/XXX	1036 Bytes	None	0 to +70°C -40 to + 85°C -40 to + 125°C	PDIP20
ST6208CM1/XXX ST6208CM6/XXX ST6208CM3/XXX			0 to +70°C -40 to + 85°C -40 to + 125°C	PSO20
ST6209CB1/XXX ST6209CB6/XXX ST6209CB3/XXX	1036 Bytes	4	0 to +70°C -40 to + 85°C -40 to + 125°C	PDIP2
ST6209CM1/XXX ST6209CM6/XXX ST6209CM3/XXX			0 to +70°C -40 to + 85°C -40 to + 125°C	PSO20
ST6210CB1/XXX ST6210CB6/XXX ST6210CB3/XXX	1836 Bytes	8	0 to +70°C -40 to + 85°C -40 to + 125°C	PDIP20
ST6210CM1/XXX ST6210CM6/XXX ST6210CM3/XXX			0 to +70°C -40 to + 85°C -40 to + 125°C	PSO20
ST6220CB1/XXX ST6220CB6/XXX ST6220CB3/XXX	3884 Bytes	8	0 to +70°C -40 to + 85°C -40 to + 125°C	PDIP20
ST6220CM1/XXX ST6220CM6/XXX ST6220CM3/XXX			0 to +70°C -40 to + 85°C -40 to + 125°C	PSO20